

With permission from William Löthman: <https://dribbble.com/shots/2112159-Hourglass-Icon>



# النص وتخطيط نظرة عامة

BlinkOn 5 - Dominik Rötsches

نوفمبر 2015



# Text & Layout Overview

BlinkOn 5 - Dominik RÃ¶ttsches

N o v e m b e r 2 0 1 5



# Text & Layout Overview

BlinkOn 5 - Dominik RÃ¶ttsches

Nove m b e r 2015



# Text & Layout Overview

BlinkOn 5 - Dominik Röttches

N o v e m b e r 2 0 1 5



# テキストとレイ アウトの概要

2 N R - B  
0 o ö I i  
1 v t D n  
5 e t o n k  
m s m k o  
b c i o  
e h n n  
r e i  
s k 5



# Text & Layout Overview

BlinkOn 5 -  
Dominik  
Röttches

N o v e m b e r  
2 0 1 5



# Text & Layout Overview

BlinkOn 5 - Dominik Röttches (drott@)

November 2015



# Agenda

- Goals & Challenges
- What does the Font code do?
- Recent Improvements
- Future Plans



# Challenges of Text & Layout (Incomplete)

- Complexity & Dependencies
- Pixel precision
- Optimal Font Matching & Fallback
- Shaping non-Latin languages
- Vertical text
- Performance
- Typographic features



# Goals of Text & Layout

- Correct and consistent display of all text in all languages
- High performance for complex text processing
- Low overhead & cost for advanced typographic features



# What does *Font.h* do?

- Measuring Text
- Drawing Text
- Selecting Text



# Measuring Text



# Measuring Text

$length = f(font, size, text)$



# Measuring Text

$length = f(font, size, direction, features, script, language, text, context)$



# Measuring Text

Times      Source Sans      Lobster

$$\text{length} = f(\text{font}, \text{size}, \text{direction}, \text{features}, \text{script}, \text{language}, \text{text}, \text{context})$$



# Measuring Text

32

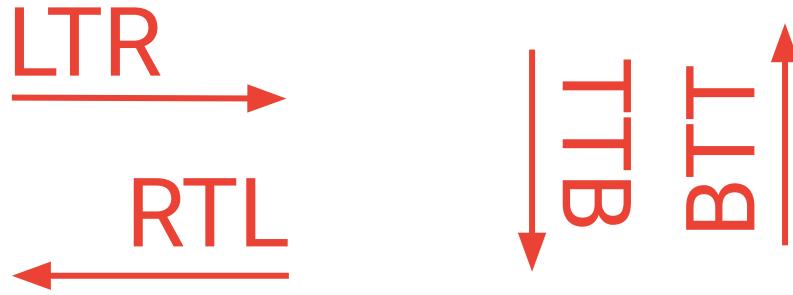
16

6

$length = f(font, \text{size}, direction, features, script, language, text, context)$



# Measuring Text



$$length = f(font, size, \textcolor{red}{direction}, features, script, language, text, context)$$



# Measuring Text

- Small Caps
- Ligatures
- Swashes, Ornamentals
- Number Forms
- ... more

$length = f(font, size, direction, features, script, language, text, context)$



# Measuring Text

- Latin
- Arabic
- Hiragana
- Cyrillic
- Traditional & Simplified Chinese
- “Common”
- ... more

$length = f(font, size, direction, features, \textcolor{red}{script}, language, text, context)$



# Measuring Text

Language	Script
English, German, Finnish...	<i>Latin</i>
Azerbaijani	<i>Cyrillic, Latin, Arabic</i>
Chinese	<i>Traditional, Simplified</i>

$$length = f(font, size, direction, features, script, language, text, context)$$



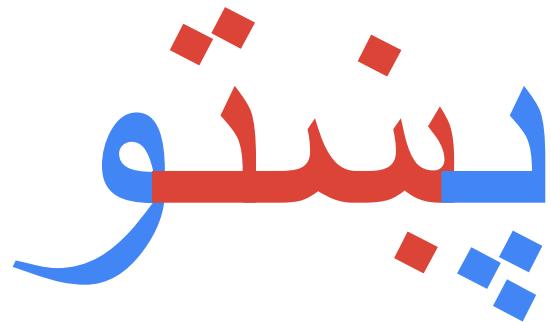
# Measuring Text

  LoremHello WorldIpsum

$$\text{length} = f(\text{font}, \text{size}, \text{direction}, \text{features}, \text{script}, \text{language}, \text{text}, \text{context})$$



# Measuring Text



بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

$$length = f(font, size, direction, features, script, language, \textcolor{red}{text}, \textcolor{blue}{context})$$



# Measuring Text

ب پشت و

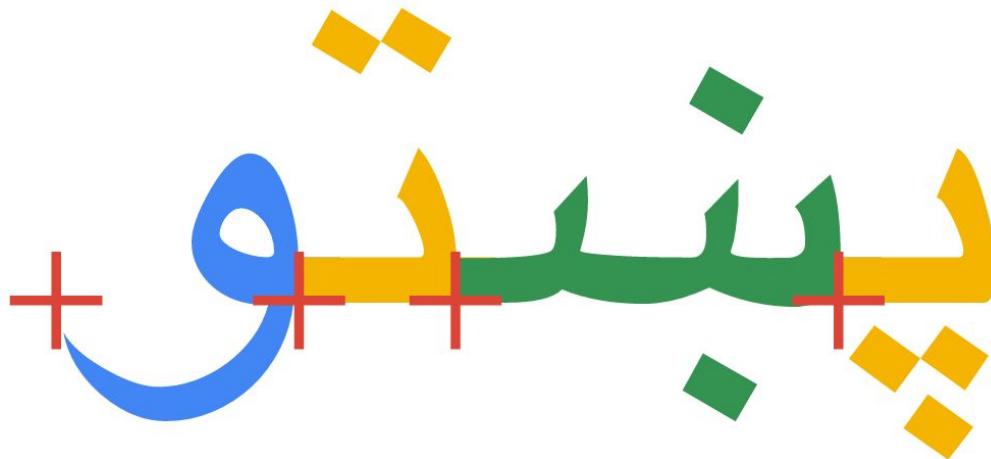
$$length = f(font, size, direction, features, script, language, \textcolor{red}{text}, \textcolor{blue}{context})$$



# Drawing Text



# Drawing Text



# Drawing Text

é a e  
s o



# Drawing Text

é a e  
s o



# Drawing Text

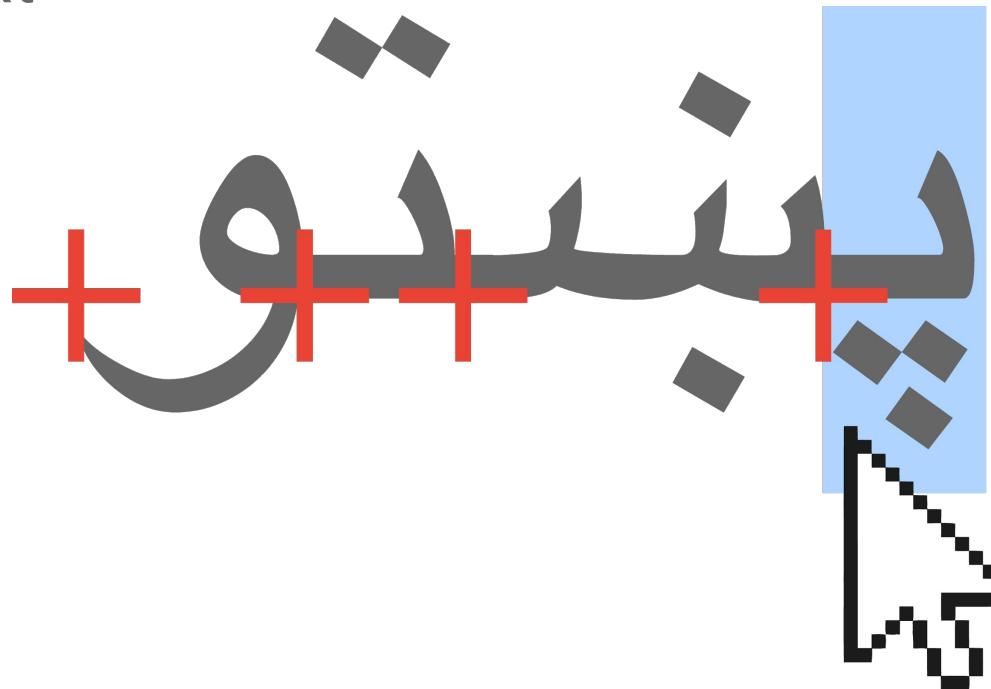
GlyphBuffer



# Selecting Text



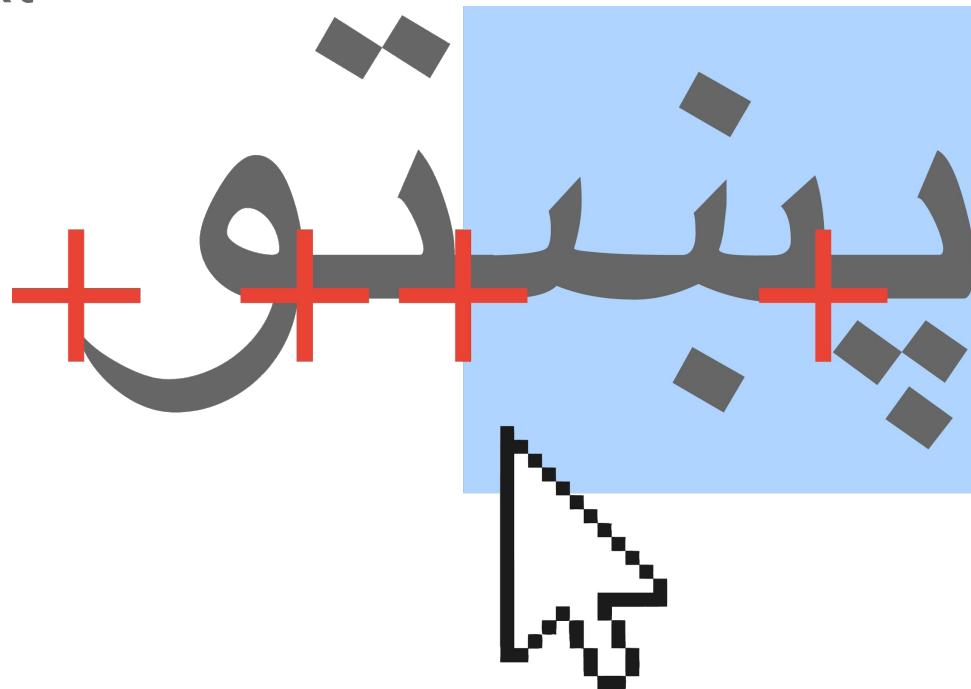
# Selecting Text



Character Position  $\Leftrightarrow$  Screen Coordinates



# Selecting Text



Character Position  $\Leftrightarrow$  Screen Coordinates



## *Font.h* does:

- Measuring Text
- Drawing Text
- Selecting Text



# Shaping



# Shaping

پ بُت و  
پ بُت و

é a a e

*f(font, size, direction, features, script, language, text, context)*



# Simple vs Complex



# Simple vs. Complex Text Path

Simple

پښتو

Lobst r

Complex

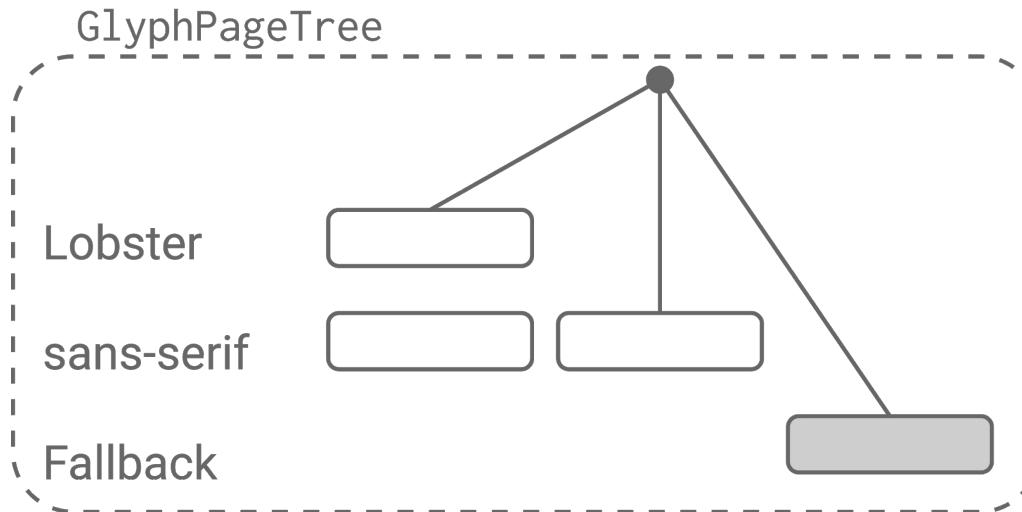
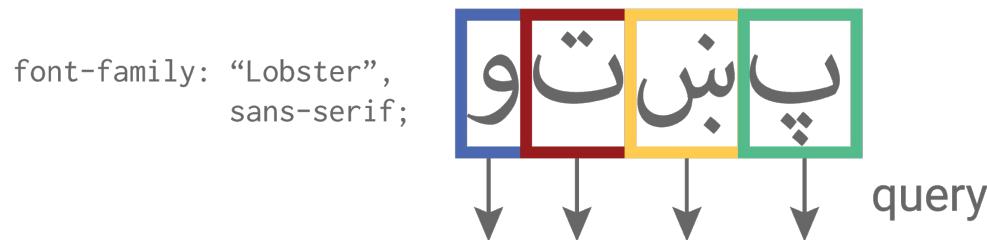
پښتو

Lobst r

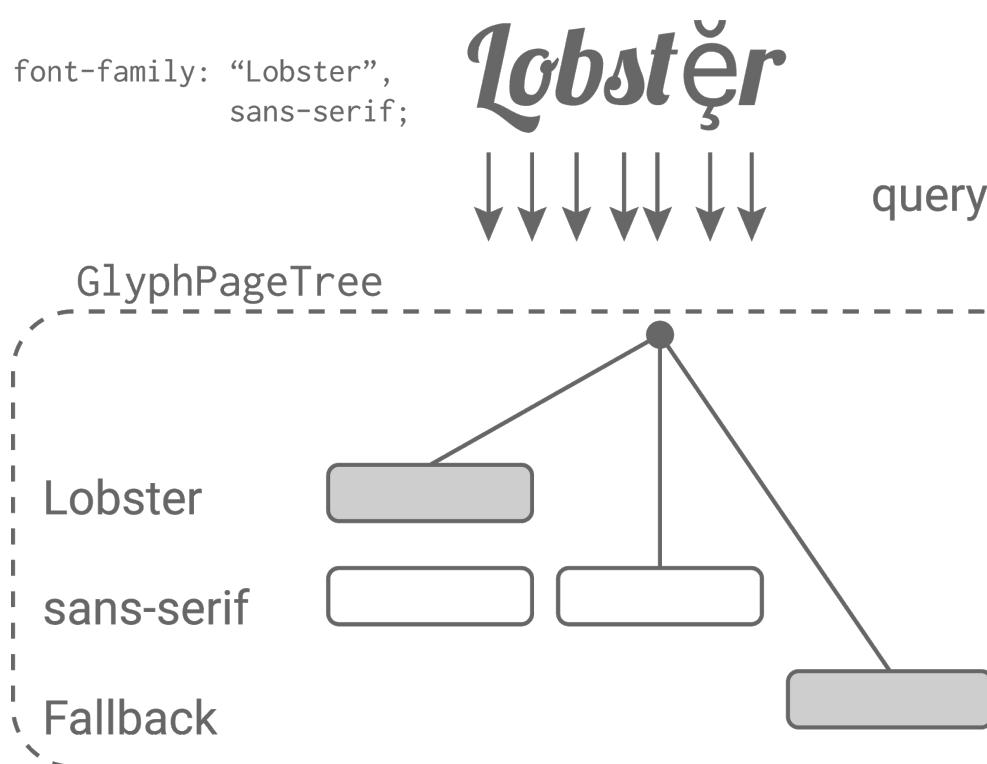
font-family: "Lobster", sans-serif;



# Simple Path



# Simple Path



# Complex Path = Full Shaping

پ بنت و

پ بنت و

The diagram shows the characters 'e', 'a', and 'a' with various red '+' marks indicating specific points of interest or features. The first 'e' has a green '+' at the top and a yellow '+' at the bottom. The first 'a' has a green '+' at the top and a red '+' at the bottom. The second 'a' has a yellow '+' at the top and a blue '+' at the bottom. A horizontal line passes through the middle of the characters.

$$length = f(font, size, direction, features, script, language, text, context)$$



# Simple vs. Complex Text Path

Simple Path	Complex Path
Typewriter-style glyph selection	<b>Shaping</b> , Contextual glyph selection
Horizontal advances only	2D glyph placement
No control over typographic features	Full typographic feature support
Fails on scripts that require contextual shaping	Comprehensive script and language support



# Recently in Text & Layout

- Switching to Complex Text
- Word Cache
- Shaper-driven Segmentation
- Improving Font Fallback
- Specification Work & Improvements  
in CSS3 Writing-modes & Fonts



# Switching to Complex



# Switching to Complex

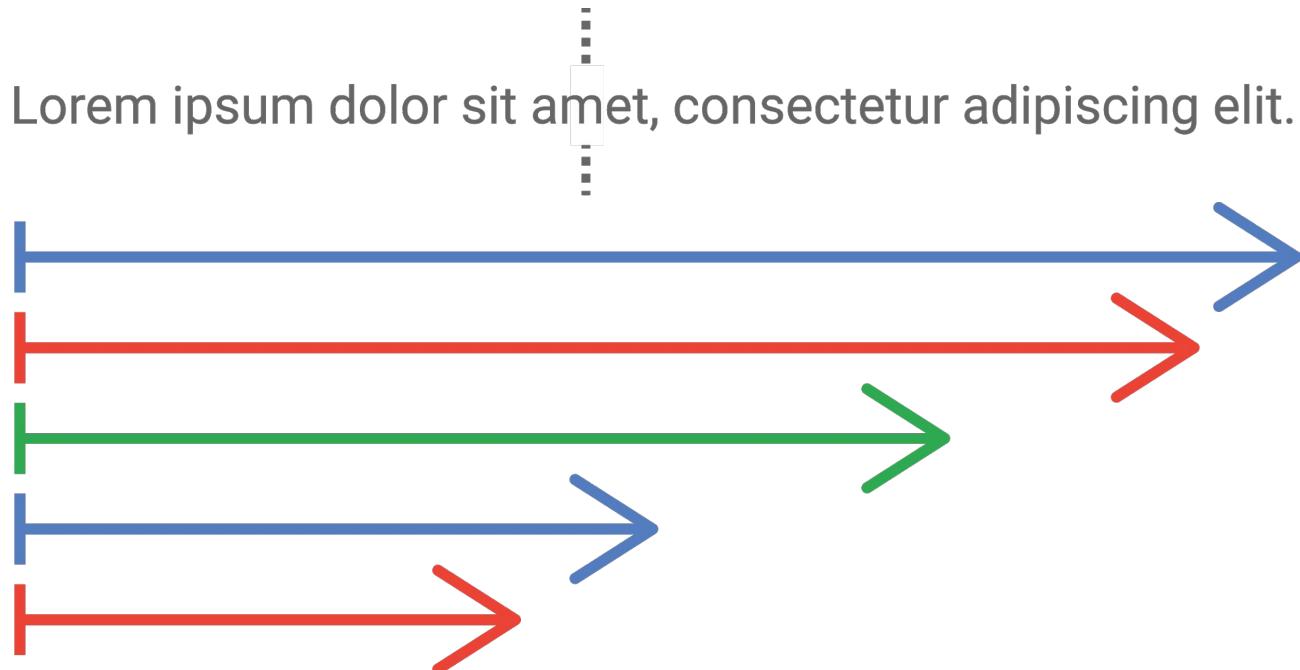
- No more rendering difference bugs: simple vs. complex
- Unified code
- Advanced typography always available
- Complex script languages become first class citizen

But:

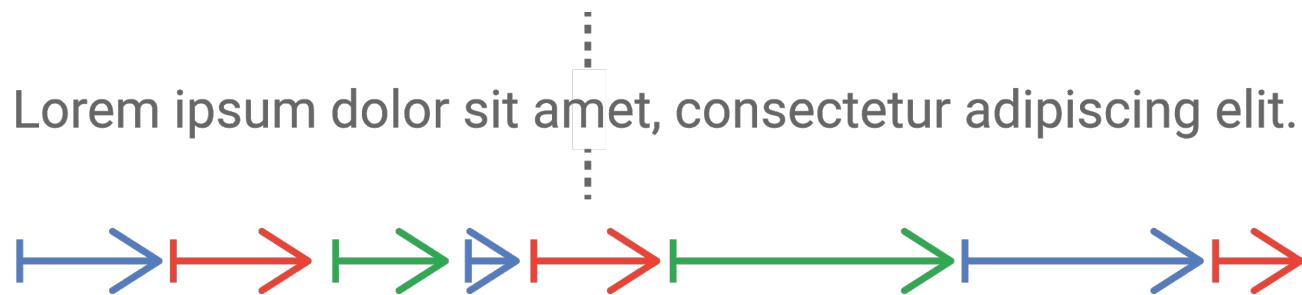
- Full shaping is more costly
- Avoid performance regressions
- Word cache!



# Layout of a TextRun - No Word Cache



# Layout of a TextRun - With Word Cache



Word Cache aids in Font.h operations:  
Width, Drawing, Selection



# Shaper Driven Segmentation

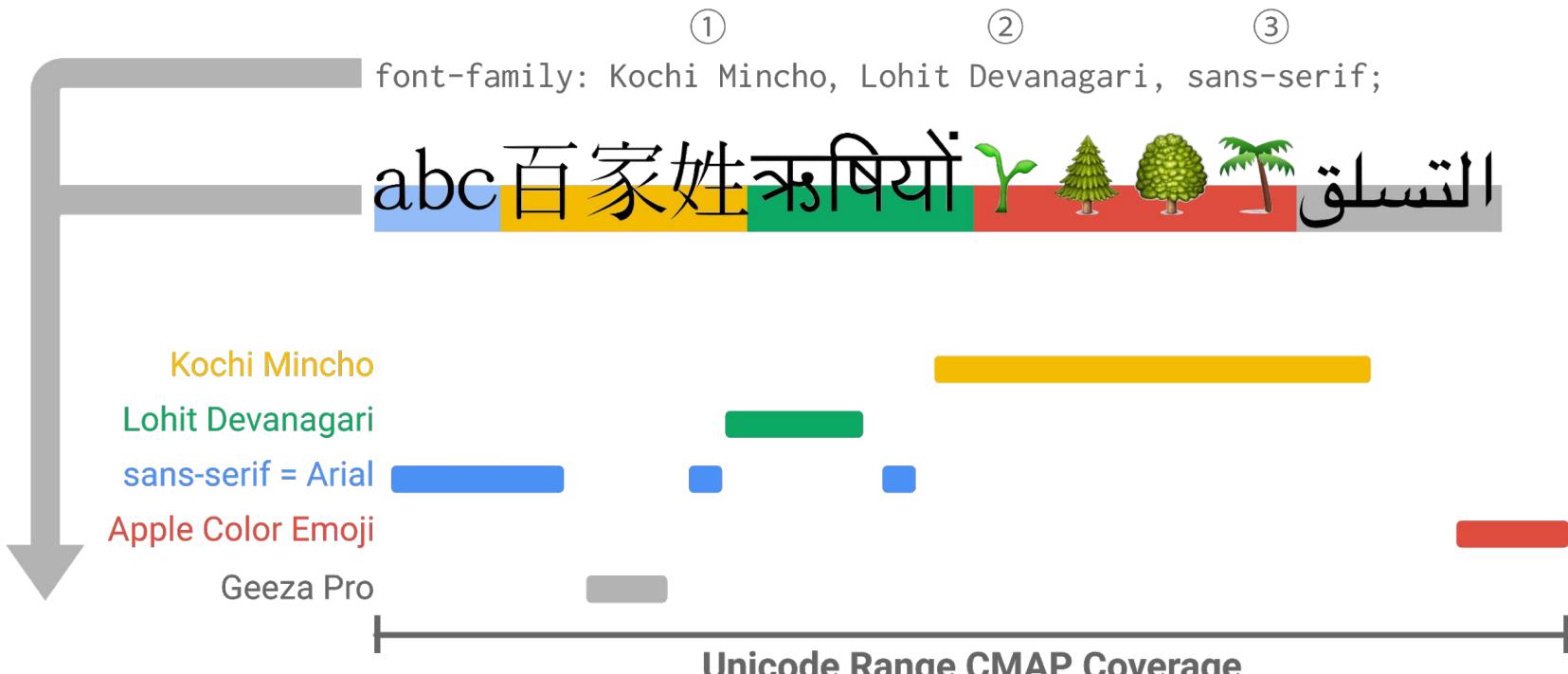
- Splitting a text run into individual pieces suitable for shaping
- Constant values for the variables of measuring text



# Font Coverage

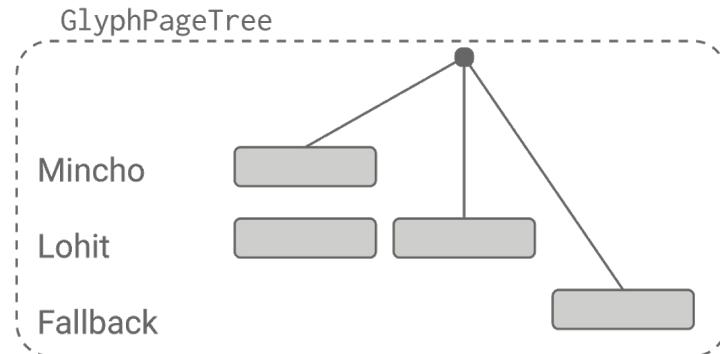


# Font Fallback



# Before: Segmentation through GlyphPageTree results

百家姓 କଷ୍ଟଧିରୋ ୟ ପ୍ରତିକାଳୀନ ହୋଦତ ଅଲେଖା ବାବା ଜାତି



# New: Clean Run Segmentation Stage

百家姓 କଷ୍ଣିଯୋं 🌳 🌲 🌴 百家姓 🌳 🌲 🌴 百家姓 حوادث التسلق 百家姓



# New: Shaper Driven Segmentation for Font Fallback

Font Fallback  
↓

Mincho

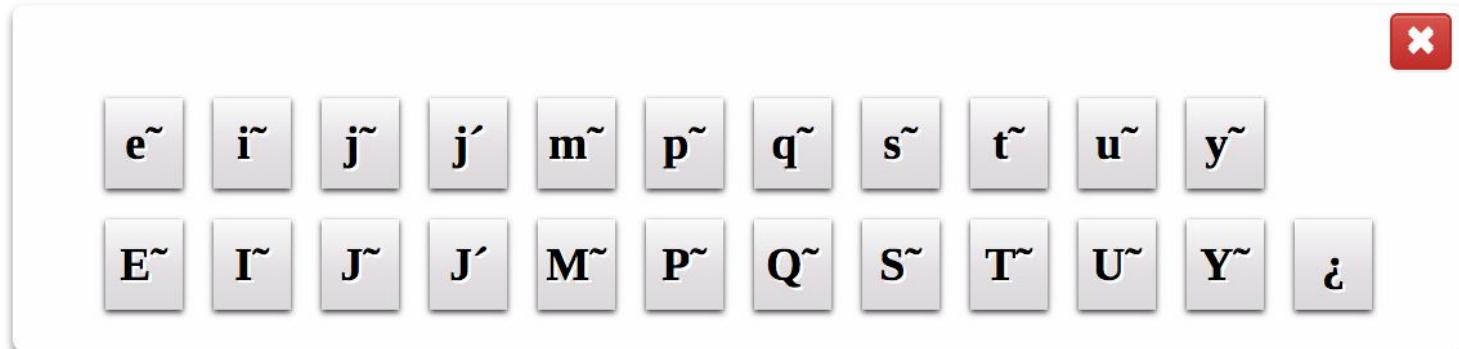
Lohit

Fallback: Emoji



# Before: Combining Marks

[crbug.com/454405](http://crbug.com/454405), reported by [javsmo](#)



Rendered Fonts

Times — 1 glyph  
Tinos — 1 glyph



# Before: Fallback Mixup for Combining Marks

U+0065 LATIN SMALL LETTER E  
U+0327 COMBINING CEDILLA  
U+0306 COMBINING BREVE



# Grapheme Cluster Based Fallback & Font Selection



Rendered Fonts

Times — 2 glyphs



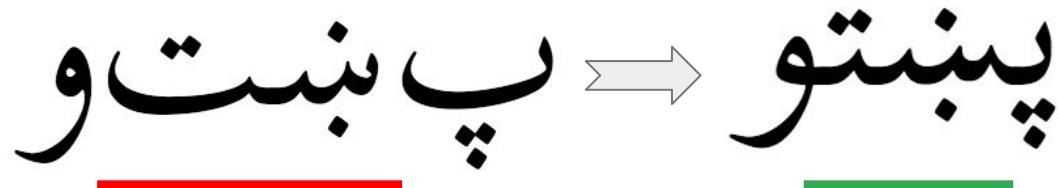
# Grapheme Cluster Based Fallback & Font Selection

U+0065 LATIN SMALL LETTER E  
U+0327 COMBINING CEDILLA  
U+0306 COMBINING BREVE



## Shaping context across font fallback

پ ب ن ت و → پ ب ن ت و



unicode-range:

U+069A, U+062A, U+646;

م ت ن → م ت ن



# Specification Work & Improvements



# CSS Font Matching Improvements

- CSS3 Fonts Module Section 5.2 § 4
- Priority order of matching
  - Choose closest **stretch** value, then
  - Choose closest **style** value, then
  - Choose closest **weight**, then
  - Match **size**
- Two main issues:
  - Stretch matching was not implemented, [#513670](#)
  - Priority order violated, [#513669](#)

The screenshot shows a browser window displaying the W3C Editor's Draft page for the CSS Fonts Module Level 3. The URL is <https://drafts.csswg.org/css-fonts/#font-style-matching>. The page content describes the priority order for font matching, starting with stretch and then moving to style, weight, and size. It includes several numbered steps and sub-steps detailing the logic for each property.

4. If a font family match occurs, the user agent assembles the set of font faces in that family and then narrows the set to a single face using other font properties in the order given below. A group of faces defined via `@font-face` rules with identical font descriptor values but differing "`unicode-range`" values are considered to be a single *composite face* for this step:

- a. "`font-stretch`" is tried first. If the matching set contains faces with width values matching the "`font-stretch`" value, faces with other width values are removed from the matching set. If there is no face that exactly matches the width value the nearest width is used instead. If the value of "`font-stretch`" is "`normal`" or one of the condensed values, narrower width values are checked first, then wider values. If the value of "`font-stretch`" is one of the expanded values, wider values are checked first, followed by narrower values. Once the closest matching width has been determined by this process, faces with other widths are removed from the matching set.
- b. "`font-style`" is tried next. If the value of "`font-style`" is "`italic`", italic faces are checked first, then oblique, then normal faces. If the value is "`oblique`", oblique faces are checked first, then italic faces and then normal faces. If the value is "`normal`", normal faces are checked first, then oblique faces, then italic faces. Faces with other style values are excluded from the matching set. User agents are permitted to distinguish between italic and oblique faces within platform font families but this is not required, so all italic or oblique faces may be treated as italic faces. However, within font families defined via `@font-face` rules, italic and oblique faces must be distinguished using the value of the "`font-style`" descriptor. For families that lack any italic or oblique faces, user agents may create artificial oblique faces, if this is permitted by the value of the "`font-synthesis`" property.
- c. "`font-weight`" is matched next, so it will always reduce the matching set to a single font face. If bolder/lighter relative weights are used, the effective weight is calculated based on the inherited weight value, as described in the definition of the "`font-weight`" property. Given the desired weight and the weights of faces in the matching set after the steps above, if the desired weight is available that face matches. Otherwise, a weight is chosen using the rules below:
  - If the desired weight is less than 400, weights below the desired weight are checked in descending order followed by weights above the desired weight in ascending order until a match is found.
  - If the desired weight is greater than 500, weights above the desired weight are checked in ascending order followed by weights below the desired weight in descending order until a match is found.
  - If the desired weight is 400, 500 is checked first and then the rule for desired weights less than 400 is used.
  - If the desired weight is 500, 400 is checked first and then the rule for desired weights less than 400 is used.
- d. "`font-size`" must be matched within a UA-dependent margin of tolerance. (Typically, sizes for scalable fonts are rounded to the nearest whole pixel, while the tolerance for bitmapped fonts could be as large as 20%.) Further computations, e.g., by "`em`" values in other properties, are based on the "`font-size`" value that is used, not the one that is specified.



# CSS Writing Mode Improvements

- Discussions in CSS WG on writing-mode attribute values concluded
- Pushed “CSS Writing Modes Level 3” to Candidate Recommendation status
- Intend to unprefix -webkit-writing-mode: to writing-mode: LGMT’ed
- Unprefixing implemented
- Contributing tests
- Test pass rate up: 85.78% (IE 71.33%, FF 86,24%)
- Spec editorship, contributions to W3C CSS Text, Text Decorations, Writing Modes, Ruby, Unicode UTR#50 Vertical Text Layout



# Recently in Text & Layout

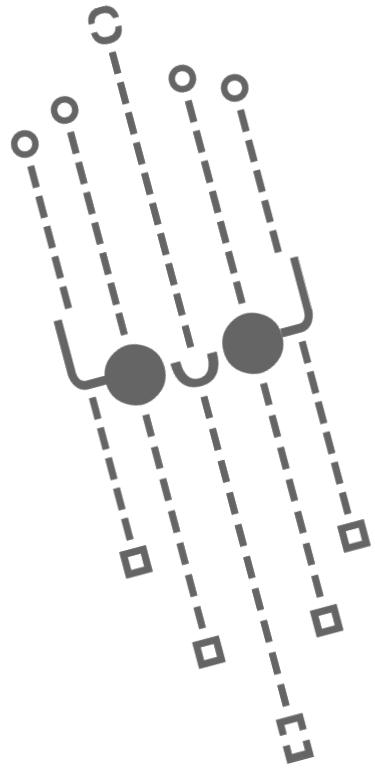
- Switching to Complex Text
- Word Cache
- Shaper-driven Segmentation
- Improving Fallback
- Spec Compliance Improvements



# Future Plans

- Fully remove Simple Path, GlyphPageTree and friends
- Improve Fallback (System APIs)
- Improve testability
- Improve WordCache caching





Thanks!

