Improving Web Fonts Performance

Vitaly Friedman
05/06/2015 • Faenza, Italy
Vitaly Friedman, editor-in-chief and co-founder of SmashingMag
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012  Design, User Experience  28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.

In this article we’ll look at what Bach has to do with modern Web challenges — Particularly the challenge of designing for devices with diverse attributes and capabilities.

Bach And “The Well-Tempered Clavier”

In 1722, Bach put together a book of solo keyboard works intended as a collection of educational pieces for young musicians. The book contained 48 pieces — a prelude and fugue in every major and minor key. Now a staple of the Western canon, it’s regarded as one of the most important works in the history of Western music. He named the book The Well-Tempered Clavier.
A Little Story
The virtue of maps, they show what can be done with limited space, they foresee that everything can happen therein.

José Saramago, The Stone Raft
Many people in the design process simply don’t know a lot about of performance consequences of their design decisions.

— Brad Frost
Your Vodafone bill

Invoice for

used by modem

<table>
<thead>
<tr>
<th>Service charges</th>
<th>15 Feb to 14 Mar</th>
<th>Mobile Internet (live!) abroad</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage charges</td>
<td>up to 14 Feb</td>
<td>300.03</td>
<td></td>
</tr>
<tr>
<td>Total before VAT</td>
<td></td>
<td>300.03</td>
<td></td>
</tr>
<tr>
<td>VAT at 23.0% on 300.03</td>
<td></td>
<td>69.00</td>
<td></td>
</tr>
<tr>
<td>Total due</td>
<td></td>
<td>369.03</td>
<td></td>
</tr>
</tbody>
</table>

for information only: please pay using the account statement

e&oe
Daten-Eurotarif in Ländergruppe 1
1 KB genaue Abrechnung
automatische Beendigung der Internetverbindung nach 59,50 €. Mehr Infos unter Nutzungskontrolle.
nur im Ausland buchbar
0,53 €/MB

Daten-Auslandstarif in Ländergruppe 2
Abrechnung in 50-KB-Blöcken
automatische Beendigung der Internetverbindung nach 59,50 €. Mehr Infos unter Nutzungskontrolle.
nur im Ausland buchbar
0,49 €/50 KB

Daten-Auslandstarif in Ländergruppe 3
Abrechnung in 50-KB-Blöcken
automatische Beendigung der Internetverbindung nach 59,50 €. Mehr Infos unter Nutzungskontrolle.
nur im Ausland buchbar
Tarif gilt auch auf Schiffen und in Flugzeugen
0,79 €/50 KB

Hier finden Sie Informationen zu den Ländergruppen (PDF) und zu den Partnernetzen.

SO BUCHEN SIE IHREN TRAVEL & SURF PASS:
• T-Mobile roaming charges for loading the full front page of Vogue.co.uk, in Moscow: €93,13
Beautés fatales de 10 à 60 ans
Avec Vanessa Paradis, Cindy Crawford, Penelope Tree...
<table>
<thead>
<tr>
<th>Daten-Eurotarif in Ländergruppe 1</th>
<th>Daten-Auslandstarif in Ländergruppe 2</th>
<th>Daten-Auslandstarif in Ländergruppe 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 KB genaue Abrechnung</td>
<td>Abrechnung in 50-KB-Blöcken</td>
<td>Abrechnung in 50-KB-Blöcken</td>
</tr>
<tr>
<td>nur im Ausland buchbar</td>
<td>nur im Ausland buchbar</td>
<td>nur im Ausland buchbar</td>
</tr>
<tr>
<td></td>
<td>Tarif gilt auch auf Schiffen und in Flugzeugen</td>
<td></td>
</tr>
</tbody>
</table>

**Preise:**

<table>
<thead>
<tr>
<th></th>
<th>€/MB</th>
<th>€/50 KB</th>
<th>€/50 KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daten-Eurotarif</td>
<td>0,53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daten-Auslandstarif in Ländergruppe 2</td>
<td>0,49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daten-Auslandstarif in Ländergruppe 3</td>
<td></td>
<td>0,79</td>
<td></td>
</tr>
</tbody>
</table>

Hier finden Sie Informationen zu den Ländergruppen (PDF) und zu den Partnernetzen.

SO BUCHEN SIE IHREN TRAVEL & SURF PASS:
There is no difference for the user between a site being **down** and a site being **inaccessible** due to loading issues caused by blocking resources or slow networks.

— Andy Hume

“Real-Life Responsive Redesign”, SmashingConf 2013
State of Font Loading
text-overflow: ellipsis

overflow: hidden
Elegant Overflow with CSS Ellipsis

overflow: hidden

text-overflow: ellipsis
Elegant Overflow with CSS Ellipsis

Written by David Walsh on September 22, 2011 · 9 Comments


Overflow with text is always a big issue, especially in a programmatic environment. There's always only so much space but variable content to add into that space. I was recently working on a table for displaying user information and noticed that longer strings were breaking the table display. The obvious solution was adding an overflow: hidden setting to the table cells, but even then the text looked unnaturally cut off. The way to make text overflow elegant is with ellipses, and CSS' text-overflow property. Let's check it out!
Elegant Overflow with CSS Ellipsis


Overflow with text is always a big issue, especially in a programmatic environment. There's always only so much space but variable content to add into that space. I was recently working on a table for displaying user information and noticed that longer strings were breaking the table display. The obvious solution was adding an `overflow: hidden` setting to the table cells, but even then the text looked unnaturally cut off. The way to make text overflow elegant is with ellipses, and CSS' text-overflow property. Let's check it out!
Web Fonts Dilemma

The choice of formats depends on browser support:

- WOFF (Web Open Font Format)
- TTF (TrueType)
- OTF (OpenType)
- EOT (Embedded OpenType)
- SVG Fonts (Scalable Vector Graphics)
- WOFF2 (Web Open Font Format 2)
Web Fonts Dilemma

• **WOFF2** has the best compression, but isn’t supported by older Android/iOS. **WOFF** is.

• Old Android and iOS support **TTF** and **OTF**; Internet Explorer 6–8 needs **EOT**.

• **SVG** doesn’t support OpenType features. Not supported in IE, Chrome or Firefox.
Web Fonts Dilemma

- **WOFF2**
  supported by older Android/iOS.

- Old Android and iOS support
  Internet Explorer 6–8 needs

- **Strategy:** WOFF/2 with TTF/OTF and EOT for IE 6–8; not SVG. Best compression always wins.
Declaring @font-face

- We can use bulletproof @font-face syntax to avoid common traps along the way:

CSS:

```css
@font-face {
    font-family: 'Elena Regular';
    src: url('elena.eot?#iefix') format('embedded-opentype'),
         url('elena.woff2') format('woff2'),
         url('elena.woff') format('woff'),
         url('elena.otf') format('opentype');
}
```
Declaring @font-face

- If you want only **smart browsers (IE9+)** to download fonts, declaration can be shorter:

  **CSS:**

  ```
  @font-face {
      font-family: 'Elena Regular';
      src: url('elena.woff2') format('woff2'),
           url('elena.woff') format('woff'),
           url('elena.otf') format('opentype');
  }
  ```
• **CSS:**

```css
@font-face {
    font-family: 'Elena Regular';
    src: url('elena.woff2') format('woff2'),
        url('elena.woff') format('woff'),
        url('elena.otf') format('opentype');
}
```

• When a font family name is used in **CSS**, browsers match it against all [@font-face](https://developer.mozilla.org/en-US/docs/Web/CSS/@font-face) rules, download web fonts, display content.
When a font family name is used in CSS, browsers match it against all `@font-face` rules, download web fonts, display content.

CSS:

```css
body {
  font-family: 'Skolar Regular', AvenirNext, Avenir, /* iOS */
  'Roboto Slab', 'Droid Serif', /* Android */
  'Segoe UI', /* Microsoft */
  Georgia, 'Times New Roman', serif; /* Fallback */
}
```
• **CSS:**

```css
body {
    font-family: 'Skolar Regular', AvenirNext, Avenir, /* iOS */
              'Roboto Slab', 'Droid Serif', /* Android */
              'Segoe UI', /* Microsoft */
              Georgia, 'Times New Roman', serif; /* Fallback */
}
```

• **HTML:**

```html
<link href='http://fonts.googleapis.com/css?family=Skolar.Regular' rel='stylesheet' type='text/css'>

<script type='text/javascript'
         src='//use.typekit.net/tbb3uid.js'></script>
<script type='text/javascript'>try{Typekit.load();}catch(e){}</script>
```
• Once DOM and CSSOM are constructed, if @font-face matches, a font will be required.

• If fonts aren’t cached yet, they will be requested, downloaded and applied, deferring rendering.
overflow: hidden

text-overflow: ellipsis
• **FOIT** (*Flash Of Invisible Text*): no content displayed until the font becomes available.

• **FOUT** (*Flash Of Unstyled Text*): show content in fallback fonts first, then switch to web fonts.

<table>
<thead>
<tr>
<th>Font loading</th>
<th>IE8</th>
<th>IE9</th>
<th>IE10</th>
<th>IE11</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Safari</th>
<th>Safari (iOS)</th>
<th>Opera</th>
<th>Android WebKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font loading</td>
<td>FOUT</td>
<td>FOUT</td>
<td>FOUT</td>
<td>FOUT</td>
<td>FOIT</td>
<td>FOIT</td>
<td>FOIT</td>
<td>FOIT</td>
<td>FOIT</td>
<td>FOIT</td>
</tr>
<tr>
<td>Timeout</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>3 sec.</td>
<td>3 sec.</td>
<td>∞</td>
<td>∞</td>
<td>3 sec.</td>
<td>∞</td>
</tr>
</tbody>
</table>
Web Fonts Rendering

• If we request fonts directly, users might be getting a FOIT, then FOUT, then web fonts.

• *Solution:* prioritize content over presentation. Display content right away, swap fonts later:
Web Fonts Rendering

• **Solution:** prioritize content over presentation. Display content right away, swap fonts later:

  • First visit: show content in fallback fonts *fast,*
  • Load web fonts *async* during rendering,
  • Cache web fonts properly (opt. localStorage),
  • Apply web fonts right away, or on later visits,
  • Next visits: apply web fonts *only* if they’re cached.
The Guardian Redesign (2013)

- **Project goals focused on mobile experience:**
  - Tackle dropping print circulation with mobile,
  - Replace the slow, rigid mobile/desktop sites,
  - First focus on “mobile” experience,
  - Long term: new RWD client-side architecture,
  - Ultimate goal: one code base, one responsive site.

- **Solution:** a mobile-first “stealth” redesign with a strong focus on progressive enhancement.
The architecture

IPHONE
ANDROID
WINDOWS PHONE

RESPONSIVE
WEBSITE

IPAD APP

READ API

CONTENT

WRITE API

CMS TOOLS

EDITORIAL TOOLS
The Guardian Redesign

- **Priority lists** for content and styles to define “core”:
  - Core content doesn’t rely on JavaScript,
  - Only one main feature image considered “core”,
  - No Ajax support for ratings, comments and live scores,
  - “Cutting the mustard” to “buckle” browsers,
  - Web fonts aren’t loaded by default (prevent FOUT).
The Guardian’s Modular Load

- Consider **at least** three types of page content:
  - Core content
    (essential HTML+CSS, usable non-JS enhanced experience);
  - Enhancement
    (JS, Geolocation, touch, enhanced CSS, Web fonts, widgets);
  - Leftovers
    (analytics, advertising, third-party content).

- Idea: load Core content first, then Enhancement on `DOMContentLoaded`, then Leftovers on `Load`. 
The Guardian’s Modular Load

• Load JS into a browser *asynchronously*. While JS is being downloaded, browser still can parse the document and show content.

• HTML/JS (for modern browsers):

```javascript
@if(isModernBrowser) {
    <script src="enhanced.js" async defer></script>
}
```
SCRIPT LOADING

appendChild(script)

<script async>

HTML

CSS

JAVASCRIPT

HTML

CSS

JAVASCRIPT
BBC’s isModernBrowser()

• We can use server-side device detection using UA strings with DeviceAtlas, WURFL etc.

• We can use client-side feature detection to split browsers into groups “HTML4” / “HTML5”.

• JS:

```javascript
if (
    'querySelector' in document &&
    'localStorage' in window &&
    'addEventListener' in window
) {
    // HTML5 browser detected; load the JS app
}
```
BBC’s isModernBrowser()

- **JS:**
  ```javascript
  if ( 
    'querySelector' in document &&
    'localStorage' in window &&
    'addEventListener' in window ) {
    // HTML5 browser detected; load the JS app
  }
  ```

- **HTML5 Browsers:**
  - IE9+, FF 3.5+, Opera 9+,
  - Safari 4+, Chrome 1+, iOS1+,
  - Android phone and tablets 2.1+,
  - Blackberry OS6+, Win 7.5+,
  - Mobile Firefox, Opera Mobile

- **HTML4 Browsers:**
  - IE8-, Blackberry OS5-,
  - Nokia S60 v6-, Nokia S40,
  - Symbian, Windows 7 Phone
  - (pre-Mango), a plethora of legacy devices.
**BBC’s isModernBrowser()**

- **JS:**
  ```javascript
  if ('visibilityState' in document) {
      // HTML5 browser detected; load the JS app
  }
  ```

- **HTML5 Browsers:**
  IE9+, FF 3.5+, Opera 9+, Safari 4+, Chrome 1+, iOS1+, Android phone and tablets 2.1+, Blackberry OS6+, Win 7.5+, Mobile Firefox, Opera Mobile

- **HTML4 Browsers:**
  IE8-, Blackberry OS5-, Nokia S60 v6-, Nokia S40, Symbian, Windows 7 Phone (pre-Mango), a plethora of legacy devices.
## Page Visibility

JavaScript API for determining whether a document is visible on the display

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global</td>
<td>unprefixed</td>
<td>85.77%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>75.16%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Browser</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox</td>
<td>31</td>
<td>31</td>
<td>36</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>Chrome</td>
<td>31</td>
<td>36</td>
<td>37</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>Safari</td>
<td>31</td>
<td>36</td>
<td>37</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>Opera</td>
<td>31</td>
<td>36</td>
<td>37</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>7</td>
<td>7</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>7</td>
<td>7</td>
<td>7.1</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.1</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4.4</td>
<td>4.4.4</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>4.1</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4.4</td>
<td>4.4.4</td>
</tr>
</tbody>
</table>

### Notes
No notes
PRE-RENDER

- **CUT THE MUSTARD?**
  - NO
    - **SUPPORT WOFF?**
      - NO
        - **FONTS IN STORAGE?**
          - NO
            - NO FONTS
          - **SHOW FONTS**
POST-RENDER

STORAGE AVAILABLE?

NO

DOWNLOAD FONTS: BASE64 ENCODED IN JSON

CACHE FONTS IN STORAGE

NO FONTS

SHOW FONTS
PageSpeed Insights

http://www.theguardian.com/uk

Mobile 88 / 100
Desktop 69 / 100

Speed

⚠️ Consider Fixing:
Prioritize visible content
  Show how to fix

Leverage browser caching
  Show how to fix

Minify HTML
  Show how to fix

✔️ 7 Passed Rules
  Show details

User Experience  BETA
Not currently a part of the overall score
Source for the Guardian's responsive site, coming soon to a desktop near you...
http://www.theguardian.com/uk?view=mobile

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>Remove heading gap</td>
<td>16 hours ago</td>
</tr>
<tr>
<td>applications</td>
<td>JsonNotFound now understands CORS</td>
<td>11 days ago</td>
</tr>
<tr>
<td>article</td>
<td>Added an AB test to try and prove that if we remove editorial linking...</td>
<td>a day ago</td>
</tr>
<tr>
<td>commercial</td>
<td>Revert &quot;Merge pull request #2625 from guardian/article-commercial-tests&quot;</td>
<td>12 days ago</td>
</tr>
<tr>
<td>common</td>
<td>Merge pull request #2655 from guardian/ab-test-page-features</td>
<td>19 hours ago</td>
</tr>
<tr>
<td>core-navigation</td>
<td>Merge pull request #2637 from guardian/html-most-popular</td>
<td>5 days ago</td>
</tr>
<tr>
<td>data</td>
<td>Cache test lookup results</td>
<td>17 days ago</td>
</tr>
<tr>
<td>dev-build</td>
<td>remove old routes</td>
<td>a day ago</td>
</tr>
<tr>
<td>dev</td>
<td>Delete sbt-launch-0.12.2.jar</td>
<td>3 months ago</td>
</tr>
<tr>
<td>diagnostics</td>
<td>Added a porter job to work out percentage conversions of Omniture and...</td>
<td>18 days ago</td>
</tr>
</tbody>
</table>

You can clone with HTTPS, SSH, or Subversion.
Targeting Mobile Users Through Google AdWords

By Tim Jensen

August 14th, 2014

Advertising, Business, Devices

0 Comments Edit

“Know your audience” has stood as a fundamental marketing principle since long before the web. When advertising online, you need to take into account one of the most basic factors of the audience you are reaching: what devices they are using.

Nothing is more frustrating than stubborn management. That’s why we published Digital Adaptation, a new book by Paul Boag on how to help management overcome legacy practices — for good.

It’s done. The Smashing Book #4, our brand new book with smart front-end
SmashingMag’s Modular Load

• Consider **three** types of page content:
  
  • Core content
    (essential HTML+CSS, usable non-JS enhanced experience);

  • Enhancement
    (JS, syntax highlighter, full CSS, Web fonts, comments);

  • Leftovers
    (analytics, advertising, Gravatar images).

• **Idea:** load Core content first, then Enhancement on `DOMContentLoaded`, then Leftovers on `Load`. 
SmashingMag’s Optimization

- Minor optimizations based on a simple principle: optimize content, *defer the rest.*
  - Load critical CSS inline and full CSS on load,
  - *Avoid JavaScript libraries* (jQuery → JavaScript),
  - Store Web fonts in localStorage + cookies,
  - Defer advertising, tracking and all non-critical CSS/JS,
  - Replaced Respond.js with IE8 stylesheet (fixed-width).
  - Optimize the *critical rendering path* for content delivery.
localFont

Implement localStorage web font caching in seconds

Drop your font family files here (*.woff, *.woff2 or *.ttf)
PageSpeed Insights

http://www.smashingmagazine.com

98 / 100 Speed

Consider Fixing:
Reduce server response time
- Show how to fix
Optimize images
- Show how to fix

8 Passed Rules
- Show details
Web Page Performance Test for
www.smashingmagazine.com

From: Amsterdam, NL - Chrome - Cable
8/28/2014 10:57:54 PM

Summary

<table>
<thead>
<tr>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>Speed Index</th>
<th>DOM Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>First View</td>
<td>2.312s</td>
<td>0.582s</td>
<td>1.004s</td>
<td>1443</td>
</tr>
<tr>
<td>Repeat View</td>
<td>1.685s</td>
<td>0.598s</td>
<td>0.993s</td>
<td>1100</td>
</tr>
</tbody>
</table>

Document Complete

<table>
<thead>
<tr>
<th>Time</th>
<th>Requests</th>
<th>Bytes In</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.312s</td>
<td>47</td>
<td>425 KB</td>
</tr>
<tr>
<td>1.685s</td>
<td>8</td>
<td>49 KB</td>
</tr>
</tbody>
</table>

Fully Loaded

<table>
<thead>
<tr>
<th>Time</th>
<th>Requests</th>
<th>Bytes In</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.159s</td>
<td>51</td>
<td>924 KB</td>
</tr>
<tr>
<td>1.685s</td>
<td>8</td>
<td>49 KB</td>
</tr>
</tbody>
</table>

Waterfall

Screen Shot

Video
Web Page Performance Test for
www.smashingmagazine.com

From: San Francisco, CA USA - Chrome - Cable
5/20/2015, 4:46:59 PM

Summary
Tester: i-1ccc34d7
Test runs: 5
Re-run the test

Performance Results (Median Run)

<table>
<thead>
<tr>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>Speed Index</th>
<th>DOM Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>First View (Run 2)</td>
<td>2.687s</td>
<td>0.301s</td>
<td>0.497s</td>
<td>693</td>
</tr>
<tr>
<td>Repeat View (Run 4)</td>
<td>1.495s</td>
<td>0.757s</td>
<td>0.213s</td>
<td>528</td>
</tr>
</tbody>
</table>

Document Complete

<table>
<thead>
<tr>
<th>Fully Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
</tr>
<tr>
<td>2.687s</td>
</tr>
<tr>
<td>3.454s</td>
</tr>
</tbody>
</table>

Test Results

Run 1:

Waterfall

Screen Shot

Video
Smart CSS Font Fallback

- With font delivery services, *Web Font Loader* gives you a granular control over font loading.

- **HTML/CSS:**

  ```html
  <html class="wf-loading">
  .wf-loading    // fonts start to load
  .wf-active     // all (or some) fonts have loaded
  .wf-inactive   // all fonts failed to load
  ```
Smart CSS Font Fallback

• We can also use CSS to *tweak font fallback metrics* to reduce the jump between the default font and the Web font...

• ...and add sensible “mobile” OS fonts into a fallback font stack and adjust their styling.

• **iOS 6.1:**
  Avenir, Avenir Next Condensed, Baskerville, Helvetica, Hoefler Text, Palatino, Optima.

• **Win Phone 7:**
Smart CSS Font Fallback

• We can also use CSS to *tweak font fallback metrics* to reduce the jump between the default font and the Web font...

• ...and add sensible “mobile” OS fonts into a fallback font stack and adjust their styling.

• **Win Phone 7:**

• **Android 4:**
  Droid Sans, Droid Sans Mono, Droid Serif, Roboto.
Smart CSS Font Fallback

• ...and add sensible “mobile” OS fonts into a fallback font stack and adjust their styling.

• CSS:

```css
.wf-loading h1 {
  font-family: 'Tablet Gothic Condensed',
              'HelveticaNeue-CondensedBlack',
              'Helvetica Neue',
              'Segoe UI', 'Roboto', sans-serif;
  font-weight: 800;
  font-stretch: condensed;
}
```
WHAT LIES BENEATH

INTERNATIONAL CONFERENCE ON TYPOGRAPHY

Kerning is the first international conference in Italy dedicated solely to typography and web typography.

On June 3rd and 4th (only workshops) and 5th (conference day) 2015 we will gather in Faenza, in the very heart of Italy, top notch speakers from the whole world of digital and web typography to create an unique event.

Kerning features two days of workshops (June 3rd and 4th) and a Conference day full of talks (June 5th). Kerning offers a very interesting chance to meet developers, gurus, managers and innovators from all over the world.

Kerning is a non-profit event. We want to help grow a community of prepared developers keeping knowledge affordable: that’s why full ticket costs just Eur. 195.00 (VAT included).

CALLING ALL TYPE LOVERS
Dealing With Faux Fonts

• If browsers find a match for family name, but not the required variation (e.g. bold), they will attempt to *generate* the required variation.

• *Solution:* use unique font-family names, and set styles to match those in @font-face declarations.
• **Solution:** use unique font-family names, and set styles to match those in `@font-face` declarations.

• **CSS:**

```css
.elena-reg {
    font-family: 'Elena Regular', sans-serif;
    font-weight: 400;
    font-style: normal;
}

.elena-italic {
    font-family: 'Elena Italic', sans-serif;
    font-weight: 400;
    font-style: italic;
}

.elena-bold {
    font-family: 'Elena Bold', sans-serif;
    font-weight: 700;
    font-style: normal;
}
```
• **Solution:** use unique font-family names, and set styles to match those in @font-face declarations.

• **CSS:**

```css
strong {
  font-family: 'Elena Bold', sans-serif;
  font-weight: 700;
  font-style: normal;
}

em {
  font-family: 'Elena Italic', sans-serif;
  font-weight: 400;
  font-style: italic;
}

em.light {
  font-family: 'Elena Light Italic', sans-serif;
  font-weight: 300;
  font-style: italic;
}
```
Setting Weights And Styles With The @font-face Declaration

By Laura Franz

If people are on your website, they’re probably either skimming quickly, looking for something, or they’ve found what they’re looking for and want to read it as easily as possible. Either way, keeping text readable will help them achieve their goal.

BOLD AND ITALIC HELP TO ORGANIZE CONTENT

A few months ago, I wrote an article on “Avoiding Faux Weights and Styles with Google Web Fonts.” I ended the article by showing that weights and styles are an important UX element when setting text. Bold and italic forms of a font help people to skim your website. They add emphasis — both strong and subtle — that can help visitors understand the organization of content before even starting to read it.
Improving Smashing Magazine’s Performance: A Case Study

By Vitaly Friedman

Today Smashing Magazine turns eight years old. Eight years is a long time on the web, yet for us it really doesn’t feel like a long journey at all. Things have changed, evolved and moved on, and we gratefully take on new challenges one at a time. To mark this special little day, we’d love to share a few things that we’ve learned over the last year about the performance challenges of this very website and about the work we’ve
SmashingMag is the only site I can read on an EDGE connection when commuting in São Paulo, every day. It makes me think why other sites aren’t optimizing for slow connections, too.

— a reader from Brazil
CSS Font Loading API

- Native browser API à la Web Font Loader, with a FontFace object representing @font-face rules.

- JavaScript:

```javascript
var elena_reg = new FontFace(  
    'Elena Regular',  
    'url(elena_reg.woff) format("woff"),', +  
    'url(elena_reg.otf) format("otf")',  
    { weight: 'regular', unicodeRange: 'U+0-7ff' }  
);```
• **JavaScript:**

```javascript
var elena_reg = new FontFace(
    'Elena Regular',
    'url(elena_reg.woff) format("woff"),
    'url(elena_reg.otf) format("otf"),
    { weight: 'regular', unicodeRange: 'U+0-7ff' }
);
```

• **JavaScript:**

```javascript
document.fonts.load('1em elena_reg')
    .then(function() {
        var docEl = document.documentElement;
        docEl.className += ' elena_reg-loaded';
    }).catch(function () {
        var docEl = document.documentElement;
        docEl.className += ' elena_reg-failed';
    });
```
**JavaScript:**

```javascript
document.fonts.load('1em elena_reg')
.then(function() {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-loaded';
}).catch(function () {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-failed';
});
```

**CSS:**

```css
.elena_reg-loaded h1 {
    font-family: "Elena Regular";
}
```
• **JavaScript:**

```javascript
document.fonts.load('1em elena_reg')
.then(function() {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-loaded';
}).catch(function () {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-failed';
});
```

• **CSS:**

```css
.elena_reg-loaded h1 {
    font-family: "Elena Regular";
    font-rendering: "block 0s swap infinite"; // FOUT
    // font-rendering: "block 3s swap infinite"; // FOIT
}
```
• **JavaScript:**

```javascript
document.fonts.load('1em elena_reg').then(function() {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-loaded';
}).catch(function () {
    var docEl = document.documentElement;
    docEl.className += ' elena_reg-failed';
});
```

• **CSS:**

```css
.elena_reg-loaded h1 {
    font-family: "Elena Regular";
    // font-rendering: "block 0s swap infinite"; // FOUT
    font-rendering: "block 3s swap 3s"; // FOIT, at most 3sec
}
```
This CSS module defines a scripting interface to font faces in CSS, allowing font faces to be easily created and loaded from script. It also provides methods to track the loading status of an individual font, or of all the fonts on an entire page.

<table>
<thead>
<tr>
<th>IE</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Android Browser</th>
<th>Chrome for Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>31</td>
<td>36</td>
<td>37</td>
<td>39</td>
<td>40</td>
<td>7</td>
<td>7.1</td>
<td>4.1</td>
</tr>
<tr>
<td>9</td>
<td>41</td>
<td>42</td>
<td>7.1</td>
<td>7.1</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4.4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>43</td>
<td>8</td>
<td>8</td>
<td>29</td>
<td>8.3</td>
<td>8</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>44</td>
<td>30</td>
<td></td>
<td>45</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td>39</td>
<td>44</td>
<td>30</td>
<td></td>
<td>45</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Better @font-face with Font Load Events

@font-face is an established staple in the diet of almost half of the web. According to the HTTP Archive, 47% of web sites make a request for at least one custom web font. What does this mean for a casual browser of the web? In this article, I make the argument that current implementations of @font-face are actually harmful to the performance and usability of the web. These problems are exacerbated by the fact that developers have started using @font-face for two completely different use cases: content fonts and icon fonts, which should be handled differently. But there is hope. We can make small changes to how these fonts load to mitigate those drawbacks and make the web work better for everyone.

First—let’s discuss what @font-face gets right.

Initiating a Font Download

What happens when you slap a fancy new @font-face custom web font into your CSS? As it turns out—not much. Just including a @font-face block doesn’t actually initiate a download of the remote font file from the server in almost all browsers (except IE8).

/* Does not download */
@font-face {
Font load events, simple, small and efficient

v1.4.12

brahstein authored 11 days ago

- src
  - Use aliased Promise library.
- test
  - Clean up dependencies.
- vendor/google
  - Use unexpected from NPM instead of a local copy.
- .gitignore
  - initial commit
- Gruntfile.js
  - Clean up dependencies.
- LICENSE
  - Add license and README.
- README.md
  - Merge branch 'master' of github.com:brahstein/fontfaceobserver
- exports.js
  - Move exports out of main source directory.
- externs.js
  - Add timeout parameter to the check method.
- fontfaceobserver.js
  - v1.4.12
- package.json
  - v1.4.12
Font Loading Revisited with Font Events

Posted by Scott on 02/16/2015

Last month we wrote about an approach we’d been using to load web fonts in a more responsible manner than browsers tend to do by default. The purpose of the approach was to avoid a typically undesirable browser behavior we often refer to as the “FOIT” (Flash of Invisible Text), in which a browser hides all text that should be styled with a custom font until that font has finished loading.
Performance Strategy

Treat web fonts as *progressive enhancement*:

- Use subsetting to minimize font’s size,
- Consider smart CSS font stacks with OS fonts,
- Avoid faux bold/italic with exact CSS definitions,
- Support WOFF/2, OTF/TTF and EOT,
- When embedding web fonts, use either `localStorage` or CSS Font Loading API,
- When using font delivery services, use `Web Font Loader` to load fonts async,
- Defer fonts loading, show fallback fonts first.
Thank you.
Image credits

- **Front cover:** *Geometric Wallpapers* by Simon C Page ([http://simoncpage.co.uk/blog/2012/03/ipad-hd-retina-wallpaper/](http://simoncpage.co.uk/blog/2012/03/ipad-hd-retina-wallpaper/))

- Beautiful maps: [http://mapsdesign.tumblr.com](http://mapsdesign.tumblr.com)


- Thanks to Tim Kadlec, Bram Stein, Andy Hume and Zach Leatherman’s research used in this talk.