Google Fonts

Making the Korean and Japanese web **beautiful & fast**
한국어 웹을 아름답고 빠르게
高速で美しい日本語のウェブを

Google

What is "Google Fonts"?

- Free, open-source, font collection
- Free, public, content delivery
- We want to serve all Google users

What is "Google Fonts"?

- Free, open-source, font collection
- Free, public, content delivery
- We want to serve all Google users

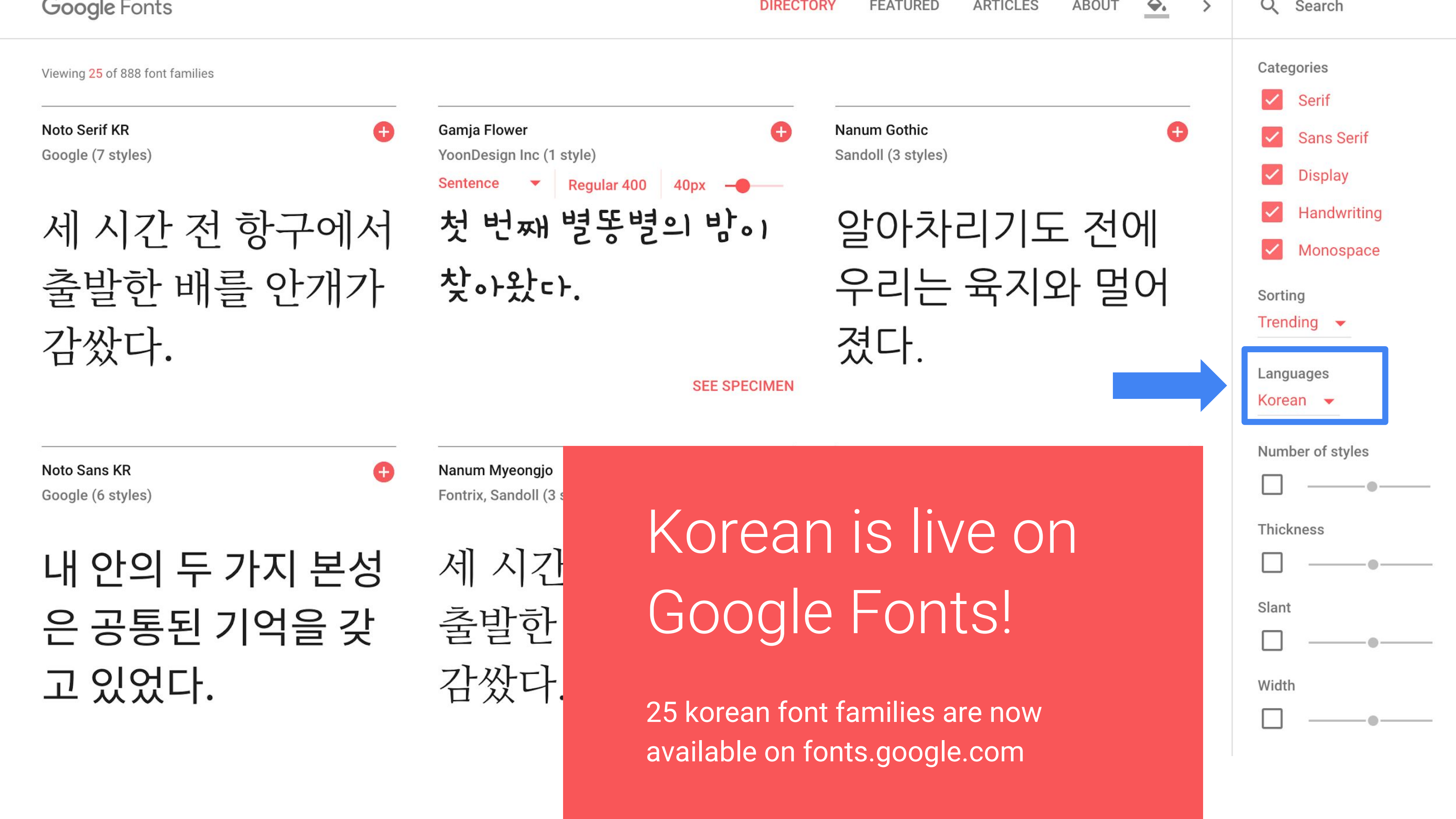
What does Google Fonts DO?

- Maintain & publish collection
- Web API for our collection
- Android API for our collection
- Helps maintain tools and supporting technologies

Google

No CJK Support :(

- Blocked due to latency for a long time
- We had Early Access (beta) support



Viewing 25 of 888 font families

Noto Serif KR
Google (7 styles)

세 시간 전 항구에서 출발한 배를 안개가 감쌌다.

Gamja Flower
YoonDesign Inc (1 style)

Sentence | Regular 400 | 40px

첫 번째 별똥별의 밤이 찾아왔다.

Nanum Gothic
Sandoll (3 styles)

알아차리기도 전에 우리는 육지와 멀어졌다.

Noto Sans KR
Google (6 styles)

내 안의 두 가지 본성은 공통된 기억을 갖고 있었다.

Nanum Myeongjo
Fontrix, Sandoll (3 s

세 시간
출발한
감쌌다.

Categories

☑ Serif
☑ Sans Serif
☑ Display
☑ Handwriting
☑ Monospace

Sorting
Trending

Languages
Korean

Number of styles
□

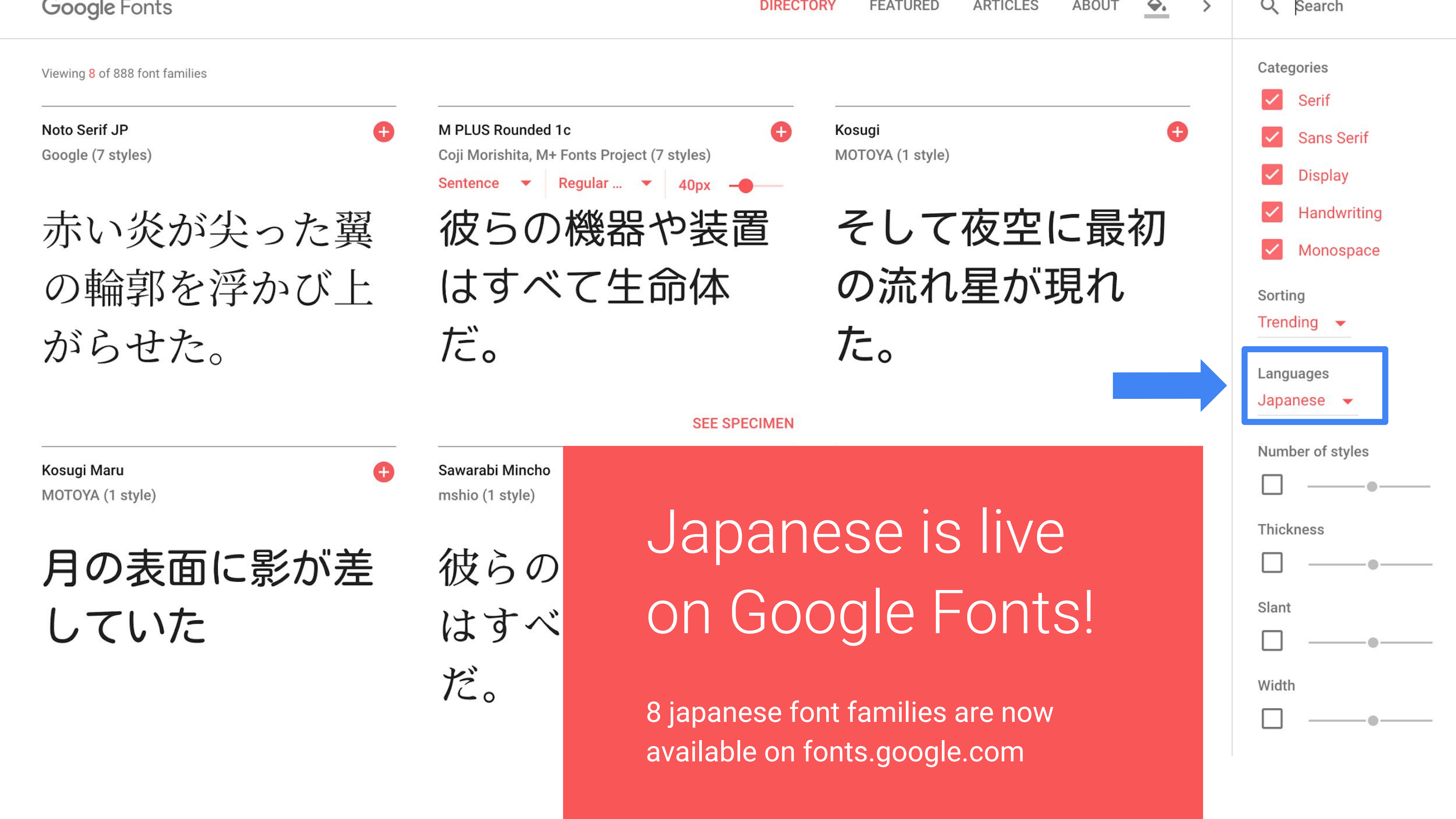
Thickness
□

Slant
□

Width
□

Korean is live on Google Fonts!

25 korean font families are now available on fonts.google.com



Viewing 8 of 888 font families

Noto Serif JP
Google (7 styles)

赤い炎が尖った翼の輪郭を浮かび上がらせた。

M PLUS Rounded 1c
Coji Morishita, M+ Fonts Project (7 styles)

Sentence ▾ Regular ... ▾ 40px

彼らの機器や装置はすべて生命体だ。

Kosugi
MOTOYA (1 style)

そして夜空に最初の流れ星が現れた。

Kosugi Maru
MOTOYA (1 style)

月の表面に影が差していた

Sawarabi Mincho
mshio (1 style)

彼らの
はすべ
だ。

Categories

☑ Serif
☑ Sans Serif
☑ Display
☑ Handwriting
☑ Monospace

Sorting
Trending ▾

Languages
Japanese ▾

Number of styles
□

Thickness
□

Slant
□

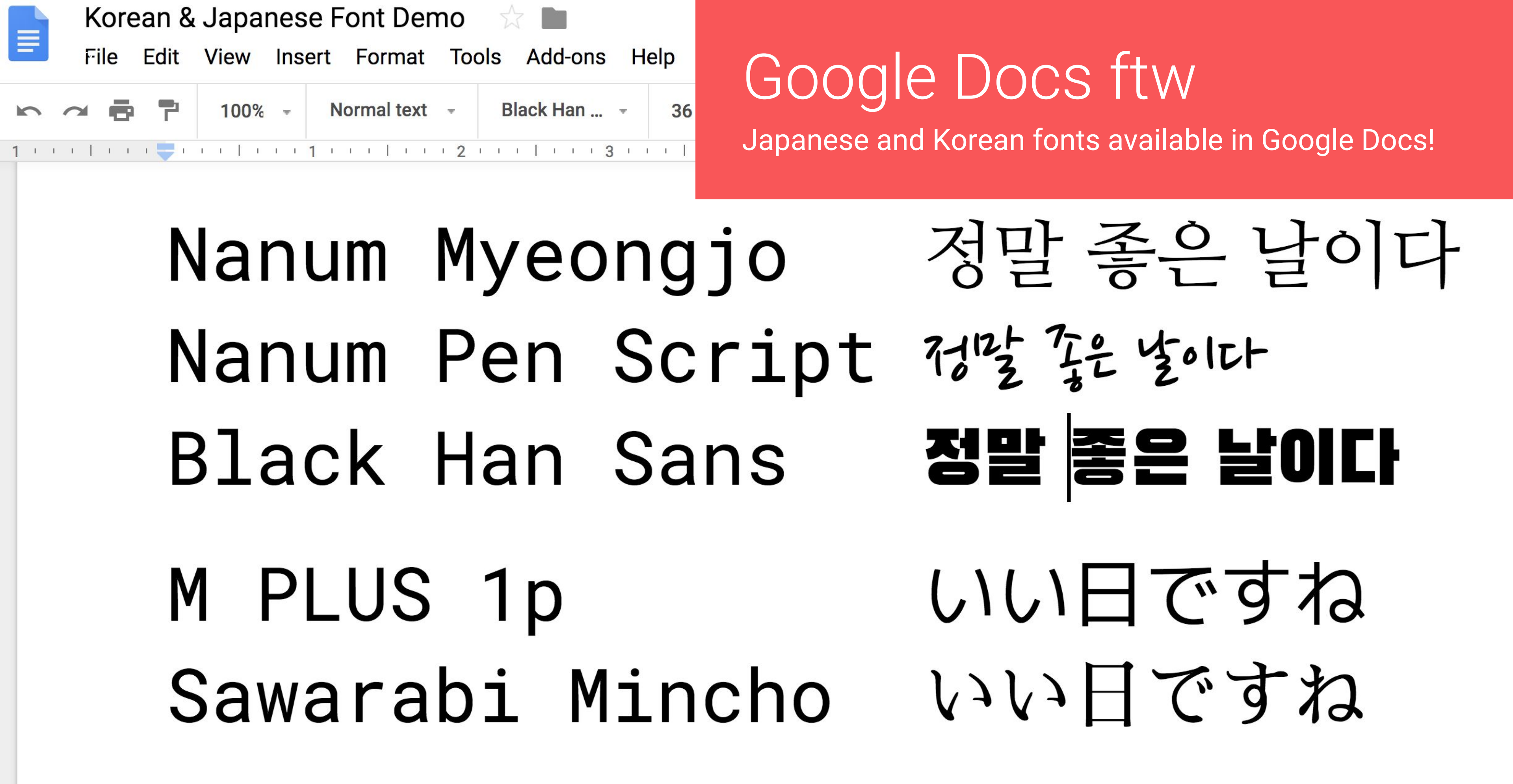
Width
□

Japanese is live on Google Fonts!

8 japanese font families are now available on fonts.google.com

Google Docs ftw

Japanese and Korean fonts available in Google Docs!



Nanum Myeongjo 정말 좋은 날이다

Nanum Pen Script 정말 좋은 날이다

Black Han Sans 정말 좋은 날이다

M PLUS 1p いい日ですね

Sawarabi Mincho いい日ですね

What took so long?

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

Google

What took so long?

- High cost per character
- Lots of characters
- We want a CSS solution
- Browser features

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

What took so long?

- High cost per character
- Lots of characters
- We want a CSS solution
- Browser features

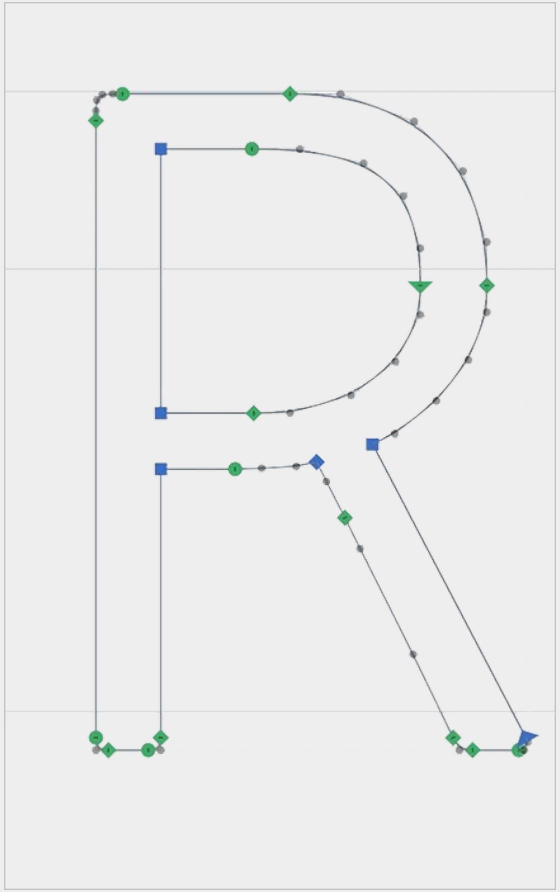
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

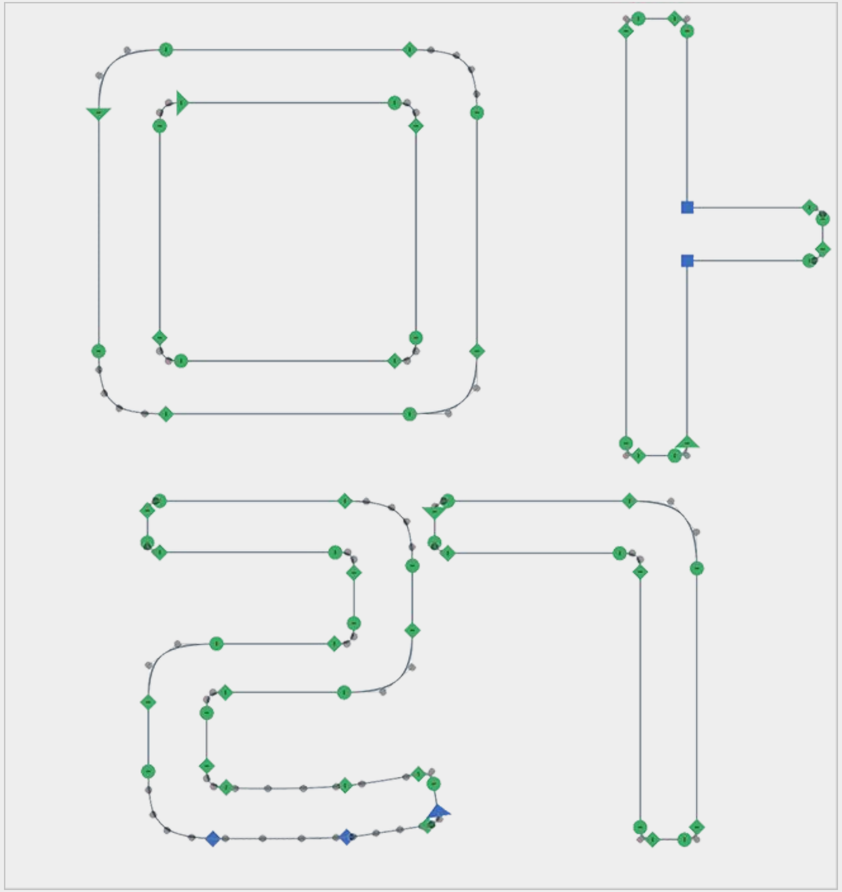
一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

R 맑

Cheap to transfer (52 pts)



Expensive to transfer (147 pts)



What took so long?

- High cost per character
- Lots of characters
- We want a CSS solution
- Browser features

Google

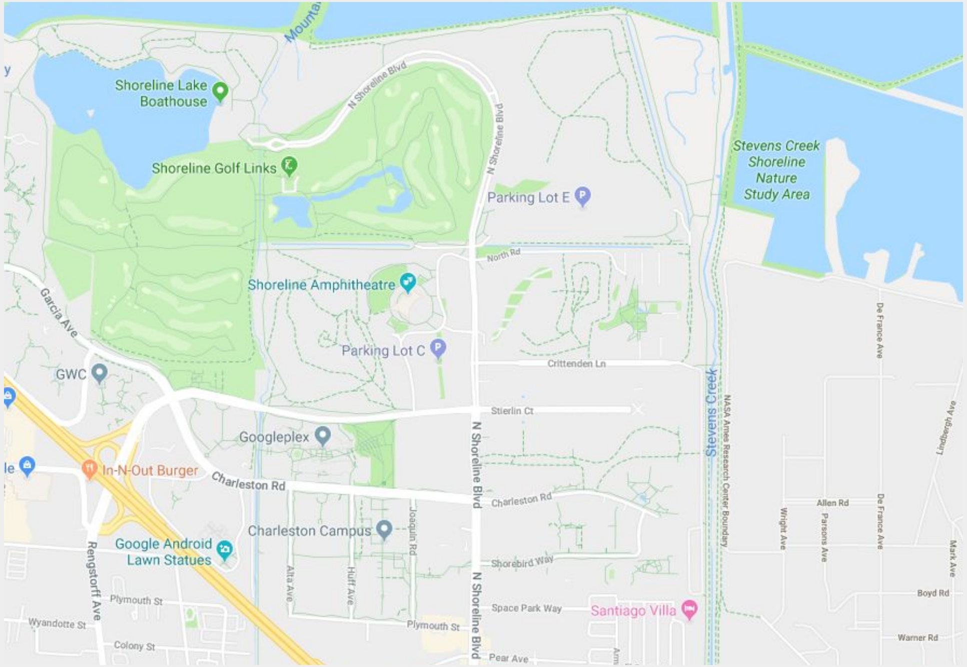
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

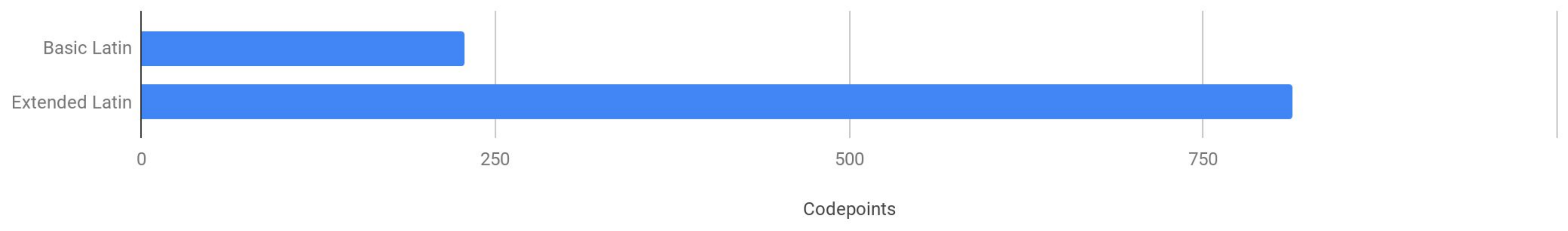
가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

Lots of characters

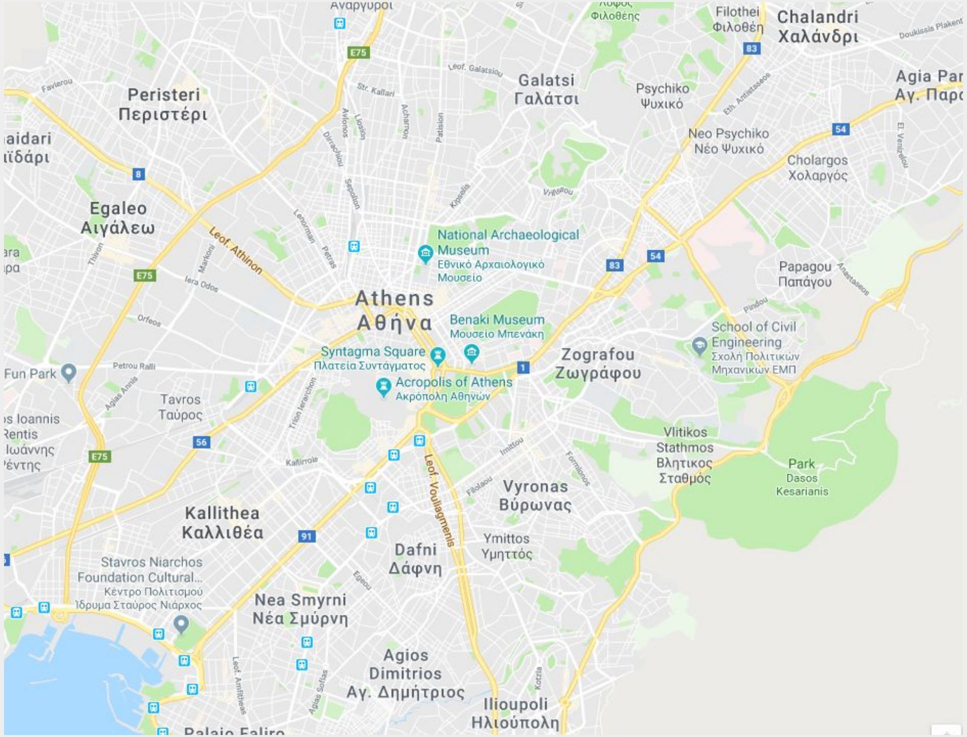
- Many US sites fit Basic Latin
- Going to Europe? Extended Latin

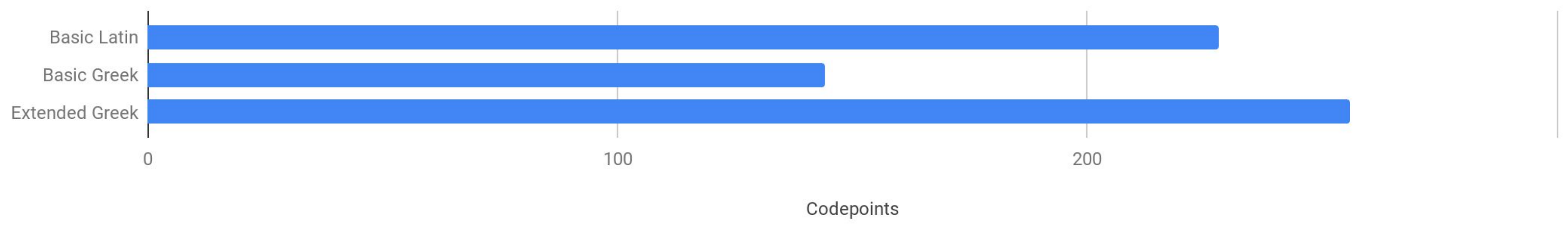




Lots of characters

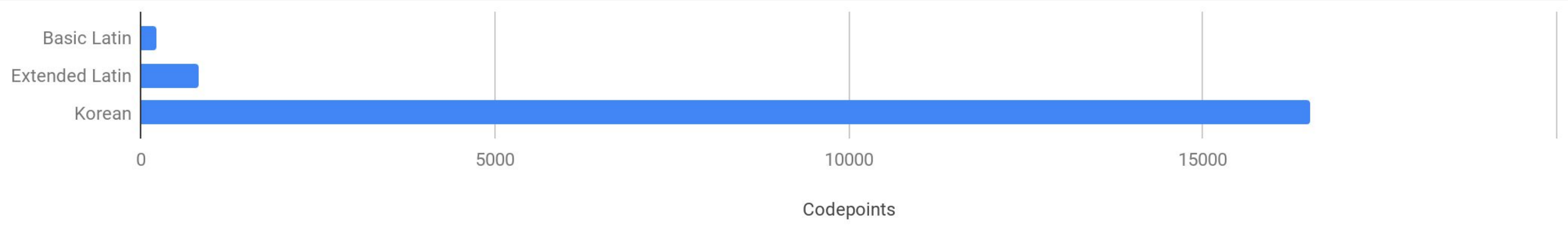
- Off we go to Greece, still fine
- Generally ≤ 1,000 characters





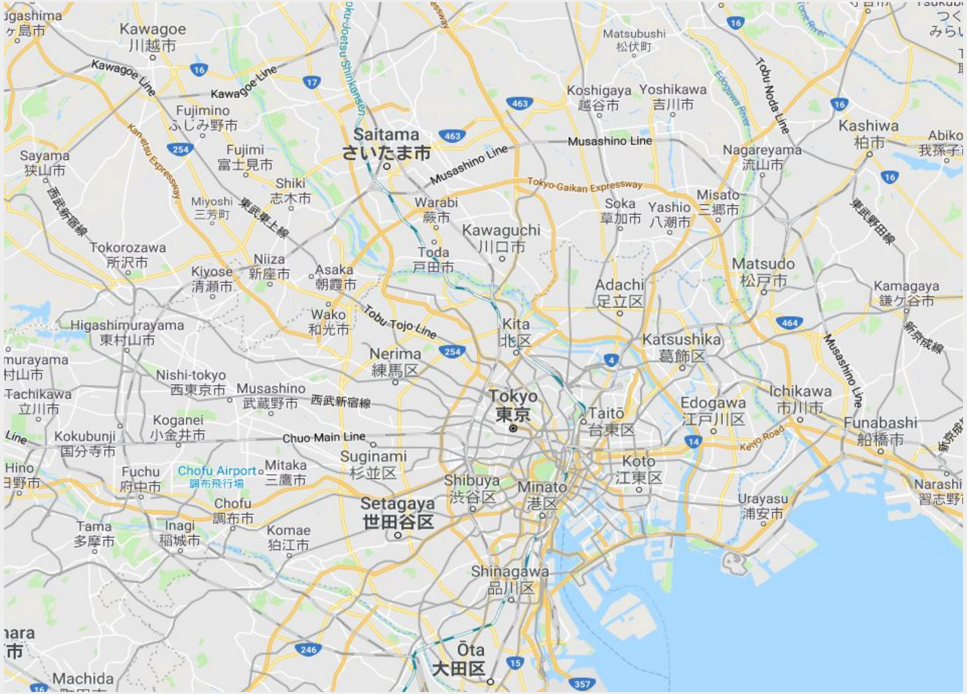
Lots of characters

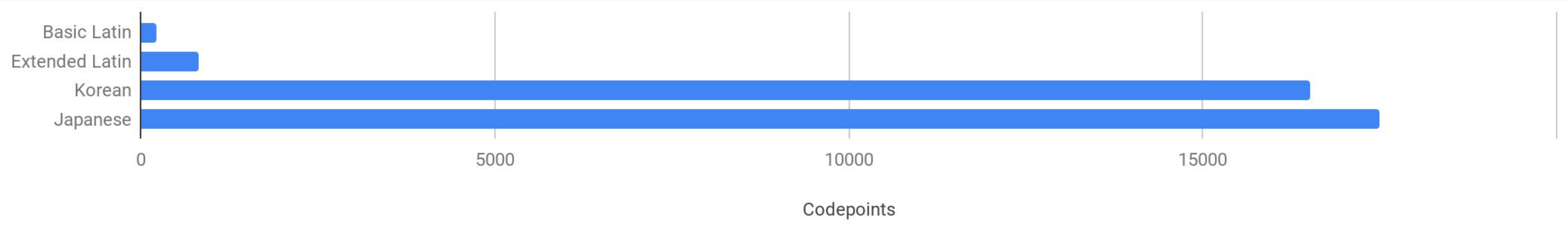
- On to Seoul
- latin ≤ 1,000 characters
- korean > 10,000 characters



Lots of characters

- Tokyo
- Even more complex characters
- > 10,000 characters





What took so long?

- High cost per character
- Lots of characters
- We want a CSS solution
- Browser features

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

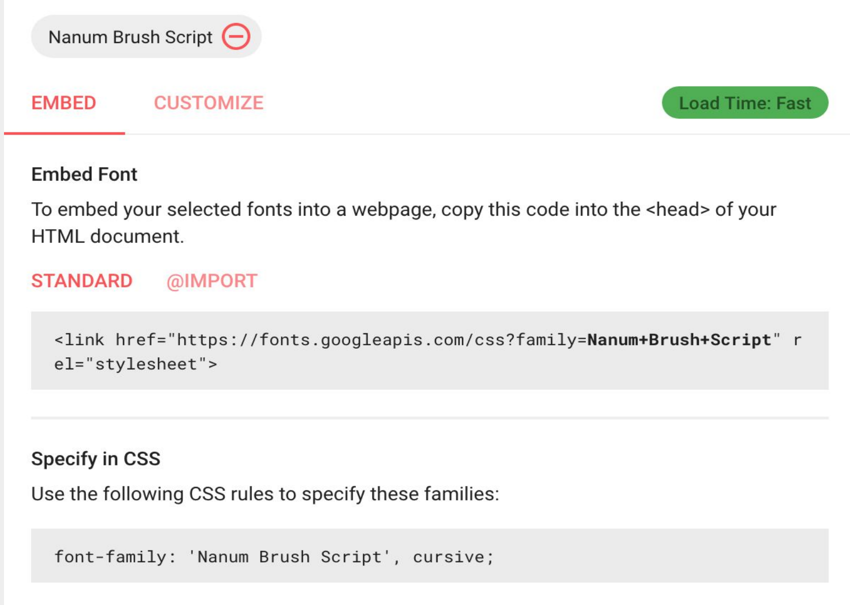
가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

We want a CSS solution

CSS only, no API changes

- API aims for simplicity and ease of use
- Should work like any other font
- Allows us to deploy ongoing optimizations transparently



Embed Font

To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.

```
<link href="https://fonts.googleapis.com/css?family=Nanum+Brush+Script" r
el="stylesheet">
```

Specify in CSS

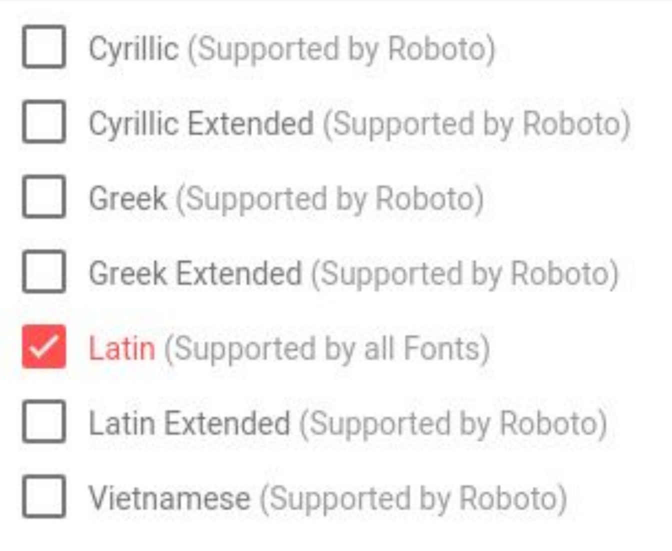
Use the following CSS rules to specify these families:

```
font-family: 'Nanum Brush Script', cursive;
```

Google

We want a CSS solution: subsetting

- 71% of web pages on httparchive.org use fonts
  - Trend toward more and larger fonts
  - https://httparchive.org/reports/page-weight
- Most pages use a small set of characters
- Cut fonts into pieces, let user pick
- Doesn't work for CJK
- Subsets are prohibitively large



□ Cyrillic (Supported by Roboto)

□ Cyrillic Extended (Supported by Roboto)

□ Greek (Supported by Roboto)

□ Greek Extended (Supported by Roboto)

☑ Latin (Supported by all Fonts)

□ Latin Extended (Supported by Roboto)

□ Vietnamese (Supported by Roboto)

What took so long?

- High cost per character
- Lots of characters
- We want a CSS solution
- Browser features

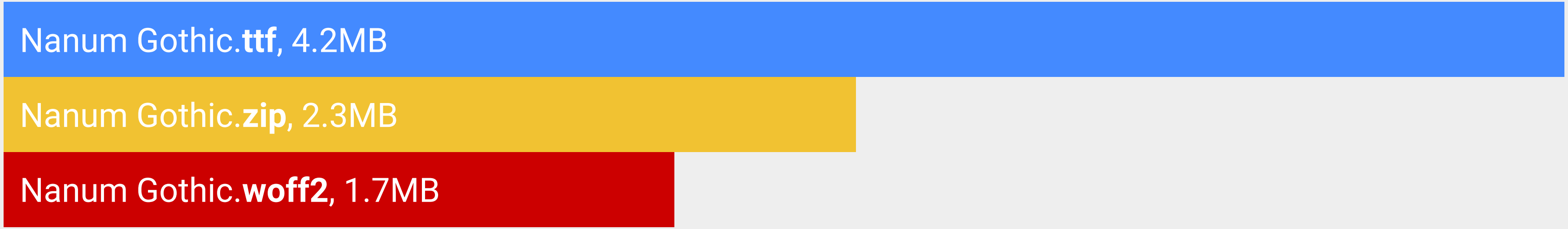
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

가각갂갃
간갅갆갇
갈갉갊갋
갌갍갎갏
감갑값갓
갔강갖갗
갘같갚갛
개객갞갟
갠갡갢갣
갤갥갦갧
갨갩갪갫
갬갭갮갯
갰갱갲갳
갴갵갶갷
갸갹갺갻
갼갽갾갿
걀걁걂걃
걄걅걆걇
걈걉걊걋
걌걍걎걏
걐걑걒걓
걔걕걖걗
걘걙걚걛

一二三四
五六七八
九十百千
上下左右
中大小月
日年早木
林山川土
空田天生
花草虫犬
人名女男
子目耳口
手足見音
力気円入
出立休先
夕本文字
学校村町
森正水火
玉王石竹
糸貝車金
雨赤青白
数多少万
半形太細
広長点丸

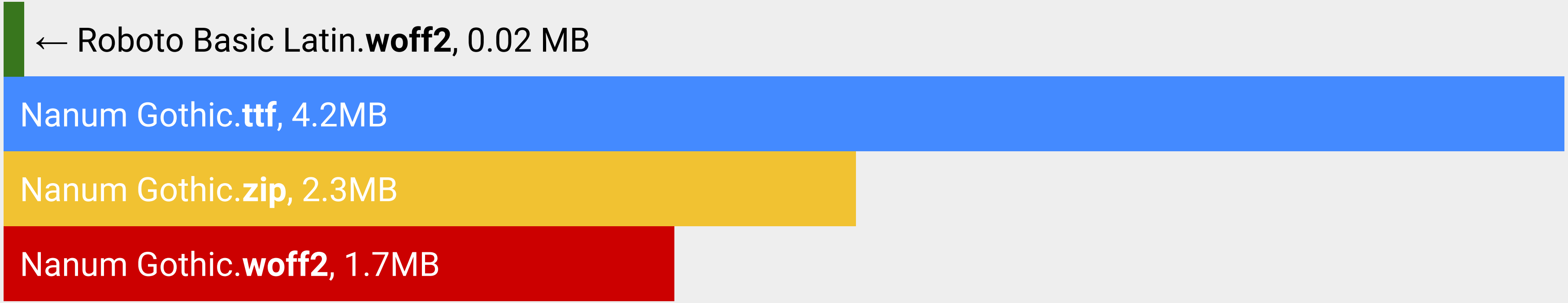
Browser features: WOFF2

- Rewrite parts of the font to be smaller
- Compress with Brotli
- 25–30% smaller than Zip



Browser features: WOFF2

- It's good … but not good enough for CJK



Browser features: unicode-range

```
http://fonts.googleapis.com/css?family=Merriweather

/* cyrillic */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/cyrillic.woff2);
    unicode-range: U+0400-045F;


/* latin */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/latin.woff2);
    unicode-range: U+0000-00FF;
```

Browser features: unicode-range

http://fonts.googleapis.com/css?family=Merriweather

```
/* cyrillic */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/cyrillic.woff2);
    unicode-range: U+0400-045F;


/* latin */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/latin.woff2);
    unicode-range: U+0000-00FF;
```

Download this

Browser features: unicode-range

http://fonts.googleapis.com/css?family=Merriweather

```
/* cyrillic */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/cyrillic.woff2);
    unicode-range: U+0400-045F;


/* latin */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/latin.woff2);
    unicode-range: U+0000-00FF;
```

Download this

If the page uses this

Browser features: unicode-range

http://fonts.googleapis.com/css?family=Merriweather

```
/* cyrillic */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/cyrillic.woff2);
    unicode-range: U+0400-045F;
```

Download this

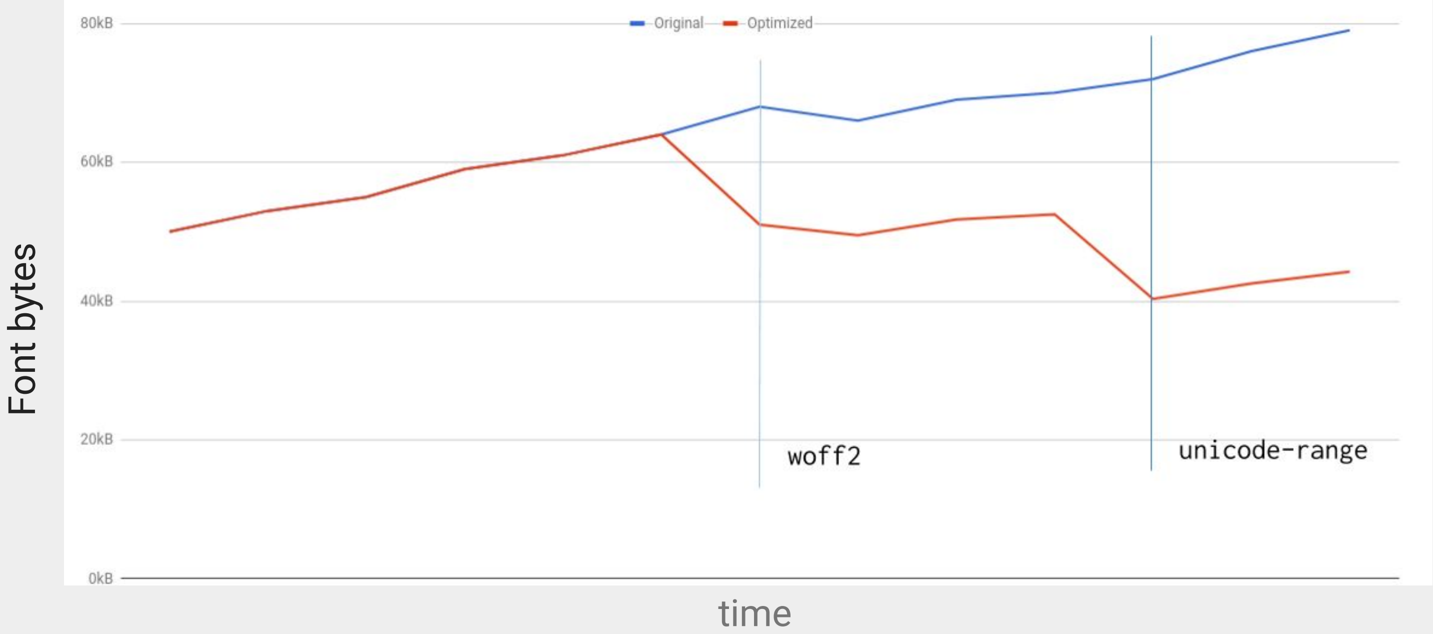
If the page uses this

```
/* latin */
@font-face
    font-family: 'Merriweather';
    src: url(http://fonts.gstatic.com/s/merriweather/latin.woff2);
    unicode-range: U+0000-00FF;
```

Download this

If the page uses this

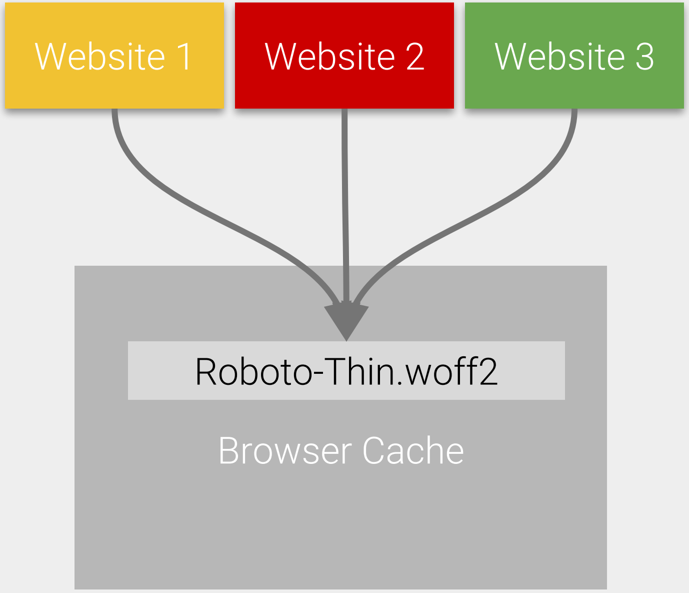
Browser features: WOFF2 + unicode-range



Google https://developers.googleblog.com/2015/02/smaller-fonts-with-woff-20-and-unicode.html

Browser features: cross-site caching

- Key benefit of centrally hosted fonts
- Less downloads as a font gets used more
- Our stats suggest this is very effective



Google

Browser features: HTTP/2

- HTTP/1.1 queues aggressively
  - < 10 fragments should be actively used

- HTTP/2 is more concurrency-enthused
  - 10's of fragments can be actively used

Browser features: Critical Mass

{Chrome, Firefox, Safari, Edge}

- unicode-range
- WOFF2
- HTTP/2

How do we ship Korean?

1. Cut the fonts into lots of pieces

2. Tell the browser about the pieces via unicode-range

Great! How exactly do you want those fonts segmented?

Google

Korean in use:
extremely skewed toward popular characters

Page	**Total characters**	**Unique Characters**
News site	> 50K	729
YouTube video + comments	1,848	394
Combined	> 50K	734

The YouTube page added only 5 more unique characters.

We should gather this type of data for a large set of Korean web pages!

Define problem space + data source

- Define "Korean"
  - Set of characters
  - 17.4k total

- Look at usage
  - Google indexes the web
  - Examine a set of Korean samples

가각갂갃간갅갆갇갈갉갊갋갌갍갎갏
갗갘같갚갛개객갞갟갠갡갢갣갤갥갦
갮갯갰갱갲갳갴갵갶갷갸갹갺갻갼갽
걅걆걇걈걉걊걋걌걍걎걏걐걑걒걓걔
걜걝걞걟걠걡걢걣걤걥걦걧걨걩걪걫
걳건걵걶걷걸걹걺걻걼걽걾걿검겁겂
겊겋게겍겎겏겐겑겒겓겔겕겖겗겘겙
겡겢겣겤겥겦겧겨격겪겫견겭겮겯결
겸겹겺겻겼경겾겿곀곁곂곃계곅곆곇
곏곐곑곒곓곔곕곖곗곘곙곚곛곜곝곞
곦곧골곩곪곫곬곭곮곯곰곱곲곳곴공
곽곾곿관괁괂괃괄괅괆괇괈괉괊괋괌
괔괕괖괗괘괙괚괛괜괝괞괟괠괡괢괣
괫괬괭괮괯괰괱괲괳괴괵괶괷괸괹괺

Google

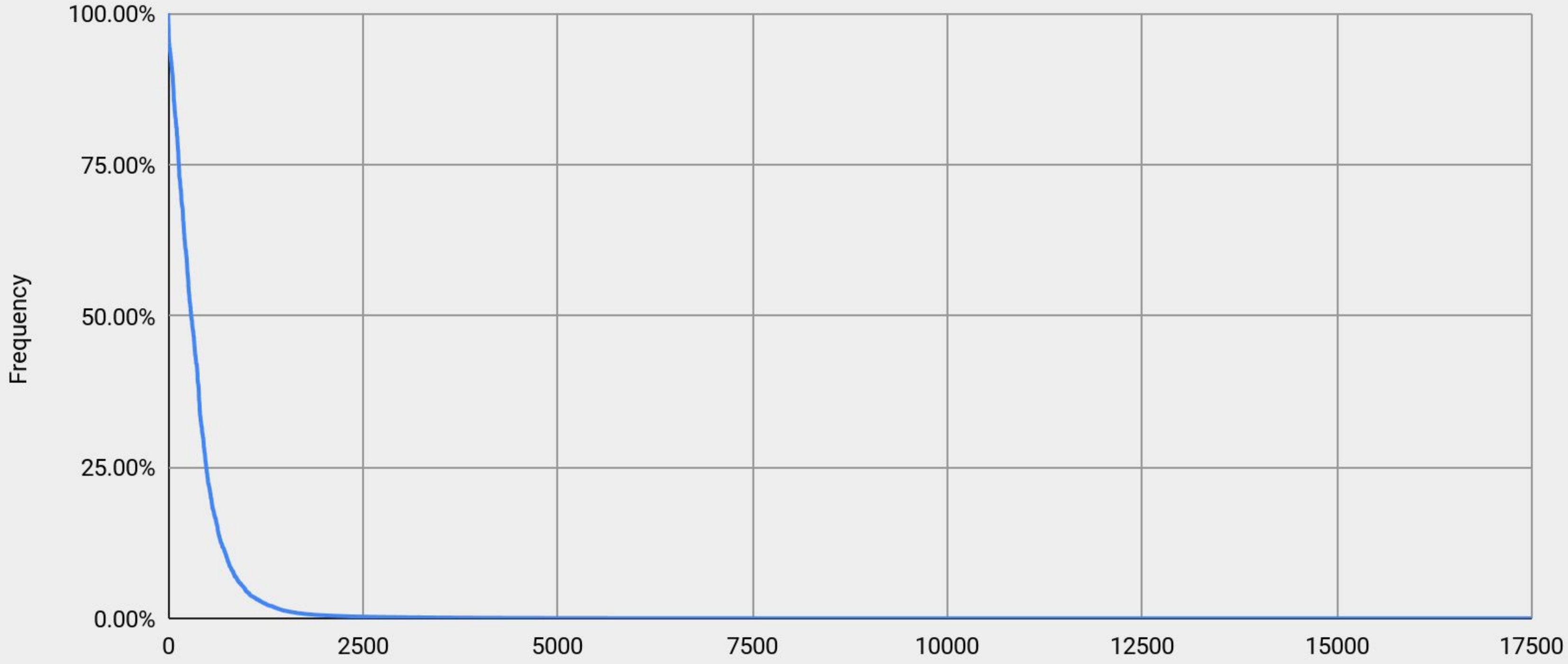
Examine data

- Find the set of characters used on each sample page
  - Ignore multiple uses on same page
- Count # of pages using each character

```
Character #Pages Description
0xB2E4        92 다 HANGUL SYLLABLE DA
0xC758         0 의 HANGUL SYLLABLE YI
0xB300      4012 대 HANGUL SYLLABLE DAE
...17k more...
```

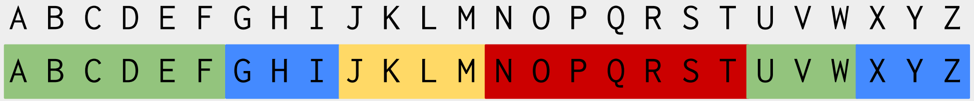
Frequency of Korean characters

Examined millions of Korean web pages



Great! How **exactly** do you want those fonts segmented?

Ways to segment



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

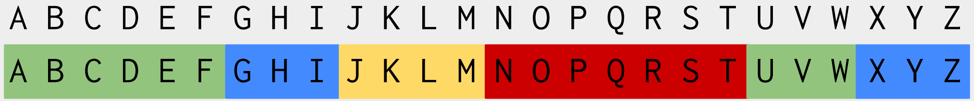
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Do Nothing

- Ship the whole font in a single segment
- Provides baseline for comparison

One giant font file

Ways to segment



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F | G H I | J K L M | N O P Q R S T | U V W | X Y Z

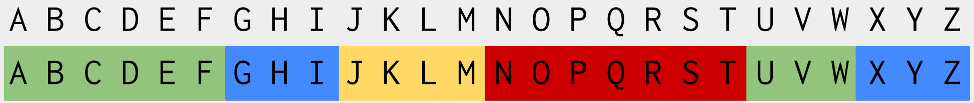
Equal Bins

- Sort by codepoint
- Divide into 100 equal size groups

...

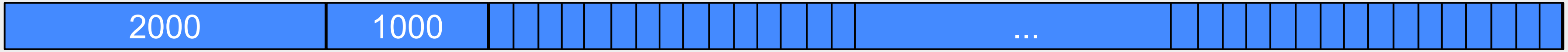
Sort by codepoint

Ways to segment



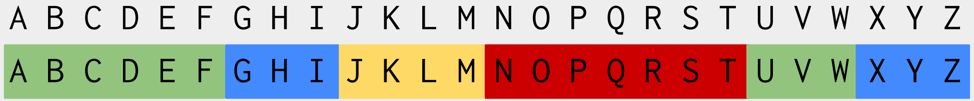
Frequency

- Sort by usage frequency
- Make 100 equal size groups



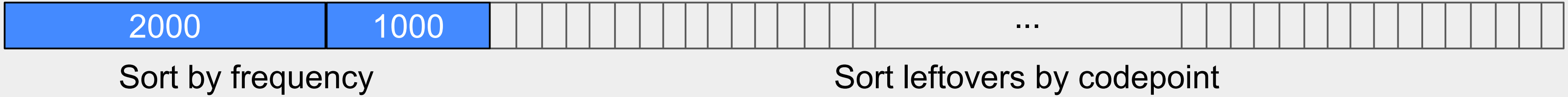
Sort by frequency

Ways to segment

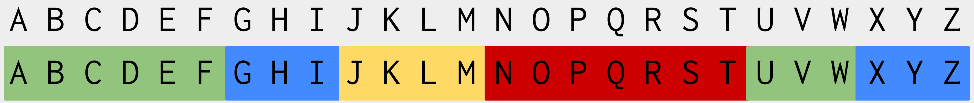


Hybrid sort

- Take two big blocks based on frequency
- Sort the rest by codepoint, make equal size groups

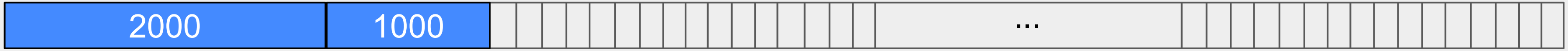


Ways to segment



Learned Sort

- Take two big blocks based on frequency
- Organize the rest using a topic model and a genetic algorithm



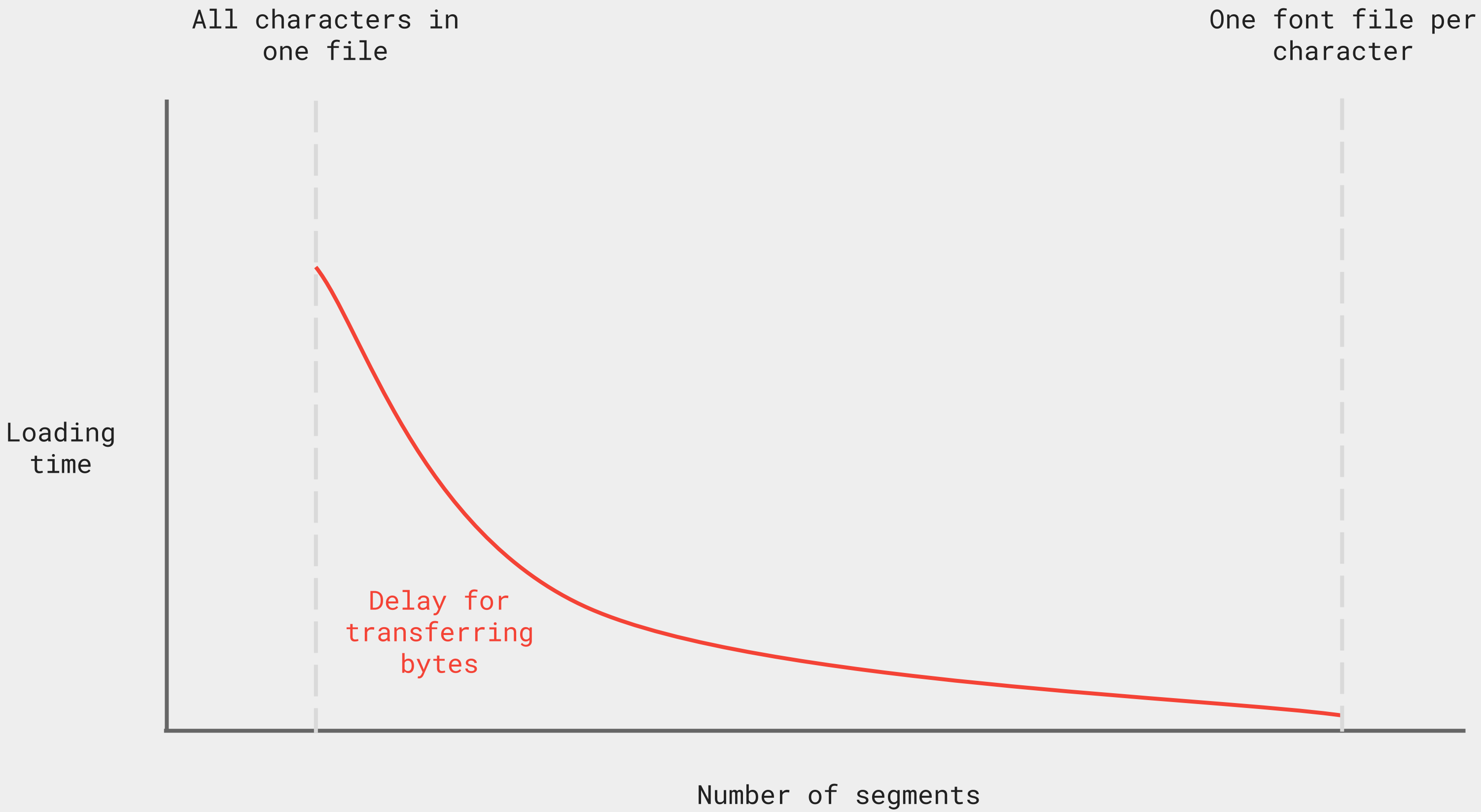
Sort by frequency

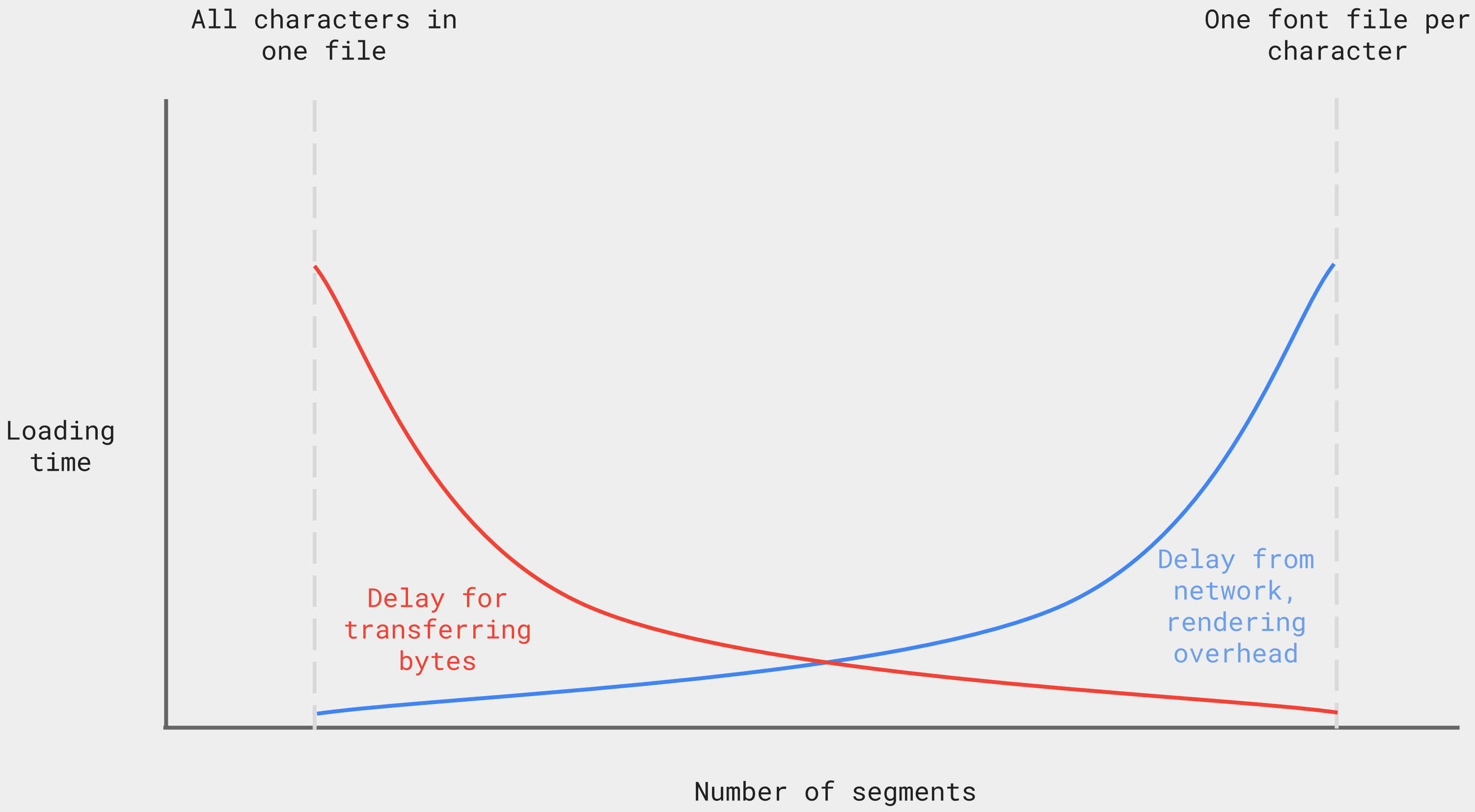
Sort by magic

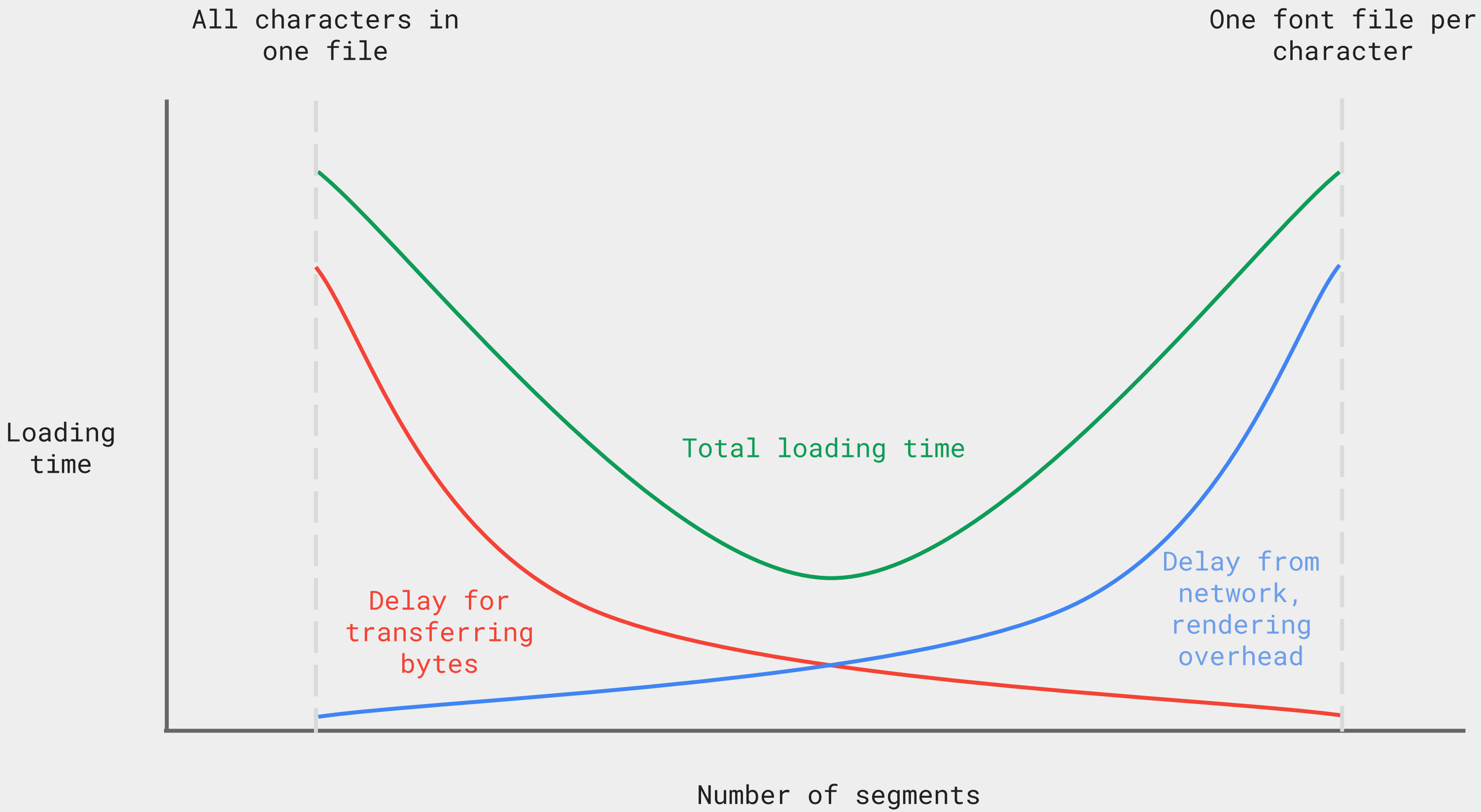
How do we compare segmentation strategies?

- Fewer bytes transferred is better
- Fewer fonts used is better

These two goals can be at odds with each other


All characters in one file
One font file per character
Loading time
Delay for transferring bytes
Number of segments


All characters in one file
One font file per character
Loading time
Delay for transferring bytes
Delay from network, rendering overhead
Number of segments


All characters in one file
One font file per character
Loading time
Total loading time
Delay for transferring bytes
Delay from network, rendering overhead
Number of segments

Test to find the best strategy

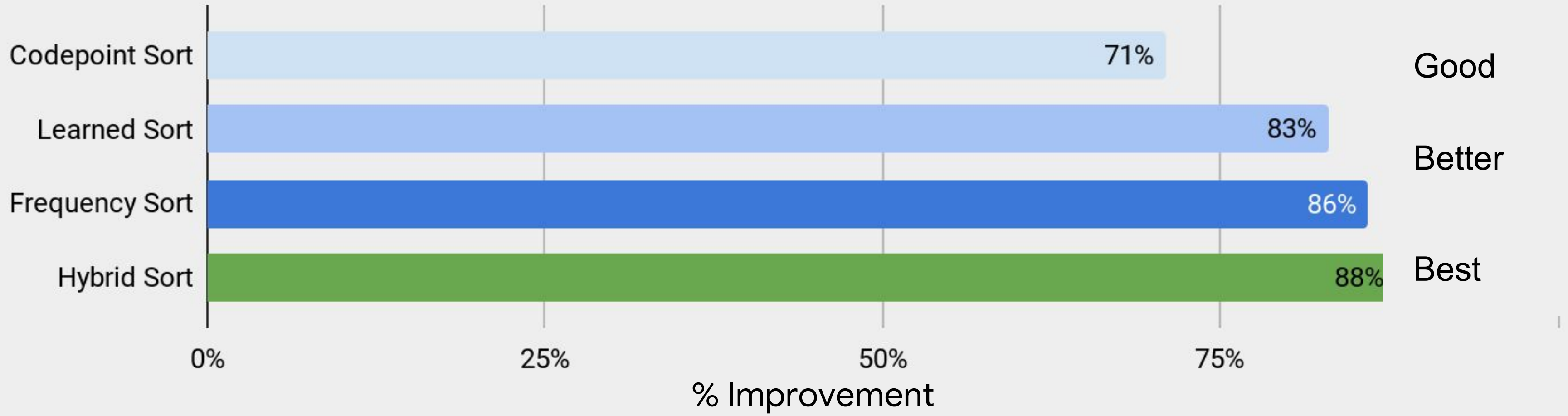
- Simulations and cost functions are fun
- Live testing is better
  - What if your cost function is wrong?
- We have N strategies we think might work
  - Send Y% of production traffic to each one
  - Let it run for a while
  - Measure and compare!
- Save >= 60% of bytes vs baseline to win

We don't always test our code
But when we do, we do it in production

Google

Results: **Hybrid Sort** is the winner

Saved from both font data and CSS (higher is better)



What we learned

Sometimes simpler is better

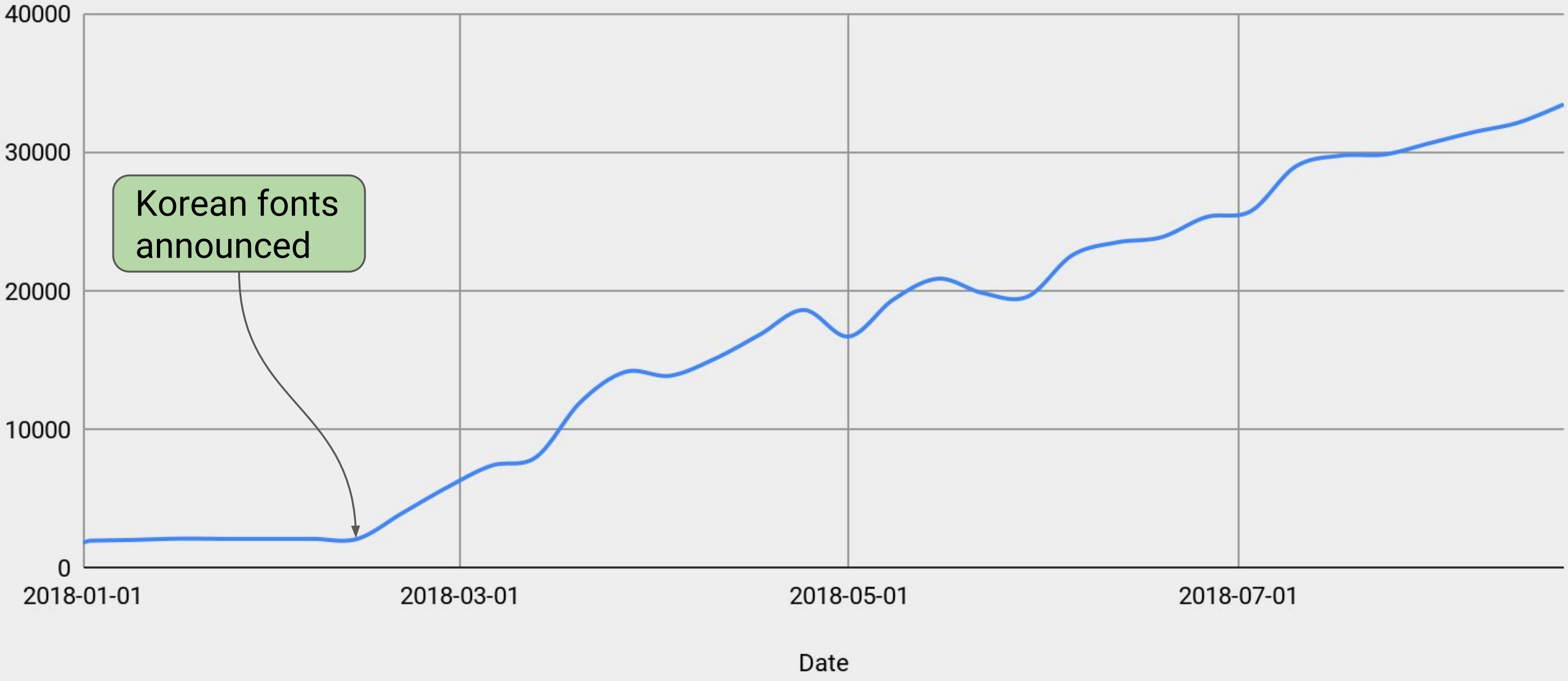
- Outperformed more complex strategies.

A few big segments + many smaller segments works surprisingly well

- Provides a good tradeoff between bytes transferred and number of font requests.

At this point we shipped Korean, much rejoicing :)

Korean adoption

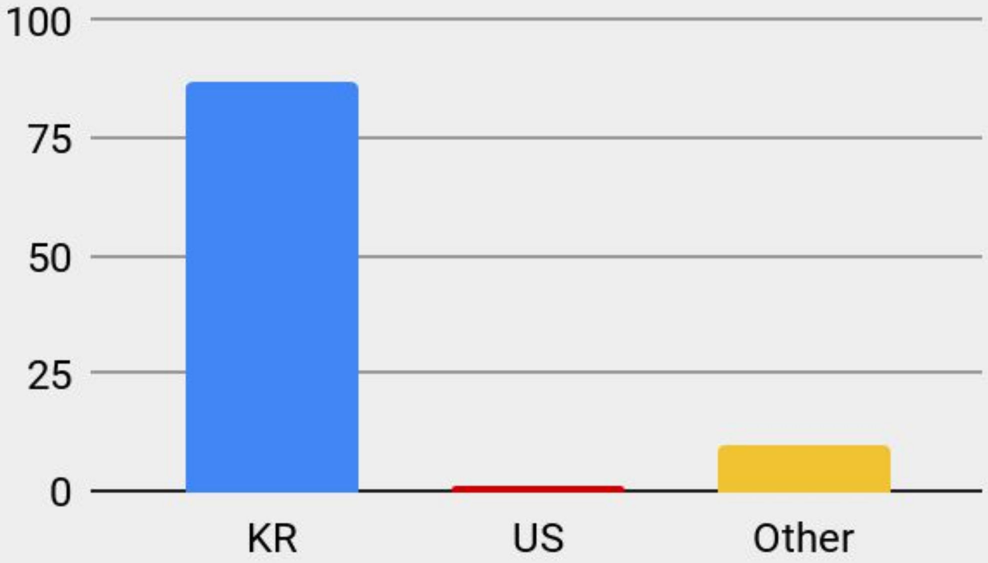


Korean usage

From fonts.google.com:

384_M

Number of times Google Fonts API served Nanum Gothic over the last week. Nanum Gothic is featured in more than 20,000 websites.



Google

Round 2, fight!

- Hybrid Sort is king of the hill
  - we can do better

How many high-frequency codepoints?

Is 3000 too many? Too few?

Consider

- More high-frequency subsets (HTTP/1 vs HTTP/2)
- Changing the boundary between high- and low-frequency codepoints

Keep

- Low-frequency characters ordered by codepoint … smaller CSS

Next batch of experiments

Adjust the high-frequency boundary: 1000, 2000

More aggressively subset high-frequency codepoints

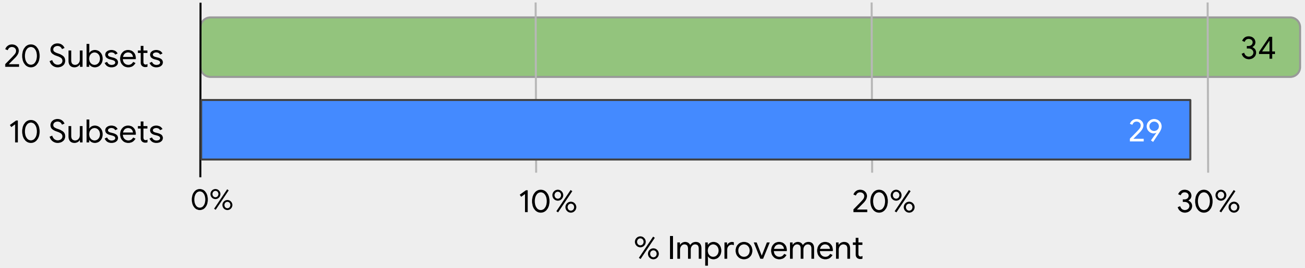
  10 and 20 subsets, rather than 2

Reminder: the baseline is Hybrid Sort

Spoiler alert: improved by 29 - 38% more … nice!

High frequency line at 3,000

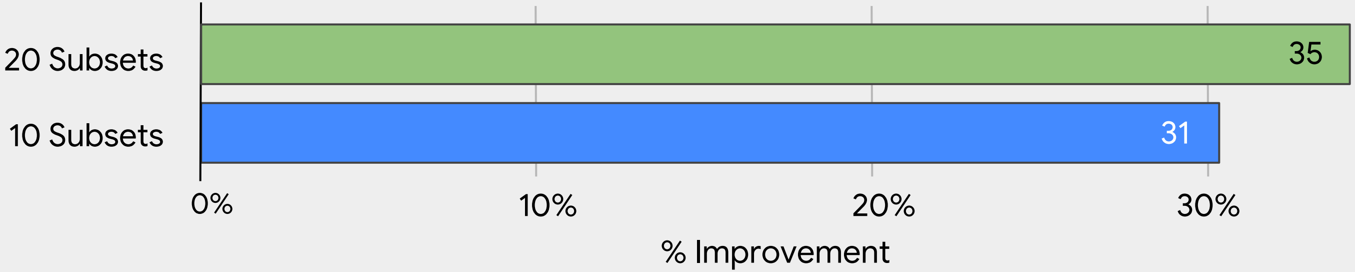
Experiments: 10 and 20 subsets



Conclusion: more high-frequency subsets performed better

High frequency line at 1,000

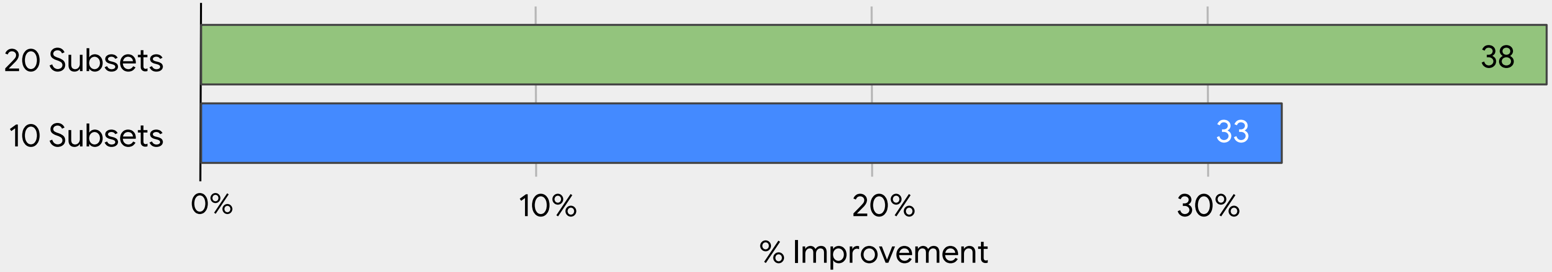
Experiments: 10 and 20 subsets



Conclusion: more high-frequency subsets performed better

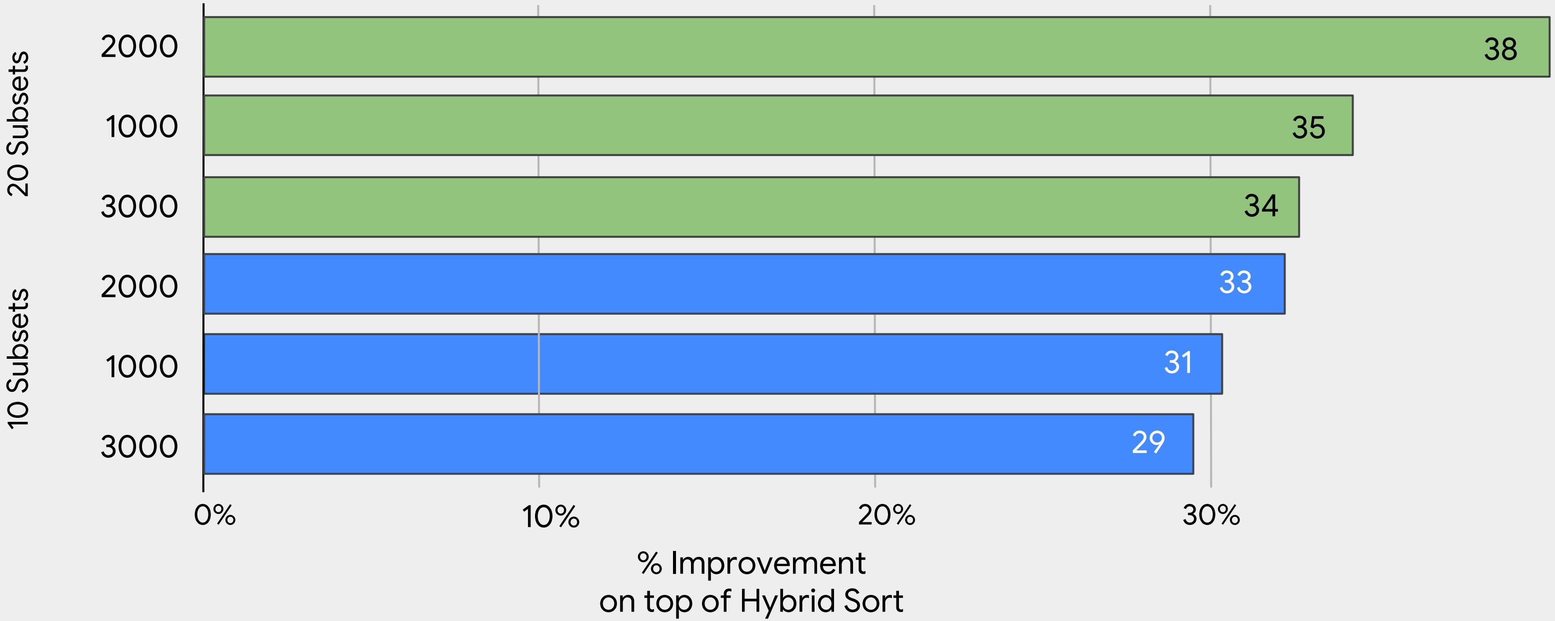
High frequency line at 2,000

Experiments: 10 and 20 subsets



Conclusion: more high-frequency subsets performed better

Overall results



Google

Conclusions for Korean fonts

Combining rounds 1 and 2, we’re saving ~93%

20 subsets always out-performed 10 subsets (regardless of the high-frequency line)

High-frequency line at 2000 performed best for both 10 and 20 subsets

- The best line appears to be somewhere between 1000 and 2000

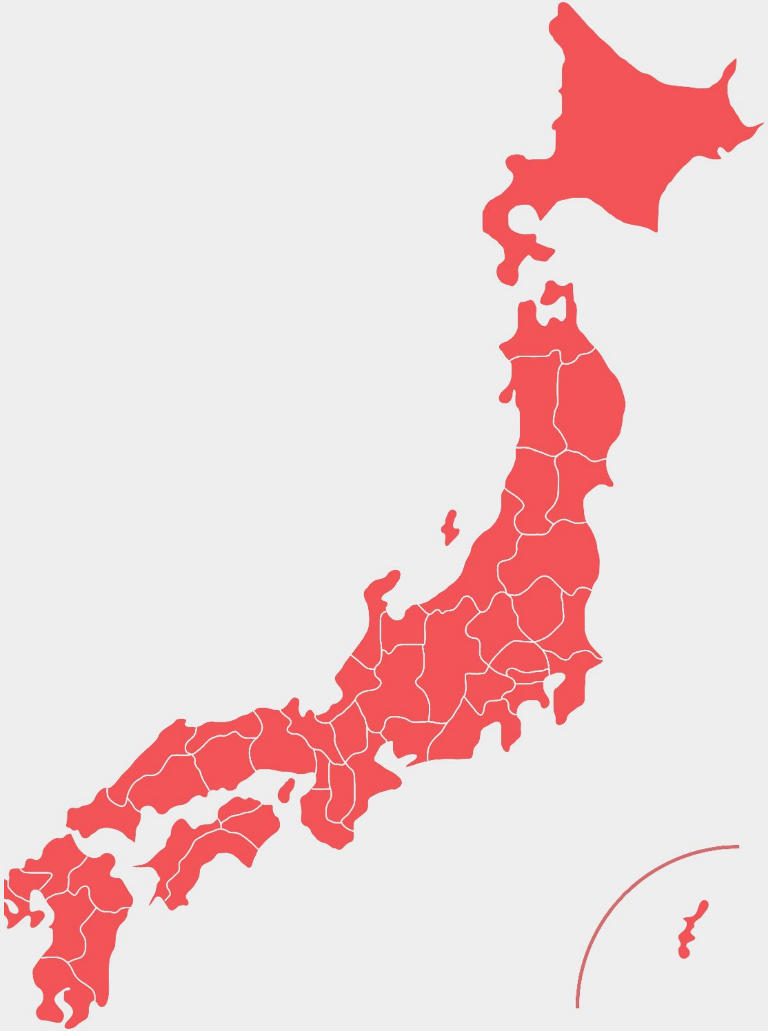
On to Japanese

Examined the character frequency

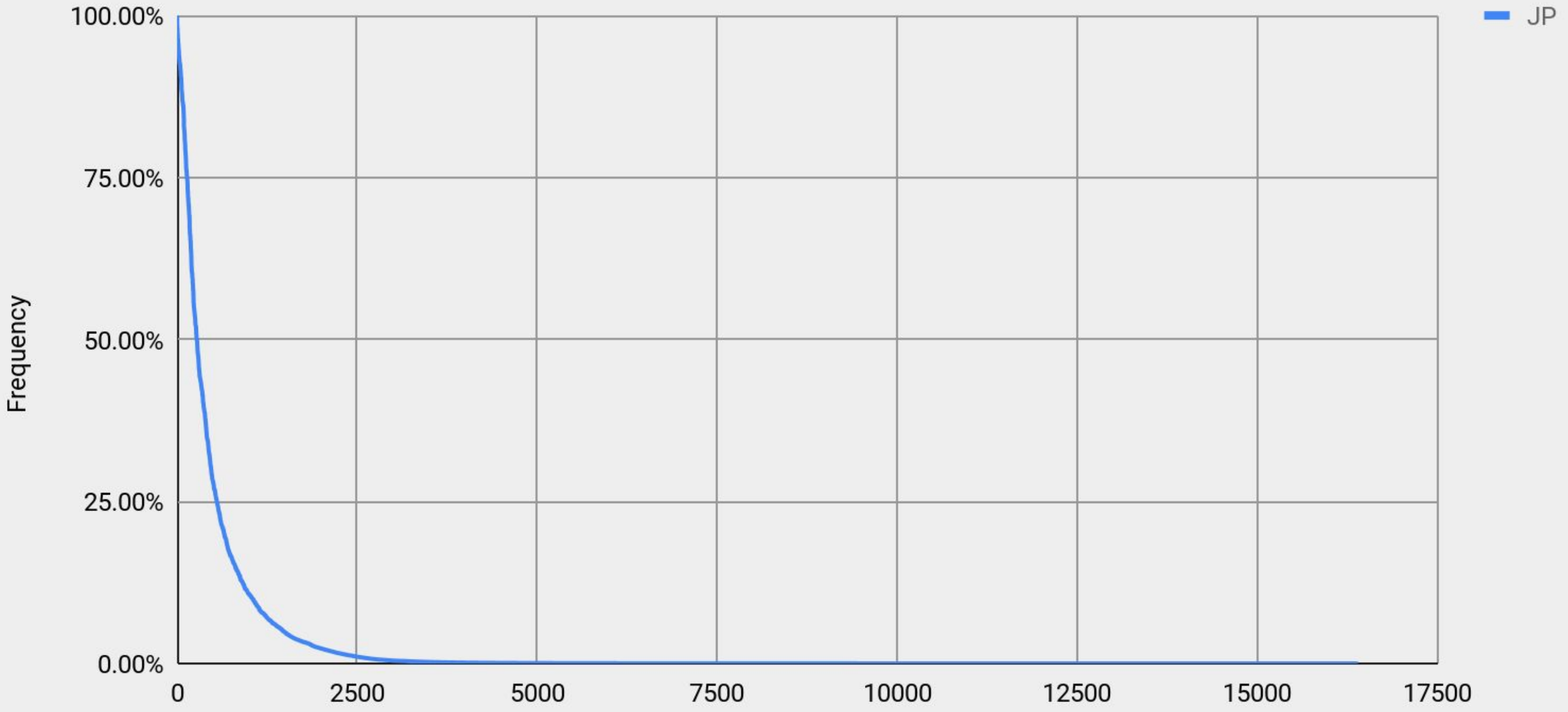
Ran similar tests

Reduced the bytes transferred by ~83%

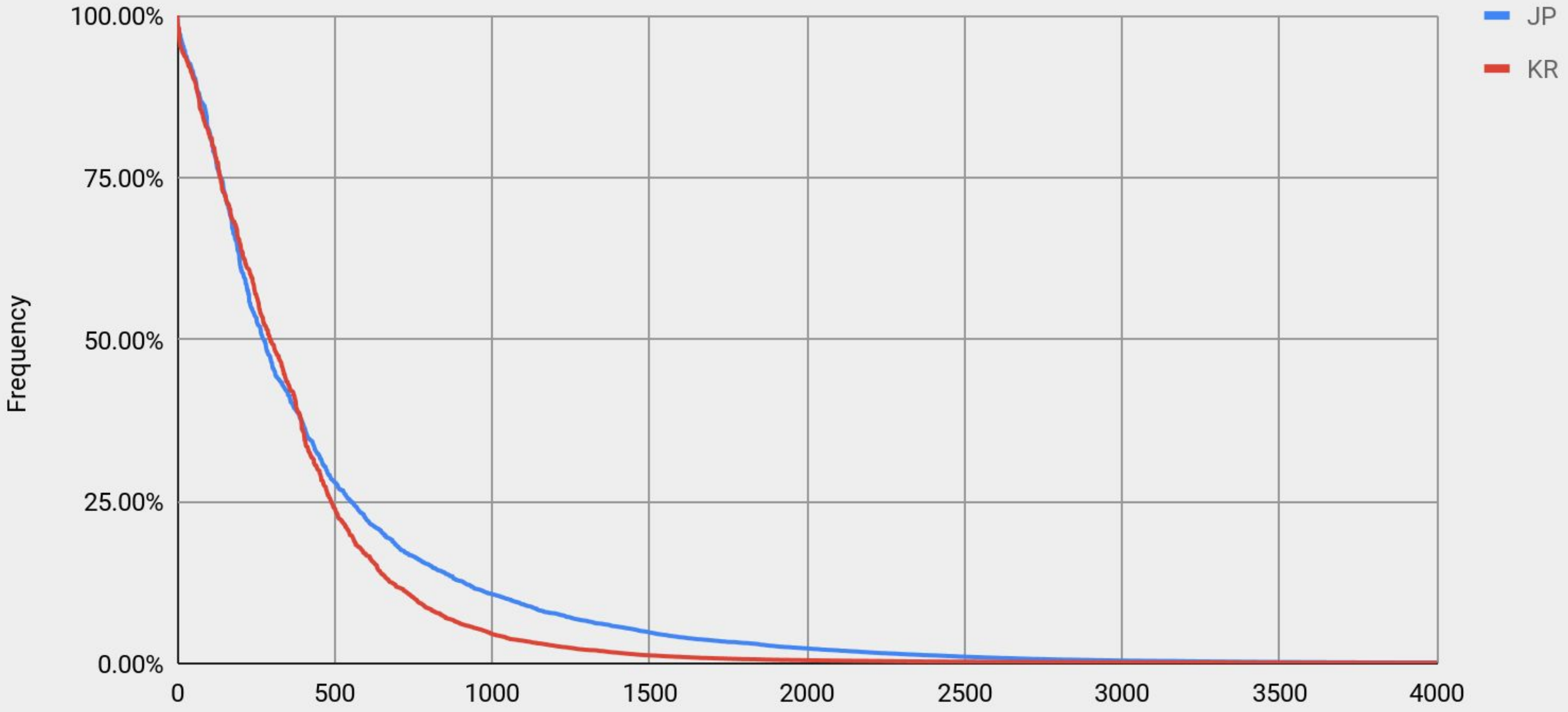
Compared to sending the whole font



Frequency of Japanese characters on the web



Japanese and Korean frequencies



Conclusions for Japanese fonts

We shipped Japanese!

Japanese characters are more complex than Korean

Japanese has more high-frequency characters

A high-frequency line of 3000 narrowly outperformed 2000

Room for improvement

Improve the preferred high-frequency line[1]

Refine the # of high-frequency subsets

Equal-sized high-frequency subsets may not be the best[1]

[1] Running these experiments right now

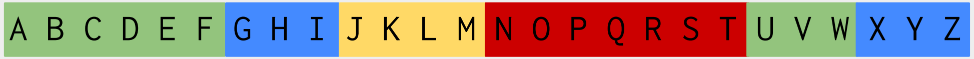
Room for improvement

Support Chinese, both simplified and traditional

使中文网络美丽而快速
使中文網絡美麗而快速

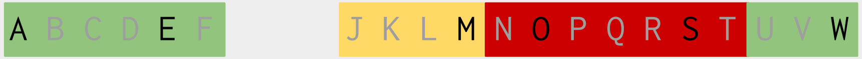
Trouble in paradise

Server: Hey buddy, here's my best guess at how I should cut up this font:



A B C D E F | G H I | J K L M | N O P Q R S T | U V W | X Y Z

Browser: I need to render "Awesome" so I'll take these pieces:



A B C D E F | J K L M | N O P Q R S T | U V W

Problems

- Layout features don't work across segments
  - AW won't kern
  - Indic, Arabic, and others rely **heavily** on layout
- We still downloaded a lot of stuff we didn't use

Transfer options

A) unicode-range
   - Cut font up, tell browser about pieces
   - This is what we do today
   - WAY better than whole font, but far from optimal
   - User often "pays" for unused codepoints

Transfer options

B) WOFF2 Deltas

- Each time user needs more codepoints, send a woff2 of the delta
- Layout will break horribly and unpredictably
- Not useful, but interesting for size comparison

Transfer options

C) Incremental Transfer

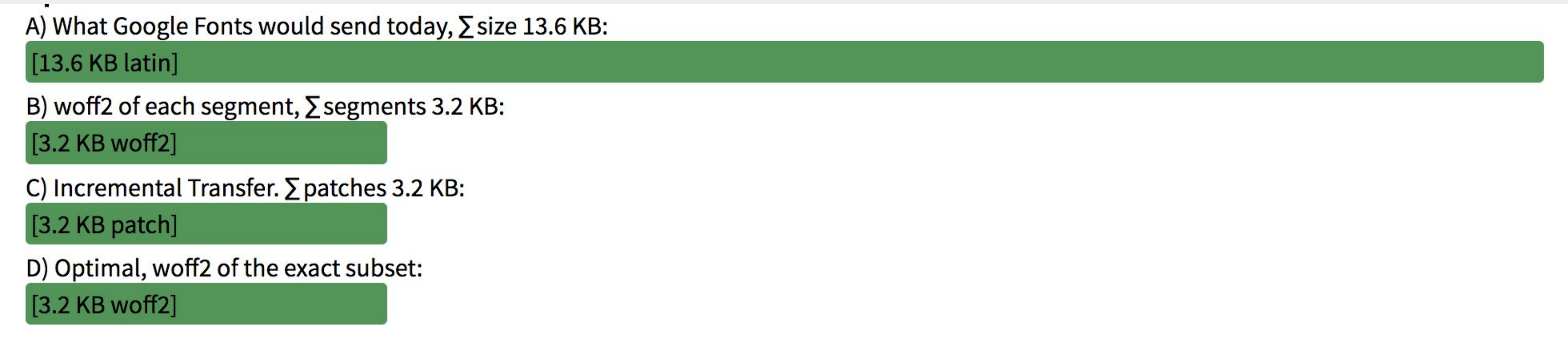
- ○ Each time user needs more codepoints, send a patch
- ○ Layout will just work
- ○ User doesn't "pay" for unused codepoints
- ○ Currently only achievable with Javascript

Transfer options

D) Send woff2 of exactly what user will need
  - Requires a priori knowledge of content user will view
    i) Currently impossible
  - Compute this as "optimal" when comparing other solutions

Trouble in paradise

Imagine we want to render "Awesome" in Montserrat



A) What Google Fonts would send today, ∑size 13.6 KB:

[13.6 KB latin]

B) woff2 of each segment, ∑segments 3.2 KB:

[3.2 KB woff2]

C) Incremental Transfer. ∑patches 3.2 KB:

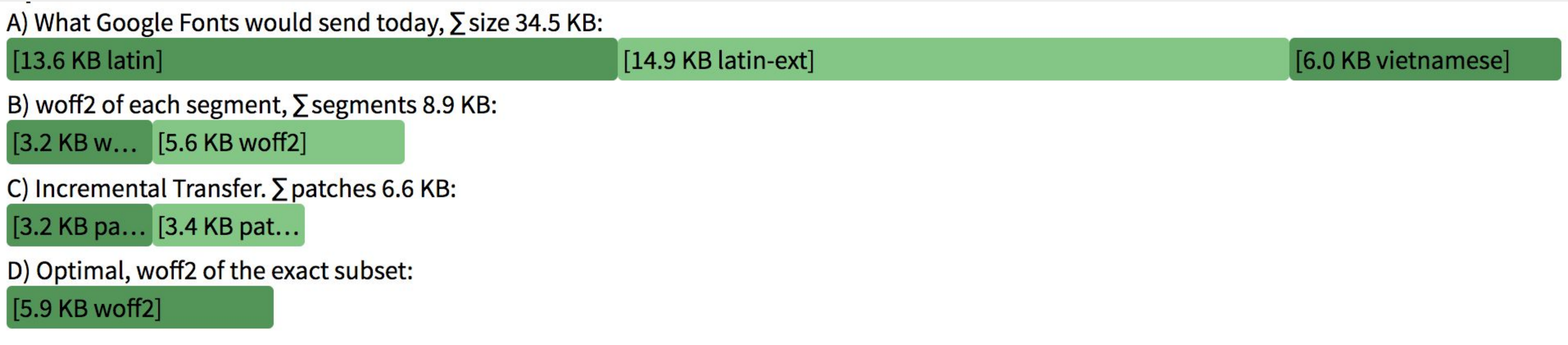
[3.2 KB patch]

D) Optimal, woff2 of the exact subset:

[3.2 KB woff2]

Trouble in paradise

Next we browse to a page that uses Vietnamese



A) What Google Fonts would send today, ∑size 34.5 KB:

[13.6 KB latin] [14.9 KB latin-ext] [6.0 KB vietnamese]

B) woff2 of each segment, ∑segments 8.9 KB:

[3.2 KB w… [5.6 KB woff2]

C) Incremental Transfer. ∑patches 6.6 KB:

[3.2 KB pa… [3.4 KB pat…

D) Optimal, woff2 of the exact subset:

[5.9 KB woff2]

Incremental Transfer: HTTP interaction

- HTTP is fine, no new protocol required :)

- Request
  - I have {current codepoints}
  - I need {desired codepoints}

- Response
  - binary patch to desired state
    - If {current codepoints} empty, a WOFF2 of desired state

- One cache key, update when you get a patch

Incremental Transfer: Subsetting

- Request
  - I have {current codepoints}
  - I need {desired codepoints}

- To compute patch

```
current = subset (current codepoints)
desired = subset (desired codepoints)
patch = binary patch (current, desired)
```

- What luck! We are building a fast subsetter :)
  - Faster Horse, https://github.com/harfbuzz/harfbuzz

Incremental Transfer: Patching

Page 1 "**Meows**"

M
e
o
s
w

Incremental Transfer: Patching

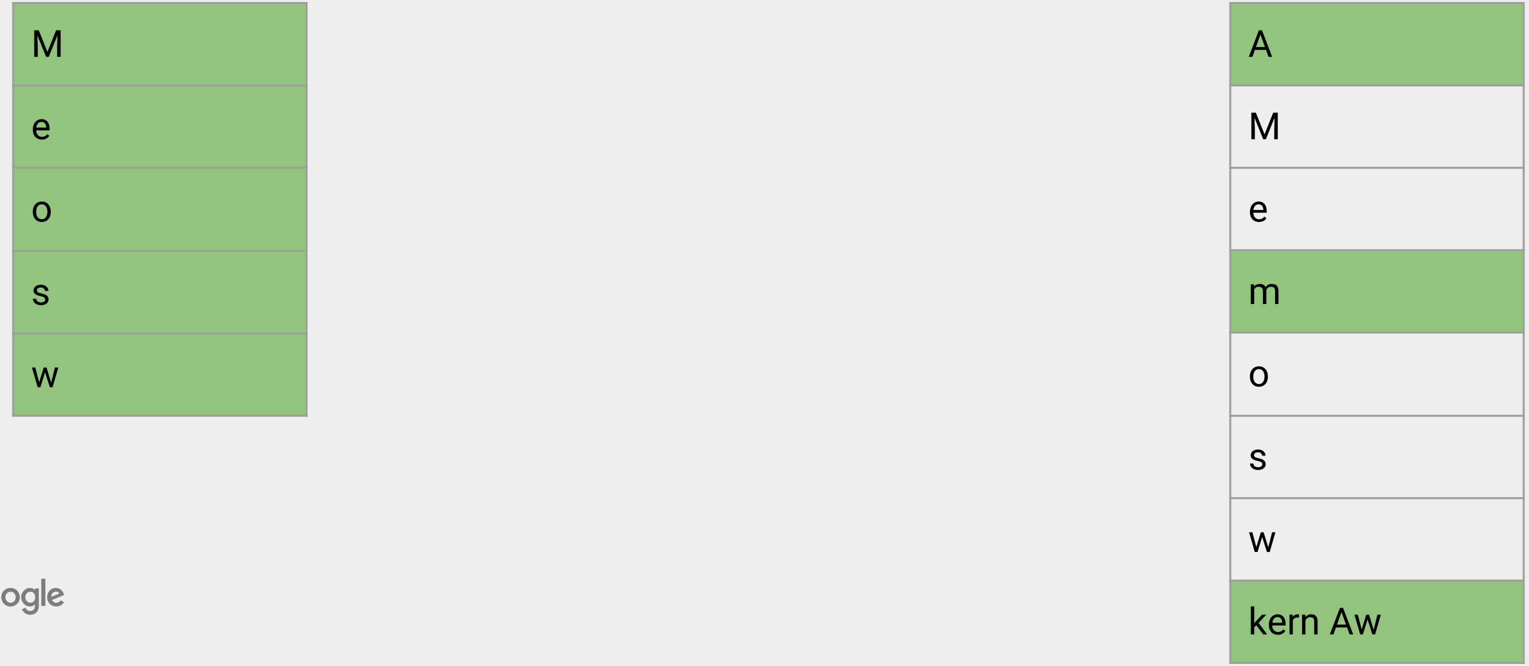
Page 1: "Meows"

🟩 M
🟩 e
🟩 o
🟩 s
🟩 w

Page 2: "Awesome"

🟩 A
M
e
🟩 m
o
s
w
🟩 kern Aw

Incremental Transfer: Patching



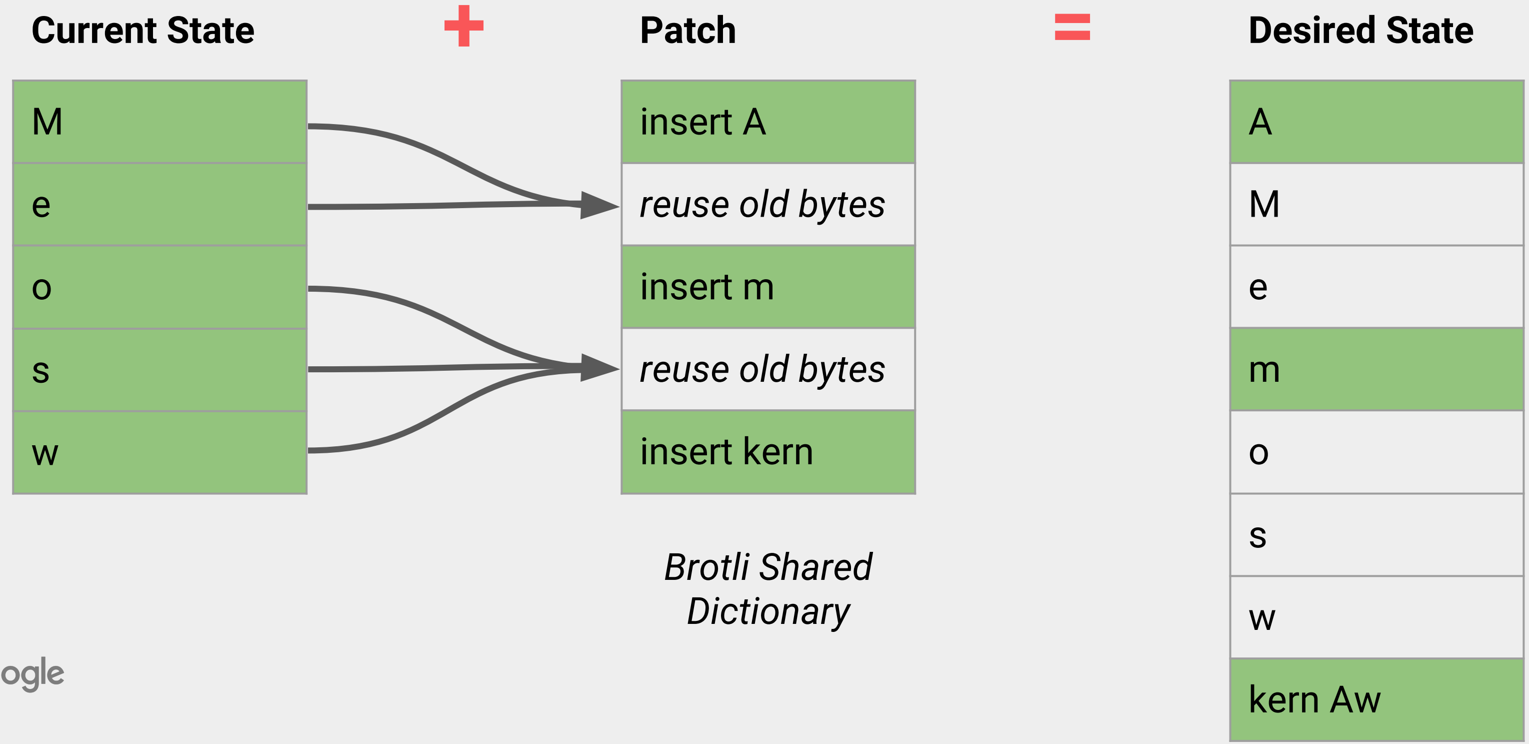
Current State

Current State
M
e
o
s
w

Desired State

Desired State
A
M
e
m
o
s
w
kern Aw

Incremental Transfer: Patching



Current State + **Patch** = **Desired State**

Current State:

- M
- e
- o
- s
- w

Patch:

- insert A
- *reuse old bytes*
- insert m
- *reuse old bytes*
- insert kern

Desired State:

- A
- M
- e
- m
- o
- s
- w
- kern Aw

Brotli Shared Dictionary

Let's standardize Incremental Transfer!

The Brotli library used by WOFF2 can handle patching

The hb-subset library can be used to do very fast subsetting

Subset <current codeponts> → current
Subset <desired codepoints> → desired
Compress desired using Brotli Shared Dictionary, dictionary = current

Proof of concept: tinyurl.com/incxfer-demo

The WebFonts Working Group is considering writing a standard :)

The End. Questions?

Speaker Evaluation

www.unicodeconference.org/eval-sp

References

fonts.google.com

(choose Japanese or Korean from Languages)

tinyurl.com/incxfer-demo