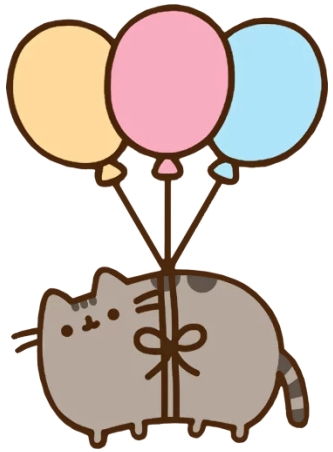
数字文本渲染 101

喵喵



$$\text{DNA} \xrightarrow{\text{转录}} \text{RNA} \xrightarrow{\text{翻译}} \text{Protein}$$

$$\text{Codepoint} \xrightarrow{\text{Shaping}} \text{Glyph} \xrightarrow{\text{Rasterization}} \text{Pixels}$$

$$\text{Codepoint} \xrightarrow{\text{Shaping}} \text{Glyph} \xrightarrow{\text{Rasterization}} \text{Pixels}$$

- 怎么生成一个字体子集
-
-

$$\text{Codepoint} \xrightarrow{\text{Shaping}} \text{Glyph} \xrightarrow{\text{Rasterization}} \text{Pixels}$$

- 怎么生成一个字体子集
- “为什么浏览器这么慢！”
-

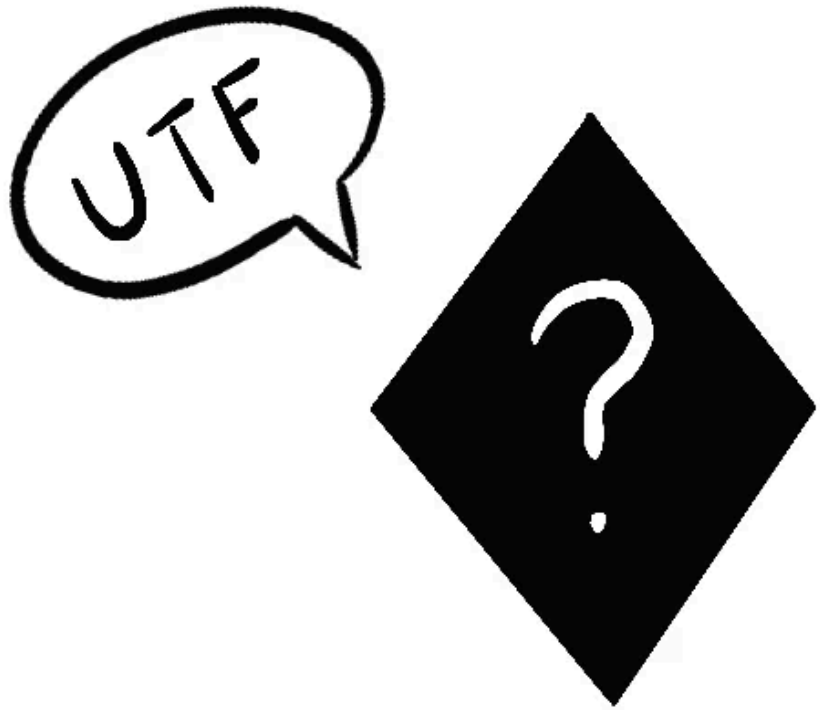
$$\text{Codepoint} \xrightarrow{\text{Shaping}} \text{Glyph} \xrightarrow{\text{Rasterization}} \text{Pixels}$$

- 怎么生成一个字体子集
- “为什么浏览器这么慢！”
- 

文字存储/编码

当然是存储在硬盘/内存/NFS mount/云/纸上....

当然是存储在硬盘/内存/NFS mount/云/纸上....



Unicode gives us 👩🏽‍👩🏻‍👦🏻

whatwg: now the mandatory encoding for all things

Unicode gives us 👩🏽‍👩🏻‍👦🏻

whatwg: now the mandatory encoding for all things

“字符” → 码位 (Codepoint)

Unicode gives us 👪🍼

whatwg: now the mandatory encoding for all things

“字符” → 码位 (Codepoint)

注意: 字符集（码位分配）和编码有区别!

- UTF-8
- UTF-16LE/BE & UCS-2
- UTF-32
- *Other* rounding errors (GB.*, BIG5, etc.)

Unicode gives us 👩🏽‍👩🏻‍👧🏻

whatwg: now the mandatory encoding for all things

“字符” → 码位 (Codepoint)

注意: 字符集（码位分配）和编码有区别!

- **UTF-8 (> 99%)**
- UTF-16LE/BE & UCS-2
- UTF-32
- *Other* rounding errors (GB.*, BIG5, etc.)

UTF-8

UTF-8

Consider UTF-16/UCS-2:

- ASCII-incompatible!
- 需要区别 LE & BE
- 支持的 Plane 比较少（可用比特比较少）
- 导致 `0xD800` - `0xDFFF` 无法使用

UTF-8

Consider UTF-16/UCS-2:

- ASCII-incompatible!
- 需要区别 LE & BE
- 支持的 Plane 比较少（可用比特比较少）
- 导致 `0xD800` - `0xDFFF` 无法使用

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0yyyzzzz			
U+0080	U+07FF	110xxxyy	10yyzzzz		
U+0800	U+FFFF	1110wwww	10xxxxyy	10yyzzzz	
U+010000	U+10FFFF	11110uvv	10vvwwww	10xxxxyy	10yyzzzz

ZWJ sequences

Array.from() =

```
[
  '👩', '', '',
  '👩', '', '',
  '👧', '', '',
  '👩', ''
]
```

ZWJ sequences

Array.from() =

[
'', '', '',
'', '', '',
'', '', '',
'', ''
]

- UTF-8 ⇒ Codepoints
- Codepoints 可能每个对应多个字符，不到一个字符，或者其他情况。

Shaping

Shaping

将文字 + 字体转换为排版与字形的过程

Wait a minute…

Wait a minute...

为什么已经开始讲字体和字形了?

It turns out...

字体需要考虑字符集!

It turns out...

字体需要考虑字符集！

Platform ID	Encoding ID	Description
3	0	Symbol
3	1	Unicode BMP
3	2	ShiftJIS
3	3	PRC
3	4	Big5
3	5	Wansung
3	6	Johab

The CMAP table

Codepoint $\rightarrow$ Glyph ID (16-bit)

The CMAP table

Codepoint → Glyph ID (16-bit)

Segmented coverage (format 12):

```
struct Group {
  start: u32,
  end: u32,
  glyphStart: u32,
}
type Subtable13 = Vec<Group>;
```

The GLYF table

$$\text{Glyph ID} \xrightarrow{\text{LOCA}} \text{Glyph Definition}$$

得到：一系列 Bezier curve 控制点

The GLYF table

$$\text{Glyph ID} \xrightarrow{\text{LOCA}} \text{Glyph Definition}$$

得到：一系列 Bezier curve 控制点

Learn / Microsoft Typography /

The TrueType Instruction Set

Article • 05/30/2024 • 5 contributors

Feedback

In this article

TrueType provides instructions for each of the following tasks and a set of general-purpose instructions. This chapter describes the TrueType instruction set. Instruction descriptions are organized by category based on their function.

- Pushing data onto the interpreter stack

[1] https://learn.microsoft.com/en-us/typography/opentype/spec/tt_instructions

2

[2] https://learn.microsoft.com/en-us/typography/opentype/spec/tt_instructions

What about…

What about…

- Ligatures
-
-
-

What about…

- Ligatures
- Kerning
-
-

What about…

- Ligatures
- Kerning
- Layout
-

What about…

- Ligatures
- Kerning
- Layout
- Variable font

Figure 1: Wrong ~~warp~~ wrap

Layout

每个字符有一个 Horizontal Advance 表示下一个字符应该在横向前进多少。

e.g. 对于空格，有 Horizontal advacement，但是 Glyph Data 里面没有任何点。

Layout

每个字符有一个 Horizontal Advance 表示下一个字符应该在横向前进多少。

e.g. 对于空格，有 Horizontal advacement，但是 Glyph Data 里面没有任何点。

- hmtx
- phantom points

What about line-breaks

UAX[3] 14: Unicode Line Breaking Algorithm

[3] Unicode® Standard Annex

What about line-breaks

UAX[4] 14: Unicode Line Breaking Algorithm

- Mandatory line breaks
- Available line breaks

[4] Unicode® Standard Annex

- UAX 29: Unicode Text Segmentation
- UAX 15: Unicode Normalization Forms

Kerning & Ligatures

最开始：KERN table

Kerning & Ligatures

最开始：KERN table

Post-CFF2:

- GPOS
-

Kerning & Ligatures

最开始：KERN table

Post-CFF2:

- GPOS
- GSUB

Kerning & Ligatures

最开始：KERN table

Post-CFF2:

- GPOS
- GSUB

Variations…

There is a GVAR table…

Variations…

There is a GVAR table…

OpenType variations	supported: outline variation data is stored in the 'gvar' table (OFF: 7.3.4); hint variation data is stored in the 'cvar' table (OFF: 7.3.2)	not supported	supported: variation data for outlines and hints is stored within the CFF2 table, partially using common formats for variation data also used in other tables, and partially interleaved within the CharString data for individual glyph descriptions

Variations…

There is a GVAR table…

Consideration	glyf	CFF	CFF2
OpenType variations	supported: outline variation data is stored in the 'gvar' table (OFF: 7.3.4); hint variation data is stored in the 'cvar' table (OFF: 7.3.2)	not supported	supported: variation data for outlines and hints is stored within the CFF2 table, partially using common formats for variation data also used in other tables, and partially interleaved within the CharString data for individual glyph descriptions

Rendering

Vector rendering 101

Vector rendering 101

```
for pixel in screen {
  pixel.color = if path.contains(pixel) {
    black
  } else {
    white
  };
}
```

Vector rendering 101

```
for pixel in screen
  pixel.color = contains(pixel) {
    black
  } else {
    white
  };
}
```

Vector rendering 101

```
for pixel in screen {
```

关于《刺客信条-幻景》

```
    write
  };
}
```

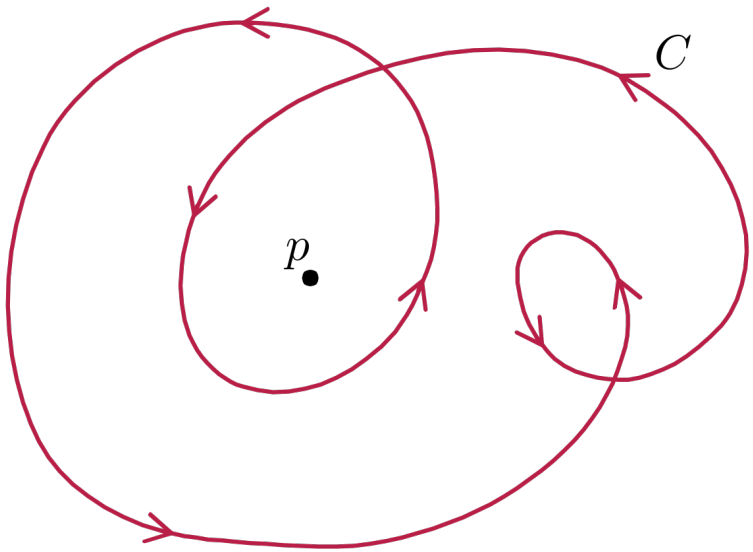
隆信彩

Vector rendering 102

Path-winding number

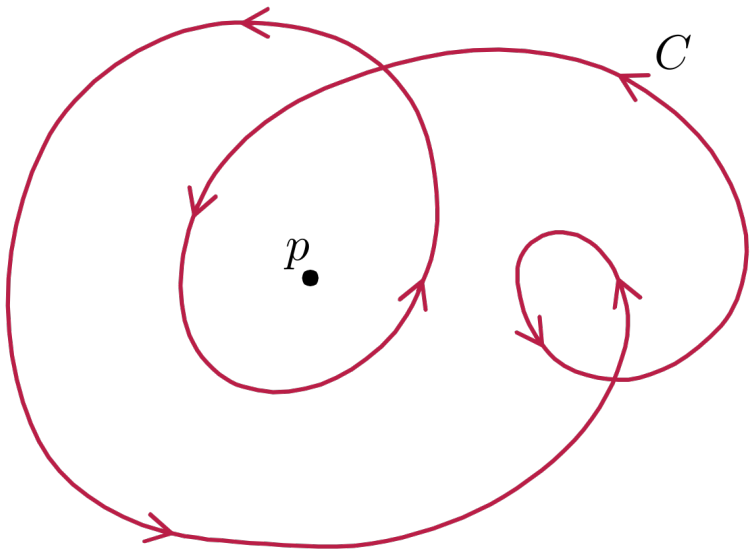
Vector rendering 102

Path-winding number



Vector rendering 102

Path-winding number



Flatten to segments

Scale animation

缩放 SVG 非常慢，为何浏览器 `transition: transform` 在文字上这么快?

群友: `Bitmap interpolation`

Scale animation

缩放 SVG 非常慢，为何浏览器 `transition: transform` 在文字上这么快?

群友: `Blur(Bitmap interpolation)`

Anti-aliasing

```
for pixel in screen {
  pixel.color = if path.contains(pixel) {
    black
  } else {
    white
  };
}
```

Anti-aliasing

```
for pixel in screen {
  pixel.color
    = black * path.intersection_ratio(pixel);
}
```

Anti-aliasing

```
for pixel in screen {
  let mut total: f64 = 0;
  for subpixels in pixel.subpixels() {
    diodes.on = path.contains(diodes);
  }
}
```

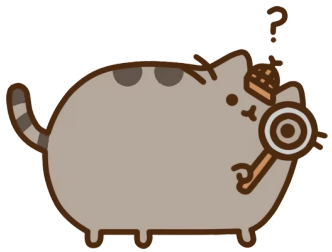
Anti-aliasing

```
for diode in screen {
  diodes.on = path.contains(diodes);
}
```

Anti-aliasing

```
for diode in screen {
  diodes.on = path.contains(diodes);
}
```

Question time!



https://layered.meow.plus