OpenType 字体功能

项目 • 2022/10/20

本主题概述了 Windows Presentation Foundation (WPF) 中 OpenType 字体技术的一些主要功能。

OpenType 字体格式

OpenType 字体格式是 TrueType® 字体格式的扩展，增加了对 PostScript 字体数据的支持。 OpenType 字体格式由 Microsoft 和 Adobe Corporation 联合开发。 无论字体包含 TrueType 边框还是 CFF (PostScript) 边框，OpenType 字体和支持 OpenType 字体的操作系统服务都向用户提供一种简单的字体安装和使用方式。

OpenType 字体格式解决了以下开发人员挑战：

- 更广泛的多平台支持。
- 更出色的国际字符集支持。
- 更优的字体数据保护。
- 更小的文件大小，让字体发布更加高效。
- 更广泛的高级版式控件支持。

> **备注**
>
> Windows SDK 包含一组可用于 Windows Presentation Foundation (WPF) 应用程序的示例 OpenType 字体。 这些字体提供本主题余下部分所述的大多数功能。 有关详细信息，请参阅**示例 OpenType 字体包**。

有关 OpenType 字体格式的详细信息，请参阅 OpenType 规范。

高级版式扩展

高级版式表格（OpenType 布局表格）扩展了具有 TrueType 或 CFF 边框的字体的功能。 OpenType 布局字体包含一些其他信息，可扩展字体功能以支持高质量国际版式。 大多数 OpenType 字体仅体现全部可用 OpenType 功能的一部分。 OpenType 字体提供以下功能。

- 字符与字形之间的丰富映射，可支持连字、定位格式、备用项以及其他字体替换功能。
- 支持二维定位和字形附加。
- 字体中包含的显式脚本和语言信息，使文本处理应用程序可相应调整其行为。

在 OpenType 规范的"字体文件表格" 部分中对 OpenType 布局表格进行了更详细的介绍。

此概述的其余部分介绍了一些直观有趣的 OpenType 功能的广度和灵活性（这些功能由 Typography 对象的属性公开）。 有关此对象的详细信息，请参阅版式类。

变量

变量用于呈现不同的版式风格，例如上标和下标。

上标和下标

通过 Variants 属性可以设置 OpenType 字体的上标和下标值。

以下文本显示 Palatino Linotype 字体的上标。

2^3 14^{th}

以下标记示例演示如何使用 Typography 对象的属性定义 Palatino Linotype 字体的上标。

XAML

```
<Paragraph FontFamily="Palatino Linotype">
  2<Run Typography.Variants="Superscript">3</Run>
  14<Run Typography.Variants="Superscript">th</Run>
</Paragraph>
```

以下文本显示 Palatino Linotype 字体的下标。

H_2O $Footnote_4$

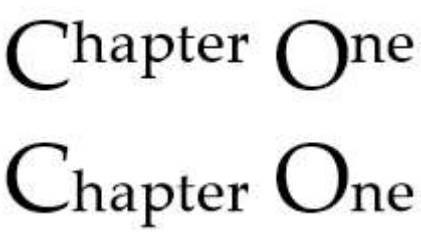
以下标记示例演示如何使用 Typography 对象的属性定义 Palatino Linotype 字体的下标。

XAML

```
<Paragraph FontFamily="Palatino Linotype">
  H<Run Typography.Variants="Subscript">2</Run>O
  Footnote<Run Typography.Variants="Subscript">4</Run>
</Paragraph>
```

上标和下标的修饰用法

也可使用上标和下标来营造混合大小写文本的修饰效果。 以下文本显示 Palatino Linotype 字体的上标和下标文本。 注意，大写字母不受影响。



以下标记示例演示如何使用 Typography 对象的属性定义字体的上标和下标。

XAML

```
<Paragraph FontFamily="Palatino Linotype"
Typography.Variants="Superscript">
  Chapter One
</Paragraph>
<Paragraph FontFamily="Palatino Linotype" Typography.Variants="Subscript">
  Chapter One
</Paragraph>
```

大写字母

大写字母是一组以大写样式字形呈现文本的版式形式。 通常情况下，当以全大写呈现文本时，字母之间的间距可能看起来很小，字母的权重和比例看起来会很大。 OpenType 支持多种大写字母的样式格式，包括小体大写字母、小号大写字母、标题和大写字母间距。 通过这些样式格式可控制大写字母的外观。

以下文本显示 Pescadero 字体的标准大写字母，其后接样式为“SmallCaps”和“AllSmallCaps”的字母。 本例中，对所有三个单词均使用相同的字体大小。

以下标记示例演示如何使用 Typography 对象的属性定义 Pescadero 字体的大写字母。 使用“SmallCaps”格式时会忽略任何前导大写字母。

XAML

```
<Paragraph FontFamily="Pescadero" FontSize="48">
  <Run>CAPITALS</Run>
  <Run Typography.Capitals="SmallCaps">Capitals</Run>
  <Run Typography.Capitals="AllSmallCaps">Capitals</Run>
</Paragraph>
```

标题大写字母

标题大写字母权重和比例更小，外观比普通大写字母更加雅致。 标题大写字母通常用于作为标题的大号字体中。 以下文本显示 Pescadero 字体的普通大写字母和标题大写字母。 请注意第二行文本的宽度更窄。

以下标记示例演示如何使用 Typography 对象的属性定义 Pescadero 字体的标题大写字母。

XAML

```
<Paragraph FontFamily="Pescadero">
  <Run Typography.Capitals="Titling">chapter one</Run>
</Paragraph>
```

大写字母间距

大写字母间距功能让你可以在使用全大写字母文本时提供更宽的间距。 大写字母通常设计为与小写字母混合使用。 大写字母和小写字母之间看起来比较美观的间距在使用全大写字母时可能会显得过紧。 以下文本显示 Pescadero 字体的普通间距和大写字母间距。

以下标记示例演示如何使用 Typography 对象的属性定义 Pescadero 字体的大写字母间距。

XAML

```
<Paragraph FontFamily="Pescadero">
  <Run Typography.CapitalSpacing="True">CHAPTER ONE</Run>
</Paragraph>
```

连字

连子是为使文本更具可读性或更加美观而由两个或更多字形形成的一个单一字形。
OpenType 字体支持四种类型的连字：

- **标准连字。** 旨在增强可读性。 标准连字包括“fi”、“fl”和“ff”。
- **上下文连字。** 旨在通过在组成连字的字符之间提供更好的联结行为来增强可读性。
- **自由连字。** 重在修饰性，并非专为可读性而设计。
- **历史连字。** 重在历史性，并非专为可读性而设计。

以下文本显示 Pericles 字体的标准连字字形。

以下标记示例演示如何使用 Typography 对象的属性定义 Pericles 字体的标准连字字形。

XAML

```
<Paragraph FontFamily="Pericles" Typography.StandardLigatures="True">
  <Run Typography.StylisticAlternates="1">FI</Run>
  <Run Typography.StylisticAlternates="1">FL</Run>
  <Run Typography.StylisticAlternates="1">TH</Run>
  <Run Typography.StylisticAlternates="1">TT</Run>
  <Run Typography.StylisticAlternates="1">TV</Run>
  <Run Typography.StylisticAlternates="1">TW</Run>
  <Run Typography.StylisticAlternates="1">TY</Run>
  <Run Typography.StylisticAlternates="1">VT</Run>
  <Run Typography.StylisticAlternates="1">WT</Run>
  <Run Typography.StylisticAlternates="1">YT</Run>
</Paragraph>
```

以下文本显示 Pericles 字体的自由连字字形。

以下标记示例演示如何使用 Typography 对象的属性定义 Pericles 字体的自由连字字形。

XAML

```
<Paragraph FontFamily="Pericles" Typography.DiscretionaryLigatures="True">
  <Run Typography.StylisticAlternates="1">CO</Run>
  <Run Typography.StylisticAlternates="1">LA</Run>
  <Run Typography.StylisticAlternates="1">LE</Run>
  <Run Typography.StylisticAlternates="1">LI</Run>
  <Run Typography.StylisticAlternates="1">LL</Run>
  <Run Typography.StylisticAlternates="1">LO</Run>
  <Run Typography.StylisticAlternates="1">LU</Run>
</Paragraph>
```

默认情况下，Windows Presentation Foundation (WPF) 中的 OpenType 字体启用标准连字。 例如，如果使用 Palatino Linotype 字体，则标准连字“fi”、“ff”和“fl”显示为组合字符字形。 请注意，每个标准连字的字符对之间彼此相连。

但是，可禁用标准连字功能，从而使“ff”等标准连字显示为两个单独的字形，而不显示为一个组合字符字形。

以下标记示例演示如何使用 Typography 对象的属性禁用 Palatino Linotype 字体的标准连字字形。

XAML

```
<!-- Set standard ligatures to false in order to disable feature. -->
<Paragraph Typography.StandardLigatures="False" FontFamily="Palatino
Linotype" FontSize="72">
  fi ff fl
</Paragraph>
```

花体

花体是使用精美修饰的装饰性字形，通常与书法相关。 以下文本显示 Pescadero 字体的标准和花体字形。

标准字形文本

花体通常用作事件公告等简短文章中的修饰元素。 以下文本使用花体强调事件名称的大写字母。

文本

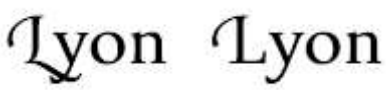
以下标记示例演示如何使用 Typography 对象的属性定义字体花体。

XAML

```
<Paragraph FontFamily="Pescadero" TextBlock.TextAlignment="Center">
  Wishing you a<LineBreak/>
  <Run Typography.StandardSwashes="1" FontSize="36">Happy New Year!</Run>
</Paragraph>
```

连接形式花体

花体字形的某些组合可能导致文本外观欠佳，例如相邻字母的下行处出现重叠。 通过连接形式花体，可使用能生成更佳外观的替代花体字形。 以下文本显示同一单词应用连接形式花体前后的外观。



以下标记示例演示如何使用 Typography 对象的属性定义 Pescadero 字体的连接形式花体。

XAML

```
<Paragraph FontFamily="Pescadero" Typography.StandardSwashes="1">
  Lyon <Run Typography.ContextualSwashes="1">L</Run>yon
</Paragraph>
```

备用项

备用项是可替代标准字形的字形。 OpenType 字体（例如以下示例中使用的 Pericles 字体）可包含用于塑造不同文本外观的备用字形。 以下文本显示 Pericles 字体的标准字形。

ANCIENT GREEK MYTHOLOGY

Pericles OpenType 字体包含其他字形，可为标准自行集提供样式备用项。 以下文本显示样式备用字形。


的文本

以下标记示例演示如何使用 Typography 对象的属性定义 Pericles 字体的样式备用字形。

XAML

```
<Paragraph FontFamily="Pericles">
  <Run Typography.StylisticAlternates="1">A</Run>NCIENT
  GR<Run Typography.StylisticAlternates="1">EE</Run>K
  MYTH<Run Typography.StylisticAlternates="1">O</Run>LOGY
</Paragraph>
```

以下文本显示 Pericles 字体的几种其他样式备用字形。

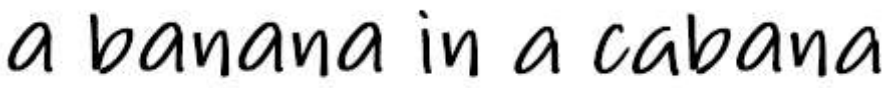
以下标记示例演示如何定义其他样式备用字形。

XAML

```
<Paragraph FontFamily="Pericles">
  <Run Typography.StylisticAlternates="1">A</Run>
  <Run Typography.StylisticAlternates="2">A</Run>
  <Run Typography.StylisticAlternates="3">A</Run>
  <Run Typography.StylisticAlternates="1">C</Run>
  <Run Typography.StylisticAlternates="1">E</Run>
  <Run Typography.StylisticAlternates="1">G</Run>
  <Run Typography.StylisticAlternates="1">O</Run>
  <Run Typography.StylisticAlternates="1">Q</Run>
  <Run Typography.StylisticAlternates="1">R</Run>
  <Run Typography.StylisticAlternates="2">R</Run>
  <Run Typography.StylisticAlternates="1">S</Run>
  <Run Typography.StylisticAlternates="1">Y</Run>
</Paragraph>
```

随机备用连接形式

随机备用连接形式为单个字符提供多种备用字形。 实现脚本类型字体时，此功能可通过使用一组随机选择的外观稍有差异的字形来模拟手写内容。 以下文本使用 Lindsey 字体的随机备用连接形式。 请注意字母“a”外观稍有变化



以下标记示例演示如何使用 Typography 对象的属性定义 Lindsey 字体的随机备用连接形式。

XAML

```
<TextBlock FontFamily="Lindsey">
  <Run Typography.ContextualAlternates="True">
    a banana in a cabana
  </Run>
</TextBlock>
```

历史形式

历史形式指过去常见的版式约定。 以下文本使用 Palatino Linotype 字体的一种历史字形形式显示短语“Boston, Massachusetts”。

以下标记示例演示如何使用 Typography 对象的属性定义 Palatino Linotype 字体的历史形式。

XAML

```
<Paragraph FontFamily="Palatino Linotype">
  <Run Typography.HistoricalForms="True">Boston, Massachusetts</Run>
</Paragraph>
```

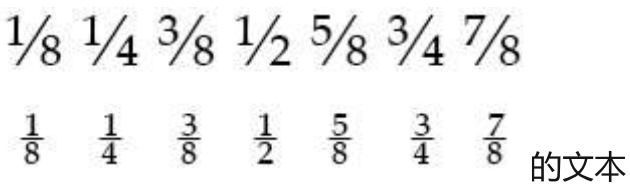
数字样式

OpenType 字体支持多种可用于文本中数值的功能。

分数

OpenType 字体支持多种分数样式，包括横式分数和竖式分数。

以下文本显示 Palatino Linotype 字体的分数样式。



的文本

以下标记示例演示如何使用 Typography 对象的属性定义 Palatino Linotype 字体的分数样式。

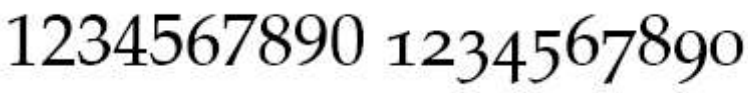
XAML

```
<Paragraph FontFamily="Palatino Linotype" Typography.Fraction="Slashed">
  1/8 1/4 3/8 1/2 5/8 3/4 7/8
</Paragraph>
<Paragraph FontFamily="Palatino Linotype" Typography.Fraction="Stacked">
  1/8 1/4 3/8 1/2 5/8 3/4 7/8
</Paragraph>
```

旧样式数字

OpenType 字体支持旧样式数字格式。 此格式对于显示不再是标准样式的数字非常有用。 以下文本以 Palatino Linotype 字体的标准和旧样式数字格式显示 18 世纪日期。

以下文本显示 Palatino Linotype 字体的标准数字，后跟旧样式数字。



以下标记示例演示如何使用 Typography 对象的属性定义 Palatino Linotype 字体的旧样式数字。

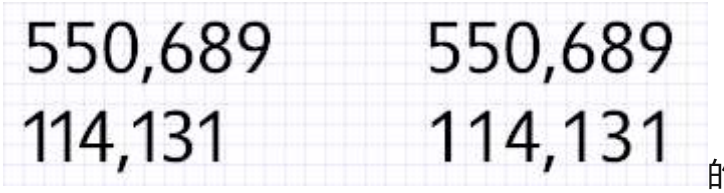
XAML

```
<Paragraph FontFamily="Palatino Linotype">
  <Run Typography.NumeralStyle="Normal">1234567890</Run>
  <Run Typography.NumeralStyle="OldStyle">1234567890</Run>
</Paragraph>
```

比例数字和表格式数字

OpenType 字体支持比例数字和表格式数字功能，可在使用数字时控制宽度对齐。 比例数字将每个数字视为具有不同的宽度—“1”窄于“5”。 表格式数字被视为宽度相等的数字，因此它们可垂直对齐，从而增强财务类型信息的可读性。

以下文本使用 Miramonte 字体显示第一列中的两个表格式数字。 请注意数字“5”和“1”之间的宽度差异。 第二列显示相同的两个数值，并通过使用表格式数字功能调整其宽度。



的文本

以下标记示例演示如何使用 Typography 对象的属性定义 Miramonte 字体的比例数字和表格式数字。

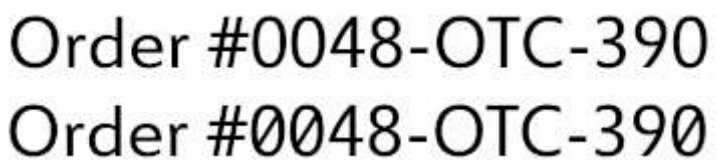
XAML

```
<TextBlock FontFamily="Miramonte">
  <Run Typography.NumeralAlignment="Proportional">114,131</Run>
</TextBlock>
<TextBlock FontFamily="Miramonte">
  <Run Typography.NumeralAlignment="Tabular">114,131</Run>
</TextBlock>
```

斜线零

OpenType 字体支持斜线零数字格式来强调字母“O”和数字“0”之间的差异。 斜线零数字通常用于财务和商务信息中的标识符。

以下文本显示使用 Miramonte 字体的订单标识符。 第一行使用标准数字。 第二行使用斜线零数字，以便更易于与大写字母“O”进行区分。



以下标记示例演示如何使用 Typography 对象的属性定义 Miramonte 字体的斜线零数字。

XAML

```
<Paragraph FontFamily="Miramonte">
  <Run>Order #0048-OTC-390</Run>
  <LineBreak/>
```

```
  <Run Typography.SlashedZero="True">Order #0048-OTC-390</Run>
</Paragraph>
```

版式类

Typography 对象公开 OpenType 字体支持的功能集。 通过在标记中设置 Typography 的属性，可轻松创作使用 OpenType 功能的文档。

以下文本显示 Pescadero 字体的标准大写字母，其后接样式为“SmallCaps”和“AllSmallCaps”的字母。 本例中，对所有三个单词均使用相同的字体大小。

以下标记示例演示如何使用 Typography 对象的属性定义 Pescadero 字体的大写字母。 使用“SmallCaps”格式时会忽略任何前导大写字母。

XAML

```
<Paragraph FontFamily="Pescadero" FontSize="48">
  <Run>CAPITALS</Run>
  <Run Typography.Capitals="SmallCaps">Capitals</Run>
  <Run Typography.Capitals="AllSmallCaps">Capitals</Run>
</Paragraph>
```

以下代码示例完成与先前的标记事例相同的任务。

C#

```
MyParagraph.FontFamily = new FontFamily("Pescadero");
MyParagraph.FontSize = 48;

Run run_1 = new Run("CAPITALS ");
MyParagraph.Inlines.Add(run_1);

Run run_2 = new Run("Capitals ");
run_2.Typography.Capitals = FontCapitals.SmallCaps;
MyParagraph.Inlines.Add(run_2);

Run run_3 = new Run("Capitals");
run_3.Typography.Capitals = FontCapitals.AllSmallCaps;
MyParagraph.Inlines.Add(run_3);

MyParagraph.Inlines.Add(new LineBreak());
```

版式类属性

下表列出了 Typography 对象的属性、值和默认设置。

properties	值	默认值
AnnotationAlternates	数值 - 字节	0
Capitals	AllPetiteCaps | AllSmallCaps | Normal | PetiteCaps | SmallCaps | Titling | Unicase	FontCapitals.Normal
CapitalSpacing	Boolean	`false`
CaseSensitiveForms	Boolean	`false`
ContextualAlternates	Boolean	`true`
ContextualLigatures	Boolean	`true`
ContextualSwashes	数值 - 字节	0
DiscretionaryLigatures	Boolean	`false`
EastAsianExpertForms	Boolean	`false`
EastAsianLanguage	HojoKanji | Jis04 | Jis78 | Jis83 | Jis90 | NlcKanji | Normal | Simplified | Traditional | TraditionalNames	FontEastAsianLanguage.Normal
EastAsianWidths	Full | Half | Normal | Proportional | Quarter | Third	FontEastAsianWidths.Normal
Fraction	Normal | Slashed | Stacked	FontFraction.Normal
HistoricalForms	Boolean	`false`
HistoricalLigatures	Boolean	`false`
Kerning	Boolean	`true`
MathematicalGreek	Boolean	`false`
NumeralAlignment	Normal | Proportional | Tabular	FontNumeralAlignment.Normal
NumeralStyle	Boolean	FontNumeralStyle.Normal
SlashedZero	Boolean	`false`
StandardLigatures	Boolean	`true`
StandardSwashes	数值 - 字节	0

properties	值	默认值
StylisticAlternates	数值 - 字节	0
StylisticSet1	Boolean	`false`
StylisticSet2	Boolean	`false`
StylisticSet3	Boolean	`false`
StylisticSet4	Boolean	`false`
StylisticSet5	Boolean	`false`
StylisticSet6	Boolean	`false`
StylisticSet7	Boolean	`false`
StylisticSet8	Boolean	`false`
StylisticSet9	Boolean	`false`
StylisticSet10	Boolean	`false`
StylisticSet11	Boolean	`false`
StylisticSet12	Boolean	`false`
StylisticSet13	Boolean	`false`
StylisticSet14	Boolean	`false`
StylisticSet15	Boolean	`false`
StylisticSet16	Boolean	`false`
StylisticSet17	Boolean	`false`
StylisticSet18	Boolean	`false`
StylisticSet19	Boolean	`false`
StylisticSet20	Boolean	`false`
Variants	Inferior | Normal | Ordinal | Ruby | Subscript | Superscript	FontVariants.Normal

另请参阅

- Typography
- OpenType 规范

- WPF 中的版式
- 示例 OpenType 字体包
- 将字体与应用程序一起打包