Pangomonium 2.0

The Ultimate Text & Graphics Engine for Hollywood

Andreas Falkenhahn

Inhaltsverzeichnis

1 Allgemeine Informationen

1.1 Einführung

Pangomonium ist ein Plugin für Hollywood, das über fortschrittliche Rendering-Module sowohl für Text als auch für Grafiken verfügt. Das Text-Rendering-Modul von Pangomonium verfügt über modernstes Layout und ermöglicht das Zeichnen von Texten in nahezu jeder Sprache der Welt. Es unterstützt komplexe Layouts wie die von rechts nach links verlaufenden oder bidirektionalen Layouts, die man in den Sprachen Arabisch und Hebräisch findet, oder vertikale Layouts in Spalten wie im Japanischen. Pangomonium kann auch komplexe Textsprachen wie Hindi verarbeiten und Schriftarten, die farbige Glyphen (Emojis) enthalten, werden ebenfalls unterstützt. Durch den vollständigen Zugriff auf die Pango-API können Sie mit dem Plugin auch alle Phasen des Textrendering-Prozesses anpassen.

Darüber hinaus verfügt Pangomonium auch über einen Lader für das beliebte SVG-Vektorbildformat. Sobald Pangomonium installiert ist, kann Hollywood "automatisch" SVG-Bilder laden und zeichnen. SVG-Bilder werden von Pangomonium als echte Vektorbilder geladen, was bedeutet, dass Sie sie ohne Qualitätsverlust nach Ihren Wünschen skalieren und transformieren können. Sie werden in jeder Auflösung perfekt gestochen scharf sein.

Darüber hinaus bietet Pangomonium auch Hollywood-Wrapper für fast alle Funktionen des beliebten Cairo-Grafik-Modul, sodass Sie direkt aus Hollywood-Skripten auf erweiterte Zeichenfunktionen für Vektorgrafiken zugreifen können. Die Verwendung der Cairo-API über Pangomonium hat den Vorteil, dass Sie beim Zeichnen von Vektorgrafiken eine präzise Kontrolle über alles haben.

Es gibt zwei Möglichkeiten, Pangomonium zu verwenden: Über die High-Level-Schnittstelle, die sich direkt in Hollywoods Text- und Vektorgrafikbibliothek einbinden lässt und diese mit den von Pangomonium bereitgestellten Funktionen erweitern kann, z.B. Zeichnen von farbigen Emojis oder Texte mit komplexen Layouts wie Arabisch oder Japanisch. Dies ist die bequemste Art, Pangomonium zu nutzen, da Sie hier nicht direkt die Pango- und Cairo-Funktionen nutzen müssen, sondern einfach die etablierten Hollywood-Befehle nutzen können.

Eine weitere Möglichkeit, Pangomonium zu nutzen, ist die Low-Level-Schnittstelle: Über diese Schnittstelle können Sie direkt aus Hollywood-Skripten auf die Pango- und Cairo-APIs zugreifen. Dies ist äußerst leistungsstark, da Sie auf Hunderte verschiedener Text- und Grafik-Rendering-Funktionen zugreifen können, sodass Sie Pangomonium genau an Ihre spezifischen Anforderungen anpassen können. Pangomonium enthält über 500 Befehle, mit denen Sie alle Ihre Träume von der Text- und Vektorgrafik-Rendering wahr werden lassen!

Schließlich enthält Pangomonium eine umfangreiche Dokumentation in verschiedenen Formaten wie PDF, HTML, AmigaGuide und CHM, die detaillierte Beschreibungen aller vom Plugin angebotenen Befehlen und Funktionen enthält.

All dies macht Pangomonium zur ultimatives Text- und Grafik-Rendering-Modul für Hollywood, die alles enthält, was Sie zum Zeichnen von Text in jeder auf dem Planeten gesprochenen Sprache benötigen.

1.2 Lizenz

FÜR DIESES PROGRAMM GIBT ES KEINE GARANTIE, SOWEIT ES DIE ANZUWENDENDEN GESETZE ZULASSEN. SOFERN ANDERSWO NICHTS GEGENTEILIGES GESCHRIEBEN STEHT STELLEN DER AUTOR UND/ODER DRITTE DAS PROGRAMM "SO WIE ES IST" ZUR VERFÜGUNG, OHNE IRGENDEINE GARANTIE, WEDER DIREKT NOCH INDIREKT. DIES BEINHALTET, IST ABER NICHT DARAUF BESCHRÄNKT, VERKÄUFLICHKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN VERWENDUNGSZWECK. DAS VOLLSTÄNDIGE RISIKO DER QUALITÄT UND AUSFÜHRBARKEIT DES PROGRAMMS LIEGT BEIM ANWENDER. SOLLTE SICH DAS PROGRAMM ALS DEFEKT HERAUSSTELLEN, LIEGEN ALLE KOSTEN FÜR SERVICE, INSTANDSETZUNG ODER NACHBESSERUNG BEIM ANWENDER.

KEIN COPYRIGHT-INHABER ODER DRITTER, DER DAS PROGRAMM WIE OBEN ERLAUBT WEITERVERKAUFT, KANN FÜR SCHÄDEN IRGENDWELCHER ART HAFTBAR GEMACHT WERDEN (DIES BEINHALTET, IST ABER NICHT BESCHRÄNKT AUF, DATENVERLUST INFOLGE UNFÄHIGKEIT DES PROGRAMMS, MIT ANDEREN PROGRAMMEN ZUSAMMENZUARBEITEN), SELBST WENN EIN SOLCHER INHABER ODER DRITTER AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE, AUSSER ES BESTEHT EINE SCHRIFTLICHE EINWILLIGUNG ODER WIRD VOM GESETZ VERLANGT.

1.3 Anforderungen

- Hollywood 10.0 oder höher
- Windows: Erfordert Windows Vista oder höher
- macOS: Erfordert mindestens 10.5 (Leopard) für PowerPC, 10.9 (Mavericks) für x64 und 11.0 (Big Sur) für arm64
- AmigaOS 3: Ihr Dateisystem muss in der Lage sein, Dateinamen zu verarbeiten, die länger als die üblichen 31 Zeichen sind, was beim klassischen FFS die Grenze darstellt; Daher dürfen Sie kein klassisches FFS mit Pangomonium verwenden. Wenn Sie Schriftarten verwenden, die Emojis oder CJK-Zeichen enthalten, benötigen Sie außerdem viel Speicher, da diese Schriftarten oft mehr als 10 bis 20 Megabyte (oder sogar mehr) groß sind und Pangomonium jede Schriftart, die Sie verwenden, in einen zusammenhängenden Speicherblock lädt. Damit benötigt es viel Speicher und darf nicht fragmentiert sein, damit weiterhin große Blöcke zugewiesen werden können.

1.4 Installation

Die Installation von Pangomonium ist unkompliziert und einfach: Kopieren Sie einfach die Datei `pangomonium.hwp` für die Plattform Ihrer Wahl in das Plugins-Verzeichnis von Hollywood. Auf allen Systemen außer AmigaOS und kompatiblen Systemen müssen Plugins in einem Verzeichnis namens `Plugins` gespeichert werden, das sich im selben Verzeichnis wie das Hauptprogramm von Hollywood befindet. Auf AmigaOS und kompatiblen Systemen müssen Plugins stattdessen in `LIBS:Hollywood` installiert werden. Unter macOS muss sich das `Plugins`-Verzeichnis im `Resources`-Verzeichnis des Anwendungspakets befinden, d.h. im `HollywoodInterpreter.app/Contents/Resources`-Verzeichnis. Beachten Sie, dass `HollywoodInterpreter.app` im `Hollywood.app`-Anwendungspaket selbst gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`.

Anschließend führen Sie den Inhalt des Ordners `Examples` mit dem Ordner `Examples` zusammen, der Teil Ihrer Hollywood-Installation ist. Alle Pangomonium-Beispiele werden dann in der Hollywood-GUI angezeigt und Sie können sie bequem über die Hollywood-GUI oder IDE starten und anzeigen.

Unter Windows sollten Sie außerdem die Datei `Pangomonium.chm` in das `Docs`-Verzeichnis Ihrer Hollywood-Installation kopieren. Anschließend können Sie die Online-Hilfe erhalten, indem Sie F1 drücken, wenn sich der Cursor über einer Pangomonium-Funktion in der Hollywood-IDE befindet.

Kopieren Sie unter Linux und macOS das `Pangomonium`-Verzeichnis, das sich im `Docs`-Verzeichnis des Pangomonium-Distributionsarchivs befindet, in das `Docs`-Verzeichnis Ihrer Hollywood-Installation. Beachten Sie, dass sich das `Docs`-Verzeichnis unter macOS im `Hollywood.app`-Anwendungspaket befindet, d.h. in `Hollywood.app/Contents/Resources/Docs`.

2 Über Pangomonium

2.1 Danksagungen

Pangomonium wurde von Andreas Falkenhahn geschrieben und basiert auf der Pango-Text-Engine und anderen Komponenten. Siehe Abschnitt 1.2 [Pangomonium license], Seite 2, für Details.

Ein besonderer Dank geht an Helmut Haake und Dominic Widmer für die Übersetzung des Handbuchs ins Deutsche. Fehler oder Verbesserungsvorschläge bzgl. des deutschen Handbuchs bitte an das Übersetzungsteam richten, welches unter `handbuch@gmx.ch` oder `https://amiga-resistance.info` erreicht werden kann.

Wenn Sie mich kontaktieren möchten, können Sie entweder eine E-Mail an `andreas@airsoftsoftwair.de` senden oder das Kontaktformular auf `http://www.hollywood-mal.com` nutzen.

2.2 Häufig gestellte Fragen

In diesem Abschnitt werden einige häufig gestellte Fragen behandelt. Bitte lesen Sie diese zuerst durch, bevor Sie im Forum nachfragen, da Ihr Problem möglicherweise hier behandelt wurde.

Q: Gibt es ein Hollywood-Forum, in dem ich mit anderen Benutzern in Kontakt treten kann?

A: Ja, schauen Sie sich bitte den Abschnitt "Community" des offiziellen Hollywood-Portals online unter `http://www.hollywood-mal.com` an.

Q: Wo kann ich um Hilfe bitten?

A: Es gibt ein aktives Forum unter `http://forums.hollywood-mal.com`. Sie können sich gerne daran beteiligen und dort Ihre Frage stellen.

Q: Ich habe einen Fehler gefunden.

A: Bitte posten Sie darüber im Abschnitt "Bugs" des Forums.

2.3 Zukunft

Hier sind einige Dinge, die auf meiner To-Do-Liste stehen

- weitere Beispiele hinzufügen

Zögern Sie nicht, mich zu kontaktieren, wenn Pangomonium eine bestimmte Funktion fehlt, die für Ihr Projekt wichtig ist.

2.4 Geschichte

Ein vollständiges Änderungsprotokoll auf Englisch von Pangomonium finden Sie in der Datei `history.txt`.

3 Verwendung von Pangomonium

3.1 Übersicht

Es gibt zwei verschiedene Möglichkeiten, Pangomonium zu verwenden: Sie können entweder direkt über eine Low-Level-Schnittstelle auf die APIs von Pango und Cairo zugreifen oder Sie können die High-Level-Schnittstelle von Pangomonium verwenden, die einige der Funktionen von Pangomonium den Standardfunktionen von Hollywood zuordnet.

Die Verwendung der High-Level-Schnittstelle ist wirklich einfach und erweitert Hollywood-Funktionen wie `SetFont()` und `TextOut()` für den Betrieb über Pangomonium, sodass sie erweiterte Textlayouts wie von rechts nach links oder farbige Glyphen (Emojis) unterstützen können. Wenn Sie nur etwas Text zeichnen möchten und keine genaue Kontrolle über den Textwiedergabeprozess benötigen, ist die High-Level-Schnittstelle die richtige Wahl für Sie.

Neben der High-Level-Schnittstelle, die Hollywoods Textbibliothek erweitert, verfügt Pangomonium auch über eine zweite High-Level-Schnittstelle. Im Gegensatz zur ersten ersetzt die zweite High-Level-Schnittstelle Hollywoods Vektorgrafikbibliothek und kann durch Aufruf des Hollywood-Befehls `SetVectorEngine()` und Übergabe von "pangomonium" als Argument aktiviert werden.

Die Low-Level-Schnittstelle, d.h. der direkte Zugriff auf die API von Pango und Cairo, ist nützlich, wenn Sie eine genauere Kontrolle über die Textwiedergabe benötigen. Über die Low-Level-Schnittstelle können Sie alle Arten von Optionen in Pangomonium konfigurieren und auf alle erweiterten Funktionen von Pango und Cairo zugreifen, sodass Sie die Kontrolle über alles bis ins kleinste Detail übernehmen können.

3.2 Verwendung der High-Level-Schnittstelle

Die Verwendung der High-Level-Schnittstelle von Pangomonium ist wirklich einfach. Es erweitert Standard-Hollywood-Textbefehle wie `SetFont()` und `TextOut()`, um erweiterte Textwiedergabefunktionen wie komplexe Layouts und farbige Glyphen wie Emojis zu unterstützen. Um Text mit Pangomonium über normale Hollywood-Befehle zu zeichnen, stellen Sie einfach sicher, dass Sie die Schriftart mit Pangomonium öffnen, indem Sie das Tag `Loader` an `SetFont()` übergeben. Sobald Sie eine Schriftart über Pangomonium geöffnet haben, werden alle Hollywood-Befehle, die Text zeichnen, dies automatisch auch über Pangomonium tun, z.B.:

```
@REQUIRE "pangomonium"

SetFont("Noto Sans CJK HK Regular", 72, {Loader = "pangomonium"})
SetFontStyle(#ANTIALIAS)

; "What is your name?" in Japanese
Local t = {21517, 21069, 12399, 12394, 12435, 12391, 12377, 12363}
For Local k = 0 To ListItems(t) - 1 Do text$ = text$ .. Chr(t[k])

TextOut(#CENTER, #CENTER, text$)
```

Der obige Code stellt die Schriftart auf Noto Sans CJK HK Regular, Größe 72, ein und zeichnet japanischen Text. Indem Sie "pangomonium" an das Tag `Loader` übergeben, weisen Sie

`SetFont()` an, die Schriftart über Pangomonium zu öffnen, anstatt über Hollywoods integrierte Schriftarten-Modul oder ein vom Host-Betriebssystem bereitgestellte Schriftarten-Modul.

Ein weiterer Vorteil der Verwendung der High-Level-Schnittstelle ist die Tatsache, dass Pangomonium eine echte vektorbasierte Texttransformation ohne Qualitätsverlust unterstützt. Die integrierten Schriftarten-Module von Hollywood unterstützen nur die verlustfreie Skalierung von Vektortext. Sobald Text gedreht wird, wird ein verlustbehafteter Algorithmus verwendet, sodass Sie beim Zeichnen von gedrehtem Text mit Pangomonium eine bessere Qualität erhalten.

Darüber hinaus können Sie mit Pangomonium die Markup-Sprache von Pango verwenden, die einige erweiterte Textformatierungen ausführen kann, die mit der in Hollywood integrierten Text-Module nicht möglich sind. Mit der Auszeichnungssprache von Pango können Sie beispielsweise unterschiedliche Schriftgrößen in einem einzelnen Textobjekt verwenden. Wenn Sie also leistungsfähigere Formatierungsfunktionen benötigen, als sie von Hollywoods integriertem Textrenderer unterstützt werden, sollten Sie stattdessen auch Pangomonium verwenden. Weitere Informationen zur Auszeichnungssprache von Pango finden Sie unter playout:SetMarkup(). Beachten Sie außerdem, dass die Auszeichnungssprache von Pango aus Kompatibilitätsgründen in Pangomonium nicht standardmäßig aktiviert ist. Wenn Sie es verwenden möchten, müssen Sie es explizit aktivieren, indem Sie das Tag `Markup` auf `True` setzen (siehe unten).

Beachten Sie, dass wir im obigen Code `SetFont()` explizit anweisen, Pangomonium zum Öffnen der Schriftart zu verwenden, indem wir `"pangomonium"` im Tag `Loader` übergeben. Alternativ können Sie Pangomonium auch global für alle Hollywood-Befehle aktivieren, die sich mit Schriftarten befassen, indem Sie einfach den Schriftartenadapter von Pangomonium installieren. Dies kann erreicht werden, indem das Tag `InstallAdapter` auf `True` gesetzt wird, wenn `@REQUIRE` Pangomonium ist, z.B. so was:

```
@REQUIRE "pangomonium", {InstallAdapter = True}
```

Wenn Sie Pangomonium global aktivieren, müssen Sie das Tag `Loader` nicht mehr mit `SetFont()` oder anderen Hollywood-Befehlen verwenden, die sich mit Schriftarten befassen. Denn wenn Pangomonium global aktiviert ist, werden Befehle wie `SetFont()` automatisch für jede zu öffnende Schriftart Pangomonium aufrufen, unabhängig davon, ob das Plugin diese Schriftart verarbeiten kann oder nicht. Beachten Sie jedoch, dass Pangomonium beim Öffnen von Schriftarten standardmäßig nie fehlschlägt. Wenn Sie es also wie oben global aktivieren, übernimmt es die Verwaltung jeder einzelnen Schriftart, die Sie öffnen, was möglicherweise nicht Ihren Wünschen entspricht. Sie können dieses Verhalten ändern, indem Sie das Benutzer-Tag `NoFail` auf `False` setzen. In diesem Fall schlägt Pangomonium fehl, wenn eine Schriftart nicht verfügbar ist. Wenn Sie jedoch möchten, dass Pangomonium alle Schriftarten in Ihrem Skript verarbeitet, kann es eine gute Idee sein, es global mit `InstallAdapter` wie oben gezeigt zu aktivieren.

Wenn Sie Pangomonium verwenden, können Sie auch einige zusätzliche Argumente an `SetFont()`, `OpenFont()` oder den Präprozessorbefehl @FONT übergeben, indem Sie die Benutzer-Tags von Hollywood verwenden. Die folgenden zusätzlichen Argumente werden von Pangomonium anerkannt:

`NoFail` Standardmäßig öffnet Pangomonium immer eine Schriftart. Wenn die angeforderte Schriftart nicht vorhanden ist, wird ein geeigneter Ersatz verwendet.

Standardmäßig öffnet Pangomonium immer eine Schriftart, unabhängig davon, welchen Namen Sie ihr geben. Wenn Sie das nicht möchten, setzen Sie dieses Tag auf `False`. Der Standardwert ist `True`.

`Markup` Wenn Sie dies auf `True` setzen, erlaubt Ihnen Pangomonium die Verwendung der Pango-Auszeichnungssprache in dem Text, den Sie an Hollywood-Aufrufe wie `TextOut()` übergeben. Standardmäßig unterstützt Pangomonium nur die Textformatierungscodes von Hollywood. Wenn Sie möchten, dass die Markup-Sprache von Pango anstelle der Textformatierungscodes von Hollywood verwendet wird, setzen Sie dieses Tag auf `True`. Weitere Informationen zur Markup-Sprache von Pango finden Sie unter playout:SetMarkup(). Der Standardwert ist `False`.

`Monochrome`
Setzen Sie dieses Tag auf `True`, wenn Pangomonium im Monochrom-Modus arbeiten soll. Dadurch wird der Speicherverbrauch um 75% reduziert, da Pangomonium nur einen statt vier Pixelkanäle zuweisen muss, Sie jedoch im Monochrom-Modus nur einfarbigen Text verwenden können, sodass Dinge wie farbige Emojis nicht korrekt gezeichnet werden. Der Standardwert ist `False`.

`NoFT2` Standardmäßig verwendet Pangomonium auf allen Plattformen den FreeType2-basierten Textrenderer von Pango. Wenn Sie das nicht möchten, setzen Sie dieses Tag auf `True`. In diesem Fall verwendet Pangomonium stattdessen den Standard-Textrenderer des Host-Betriebssystems (z.B. DirectWrite unter Windows, Core Text unter macOS usw.). Beachten Sie, dass auf AmigaOS und kompatiblen Versionen der einzige verfügbare Renderer FreeType2 ist, sodass Sie dieses Tag auf AmigaOS nicht auf `True` setzen dürfen. Der Standardwert ist `False`.

So können Sie Benutzer-Tags an Pangomonium übergeben:

```
@REQUIRE "pangomonium"
SetFont("Arial", 72, {Loader = "pangomonium", UserTags = {Markup = True}})
SetFontStyle(#ANTIALIAS)
TextOut(#CENTER, #CENTER, "<span bgcolor=\"red\">Hello</span>")
```

Der obige Code zeigt, wie die Markup-Sprache von Pango in der High-Level-Schnittstelle von Pangomonium aktiviert wird.

3.3 Verwendung der Vektorgrafik-Schnittstelle

Neben der High-Level-Schnittstelle, die Hollywoods Textfunktionen erweitert, verfügt Pangomonium auch über eine zweite High-Level-Schnittstelle, die als Ersatz für Hollywoods Vektorgrafikbibliothek verwendet werden kann. Sie können den Befehl `SetVectorEngine()` von Hollywood verwenden, um Pangomonium zum Standardrenderer für Vektorgrafiken zu machen. Das bedeutet, dass alle Vektorgrafiken von Pangomonium und nicht von Hollywood gezeichnet werden, sodass Hollywood-Befehle wie `DrawPath()` und alle anderen Befehle aus der Vektorgrafikbibliothek von Pangomonium verwaltet werden.

Um Pangomonium zum Standard-Renderer für Vektorgrafiken zu machen, gehen Sie einfach wie folgt vor:

```
SetVectorEngine("pangomonium")
```

3.4 Laden von SVG-Bildern

Pangomonium kann auch SVG-Bilder laden und zeichnen. Sobald das Plugin installiert ist, kann Hollywood "automatisch" SVG-Bilder öffnen. Wenn Sie also beispielsweise `LoadBrush()` für ein SVG-Bild verwenden, lädt Pangomonium es automatisch. Beachten Sie, dass empfohlen wird, das Tag `Loader` zu verwenden, um anzugeben, welches Plugin zum Laden des SVG-Bilds verwendet werden soll, da es ein anderes Hollywood-Plugin gibt, das SVG-Bilder laden kann, z.B.:

```
; Dieser Code veranlasst Pangomonium, das Bild zu laden
LoadBrush(1, "test.svg", {Loader = "pangomonium"})

; Dieser Code sorgt dafür, dass das SVGimage.hwp-Plugin das Bild lädt
LoadBrush(1, "test.svg", {Loader = "svgimage"})
```

Wenn Sie das Tag `Loader` nicht angeben und zwei Plugins installiert haben, die SVG-Bilder laden können, ist das Plugin, das Hollywood zuerst geladen hat, dasjenige, das Ihr SVG-Bild lädt. Da die Reihenfolge, in der Hollywood-Plugins geladen werden, jedoch vom Dateisystem abhängt, können Sie sich auf nichts wirklich verlassen. Daher ist es besser, explizit das Tag `Loader` zu verwenden, um Hollywood mitzuteilen, welches Plugin zum Laden des SVG-Bilds verwendet werden soll. Um zu sehen, welches Plugin Ihr SVG-Bild geladen hat, können Sie das Attribut `#ATTRLOADER` wie folgt verwenden:

```
DebugPrint(GetAttribute(#BRUSH, 1, #ATTRLOADER))
```

Wenn es ein SVG-Bild lädt, erstellt Pangomonium immer einen Vektorpinsel für Sie. Das bedeutet, dass Sie es mit Befehlen wie `ScaleBrush()`, `RotateBrush()` und `TransformBrush()` ohne Qualitätsverlust transformieren können. Falls Ebenen aktiviert sind und Sie mit `DisplayBrush()` einen Vektorpinsel anzeigen, können die Ebenengrafiken auch ohne Qualitätsverlust transformiert werden. Das Gleiche gilt für BGPics, die Vektorgrafiken verwenden.

Bitte beachten Sie, dass Pangomonium standardmäßig auch immer Bilder mit Alphakanal erstellt. Wenn Sie das nicht möchten, müssen Sie den Alphakanal nach dem Laden des Bildes manuell mit `DeleteAlphaChannel()` löschen. Falls Sie ein SVG-Bild als BGPic verwenden möchten, können Sie dieses Problem auch umgehen, indem Sie einen Hintergrund für dieses BGPic definieren, z.B. so:

```
@BGPIC 1, "test.svg", {FillStyle = #FILLCOLOR, FillColor = #WHITE,
                       Loader = "pangomonium"}
```

3.5 Verwendung der Low-Level-Schnittstelle

Die Verwendung der Low-Level-Schnittstelle von Pangomonium ist schwieriger als die Verwendung der High-Level-Schnittstelle, da Sie damit direkt auf die APIs von Pango und Cairo zugreifen können. Das bedeutet, dass Sie sich zunächst mit diesen APIs vertraut machen sollten, damit Sie wissen, wie sie konzipiert sind und wie sie Ihren Zwecken dienen können.

Zusätzlich zu den Kern-APIs von Pango und Cairo bietet Pangomonium auch einige spezielle Funktionen, die zur Überbrückung von Pango/Cairo-Aufrufen und Hollywood-Befehlen verwendet werden können. Eine der Schlüsselfunktionen, die als solche Brücke dient, ist csurface:ToBrush(), mit der Sie eine Cairo-Oberfläche in einen Hollywood-Pinsel umwandeln

können. Umgekehrt ist es mit der Funktion cairo.ImageSurfaceFromBrush() auch möglich, Cairo-Oberflächen aus Hollywood-Pinseln zu erstellen.

Durch die Verwendung dieser Überbrückungsfunktionen können Sie Pango/Cairo-APIs direkt verwenden und das Ergebnis dann in einen Hollywood-Pinsel konvertieren. Das Zeichnen von Text mit Pango umfasst beispielsweise normalerweise die folgenden Schritte:

1. Erstellung einen Pango-Zeichensatz
2. Erstellung einen Pango-Kontext aus dem Zeichensatz
3. Erstellung aus dem Kontext ein Pango-Layout
4. Erstellung einer Pango-Schriftartenbeschreibung und weist sie dem Layout zu
5. Laden einer Schriftart in den Zeichensatz
6. Festlegen des Texts, der gerendert werden soll
7. Erstellung eine Cairo-Oberfläche
8. Erstellung von der Oberfläche aus einen Cairo-Kontext
9. Anzeigen des Pango-Layout im Cairo-Kontext
10. Umwandlung der Cairo-Oberfläche in einen Hollywood-Pinsel

Wie Sie sehen, ist es ziemlich aufwändig, mit Pangomonium Text zu zeichnen, wenn die Pango/Cairo-APIs direkt verwendet werden. Es ist viel einfacher, dies mit der High-Level-Schnittstelle zu tun, aber nur die Low-Level-Schnittstelle ermöglicht Ihnen den Zugriff auf alle Funktionen von Pango und Cairo.

So sehen die oben beschriebenen Schritte im tatsächlichen Code aus:

```
@REQUIRE "pangomonium"
@DISPLAY {Color = #WHITE}

fontmap = pango.FontMap(#CAIRO_FONT_TYPE_FT)
context = fontmap:CreateContext()
layout = pango.Layout(context)
fontdesc = pango.FontDescription("DejaVu Sans 72")
layout:SetFontDescription(fontdesc)
fontmap:LoadFont(context, fontdesc)
layout:SetText("Hello World")
img = cairo.ImageSurface(#CAIRO_FORMAT_ARGB32, 640, 480)
cr = cairo.Context(img)
cr:ShowLayout(layout)
img:ToBrush(1)
DisplayBrush(1, 0, 0)
```

Wenn Sie die Low-Level-Schnittstelle verwenden, müssen Sie auch daran denken, Objekte zu löschen, wenn sie nicht mehr verwendet werden. Dies kann durch den Aufruf der `Free()`-Funktionen der einzelnen Objekte erfolgen, z.B. pcontext:Free(), um einen Pango-Kontext zu löschen. Alternativ können Sie das Objekt auch auf Null setzen. Dies signalisiert Hollywoods Speicherbereiniger, dass das Objekt nicht mehr verwendet wird und gelöscht werden kann. Beachten Sie jedoch, dass die Speicherbereinigung nur dann erfolgt, wenn wirklich nirgendwo mehr Verweise auf das Objekt vorhanden sind. Daher ist es möglicherweise sicherer, die `Free()`-Funktionen explizit aufzurufen, anstatt Objekte auf Null zu setzen.

Wenn Sie den Code aus dem obigen Beispiel in eine Funktion einfügen und nur lokale Variablen verwenden, können Sie jedoch auf den expliziten Aufruf von `Free()` für alle Objekte verzichten, da lokale Variablen automatisch vom Speicherbereiniger gelöscht werden, sobald die Funktion beendet ist. Betrachten Sie den folgenden Code, der mit dem oben genannten identisch ist, mit dem einzigen Unterschied, dass der Code jetzt in eine Funktion eingeschlossen ist:

```
Function p_DrawText(t$, brush)
   Local fontmap = pango.FontMap()
   Local context = fontmap:CreateContext()
   Local layout = pango.Layout(context)
   Local fontdesc = pango.FontDescription("DejaVu Sans 72")
   layout:SetFontDescription(fontdesc)
   fontmap:LoadFont(context, fontdesc)
   layout:SetText(t$)
   Local img = cairo.ImageSurface(#CAIRO_FORMAT_ARGB32, 640, 480)
   Local cr = cairo.Context(img)
   cr:ShowLayout(layout)
   img:ToBrush(brush)
EndFunction
```

Da alle Pango- und Cairo-Objekte in lokalen Variablen gespeichert werden, wird alles für den Speicherbereiniger markiert, sobald die Funktion beendet ist. Daher ist es in diesem Fall nicht notwendig, Objekte explizit durch den Aufruf von `Free()` tz löschen, auch wenn es dennoch als gute Vorgehensweise angesehen werden könnte.

Weitere Informationen zu den APIs Pango und Cairo finden Sie in den folgenden Kapiteln.

4 Cairo-Befehle

4.1 cairo.Context

BEZEICHNUNG

cairo.Context – Erstellt einen Cairo-Kontext

ÜBERSICHT

```
handle = cairo.Context(target)
```

BESCHREIBUNG

Erstellt einen neuen Cairo-Kontext, in dem alle Grafikstatusparameter auf Standardwerte gesetzt sind und `target` als Zieloberfläche dient. Die Zieloberfläche sollte mit einem Backend-spezifischem Befehl wie cairo.ImageSurface() (oder einem anderen Cairo-Oberflächen-Backend-Konstruktor) erstellt werden.

Dieser Befehl verweist auf `target`, sodass Sie sofort csurface:Free() darauf aufrufen können, wenn Sie keinen separaten Verweis darauf pflegen müssen.

Dieser Befehl gibt einen neu zugewiesenen Cairo-Kontext mit einem Referenzzähler von 1 zurück. Der anfängliche Referenzzähler sollte mit ccontext:Free() gelöscht werden, wenn Sie mit der Verwendung des Cairo-Kontexts fertig sind. Dieser Befehl gibt niemals Null zurück. Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Cairo-Kontextobjekt zurückgegeben, für das ccontext:Status() `#CAIRO_STATUS_NO_MEMORY` zurückgibt. Wenn Sie versuchen, auf eine Oberfläche abzuzielen, die das Schreiben nicht unterstützt (z.B. eine Cairo MIME-Oberfläche), wird ein `#CAIRO_STATUS_WRITE_ERROR` ausgelöst. Sie können dieses Objekt normal verwenden, es wird jedoch keine Zeichnung erstellt.

EINGABEN

`target` Zieloberfläche für den Kontext

RÜCKGABEWERTE

`handle` ein neu zugewiesener Cairo Kontext

4.2 cairo.FontFace

BEZEICHNUNG

cairo.FontFace – Erstellt eine Schriftart

ÜBERSICHT

```
font = cairo.FontFace(pattern)
```

BESCHREIBUNG

Erstellt eine neue Schriftart für das FreeType-Schriftart-Backend basierend auf einem Fontconfig-Muster. Diese Schriftart kann dann mit ccontext:SetFontFace() oder cairo.ScaledFont() verwendet werden.

Das Argument `pattern` kann entweder eine Zeichenkette im von Fontconfig verwendeten Format sein, z.B. `"Ubuntu:style=italic:weight=200"` oder es kann eine Tabelle sein, die

die folgenden Tags enthalten kann, die jeweils ein einzelnes Element eines Fontconfig-Musters beschreiben:

`Family` Der Name der Schriftfamilie.

`Style` Schriftstil (als Zeichenkette übergeben). Überschreibt die Stärke und Neigung.

`Slant` Schriftschräge. Dies kann eine der folgenden Konstanten sein:

```
#FC_SLANT_ROMAN
#FC_SLANT_ITALIC
#FC_SLANT_OBLIQUE
```

`Weight` Schriftstärke. Dies kann eine der folgenden Konstanten sein:

```
#FC_WEIGHT_THIN
#FC_WEIGHT_EXTRALIGHT
#FC_WEIGHT_ULTRALIGHT
#FC_WEIGHT_LIGHT
#FC_WEIGHT_DEMILIGHT
#FC_WEIGHT_SEMILIGHT
#FC_WEIGHT_BOOK
#FC_WEIGHT_REGULAR
#FC_WEIGHT_NORMAL
#FC_WEIGHT_MEDIUM
#FC_WEIGHT_DEMIBOLD
#FC_WEIGHT_SEMIBOLD
#FC_WEIGHT_BOLD
#FC_WEIGHT_EXTRABOLD
#FC_WEIGHT_ULTRABOLD
#FC_WEIGHT_BLACK
#FC_WEIGHT_HEAVY
#FC_WEIGHT_EXTRABLACK
#FC_WEIGHT_ULTRABLACK
```

`Size` Schriftgröße in Punkt.

`Aspect` Streckt Glyphen vor der Darstellung horizontal aus.

`PixelSize`
Schriftgröße in Pixel.

`Spacing` Schriftabstand. Dies kann eine der folgenden Konstanten sein:

```
#FC_PROPORTIONAL
#FC_DUAL
#FC_MONO
#FC_CHARCELL
```

`Width` Schriftbreite. Dies kann eine der folgenden Konstanten sein:

```
#FC_WIDTH_ULTRACONDENSED
#FC_WIDTH_EXTRACONDENSED
#FC_WIDTH_CONDENSED
```

```
#FC_WIDTH_SEMICONDENSED
#FC_WIDTH_NORMAL
#FC_WIDTH_SEMIEXPANDED
#FC_WIDTH_EXPANDED
#FC_WIDTH_EXTRAEXPANDED
#FC_WIDTH_ULTRAEXPANDED
```

`File` Der Dateiname, der die Schriftart enthält.

`Index` Der Index der Schriftart innerhalb der Datei.

Dieser Befehl gibt eine neu erstellte Schriftart "Cairo" zurück. Löschen Sie sie mit cfontface:Free(), wenn Sie sie nicht mehr verwenden.

EINGABEN

`pattern` ein Fontconfig-Muster, das entweder als Zeichenkette oder Tabelle übergeben wird

RÜCKGABEWERTE

`font` eine neu erstellte Cairo-Schriftart.

4.3 cairo.FontOptions

BEZEICHNUNG

cairo.FontOptions – Erstellt ein neues Schriftartoptionsobjekt

ÜBERSICHT

```
handle = cairo.FontOptions()
```

BESCHREIBUNG

Ordnet ein neues Schriftartoptionsobjekt zu, wobei alle Optionen auf Standardwerte initialisiert werden.

Dieser Befehl gibt ein neu zugewiesenes Cairo-Schriftartoptionsobjekt zurück. Kann mit cfontoptions:Free() gelöscht werden. Er gibt immer ein gültiges Handle zurück. Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Fehlerobjekt zurückgegeben, bei dem alle Operationen auf dem Objekt nichts bewirken. Sie können dies mit cfontoptions:Status() überprüfen.

EINGABEN

keine

RÜCKGABEWERTE

`handle` ein neu zugewiesenes Cairo-Schriftartoptionsobjekt

4.4 cairo.Glyphs

BEZEICHNUNG

cairo.Glyphs – Erstellt ein neues Glyphen-Array

ÜBERSICHT

```
handle = cairo.Glyphs(n)
```

BESCHREIBUNG

Weist ein neues Glyphen-Array zu, das genügend Platz zum Speichern von `n` Glyphen bietet. Anschließend können Sie das einzelne Glyph an einem bestimmten Index festlegen und abrufen, indem Sie die Funktion cglyphs:Get() und cglyphs:Set() verwenden. Standardmäßig werden alle Glyphen-IDs und Offsets im Objekt auf 0 gesetzt.

Eine Schriftart ist (vereinfacht ausgedrückt) eine Sammlung von Formen, die zum Zeichnen von Text verwendet werden. Eine Glyphe ist eine dieser Formen. Für ein einzelnes Zeichen können mehrere Glyphen vorhanden sein (z.B. Alternativen zur Verwendung in unterschiedlichen Kontexten), oder eine Glyphe kann eine Ligatur mehrerer Zeichen sein. Cairo bietet keine Möglichkeit, Eingabetext in Glyphen umzuwandeln. Um also die Cairo-Schnittstellen zu verwenden, die Arrays von Glyphen akzeptieren, müssen Sie direkt auf das entsprechende zugrunde liegende Schriftartensystem zugreifen.

EINGABEN

keine

RÜCKGABEWERTE

`handle` ein neu zugewiesenes Cairo-Glyphen-Array

4.5 cairo.ImageSurface

BEZEICHNUNG

cairo.ImageSurface – Erstellt eine Bildoberfläche

ÜBERSICHT

```
handle = cairo.ImageSurface(format, width, height)
```

BESCHREIBUNG

Erstellt eine Bildoberfläche mit dem angegebenen Format und den angegebenen Abmessungen. Zunächst wird der Oberflächeninhalt auf 0 gesetzt. (In jedem Pixel ist insbesondere jede zum Format gehörende Farbe oder jeder Alphakanal 0. Der Inhalt von Bits innerhalb eines Pixels, die jedoch nicht zum angegebenen Format gehören, ist undefiniert.)

Das `format-Argument` kann eines der folgenden Pixelformate sein:

`#CAIRO_FORMAT_ARGB32`
: Jedes Pixel ist eine 32-Bit-Größe mit Alpha in den oberen 8 Bits, dann Rot, dann Grün und dann Blau. Die 32-Bit-Mengen werden nativ-endian gespeichert. Es wird vormultipliziertes Alpha verwendet. (Das heißt, 50 % transparentes Rot ist $80800000, nicht $80ff0000.

`#CAIRO_FORMAT_RGB24`
: Jedes Pixel ist eine 32-Bit-Menge, wobei die oberen 8 Bits ungenutzt bleiben. Rot, Grün und Blau werden in dieser Reihenfolge in den verbleibenden 24 Bits gespeichert.

`#CAIRO_FORMAT_A8`
: Jedes Pixel ist eine 8-Bit-Größe, die einen Alphawert enthält.

`#CAIRO_FORMAT_A1`
: Jedes Pixel ist eine 1-Bit-Größe, die einen Alphawert enthält. Pixel werden zu 32-Bit-Mengen zusammengepackt. Die Reihenfolge der Bits entspricht der Endianness der Plattform. Auf einer Big-Endian-Maschine befindet sich das erste Pixel im obersten Bit, auf einer Little-Endian-Maschine befindet sich das erste Pixel im niedrigstwertigen Bit.

`#CAIRO_FORMAT_RGB16_565`
: Jedes Pixel ist eine 16-Bit-Menge mit Rot in den oberen 5 Bits, Grün in den mittleren 6 Bits und Blau in den unteren 5 Bits.

`#CAIRO_FORMAT_RGB30`
: Wie RGB24, aber mit 10bpc.

Dieser Befehl gibt ein Handle für die neu erstellte Oberfläche zurück. Der Aufrufer hat die zurückgegebenen Oberfläche zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber er gibt ein Handle auf eine "Null"-Oberfläche zurück, wenn ein Fehler auftritt, z.B. nicht genügend Speicher. Sie können csurface:Status() verwenden, um dies zu überprüfen.

EINGABEN

`format` Format der Pixel in der zu erstellenden Oberfläche (siehe oben).

`width` Breite der Oberfläche in Pixel

`height` Höhe der Oberfläche in Pixel

RÜCKGABEWERTE

`handle` Bildoberfläche

4.6 cairo.Path

BEZEICHNUNG

cairo.Path – Konstruiert einen neuen Pfad

ÜBERSICHT

```
p = cairo.Path(table)
```

BESCHREIBUNG

Konstruiert einen neuen Cairo-Pfad aus einer Tabellenquelle. Die Tabelle kann mehrere Untertabellen enthalten, die jeweils ein Pfadelement beschreiben. Jede Untertabelle in `table` muss ein Feld `Type` enthalten, das den Pfadelementtyp der Untertabelle angibt. Zusätzliche Untertabellenelemente, die initialisiert werden müssen, hängen vom Elementtyp ab, der im Feld `Type` festgelegt ist.

Die folgenden Elemente werden für jede Untertabelle in `table` erkannt:

`Type` Beschreibt den Typ des Pfadelements. Dies kann einer der folgenden Typen sein:

`#CAIRO_PATH_MOVE_TO`
: Eine Verschiebungsoperation. Sie müssen auch die Felder `x1` und `y1` setzen.

`#CAIRO_PATH_LINE_TO`
: Eine Linienoperation. Sie müssen auch die Felder `x1` und `y1` setzen.

`#CAIRO_PATH_CURVE_TO`
: Eine Kurvenoperation. Sie müssen auch die Felder `x1`, `y1`, `x2`, `y2`, `x3` und `y3` festlegen.

`#CAIRO_PATH_CLOSE_PATH`
: Einen Operations-Pfad abschließen. Es müssen keine zusätzlichen Felder festgelegt werden.

`x1, y1`
: Kontrollpunkte werden von `#CAIRO_PATH_MOVE_TO`, `#CAIRO_PATH_LINE_TO`, und `#CAIRO_PATH_CURVE_TO` benötigt.

`x2, y2, x3, y3`
: Zusätzliche Kontrollpunkte, wird nur von `#CAIRO_PATH_CURVE_TO` benötigt.

EINGABEN

`table`
: Tabelle, die den Pfad beschreibt

RÜCKGABEWERTE

`p`
: ein neu zugewiesener Cairo-Pfad

4.7 cairo.ScaledFont

BEZEICHNUNG

cairo.ScaledFont – Erstellt eine skalierte Schriftart

ÜBERSICHT

```
font = cairo.ScaledFont(font_face, font_matrix, ctm, options)
```

BESCHREIBUNG

Erstellt ein skaliertes Cairo-Schriftartobjekt aus einer Schriftart und Matrizen, die die Größe der Schriftart und die Umgebung, in der sie verwendet wird, beschreiben. Das Argument `options` muss ein Cairo-Schriftartoptionshandle sein, das mit cairo.FontOptions() oder cfontoptions:Copy() erstellt wurde. Die Argumente `font_matrix` und `ctm` müssen Cairo-Matrizen sein.

Im einfachsten Fall einer `N`-Punktschriftart ist `font_matrix` nur eine Skalierung um `N`, kann aber auch verwendet werden, um die Schrift zu scheren oder entlang der beiden Achsen ungleichmäßig zu strecken.

Dieser Befehl gibt eine neu erstellte skalierte Cairo-Schriftart zurück, die mit cscaledfont:Free() gelöscht werden kann.

EINGABEN

`font_face`
: eine Schriftart aus Cairo

`font_matrix`
: Schriftartbereich-zu-Benutzerbereich-Transformationsmatrix für die Schriftart

`ctm`
: Benutzer-zu-Gerät-Transformationsmatrix, mit der die Schriftart verwendet wird

`options`
: Optionen, die beim Abrufen von Metriken für die Schriftart und beim Rendern damit verwendet werden sollen

RÜCKGABEWERTE

`font`
: eine neu erstellte Cairo-Schriftart

4.8 cairo.ImageSurfaceFromBrush

BEZEICHNUNG

cairo.ImageSurfaceFromBrush – Erstellt eine Bildoberfläche mit einem Hollywood-Pinsel

ÜBERSICHT

```
handle = cairo.ImageSurfaceFromBrush(id)
```

BESCHREIBUNG

Erstellt eine Bildoberfläche aus dem durch `id` angegebenen Hollywood-Pinsel. Wenn der Pinsel transparente Bereiche hat, verwendet die Bildoberfläche `#CAIRO_FORMAT_ARGB32`, andernfalls das Pixelformat `#CAIRO_FORMAT_RGB24`.

Dieser Befehl gibt ein Handle für die neu erstellte Oberfläche zurück. Der Aufrufer ordnete die Oberfläche zu und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

EINGABEN

`id`
: Hollywood-Pinsel zum Umwandeln in eine Bildoberfläche

RÜCKGABEWERTE

`handle`
: Bildoberfläche

4.9 cairo.Matrix

BEZEICHNUNG

cairo.Matrix – Erstellt eine Matrix

ÜBERSICHT

```
m = cairo.Matrix([xx, yx, xy, yy, x0, y0])
```

BESCHREIBUNG

Erstellt eine Matrix und initialisiert optional ihre affine Transformation auf die durch `xx`, `yx`, `xy`, `yy`, `x0`, `y0` angegebenen Koeffizienten. Ausgelassene Koeffizienten werden auf 0 gesetzt.

EINGABEN

`xx`
: optional: xx Komponente der affinen Transformation (Standardwert 0)

`yx`	optional: yx Komponente der affinen Transformation (Standardwert 0)
`xy`	optional: xy Komponente der affinen Transformation (Standardwert 0)
`yy`	optional: yy Komponente der affinen Transformation (Standardwert 0)
`x0`	optional: X Verschiebungskomponente der affinen Transformation (Standardwert 0)
`y0`	optional: Y Verschiebungskomponente der affinen Transformation (Standardwert 0)

RÜCKGABEWERTE

`m`	Matrixobjekt

4.10 cairo.MatrixIdentity

BEZEICHNUNG

cairo.MatrixIdentity – Erstellt eine Identitätsmatrix

ÜBERSICHT

```
m = cairo.MatrixIdentity()
```

BESCHREIBUNG

Erstellt eine Matrix und initialisiert ihre affine Transformation in eine Identitätstransformation.

EINGABEN

keine

RÜCKGABEWERTE

`m`	Identitätsmatrixobjekt

4.11 cairo.PatternForSurface

BEZEICHNUNG

cairo.PatternForSurface – Erstellt ein Muster für die Oberfläche

ÜBERSICHT

```
pat = cairo.PatternForSurface(surface)
```

BESCHREIBUNG

Erstellt ein neues Cairo-Muster für die angegebene Oberfläche.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

`surface` die Oberfläche

RÜCKGABEWERTE

`pat` das neu geschaffene Cairo-Muster

4.12 cairo.PatternLinear

BEZEICHNUNG

cairo.PatternLinear – Erstellt ein lineares Muster

ÜBERSICHT

```
pat = cairo.PatternLinear(x0, y0, x1, y1)
```

BESCHREIBUNG

Erstellt ein neues Cairo-Muster mit linearem Farbverlauf entlang der durch (x0, y0) und (x1, y1) definierten Linie. Vor der Verwendung des Verlaufsmusters sollten mit cpattern:AddColorStopRGB() oder cpattern:AddColorStopRGBA() eine Reihe von Farbstopps definiert werden.

Hinweis: Die Koordinaten liegen hier im Musterbereich. Für ein neues Muster ist der Musterbereich identisch mit dem Benutzerbereich, aber die Beziehung zwischen den Räumen kann mit cpattern:SetMatrix() geändert werden.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

`x0` x Koordinate des Startpunkts

`y0` y Koordinate des Startpunkts

`x1` x Koordinate des Endpunktes

`y1` y Koordinate des Endpunktes

RÜCKGABEWERTE

`pat` das neu geschaffene Cairo-Muster

4.13 cairo.PatternMesh

BEZEICHNUNG

cairo.PatternMesh – Erstellt ein Netzmuster

ÜBERSICHT

```
pat = cairo.PatternMesh()
```

BESCHREIBUNG

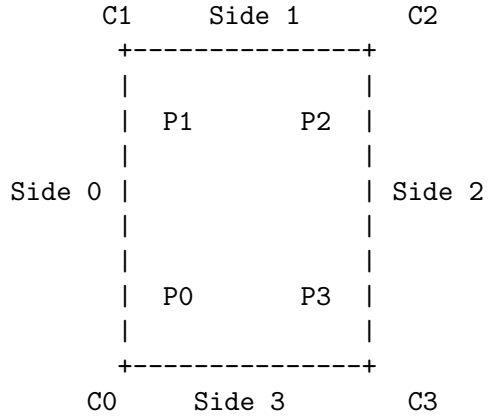
Erstellt ein neues Netzmuster.

Netzmuster sind Tensorprodukt-Patchnetze (Schattierungen vom Typ 7 im PDF). Netzmuster können auch verwendet werden, um andere Arten von Schattierungen zu erstellen, bei denen es sich um Spezialfälle von Tensorprodukt-Patchnetzen handelt, z.B. Coons-Patchnetze (Schattierung vom Typ 6 in PDF) und Gouraud-schattierte Dreiecksnetze (Schattierung vom Typ 4 und 5 in PDF).

Netzmuster bestehen aus einem oder mehreren Tensorprodukt-Patches, die vor der Verwendung des Netzmusters definiert werden sollten. Die Verwendung eines Netzmusters mit einem teilweise definierten Patch als Quelle oder Maske führt dazu, dass der Kontext in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt wird.

Ein Tensorprodukt-Patch wird durch 4 Bézier-Kurven (Seite 0, 1, 2, 3) und durch 4 zusätzliche Kontrollpunkte (P0, P1, P2, P3) definiert, die eine weitere Kontrolle über den Patch ermöglichen und die Definition des Tensorprodukt-Patches vervollständigen. Die Ecke C0 ist der erste Punkt des Patches.

Degenerierte Seiten sind zulässig, daher können gerade Linien verwendet werden. Eine Linie mit der Länge Null auf einer Seite kann zum Erstellen dreiseitiger Patches verwendet werden.



Jeder Patch wird erstellt, indem zuerst cpattern:BeginPatch() und dann cpattern:MoveTo() aufgerufen werden, um den ersten Punkt im Patch (C0) anzugeben. Anschließend werden die Seiten mit Aufrufen von cpattern:CurveTo() und cpattern:LineTo() angegeben.

Die vier zusätzlichen Kontrollpunkte (P0, P1, P2, P3) in einem Patch können mit cpattern:SetControlPoint() angegeben werden.

An jeder Ecke des Patches (C0, C1, C2, C3) kann mit cpattern:SetCornerColorRGB() oder cpattern:SetCornerColorRGBA() eine Farbe angegeben werden. Jede Ecke, deren Farbe nicht explizit angegeben ist, wird standardmäßig auf transparentes Schwarz gesetzt.

Ein Coons-Patch ist ein Sonderfall des Tensorprodukt-Patches, bei dem die Kontrollpunkte implizit durch die Seiten des Patches definiert werden. Der Standardwert für alle nicht angegebenen Kontrollpunkte ist der implizite Wert für einen Coons-Patch.

Wenn also keine Kontrollpunkte angegeben sind, handelt es sich bei dem Patch um einen Coons-Patch.

Ein Dreieck ist ein Sonderfall des Tensorprodukt-Patches, bei dem die Kontrollpunkte implizit durch die Seiten des Patches definiert werden, alle Seiten Linien sind und eine davon die Länge 0 hat, d.h. wenn der Patch mit nur 3 Linien angegeben wird, es ein Dreieck ist. Wenn die durch die 0-Längenseite verbundenen Ecken die gleiche Farbe haben, ist der Patch ein Gouraud-schattiertes Dreieck.

Patches können anders als im obigen Diagramm ausgerichtet sein. Beispielsweise könnte der erste Punkt oben links liegen. Das Diagramm zeigt nur die Beziehung zwischen den Seiten, Ecken und Kontrollpunkten. Unabhängig davon, wo sich der erste Punkt befindet, ist beim Festlegen von Farben immer Ecke 0 der erste Punkt, Ecke 1 der Punkt zwischen Seite 0 und Seite 1 usw.

Durch den Aufruf von cpattern:EndPatch() wird der aktuelle Patch abgeschlossen. Wenn weniger als 4 Seiten definiert wurden, wird die erste fehlende Seite als Linie vom aktuellen Punkt zum ersten Punkt des Patches (C0) definiert und die anderen Seiten sind degenerierte Linien von C0 bis C0. Die Ecken zwischen den hinzugefügten Seiten stimmen alle mit C0 des Patches überein und ihre Farbe wird auf die gleiche Farbe wie C0 eingestellt.

Weitere Patches können durch zusätzliche Aufrufe hinzugefügt werden cpattern:BeginPatch() / cpattern:EndPatch().

```
pat = cairo.PatternMesh()

; Coons-Patch hinzufügen
pat:BeginPatch()
pat:MoveTo(0, 0)
pat:CurveTo(30, -30, 60, 30, 100, 0)
pat:CurveTo(60, 30, 130, 60, 100, 100)
pat:CurveTo(60, 70, 30, 130,  0, 100)
pat:CurveTo(30, 70, -30, 30,  0, 0)
pat:SetCornerColorRGB(0, 1, 0, 0)
pat:SetCornerColorRGB(1, 0, 1, 0)
pat:SetCornerColorRGB(2, 0, 0, 1)
pat:SetCornerColorRGB(3, 1, 1, 0)
pat:EndPatch()

; Gouraud-schattiertes Dreieck hinzufügen
pat:BeginPatch()
pat:MoveTo(100, 100)
pat:LineTo(130, 130)
pat:LineTo(130, 70)
pat:SetCornerColorRGB(0, 1, 0, 0)
pat:SetCornerColorRGB(1, 0, 1, 0)
pat:SetCornerColorRGB(2, 0, 0, 1)
pat:EndPatch()
```

Wenn sich zwei Patches überlappen, wird der zuletzt hinzugefügte Patch über den ersten gezeichnet.

Wenn sich ein Patch über sich selbst faltet, werden die Punkte anhand ihrer Parameterkoordinaten innerhalb des Patches sortiert. Die v-Koordinate reicht von 0 bis 1, wenn man sich von Seite 3 zu Seite 1 bewegt; Die U-Koordinate reicht von 0 bis 1, wenn man von Seite 0 zu Seite 2 wechselt. Punkte mit höheren V-Koordinaten verbergen Punkte mit niedrigeren V-Koordinaten. Wenn zwei Punkte die gleiche v-Koordinate haben, liegt der Punkt mit der höheren u-Koordinate oben. Dies bedeutet, dass Punkte, die näher an Seite 1 liegen, über Punkten liegen, die näher an Seite 3 liegen. Wenn dies nicht ausreicht, um zu entscheiden, welcher Punkt darüber liegt (z.B. wenn beide Punkte zu Seite 1 oder Seite 3 gehören), liegen Punkte, die näher an Seite 2 liegen, über Punkten, die näher an Seite 0 liegen.

Eine vollständige Definition von Tensorprodukt-Patches finden Sie in der PDF-Spezifikation (ISO32000), in der die Parametrisierung ausführlich beschrieben wird.

Hinweis: Die Koordinaten liegen immer im Musterbereich. Für ein neues Muster ist der Musterbereich identisch mit dem Benutzerbereich, aber die Beziehung zwischen den Räumen kann mit cpattern:SetMatrix() geändert werden.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

keine

RÜCKGABEWERTE

`pat` das neu geschaffene Cairo-Muster

4.14 cairo.PatternRadial

BEZEICHNUNG

cairo.PatternRadial – Erstellt ein radiales Muster

ÜBERSICHT

```
pat = cairo.PatternRadial(cx0, cy0, radius0, cx1, cy1, radius1)
```

BESCHREIBUNG

Erstellt ein neues Cairo-Muster mit radialem Farbverlauf zwischen den beiden durch (cx0, cy0, radius0) und (cx1, cy1, radius1) definierten Kreisen. Vor der Verwendung des Verlaufsmusters sollten mit cpattern:AddColorStopRGB() oder cpattern:AddColorStopRGBA() eine Reihe von Farbstopps definiert werden.

Hinweis: Die Koordinaten liegen hier im Musterbereich. Für ein neues Muster ist der Musterbereich identisch mit dem Benutzerbereich, aber die Beziehung zwischen den Räumen kann mit cpattern:SetMatrix() geändert werden.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

`cx0`	x Koordinate für den Mittelpunkt des Startkreises
`cy0`	y Koordinate für den Mittelpunkt des Startkreises
`radius0`	Radius des Startkreises
`cx1`	x Koordinate für den Mittelpunkt des Endkreises
`cy1`	y Koordinate für den Mittelpunkt des Endkreises
`radius1`	Radius des Endkreises

RÜCKGABEWERTE

`x`	das neu geschaffene Cairo-Muster

4.15 cairo.PatternRGB

BEZEICHNUNG

cairo.PatternRGB – Erstellt ein RGB-Muster

ÜBERSICHT

```
pat = cairo.PatternRGB(red, green, blue)
```

BESCHREIBUNG

Erstellt ein neues Cairo-Muster, das einer undurchsichtigen Farbe entspricht. Die Farbkomponenten sind Gleitkommazahlen im Bereich von 0 bis 1. Wenn die übergebenen Werte außerhalb dieses Bereichs liegen, werden sie geklemmt.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGB() angegeben.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

`red`	Rotanteil der Farbe
`green`	Grünanteil der Farbe
`blue`	Blauanteil der Farbe

RÜCKGABEWERTE

`pat`	das neu geschaffene Cairo-Muster

4.16 cairo.PatternRGBA

BEZEICHNUNG

cairo.PatternRGBA – Erstellt ein RGBA-Muster

ÜBERSICHT

```
pat = cairo.PatternRGBA(red, green, blue, alpha)
```

BESCHREIBUNG

Erstellt ein neues Cairo-Muster, das einer durchscheinenden Farbe entspricht. Die Farbkomponenten sind Gleitkommazahlen im Bereich von 0 bis 1. Wenn die übergebenen Werte außerhalb dieses Bereichs liegen, werden sie geklemmt.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGB() angegeben.

Dieser Befehl gibt bei Erfolg das neu erstellte Cairo-Muster zurück oder ein Fehlermuster, wenn kein Speicher vorhanden ist. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber wenn ein Fehler aufgetreten ist, wird der Musterstatus auf einen Fehler gesetzt. Um den Status eines Musters zu überprüfen, verwenden Sie cpattern:Status().

EINGABEN

`red`	Rotanteil der Farbe
`green`	Grünanteil der Farbe
`blue`	Blauanteil der Farbe
`alpha`	Alpha-Komponente der Farbe

RÜCKGABEWERTE

`pat`	das neu geschaffene Cairo-Muster

4.17 cairo.PDFSurface

BEZEICHNUNG

cairo.PDFSurface – Erstellt eine PDF-Oberfläche

ÜBERSICHT

```
handle = cairo.PDFSurface(filename$, width_in_points, height_in_points)
```

BESCHREIBUNG

Erstellt eine PDF-Oberfläche mit der angegebenen Größe in Punkten, die in `filename$` geschrieben werden soll.

Dieser Befehl gibt ein Handle für die neu erstellte Oberfläche zurück. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber sie gibt ein Handle auf eine "Null"-Oberfläche zurück, wenn ein Fehler auftritt, z.B. nicht genügend Speicher. Sie können csurface:Status() verwenden, um dies zu überprüfen.

EINGABEN

`filename$`
: ein Dateiname für die PDF-Ausgabe (muss beschreibbar sein).

`width_in_points`
: Breite der Oberfläche in Punkten (1 Punkt == 1/72,0 Zoll)

`height_in_points`
: Höhe der Oberfläche in Punkten (1 Punkt == 1/72,0 Zoll)

RÜCKGABEWERTE

`handle` PDF-Oberfläche

4.18 cairo.PDFVersionToString

BEZEICHNUNG

cairo.PDFVersionToString – Gibt die Zeichenkette der PDF-Version zurück

ÜBERSICHT

```
s$ = cairo.PDFVersionToString(version)
```

BESCHREIBUNG

Ruft die Zeichenkettendarstellung der in `version` angegebenen ID ab. Dieser Befehl gibt Null zurück, wenn `version` nicht gültig ist. `version` kann eine der folgenden Konstanten sein:

```
#CAIRO_PDF_VERSION_1_4
#CAIRO_PDF_VERSION_1_5
```

Dieser Befehl gibt die Zeichenkette zurück, die der angegebenen Version zugeordnet ist.

EINGABEN

`version` eine Versions-ID

RÜCKGABEWERTE

`s$` Zeichenkettendarstellung der angegebenen Version

4.19 cairo.Region

BEZEICHNUNG

cairo.Region – Erstellt eine Region

ÜBERSICHT

```
reg = cairo.Region([rects])
```

BESCHREIBUNG

Weist ein neues Regionsobjekt zu und initialisiert es optional für die Vereinigung aller angegebenen `rects`. Wenn angegeben, muss der Parameter `rects` eine Tabelle sein, die eine beliebige Anzahl von Untertabellen enthält, die jeweils ein Rechteck beschreiben. Für die Untertabellen müssen die Felder `x`, `y`, `width` und `height` initialisiert sein. Wenn der Parameter `rects` weggelassen wird, wird ein leerer Bereich erstellt.

Dieser Befehl gibt eine neu zugewiesene Cairo Region zurück. Wird gelöscht mit cregion:Free(). Dieser Befehl gibt immer ein gültiges Handle zurück. Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Fehlerobjekt zurückgegeben, bei dem alle Operationen auf dem Objekt nichts bewirken. Sie können dies mit cregion:Status() überprüfen.

EINGABEN

`rects` Optional: Tabelle mit Rechtecken, mit der die Region initialisiert werden soll

RÜCKGABEWERTE

`reg` eine neu zugewiesene Cairo Region

4.20 cairo.StatusToString

BEZEICHNUNG

cairo.StatusToString – Gibt eine Zeichenkette aus dem Statuscode zurück

ÜBERSICHT

```
s$ = cairo.StatusToString(status)
```

BESCHREIBUNG

Bietet eine für Menschen lesbare Beschreibung des durch `status` angegebenen Statuscodes. Eine Liste der Statuscodes finden Sie unter ccontext:Status().

EINGABEN

`status` ein Cairo-Statuscode

RÜCKGABEWERTE

`s$` Zeichenkettendarstellung des Statuscodes

4.21 cairo.SVGSurface

BEZEICHNUNG

cairo.SVGSurface – Erstellt eine SVG-Oberfläche

ÜBERSICHT

```
handle = cairo.SVGSurface(filename$, width_in_points, height_in_points)
```

BESCHREIBUNG

Erstellt eine SVG-Oberfläche mit der angegebenen Größe in Punkten, die in `filename$` geschrieben werden soll.

Das SVG-Oberflächen-Backend erkennt die folgenden MIME-Typen für die an eine Oberfläche angehängten Daten (siehe csurface:SetMimeData()), wenn sie als Quellmuster zum Zeichnen auf dieser Oberfläche verwendet werden:

```
#CAIRO_MIME_TYPE_JPEG
#CAIRO_MIME_TYPE_PNG
#CAIRO_MIME_TYPE_URI
```

Wenn einer von ihnen angegeben ist, gibt das SVG-Backend den Inhalt der MIME-Daten anstelle eines Oberflächen-Snapshots (PNG, Base64-kodiert) im entsprechenden Bild-Tag aus.

Zunächst wird der inoffizielle MIME-Typ `#CAIRO_MIME_TYPE_URI` untersucht. Falls vorhanden, wird der URI unverändert ausgegeben: Die Sicherstellung der Richtigkeit des URI bleibt dem Clientcode überlassen.

Wenn `#CAIRO_MIME_TYPE_URI` nicht vorhanden ist, aber `#CAIRO_MIME_TYPE_JPEG` oder `#CAIRO_MIME_TYPE_PNG` angegeben ist, werden die entsprechenden Daten Base64-codiert und ausgegeben.

Wenn `#CAIRO_MIME_TYPE_UNIQUE_ID` vorhanden ist, werden alle Oberflächen mit derselben eindeutigen Kennung nur einmal eingebettet.

Dieser Befehl gibt ein Handle für die neu erstellte Oberfläche zurück. Der Aufrufer hat die zurückgegebenen Oberfläche zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Dieser Befehl gibt immer ein gültiges Handle zurück, aber sie gibt ein Handle auf eine "Null"-Oberfläche zurück, wenn ein Fehler auftritt, z.B. nicht genügend Speicher. Sie können csurface:Status() verwenden, um dies zu überprüfen.

EINGABEN

`filename$`
: ein Dateiname für die SVG-Ausgabe (muss beschreibbar sein)

`width_in_points`
: Breite der Oberfläche in Punkten (1 Punkt == 1/72,0 Zoll)

`height_in_points`
: Höhe der Oberfläche in Punkten (1 Punkt == 1/72,0 Zoll)

RÜCKGABEWERTE

`handle` SVG-Oberfläche

4.22 cairo.SVGVersionToString

BEZEICHNUNG

cairo.SVGVersionToString – Holt sich die Zeichenkette aus der SVG-Version

ÜBERSICHT

```
s$ = cairo.SVGVersionToString(version)
```

BESCHREIBUNG

Ruft die Zeichenkettendarstellung der in `version` angegebenen ID ab. Dieser Befehl gibt Null zurück, wenn `version` nicht gültig ist. `version` kann eine der folgenden Konstanten sein:

```
#CAIRO_SVG_VERSION_1_1
#CAIRO_SVG_VERSION_1_2
```

Dieser Befehl gibt die Zeichenkette zurück, die der angegebenen Version zugeordnet ist.

EINGABEN

`version` eine Versions-ID

RÜCKGABEWERTE

`s$` Zeichenkettendarstellung der angegebenen Version

4.23 cairo.ToyFontFace

BEZEICHNUNG

cairo.ToyFontFace – Erstellt eine Toy-Schriftart

ÜBERSICHT

```
font = cairo.ToyFontFace(family$, slant, weight)
```

BESCHREIBUNG

Erstellt eine Schriftart aus einem Triplett aus `family$`, `slant` und `weight`. Diese Schriftarten werden bei der Implementierung der Cairo-Kontext-"Toy"-Schriftart-API verwendet.

Der Parameter-slant kann eine der folgenden Konstanten sein:

```
#CAIRO_FONT_SLANT_NORMAL
#CAIRO_FONT_SLANT_ITALIC
#CAIRO_FONT_SLANT_OBLIQUE
```

Der Parameter `weight` kann eine der folgenden Konstanten sein:

```
#CAIRO_FONT_WEIGHT_NORMAL
#CAIRO_FONT_WEIGHT_BOLD
```

Wenn `family$` die Zeichenkette mit der Länge Null `""` ist, wird die plattformspezifische Standardfamilie angenommen. Die Standardfamilie kann dann mit cfontface:GetFamily() abgefragt werden.

Die Funktion ccontext:SelectFontFace() verwendet dies, um Schriftarten zu erstellen. Einschränkungen und andere Details zu Toy-Schriftarten finden Sie in dieser Funktion.

Dieser Befehl gibt eine neu erstellte Schriftart "Cairo" zurück. Wird gelöscht mit cfontface:Free(), wenn Sie sie nicht mehr verwenden.

EINGABEN

`family$` ein Schriftfamilienname

`slant` die Neigung für die Schriftart (siehe oben)

`weight` die Stärke der Schriftart (siehe oben)

RÜCKGABEWERTE

`font` eine neu erstellte Schriftart "Cairo"

4.24 cairo.Version

BEZEICHNUNG

cairo.Version – Ermittelt die Cairo-Version

ÜBERSICHT

```
ver$ = cairo.MatrixIdentity()
```

BESCHREIBUNG

Gibt die Version der Cairo-Bibliothek als für Menschen lesbare Zeichenkette der Form "X.Y.Z" zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`ver$` von Pangomonium verwendete Cairo Version

5 Cairo-Kontext

5.1 ccontext:AppendPath

BEZEICHNUNG

ccontext:AppendPath – Hängt einen Pfad an

ÜBERSICHT

```
ccontext:AppendPath(path)
```

BESCHREIBUNG

Hängt `path` an den aktuellen Pfad an. `path` kann entweder der Rückgabewert von ccontext:CopyPath() oder ccontext:CopyPathFlat() sein oder er kann mit cairo.Path() erstellt werden.

EINGABEN

`path`	Pfad, der angehängt werden soll

5.2 ccontext:Arc

BEZEICHNUNG

ccontext:Arc – Fügt einen zunehmenden Bogen zum aktuellen Pfad hinzu

ÜBERSICHT

```
ccontext:Arc(xc, yc, radius, angle1, angle2)
```

BESCHREIBUNG

Fügt dem aktuellen Pfad einen Kreisbogen mit dem angegebenen `radius` hinzu. Der Bogen ist bei (`xc`, `yc`) zentriert, beginnt bei `angle1` und verläuft in Richtung zunehmender Winkel, um bei `angle2` zu enden. Wenn `angle2` kleiner als `angle1` ist, wird er schrittweise um 2*#PI erhöht, bis er größer als `Winkel1` ist.

Wenn ein aktueller Punkt vorhanden ist, wird dem Pfad ein erstes Liniensegment hinzugefügt, um den aktuellen Punkt mit dem Anfang des Bogens zu verbinden. Wenn diese Anfangslinie unerwünscht ist, kann sie vermieden werden, indem ccontext:NewSubPath() vor dem Aufruf von ccontext:Arc() aufgerufen wird.

Winkel werden im Bogenmaß gemessen. Ein Winkel von 0,0 liegt in Richtung der positiven X-Achse (im Benutzerbereich). Ein Winkel von #PI/2,0 Bogenmaß (90 Grad) liegt in Richtung der positiven Y-Achse (im Benutzerbereich). Die Winkel nehmen in Richtung von der positiven X-Achse zur positiven Y-Achse zu. Bei der Standardtransformationsmatrix nehmen die Winkel also im Uhrzeigersinn zu.

Um von Grad in Bogenmaß umzurechnen, verwenden Sie Grad * `#PI` / 180.

Diese Funktion gibt den Bogen in Richtung zunehmender Winkel an. Sehen Sie sich ccontext:ArcNegative() an, um den Bogen in Richtung abnehmender Winkel zu erhalten.

Der Bogen ist im Benutzerbereich kreisförmig. Um einen elliptischen Bogen zu erhalten, können Sie die aktuelle Transformationsmatrix um unterschiedliche Beträge in X- und

Y-Richtung skalieren. Zum Beispiel, um eine Ellipse in das durch `x`, `y`, `width`, `height` gegebene Feld zu zeichnen:

```
ctx:Save()
ctx:Translate(x + width / 2, y + height / 2)
ctx:Scale(width / 2, height / 2)
ctx:Arc(0, 0, 1, 0, 2 * #PI)
ctx:Restore()
```

EINGABEN

`xc`	X Position des Mittelpunkts des Bogens
`yc`	Y Position des Mittelpunkts des Bogens
`radius`	der Radius des Bogens
`angle1`	der Startwinkel im Bogenmaß
`angle2`	der Endwinkel im Bogenmaß

5.3 ccontext:ArcNegative

BEZEICHNUNG

ccontext:ArcNegative – Fügt einen abnehmenden Bogen zum aktuellen Pfad hinzu

ÜBERSICHT

`ccontext:ArcNegative(xc, yc, radius, angle1, angle2)`

BESCHREIBUNG

Fügt dem aktuellen Pfad einen Kreisbogen mit dem angegebenen `radius` hinzu. Der Bogen ist bei (`xc`, `yc`) zentriert, beginnt bei `angle1` und verläuft in Richtung abnehmender Winkel, um bei `angle2` zu enden. Wenn `angle2` größer als `angle1` ist, wird er schrittweise um 2*#PI verringert, bis er kleiner als `angle1` ist.

Weitere Details finden Sie unter ccontext:Arc(). Diese Funktion unterscheidet sich nur in der Richtung des Bogens zwischen den beiden Winkeln.

EINGABEN

`xc`	X Position des Mittelpunkts des Bogens
`yc`	Y Position des Mittelpunkts des Bogens
`radius`	der Radius des Bogens
`angle1`	der Startwinkel im Bogenmaß
`angle2`	der Endwinkel im Bogenmaß

5.4 ccontext:Clip

BEZEICHNUNG

ccontext:Clip – Legt eine neue Clip-Region fest

ÜBERSICHT

```
ccontext:Clip()
```

BESCHREIBUNG

Erstellt einen neuen Clip-Bereich, indem er den aktuellen Clip-Bereich mit dem aktuellen Pfad schneidet, wie er von ccontext:Fill() und gemäß der aktuellen Füllregel gefüllt würde (siehe ccontext:SetFillRule()).

Nach ccontext:Clip() wird der aktuelle Pfad aus dem Cairo-Kontext gelöscht.

Der aktuelle Clip-Bereich wirkt sich auf alle Zeichenvorgänge aus, indem er alle Änderungen an der Oberfläche, die außerhalb des aktuellen Clip-Bereichs liegen, effektiv ausblendet.

Durch den Aufruf von ccontext:Clip() kann der Clipbereich nur kleiner und niemals größer gemacht werden. Da der aktuelle Clip jedoch Teil des Grafikstatus ist, kann eine vorübergehende Einschränkung des Clipbereichs durch den Aufruf von ccontext:Clip() innerhalb eines ccontext:Save() / ccontext:Restore()-Paar erreicht werden. Die einzige andere Möglichkeit, die Größe des Clip-Bereichs zu erhöhen, ist ccontext:ResetClip().

EINGABEN

Keine

5.5 ccontext:ClipExtents

BEZEICHNUNG

ccontext:ClipExtents – Gibt die Clip-Ausdehnungen zurück

ÜBERSICHT

```
x1, y1, x2, y2 = ccontext:ClipExtents()
```

BESCHREIBUNG

Berechnet einen Begrenzungsrahmen in Benutzerkoordinaten, der den Bereich innerhalb des aktuellen Clips abdeckt.

EINGABEN

Keine

RÜCKGABEWERTE

`x1`	links von den resultierenden Ausmaßen
`y1`	oberhalb der resultierenden Ausmaße
`x2`	rechts der resultierenden Ausmaßen
`y2`	unterhalb der resultierenden Ausmaße

5.6 ccontext:ClipPreserve

BEZEICHNUNG

ccontext:ClipPreserve – Legt einen neuen Clipbereich fest und behält den Pfad bei

ÜBERSICHT

`ccontext:ClipPreserve()`

BESCHREIBUNG

Erstellt einen neuen Clip-Bereich, indem er den aktuellen Clip-Bereich mit dem aktuellen Pfad schneidet, wie er von ccontext:Fill() und gemäß der aktuellen Füllregel gefüllt würde (siehe ccontext:SetFillRule()).

Im Gegensatz zu ccontext:Clip() behält ccontext:ClipPreserve() den Pfad im Cairo-Kontext bei.

Der aktuelle Clip-Bereich wirkt sich auf alle Zeichenvorgänge aus, indem er alle Änderungen an der Oberfläche, die außerhalb des aktuellen Clip-Bereichs liegen, effektiv ausblendet.

Der Aufruf von ccontext:ClipPreserve() kann den Clipbereich nur verkleinern, niemals vergrößern. Da der aktuelle Clip jedoch Teil des Grafikstatus ist, kann eine vorübergehende Einschränkung des Clipbereichs durch den Aufruf von ccontext:ClipPreserve() innerhalb eines ccontext:Save() / ccontext:Restore()-Paar erreicht werden. Die einzige andere Möglichkeit, die Größe des Clip-Bereichs zu erhöhen, ist ccontext:ResetClip().

EINGABEN

Keine

5.7 ccontext:ClosePath

BEZEICHNUNG

ccontext:ClosePath – Schliesst den Pfad

ÜBERSICHT

`ccontext:ClosePath()`

BESCHREIBUNG

Fügt dem Pfad ein Liniensegment vom aktuellen Punkt bis zum Anfang des aktuellen Unterpfads hinzu (der zuletzt an ccontext:MoveTo() übergebene Punkt) und schließt diesen Unterpfad. Nach diesem Aufruf befindet sich der aktuelle Punkt am verbundenen Endpunkt des Unterpfads.

Das Verhalten von ccontext:ClosePath() unterscheidet sich vom einfachen Aufruf von ccontext:LineTo() mit der entsprechenden Koordinate beim Zeichnen. Wenn ein geschlossener Unterpfad gestrichelt wird, gibt es an den Enden des Unterpfads keine Kappen. Stattdessen gibt es eine Linienverbindung, die das letzte und das anfängliche Segment des Unterpfads verbindet.

Wenn vor dem Aufruf von ccontext:ClosePath() kein aktueller Punkt vorhanden ist, hat diese Funktion keine Auswirkung.

Hinweis: Jeder Aufruf von ccontext:ClosePath() platziert ein explizites Element `MOVE_TO` im Pfad unmittelbar nach dem Element `CLOSE_PATH`, was beispielsweise in ccontext:CopyPath() zu sehen ist. Dies kann in einigen Fällen die Pfadverarbeitung vereinfachen, da es möglicherweise nicht erforderlich ist, den "letzten move_to-Punkt" während der Verarbeitung zu speichern, da `MOVE_TO` unmittelbar nach `CLOSE_PATH` diesen Punkt bereitstellt.

EINGABEN
Keine

5.8 ccontext:CopyPage

BEZEICHNUNG
ccontext:CopyPage – Kopiert eine Seite

ÜBERSICHT
```
ccontext:CopyPage()
```

BESCHREIBUNG
Gibt die aktuelle Seite für Backends aus, die mehrere Seiten unterstützen, löscht sie jedoch nicht, sodass der Inhalt der aktuellen Seite auch für die nächste Seite beibehalten wird. Verwenden Sie ccontext:ShowPage(), wenn Sie nach der Ausgabe eine leere Seite erhalten möchten.

Dies ist eine praktische Funktion, die einfach csurface:CopyPage() für das Ziel des Cairo-Kontexts aufruft.

EINGABEN
Keine

5.9 ccontext:CopyPath

BEZEICHNUNG
ccontext:CopyPath – Kopiert einen Pfad

ÜBERSICHT
```
handle = ccontext:CopyPath()
```

BESCHREIBUNG
Erstellt eine Kopie des aktuellen Pfads und gibt sie als Cairo-Pfadobjekt an den Benutzer zurück. Verwenden Sie cpath:Get(), um die einzelnen Pfadelemente abzurufen.

Diese Funktion gibt immer ein gültiges Handle zurück, das Ergebnis enthält jedoch keine Daten, wenn eine der folgenden Bedingungen zutrifft:

1. Wenn nicht genügend Speicher zum Kopieren des Pfads vorhanden ist. In diesem Fall wird der Pfadstatus auf `#CAIRO_STATUS_NO_MEMORY` gesetzt.
2. Wenn sich der Cairo-Kontext bereits in einem Fehlerzustand befindet. In diesem Fall enthält der Pfadstatus denselben Status, der von ccontext:Status() zurückgegeben würde.

Diese Funktion gibt die Kopie des aktuellen Pfads zurück. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpath:Free() aufrufen, wenn er damit fertig ist.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die Kopie des aktuellen Pfades

5.10 ccontext:CopyPathFlat

BEZEICHNUNG

ccontext:CopyPathFlat – Kopiert die vereinfachte Version des Pfads

ÜBERSICHT

```
handle = ccontext:CopyPathFlat()
```

BESCHREIBUNG

Ruft eine vereinfachte Kopie des aktuellen Pfads ab und gibt sie als Cairo-Pfadobjekt an den Benutzer zurück. Verwenden Sie cpath:Get(), um die einzelnen Pfadelemente abzurufen.

Diese Funktion ähnelt ccontext:CopyPath(), außer dass alle Kurven im Pfad mit stückweise linearen Näherungen angenähert werden (genau innerhalb des aktuellen Toleranzwerts). Das heißt, das Ergebnis enthält garantiert keine Elemente vom Typ `#CAIRO_PATH_CURVE_TO`, sondern wird stattdessen durch eine Reihe von #CAIRO_PATH_LINE_TO-Elementen ersetzt.

Diese Funktion gibt immer ein gültiges Handle zurück, das Ergebnis enthält jedoch keine Daten, wenn eine der folgenden Bedingungen zutrifft:

1. Wenn nicht genügend Speicher zum Kopieren des Pfads vorhanden ist. In diesem Fall wird der Pfadstatus auf `#CAIRO_STATUS_NO_MEMORY` gesetzt.
2. Wenn sich der Cairo-Kontext bereits in einem Fehlerzustand befindet. In diesem Fall enthält der Pfadstatus denselben Status, der von ccontext:Status() zurückgegeben würde.

Diese Funktion gibt die Kopie des aktuellen Pfads zurück. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte cpath:Free() aufrufen, wenn er damit fertig ist.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die Kopie des aktuellen Pfades

5.11 ccontext:CurveTo

BEZEICHNUNG

ccontext:CurveTo – Fügt dem Pfad einen kubischen Bézier-Spline hinzu

ÜBERSICHT

```
ccontext:CurveTo(x1, y1, x2, y2, x3, y3)
```

BESCHREIBUNG

Fügt einen kubischen Bézier-Spline zum Pfad vom aktuellen Punkt zur Position (`x3`, `y3`) in Benutzerbereichkoordinaten hinzu, wobei (`x1`, `y1`) und (`x2`, `y2`) als Kontrollpunkte verwendet werden. Nach diesem Aufruf ist der aktuelle Punkt (`x3`, `y3`).

Wenn es vor dem Aufruf von ccontext:CurveTo() keinen aktuellen Punkt gibt, verhält sich Diese Funktion so, als ob ihr ein Aufruf von ccontext:MoveTo() mit `x1` und `y1` vorausgegangen wäre.

EINGABEN

`x1`	die X-Koordinate des ersten Kontrollpunkts y1 -die Y-Koordinate des ersten Kontrollpunkts
`x2`	die X-Koordinate des zweiten Kontrollpunkts
`y2`	die Y-Koordinate des zweiten Kontrollpunkts
`x3`	die X-Koordinate des Endes der Kurve
`y3`	die Y-Koordinate des Endes der Kurve

5.12 ccontext:DeviceToUser

BEZEICHNUNG

ccontext:DeviceToUser – Transformiert den Geräte- in den Benutzerbereich

ÜBERSICHT

```
ux, uy = ccontext:DeviceToUser(dx, dy)
```

BESCHREIBUNG

Transformiert eine Koordinate vom Gerätebereich in den Benutzerbereich, indem Sie den angegebenen Punkt mit der Umkehrung der aktuellen Transformationsmatrix (CTM) multiplizieren.

EINGABEN

`dx`	X-Wert der Koordinate (Gerätebereich)
`dy`	Y-Wert der Koordinate (Gerätebereich)

RÜCKGABEWERTE

`ux`	X-Wert der Koordinate (Benutzerbereich)
`uy`	Y-Wert der Koordinate (Benutzerbereich)

5.13 ccontext:DeviceToUserDistance

BEZEICHNUNG

ccontext:DeviceToUserDistance – Transformiert den Abstand zwischen Gerät und Benutzer

ÜBERSICHT

```
ux, uy = ccontext:DeviceToUserDistance(dx, dy)
```

BESCHREIBUNG

Transformiert einen Distanzvektor vom Gerätebereich in den Benutzerbereich. Diese Funktion ähnelt ccontext:DeviceToUser(), außer dass die Verschiebungskomponenten des inversen CTM bei der Transformation ignoriert werden (`dx`, `dy`).

EINGABEN

`dx`	X-Komponente eines Distanzvektors (Gerätebereich)
`dy`	Y-Komponente eines Distanzvektors (Gerätebereich)

EINGABEN

`ux`	X-Komponente eines Distanzvektors (Benutzerbereich)
`uy`	Y-Komponente eines Distanzvektors (Benutzerbereich)

5.14 ccontext:ErrorUnderlinePath

BEZEICHNUNG

ccontext:ErrorUnderlinePath – Fügt dem Pfad eine verschnörkelte Linie hinzu

ÜBERSICHT

```
ccontext:ErrorUnderlinePath(x, y, width, height)
```

BESCHREIBUNG

Fügt dem aktuellen Pfad im angegebenen Cairo-Kontext eine verschnörkelte Linie hinzu, die ungefähr das angegebene Rechteck im Stil einer Unterstreichung abdeckt, die zur Anzeige eines Rechtschreibfehlers verwendet wird.

Die Breite der Unterstreichung wird auf eine ganze Zahl von Aufwärts-/Abwärtssegmenten gerundet und das resultierende Rechteck wird im ursprünglichen Rechteck zentriert.

EINGABEN

`x`	die X-Koordinate einer Ecke des Rechtecks
`y`	die Y-Koordinate einer Ecke des Rechtecks
`width`	nicht negative Breite des Rechtecks
`height`	nicht negative Höhe des Rechtecks

5.15 ccontext:Fill

BEZEICHNUNG

ccontext:Fill – Füllt den aktuellen Pfad

ÜBERSICHT

```
ccontext:Fill()
```

BESCHREIBUNG

Ein Zeichenoperator, der den aktuellen Pfad gemäß der aktuellen Füllregel füllt. Jeder Unterpfad wird implizit geschlossen, bevor er gefüllt wird. Nach ccontext:Fill() wird der aktuelle Pfad aus dem Cairo-Kontext gelöscht. Siehe ccontext:SetFillRule() und ccontext:FillPreserve().

EINGABEN

Keine

5.16 ccontext:FillExtents

BEZEICHNUNG

ccontext:FillExtents – Gibt die Füllausdehnungen zurück

ÜBERSICHT

```
x1, y1, x2, y2 = ccontext:FillExtents()
```

BESCHREIBUNG

Berechnet einen Begrenzungsrahmen in Benutzerkoordinaten, der den Bereich abdeckt, der von einer ccontext:Fill()-Operation unter Berücksichtigung der aktuellen Pfad- und Füllparameter betroffen wäre (der "eingefärbte" Bereich). Wenn der aktuelle Pfad leer ist, wird ein leeres Rechteck ((0,0), (0,0)) zurückgegeben. Flächenmaße und Zuschnitt werden nicht berücksichtigt.

Im Gegensatz zu ccontext:PathExtents(), das ähnlich ist, aber für einige Pfade ohne eingefärbte Fläche, z.B. ein einfaches Liniensegment, Ausdehnungen ungleich Null zurückgibt.

Beachten Sie, dass ccontext:FillExtents() notwendigerweise mehr Arbeit leisten muss, um die genauen eingefärbten Bereiche unter Berücksichtigung der Füllregel zu berechnen. Daher kann ccontext:PathExtents() aus Leistungsgründen wünschenswerter sein, wenn die nicht eingefärbten Pfadausdehnungen erwünscht sind.

Siehe ccontext:Fill(), ccontext:SetFillRule() und ccontext:FillPreserve().

EINGABEN

Keine

RÜCKGABEWERTE

`x1`	links von den resultierenden Ausmaßen
`y1`	oberhalb der resultierenden Ausmaßen
`x2`	rechts von den resultierenden Ausmaßen
`y2`	unterhalb der resultierenden Ausmaßen

5.17 ccontext:FillPreserve

BEZEICHNUNG
ccontext:FillPreserve – Füllt und behält den Pfad

ÜBERSICHT
`ccontext:FillPreserve()`

BESCHREIBUNG
Ein Zeichenoperator, der den aktuellen Pfad gemäß der aktuellen Füllregel füllt. Jeder Unterpfad wird implizit geschlossen, bevor er gefüllt wird. Im Gegensatz zu ccontext:Fill() behält ccontext:FillPreserve() den Pfad im Cairo-Kontext bei.

Siehe ccontext:SetFillRule() und ccontext:Fill().

EINGABEN
Keine

5.18 ccontext:FontExtents

BEZEICHNUNG
ccontext:FontExtents – Gibt die Schriftausdehnungen der Schriftart zurück

ÜBERSICHT
`t = ccontext:FontExtents()`

BESCHREIBUNG
Ruft die Schriftausdehnungen für die aktuell ausgewählte Schriftart ab. Dadurch wird eine Tabelle zurückgegeben, in der die folgenden Elemente initialisiert sind:

`Aufstieg` Der Abstand, um den sich die Schriftart über der Grundlinie erstreckt. Beachten Sie, dass dies nicht immer genau dem Maximum der Ausmaße aller Glyphen in der Schriftart entspricht, sondern eher ausgewählt wird, um die Absicht des Schriftartendesigners auszudrücken, wie die Schriftart an den darüber liegenden Elementen ausgerichtet werden soll.

`Abstieg` Der Abstand, um den sich die Schriftart unterhalb der Grundlinie erstreckt. Dieser Wert ist positiv für typische Schriftarten, die Teile unterhalb der Grundlinie enthalten. Beachten Sie, dass dies nicht immer genau dem Maximum der Ausmaße aller Glyphen in der Schriftart entspricht, sondern vielmehr ausgewählt wird, um die Absicht des Schriftartendesigners auszudrücken, wie die Schriftart an den darunter liegenden Elementen ausgerichtet werden soll.

`Höhe` Der empfohlene vertikale Abstand zwischen den Grundlinien beim Festlegen aufeinanderfolgender Textzeilen mit der Schriftart. Dies ist um einen Betrag größer als `Aufstieg + Abstieg`, der als Zeilenabstand oder externer Zeilenabstand bekannt ist. Wenn der Platz knapp ist, können die meisten Schriftarten nur mit einem Zeilenabstand von `Aufstieg + Abstieg` eingestellt werden.

`MaxXAdvance`
: Der maximale Abstand in X-Richtung, um den der Ursprung für eine beliebige Glyphe in der Schriftart vorgeschoben wird.

`MaxYAdvance`
: Der maximale Abstand in Y-Richtung, um den der Ursprung für ein beliebiges Glyph in der Schriftart vorgeschoben wird. Bei normalen Schriftarten, die für horizontales Schreiben verwendet werden, beträgt dieser Wert Null (die Schriften Ostasiens werden manchmal vertikal geschrieben).

Alle Werte werden im aktuellen Benutzerbereich-Koordinatensystem angegeben.

Da Schriftmetriken in Benutzerbereichkoordinaten vorliegen, sind sie größtenteils, aber nicht vollständig, unabhängig von der aktuellen Transformationsmatrix. Wenn Sie ccontext:Scale() mit den Koeffizienten (2.0, 2.0) aufrufen, wird der Text doppelt so groß gezeichnet, aber die gemeldeten Textausdehnungen werden nicht verdoppelt. Sie ändern sich aufgrund von Hinweisen geringfügig (Sie können also nicht davon ausgehen, dass die Metriken unabhängig von der Transformationsmatrix sind), bleiben ansonsten aber unverändert.

EINGABEN

Keine

RÜCKGABEWERTE

`t` Tabelle mit den Schriftausdehnungen (siehe oben)

5.19 ccontext:Free

BEZEICHNUNG

ccontext:Free – Löscht einen Kontext

ÜBERSICHT

```
ccontext:Free()
```

BESCHREIBUNG

Verringert die Referenzanzahl im Cairo-Kontext um eins. Wenn das Ergebnis Null ist, werden der Cairo-Kontext gelöscht und alle zugehörigen Ressourcen freigegeben. Siehe ccontext:Reference().

EINGABEN

Keine

5.20 ccontext:GetAntialias

BEZEICHNUNG

ccontext:GetAntialias – Gibt den aktuellen Antialias-Modus zurück

ÜBERSICHT

```
mode = ccontext:GetAntialias()
```

BESCHREIBUNG

Ruft den aktuellen Form-Antialiasing-Modus ab, wie durch ccontext:SetAntialias() festgelegt.

Siehe ccontext:SetAntialias() für eine Liste der Antialiasing-Modi.

EINGABEN

Keine

RÜCKGABEWERTE

`mode` der aktuelle Form-Antialiasing-Modus

5.21 ccontext:GetCurrentPoint

BEZEICHNUNG

ccontext:GetCurrentPoint – Gibt den aktuellen Punkt zurück

ÜBERSICHT

```
x, y = ccontext:GetCurrentPoint()
```

BESCHREIBUNG

Ruft den aktuellen Punkt des aktuellen Pfads ab, der konzeptionell der letzte Punkt ist, den der Pfad bisher erreicht hat.

Der aktuelle Punkt wird im Benutzerbereich-Koordinatensystem zurückgegeben. Wenn kein aktueller Punkt definiert ist oder sich der Kontext in einem Fehlerstatus befindet, werden `x` und `y` beide auf 0,0 gesetzt. Dies kann vorab mit ccontext:HasCurrentPoint() überprüft werden.

Die meisten Pfadkonstruktionsfunktionen ändern den aktuellen Punkt. Im Folgenden finden Sie Einzelheiten dazu, wie sie sich auf den aktuellen Punkt auswirken:

ccontext:NewPath() ccontext:NewSubPath() ccontext:AppendPath() ccontext:ClosePath() ccontext:MoveTo() ccontext:LineTo() ccontext:CurveTo() ccontext:RelMoveTo() ccontext:RelLineTo() ccontext:RelCurveTo() ccontext:Arc() ccontext:ArcNegative() ccontext:Rectangle() ccontext:TextPath()

Einige Funktionen verwenden und ändern den aktuellen Punkt, ändern aber ansonsten nicht den aktuellen Pfad:

ccontext:ShowText().

Einige Funktionen setzen den aktuellen Pfad und damit den aktuellen Punkt zurück:

ccontext:Fill() ccontext:Stroke()

EINGABEN

Keine

RÜCKGABEWERTE

`x` Rückgabewert für die X-Koordinate des aktuellen Punktes

`y` Rückgabewert für die Y-Koordinate des aktuellen Punktes

5.22 ccontext:GetDash

BEZEICHNUNG
ccontext:GetDash – Gibt das aktuelle Dash-Array zurück

ÜBERSICHT
```
dashes, offset = ccontext:GetDash()
```

BESCHREIBUNG
Ruft das aktuelle Dash-Array ab.

EINGABEN
Keine

RÜCKGABEWERTE

`dashes`	aktuelles Dash-Array
`offset`	aktueller Strichversatz

5.23 ccontext:GetDashCount

BEZEICHNUNG
ccontext:GetDashCount – Gibt die aktuelle Strichmusterzahl zurück

ÜBERSICHT
```
count = ccontext:GetDashCount()
```

BESCHREIBUNG
Diese Funktion gibt die Länge des Strichmuster-Arrays im Cairo-Kontext oder 0 zurück, wenn Strichmuster derzeit nicht wirksam sind.

Siehe ccontext:SetDash() und ccontext:GetDash().

EINGABEN
Keine

RÜCKGABEWERTE

`count`	die Länge des Strichmuster-Arrays oder 0, wenn kein Strichmuster-Array festgelegt ist

5.24 ccontext:GetFillRule

BEZEICHNUNG
ccontext:GetFillRule – Gibt die aktuelle Füllregel zurück

ÜBERSICHT
```
fillrule = ccontext:GetFillRule()
```

BESCHREIBUNG
Ruft die aktuelle Füllregel ab, welche mit ccontext:SetFillRule() festgelegt wurde. Eine Liste der Füllregeln finden Sie unter ccontext:SetFillRule().

EINGABEN
Keine

RÜCKGABEWERTE
`fillrule` die aktuelle Füllregel

5.25 ccontext:GetFontFace

BEZEICHNUNG
ccontext:GetFontFace – Gibt die aktuelle Schriftart zurück

ÜBERSICHT
```
face = ccontext:GetFontFace()
```

BESCHREIBUNG
Ruft die aktuelle Schriftart für einen Cairo-Kontext ab.

Diese Funktion gibt die aktuelle Schriftart zurück. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie cfontface:Reference() aufrufen.

Diese Funktion gibt niemals Null zurück. Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Nil-Font-Face-Objekt zurückgegeben, bei dem cfontface:Status() `#CAIRO_STATUS_NO_MEMORY` zurückgibt. Die Verwendung dieses Nil-Objekts führt dazu, dass sein Fehlerstatus auf andere Objekte übertragen wird, an die es übergeben wird. Beispielsweise löst der Aufruf von ccontext:SetFontFace() mit einer Null-Schriftart einen Fehler aus, der das Cairo-Kontextobjekt herunterfährt.

EINGABEN
Keine

RÜCKGABEWERTE
`face` die aktuelle Schriftart

5.26 ccontext:GetFontMatrix

BEZEICHNUNG
ccontext:GetFontMatrix – Gibt die aktuelle Schriftartenmatrix zurück

ÜBERSICHT
```
m = ccontext:GetFontMatrix(matrix)
```

BESCHREIBUNG
Gibt die aktuelle Schriftartmatrix als Cairo-Matrixobjekt zurück. Siehe ccontext:SetFontMatrix().

EINGABEN
Keine

RÜCKGABEWERTE
`m` Matrixobjekt

5.27 ccontext:GetFontOptions

BEZEICHNUNG

ccontext:GetFontOptions – Gibt die Schriftartoptionen zurück

ÜBERSICHT

```
ccontext:GetFontOptions(options)
```

BESCHREIBUNG

Ruft über ccontext:SetFontOptions() festgelegte Schriftart-Rendering-Optionen ab. Beachten Sie, dass die zurückgegebenen Optionen keine von der zugrunde liegenden Oberfläche abgeleiteten Optionen enthalten. Sie sind im wahrsten Sinne des Wortes die Optionen, die an ccontext:SetFontOptions() übergeben werden.

EINGABEN

`options` ein Cairo-Schriftartoptionsobjekt, in dem die abgerufenen Optionen gespeichert werden; alle vorhandenen Werte werden überschrieben

5.28 ccontext:GetGroupTarget

BEZEICHNUNG

ccontext:GetGroupTarget – Gibt das Gruppenziel zurück

ÜBERSICHT

```
handle = ccontext:GetGroupTarget()
```

BESCHREIBUNG

Ruft die aktuelle Zieloberfläche für den Kontext ab. Dies ist entweder die ursprüngliche Zieloberfläche, wie sie an cairo.Context() übergeben wurde, oder die Zieloberfläche für die aktuelle Gruppe, wie sie durch den letzten Aufruf von ccontext:PushGroup() oder ccontext:PushGroupWithContent() gestartet wurde.

Diese Funktion gibt immer ein gültiges Handle zurück, das Ergebnis kann jedoch eine "Null"-Oberfläche sein, wenn sich der Cairo-Kontext bereits in einem Fehlerzustand befindet. Eine Nulloberfläche wird dadurch angezeigt, dass csurface:Status() sich von `#CAIRO_STATUS_SUCCESS` unterscheidet.

Diese Funktion gibt die Zieloberfläche zurück. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie csurface:Reference() aufrufen.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die Zieloberfläche (im Besitz von Cairo)

5.29 ccontext:GetLineCap

BEZEICHNUNG

ccontext:GetLineCap – Gibt den aktuellen Zeilenumbruchstil zurück

ÜBERSICHT

```
cap = ccontext:GetLineCap()
```

BESCHREIBUNG

Ruft den aktuellen Zeilenkopfstil ab, wie durch ccontext:SetLineCap() festgelegt. Eine Liste der Zeilenumbruchstile finden Sie unter ccontext:SetLineCap().

EINGABEN

Keine

RÜCKGABEWERTE

`cap` der aktuelle Zeilenumbruchstil

5.30 ccontext:GetLineJoin

BEZEICHNUNG

ccontext:GetLineJoin – Gibt den aktuellen Linienverbindungsstil zurück

ÜBERSICHT

```
join = ccontext:GetLineJoin()
```

BESCHREIBUNG

Ruft den aktuellen Linienverbindungsstil ab, wie durch ccontext:SetLineJoin() festgelegt. Eine Liste der Linienverbindungsstile finden Sie unter ccontext:SetLineJoin().

EINGABEN

Keine

RÜCKGABEWERTE

`join` der aktuelle Linienverbindungsstil

5.31 ccontext:GetLineWidth

BEZEICHNUNG

ccontext:GetLineWidth – Gibt die aktuelle Linienbreite zurück

ÜBERSICHT

```
width = ccontext:GetLineWidth()
```

BESCHREIBUNG

Diese Funktion gibt den aktuellen Wert der Linienbreite genau so zurück, wie er durch ccontext:SetLineWidth() festgelegt wurde. Beachten Sie, dass der Wert auch dann unverändert bleibt, wenn sich die CTM zwischen den Aufrufen von ccontext:SetLineWidth() und ccontext:GetLineWidth() geändert hat.

EINGABEN

Keine

RÜCKGABEWERTE

`width` die aktuelle Linienbreite

5.32 ccontext:GetMatrix

BEZEICHNUNG
ccontext:GetMatrix – Gibt die aktuelle Transformationsmatrix zurück

ÜBERSICHT
```
m = ccontext:GetMatrix(matrix)
```

BESCHREIBUNG
Gibt die aktuelle Transformationsmatrix (CTM) als Cairo-Matrixobjekt zurück. Verwenden Sie cmatrix:Get(), um die einzelnen Matrixkoeffizienten abzufragen.

EINGABEN
Keine

RÜCKGABEWERTE
`m` aktuelle Transformationsmatrix

5.33 ccontext:GetMiterLimit

BEZEICHNUNG
ccontext:GetMiterLimit – Gibt die aktuelle Gehrungsgrenze zurück

ÜBERSICHT
```
limit = ccontext:GetMiterLimit()
```

BESCHREIBUNG
Ruft das aktuelle Gehrungslimit ab, wie durch ccontext:SetMiterLimit() festgelegt.

EINGABEN
Keine

RÜCKGABEWERTE
`limit` die aktuelle Gehrungsgrenze

5.34 ccontext:GetOperator

BEZEICHNUNG
ccontext:GetOperator – Gibt den aktuellen Compositing-Operator zurück

ÜBERSICHT
```
op = ccontext:GetOperator()
```

BESCHREIBUNG
Ruft den aktuellen Compositing-Operator für einen Cairo-Kontext ab. Eine Liste der Compositing-Operatoren finden Sie unter ccontext:SetOperator().

EINGABEN
Keine

RÜCKGABEWERTE
`op` der aktuelle Compositing-Operator

5.35 ccontext:GetReferenceCount

BEZEICHNUNG
ccontext:GetReferenceCount – Gibt die Referenzanzahl zurück

ÜBERSICHT
```
count = ccontext:GetReferenceCount()
```

BESCHREIBUNG
Gibt den aktuellen Referenzzähler des Kontexts zurück.

EINGABEN
Keine

RÜCKGABEWERTE
`count` der aktuelle Referenzzähler

5.36 ccontext:GetScaledFont

BEZEICHNUNG
ccontext:GetScaledFont – Gibt die aktuelle skalierte Schriftart zurück

ÜBERSICHT
```
font = ccontext:GetScaledFont()
```

BESCHREIBUNG
Ruft die aktuelle skalierte Schriftart für einen Ciro-Kontext ab.

Diese Funktion gibt die aktuell skalierte Schriftart zurück. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie cscaledfont:Reference() aufrufen.

Diese Funktion gibt niemals Null zurück. Wenn kein Speicher zugewiesen werden kann, wird ein spezielles "Null"-skaliertes Schriftartobjekt zurückgegeben, bei dem cscaledfont:Status() `#CAIRO_STATUS_NO_MEMORY` zurückgibt. Die Verwendung dieses Null-Objekts führt dazu, dass sein Fehlerstatus auf andere Objekte übertragen wird, an die es übergeben wird. Beispielsweise löst der Aufruf von ccontext:SetScaledFont() mit einer Null-Schriftart einen Fehler aus, der das Cairo-Kontextobjekt herunterfährt.

EINGABEN
Keine

RÜCKGABEWERTE
`font` die aktuell skalierte Schriftart (im Besitz von Cairo)

5.37 ccontext:GetSource

BEZEICHNUNG
ccontext:GetSource – Gibt das aktuelle Quellmuster zurück

ÜBERSICHT
```
pat = ccontext:GetSource()
```

BESCHREIBUNG

Ruft das aktuelle Quellmuster für den Cairo-Kontext ab.

Diese Funktion gibt das aktuelle Quellmuster zurück. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie cpattern:Reference() aufrufen.

EINGABEN

Keine

RÜCKGABEWERTE

`pat` das aktuelle Quellmuster (im Besitz von Cairo)

5.38 ccontext:GetTarget

BEZEICHNUNG

ccontext:GetTarget – Gibt die Zieloberfläche zurück

ÜBERSICHT

```
surface = ccontext:GetTarget()
```

BESCHREIBUNG

Ruft die Zieloberfläche für den Cairo-Kontext ab, wie sie an cairo.Context() übergeben wird.

Diese Funktion gibt immer ein gültiges Handle zurück, das Ergebnis kann jedoch eine "Null"-Oberfläche sein, wenn sich der Kontext bereits in einem Fehlerzustand befindet. Eine Nulloberfläche wird dadurch angezeigt, dass csurface:Status() sich von `#CAIRO_STATUS_SUCCESS` unterscheidet.

Diese Funktion gibt die Zieloberfläche zurück. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie csurface:Reference() aufrufen.

EINGABEN

Keine

RÜCKGABEWERTE

`surface` die Zieloberfläche (im Besitz von Cairo)

5.39 ccontext:GetTolerance

BEZEICHNUNG

ccontext:GetTolerance – Gibt den aktuellen Toleranzwert zurück

ÜBERSICHT

```
t = ccontext:GetTolerance()
```

BESCHREIBUNG

Ruft den aktuellen Toleranzwert ab, wie durch ccontext:SetTolerance() festgelegt.

EINGABEN

Keine

RÜCKGABEWERTE

`t` der aktuelle Toleranzwert

5.40 ccontext:GlyphExtents

BEZEICHNUNG

ccontext:GlyphExtents – Gibt die Glyphen-Ausdehnungen zurück

ÜBERSICHT

```
t = ccontext:GlyphExtents(glyphs[, offset, num_glyphs])
```

BESCHREIBUNG

Ruft die Grenzen für ein Glyphen-Array ab. Die Grenzen beschreiben ein Benutzerbereichrechteck, das den "eingefärbten" Teil der Glyphen umschließt, wie sie von ccontext:ShowGlyphs() gezeichnet würden. Darüber hinaus geben die Werte `XAdvance` und `YAdvance` den Betrag an, um den der aktuelle Punkt durch ccontext:ShowGlyphs() vorgezogen würde.

Beachten Sie, dass Leerzeichen keinen Einfluss auf die Größe des Rechtecks haben (Felder `Width` und `Height`).

Diese Funktion gibt eine Tabelle zurück, die die Glyphenausdehnungen beschreibt. Eine Beschreibung aller Tabellenfelder finden Sie unter ccontext:TextExtents().

EINGABEN

`glyphs` ein Glyphen-Array

`offset` Optional: Offset in das Array, das das Startzeichen angibt (standardmäßig 0, was das erste Zeichen bedeutet).

`num_glyphs` Optional: Anzahl der anzuzeigenden Glyphen (Standard ist -1, was alle Glyphen bedeutet).

RÜCKGABEWERTE

`t` Tabelle mit den Glyphenausdehnungen

5.41 ccontext:GlyphPath

BEZEICHNUNG

ccontext:GlyphPath – Fügt die Glyphen zum Pfad hinzu

ÜBERSICHT

```
ccontext:GlyphPath(glyphs[, offset, num_glyphs])
```

BESCHREIBUNG

Fügt geschlossene Pfade für die Glyphen zum aktuellen Pfad hinzu. Wenn der generierte Pfad gefüllt ist, erzielt er einen ähnlichen Effekt wie ccontext:ShowGlyphs().

EINGABEN

`glyphs` ein Glyphen-Array

`offset` Optional: Offset in das Array, das das Startzeichen angibt (standardmäßig 0, was das erste Zeichen bedeutet).

`num_glyphs`
Optional: Anzahl der anzuzeigenden Glyphen (Standard ist -1, was alle Glyphen bedeutet).

5.42 ccontext:GlyphStringPath

BEZEICHNUNG
ccontext:GlyphStringPath – Fügt die Glyphen zum Zeichenketten-Pfad hinzu

ÜBERSICHT
```
ccontext:GlyphStringPath(font, glyphs)
```

BESCHREIBUNG
Fügt die Glyphen in `glyphs` zum aktuellen Pfad im angegebenen Cairo-Kontext hinzu.

Der Ursprung der Glyphen (der linke Rand der Grundlinie) liegt am aktuellen Punkt des Cairo-Kontexts.

EINGABEN

`font` ein Pango-Schriftartobjekt

`glyphs` ein Pango-Glyphen-Zeichenketten-Objekt

5.43 ccontext:HasCurrentPoint

BEZEICHNUNG
ccontext:HasCurrentPoint – Gibt den aktuellen Punkt zurück

ÜBERSICHT
```
ok = ccontext:HasCurrentPoint()
```

BESCHREIBUNG
Gibt zurück, ob ein aktueller Punkt auf dem aktuellen Pfad definiert ist. Einzelheiten zum aktuellen Punkt finden Sie unter ccontext:GetCurrentPoint().

EINGABEN
Keine

RÜCKGABEWERTE

`ok` ob ein aktueller Punkt definiert ist

5.44 ccontext:IdentityMatrix

BEZEICHNUNG
ccontext:IdentityMatrix – Stellt die aktuelle Transformationsmatrix zurück

ÜBERSICHT

```
ccontext:IdentityMatrix()
```

BESCHREIBUNG

Setzt die aktuelle Transformationsmatrix (CTM) zurück, indem sie auf die Identitätsmatrix gesetzt wird. Das heißt, die Benutzerbereich- und Gerätebereich-Achsen werden ausgerichtet und eine Benutzerbereich-Einheit wird in eine Gerätebereich-Einheit umgewandelt.

EINGABEN

Keine

5.45 ccontext:InClip

BEZEICHNUNG

ccontext:InClip – Überprüft, ob der Punkt innerhalb des Clipbereichs liegt

ÜBERSICHT

```
ok = ccontext:InClip(x, y)
```

BESCHREIBUNG

Testet, ob der angegebene Punkt innerhalb des Bereichs liegt, der durch den aktuellen Clip sichtbar wäre, d.h. dem Bereich, der durch eine ccontext:Paint()-Operation gefüllt würde.

Siehe ccontext:Clip() und ccontext:ClipPreserve().

Diese Funktion gibt einen Wert ungleich Null zurück, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt.

EINGABEN

`x`	X-Koordinate des zu testenden Punkts
`y`	Y-Koordinate des zu testenden Punkts

RÜCKGABEWERTE

`ok`	ein Wert ungleich Null, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt

5.46 ccontext:InFill

BEZEICHNUNG

ccontext:InFill – Überprüft, ob der Punkt innerhalb des Füllbereichs liegt

ÜBERSICHT

```
ok = ccontext:InFill(x, y)
```

BESCHREIBUNG

Testet, ob der angegebene Punkt innerhalb des Bereichs liegt, der von einer ccontext:Fill()-Operation unter Berücksichtigung des aktuellen Pfads und der Füllparameter betroffen wäre. Flächenmaße und Zuschnitt werden nicht berücksichtigt.

Siehe ccontext:Fill(), ccontext:SetFillRule() und ccontext:FillPreserve().

Diese Funktion gibt einen Wert ungleich Null zurück, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt.

EINGABEN

`x` X-Koordinate des zu testenden Punkts

`y` Y-Koordinate des zu testenden Punkts

RÜCKGABEWERTE

`ok` ein Wert ungleich Null, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt

5.47 ccontext:InStroke

BEZEICHNUNG

ccontext:InStroke – Überprüft, ob der Punkt innerhalb des Strichbereichs liegt

ÜBERSICHT

```
ok = ccontext:InStroke(x, y)
```

BESCHREIBUNG

Testet, ob der angegebene Punkt innerhalb des Bereichs liegt, der von einer ccontext:Stroke()-Operation unter Berücksichtigung der aktuellen Pfad- und Strichparameter betroffen wäre. Flächenmaße und Zuschnitt werden nicht berücksichtigt.

Siehe ccontext:Stroke(), ccontext:SetLineWidth(), ccontext:SetLineJoin(), ccontext:SetLineCap(), ccontext:SetDash() und ccontext:StrokePreserve().

Diese Funktion gibt einen Wert ungleich Null zurück, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt.

EINGABEN

`x` X-Koordinate des zu testenden Punkts

`y` Y-Koordinate des zu testenden Punkts

RÜCKGABEWERTE

`ok` ein Wert ungleich Null, wenn der Punkt innerhalb liegt, oder Null, wenn er außerhalb liegt

5.48 ccontext:IsNull

BEZEICHNUNG

ccontext:IsNull – Überprüft, ob der Kontext ungültig ist

ÜBERSICHT

```
bool = ccontext:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn der Kontext `NULL` ist, also ungültig. Wenn Funktionen, die Kontexte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen den Kontext einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Kontextzuordnung fehlgeschlagen ist. In diesem Fall ist der Kontext `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn der Kontext `NULL` ist, andernfalls `False`

5.49 ccontext:LayoutLinePath

BEZEICHNUNG

ccontext:LayoutLinePath – Fügt Text aus der Layoutzeile hinzu

ÜBERSICHT

```
ccontext:LayoutLinePath(line)
```

BESCHREIBUNG

Fügt den Text aus der durch `line` angegebenen Pango-Layoutzeile zum aktuellen Pfad im angegebenen Cairo-Kontext hinzu.

Der Ursprung der Glyphen (der linke Rand der Linie) liegt am aktuellen Punkt des Cairo-Kontexts.

EINGABEN

`line` ein Pango-Layout-Linienobjekt

5.50 ccontext:LayoutPath

BEZEICHNUNG

ccontext:LayoutPath – Fügt Text aus dem Layout hinzu

ÜBERSICHT

```
ccontext:LayoutPath(layout)
```

BESCHREIBUNG

Fügt den Text aus dem durch `layout` angegebenen Pango-Layout zum aktuellen Pfad im angegebenen Cairo-Kontext hinzu.

Die obere linke Ecke des Pango-Layouts befindet sich am aktuellen Punkt des Cairo-Kontexts.

EINGABEN

`layout` ein Pango-Layoutobjekt

5.51 ccontext:LineTo

BEZEICHNUNG

ccontext:LineTo – Fügt eine Linie zum Pfad hinzu

ÜBERSICHT

```
ccontext:LineTo(x, y)
```

BESCHREIBUNG

Fügt eine Linie zum Pfad vom aktuellen Punkt zur Position (`x`, `y`) in Benutzerbereichkoordinaten hinzu. Nach diesem Aufruf ist der aktuelle Punkt (`x`, `y`).

Wenn es vor dem Aufruf von ccontext:LineTo() keinen aktuellen Punkt gibt, verhält sich diese Funktion wie ccontext:MoveTo() mit `x` und `y` als Parametern.

EINGABEN

`x`	die X-Koordinate des Endes der neuen Linie
`y`	die Y-Koordinate des Endes der neuen Linie

5.52 ccontext:Mask

BEZEICHNUNG

ccontext:Mask – Zeichnet die Farbquelle als Maske

ÜBERSICHT

```
ccontext:Mask(pattern)
```

BESCHREIBUNG

Ein Zeichenoperator, der die aktuelle Quelle unter Verwendung des Alphakanals von `pattern` als Maske zeichnet. Undurchsichtige Bereiche von `pattern` werden mit der Quelle bemalt, transparente Bereiche werden nicht bemalt.

EINGABEN

`pattern`	ein Cairo-Musterobjekt

5.53 ccontext:MaskSurface

BEZEICHNUNG

ccontext:MaskSurface – Zeichnet die Farbquelle als Oberflächenmaske

ÜBERSICHT

```
ccontext:MaskSurface(surface, surface_x, surface_y)
```

BESCHREIBUNG

Ein Zeichenoperator, der die aktuelle Quelle unter Verwendung des Alphakanals von `surface` als Maske zeichnet. Undurchsichtige Bereiche der `Oberfläche` werden mit der Quelle bemalt, transparente Bereiche werden nicht bemalt.

EINGABEN

`surface`	ein Cairo Oberflächenobjekt

`surface_x`
: X-Koordinate, an der der Ursprung der `Oberfläche` platziert werden soll

`surface_y`
: Y-Koordinate, an der der Ursprung der `Oberfläche` platziert werden soll

5.54 ccontext:MoveTo

BEZEICHNUNG
ccontext:MoveTo – Beginnt einen neuen Unterpfad

ÜBERSICHT
`ccontext:MoveTo(x, y)`

BESCHREIBUNG
Beginnt einen neuen Unterpfad. Nach diesem Aufruf ist der aktuelle Punkt (`x`, `y`).

EINGABEN

`x`	die X-Koordinate der neuen Position
`y`	die Y-Koordinate der neuen Position

5.55 ccontext:NewPath

BEZEICHNUNG
ccontext:NewPath – Löscht den aktuellen Pfad

ÜBERSICHT
`ccontext:NewPath()`

BESCHREIBUNG
Löscht den aktuellen Pfad. Nach diesem Aufruf gibt es keinen Pfad und keinen aktuellen Punkt.

EINGABEN
Keine

5.56 ccontext:NewSubPath

BEZEICHNUNG
ccontext:NewSubPath – Beginnt einen neuen Unterpfad

ÜBERSICHT
`ccontext:NewSubPath()`

BESCHREIBUNG
Beginnt einen neuen Unterpfad. Beachten Sie, dass der vorhandene Pfad davon nicht betroffen ist. Nach diesem Aufruf wird es keinen aktuellen Punkt mehr geben.

In vielen Fällen ist dieser Aufruf nicht erforderlich, da neue Unterpfade häufig mit ccontext:MoveTo() gestartet werden.

Ein Aufruf von ccontext:NewSubPath() ist besonders nützlich, wenn ein neuer Unterpfad mit einem der ccontext:Arc()-Aufrufe beginnt. Dies erleichtert die Arbeit, da es nicht mehr notwendig ist, die Anfangskoordinaten des Bogens für einen Aufruf von ccontext:MoveTo() manuell zu berechnen.

EINGABEN

Keine

5.57 ccontext:Paint

BEZEICHNUNG

ccontext:Paint – Zeichnet die aktuelle Quelle

ÜBERSICHT

```
ccontext:Paint()
```

BESCHREIBUNG

Ein Zeichenoperator, der die aktuelle Quelle überall im aktuellen Clipbereich zeichnet.

EINGABEN

Keine

5.58 ccontext:PaintWithAlpha

BEZEICHNUNG

ccontext:PaintWithAlpha – Zeichnet die aktuelle Quelle mit Alpha-Werten

ÜBERSICHT

```
ccontext:PaintWithAlpha(alpha)
```

BESCHREIBUNG

Ein Zeichenoperator, der die aktuelle Quelle überall innerhalb der aktuellen Clip-Region unter Verwendung einer Maske mit konstantem Alpha-Wert `alpha` zeichnet. Der Effekt ist ähnlich wie bei ccontext:Paint(), aber die Zeichnung wird jedoch mit dem Alphawert ausgeblendet.

EINGABEN

`alpha` Alphawert, zwischen 0 (transparent) und 1 (undurchsichtig)

5.59 ccontext:PangoContext

BEZEICHNUNG

ccontext:PangoContext – Erstellt einen Pango-Kontext

ÜBERSICHT

```
handle = ccontext:PangoContext()
```

BESCHREIBUNG

Erzeugt einen Pango-Kontext, der so eingerichtet ist, dass er mit der aktuellen Transformation und der Zielfläche des Cairo-Kontextes entspricht.

Dieser Kontext kann dann verwendet werden, um ein Layout mit pango.Layout() zu erstellen.

Diese Funktion ist eine Komfortfunktion, die einen Pango-Kontext mit der Standard-Schriftartenzuordnung erstellt und dann auf den Cairo-Kontext aktualisiert. Wenn Sie nur ein Layout zur Verwendung mit dem Cairo-Kontext erstellen und nicht direkt auf den Pango-Kontext zugreifen müssen, können Sie stattdessen ccontext:PangoLayout() verwenden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` den neu erstellten Pango-Kontext

5.60 ccontext:PangoLayout

BEZEICHNUNG

ccontext:PangoLayout – Erstellt ein Pango Layout

ÜBERSICHT

```
handle = ccontext:PangoLayout()
```

BESCHREIBUNG

Erzeugt ein Pango-Layout, das auf die aktuelle Transformation die der Zielfläche des Cairo-Kontextes entspricht.

Dieses Layout kann dann zur Textmessung mit Funktionen wie playout:GetSize() oder zum Zeichnen mit Funktionen wie ccontext:ShowLayout() benutzt werden. Wenn Sie die Transformation oder die Zieloberfläche für den Cairo-Kontext ändern, müssen Sie ccontext:UpdateLayout() aufrufen.

Diese Funktion ist der bequemste Weg, um Cairo mit Pango zu verwenden, allerdings ist sie etwas ineffizient, da sie für jedes Layout ein eigenes Pango-Kontextobjekt für jedes Layout erstellt. Dies könnte in einer Anwendung von Bedeutung sein, in der große Mengen an Text layoutet werden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` das neu erstellte Pango-Layout

5.61 ccontext:PathExtents

BEZEICHNUNG

ccontext:PathExtents – Gibt die Pfadausdehnungen zurück

ÜBERSICHT

```
x1, y1, x2, y2 = ccontext:PathExtents()
```

BESCHREIBUNG

Berechnet einen Begrenzungsrahmen in Benutzerbereichkoordinaten, der die Punkte auf dem aktuellen Pfad abdeckt. Wenn der aktuelle Pfad leer ist, wird ein leeres Rechteck ((0,0), (0,0)) zurück gegeben. Strichparameter, Füllregel, Oberflächenabmessungen und Clipping werden nicht berücksichtigt.

Im Gegensatz dazu geben ccontext:FillExtents() und ccontext:StrokeExtents() nur die Ausmaße des Bereichs zurück, der durch die entsprechenden Zeichenvorgänge "eingefärbt" würde.

Das Ergebnis von ccontext:PathExtents() wird als äquivalent zum Grenzwert von ccontext:StrokeExtents() mit `#CAIRO_LINE_CAP_ROUND` definiert, wenn sich die Linienbreite 0,0 annähert (jedoch niemals den von ccontext:StrokeExtents() für eine Linienbreite von 0,0 zurückgegebenen leeren Rechteckwert erreicht).

Konkret bedeutet dies, dass Teilpfade mit Nullfläche, wie z.B. ccontext:MoveTo();ccontext:LineTo()-Segmente (selbst in degenerierten Fällen, in denen die Koordinaten beider Aufrufe identisch sind), als Beitrag zu den Ausdehnungen berücksichtigt werden. Ein einzelnes ccontext:MoveTo() trägt jedoch nicht zu den Ergebnissen von ccontext:PathExtents() bei.

EINGABEN

Keine

RÜCKGABEWERTE

`x1`	links von den resultierenden Ausmaße
`y1`	Oberseite der resultierenden Ausmaße
`x2`	rechts von den resultierenden Ausmaße
`y2`	Unterseite der resultierenden Ausmaße

5.62 ccontext:PopGroup

BEZEICHNUNG

ccontext:PopGroup – Beendet die Umleitung auf eine Gruppe

ÜBERSICHT

```
pat = ccontext:PopGroup()
```

BESCHREIBUNG

Beendet die Umleitung, die durch einen Aufruf von ccontext:PushGroup() oder ccontext:PushGroupWithContent() begonnen wurde, und gibt ein neues Muster zurück, das die Ergebnisse aller für die Gruppe durchgeführten Zeichenvorgänge enthält.

Die Funktion ccontext:PopGroup() ruft ccontext:Restore() auf (und gleicht damit einen Aufruf von ccontext:Save() durch die Push-Gruppenfunktion aus), sodass Änderungen am Grafikstatus außerhalb der Gruppe nicht sichtbar sind.

Diese Funktion gibt ein neu erstelltes (Oberflächen-)Muster zurück, das die Ergebnisse aller für die Gruppe durchgeführten Zeichenvorgänge enthält. Der Aufrufer hat das

zurückgegebenen Objekt zugewiesen und sollte cpattern:Free() aufrufen, wenn er damit fertig ist.

EINGABEN
Keine

RÜCKGABEWERTE
`pat` ein neu erstelltes Muster

5.63 ccontext:PopGroupToSource

BEZEICHNUNG
ccontext:PopGroupToSource – Installiert die Gruppe als Quellmuster

ÜBERSICHT
`ccontext:PopGroupToSource()`

BESCHREIBUNG
Beendet die Umleitung, die durch einen Aufruf von ccontext:PushGroup() oder ccontext:PushGroupWithContent() begonnen wurde, und installiert das resultierende Muster als Quellmuster im angegebenen Cairo-Kontext.

Das Verhalten dieser Funktion entspricht der Abfolge von folgenden Operationen:

```
group = cr:PopGroup()
cr:SetSource(group)
group:Free()
```

Dies ist jedoch praktischer, da keine Variable zum Speichern des kurzlebigen Handles für das Muster erforderlich ist.

Die Funktion ccontext:PopGroup() ruft ccontext:Restore() auf (und gleicht damit einen Aufruf von ccontext:Save() durch die Push-Gruppenfunktion aus), sodass Änderungen am Grafikstatus außerhalb der Gruppe nicht sichtbar sind.

EINGABEN
Keine

5.64 ccontext:PushGroup

BEZEICHNUNG
ccontext:PushGroup – Leitet auf eine Gruppe um

ÜBERSICHT
`ccontext:PushGroup()`

BESCHREIBUNG
Leitet die Zeichnung vorübergehend auf eine Zwischenoberfläche um, die als Gruppe bezeichnet wird. Die Umleitung dauert so lange, bis die Gruppe durch einen Aufruf von ccontext:PopGroup() oder ccontext:PopGroupToSource() abgeschlossen ist. Diese Aufrufe stellen der Gruppe das Ergebnis jeder Zeichnung als Muster bereit (entweder als explizites Objekt oder als Quellmuster festgelegt).

Diese Gruppenfunktionalität kann für die Durchführung von Zwischenkompositionen praktisch sein. Eine häufige Verwendung einer Gruppe besteht darin, Objekte innerhalb der Gruppe undurchsichtig darzustellen (so dass sie sich gegenseitig verdecken) und das Ergebnis dann durchscheinend auf das Ziel zu übertragen.

Gruppen können beliebig tief verschachtelt werden, indem ausgewogene Aufrufe an ccontext:PushGroup() / ccontext:PopGroup() durchgeführt werden. Bei jedem Aufruf wird die neue Zielgruppe auf bzw. von einem Stapel verschoben bzw. entfernt.

Die Funktion ccontext:PushGroup() ruft ccontext:Save() auf, sodass Änderungen am Grafikstatus außerhalb der Gruppe nicht sichtbar sind. Die pop_group-Funktionen rufen ccontext:Restore() auf.

Standardmäßig hat die Zwischengruppe den Inhaltstyp `#CAIRO_CONTENT_COLOR_ALPHA`. Andere Inhaltstypen können für die Gruppe ausgewählt werden, indem stattdessen ccontext:PushGroupWithContent() verwendet wird.

Als Beispiel sehen Sie hier, wie man einen Pfad mit Transluzenz füllen und streichen könnte, ohne dass jedoch ein Teil der Füllung unter dem Strich sichtbar wäre:

```
cr:PushGroup()
cr:SetSource(fill_pattern)
cr:FillPreserve()
cr:SetSource(stroke_pattern)
cr:Stroke()
cr:PopGroupToSource()
cr:PaintWithAlpha(alpha)
```

EINGABEN

Keine

5.65 ccontext:PushGroupWithContent

BEZEICHNUNG

ccontext:PushGroupWithContent – Leitet auf eine Gruppe mit Inhalt um

ÜBERSICHT

`ccontext:PushGroupWithContent(content)`

BESCHREIBUNG

Leitet die Zeichnung vorübergehend auf eine Zwischenoberfläche um, die als Gruppe bezeichnet wird. Die Umleitung dauert so lange, bis die Gruppe durch einen Aufruf von ccontext:PopGroup() oder ccontext:PopGroupToSource() abgeschlossen ist. Diese Aufrufe stellen der Gruppe das Ergebnis jeder Zeichnung als Muster bereit (entweder als explizites Objekt oder als Quellmuster festgelegt).

Die Gruppe hat den Inhaltstyp `content`. Die Möglichkeit, diesen Inhaltstyp zu steuern, ist der einzige Unterschied zwischen dieser Funktion und ccontext:PushGroup(), den Sie für eine detailliertere Beschreibung des Gruppenrenderings beachten sollten.

Siehe csurface:GetContent() für eine Liste der unterstützten Inhaltstypen.

EINGABEN

`content` Konstante, die den Typ der Gruppe angibt, die erstellt wird (siehe oben).

5.66 ccontext:Rectangle

BEZEICHNUNG
ccontext:Rectangle – Fügt dem Pfad ein Rechteck hinzu

ÜBERSICHT
`ccontext:Rectangle(x, y, width, height)`

BESCHREIBUNG
Fügt dem aktuellen Pfad an der Position (`x`, `y`) in Benutzerbereichkoordinaten ein geschlossenes Unterpfadrechteck der angegebenen Größe hinzu.

Diese Funktion ist logisch äquivalent zu:

```
cr:MoveTo(x, y)
cr:RelLineTo(width)
cr:RelLineTo(0, height)
cr:RelLineTo(-width, 0)
cr:ClosePath()
```

EINGABEN

`x`	die X-Koordinate der oberen linken Ecke des Rechtecks
`y`	die Y-Koordinate der oberen linken Ecke des Rechtecks
`width`	die Breite des Rechtecks
`height`	die Höhe des Rechtecks

5.67 ccontext:Reference

BEZEICHNUNG
ccontext:Reference – Erhöht die Referenzanzahl für den Kontext

ÜBERSICHT
`ccontext:Reference()`

BESCHREIBUNG
Erhöht die Referenzanzahl für den Kontext um eins. Dadurch wird verhindert, dass der Kontext gelöscht wird, bis ein passender Aufruf von ccontext:Free() erfolgt.

Verwenden Sie ccontext:GetReferenceCount(), um die Anzahl der Verweise auf einen Cairo-Kontext abzurufen.

EINGABEN
Keine

5.68 ccontext:RelCurveTo

BEZEICHNUNG

ccontext:RelCurveTo – Fügt einen relativen kubischen Bézier-Spline hinzu

ÜBERSICHT

```
ccontext:RelCurveTo(dx1, dy1, dx2, dy2, dx3, dy3)
```

BESCHREIBUNG

Relativkoordinatenversion von ccontext:CurveTo(). Alle Versätze beziehen sich auf den aktuellen Punkt. Fügt einen kubischen Bézier-Spline zum Pfad vom aktuellen Punkt zu einem um (`dx3`, `dy3`) vom aktuellen Punkt versetzten Punkt hinzu und verwendet dabei um (`dx1`, `dy1`) und (`dx2`, `dy2`) versetzte Punkte als Kontrollpunkte. Nach diesem Aufruf wird der aktuelle Punkt um (`dx3`, `dy3`) verschoben.

Bei einem aktuellen Punkt von (x, y) ist der Aufruf dieser Funktion mit `dx1`, `dy1`, `dx2`, `dy2`, `dx3`, `dy3` logisch äquivalent zum Aufruf von ccontext:CurveTo() mit `x+dx1`, `y+dy1`, `x+dx2`, `y+dy2`, `x+dx3`, `y+dy3`.

Es ist ein Fehler, diese Funktion ohne aktuellen Punkt aufzurufen. Dies führt dazu, dass der Kontext mit dem Status `#CAIRO_STATUS_NO_CURRENT_POINT` heruntergefahren wird.

EINGABEN

`dx1`	der X-Versatz zum ersten Kontrollpunkt
`dy1`	der Y-Versatz zum ersten Kontrollpunkt
`dx2`	der X-Versatz zum zweiten Kontrollpunkt
`dy2`	der Y-Versatz zum zweiten Kontrollpunkt
`dx3`	der X-Versatz zum Ende der Kurve
`dy3`	der Y-Versatz zum Ende der Kurve

5.69 ccontext:RelLineTo

BEZEICHNUNG

ccontext:RelLineTo – Fügt eine relative Linie hinzu

ÜBERSICHT

```
ccontext:RelLineTo(dx, dy)
```

BESCHREIBUNG

Relativkoordinatenversion von ccontext:LineTo(). Fügt eine Linie zum Pfad vom aktuellen Punkt zu einem Punkt hinzu, der im Benutzerbereich um (`dx`, `dy`) vom aktuellen Punkt versetzt ist. Nach diesem Aufruf wird der aktuelle Punkt um (`dx`, `dy`) verschoben.

Bei einem aktuellen Punkt von (x, y) ist der Aufruf dieser Funktion mit `dx`, `dy` logisch äquivalent zum Aufruf von ccontext:LineTo() mit `x+dx` und `y+dy` .

Es ist ein Fehler, diese Funktion ohne aktuellen Punkt aufzurufen. Dies führt dazu, dass der Kontext mit dem Status `#CAIRO_STATUS_NO_CURRENT_POINT` heruntergefahren wird.

EINGABEN

`dx`	der X-Versatz zum Ende der neuen Zeile

`dy` der Y-Versatz zum Ende der neuen Zeile

5.70 ccontext:RelMoveTo

BEZEICHNUNG

ccontext:RelMoveTo – Beginnt einen neuen Unterpfad mit einem relativen Punkt

ÜBERSICHT

```
ccontext:RelMoveTo(dx, dy)
```

BESCHREIBUNG

Beginnt einen neuen Unterpfad. Nach diesem Aufruf wird der aktuelle Punkt um (`x`, `y`) verschoben.

Bei einem aktuellen Punkt von (x, y) ist der Aufruf dieser Funktion mit `dx`, `dy` logisch äquivalent zum Aufruf von ccontext:MoveTo() mit `x+dx` und `y+dy` .

Es ist ein Fehler, diese Funktion ohne aktuellen Punkt aufzurufen. Dies führt dazu, dass der Kontext mit dem Status `#CAIRO_STATUS_NO_CURRENT_POINT` heruntergefahren wird.

EINGABEN

`dx` der X Versatz

`dy` der Y Versatz

5.71 ccontext:ResetClip

BEZEICHNUNG

ccontext:ResetClip – Stellt die aktuelle Clip-Region zurück

ÜBERSICHT

```
ccontext:ResetClip()
```

BESCHREIBUNG

Setzt den aktuellen Clipbereich auf seinen ursprünglichen, uneingeschränkten Zustand zurück. Das heißt, Sie stellen den Clipbereich auf eine unendlich große Form ein, die die Zieloberfläche enthält. Wenn die Unendlichkeit zu schwer zu erfassen ist, kann man sich gleichermaßen vorstellen, dass der Clip-Bereich auf die genauen Grenzen der Zieloberfläche zurückgesetzt wird.

Beachten Sie, dass Code, der wiederverwendbar sein soll, ccontext:ResetClip() nicht aufrufen sollte, da dies zu Ergebnissen führt, die von übergeordnetem Code, der ccontext:Clip() aufruft, unerwartet sind. Erwägen Sie die Verwendung von ccontext:Save() und ccontext:Restore() rund um ccontext:Clip() als robusteres Mittel zur vorübergehenden Einschränkung des Clipbereichs.

EINGABEN

Keine

5.72 ccontext:Restore

BEZEICHNUNG
ccontext:Restore – Stellt den gespeicherten Zustand wieder her

ÜBERSICHT
`ccontext:Restore()`

BESCHREIBUNG
Stellt den Kontext in dem Zustand wieder her, der durch einen vorherigen Aufruf von ccontext:Save() gespeichert wurde, und entfernt diesen Zustand aus dem Stapel der gespeicherten Zustände.

EINGABEN
Keine

5.73 ccontext:Rotate

BEZEICHNUNG
ccontext:Rotate – Dreht die aktuelle Transformationsmatrix

ÜBERSICHT
`ccontext:Rotate(angle)`

BESCHREIBUNG
Ändert die aktuelle Transformationsmatrix (CTM), indem die Benutzerbereichachsen um `Winkel` im Bogenmaß gedreht werden. Die Drehung der Achsen erfolgt nach einer eventuell vorhandenen Transformation des Benutzerbereichs. Die Drehrichtung für positive Winkel verläuft von der positiven X-Achse zur positiven Y-Achse.

EINGABEN

`angle`	Winkel (im Bogenmaß), um den die Benutzerbereichachsen gedreht werden

5.74 ccontext:Save

BEZEICHNUNG
ccontext:Save – Speichert den aktuellen Zustand

ÜBERSICHT
`ccontext:Save()`

BESCHREIBUNG
Erstellt eine Kopie des aktuellen Status des Kontexts und speichert sie auf einem internen Stapel gespeicherter Status für den Kontext. Wenn ccontext:Restore() aufgerufen wird, wird der Kontext auf den gespeicherten Zustand zurückgesetzt. Mehrere Aufrufe von ccontext:Save() und ccontext:Restore() können verschachtelt werden; jeder Aufruf von ccontext:Restore() stellt den Status aus dem passenden gepaarten ccontext:Save() wieder her.

Es ist nicht notwendig, alle gespeicherten Zustände zu löschen, bevor ein Cairo-Kontext gelöscht wird. Wenn der Referenzzähler eines Cairo-Kontexts als Reaktion auf einen

Aufruf von ccontext:Free() auf Null sinkt, werden alle gespeicherten Zustände zusammen mit dem Cairo-Kontext gelöscht und freigegeben.

EINGABEN
Keine

5.75 ccontext:Scale

BEZEICHNUNG
ccontext:Scale – Skaliert die aktuelle Transformationsmatrix

ÜBERSICHT
```
ccontext:Scale(sx, sy)
```

BESCHREIBUNG
Ändert die aktuelle Transformationsmatrix (CTM), indem die X- und Y-Benutzerbereichachsen um `sx` bzw. `sy` skaliert werden. Die Skalierung der Achsen erfolgt nach einer eventuell vorhandenen Transformation des Benutzerbereichs.

EINGABEN

`sx`	Skalierungsfaktor für die X-Dimension
`sy`	Skalierungsfaktor für die Y-Dimension

5.76 ccontext:SelectFontFace

BEZEICHNUNG
ccontext:SelectFontFace – Wählt die Schriftart aus

ÜBERSICHT
```
ccontext:SelectFontFace(family$, slant, weight)
```

BESCHREIBUNG
Hinweis: Der Funktionsaufruf ccontext:SelectFontFace() ist Teil dessen, was die Cairoer Designer als "Toy"-Text-API bezeichnen. Es ist praktisch für kurze Demos und einfache Programme, wird aber voraussichtlich nicht für ernsthafte Textanwendungen geeignet sein.

Wählt eine Familie und einen Schriftstil aus einer vereinfachten Beschreibung wie Familienname, Neigung und Stärke aus. Cairo bietet keine Funktion zum Auflisten verfügbarer Familiennamen im System (dies ist ein "Toy", denken Sie daran), aber die standardmäßigen generischen CSS2-Familiennamen ("serif", "sans-serif", "cursive", "fantasy", "Monospace") funktionieren wahrscheinlich wie erwartet.

Wenn `family$` mit der Zeichenkette "cairo:" beginnt oder keine nativen Schriftarten-Backends kompiliert sind, verwendet Cairo eine interne Schriftartenfamilie. Die interne Schriftfamilie erkennt viele Modifikatoren in der Zeichenkette `family$`, insbesondere die Zeichenkette "monospace". Das heißt, der Familienname "cairo:monospace" verwendet die Monospace-Version der internen Schriftfamilie.

Informationen zur "echten" Schriftartenauswahl finden Sie in den Schriftarten-Backend-spezifischen Schriftarten-Erstellungsfunktionen für das Schriftarten-Backend, das Sie verwenden, z.B. cairo.FontFace(). Die resultierende Schriftart könnte dann mit cairo.ScaledFont() und ccontext:SetScaledFont() verwendet werden.

Ebenso können Sie bei Verwendung der "echten" Schriftartenunterstützung direkt das zugrunde liegende Schriftartensystem aufrufen, z.B. "fontconfig" oder "freetype", um Vorgänge wie das Auflisten verfügbarer Schriftarten usw. durchzuführen.

Es wird erwartet, dass die meisten Anwendungen eine umfassendere Schriftartenverarbeitungs- und Textlayoutbibliothek (z.B. Pango) in Verbindung mit Cairo verwenden müssen.

Wenn Text ohne einen Aufruf von ccontext:SelectFontFace(), ccontext:SetFontFace() oder ccontext:SetScaledFont() gezeichnet wird, ist die Standardfamilie plattformspezifisch, aber im Wesentlichen "serifenlos".

Der `slant-Parameter` kann eine der folgenden Konstanten sein:

```
#CAIRO_FONT_SLANT_NORMAL
#CAIRO_FONT_SLANT_ITALIC
#CAIRO_FONT_SLANT_OBLIQUE
```

Der `weight-Parameter` kann eine der folgenden Konstanten sein:

```
#CAIRO_FONT_WEIGHT_NORMAL
#CAIRO_FONT_WEIGHT_BOLD
```

Die Standardneigung ist `#CAIRO_FONT_SLANT_NORMAL` und die Standardstärke ist `#CAIRO_FONT_WEIGHT_NORMAL`.

Diese Funktion entspricht einem Aufruf von cairo.ToyFontFace() gefolgt von ccontext:SetFontFace().

EINGABEN

`family$`	ein Schriftfamilienname
`slant`	die Neigung für die Schriftart (siehe oben)
`weight`	die Stärke der Schriftart (siehe oben)

5.77 ccontext:SetAntialias

BEZEICHNUNG

ccontext:SetAntialias – Legt das Antialias fest

ÜBERSICHT

```
ccontext:SetAntialias(antialias)
```

BESCHREIBUNG

Legt den Antialiasing-Modus des Rasters fest, der zum Zeichnen von Formen verwendet wird. Der Parameter `antialias` kann eine der folgenden Konstanten sein:

`#CAIRO_ANTIALIAS_DEFAULT`
: Verwendt das Standard-Antialiasing für das Subsystem und das Zielgerät.

`#CAIRO_ANTIALIAS_NONE`
: Verwendt eine Bilevel-Alphamaske.

`#CAIRO_ANTIALIAS_GRAY`
: Führt einfarbiges Antialiasing durch (z.B. mit Grautönen für schwarzen Text auf weißem Hintergrund).

`#CAIRO_ANTIALIAS_SUBPIXEL`
: Führt Antialiasing durch, indem Sie die Reihenfolge der Subpixelelemente auf Geräten wie LCD-Panels nutzen.

`#CAIRO_ANTIALIAS_FAST`
: Führt Antialiasing durch, aber die Geschwindigkeit wird der Qualität vorgezogen.

`#CAIRO_ANTIALIAS_GOOD`
: Das Backend sollte Qualität und Leistung in Einklang bringen.

`#CAIRO_ANTIALIAS_BEST`
: Das Backend rendert mit höchster Qualität und nimmt gegebenenfalls Geschwindigkeitseinbußen hin.

Beachten Sie, dass der `antialias-Wert` ein Hinweis ist und ein bestimmtes Backend einen bestimmten Wert unterstützen kann oder auch nicht. Derzeit unterstützt kein Backend `#CAIRO_ANTIALIAS_SUBPIXEL` beim Zeichnen von Formen.

Beachten Sie, dass diese Option keinen Einfluss auf die Textwiedergabe hat, siehe stattdessen cfontoptions:SetAntialias().

EINGABEN

`antialias`
: der neue Antialiasing-Modus (siehe oben).

5.78 ccontext:SetDash

BEZEICHNUNG

ccontext:SetDash – Legt das Strichmuster fest

ÜBERSICHT

```
ccontext:SetDash(offset[, dash1, ...])
```

BESCHREIBUNG

Legt das Strichmuster fest, das von ccontext:Stroke() verwendet werden soll. Ein Strichmuster wird durch eine Reihe positiver Strichwerte angegeben, beginnend mit `dash1`. Jeder Wert gibt die Länge der abwechselnden `"Ein"`- und `"Aus"`-Abschnitte des Strichmusters an. Der `offset` gibt einen Versatz im Muster an, bei dem der Strich beginnt.

Auf jedes `"Ein"`-Segment werden Obergrenzen angewendet, als wäre das Segment ein separater Unterpfad. Insbesondere ist es gültig, eine `"Ein"`-Länge von 0,0 mit `#CAIRO_LINE_CAP_ROUND` oder `#CAIRO_LINE_CAP_SQUARE` zu verwenden, um Punkte oder Quadrate entlang eines Pfads zu verteilen.

Hinweis: Die Längenwerte sind in Benutzerbereicheinheiten angegeben, die zum Zeitpunkt des Zeichnens ausgewertet wurden. Dies ist nicht unbedingt derselbe wie der Benutzerbereich zum Zeitpunkt von ccontext:SetDash().

Wenn Sie die optionalen Argumente `dash1` usw. weglassen, ist die Strichelung deaktiviert.

Wenn Sie nur einen einzelnen Strichmusterwert übergeben, wird ein symmetrisches Muster mit abwechselnd ein- und ausgeschalteten Teilen der Größe angenommen, die durch den einzelnen Wert in `dash1` angegeben wird.

Wenn ein Strichmusterwert negativ ist oder alle Werte 0 sind, wird der Cairo-Kontext in einen Fehlerzustand mit dem Status `#CAIRO_STATUS_INVALID_DASH` versetzt.

EINGABEN

`offset` ein Versatz im Strichmuster, bei dem der Strich beginnen soll

`dash1` Optional: Wert für den ersten Strichmusterabschnitt (im Abschnitt).

`...` Optional: Weitere Werte, die abwechselnde Längen der Ein- und Aus-Strichmusterabschnitte angeben

5.79 ccontext:SetFillRule

BEZEICHNUNG

ccontext:SetFillRule – Legt die aktuelle Füllregel fest

ÜBERSICHT

```
ccontext:SetFillRule(fill_rule)
```

BESCHREIBUNG

Legt die aktuelle Füllregel im Cairo-Kontext fest. Die Füllregel wird verwendet, um zu bestimmen, welche Regionen innerhalb oder außerhalb eines komplexen (möglicherweise sich selbst schneidenden) Pfads liegen. Die aktuelle Füllregel betrifft sowohl ccontext:Fill() als auch ccontext:Clip(). Die folgenden Werte können für `fill_rule` übergeben werden:

`#CAIRO_FILL_RULE_WINDING`
: Wenn der Pfad den Strahl von links nach rechts kreuzt, zählt +1. Wenn der Pfad den Strahl von rechts nach links kreuzt, zählt -1. Links und rechts werden aus der Perspektive des Blicks entlang des Strahls vom Startpunkt aus bestimmt. Wenn die Gesamtzahl ungleich Null ist, wird der Punkt aufgefüllt.

`#CAIRO_FILL_RULE_EVEN_ODD`
: Zählt die Gesamtzahl der Schnittpunkte, unabhängig von der Ausrichtung der Kontur. Wenn die Gesamtzahl der Schnittpunkte ungerade ist, wird der Punkt gefüllt.

Die Standardfüllregel ist `#CAIRO_FILL_RULE_WINDING`.

EINGABEN

`fill_rule`
: eine Füllregel (siehe oben).

5.80 ccontext:SetFontFace

BEZEICHNUNG

ccontext:SetFontFace – Legt die Schriftart fest

ÜBERSICHT

```
ccontext:SetFontFace(font_face)
```

BESCHREIBUNG

Ersetzt das aktuelle Schriftartobjekt im Cairo-Kontext durch `font_face`. Die ersetzte Schriftart im Cairo-Kontext wird gelöscht, wenn keine weiteren Verweise darauf vorhanden sind.

EINGABEN

`font_face`
: ein Schriftartobjekt oder `Nil`, um die Standardschriftart wiederherzustellen

5.81 ccontext:SetFontMatrix

BEZEICHNUNG

ccontext:SetFontMatrix – Legt die Schriftartmatrix fest

ÜBERSICHT

```
ccontext:SetFontMatrix(matrix)
```

BESCHREIBUNG

Setzt die aktuelle Schriftartmatrix auf `matrix`. Die Schriftartmatrix ermöglicht eine Transformation vom Designbereich der Schriftart (in diesem Bereich beträgt das em-Quadrat 1 Einheit mal 1 Einheit) in den Benutzerbereich. Normalerweise wird eine einfache Skalierung verwendet (siehe ccontext:SetFontSize()), aber eine komplexere Schriftmatrix kann verwendet werden, um die Schrift zu scheren oder entlang der beiden Achsen ungleichmäßig zu strecken.

EINGABEN

`matrix`
: ein Matrixobjekt, das eine Transformation beschreibt, die auf die aktuelle Schriftart angewendet werden soll

5.82 ccontext:SetFontOptions

BEZEICHNUNG

ccontext:SetFontOptions – Legt die Schriftartoptionen fest

ÜBERSICHT

```
ccontext:SetFontOptions(options)
```

BESCHREIBUNG

Legt eine Reihe benutzerdefinierter Schriftart-Rendering-Optionen für den Cairo-Kontext fest. Rendering-Optionen werden durch Zusammenführen dieser Optionen mit den von der zugrunde liegenden Oberfläche abgeleiteten Optionen abgeleitet. Wenn der

Wert in `options` einen Standardwert hat (wie `#CAIRO_ANTIALIAS_DEFAULT`), dann wird der Wert von der Oberfläche verwendet.

EINGABEN

`options` Schriftartoptionen-Objekt, das verwendet werden soll

5.83 ccontext:SetFontSize

BEZEICHNUNG

ccontext:SetFontSize – Stellt die Schriftgröße ein

ÜBERSICHT

`ccontext:SetFontSize(size)`

BESCHREIBUNG

Setzt die aktuelle Schriftartmatrix auf eine Skalierung um einen Faktor von `size` und ersetzt jede Schriftartmatrix, die zuvor mit ccontext:SetFontSize() oder ccontext:SetFontMatrix() festgelegt wurde. Daraus ergibt sich eine Schriftgröße von `size`-Benutzerbereicheinheiten. Genauer gesagt führt diese Matrix dazu, dass das em-Quadrat der Schriftart ein `size-mal-§size-Quadrat` im Benutzerbereich ist.

Wenn Text ohne Aufruf von ccontext:SetFontSize(), ccontext:SetFontMatrix() oder ccontext:SetScaledFont() gezeichnet wird, ist die Standardschriftgröße 10,0.

EINGABEN

`size` die neue Schriftgröße in Benutzerbereicheinheiten

5.84 ccontext:SetLineCap

BEZEICHNUNG

ccontext:SetLineCap – Legt den Zeilenkopfstil fest

ÜBERSICHT

`ccontext:SetLineCap(line_cap)`

BESCHREIBUNG

Legt den aktuellen Zeilenkopfstil im Kairo-Kontext fest. Der Parameter `line_cap` kann eine der folgenden Konstanten sein:

`#CAIRO_LINE_CAP_BUTT`
: Beginnt (stoppt) die Linie genau am Startpunkt (Endpunkt).

`#CAIRO_LINE_CAP_ROUND`
: Verwendet einen runden Abschluss, der Mittelpunkt des Kreises ist der Endpunkt.

`#CAIRO_LINE_CAP_SQUARE`
: Verwendet ein quadratisches Ende. Die Mitte des Quadrats ist der Endpunkt.

Wie bei den anderen Strichparametern wird der aktuelle Zeilenkopfstil von ccontext:Stroke() und ccontext:StrokeExtents() untersucht, hat jedoch keine Auswirkung bei der Pfadkonstruktion.

Der Standard-Zeilenkopfstil ist `#CAIRO_LINE_CAP_BUTT`.

EINGABEN

`line_cap` ein Zeilenumbruchstil (siehe oben).

5.85 ccontext:SetLineJoin

BEZEICHNUNG

ccontext:SetLineJoin – Legt den Linienverbindungsstil fest

ÜBERSICHT

```
ccontext:SetLineJoin(line_join)
```

BESCHREIBUNG

Legt den aktuellen Linienverbindungsstil im Cairo-Kontext fest. Der Parameter `line_join` kann eine der folgenden Konstanten sein:

`#CAIRO_LINE_JOIN_MITER`
: Verwendet eine scharfe (abgewinkelte) Ecke. Siehe Abschnitt 5.88 [ccontext:SetMiterLimit], Seite 75, für Details.

`#CAIRO_LINE_JOIN_ROUND`
: Verwendet eine abgerundete Verbindung. Der Mittelpunkt des Kreises ist der Verbindungspunkt.

`#CAIRO_LINE_JOIN_BEVEL`
: Verwendet eine abgeschnittene Verbindung. Die Verbindung wird auf halber Linienbreite vom Verbindungspunkt abgeschnitten.

Wie bei den anderen Strichparametern wird der aktuelle Linienverbindungsstil von ccontext:Stroke() und ccontext:StrokeExtents() untersucht, hat aber bei der Pfadkonstruktion keine Auswirkung.

Der Standardlinienverbindungsstil ist `#CAIRO_LINE_JOIN_MITER`.

EINGABEN

`line_join`
: ein Linienverbindungsstil (siehe oben).

5.86 ccontext:SetLineWidth

BEZEICHNUNG

ccontext:SetLineWidth – Stellt die Linienbreite ein

ÜBERSICHT

```
ccontext:SetLineWidth(width)
```

BESCHREIBUNG

Legt die aktuelle Linienbreite im Cairo-Kontext fest. Der Linienbreitenwert gibt den Durchmesser eines Stifts an, der im Benutzerbereich kreisförmig ist (obwohl der Stift im Gerätebereich aufgrund der Skalierung/Scherung/Rotation des CTM im Allgemeinen eine Ellipse sein kann).

Hinweis: Wenn sich die obige Beschreibung auf Benutzerbereich und CTM bezieht, bezieht sie sich auf den Benutzerbereich und die CTM, die zum Zeitpunkt des Zeichnungsvorgangs gültig waren, nicht auf den Benutzerbereich und die CTM, die zum Zeitpunkt des Aufrufs von ccontext:SetLineWidth() gültig waren. Die einfachste Verwendung macht beide Räume identisch. Das heißt, wenn es zwischen einem Aufruf von ccontext:SetLineWidth() und dem Zeichnungsvorgang keine Änderung am CTM gibt, kann man einfach Benutzerbereichswerte an ccontext:SetLineWidth() übergeben und ignorieren diesen Hinweis.

Wie bei den anderen Linienparametern wird die aktuelle Linienbreite von ccontext:Stroke() und ccontext:StrokeExtents() untersucht, hat aber bei der Pfadkonstruktion keine Auswirkung.

Der Standardwert für die Linienstärke beträgt 2.0.

EINGABEN

`width` eine Linienstärke

5.87 ccontext:SetMatrix

BEZEICHNUNG

ccontext:SetMatrix – Legt die aktuelle Transformationsmatrix fest

ÜBERSICHT

```
ccontext:SetMatrix(matrix)
```

BESCHREIBUNG

Ändert die aktuelle Transformationsmatrix (CTM), indem sie gleich `matrix` gesetzt wird.

EINGABEN

`matrix` eine Transformationsmatrix vom Benutzerbereich zum Gerätebereich

5.88 ccontext:SetMiterLimit

BEZEICHNUNG

ccontext:SetMiterLimit – Stellt die aktuelle Gehrungsgrenze ein

ÜBERSICHT

```
ccontext:SetMiterLimit(limit)
```

BESCHREIBUNG

Legt die aktuelle Gehrungsgrenze im Cairo-Kontext fest.

Wenn der aktuelle Linienverbindungsstil auf `#CAIRO_LINE_JOIN_MITER` eingestellt ist (siehe ccontext:SetLineJoin()), wird die Gehrungsgrenze verwendet, um zu bestimmen,

ob die Linien mit einer Abschrägung statt mit einer Gehrung verbunden werden sollen. Cairo dividiert die Länge der Gehrung durch die Linienbreite. Wenn das Ergebnis größer als die Gehrungsgrenze ist, wird der Stil in eine Abschrägung umgewandelt.

Wie bei den anderen Linienparametern wird der aktuelle Grenzwert für die Liniengehrung von ccontext:Stroke() und ccontext:StrokeExtents() untersucht, hat jedoch keine Auswirkung auf die Pfadkonstruktion.

Der standardmäßige Gehrungsgrenzwert ist 10.0, wodurch Verbindungen mit Innenwinkeln von weniger als 11 Grad in Abschrägungen statt in Gehrungen umgewandelt werden. Als Referenz: Bei einem Gehrungsgrenzwert von 2.0 liegt der Gehrungsschnitt bei 60 Grad und bei einem Gehrungsgrenzwert von 1.414 bei 90 Grad.

Eine Gehrungsgrenze für einen gewünschten Winkel kann berechnet werden als:

```
miterlimit = 1/sin(angle/2)
```

EINGABEN

`limit` einzustellende Gehrungsgrenze

5.89 ccontext:SetOperator

BEZEICHNUNG

ccontext:SetOperator – Legt den aktuellen Kompositionsoperator fest

ÜBERSICHT

`ccontext:SetOperator(op)`

BESCHREIBUNG

Legt den Kompositionsoperator fest, der für alle Zeichenvorgänge verwendet werden soll. Der Parameter `op` kann eine der folgenden Konstanten sein:

`#CAIRO_OPERATOR_CLEAR`
: Zielebene löschen (begrenzt).

`#CAIRO_OPERATOR_SOURCE`
: Zielebene ersetzen (begrenzt).

`#CAIRO_OPERATOR_OVER`
: Zeichnet die Quellebene über der Zielebene (begrenzt).

`#CAIRO_OPERATOR_IN`
: Zeichnet die Quelle dort, wo sich der Zielinhalt befand (unbegrenzt).

`#CAIRO_OPERATOR_OUT`
: Zeichnet die Quelle dort, wo es keinen Zielinhalt gab (unbegrenzt).

`#CAIRO_OPERATOR_ATOP`
: Zeichnet die Quelle über den Zielinhalt und nur dort.

`#CAIRO_OPERATOR_DEST`
: Ignoriert die Quelle.

`#CAIRO_OPERATOR_DEST_OVER`
: Zeichnet das Ziel über der Quelle.

`#CAIRO_OPERATOR_DEST_IN`
: Beläst das Ziel nur dort, wo Quellinhalt vorhanden war (unbegrenzt).

`#CAIRO_OPERATOR_DEST_OUT`
: Beläst das Ziel nur dort, wo kein Quellinhalt vorhanden ist.

`#CAIRO_OPERATOR_DEST_ATOP`
: Beläst das Ziel über dem Quellinhalt und nur dort (unbegrenzt).

`#CAIRO_OPERATOR_XOR`
: Quelle und Ziel werden dort angezeigt, wo es nur eines davon gibt.

`#CAIRO_OPERATOR_ADD`
: Quell- und Zielebene werden akkumuliert.

`#CAIRO_OPERATOR_SATURATE`
: Wie oben, aber unter der Annahme, dass Quelle und Ziel nicht gemeinsame Geometrien haben.

`#CAIRO_OPERATOR_MULTIPLY`
: Quell- und Zielebene werden multipliziert. Dadurch ist das Ergebnis mindestens so dunkel wie die dunkelste Eingabe.

`#CAIRO_OPERATOR_SCREEN`
: Quelle und Ziel werden ergänzt und vervielfacht. Dadurch ist das Ergebnis mindestens so hell wie die hellste Eingabe.

`#CAIRO_OPERATOR_OVERLAY`
: Je nach Helligkeit der Zielfarbe wird es multipliziert oder gerastert.

`#CAIRO_OPERATOR_DARKEN`
: Ersetzt das Ziel durch die Quelle, wenn es dunkler ist, andernfalls bleibt die Quelle erhalten.

`#CAIRO_OPERATOR_LIGHTEN`
: Ersetzt das Ziel durch die Quelle, wenn es heller ist, andernfalls bleibt die Quelle erhalten.

`#CAIRO_OPERATOR_COLOR_DODGE`
: Hellt die Zielfarbe auf, um die Quellfarbe widerzuspiegeln.

`#CAIRO_OPERATOR_COLOR_BURN`
: Verdunkelt die Zielfarbe, um die Quellfarbe widerzuspiegeln.

`#CAIRO_OPERATOR_HARD_LIGHT`
: Multipliziert oder rastert, abhängig von der Quellfarbe.

`#CAIRO_OPERATOR_SOFT_LIGHT`
: Je nach Quellfarbe wird es dunkler oder heller.

`#CAIRO_OPERATOR_DIFFERENCE`
: Nimmt den Unterschied zwischen Quell- und Zielfarbe.

`#CAIRO_OPERATOR_EXCLUSION`
: Erzeugt einen ähnlichen Effekt wie "DIFFERENCE", jedoch mit geringerem Kontrast.

`#CAIRO_OPERATOR_HSL_HUE`
: Erstellt eine Farbe mit dem Farbton der Quelle und der Sättigung und Leuchtkraft des Ziels.

`#CAIRO_OPERATOR_HSL_SATURATION`
: Erstellt eine Farbe mit der Sättigung der Quelle und dem Farbton und der Leuchtkraft des Ziels. Das Malen mit diesem Modus auf eine graue Fläche führt zu keiner Veränderung.

`#CAIRO_OPERATOR_HSL_COLOR`
: Erstellt eine Farbe mit dem Farbton und der Sättigung der Quelle und der Leuchtkraft des Ziels. Dadurch bleiben die Graustufen des Ziels erhalten und eignen sich zum Einfärben von Schwarzweißbildern oder zum Tönen von Farbbildern.

`#CAIRO_OPERATOR_HSL_LUMINOSITY`
: Erstellt eine Farbe mit der Leuchtkraft der Quelle und dem Farbton und der Sättigung des Ziels. Dies erzeugt einen umgekehrten Effekt zu `#CAIRO_OPERATOR_HSL_COLOR`.

Der Standardoperator ist `#CAIRO_OPERATOR_OVER`.

EINGABEN

`op` ein Kompositionsoperator (siehe oben).

5.90 ccontext:SetScaledFont

BEZEICHNUNG

ccontext:SetScaledFont – Legt die skalierte Schriftart fest

ÜBERSICHT

```
ccontext:SetScaledFont(scaled_font)
```

BESCHREIBUNG

Ersetzt das aktuelle Schriftbild, die Schriftmatrix und die Schriftoptionen im Cairo-Kontext durch die des `scaled_font`. Mit Ausnahme einiger Verschiebungen sollte der aktuelle CTM des Cairo-Kontexts mit dem des `scale_font` identisch sein, auf den über cscaledfont:GetCTM() zugegriffen werden kann.

EINGABEN

`scaled_font`
: ein skaliertes Schriftartobjekt

5.91 ccontext:SetSource

BEZEICHNUNG

ccontext:SetSource – Legt das Quellmuster fest

ÜBERSICHT

```
ccontext:SetSource(source)
```

BESCHREIBUNG

Setzt das Quellmuster im Kontext auf `source`. Dieses Muster wird dann für alle nachfolgenden Zeichenvorgänge verwendet, bis ein neues Quellmuster festgelegt wird.

Hinweis: Die Transformationsmatrix des Musters ist an den zum Zeitpunkt von ccontext:SetSource() gültigen Benutzerbereich gebunden. Dies bedeutet, dass weitere Änderungen der aktuellen Transformationsmatrix keinen Einfluss auf das Quellmuster haben. Siehe cpattern:SetMatrix().

Das Standardquellmuster ist ein durchgehendes Muster in undurchsichtigem Schwarz, das heißt, es entspricht ccontext:SetSourceRGB(), wobei alle Argumente auf 0.0 gesetzt sind.

EINGABEN

`source`	ein Cairo-Musterobjekt, das als Quelle für nachfolgende Zeichenvorgänge verwendet werden soll

5.92 ccontext:SetSourceRGB

BEZEICHNUNG

ccontext:SetSourceRGB – Stellt die RGB-Quelle ein

ÜBERSICHT

```
ccontext:SetSourceRGB(red, green, blue)
```

BESCHREIBUNG

Setzt das Quellmuster im Kontext auf eine undurchsichtige Farbe. Diese undurchsichtige Farbe wird dann für alle nachfolgenden Zeichenvorgänge verwendet, bis ein neues Quellmuster festgelegt wird.

Die Farbkomponenten sind Gleitkommazahlen im Bereich von 0 bis 1. Wenn die übergebenen Werte außerhalb dieses Bereichs liegen, werden sie geklemmt.

Das Standardquellmuster ist ein durchgehendes Muster in undurchsichtigem Schwarz, das heißt, es entspricht ccontext:SetSourceRGB(), wobei alle Argumente auf 0.0 gesetzt sind.

EINGABEN

`red`	roter Farbanteil
`green`	grüner Farbanteil
`blue`	blauer Farbanteil

5.93 ccontext:SetSourceRGBA

BEZEICHNUNG

ccontext:SetSourceRGBA – Legt die RGBA-Quelle fest

ÜBERSICHT

```
ccontext:SetSourceRGBA(red, green, blue, alpha)
```

BESCHREIBUNG

Setzt das Quellmuster im Kontext auf eine durchscheinende Farbe. Diese Farbe wird dann für alle nachfolgenden Zeichenvorgänge verwendet, bis ein neues Quellmuster festgelegt wird.

Die Farb- und Alphakomponenten sind Gleitkommazahlen im Bereich von 0 bis 1. Wenn die übergebenen Werte außerhalb dieses Bereichs liegen, werden sie geklemmt.

Beachten Sie, dass die Farb- und Alphawerte nicht vormultipliziert werden.

Das Standardquellmuster ist ein durchgehendes Muster in undurchsichtigem Schwarz, das heißt, es entspricht ccontext:SetSourceRGB(), wobei alle Argumente auf 0.0 gesetzt sind.

EINGABEN

`red`	roter Farbanteil
`green`	grüner Farbanteil
`blue`	blauer Farbanteil
`alpha`	Alpha-Komponente der Farbe

5.94 ccontext:SetSourceSurface

BEZEICHNUNG

ccontext:SetSourceSurface – Legt die Quelloberfläche fest

ÜBERSICHT

```
ccontext:SetSourceSurface(surface, x, y)
```

BESCHREIBUNG

Dies ist eine praktische Funktion zum Erstellen eines Musters aus `surface` und zum festlegen als Quelle im Kontext mit ccontext:SetSource().

Die Parameter `x` und `y` geben die Benutzerbereichkoordinate an, an der der Oberflächenursprung erscheinen soll. Der Oberflächenursprung ist die obere linke Ecke, bevor eine Transformation angewendet wurde. Die Parameter `x` und `y` werden negiert und dann als Verschiebungswerte in der Mustermatrix festgelegt.

Abgesehen von der anfänglichen Verschiebungsmustermatrix, wie oben beschrieben, werden alle anderen Musterattribute, wie z.B. der Erweiterungsmodus, auf die Standardwerte wie in cairo.PatternForSurface() gesetzt. Das resultierende Muster kann mit ccontext:GetSource() abgefragt werden, sodass diese Attribute bei Bedarf geändert werden können (z.B. um ein sich wiederholendes Muster mit cpattern:SetExtend() zu erstellen).

EINGABEN

`surface`	eine Oberfläche, die zum festlegen des Quellmusters verwendet werden soll
`x`	X-Koordinate im Benutzerbereich für den Oberflächenursprung
`y`	Y-Koordinate im Benutzerbereich für den Oberflächenursprung

5.95 ccontext:SetTolerance

BEZEICHNUNG

ccontext:SetTolerance – Stellt die Toleranz ein

ÜBERSICHT

```
ccontext:SetTolerance(tolerance)
```

BESCHREIBUNG

Legt die Toleranz fest, die beim Konvertieren von Pfaden in Trapeze verwendet wird. Gekrümmte Abschnitte des Pfads werden so lange unterteilt, bis die maximale Abweichung zwischen dem ursprünglichen Pfad und der polygonalen Näherung kleiner als `Toleranz` ist. Der Standardwert ist 0.1. Ein größerer Wert führt zu einer besseren Leistung, ein kleinerer Wert zu einem besseren Erscheinungsbild. Eine Verringerung des Werts gegenüber dem Standardwert von 0.1 wird das Erscheinungsbild wahrscheinlich nicht wesentlich verbessern. Die Genauigkeit von Pfaden innerhalb Cairos wird durch die Präzision ihrer internen Arithmetik begrenzt, und die vorgeschriebene `tolerance` ist auf den kleinsten darstellbaren internen Wert beschränkt.

EINGABEN

`tolerance`
: die Toleranz in Geräteeinheiten (normalerweise Pixel).

5.96 ccontext:ShowErrorUnderline

BEZEICHNUNG

ccontext:ShowErrorUnderline – Unterstreicht die Fehler beim Zeichnen

ÜBERSICHT

```
ccontext:ShowErrorUnderline(x, y, width, height)
```

BESCHREIBUNG

Zeichnet im angegebenen Cairo-Kontext eine verschnörkelte Linie, die ungefähr das angegebene Rechteck im Stil einer Unterstreichung abdeckt, die zur Anzeige eines Rechtschreibfehlers verwendet wird.

Die Breite der Unterstreichung wird auf eine ganze Zahl von Aufwärts-/Abwärtssegmenten gerundet und das resultierende Rechteck wird im ursprünglichen Rechteck zentriert.

EINGABEN

`x` die X-Koordinate einer Ecke des Rechtecks

`y` die Y-Koordinate einer Ecke des Rechtecks

`width` nicht negative Breite des Rechtecks

`height` nicht negative Höhe des Rechtecks

5.97 ccontext:ShowGlyphItem

BEZEICHNUNG

ccontext:ShowGlyphItem – Zeigt ein Glyphenelement an

ÜBERSICHT

```
ccontext:ShowGlyphItem(text$, glyph_item)
```

BESCHREIBUNG

Zeichnet die Glyphen in `glyph_item` im angegebenen Cairo-Kontext und bettet den mit den Glyphen verknüpften Text in die Ausgabe ein, wenn das Ausgabeformat dies unterstützt (z.B. PDF). Andernfalls verhält es sich ähnlich wie ccontext:ShowGlyphString().

Der Ursprung der Glyphen (der linke Rand der Grundlinie) wird am aktuellen Punkt des Cairo-Kontexts gezeichnet.

Beachten Sie, dass `text` der Anfang des Textes für das Layout ist, der dann durch das Offset-Element des Glyphenelements indiziert wird.

EINGABEN

`text$` der Text, auf den sich `glyph_item` bezieht

`glyph_item`
ein Pango-Glyphen-Elementobjekt

5.98 ccontext:ShowGlyphs

BEZEICHNUNG

ccontext:ShowGlyphs – Zeigt ein Glyphen an

ÜBERSICHT

```
ccontext:ShowGlyphs(glyphs[, offset, num_glyphs])
```

BESCHREIBUNG

Ein Zeichenoperator, der die Form aus einem Glyphen-Versatz generiert und entsprechend der aktuellen Schriftart, Schriftgröße (Schriftmatrix) und Schriftartoptionen gerendert wird.

EINGABEN

`glyphs` Glyphen-Versatz, das angezeigt werden soll

`offset` Optional: Versatz im Array, das das Startzeichen angibt (standardmäßig 0, was das erste Zeichen bedeutet).

`num_glyphs`
Optional: Anzahl der anzuzeigenden Glyphen (Standard ist -1, was alle Glyphen bedeutet).

5.99 ccontext:ShowGlyphString

BEZEICHNUNG

ccontext:ShowGlyphString – Zeigt die Glyphen-Zeichenkette an

ÜBERSICHT

```
ccontext:ShowGlyphString(font, glyphs)
```

BESCHREIBUNG

Zeichnet die Glyphen in `glyphs` im angegebenen Cairo-Kontext.

Der Ursprung der Glyphen (der linke Rand der Grundlinie) wird am aktuellen Punkt des Cairo-Kontexts gezeichnet.

EINGABEN

`font`	ein Pango-Schriftartobjekt
`glyphs`	ein Pango-Glyphen-Zeichenkette-Objekt

5.100 ccontext:ShowLayout

BEZEICHNUNG

ccontext:ShowLayout – Zeigt das Layout an

ÜBERSICHT

```
ccontext:ShowLayout(layout)
```

BESCHREIBUNG

Zeichnet ein Pango-Layout im angegebenen Cairo-Kontext.

Die obere linke Ecke des Pango-Layouts wird am aktuellen Punkt des Cairo-Kontexts gezeichnet.

EINGABEN

`layout`	ein Pango-Layoutobjekt

5.101 ccontext:ShowLayoutLine

BEZEICHNUNG

ccontext:ShowLayoutLine – Zeigt die Layoutzeile an

ÜBERSICHT

```
ccontext:ShowLayoutLine(line)
```

BESCHREIBUNG

Zeichnet eine Pango-Layoutzeile im angegebenen Cairo-Kontext.

Der Ursprung der Glyphen (der linke Rand der Linie) wird am aktuellen Punkt des Cairo-Kontexts gezeichnet.

EINGABEN

`line`	ein Pango-Layout-Zeilenobjekt

5.102 ccontext:ShowPage

BEZEICHNUNG

ccontext:ShowPage – Zeigt die Seite an

ÜBERSICHT

```
ccontext:ShowPage()
```

BESCHREIBUNG

Gibt die aktuelle Seite für Backends aus, die mehrere Seiten unterstützen, und löscht sie. Verwenden Sie ccontext:CopyPage(), wenn Sie die Seite nicht löschen möchten.

Dies ist eine praktische Funktion, die einfach csurface:ShowPage() auf dem Ziel von `cr` aufruft.

EINGABEN

Keine

5.103 ccontext:ShowText

BEZEICHNUNG

ccontext:ShowText – Zeigt den Text an

ÜBERSICHT

```
ccontext:ShowText(s$)
```

BESCHREIBUNG

Ein Zeichenoperator, der die Form aus der in `s$` übergebenen Zeichenkette generiert und entsprechend der aktuellen Schriftart, Schriftgröße (Schriftmatrix) und Schriftartoptionen gerendert wird.

Diese Funktion berechnet zunächst eine Reihe von Glyphen für die TextZeichenkette. Die erste Glyphe wird so platziert, dass ihr Ursprung am aktuellen Punkt liegt. Der Ursprung jedes nachfolgenden Glyphen wird gegenüber dem des vorherigen Glyphen um die Vorrückungswerte des vorherigen Glyphen versetzt.

Nach diesem Aufruf wird der aktuelle Punkt zum Ursprung verschoben, an dem die nächste Glyphe in derselben Reihenfolge platziert werden würde. Das heißt, der aktuelle Punkt liegt am Ursprung der endgültigen Glyphe, versetzt um seine Vorwärtswerte. Dies ermöglicht die einfache Anzeige einer einzelnen logischen Zeichenkette mit mehreren Aufrufen von ccontext:ShowText().

Hinweis: Der Funktionsaufruf ccontext:ShowText() ist Teil dessen, was die Cairoer Designer als "Toy"-Text-API bezeichnen. Es ist praktisch für kurze Demos und einfache Programme, wird aber voraussichtlich nicht für ernsthafte Textanwendungen geeignet sein. Siehe ccontext:ShowGlyphs() für die "echte" Textanzeige-API in Cairo.

EINGABEN

`s$`	Zeichenkette, die angezeigt werden soll

5.104 ccontext:Status

BEZEICHNUNG

ccontext:Status – Gibt den Kontextstatus zurück

ÜBERSICHT

```
status = ccontext:Status()
```

BESCHREIBUNG

Prüft, ob für diesen Kontext zuvor ein Fehler aufgetreten ist. Die folgenden Statuscodes sind derzeit definiert:

`#CAIRO_STATUS_SUCCESS`
: Es ist kein Fehler aufgetreten.

`#CAIRO_STATUS_NO_MEMORY`
: Kein Speicher mehr.

`#CAIRO_STATUS_INVALID_RESTORE`
: ccontext:Restore() wird ohne Übereinstimmung mit ccontext:Save() aufgerufen.

`#CAIRO_STATUS_INVALID_POP_GROUP`
: Keine gespeicherte Gruppe zum Pop, d.h. ccontext:PopGroup() ohne passende ccontext:PushGroup().

`#CAIRO_STATUS_NO_CURRENT_POINT`
: Kein aktueller Punkt definiert.

`#CAIRO_STATUS_INVALID_MATRIX`
: Ungültige Matrix (nicht invertierbar).

`#CAIRO_STATUS_INVALID_STATUS`
: Ungültiger Wert für einen Eingabe-Cairo-Status.

`#CAIRO_STATUS_NULL_POINTER`
: NULL-Zeiger.

`#CAIRO_STATUS_INVALID_STRING`
: Eingabezeichenkette ist UTF-8 ungültig.

`#CAIRO_STATUS_INVALID_PATH_DATA`
: Eingabepfaddaten ungültig.

`#CAIRO_STATUS_READ_ERROR`
: Fehler beim Lesen aus dem Eingabestream.

`#CAIRO_STATUS_WRITE_ERROR`
: Fehler beim Schreiben in den Ausgabestream.

`#CAIRO_STATUS_SURFACE_FINISHED`
: Zieloberfläche ist fertig.

`#CAIRO_STATUS_SURFACE_TYPE_MISMATCH`
: Der Oberflächentyp ist für den Vorgang nicht geeignet.

`#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`
: Der Mustertyp ist für den Vorgang nicht geeignet.

`#CAIRO_STATUS_INVALID_CONTENT`
: Ungültiger Wert für einen eingegebenen Cairo-Inhalt.

`#CAIRO_STATUS_INVALID_FORMAT`
: Ungültiger Wert für ein Eingabe-Cairo-Format.

`#CAIRO_STATUS_INVALID_VISUAL`
: Ungültiger Wert für eine visuelle Eingabe.

`#CAIRO_STATUS_FILE_NOT_FOUND`
: Datei nicht gefunden.

`#CAIRO_STATUS_INVALID_DASH`
: Ungültiger Wert für eine Strichmustereinstellung.

`#CAIRO_STATUS_INVALID_DSC_COMMENT`
: Ungültiger Wert für einen DSC-Kommentar.

`#CAIRO_STATUS_INVALID_INDEX`
: Ungültiger Index an Get-Funktion übergeben.

`#CAIRO_STATUS_CLIP_NOT_REPRESENTABLE`
: Der Clipbereich kann im gewünschten Format nicht dargestellt werden.

`#CAIRO_STATUS_TEMP_FILE_ERROR`
: Fehler beim Erstellen oder Schreiben in eine temporäre Datei.

`#CAIRO_STATUS_INVALID_STRIDE`
: Ungültiger Wert für die Schrittweite.

`#CAIRO_STATUS_FONT_TYPE_MISMATCH`
: Die Schriftart ist für den Vorgang nicht geeignet.

`#CAIRO_STATUS_USER_FONT_IMMUTABLE`
: Die Benutzerschriftart ist unveränderlich.

`#CAIRO_STATUS_USER_FONT_ERROR`
: In einer Callbak-Funktion für Benutzerschriftarten ist ein Fehler aufgetreten.

`#CAIRO_STATUS_NEGATIVE_COUNT`
: Negative Zahl wird dort verwendet, wo sie nicht zulässig ist.

`#CAIRO_STATUS_INVALID_CLUSTERS`
: Eingabecluster stellen nicht das zugehörige Text- und Glyphen-Array dar.

`#CAIRO_STATUS_INVALID_SLANT`
: Ungültiger Wert für eine Eingabeneigung der Cairo-Schriftart.

`#CAIRO_STATUS_INVALID_WEIGHT`
: Ungültiger Wert für eine eingegebene Cairo-Schriftstärke.

`#CAIRO_STATUS_INVALID_SIZE`
: Ungültiger Wert (normalerweise zu groß) für die Größe der Eingabe (Oberfläche, Muster usw.).

`#CAIRO_STATUS_USER_FONT_NOT_IMPLEMENTED`
: Benutzerschriftartmethode nicht implementiert.

`#CAIRO_STATUS_DEVICE_TYPE_MISMATCH`
: Der Gerätetyp ist für den Vorgang nicht geeignet.

`#CAIRO_STATUS_DEVICE_ERROR`
: Ein Vorgang am Gerät verursachte einen nicht näher bezeichneten Fehler.

`#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION`
: Außerhalb des Anfangs-Ende-Patchpaars wurde ein Netzmusterkonstruktionsvorgang verwendet

`#CAIRO_STATUS_DEVICE_FINISHED`
: Zielgerät ist fertig

`#CAIRO_STATUS_JBIG2_GLOBAL_MISSING`
: `#CAIRO_MIME_TYPE_JBIG2_GLOBAL_ID` wurde für mindestens ein Bild verwendet, es wurde jedoch kein Bild bereitgestellt für `#CAIRO_MIME_TYPE_JBIG2_GLOBAL`.

`#CAIRO_STATUS_PNG_ERROR`
: Beim Lesen oder Schreiben in eine PNG-Datei ist in libpng ein Fehler aufgetreten.

`#CAIRO_STATUS_FREETYPE_ERROR`
: In libfreetype ist ein Fehler aufgetreten.

`#CAIRO_STATUS_WIN32_GDI_ERROR`
: In der Windows-Grafikgeräteschnittstelle ist ein Fehler aufgetreten.

`#CAIRO_STATUS_TAG_ERROR`
: Ungültiger Tag-Name, ungültige Attribute oder Verschachtelung.

`#CAIRO_STATUS_DWRITE_ERROR`
: In der Windows Direct Write API ist ein Fehler aufgetreten.

`#CAIRO_STATUS_SVG_FONT_ERROR`
: Beim Rendern von OpenType-SVG-Schriftarten ist ein Fehler aufgetreten.

EINGABEN

Keine

RÜCKGABEWERTE

`status` den aktuellen Stand dieses Kontextes

5.105 ccontext:Stroke

BEZEICHNUNG

ccontext:Stroke – Zeichnet den aktuellen Linienpfad

ÜBERSICHT

```
ccontext:Stroke()
```

BESCHREIBUNG

Ein Zeichenoperator, der den aktuellen Pfad entsprechend den aktuellen Einstellungen für Linienbreite, Linienverbindung, Linienkopf und Strichmuster zeichnet. Nach ccontext:Stroke() wird der aktuelle Pfad aus dem Cairo-Kontext gelöscht. Siehe ccontext:SetLineWidth(), ccontext:SetLineJoin(), ccontext:SetLineCap(), ccontext:SetDash() und ccontext:StrokePreserve().

Hinweis: Degenerierte Segmente und Unterpfade werden speziell behandelt und liefern ein brauchbares Ergebnis. Dies kann zu zwei unterschiedlichen Situationen führen:

1. "Ein"-Segmente mit Nulllänge, festgelegt in ccontext:SetDash(). Wenn der Zeilenkopfstil `#CAIRO_LINE_CAP_ROUND` oder `#CAIRO_LINE_CAP_SQUARE` ist, werden diese Segmente als kreisförmige Punkte bzw. Quadrate gezeichnet. Im Fall von `#CAIRO_LINE_CAP_SQUARE` wird die Ausrichtung der Quadrate durch die Richtung des zugrunde liegenden Pfades bestimmt.
2. Ein durch ccontext:MoveTo() erstellter Unterpfad, gefolgt von einem ccontext:ClosePath() oder einem oder mehreren Aufrufen von ccontext:LineTo() zur gleichen Koordinate wie das ccontext:MoveTo(). Wenn der Zeilekopfstil `#CAIRO_LINE_CAP_ROUND` ist, werden diese Unterpfade als kreisförmige Punkte gezeichnet. Beachten Sie, dass im Fall von `#CAIRO_LINE_CAP_SQUARE` ein degenerierter Unterpfad überhaupt nicht gezeichnet wird (da die korrekte Ausrichtung unbestimmt ist).

In keinem Fall führt der Zeilenkopfstil `#CAIRO_LINE_CAP_BUTT` dazu, dass irgendetwas gezeichnet wird, weder bei degenerierten Segmenten noch bei Unterpfaden.

EINGABEN

Keine

5.106 ccontext:StrokeExtents

BEZEICHNUNG

ccontext:StrokeExtents – Gibt die Strichausmaße zurück

ÜBERSICHT

```
x1, y1, x2, y2 = ccontext:StrokeExtents()
```

BESCHREIBUNG

Berechnet einen Begrenzungsrahmen in Benutzerkoordinaten, der den Bereich abdeckt, der durch eine ccontext:Stroke()-Operation unter Berücksichtigung der aktuellen Pfad- und Strichmusterparameter betroffen wäre, den "eingefärbten" Bereich. Wenn der aktuelle Pfad leer ist, wird ein leeres Rechteck ((0,0), (0,0)) zurückgegeben. Flächenmaße und Zuschnitt werden nicht berücksichtigt.

Beachten Sie, dass ccontext:StrokeExtents() ein leeres Rechteck zurückgibt, wenn die Linienbreite genau auf Null gesetzt ist. Im Gegensatz dazu kann ccontext:PathExtents() zur Berechnung der nicht leeren Grenzen verwendet werden, wenn sich die Linienbreite Null nähert.

Beachten Sie, dass ccontext:StrokeExtents() zwangsläufig mehr Arbeit leisten muss, um die genauen eingefärbten Bereiche im Hinblick auf die Strichmusterparameter

zu berechnen, sodass ccontext:PathExtents() aus Leistungsgründen möglicherweise wünschenswerter ist, wenn nicht Eingefärbte Pfadlängen erwünscht sind.

Siehe ccontext:Stroke(), ccontext:SetLineWidth(), ccontext:SetLineJoin(), ccontext:SetLineCap(), ccontext:SetDash() und ccontext:StrokePreserve().

EINGABEN

Keine

RÜCKGABEWERTE

`x1`	links von den resultierenden Ausmaßen
`y1`	oberhalb der resultierenden Ausmaßen
`x2`	rechts von den resultierenden Ausmaßen
`y2`	unterhalb der resultierenden Ausmaßen

5.107 ccontext:StrokePreserve

BEZEICHNUNG

ccontext:StrokePreserve – Zeichnet den Strichpfad und behält den Pfad bei

ÜBERSICHT

```
ccontext:StrokePreserve()
```

BESCHREIBUNG

Ein Zeichenoperator, der den aktuellen Pfad entsprechend den aktuellen Einstellungen für Linienbreite, Linienverbindung, Linienkopf und Strichmuster zeichnet. Im Gegensatz zu ccontext:Stroke() behält ccontext:StrokePreserve() den Pfad innerhalb des Cairo-Kontexts bei.

Siehe ccontext:SetLineWidth(), ccontext:SetLineJoin(), ccontext:SetLineCap() und ccontext:SetDash().

EINGABEN

Keine

5.108 ccontext:TagBegin

BEZEICHNUNG

ccontext:TagBegin – Markiert den Anfang eines Tags

ÜBERSICHT

```
ccontext:TagBegin(tag_name$, attributes$)
```

BESCHREIBUNG

Markiert den Anfang der Struktur `tag_name$`. Rufen Sie ccontext:TagEnd() mit demselben `tag_name$` auf, um das Ende der Struktur zu markieren. Der Parameter `tag_name$` kann eine der folgenden Konstanten sein:

`#CAIRO_TAG_DEST`
: Erstellt ein Ziel für einen Hyperlink.

`#CAIRO_TAG_LINK`
: Hyperlink erstellen.

Die Zeichenkette `attributes$` hat die Form "key1=value2 key2=value2 ...". Die Werte können boolesche Werte (wahr/falsch oder 1/0), Ganzzahlen, Gleitkommazahlen, Zeichenketten oder ein Array sein.

Zeichenkettenwerte werden in einfache Anführungszeichen (') eingeschlossen. Einfache Anführungszeichen und Backslashes (Rückstrich: \) innerhalb der Zeichenkette sollten mit einem Backslash (Rückstrich: \) maskiert werden.

Boolesche Werte können nur durch Angabe des Schlüssels auf "true" gesetzt werden. Beispielsweise ist die AttributZeichenkette "key" das Äquivalent zu "key=true".

Arrays werden in "[]" eingeschlossen. z.B. "rect=[1.2 4.3 2.0 3.0]".

Wenn keine Attribute erforderlich sind, kann `attributes$` eine leere Zeichenkette sein.

Die Liste der Tags und Attribute finden Sie in der Beschreibung der Tags und Links im Cairo-Handbuch.

Eine ungültige Verschachtelung von Tags oder ungültige Attribute führen dazu, dass der Kontext mit dem Status `#CAIRO_STATUS_TAG_ERROR` heruntergefahren wird.

Siehe ccontext:TagEnd().

EINGABEN

`tag_name$`
: Tag-Name (siehe oben).

`attributes$`
: Tag-Attribute (siehe oben).

BEISPIEL

```
cr:TagBegin(#CAIRO_TAG_LINK, "dest='mydest' internal")
cr:MoveTo(50, 50)
cr:ShowText("This is a hyperlink.")
cr:TagEnd(#CAIRO_TAG_LINK)
```

Der obige Code erstellt einen Hyperlink.

5.109 ccontext:TagEnd

BEZEICHNUNG

ccontext:TagEnd – Markiert das Ende eines Tags

ÜBERSICHT

```
ccontext:TagEnd(tag_name$)
```

BESCHREIBUNG

Markiert das Ende der Struktur `tag_name$`. Eine Liste der Tag-Namen finden Sie unter ccontext:TagBegin().

Eine ungültige Verschachtelung von Tags führt dazu, dass der Kontext mit dem Status `#CAIRO_STATUS_TAG_ERROR` heruntergefahren wird.

EINGABEN

`tag_name$`
: Tag-Name

5.110 ccontext:TextExtents

BEZEICHNUNG

ccontext:TextExtents – Gibt die Textausdehnungen zurück

ÜBERSICHT

```
t = ccontext:TextExtents(s$)
```

BESCHREIBUNG

Ruft die Grenzen für eine TextZeichenkette ab. Die Grenzen beschreiben ein Rechteck im Benutzerbereich, das den "eingefärbten" Teil des Texts umschließt (wie er von ccontext:ShowText() gezeichnet würde). Darüber hinaus geben die Werte `XAdvance` und `YAdvance` den Betrag an, um den der aktuelle Punkt durch ccontext:ShowText() vorgezogen würde.

Diese Funktion gibt eine Tabelle zurück, die die Textbereiche beschreibt. Die Tabelle enthält die folgenden Felder:

`XBearing` Der horizontale Abstand vom Ursprung zum ganz linken Teil der gezeichneten Glyphen. Positiv, wenn die Glyphen vollständig rechts vom Ursprung liegen.

`YBearing` Der vertikale Abstand vom Ursprung bis zum obersten Teil der gezeichneten Glyphen. Nur positiv, wenn die Glyphen vollständig unterhalb des Ursprungs liegen; wird normalerweise negativ sein.

`Width` Breite der Glyphen die gezeichnet wird.

`Height` Höhe der Glyphen die gezeichnet wird.

`XAdvance` Abstand, der nach dem Zeichnen dieser Glyphen in X-Richtung vorgeschoben werden soll.

`YAdvance` Entfernung, die nach dem Zeichnen dieser Glyphen in Y-Richtung vorgeschoben werden soll. Wird normalerweise Null sein, mit Ausnahme des vertikalen Textlayouts, wie es in ostasiatischen Sprachen vorkommt.

Beachten Sie, dass Leerzeichen keinen direkten Einfluss auf die Größe des Rechtecks haben (`Width` und `Height`). Sie tragen indirekt dazu bei, indem sie die Position von Nicht-Leerzeichen ändern. Insbesondere nachgestellte Leerzeichen haben wahrscheinlich keinen Einfluss auf die Größe des Rechtecks, wohl aber auf die Werte `XAdvance` und `YAdvance`.

Beachten Sie außerdem, dass Textausdehnungen größtenteils, aber nicht vollständig, unabhängig von der aktuellen Transformationsmatrix sind, da sie in Benutzerbereichkoordinaten angegeben werden. Wenn Sie ccontext:Scale() mit Skalierungskoeffizienten von 2.0 auf jeder Achse aufrufen, wird der Text doppelt so groß gezeichnet, aber die gemeldeten Textausdehnungen werden nicht verdoppelt. Sie ändern sich aufgrund von Hinweisen

geringfügig (Sie können also nicht davon ausgehen, dass die Metriken unabhängig von der Transformationsmatrix sind), bleiben ansonsten aber unverändert.

EINGABEN

`s$` Textfolge

RÜCKGABEWERTE

`t` Tabelle mit den Textumfängen (siehe oben).

5.111 ccontext:TextPath

BEZEICHNUNG

ccontext:TextPath – Fügt einen Text zum Pfad hinzu

ÜBERSICHT

```
ccontext:TextPath(s$)
```

BESCHREIBUNG

Fügt geschlossene Pfade für Text zum aktuellen Pfad hinzu. Wenn der generierte Pfad gefüllt ist, erzielt er einen ähnlichen Effekt wie ccontext:ShowText().

Die Textkonvertierung und -positionierung erfolgt ähnlich wie bei ccontext:ShowText().

Wie bei ccontext:ShowText() wird nach diesem Aufruf der aktuelle Punkt an den Ursprung verschoben, an dem das nächste Glyph in derselben Reihenfolge platziert werden würde. Das heißt, der aktuelle Punkt befindet sich am Ursprung des endgültigen Glyphen, versetzt um seine Vorwärtswerte. Dies ermöglicht die Verkettung mehrerer Aufrufe von ccontext:TextPath(), ohne dazwischen den aktuellen Punkt festlegen zu müssen.

Hinweis: Der Funktionsaufruf ccontext:TextPath() ist Teil dessen, was die Cairoer Designer als "Toy"-Text-API bezeichnen. Es ist praktisch für kurze Demos und einfache Programme, wird aber voraussichtlich nicht für ernsthafte Textanwendungen geeignet sein. Siehe ccontext:GlyphPath() für die "echte" Textpfad-API in Cairo.

EINGABEN

`s$` Textfolge

5.112 ccontext:Transform

BEZEICHNUNG

ccontext:Transform – Ändert die aktuelle Transformationsmatrix

ÜBERSICHT

```
ccontext:Transform(matrix)
```

BESCHREIBUNG

Ändert die aktuelle Transformationsmatrix (CTM), indem `matrix` als zusätzliche Transformation angewendet wird. Die neue Transformation des Benutzerbereichs erfolgt nach jeder bestehenden Transformation.

EINGABEN

`matrix` eine Transformationsmatrix, die auf die Benutzerbereichachsen angewendet werden soll

5.113 ccontext:Translate

BEZEICHNUNG

ccontext:Translate – Verschiebt die Transformationsmatrix

ÜBERSICHT

```
ccontext:Translate(tx, ty)
```

BESCHREIBUNG

Ändert die aktuelle Transformationsmatrix (CTM), indem der Ursprung des Benutzerbereichs um (`tx`, `ty`) verschoben wird. Dieser Offset wird als Benutzerbereich-Koordinate entsprechend dem CTM interpretiert, der vor dem neuen Aufruf von ccontext:Translate() vorhanden war. Mit anderen Worten: Die Verschiebung des Ursprung-Benutzerbereich erfolgt nach einer eventuell vorhandenen Transformation.

EINGABEN

`tx` Betrag, der in X-Richtung verschoben werden soll

`ty` Betrag, der in Y-Richtung verschoben werden soll

5.114 ccontext:UpdateLayout

BEZEICHNUNG

ccontext:UpdateLayout – Aktualisiert das Pango-Layout

ÜBERSICHT

```
ccontext:UpdateLayout(layout)
```

BESCHREIBUNG

Aktualisiert den privaten Pango-Kontext eines Pango-Layouts, das mit ccontext:PangoLayout() erstellt wurde, damit er mit der aktuellen Transformation und Zieloberfläche eines Cairo-Kontexts übereinstimmt.

EINGABEN

`layout` ein Pango-Layout von ccontext:PangoLayout()

5.115 ccontext:UserToDevice

BEZEICHNUNG

ccontext:UserToDevice – Transformiert den Benutzer- in den Gerätebereich

ÜBERSICHT

```
dx, dy = ccontext:UserToDevice(ux, uy)
```

BESCHREIBUNG

Transformiert eine Koordinate vom Benutzerbereich in den Gerätebereich, indem der angegebenen Punkt mit der aktuellen Transformationsmatrix (CTM) multipliziert wird.

EINGABEN

`ux` X-Wert der Koordinate (Benutzerbereich)

`uy` Y-Wert der Koordinate (Benutzerbereich)

RÜCKGABEWERTE

`dx` X-Wert der Koordinate (Gerätebereich)

`dy` Y-Wert der Koordinate (Gerätebereich)

5.116 ccontext:UserToDeviceDistance

BEZEICHNUNG

ccontext:UserToDeviceDistance – Transformiert einen Distanzvektor

ÜBERSICHT

```
dx, dy = ccontext:UserToDeviceDistance(ux, uy)
```

BESCHREIBUNG

Transformiert einen Distanzvektor vom Benutzerbereich in den Gerätebereich. Diese Funktion ähnelt ccontext:UserToDevice(), außer dass die Verschiebungskomponenten (`dx`, `dy`) des CTM bei der Transformation ignoriert werden.

EINGABEN

`ux` X-Komponente eines Distanzvektors (Benutzerbereich)

`uy` Y-Komponente eines Distanzvektors (Benutzerbereich)

RÜCKGABEWERTE

`dx` X-Komponente eines Distanzvektors (Gerätebereich)

`dy` Y-Komponente eines Distanzvektors (Gerätebereich)

6 Cairo-Schriftart

6.1 cfontface:Free

BEZEICHNUNG
cfontface:Free – Löscht eine Schriftart

ÜBERSICHT
`cfontface:Free()`

BESCHREIBUNG
Verringert die Referenzanzahl auf der Schriftart um eins. Wenn das Ergebnis Null ist, wird die Schriftart und alle zugehörigen Ressourcen freigegeben. Siehe cfontface:Reference().

EINGABEN
Keine

6.2 cfontface:GetFamily

BEZEICHNUNG
cfontface:GetFamily – Gibt den Familiennamen zurück

ÜBERSICHT
`f$ = cfontface:GetFamily()`

BESCHREIBUNG
Ruft den Familiennamen einer Toy-Schriftart ab.

EINGABEN
Keine

RÜCKGABEWERTE
`f$` der Familienname der Schriftart

6.3 cfontface:GetReferenceCount

BEZEICHNUNG
cfontface:GetReferenceCount – Gibt die Referenzanzahl zurück

ÜBERSICHT
`count = cfontface:GetReferenceCount()`

BESCHREIBUNG
Gibt den aktuellen Referenzzähler der Schriftart zurück.

EINGABEN
Keine

RÜCKGABEWERTE
`count` die aktuelle Referenzanzahl der Schriftart

6.4 cfontface:GetSlant

BEZEICHNUNG
cfontface:GetSlant – Gibt die Neigung einer Toy-Schriftart zurück

ÜBERSICHT
`slant = cfontface:GetSlant()`

BESCHREIBUNG
Ermittelt die Neigung einer Toy-Schriftart. Dies wird eine der folgenden Konstanten sein:

```
#CAIRO_FONT_SLANT_NORMAL
#CAIRO_FONT_SLANT_ITALIC
#CAIRO_FONT_SLANT_OBLIQUE
```

EINGABEN
Keine

RÜCKGABEWERTE

`slant` der Neigungswert (siehe oben).

6.5 cfontface:GetType

BEZEICHNUNG
cfontface:GetType – Gibt den Typ des Schriftart-Backend zurück

ÜBERSICHT
`type = cfontface:GetType()`

BESCHREIBUNG
Diese Funktion gibt den Typ des Backends zurück, der zum Erstellen einer Schriftart verwendet wird. Dies wird eine der folgenden Konstanten sein:

`#CAIRO_FONT_TYPE_TOY`
: Die Schriftart wurde mit der Toy-Schriftart-API von Cairo erstellt.

`#CAIRO_FONT_TYPE_FT:`
: Die Schriftart ist vom Typ FreeType.

`#CAIRO_FONT_TYPE_WIN32`
: Die Schriftart ist vom Typ Win32.

`#CAIRO_FONT_TYPE_QUARTZ`
: Die Schriftart ist vom Typ Quartz.

`#CAIRO_FONT_TYPE_USER`
: Die Schriftart wurde mit der Benutzer-Schriftarten-API von Cairo erstellt.

`#CAIRO_FONT_TYPE_DWRITE`
: Die Schriftart ist vom Typ Win32 DWrite.

EINGABEN
Keine

RÜCKGABEWERTE

`type` der Schriftarttyp (siehe oben).

6.6 cfontface:GetWeight

BEZEICHNUNG

cfontface:GetWeight – Gibt die Stärke einer Toy-Schriftart zurück

ÜBERSICHT

```
weight = cfontface:GetWeight()
```

BESCHREIBUNG

Ruft die Stärke einer Toy-Schriftart ab. Dies wird eine der folgenden Konstanten sein:

```
#CAIRO_FONT_WEIGHT_NORMAL
#CAIRO_FONT_WEIGHT_BOLD
```

EINGABEN

Keine

RÜCKGABEWERTE

`weight` der Stärkewert (siehe oben).

6.7 cfontface:IsNull

BEZEICHNUNG

cfontface:IsNull – Überprüft, ob die Schriftart ungültig ist

ÜBERSICHT

```
bool = cfontface:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Schriftart `NULL` ist, also ungültig. Wenn Funktionen, die Schriftarten zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Schriftart einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Schriftartzuordnung fehlgeschlagen ist. In diesem Fall ist die Schriftart `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn die Schriftart `NULL` ist, andernfalls `False`

6.8 cfontface:Reference

BEZEICHNUNG

cfontface:Reference – Erhöht die Referenzanzahl der Schriftart

ÜBERSICHT

```
cfontface:Reference()
```

BESCHREIBUNG

Erhöht die Referenzanzahl auf der Schriftart um eins. Dadurch wird verhindert, dass die Schriftart gelöscht wird, bis ein entsprechender Aufruf von cfontface:Free() erfolgt.

Verwenden Sie cfontface:GetReferenceCount(), um die Anzahl der Verweise auf eine Cairo-Schriftart zu ermitteln.

EINGABEN

Keine

6.9 cfontface:Status

BEZEICHNUNG

cfontface:Status – Ruft den Status der Schriftart ab

ÜBERSICHT

```
status = cfontface:Status()
```

BESCHREIBUNG

Überprüft, ob für diese Schriftart zuvor ein Fehler aufgetreten ist.

Diese Funktion gibt `#CAIRO_STATUS_SUCCESS` oder einen Fehler wie `#CAIRO_STATUS_NO_MEMORY` zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`status` ein Cairo-Statuscode

7 Schriftartenoptionen für Cairo

7.1 cfontoptions:Copy

BEZEICHNUNG
cfontoptions:Copy – Kopiert das Schriftartoptionsobjekt

ÜBERSICHT
`handle = cfontoptions:Copy()`

BESCHREIBUNG
Ordnet ein neues Schriftartoptionsobjekt zu, das die Optionswerte vom Original kopiert.

Diese Funktion gibt ein neu zugewiesenes Cairo-Schriftartoptionsobjekt zurück. Die Funktion cfontoptions:Free() gibt immer ein gültiges Handle zurück; Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Fehlerobjekt zurückgegeben, bei dem alle Operationen auf dem Objekt nichts bewirken. Sie können dies mit cfontoptions:Status() überprüfen.

EINGABEN
Keine

RÜCKGABEWERTE

`handle` ein neu zugewiesenes Cairo-Schriftartoptionsobjekt

7.2 cfontoptions:Equal

BEZEICHNUNG
cfontoptions:Equal – Prüft auf Gleichheit

ÜBERSICHT
`ok = cfontoptions:Equal(other)`

BESCHREIBUNG
Vergleicht zwei Schriftartoptionsobjekte auf Gleichheit.

Diese Funktion gibt `True` zurück, wenn alle Felder der beiden Schriftartoptionsobjekte übereinstimmen. Beachten Sie, dass diese Funktion `False` zurückgibt, wenn eines der Objekte fehlerhaft ist.

EINGABEN

`other` ein weiteres Cairo-Schriftartoptionsobjekt

RÜCKGABEWERTE

`ok` `True`, wenn alle Felder der beiden Schriftartoptionsobjekte übereinstimmen

7.3 cfontoptions:Free

BEZEICHNUNG
cfontoptions:Free – Löscht ein Schriftartoptionsobjekt

ÜBERSICHT
`cfontoptions:Free()`

BESCHREIBUNG
Löscht ein Kairo-Schriftartoptionsobjekt, das mit cairo.FontOptions() oder cfontoptions:Copy() erstellt wurde.

EINGABEN
Keine

7.4 cfontoptions:GetAntialias

BEZEICHNUNG
cfontoptions:GetAntialias – Gibt den Antialias-Modus zurück

ÜBERSICHT
`mode = cfontoptions:GetAntialias()`

BESCHREIBUNG
Ruft den Antialiasing-Modus für das Schriftartoptionsobjekt ab. Eine Liste der Antialiasing-Modi finden Sie unter ccontext:SetAntialias().

EINGABEN
Keine

RÜCKGABEWERTE
`mode` der Antialiasing-Modus

7.5 cfontoptions:GetHintMetrics

BEZEICHNUNG
cfontoptions:GetHintMetrics – Gibt die Hinweismetriken zurück

ÜBERSICHT
`metrics = cfontoptions:GetHintMetrics()`

BESCHREIBUNG
Ruft den Metrik-Hinweismodus für das Schriftartoptionsobjekt ab. Eine Liste der Hinweismetriken finden Sie unter cfontoptions:SetHintMetrics().

EINGABEN
Keine

RÜCKGABEWERTE
`metrics` der Metrik-Hinweismodus für das Schriftartoptionsobjekt

7.6 cfontoptions:GetHintStyle

BEZEICHNUNG
cfontoptions:GetHintStyle – Ruft den Hinweisstil ab

ÜBERSICHT
```
style = cfontoptions:GetHintStyle()
```

BESCHREIBUNG
Ruft den Hinweisstil für Schriftartkonturen für das Schriftartoptionsobjekt ab. Eine Liste der Hinweisstile finden Sie unter cfontoptions:SetHintStyle().

EINGABEN
Keine

RÜCKGABEWERTE

`style`	der Hinweisstil für das Schriftartoptionsobjekt

7.7 cfontoptions:GetSubpixelOrder

BEZEICHNUNG
cfontoptions:GetSubpixelOrder – Ruft die Subpixel-Reihenfolge ab

ÜBERSICHT
```
order = cfontoptions:GetSubpixelOrder()
```

BESCHREIBUNG
Ruft die Subpixelreihenfolge für das Schriftartoptionsobjekt ab. Eine Liste der Subpixel-Reihenfolgen finden Sie unter cfontoptions:SetSubpixelOrder().

EINGABEN
Keine

RÜCKGABEWERTE

`order`	die Subpixelreihenfolge für das Schriftartoptionsobjekt

7.8 cfontoptions:GetVariations

BEZEICHNUNG
cfontoptions:GetVariations – Ruft die Schriftartvariationen ab

ÜBERSICHT
```
v$ = cfontoptions:GetVariations()
```

BESCHREIBUNG
Ruft die OpenType-Schriftartvariationen für das Schriftartoptionsobjekt ab. Weitere Informationen zum Zeichenkettenformat finden Sie unter cfontoptions:SetVariations().

EINGABEN
Keine

RÜCKGABEWERTE

`v$` die Schriftartvariationen für das Schriftartoptionsobjekt

7.9 cfontoptions:Hash

BEZEICHNUNG

cfontoptions:Hash – Berechnet den Objekt-Hash

ÜBERSICHT

```
v = cfontoptions:Hash()
```

BESCHREIBUNG

Berechnet einen Hash für das Schriftartoptionsobjekt. Dieser Wert ist nützlich, wenn ein Objekt, das ein Cairo-Schriftartoptionsobjekt enthält, in einer Hash-Tabelle gespeichert wird.

Diese Funktion gibt den Hashwert für das Schriftartoptionsobjekt zurück. Der Rückgabewert kann in einen 32-Bit-Typ umgewandelt werden, wenn ein 32-Bit-Hash-Wert benötigt wird.

EINGABEN

Keine

RÜCKGABEWERTE

`v` der Hashwert für das Schriftartoptionsobjekt

7.10 cfontoptions:IsNull

BEZEICHNUNG

cfontoptions:IsNull – Überprüft, ob das Schriftartoptionsobjekt ungültig ist

ÜBERSICHT

```
bool = cfontoptions:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Schriftartoptionsobjekt `NULL`, also ungültig, ist. Wenn Funktionen, die Objekte zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Objektzuordnung fehlgeschlagen ist. In diesem Fall ist das Objekt `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Objekt `NULL` ist, andernfalls `False`

7.11 cfontoptions:Merge

BEZEICHNUNG

cfontoptions:Merge – Führt die Schriftartoptionen zusammen

ÜBERSICHT

```
cfontoptions:Merge(other)
```

BESCHREIBUNG

Führt nicht standardmäßige Optionen von `other` in das Schriftartoptionsobjekt ein und ersetzt vorhandene Werte. Man kann sich diesen Vorgang als etwas Ähnliches vorstellen, als würde man `other` mit dem Vorgang `#CAIRO_OPERATOR_OVER` auf dem Schriftartoptionsobjekt zusammensetzen.

EINGABEN

`other` ein weiteres Cairo-Schriftartoptionsobjekt

7.12 cfontoptions:SetAntialias

BEZEICHNUNG

cfontoptions:SetAntialias – Stellt den Antialias-Modus ein

ÜBERSICHT

```
cfontoptions:SetAntialias(antialias)
```

BESCHREIBUNG

Legt den Antialiasing-Modus für das Schriftartoptionsobjekt fest. Dies gibt die Art des Antialiasings an, das beim Rendern von Text ausgeführt werden soll. Eine Liste der Antialiasing-Modi finden Sie unter ccontext:SetAntialias().

EINGABEN

`antialias`
der neue Antialiasing-Modus

7.13 cfontoptions:SetHintMetrics

BEZEICHNUNG

cfontoptions:SetHintMetrics – Legt die Hinweismetriken fest

ÜBERSICHT

```
cfontoptions:SetHintMetrics(hint_metrics)
```

BESCHREIBUNG

Legt den Metrik-Hinweismodus für das Schriftartoptionsobjekt fest. Dies steuert, ob Metriken in Ganzzahlwerte in Geräteeinheiten quantisiert werden.

Die folgenden Konstanten können im Argument `hint_metrics` übergeben werden:

`#CAIRO_HINT_METRICS_DEFAULT`
Hinweismetriken in der Standardmethode für das Schriftarten-Backend und das Zielgerät.

`#CAIRO_HINT_METRICS_OFF`
: Es gibt keine Hinweise auf Schriftmetriken.

`#CAIRO_HINT_METRICS_ON`
: Hinweis-Font-Metriken.

EINGABEN

`hint_metrics`
: der neue Metrik-Hinweismodus

7.14 cfontoptions:SetHintStyle

BEZEICHNUNG

cfontoptions:SetHintStyle – Legt den Hinweisstil fest

ÜBERSICHT

`cfontoptions:SetHintStyle(hint_style)`

BESCHREIBUNG

Legt den Hinweisstil für Schriftartumrisse für das Schriftartoptionsobjekt fest. Dies steuert, ob Schriftumrisse an das Pixelraster angepasst werden sollen und, wenn ja, ob die Wiedergabetreue oder der Kontrast optimiert werden sollen.

Die folgenden Konstanten können im Argument `hint_style` übergeben werden:

`#CAIRO_HINT_STYLE_DEFAULT`
: Verwendet den Standard-Hinweisstil für das Schriftart-Backend und das Zielgerät.

`#CAIRO_HINT_STYLE_NONE`
: Deutet Umrisse nicht an.

`#CAIRO_HINT_STYLE_SLIGHT`
: Deutet die Umrisse leicht an, um den Kontrast zu verbessern und gleichzeitig eine gute Wiedergabetreue zu den Originalformen beizubehalten.

`#CAIRO_HINT_STYLE_MEDIUM`
: Andeutung von Konturen mit mittlerer Stärke, die einen Kompromiss zwischen der Treue zu den ursprünglichen Formen und dem Kontrast bieten.

`#CAIRO_HINT_STYLE_FULL`
: Deutet den Umrisse an, um den Kontrast zu maximieren.

EINGABEN

`hint_style`
: der neue Hinweisstil

7.15 cfontoptions:SetSubpixelOrder

BEZEICHNUNG

cfontoptions:SetSubpixelOrder – Legt die Subpixelreihenfolge fest

ÜBERSICHT

```
cfontoptions:SetSubpixelOrder(subpixel_order)
```

BESCHREIBUNG

Legt die Subpixelreihenfolge für das Schriftartoptionsobjekt fest. Die Subpixelreihenfolge gibt die Reihenfolge der Farbelemente innerhalb jedes Pixels auf dem Anzeigegerät an, wenn mit dem Antialiasing-Modus `#CAIRO_ANTIALIAS_SUBPIXEL` gerendert wird.

Die folgenden Konstanten können im Argument `subpixel_order` übergeben werden:

`#CAIRO_SUBPIXEL_ORDER_DEFAULT`
: Verwendet die Standard-Subpixelreihenfolge für das Zielgerät.

`#CAIRO_SUBPIXEL_ORDER_RGB`
: Subpixelelemente sind horizontal angeordnet, mit Rot auf der linken Seite.

`#CAIRO_SUBPIXEL_ORDER_BGR`
: Subpixel-Elemente sind horizontal angeordnet, mit Blau auf der linken Seite.

`#CAIRO_SUBPIXEL_ORDER_VRGB`
: Subpixel-Elemente sind vertikal angeordnet, wobei die Farbe oben Rot ist

`#CAIRO_SUBPIXEL_ORDER_VBGR`
: Subpixel-Elemente sind vertikal angeordnet, wobei die Farbe oben Blau ist.

EINGABEN

`subpixel_order`
: die neue Subpixelreihenfolge

7.16 cfontoptions:SetVariations

BEZEICHNUNG

cfontoptions:SetVariations – Legt die Varianten fest

ÜBERSICHT

```
cfontoptions:SetVariations(variations$)
```

BESCHREIBUNG

Legt die OpenType-Schriftartvarianten für das Schriftartoptionsobjekt fest. Schriftartvarianten werden als Zeichenkette mit einem Format angegeben, das den CSS-Schriftartvarianteneinstellungen ähnelt. Die Zeichenkette enthält eine durch Kommas getrennte Liste von Achsenzuweisungen, wobei jede Zuweisung aus einem vierstelligen Achsennamen und einem Wert besteht, getrennt durch Leerzeichen und optionales Gleichheitszeichen.

Beispiele:

```
wght=200,wdth=140.5
wght 200 , wdth 140.5
```

EINGABEN

`variations$`
: die neuen Schriftvarianten

7.17 cfontoptions:Status

BEZEICHNUNG

cfontoptions:Status – Gibt den Status der Schriftartoptionen zurück

ÜBERSICHT

```
status = cfontoptions:Status()
```

BESCHREIBUNG

Überprüft, ob für dieses Schriftartoptionsobjekt zuvor ein Fehler aufgetreten ist.

Diese Funktion gibt `#CAIRO_STATUS_SUCCESS`, `#CAIRO_STATUS_NO_MEMORY` oder `#CAIRO_STATUS_NULL_POINTER` zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`status` ein Cairo-Statuscode (siehe oben).

8 Cairo-Glyphen

8.1 cglyphs:Free

BEZEICHNUNG
cglyphs:Free – Löscht ein Glyphen-Array

ÜBERSICHT
`cglyphs:Free()`

BESCHREIBUNG
Löscht ein von cairo.Glyphs() zugewiesenes Glyphen-Array.

EINGABEN
Keine

8.2 cglyphs:Get

BEZEICHNUNG
cglyphs:Get – Gibt den Glyphenindex und den Versatz zurück

ÜBERSICHT
`index, x, y = cglyphs:Get(n)`

BESCHREIBUNG
Gibt Informationen über die Glyphe am Index `n` im Glyphen-Array zurück. Der Rückgabewert `index` gibt den Glyphenindex in der Schriftart an. Die genaue Interpretation des Glyphenindex hängt von der verwendeten Schriftartentechnologie ab. `x` und `y` geben den Versatz in X- und Y-Richtung zwischen dem Ursprung, der zum Zeichnen oder Messen der Zeichenkette verwendet wird, und dem Ursprung dieser Glyphe an.

EINGABEN

`n`	welche Glyphe man bekommen soll

RÜCKGABEWERTE

`index`	glyph ID
`x`	x-Versatz der Glyphe
`y`	y-Versatz der Glyphe

8.3 cglyphs:Set

BEZEICHNUNG
cglyphs:Set – Legt den Glyphenindex und den Versatz fest

ÜBERSICHT
`cglyphs:Set(n, index, x, y)`

BESCHREIBUNG

Legt Informationen über die Glyphe am Index `n` im Glyphen-Array fest. `index` gibt den Glyphenindex in der Schriftart an. Die genaue Interpretation des Glyphenindex hängt von der verwendeten Schriftartentechnologie ab. `x` und `y` geben den Versatz in X- und Y-Richtung zwischen dem Ursprung, der zum Zeichnen oder Messen der Zeichenkette verwendet wird, und dem Ursprung dieser Glyphe an.

EINGABEN

`n`	Glyphe, die festgelegt werden soll
`index`	Glyphen ID
`x`	x-Versatz der Glyphe
`y`	y-Versatz der Glyphe

9 Cairo-Matrix

9.1 cmatrix:Get

BEZEICHNUNG

cmatrix:Get – Gibt eine affine Matrixtransformation zurück

ÜBERSICHT

```
xx, yx, xy, yy, x0, y0 = cmatrix:Get()
```

BESCHREIBUNG

Ruft die affine Transformation aus der Matrix ab und gibt sie zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`xx`	xx-Komponente der affinen Transformation
`yx`	yx-Komponente der affinen Transformation
`xy`	xy-Komponente der affinen Transformation
`yy`	yy-Komponente der affinen Transformation
`x0`	X-Translationskomponente der affinen Transformation
`y0`	Y-Translationskomponente der affinen Transformation

9.2 cmatrix:Init

BEZEICHNUNG

cmatrix:Init – Initialisiert die Matrix

ÜBERSICHT

```
cmatrix:Init(xx, yx, xy, yy, x0, y0)
```

BESCHREIBUNG

Setzt `matrix` als die durch `xx`, `yx`, `xy`, `yy`, `x0`, `y0` gegebene affine Transformation. Die Transformation ist gegeben durch:

```
x_new = xx * x + xy * y + x0;
y_new = yx * x + yy * y + y0;
```

EINGABEN

`xx`	xx-Komponente der affinen Transformation
`yx`	yx-Komponente der affinen Transformation
`xy`	xy-Komponente der affinen Transformation
`yy`	yy-Komponente der affinen Transformation
`x0`	X-Translationskomponente der affinen Transformation
`y0`	Y-Translationskomponente der affinen Transformation

9.3 cmatrix:InitIdentity

BEZEICHNUNG
cmatrix:InitIdentity – Initialisiert die Identität

ÜBERSICHT
`cmatrix:InitIdentity()`

BESCHREIBUNG
Ändert die Matrix in eine Identitätstransformation.

EINGABEN
Keine

9.4 cmatrix:InitRotate

BEZEICHNUNG
cmatrix:InitRotate – Initialisiert die Drehrichtung

ÜBERSICHT
`cmatrix:InitRotate(radians)`

BESCHREIBUNG
Initialisiert die Matrix mit einer Transformation, die sich um `radians` dreht. Die Drehrichtung ist so definiert, dass positive Winkel in der Richtung von der positiven X-Achse zur positiven Y-Achse rotieren. Bei der standardmäßigen Achsenausrichtung von Cairo drehen sich positive Winkel im Uhrzeigersinn.

EINGABEN

`radians` Drehwinkel im Bogenmaß

9.5 cmatrix:InitScale

BEZEICHNUNG
cmatrix:InitScale – Initialisiert die Skalierung

ÜBERSICHT
`cmatrix:InitScale(sx, sy)`

BESCHREIBUNG
Initialisiert die Matrix mit einer Transformation, die in der X- bzw. Y-Dimension um `sx` und `sy` skaliert.

EINGABEN

`sx` Skalierungsfaktor in X-Richtung

`sy` Skalierungsfaktor in Y-Richtung

9.6 cmatrix:InitTranslate

BEZEICHNUNG

cmatrix:InitTranslate – Initialisiert die Verschiebung

ÜBERSICHT

```
cmatrix:InitTranslate(tx, ty)
```

BESCHREIBUNG

Initialisiert die Matrix mit einer Transformation, die durch `tx` und `ty` in die X- bzw. Y-Dimensionen übersetzt wird.

EINGABEN

`tx`	Betrag, der in X-Richtung verschoben werden soll
`ty`	Betrag, der in Y-Richtung verschoben werden soll

9.7 cmatrix:Invert

BEZEICHNUNG

cmatrix:Invert – Kehrt die Matrix um

ÜBERSICHT

```
status = cmatrix:Invert()
```

BESCHREIBUNG

Ändert die Matrix so, dass sie die Umkehrung ihres ursprünglichen Werts ist. Nicht alle Transformationsmatrizen haben Umkehrungen; Wenn die Matrix Punkte zusammenbricht (sie ist entartet), dann hat sie keine Umkehrung und diese Funktion schlägt fehl.

Wenn die Matrix eine Umkehrung hat, wird die Matrix in die Umkehrungsmatrix geändert und `#CAIRO_STATUS_SUCCESS` zurückgegeben. Andernfalls wird `#CAIRO_STATUS_INVALID_MATRIX` zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`status`	Statuscode, der Erfolg oder Fehler anzeigt (siehe oben).

9.8 cmatrix:Multiply

BEZEICHNUNG

cmatrix:Multiply – Multipliziert die Matrix

ÜBERSICHT

```
cmatrix:Multiply(a, b)
```

BESCHREIBUNG

Multipliziert die affinen Transformationen in `a` und `b` miteinander und speichert das Ergebnis in der Zielmatrix. Der Effekt der resultierenden Transformation besteht darin,

dass zunächst die Transformation in `a` auf die Koordinaten und dann die Transformation in `b` auf die Koordinaten angewendet wird.

Es ist zulässig, dass die Zielmatrix entweder mit `a` oder `b` identisch ist.

EINGABEN

`a`	eine Cairo-Matrix
`b`	eine Cairo-Matrix

9.9 cmatrix:Rotate

BEZEICHNUNG

cmatrix:Rotate – Dreht die Matrix

ÜBERSICHT

```
cmatrix:Rotate(radians)
```

BESCHREIBUNG

Wendet eine Drehung um `radians` auf die Transformation in der Matrix an. Der Effekt der neuen Transformation besteht darin, dass zunächst die Koordinaten um `radians` gedreht werden und dann die ursprüngliche Transformation auf die Koordinaten angewendet wird.

Die Drehrichtung ist so definiert, dass positive Winkel in der Richtung von der positiven X-Achse zur positiven Y-Achse rotieren. Bei der Standardachsenausrichtung von Cairo drehen sich positive Winkel im Uhrzeigersinn.

EINGABEN

`radians`	Drehwinkel im Bogenmaß

9.10 cmatrix:Scale

BEZEICHNUNG

cmatrix:Scale – Skaliert die Matrix

ÜBERSICHT

```
cmatrix:Scale(sx, sy)
```

BESCHREIBUNG

Wendet die Skalierung um `sx`, `sy` auf die Transformation in der Matrix an. Der Effekt der neuen Transformation besteht darin, dass zunächst die Koordinaten um `sx` und `sy` skaliert werden und dann die ursprüngliche Transformation auf die Koordinaten angewendet wird.

EINGABEN

`sx`	Skalierungsfaktor in X-Richtung
`sy`	Skalierungsfaktor in Y-Richtung

9.11 cmatrix:TransformDistance

BEZEICHNUNG

cmatrix:TransformDistance – Transformiert den Distanzvektor

ÜBERSICHT

```
tx, ty = cmatrix:TransformDistance(dx, dy)
```

BESCHREIBUNG

Transformiert den Distanzvektor (`dx`,`dy`) durch die Matrix. Dies ähnelt cmatrix:TransformPoint(), außer dass die Verschiebungskomponenten der Transformation ignoriert werden. Die Berechnung des zurückgegebenen Vektors ist wie folgt:

```
dx2 = dx1 * a + dy1 * c;
dy2 = dx1 * b + dy1 * d;
```

Affine Transformationen sind ortsinvariant, sodass derselbe Vektor immer in denselben Vektor transformiert wird. Wenn sich (`x1`,`y1`) in (`x2`,`y2`) umwandelt, dann wird (`x1`+§`dx1`,`y1`+§`dy1`) für alle Werte von `x1` und in (`x1`+§`dx2`,`y1`+§`dy2`) umgewandelt nach `x2`.

EINGABEN

`dx`	transformierte X-Komponente eines Distanzvektors
`dy`	transformierte Y-Komponente eines Distanzvektors

RÜCKGABEWERTE

`tx`	transformierte X-Komponente eines Distanzvektors
`ty`	transformierte Y-Komponente eines Distanzvektors

9.12 cmatrix:TransformPoint

BEZEICHNUNG

cmatrix:TransformPoint – Transformatiert den Punkt

ÜBERSICHT

```
tx, ty = cmatrix:TransformPoint(x, y)
```

BESCHREIBUNG

Transformiert den Punkt (`x`, `y`) durch die Matrix.

EINGABEN

`x`	X Position
`y`	Y Position

RÜCKGABEWERTE

`tx`	transformierte X-Position
`ty`	transformierte Y-Position

9.13 cmatrix:Translate

BEZEICHNUNG

cmatrix:Translate – Verschiebt die Matrix

ÜBERSICHT

```
cmatrix:Translate(tx, ty)
```

BESCHREIBUNG

Wendet eine Verschiebung um `tx`, `ty` auf die Transformation in der Matrix an. Der Effekt der neuen Transformation besteht darin, dass zunächst die Koordinaten um `tx` und `ty` verschoben werden und dann die ursprüngliche Transformation auf die Koordinaten angewendet wird.

EINGABEN

`tx`	Betrag, der in X-Richtung verschoben werden soll
`ty`	Betrag, der in Y-Richtung verschoben werden soll

10 Cairo-Pfad

10.1 cpath:Free

BEZEICHNUNG
cpath:Free – Löscht einen Pfad

ÜBERSICHT
`cpath:Free()`

BESCHREIBUNG
Löscht einen von cairo.Path() zugewiesenen Pfad.

EINGABEN
Keine

10.2 cpath:Get

BEZEICHNUNG
cpath:Get – Gibt den Pfad zurück

ÜBERSICHT
`t = cpath:Get()`

BESCHREIBUNG
Gibt die einzelnen Elemente zurück, aus denen der Pfad besteht. Die einzelnen Pfadelemente werden als Tabelle zurückgegeben, die eine Reihe von Untertabellen enthält, die jeweils ein einzelnes Pfadelement beschreiben. Eine Beschreibung der gültigen Felder in den Untertabellen finden Sie unter cairo.Path().

EINGABEN
Keine

RÜCKGABEWERTE

`t`	Tabelle mit allen Pfadelementen

11 Cairo-Muster

11.1 cpattern:AddColorStopRGB

BEZEICHNUNG

cpattern:AddColorStopRGB – Fügt einen RGB-Farbstopp hinzu

ÜBERSICHT

`cpattern:AddColorStopRGB(offset, red, green, blue)`

BESCHREIBUNG

Fügt einem Verlaufsmuster einen undurchsichtigen Farbstopp hinzu. Der Versatz gibt die Position entlang des Kontrollvektors des Farbverlaufs an. Beispielsweise verläuft der Kontrollvektor eines linearen Gradienten von (x0,y0) bis (x1,y1), während der Kontrollvektor eines radialen Gradienten von einem beliebigen Punkt auf dem Startkreis zum entsprechenden Punkt auf dem Endkreis verläuft.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGB() angegeben.

Wenn zwei (oder mehr) Stopps mit identischen Versatz-Werten angegeben werden, werden sie entsprechend der Reihenfolge sortiert, in der die Stopps hinzugefügt werden (früher hinzugefügte Stopps werden weniger verglichen als später hinzugefügte Stopps). Dies kann nützlich sein, um zuverlässig scharfe Farbübergänge anstelle der typischen Mischung zu erzielen.

Hinweis: Wenn das Muster kein Verlaufsmuster ist (z.B. ein lineares oder radiales Muster), wird das Muster in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt.

EINGABEN

`offset`	ein Versatz im Bereich [0.0 .. 1.0]
`red`	roter Farbanteil
`green`	grüner Farbanteil
`blue`	blauer Farbanteil

11.2 cpattern:AddColorStopRGBA

BEZEICHNUNG

cpattern:AddColorStopRGBA – Fügt einen RGBA-Farbstopp hinzu

ÜBERSICHT

`cpattern:AddColorStopRGBA(offset, red, green, blue, alpha)`

BESCHREIBUNG

Fügt einem Verlaufsmuster einen durchscheinenden Farbstopp hinzu. Der Versatz gibt die Position entlang des Kontrollvektors des Farbverlaufs an. Beispielsweise verläuft der Kontrollvektor eines linearen Gradienten von (x0,y0) bis (x1,y1), während der Kontrollvektor eines radialen Gradienten von einem beliebigen Punkt auf dem Startkreis zum entsprechenden Punkt auf dem Endkreis verläuft.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGBA() angegeben.

Wenn zwei (oder mehr) Stopps mit identischen Versatz-Werten angegeben werden, werden sie entsprechend der Reihenfolge sortiert, in der die Stopps hinzugefügt werden (früher hinzugefügte Stopps werden weniger verglichen als später hinzugefügte Stopps). Dies kann nützlich sein, um zuverlässig scharfe Farbübergänge anstelle der typischen Mischung zu erzielen.

Hinweis: Wenn das Muster kein Verlaufsmuster ist (z.B. ein lineares oder radiales Muster), wird das Muster in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt.

EINGABEN

`offset`	ein Versatz im Bereich [0.0 .. 1.0]
`red`	roter Farbanteil
`green`	grüner Farbanteil
`blue`	blauer Farbanteil
`alpha`	Alpha-Komponente der Farbe

11.3 cpattern:BeginPatch

BEZEICHNUNG

cpattern:BeginPatch – Beginnt einen Patch

ÜBERSICHT

```
cpattern:BeginPatch()
```

BESCHREIBUNG

Beginnt einen Patch in einem Netzmuster.

Nach dem Aufruf dieser Funktion sollte die Patchform mit cpattern:MoveTo(), cpattern:LineTo() und cpattern:CurveTo() definiert werden.

Nach der Definition des Patches muss cpattern:EndPatch() aufgerufen werden, bevor das Muster als Quelle oder Maske verwendet werden kann.

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn das Muster bereits über einen aktuellen Patch verfügt, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

Keine

11.4 cpattern:CurveTo

BEZEICHNUNG

cpattern:CurveTo – Fügt eine kubische Bézier-Spline hinzu

ÜBERSICHT

`cpattern:CurveTo(x1, y1, x2, y2, x3, y3)`

BESCHREIBUNG

Fügt dem aktuellen Patch einen kubischen Bézier-Spline vom aktuellen Punkt zur Position (`x3`, `y3`) in Musterbereichkoordinaten hinzu, wobei (`x1`, `y1`) und (`x2`, `y2`) als Kontrollpunkte verwendet werden.

Wenn der aktuelle Patch vor dem Aufruf von cpattern:CurveTo() keinen aktuellen Punkt hat, verhält sich diese Funktion so, als ob ihr ein Aufruf von cpattern:MoveTo() mit `x1` und `y1` als Parametern vorausgegangen wäre.

Nach diesem Aufruf ist der aktuelle Punkt (`x3`, `y3`).

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn das Muster keinen aktuellen Patch hat oder der aktuelle Patch bereits 4 Seiten hat, wird es in einen Fehlerstatus `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`x1`	die X-Koordinate des ersten Kontrollpunkts
`y1`	die Y-Koordinate des ersten Kontrollpunkts
`x2`	die X-Koordinate des zweiten Kontrollpunkts
`y2`	die Y-Koordinate des zweiten Kontrollpunkts
`x3`	die X-Koordinate vom Ende der Kurve
`y3`	die Y-Koordinate vom Ende der Kurve

11.5 cpattern:EndPatch

BEZEICHNUNG

cpattern:EndPatch – Beendet einen Patch

ÜBERSICHT

`cpattern:EndPatch()`

BESCHREIBUNG

Zeigt das Ende des aktuellen Patches in einem Netzmuster an.

Wenn der aktuelle Patch weniger als 4 Seiten hat, wird er mit einer geraden Linie vom aktuellen Punkt zum ersten Punkt des Patches geschlossen, als ob cpattern:LineTo() verwendet würde.

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn das Muster keinen aktuellen Patch hat oder der aktuelle Patch bereits 4 Seiten hat, wird es in einen Fehlerstatus `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

Keine

11.6 cpattern:Free

BEZEICHNUNG
cpattern:Free – Löscht ein Muster

ÜBERSICHT
`cpattern:Free()`

BESCHREIBUNG
Verringert den Referenzzähler für das Muster um eins. Wenn das Ergebnis Null ist, werden das Muster und alle zugehörigen Ressourcen gelöscht und freigegeben. Siehe cpattern:Reference().

EINGABEN
Keine

11.7 cpattern:GetColorStopCount

BEZEICHNUNG
cpattern:GetColorStopCount – Ermittelt die Anzahl der Farbstopps

ÜBERSICHT
`status, count = cpattern:GetColorStopCount()`

BESCHREIBUNG
Ruft die Anzahl der im angegebenen Farbverlaufsmuster angegebenen Farbstopps ab.

Diese Funktion gibt bei Erfolg `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein Farbverlaufsmuster ist.

EINGABEN
Keine

RÜCKGABEWERTE

`status`	Statuscode (siehe oben)
`count`	Anzahl der Farbstopps

11.8 cpattern:GetColorStopRGBA

BEZEICHNUNG
cpattern:GetColorStopRGBA – Gibt die RGBA-Farbstopps zurück

ÜBERSICHT
`status, offset, r, g, b, a = cpattern:GetColorStopRGBA(index)`

BESCHREIBUNG
Ruft die Farb- und Versatzinformationen am angegebenen `index` für ein Farbverlaufsmuster ab. Die Werte von `index` reichen von 0 bis n-1, wobei n die von cpattern:GetColorStopCount() zurückgegebene Zahl ist.

Beachten Sie, dass die Farb- und Alphawerte nicht vormultipliziert werden.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_INVALID_INDEX` zurück, wenn `index` für das angegebene Muster nicht gültig ist. Wenn das Muster kein Farbverlaufsmuster ist, wird `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurückgegeben.

EINGABEN

`index`	Index des Stopps, für den Daten zurückgegeben werden sollen

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_INVALID_INDEX`
`offset`	Versatz des Stopps
`r`	roter Farbanteil
`g`	grüner Farbanteil
`b`	blauer Farbanteil
`a`	Alpha-Komponente der Farbe

11.9 cpattern:GetControlPoint

BEZEICHNUNG

cpattern:GetControlPoint – Gibt den Kontrollpunkt zurück

ÜBERSICHT

```
status, x, y = cpattern:GetControlPoint(patch_num, point_num)
```

BESCHREIBUNG

Ruft den Kontrollpunkt `point_num` des Patches `patch_num` für ein Netzmuster ab.

patch_num kann zwischen 0 und n-1 liegen, wobei `n` die von cpattern:GetPatchCount() zurückgegebene Zahl ist.

Gültige Werte für `point_num` liegen zwischen 0 und 3 und identifizieren die Kontrollpunkte wie in cairo.PatternMesh() erläutert.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_INVALID_INDEX` zurück, wenn `patch_num` oder `point_num` für das Muster nicht gültig ist. Wenn das Muster kein Netzmuster ist, wird `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurückgegeben.

EINGABEN

`patch_num`	die Patch-Nummer, für die Daten zurückgegeben werden sollen
`point_num`	die Kontrollpunktnummer, für die Daten zurückgegeben werden sollen

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_INVALID_INDEX`
`x`	x-Koordinate des Kontrollpunkts
`y`	y-Koordinate des Kontrollpunkts

11.10 cpattern:GetCornerColorRGBA

BEZEICHNUNG

cpattern:GetCornerColorRGBA – Gibt die RGBA-Eckfarbe zurück

ÜBERSICHT

```
status, r, g, b, a = cpattern:GetCornerColorRGBA(patch_num, corner_num)
```

BESCHREIBUNG

Ruft die Farbinformationen in der Ecke `corner_num` des Patches `patch_num` für ein Netzmuster ab.

patch_num kann zwischen 0 und n-1 liegen, wobei `n` die von cpattern:GetPatchCount() zurückgegebene Zahl ist.

Gültige Werte für `corner_num` liegen zwischen 0 und 3 und identifizieren die Ecken wie in cairo.PatternMesh() erläutert.

Beachten Sie, dass die Farb- und Alphawerte nicht vormultipliziert werden.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_INVALID_INDEX` zurück, wenn `patch_num` oder `corner_num` für das Muster nicht gültig ist. Wenn das Muster kein Netzmuster ist, wird `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurückgegeben.

EINGABEN

`patch_num`
: die Patch-Nummer, für die Daten zurückgegeben werden sollen

`corner_num`
: die Ecknummer, für die Daten zurückgegeben werden sollen

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_INVALID_INDEX`
`r`	roter Farbanteil
`g`	grüner Farbanteil
`b`	blauer Farbanteil
`a`	Alpha-Komponente der Farbe

11.11 cpattern:GetExtend

BEZEICHNUNG

cpattern:GetExtend – Gibt den Erweiterungsmodus zurück

ÜBERSICHT

```
mode = cpattern:GetExtend()
```

BESCHREIBUNG

Ruft den aktuellen Erweiterungsmodus für ein Muster ab. Einzelheiten zur Semantik jeder Erweiterungsstrategie finden Sie unter cpattern:SetExtend().

EINGABEN
Keine

RÜCKGABEWERTE

`mode` Die aktuelle Erweiterungsstrategie, die zum Zeichnen des Musters verwendet wird

11.12 cpattern:GetFilter

BEZEICHNUNG
cpattern:GetFilter – Gibt den aktuellen Filter zurück

ÜBERSICHT

```
filter = cpattern:GetFilter()
```

BESCHREIBUNG
Ruft den aktuellen Filter für ein Muster ab. Einzelheiten zu jedem Filter finden Sie unter cpattern:SetFilter().

EINGABEN
Keine

RÜCKGABEWERTE

`filter` Der aktuelle Filter, der zum Ändern der Größe des Musters verwendet wird

11.13 cpattern:GetLinearPoints

BEZEICHNUNG
cpattern:GetLinearPoints – Gibt die Gradientenendpunkte zurück

ÜBERSICHT

```
status, x0, y0, x1, y1 = cpattern:GetLinearPoints()
```

BESCHREIBUNG
Ruft die Verlaufsendpunkte für einen linearen Verlauf ab.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein lineares Farbverlaufsmuster ist.

EINGABEN
Keine

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`

`x0` x-Koordinate des ersten Punktes

`y0` y-Koordinate des ersten Punktes

`x1` x-Koordinate des zweiten Punktes

`y1` y-Koordinate des zweiten Punktes

11.14 cpattern:GetMatrix

BEZEICHNUNG
cpattern:GetMatrix – Gibt die Transformationsmatrix des Musters zurück

ÜBERSICHT
```
m = cpattern:GetMatrix(matrix)
```

BESCHREIBUNG
Gibt die Transformationsmatrix des Musters zurück.

EINGABEN
Keine

RÜCKGABEWERTE

`m`	aktuelle Transformationsmatrix

11.15 cpattern:GetPatchCount

BEZEICHNUNG
cpattern:GetPatchCount – Gibt die Patch-Anzahl zurück

ÜBERSICHT
```
status, count = cpattern:GetPatchCount()
```

BESCHREIBUNG
Ruft die Anzahl der im angegebenen Netzmuster angegebenen Patches ab.

Die Zahl umfasst nur Patches, die durch den Aufruf von cpattern:EndPatch() abgeschlossen wurden. Beispielsweise wird es bei der Definition des ersten Patches 0 sein.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein Netzmuster ist.

EINGABEN
Keine

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`
`count`	Anzahl der Patches

11.16 cpattern:GetRadialCircles

BEZEICHNUNG
cpattern:GetRadialCircles – Gibt die radiale Kreise zurück

ÜBERSICHT
```
status, x0, y0, r0, x1, y1, r1 = cpattern:GetRadialCircles()
```

BESCHREIBUNG
Ruft die Farbverlaufsendpunktkreise für einen radialen Farbverlauf ab, die jeweils als Mittelkoordinate und Radius angegeben werden.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein radiales Farbverlaufsmuster ist.

EINGABEN

Keine

RÜCKGABEWERTE

`x`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`
`x0`	x-Koordinate des Mittelpunkts des ersten Kreises
`y0`	y-Koordinate des Mittelpunkts des ersten Kreises
`r0`	Radius des ersten Kreises
`x1`	x-Koordinate des Mittelpunkts des zweiten Kreises
`y1`	y-Koordinate des Mittelpunkts des zweiten Kreises
`r1`	Radius des zweiten Kreises

11.17 cpattern:GetReferenceCount

BEZEICHNUNG

cpattern:GetReferenceCount – Gibt die Referenzanzahl zurück

ÜBERSICHT

```
count = cpattern:GetReferenceCount()
```

BESCHREIBUNG

Gibt den aktuellen Referenzzähler des Musters zurück.

Diese Funktion gibt den aktuellen Referenzzähler des Musters zurück. Wenn das Objekt ein Nullobjekt ist, wird 0 zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`count`	der aktuelle Referenzzähler des Musters

11.18 cpattern:GetRGBA

BEZEICHNUNG

cpattern:GetRGBA – Gibt die RGBA-Volltonfarbe zurück

ÜBERSICHT

```
status, red, green, blue, alpha = cpattern:GetRGBA()
```

BESCHREIBUNG

Ruft die Volltonfarbe für ein Volltonmuster ab.

Beachten Sie, dass die Farb- und Alphawerte nicht vormultipliziert werden.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein einfarbiges Muster ist.

EINGABEN
Keine

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`
`red`	roter Farbanteil
`green`	grüner Farbanteil
`blue`	blauer Farbanteil
`alpha`	Alpha-Komponente der Farbe

11.19 cpattern:GetSurface

BEZEICHNUNG
cpattern:GetSurface – Gibt die Musteroberfläche zurück

ÜBERSICHT
```
status, surface = cpattern:GetSurface()
```

BESCHREIBUNG
Ruft die Oberfläche eines Oberflächenmusters ab. Die zurückgegebene Referenz gehört dem Muster. Der Aufrufer sollte csurface:Reference() aufrufen, wenn die Oberfläche beibehalten werden soll.

Diese Funktion gibt auch `#CAIRO_STATUS_SUCCESS` bei Erfolg oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` zurück, wenn das Muster kein Oberflächenmuster ist.

EINGABEN
Keine

RÜCKGABEWERTE

`status`	`#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`
`surface`	Oberfläche des Musters

11.20 cpattern:GetType

BEZEICHNUNG
cpattern:GetType – Gibt den Typ zurück

ÜBERSICHT
```
type = cpattern:GetType()
```

BESCHREIBUNG
Ruft den Typ des Musters ab. Dies wird einer der folgenden Typen sein:

`#CAIRO_PATTERN_TYPE_SOLID`
: Das Muster ist einfarbig (einheitlich). Es kann undurchsichtig oder durchscheinend sein.

`#CAIRO_PATTERN_TYPE_SURFACE`
: Das Muster basiert auf einer Oberfläche (einem Bild).

`#CAIRO_PATTERN_TYPE_LINEAR`
: Das Muster ist ein linearer Farbverlauf.

`#CAIRO_PATTERN_TYPE_RADIAL`
: Das Muster ist ein radialer Farbverlauf.

`#CAIRO_PATTERN_TYPE_MESH`
: Das Muster ist ein Netz.

`#CAIRO_PATTERN_TYPE_RASTER_SOURCE`
: Das Muster ist ein Benutzermuster, das Rasterdaten bereitstellt.

EINGABEN

Keine

RÜCKGABEWERTE

`type` die Art des Musters

11.21 cpattern:IsNull

BEZEICHNUNG

cpattern:IsNull – Überprüft, ob das Muster ungültig ist

ÜBERSICHT

```
bool = cpattern:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Muster `NULL` ist, also ungültig. Wenn Funktionen, die Muster zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Muster einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Musterzuordnung fehlgeschlagen ist. In diesem Fall ist das Muster `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Muster `NULL` ist, andernfalls `False`

11.22 cpattern:LineTo

BEZEICHNUNG

cpattern:LineTo – Fügt eine Zeile zum Patch hinzu

ÜBERSICHT

```
cpattern:LineTo(x, y)
```

BESCHREIBUNG

Fügt dem aktuellen Patch eine Linie vom aktuellen Punkt zur Position (`x`, `y`) in Musterbereichkoordinaten hinzu.

Wenn es vor dem Aufruf von cpattern:LineTo() keinen aktuellen Punkt gibt, verhält sich diese Funktion wie cpattern:MoveTo() mit den Argumenten `x` und `y`.

Nach diesem Aufruf ist der aktuelle Punkt (`x`, `y`).

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn das Muster keinen aktuellen Patch hat oder der aktuelle Patch bereits 4 Seiten hat, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`x`	die X-Koordinate des Endes der neuen Zeile
`y`	die Y-Koordinate des Endes der neuen Zeile

11.23 cpattern:MoveTo

BEZEICHNUNG

cpattern:MoveTo – Bewegt zum Punkt

ÜBERSICHT

```
cpattern:MoveTo(x, y)
```

BESCHREIBUNG

Definiert den ersten Punkt des aktuellen Patches in einem Netzmuster.

Nach diesem Aufruf ist der aktuelle Punkt (`x`, `y`).

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn das Muster keinen aktuellen Patch hat oder der aktuelle Patch bereits 4 Seiten hat, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`x`	die X-Koordinate der neuen Position
`y`	die Y-Koordinate der neuen Position

11.24 cpattern:Reference

BEZEICHNUNG

cpattern:Reference – Erhöht die Referenzanzahl für die Muster

ÜBERSICHT

```
cpattern:Reference()
```

BESCHREIBUNG

Erhöht die Referenzanzahl für das Muster um eins. Dadurch wird verhindert, dass das Muster gelöscht wird, bis ein passender Aufruf von cpattern:Free() erfolgt.

Verwenden Sie cpattern:GetReferenceCount(), um die Anzahl der Referenzen auf ein Cairo-Muster abzurufen.

EINGABEN

Keine

11.25 cpattern:SetControlPoint

BEZEICHNUNG

cpattern:SetControlPoint – Legt den Kontrollpunkt fest

ÜBERSICHT

```
cpattern:SetControlPoint(point_num, x, y)
```

BESCHREIBUNG

Legt einen internen Kontrollpunkt des aktuellen Patches fest.

Gültige Werte für `point_num` liegen zwischen 0 und 3 und identifizieren die Kontrollpunkte wie in cairo.PatternMesh() erläutert.

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn `point_num` ungültig ist, wird das Muster in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_INDEX` versetzt. Wenn das Muster keinen aktuellen Patch hat, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`point_num` Der Kontrollpunkt, für den die Position festgelegt werden soll

`x` die X-Koordinate des Kontrollpunkts

`y` die Y-Koordinate des Kontrollpunkts

11.26 cpattern:SetCornerColorRGB

BEZEICHNUNG

cpattern:SetCornerColorRGB – Stellt die RGB-Eckfarbe ein

ÜBERSICHT

```
cpattern:SetCornerColorRGB(corner_num, red, green, blue)
```

BESCHREIBUNG

Legt die Farbe einer Ecke des aktuellen Patches in einem Netzmuster fest.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGB() angegeben.

Gültige Werte für `corner_num` liegen zwischen 0 und 3 und identifizieren die Ecken wie in cairo.PatternMesh() erläutert.

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn `corner_num` ungültig ist, wird das Muster in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_INDEX` versetzt. Wenn das Muster keinen aktuellen Patch hat, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`corner_num`
: die Ecke, für die die Farbe festgelegt werden soll

`red`
: roter Farbanteil

`green`
: grüner Farbanteil

`blue`
: blauer Farbanteil

11.27 cpattern:SetCornerColorRGBA

BEZEICHNUNG

cpattern:SetCornerColorRGBA – Stellt die RGBA-Eckfarbe ein

ÜBERSICHT

```
cpattern:SetCornerColorRGBA(corner_num, red, green, blue, alpha)
```

BESCHREIBUNG

Legt die Farbe einer Ecke des aktuellen Patches in einem Netzmuster fest.

Die Farbe wird auf die gleiche Weise wie in ccontext:SetSourceRGBA() angegeben.

Gültige Werte für `corner_num` liegen zwischen 0 und 3 und identifizieren die Ecken wie in cairo.PatternMesh() erläutert.

Hinweis: Wenn es sich bei dem Muster nicht um ein Netzmuster handelt, wird es in den Fehlerstatus mit dem Status `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH` versetzt. Wenn `corner_num` ungültig ist, wird das Muster in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_INDEX` versetzt. Wenn das Muster keinen aktuellen Patch hat, wird es in einen Fehlerstatus mit dem Status `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` versetzt.

EINGABEN

`corner_num`
: die Ecke, für die die Farbe festgelegt werden soll

`red`
: roter Farbanteil

`green`
: grüner Farbanteil

`blue`
: blauer Farbanteil

`alpha`
: Alpha-Komponente der Farbe

11.28 cpattern:SetExtend

BEZEICHNUNG

cpattern:SetExtend – Stellt den Erweiterungsmodus ein

ÜBERSICHT

`cpattern:SetExtend(extend)`

BESCHREIBUNG

Legt den Modus fest, der zum Zeichnen außerhalb des Musterbereichs verwendet werden soll. Der Parameter `extend` kann auf eine der folgenden Konstanten gesetzt werden:

`#CAIRO_EXTEND_NONE`
: Pixel außerhalb des Quellmusters sind vollständig transparent.

`#CAIRO_EXTEND_REPEAT`
: Das Muster wird durch Wiederholung gekachelt.

`#CAIRO_EXTEND_REFLECT`
: Durch die Spiegelung an den Rändern wird das Muster gekachelt.

`#CAIRO_EXTEND_PAD`
: Pixel außerhalb des Musters kopieren das nächste Pixel der Quelle.

Der Standarderweiterungsmodus ist `#CAIRO_EXTEND_NONE` für Oberflächenmuster und `#CAIRO_EXTEND_PAD` für Verlaufsmuster.

EINGABEN

`extend` Erweiterungsmodus, der beschreibt, wie der Bereich außerhalb des Musters gezeichnet wird (siehe oben).

11.29 cpattern:SetFilter

BEZEICHNUNG

cpattern:SetFilter – Stellt den Filter ein

ÜBERSICHT

`cpattern:SetFilter(filter)`

BESCHREIBUNG

Legt den Filter fest, der für die Größenänderung verwendet werden soll, wenn dieses Muster verwendet wird. Der Parameter `filter` kann auf eine der folgenden Konstanten gesetzt werden:

`#CAIRO_FILTER_FAST`
: Ein Hochleistungsfilter mit einer ähnlichen Qualität wie `#CAIRO_FILTER_NEAREST`.

`#CAIRO_FILTER_GOOD`
: Ein Filter mit angemessener Leistung und einer ähnlichen Qualität wie `#CAIRO_FILTER_BILINEAR`.

`#CAIRO_FILTER_BEST`
: Die höchste verfügbare Qualität ist möglicherweise nicht für die interaktive Nutzung geeignet.

`#CAIRO_FILTER_NEAREST`
: Filterung nach dem nächsten Nachbarn.

`#CAIRO_FILTER_BILINEAR`
: Lineare Interpolation in zwei Dimensionen.

`#CAIRO_FILTER_GAUSSIAN`
: Dieser Filterwert ist derzeit nicht implementiert und sollte im aktuellen Code nicht verwendet werden.

Beachten Sie, dass Sie die Filterung möglicherweise auch dann steuern möchten, wenn Sie kein explizites Cairo-Musterobjekt haben, beispielsweise wenn Sie ccontext:SetSourceSurface() verwenden. In diesen Fällen ist es praktisch, ccontext:GetSource() zu verwenden, um Zugriff auf das Muster zu erhalten, das Cairo implizit erstellt. Zum Beispiel:

```
ctx:SetSourceSurface(image, x, y)
pat = ctx:GetSource()
pat:SetFilter(#CAIRO_FILTER_NEAREST)
```

EINGABEN

`filter` Filter, der zum Ändern der Größe des Musters verwendet werden soll

11.30 cpattern:SetMatrix

BEZEICHNUNG

cpattern:SetMatrix – Legt die Matrix fest

ÜBERSICHT

```
cpattern:SetMatrix(matrix)
```

BESCHREIBUNG

Setzt die Transformationsmatrix des Musters auf `matrix`. Diese Matrix ist eine Transformation vom Benutzerbereich zum Musterbereich.

Wenn ein Muster zum ersten Mal erstellt wird, verfügt es immer über die Identitätsmatrix für seine Transformationsmatrix, was bedeutet, dass der Musterbereich zunächst mit dem Benutzerbereich identisch ist.

Wichtig: Bitte beachten Sie, dass die Richtung dieser Transformationsmatrix vom Benutzerbereich zum Musterbereich verläuft. Das heißt, wenn Sie sich den Fluss von einem Muster zum Benutzerbereich (und weiter zum Gerätebereich) vorstellen, werden die Koordinaten in diesem Fluss durch die Umkehrung der Mustermatrix transformiert.

Wenn Sie beispielsweise möchten, dass ein Muster doppelt so groß erscheint wie standardmäßig, ist der richtige Code zu verwenden:

```
m = cairo.Matrix()
m:InitScale(0.5, 0.5)
```

```
pat:SetMatrix(m)
```

Die Verwendung von Werten von 2.0 anstelle von 0.5 im obigen Code würde dazu führen, dass das Muster in der Hälfte seiner Standardgröße angezeigt wird.

Bitte beachten Sie auch die Ausführungen der Benutzerbereich-Sperrsemantik von ccontext:SetSource().

EINGABEN

`matrix` ein Cairo-Matrixobjekt

11.31 cpattern:Status

BEZEICHNUNG

cpattern:Status – Gibt den Musterstatus zurück

ÜBERSICHT

`cpattern:Status()`

BESCHREIBUNG

Überprüft, ob für dieses Muster zuvor ein Fehler aufgetreten ist.

Diese Funktion gibt `#CAIRO_STATUS_SUCCESS`, `#CAIRO_STATUS_NO_MEMORY`, `#CAIRO_STATUS_INVALID_MATRIX`, `#CAIRO_STATUS_PATTERN_TYPE_MISMATCH`, oder `#CAIRO_STATUS_INVALID_MESH_CONSTRUCTION` zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`status` ein Cairo-Statuswert (siehe oben)

12 Cairo-Region

12.1 cregion:ContainsPoint

BEZEICHNUNG

cregion:ContainsPoint – Überprüft, ob der Punkt in der Region liegt

ÜBERSICHT

```
ok = cregion:ContainsPoint(x, y)
```

BESCHREIBUNG

Prüft, ob (`x`, `y`) in der Region enthalten ist.

EINGABEN

`x`	die x-Koordinate eines Punktes
`y`	die y-Koordinate eines Punktes

RÜCKGABEWERTE

`ok`	`True`, wenn (`x`, `y`) in der Region enthalten ist, `False`, wenn dies nicht der Fall ist

12.2 cregion:ContainsRectangle

BEZEICHNUNG

cregion:ContainsRectangle – Überprüft, ob sich das Rechteck in der Region befindet

ÜBERSICHT

```
overlap = cregion:ContainsRectangle(rectangle)
```

BESCHREIBUNG

Überprüft, ob `rectangle` innerhalb, außerhalb oder teilweise in der Region liegt. Der Parameter `rectangle` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

Diese Funktion gibt `#CAIRO_REGION_OVERLAP_IN`, wenn `rectangle` vollständig innerhalb der Region liegt, `#CAIRO_REGION_OVERLAP_OUT`, wenn `rectangle` vollständig außerhalb der Region liegt, oder `#CAIRO_REGION_OVERLAP_PART` zurück, wenn `rectangle` teilweise innerhalb und teilweise außerhalb der Region liegt.

EINGABEN

`rectangle`	eine Tabelle, die ein Rechteck beschreibt (siehe oben).

RÜCKGABEWERTE

`overlap`	eine Überlappungskonstante (siehe oben).

12.3 cregion:Copy

BEZEICHNUNG
cregion:Copy – Kopiert eine Region

ÜBERSICHT
`reg = cregion:Copy()`

BESCHREIBUNG
Ordnet ein neues Regionsobjekt zu, das den Bereich vom Original kopiert.

Diese Funktion gibt eine neu zugewiesene Cairo Region zurück. Wird mit cregion:Free() gelöscht. Diese Funktion gibt immer ein gültiges Handle zurück; Wenn kein Speicher zugewiesen werden kann, wird ein spezielles Fehlerobjekt zurückgegeben, bei dem alle Operationen auf dem Objekt nichts bewirken. Sie können dies mit cregion:Status() überprüfen.

EINGABEN
Keine

RÜCKGABEWERTE

`reg` eine neu zugewiesene Region Cairo

12.4 cregion:Equal

BEZEICHNUNG
cregion:Equal – Überprüft die Regionsgleichheit

ÜBERSICHT
`ok = cregion:Equal(region_b)`

BESCHREIBUNG
Vergleicht, ob `region_b` der Region entspricht.

Diese Funktion gibt `True` zurück, wenn beide Regionen dieselbe Abdeckung enthielten, `False`, wenn dies nicht der Fall ist oder sich eine Region in einem Fehlerstatus befindet.

EINGABEN

`region_b` eine Region von Cairo

RÜCKGABEWERTE

`ok` `True`, wenn beide Regionen dieselbe Abdeckung enthielten, andernfalls `False`

12.5 cregion:Free

BEZEICHNUNG
cregion:Free – Löscht eine Region

ÜBERSICHT
`cregion:Free()`

BESCHREIBUNG
Löscht ein Cairo-Regionsobjekt, das mit cairo.Region() oder cregion:Copy() erstellt wurde.

EINGABEN
Keine

12.6 cregion:GetExtents

BEZEICHNUNG
cregion:GetExtents – Gibt die Regionsausdehnungen zurück

ÜBERSICHT
```
rc = cregion:GetExtents()
```

BESCHREIBUNG
Ruft das umgrenzende Rechteck der Region ab. Dadurch wird eine Tabelle zurückgegeben, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN
Keine

RÜCKGABEWERTE

`rc`	Begrenzungsrechteck der Region

12.7 cregion:GetRectangle

BEZEICHNUNG
cregion:GetRectangle – Gibt ein Rechteck aus der Region zurück

ÜBERSICHT
```
rc = cregion:GetRectangle(nth)
```

BESCHREIBUNG
Gibt das `nth` Rechteck aus der Region zurück. Dadurch wird eine Tabelle zurückgegeben, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

`nth`	eine Zahl, die angibt, welches Rechteck zurückgegeben werden soll

RÜCKGABEWERTE

`rc`	Rechteck am angegebenen Index

12.8 cregion:Intersect

BEZEICHNUNG
cregion:Intersect – Prüft, ob sich Regionen überschneiden

ÜBERSICHT
`status = cregion:Intersect(other)`

BESCHREIBUNG
Berechnet den Schnittpunkt der Region mit `other`.

EINGABEN

`other` eine andere Region von Cairo

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.9 cregion:IntersectRectangle

BEZEICHNUNG
cregion:IntersectRectangle – Prüft, ob sich ein Rechteck mit der Region überschneidet

ÜBERSICHT
`status = cregion:IntersectRectangle(rectangle)`

BESCHREIBUNG
Berechnet den Schnittpunkt der Region mit `Rechteck`. Der Parameter `rectangle` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

`rectangle`
eine Tabelle, die ein Rechteck enthält (siehe oben).

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.10 cregion:IsEmpty

BEZEICHNUNG
cregion:IsEmpty – Überprüft, ob die Region leer ist

ÜBERSICHT
`ok = cregion:IsEmpty()`

BESCHREIBUNG
Überprüft, ob die Region leer ist.

EINGABEN
Keine

RÜCKGABEWERTE

`ok` `True`, wenn die Region leer ist, `False`, wenn dies nicht der Fall ist

12.11 cregion:IsNull

BEZEICHNUNG

cregion:IsNull – Überprüft, ob die Region ungültig ist

ÜBERSICHT

```
bool = cregion:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Region `NULL`, also ungültig, ist. Wenn Funktionen, die Regionen zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Region einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Regionszuordnung fehlgeschlagen ist. In diesem Fall ist die Region `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool`	`True`, wenn die Region `NULL` ist, andernfalls `False`

12.12 cregion:NumRectangles

BEZEICHNUNG

cregion:NumRectangles – Gibt die Anzahl Rechtecke zurück

ÜBERSICHT

```
count = cregion:NumRectangles()
```

BESCHREIBUNG

Gibt die Anzahl der in der Region enthaltenen Rechtecke zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`count`	die Anzahl der in der Region enthaltenen Rechtecke

12.13 cregion:Reference

BEZEICHNUNG

cregion:Reference – Erhöht die Referenzanzahl für die Region

ÜBERSICHT

```
cregion:Reference()
```

BESCHREIBUNG

Erhöht die Referenzanzahl für die Region um eins. Dadurch wird verhindert, dass die Region gelöscht wird, bis ein passender Aufruf von cregion:Free() erfolgt.

EINGABEN
Keine

12.14 cregion:Status

BEZEICHNUNG
cregion:Status – Gibt den Regionsstatus zurück

ÜBERSICHT
`status = cregion:Status()`

BESCHREIBUNG
Überprüft, ob für dieses Regionsobjekt zuvor ein Fehler aufgetreten ist.

EINGABEN
Keine

RÜCKGABEWERTE
`status` ein Cairo-Statuscode

12.15 cregion:Subtract

BEZEICHNUNG
cregion:Subtract – Subtrahiert die Region

ÜBERSICHT
`status = cregion:Subtract(other)`

BESCHREIBUNG
Subtrahiert die durch `other` angegebene Region von der Region.

EINGABEN
`other` eine andere Region von Cairo

RÜCKGABEWERTE
`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.16 cregion:SubtractRectangle

BEZEICHNUNG
cregion:SubtractRectangle – Subtrahiert das Rechteck

ÜBERSICHT
`status = cregion:SubtractRectangle(rectangle)`

BESCHREIBUNG
Subtrahiert `rectangle` von der Region. Der Parameter `rectangle` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

`rectangle`
: eine Tabelle, die ein Rechteck enthält (siehe oben)

RÜCKGABEWERTE

`x` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.17 cregion:Translate

BEZEICHNUNG

cregion:Translate – Verschiebt die Region

ÜBERSICHT

```
cregion:Translate(dx, dy)
```

BESCHREIBUNG

Verschiebt die Region um (`dx`, `dy`).

EINGABEN

`dx` Betrag, der in x-Richtung verschoben werden soll

`dy` Betrag, der in y-Richtung verschoben werden soll

12.18 cregion:Union

BEZEICHNUNG

cregion:Union – Prüft, ob die Regionen vereinigt sind

ÜBERSICHT

```
status = cregion:Union(other)
```

BESCHREIBUNG

Berechnet die Vereinigung der aktuellen Region und der durch `other` angegebenen Region.

EINGABEN

`other` eine andere Region von Cairo

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.19 cregion:UnionRectangle

BEZEICHNUNG

cregion:UnionRectangle – Berechnet die Regionsvereinigung mit dem Rechteck

ÜBERSICHT

```
status = cregion:UnionRectangle(rectangle)
```

BESCHREIBUNG
Berechnet die Vereinigung der aktuellen Region und des durch `rectangle` angegebenen Rechtecks. Der Parameter `rectangle` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

`rectangle`
eine Tabelle, die ein Rechteck enthält (siehe oben).

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.20 cregion:Xor

BEZEICHNUNG
cregion:Xor – Berechnet die exklusive Differenz der Region

ÜBERSICHT
```
status = cregion:Xor(other)
```

BESCHREIBUNG
Berechnet die exklusive Differenz der aktuellen Region mit der durch `other` angegebenen Region. Das heißt, die aktuelle Region wird so eingestellt, dass sie alle Gebiete enthält, die sich entweder in der aktuellen Region oder in `other`, aber nicht in beiden befinden.

EINGABEN

`other` eine andere Region von Cairo

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

12.21 cregion:XorRectangle

BEZEICHNUNG
cregion:XorRectangle – Berechnet die exklusive Differenz der Region zum Rechteck

ÜBERSICHT
```
status = cregion:XorRectangle(rectangle)
```

BESCHREIBUNG
Berechnet die exklusive Differenz der aktuellen Region mit `rectangle`. Das heißt, die aktuelle Region wird so eingestellt, dass sie alle Bereiche enthält, die entweder in der aktuellen Region oder in `rectangle` liegen, aber nicht in beiden. Der Parameter `rectangle` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

`rectangle`
eine Tabelle, die ein Rechteck enthält (siehe oben).

RÜCKGABEWERTE

`status` `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY`

13 Cairo-Skalierte-Schriftart

13.1 cscaledfont:Extents

BEZEICHNUNG
cscaledfont:Extents – Gibt die Schriftartenausdehnungen zurück

ÜBERSICHT
```
t = cscaledfont:Extents(extents)
```

BESCHREIBUNG
Ruft die Metriken für eine in Cairo skalierte Schriftart ab. Dies gibt eine Tabelle mit den Schriftartbereichen zurück. Eine Erläuterung der einzelnen Tabellenelemente finden Sie unter ccontext:FontExtents().

EINGABEN

`t`	Tabelle mit den Schriftausdehnungen (siehe oben).

13.2 cscaledfont:Free

BEZEICHNUNG
cscaledfont:Free – Löscht eine skalierte Schriftart

ÜBERSICHT
```
cscaledfont:Free()
```

BESCHREIBUNG
Verringert die Referenzanzahl für die Schriftart um eins. Wenn das Ergebnis Null ist, werden die Schriftart und alle zugehörigen Ressourcen gelöscht und freigegeben. Siehe cscaledfont:Reference().

EINGABEN
Keine

13.3 cscaledfont:GetCTM

BEZEICHNUNG
cscaledfont:GetCTM – Gibt die aktuelle Transformationsmatrix zurück

ÜBERSICHT
```
m = cscaledfont:GetCTM()
```

BESCHREIBUNG
Gibt den CTM zurück, mit dem die skalierte Schriftart erstellt wurde. Beachten Sie, dass die Verschiebungsoffsets (x0, y0) des CTM von cairo.ScaledFont() ignoriert werden. Die von dieser Funktion zurückgegebene Matrix hat also immer 0,0 als x0,y0.

EINGABEN

`m`	aktuelle Transformationsmatrix

13.4 cscaledfont:GetFontFace

BEZEICHNUNG

cscaledfont:GetFontFace – Gibt die Schriftart zurück

ÜBERSICHT

```
font = cscaledfont:GetFontFace()
```

BESCHREIBUNG

Ruft die Schriftart ab, die diese skalierte Schriftart verwendet. Dies könnte die an cairo.ScaledFont() übergebene Schriftart sein, dies gilt jedoch nicht für alle möglichen Fälle.

Diese Funktion gibt die Schriftart Cairo zurück, mit der die skalierte Schriftart erstellt wurde. Dieses Objekt ist Eigentum von Cairo. Um einen Verweis darauf beizubehalten, müssen Sie cscaledfont:Reference() aufrufen.

EINGABEN

Keine

RÜCKGABEWERTE

`font`	die Schriftart Cairo, mit der die skalierte Schriftart erstellt wurde

13.5 cscaledfont:GetFontMatrix

BEZEICHNUNG

cscaledfont:GetFontMatrix – Ruft die Schriftartenmatrix ab

ÜBERSICHT

```
m = cscaledfont:GetFontMatrix()
```

BESCHREIBUNG

Gibt die Schriftartmatrix zurück, mit der die skalierte Schriftart erstellt wurde.

EINGABEN

`m`	eine Cairo-Matrix

13.6 cscaledfont:GetFontOptions

BEZEICHNUNG

cscaledfont:GetFontOptions – Gibt die Schriftartoptionen zurück

ÜBERSICHT

```
opt = cscaledfont:GetFontOptions(options)
```

BESCHREIBUNG

Gibt die Schriftartoptionen zurück, mit denen die skalierte Schriftart erstellt wurde.

EINGABEN

`opt`	ein Cairo-Schriftartoptionsobjekt

13.7 cscaledfont:GetReferenceCount

BEZEICHNUNG
cscaledfont:GetReferenceCount – Gibt die Referenzanzahl zurück

ÜBERSICHT
```
count = cscaledfont:GetReferenceCount()
```

BESCHREIBUNG
Gibt den aktuellen Referenzzähler der skalierten Schriftart zurück.

EINGABEN
Keine

RÜCKGABEWERTE

`count`	der aktuelle Referenzzähler der skalierten Schriftart

13.8 cscaledfont:GetScaleMatrix

BEZEICHNUNG
cscaledfont:GetScaleMatrix – Gibt die Skalierungsmatrix zurück

ÜBERSICHT
```
m = cscaledfont:GetScaleMatrix()
```

BESCHREIBUNG
Gibt die Skalierungsmatrix der skalierten Schriftart zurück. Die Skalierungsmatrix ist das Produkt der Schriftartenmatrix und des mit der skalierten Schriftart verbundenen CTM und stellt somit die Matrixzuordnung vom Schriftartenbereich zum Gerätebereich dar.

EINGABEN

`m`	Skalierungsmatrix der Schriftart

13.9 cscaledfont:GetType

BEZEICHNUNG
cscaledfont:GetType – Gibt den Typ der Schriftart zurück

ÜBERSICHT
```
type = cscaledfont:GetType()
```

BESCHREIBUNG
Diese Funktion gibt den Typ des Backends zurück, der zum Erstellen einer skalierten Schriftart verwendet wird. Eine Liste möglicher Schriftarten finden Sie unter cfontface:GetType(). Beachten Sie, dass diese Funktion niemals `#CAIRO_FONT_TYPE_TOY` zurückgibt.

EINGABEN
Keine

RÜCKGABEWERTE

`type` der Typ der skalierten Schriftart

13.10 cscaledfont:GlyphExtents

BEZEICHNUNG

cscaledfont:GlyphExtents – Gibt die Glyphen-Ausdehnungen zurück

ÜBERSICHT

```
t = cscaledfont:GlyphExtents(glyphs[, offset, num_glyphs])
```

BESCHREIBUNG

Ruft die Grenzen für ein Glyphen-Array ab. Die Grenzen beschreiben ein Benutzerbereichrechteck, das den "eingefärbten" Teil der Glyphen umschließt, wie sie von ccontext:ShowGlyphs() gezeichnet würden, wenn der Grafikstatus "cairo" auf die gleiche Schriftart, Schriftartmatrix und CTM eingestellt wäre und Schriftartoptionen wie die skalierte Schriftart. Darüber hinaus geben die Werte `XAdvance` und `YAdvance` den Betrag an, um den der aktuelle Punkt durch ccontext:ShowGlyphs() ausgedehnt würde.

Beachten Sie, dass Leerzeichen Glyphen nicht zur Größe des Rechtecks beitragen, wie sie in den Feldern `Width` und `Height` der Ausmaße zurückgegeben werden.

Diese Funktion gibt eine Tabelle zurück, die die Glyphenausdehnungen enthält. Eine Beschreibung aller Tabellenfelder finden Sie unter ccontext:TextExtents().

EINGABEN

`glyphs` ein Glyphen-Array, das Glyphen-IDs mit X- und Y-Versatz enthält

`offset` Optional: Versatz des Array, das das Startzeichen angibt (standardmäßig 0, was das erste Zeichen bedeutet).

`num_glyphs` Optional: Anzahl der anzuzeigenden Glyphen (Standard ist -1, was alle Glyphen bedeutet)

RÜCKGABEWERTE

`t` Tabelle mit den Glyphenausdehnungen (siehe oben).

13.11 cscaledfont:IsNull

BEZEICHNUNG

cscaledfont:IsNull – Überprüft, ob die skalierte Schriftart ungültig ist

ÜBERSICHT

```
bool = cscaledfont:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die skalierte Schriftart `NULL`, also ungültig, ist. Wenn Funktionen, die Objekte zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern

setzen das Objekt einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Objektzuordnung fehlgeschlagen ist. In diesem Fall ist das Objekt `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Objekt `NULL` ist, andernfalls `False`

13.12 cscaledfont:Reference

BEZEICHNUNG

cscaledfont:Reference – Erhöht die Referenzanzahl der skalierten Schriftart

ÜBERSICHT

```
cscaledfont:Reference()
```

BESCHREIBUNG

Erhöht die Referenzanzahl der skalierten Schriftart um eins. Dadurch wird verhindert, dass die skalierte Schriftart gelöscht wird, bis ein entsprechender Aufruf von cscaledfont:Free() erfolgt.

Verwenden Sie cscaledfont:GetReferenceCount(), um die Anzahl der Verweise auf eine skalierte Cairo-Schriftart abzurufen.

EINGABEN

Keine

13.13 cscaledfont:Status

BEZEICHNUNG

cscaledfont:Status – Gibt den Status der skalierten Schriftart zurück

ÜBERSICHT

```
status = cscaledfont:Status()
```

BESCHREIBUNG

Überprüft, ob für diese skalierte Schriftart zuvor ein Fehler aufgetreten ist.

Diese Funktion gibt bei Erfolg `#CAIRO_STATUS_SUCCESS` oder einen Fehler wie `#CAIRO_STATUS_NO_MEMORY` zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`status` ein Cairo-Statuscode (siehe oben)

13.14 cscaledfont:TextExtents

BEZEICHNUNG

cscaledfont:TextExtents – Gibt die Textausdehnungen zurück

ÜBERSICHT

```
t = cscaledfont:TextExtents(s$)
```

BESCHREIBUNG

Ruft die Ausdehnung für eine TextZeichenkette ab. Die Ausdehnung beschreiben ein Rechteck im Benutzerbereich, das den "eingefärbten" Teil des am Ursprung (0.0) gezeichneten Texts umschließt, wie er von ccontext:ShowText() gezeichnet würde, wenn der Cairographics-Status auf dieselbe Schriftart, Schriftmatrix, ctm und Schriftoptionen wie die skalierte Schriftart eingestellt wäre. Zusätzlich geben die Werte `XAdvance` und `YAdvance` an, um wie viel der aktuelle Punkt durch ccontext:ShowText() vorgerückt würde.

Beachten Sie, dass Leerzeichen nicht direkt zur Größe des Rechtecks beitragen, wie sie in den Feldern `Width` und `Height` der Ausmaße enthalten ist. Sie tragen indirekt dazu bei, indem sie die Position von Nicht-Leerzeichen ändern. Insbesondere nachgestellte Leerzeichen haben wahrscheinlich keinen Einfluss auf die Größe des Rechtecks, wohl aber auf die Werte `XAdvance` und `YAdvance`.

Diese Funktion gibt eine Tabelle zurück, die die Textbereiche enthält. Eine Beschreibung aller Tabellenfelder finden Sie unter ccontext:TextExtents().

EINGABEN

`s$`	eine Textfolge

RÜCKGABEWERTE

`t`	Tabelle mit den Textumfängen (siehe oben)

14 Cairo-Oberfläche

14.1 csurface:AddOutline

BEZEICHNUNG

csurface:AddOutline – Fügt die Gliederung hinzu

ÜBERSICHT

```
id = csurface:AddOutline(parent_id, name$, link$, flags)
```

BESCHREIBUNG

Bei PDF-Oberflächen wird dadurch der Dokumentgliederungshierarchie ein Element mit dem Namen `name$` hinzugefügt, das auf den durch `link$` angegebenen Speicherort verweist. Link-Attribute haben dieselben Schlüssel und Werte wie das Link-Tag, mit Ausnahme des Attributs `rect`. Das Element ist ein untergeordnetes Element des Elements mit der ID `parent_id`. Verwenden Sie `#CAIRO_PDF_OUTLINE_ROOT` als übergeordnete ID von Elementen der obersten Ebene. Der Parameter `flags` kann auf eine Kombination der folgenden Flags gesetzt werden:

`#CAIRO_PDF_OUTLINE_FLAG_OPEN`
: Das Gliederungselement wird standardmäßig im PDF-Viewer geöffnet.

`#CAIRO_PDF_OUTLINE_FLAG_BOLD`
: Das Gliederungselement wird vom Betrachter in Fettschrift angezeigt.

`#CAIRO_PDF_OUTLINE_FLAG_ITALIC`
: Das Gliederungselement wird vom Betrachter in kursiver Schrift angezeigt.

Diese Funktion gibt die ID für das hinzugefügte Element zurück.

EINGABEN

`parent_id`
: die ID des übergeordneten Elements oder `#CAIRO_PDF_OUTLINE_ROOT`, wenn es sich um ein Element der obersten Ebene handelt

`name$`
: der Name der Gliederung

`link$`
: die Linkattribute, die angeben, wohin diese Gliederung verlinkt

`flags`
: Gliederungselement-Flags (siehe oben).

RÜCKGABEWERTE

`id`
: ID für das hinzugefügte Element

14.2 csurface:CopyPage

BEZEICHNUNG

csurface:CopyPage – Kopiert eine Seite

ÜBERSICHT

```
csurface:CopyPage()
```

BESCHREIBUNG

Gibt die aktuelle Seite für Backends aus, die mehrere Seiten unterstützen, löscht sie jedoch nicht, sodass der Inhalt der aktuellen Seite für die nächste Seite beibehalten wird. Verwenden Sie csurface:ShowPage(), wenn Sie nach der Ausgabe eine leere Seite erhalten möchten.

Hierfür gibt es eine praktische Funktion, die einen Cairo-Kontext benötigt, nämlich ccontext:CopyPage().

EINGABEN

Keine

14.3 csurface:CreateForRectangle

BEZEICHNUNG

csurface:CreateForRectangle – Erstellt eine Fläche als Rechteck

ÜBERSICHT

```
handle = csurface:CreateForRectangle(x, y, width, height)
```

BESCHREIBUNG

Erstellt eine neue Fläche, die ein Rechteck innerhalb der Zielfläche ist. Alle auf dieser Oberfläche gezeichneten Operationen werden dann ausgeschnitten und auf die Zieloberfläche übertragen. Nichts, was über diese Unteroberfläche außerhalb ihrer Grenzen gezeichnet wird, wird auf die Zieloberfläche gezeichnet. Dies macht dies zu einer nützlichen Funktion, um eingeschränkte untergeordnete Oberflächen an Bibliotheksroutinen zu übergeben, die direkt auf die übergeordnete Oberfläche zeichnen, d.h. ohne weitere Backend-Zuweisungen, doppelte Pufferung oder Kopien .

Die Semantik von Untergrundflächen ist noch nicht endgültig festgelegt, es sei denn, das Rechteck besteht aus vollständigen Geräteeinheiten, ist in den Ausmaßen der Zieloberfläche enthalten und die Gerätetransformationen des Ziels oder Untergrunds werden nicht geändert.

Diese Funktion gibt ein Handle für die neu zugewiesene Oberfläche zurück. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Diese Funktion gibt immer ein gültiges Handle zurück, sie gibt jedoch ein Handle für eine "Null"-Oberfläche zurück, wenn sich die Quelloberfläche bereits in einem Fehlerzustand befindet oder ein anderer Fehler auftritt.

EINGABEN

`x`	der x-Ursprung der Unteroberfläche von der oberen linken Ecke der Zieloberfläche (in Gerätebereicheinheiten).
`y`	der y-Ursprung der Unteroberfläche von der oberen linken Ecke der Zieloberfläche (in Gerätebereicheinheiten).
`width`	Breite des Untergrunds (in Gerätebereicheinheiten)
`height`	Höhe des Untergrunds (in Gerätebereicheinheiten)

RÜCKGABEWERTE

`handle` Handle auf eine neue Oberfläche

14.4 csurface:CreateSimilar

BEZEICHNUNG

csurface:CreateSimilar – Erstellt eine ähnliche Oberfläche

ÜBERSICHT

```
handle = csurface:CreateSimilar(content, width, height)
```

BESCHREIBUNG

Erstellt eine neue Oberfläche, die möglichst kompatibel zu einer vorhandenen Oberfläche ist. Beispielsweise verfügt die neue Oberfläche über denselben Gerätemaßstab, dieselbe Ersatzauflösung und dieselben Schriftartoptionen wie die Quelloberfläche. Im Allgemeinen verwendet die neue Oberfläche auch dasselbe Backend wie die Quelloberfläche, es sei denn, dies ist aus irgendeinem Grund nicht möglich. Der Typ der zurückgegebenen Oberfläche kann mit csurface:GetType() untersucht werden.

Der Parameter `content` kann auf eine der folgenden Konstanten gesetzt werden:

`#CAIRO_CONTENT_COLOR`
: Die Oberfläche nimmt nur Farbinhalte auf.

`#CAIRO_CONTENT_ALPHA`
: Die Oberfläche enthält nur Alpha-Inhalte.

`#CAIRO_CONTENT_COLOR_ALPHA`
: Die Oberfläche enthält Farbe und Alphainhalte.

Anfänglich sind alle Oberflächeninhalte 0 (transparent, wenn der Inhalt transparent ist, andernfalls schwarz).

Verwenden Sie csurface:CreateSimilarImage(), wenn Sie eine Bildoberfläche benötigen, die schnell auf die Zieloberfläche gemalt werden kann.

Diese Funktion gibt ein Handle für die neu zugewiesene Oberfläche zurück. Der Aufrufer hat das zurückgegebenen Objekt zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Diese Funktion gibt immer ein gültiges Handle zurück, sie gibt jedoch ein Handle für eine "Null"-Oberfläche zurück, wenn sich die Quelloberfläche bereits in einem Fehlerzustand befindet oder ein anderer Fehler auftritt.

EINGABEN

`content` den Inhalt für die neue Oberfläche (siehe oben)

`width` Breite der neuen Oberfläche (in Gerätebereicheinheiten)

`height` Höhe der neuen Oberfläche (in Gerätebereicheinheiten)

RÜCKGABEWERTE

`handle` Handle auf eine neue Oberfläche

14.5 csurface:CreateSimilarImage

BEZEICHNUNG

csurface:CreateSimilarImage – Erstellt eine ähnliche Bildoberfläche

ÜBERSICHT

```
handle = csurface:CreateSimilarImage(format, width, height)
```

BESCHREIBUNG

Erstellt eine neue Bildoberfläche, die möglichst kompatibel zum Hochladen und zur Verwendung in Verbindung mit einer vorhandenen Oberfläche ist. Diese Oberfläche kann jedoch weiterhin wie jede normale Bildoberfläche verwendet werden. Im Gegensatz zu csurface:CreateSimilar() erbt die neue Bildoberfläche nicht den Gerätemaßstab von der Quelloberfläche. Der Parameter `format` muss auf eine Pixelformatkonstante festgelegt werden. Siehe Abschnitt 4.5 [cairo.ImageSurface], Seite 16, für Details.

Anfänglich sind alle Oberflächeninhalte 0 (transparent, wenn der Inhalt transparent ist, andernfalls schwarz).

Verwenden Sie csurface:CreateSimilar(), wenn Sie keine Bildoberfläche benötigen.

Diese Funktion gibt ein Handle für die neu zugewiesene Bildoberfläche zurück. Der Aufrufer hat die zurückgegebenen Oberfläche zugewiesen und sollte csurface:Free() aufrufen, wenn er damit fertig ist.

Diese Funktion gibt immer ein gültiges Handle zurück, aber sie gibt ein Handle auf eine "Null"-Oberfläche zurück, wenn sich `other` bereits in einem Fehlerzustand befindet oder ein anderer Fehler auftritt.

EINGABEN

`format`	das Format für die neue Oberfläche
`width`	Breite der neuen Oberfläche (in Pixel)
`height`	Höhe der neuen Oberfläche (in Pixel)

RÜCKGABEWERTE

`handle`	Handle auf eine neue Oberfläche

14.6 csurface:Finish

BEZEICHNUNG

csurface:Finish – Vervollständigt die Oberfläche

ÜBERSICHT

```
csurface:Finish()
```

BESCHREIBUNG

Diese Funktion vervollständigt die Oberfläche und löscht alle Verweise auf externe Ressourcen. Für das Xlib-Backend bedeutet dies beispielsweise, dass Cairo nicht mehr auf das Zeichenbare zugreift, das somit gelöscht und freigegeben werden kann. Nach dem Aufruf von csurface:Finish() sind die einzigen gültigen Vorgänge auf einer Oberfläche die Überprüfung des Status, das Abrufen und Festlegen des Benutzers, das Referenzieren

und Löschen sowie das Leeren und Beenden der Oberfläche. Ein weiteres Zeichnen zur Oberfläche hat keine Auswirkungen auf die Oberfläche, sondern löst stattdessen einen #CAIRO_STATUS_SURFACE_FINISHED-Fehler aus.

Wenn der letzte Aufruf von csurface:Free() den Referenzzähler auf Null verringert, ruft Cairo csurface:Finish() auf, falls es noch nicht aufgerufen wurde, bevor die mit der Oberfläche verknüpften Ressourcen gelöscht und freigegeben werden.

EINGABEN
Keine

14.7 csurface:Flush

BEZEICHNUNG
csurface:Flush – Erstellt alle Zeichnungen für die Oberfläche

ÜBERSICHT
`csurface:Flush()`

BESCHREIBUNG
Erstellt alle ausstehenden Zeichnungen für die Oberfläche und stellen auch alle vorübergehenden Änderungen wieder her, die Cairo am Zustand der Oberfläche vorgenommen hat. Diese Funktion muss aufgerufen werden, bevor vom Zeichnen auf der Oberfläche mit Cairo zum direkten Zeichnen darauf mit nativen APIs oder vor dem Zugriff auf den Speicher außerhalb von Cairo gewechselt wird. Wenn die Oberfläche keinen direkten Zugriff unterstützt, führt diese Funktion nichts aus.

EINGABEN
Keine

14.8 csurface:Free

BEZEICHNUNG
csurface:Free – Löscht eine Oberfläche

ÜBERSICHT
`csurface:Free()`

BESCHREIBUNG
Verringert die Referenzanzahl auf der Oberfläche um eins. Wenn das Ergebnis Null ist, werden die Oberfläche und alle zugehörigen Ressourcen gelöscht und freigegeben. Siehe csurface:Reference().

EINGABEN
Keine

14.9 csurface:GetContent

BEZEICHNUNG
csurface:GetContent – Gibt den Inhaltstyp zurück

ÜBERSICHT
```
content = csurface:GetContent()
```

BESCHREIBUNG
Diese Funktion gibt den Inhaltstyp der Oberfläche zurück, der angibt, ob die Oberfläche Farb- und/oder Alphainformationen enthält. Dies kann eine der folgenden Konstanten sein:

`#CAIRO_CONTENT_COLOR`
: Die Oberfläche enthält nur Farbinhalte.

`#CAIRO_CONTENT_ALPHA`
: Die Oberfläche enthält nur Alpha-Inhalte.

`#CAIRO_CONTENT_COLOR_ALPHA`
: Die Oberfläche enthält Farb- und Alphainhalte.

EINGABEN
Keine

RÜCKGABEWERTE

`content` der Inhaltstyp der Oberfläche

14.10 csurface:GetDeviceOffset

BEZEICHNUNG
csurface:GetDeviceOffset – Gibt den Geräte-Versatz zurück

ÜBERSICHT
```
x_offset, y_offset = csurface:GetDeviceOffset()
```

BESCHREIBUNG
Diese Funktion gibt den vorherigen Geräteversatz zurück, der durch csurface:SetDeviceOffset() festgelegt wurde.

EINGABEN
Keine

RÜCKGABEWERTE

`x_offset` der Versatz in X-Richtung in Geräteeinheiten

`y_offset` der Versatz in Y-Richtung in Geräteeinheiten

14.11 csurface:GetDeviceScale

BEZEICHNUNG

csurface:GetDeviceScale – Gibt die Geräteskalierung zurück

ÜBERSICHT

```
x_scale, y_scale = csurface:GetDeviceScale()
```

BESCHREIBUNG

Diese Funktion gibt die vorherige Geräteskalierung zurück, die von csurface:SetDeviceScale() festgelegt wurde.

EINGABEN

Keine

RÜCKGABEWERTE

`x_scale` die Skalierung in X-Richtung in Geräteeinheiten

`y_scale` die Skalierung in Y-Richtung in Geräteeinheiten

14.12 csurface:GetDocumentUnit

BEZEICHNUNG

csurface:GetDocumentUnit – Gibt die Dokumenteneinheit zurück

ÜBERSICHT

```
unit = csurface:GetDocumentUnit()
```

BESCHREIBUNG

Ermittelt die Einheit der SVG-Oberfläche.

Wenn die als Argument übergebene Oberfläche keine SVG-Oberfläche ist, setzt die Funktion den Fehlerstatus auf `#CAIRO_STATUS_SURFACE_TYPE_MISMATCH` und gibt `#CAIRO_SVG_UNIT_USER` zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`unit` die SVG-Einheit der SVG-Oberfläche

14.13 csurface:GetFallbackResolution

BEZEICHNUNG

csurface:GetFallbackResolution – Gibt die Fallback-Auflösung zurück

ÜBERSICHT

```
xppi, yppi = csurface:GetFallbackResolution()
```

BESCHREIBUNG

Diese Funktion gibt die vorherige Fallback-Auflösung zurück, die von csurface:SetFallbackResolution() festgelegt wurde, oder die Standard-Fallback-Auflösung, wenn diese nie festgelegt wurde.

EINGABEN
Keine

RÜCKGABEWERTE

`xppi` horizontale Pixel pro Zoll

`yppi` vertikale Pixel pro Zoll

14.14 csurface:GetFontOptions

BEZEICHNUNG
csurface:GetFontOptions – Gibt die Schriftartoptionen zurück

ÜBERSICHT
```
csurface:GetFontOptions(options)
```

BESCHREIBUNG
Ruft die Standard-Schriftart-Rendering-Optionen für die Oberfläche ab. Dies ermöglicht es für Anzeigeoberflächen, die richtige Subpixelreihenfolge für das Rendern ihnen zurück zu melden, bei Druckoberflächen die Hinweise auf Metriken zu deaktivieren usw. Das Ergebnis kann dann mit cairo.ScaledFont() verwendet werden.

EINGABEN

`options` ein Cairo-Schriftartoptionsobjekt, in dem die abgerufenen Optionen gespeichert werden; alle vorhandenen Werte werden überschrieben

14.15 csurface:GetFormat

BEZEICHNUNG
csurface:GetFormat – Gibt das Pixelformat der Oberfläche zurück

ÜBERSICHT
```
format = csurface:GetFormat()
```

BESCHREIBUNG
Ermittelt das Pixelformat der Oberfläche. Siehe Abschnitt 4.5 [cairo.ImageSurface], Seite 16, für eine Liste von Pixelformaten.

EINGABEN
Keine

RÜCKGABEWERTE

`format` Pixelformat der Oberfläche

14.16 csurface:GetHeight

BEZEICHNUNG
csurface:GetHeight – Gibt die Oberflächenhöhe zurück

ÜBERSICHT
```
height = csurface:GetHeight()
```

BESCHREIBUNG
Ermittelt die Höhe der Bildoberfläche in Pixel.

EINGABEN
Keine

RÜCKGABEWERTE
`height` die Höhe der Oberfläche in Pixel

14.17 csurface:GetMimeData

BEZEICHNUNG
csurface:GetMimeData – Gibt die Mime-Daten zurück

ÜBERSICHT
```
data$ = csurface:GetMimeData(mime_type$)
```

BESCHREIBUNG
Gibt Mime-Daten zurück, die zuvor mithilfe des angegebenen Mime-Typs an die Oberfläche angehängt wurden. Wenn mit dem angegebenen Mime-Typ keine Daten angehängt wurden, wird `data$` auf Null gesetzt.

EINGABEN
`mime_type$`
der MIME-Typ der Bilddaten

RÜCKGABEWERTE
`data$` Die Bilddaten werden an die Oberfläche angehängt

14.18 csurface:GetReferenceCount

BEZEICHNUNG
csurface:GetReferenceCount – Gibt die Referenzanzahl zurück

ÜBERSICHT
```
count = csurface:GetReferenceCount()
```

BESCHREIBUNG
Gibt die aktuelle Referenzanzahl der Oberfläche zurück.

EINGABEN
Keine

RÜCKGABEWERTE

`count` Referenzanzahl der Oberfläche

14.19 csurface:GetType

BEZEICHNUNG

csurface:GetType – Gibt den Oberflächentyp zurück

ÜBERSICHT

```
type = csurface:GetType()
```

BESCHREIBUNG

Diese Funktion gibt den Typ des Backends zurück, das zum Erstellen einer Oberfläche verwendet wird. Dies kann einer der folgenden Typen sein:

`#CAIRO_SURFACE_TYPE_IMAGE`
: Die Oberfläche ist vom Typ Bild.

`#CAIRO_SURFACE_TYPE_PDF`
: Die Oberfläche ist vom Typ pdf.

`#CAIRO_SURFACE_TYPE_PS`
: Die Oberfläche ist vom Typ ps.

`#CAIRO_SURFACE_TYPE_SVG`
: Die Oberfläche ist vom Typ svg.

`#CAIRO_SURFACE_TYPE_QUARTZ`
: Die Oberfläche ist vom Typ quartz.

`#CAIRO_SURFACE_TYPE_QUARTZ_IMAGE`
: Die Oberfläche ist vom Typ Quarzbild.

`#CAIRO_SURFACE_TYPE_WIN32`
: Die Oberfläche ist vom Typ win32.

EINGABEN

Keine

RÜCKGABEWERTE

`type` die Art der Oberfläche (siehe oben)

14.20 csurface:GetWidth

BEZEICHNUNG

csurface:GetWidth – Gibt die Oberflächenbreite zurück

ÜBERSICHT

```
width = csurface:GetWidth()
```

BESCHREIBUNG

Ermittelt die Breite der Bildoberfläche in Pixel.

EINGABEN
Keine

RÜCKGABEWERTE
width -die Breite der Oberfläche in Pixel

14.21 csurface:IsNull

BEZEICHNUNG
csurface:IsNull – Überprüft, ob die Oberfläche ungültig ist

ÜBERSICHT
```
bool = csurface:IsNull()
```

BESCHREIBUNG
Gibt `True` zurück, wenn die Oberfläche `NULL` ist, also ungültig. Wenn Funktionen, die Oberflächen zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Oberfläche einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Oberflächenzuordnung fehlgeschlagen ist. In diesem Fall ist die Oberfläche `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN
Keine

RÜCKGABEWERTE

`bool` `True`, wenn die Oberfläche `NULL` ist, andernfalls `False`

14.22 csurface:MarkDirty

BEZEICHNUNG
csurface:MarkDirty – Markiert eine unsaubere Oberfläche

ÜBERSICHT
```
csurface:MarkDirty()
```

BESCHREIBUNG
Teilt Cairo mit, dass das Zeichnen auf der Oberfläche mit anderen Mitteln als Cairo erfolgt ist und dass Cairo alle zwischengespeicherten Bereiche erneut lesen soll. Beachten Sie, dass Sie csurface:Flush() aufrufen müssen, bevor Sie eine solche Zeichnung durchführen.

EINGABEN
Keine

14.23 csurface:MarkDirtyRectangle

BEZEICHNUNG
csurface:MarkDirtyRectangle – Markiert ein Dirty Rectangle (unsauberes Rechteck)

ÜBERSICHT
`csurface:MarkDirtyRectangle(x, y, width, height)`

BESCHREIBUNG
Wie csurface:MarkDirty(), aber das Zeichnen wurde nur für das angegebene Rechteck durchgeführt, sodass Cairo zwischengespeicherte Inhalte für andere Teile der Oberfläche behalten kann.

Alle zwischengespeicherten Clip-Sets auf der Oberfläche werden durch diese Funktion zurückgesetzt, um sicherzustellen, dass zukünftige Cairo-Anrufe über das Clip-Set verfügen, das sie erwarten.

EINGABEN

`x`	X-Koordinate des Dirty Rectangle
`y`	Y-Koordinate des Dirty Rectangle
`width`	Breite des Dirty Rectangle
`height`	Höhe des Dirty Rectangle

14.24 csurface:Reference

BEZEICHNUNG
csurface:Reference – Erhöht die Referenzanzahl der Oberflächen

ÜBERSICHT
`csurface:Reference()`

BESCHREIBUNG
Erhöht die Referenzanzahl auf der Oberfläche um eins. Dadurch wird verhindert, dass die Oberfläche gelöscht wird, bis ein passender Aufruf von csurface:Free() erfolgt.

Verwenden Sie csurface:GetReferenceCount(), um die Anzahl der Referenzen auf eine Oberfläche abzurufen.

EINGABEN
Keine

14.25 csurface:RestrictToVersion

BEZEICHNUNG
csurface:RestrictToVersion – Beschränkt auf eine Version

ÜBERSICHT
`csurface:RestrictToVersion(version)`

BESCHREIBUNG

Beschränkt die generierte SVG- oder PDF-Datei auf `version`. Für PDF-Oberflächen müssen Sie hier eine PDF-Versionskonstante übergeben. Eine Liste der verfügbaren PDF-Versionen finden Sie unter cairo.PDFSurface(). Für SVG-Oberflächen müssen Sie hier eine SVG-Versionskonstante übergeben. Eine Liste der verfügbaren SVG-Versionen finden Sie unter cairo.SVGSurface().

Diese Funktion sollte nur dann aufgerufen werden, bevor Zeichenvorgänge auf der angegebenen Oberfläche ausgeführt wurden. Der einfachste Weg, dies zu tun, besteht darin, diese Funktion direkt nach dem Erstellen der Oberfläche aufzurufen.

EINGABEN

`version` PDF- oder SVG-Version (siehe oben)

14.26 csurface:SetDeviceOffset

BEZEICHNUNG

csurface:SetDeviceOffset – Stellt den Geräte-Versatz ein

ÜBERSICHT

```
csurface:SetDeviceOffset(x_offset, y_offset)
```

BESCHREIBUNG

Legt einen Versatz fest, der zu den Gerätekoordinaten hinzugefügt wird, die vom CTM beim Zeichnen auf die Oberfläche ermittelt werden. Ein Anwendungsfall für diese Funktion besteht darin, eine Oberfläche zu erstellen, die die Zeichnung für einen Teil einer Bildschirmoberfläche auf eine für den Benutzer der Cairo-API völlig unsichtbare Weise auf eine Offscreen-Oberfläche umleitet. Das Festlegen einer Transformation über ccontext:Translate() reicht hierfür nicht aus, da Funktionen wie ccontext:DeviceToUser() den verborgenen Versatz offenlegen.

Beachten Sie, dass sich der Versatz sowohl auf das Zeichnen zur Oberfläche als auch auf die Verwendung der Oberfläche in einem Quellmuster auswirkt.

EINGABEN

`x_offset` der Versatz in X-Richtung in Geräteeinheiten

`y_offset` der Versatz in Y-Richtung in Geräteeinheiten

14.27 csurface:SetDeviceScale

BEZEICHNUNG

csurface:SetDeviceScale – Stellt die Geräteskalierung ein

ÜBERSICHT

```
csurface:SetDeviceScale(x_scale, y_scale)
```

BESCHREIBUNG

Legt einen Maßstab fest, der mit den Gerätekoordinaten multipliziert wird, die vom CTM beim Zeichnen auf die Oberfläche ermittelt werden. Eine häufige Verwendung hierfür ist

das Rendern auf Anzeigegeräten mit sehr hoher Auflösung mit einem Skalierungsfaktor, sodass Code, der davon ausgeht, dass 1 Pixel eine bestimmte Größe hat, weiterhin funktioniert. Das Festlegen einer Transformation über ccontext:Scale() reicht hierfür nicht aus, da Funktionen wie ccontext:DeviceToUser() die verborgene Skala offenlegen.

Beachten Sie, dass sich der Maßstab sowohl auf das Zeichnen auf die Oberfläche als auch auf die Verwendung der Oberfläche in einem Quellmuster auswirkt.

EINGABEN

`x_scale` ein Skalierungsfaktor in X-Richtung

`y_scale` ein Skalierungsfaktor in Y-Richtung

14.28 csurface:SetDocumentUnit

BEZEICHNUNG

csurface:SetDocumentUnit – Stellt die Dokumenteinheit ein

ÜBERSICHT

```
csurface:SetDocumentUnit(unit)
```

BESCHREIBUNG

Verwendet die angegebene Einheit für die Breite und Höhe der generierten SVG-Datei. Unter csurface:GetDocumentUnit() finden Sie eine Liste der verfügbaren Einheitenwerte, die hier verwendet werden können.

Diese Funktion kann jederzeit vor dem Generieren der SVG-Datei aufgerufen werden.

Um jedoch das Risiko von Mehrdeutigkeiten zu minimieren, wird empfohlen, sie aufzurufen, bevor Zeichenvorgänge auf der gegebenen Oberfläche ausgeführt wurden, um klarer zu machen, welche Einheit bei den Zeichenvorgängen verwendet wird.

Der einfachste Weg, dies zu tun, besteht darin, diese Funktion direkt nach dem Erstellen der SVG-Oberfläche aufzurufen.

Beachten Sie, dass die Standardeinheit für von Cairo generierte SVG-Dokumente die Benutzereinheit ist, wenn diese Funktion nie aufgerufen wird.

EINGABEN

`unit` SVG-Einheit

14.29 csurface:SetFallbackResolution

BEZEICHNUNG

csurface:SetFallbackResolution – Legt die Fallback-Auflösung fest

ÜBERSICHT

```
csurface:SetFallbackResolution(x_pixels_per_inch, y_pixels_per_inch)
```

BESCHREIBUNG

Legt die horizontale und vertikale Auflösung für Bild-Fallbacks fest.

Wenn bestimmte Vorgänge nicht nativ von einem Backend unterstützt werden, greift Cairo zurück, indem es Vorgänge auf ein Bild rendert und dieses Bild dann über die Ausgabe legt. Für Backends, die nativ vektororientiert sind, kann diese Funktion verwendet werden, um die Auflösung festzulegen, die für diese Bild-Fallbacks verwendet wird (größere Werte führen zu detaillierteren Bildern, aber auch zu größeren Dateigrößen).

Einige Beispiele für nativ vektororientierte Backends sind die Backends ps, pdf und svg.

Für Backends, die nativ rasterorientiert sind, sind Bild-Fallbacks weiterhin möglich, sie werden jedoch immer mit der nativen Geräteauflösung durchgeführt. Diese Funktion hat also keine Auswirkung auf diese Backends.

Hinweis: Die Fallback-Auflösung wird erst wirksam, wenn eine Seite fertiggestellt wird (mit ccontext:ShowPage() oder ccontext:CopyPage()), daher gibt es derzeit keine Möglichkeit, mehr als eine Fallback-Auflösung auf einer einzigen Seite zu haben.

Die standardmäßige Fallback-Auflösung beträgt in beiden Dimensionen 300 Pixel pro Zoll.

EINGABEN

`x_pixels_per_inch`
: horizontale Einstellung für Pixel pro Zoll

`y_pixels_per_inch`
: vertikale Einstellung für Pixel pro Zoll

14.30 csurface:SetMetadata

BEZEICHNUNG

csurface:SetMetadata – Legt die Metadaten für die PDF-Oberfläche fest

ÜBERSICHT

```
csurface:SetMetadata(metadata, s$)
```

BESCHREIBUNG

Legt Dokumentmetadaten für die PDF-Oberfläche fest. Die Werte `#CAIRO_PDF_METADATA_CREATE_DATE` und `#CAIRO_PDF_METADATA_MOD_DATE` müssen im ISO-8601-Format vorliegen: JJJJ-MM-TTThh:mm:ss. Eine optionale Zeitzone der Form `"[+/-]hh:mm"` oder `"Z"` für UTC-Zeit kann angehängt werden. Alle anderen Metadatenwerte können beliebige Zeichenketten sein.

Der Parameter `metadata` kann eine der folgenden Konstanten sein:

`#CAIRO_PDF_METADATA_TITLE`
: Der Dokumenttitel

`#CAIRO_PDF_METADATA_AUTHOR`
: Der Dokumentautor

`#CAIRO_PDF_METADATA_SUBJECT`
: Der Betreff des Dokuments

`#CAIRO_PDF_METADATA_KEYWORDS`
: Die Schlüsselwörter des Dokuments

`#CAIRO_PDF_METADATA_CREATOR`
: Der Dokumentersteller

`#CAIRO_PDF_METADATA_CREATE_DATE`
: Das Erstellungsdatum des Dokuments

`#CAIRO_PDF_METADATA_MOD_DATE`
: Das Änderungsdatum des Dokuments

Zum Beispiel:

```
surface:SetMetadata(#CAIRO_PDF_METADATA_TITLE, "My Document")
surface:SetMetadata(#CAIRO_PDF_METADATA_CREATE_DATE, "2015-12-31T23:59+02:00")
```

EINGABEN

`metadata` das festzulegende Metadatenelement (siehe oben).

`s$` Metadatenwert

14.31 csurface:SetMimeData

BEZEICHNUNG

csurface:SetMimeData – Legt die Mime-Daten fest

ÜBERSICHT

```
status = csurface:SetMimeData(mime_type[, data$])
```

BESCHREIBUNG

Hängt ein Bild im Format `mime_type` an die Oberfläche an. Um die Daten von einer Oberfläche zu entfernen, rufen Sie diese Funktion auf, ohne das Argument `data$` anzugeben.

Die angehängten Bild- (oder Dateinamen-)Daten können später von Backends, die sie unterstützen (derzeit: PDF-, PS-, SVG- und Win32-Druckoberflächen), verwendet werden, um diese Daten auszugeben, anstatt einen Schnappschuss der Oberfläche zu erstellen. Dieser Ansatz ist tendenziell schneller und erfordert weniger Arbeitsspeicher und Speicherplatz.

Die erkannten MIME-Typen sind die folgenden: `#CAIRO_MIME_TYPE_JPEG`, `#CAIRO_MIME_TYPE_PNG`, `#CAIRO_MIME_TYPE_JP2`, `#CAIRO_MIME_TYPE_URI`, `#CAIRO_MIME_TYPE_UNIQUE_ID`, `#CAIRO_MIME_TYPE_JBIG2`, `#CAIRO_MIME_TYPE_JBIG2_GLOBAL`, `#CAIRO_MIME_TYPE_JBIG2_GLOBAL_ID`, `#CAIRO_MIME_TYPE_CCITT_FAX`, `#CAIRO_MIME_TYPE_CCITT_FAX_PARAMS`.

Einzelheiten darüber, welche MIME-Typen verarbeitet werden können, finden Sie in den entsprechenden Backend-Oberflächendokumenten. Achtung: Die zugehörigen MIME-Daten werden verworfen, wenn Sie anschließend auf der Oberfläche zeichnen. Verwenden Sie diese Funktion mit Vorsicht.

Auch wenn ein Backend einen MIME-Typ unterstützt, bedeutet das nicht, dass Cairo immer die angehängten MIME-Daten verwenden kann. Wenn das Backend beispielsweise den Compositing-Vorgang, der zum Anwenden der MIME-Daten auf das Backend verwendet wird, nicht nativ unterstützt. In diesem Fall werden die MIME-Daten ignoriert.

Um in allen Fällen ein Bild anzuwenden, ist es daher am besten, eine Bildoberfläche zu erstellen, die die dekodierten Bilddaten enthält, und dann die MIME-Daten daran anzuhängen. Dadurch wird sichergestellt, dass das Bild immer verwendet wird, während die MIME-Daten nach Möglichkeit weiterhin verwendet werden können.

Diese Funktion gibt bei Erfolg `#CAIRO_STATUS_SUCCESS` oder `#CAIRO_STATUS_NO_MEMORY` zurück, wenn kein Slot für die Benutzerdaten zugewiesen werden konnte.

EINGABEN

`mime_type`
der MIME-Typ der Bilddaten (siehe oben).

`data$` Optional: die Bilddaten, die an die Oberfläche angehängt werden sollen

RÜCKGABEWERTE

`status` ein Cairo-Statuscode

14.32 csurface:SetPageLabel

BEZEICHNUNG

csurface:SetPageLabel – Legt die Seitenbeschriftung fest

ÜBERSICHT

```
csurface:SetPageLabel(s$)
```

BESCHREIBUNG

Legt die Seitenbeschriftung für die aktuelle Seite einer PDF-Oberfläche fest.

EINGABEN

`s$` die gewünschte Seitenbeschriftung

14.33 csurface:SetSize

BEZEICHNUNG

csurface:SetSize – Legt die PDF-Oberflächengröße fest

ÜBERSICHT

```
csurface:SetSize(width_in_points, height_in_points)
```

BESCHREIBUNG

Ändert die Größe einer PDF-Oberfläche für die aktuelle (und nachfolgende) Seite.

Diese Funktion sollte nur aufgerufen werden, bevor Zeichenvorgänge auf der aktuellen Seite ausgeführt wurden. Der einfachste Weg, dies zu tun, besteht darin, diese Funktion unmittelbar nach dem Erstellen der Oberfläche oder unmittelbar nach dem Fertigstellen einer Seite mit entweder ccontext:ShowPage() oder ccontext:CopyPage() aufzurufen.

EINGABEN

`width_in_points`
neue Oberflächenbreite in Punkten (1 Punkt == 1/72,0 Zoll)

`height_in_points`
: neue Oberflächenhöhe in Punkten (1 Punkt == 1/72,0 Zoll)

14.34 csurface:SetThumbnailSize

BEZEICHNUNG

csurface:SetThumbnailSize – Stellt die Miniaturbildgröße ein

ÜBERSICHT

```
csurface:SetThumbnailSize(width, height)
```

BESCHREIBUNG

Legt die Miniaturbildgröße für die aktuelle und alle nachfolgenden Seiten einer PDF-Oberfläche fest. Wenn Sie eine Breite oder Höhe von 0 festlegen, werden Miniaturansichten für die aktuelle und die folgenden Seiten deaktiviert.

EINGABEN

`width` Miniaturbildbreite

`height` Miniaturbildhöhe

14.35 csurface:ShowPage

BEZEICHNUNG

csurface:ShowPage – Zeigt die Seite an

ÜBERSICHT

```
csurface:ShowPage()
```

BESCHREIBUNG

Gibt die aktuelle Seite für Backends aus, die mehrere Seiten unterstützen, und löscht sie. Verwenden Sie csurface:CopyPage(), wenn Sie die Seite nicht löschen möchten.

Hierfür gibt es eine praktische Funktion, die einen Cairo-Kontext benötigt, nämlich ccontext:ShowPage().

EINGABEN

Keine

14.36 csurface:Status

BEZEICHNUNG

csurface:Status – Gibt den Oberflächenstatus zurück

ÜBERSICHT

```
status = csurface:Status()
```

BESCHREIBUNG

Überprüft, ob für diese Oberfläche zuvor ein Fehler aufgetreten ist.

Diese Funktion gibt

```
#CAIRO_STATUS_SUCCESS
#CAIRO_STATUS_NULL_POINTER
#CAIRO_STATUS_NO_MEMORY
#CAIRO_STATUS_READ_ERROR
#CAIRO_STATUS_INVALID_CONTENT
#CAIRO_STATUS_INVALID_FORMAT
#CAIRO_STATUS_INVALID_VISUAL
```

zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`status` ein Cairo-Statuswert (siehe oben)

14.37 csurface:SupportsMimeType

BEZEICHNUNG

csurface:SupportsMimeType – Unterstützt den Mime-Typ

ÜBERSICHT

```
ok = csurface:SupportsMimeType(mime_type)
```

BESCHREIBUNG

Gibt zurück, ob die Oberfläche `mime_type` unterstützt. Eine Liste der MIME-Typen finden Sie unter csurface:SetMimeData().

Diese Funktion gibt `True` zurück, wenn die Oberfläche `mime_type` unterstützt, andernfalls `False`.

EINGABEN

`mime_type`
der Mime-Typ

RÜCKGABEWERTE

`ok` boolescher Wert, der angibt, ob die Oberfläche `mime_type` unterstützt

14.38 csurface:ToBrush

BEZEICHNUNG

csurface:ToBrush – Konvertiert die Oberfläche in Hollywood-Pinsel

ÜBERSICHT

```
[id] = csurface:ToBrush(brush[, color])
```

BESCHREIBUNG

Konvertiert die Bildoberfläche in den durch `brush` angegebenen Hollywood-Pinsel. Wenn Sie `Nil` in `brush` übergeben, wird die automatische ID-Auswahl verwendet und diese Funktion gibt die ID des Pinsels zurück. Das optionale Argument `color` wird nur

verwendet, wenn die Bildoberfläche keine Farbkanäle hat (z.B. wenn das Pixelformat `#CAIRO_FORMAT_A8` ist). In diesem Fall werden die Farbkanäle des Pinsels mit der in `color` angegebenen Farbe gefüllt.

EINGABEN

`brush` Kennung des zu erstellenden Pinsels oder Null für die automatische ID-Auswahl

`color` Optional: Füllfarbe, die verwendet werden soll, wenn die Oberfläche keine Farbkanäle hat

RÜCKGABEWERTE

`id` Optional: Hanlde zum neuen Pinsel; wird nur zurückgegeben, wenn Sie `Nil` in `brush` angegeben haben

14.39 csurface:WriteToPNG

BEZEICHNUNG

csurface:WriteToPNG – Schreibt die Oberfläche in ein PNG-Bild

ÜBERSICHT

```
status = csurface:WriteToPNG(filename$)
```

BESCHREIBUNG

Schreibt den Inhalt der Oberfläche als PNG-Bild in eine neue Datei, die durch `filename$` angegeben wird.

Diese Funktion gibt `#CAIRO_STATUS_SUCCESS` zurück, wenn die PNG-Datei erfolgreich geschrieben wurde. Andernfalls:

`#CAIRO_STATUS_NO_MEMORY`
Wenn für den Vorgang kein Speicher zugewiesen werden konnte.

`#CAIRO_STATUS_SURFACE_TYPE_MISMATCH`
Wenn die Oberfläche keine Pixelinhalte hat.

`#CAIRO_STATUS_WRITE_ERROR`
Wenn beim Versuch, die Datei zu schreiben, ein E/A-Fehler auftritt.

`#CAIRO_STATUS_PNG_ERROR`
Wenn libpng einen Fehler zurückgegeben hat.

EINGABEN

`filename$`
Dateiname des PNG-Bildes

RÜCKGABEWERTE

`status` ein Cairo-Statuswert (siehe oben).

15 Pango-Funktionen

15.1 pango.Attribute

BEZEICHNUNG

pango.Attribute – Erstellt ein Pango-Attribut

ÜBERSICHT

```
attr = pango.Attribute(type$[, ...])
```

BESCHREIBUNG

Erstellt ein neues Pango-Attribut des angegebenen `type$`. Die anderen Parameter, die an diese Funktion übergeben werden müssen, hängen vom angegebenen Typ ab.

Die folgenden Typen werden derzeit unterstützt:

`AllowBreaks`
: Erstellt ein neues Attribut "Pausen zulassen". Sie müssen ein zusätzliches boolesches Argument übergeben, das angibt, ob Pausen zulässig sind. Bei deaktivierten Pausen wird die Reichweite soweit wie möglich in einem einzigen Lauf gehalten.

`Background`
: Erstellt ein neues Hintergrundfarbattribut. Sie müssen die drei zusätzlichen Argumente übergeben, die die roten, grünen und blauen Komponenten der Hintergrundfarbe angeben und jeweils zwischen 0 und 65535 liegen müssen.

`BackgroundAlpha`
: Erstellt ein neues Alpha-Hintergrund-Attribut. Sie müssen das zusätzliche Argument `alpha` übergeben, das den Alpha-Hintergrundwert angibt und zwischen 0 und 65535 liegen muss.

`Fallback`
: Erstellt ein neues Schriftarten-Fallback-Attribut. Sie müssen ein zusätzliches boolesches Argument übergeben, das angibt, ob Fallback aktiviert oder deaktiviert werden soll. Wenn das Fallback deaktiviert ist, werden nur Zeichen aus der am besten passenden Schriftart im System verwendet. Es wird kein Fallback auf andere Schriftarten im System durchgeführt, die möglicherweise die Zeichen im Text enthalten.

`Family`
: Erstellt ein neues Schriftfamilienattribut. Sie müssen eine Zeichenkette übergeben, die den Familiennamen oder eine durch Kommas getrennte Liste von Familien enthält.

`FontDesc`
: Erstellt ein neues Schriftbeschreibungsattribut. Als zusätzliches Argument benötigen Sie ein Pango-Schriftbeschreibungsobjekt. Dieses Attribut ermöglicht das gleichzeitige Setzen der Attribute `Family`, `Style`, `Weight`, `Variant`, `Stretch` und `Size`.

`Foreground`
: Erstellt ein neues Vordergrundfarbattribut. Sie müssen die drei zusätzlichen Argumente übergeben, die die roten, grünen und blauen Komponenten der Vordergrundfarbe angeben und jeweils zwischen 0 und 65535 liegen müssen.

`ForegroundAlpha`
: Erstellt ein neues Alpha-Vordergrund-Attribut. Sie müssen einen zusätzlichen Wert übergeben, der den Alpha-Vordergrundwert angibt und zwischen 0 und 65535 liegen muss.

`Gravity`
: Erstellt ein neues Ausrichtungsattribut. Sie müssen ein zusätzliches Argument übergeben, das die Ausrichtung angibt. Eine Liste der Ausrichtungskonstanten finden Sie unter pcontext:SetBaseGravity().

`GravityHint`
: Erstellt ein neues Ausrichtungshinweis-Attribut. Sie müssen ein zusätzliches Argument übergeben, das den Ausrichtungshinweis angibt. Eine Liste mit Hinweisen zur Ausrichtung finden Sie unter pcontext:SetGravityHint().

`InsertHyphens`
: Erstellt ein neues Attribut "Bindestriche einfügen". Pango fügt Bindestriche ein, wenn Zeilen in der Mitte eines Wortes umgebrochen werden. Setzen Sie dieses Attribut auf `False`, um den Bindestrich zu unterdrücken.

`Language`
: Erstellt ein neues Sprach-Tag-Attribut. Sie müssen ein zusätzliches Argument übergeben, das ein Pango-Sprachobjekt enthält.

`LetterSpacing`
: Erstellt ein neues Attribut für den Buchstabenabstand. Sie müssen ein zusätzliches Argument übergeben, das den zusätzlichen Abstand angibt, der zwischen den Graphem des Textes hinzugefügt werden soll (in Pango-Einheiten).

`Rise`
: Erstellt ein neues Attribut für die Grundlinienverschiebung. Sie müssen ein zusätzliches Argument übergeben, das den Betrag angibt, um den der Text vertikal verschoben werden soll, in Pango-Einheiten. Positive Werte verschieben den Text nach oben.

`Scale`
: Erstellt ein neues Skalierungsattribut für die Schriftgröße. Die Größe der Basisschriftart für den betroffenen Text wird mit dem Skalierungsfaktor multipliziert, der als zusätzliches Argument an dieses Attribut übergeben wird.

`Shape`
: Erstellt ein neues Formattribut. Eine Form wird verwendet, um dem Ergebnis der Formung einer bestimmten Glyphe eine bestimmte Einfärbung und ein logisches Rechteck aufzuerlegen. Dies kann beispielsweise zum Einbetten eines Bildes oder eines Widgets in ein Pango-Layout verwendet werden. Sie müssen zwei zusätzliche Tabellenargumente übergeben: Das erste ist das Einfärberechteck, das jedem Zeichen zugewiesen wird, das zweite ist das logische Rechteck, das jedem Zeichen zugewiesen wird. Beide Argumente müssen Tabellen mit den initialisierten Feldern `x`, `y`, `width` und `height` sein. Darüber hinaus können Sie ein optionales drittes Argument übergeben, das als Benutzerdaten gilt und einen beliebigen Typ haben kann.

`Show`
: Erstellt ein neues Attribut, das beeinflusst, wie unsichtbare Zeichen dargestellt werden. Sie müssen ein zusätzliches Argument übergeben, das auf eines der folgenden Flags gesetzt werden kann:

`#PANGO_SHOW_NONE`
: Keine Sonderbehandlung für unsichtbare Zeichen.

`#PANGO_SHOW_SPACES`
: Macht Leerzeichen, Tabulatoren und Zeilenumbrüche sichtbar.

`#PANGO_SHOW_LINE_BREAKS`
: Macht Zeilenumbrüche sichtbar.

`#PANGO_SHOW_IGNORABLES`
: Macht standardmäßig ignorierbare Unicode-Zeichen sichtbar.

`Size`
: Erstellt ein neues Schriftgrößenattribut in Bruchteilen. Sie müssen ein zusätzliches Argument übergeben, das die Schriftgröße in #PANGO_SCALE-teln eines Punktes angibt.

`SizeAbsolute`
: Erstellt ein neues Schriftgrößenattribut in Gerätepunkten. Sie müssen ein zusätzliches Argument übergeben, das die Schriftgröße in #PANGO_SCALE-teln einer Geräteeinheit angibt.

`Stretch`
: Erstellt ein neues Attribut zur Dehnung der Schriftart. Sie müssen ein zusätzliches Argument übergeben, das den Schriftart-Dehnungsmodus angibt. Eine Liste der Pango-Dehnungsmodi finden Sie unter pfontdesc:SetStretch().

`Strikethrough`
: Erstellt ein neues Durchgestrichen-Attribut. Sie müssen ein zusätzliches boolesches Argument übergeben, das angibt, ob der Text durchgestrichen werden soll.

`StrikethroughColor`
: Erstellt ein neues durchgestrichenes Farbattribut. Sie müssen die drei zusätzlichen Argumente `red`, `green`, `blue` übergeben, die die durchgestrichene Farbe angeben und für jeden Wert zwischen 0 und 65535 liegen müssen.

`Style`
: Erstellt ein neues Stilattribut für die Schriftartschräge. Sie müssen ein zusätzliches Argument übergeben, das den Schriftschnittstil angibt. Eine Liste der Pango-Stile finden Sie unter pfontdesc:SetStyle().

`Underline`
: Erstellt ein neues Attribut im Unterstreichungsstil. Sie müssen ein zusätzliches Argument übergeben, das den Unterstreichungsstil angibt. Die folgenden Unterstreichungsstile werden derzeit unterstützt:

  `#PANGO_UNDERLINE_NONE`
  : Es sollte keine Unterstreichung gezeichnet werden.

  `#PANGO_UNDERLINE_SINGLE`
  : Es sollte eine einzelne Unterstreichung gezeichnet werden.

  `#PANGO_UNDERLINE_DOUBLE`
  : Es sollte eine doppelte Unterstreichung gezeichnet werden.

`#PANGO_UNDERLINE_LOW`
: Eine einzelne Unterstreichung sollte an einer Position unterhalb der Freihandbereiche des zu unterstreichenden Textes gezeichnet werden. Dies sollte nur zum Unterstreichen einzelner Zeichen verwendet werden, beispielsweise für Tastaturkürzel. `#PANGO_UNDERLINE_SINGLE` sollte für längere Textteile verwendet werden.

`#PANGO_UNDERLINE_ERROR`
: Unten sollte ein Unterstrich gezeichnet werden, der auf einen Fehler hinweist. Der genaue Darstellungsstil hängt vom verwendeten PangoRenderer ab, typische Stile umfassen jedoch wellenförmige oder gepunktete Linien. Diese Unterstreichung wird normalerweise verwendet, um auf einen Fehler wie etwa einen möglichen Schreibfehler hinzuweisen. In einigen Fällen kann automatisch eine Kontrastfarbe verwendet werden.

`#PANGO_UNDERLINE_SINGLE_LINE`
: Wie PANGO_UNDERLINE_SINGLE, jedoch kontinuierlich über mehrere Läufe hinweg gezeichnet.

`#PANGO_UNDERLINE_DOUBLE_LINE`
: Wie PANGO_UNDERLINE_DOUBLE, jedoch kontinuierlich über mehrere Läufe hinweg gezeichnet.

`#PANGO_UNDERLINE_ERROR_LINE`
: Wie PANGO_UNDERLINE_ERROR, aber kontinuierlich über mehrere Läufe hinweg gezeichnet.

`UnderlineColor`
: Erstellt ein neues Unterstreichungsfarbattribut. Sie müssen die drei zusätzlichen Argumente `red`, `green`, `blue` übergeben, die die Unterstreichungsfarbe angeben und für jeden Wert zwischen 0 und 65535 liegen müssen.

`Variant` Erstellt ein neues Schriftartvariantenattribut. Sie müssen ein zusätzliches Argument übergeben, das den Stil der Schriftartvariante angibt. Eine Liste der Pango-Varianten finden Sie unter pfontdesc:SetVariant().

`Weight` Erstellt ein neues Attribut für die Schriftstärke. Sie müssen ein zusätzliches Argument übergeben, das die Schriftstärke angibt. Dies kann eine Zahl oder eine der vordefinierten Schriftstärken sein. Eine Liste der Pango-Schriftstärken finden Sie unter pfontdesc:SetWeight().

Diese Funktion gibt das neu zugewiesene Pango-Attribut zurück, das mit pattribute:Free() gelöscht werden sollte.

EINGABEN

Keine

RÜCKGABEWERTE

`attr` das neu zugewiesene Pango-Attribut

15.2 pango.AttrList

BEZEICHNUNG
pango.AttrList – Erstellt eine Attributliste

ÜBERSICHT
```
handle = pango.AttrList()
```

BESCHREIBUNG
Erstellt eine neue leere Attributliste mit einer Referenzanzahl von eins.

Diese Funktion gibt die neu zugewiesene Pango-Attributliste zurück, die mit pattrlist:Free gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` die neu zugewiesene Attributliste

15.3 pango.Context

BEZEICHNUNG
pango.Context – Erstellt einen Pango-Kontext

ÜBERSICHT
```
handle = pango.Context()
```

BESCHREIBUNG
Erstellt einen neuen Pango-Kontext, der mit Standardwerten initialisiert wird.

Diese Funktion ist nicht besonders nützlich, da ihr immer ein pcontext:SetFontMap()-Aufruf folgen sollte und die Funktion pfontmap:CreateContext() diese beiden Schritte zusammen ausführt. Benutzern wird daher empfohlen, diese zu verwenden.

Diese Funktion gibt den neu zugewiesenen Pango-Kontext zurück, der mit pcontext:Free() gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` der neu zugewiesene Pango-Kontext

15.4 pango.Coverage

BEZEICHNUNG
pango.Coverage – Erstellt eine Pango-Zeichenbereichsabdeckung

ÜBERSICHT
```
handle = pango.Coverage()
```

BESCHREIBUNG

Erstellt eine neue Pango-Zeichenbereichsabdeckung.

Diese Funktion gibt die neu zugewiesene Pango-Zeichenbereichsabdeckung zurück, initialisiert auf `#PANGO_COVERAGE_NONE` mit einem Referenzzähler von eins, die mit pcoverage:Free() gelöscht werden sollte.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die neu zugewiesene Pango-Zeichenbereichsabdeckung

15.5 pango.ExtentsToPixels

BEZEICHNUNG

pango.ExtentsToPixels – Konvertiert die Ausmaße von Pango- in Geräteeinheiten

ÜBERSICHT

```
r1, r2 = pango.ExtentsToPixels(inclusive, nearest)
```

BESCHREIBUNG

Konvertiert Ausmaße von Pango-Einheiten in Geräteeinheiten. Die Umrechnung erfolgt durch Division durch den #PANGO_SCALE-Faktor und Rundung. Die Argumente `inclusive` und `nearest` müssen Tabellen sein, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind. Wenn Sie eines der Rechtecke nicht benötigen, können Sie es auch auf Null setzen.

Das Rechteck `inklusive` wird konvertiert, indem die x/y-Koordinaten auf den Boden gelegt und um die Breite/Höhe erweitert werden, sodass das endgültige Rechteck das ursprüngliche Rechteck vollständig enthält.

Das `nearest` Rechteck wird konvertiert, indem die Koordinaten des Rechtecks auf die nächste Geräteeinheit (Pixel) gerundet werden.

Die Regel für das zu verwendende Argument lautet: Wenn Sie möchten, dass das resultierende Rechteck des Gerätebereichs das ursprüngliche Rechteck vollständig enthält, übergeben Sie es als `inklusive`. Wenn Sie möchten, dass zwei sich berührende, aber nicht überlappende Rechtecke sich nach dem Runden auf Geräteeinheiten weiterhin berühren, aber nicht überlappen, übergeben Sie sie als `nearest`.

EINGABEN

`inclusive` Rechteck zum Runden auf einschließlich Pixel oder Null

`nearest` Rechteck zum Runden auf die nächsten Pixel oder Null

RÜCKGABEWERTE

`r1` abgerundetes inklusives Rechteck

`r2` abgerundetes nächstgelegenes Rechteck

15.6 pango.FontDescription

BEZEICHNUNG

pango.FontDescription – Erstellt eine Schriftbeschreibung

ÜBERSICHT

```
desc = pango.FontDescription([s$])
```

BESCHREIBUNG

Erstellt eine neue Schriftartbeschreibung. Wenn der Parameter `s$` weggelassen wird, werden alle Felder auf Standardwerte initialisiert. Andernfalls wird die Schriftartbeschreibung gemäß der Zeichenkettenspezifikation initialisiert. Falls angegeben, muss `s$` diese Form haben:

```
[FAMILY_LIST] [STYLE_OPTIONS] [SIZE] [VARIATIONS]
```

Dabei ist `FAMILY_LIST` eine durch Kommas getrennte Liste von Familien, die optional durch ein Komma abgeschlossen werden kann, `STYLE_OPTIONS` ist eine durch Leerzeichen getrennte Liste von Wörtern, wobei jedes Wort Stil, Variante, Schriftstärke, Dehnung oder Ausrichtung beschreibt und `SIZE` eine Dezimalzahl ist (Größe in Punkten) oder optional gefolgt vom Einheitenmodifikator "px" für absolute Größe. `VARIATIONS` ist eine durch Kommas getrennte Liste von Schriftartvariationsspezifikationen der Form "axis=value" (das =-Zeichen ist optional).

Die folgenden Wörter werden als Stile verstanden:

```
"Normal"
"Roman"
"Oblique"
"Italic"
```

Die folgenden Wörter gelten als Varianten:

```
"Small-Caps"
"All-Small-Caps"
"Petite-Caps"
"All-Petite-Caps"
"Unicase"
"Title-Caps"
```

Die folgenden Wörter sind für die Schriftstärke zugelassen:

```
"Thin"
"Ultra-Light"
"Extra-Light"
"Light"
"Semi-Light"
"Demi-Light"
"Book"
"Regular"
"Medium"
"Semi-Bold"
"Demi-Bold"
"Bold"
"Ultra-Bold"
```

```
"Extra-Bold"
"Heavy"
"Black"
"Ultra-Black"
"Extra-Black"
```

Die folgenden Wörter können für Dehnungswerte verwendet werden:

```
"Ultra-Condensed"
"Extra-Condensed"
"Condensed"
"Semi-Condensed"
"Semi-Expanded"
"Expanded"
"Extra-Expanded"
"Ultra-Expanded"
```

Die folgenden Wörter werden als Ausrichtungswerte verstanden:

```
"Not-Rotated"
"South"
"Upside-Down"
"North"
"Rotated-Left"
"East"
"Rotated-Right"
"West"
```

Eine der Optionen kann fehlen. Wenn `FAMILY_LIST` fehlt, wird das Feld `FamilyName` der resultierenden Schriftartenbeschreibung mit Null initialisiert. Wenn `STYLE_OPTIONS` fehlt, werden alle Stiloptionen auf die Standardwerte gesetzt. Fehlt `SIZE`, wird die Größe in der resultierenden Schriftbeschreibung auf 0 gesetzt.

Ein typisches Beispiel:

```
Cantarell Italic Light 15 wght=200
```

EINGABEN

`s$` Optional: eine Zeichenkette, die die zu ladende Schriftart beschreibt

RÜCKGABEWERTE

`desc` eine neue Pango-Schriftbeschreibung

15.7 pango.FontMap

BEZEICHNUNG

pango.FontMap – Erstellt einen Zeichensatz

ÜBERSICHT

```
handle = pango.FontMap([fonttype])
```

BESCHREIBUNG

Erstellt ein neues Zeichensatz-Objekt.

Ein Zeichensatz wird zum Zwischenspeichern von Informationen über verfügbare Schriftarten verwendet und enthält bestimmte globale Parameter wie die Auflösung. In den meisten Fällen können Sie stattdessen pango.GetDefaultFontMap() verwenden.

Beachten Sie, dass der Typ des zurückgegebenen Objekts von der jeweiligen Schriftart abhängt, für die Cairo kompiliert wurde. Alternativ können Sie auch das optionale Argument `fonttype` übergeben, um explizit ein zu verwendendes Backend anzugeben. Eine Liste der Schriftarten finden Sie unter cfontface:GetType().

Diese Funktion gibt die neu zugewiesene Pango-Schriftartzuordnung zurück, die mit pfontmap:Free() gelöscht werden sollte.

EINGABEN

`fonttype` Optional: der zu verwendende Cairo-Schrifttyp

RÜCKGABEWERTE

`handle` die neu zugewiesene Pango-Zeichensatz

15.8 pango.GetDefaultFontMap

BEZEICHNUNG

pango.GetDefaultFontMap – Gibt die Standardschriftartzuordnung zurück

ÜBERSICHT

```
handle = pango.GetDefaultFontMap()
```

BESCHREIBUNG

Ruft eine Standardschriftartzuordnung zur Verwendung mit Cairo ab.

Beachten Sie, dass der Typ des zurückgegebenen Objekts von der jeweiligen Schriftart abhängt, für die Cairo kompiliert wurde.

Die standardmäßige Cairo-Schriftartzuordnung kann mithilfe von pango.SetDefaultFontMap() geändert werden. Dies kann beispielsweise verwendet werden, um das Cairo-Schriftarten-Backend zu ändern, das die Standard-Schriftartzuordnung verwendet.

Das zurückgegebene Objekt ist Eigentum von Pango und darf nicht gelöscht werden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die standardmäßige Pango-Schriftartzuordnung

15.9 pango.GetDefaultLanguage

BEZEICHNUNG

pango.GetDefaultLanguage – Gibt die Standardsprache zurück

ÜBERSICHT

```
lang = pango.GetDefaultLanguage()
```

BESCHREIBUNG

Gibt die Pango-Sprache für das aktuelle Gebietsschema des Prozesses zurück. Beachten Sie, dass sich die Standardsprache im Laufe der Lebensdauer einer Anwendung ändern kann.

EINGABEN

Keine

RÜCKGABEWERTE

`lang` die Standardsprache als Pango-Sprache

15.10 pango.GlyphString

BEZEICHNUNG

pango.GlyphString – Erstellt eine Pango-Glyphen-Zeichenkette

ÜBERSICHT

```
gstr = pango.GlyphString()
```

BESCHREIBUNG

Erstellt eine neue Pango-Glyphen-Zeichenkette.

Diese Funktion gibt die neu zugewiesene Pango-Glyphen-Zeichenkette zurück, die mit pglyphstring:Free() gelöscht werden sollte.

EINGABEN

Keine

RÜCKGABEWERTE

`gstr` die neu zugewiesene Pango-Glyphen-Zeichenkette

15.11 pango.GravityForMatrix

BEZEICHNUNG

pango.GravityForMatrix – Gibt die Ausrichtung für die Matrix zurück

ÜBERSICHT

```
gravity = pango.GravityForMatrix(matrix)
```

BESCHREIBUNG

Ermittelt die Ausrichtung, die am besten zur Ausrichtungskomponente in einer Pango-Matrix passt.

Diese Funktion gibt die Ausrichtung von `matrix` zurück, die niemals `#PANGO_GRAVITY_AUTO` sein wird.

Siehe pcontext:SetBaseGravity() für eine Liste der Ausrichtungskonstanten.

EINGABEN

`matrix` Pango-Matrix

RÜCKGABEWERTE

`gravity` die Ausrichtung von `matrix`

15.12 pango.GravityForScript

BEZEICHNUNG

pango.GravityForScript – Gibt die Ausrichtung für das Skript zurück

ÜBERSICHT

```
gravity = pango.GravityForScript(script, base_gravity, hint)
```

BESCHREIBUNG

Gibt die Ausrichtung zurück, die beim Auslegen eines Pango-Elements verwendet werden soll. Die Ausrichtung wird anhand des Skripts, der Grundausrichtung und des Hinweises bestimmt.

Wenn `base_gravity` `#PANGO_GRAVITY_AUTO` ist, wird es zunächst durch die bevorzugte Ausrichtung von `script` ersetzt. Um die bevorzugte Ausrichtung eines Skripts zu erhalten, übergeben Sie `#PANGO_GRAVITY_AUTO` und `#PANGO_GRAVITY_HINT_STRONG`.

Diese Funktion gibt die aufgelöste Ausrichtung zurück, die für die Verwendung für einen Textlauf mit `script` geeignet ist.

Siehe pcontext:SetBaseGravity() für eine Liste der Ausrichtungskonstanten. Siehe pcontext:SetGravityHint() für eine Liste mit Hinweisen zur Ausrichtung. Siehe planguage:GetScripts() für eine Liste der Skripte.

EINGABEN

`script` das abzufragende Pango-Skript

`base_gravity`
Grundausrichtung des Absatzes

`hint` Orientierungshinweis

RÜCKGABEWERTE

`gravity` Aufgelöste Ausrichtung, geeignet für die Verwendung für einen Textlauf

15.13 pango.GravityForScriptAndWidth

BEZEICHNUNG

pango.GravityForScriptAndWidth – Gibt die Ausrichtung für Skript und Breite zurück

ÜBERSICHT

```
g = pango.GravityForScriptAndWidth(script, wide, base_gravity, hint)
```

BESCHREIBUNG

Gibt die Ausrichtung zurück, die beim Anordnen von einem einzelnen Zeichen oder Pango-Einheit verwendet werden soll.

Die Ausrichtung wird anhand der Schrift, der ostasiatischen Breite, der Grundausrichtung und des Hinweises bestimmt.

Diese Funktion ähnelt pango.GravityForScript(), außer dass diese Funktion auch zwischen schmalen/halbbreiten und breiten/vollbreiten Zeichen unterscheidet. Breite Zeichen/Zeichen voller Breite stehen immer *aufrecht*, d.h. sie nehmen immer die Grundausrichtung an, während schmale Zeichen/Zeichen voller Breite immer im vertikalen Kontext gedreht werden.

Wenn `base_gravity` `#PANGO_GRAVITY_AUTO` ist, wird es zunächst durch die bevorzugte Ausrichtung von `script` ersetzt.

Diese Funktion gibt die aufgelöste Ausrichtung zurück, die für die Verwendung für einen Textlauf mit `script` und `wide` geeignet ist.

Siehe pcontext:SetBaseGravity() für eine Liste der Ausrichtungskonstanten. Siehe pcontext:SetGravityHint() für eine Liste mit Hinweisen zur Ausrichtung. Siehe planguage:GetScripts() für eine Liste der Skripte.

EINGABEN

`script` das abzufragende Pango-Skript

`wide` `True` für breite Zeichen

`base_gravity`
Grundausrichtung des Absatzes

`hint` Orientierungshinweis

RÜCKGABEWERTE

`g` Aufgelöste Ausrichtung, geeignet für die Verwendung für einen Textlauf

15.14 pango.GravityToRotation

BEZEICHNUNG

pango.GravityToRotation – Konvertiert die Ausrichtung in Rotation

ÜBERSICHT

```
angle = pango.GravityToRotation(gravity)
```

BESCHREIBUNG

Konvertiert einen Pango-Ausrichtungswert in seine natürliche Rotation im Bogenmaß.

Beachten Sie, dass pmatrix:Rotate() den Winkel in Grad und nicht im Bogenmaß annimmt. Um also pmatrix:Rotate() mit der Ausgabe dieser Funktion aufzurufen, sollten Sie sie mit (180 / `#PI`) multiplizieren.

Diese Funktion gibt den Rotationswert zurück, der der `gravity` entspricht.

Siehe pcontext:SetBaseGravity() für eine Liste der Ausrichtungskonstanten.

EINGABEN

`gravity` Die abzufragende Ausrichtung sollte nicht `#PANGO_GRAVITY_AUTO` sein

RÜCKGABEWERTE

`angle` der Rotationswert entspricht der `gravity`

15.15 pango.Item

BEZEICHNUNG

pango.Item – Erstellt eine Pango-Elementstruktur

ÜBERSICHT

```
item = pango.Item()
```

BESCHREIBUNG

Erstellt eine neue Pango-Elementstruktur, die mit Standardwerten initialisiert wird.

Diese Funktion gibt das neu zugewiesene Pango-Element zurück, das mit pitem:Free() gelöscht werden sollte.

EINGABEN

Keine

RÜCKGABEWERTE

`item` das neu zugewiesene Pango-Element

15.16 pango.Language

BEZEICHNUNG

pango.Language – Erstellt eine Sprache aus einer Zeichenkette

ÜBERSICHT

```
lang = pango.Language(tag$)
```

BESCHREIBUNG

Konvertiert ein durch `tag$` angegebenes Sprach-Tag in ein Pango-Sprachobjekt. Das Sprach-Tag muss im RFC-3066-Format vorliegen. Pango-Sprachhandle können effizient kopiert und mit anderen Sprach-Tags verglichen werden.

Diese Funktion kanonisiert zunächst die Zeichenkette, indem sie sie in Kleinbuchstaben umwandelt, '_' auf '-', zuordnet und alle Zeichen außer Buchstaben und '-' entfernt.

Verwenden Sie pango.GetDefaultLanguage(), wenn Sie die Pango-Sprache für das aktuelle Gebietsschema des Prozesses abrufen möchten.

EINGABEN

Keine

RÜCKGABEWERTE

`lang` ein Pango-Sprachobjekt

15.17 pango.Layout

BEZEICHNUNG

pango.Layout – Erstellt ein Pango-Layout

ÜBERSICHT

```
handle = pango.Layout(context)
```

BESCHREIBUNG

Erstellt ein neues Pango-Layoutobjekt mit Attributen, die auf Standardwerte für einen bestimmten Pango-Kontext initialisiert sind.

EINGABEN

`context` ein Pango-Kontext

RÜCKGABEWERTE

`handle` das neu zugewiesene Pango-Layout

15.18 pango.Matrix

BEZEICHNUNG

pango.Matrix – Erstellt eine Matrix

ÜBERSICHT

```
m = pango.Matrix([xx, xy, yx, yy, x0, y0])
```

BESCHREIBUNG

Erstellt eine Matrix und initialisiert optional ihre affine Transformation auf die durch `xx`, `xy`, `yx`, `yy`, `x0`, `y0` angegebenen Koeffizienten. Ausgelassene Koeffizienten werden auf 0 gesetzt.

EINGABEN

`xx`	Optional: xx-Komponente der affinen Transformation (standardmäßig 0)
`xy`	Optional: xy-Komponente der affinen Transformation (standardmäßig 0)
`yx`	Optional: yx-Komponente der affinen Transformation (standardmäßig 0)
`yy`	Optional: yy-Komponente der affinen Transformation (standardmäßig 0)
`x0`	Optional: X-Verschiebungskomponente der affinen Transformation (standardmäßig 0)
`y0`	Optional: Y-Verschiebungskomponente der affinen Transformation (standardmäßig 0)

RÜCKGABEWERTE

`m`	Matrixobjekt

15.19 pango.MatrixIdentity

BEZEICHNUNG

pango.MatrixIdentity – Erstellt eine Identitätsmatrix

ÜBERSICHT

```
m = pango.MatrixIdentity()
```

BESCHREIBUNG

Erstellt eine Matrix und initialisiert ihre affine Transformation in eine Identitätstransformation.

EINGABEN

Keine

RÜCKGABEWERTE

`m` Identitätsmatrixobjekt

15.20 pango.SetDefaultFontMap

BEZEICHNUNG

pango.SetDefaultFontMap – Legt die Standard-Schriftartzuordnung fest

ÜBERSICHT

```
pango.SetDefaultFontMap([fontmap])
```

BESCHREIBUNG

Legt eine Standard-Schriftartzuordnung zur Verwendung mit Cairo fest.

Dies kann beispielsweise verwendet werden, um das Cairo-Schriftart-Backend zu ändern, das die Standard-Schriftartzuordnung verwendet. Der alte Standard-Zeichensatz ist nicht referenziert und es wird auf den neue Zeichensatz verwiesen.

Wenn Sie den Parameter `fontmap` weglassen, wird der aktuelle Standard-Zeichensatz gelöscht und bei Bedarf ein neuer Standard-Zeichensatz mit pango.FontMap() erstellt.

EINGABEN

`fontmap` Optional: Pango-Zeichensatz als Standard festlegen

15.21 pango.SetFontconfig

BEZEICHNUNG

pango.SetFontconfig – Legt den Fontconfig-Parameter fest

ÜBERSICHT

```
pango.SetFontconfig(parm$, val$[, ...])
```

BESCHREIBUNG

Mit dieser Funktion können individuelle Fontconfig-Einstellungen konfiguriert werden. Fontconfig wird von Pango zur Schriftartenverwaltung verwendet. Die folgenden Fontconfig-Einstellungen können derzeit konfiguriert und als `parm` übergeben werden:

`FontDir` Fügt das angegebene Verzeichnis zur Liste der von Fontconfig nach Schriftarten durchsuchten Verzeichnisse hinzu. Wenn Sie `True` als drittes Argument an diese Funktion übergeben, wird die vorhandene Liste der Schriftartenverzeichnisse gelöscht, sodass `val$` das einzige Verzeichnis ist, in dem Fontconfig nach Schriftarten sucht. Wenn Sie das dritte Argument weglassen oder es auf `False` setzen, wird das angegebene Verzeichnis über den vorhandenen Schriftartverzeichnissen hinzugefügt.

`CacheDir` Legt das von Fontconfig verwendete Cache-Verzeichnis fest.

`ConfigDir`

Legt das Verzeichnis fest, in dem Fontconfig nach Konfigurationsdateien sucht und diese speichert.

`ConfigFile`
: Legt die Konfigurationsdatei fest, die Fontconfig verwenden soll.

EINGABEN

`parm$`	zu ändernde Einstellung (gültige Typen siehe oben)
`val$`	neuer Wert für die Fontconfig-Einstellung
`...`	Optional: weitere Parameter abhängig vom in `parm` übergebenen Typ

15.22 pango.Shape

BEZEICHNUNG

pango.Shape – Konvertiert den Text in Glyphen

ÜBERSICHT

```
pango.Shape(text$, analysis, glyphs)
```

BESCHREIBUNG

Wandelt die Zeichen in `text$` in Glyphen um.

Konvertiert anhand eines Textsegments und der entsprechenden von pcontext:Itemize() zurückgegebenen Pango-Analysestruktur die Zeichen in Glyphen. Sie können auch nur eine TeilZeichenkette des Elements von pcontext:Itemize() übergeben.

Es wird empfohlen, stattdessen pango.ShapeFull() zu verwenden, da diese API die Gestaltung von Interaktionen über Textelementgrenzen hinweg ermöglicht.

Beachten Sie, dass die zusätzlichen Attribute in der `analyis`, die von pcontext:Itemize() zurückgegeben wird, Indizes haben, die relativ zum gesamten Absatz sind. Sie müssen daher den Elementversatz von ihren Indizes subtrahieren, bevor Sie diese Funktion aufrufen.

EINGABEN

`text$`	der zu verarbeitende Text
`analysis`	Pango-Analyseobjekt von pcontext:Itemize()
`glyphs`	Pango-Glyphen-Zeichenkettenobjekt, in dem Ergebnisse gespeichert werden sollen

15.23 pango.ShapeFull

BEZEICHNUNG

pango.ShapeFull – Konvertiert den Text in Glyphen

ÜBERSICHT

```
pango.ShapeFull(item_text$, paragraph_text$, analysis, glyphs[, flags])
```

BESCHREIBUNG

Wandelt die Zeichen in `item_text$` in Glyphen um.

Konvertiert anhand eines Textsegments und der entsprechenden von pcontext:Itemize() zurückgegebenen Pango-Analysestruktur die Zeichen in Glyphen. Sie können auch nur eine TeilZeichenkette des Elements von pcontext:Itemize() übergeben.

Dies ähnelt pango.Shape(), außer dass optional auch der vollständige Absatztext als Eingabe verwendet werden kann, der dann zur Durchführung bestimmter interelementübergreifender Gestaltungsinteraktionen verwendet wird. Wenn Sie Zugriff auf den umfassenderen Text haben, zu dem `item_text$` gehört, geben Sie den umfassenderen Text als `paragraph_text$` an. Wenn `paragraph_text$` Null ist, wird stattdessen der Elementtext verwendet.

Beachten Sie, dass die zusätzlichen Attribute in der `analyis`, die von pcontext:Itemize() zurückgegeben wird, Indizes haben, die relativ zum gesamten Absatz sind. Sie übergeben also nicht den vollständigen Absatztext als `paragraph_text$`, sondern müssen den Elementversatz von ihren Indizes subtrahieren, bevor Sie pcontext:Itemize() aufrufen.

Mit dem optionalen Argument `flags` kann der Gestaltungsprozess beeinflusst werden. Die folgenden Flags werden derzeit erkannt:

`#PANGO_SHAPE_NONE`
: Standardwert.

`#PANGO_SHAPE_ROUND_POSITIONS`
: Rundet Glyphenpositionen und -breiten auf ganze Geräteeinheiten. Diese Option sollte festgelegt werden, wenn der Zielrenderer keine Subpixelpositionierung von Glyphen durchführen kann.

EINGABEN

`item_text$`
: Text zum Formen

`paragraph_text$`
: Text des Absatzes oder Null (siehe Details)

`analysis` Pango-Analyseobjekt von pcontext:Itemize()

`glyphs` Glyphen-Zeichenkette, in der Ergebnisse gespeichert werden sollen

`flags` Optional: zusätzliche Flags (siehe oben)

15.24 pango.TabArray

BEZEICHNUNG

pango.TabArray – Erstellt ein Tab-Array aus Tabstopps

ÜBERSICHT

```
tabs = pango.TabArray(initial_size, positions_in_pixels)
```

BESCHREIBUNG

Erstellt ein Array von `initial_size`-Tabstopps. Tabstopps werden in Pixeleinheiten angegeben, wenn `positions_in_pixels` `True` ist, andernfalls in Pango-Einheiten. Alle Anschläge stehen zunächst auf Position 0.

Diese Funktion gibt das neu zugewiesene Pango-Tab-Array zurück, das mit ptabarray:Free() gelöscht werden sollte.

EINGABEN

`initial_size`
: Die anfängliche Anzahl der zuzuordnenden Tabstopps kann 0 positions_in_pixels sein - unabhängig davon, ob die Positionen in Pixeleinheiten angegeben sind

RÜCKGABEWERTE

`tabs` das neu zugewiesene Pango-Tab-Array

15.25 pango.TabArrayWithPositions

BEZEICHNUNG

pango.TabArrayWithPositions – Erstellt ein Tab-Array

ÜBERSICHT

```
tabs = pango.TabArrayWithPositions(positions_in_pixels[, align1, pos1, ...])
```

BESCHREIBUNG

Erstellt ein Pango-Tab-Array und ermöglicht Ihnen die Angabe der Ausrichtung und Position jedes Tabstopps. Tabstopps werden in Pixeleinheiten angegeben, wenn `positions_in_pixels` `True` ist, andernfalls in Pango-Einheiten.

Für jeden Tabstopp müssen Sie eine Ausrichtung und eine Position angeben. Der Parameter `align` kann eine der folgenden Konstanten sein:

`#PANGO_TAB_LEFT`
: Der Text erscheint rechts von der Tabstopp-Position.

`#PANGO_TAB_RIGHT`
: Der Text wird links von der Tabstopp-Position angezeigt, bis der verfügbare Platz ausgefüllt ist.

`#PANGO_TAB_CENTER`
: Der Text wird an der Tabstopp-Position zentriert, bis der verfügbare Platz ausgefüllt ist.

`#PANGO_TAB_DECIMAL`
: Text vor dem ersten Vorkommen des Dezimalpunkts erscheint links von der Tabstopp-Position (bis der verfügbare Platz gefüllt ist), der Rest rechts.

Diese Funktion gibt das neu zugewiesene Pango-Tab-Array zurück, das mit ptabarray:Free() gelöscht werden sollte.

EINGABEN

`positions_in_pixels`
: ob Positionen in Pixeleinheiten angegeben sind

`align1` Ausrichtung des ersten Tabstopps (siehe oben)

`pos1` Position des ersten Tabstopps

`...` zusätzliche Ausrichtungs-/Positionspaare

RÜCKGABEWERTE

`tabs` das neu zugewiesene Pango-Tab-Array

15.26 pango.Version

BEZEICHNUNG

pango.Version – Gibt die Pango-Version zurück

ÜBERSICHT

```
ver$ = pango.Version()
```

BESCHREIBUNG

Gibt die zur Laufzeit verfügbare Version von Pango zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`ver$` Pango Version

16 Pango-Analyse

16.1 panalysis:Get

BEZEICHNUNG

panalysis:Get – Gibt die Analyseinformationen zurück

ÜBERSICHT

```
table = panalysis:Get()
```

BESCHREIBUNG

Ruft Informationen zu einem Pango-Analyseobjekt ab. Die Daten werden als Tabelle zurückgegeben. Die folgenden Felder werden in der Tabelle initialisiert:

`Font` Die Schriftart für dieses Segment. Dies ist ein Pango-Schriftart-Handle. Es ist Eigentum des Pango-Analyseobjekts und darf nicht gelöscht werden.

`Level` Die bidirektionale Ebene für dieses Segment.

`Gravity` Die Glyphenausrichtung für dieses Segment. Eine Liste der Ausrichtungskonstanten finden Sie unter pcontext:SetBaseGravity().

`Flags` Boolesche Flags für dieses Segment.

`Script` Das erkannte Skript für dieses Segment. Eine Liste der unterstützten Skripte finden Sie unter planguage:GetScripts().

`Language` Die erkannte Sprache für dieses Segment. Dies ist ein Pango-Sprachhandle. Es ist Eigentum des Pango-Analyseobjekts und darf nicht gelöscht werden.

`ExtraAttrs`
Zusätzliche Attribute für dieses Segment. Dies ist eine Tabelle mit Pango-Attributen. Die Pango-Attributobjekte sind Eigentum des Pango-Analyseobjekts und dürfen nicht gelöscht werden.

EINGABEN

Keine

RÜCKGABEWERTE

`table` eine Tabelle mit den Analysedaten

17 Pango-Attribute

17.1 pattribute:Copy

BEZEICHNUNG
pattribute:Copy – Kopiert ein Attribut

ÜBERSICHT
```
attr = pattribute:Copy()
```

BESCHREIBUNG
Erstellt eine Kopie eines Attributs.

Diese Funktion gibt das neu zugewiesene Pango-Attribut zurück, das mit pattribute:Free() gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE

`attr` die neu zugewiesene Attributkopie

17.2 pattribute:Equal

BEZEICHNUNG
pattribute:Equal – Prüft, ob Atribute gleich sind

ÜBERSICHT
```
ok = pattribute:Equal(attr2)
```

BESCHREIBUNG
Vergleicht das Attribut mit `attr2` auf Gleichheit.

Dadurch wird nur der tatsächliche Wert der beiden Attribute verglichen und nicht die Bereiche, für die die Attribute gelten.

Diese Funktion gibt `True` zurück, wenn die beiden Attribute den gleichen Wert haben.

EINGABEN

`attr2` ein weiteres Pango-Attributobjekt

RÜCKGABEWERTE

`ok` `True`, wenn die beiden Attribute den gleichen Wert haben, andernfalls `False`

17.3 pattribute:Free

BEZEICHNUNG
pattribute:Free – Löscht ein Attribut

ÜBERSICHT
```
pattribute:Free()
```

BESCHREIBUNG
Löscht ein Pango-Attribut und gibt den gesamten zugehörigen Speicher frei.

EINGABEN
Keine

17.4 pattribute:GetRange

BEZEICHNUNG
pattribute:GetRange – Gibt den Attributbereich zurück

ÜBERSICHT
```
start_index, end_index = pattribute:GetRange()
```

BESCHREIBUNG
Ruft den Bereich ab, für den der Wert im typspezifischen Teil des Attributs gilt.

EINGABEN
Keine

RÜCKGABEWERTE

`start_index`
: der Startindex des Bereichs (in Bytes)

`end_index`
: Endindex des Bereichs (in Bytes); Das Zeichen an diesem Index ist nicht im Bereich enthalten

17.5 pattribute:GetType

BEZEICHNUNG
pattribute:GetType – Gibt den Attributtyp zurück

ÜBERSICHT
```
type = pattribute:GetType()
```

BESCHREIBUNG
Ruft den Attributtyp ab. Dies kann eine der folgenden Konstanten sein:

```
#PANGO_ATTR_ABSOLUTE_SIZE
#PANGO_ATTR_ALLOW_BREAKS
#PANGO_ATTR_BACKGROUND
#PANGO_ATTR_BACKGROUND_ALPHA
#PANGO_ATTR_FALLBACK
#PANGO_ATTR_FAMILY
#PANGO_ATTR_FONT_DESC
#PANGO_ATTR_FONT_FEATURES
#PANGO_ATTR_FOREGROUND
#PANGO_ATTR_FOREGROUND_ALPHA
#PANGO_ATTR_GRAVITY
```

```
#PANGO_ATTR_GRAVITY_HINT
#PANGO_ATTR_INSERT_HYPHENS
#PANGO_ATTR_INVALID
#PANGO_ATTR_LANGUAGE
#PANGO_ATTR_LETTER_SPACING
#PANGO_ATTR_RISE
#PANGO_ATTR_SCALE
#PANGO_ATTR_SHAPE
#PANGO_ATTR_SHOW
#PANGO_ATTR_SIZE
#PANGO_ATTR_STRETCH
#PANGO_ATTR_STRIKETHROUGH
#PANGO_ATTR_STRIKETHROUGH_COLOR
#PANGO_ATTR_STYLE
#PANGO_ATTR_UNDERLINE
#PANGO_ATTR_UNDERLINE_COLOR
#PANGO_ATTR_VARIANT
#PANGO_ATTR_WEIGHT
```

EINGABEN

Keine

RÜCKGABEWERTE

`type` der Typ des Attributs

17.6 pattribute:GetValue

BEZEICHNUNG

pattribute:GetValue – Gibt den Attributwert zurück

ÜBERSICHT

```
v = pattribute:GetValue()
```

BESCHREIBUNG

Ruft den Attributwert ab. Der Attributwert ist der Wert, der beim Aufruf von pango.Attribute() an das Attribut übergeben wird. Je nach Attributtyp können im Attribut auch mehrere Werte vorhanden sein.

EINGABEN

Keine

RÜCKGABEWERTE

`v` der Wert des Attributs

17.7 pattribute:IsNull

BEZEICHNUNG

pattribute:IsNull – Überprüft, ob das Attribut ungültig ist

ÜBERSICHT

```
bool = pattribute:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Attribut `NULL` ist, also ungültig. Wenn Funktionen, die Objekte zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Objektzuordnung fehlgeschlagen ist. In diesem Fall ist das Attribut `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Attribut `NULL` ist, andernfalls `False`

17.8 pattribute:SetRange

BEZEICHNUNG

pattribute:SetRange – Legt den Attributbereich fest

ÜBERSICHT

```
pattribute:SetRange(start_index, end_index)
```

BESCHREIBUNG

Legt den Bereich fest, für den der Wert im typspezifischen Teil des Attributs gilt.

EINGABEN

`start_index`
: der Startindex des Bereichs (in Bytes)

`end_index`
: Endindex des Bereichs (in Bytes); Das Zeichen an diesem Index ist nicht im Bereich enthalten

18 Pango-Attributliste

18.1 pattrlist:Change

BEZEICHNUNG
pattrlist:Change – Fügt ein Attribut in die Liste ersetzend ein

ÜBERSICHT
`pattrlist:Change(attr)`

BESCHREIBUNG
Fügt das angegebene Attribut in die Pango-Attributliste ein.

Es ersetzt alle Attribute desselben Typs in diesem Segment und wird mit allen angrenzenden Attributen zusammengeführt, die identisch sind.

Diese Funktion ist beim Erstellen einer Attributliste der Reihe nach langsamer als pattrlist:Insert() (möglicherweise viel langsamer bei großen Listen). Allerdings eignet sich pattrlist:Insert() nicht zum kontinuierlichen Ändern einer Reihe von Attributen, da vorhandene Attribute niemals entfernt oder kombiniert werden.

EINGABEN

`attr` das einzufügende Attribut

18.2 pattrlist:Copy

BEZEICHNUNG
pattrlist:Copy – Kopiert die Attributliste

ÜBERSICHT
`list = pattrlist:Copy()`

BESCHREIBUNG
Kopiert die Attributliste und gibt eine identische neue Liste zurück.

EINGABEN
Keine

RÜCKGABEWERTE

`list` die neu zugewiesene Attributliste

18.3 pattrlist:Free

BEZEICHNUNG
pattrlist:Free – Löscht eine Attributliste

ÜBERSICHT
`pattrlist:Free()`

BESCHREIBUNG
Verringert den Referenzzähler der angegebenen Attributliste um eins.

Wenn das Ergebnis Null ist, wird die Attributliste und die darin enthaltenen Attribute alle gelöscht und freigegeben.

EINGABEN
Keine

18.4 pattrlist:GetAttributes

BEZEICHNUNG
pattrlist:GetAttributes – Gibt die Attribute als Tabelle zurück

ÜBERSICHT
```
t = pattrlist:GetAttributes()
```

BESCHREIBUNG
Ruft eine Liste aller Attribute in der Liste ab. Diese Funktion gibt eine Tabelle zurück, die alle Attribute in der Liste enthält. Sie müssen die einzelnen Attribute mit pattribute:Free() löschen.

EINGABEN
Keine

RÜCKGABEWERTE

`t`	Tabelle mit allen Attributen in der Liste

18.5 pattrlist:Insert

BEZEICHNUNG
pattrlist:Insert – Fügt ein Attribut in die Liste mit Startindex ein

ÜBERSICHT
```
pattrlist:Insert(attr)
```

BESCHREIBUNG
Fügt das angegebene Attribut in die Pango-Attributliste ein.

Es wird nach allen anderen Attributen mit passendem Startindex eingefügt. Sie können pattribute:SetRange() verwenden, um den Startindex festzulegen.

EINGABEN

`attr`	das einzufügende Attribut

18.6 pattrlist:InsertBefore

BEZEICHNUNG
pattrlist:InsertBefore – Fügt ein Attribut vor allen anderen ein

ÜBERSICHT

```
pattrlist:InsertBefore(attr)
```

BESCHREIBUNG

Fügt das angegebene Attribut in die Pango-Attributliste ein.

Es wird vor allen anderen Attributen mit passendem Startindex eingefügt. Sie können pattribute:SetRange() verwenden, um den Startindex festzulegen.

EINGABEN

`attr`	das einzufügende Attribut

18.7 pattrlist:IsNull

BEZEICHNUNG

pattrlist:IsNull – Überprüft, ob die Attributliste ungültig ist

ÜBERSICHT

```
bool = pattrlist:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Attributliste `NULL` ist, also ungültig. Wenn Funktionen, die Objekte zuordnen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Objektzuordnung fehlgeschlagen ist. In diesem Fall ist die Attributliste `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool`	`True`, wenn die Attributliste `NULL` ist, andernfalls `False`

18.8 pattrlist:Reference

BEZEICHNUNG

pattrlist:Reference – Erhöht die Referenzanzahl der Attributlisten

ÜBERSICHT

```
pattrlist:Reference()
```

BESCHREIBUNG

Erhöht die Referenzanzahl der angegebenen Attributliste um eins.

EINGABEN

Keine

18.9 pattrlist:Splice

BEZEICHNUNG

pattrlist:Splice – Spleißt eine Attributliste

ÜBERSICHT

```
pattrlist:Splice(other, pos, len)
```

BESCHREIBUNG

Diese Funktion öffnet eine Lücke in der Liste, füllt sie von links mit Attributen und fügt dann die durch `other` angegebene Attributliste über der Lücke zusammen.

Diese Operation entspricht dem Strecken jedes Attributs, das an der Position `pos` in der Liste gilt, um einen Betrag `len` und dem anschließenden Aufruf von pattrlist:Change() mit einer Kopie jedes Attributs in `other` der Reihe nach (in der Position um `pos` versetzt und in der Länge auf `len` begrenzt).

Dieser Vorgang erweist sich beispielsweise zum Einfügen einer Vorbearbeitungs-Zeichenkette in die Mitte eines Bearbeitungspuffers als nützlich.

Aus Gründen der Abwärtskompatibilität verhält sich die Funktion anders, wenn `len` den Wert 0 hat. In diesem Fall sind die Attribute von `other` nicht auf `len` beschränkt, sondern werden einfach über die Liste gelegt.

Dieser Modus eignet sich zum Zusammenführen zweier Attributlisten.

EINGABEN

`other`	eine weitere Pango-Attributliste
`pos`	die Position in der Liste, an der `other` eingefügt werden soll
`len`	die Länge des gespleißten Segments; Beachten Sie, dass dies angegeben werden muss, da die Attribute in `other` möglicherweise nur in einem Unterabschnitt dieses Bereichs vorhanden sind

18.10 pattrlist:Update

BEZEICHNUNG

pattrlist:Update – Aktualisiert die Attributindizes

ÜBERSICHT

```
pattrlist:Update(pos, remove, add)
```

BESCHREIBUNG

Aktualisiert die Attributindizes in der Liste, wenn sich der Text ändert, auf den sie sich beziehen.

Die von dieser Funktion vorgenommene Änderung besteht darin, `remove-Bytes` an der Position `pos` zu entfernen und stattdessen `add-Bytes` einzufügen.

Attribute, die vollständig im Bereich (`pos`, `pos + remove`) liegen, werden entfernt.

Attribute, die innerhalb des Bereichs (`pos`, `pos + remove`) beginnen oder enden, werden gekürzt, um die Entfernung widerzuspiegeln.

Die Start- und Endpositionen der Attribute werden aktualisiert, wenn sie hinter `pos + remove` liegen.

EINGABEN

`pos`	die Position der Änderung
`remove`	die Anzahl der entfernten Bytes
`add`	die Anzahl der hinzugefügten Bytes

19 Pango-Kontext

19.1 pcontext:Changed

BEZEICHNUNG
pcontext:Changed – Erzwingt einen Kontextwechsel

ÜBERSICHT
`pcontext:Changed()`

BESCHREIBUNG
Erzwingt eine Änderung des Kontexts, was dazu führt, dass jedes Pango-Layout, das diesen Kontext verwendet, neu gestaltet wird.

Diese Funktion ist nur nützlich, wenn ein neues Backend für Pango implementiert wird, was Anwendungen nicht tun werden. Backends sollten diese Funktion aufrufen, wenn sie zusätzliche Daten an den Kontext angehängt haben und diese Daten geändert werden.

EINGABEN
Keine

19.2 pcontext:Free

BEZEICHNUNG
pcontext:Free – Verringert/löscht die Verweise auf einen Pango-Kontext

ÜBERSICHT
`pcontext:Free()`

BESCHREIBUNG
Verringert die Anzahl der Verweise auf einen Pango-Kontext um eins.

Ist das Ergebnis gleich Null, werden der Kontext gelöscht und der gesamte zugehörige Speicher freigegeben.

EINGABEN
Keine

19.3 pcontext:GetBaseDir

BEZEICHNUNG
pcontext:GetBaseDir – Gibt die Basisrichtung für Kontext zurück

ÜBERSICHT
`dir = pcontext:GetBaseDir()`

BESCHREIBUNG
Gibt die Basisrichtung für den Kontext zurück. Siehe pcontext:SetBaseDir() für weitere Informationen.

EINGABEN
Keine

RÜCKGABEWERTE
`dir` die Basisrichtung für den Kontext

19.4 pcontext:GetBaseGravity

BEZEICHNUNG
pcontext:GetBaseGravity – Gibt die Basisausrichtung für den Kontext zurück

ÜBERSICHT
```
gravity = pcontext:GetBaseGravity()
```

BESCHREIBUNG
Gibt die Basisausrichtung für den Kontext zurück. Siehe pcontext:SetBaseGravity() für weitere Informationen.

EINGABEN
Keine

RÜCKGABEWERTE
`gravity` die Basisausrichtung für den Kontext

19.5 pcontext:GetFontDescription

BEZEICHNUNG
pcontext:GetFontDescription – Gibt die Beschreibung der Schriftart zurück

ÜBERSICHT
```
handle = pcontext:GetFontDescription()
```

BESCHREIBUNG
Gibt die Standardschriftartbeschreibung für den Kontext zurück.

Diese Funktion gibt ein Handle auf die Standardschriftbeschreibung des Kontexts zurück. Dieser Wert darf nicht verändert oder gelöscht werden.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` ein Handle auf die Standardschriftart des Kontexts

19.6 pcontext:GetFontMap

BEZEICHNUNG
pcontext:GetFontMap – Gibt den Zeichensatz zurück

ÜBERSICHT

```
handle = pcontext:GetFontMap()
```

BESCHREIBUNG

Gibt den Pango-Zeichensatz zurück, der für die Suche nach Schriftarten für diesen Kontext verwendet wird.

Diese Funktion gibt den Zeichensatz für den Pango-Kontext zurück. Dieser Wert ist Pango zugewiesen und sollte nicht gelöscht werden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` der Zeichensatz für den Pango-Kontext

19.7 pcontext:GetFontOptions

BEZEICHNUNG

pcontext:GetFontOptions – Gibt die Schriftart-Optionen zurück

ÜBERSICHT

```
handle = pcontext:GetFontOptions()
```

BESCHREIBUNG

Gibt alle Schriftart-Rendering-Optionen zurück, die zuvor mit pcontext:SetFontOptions() gesetzt wurden.

Diese Funktion meldet keine Optionen, die mit pcontext:UpdateContext() von der Zieloberfläche abgeleitet werden.

Diese Funktion gibt das zuvor im Kontext festgelegte Cairo-Schriftoptionsobjekt oder `NULL` zurück, wenn keine Optionen gesetzt wurden. Das zurückgegebene Objekt ist dem Kontexts zugewiesen und darf nicht verändert oder gelöscht werden. Sie können cfontoptions:IsNull() verwenden, um zu prüfen, ob das zurückgegebene Objekt `NULL` ist.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` ein Cairo-Schriftart-Optionen-Objekt

19.8 pcontext:GetGravity

BEZEICHNUNG

pcontext:GetGravity – Gibt die Ausrichtung zurück

ÜBERSICHT

```
gravity = pcontext:GetGravity()
```

BESCHREIBUNG

Gibt die Ausrichtung für den Kontext zurück.

Dies ist ähnlich wie pcontext:GetBaseGravity(), außer wenn die Basisausrichtung `#PANGO_GRAVITY_AUTO` ist, für die pango.GravityForMatrix() verwendet wird, um die Ausrichtung aus der aktuellen Kontextmatrix zurückzugeben.

EINGABEN
Keine

RÜCKGABEWERTE

`gravity` die aufgelöste Ausrichtung für den Kontext

19.9 pcontext:GetGravityHint

BEZEICHNUNG
pcontext:GetGravityHint – Gibt den Ausrichtungshinweis zurück

ÜBERSICHT
```
hint = pcontext:GetGravityHint()
```

BESCHREIBUNG
Gibt den Ausrichtungshinweis für den Kontext zurück. Siehe pcontext:SetGravityHint() für Details.

EINGABEN
Keine

RÜCKGABEWERTE

`hint` den Ausrichtungshinweis für den Kontext

19.10 pcontext:GetLanguage

BEZEICHNUNG
pcontext:GetLanguage – Gibt die Sprache zurück

ÜBERSICHT
```
lang = pcontext:GetLanguage()
```

BESCHREIBUNG
Gibt den globalen Sprach-Tag für den Kontext zurück. Der Rückgabewert ist ein Handle auf ein Pango-Sprachobjekt.

EINGABEN
Keine

RÜCKGABEWERTE

`lang` Handle zum globalen Sprach-Tag

19.11 pcontext:GetMatrix

BEZEICHNUNG

pcontext:GetMatrix – Gibt die Matrix zurück

ÜBERSICHT

```
m = pcontext:GetMatrix()
```

BESCHREIBUNG

Gibt die Transformationsmatrix zurück, die beim Rendern mit diesem Kontext angewendet wird. Siehe pcontext:SetMatrix() für weitere Informationen.

EINGABEN

Keine

RÜCKGABEWERTE

`m` ein Pango-Matrix-object

19.12 pcontext:GetMetrics

BEZEICHNUNG

pcontext:GetMetrics – Gibt die Metriken zurück

ÜBERSICHT

```
metrics = pcontext:GetMetrics(desc, language)
```

BESCHREIBUNG

Gibt die allgemeinen Metrik-Informationen für eine bestimmte Schriftartbeschreibung zurück.

Da die Metriken für verschiedene Skripte sehr unterschiedlich sein können, kann ein Sprach-Tag angegeben werden, um darauf hinzuweisen, dass die Metriken abgerufen werden sollen, die dem/den von dieser Sprache verwendeten Skript(s) entsprechen.

Die Beschreibung der Pango-Schriftart wird auf dieselbe Weise interpretiert wie bei pcontext:Itemize(), und der Familienname kann eine durch Komma getrennte Liste von Namen sein. Wenn Zeichen aus mehreren dieser Familien zum Rendern der Zeichenkette verwendet werden, sind die zurückgegebenen Schriftarten eine Zusammensetzung der Metriken für die Schriftarten, die für die einzelnen Familien geladen wurden.

Diese Funktion gibt ein Pango-Schriftart-Metriken-Objekt zurück. Der Aufrufer muss pfontmetrics:Free() aufrufen, wenn er das Objekt nicht mehr benötigt.

EINGABEN

`desc` eine Pango-Schriftbeschreibungsstruktur; `Nil` bedeutet, dass die Schriftbeschreibung aus dem Kontext verwendet wird

`language` Sprach-Tag, mit dem bestimmt wird, für welches Skript die Metriken abgerufen werden sollen; `Nil` bedeutet, dass der Sprach-Tag aus dem Kontext verwendet wird; wenn im Kontext kein Sprach-Tag gesetzt ist, werden Metriken für die Standardsprache (wie durch pango.GetDefaultLanguage() bestimmt) zurückgegeben

RÜCKGABEWERTE

`metrics` ein Pango-Schriftart-Metriken-Objekt

19.13 pcontext:GetResolution

BEZEICHNUNG

pcontext:GetResolution – Gibt die Kontextauflösung zurück

ÜBERSICHT

```
res = pcontext:GetResolution()
```

BESCHREIBUNG

Ruft die Auflösung für den Kontext ab.

Diese Funktion gibt die Auflösung in "Punkten pro Zoll" zurück. Ein negativer Wert wird zurückgegeben, wenn zuvor keine Auflösung festgelegt wurde.

Siehe pcontext:SetResolution() für Details.

EINGABEN

Keine

RÜCKGABEWERTE

`res` die Kontextauflösung in "Punkten pro Zoll".

19.14 pcontext:GetRoundGlyphPositions

BEZEICHNUNG

pcontext:GetRoundGlyphPositions – Gibt zurück, ob Glyphenpositionen gerundet werden sollen

ÜBERSICHT

```
ok = pcontext:GetRoundGlyphPositions()
```

BESCHREIBUNG

Gibt zurück, ob beim Rendern von Schriften mit diesem Kontext die Glyphenpositionen und -breiten gerundet werden sollen.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` `True` oder `False`, um anzugeben, ob die Positionen und Breiten der Glyphen gerundet werden sollen oder nicht

19.15 pcontext:GetSerial

BEZEICHNUNG

pcontext:GetSerial – Gibt die Seriennummer des Kontextes zurück

ÜBERSICHT

```
s = pcontext:GetSerial()
```

BESCHREIBUNG

Gibt die aktuelle Seriennummer des Kontexts zurück.

Die Seriennummer wird auf eine kleine Zahl größer als Null initialisiert, wenn ein neuer Kontext erstellt wird und wird immer dann erhöht, wenn der Kontext mit einer der Set-Funktionen geändert wird oder die Pango-Zeichensatz, die zum Auffinden von Schriftarten verwendet wird, geändert wurde. Die Seriennummer kann umgebrochen werden, wird aber nie den Wert 0 haben. Da sie umgebrochen werden kann, sollte man sie nie mit "kleiner als" vergleichen, sondern immer "nicht gleich" verwenden.

Dies kann verwendet werden, um automatisch Änderungen an einem Pango-Kontext zu erkennen, und ist nur nützlich, wenn Objekte implementiert werden, die aktualisiert werden müssen, wenn sich ihr Pango-Kontext ändert, wie z.B. Pango-Layout.

EINGABEN

Keine

RÜCKGABEWERTE

`s`	die aktuelle Seriennummer des Kontexts

19.16 pcontext:IsNull

BEZEICHNUNG

pcontext:IsNull – Prüft, ob der Kontext ungültig ist

ÜBERSICHT

```
bool = pcontext:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn der Kontext `NULL` ist, d.h. ungültig. Wenn Funktionen, die Kontexte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen den Kontext einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Kontextzuweisung fehlgeschlagen ist. In diesem Fall wird der Kontext auf `NULL` gesetzt.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktionen ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool`	`True`, wenn der Kontext `NULL` ist, sonst `False`

19.17 pcontext:Itemize

BEZEICHNUNG

pcontext:Itemize – Unterteilt einen Textabschnitt in Segmente

ÜBERSICHT

```
list = pcontext:Itemize(text$, start_index, length, attrs[, base_dir])
```

BESCHREIBUNG

Unterteilt einen Textabschnitt in Segmente mit einheitlicher Richtungsebene und Schriftart. Jedes Byte des Textes ist in genau einem der Elemente in der zurückgegebenen Liste enthalten. Die erzeugte Liste der Elemente ist in logischer Reihenfolge (die Startoffsets der Elemente sind aufsteigend).

Optional können Sie die Basisrichtung im Argument `base_dir` angeben. Die Basisrichtung wird beim Berechnen bidirektionaler Ebenen verwendet. Wenn `base_dir` weggelassen wird, wird die Basisrichtung aus dem Kontext ermittelt. Eine Liste der Basisrichtungen finden Sie unter pcontext:SetBaseDir().

Beachten Sie, dass Sie die einzelnen Pango-Elemente in der Liste mit pitem:Free() löschen müssen.

EINGABEN

`text$` der aufzulistende Text

`start_index` erstes zu verarbeitendes Byte im Text

`length` die Anzahl der zu verarbeitenden Bytes (nicht Zeichen) nach `start_index`. Dieser Wert muss >= 0 sein

`attrs` eine Pango-Attributliste mit der Menge der Attribute, die für Text gelten

`base_dir` optional: Basisrichtung, die für die bidirektionale Verarbeitung zu verwenden ist

RÜCKGABEWERTE

`list` eine Tabelle mit einer Liste von Pango-Elementen; der Aufrufer ist dafür verantwortlich, jedes Element mit pitem:Free() zu löschen

19.18 pcontext:ListFamilies

BEZEICHNUNG

pcontext:ListFamilies – Listet die Schriftfamilien auf

ÜBERSICHT

```
t = pcontext:ListFamilies()
```

BESCHREIBUNG

Listet alle Schriftfamilien für einen Kontext auf. Die Schriftfamilien werden als Tabelle zurückgegeben, die eine Liste von Pango-Schriftfamilien-Handles enthält. Die Schriftfamilien-Handles sind dem Kontext zugewiesen und dürfen nicht gelöscht werden.

EINGABEN

Keine

RÜCKGABEWERTE

`t` Tabelle mit allen Schriftfamilien

19.19 pcontext:LoadFont

BEZEICHNUNG

pcontext:LoadFont – Lädt eine Schriftart

ÜBERSICHT

```
font = pcontext:LoadFont(desc)
```

BESCHREIBUNG

Lädt die Schriftart in eine der Zeichensätze im Kontext, die am ehesten mit der in `desc` übergebenen Pango-Schriftartbeschreibung übereinstimmt.

Diese Funktion gibt die neu zugewiesene Pango-Schriftart zurück, die geladen wurde, oder `NULL`, wenn keine Schriftart übereinstimmt. Sie können pfont:IsNull() verwenden, um zu prüfen, ob das zurückgegebene Objekt eine NULL-Schriftart enthält.

EINGABEN

`desc` eine Pango-Schriftartbeschreibung, die die zu ladende Schriftart beschreibt

RÜCKGABEWERTE

`font` die neu zugewiesene Pango-Schriftart

19.20 pcontext:LoadFontset

BEZEICHNUNG

pcontext:LoadFontset – Lädt einen Schriftsatz

ÜBERSICHT

```
fontset = pcontext:LoadFontset(desc, language)
```

BESCHREIBUNG

Lädt eine Reihe von Schriftarten in den Kontext, die zum Rendern einer Schriftart verwendet werden können, die `desc` entspricht.

Diese Funktion gibt den neu zugewiesenen Pango-Schriftsatz zurück, der geladen wurde, oder `NULL`, wenn kein Schriftsatz übereinstimmt. Sie können pfontset:IsNull() verwenden, um zu prüfen, ob das zurückgegebene Objekt einen NULL-Schriftsatz enthält.

EINGABEN

`desc` eine Pango-Schriftartbeschreibung, die die zu ladenden Schriftarten beschreibt

`language` eine Pango-Sprache, für die die Schriftarten verwendet werden sollen

RÜCKGABEWERTE

`fontset` den neu zugewiesenen Schriftsatz

19.21 pcontext:Reference

BEZEICHNUNG
pcontext:Reference – Erhöht die Anzahl der Referenzen der Pango-Kontexte

ÜBERSICHT
`pcontext:Reference()`

BESCHREIBUNG
Erhöht die Anzahl der Verweise auf den Pango-Kontext um eins.

EINGABEN
Keine

19.22 pcontext:SetBaseDir

BEZEICHNUNG
pcontext:SetBaseDir – Legt die Basisrichtung für den Kontext fest

ÜBERSICHT
`pcontext:SetBaseDir(direction)`

BESCHREIBUNG
Legt die Basisrichtung für den Kontext fest. Der Parameter `direction` kann auf eine der folgenden Konstanten gesetzt werden:

`#PANGO_DIRECTION_LTR`
: Eine starke Links-nach-Rechts-Richtung.

`#PANGO_DIRECTION_RTL`
: Eine starke Rechts-nach-Links-Richtung.

`#PANGO_DIRECTION_WEAK_LTR`
: Eine schwache Links-nach-Rechts-Richtung.

`#PANGO_DIRECTION_WEAK_RTL`
: Eine schwache Rechts-nach-Links-Richtung.

`#PANGO_DIRECTION_NEUTRAL`
: Keine Richtung festgelegt.

Die Basisrichtung wird bei der Anwendung des Unicode-Bidirectional-Algorithmus verwendet. Wenn die `direction` `#PANGO_DIRECTION_LTR` oder `#PANGO_DIRECTION_RTL` ist, dann wird der Wert als Absatzrichtung im Unicode-Bidirectional-Algorithmus verwendet. Ein Wert von `#PANGO_DIRECTION_WEAK_LTR` oder `#PANGO_DIRECTION_WEAK_RTL` wird nur für Absätze verwendet, die selbst keine starken Zeichen enthalten.

EINGABEN

`direction`
: die neue Basisrichtung

19.23 pcontext:SetBaseGravity

BEZEICHNUNG

pcontext:SetBaseGravity – Stellt die Basisausrichtung ein

ÜBERSICHT

```
pcontext:SetBaseGravity(gravity)
```

BESCHREIBUNG

Legt die Basisausrichtung für den Kontext fest. Der Parameter `gravity` kann auf eine der folgenden Konstanten gesetzt werden:

`#PANGO_GRAVITY_SOUTH`
: Die Glyphen stehen aufrecht (Voreinstellung).

`#PANGO_GRAVITY_EAST`
: Die Glyphen werden um 90 Grad gegen den Uhrzeigersinn gedreht.

`#PANGO_GRAVITY_NORTH`
: Die Glyphen stehen auf dem Kopf.

`#PANGO_GRAVITY_WEST`
: Die Glyphen werden um 90 Grad im Uhrzeigersinn gedreht.

`#PANGO_GRAVITY_AUTO`
: Die Basisausrichtung wird beim Darstellen von vertikalem Text verwendet.

EINGABEN

`gravity` die neue Basisausrichtung (siehe oben)

19.24 pcontext:SetFontDescription

BEZEICHNUNG

pcontext:SetFontDescription – Stellt die Schriftartbeschreibung ein

ÜBERSICHT

```
pcontext:SetFontDescription(desc)
```

BESCHREIBUNG

Legt die Standard-Schriftbeschreibung für den Kontext fest. Der Parameter `desc` kann auch `Nil` sein, um die Schriftbeschreibung auf einen Standardwert zurückzusetzen.

EINGABEN

`desc` die neue Pango-Schriftartbeschreibung oder `Nil`

19.25 pcontext:SetFontMap

BEZEICHNUNG

pcontext:SetFontMap – Setzt den Zeichensatz

ÜBERSICHT

```
pcontext:SetFontMap(font_map)
```

BESCHREIBUNG

Legt den Zeichensatz fest, der bei der Suche nach Schriftarten in diesem Kontext durchsucht werden soll. Der Parameter `font_map` kann auch `Nil` sein, um den Zeichensatz auf einen Standardwert zurückzusetzen.

Dies ist nur für die interne Verwendung durch Pango-Backends gedacht. Ein Pango-Kontext, der über eine der empfohlenen Funktionen bezogen wird, sollte bereits ein geeigneten Zeichensatz haben.

EINGABEN

`font_map` der zu setzende Pango-Zeichensatz oder `Nil`

19.26 pcontext:SetFontOptions

BEZEICHNUNG

pcontext:SetFontOptions – Stellt die Schriftartoptionen ein

ÜBERSICHT

```
pcontext:SetFontOptions(options)
```

BESCHREIBUNG

Legt die Schriftartoptionen fest, die beim Rendern von Text mit diesem Kontext verwendet werden. Diese Optionen haben Vorrang vor allen Optionen, die pcontext:UpdateContext() von der Zieloberfläche ableitet. Diese Funktion erstellt eine Kopie des ihr übergebenen Objekt `options`.

EINGABEN

`options` ein Cairo-Schriftart-Optionsobjekt oder `Nil`, um alle zuvor festgelegten Optionen aufzuheben

19.27 pcontext:SetGravityHint

BEZEICHNUNG

pcontext:SetGravityHint – Stellt den Ausrichtungshinweis ein

ÜBERSICHT

```
pcontext:SetGravityHint(hint)
```

BESCHREIBUNG

Legt den Ausrichtungshinweis für den Kontext fest. Der Parameter `hint` kann eine der folgenden Konstanten sein:

`#PANGO_GRAVITY_HINT_NATURAL`
: Skripte erhalten ihre natürliche Ausrichtung auf der Grundlage der Basisausrichtung und des Skripts. Dies ist die Voreinstellung.

`#PANGO_GRAVITY_HINT_STRONG`
: Verwendet unabhängig vom Skript immer die Basisausrichtung.

`#PANGO_GRAVITY_HINT_LINE`
: Für Schriftarten, die nicht in ihrer natürlichen Ausrichtung verlaufen (z.B. Latein in östlicher Ausrichtung), wählen Sie die Ausrichtung pro Schriftart so, dass jede Schriftart den Zeilenverlauf respektiert. Das bedeutet, dass beispielsweise Latein und Arabisch entgegengesetzte Ausrichtung haben und beide von oben nach unten fließen.

Die Ausrichtung wird für das Anordnen von vertikalem Text verwendet und ist nur relevant, wenn die Ausrichtung des Kontexts, die von pcontext:GetGravity() zurückgegeben wird, auf `#PANGO_GRAVITY_EAST` oder `#PANGO_GRAVITY_WEST` gesetzt ist.

EINGABEN

`hint` den neuen Ausrichtungshinweis (siehe oben)

19.28 pcontext:SetLanguage

BEZEICHNUNG

pcontext:SetLanguage – Setzt die Sprache

ÜBERSICHT

```
pcontext:SetLanguage(language)
```

BESCHREIBUNG

Setzt das globale Sprach-Tag für den Kontext. Der Parameter `language` kann auch `Nil` sein, um die Sprache auf die Standardeinstellung zurückzusetzen.

Die Standardsprache des laufenden Prozesses kann mit pango.GetDefaultLanguage() ermittelt werden.

EINGABEN

`language` das neue Sprach-Tag oder `Nil`

19.29 pcontext:SetMatrix

BEZEICHNUNG

pcontext:SetMatrix – Setzt die Matrix

ÜBERSICHT

```
pcontext:SetMatrix(matrix)
```

BESCHREIBUNG

Legt die Transformationsmatrix fest, die beim Rendern mit diesem Kontext angewendet wird.

Beachten Sie, dass die Metriken in den Koordinaten des Benutzerbereichs vor der Anwendung der Matrix angegeben werden, nicht in den Koordinaten des Gerätebereichs nach der Anwendung der Matrix. Sie skalieren also nicht mit der Matrix, obwohl sie sich bei verschiedenen Matrizen leicht ändern können, je nachdem, wie der Text an das Pixelraster angepasst wird.

EINGABEN

`matrix` eine Pango-Matrix

19.30 pcontext:SetResolution

BEZEICHNUNG

pcontext:SetResolution – Setzt die Kontextauflösung

ÜBERSICHT

```
pcontext:SetResolution(dpi)
```

BESCHREIBUNG

Setzt die Auflösung für den Kontext.

Dies ist ein Skalierungsfaktor zwischen den in einer Pango-Schriftbeschreibung angegebenen Punkten und den Cairo-Einheiten. Der Standardwert ist 96, was bedeutet, dass eine 10-Punkt-Schrift 13 Einheiten hoch ist. (10 * 96. / 72. = 13.3).

Beachten Sie, dass die Auflösung zwar in "Punkte pro Zoll" angegeben werden muss, es sich aber nicht um physikalische Zoll handelt. Die Terminologie ist konventionell. Ein Wert von 0 oder ein negativer Wert bedeutet, dass die Auflösung aus dem Zeichensatz verwendet werden soll.

EINGABEN

`dpi` die Auflösung in "Punkte pro Zoll".

19.31 pcontext:SetRoundGlyphPositions

BEZEICHNUNG

pcontext:SetRoundGlyphPositions – Setzt, ob Glyphenpositionen gerundet werden sollen

ÜBERSICHT

```
pcontext:SetRoundGlyphPositions(round_positions)
```

BESCHREIBUNG

Legt fest, ob beim Rendern von Schriftarten mit diesem Kontext die Glyphenpositionen und -breiten auf ganze Positionen in Geräteeinheiten gerundet werden sollen.

Dies ist nützlich, wenn der Renderer nicht mit der Subpixel-Positionierung von Glyphen umgehen kann.

Der voreingestellte Wert ist die Rundung der Glyphenpositionen, um mit dem früheren Verhalten von Pango kompatibel zu bleiben.

EINGABEN

`round_positions`
: boolescher Wert, der angibt, ob Glyphenpositionen gerundet werden sollen (`True`) oder nicht (`False`)

19.32 pcontext:SetShapeRenderer

BEZEICHNUNG

pcontext:SetShapeRenderer – Setzt den Form-Renderer

ÜBERSICHT

```
pcontext:SetShapeRenderer([func[, userdata]])
```

BESCHREIBUNG

Legt die Callback-Funktion für den Kontext fest, die für das Rendern von Attributen des Typs `#PANGO_ATTR_SHAPE` verwendet wird.

Die Callback-Funktion erhält drei bis vier Argumente: Das erste Argument ist ein Handle zu einem Cairo-Kontext, das zweite Argument ist ein Pango-Attribut vom Typ shape und das dritte Argument ist ein Boolean, das angibt, ob nur der shape-Pfad an den aktuellen Pfad des Cairo-Kontexts angehängt werden soll und kein Füllen/Streichen erfolgen soll. Wenn Sie den Parameter `userdata` angeben, wird er als vierter Parameter an die Callback-Funktion weitergeleitet.

Um das Rendern von Formen zu deaktivieren, rufen Sie diese Funktion ohne Argumente auf.

Beachten Sie, dass Sie sicherstellen müssen, dass das Objekt, für das Sie diese Funktion aufrufen, so lange gültig bleibt, wie Sie den Shape-Renderer benötigen. Stellen Sie also sicher, dass Sie es nicht auf `Nil` setzen und dass das Objekt nicht in irgendeiner Weise vom Speicherbereiniger/Garbage Collector gelöscht wird. Siehe das Beispiel ShapeText.hws, das mit Pangomonium geliefert wird, wie man sicherstellt, dass das Objekt, für das man diese Funktion aufruft, nicht Speicherbereiniger/Garbage Collector gelöscht wird.

EINGABEN

`func`	optional: Callback-Funktion für das Rendern von Attributen des Typs `#PANGO_ATTR_SHAPE`
`userdata`	optional: Benutzerdaten, die an `func` übergeben werden

19.33 pcontext:UpdateContext

BEZEICHNUNG

pcontext:UpdateContext – Aktualiert den Kontext

ÜBERSICHT

```
pcontext:UpdateContext(context)
```

BESCHREIBUNG

Aktualisiert einen Pango-Kontext, der zuvor für die Verwendung mit Cairo erstellt wurde, damit er mit der aktuellen Transformation und der Zielfläche des in `context` übergebenen Cairo-Kontextes übereinstimmt.

Wenn für den Kontext Layouts erstellt wurden, ist es notwendig, playout:ContextChanged() für diese Layouts aufzurufen.

EINGABEN

`context`	ein Cairo-Kontext

20 Pango-Abdeckung

20.1 pcoverage:Copy

BEZEICHNUNG
pcoverage:Copy – Kopiert die Zeichenbereichsabdeckung

ÜBERSICHT
`handle = pcoverage:Copy()`

BESCHREIBUNG
Kopieren Sie eine bestehende Pango-Zeichenbereichsabdeckung.

Diese Funktion gibt die neu zugewiesene Pango-Zeichenbereichsabdeckung zurück, mit einer Referenzzahl von eins, die mit pcoverage:Free() gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` den neu zugewiesenen Pango-Zeichenbereichsabdeckung

20.2 pcoverage:Free

BEZEICHNUNG
pcoverage:Free – Verringert die Anzahl der Zeichenbereichsabdeckung

ÜBERSICHT
`pcoverage:Free()`

BESCHREIBUNG
Verringert die Anzahl der Verweise auf die Pango-Zeichenbereichsabdeckung um eins. Wenn das Ergebnis Null ist, wird die Zeichenbereichsabdeckung gelöscht und den gesamten zugehörigen Speicher frei gegeben.

EINGABEN
Keine

20.3 pcoverage:Get

BEZEICHNUNG
pcoverage:Get – Gibt den Zeichenbereichsabdeckungsgrad zurück

ÜBERSICHT
`level = pcoverage:Get(index)`

BESCHREIBUNG
Gibt zurück, ob ein bestimmter Index abgedeckt ist. Diese Funktion gibt den Abdeckungsgrad der Zeichenbereichsabdeckung für das Zeichen `index` zurück.

EINGABEN

`index` der zu prüfende Index

RÜCKGABEWERTE

`level` der Abdeckungsgrad für das Zeichen `index`

20.4 pcoverage:IsNull

BEZEICHNUNG

pcoverage:IsNull – Prüft, ob die Zeichenbereichsabdeckung ungültig ist

ÜBERSICHT

```
bool = pcoverage:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Zeichenbereichsabdeckung `NULL` ist, d.h. ungültig. Wenn Funktionen, die Zeichenbereichsabdeckungen zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Zeichenbereichsabdeckung einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Zuweisung der Zeichenbereichsabdeckung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktionen ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True` wenn die Abdeckung `NULL` ist, sonst `False`

20.5 pcoverage:Reference

BEZEICHNUNG

pcoverage:Reference – Erhöht die Anzahl der Referenzen der Pango-Zeichenbereichsabdeckung

ÜBERSICHT

```
pcoverage:Reference()
```

BESCHREIBUNG

Erhöht die Anzahl der Referenzen in der Pango-Zeichenbereichsabdeckung um eins.

EINGABEN

Keine

20.6 pcoverage:Set

BEZEICHNUNG

pcoverage:Set – Setzt den Abdeckungsgrad

ÜBERSICHT

```
pcoverage:Set(index, level)
```

BESCHREIBUNG

Ändet einen bestimmten Index innerhalb der Abdeckung.

EINGABEN

`index`	der zu ändernde Index
`level`	der neue Abdeckungsgrad für `index`

21 Pango-Schriftart

21.1 pfont:Describe

BEZEICHNUNG

pfont:Describe – Gibt die Schriftartbeschreibung zurück (Größe in Punkt)

ÜBERSICHT

```
desc = pfont:Describe()
```

BESCHREIBUNG

Gibt eine Beschreibung der Schriftart zurück, wobei die Schriftgröße in Punkt angegeben wird.

Verwenden Sie pfont:DescribeWithAbsoluteSize(), wenn Sie die Schriftgröße in Geräteeinheiten wünschen.

EINGABEN

Keine

RÜCKGABEWERTE

`desc` ein neu zugewiesenes Pango-Schriftbeschreibungsobjekt

21.2 pfont:DescribeWithAbsoluteSize

BEZEICHNUNG

pfont:DescribeWithAbsoluteSize – Gibt die Schriftartbeschreibung zurück (Größe in Geräteeinheiten)

ÜBERSICHT

```
desc = pfont:DescribeWithAbsoluteSize()
```

BESCHREIBUNG

Gibt eine Beschreibung der Schriftart zurück, wobei die absolute Schriftgröße in Geräteeinheiten angegeben wird.

Verwenden Sie pfont:Describe(), wenn Sie die Schriftgröße in Punkt angeben möchten.

Diese Funktion gibt ein neu zugewiesenes Pango-Schriftbeschreibungsobjekt zurück. Sie sind dafür verantwortlich, dass dieses Objekt mit pfontdesc:Free() gelöscht wird.

EINGABEN

Keine

RÜCKGABEWERTE

`desc` ein neu zugewiesenes Pango-Schriftbeschreibungsobjekt

21.3 pfont:Free

BEZEICHNUNG
pfont:Free – Verringert die Anzahl der Schriftartenzuordnung

ÜBERSICHT
```
pfont:Free()
```

BESCHREIBUNG
Verringert die Anzahl der Zuordnung einer Pango-Schriftart um eins.

Ist das Ergebnis gleich Null, werden die Schriftartenzuordnung gelöscht und der gesamte zugehörige Speicher freigegeben.

EINGABEN
Keine

21.4 pfont:GetCoverage

BEZEICHNUNG
pfont:GetCoverage – Gibt die Abdeckung des Zeichensatzes zurück

ÜBERSICHT
```
cov = pfont:GetCoverage(language)
```

BESCHREIBUNG
Berechnet die Abdeckung des Zeichensatzes für eine bestimmte Schriftart und ein Sprach-Tag. Der Parameter `language` muss auf ein Pango-Sprachobjekt gesetzt werden.

Diese Funktion gibt ein neu zugewiesenes Pango-Abdeckungs-Objekt zurück. Sie sind dafür verantwortlich, dass dieses Objekt mit pcoverage:Free(). gelöscht wird.

EINGABEN

`language` das Sprachen-Tag

RÜCKGABEWERTE

`cov` eine neu zugewiesene Pango-Abdeckung

21.5 pfont:GetFontMap

BEZEICHNUNG
pfont:GetFontMap – Gibt den Zeichensatz der Schriftart zurück

ÜBERSICHT
```
fontmap = pfont:GetFontMap()
```

BESCHREIBUNG
Gibt den Zeichensatz zurück, für die die Schriftart erstellt wurde.

Beachten Sie, dass die Schriftart einen *schwachen* Verweis auf die Schriftartenzuordnung beibehält. Wenn also alle Verweise auf die Schriftartenzuordnung gelöscht werden, wird die Schriftartenzuordnung beendet, selbst wenn es Schriftarten gibt, die mit der

Schriftartenzuordnung erstellt wurden und noch aktiv sind. In diesem Fall wird diese Funktion ein NULL-Objekt zurückgeben.

Es liegt in der Verantwortung des Benutzers, dafür zu sorgen, dass die Schriftartenzuordnung aufrechterhalten wird. In den meisten Fällen ist dies kein Problem, da ein Pango-Kontext einen Verweis auf den Zeichensatz der Schriftart enthält.

Sie dürfen den zurückgegebene Zeichensatz nicht löschen bzw. freigeben.

EINGABEN

Keine

RÜCKGABEWERTE

`fontmap` den Pango-Zeichensatz

21.6 pfont:GetGlyphExtents

BEZEICHNUNG

pfont:GetGlyphExtents – Gibt die Glyphenausmaße zurück

ÜBERSICHT

```
ink_rect, logical_rect = pfont:GetGlyphExtents(glyph)
```

BESCHREIBUNG

Ermittelt die logischen und drucktechnischen Ausmaße einer Glyphe innerhalb einer Schriftart. Diese Funktion gibt zwei Tabellen zurück, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Das Koordinatensystem für jedes Rechteck hat seinen Ursprung an der Grundlinie und dem horizontalen Ursprung des Zeichens mit ansteigenden Koordinaten nach rechts und unten. Die Makros PANGO_ASCENT(), PANGO_DESCENT(), PANGO_LBEARING() und PANGO_RBEARING() können verwendet werden, um von dem Ausmaß-Rechteck in traditionellere Schriftmetriken zu konvertieren. Die Einheiten der Rechtecke sind in 1/#PANGO_SCALE einer Geräteeinheit.

Wenn die Schriftart ein NULL-Objekt ist, setzt diese Funktion einige vernünftige Werte in den Ausgabevariablen und kehrt zurück.

EINGABEN

`glyph` der Glyphenindex

RÜCKGABEWERTE

`ink_rect` Tabelle mit den Ausmaßen der gezeichneten Glyphe

`logical_rect`
Tabelle mit den logischen Ausmaßen der Glyphe

21.7 pfont:GetMetrics

BEZEICHNUNG

pfont:GetMetrics – Gibt die Schriftartmetriken zurück

ÜBERSICHT

```
metrics = pfont:GetMetrics([language])
```

BESCHREIBUNG

Gibt die allgemeine metrische Informationen für eine Schriftart zurück.

Da die Metriken für verschiedene Skripte sehr unterschiedlich sein können, kann ein Sprach-Tag angegeben werden, um anzuzeigen, dass die Metriken abgerufen werden sollen, die dem/den von dieser Sprache verwendeten Skript(s) entsprechen. Wenn er angegeben wird, muss der Parameter `language` ein Pango-Sprachobjekt sein.

Wenn die Schriftart ein NULL-Objekt ist, setzt diese Funktion einige vernünftige Werte in den Ausgabevariablen und kehrt zurück.

Diese Funktion gibt ein Pango-Fontmetrics-Objekt zurück. Der Aufrufer muss pfontmetrics:Free() aufrufen, wenn er das Objekt nicht mehr benötigt.

EINGABEN

`language` optional: Sprach-Tag, mit dem festgelegt wird, für welches Skript die Metriken abgerufen werden sollen; wenn dieser Parameter weggelassen wird, werden Metriken für die gesamte Schrift zurückgegeben

RÜCKGABEWERTE

`metrics` ein Pango-Schriftmetrik-Objekt

21.8 pfont:GetScaledFont

BEZEICHNUNG

pfont:GetScaledFont – Gibt die skalierte Schrift zurück

ÜBERSICHT

```
handle = pfont:GetScaledFont()
```

BESCHREIBUNG

Ermitelt das von der Pango-Schriftart verwendete Cairo-Scaled-Font-Objekt. Die skalierte Schriftart kann mit cscaledfont:Reference() referenziert und verwendet werden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` ein Cairo-Scaled-Font-Objekt

21.9 pfont:HasChar

BEZEICHNUNG

pfont:HasChar – Prüft, ob die Schriftart Glyphen hat

ÜBERSICHT

```
ok = pfont:HasChar(wc)
```

BESCHREIBUNG
Gibt zurück, ob die Schriftart eine Glyphe für dieses Zeichen bereitstellt.

EINGABEN

`wc` ein Unicode-Zeichen

RÜCKGABEWERTE

`ok` `True`, wenn die Schriftart die Glyphe darstellen kann, sonst `False`

21.10 pfont:IsNull

BEZEICHNUNG
pfont:IsNull – Prüft, ob die Schriftart ungültig ist

ÜBERSICHT

```
bool = pfont:IsNull()
```

BESCHREIBUNG
Gibt `True` zurück, wenn die Schriftart `NULL` ist, d.h. ungültig. Wenn Funktionen, die Schriftarten zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Schriftart einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Zuweisung der Schriftart fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN
Keine

RÜCKGABEWERTE

`bool` `True` wenn die Schriftart `NULL` ist, sonst `False`

21.11 pfont:Reference

BEZEICHNUNG
pfont:Reference – Erhöht die Anzahl der Referenzen der Pango-Schriftarten

ÜBERSICHT

```
pfont:Reference()
```

BESCHREIBUNG
Erhöht die Anzahl der Verweise auf die Pango-Schriftarten um eins.

EINGABEN
Keine

22 Pango-Schriftbeschreibungen

22.1 pfontdesc:BetterMatch

BEZEICHNUNG

pfontdesc:BetterMatch – Vergleicht zwei Schriftbeschreibungen miteinander

ÜBERSICHT

```
ok = pfontdesc:BetterMatch(old_match, new_match)
```

BESCHREIBUNG

Ermittelt, ob die Stilattribute von `new_match` der Schriftbeschreibung besser entsprechen als die von `old_match`, oder, wenn `old_match` gleich Null ist, ob `new_match` überhaupt eine Übereinstimmung ist.

Eine ungefähre Übereinstimmung wird für die Stärke und Stil vorgenommen; andere Stilattribute müssen genau übereinstimmen. Stilattribute sind alle Attribute mit Ausnahme von familien- und größenbezogenen Attributen. Der ungefähre Abgleich für den Stil berücksichtigt `#PANGO_STYLE_OBLIQUE` und `#PANGO_STYLE_ITALIC` als Übereinstimmungen, aber nicht so gut, wie wenn die Stile gleich sind.

Beachten Sie, dass `old_match` mit der Schriftartbeschreibung übereinstimmen muss.

EINGABEN

`old_match`
: ein Pango-Schriftbeschreibungsobjekt oder `Nil`

`new_match`
: ein Pango-Schriftbeschreibungsobjekt

RÜCKGABEWERTE

`ok` `True`, wenn `new_match` eine bessere Übereinstimmung ist, sonst `False`

22.2 pfontdesc:Copy

BEZEICHNUNG

pfontdesc:Copy – Kopiert eine Schriftartbeschreibung

ÜBERSICHT

```
desc = pfontdesc:Copy()
```

BESCHREIBUNG

Erstellt eine Kopie der Beschreibung einer Pango-Schriftart.

Diese Funktion gibt die neu zugewiesene Pango-Schriftbeschreibung zurück, die mit pfontdesc:Free() wieder gelöscht werden sollte.

EINGABEN

Keine

RÜCKGABEWERTE

`desc` die neu zugewiesene Beschreibung der Pango-Schriftart

22.3 pfontdesc:Equal

BEZEICHNUNG

pfontdesc:Equal – Vergleicht zwei Schriftbeschreibungen auf Gleichheit

ÜBERSICHT

```
ok = pfontdesc:Equal(desc2)
```

BESCHREIBUNG

Vergleicht zwei Schriftbeschreibungen auf Gleichheit.

Zwei Schriftartenbeschreibungen gelten als gleich, wenn die von ihnen beschriebenen Schriftarten nachweislich identisch sind. Das bedeutet, dass ihre Masken nicht übereinstimmen müssen, solange die anderen Felder alle gleich sind. (Zwei Schriftbeschreibungen können dazu führen, dass identische Schriftarten geladen werden, aber dennoch `False` ergeben).

Diese Funktion gibt `True` zurück, wenn die beiden Schriftbeschreibungen identisch sind, andernfalls `False`.

EINGABEN

`desc2`	ein weiteres Pango-Schriftbeschreibungsobjekt

RÜCKGABEWERTE

`ok`	`True`, wenn die beiden Schriftbeschreibungen identisch sind, sonst `False`

22.4 pfontdesc:Free

BEZEICHNUNG

pfontdesc:Free – Löscht eine Schriftbeschreibungen

ÜBERSICHT

```
pfontdesc:Free()
```

BESCHREIBUNG

Löscht eine Schriftbeschreibung.

EINGABEN

Keine

22.5 pfontdesc:GetFamily

BEZEICHNUNG

pfontdesc:GetFamily – Gibt den Familienname zurück

ÜBERSICHT

```
family$ = pfontdesc:GetFamily()
```

BESCHREIBUNG

Ruft das Feld für den Familiennamen einer Schriftbeschreibung ab. Siehe pfontdesc:SetFamily() für weitere Informationen.

Diese Funktion gibt das Feld für den Familiennamen der Schriftartbeschreibung zurück, oder Null, wenn es vorher nicht gesetzt wurde.

EINGABEN

Keine

RÜCKGABEWERTE

`family$` der Familienname

22.6 pfontdesc:GetGravity

BEZEICHNUNG

pfontdesc:GetGravity – Gibt die Ausrichtung der Schriftart zurück

ÜBERSICHT

```
gravity = pfontdesc:GetGravity()
```

BESCHREIBUNG

Gibt das Ausrichtungsfeld einer Schriftbeschreibung zurück. Siehe pfontdesc:SetGravity() für weitere Informationen.

Diese Funktion gibt das Ausrichtungsfeld für die Schriftbeschreibung zurück. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`gravity` das Ausrichtungsfeld für die Schriftbeschreibung

22.7 pfontdesc:GetSetFields

BEZEICHNUNG

pfontdesc:GetSetFields – Stellt fest, welche Felder gesetzt wurden

ÜBERSICHT

```
mask = pfontdesc:GetSetFields()
```

BESCHREIBUNG

Ermittelt, welche Felder in einer Schriftbeschreibung gesetzt wurden.

Diese Funktion gibt eine Bitmaske mit gesetzten Bits zurück, die den Feldern in der Schriftbeschreibung entsprechen, die gesetzt wurden. Die folgenden Bits können gesetzt werden:

`#PANGO_FONT_MASK_FAMILY`
: Die Schriftfamilie ist festgelegt.

`#PANGO_FONT_MASK_STYLE`
: Der Schriftstil ist festgelegt.

`#PANGO_FONT_MASK_VARIANT`
: Die Schriftvariante ist festgelegt.

`#PANGO_FONT_MASK_WEIGHT`
: Der Schriftstärke ist festgelegt.

`#PANGO_FONT_MASK_STRETCH`
: Die Schriftausdehnung ist festgelegt.

`#PANGO_FONT_MASK_SIZE`
: Die Schriftgröße ist festgelegt.

`#PANGO_FONT_MASK_GRAVITY`
: Die Ausrichtung der Schrift ist festgelegt.

`#PANGO_FONT_MASK_VARIATIONS`
: OpenType-Schriftvarianten sind festgelegt.

EINGABEN

Keine

RÜCKGABEWERTE

`mask` eine Bitmaske mit gesetzten Bits, die den eingestellten Feldern entsprechen (siehe oben)

22.8 pfontdesc:GetSize

BEZEICHNUNG

pfontdesc:GetSize – Gibt das Größefeld zurück

ÜBERSICHT

```
size = pfontdesc:GetSize()
```

BESCHREIBUNG

Ruft das Größenfeld einer Schriftbeschreibung ab. Siehe pfontdesc:SetSize() für weitere Informationen.

Diese Funktion gibt die Größen für die Schriftbeschreibung in Punkt oder Geräteeinheiten zurück. Sie müssen pfontdesc:GetSizeIsAbsolute() aufrufen, um herauszufinden, was der Fall ist. Gibt 0 zurück, wenn das Größenfeld noch nicht gesetzt oder wenn es explizit auf 0 gesetzt wurde. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`size` die Größe in Punkt

22.9 pfontdesc:GetSizeIsAbsolute

BEZEICHNUNG

pfontdesc:GetSizeIsAbsolute – Stellt fest, ob die Größe in Geräteeinheiten angegeben ist

ÜBERSICHT

```
isabs = pfontdesc:GetSizeIsAbsolute()
```

BESCHREIBUNG

Bestimmt, ob die Größe der Schrift in Punkten (nicht absolut) oder in Geräteeinheiten (absolut) angegeben wird.

Siehe pfontdesc:SetSize() und pfontdesc:SetAbsoluteSize() für mehr Informationen.

Diese Funktion gibt zurück, ob die Größe für die Schriftbeschreibung in Punkten oder Geräteeinheiten angegeben ist. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Größenfeld der Schriftbeschreibung explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`isabs` ob die Größe für die Schriftbeschreibung in absoluten Einheiten angegeben werden soll

22.10 pfontdesc:GetStretch

BEZEICHNUNG

pfontdesc:GetStretch – Gibt das Feld der Schriftausdehnung zurück

ÜBERSICHT

```
stretch = pfontdesc:GetStretch()
```

BESCHREIBUNG

Ruft das Schriftausdehnungfeld einer Schriftbeschreibung ab. Siehe pfontdesc:SetStretch() für weitere Informationen.

Diese Funktion gibt das Schriftausdehnungfeld für die Schriftbeschreibung zurück. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`stretch` das Schriftausdehnung für die Schriftbeschreibung

22.11 pfontdesc:GetStyle

BEZEICHNUNG

pfontdesc:GetStyle – Gibt das Stilfeld zurück

ÜBERSICHT

```
style = pfontdesc:GetStyle()
```

BESCHREIBUNG

Ruft das Stilfeld einer Pango-Schriftbeschreibung ab. Siehe pfontdesc:SetStyle() für weitere Informationen.

Diese Funktion gibt das Stilfeld für die Schriftbeschreibung zurück. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`style` das Stilfeld für die Schriftbeschreibung

22.12 pfontdesc:GetVariant

BEZEICHNUNG

pfontdesc:GetVariant – Gibt das Schriftvariantefeld zurück

ÜBERSICHT

```
var = pfontdesc:GetVariant()
```

BESCHREIBUNG

Ruft das Schriftvariantefeld einer Pango-Schriftbeschreibung ab. Siehe pfontdesc:SetVariant() für weitere Informationen.

Diese Funktion gibt das Variantenfeld für die Schriftbeschreibung zurück. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN

Keine

RÜCKGABEWERTE

`var` das Schriftvariantefeld für die Schriftbeschreibung

22.13 pfontdesc:GetVariations

BEZEICHNUNG

pfontdesc:GetVariations – Gibt das Schriftvariantenfeld zurück

ÜBERSICHT

```
var$ = pfontdesc:GetVariations()
```

BESCHREIBUNG

Ruft das Feld Variationen einer Schriftbeschreibung ab. Siehe pfontdesc:SetVariations() für weitere Informationen.

Diese Funktion gibt das OpenType-Schriftvariantenfeld für die Schriftbeschreibung zurück, oder Null, wenn es vorher nicht gesetzt wurde.

EINGABEN
Keine

RÜCKGABEWERTE
`var$` das Schriftvariantenfeld für die Beschreibung der Schriftart

22.14 pfontdesc:GetWeight

BEZEICHNUNG
pfontdesc:GetWeight – Gibt das Schriftstärkefeld zurück

ÜBERSICHT
```
weight = pfontdesc:GetWeight()
```

BESCHREIBUNG
Ruft das Feld Schriftstärke einer Schriftbeschreibung ab. Siehe pfontdesc:SetWeight() für weitere Informationen.

Diese Funktion gibt das Schriftstärkefeld für die Schriftbeschreibung zurück. Verwenden Sie pfontdesc:GetSetFields(), um herauszufinden, ob das Feld explizit gesetzt wurde oder nicht.

EINGABEN
Keine

RÜCKGABEWERTE
`weight` das Feld Schriftstärke für die Beschreibung der Schriftart

22.15 pfontdesc:IsNull

BEZEICHNUNG
pfontdesc:IsNull – Prüft, ob die Schriftbeschreibung ungültig ist

ÜBERSICHT
```
bool = pfontdesc:IsNull()
```

BESCHREIBUNG
Gibt `True` zurück, wenn die Schriftbeschreibung `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN
Keine

RÜCKGABEWERTE
`bool` `True` wenn die Schriftbeschreibung `NULL` ist, sonst `False`

22.16 pfontdesc:Merge

BEZEICHNUNG

pfontdesc:Merge – Fügt Felder in die Schriftbeschreibung ein

ÜBERSICHT

```
pfontdesc:Merge(desc_to_merge, replace_existing)
```

BESCHREIBUNG

Fügt die Felder, die in `desc_to_merge` festgelegt sind, in die Felder der Schriftbeschreibung ein.

Wenn `replace_existing` `False` ist, sind nur Felder in der Schriftbeschreibung betroffen, die noch nicht gesetzt sind. Ist `replace_existing` `True`, werden auch Felder ersetzt, die bereits gesetzt sind.

EINGABEN

`desc_to_merge`
: das Pango-Schriftbeschreibungsobjekt

`replace_existing`
: wenn `True`, werden die Felder in der Schriftbeschreibung durch die entsprechenden Werte aus `desc_to_merge` ersetzt, auch wenn sie bereits vorhanden sind

22.17 pfontdesc:SetAbsoluteSize

BEZEICHNUNG

pfontdesc:SetAbsoluteSize – Stellt die absolute Größe ein

ÜBERSICHT

```
pfontdesc:SetAbsoluteSize(size)
```

BESCHREIBUNG

Legt das Größenfeld einer Schriftbeschreibung in Geräteeinheiten fest.

Dies schließt sich gegenseitig mit pfontdesc:SetSize() aus, der die Schriftgröße in Punkten festlegt.

EINGABEN

`size`
: die neue Größe in Pango-Einheiten; eine Geräteeinheit besteht aus `#PANGO_SCALE` Pango-Einheiten; für ein Ausgabe-Backend, bei dem eine Geräteeinheit ein Pixel ist, ergibt ein Größenwert von 10 * `#PANGO_SCALE` eine 10-Pixel-Schrift

22.18 pfontdesc:SetFamily

BEZEICHNUNG

pfontdesc:SetFamily – Stellt die Schriftfamilie ein

ÜBERSICHT

```
pfontdesc:SetFamily(family$)
```

BESCHREIBUNG

Legt das Feld für den Schriftfamilienamen einer Schriftartbeschreibung fest.

Der Familienname steht für eine Familie von verwandten Schriftstilen und wird in eine bestimmte Pango-Schriftfamilie aufgelöst. Bei einigen Verwendungen der Pango-Schriftartbeschreibung ist es auch möglich, eine durch Kommata getrennte Liste von Familiennamen für dieses Feld zu verwenden.

EINGABEN

`family$` eine Zeichenkette, die den Schriftfamilienamen darstellt

22.19 pfontdesc:SetGravity

BEZEICHNUNG

pfontdesc:SetGravity – Stellt die Ausrichtung der Schrift ein

ÜBERSICHT

```
pfontdesc:SetGravity(gravity)
```

BESCHREIBUNG

Legt das Ausrichtungsfeld einer Schriftbeschreibung fest.

Das Gravitationsfeld gibt an, wie die Glyphen gedreht werden sollen. Wenn `gravity` den Wert `#PANGO_GRAVITY_AUTO` hat, wird die Ausrichtungsmaske in der Schriftbeschreibung deaktiviert.

Diese Funktion ist für den Benutzer nur selten nützlich. Die Ausrichtung sollte normalerweise in einem Pango-Kontext eingestellt werden.

Siehe pcontext:SetBaseGravity() für eine Liste von Konstanten, die im Argument `gravity` übergeben werden können.

EINGABEN

`gravity` die Ausrichtung für die Schriftbeschreibung

22.20 pfontdesc:SetSize

BEZEICHNUNG

pfontdesc:SetSize – Stellt die Größe ein

ÜBERSICHT

```
pfontdesc:SetSize(size)
```

BESCHREIBUNG

Legt das Größenfeld einer Schriftbeschreibung in Punktbruchteilen fest.

Dies schließt sich gegenseitig mit pfontdesc:SetAbsoluteSize() aus.

EINGABEN

`size` die Größe der Schrift in Punkten, skaliert mit `#PANGO_SCALE` (d.h. ein Größenwert von 10 * `#PANGO_SCALE` ist eine 10-Punkt-Schrift; der Umrechnungsfaktor zwischen Punkten und Geräteeinheiten hängt von der Systemkonfiguration und dem Ausgabegerät ab; für die Bildschirmdarstellung ist ein logischer DPI von 96 üblich, in diesem Fall entspricht eine 10-Punkt-Schrift einer 10 * (96 / 72) = 13,3-Pixel-Schrift. Verwenden Sie pfontdesc:SetAbsoluteSize(), wenn Sie eine bestimmte Größe in Geräteeinheiten benötigen

22.21 pfontdesc:SetStretch

BEZEICHNUNG

pfontdesc:SetStretch – Stellt das Schriftausdehnungfeld ein

ÜBERSICHT

`pfontdesc:SetStretch(stretch)`

BESCHREIBUNG

Legt das Schriftausdehnungfeld einer Schriftbeschreibung fest. Das Schriftausdehnungfeld gibt an, wie schmal oder breit die Schrift sein soll.

Die folgenden Konstanten können im Parameter `stretch` übergeben werden:

`#PANGO_STRETCH_ULTRA_CONDENSED`
: Ultraschmal

`#PANGO_STRETCH_EXTRA_CONDENSED`
: Extraschmal

`#PANGO_STRETCH_CONDENSED`
: Schmal

`#PANGO_STRETCH_SEMI_CONDENSED`
: Halbschmal

`#PANGO_STRETCH_NORMAL`
: Normal

`#PANGO_STRETCH_SEMI_EXPANDED`
: Halbbreit

`#PANGO_STRETCH_EXPANDED`
: Breit

`#PANGO_STRETCH_EXTRA_EXPANDED`
: Extrabreit

`#PANGO_STRETCH_ULTRA_EXPANDED`
: Ultrabreit

EINGABEN

`stretch` die Schriftausdehnung für die Schriftbeschreibung (siehe oben)

22.22 pfontdesc:SetStyle

BEZEICHNUNG

pfontdesc:SetStyle – Stellt das Stilfeld ein

ÜBERSICHT

```
pfontdesc:SetStyle(style)
```

BESCHREIBUNG

Legt das Stilfeld einer Pango-Schriftbeschreibung fest.

Das Stilfeld beschreibt, ob die Schrift schräg ist und auf welche Weise. Es kann entweder `#PANGO_STYLE_NORMAL`, `#PANGO_STYLE_ITALIC` oder `#PANGO_STYLE_OBLIQUE` sein.

Die meisten Schriften haben entweder einen kursiven oder einen schrägen (oblique) Stil, aber nicht beides. Der Schriftabgleich in Pango gleicht kursive Spezifikationen mit schrägen Schriften ab und umgekehrt, wenn keine exakte Übereinstimmung gefunden wird.

EINGABEN

`style` der Stil für die Schriftbeschreibung (siehe oben)

22.23 pfontdesc:SetVariant

BEZEICHNUNG

pfontdesc:SetVariant – Stellt das Schriftvariantenfeld ein

ÜBERSICHT

```
pfontdesc:SetVariant(variant)
```

BESCHREIBUNG

Legt das Schriftvarianten einer Schriftbeschreibung fest.

Das Variantenfeld kann entweder `#PANGO_VARIANT_NORMAL` oder `#PANGO_VARIANT_SMALL_CAPS` sein.

EINGABEN

`variant` den Schriftvariantentyp für die Schriftbeschreibung (siehe oben)

22.24 pfontdesc:SetVariations

BEZEICHNUNG

pfontdesc:SetVariations – Stellt das Schriftvariantenfeld ein

ÜBERSICHT

```
pfontdesc:SetVariations(variations$)
```

BESCHREIBUNG

Legt das Schriftvariantenfeld einer Schriftbeschreibung fest.

OpenType-Schriftvarianten ermöglichen die Auswahl einer Schriftart durch die Angabe von Werten für eine Reihe von Achsen, wie Breite oder Stärke.

Das Format der Schriftvariantenkette ist

```
AXIS1=VALUE,AXIS2=VALUE...
```

wobei jedes AXIS ein 4-Zeichen-Tag ist, das eine Schriftachse identifiziert, und jedes VALUE eine Fließkommazahl. Unbekannte Achsen werden ignoriert und die Werte werden auf ihren zulässigen Bereich begrenzt.

Pango verfügt derzeit nicht über eine Möglichkeit, unterstützte Achsen einer Schriftart zu finden. Sowohl harfbuzz als auch freetype haben eine API für diese Aufgabe.

EINGABEN

`variations$`
: eine Zeichenkette, die die Schriftvariante darstellt

22.25 pfontdesc:SetWeight

BEZEICHNUNG

pfontdesc:SetWeight – Stellt das Stärkefeld ein

ÜBERSICHT

`pfontdesc:SetWeight(weight)`

BESCHREIBUNG

Legt das Stärkefeld einer Schriftbeschreibung fest.

Das Feld Stärke gibt an, wie fett oder hell die Schrift sein soll. Dies kann ein numerischer Wert zwischen 100 und 1000 oder eine der folgenden vordefinierten Stärkekonstanten sein:

`#PANGO_WEIGHT_THIN`
: fein, dünn (= 100).

`#PANGO_WEIGHT_ULTRALIGHT`
: ultraleicht (= 200).

`#PANGO_WEIGHT_LIGHT`
: leicht, mager (= 300).

`#PANGO_WEIGHT_SEMILIGHT`
: halbleicht, halbmager (= 350).

`#PANGO_WEIGHT_BOOK`
: Buch, Werk (= 380).

`#PANGO_WEIGHT_NORMAL`
: normal, regulär (= 400).

`#PANGO_WEIGHT_MEDIUM`
: medium, leichthalbfett (= 500).

`#PANGO_WEIGHT_SEMIBOLD`
: halbfett (= 600).

`#PANGO_WEIGHT_BOLD`
: fett (= 700).

`#PANGO_WEIGHT_ULTRABOLD`
: ultrafett (= 800).

`#PANGO_WEIGHT_HEAVY`
: kräftig, fett, schwer (= 900).

`#PANGO_WEIGHT_ULTRAHEAVY`
: ultrafett (= 1000).

EINGABEN

`weight` die Schriftstärke für die Schriftbeschreibung (siehe oben)

22.26 pfontdesc:ToFilename

BEZEICHNUNG

pfontdesc:ToFilename – Erstellt einen Dateiname der Schriftbeschreibung

ÜBERSICHT

```
f$ = pfontdesc:ToFilename()
```

BESCHREIBUNG

Erzeugt eine Dateinamendarstellung einer Schriftartbeschreibung.

Der Dateiname ist identisch mit dem Ergebnis des Aufrufs der Funktion pfontdesc:ToString(), jedoch mit Unterstrichen anstelle von Zeichen, die in Dateinamen untypisch sind, und nur in Kleinbuchstaben.

EINGABEN

Keine

RÜCKGABEWERTE

`f$` Dateinamen der Schriftartbeschreibung

22.27 pfontdesc:ToString

BEZEICHNUNG

pfontdesc:ToString – Erstellt eine Zeichenkette der Schriftbeschreibung

ÜBERSICHT

```
s$ = pfontdesc:ToString()
```

BESCHREIBUNG

Erzeugt eine Zeichenkettedarstellung einer Schriftartbeschreibung.

Siehe pango.FontDescription() für eine Beschreibung des Formats der Zeichenkettedarstellung. Die Familienliste in der Zeichenkettebeschreibung wird nur dann mit einem Komma abgeschlossen, wenn das letzte Wort der Liste eine gültige Stiloption ist.

EINGABEN

Keine

RÜCKGABEWERTE

`s$` Zeichenkettedarstellung der Schriftartbeschreibung

22.28 pfontdesc:UnsetFields

BEZEICHNUNG

pfontdesc:UnsetFields – Setzt einige Felder zurück

ÜBERSICHT

```
pfontdesc:UnsetFields(to_unset)
```

BESCHREIBUNG

Setzt einige der Felder in einer Pango-Schriftbeschreibung zurück, somit werden die nicht gesetzten Felder auf ihre Standardwerte zurückgesetzt.

Siehe pfontdesc:GetSetFields() für eine Liste der unterstützten Felder.

EINGABEN

`to_unset` Bitmaske der zu löschenden Felder in der Schriftbeschreibung

23 Pango-Schriftschnitte

23.1 pfontface:Describe

BEZEICHNUNG
pfontface:Describe – Gibt die Schriftschnittbeschreibung zurück

ÜBERSICHT
```
desc = pfontface:Describe()
```

BESCHREIBUNG
Gibt die Schriftschnittbeschreibung zurück, die dem Schriftschnitt entspricht.

Die sich daraus ergebende Schriftschnitbeschreibung enthält die Familie, den Stil, die Variante, die Stärke und die Dehnung des Schriftschnitts, aber das Größenfeld ist nicht gesetzt.

Diese Funktion gibt eine neu erstellte Pango-Schriftbeschreibungsstruktur zurück, die die Beschreibung des Schriftschnitts enthält. Verwenden Sie pfontdesc:Free(), um das Ergebnis zu löschen.

EINGABEN
Keine

RÜCKGABEWERTE

`desc` eine neu erstellte Pango-Schriftbeschreibungsstruktur

23.2 pfontface:GetFaceName

BEZEICHNUNG
pfontface:GetFaceName – Gibt den Namen des Schriftschnitts zurück

ÜBERSICHT
```
name$ = pfontface:GetFaceName()
```

BESCHREIBUNG
Ermittelt den Namen, der den Stil dieses Schriftschnitts darstellt.

Beachten Sie, dass eine Schriftfamilie mehrere Schriftschnitte mit demselben Namen enthalten kann (z.B. ein variabler und ein nicht-variabler Schriftschnitt für denselben Stil).

EINGABEN
Keine

RÜCKGABEWERTE

`name$` der Name des Schriftschnitts

23.3 pfontface:IsNull

BEZEICHNUNG

pfontface:IsNull – Prüft, ob der Schriftschnitt ungültig ist

ÜBERSICHT

```
bool = pfontface:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn der Schriftschnitt `NULL` ist, d.h. ungültig. Wenn Funktionen, die Schriftschnitte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen den Schriftschnitt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob der Schriftschnitt fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True` wenn die Schriftschnitt `NULL` ist, sonst `False`

23.4 pfontface:IsSynthesized

BEZEICHNUNG

pfontface:IsSynthesized – Prüft, ob der Schriftschnitt synthetisiert wurde

ÜBERSICHT

```
ok = pfontface:IsSynthesized()
```

BESCHREIBUNG

Gibt zurück, ob ein Pango-Schriftschnitt synthetisiert wurde.

Dies ist dann der Fall, wenn das zugrundeliegende Schriftart-Rendering-Modul diesen Schriftschnitt aus einem anderen Schriftschnitt erzeugt, indem es ihn schert, hervorhebt, aufhellt oder in anderer Weise verändert.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` boolescher Wert, der angibt, ob der Schriftschnitt synthetisiert wurde (`True`) oder nicht (`False`)

23.5 pfontface:ListSizes

BEZEICHNUNG

pfontface:ListSizes – Gibt die Grössen von Bitmap-Schriften zurück

ÜBERSICHT

```
t = pfontface:ListSizes()
```

BESCHREIBUNG

Listet die verfügbaren Größen für eine Schriftart auf. Diese Funktion gibt eine Tabelle mit einer Liste von Größen für die Schriftart zurück. Die zurückgegebenen Größen sind in Pango-Einheiten angegeben und werden in aufsteigender Reihenfolge sortiert.

Dies gilt nur für Bitmap-Schriften. Bei skalierbaren Schriftarten wird eine leere Tabelle zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`t`	table containing a list of font sizes

24 Pango-Schriftartenfamilie

24.1 pfontfamily:GetName

BEZEICHNUNG

pfontfamily:GetName – Gibt den Namen der Schriftartenfamilie zurück

ÜBERSICHT

```
name$ = pfontfamily:GetName()
```

BESCHREIBUNG

Ermittelt den Namen der Schriftartenfamilie.

Der Name ist unter allen Schriften für das Schriftarten-Backend eindeutig und kann in einer Pango-Schriftbeschreibung verwendet werden, um anzugeben, dass eine Schrift aus dieser Familie gewünscht ist.

EINGABEN

Keine

RÜCKGABEWERTE

`name$` der Name der Schriftartenfamilie

24.2 pfontfamily:IsMonospace

BEZEICHNUNG

pfontfamily:IsMonospace – Prüft, ob die Familie monospace ist (nichtproportional)

ÜBERSICHT

```
ok = pfontfamily:IsMonospace()
```

BESCHREIBUNG

Eine Monospace-Schrift ist eine Schriftart für die Textdarstellung, bei der die Zeichen ein regelmäßiges Raster bilden und somit nichtproportional sind.

Für westliche Sprachen würde dies bedeuten, dass die Vorlaufbreite aller Zeichen gleich ist, aber diese Kategorisierung umfasst auch asiatische Schriftarten, die Zeichen mit doppelter Breite enthalten: Zeichen, die zwei Rasterzellen belegen.

Die Größe der Rasterzellen lässt sich am besten durch die Funktion pfontmetrics:GetApproximateDigitWidth() ermitteln, da die Ergebnisse von pfontmetrics:GetApproximateCharWidth() durch Zeichen mit doppelter Breite beeinflusst werden können.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` `True`, wenn die Familie monospace ist, sonst `False`

24.3 pfontfamily:IsNull

BEZEICHNUNG

pfontfamily:IsNull – Prüft, ob die Schriftartenfamilie ungültig ist

ÜBERSICHT

```
bool = pfontfamily:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Schriftartenfamilie `NULL` ist, d.h. ungültig. Wenn Funktionen, die Schriftartenfamilie zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Schriftartenfamilie einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Schriftartenfamilie fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True` wenn die Schriftartenfamilie `NULL` ist, sonst `False`

24.4 pfontfamily:IsVariable

BEZEICHNUNG

pfontfamily:IsVariable – Prüft, ob die Familie variabel ist

ÜBERSICHT

```
ok = pfontfamily:IsVariable()
```

BESCHREIBUNG

Eine variable Schriftart ist eine Schriftart, deren Designachsen verändert werden können, um verschiedene Schriftarten zu erzeugen.

Solche Achsen werden auch als Variationen bezeichnet. Siehe pfontdesc:SetVariations() für weitere Informationen.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` `True`, wenn die Familie variabel ist, sonst `False`

24.5 pfontfamily:ListFaces

BEZEICHNUNG

pfontfamily:ListFaces – Gibt die Schriftschnitte innerhalb der Schriftartfamilie zurück

ÜBERSICHT

```
t = pfontfamily:ListFaces()
```

BESCHREIBUNG

Listet die verschiedenen Schriftschnitte auf, aus denen die Schriftartfamilie besteht. Die Schriftschnitte einer Familie haben ein gemeinsames Design, unterscheiden sich aber in Neigung, Stärke, Breite und anderen Aspekten.

Diese Funktion gibt eine Tabelle zurück, die alle Familien-Schriftschnitte als Pango-Schriftschnitt-Objekte enthält. Sie dürfen die einzelnen Pango-Schriftschnitt-Objekte nicht löschen, da sie zur Schriftartenfamilie gehören. Beachten Sie, dass die zurückgegebenen Schriftschnitte nicht in einer bestimmten Reihenfolge stehen und mehrere Schriftschnitte denselben Namen oder dieselben Merkmale haben können.

EINGABEN

Keine

RÜCKGABEWERTE

`t` Tabelle mit einer Liste von Pango-Schriftschnitt-Objekten

25 Pango-Schriftartenzuordnung

25.1 pfontmap:Changed

BEZEICHNUNG
pfontmap:Changed – Erzwingt eine Änderung des Kontexts

ÜBERSICHT
`pfontmap:Changed()`

BESCHREIBUNG
Erzwingt eine Änderung des Kontexts, was dazu führt, dass sich jeder Pango-Kontext, der diese Schriftartenzuordnung verwendet, ändert.

Diese Funktion ist nur nützlich, wenn ein neues Backend für Pango implementiert wird, was Anwendungen nicht tun werden. Backends sollten diese Funktion aufrufen, wenn sie zusätzliche Daten an den Kontext angehängt haben und diese Daten geändert werden.

EINGABEN
Keine

25.2 pfontmap:CreateContext

BEZEICHNUNG
pfontmap:CreateContext – Erstellt einen Kontext

ÜBERSICHT
`context = pfontmap:CreateContext()`

BESCHREIBUNG
Erzeugt einen Pango-Kontext, der mit der Schriftartenzuordnung verbunden ist.

Dies ist äquivalent zu pango.Context() gefolgt von pcontext:SetFontMap().

Diese Funktion gibt den neu zugewiesenen Pango-Kontext zurück, der mit pcontext:Free() wieder gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE
`context` den neu zugewiesenen Pango-Kontext

25.3 pfontmap:Free

BEZEICHNUNG
pfontmap:Free – Verringert die Anzahl der Schriftartenzuordnung

ÜBERSICHT
`pfontmap:Free()`

BESCHREIBUNG
Verringert die Anzahl der Verweise auf eine Pango-Schriftart um eins.

Ist das Ergebnis gleich Null, werden die Schriftartenzuordnung gelöscht und der gesamte zugehörige Speicher freigegeben.

EINGABEN
Keine

25.4 pfontmap:GetFontType

BEZEICHNUNG
pfontmap:GetFontType – Gibt den Typ der Schriftart zurück

ÜBERSICHT
```
type = pfontmap:GetFontType()
```

BESCHREIBUNG
Gibt den Typ des Cairo-Schriftarten-Backends zurück, den die Schriftartenzuordnung verwendet. Siehe cfontface:GetType() für eine Liste der Cairo-Schrifttypen.

EINGABEN
Keine

RÜCKGABEWERTE

`type` der Cairo-Schriftarten-Backend-Typ (siehe oben)

25.5 pfontmap:GetResolution

BEZEICHNUNG
pfontmap:GetResolution – Gibt die Auflösung zurück

ÜBERSICHT
```
dpi = pfontmap:GetResolution()
```

BESCHREIBUNG
Gibt die Auflösung für die Schriftartenzuordnung zurück.

Siehe pfontmap:SetResolution() für mehr Informationen.

EINGABEN
Keine

RÜCKGABEWERTE

`dpi` die Auflösung in "Punkte pro Zoll"

25.6 pfontmap:GetSerial

BEZEICHNUNG

pfontmap:GetSerial – Gibt die Seriennummer der Schriftartenzuordnung zurück

ÜBERSICHT

```
s = pfontmap:GetSerial()
```

BESCHREIBUNG

Gibt die aktuelle Seriennummer der Schriftartenzuordnung zurück.

Die Seriennummer wird auf eine kleine Zahl größer als Null initialisiert, wenn eine neue Schriftartenzuordnung erstellt wird, und wird jedes Mal erhöht, wenn die Schriftartenzuordnung geändert wird. Sie kann umbrechen, wird aber nie den Wert 0 haben. Da sie umgebrochen werden kann, sollte man sie nie mit "kleiner als" vergleichen, sondern immer "nicht gleich" verwenden.

Die Schriftartenzuordnung kann nur über die Backend-spezifische API geändert werden, z.B. durch Änderung der Auflösung der Schriftartenzuordnung.

Diese Funktion gibt die aktuelle Seriennummer der Schriftartenzuordnung zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`s` die aktuelle Seriennummer der Schriftartenzuordnung

25.7 pfontmap:IsNull

BEZEICHNUNG

pfontmap:IsNull – Prüft, ob die Schriftartenzuordnung ungültig ist

ÜBERSICHT

```
bool = pfontmap:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Schriftartenzuordnung `NULL` ist, d.h. ungültig. Wenn Funktionen, die Schriftartenzuordnung zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen die Schriftartenzuordnung einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Schriftartenzuordnung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True` wenn die Schriftartenzuordnung `NULL` ist, sonst `False`

25.8 pfontmap:ListFamilies

BEZEICHNUNG
pfontmap:ListFamilies – Gibt eine Liste aller Schriftartenfamilie zurück

ÜBERSICHT
```
t = pfontmap:ListFamilies()
```

BESCHREIBUNG
Listet alle Familien für eine Schriftartenzuordnung auf. Die Familien werden als Tabelle zurückgegeben, die eine Liste von Pango-Schriftfamilien-Handles in beliebiger Reihenfolge enthält. Die Schriftfamilien-Handles sind der Schriftartenzuordnung zugewiesen und dürfen nicht gelöscht werden.

EINGABEN
Keine

RÜCKGABEWERTE

`t` Tabelle mit allen Schriftartenfamilien

25.9 pfontmap:LoadFont

BEZEICHNUNG
pfontmap:LoadFont – Lädt eine Schriftart

ÜBERSICHT
```
font = pfontmap:LoadFont(context, desc)
```

BESCHREIBUNG
Lädt die Schriftart aus der Schriftartenzuordnung, die `desc` am ehesten entspricht.

Diese Funktion gibt die neu zugewiesene, geladene Pango-Schriftart zurück oder ein NULL-Objekt, wenn keine Schriftart gefunden wurde. Sie können die Funktion pfont:IsNull() verwenden, um zu prüfen, ob eine Schriftart `NULL` ist.

EINGABEN

`context` den Pango-Kontext, in dem die Schriftart verwendet werden soll

`desc` eine Pango-Schriftartbeschreibung, die die zu ladende Schriftart beschreibt

RÜCKGABEWERTE

`font` die neu zugewiesene Pango-Schriftart

25.10 pfontmap:LoadFontset

BEZEICHNUNG
pfontmap:LoadFontset – Lädt einen Schriftsatz

ÜBERSICHT
```
pfontmap:LoadFontset(context, desc, language)
```

BESCHREIBUNG

Laden Sie eine Reihe von Schriftarten in die Schriftartenzuordnung, die zum Rendern einer Schriftart verwendet werden können, die `desc` entspricht.

Diese Funktion gibt den neu zugewiesenen 'Pango-Schriftsatz' zurück, der geladen wurde, oder Null, wenn keine Schriftart übereinstimmt.

EINGABEN

`context`	den Pango-Kontext, in dem die Schriftart verwendet werden soll
`desc`	eine Pango-Schriftartbeschreibung, die die zu ladende Schriftart beschreibt
`language`	eine Pango-Sprache, für die die Schriftarten verwendet werden sollen

RÜCKGABEWERTE

`x`	die neu zugewiesenen

25.11 pfontmap:Reference

BEZEICHNUNG

pfontmap:Reference – Erhöht die Anzahl der Referenzen der Pango-Schriftartenzuordnungen

ÜBERSICHT

```
pfontmap:Reference()
```

BESCHREIBUNG

Erhöht die Anzahl der Verweise auf die Pango-Schriftartenzuordnung um eins.

EINGABEN

Keine

25.12 pfontmap:SetResolution

BEZEICHNUNG

pfontmap:SetResolution – Stellt die Auflösung ein

ÜBERSICHT

```
pfontmap:SetResolution(dpi)
```

BESCHREIBUNG

Legt die Auflösung für die Schriftartenzuordnung fest.

Dies ist ein Skalierungsfaktor zwischen den in einer Pango-Schriftbeschreibung angegebenen Punkten und den Cairo-Einheiten. Der Standardwert ist 96, was bedeutet, dass eine 10-Punkt-Schrift 13 Einheiten hoch ist (10 * 96. / 72. = 13.3).

EINGABEN

`dpi`	die Auflösung in "Punkte pro Zoll"; es handelt sich nicht um physikalische Zollwerte, sondern um eine konventionelle Terminologie

26 Pango-Schriftarten-Metriken

26.1 pfontmetrics:Free

BEZEICHNUNG
pfontmetrics:Free – Verringert die Anzahl der Schriftart-Metriken

ÜBERSICHT
`pfontmetrics:Free()`

BESCHREIBUNG
Verringert die Anzahl der Verweise auf ein Schriftmetrikobjekt um eins.

Ist das Ergebnis gleich Null, wird das Objekt gelöscht und der zugehörige Speicher freigegeben.

EINGABEN
Keine

26.2 pfontmetrics:GetApproximateCharWidth

BEZEICHNUNG
pfontmetrics:GetApproximateCharWidth – Gibt die ungefähre Zeichenbreite zurück

ÜBERSICHT
`width = pfontmetrics:GetApproximateCharWidth()`

BESCHREIBUNG
Ermittelt die ungefähre Zeichenbreite für eine Schriftmetrikstruktur.

Es handelt sich hierbei lediglich um einen repräsentativen Wert, der z.B. für die Bestimmung der Anfangsgröße eines Fensters nützlich ist. Tatsächliche Zeichen im Text werden breiter und schmaler sein als dieser Wert.

EINGABEN
Keine

RÜCKGABEWERTE

`width` die ungefähre Zeichenbreite in Pango-Einheiten

26.3 pfontmetrics:GetApproximateDigitWidth

BEZEICHNUNG
pfontmetrics:GetApproximateDigitWidth – Gibt die ungefähre Ziffernbreite zurück

ÜBERSICHT
`width = pfontmetrics:GetApproximateDigitWidth()`

BESCHREIBUNG
Ermittelt die ungefähre Ziffernbreite für eine Schriftmetrikstruktur.

Dies ist lediglich ein repräsentativer Wert, der z.B. für die Bestimmung der Anfangsgröße eines Fensters nützlich ist. Tatsächliche Ziffern im Text können breiter oder schmaler sein als dieser Wert, obwohl dieser Wert für Ziffern im Allgemeinen etwas genauer ist als das Ergebnis von pfontmetrics:GetApproximateCharWidth(), wenn es für Ziffern verwendet wird.

EINGABEN

Keine

RÜCKGABEWERTE

`width` die ungefähre Ziffernbreite in Pango-Einheiten

26.4 pfontmetrics:GetAscent

BEZEICHNUNG

pfontmetrics:GetAscent – Gibt die Versalhöhe zurück

ÜBERSICHT

```
ascent = pfontmetrics:GetAscent()
```

BESCHREIBUNG

Gibt die Versalhöhe aus einer Schriftmetrikstruktur zurück.

Die Versalhöhe ist der Abstand von der Grundlinie bis zum logischen oberen Rand einer Textzeile. Der logische obere Rand kann über oder unter der tatsächlich gezeichneten Zeichenfarbe liegen. Es ist notwendig, den Text anzulegen, um herauszufinden, wo sich die Zeichenfarbe befinden wird.

EINGABEN

Keine

RÜCKGABEWERTE

`ascent` die Versalhöhe in Pango-Einheiten

26.5 pfontmetrics:GetDescent

BEZEICHNUNG

pfontmetrics:GetDescent – Gibt die Unterlänge zurück

ÜBERSICHT

```
descent = pfontmetrics:GetDescent()
```

BESCHREIBUNG

Gibt die Unterlänge aus einer Schriftmetrikstruktur zurück.

Die Unterlänge ist der Abstand zwischen der Grundlinie und der p-Linie einer Textzeile. Die p-Linie kann über oder unter der tatsächlich Zeichenfarbe liegen. Um herauszufinden, wo sich die Zeichenfarbe befinden wird, ist es notwendig, den Text anzulegen.

EINGABEN

Keine

RÜCKGABEWERTE

`descent` die Unterlänge in Pango-Einheiten

26.6 pfontmetrics:GetHeight

BEZEICHNUNG

pfontmetrics:GetHeight – Gibt die Zeilenhöhe zurück

ÜBERSICHT

```
height = pfontmetrics:GetHeight()
```

BESCHREIBUNG

Ermittelt die Zeilenhöhe aus einer Schriftmetrikstruktur.

Die Zeilenhöhe ist der empfohlene Abstand zwischen aufeinanderfolgenden Grundlinien in umbrochenem Text mit dieser Schriftart.

Wenn die Zeilenhöhe nicht verfügbar ist, wird 0 zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`height` die Zeilenhöhe in Pango-Einheiten

26.7 pfontmetrics:GetStrikethroughPosition

BEZEICHNUNG

pfontmetrics:GetStrikethroughPosition – Gibt die durchgestrichene Linienposition zurück

ÜBERSICHT

```
pos = pfontmetrics:GetStrikethroughPosition()
```

BESCHREIBUNG

Gibt die vorgeschlagene Position für das Durchstreichen zurück.

Der zurückgegebene Wert ist der Abstand *über* der Grundlinie des oberen Teils des Durchstreichens.

EINGABEN

Keine

RÜCKGABEWERTE

`pos` die vorgeschlagene durchgestrichene Position in Pango-Einheiten

26.8 pfontmetrics:GetStrikethroughThickness

BEZEICHNUNG

pfontmetrics:GetStrikethroughThickness – Gibt die Stärke der durchgestrichenen Linie zurück

ÜBERSICHT

```
thick = pfontmetrics:GetStrikethroughThickness()
```

BESCHREIBUNG

Gibt die vorgeschlagene Stärke für das Durchstreichen zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`thick` die vorgeschlagene Stärke fürs Durchstreichen in Pango-Einheiten

26.9 pfontmetrics:GetUnderlinePosition

BEZEICHNUNG

pfontmetrics:GetUnderlinePosition – Gibt die Position der Unterstreichung zurück

ÜBERSICHT

```
pos = pfontmetrics:GetUnderlinePosition()
```

BESCHREIBUNG

Gibt die vorgeschlagene Position zum Zeichnen der Unterstreichung zurück.

Der zurückgegebene Wert ist der Abstand des oberen Endes der Unterstreichung *über* der Grundlinie. Da bei den meisten Schriftarten die Unterstreichung unterhalb der Grundlinie liegt, ist dieser Wert in der Regel negativ.

EINGABEN

Keine

RÜCKGABEWERTE

`pos` die vorgeschlagene Unterstreichungsposition in Pango-Einheiten

26.10 pfontmetrics:GetUnderlineThickness

BEZEICHNUNG

pfontmetrics:GetUnderlineThickness – Gibt die Stärke der Unterstreichungslinie zurück

ÜBERSICHT

```
thick = pfontmetrics:GetUnderlineThickness()
```

BESCHREIBUNG

Gibt die vorgeschlagene Stärke für die Unterstreichung zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`thick` die vorgeschlagene Unterstreichungsstärke in Pango-Einheiten

26.11 pfontmetrics:IsNull

BEZEICHNUNG

pfontmetrics:IsNull – Prüft, ob das Schriftart-Metrik-Objekt ungültig ist

ÜBERSICHT

```
bool = pfontmetrics:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Objekt `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Objekt `NULL` ist, sonst `False`

26.12 pfontmetrics:Reference

BEZEICHNUNG

pfontmetrics:Reference – Erhöht die Anzahl der Referenzen der Schriftmetrikstrukturen

ÜBERSICHT

```
pfontmetrics:Reference()
```

BESCHREIBUNG

Erhöht die Anzahl der Verweise auf eine Schriftmetrikstruktur um eins.

EINGABEN

Keine

27 Pango-Schriftartensatz

27.1 pfontset:ForEach

BEZEICHNUNG
pfontset:ForEach – Durchläuft alle Schriftarten

ÜBERSICHT
`pfontset:ForEach(func[, userdata])`

BESCHREIBUNG
Durchläuft alle Schriftarten in einem Schriftartensatz und ruft die Callback-Funktion `func` für jede einzelne Schriftart auf. Wenn `func` `True` zurückgibt, wird der Durchlauf beendet.

Die Callback-Funktion erhält im ersten Parameter den Schriftartensatz und im zweiten Parameter die Pango-Schriftart. Wenn Sie den Parameter `userdata` angeben, wird er als dritter Parameter an die Callback-Funktion übergeben.

EINGABEN

`func`	eine Callback-Funktion
`data`	zusätzliche Daten, die an die Callback-Funktion übergeben werden

27.2 pfontset:Free

BEZEICHNUNG
pfontset:Free – Löscht den Schriftartensatz

ÜBERSICHT
`pfontset:Free()`

BESCHREIBUNG
Löscht einen Schriftartensatz.

EINGABEN
Keine

27.3 pfontset:GetFont

BEZEICHNUNG
pfontset:GetFont – Gibt die Schriftart zurück

ÜBERSICHT
`font = pfontset:GetFont(wc)`

BESCHREIBUNG
Gibt die Schriftart im Schriftartensatz zurück, die die beste Glyphe für ein Unicode-Zeichen enthält. Sie müssen die von dieser Funktion zurückgegebene Schriftart mit pfont:Free() löschen.

EINGABEN

`wc` ein Unicode-Zeichen

RÜCKGABEWERTE

`font` ein Pango-Schriftart-Objekt

27.4 pfontset:GetMetrics

BEZEICHNUNG

pfontset:GetMetrics – Gibt die Metriken für alle Schriftarten zurück

ÜBERSICHT

```
metrics = pfontset:GetMetrics()
```

BESCHREIBUNG

Ermittelt allgemeine Metrikinformationen für die Schriftarten im Schriftartensatz. Sie müssen das von dieser Funktion zurückgegebene Schriftmetrikobjekt mit pfontmetrics:Free() löschen.

EINGABEN

Keine

RÜCKGABEWERTE

`metrics` ein Pango-Schriftmetrik-Objekt

27.5 pfontset:IsNull

BEZEICHNUNG

pfontset:IsNull – Prüft, ob die Schriftart ungültig ist

ÜBERSICHT

```
bool = pfontset:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn der Schriftartensatz `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn der Schriftartensatz `NULL` ist, sonst `False`

27.6 pfontset:Reference

BEZEICHNUNG
pfontset:Reference – Erhöht die Anzahl der Referenzen der Pango-Schriftsätze

ÜBERSICHT
`pfontset:Reference()`

BESCHREIBUNG
Erhöht die Anzahl der Verweise auf den Pango-Schriftsatz um eins.

EINGABEN
Keine

28 Pango-Glyphelement

28.1 pglyphitem:Copy

BEZEICHNUNG
pglyphitem:Copy – Kopiert ein Glyphenelement

ÜBERSICHT
```
handle = pglyphitem:Copy()
```

BESCHREIBUNG
Erstellt eine tiefe Kopie einer bestehenden Pango-Glyphenelementstruktur.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` das neu zugewiesene Pango-Glyphenelement

28.2 pglyphitem:Free

BEZEICHNUNG
pglyphitem:Free – Löscht ein Glyphenelement

ÜBERSICHT
```
pglyphitem:Free()
```

BESCHREIBUNG
Löscht ein Pango-Glyphenelement und gibt die Ressourcen frei.

EINGABEN
Keine

28.3 pglyphitem:Get

BEZEICHNUNG
pglyphitem:Get – Gibt die Glyphenelementdaten zurück

ÜBERSICHT
```
t = pglyphitem:Get()
```

BESCHREIBUNG
Gibt die Informationen über das Glyphenelement zurück. Die Daten werden als Tabelle zurückgegeben. Eine Beschreibung aller Felder, die in der Tabelle festgelegt werden, finden Sie unter pglyphitem:Set().

EINGABEN
Keine

RÜCKGABEWERTE
`t` Tabelle mit Informationen über das Glyphenelement (siehe oben)

28.4 pglyphitem:GetItem

BEZEICHNUNG
pglyphitem:GetItem – Gibt das Glyphenelement zurück

ÜBERSICHT
```
item = pglyphitem:GetItem()
```

BESCHREIBUNG
Gibt das Pango-Element zurück, das dem Glyphenelement zugeordnet ist. Sie dürfen dieses Objekt nicht löschen.

EINGABEN
Keine

RÜCKGABEWERTE
`item` ein Pango-Element-Objekt

28.5 pglyphitem:GetGlyphString

BEZEICHNUNG
pglyphitem:GetGlyphString – Gibt die Glyphen-Zeichenkette zurück

ÜBERSICHT
```
glyph_string = pglyphitem:GetGlyphString()
```

BESCHREIBUNG
Gibt die Glyphen-Zeichenkette zurück, die dem Glyphenelement zugeordnet ist. Sie dürfen dieses Objekt nicht löschen.

EINGABEN
Keine

RÜCKGABEWERTE
`glyph_string`
ein Pango-Glyphen-Zeichenkette-Objekt

28.6 pglyphitem:GetLogicalWidths

BEZEICHNUNG
pglyphitem:GetLogicalWidths – Gibt die logische Breite zurück

ÜBERSICHT
```
logical_widths = pglyphitem:GetLogicalWidths(text$)
```

BESCHREIBUNG
Bei einem Pango-Glyphenelement und dem entsprechenden Text wird die Breite jedes Zeichens bestimmt. Die einzelnen Breiten werden in einer Tabelle zurückgegeben. Wenn mehrere Zeichen einen einzigen Cluster bilden, wird die Breite des gesamten Clusters gleichmäßig auf die Zeichen aufgeteilt.

Siehe auch pglyphstring:GetLogicalWidths().

EINGABEN

`text$` Text, dem das Glyphenelement entspricht

RÜCKGABEWERTE

`logical_widths`
Tabelle mit den resultierenden Zeichenbreiten

28.7 pglyphitem:IsNull

BEZEICHNUNG

pglyphitem:IsNull – Prüft, ob das Glyphenelement ungültig ist

ÜBERSICHT

```
bool = pglyphitem:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Glyphenelement `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist. In diesem Fall wird das Glyphenelement auf `NULL` gesetzt.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Glyphenelement `NULL` ist, sonst `False`

28.8 pglyphitem:Set

BEZEICHNUNG

pglyphitem:Set – Setzt die Glyphenelementdaten

ÜBERSICHT

```
pglyphitem:Set(table)
```

BESCHREIBUNG

Setzt die Daten eines oder mehrerer Datenfelder innerhalb des Glyphenelements. Sie müssen dieser Funktion eine Tabelle übergeben, die aus einem oder mehreren Schlüssel-Wert-Paaren besteht, die jeweils ein einzelnes Datenelement festlegen.

Die folgenden Schlüssel werden im Parameter `table` erkannt:

`YOffset` Verschiebt die Grundlinie, relativ zur Grundlinie der enthaltenden Linie. Positive Werte verschieben nach oben.

`StartXOffset`
: Verschiebt horizontal, die vor dem Glyphenelement angewendet wird. Positive Werte verschieben nach rechts.

`EndXOffset`
: Verschiebt horizontal, die nach dem Glyphenelement angewendet wird. Positive Werte verschieben nach rechts.

EINGABEN

`table` Tabelle mit Schlüssel-Wert-Paaren mit den zu setzenden Daten (siehe oben)

28.9 pglyphitem:SetItem

BEZEICHNUNG
pglyphitem:SetItem – Setzt das Glyphenelement

ÜBERSICHT
`pglyphitem:SetItem(item)`

BESCHREIBUNG
Setzt das Pango-Element innerhalb des Glyphenelements auf `item`. Beachten Sie, dass `item` von dieser Funktion nicht referenziert wird, so dass Sie sicherstellen müssen, dass es während des Lebenszyklus des Glyphenelements gültig bleibt.

EINGABEN

`item` ein Pango-Element-Objekt

28.10 pglyphitem:SetGlyphString

BEZEICHNUNG
pglyphitem:SetGlyphString – Setzt die Glyphen-Zeichenkette

ÜBERSICHT
`pglyphitem:SetGlyphString(glyph_string)`

BESCHREIBUNG
Setzt die Glyphen-Zeichenkette innerhalb des Glyphenelements auf `glyph_string`. Beachten Sie, dass `glyph_string` von dieser Funktion nicht referenziert wird, so dass Sie sicherstellen müssen, dass sie während des Lebenszyklus des Glyphenelements gültig bleibt.

EINGABEN

`glyph_string`
: ein Pango-Glyphen-Zeichenkette-Objekt

28.11 pglyphitem:Split

BEZEICHNUNG

pglyphitem:Split – Teilt ein Element auf

ÜBERSICHT

```
handle = pglyphitem:Split(text$, split_index)
```

BESCHREIBUNG

Ändert das Glyphenelement so, dass es nur den Text nach `split_index` abdeckt und gibt ein neues Element zurück, das den Text vor `split_index` abdeckt, der zuvor im Glyphenelement enthalten war.

Sie können sich `split_index` als die Länge des zurückgegebenen Elements vorstellen. `split_index` darf nicht 0 sein, und darf nicht größer oder gleich der Länge des Quellglyphenelements sein (d.h. jedem Element muss mindestens ein Byte zugewiesen sein, Sie können kein Element mit einer Länge von Null erstellen).

Diese Funktion gibt das neu zugewiesene Element zurück, das den Text vor `split_index` repräsentiert und das mit pglyphitem:Free() gelöscht werden sollte.

EINGABEN

`text$` Text, für den die Positionen im Glyphenelement gelten

`split_index`
: Byte-Index der Position, an der das Element geteilt werden soll, bezogen auf den Anfang des Elements

RÜCKGABEWERTE

`handle` das neu zugewiesene Glyphenelement

29 Pango-Glyphen-Zeichenkette

29.1 pglyphstring:Copy

BEZEICHNUNG

pglyphstring:Copy – Kopiert eine Glyphen-Zeichenkette

ÜBERSICHT

```
gstr = pglyphstring:Copy()
```

BESCHREIBUNG

Kopiert eine Glyphen-Zeichenkette und den zugehörigen Speicher.

EINGABEN

Keine

RÜCKGABEWERTE

`gstr` die neu zugewiesene Pango-Glyphen-Zeichenkette

29.2 pglyphstring:Extents

BEZEICHNUNG

pglyphstring:Extents – Berechnet die Ausmaße von Glyphen-Zeichenketten

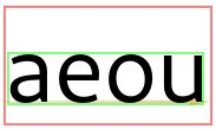
ÜBERSICHT

```
ink_rect, logical_rect = pglyphstring:Extents(font)
```

BESCHREIBUNG

Berechnet die logischen und farblichen Ausmaße einer Glyphen-Zeichenkette. Dies gibt zwei Tabellen zurück, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind. Siehe die Dokumentation zu pfont:GetGlyphExtents() für Details zur Interpretation der Rechtecke.

Beispiele für logische (rot) und farbige (grün) Felder:



EINGABEN

`font` eine Pango-Schriftart

RÜCKGABEWERTE

`ink_rect` Tabelle mit den Ausmaßen der gezeichneten Glyphen-Zeichenkette

`logical_rect`
: Tabelle mit den logischen Ausmaßen der Glyphen-Zeichenkette

29.3 pglyphstring:ExtentsRange

BEZEICHNUNG

pglyphstring:ExtentsRange – Berechnet die Ausmaße

ÜBERSICHT

```
ink_rect, logical_rect = pglyphstring:ExtentsRange(start, end, font)
```

BESCHREIBUNG

Berechnet die Ausmaße eines Teilbereichs einer Glyphen-Zeichenkette. Es werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Die Ausmaße sind relativ zum Beginn des Bereichs der Glyphen-Zeichenkette (der Ursprung ihres Koordinatensystems liegt am Beginn des Bereichs, nicht am Beginn der gesamten Glyphen-Zeichenkette).

EINGABEN

`start` Startindex

`end` Endindex (der Bereich ist die Menge der Bytes mit Indizes, die so beschaffen sind, dass start <= Index < end)

`font` eine Pango-Schriftart

RÜCKGABEWERTE

`ink_rect` Tabelle mit den Ausmaßen der gezeichneten Glyphen-Zeichenkette

`logical_rect`
: Tabelle mit den logischen Ausmaßen der Glyphen-Zeichenkette

29.4 pglyphstring:Free

BEZEICHNUNG

pglyphstring:Free – Löscht die Glyphen-Zeichenkette

ÜBERSICHT

```
pglyphstring:Free()
```

BESCHREIBUNG

Löscht eine Glyphen-Zeichenkette und gibt den zugehörigen Speicher frei.

EINGABEN

Keine

29.5 pglyphstring:Get

BEZEICHNUNG

pglyphstring:Get – Gibt die Glypheninfo zurück

ÜBERSICHT

```
table = pglyphstring:Get(index)
```

BESCHREIBUNG

Gibt die Positionsinformationen und die visuellen Attribute für die Glyphe unter `index` ab. Die Informationen werden in einer Tabelle zurückgegeben. Siehe pglyphstring:Set() für eine Beschreibung aller Tabellenfelder, die von dieser Funktion gesetzt werden.

EINGABEN

`index` die zu verwendende Glyphe

RÜCKGABEWERTE

`table` eine Tabelle mit den Glypheninformationen

29.6 pglyphstring:GetLogicalWidths

BEZEICHNUNG

pglyphstring:GetLogicalWidths – Gibt die logischen Breiten zurück

ÜBERSICHT

```
t = pglyphstring:GetLogicalWidths(text$, embedding_level)
```

BESCHREIBUNG

Gibt bei einer Pango-Glyphen-Zeichenkette und dem entsprechenden Text die Breite jedes Zeichens. Wenn mehrere Zeichen einen einzigen Cluster bilden, wird die Breite des gesamten Clusters gleichmäßig auf die Zeichen aufgeteilt. Die einzelnen Breiten werden in einer Tabelle zurückgegeben.

Siehe auch pglyphitem:GetLogicalWidths().

EINGABEN

`text$` der den Glyphen entsprechende Text

`embedding_level`
die Einbettungsebene der Zeichenkette

RÜCKGABEWERTE

`logical_widths`
Tabelle mit den resultierenden Zeichenbreiten

29.7 pglyphstring:GetWidth

BEZEICHNUNG

pglyphstring:GetWidth – Gibt die Breite zurück

ÜBERSICHT

```
width = pglyphstring:GetWidth()
```

BESCHREIBUNG

Berechnet die logische Breite der Zeichenkette.

Dies kann auch mit pglyphstring:Extents() berechnet werden. Da diese Funktion jedoch nur die Breite berechnet, ist sie viel schneller. Dies ist eigentlich nur ein Komfortbefehl, der die Summe von `geometry.width` für jede Glyphe in der Glyphen-Zeichenkette berechnet.

EINGABEN

Keine

RÜCKGABEWERTE

`width` die logische Breite der Glyphen-Zeichenkette

29.8 pglyphstring:IndexToX

BEZEICHNUNG

pglyphstring:IndexToX – Konvertiert von der Zeichen- zur x-Position

ÜBERSICHT

```
x_pos = pglyphstring:IndexToX(text$, analysis, index, trailing)
```

BESCHREIBUNG

Konvertiert von der Zeichenposition zur x-Position.

Die X-Position wird vom linken Rand des Laufs aus gemessen. Die Zeichenpositionen werden, sofern verfügbar, mithilfe von Schriftmetriken für Ligaturen ermittelt und andernfalls durch Aufteilung jedes Clusters in gleiche Teile berechnet.

EINGABEN

`text$` den Text für den Lauf

`analysis` die von der Funktion pcontext:Itemize() zurückgegebenen Analyseinformationen; dies muss ein Pango-Analyseobjekt sein

`index` den Byte-Index innerhalb von `text$`

`trailing` ob das Ergebnis für den Anfang (`False`) oder das Ende (`True`) des Zeichens berechnet werden soll

RÜCKGABEWERTE

`x_pos` Speicherort für das Ergebnis

29.9 pglyphstring:IsNull

BEZEICHNUNG

pglyphstring:IsNull – Prüft, ob die Glyphen-Zeichenkette ungültig ist

ÜBERSICHT

```
bool = pglyphstring:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Zeichenkette `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn die Glyphen-Zeichenkette `NULL` ist, sonst `False`

29.10 pglyphstring:Set

BEZEICHNUNG

pglyphstring:Set – Stellt die Glypheninfo ein

ÜBERSICHT

```
pglyphstring:Set(index, table)
```

BESCHREIBUNG

Setzt Positionierungsinformationen und visuelle Attribute für die Glyphe bei `index`. Die zu setzenden Daten müssen als ein oder mehrere Schlüssel-Wert-Paare im Argument `table` übergeben werden. Die folgenden Schlüssel werden derzeit unterstützt:

`Glyph` Die Glyphe selbst.

`Width` Die logische Breite, die für das Zeichen verwendet werden soll.

`XOffset` Horizontaler Versatz von der nominalen Zeichenposition.

`YOffset` Vertikaler Versatz von der nominalen Zeichenposition.

`IsClusterStart`
: Wird für die erste logische Glyphe in jedem Cluster festgelegt.

`LogCluster`
: Logische Cluster-Info, indiziert durch den Byte-Index innerhalb des Textes, der der Glyphen-Zeichenkette entspricht.

EINGABEN

`index` die zu verwendende Glyphe

`table` eine Tabelle, die aus Schlüssel-Wert-Paaren der zu setzenden Daten besteht

29.11 pglyphstring:SetSize

BEZEICHNUNG
pglyphstring:SetSize – Ändert die Größe der Glyphen-Zeichenkette

ÜBERSICHT
```
pglyphstring:SetSize(new_len)
```

BESCHREIBUNG
Ändert die Größe einer Zeichenkette auf die angegebene Länge.

EINGABEN

`new_len` die neue Länge der Zeichenkette

29.12 pglyphstring:XToIndex

BEZEICHNUNG
pglyphstring:XToIndex – Konvertiert von x-Offset in Zeichenposition

ÜBERSICHT
```
index, trailing = pglyphstring:XToIndex(text$, analysis, x_pos)
```

BESCHREIBUNG
Konvertierung von x-Offset in Zeichenposition.

Die Zeichenpositionen werden durch Aufteilung jedes Clusters in gleiche Teile berechnet. In Skripten, in denen die Positionierung innerhalb eines Clusters nicht erlaubt ist (z.B. Thai), ist der zurückgegebene Wert möglicherweise keine gültige Cursorposition. Der Aufrufer muss das Ergebnis mit den logischen Attributen für den Text kombinieren, um die gültige Cursorposition zu berechnen.

EINGABEN

`text$` den Text für den Lauf

`analysis` die von der Funktion pcontext:Itemize() zurückgegebenen Analyseinformationen; dies muss ein Pango-Analyseobjekt sein

`x_pos` der x-Versatz (in Pango-Einheiten)

RÜCKGABEWERTE

`index` berechneter Byte-Index innerhalb von `text$`

`trailing` Boolescher Wert, der angibt, ob der Benutzer auf den vorderen oder den hinteren Rand des Zeichens geklickt hat

30 Pango-Element

30.1 pitem:Copy

BEZEICHNUNG
pitem:Copy – Kopiert ein Pango-Element

ÜBERSICHT
`item = pitem:Copy()`

BESCHREIBUNG
Kopiert eine vorhandene Pango-Elementstruktur.

EINGABEN
Keine

RÜCKGABEWERTE
`item` das neu zugewiesenen Pango-Element

30.2 pitem:Free

BEZEICHNUNG
pitem:Free – Löscht ein Element

ÜBERSICHT
`pitem:Free()`

BESCHREIBUNG
Löscht ein Pango-Element und gibt den gesamten zugehörigen Speicher frei.

EINGABEN
Keine

30.3 pitem:Get

BEZEICHNUNG
pitem:Get – Gibt die Elementdaten zurück

ÜBERSICHT
`table = pitem:Get()`

BESCHREIBUNG
Gibt die Daten für das Element in Form einer Tabelle zurück. Siehe pitem:Set() für Informationen darüber, welche Tabellenfelder gesetzt werden.

EINGABEN
Keine

RÜCKGABEWERTE
`table` eine Tabelle mit den Elementdaten

30.4 pitem:IsNull

BEZEICHNUNG

pitem:IsNull – Prüft, ob das Element ungültig ist

ÜBERSICHT

```
bool = pitem:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Element `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob ein Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Element `NULL` ist, sonst `False`

30.5 pitem:Set

BEZEICHNUNG

pitem:Set – Setzt die Elementdaten

ÜBERSICHT

```
pitem:Set(table)
```

BESCHREIBUNG

Setzt Daten für das Element. Die zu setzenden Daten müssen als ein oder mehrere Schlüssel-Wert-Paare im Argument `table` übergeben werden. Die folgenden Schlüssel werden derzeit unterstützt:

`Offset` Länge dieses Elements in Bytes.

`NumChars` Anzahl der Unicode-Zeichen im Element.

`Analysis` Analyseergebnisse für das Element. Dies ist ein Pango-Analyseobjekt.

EINGABEN

`table` eine Tabelle, die aus Schlüssel-Wert-Paaren der zu setzenden Daten besteht

30.6 pitem:Split

BEZEICHNUNG

pitem:Split – Teilt ein Element auf

ÜBERSICHT

```
item = pitem:Split(split_index, split_offset)
```

BESCHREIBUNG

Ändert das Elemnt so, dass es nur den Text nach `split_index` enthält, und gibt ein neues Element zurück, das den Text vor `split_index` enthält, das im ursprünglichen Element enthalten war.

Sie können sich `split_index` als die Länge des zurückgegebenen Elements vorstellen. `split_index` darf nicht 0 sein, und er darf nicht größer oder gleich der Länge des Elements sein (d.h. jedes Element muss mindestens ein Byte zugeordnet sein, Sie können kein Element mit einer Länge von Null erstellen). `split_offset` ist die Länge des ersten Elements in Zeichen und muss angegeben werden, da der Text, der zur Erzeugung des Elements verwendet wird, nicht zur Verfügung steht, so dass diese Funktion die Zeichenlänge des geteilten Elements selbst nicht zählen kann.

Diese Funktion gibt ein neues Element zurück, das den Text vor `split_index` repräsentiert und das mit pitem:Free() gelöscht werden sollte.

EINGABEN

`split_index`
: Byte-Index der Position, an der das Element geteilt werden soll, bezogen auf den Anfang des Elements

`split_offset`
: Anzahl der Zeichen zwischen dem Beginn des Eintrags und `split_index`

RÜCKGABEWERTE

`item` new item representing text before `split_index`

31 Pango-Sprache

31.1 planguage:GetSampleString

BEZEICHNUNG

planguage:GetSampleString – Gibt einen Beispieltext zurück

ÜBERSICHT

```
s$ = planguage:GetSampleString()
```

BESCHREIBUNG

Gibt eine Zeichenkette zurück, die die für die Darstellung einer bestimmten Sprache erforderlichen Zeichen repräsentiert.

Der Beispieltext kann ein Pangramm sein, muss es aber nicht. Er ist so gewählt, dass er einen normalen Text in der Sprache darstellt und die besonderen Anforderungen der Sprache an die Schriftart aufzeigt. Er eignet sich zur Verwendung als Beispieltext in einem Schriftauswahldialog.

Wenn Pango keine Beispielzeichenkette für die Sprache hat, wird das klassische "The quick brown fox..." zurückgegeben. Dies lässt sich feststellen, indem der zurückgegebene Handle-Wert mit dem für den (nicht vorhandenen) Sprachcode "xx" verglichen wird.

EINGABEN

Keine

RÜCKGABEWERTE

`s$` Beispieltext für die Sprache

31.2 planguage:GetScripts

BEZEICHNUNG

planguage:GetScripts – Gibt die Sprache der verwendeten Skripte zurück

ÜBERSICHT

```
t = planguage:GetScripts()
```

BESCHREIBUNG

Ermittelt die zum Schreiben der Sprache verwendeten Skripte. Die einzelnen Skripte werden in einer Tabelle zurückgegeben. Die Liste der zurückgegebenen Skripte beginnt mit dem Skript, das die Sprache am häufigsten verwendet und geht weiter bis zu dem, das sie am wenigsten verwendet.

Die meisten Sprachen verwenden nur eine Schrift, einige aber auch zwei (z.B. Latein und Kyrillisch) und einige wenige drei (z.B. Japanisch).

Die folgenden Skripte sind derzeit definiert:

`#PANGO_SCRIPT_INVALID_CODE`
: Signalisiert ein ungültiges Skript.

`#PANGO_SCRIPT_COMMON`
: Ein Zeichen, das von mehreren verschiedenen Skripten verwendet wird.

`#PANGO_SCRIPT_INHERITED`
: Eine Zeichenglyphe, die ihre Schrift von der Basisglyphe übernimmt, an die sie angehängt ist.

`#PANGO_SCRIPT_ARABIC`
: Arabisch.

`#PANGO_SCRIPT_ARMENIAN`
: Armenisch.

`#PANGO_SCRIPT_BENGALI`
: Bengalisch.

`#PANGO_SCRIPT_BOPOMOFO`
: Bopomofo.

`#PANGO_SCRIPT_CHEROKEE`
: Cherokee.

`#PANGO_SCRIPT_COPTIC`
: Koptisch.

`#PANGO_SCRIPT_CYRILLIC`
: Kyrillisch.

`#PANGO_SCRIPT_DESERET`
: Deseret.

`#PANGO_SCRIPT_DEVANAGARI`
: Devanagari.

`#PANGO_SCRIPT_ETHIOPIC`
: Äthiopisch.

`#PANGO_SCRIPT_GEORGIAN`
: Georgisch.

`#PANGO_SCRIPT_GOTHIC`
: Gotisch.

`#PANGO_SCRIPT_GREEK`
: Griechisch.

`#PANGO_SCRIPT_GUJARATI`
: Gujarati.

`#PANGO_SCRIPT_GURMUKHI`
: Gurmukhi.

`#PANGO_SCRIPT_HAN`
: Han.

`#PANGO_SCRIPT_HANGUL`
: Hangul.

`#PANGO_SCRIPT_HEBREW`
: Hebräisch.

`#PANGO_SCRIPT_HIRAGANA`
: Hiragana.

`#PANGO_SCRIPT_KANNADA`
: Kannada.

`#PANGO_SCRIPT_KATAKANA`
: Katakana.

`#PANGO_SCRIPT_KHMER`
: Khmer.

`#PANGO_SCRIPT_LAO`
: Lao.

`#PANGO_SCRIPT_LATIN`
: Lateinisch.

`#PANGO_SCRIPT_MALAYALAM`
: Malayalam.

`#PANGO_SCRIPT_MONGOLIAN`
: Mongolisch.

`#PANGO_SCRIPT_MYANMAR`
: Myanmar.

`#PANGO_SCRIPT_OGHAM`
: Ogham.

`#PANGO_SCRIPT_OLD_ITALIC`
: Alt-Kursiv.

`#PANGO_SCRIPT_ORIYA`
: Oriya.

`#PANGO_SCRIPT_RUNIC`
: Runen.

`#PANGO_SCRIPT_SINHALA`
: Singhalesisch.

`#PANGO_SCRIPT_SYRIAC`
: Syrisch.

`#PANGO_SCRIPT_TAMIL`
: Tamilisch.

`#PANGO_SCRIPT_TELUGU`
: Telugu.

`#PANGO_SCRIPT_THAANA`
: Thaana.

`#PANGO_SCRIPT_THAI`
: Thailändisch.

`#PANGO_SCRIPT_TIBETAN`
: Tibetisch.

`#PANGO_SCRIPT_CANADIAN_ABORIGINAL`
: Kanadische Ureinwohner.

`#PANGO_SCRIPT_YI`
: Yi.

`#PANGO_SCRIPT_TAGALOG`
: Tagalog.

`#PANGO_SCRIPT_HANUNOO`
: Hanunoo.

`#PANGO_SCRIPT_BUHID`
: Buhid.

`#PANGO_SCRIPT_TAGBANWA`
: Tagbanwa.

`#PANGO_SCRIPT_BRAILLE`
: Blindenschrift.

`#PANGO_SCRIPT_CYPRIOT`
: Zypriotisch.

`#PANGO_SCRIPT_LIMBU`
: Limbu.

`#PANGO_SCRIPT_OSMANYA`
: Osmanisch.

`#PANGO_SCRIPT_SHAVIAN`
: Shavian.

`#PANGO_SCRIPT_LINEAR_B`
: Linear B.

`#PANGO_SCRIPT_TAI_LE`
: Tai Le.

`#PANGO_SCRIPT_UGARITIC`
: Ugaritisch.

`#PANGO_SCRIPT_NEW_TAI_LUE`
: Neu-Tai Lue.

`#PANGO_SCRIPT_BUGINESE`
: Buginesisch.

`#PANGO_SCRIPT_GLAGOLITIC`
: Glagolitisch.

`#PANGO_SCRIPT_TIFINAGH`
: Tifinagh.

`#PANGO_SCRIPT_SYLOTI_NAGRI`
: Syloti Nagri.

`#PANGO_SCRIPT_OLD_PERSIAN`
: Altpersisch.

`#PANGO_SCRIPT_KHAROSHTHI`
: Kharoshthi.

`#PANGO_SCRIPT_UNKNOWN`
: Ein nicht zugewiesener Codepunkt.

`#PANGO_SCRIPT_BALINESE`
: Balinesisch.

`#PANGO_SCRIPT_CUNEIFORM`
: Keilschrift.

`#PANGO_SCRIPT_PHOENICIAN`
: Phönizisch.

`#PANGO_SCRIPT_PHAGS_PA`
: Phags-pa.

`#PANGO_SCRIPT_NKO`
: NKo.

`#PANGO_SCRIPT_KAYAH_LI`
: Kayah Li.

`#PANGO_SCRIPT_LEPCHA`
: Lepcha.

`#PANGO_SCRIPT_REJANG`
: Rejang.

`#PANGO_SCRIPT_SUNDANESE`
: Sundanisch.

`#PANGO_SCRIPT_SAURASHTRA`
: Saurashtra.

`#PANGO_SCRIPT_CHAM`
: Cham.

`#PANGO_SCRIPT_OL_CHIKI`
: Ol Chiki.

`#PANGO_SCRIPT_VAI`
: Vai.

`#PANGO_SCRIPT_CARIAN`
: Karian.

`#PANGO_SCRIPT_LYCIAN`
: Lykisch.

`#PANGO_SCRIPT_LYDIAN`
: Lydisch.

`#PANGO_SCRIPT_BATAK`
: Batak.

`#PANGO_SCRIPT_BRAHMI`
: Brahmi.

`#PANGO_SCRIPT_MANDAIC`
: Mandäisch.

`#PANGO_SCRIPT_CHAKMA`
: Chakma.

`#PANGO_SCRIPT_MEROITIC_CURSIVE`
: Meroitische Kursivschrift.

`#PANGO_SCRIPT_MEROITIC_HIEROGLYPHS`
: Meroitische Hieroglyphen.

`#PANGO_SCRIPT_MIAO`
: Miao.

`#PANGO_SCRIPT_SHARADA`
: Sharada.

`#PANGO_SCRIPT_SORA_SOMPENG`
: Sora Sompeng.

`#PANGO_SCRIPT_TAKRI`
: Takri.

`#PANGO_SCRIPT_BASSA_VAH`
: Bassa.

`#PANGO_SCRIPT_CAUCASIAN_ALBANIAN`
: Kaukasischer Albaner.

`#PANGO_SCRIPT_DUPLOYAN`
: Duployan.

`#PANGO_SCRIPT_ELBASAN`
: Elbasan.

`#PANGO_SCRIPT_GRANTHA`
: Grantha.

`#PANGO_SCRIPT_KHOJKI`
: Kjohki.

`#PANGO_SCRIPT_KHUDAWADI`
: Khudawadi, Sindhi.

`#PANGO_SCRIPT_LINEAR_A`
: Linear A.

`#PANGO_SCRIPT_MAHAJANI`
: Mahajani.

`#PANGO_SCRIPT_MANICHAEAN`
: Manichäisch.

`#PANGO_SCRIPT_MENDE_KIKAKUI`
: Mende Kikakui.

`#PANGO_SCRIPT_MODI`
: Modi.

`#PANGO_SCRIPT_MRO`
: Mro.

`#PANGO_SCRIPT_NABATAEAN`
: Nabatäische.

`#PANGO_SCRIPT_OLD_NORTH_ARABIAN`
: Alt-Nordarabisch.

`#PANGO_SCRIPT_OLD_PERMIC`
: Alt-Permisch.

`#PANGO_SCRIPT_PAHAWH_HMONG`
: Pahawh Hmong.

`#PANGO_SCRIPT_PALMYRENE`
: Palmyrenäisch.

`#PANGO_SCRIPT_PAU_CIN_HAU`
: Pau Cin Hau.

`#PANGO_SCRIPT_PSALTER_PAHLAVI`
: Psalter Pahlavi.

`#PANGO_SCRIPT_SIDDHAM`
: Siddham.

`#PANGO_SCRIPT_TIRHUTA`
: Tirhuta.

`#PANGO_SCRIPT_WARANG_CITI`
: Warang Citi.

`#PANGO_SCRIPT_AHOM`
: Ahom.

`#PANGO_SCRIPT_ANATOLIAN_HIEROGLYPHS`
: Anatolische Hieroglyphen.

`#PANGO_SCRIPT_HATRAN`
: Hatran.

`#PANGO_SCRIPT_MULTANI`
: Multani.

`#PANGO_SCRIPT_OLD_HUNGARIAN`
: Alt-Ungarisch.

`PANGO_SCRIPT_SIGNWRITING`
: Gebärdenschrift.

EINGABEN

Keine

RÜCKGABEWERTE

`t` Tabelle mit den einzelnen Skripten (siehe oben)

31.3 planguage:IncludesScript

BEZEICHNUNG

planguage:IncludesScript – Prüft, ob die Sprache ein Skript enthält

ÜBERSICHT

```
ok = planguage:IncludesScript(script)
```

BESCHREIBUNG

Überprüft, ob `script` eines der Skripte ist, die zum Schreiben der Sprache verwendet werden.

Der zurückgegebene Wert ist konservativ. Wenn nichts über das Sprach-Tag der Sprache bekannt ist, wird `True` zurückgegeben, da, soweit Pango weiß, `script` zum Schreiben der Sprache verwendet werden könnte.

Diese Funktion wird im Pango-Zuordnungsprozess verwendet, um festzustellen, ob ein geliefertes Sprach-Tag für einen bestimmten Textabschnitt relevant ist. In den meisten Fällen ist sie für Anwendungen wahrscheinlich nicht nützlich.

Eine Liste der unterstützten Skripte finden Sie unter planguage:GetScripts().

EINGABEN

`script` eine Skript-Kennung

RÜCKGABEWERTE

`ok` `True`, wenn `script` eines der Skripte ist, die zum Schreiben der Sprache verwendet werden, ansonsten `False`

31.4 planguage:IsNull

BEZEICHNUNG

planguage:IsNull – Prüft, ob die Sprache ungültig ist

ÜBERSICHT

```
bool = planguage:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Sprache `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen

das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu überprüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob ein Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn die Sprache `NULL` ist, sonst `False`

31.5 planguage:Matches

BEZEICHNUNG

planguage:Matches – Prüft, ob die Sprachkennzeichnung mit den Sprachbereichen übereinstimmt

ÜBERSICHT

```
ok = planguage:Matches(range_list$)
```

BESCHREIBUNG

Prüft, ob ein Sprachkennzeichnung mit einem der Elemente in einer Liste von Sprachbereichen übereinstimmt. Die Liste der Sprachbereiche muss als Zeichenkette übergeben werden, getrennt durch `";"`, `":"`, `","` oder Leerzeichen. Jedes Element muss entweder `"*"` oder ein RFC 3066-Sprachbereich sein, der wie bei pango.Language() kanonisiert ist.

Eine Sprachkennzeichnung wird als mit einem Bereich in der Liste übereinstimmend betrachtet, wenn der Bereich `"*"` ist, der Bereich genau der Kennzeichnung entspricht oder der Bereich ein Präfix der Kennzeichnung und das Zeichen danach in der Kennzeichnung `"-"` ist.

Diese Funktion gibt `True` zurück, wenn eine Übereinstimmung gefunden wurde.

EINGABEN

`range_list$`
: eine Liste von Sprachbereichen (Format siehe oben)

RÜCKGABEWERTE

`ok` `True`, wenn eine Übereinstimmung gefunden wurde, sonst `False`

31.6 planguage:ToString

BEZEICHNUNG

planguage:ToString – Wandelt die Sprachkennzeichnung in RFC-3066 um

ÜBERSICHT

```
f$ = planguage:ToString()
```

BESCHREIBUNG

Ruft die Zeichenkette im RFC-3066-Format ab, die die angegebene Sprachkennzeichnung darstellt.

EINGABEN

Keine

RÜCKGABEWERTE

`f$` eine Zeichenkette, die das Sprachkennzeichen darstellt

32 Pango-Layout

32.1 playout:ContextChanged

BEZEICHNUNG
playout:ContextChanged – Erzwingt eine Neuberechnung des Layouts

ÜBERSICHT
`playout:ContextChanged()`

BESCHREIBUNG
Erzwingt die Neuberechnung aller Zustände im Pango-Layout, die möglicherweise vom Kontext des Layouts abhängen.

Diese Funktion sollte aufgerufen werden, wenn Sie nach der Erstellung des Layouts Änderungen am Kontext vornehmen.

EINGABEN
Keine

32.2 playout:Copy

BEZEICHNUNG
playout:Copy – Kopiert das Layout

ÜBERSICHT
`handle = playout:Copy()`

BESCHREIBUNG
Erzeugt eine tiefe Kopie des Layouts nach Werten.

Die Attributliste, das Registerfeld und der Text aus dem Originallayout werden alle wertmäßig kopiert.

EINGABEN
Keine

RÜCKGABEWERTE
`handle` das neu zugewiesene Pango-Layout

32.3 playout:Free

BEZEICHNUNG
playout:Free – Löscht ein Layout

ÜBERSICHT
`playout:Free()`

BESCHREIBUNG
Verringert die Referenzanzahl eines Pango-Layouts um eins.

Ist das Ergebnis gleich Null, werden das Layout und der gesamte zugehörige Speicher gelöscht.

EINGABEN
Keine

32.4 playout:GetAlignment

BEZEICHNUNG
playout:GetAlignment – Gibt die Layout-Ausrichtung zurück

ÜBERSICHT
```
align = playout:GetAlignment()
```

BESCHREIBUNG
Ruft die Ausrichtung für das Layout ab, wie Teillinien innerhalb des verfügbaren horizontalen Bereichs positioniert werden. Eine Liste der Ausrichtungen finden Sie unter playout:SetAlignment().

EINGABEN
Keine

RÜCKGABEWERTE
`align` die Ausrichtung

32.5 playout:GetAttributes

BEZEICHNUNG
playout:GetAttributes – Gibt die Attributliste zurück

ÜBERSICHT
```
attrlist = playout:GetAttributes()
```

BESCHREIBUNG
Ruft die Attributliste für das Layout ab, falls vorhanden. Wenn im Layout keine Attributliste vorhanden ist, wird ein NULL-Objekt zurückgegeben. Sie können pattrlist:IsNull() verwenden, um zu überprüfen, ob eine Attributliste `NULL` ist.

EINGABEN
Keine

RÜCKGABEWERTE
`attrlist` eine Pango Attributliste

32.6 playout:GetAutoDir

BEZEICHNUNG
playout:GetAutoDir – Gibt die automatische Verzeichniseinstellung des Layouts zurück

ÜBERSICHT
```
autodir = playout:GetAutoDir()
```

BESCHREIBUNG

Ermittelt, ob die Basisrichtung für das Layout entsprechend seinem Inhalt berechnet werden soll.

Siehe playout:SetAutoDir() für mehr Informationen.

Diese Funktion gibt `True` zurück, wenn die bidirektionale Basisrichtung aus dem Inhalt des Layouts berechnet wird, andernfalls `False`.

EINGABEN

Keine

RÜCKGABEWERTE

`autodir` `True`, wenn AutoDir eingeschaltet ist, sonst `False`

32.7 playout:GetBaseline

BEZEICHNUNG

playout:GetBaseline – Ermittelt die Grundlinie

ÜBERSICHT

```
baseline = playout:GetBaseline()
```

BESCHREIBUNG

Ruft die Y-Position der Grundlinie der ersten Zeile im Layout ab.

EINGABEN

Keine

RÜCKGABEWERTE

`baseline` Grundlinie der ersten Zeile, vom oberen Rand des Layouts

32.8 playout:GetCharacterCount

BEZEICHNUNG

playout:GetCharacterCount – Ermittelt die Anzahl der Zeichen

ÜBERSICHT

```
n = playout:GetCharacterCount()
```

BESCHREIBUNG

Gibt die Anzahl der Unicode-Zeichen im Text des Layouts zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`n` die Anzahl der Unicode-Zeichen

32.9 playout:GetContext

BEZEICHNUNG
playout:GetContext – Gibt den Pango-Kontext zurück

ÜBERSICHT
```
ctx = playout:GetContext()
```

BESCHREIBUNG
Ruft den für dieses Layout verwendeten Pango-Kontext ab. Der Kontext gehört dem Layout und darf nicht gelöscht werden.

EINGABEN
Keine

RÜCKGABEWERTE
`ctx` den Pango-Kontext für das Layout

32.10 playout:GetCursorPos

BEZEICHNUNG
playout:GetCursorPos – Ermittelt die Cursor-Position

ÜBERSICHT
```
strong_pos, weak_pos = playout:GetCursorPos(index)
```

BESCHREIBUNG
Bestimmt bei einem gegebenen `index` innerhalb eines Layouts die Positionen des starken und schwachen Cursors, wenn sich die Einfügemarke an diesem Index befindet. Sowohl `strongpos` als auch `weakpos` sind Tabellen, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind. Die Position jedes Cursors wird als Rechteck mit der Breite Null und der Höhe der Laufgrenzen gespeichert.

Die starke Cursor-Position ist der Ort, an dem Zeichen mit der gleichen Richtungsangabe wie die Basisrichtung des Layouts eingefügt werden. Die schwache Cursor-Position ist der Ort, an dem Zeichen mit einer der Basisrichtung des Layouts entgegengesetzten Richtungsangabe eingefügt werden.

Das folgende Beispiel zeigt Text mit einem starken und einem schwachen Cursor.

Der starke Cursor hat einen kleinen Pfeil, der nach rechts zeigt, der schwache Cursor zeigt nach links. Wenn Sie in dieser Situation ein 'c' eingeben, wird das Zeichen nach dem 'b' eingefügt, und wenn Sie ein anderes hebräisches Zeichen wie "×" eingeben, wird es am Ende eingefügt.

EINGABEN

`index` der Byte-Index des Cursors

RÜCKGABEWERTE

`strong_pos`
starke Cursor-Position

`weak_pos` schwache Cursor-Position

32.11 playout:GetEllipsize

BEZEICHNUNG

playout:GetEllipsize – Gibt den Ellipsisierungstyp zurück

ÜBERSICHT

```
mode = playout:GetEllipsize()
```

BESCHREIBUNG

Gibt die Art der Ellipsenbildung zurück, die für das Layout durchgeführt wird. Siehe playout:SetEllipsize() für weitere Informationen.

Verwenden Sie playout:IsEllipsized(), um abzufragen, ob Absätze tatsächlich mit Ellipsen versehen wurden.

EINGABEN

keine

RÜCKGABEWERTE

`mode` der aktuelle Ellipsisierungsmodus für das Layout

32.12 playout:GetExtents

BEZEICHNUNG

playout:GetExtents – Gibt die Layout-Ausmaße zurück

ÜBERSICHT

```
ink_rect, logical_rect = playout:GetExtents()
```

BESCHREIBUNG

Berechnet die logischen und farblichen Ausdehnungen des Layouts.

Logische Ausdehnungen sind normalerweise das, was Sie für die Positionierung von Dingen benötigen. Beachten Sie, dass beide Ausdehnungen x und y ungleich Null sein können. Sie können diese verwenden, um die Stelle zu verschieben, an der Sie das Layout rendern. Dies nicht zu tun, ist ein sehr typischer Fehler, der sich darin zeigt, dass Layouts von rechts nach links in einem Layout mit einer festgelegten Breite nicht korrekt positioniert werden.

Die Ausdehnungen werden in Layoutkoordinaten und in Pango-Einheiten angegeben; die Layoutkoordinaten beginnen in der linken oberen Ecke des Layouts.

Diese Funktion gibt Tabellen zurück, bei denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

keine

RÜCKGABEWERTE

`ink_rect` Ausmaße des gezeichneten Layouts

`logical_rect`
logische Ausdehnung des Layouts

32.13 playout:GetFontDescription

BEZEICHNUNG

playout:GetFontDescription – Gibt die Beschreibung der Schriftart zurück

ÜBERSICHT

```
fontdesc = playout:GetFontDescription()
```

BESCHREIBUNG

Ruft die Schriftartbeschreibung für das Layout ab, falls vorhanden. Diese Funktion gibt ein Handle auf die Schriftartbeschreibung des Layouts oder ein NULL-Objekt zurück, wenn die Schriftartbeschreibung aus dem Kontext des Layouts geerbt wird. Sie können pfontdesc:IsNull() verwenden, um zu überprüfen, ob ein Objekt `NULL` ist.

EINGABEN

keine

RÜCKGABEWERTE

`fontdesc` ein Handle zu einer Schriftartbeschreibung

32.14 playout:GetHeight

BEZEICHNUNG

playout:GetHeight – Gibt die Höhe des Layouts zurück

ÜBERSICHT

```
height = playout:GetHeight()
```

BESCHREIBUNG

Gibt die Höhe des für die Ellipsenbildung verwendeten Layouts zurück. Siehe playout:SetHeight() für Details.

Diese Funktion gibt die Höhe zurück, in Pango-Einheiten, falls positiv, oder die Anzahl der Zeilen, falls negativ.

EINGABEN

keine

RÜCKGABEWERTE

`height` Layout-Höhe

32.15 playout:GetIndent

BEZEICHNUNG

playout:GetIndent – Gibt die Absatzeinrückung zurück

ÜBERSICHT

```
indent = playout:GetIndent()
```

BESCHREIBUNG

Gibt die Breite der Absatzeinrückung in Pango-Einheiten zurück. Ein negativer Wert bedeutet einen hängenden Einrückung.

EINGABEN

keine

RÜCKGABEWERTE

`indent` die Einrückung in Pango-Einheiten

32.16 playout:GetIter

BEZEICHNUNG

playout:GetIter – Gibt den Layout-Iterator zurück

ÜBERSICHT

```
it = playout:GetIter()
```

BESCHREIBUNG

Gibt einen Iterator zurück, der über die visuellen Ausmaße des Layouts iteriert. Der Aufrufer ist dafür verantwortlich, den Iterator mit playoutiter:Free() zu löschen.

EINGABEN

keine

RÜCKGABEWERTE

`it` der neue Pango-Layout-Iterator

32.17 playout:GetJustify

BEZEICHNUNG

playout:GetJustify – Gibt die Zeilenausrichtung zurück (Blocksatz)

ÜBERSICHT

```
justify = playout:GetJustify()
```

BESCHREIBUNG

Ermittelt, ob jede vollständige Zeile so gestreckt werden soll, dass sie die gesamte Breite des Layouts ausfüllt oder nicht (Blocksatz).

EINGABEN

keine

RÜCKGABEWERTE

`justify` ein boolescher Wert

32.18 playout:GetLine

BEZEICHNUNG

playout:GetLine – Gibt die Zeile zurück

ÜBERSICHT

```
handle = playout:GetLine(line)
```

BESCHREIBUNG

Gibt eine bestimmte Zeile aus einem Pango-Layout zurück.

Verwenden Sie die schnellere Funktion playout:GetLineReadonly(), wenn Sie den Inhalt der Zeile (Glyphen, Glyphenbreiten usw.) nicht ändern wollen.

Diese Funktion gibt die angeforderte Pango-Layoutzeile zurück oder ein NULL-Objekt, wenn der Index außerhalb des Bereichs liegt. Die zurückgegebene Pango-Layoutzeile gehört zum Layout und darf nicht gelöscht werden. Sie kann referenziert und beibehalten werden, wird aber ungültig, wenn Änderungen am Pango-Layout vorgenommen werden.

EINGABEN

`line` der Index einer Zeile, der zwischen 0 und der Anzahl der Zeilen im Layout minus eins liegen muss

RÜCKGABEWERTE

`handle` ein Pango-Layout-Zeilenobjekt

32.19 playout:GetLineCount

BEZEICHNUNG

playout:GetLineCount – Gibt die Anzahl der Zeilen im Layout zurück

ÜBERSICHT

```
n = playout:GetLineCount()
```

BESCHREIBUNG
Ermittelt die Anzahl der Zeilen für das Layout.

EINGABEN
keine

RÜCKGABEWERTE

`n`	die Zeilenzahl

32.20 playout:GetLineReadonly

BEZEICHNUNG
playout:GetLineReadonly – Gibt die Layoutzeile im Nur-Lese-Modus zurück

ÜBERSICHT
```
handle = playout:GetLineReadonly(line)
```

BESCHREIBUNG
Gibt eine bestimmte Zeile aus einem Pango-Layout zurück.

Dies ist eine schnellere Alternative zu playout:GetLine(), aber der Benutzer kann den Inhalt der Zeile (Glyphen, Glyphenbreiten usw.) nicht ändern.

Diese Funktion gibt die angeforderte Pango-Layoutzeile zurück oder ein NULL-Objekt, wenn der Index außerhalb des Bereichs liegt. Die zurückgegebene Pango-Layoutzeile gehört zum Layout und darf nicht gelöscht werden. Sie kann referenziert und beibehalten werden, wird aber ungültig, wenn Änderungen am Pango-Layout vorgenommen werden.

EINGABEN

`line`	der Index einer Zeile, der zwischen 0 und der Anzahl der Zeilen im Layout minus eins liegen muss

RÜCKGABEWERTE

`handle`	ein Pango-Layout-Zeilenobjekt

32.21 playout:GetLineSpacing

BEZEICHNUNG
playout:GetLineSpacing – Gibt die Zeilenabstände zurück

ÜBERSICHT
```
spacing = playout:GetLineSpacing()
```

BESCHREIBUNG
Ruft den Zeilenabstandsfaktor des Layouts ab. Siehe playout:SetLineSpacing() für weitere Informationen.

EINGABEN
keine

RÜCKGABEWERTE

`spacing`	der Zeilenabstandsfaktor

32.22 playout:GetLines

BEZEICHNUNG
playout:GetLines – Gibt die Layoutzeilen zurück

ÜBERSICHT
```
list = playout:GetLines()
```

BESCHREIBUNG
Gibt die Zeilen des Layouts als Liste (Tabelle) zurück.

Verwenden Sie den schnelleren playout:GetLinesReadonly(), wenn Sie den Inhalt der Zeilen (Glyphen, Glyphenbreiten usw.) nicht ändern wollen.

Diese Funktion gibt eine Liste mit den Zeilen des Layouts zurück. Die Pango-Layout-Zeilenobjekte in der Liste müssen mit Vorsicht verwendet werden. Sie werden bei jeder Änderung des Textes oder der Eigenschaften des Layouts ungültig.

EINGABEN
keine

RÜCKGABEWERTE
`list` Tabelle mit einer Liste von Pango-Layoutzeilenobjekten

32.23 playout:GetLinesReadonly

BEZEICHNUNG
playout:GetLinesReadonly – Gibt die Layoutzeilen im Nur-Lese-Modus zurück

ÜBERSICHT
```
list = playout:GetLinesReadonly()
```

BESCHREIBUNG
Gibt die Zeilen des Layouts als Liste (Tabelle) nur zum Lesen zurück.

Dies ist eine schnellere Alternative zu playout:GetLine(), aber der Benutzer kann den Inhalt der Zeile (Glyphen, Glyphenbreiten usw.) nicht ändern.

Diese Funktion gibt eine Liste mit den Zeilen des Layouts zurück. Die Pango-Layout-Zeilenobjekte in der Liste müssen mit Vorsicht verwendet werden. Sie werden bei jeder Änderung des Textes oder der Eigenschaften des Layouts ungültig.

EINGABEN
keine

RÜCKGABEWERTE
`list` Tabelle mit einer Liste von Pango-Layoutzeilenobjekten

32.24 playout:GetLogAttrs

BEZEICHNUNG
playout:GetLogAttrs – Gibt die logischen Attribute für Zeichen zurück

ÜBERSICHT

```
t = playout:GetLogAttrs()
```

BESCHREIBUNG

Gibt ein Array mit logischen Attributen für jedes Zeichen im Layout zurück. Diese Funktion gibt eine Tabelle zurück, die eine Untertabelle für jedes Zeichen im Layout enthält. Für jede Untertabelle werden die folgenden Felder initialisiert:

`IsLineBreak`
: Wenn gesetzt, kann die Zeile vor dem Zeichen umbrechen.

`IsCharBreak`
: Falls gesetzt, kann hier beim Zeichenumbruch ein Umbruch entstehen.

`IsWhite` Ist ein Leerzeichen.

`IsCursorPosition`
: Wenn dieses Flag gesetzt ist, kann der Cursor vor einem Zeichen stehen, d.h. es handelt sich um eine Graphemgrenze oder das erste Zeichen im Text. Dieses Flag implementiert die Unicode-Semantik der Graphem-Clustergrenzen.

`IsWordStart`
: Ist das erste Zeichen eines Wortes.

`IsWordEnd`
: Ist das erste Nicht-Wort-Zeichen nach einem Wort. Beachten Sie, dass in degenerierten Fällen sowohl `IsWordStart` als auch `IsWordEnd` für ein Zeichen gesetzt sein können.

`IsSentenceBoundary`
: Ist eine Satzgrenze. Es gibt zwei Möglichkeiten, Sätze zu unterteilen. Die erste ordnet alle Leer-/Kontroll-/Formatzeichen zwischen den Sätzen einem Satz zu, so dass alle Zeichen in einem Satz sind. `IsSentenceBoundary` bezeichnet die Grenzen dort. Bei der zweiten Möglichkeit werden die Leerzeichen zwischen den Sätzen usw. keinem Satz zugeordnet, so dass `IsSentenceStart` / `IsSentenceEnd` die Grenzen dieser Sätze markiert.

`IsSentenceStart`
: Ist das erste Zeichen in einem Satz.

`IsSentenceEnd`
: Ist das erste Zeichen nach einem Satz. Beachten Sie, dass in degenerierten Fällen sowohl `IsSentenceStart` als auch `IsSentenceEnd` für ein bestimmtes Zeichen gesetzt sein können (z.B. kein Leerzeichen nach einem Punkt, so dass der nächste Satz sofort beginnt).

`BackspaceDeletesCharacter`
: Wenn diese Option gesetzt ist, löscht die Rücktaste ein Zeichen und nicht den gesamten Graphem-Cluster. Dieses Feld ist nur an Graphemgrenzen sinnvoll (wenn `IsCursorPosition` gesetzt ist). In einigen Sprachen wird das gesamte Graphem (z.B. Buchstabe + diakritische Zeichen) als Einheit betrachtet, während in anderen Sprachen jedes zerlegte Zeichen des Graphems eine Einheit darstellt. In der Standardimplementierung von pango-Umbruch (pango

break) ist dieses Bit bei allen Graphemgrenzen gesetzt, außer bei denen, die auf lateinische, kyrillische oder griechische Basiszeichen folgen.

`IsExpandableSpace`
: Ist ein Leerzeichen, das möglicherweise zu Ausrichtungszwecken erweitert werden kann.

`IsWordBoundary`
: Ist eine Wortgrenze, wie in UAX#29 definiert. Genauer gesagt bedeutet dies, dass es sich nicht um eine Position in der Mitte eines Wortes handelt. Zum Beispiel werden beide Seiten eines Satzzeichens als Wortgrenzen betrachtet. Dieses Flag ist besonders nützlich, wenn Text Wort für Wort ausgewählt wird. Dieses Flag implementiert die Semantik von Unicodes-Wortgrenzen.

Die Anzahl der in der Tabelle zurückgegebenen Attribute ist um eins höher als die Gesamtzahl der Zeichen im Layout, da es sowohl für die Position vor dem ersten Zeichen als auch für die Position nach dem letzten Zeichen Attribute geben muss.

EINGABEN

keine

RÜCKGABEWERTE

`t` Tabelle mit einer Reihe von logischen Attributen

32.25 playout:GetPixelExtents

BEZEICHNUNG

playout:GetPixelExtents – Gibt die Pixelausmaße des Layouts zurück

ÜBERSICHT

```
ink_rect, logical_rect = playout:GetPixelExtents()
```

BESCHREIBUNG

Berechnet die logischen und farblichen Ausdehnungen des Layouts in Geräteeinheiten. Beide Rückgabewerte sind Tabellen, bei denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Diese Funktion ruft einfach playout:GetExtents() auf, gefolgt von zwei Aufrufen von pango.ExtentsToPixels(), wobei `ink_rect` und `logical_rect` so gerundet werden, dass die gerundeten Rechtecke die ungerundeten Rechtecke vollständig enthalten (d.h. sie werden als erstes Argument an pango.ExtentsToPixels() übergeben).

EINGABEN

keine

RÜCKGABEWERTE

`ink_rect` Ausmaße des gezeichneten Layouts

`logical_rect`
: logische Ausmaße des Layouts

32.26 playout:GetPixelSize

BEZEICHNUNG

playout:GetPixelSize – Gibt die Layout-Abmessungen zurück

ÜBERSICHT

```
width, height = playout:GetPixelSize()
```

BESCHREIBUNG

Gibt die logische Breite und Höhe eines Pango-Layouts in Geräteeinheiten zurück.

playout:GetSize() gibt die mit `#PANGO_SCALE` skalierte Breite und Höhe zurück. Dies ist einfach eine Komfortfunktion von playout:GetPixelExtents().

EINGABEN

keine

RÜCKGABEWERTE

`width`	logische Breite
`height`	logische Höhe

32.27 playout:GetSerial

BEZEICHNUNG

playout:GetSerial – Gibt die Seriennummer des Layouts zurück

ÜBERSICHT

```
s = playout:GetSerial()
```

BESCHREIBUNG

Gibt die aktuelle Seriennummer des Layouts zurück.

Die Seriennummer wird bei der Erstellung eines neuen Layouts mit einer kleinen Zahl größer als Null initialisiert und immer dann erhöht, wenn das Layout mit einer der Set-Funktionen geändert wird oder sich der verwendete Pango-Kontext geändert hat. Die Serie kann umgebrochen werden, wird aber nie den Wert 0 haben. Da sie umgebrochen werden kann, sollte man sie nie mit "kleiner als" vergleichen, sondern immer "nicht gleich" verwenden.

Dies kann verwendet werden, um Änderungen an einem Pango-Layout automatisch zu erkennen und ist zum Beispiel nützlich, um zu entscheiden, ob ein Layout neu gezeichnet werden muss. Um zu erzwingen, dass die Serie erhöht wird, verwenden Sie playout:ContextChanged().

EINGABEN

keine

RÜCKGABEWERTE

`s`	die aktuelle Seriennummer des Layouts

32.28 playout:GetSingleParagraphMode

BEZEICHNUNG
playout:GetSingleParagraphMode – Gibt den Einzelabsatzmodus zurück

ÜBERSICHT
```
bool = playout:GetSingleParagraphMode()
```

BESCHREIBUNG
Ermittelt, ob sich das Layout im Einzelabsatzmodus befindet. Siehe playout:SetSingleParagraphMode() für Details.

Diese Funktion gibt `True` zurück, wenn das Layout die Absätze nicht an Absatztrennzeichen umbricht, andernfalls `False`.

EINGABEN
keine

RÜCKGABEWERTE

`bool`	`True`, wenn das Layout keine Absätze umbricht, sonst `False`

32.29 playout:GetSize

BEZEICHNUNG
playout:GetSize – Gibt die Layout-Abmessungen zurück

ÜBERSICHT
```
width, height = playout:GetSize()
```

BESCHREIBUNG
Gibt die logische Breite und Höhe eines Pango-Layouts in Pango-Einheiten zurück.

Dies ist lediglich eine Komfortfunktion für playout:GetExtents().

EINGABEN
keine

RÜCKGABEWERTE

`width`	logische Breite
`height`	logische Höhe

32.30 playout:GetSpacing

BEZEICHNUNG
playout:GetSpacing – Gibt den Zeilenabstand zurück

ÜBERSICHT
```
spacing = playout:GetSpacing()
```

BESCHREIBUNG
Gibt den Abstand zwischen den Zeilen des Layouts zurück.

EINGABEN
keine

RÜCKGABEWERTE
`spacing` den Abstand in Pango-Einheiten

32.31 playout:GetTabs

BEZEICHNUNG
playout:GetTabs – Gibt das Tabulator-Array zurück

ÜBERSICHT
```
handle = playout:GetTabs()
```

BESCHREIBUNG
Gibt das aktuelle Pango-Tabulator-Array zurück, das von diesem Layout verwendet wird.

Wenn kein Pango-Tabulator-Array gesetzt wurde, werden die Standard-Tabulatoren verwendet und ein NULL-Objekt zurückgegeben. Sie können die Funktion ptabarray:IsNull() verwenden, um zu überprüfen, ob ein Tabulator-Array `NULL` ist. Standardmäßig werden die Tabulatoren alle 8 Leerzeichen gesetzt.

Der Rückgabewert sollte, wenn er nicht mehr gebraucht wird, mit ptabarray:Free() gelöscht werden.

EINGABEN
keine

RÜCKGABEWERTE
`handle` ein Pango-Tabulator-Array

32.32 playout:GetText

BEZEICHNUNG
playout:GetText – Gibt den Text zurück

ÜBERSICHT
```
s$ = playout:GetText()
```

BESCHREIBUNG
Gibt den Text im Layout zurück.

EINGABEN
keine

RÜCKGABEWERTE
`s$` den Text im Layout

32.33 playout:GetUnknownGlyphsCount

BEZEICHNUNG

playout:GetUnknownGlyphsCount – Gibt die Anzahl unbekannter Glyphen zurück

ÜBERSICHT

```
n = playout:GetUnknownGlyphsCount()
```

BESCHREIBUNG

Zählt die Anzahl der unbekannten Glyphen im Layout und gibt die Anzahl zurück.

Diese Funktion kann verwendet werden, um festzustellen, ob es Schriftarten gibt, die alle Zeichen in einer bestimmten Zeichenkette darstellen können. In Kombination mit `#PANGO_ATTR_FALLBACK` können Sie außerdem überprüfen, ob eine bestimmte Schriftart alle Zeichen in der Zeichenkette unterstützt.

EINGABEN

keine

RÜCKGABEWERTE

`n` die Anzahl der unbekannten Glyphen im Layout

32.34 playout:GetWidth

BEZEICHNUNG

playout:GetWidth – Gibt die Umbruchbreite zurück

ÜBERSICHT

```
width = playout:GetWidth()
```

BESCHREIBUNG

Gibt die Breite zurück, auf die die Zeilen des Pango-Layouts umgebrochen werden sollen. Diese Funktion gibt die Breite in Pango-Einheiten zurück oder -1, wenn keine Breite festgelegt wurde.

EINGABEN

keine

RÜCKGABEWERTE

`width` die Breite in Pango-Einheiten, oder -1, wenn keine Breite festgelegt wurde

32.35 playout:GetWrap

BEZEICHNUNG

playout:GetWrap – Gibt den Umbruchmodus zurück

ÜBERSICHT

```
mode = playout:GetWrap()
```

BESCHREIBUNG

Gibt den Umbruchmodus für das Layout zurück. Siehe playout:SetWrap() für eine Liste der unterstützten Umbruchmodi.

Verwenden Sie playout:IsWrapped(), um abzufragen, ob Absätze tatsächlich umgebrochen wurden.

EINGABEN

keine

RÜCKGABEWERTE

`mode` aktiver Umbruchmodus

32.36 playout:IndexToLineX

BEZEICHNUNG

playout:IndexToLineX – Konvertiert den Index zur Zeile x

ÜBERSICHT

```
line, x_pos = playout:IndexToLineX(index, trailing)
```

BESCHREIBUNG

Konvertiert von Byte `Index` innerhalb des Layouts in Zeile und X-Position. Die X-Position wird vom linken Rand der Linie aus gemessen.

EINGABEN

`index` der Byte-Index eines Graphems innerhalb des Layouts

`trailing` eine Ganzzahl, die den Rand des Graphems angibt, dessen Position ermittelt werden soll. Wenn >0, der hintere Rand des Graphems: wenn 0, der vordere Rand des Graphems

RÜCKGABEWERTE

`line` Zeilenindex (liegt zwischen 0 und der Anzahl der Layoutzeilen minus 1)

`x_pos` Position innerhalb der Zeile (`#PANGO_SCALE` Einheiten pro Geräteeinheit)

32.37 playout:IndexToPos

BEZEICHNUNG

playout:IndexToPos – Gibt die Position des Graphems auf dem Bildschirm zurück

ÜBERSICHT

```
pos = playout:IndexToPos(index)
```

BESCHREIBUNG

Konvertiert von einem Index innerhalb eines Pango-Layouts in die auf dem Bildschirm entsprechende Position des Graphems (an diesem Index).

Der Rückgabewert ist eine Tabelle mit den Feldern `x`, `y`, `width` und `height` zur Beschreibung eines Rechtecks. Beachten Sie, dass `x` immer der vordere Rand des Graphems ist und `x+width` der hintere Rand des Graphems. Wenn die Richtung des Graphems von rechts nach links ist, ist `width` negativ.

EINGABEN

`index` Byte-Index innerhalb des Layouts

RÜCKGABEWERTE

`pos` Rechteck, das die Position des Graphems enthält

32.38 playout:IsEllipsized

BEZEICHNUNG

playout:IsEllipsized – Überprüft auf ellipsenförmige Absätze

ÜBERSICHT

```
bool = playout:IsEllipsized()
```

BESCHREIBUNG

Diese Funktion gibt zurück, ob das Layout Absätze ellipsenförmig gestalten musste.

Dies gibt `True` zurück, wenn der Ellipsenmodus für das Layout nicht `#PANGO_ELLIPSIZE_NONE` ist, eine positive Breite im Layout eingestellt ist und es Absätze gibt, die diese Breite überschreiten und ellipsenförmig gestaltet werden müssen.

EINGABEN

keine

RÜCKGABEWERTE

`bool` `True`, wenn Absätze ellipsenförmig gestaltet werden mussten, sonst `False`

32.39 playout:IsNull

BEZEICHNUNG

playout:IsNull – Überprüft, ob das Layout ungültig ist

ÜBERSICHT

```
bool = playout:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Layout `NULL` ist, d.h. ungültig. Wenn Funktionen, die Layouts zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Layout einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Layoutzuweisung fehlgeschlagen ist. In diesem Fall wird das Layout auf `NULL` gesetzt.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Mit dieser Funktion können Sie überprüfen, ob ein Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

keine

RÜCKGABEWERTE

`bool` `True`, wenn das Layout `NULL` ist, sonst `False`

32.40 playout:IsWrapped

BEZEICHNUNG

playout:IsWrapped – Überprüft, ob Absätze umgebrochen sind

ÜBERSICHT

```
bool = playout:IsWrapped()
```

BESCHREIBUNG

Diese Funktion fragt ab, ob das Layout Absätze umgebrochen hat.

Dies gibt `True` zurück, wenn eine positive Breite im Layout eingestellt ist, der Ellipsenmodus des Layouts auf `#PANGO_ELLIPSIZE_NONE` gesetzt ist und es Absätze gibt, die die Layoutbreite überschreiten und umgebrochen werden müssen.

EINGABEN

keine

RÜCKGABEWERTE

`bool` `True`, wenn Absätze umgebrochen werden mussten, sonst `False`

32.41 playout:MoveCursorVisually

BEZEICHNUNG

playout:MoveCursorVisually – Bewegt den Cursor visuell

ÜBERSICHT

```
nidx, ntrail = playout:MoveCursorVisually(strong, oidx, otrail, dir)
```

BESCHREIBUNG

Berechnet eine neue Cursorposition aus einer alten Position und einer Richtung.

Wenn `dir` positiv ist, wird die neue Position dazu führen, dass der starke oder schwache Cursor eine Position rechts von der Position der alten Cursorposition angezeigt wird. Wenn `dir` negativ ist, wird es nach links bewegt.

Bei bidirektionalem Text hängt die Übereinstimmung zwischen logischer und visueller Reihenfolge von der Richtung des aktuellen Laufs ab und es können Sprünge auftreten, wenn der Cursor vom über das Ende eines Laufs hinaus bewegt wird.

Die Bewegung erfolgt hier in Cursorpositionen, nicht in Zeichen, so dass ein einziger Aufruf von dieser Funktion den Cursor über mehrere Zeichen bewegen kann, wenn mehrere Zeichen zu einem einzigen Graphem kombiniert werden.

EINGABEN

`strong` ob der sich bewegende Cursor der starke oder der schwache Cursor ist; der starke Cursor ist der Cursor, der der Texteinfügung in der Basisrichtung für das Layout entspricht

`oidx` der Byte-Index der aktuellen Cursorposition

`otrail` wenn 0, befand sich der Cursor an der vorderen Kante des durch `oidx` angegebenen Graphems; wenn > 0, befand er sich an der hinteren Kante

`dir` Richtung, in die sich der Cursor bewegen soll; ein negativer Wert bedeutet eine Bewegung nach links

RÜCKGABEWERTE

`nidx` neuer Cursor-Byte-Index; ein Wert von -1 bedeutet, dass der Cursor vom Anfang des Layouts verschoben wurde; ein Wert von `#G_MAXINT` bedeutet, dass der Cursor vom Ende des Layouts verschoben wurde

`ntrail` Anzahl der Zeichen, die von der für `nidx` zurückgegebenen Position vorwärts zu bewegen sind, um die Position zu erhalten, an der der Cursor angezeigt werden soll; dies ermöglicht die Unterscheidung zwischen der Position am Anfang einer Zeile und der Position am Ende der vorangegangenen Zeile; `nind` befindet sich immer in der Zeile, in der der Cursor angezeigt werden soll

32.42 playout:Reference

BEZEICHNUNG

playout:Reference – Erhöht die Anzahl der Referenzen der Pango-Layouts

ÜBERSICHT

```
playout:Reference()
```

BESCHREIBUNG

Erhöhen Sie die Anzahl der Verweise im Pango-Layout um eins.

EINGABEN

keine

32.43 playout:SetAlignment

BEZEICHNUNG

playout:SetAlignment – Legt die Ausrichtung fest

ÜBERSICHT

```
playout:SetAlignment(alignment)
```

BESCHREIBUNG

Legt die Ausrichtung für das Layout fest: wie Teilzeilen innerhalb des verfügbaren horizontalen Bereichs positioniert werden. Der Parameter `alignment` kann eine der folgenden Konstanten sein:

`#PANGO_ALIGN_LEFT`
Platziert den gesamten verfügbaren Bereich auf der rechten Seite.

`#PANGO_ALIGN_CENTER`
Zentriert die Zeile innerhalb des verfügbaren Bereichs.

`#PANGO_ALIGN_RIGHT`
Platziert den gesamten verfügbaren Bereich auf der linken Seite.

Die Standardausrichtung ist `#PANGO_ALIGN_LEFT`.

EINGABEN

`alignment`
gewünschte Ausrichtung (siehe oben)

32.44 playout:SetAttributes

BEZEICHNUNG

playout:SetAttributes – Legt die Attribute fest

ÜBERSICHT

```
playout:SetAttributes(attrs)
```

BESCHREIBUNG

Legt die Textattribute für ein Layoutobjekt fest.

Diese Funktion verweist auf `attrs`, so dass der Aufrufer den Verweis auf `attrs` aufheben kann.

EINGABEN

`attrs` ein Pango-Attribut-Listenobjekt

32.45 playout:SetAutoDir

BEZEICHNUNG

playout:SetAutoDir – Setzt die automatische Richtung

ÜBERSICHT

```
playout:SetAutoDir(auto_dir)
```

BESCHREIBUNG

Legt fest, ob die Basisrichtung für das Layout entsprechend seinem Inhalt berechnet werden soll.

Wenn dieses Flag aktiviert ist (voreingestellt), werden Absätze im Layout, die mit starken Rechts-nach-Links-Zeichen beginnen (vor allem Arabisch und Hebräisch), von rechts nach links angeordnet, Absätze mit Buchstaben aus anderen Schriften von links nach rechts. Absätze mit ausschließlich neutralen Zeichen werden von den umgebenden Absätzen ausgerichtet.

Bei `False` erfolgt die Wahl zwischen Links-nach-Rechts- und Rechts-nach-Links-Layout entsprechend der Basisrichtung des Pango-Kontexts des Layouts. Siehe pcontext:SetBaseDir() für Einzelheiten.

Wenn die automatisch berechnete Richtung eines Absatzes von der Basisrichtung des Kontexts abweicht, werden die Interpretationen von `#PANGO_ALIGN_LEFT` und `#PANGO_ALIGN_RIGHT` vertauscht.

EINGABEN

`auto_dir` wenn `True`, Berechnung der bidirektionalen Basisrichtung aus dem Inhalt des Layouts

32.46 playout:SetEllipsize

BEZEICHNUNG

playout:SetEllipsize – Setzt die Ellipsisierung

ÜBERSICHT

`playout:SetEllipsize(ellipsize)`

BESCHREIBUNG

Legt die Art der Ellipsisierung fest, die für das Layout durchgeführt wird.

Je nach Ellipsisierungsmodus `ellipsize` wird Text vom Anfang, der Mitte oder dem Ende des Textes entfernt, sodass er in die mit playout:SetWidth() und playout:SetHeight() festgelegte Breite und Höhe des Layouts passt.

Wenn das Layout Zeichen wie Zeilenumbrüche enthält, die ein Layout mit mehreren Absätzen erzwingen, hängt es von der eingestellten Höhe des Layouts ab, ob jeder Absatz einzeln oder das gesamte Layout als Ganzes ellipsenförmig dargestellt wird.

Die folgenden Werte sind für `ellipsize` möglich:

`#PANGO_ELLIPSIZE_NONE`
: Keine Ellipsenbildung.

`#PANGO_ELLIPSIZE_START`
: Lässt Zeichen am Anfang des Textes aus.

`#PANGO_ELLIPSIZE_MIDDLE`
: Lässt Zeichen in der Mitte des Textes aus.

`#PANGO_ELLIPSIZE_END`
: Lässt Zeichen am Ende des Textes aus.

Voreingestellt ist `#PANGO_ELLIPSIZE_NONE`.

Siehe playout:SetHeight() für Details.

EINGABEN

`ellipsize`
: der neue Ellipsisierungsmodus für das Layout (siehe oben)

32.47 playout:SetFontDescription

BEZEICHNUNG

playout:SetFontDescription – Legt die Schriftartbeschreibung fest

ÜBERSICHT

`playout:SetFontDescription(desc)`

BESCHREIBUNG

Legt die Standardschriftbeschreibung für das Layout fest.

Wenn im Layout keine Schriftbeschreibung eingestellt ist, wird die Schriftbeschreibung aus dem Kontext des Layouts verwendet.

EINGABEN

`desc` die neue Pango-Schriftartbeschreibung, um die aktuelle Schriftartbeschreibung aufzuheben

32.48 playout:SetHeight

BEZEICHNUNG

playout:SetHeight – Stellt die Höhe für die Ellipsisierung ein

ÜBERSICHT

```
playout:SetHeight(height)
```

BESCHREIBUNG

Legt die Höhe fest, in der das Pango-Layout ellipsenförmig dargestellt werden soll.

Es gibt zwei verschiedene Verhaltensweisen, je nachdem, ob `height` positiv oder negativ ist.

Wenn `height` positiv ist, ist dies die maximale Höhe des Layouts. Es werden nur Zeilen angezeigt, die in das Layout passen, und wenn Text ausgelassen wird, wird eine Ellipse hinzugefügt. In jedem Absatz wird mindestens eine Zeile angezeigt, unabhängig davon, wie klein der Wert für die Höhe ist. Bei einem Wert von Null wird genau eine Zeile für das gesamte Layout angezeigt.

Wenn `height` negativ ist, handelt es sich um die (negative) maximale Anzahl von Zeilen pro Absatz. Das heißt, die Gesamtzahl der angezeigten Zeilen kann durchaus höher sein als dieser Wert, wenn das Layout mehrere Textabsätze enthält. Der Standardwert -1 bedeutet, dass die erste Zeile eines jeden Absatzes ellipsenförmig dargestellt wird. Dieses Verhalten kann in Zukunft geändert werden, um pro Layout statt pro Absatz zu agieren.

Die Höheneinstellung hat nur Auswirkungen, wenn eine positive Breite im Layout eingestellt ist und der Ellipsisierungsmodus des Layouts nicht `#PANGO_ELLIPSIZE_NONE` ist. Das Verhalten ist undefiniert, wenn eine andere Höhe als -1 eingestellt ist und der Ellipsisierungsmodus auf `#PANGO_ELLIPSIZE_NONE` gesetzt ist. Dies kann sich in Zukunft ändern.

EINGABEN

`height` die gewünschte Höhe des Layouts in Pango-Einheiten, falls positiv, oder die gewünschte Anzahl von Zeilen, falls negativ

32.49 playout:SetIndent

BEZEICHNUNG

playout:SetIndent – Stellt den Einzug ein

ÜBERSICHT

```
playout:SetIndent(indent)
```

BESCHREIBUNG

Legt die Breite in Pango-Einheiten fest, um jeden Absatz einzurücken.

Ein negativer Wert von `indent` bewirkt einen hängenden Einzug. Das heißt, die erste Zeile hat die volle Breite, und die nachfolgenden Zeilen werden um den absoluten Wert von `indent` eingerückt.

Die Einstellung für den Einzug wird ignoriert, wenn die Layout-Ausrichtung auf `#PANGO_ALIGN_CENTER` gesetzt ist.

Voreingestellt ist 0.

EINGABEN

`indent` der Betrag, um den eingerückt werden soll

32.50 playout:SetJustify

BEZEICHNUNG

playout:SetJustify – Setzt den Blocksatz

ÜBERSICHT

```
playout:SetJustify(justify)
```

BESCHREIBUNG

Legt fest, ob jede komplette Zeile so gedehnt werden soll, dass sie die gesamte Breite des Layouts ausfüllt (Blocksatz).

Die Dehnung erfolgt normalerweise durch das Hinzufügen von Leerzeichen. Bei manchen Schriftsystemen (z.B. Arabisch) kann die Ausrichtung jedoch auch auf komplexere Weise erfolgen, beispielsweise durch die Erweiterung der Zeichen.

Beachten Sie, dass Tabulatoren und Blocksatz in Konflikt zueinander stehen: Durch den Blocksatz wird der Inhalt von seinen mit Tabulatoren ausgerichteten Positionen verschoben.

Der voreingestellte Wert ist `False`.

EINGABEN

`justify` ob die Zeilen im Layout im Blocksatz angeordnet werden sollen oder nicht

32.51 playout:SetLineSpacing

BEZEICHNUNG

playout:SetLineSpacing – Stellt den Zeilenabstand ein

ÜBERSICHT

```
playout:SetLineSpacing(factor)
```

BESCHREIBUNG

Legt einen Faktor für den Zeilenabstand fest. Typische Werte sind: 0 ; 1 ; 1,5 ; 2. Voreingestellt ist 0.

Wenn `factor` ungleich Null ist, werden die Zeilen so platziert, dass

```
baseline2 = baseline1 + factor * height2
```

`height2` die Zeilenhöhe der zweiten Zeile ist (wie durch die Schriftart(en) bestimmt). In diesem Fall wird der mit playout:SetSpacing() eingestellte Abstand ignoriert.

Ist `factor` gleich Null (Standardeinstellung), werden die Abstände wie bisher angewendet.

Hinweis: Für eine Semantik, die näher an der CSS-Eigenschaft Zeilenhöhe liegt, verwenden Sie das Attribut Zeilenhöhe in einer Pango-Attributliste.

EINGABEN

`factor` der neue Faktor für den Zeilenabstand

32.52 playout:SetMarkup

BEZEICHNUNG

playout:SetMarkup – Setzt die Textformatierung

ÜBERSICHT

```
playout:SetMarkup(text$[, length])
```

BESCHREIBUNG

Legt den Layouttext und die Attributliste aus formatiertem Text fest. Dies ersetzt den aktuellen Text und die Attributliste. Es ist dasselbe wie playout:SetMarkupWithAccel(), aber der formatierte Text wird nicht nach Tastaturkürzeln gescannt.

Mit der Auszeichnungssprache Pango können Sie Textformatierungen mithilfe von XML-Tags vornehmen. Das am häufigsten verwendete Tag in der Pango-Auszeichnungssprache ist das Tag `<span>`. Damit können Sie eine Reihe von Zeichen formatieren. Das Tag `<span>` unterstützt die folgenden Attribute:

`font`

`font_desc`
: Eine Zeichenkette zur Beschreibung der Schriftart, z.B. "Sans Italic 12". Siehe pango.FontDescription() für eine Beschreibung des Formats der Zeichenketten-Darstellung. Beachten Sie, dass alle anderen `<span>`-Attribute diese Beschreibung überschreiben. Wenn Sie also "Sans Italic" und auch ein Attribut style="normal" haben, erhalten Sie Sans normal, nicht kursiv.

`font_family`
`face` Der Name einer Schriftartenfamilie.

`font_size`
`size` Schriftgröße in 1024stel eines Punktes oder in Punkten (z.B. "12.5pt") oder eine der absoluten Größen "xx-small", "x-small", "small", "medium", "large", "x-large", "xx-large" oder ein Prozentsatz (z.B. "200%") oder eine der relativen Größen "smaller" oder "larger". Wenn Sie eine absolute Größe angeben wollen, ist es in der Regel einfacher, die Möglichkeit zu nutzen, mit "font" eine partielle Schriftbeschreibung anzugeben; Sie können font="12.5" statt size="12800" oder size="12.5pt" verwenden.

`font_style`
`style` Eine der Optionen "normal", "oblique", "italic".

`font_weight`
`weight` Eine der Optionen "ultralight", "light", "normal", "bold", "ultrabold", "heavy" oder eine numerische Stärke.

`font_variant`
`variant` Eine der Optionen "normal", "small-caps", "all-small-caps", "petite-caps", "all-petite-caps", "unicase", "title-caps".

`font_stretch`
`stretch` Eine der Optionen "ultracondensed", "extracondensed", "condensed", "semicondensed", "normal", "semiexpanded", "expanded", "extraexpanded", "ultraexpanded".

`font_features`
Eine durch Kommata getrennte Liste von Einstellungen für OpenType-Schriftarten, in der gleichen Syntax, wie sie von CSS akzeptiert wird. Z.B.: font_features='dlig=1, -kern, afrc on'.

`foreground`
`fgcolor`

`color` Eine RGB-Farbangabe wie "#00FF00" oder ein Farbname wie "red". Eine RGBA-Farbspezifikation wie "#00FF007F" wird so interpretiert, dass sie sowohl eine Vordergrundfarbe als auch ein Vordergrund-Alpha angibt.

`background`
`bgcolor` Eine RGB-Farbangabe wie "#00FF00" oder ein Farbname wie "red". Eine RGBA-Farbspezifikation wie "#00FF007F" wird so interpretiert, dass sie sowohl eine Hintergrundfarbe als auch ein Hintergrund-Alpha angibt.

`alpha`

`fgalpha` Ein Alphawert für die Vordergrundfarbe, entweder eine einfache ganze Zahl zwischen 1 und 65536 oder ein Prozentwert wie "50%".

`background_alpha`
`bgalpha` Ein Alphawert für die Hintergrundfarbe, entweder eine einfache ganze Zahl zwischen 1 und 65536 oder ein Prozentwert wie "50%".

`underline`
Eine der Optionen "none", "single", "double", "low", "error".

`underline_color`
Die Farbe der Unterstreichungen; eine RGB-Farbangabe wie "#00FF00" oder ein Farbname wie "red".

`overline` Eine der Optionen "none" oder "single".

`overline_color`
Die Farbe der Überschriften; eine RGB-Farbangabe wie "#00FF00" oder ein Farbname wie "red".

`rise` Vertikale Verschiebung, in Pango-Einheiten oder in Punkten (z.B. "5pt"). Kann für tiefgestellter Schrift negativ, für hochgestellter Schrift positiv sein.

`baseline_shift`
Vertikale Verschiebung. Im Gegensatz zu "rise" sind die Attribute "baseline_shift" kumulativ. Der Wert kann in Pango-Einheiten oder in Punkten (z.B. "5pt") oder "superscript" oder "subscript" sein.

`font_scale`
Ändert die Schriftgröße. Die möglichen Werte sind "superscript", "subscript" oder "small-caps". Dies ähnelt den Werten von font_size wie "smaller" oder "larger", verwendet jedoch Schriftmetriken, um die neue Größe zu ermitteln.

`strikethrough`
"true" oder "false", ob der Text durchgestrichen werden soll oder nicht.

`strikethrough_color`
Die Farbe der durchgestrichenen Zeile; eine RGB-Farbangabe wie "#00FF00" oder ein Farbname wie "red".

`fallback` "true" oder "false" ob Fallback aktiviert werden soll oder nicht. Wenn diese Option deaktiviert ist, werden nur Zeichen aus der am besten passenden Schriftart auf dem System verwendet. Es wird kein Fallback auf andere Schriftarten auf dem System durchgeführt, die die Zeichen im Text enthalten könnten. Fallback ist standardmäßig aktiviert. Die meisten Anwendungen sollten Fallback nicht deaktivieren.

`lang` Ein Sprachcode, der die Textsprache angibt.

`letter_spacing`
Abstand zwischen den Buchstaben in 1024stel eines Punktes.

`gravity` Eine der Optionen "south", "east", "north", "west", "auto".

`gravity_hint`
Eine der Optionen "natural", "strong", "line".

`show` Gibt an, welche Sonderzeichen sichtbar angezeigt werden sollen. Der Wert kann "none" oder eine Kombination aus "spaces", "line-breaks" und "ignorables" sein, welche durch "|" getrennt sind.

`insert_hyphens`
"true" oder "false" um anzugeben, ob beim Zeilenumbruch in der Mitte von Wörtern Bindestriche eingefügt werden sollen oder nicht.

`allow_breaks`
"true" oder "false" um anzugeben, ob ein Zeilenumbruch erlaubt ist oder nicht.

`line_height`
Setzt die Zeilenhöhe. Der Wert kann entweder ein Faktor (< 1024) sein, der zum Hochskalieren der logischen Ausdehnung von Läufen verwendet wird, oder ein absoluter Wert (in 1024stel eines Punktes).

`text_transform`
: Gibt an, wie Zeichen während der Formgebung umgewandelt werden. Die Werte können "none", "lowercase", "uppercase" oder "capitalize" sein.

`segment`
: Setzt die Wort- oder Satzgrenzen. Der Wert kann "word" oder "sentence" sein, um anzugeben, dass die Spanne als ein einzelnes Wort oder ein Satz behandelt werden soll. Sich überschneidende Segmente werden entsprechend aufgeteilt und die Zeilenumbrüche werden entsprechend angepasst.

Neben dem Tag `<span>` unterstützt die Auszeichnungssprache Pango auch die folgenden Komfort-Tags:

`<b>`
: Fett

`<big>`
: Vergrößert die Schrift relativ, entspricht `<span size="larger">`.

`<i>`
: Italic

`<s>`
: Durchgestrichen

`<sub>`
: Tiefgestellt

`<sup>`
: Hochgestellt

`<small>`
: Verkleinert die Schrift relativ, entspricht `<span size="smaller">`.

`<tt>`
: Schriftart Monospace

`<u>`
: Unterstrichen

Hier ist ein Beispiel für formatierten Text:

```
<span foreground="blue" size="x-large">Blue text</span> is <i>cool</i>.
```

Beachten Sie, dass Sie auch XML-Funktionen wie numerische Zeichenentitäten verwenden können, z.B. können Sie `&#169;` für das Copyright-Zeichen usw. verwenden.

EINGABEN

`text$`
: formatierter Text

`length`
: optional: Länge des formatierten Textes in Bytes (Standardwert ist -1, was bedeutet, dass die Länge der Zeichenkette verwendet wird)

32.53 playout:SetMarkupWithAccel

BEZEICHNUNG

playout:SetMarkupWithAccel – Formatiert mit Kürzel

ÜBERSICHT

```
accel_char = playout:SetMarkupWithAccel(text$, length, accel_marker)
```

BESCHREIBUNG

Legt den Layouttext und die Attributliste aus dem formatierten Text fest. Ersetzt die aktuelle Text- und die Attributliste.

Wenn `accel_marker` ungleich Null ist, markiert das angegebene Zeichen das darauf folgende Zeichen als Kürzel. Beispielsweise könnte `accel_marker` ein kaufmännisches

Und (&) oder ein Unterstrich (_) sein. Alle Zeichen, die als Kürzel markiert sind, erhalten das Attribut `#PANGO_UNDERLINE_LOW` und das erste so gekennzeichnete Zeichen wird in `accel_char` zurückgegeben. Zwei aufeinander folgende `accel_marker`-Zeichen ergeben ein einzelnes literales `accel_marker`-Zeichen.

EINGABEN

`text$` formatierter Text

`length` Länge des markierten Textes in Bytes (-1 für die Länge von `text$`)

`accel_marker`
Kürzel im Text

RÜCKGABEWERTE

`accel_char`
Rückgabeort für das erste lokalisierte Kürzel

32.54 playout:SetSingleParagraphMode

BEZEICHNUNG
playout:SetSingleParagraphMode – Stellt den Einzelabsatzmodus ein

ÜBERSICHT
`playout:SetSingleParagraphMode(setting)`

BESCHREIBUNG
Legt den Einzelabsatzmodus des Layouts fest.

Wenn `setting` `True` ist, werden Zeilenumbrüche und ähnliche Zeichen nicht als Absatztrennzeichen behandelt. Stattdessen wird der gesamte Text in einem einzigen Absatz gehalten und eine Glyphe für Absatztrennzeichen angezeigt. Wird verwendet, wenn Sie die Bearbeitung von Zeilenumbrüchen in einer einzelnen Textzeile zulassen möchten.

Voreingestellt ist `False`.

EINGABEN

`setting` die neue Einstellung

32.55 playout:SetSpacing

BEZEICHNUNG
playout:SetSpacing – Stellt den Abstand ein

ÜBERSICHT
`playout:SetSpacing(spacing)`

BESCHREIBUNG
Stellt den Abstand in Pango-Einheiten zwischen den Zeilen des Layouts fest.

Beim Platzieren von Zeilen mit Abstand ordnet Pango die Dinge so an, dass

```
line2.top = line1.bottom + spacing
```

Der voreingestellte Wert ist 0.

Hinweis: Seit Version 1.44 verwendet Pango die Zeilenhöhe (wie von der Schriftart bestimmt) für die Platzierung von Zeilen, wenn der Zeilenabstandsfaktor mit playout:SetLineSpacing() auf einen Wert ungleich Null gesetzt wurde. In diesem Fall wird der mit dieser Funktion in `spacing` eingestellte Abstand ignoriert.

Nächster Hinweis: Für eine Semantik, die näher an der CSS-Eigenschaft line-height liegt, siehe das Pango-Attribut line height.

EINGABEN

`spacing` die Größe des Abstandes

32.56 playout:SetTabs

BEZEICHNUNG

playout:SetTabs – Setzt die Tabulatoren

ÜBERSICHT

```
playout:SetTabs([tabs])
```

BESCHREIBUNG

Legt die für das Layout zu verwendenden Tabulatoren fest und setzt die voreingestellten Tabulatoren außer Kraft.

Das Pango-Layout setzt den Inhalt immer dann an die nächste Tabulatorposition, wenn es auf ein Tabulatorzeichen (U+0009) trifft.

Standardmäßig sind die Tabulatoren alle 8 Leerzeichen gesetzt. Wenn der Parameter `tabs` weggelassen wird, werden die Standardtabulatoren wiederhergestellt. `tabs` wird in das Layout kopiert; Sie müssen Ihre Kopie von `tabs` selbst löschen.

Beachten Sie, dass Tabulatoren und der Blocksatz in Konflikt zueinander stehen: Durch den Blocksatz wird der Inhalt von seinen mit Tabulatoren ausgerichteten Positionen wegbewegt. Das Gleiche gilt für eine andere Ausrichtungen als `#PANGO_ALIGN_LEFT`.

EINGABEN

`tabs` optional: ein Pango-Tabulator-Array

32.57 playout:SetText

BEZEICHNUNG

playout:SetText – Setzt den Text

ÜBERSICHT

```
playout:SetText(text$[, length])
```

BESCHREIBUNG

Legt den Text des Layouts fest.

Diese Funktion validiert `text` und gibt ungültige UTF-8 mit einer Platzhalterglyphe wieder.

Wenn Sie zuvor playout:SetMarkup() oder playout:SetMarkupWithAccel() für das Layout verwendet haben, sollten Sie playout:SetAttributes() aufrufen, um die im Layout gesetzten Textformatierung zu löschen, da diese Funktion keine Attribute löscht.

EINGABEN

`text$` der Text

`length` optional: maximale Länge von `text` in Bytes (voreingestellt ist -1, was die Länge von `text$` bedeutet)

32.58 playout:SetWidth

BEZEICHNUNG

playout:SetWidth – Stellt die Umbruchbreite ein

ÜBERSICHT

```
playout:SetWidth(width)
```

BESCHREIBUNG

Legt die Breite fest, auf die die Zeilen des Pango-Layouts umgebrochen oder ellipsenförmig werden sollen.

Der Standardwert ist -1, was bedeutet, dass kein Umbruch oder keine Ellipsenbildung angefordert wird.

EINGABEN

`width` die gewünschte Breite in Pango-Einheiten, oder -1, um anzugeben, dass kein Umbruch oder keine Ellipsenbildung durchgeführt werden soll.

32.59 playout:SetWrap

BEZEICHNUNG

playout:SetWrap – Stellt den Umbruchmodus ein

ÜBERSICHT

```
playout:SetWrap(wrap)
```

BESCHREIBUNG

Legt den Umbruchmodus fest. Der Umbruchmodus ist nur wirksam, wenn im Layout mit playout:SetWidth() eine Breite festgelegt wurde. Um den Umbruch zu deaktivieren, setzen Sie die Breite auf -1.

Die folgenden Umbruchmodi werden derzeit unterstützt:

`#PANGO_WRAP_WORD`
: Umbruch von Zeilen an Wortgrenzen.

`#PANGO_WRAP_CHAR`
: Zeilenumbruch an Zeichengrenzen.

`#PANGO_WRAP_WORD_CHAR`
: Zeilenumbruch an Wortgrenzen, aber Rückgriff auf Zeichengrenzen, wenn nicht genügend Platz für ein ganzes Wort vorhanden ist.

Voreingestellt ist `#PANGO_WRAP_WORD`.

EINGABEN

`wrap` der Umbruchmodus (siehe oben)

32.60 playout:XYToIndex

BEZEICHNUNG

playout:XYToIndex – Wandelt XY in Byte-Index um

ÜBERSICHT

```
inside, index, trailing = playout:XYToIndex(x, y)
```

BESCHREIBUNG

Konvertiert die X- und Y-Position innerhalb eines Layouts in den Byte-Index des Zeichens an dieser logischen Position.

Liegt die Y-Position nicht innerhalb des Layouts, wird die nächstgelegene Position gewählt (die Position wird innerhalb des Layouts festgehalten). Liegt die X-Position nicht innerhalb des Layouts, wird der Anfang oder das Ende der Zeile gewählt, wie für playoutline:XToIndex() beschrieben. Wenn entweder die X- oder die Y-Position nicht innerhalb des Layouts liegt, gibt die Funktion `False` zurück; bei einem exakten Treffer gibt sie `True` zurück.

Diese Funktion gibt `True` zurück, wenn die Koordinaten innerhalb des Textes waren, ansonsten `False`.

EINGABEN

`x` der X-Versatz (in Pango-Einheiten) vom linken Rand des Layouts

`y` der Y-Versatz (in Pango-Einheiten) vom linken Rand des Layouts

RÜCKGABEWERTE

`inside` `True`, wenn die Koordinaten innerhalb des Textes waren, sonst `False`

`index` berechneter Byte-Index

`trailing` Ganzzahl, die angibt, an welcher Stelle des Graphems der Benutzer geklickt hat; sie ist entweder Null oder die Anzahl der Zeichen im Graphem; 0 steht für den Anfang des Graphems

33 Pango-Layout-Iterator

33.1 playoutiter:AtLastLine

BEZEICHNUNG
playoutiter:AtLastLine – Überprüft, ob der Iterator in der letzten Zeile ist

ÜBERSICHT
```
ok = playoutiter:AtLastLine()
```

BESCHREIBUNG
Ermittelt, ob sich der Iterator in der letzten Zeile des Layouts befindet.

EINGABEN
Keine

RÜCKGABEWERTE

`ok`	`True`, wenn der Iterator in der letzten Zeile steht

33.2 playoutiter:Copy

BEZEICHNUNG
playoutiter:Copy – Kopiert einen Layout-Iterator

ÜBERSICHT
```
iter = playoutiter:Copy()
```

BESCHREIBUNG
Kopiert einen Pango-Layout-Iterator.

Diese Funktion gibt den neu zugewiesenen Pango-Layout-Iterator zurück.

EINGABEN
Keine

RÜCKGABEWERTE

`iter`	der neu zugewiesene Pango-Layout-Iterator

33.3 playoutiter:Free

BEZEICHNUNG
playoutiter:Free – Löscht einen Layout-Iterator

ÜBERSICHT
```
playoutiter:Free()
```

BESCHREIBUNG
Löscht einen Iterator, der nicht mehr in Gebrauch ist.

EINGABEN
Keine

33.4 playoutiter:GetBaseline

BEZEICHNUNG
playoutiter:GetBaseline – Ermittelt die Grundlinie

ÜBERSICHT
```
baseline = playoutiter:GetBaseline()
```

BESCHREIBUNG
Ermittelt die Y-Position der Grundlinie der aktuellen Zeile in Layoutkoordinaten.

Der Ursprung der Layoutkoordinaten befindet sich oben links im gesamten Layout.

EINGABEN
Keine

RÜCKGABEWERTE

`baseline` Grundlinie der aktuellen Zeile

33.5 playoutiter:GetCharExtents

BEZEICHNUNG
playoutiter:GetCharExtents – Ermittelt die Zeichenausmaße

ÜBERSICHT
```
logical_rect = playoutiter:GetCharExtents()
```

BESCHREIBUNG
Ermittelt die Ausmaße des aktuellen Zeichens in Layoutkoordinaten. Dadurch wird eine Tabelle zurückgegeben, in der die Felder `x`, `y`, `width` und `height` initialisiert sind.

Der Ursprung der Layoutkoordinaten befindet sich oben links im gesamten Layout.

Für Zeichen können nur logische Ausmaße sinnvoll ermittelt werden; Farbausmaße sind nur bis zur Ebene der Cluster sinnvoll.

EINGABEN
Keine

RÜCKGABEWERTE

`logical_rect`
Rechteck gefüllt mit logischen Ausmaße

33.6 playoutiter:GetClusterExtents

BEZEICHNUNG
playoutiter:GetClusterExtents – Gibt die Cluster-Ausmaße zurück

ÜBERSICHT
```
ink_rect, logical_rect = playoutiter:GetClusterExtents()
```

BESCHREIBUNG

Ruft die Ausmaße des aktuellen Clusters in Layoutkoordinaten ab. Somit werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Der Ursprung der Layoutkoordinaten befindet sich oben links im gesamten Layout.

EINGABEN

Keine

RÜCKGABEWERTE

`ink_rect` Rechteck gefüllt mit Farbausmaße

`logical_rect`
Rechteck gefüllt mit logischen Ausmaßen

33.7 playoutiter:GetIndex

BEZEICHNUNG

playoutiter:GetIndex – Gibt den aktuellen Byte-Index zurück

ÜBERSICHT

```
idx = playoutiter:GetIndex()
```

BESCHREIBUNG

Ermittelt den aktuellen Byte-Index.

Beachten Sie, dass die Vorwärtsiteration um Zeichen in visueller und nicht in logischer Reihenfolge erfolgt, sodass Indizes möglicherweise nicht aufeinanderfolgend sind. Außerdem kann der Index der Länge des Textes im Layout entsprechen, wenn er auf `Nil` ausgeführt wird (siehe playoutiter:GetRun()).

EINGABEN

Keine

RÜCKGABEWERTE

`idx` aktueller Byte-Index

33.8 playoutiter:GetLayout

BEZEICHNUNG

playoutiter:GetLayout – Gibt den Layout-Iterator zurück

ÜBERSICHT

```
handle = playoutiter:GetLayout()
```

BESCHREIBUNG

Ruft das einem Pango-Layout-Iterator zugeordnete Layout ab. Dies gehört zum Iterator und darf nicht gelöscht werden.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` das dem Iterator zugeordnete Layout

33.9 playoutiter:GetLayoutExtents

BEZEICHNUNG

playoutiter:GetLayoutExtents – Gibt die Layout-Ausmaße zurück

ÜBERSICHT

```
ink_rect, logical_rect = playoutiter:GetLayoutExtents()
```

BESCHREIBUNG

Ruft die Ausmaße des Pango-Layouts ab, über das iteriert wird. Dadurch werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

EINGABEN

Keine

RÜCKGABEWERTE

`ink_rect` Rechteck gefüllt mit Farbausmaße

`logical_rect`
Rechteck gefüllt mit logischen Ausmaßen

33.10 playoutiter:GetLine

BEZEICHNUNG

playoutiter:GetLine – Ermittelt die aktuelle Zeile

ÜBERSICHT

```
handle = playoutiter:GetLine()
```

BESCHREIBUNG

Ruft die aktuelle Zeile ab. Die zurückgegebene Pango-Layoutzeile gehört zum Layout-Iterator und darf nicht gelöscht werden.

Verwenden Sie das schnellere playoutiter:GetLineReadonly(), wenn Sie nicht vorhaben, den Inhalt der Zeile (Glyphen, Glyphenbreiten usw.) zu ändern.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die aktuelle Zeile

33.11 playoutiter:GetLineExtents

BEZEICHNUNG

playoutiter:GetLineExtents – Ermittelt die Zeilenausmaße

ÜBERSICHT

```
ink_rect, logical_rect = playoutiter:GetLineExtents()
```

BESCHREIBUNG

Ermittelt die Ausmaße der aktuellen Zeile. Dadurch werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Die Ausmaße werden in Layoutkoordinaten angegeben (Ursprung ist die linke obere Ecke des gesamten Pango-Layouts). Daher haben die von dieser Funktion zurückgegebenen Ausmaße die gleiche Breite/Höhe, aber nicht das gleiche x/y-Verhältnis wie die von playoutline:GetExtents() zurückgegebenen Ausmaße.

EINGABEN

Keine

RÜCKGABEWERTE

`ink_rect` Rechteck gefüllt mit Farbausmaße

`logical_rect`
Rechteck gefüllt mit logischen Ausmaßen

33.12 playoutiter:GetLineReadonly

BEZEICHNUNG

playoutiter:GetLineReadonly – Ermittelt die Zeile im Nur-Lese-Zugriff/schreibgeschützt

ÜBERSICHT

```
handle = playoutiter:GetLineReadonly()
```

BESCHREIBUNG

Ruft die aktuelle Zeile für den Nur-Lese-Zugriff/schreibgeschützten Zugriff ab. Die zurückgegebene Pango-Layoutzeile gehört zum Layout-Iterator und darf nicht gelöscht werden.

Dies ist eine schnellere Alternative zu playoutiter:GetLine(), es ist jedoch nicht möglich, dass der Benutzer den Inhalt der Zeile (Glyphen, Glyphenbreiten usw.) ändert.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` die aktuelle Zeile

33.13 playoutiter:GetLineYRange

BEZEICHNUNG

playoutiter:GetLineYRange – Gibt den y-Bereich zurück

ÜBERSICHT

```
y0, y1 = playoutiter:GetLineYrange()
```

BESCHREIBUNG

Teilt den vertikalen Bereich im Pango-Layout, über den iteriert wird, zwischen den Zeilen im Layout auf und gibt den Bereich zurück, der zur aktuellen Zeile gehört.

Der Bereich einer Zeile umfasst die logischen Ausmaße der Zeile plus die Hälfte des Abstands über und unter der Zeile, wenn playout:SetSpacing() aufgerufen wurde, um den Layoutabstand festzulegen. Die Y-Positionen sind in Layout-Koordinaten angegeben (Ursprung links oben im gesamten Layout).

EINGABEN

Keine

RÜCKGABEWERTE

`y0`	Anfang der Zeile
`y1`	Zeilenende

33.14 playoutiter:GetRun

BEZEICHNUNG

playoutiter:GetRun – Gibt den aktuellen Lauf zurück

ÜBERSICHT

```
handle = playoutiter:GetRun()
```

BESCHREIBUNG

Gibt den aktuellen Lauf zurück. Der Lauf wird als Pango-Glyph-Element-Objekt zurückgegeben und darf nicht gelöscht werden.

Bei der Iteration nach Lauf gibt es am Ende jeder Zeile eine Position mit einem NULL-Lauf, so dass diese Funktion ein NULL-Handle zurückgeben kann. Der NULL-Lauf am Ende jeder Zeile stellt sicher, dass alle Zeilen mindestens einen Lauf haben, auch Zeilen, die nur aus einem Zeilenumbruch bestehen. Sie können pglyphitem:IsNull() verwenden, um zu prüfen, ob ein Glyphenelement `NULL` ist.

Verwenden Sie die schnellere Funktion playoutiter:GetRunReadonly(), wenn Sie den Inhalt des Laufs (Glyphen, Glyphenbreiten usw.) nicht ändern wollen.

EINGABEN

Keine

RÜCKGABEWERTE

`handle`	der aktuelle Lauf als Pango-Glyphen-Element

33.15 playoutiter:GetRunExtents

BEZEICHNUNG

playoutiter:GetRunExtents – Ruft die Ausmaße des Laufs ab

ÜBERSICHT

```
ink_rect, logical_rect = playoutiter:GetRunExtents()
```

BESCHREIBUNG

Ruft die Ausmaße des aktuellen Laufs in Layoutkoordinaten ab. Es werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Der Ursprung der Layoutkoordinaten befindet sich links oben im gesamten Layout.

EINGABEN

Keine

RÜCKGABEWERTE

`ink_rect` Rechteck gefüllt mit Farbe

`logical_rect`
Rechteck gefüllt mit logischen Ausdehnungen

33.16 playoutiter:GetRunReadonly

BEZEICHNUNG

playoutiter:GetRunReadonly – Gibt den aktuellen Lauf für Nur-Lese-Zugriff zurück

ÜBERSICHT

```
handle = playoutiter:GetRunReadonly()
```

BESCHREIBUNG

Gibt den aktuellen Lauf für den Nur-Lese-Zugriff zurück. Der Lauf wird als Pango-Glyph-Element-Objekt zurückgegeben und darf nicht gelöscht werden.

Bei der Iteration durch den Lauf gibt es am Ende jeder Zeile eine Position mit einem NULL-Lauf, so dass diese Funktion ein NULL-Objekt zurückgeben kann. Der NULL-Lauf am Ende jeder Zeile stellt sicher, dass alle Zeilen mindestens einen Lauf haben, auch Zeilen, die nur aus einem Zeilenumbruch bestehen. Sie können pglyphitem:IsNull() verwenden, um zu prüfen, ob ein Glyph-Element `NULL` ist.

Dies ist eine schnellere Alternative zur Funktion playoutiter:GetRun(), aber der Benutzer kann den Inhalt des Laufs (Glyphen, Glyphenbreiten usw.) nicht ändern.

EINGABEN

Keine

RÜCKGABEWERTE

`handle` der aktuelle Lauf als Pango-Glyphen-Element

33.17 playoutiter:IsNull

BEZEICHNUNG

playoutiter:IsNull – Überprüft, ob der Iterator ungültig ist

ÜBERSICHT

```
bool = playoutiter:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn der Iterator `NULL` ist, d.h. ungültig. Wenn Funktionen, die Iteratoren zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen den Iterator einfach auf `NULL`. Mit dieser Funktion können Sie überprüfen, ob die Iterator-Zuweisung fehlgeschlagen ist. In diesem Fall ist der Iterator `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn der Iterator `NULL` ist, sonst `False`

33.18 playoutiter:NextChar

BEZEICHNUNG

playoutiter:NextChar – Verschiebt den Iterator zum nächsten Zeichen

ÜBERSICHT

```
ok = playoutiter:NextChar()
```

BESCHREIBUNG

Verschiebt den Iterator vorwärts zum nächsten Zeichen in der visuellen Reihenfolge.

Diese Funktion gibt zurück, ob eine Verschiebung möglich war. Wenn sich der Iterator bereits am Ende des Layouts befand, wird `False` zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` ob eine Verschiebung möglich war

33.19 playoutiter:NextCluster

BEZEICHNUNG

playoutiter:NextCluster – Verschiebt den Iterator zum nächsten Cluster

ÜBERSICHT

```
ok = playoutiter:NextCluster()
```

BESCHREIBUNG

Verschiebt den Iterator zum nächsten Cluster in visueller Reihenfolge.

Diese Funktion gibt zurück, ob eine Verschiebung möglich war. Wenn sich der Iterator bereits am Ende des Layouts befand, wird `False` zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` ob eine Verschiebung möglich war

33.20 playoutiter:NextLine

BEZEICHNUNG

playoutiter:NextLine – Verschiebt den Iterator zur nächsten Zeile

ÜBERSICHT

```
ok = playoutiter:NextLine()
```

BESCHREIBUNG

Verschiebt den Iterator vorwärts zum Anfang der nächsten Zeile.

Diese Funktion gibt zurück, ob eine Verschiebung möglich war. Wenn der Iterator bereits in der letzten Zeile steht, wird `False` zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` ob eine Verschiebung möglich war

33.21 playoutiter:NextRun

BEZEICHNUNG

playoutiter:NextRun – Verschiebt den Iterator zum nächsten Lauf

ÜBERSICHT

```
ok = playoutiter:NextRun()
```

BESCHREIBUNG

Bewegt den Iterator in der visuellen Reihenfolge zum nächsten Lauf vorwärts.

Diese Funktion gibt zurück, ob eine Verschiebung möglich war. Wenn der Iterator bereits am Ende des Layouts war, wird `False` zurückgegeben.

EINGABEN

Keine

RÜCKGABEWERTE

`ok` ob eine Verschiebung möglich war

34 Pango-Layout-Zeile

34.1 playoutline:Free

BEZEICHNUNG
playoutline:Free – Löscht die Layoutzeile

ÜBERSICHT
`playoutline:Free()`

BESCHREIBUNG
Verringert die Referenzanzahl einer Pango-Layoutzeile um eins.

Ist das Ergebnis gleich Null, werden die Zeile und der gesamte zugehörige Speicher gelöscht.

EINGABEN
Keine

34.2 playoutline:GetExtents

BEZEICHNUNG
playoutline:GetExtents – Ermittelt die Maße der Layoutzeile

ÜBERSICHT
`ink_rect, logical_rect = playoutline:GetExtents()`

BESCHREIBUNG
Berechnet die logischen und farblichen Ausdehnungen einer Layoutzeile. Es werden zwei Tabellen zurückgegeben, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Siehe pfont:GetGlyphExtents() für Einzelheiten über die Interpretation der Rechtecke.

EINGABEN
Keine

RÜCKGABEWERTE

`ink_rect` Ausdehnung der ausgegebenen Glyphen-Zeichenkette

`logical_rect`
logische Ausdehnung der Zeichenkette

34.3 playoutline:GetHeight

BEZEICHNUNG
playoutline:GetHeight – Ermittelt die Zeilenhöhe

ÜBERSICHT
`height = playoutline:GetHeight()`

BESCHREIBUNG

Berechnet die Höhe der Zeile als das Maximum der Höhen der in dieser Zeile verwendeten Schriftarten.

Beachten Sie, dass der tatsächliche Abstand zwischen den Textzeilen von anderen Faktoren beeinflusst wird, wie playout:SetSpacing() und playout:SetLineSpacing().

EINGABEN

Keine

RÜCKGABEWERTE

`height` Rückgabestelle für die Zeilenhöhe

34.4 playoutline:GetLength

BEZEICHNUNG

playoutline:GetLength – Ermittelt die Zeilenlänge

ÜBERSICHT

```
len = playoutline:GetLength()
```

BESCHREIBUNG

Gibt die Länge der Zeile in Bytes zurück.

EINGABEN

Keine

RÜCKGABEWERTE

`len` die Länge der Zeile

34.5 playoutline:GetPixelExtents

BEZEICHNUNG

playoutline:GetPixelExtents – Gibt die Pixelmaße zurück

ÜBERSICHT

```
ink_rect, logical_rect = playoutline:GetPixelExtents()
```

BESCHREIBUNG

Berechnet die logischen und farblichen Ausdehnungen von `layout_line` in Geräteeinheiten. Dies gibt zwei Tabellen zurück, in denen die Felder `x`, `y`, `width` und `height` initialisiert sind.

Diese Funktion ruft einfach playoutline:GetExtents() auf, gefolgt von zwei Aufrufen von pango.ExtentsToPixels(), wobei `ink_rect` und `logical_rect` so gerundet werden, dass die abgerundeten Rechtecke das ungerundete vollständig enthalten (d.h. sie werden als erstes Argument an pango.ExtentsToPixels() übergeben).

EINGABEN

Keine

RÜCKGABEWERTE

`ink_rect` Ausdehnung der gezeichneten Glyphen-Zeichenkette

`logical_rect`
logische Ausdehnung der Glyphen-Zeichenkette

34.6 playoutline:GetRuns

BEZEICHNUNG

playoutline:GetRuns – Gibt den Lauf in der Zeile zurück

ÜBERSICHT

```
table = playoutline:GetRuns()
```

BESCHREIBUNG

Ruft alle Läufe in der Layoutzeile ab. Die Läufe werden in einer Tabelle zurückgegeben, wobei jedes Tabellenelement ein Pango-Glyphenobjekt ist, das nicht freigegeben werden darf.

EINGABEN

Keine

RÜCKGABEWERTE

`table` Tabelle mit allen Läufen in der Zeile als Pango-Glyphelemente

34.7 playoutline:GetXRanges

BEZEICHNUNG

playoutline:GetXRanges – Gibt die x-Bereiche zurück

ÜBERSICHT

```
ranges = playoutline:GetXRanges(start_index, end_index)
```

BESCHREIBUNG

Ruft eine Liste der visuellen Bereiche ab, die einem bestimmten logischen Bereich entsprechen. Dies gibt eine Liste mit einer Reihe von Tabellen zurück, in denen die Felder `x1` und `x2` zur Beschreibung eines Bereichs initialisiert sind.

Diese Liste ist nicht unbedingt minimal - es kann aufeinanderfolgende Bereiche geben, die aneinandergrenzen. Die Bereiche werden von links nach rechts sortiert. Die Bereiche beziehen sich auf den linken Rand des gesamten Layouts, nicht auf die Zeile.

EINGABEN

`start_index`
Startbyte-Index des logischen Bereichs; Wenn dieser Wert kleiner als der Startindex der Zeile ist, erstreckt sich der erste Bereich bis zur Vorderkante des Layouts; andernfalls beginnt es an der Vorderkante des ersten Zeichens

`end_index`
Endbyte-Index des logischen Bereichs; Wenn dieser Wert größer als der Endindex der Zeile ist, erstreckt sich der letzte Bereich bis zur Hinterkante des Layouts; andernfalls endet es an der Hinterkante des letzten Zeichens

RÜCKGABEWERTE

`ranges` eine Liste mit einer Reihe von Bereichen

34.8 playoutline:IndexToX

BEZEICHNUNG

playoutline:IndexToX – Konvertiert den Index in x-Position

ÜBERSICHT

```
x_pos = playoutline:IndexToX(index, trailing)
```

BESCHREIBUNG

Konvertiert einen Index innerhalb einer Zeile in eine X-Position.

EINGABEN

`index` Byte-Versatz eines Graphems innerhalb des Layouts

`trailing` eine ganze Zahl, die den Rand des Graphems angibt, dessen Position abgerufen werden soll; wenn > 0, der hintere Rand des Graphems, wenn 0, der vordere Rand des Graphems

RÜCKGABEWERTE

`x_pos` x_offset in Pango-Einheiten

34.9 playoutline:IsNull

BEZEICHNUNG

playoutline:IsNull – Überprüft, ob die Layout-Zeile ungültig ist

ÜBERSICHT

```
bool = playoutline:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn die Layout-Zeile `NULL`, d.h. ungültig ist. Wenn Funktionen, die Layoutzeilen zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Mit dieser Funktion können Sie prüfen, ob die Objektzuweisung fehlgeschlagen ist. In diesem Fall ist das Objekt `NULL`.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, falls das Objekt nicht existiert. Mit dieser Funktion können Sie prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn die Layoutzeile `NULL` ist, sonst `False`

34.10 playoutline:Reference

BEZEICHNUNG

playoutline:Reference – Erhöht die Anzahl der Verweise der Pango-Layoutzeilen

ÜBERSICHT

```
playoutline:Reference()
```

BESCHREIBUNG

Erhöht die Anzahl der Verweise auf eine Pango-Layoutzeile um eins.

EINGABEN

Keine

34.11 playoutline:XToIndex

BEZEICHNUNG

playoutline:XToIndex – Konvertiert von x-Offset in Index

ÜBERSICHT

```
inside, index, trailing = playoutline:XToIndex(x_pos)
```

BESCHREIBUNG

Konvertiert den x-Offset in den Byte-Index des entsprechenden Zeichens im Text des Layouts.

Wenn `x_pos` außerhalb der Zeile liegt, verweisen `index` und `trailing` auf die allererste oder allerletzte Position in der Zeile. Diese Bestimmung basiert auf der festgelegten Richtung des Absatzes: Wenn die festgelegte Richtung beispielsweise von rechts nach links verläuft, dann führt eine X-Position rechts von der Zeile (danach) dazu, dass 0 in `index` und `trailing` gespeichert wird. Eine X-Position links von der Zeile führt dazu, dass `index` auf das (logische) letzte Graphem in der Zeile verweist und `trailing` auf die Anzahl der Zeichen in diesem Graphem gesetzt wird. Das Umgekehrte gilt für eine Zeile von links nach rechts.

Diese Funktion gibt `False` zurück, wenn `x_pos` außerhalb der Zeile war, `True`, wenn innerhalb der Zeile

EINGABEN

`x_pos`	der X-Versatz (in Pango-Einheiten) vom linken Rand der Zeile

RÜCKGABEWERTE

`inside`	`False`, wenn `x_pos` außerhalb der Linie lag, `True`, wenn innerhalb
`index`	berechneter Byte-Index für das Graphem, auf das der Benutzer geklickt hat
`trailing`	Ganzzahl, die angibt, an welcher Stelle des Graphems der Benutzer geklickt hat; sie ist entweder Null oder die Anzahl der Zeichen im Graphem; 0 steht für den Anfang des Graphems

35 Pango-Matrix

35.1 pmatrix:Concat

BEZEICHNUNG
pmatrix:Concat – Konkat die Matrix

ÜBERSICHT
`pmatrix:Concat(new_matrix)`

BESCHREIBUNG
Ändert die durch matrix dargestellte Transformation in die Transformation, die sich ergibt, wenn zuerst die durch `new_matrix` gegebene Transformation und dann die ursprüngliche Transformation angewendet wird.

EINGABEN

`new_matrix`
eine Pango-Matrix

35.2 pmatrix:Get

BEZEICHNUNG
pmatrix:Get – Gibt eine affine Matrixtransformation zurück

ÜBERSICHT
`xx, xy, yx, yy, x0, y0 = pmatrix:Get()`

BESCHREIBUNG
Ermittelt die affine Transformation aus der Matrix und gibt sie zurück.

EINGABEN
keine

RÜCKGABEWERTE

`xx`	xx-Komponente der affinen Transformation
`xy`	xy-Komponente der affinen Transformation
`yx`	yx-Komponente der affinen Transformation
`yy`	yy-Komponente der affinen Transformation
`x0`	X-Translationskomponente der affinen Transformation
`y0`	Y-Translationskomponente der affinen Transformation

35.3 pmatrix:GetFontScaleFactor

BEZEICHNUNG

pmatrix:GetFontScaleFactor – Gibt den Skalierungsfaktor der Schriftart zurück

ÜBERSICHT

```
yscale = pmatrix:GetFontScaleFactor()
```

BESCHREIBUNG

Gibt den Skalierungsfaktor einer Matrix für die Höhe der Schriftart zurück.

Das heißt, der Skalierungsfaktor in der Richtung senkrecht zum Vektor, dem die X-Koordinate zugeordnet ist. Wenn auch die Skalierung in der X-Achse benötigt wird, verwenden Sie pmatrix:GetFontScaleFactors().

EINGABEN

Keine

RÜCKGABEWERTE

`yscale` der Skalierungsfaktor der Matrix für die Höhe der Schriftart

35.4 pmatrix:GetFontScaleFactors

BEZEICHNUNG

pmatrix:GetFontScaleFactors – Gibt die Schriftart-Skalierungsfaktoren zurück

ÜBERSICHT

```
xscale, yscale = pmatrix:GetFontScaleFactors()
```

BESCHREIBUNG

Berechnet den Skalierungsfaktor einer Matrix anhand der Breite und Höhe der Schriftart.

Das heißt, `xscale` ist der Skalierungsfaktor in Richtung der X-Achse und `yscale` ist der Skalierungsfaktor in der Richtung senkrecht zum Vektor, dem die X-Koordinate zugeordnet ist.

Beachten Sie, dass die ausgegebenen Zahlen immer nicht-negativ sein werden.

EINGABEN

Keine

RÜCKGABEWERTE

`xscale` Skalierungsfaktor in x-Richtung

`yscale` Skalierungsfaktor senkrecht zur x-Richtung

35.5 pmatrix:Init

BEZEICHNUNG

pmatrix:Init – Initialisiert die Anfangsmatrix

ÜBERSICHT

```
pmatrix:Init(xx, xy, yx, yy, x0, y0)
```

BESCHREIBUNG
Setzt `matrix` als die durch `xx`, `xy`, `yx`, `yy`, `x0`, `y0` gegebene affine Transformation.

EINGABEN

`xx`	xx-Komponente der affinen Transformation
`xy`	xy-Komponente der affinen Transformation
`yx`	yx-Komponente der affinen Transformation
`yy`	yy-Komponente der affinen Transformation
`x0`	X-Translationskomponente der affinen Transformation
`y0`	Y-Translationskomponente der affinen Transformation

35.6 pmatrix:InitIdentity

BEZEICHNUNG
pmatrix:InitIdentity – Initialisiert die Anfangsidentität

ÜBERSICHT
`pmatrix:InitIdentity()`

BESCHREIBUNG
Ändert die Matrix in eine Identitätstransformation.

EINGABEN
Keine

35.7 pmatrix:Rotate

BEZEICHNUNG
pmatrix:Rotate – Dreht die Matrix

ÜBERSICHT
`pmatrix:Rotate(degrees)`

BESCHREIBUNG
Ändert die durch die Matrix dargestellte Transformation in die Transformation, die sich ergibt, wenn man zuerst um `degrees` Grad gegen den Uhrzeigersinn dreht und dann die ursprüngliche Transformation anwendet.

EINGABEN

`degrees`	Grad, um gegen den Uhrzeigersinn zu drehen

35.8 pmatrix:Scale

BEZEICHNUNG
pmatrix:Scale – Skaliert die Matrix

ÜBERSICHT
`pmatrix:Scale(scale_x, scale_y)`

BESCHREIBUNG
Ändert die durch die Matrix dargestellte Transformation in die Transformation, die durch die erste Skalierung um `scale_x` in X-Richtung und `scale_y` in Y-Richtung ergibt, und wendet dann die ursprüngliche Transformation an.

EINGABEN

`scale_x`	Wert, um den in X-Richtung skaliert wird
`scale_y`	Wert, um den in Y-Richtung skaliert wird

35.9 pmatrix:TransformDistance

BEZEICHNUNG
pmatrix:TransformDistance – Transformiert den Abstand

ÜBERSICHT
`tx, ty = pmatrix:TransformDistance(dx, dy)`

BESCHREIBUNG
Transformiert den Abstandsvektor (`dx`,`dy`) durch die Matrix.

Dies ähnelt pmatrix:TransformPoint(), außer dass die Translationskomponenten der Transformation ignoriert werden. Die Berechnung des zurückgegebenen Vektors ist wie folgt:

```
dx2 = dx1 * xx + dy1 * xy;
dy2 = dx1 * yx + dy1 * yy;
```

Affine Transformationen sind positionsinvariant, sodass derselbe Vektor immer in denselben Vektor transformiert wird. Wenn (`x1`,`y1`) in (`x2`,`y2`) transformiert, dann wird (`x1`+§`dx1`,`y1`+§`dy1`) für alle Werte von `x1` und `x2` in (`x1`+§`dx2`,`y1`+§`dy2`) transformiert.

EINGABEN

`dx`	X-Komponente eines Abstandsvektors
`dy`	Y-Komponente eines Abstandsvektors

RÜCKGABEWERTE

`tx`	transformierte X-Komponente eines Abstandsvektors
`ty`	transformierte Y-Komponente eines Abstandsvektors

35.10 pmatrix:TransformPixelRectangle

BEZEICHNUNG

pmatrix:TransformPixelRectangle – Transformiert das Pixelrechteck

ÜBERSICHT

```
rc = pmatrix:TransformPixelRectangle(rect)
```

BESCHREIBUNG

Transformiert zuerst das Rechteck `rect` mithilfe der Matrix und berechnet dann den Begrenzungsrahmen des transformierten Rechtecks. Der Parameter `rect` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind. In der Rückgabetabelle werden diese Felder ebenfalls initialisiert.

Diese Funktion ist beispielsweise nützlich, wenn Sie ein gedrehtes Pango-Layout in einen Bildpuffer zeichnen möchten und wissen möchten, wie groß das Bild sein soll und um wie viel Sie das Layout beim Rendern verschieben sollten.

Für eine bessere Genauigkeit sollten Sie pmatrix:TransformRectangle() für das ursprüngliche Rechteck in Pango-Einheiten verwenden und anschließend mit dem ersten Argument von pango.ExtentsToPixels() in Pixel umrechnen.

EINGABEN

`rect` Begrenzungsrahmen in Pixeln

RÜCKGABEWERTE

`rc` transformiertes Rechteck

35.11 pmatrix:TransformPoint

BEZEICHNUNG

pmatrix:TransformPoint – Transformiert den Punkt

ÜBERSICHT

```
tx, ty = pmatrix:TransformPoint(x, y)
```

BESCHREIBUNG

Transformiert den Punkt (`x`, `y`) durch die Matrix.

EINGABEN

`x` X-Position

`y` Y-Position

RÜCKGABEWERTE

`tx` transformierte X-Position

`ty` transformierte Y-Position

35.12 pmatrix:TransformRectangle

BEZEICHNUNG

pmatrix:TransformRectangle – Transformiert das Rechteck

ÜBERSICHT

```
rc = pmatrix:TransformRectangle(rect)
```

BESCHREIBUNG

Transformiert zunächst das Rechteck `rect` mithilfe der Matrix und berechnet dann den Begrenzungsrahmen des transformierten Rechtecks. Der Parameter `rect` muss eine Tabelle sein, in der die Felder `x`, `y`, `width` und `height` initialisiert sind. In der Rückgabetabelle werden diese Felder ebenfalls initialisiert.

Diese Funktion ist beispielsweise nützlich, wenn Sie ein gedrehtes Pango-Layout in einen Bildpuffer zeichnen wollen und wissen möchten, wie groß das Bild sein soll und wie stark Sie das Layout beim Rendern verschieben müssen.

Wenn Sie ein Rechteck in Pixel haben, verwenden Sie pmatrix:TransformPixelRectangle().

Wenn Sie das Rechteck in Pango-Einheiten haben und es in einen transformierten Pixel-Begrenzungsrahmen konvertieren möchten, ist es genauer, es zuerst zu transformieren (mit dieser Funktion) und das Ergebnis an pango.ExtentsToPixels(), das erste Argument, für ein abgerundetes Rechteck zu übergeben. Es gibt jedoch triftige Gründe, die dafür sprechen, erst in Pixel zu konvertieren und dann zu transformieren, beispielsweise wenn die transformierten Koordinaten in Pango-Einheiten überlaufen (z.B. bei einer großen Matrix-Translation).

EINGABEN

`rect`	Begrenzungsrahmen in Pango-Einheiten

RÜCKGABEWERTE

`rc`	transformiertes Rechteck

35.13 pmatrix:Translate

BEZEICHNUNG

pmatrix:Translate – Ändert die Matrix

ÜBERSICHT

```
pmatrix:Translate(tx, ty)
```

BESCHREIBUNG

Ändert die durch die Matrix dargestellte Transformation in die Transformation, die zuerst durch die Translation durch (`tx`, `ty`) und dann die ursprüngliche Transformation anwendet.

EINGABEN

`tx`	Wert für die Verschiebung in X-Richtung
`ty`	Wert für die Verschiebung in Y-Richtung

36 Pango-Tabulator-Array

36.1 ptabarray:Copy

BEZEICHNUNG
ptabarray:Copy – Kopiert ein Tabulator-Array

ÜBERSICHT
```
tabs = ptabarray:Copy()
```

BESCHREIBUNG
Kopiert einen Pango-Tabulator-Array.

Diese Funktion gibt das neu zugewiesene Pango-Tabulator-Array zurück, das mit ptabarray:Free() wieder gelöscht werden sollte.

EINGABEN
Keine

RÜCKGABEWERTE

`tabs`	das neu zugewiesene Pango-Tabulator-Array

36.2 ptabarray:Free

BEZEICHNUNG
ptabarray:Free – Löscht ein Tabulator-Array

ÜBERSICHT
```
ptabarray:Free()
```

BESCHREIBUNG
Löscht ein Tabulator-Array und gibt die zugehörigen Ressourcen frei.

EINGABEN
Keine

36.3 ptabarray:GetPositionsInPixels

BEZEICHNUNG
ptabarray:GetPositionsInPixels – Überprüft, wie die Position angebeben ist

ÜBERSICHT
```
ok = ptabarray:GetPositionsInPixels()
```

BESCHREIBUNG
Gibt `True` zurück, wenn die Tabulatorpositionen in Pixeln angegeben, `False`, wenn sie in Pango-Einheiten angegeben sind.

EINGABEN
Keine

RÜCKGABEWERTE

`ok` ob die Positionen in Pixeln oder Pango-Einheiten angegeben sind

36.4 ptabarray:GetSize

BEZEICHNUNG

ptabarray:GetSize – Gibt die Anzahl der Tabstopps zurück

ÜBERSICHT

```
n = ptabarray:GetSize()
```

BESCHREIBUNG

Ermittelt die Anzahl der Tabstopps im Tabulator-Array.

EINGABEN

Keine

RÜCKGABEWERTE

`n` die Anzahl der Tabstopps im Array

36.5 ptabarray:GetTab

BEZEICHNUNG

ptabarray:GetTab – Gibt die Ausrichtung und Position eines Tabstopps zurück

ÜBERSICHT

```
alignment, location = ptabarray:GetTab(tab_index)
```

BESCHREIBUNG

Ermittelt die Ausrichtung und Position eines Tabstopps.

EINGABEN

`tab_index`
der Tabstopp-Index

RÜCKGABEWERTE

`alignment`
die Ausrichtung des Tabstopps

`location` die Position des Tabstopps

36.6 ptabarray:GetTabs

BEZEICHNUNG

ptabarray:GetTabs – Gibt die Tabulatoren zurück

ÜBERSICHT

```
tabs = ptabarray:GetTabs()
```

BESCHREIBUNG

Liefert eine Tabelle mit den Ausrichtungen und Positionen aller Tabstopps. Die zurückgegebene Tabelle enthält eine Untertabelle für jeden Tabstopp. In jeder Untertabelle sind die Felder `Align` und `Pos` auf die jeweilige Ausrichtung und Position des Tabulatorstopps initialisiert.

EINGABEN

Keine

RÜCKGABEWERTE

`tabs` Tabelle mit den Ausrichtungen und Positionen aller Tabstopps

36.7 ptabarray:IsNull

BEZEICHNUNG

ptabarray:IsNull – Prüft, ob das Tabulator-Array ungültig ist

ÜBERSICHT

```
bool = ptabarray:IsNull()
```

BESCHREIBUNG

Gibt `True` zurück, wenn das Tabulator-Array `NULL` ist, d.h. ungültig. Wenn Funktionen, die Objekte zuweisen, fehlschlagen, geben sie möglicherweise keinen Fehler aus, sondern setzen das Objekt einfach auf `NULL`. Sie können diese Funktion verwenden, um zu prüfen, ob die Objektzuweisung fehlgeschlagen ist.

Außerdem können bestimmte Get-Funktionen ein NULL-Objekt zurückgeben, wenn das Objekt nicht existiert. Sie können diese Funktion verwenden, um zu prüfen, ob eine Get-Funktion ein NULL-Handle zurückgegeben hat.

EINGABEN

Keine

RÜCKGABEWERTE

`bool` `True`, wenn das Tabulator-Array `NULL` ist, sonst `False`

36.8 ptabarray:Resize

BEZEICHNUNG

ptabarray:Resize – Ändert die Größe des Tabulator-Array

ÜBERSICHT

```
ptabarray:Resize(new_size)
```

BESCHREIBUNG

Ändert die Größe eines Tabulator-Arrays auf die durch `new_size` angegebene Größe.

Sie müssen anschließend alle Tabulator initialisieren, die als Ergebnis der Erweiterung des Arrays hinzugefügt wurden.

EINGABEN

`new_size` neue Größe des Arrays

36.9 ptabarray:SetTab

BEZEICHNUNG

ptabarray:SetTab – Setzt den Tabulatorstopp

ÜBERSICHT

```
ptabarray:SetTab(tab_index, alignment, location)
```

BESCHREIBUNG

Legt die Ausrichtung und Position eines Tabulatorstopps fest. Siehe pango.TabArrayWithPositions für eine Liste der unterstützten Ausrichtungen für den Parameter `alignment`.

EINGABEN

`tab_index`
: der Index eines Tabulatorstopps

`alignment`
: Tabulatorausrichtung

`location` Tabulatorposition in Pango-Einheiten

Anhang A Licenses

A.1 LGPL license

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent

license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any

derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the

Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

A.2 HarfBuzz license

HarfBuzz is licensed under the so-called "Old MIT" license. Details follow. For parts of HarfBuzz that are licensed under different licenses see individual files names COPYING in subdirectories where applicable.

A.3 Expat license

distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

A.4 Fontconfig license

Copyright (C) 2000,2001,2002,2003,2004,2006,2007 Keith Packard
Copyright 2005 Patrick Lam
Copyright 2007 Dwayne Bailey and Translate.org.za
Copyright 2009 Roozbeh Pournader
Copyright 2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020 Red Hat, Inc.
Copyright 2008 Danilo Aegan
Copyright 2012 Google, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of the author(s) not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. The authors make no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THE AUTHOR(S) DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL THE AUTHOR(S) BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

A.5 Pixman license

The following is the MIT license, agreed upon by most contributors. Copyright holders of new code should use this license statement where possible. They may also add themselves to the list below.

Copyright 1987, 1988, 1989, 1998 The Open Group
Copyright 1987, 1988, 1989 Digital Equipment Corporation
Copyright 1999, 2004, 2008 Keith Packard

A.6 Libxml2 license

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Index

C

P