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.
Front-End Challenges
What if you want to nest one link inside another? E.g. in sidenotes, footnotes or articles, when you want the *entire excerpt to be linked*, but the excerpt could potentially also contain links which you can’t strip out from the markup?

— Roman Komarov
Things are getting better Afghanistan—so US troops could stay longer. On his first visit to the country, new Pentagon chief Ashton Carter said he would review the US mission there. He cited a more positive political atmosphere in Kabul as a reason for a slower withdrawal of American troops. Read more...

Russia is threatened with new sanctions over Ukraine. US secretary of state John Kerry says stern measures may be necessary as Moscow flouts the terms of the latest truce. Germany agrees. Ukraine says more Russian troops and tanks have crossed into its territory. Read more...
• HTML:

```
<a href="#url1">When the crisis was over,
    <a href="#url2">Mr. Jones</a>
    left the region immediately.</a>
```

• Browser parser reads it as:

```
<a href="#url1">When the crisis was over,</a>
    <a href="#url2">Mr. Jones</a>
    left the region immediately.
```
When the crisis was over, Mr. Jones left the region immediately.

Browser parser reads it as:

When the crisis was over, Mr. Jones left the region immediately.
• HTML:

```html
<a href="#url1">When the crisis was over,
  <object><a href="#url2">Mr. Jones</a></object>
  left the region immediately.</a>
```

• Browser parser reads it as:

```html
<a href="#url1">When the crisis was over,</a>
<a href="#url2">Mr. Jones</a>
<a href="#url1">left the region immediately.</a>
```
• HTML:

```html
<a href="#url1">When the crisis was over,
  <object><a href="#url2">Mr. Jones</a></object>
  left the region immediately.</a>
```

• Browser parser reads it as:

```html
<a href="#url1">When the crisis was over,</a>
<a href="#url2">Mr. Jones</a>
<a href="#url1">left the region immediately.</a>
```
• **HTML:**

```html
<a href="#url1">When the crisis was over, 
  <object><a href="#url2">Mr. Jones</a></object>
  left the region immediately.</a>
```

• **Browser parser reads it as:**

```html
<a href="#url1">When the crisis was over,</a> 
<a href="#url2">Mr. Jones</a> 
<a href="#url1">left the region immediately.</a>
```

• **Works well in modern browsers:**

IE 9+, Firefox 4+, Opera 9+, Safari 5.1+, Chrome 14+.

For old IE, we can use Cond. comments inside `<object>`.
Nested links

The problem

HTML specification have a lot of different restrictions. And I have my doubts about the feasibility of many of those. One example which I stumble upon rather often — nested links.

Spec straightly forbids such nesting:

The a element

[...]

This image is magically clickable

**This headline is clickable as a normal link**

Lorem ipsum dolor sit amet. [another link](#) An hoverable abbreviation
Faux block-level links

Block-level links are easy - give an `a` tag `display:block` and off you go. Unfortunately it all falls apart when you need to add additional links inside it, as HTML doesn't let you nest links.

We can fake this though. Have the container be a regular div, with multiple links inside it then mark one link as the 'faux-block' link to be absolutely positioned to cover the entire container. A bit of z-index to ensure the other links sit atop of this faux link and we're done. Because users click on a regular `a` tag instead of using JavaScript events things like right click and middle click continue to work as expected. We use this technique for the BBC Programmes programme object so we can link to iPlayer as an image cut-out while the whole box links to the programme's main page.

Here's a simplistic example:

I am an example header

This entire box links somewhere, thanks to faux block links. I am some example text with a custom link that sits within the block

EDIT 2015-11-02: @IschaGast came up with an improvement to this, that leverages :before so you don't need the additional link to cover the whole area. I've not tested it for full accessibility in all browsers but it looks like a solid start to an alternative implementation.
SVG Embedding

- `<img src="image.svg"/>`, `background: url(image.svg)`; block access to SVG paths via CSS.

- With “inline” SVG or Data URI encoding we can style SVG via CSS and avoid one HTTP-request.

```html
<body>
  <svg ...
  <path class="logo" ...
</svg>
</body>
```
SVG Embedding

- **Alternative:** using SVG as an `<object>` avoids caching issues with CSS styling *within* SVG.

```html
<object type="image/svg+xml" data="image.svg">
  <!-- fallback image in CSS -->
</object>

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="component.css" type="text/css"?>
<svg xmlns="http://www.w3.org/2000/svg">…</svg>
```
By default, *broken images* look pretty unspectacular. Is there any way to improve the experience by *changing the styling* if images are actually broken?
This image is broken! Ugly, isn’t it?
The `<img>` element is a *replaced* element. This is an element “whose appearance and dimensions are defined by an *external resource*. Pseudo-elements typically shouldn’t work with it.
We're sorry, the image below is broken :(  

![Kanye Laughing](http://bitsofco.de/broken.jpg)
Kanye Laughing
Broken Image of Kanye Laughing
```css
div {
  /* Same as first example */
  min-height: 50px;
}

div:before {
  content: " ";
  display: block;

  position: absolute;
  top: -10px;
  left: 0;
  height: calc(100% + 10px);
  width: 100%;
  background-color: rgb(230, 230, 230);
  border: 2px dotted rgb(200, 200, 200);
  border-radius: 5px;
}
```
```css
img::after {
  content: "\f127" " Broken Image of " attr(alt);
  display: block;
  font-size: 16px;
  font-style: normal;
  font-family: FontAwesome;
  color: rgb(100, 100, 100);

  position: absolute;
  top: 5px;
  left: 0;
  width: 100%;
  text-align: center;
}
```
<table>
<thead>
<tr>
<th>Browser</th>
<th>Alt Text</th>
<th>:before</th>
<th>:after</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome (Desktop and Android)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Firefox (Desktop and Android)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Opera (Desktop)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>✓ **</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Safari (Desktop and iOS)</td>
<td>✓ *</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>iOS Webview (Chrome, Firefox, others)</td>
<td>✓ *</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

* The alt text will only show if the width of the image is large enough to accommodate it. If no width is specified for the img, the alt text may not be displayed at all

** Font styling not applied
Styling Broken Images

Broken images are ugly.

![This image is broken! Ugly, isn't it?](image.png)

But they don’t always have to be. We can use CSS to apply styles to the `<img>` element to provide a better experience than the default.

**Two Facts About The `<img>` Element**

To understand how we can style broken images, there are two facts about the way the `<img>` element behaves that we need to understand first.

1. **We can apply regular typography-related styling to the `<img>` element.** These styles will be applied to the alternative text, if it is displayed, and will not affect the working image.

2. **The `<img>` element is a replaced element.** This is an element “whose appearance and
To achieve *fluid typography*, we can combine the `calc()` function in CSS with viewport units (vw/vh/vmin/vmax). But what if you want to apply a *modular scale* to font sizes?
We can get perfectly fluid type with

```html
{ font-size: calc(1em + 1vw); }
```

but it gives us little control over the rate at which viewport units change. Media queries? Well, with them usually there is an annoying “visual” jump between fixed and fluid values.

— Mike Riethmuller
Using `calc` in combination with `vw` and `vh` units for `font-size` to create text that always fills the viewport. No matter what ratio.
The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog
...E.g. if we wanted to choose a font-size of 16px at a screen resolution of 400px and then transition to 24px at a resolution of 800px, we couldn’t do it without a breakpoint.

— Mike Riethmuller
<table>
<thead>
<tr>
<th></th>
<th>1vw</th>
<th>2vw</th>
<th>3vw</th>
<th>4vw</th>
<th>5vw</th>
</tr>
</thead>
<tbody>
<tr>
<td>400px</td>
<td>4px</td>
<td>8px</td>
<td>12px</td>
<td>16px</td>
<td>20px</td>
</tr>
<tr>
<td>500px</td>
<td>5px</td>
<td>10px</td>
<td>15px</td>
<td>20px</td>
<td>25px</td>
</tr>
<tr>
<td>600px</td>
<td>6px</td>
<td>12px</td>
<td>18px</td>
<td>24px</td>
<td>30px</td>
</tr>
<tr>
<td>700px</td>
<td>7px</td>
<td>14px</td>
<td>21px</td>
<td>28px</td>
<td>35px</td>
</tr>
<tr>
<td>800px</td>
<td>8px</td>
<td>16px</td>
<td>24px</td>
<td>32px</td>
<td>40px</td>
</tr>
</tbody>
</table>
font-size: calc(16px + (24 - 16) * (100vw - 400px) / (800 - 400));
You choose the *min* and *max* font-size and the *screen sizes*, over which the font should scale and plug them into the equation. You can use any unit type including ems, rems or px.

— Mike Riethmuller
Fluid modular scale headings

These heading scale between a modular scale of 1.067 and 1.333. Resize the window to see the effect.

Fluid modular scale
Fluid modular scale
Fluid modular scale
Fluid modular scale
Fluid modular scale
Truly Fluid Typography With vh And vw Units

By Michael Riethmuller

May 10th, 2016 | CSS, Responsive Web Design, Typography

Embracing fluid typography might be easier than you think. It has wide browser support, is simple to implement and can be achieved without losing control over many important aspects of design.

Unlike responsive typography, which changes only at set breakpoints, fluid typography resizes smoothly to match any device width. It is an intuitive option for a web in which we have a practically infinite number of screen sizes to support. Yet, for some reason, it is still used far less than responsive techniques.

This might be because typography is so deeply rooted in the centuries-old history of typesetting. The concept of having “fluid” anything is often at odds with this tradition. In print, dimensions have always been fixed, but they don’t need to be on the web.
calc() as CSS unit value

Method of allowing calculated values for length units, i.e. width:
calc(100% - 3em)

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>Edge</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>11</td>
<td>13</td>
<td>47</td>
<td>51</td>
<td>9.1</td>
<td>38</td>
<td>9.3</td>
<td>8</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>48</td>
<td>52</td>
<td>10</td>
<td>39</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>49</td>
<td>53</td>
<td>TP</td>
<td>TP</td>
<td>TP</td>
<td>TP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>54</td>
<td>TP</td>
<td>TP</td>
<td>TP</td>
<td>TP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes | Known issues (9) | Resources (6) | Feedback

Support can be somewhat emulated in older versions of IE using the non-standard expression() syntax.

Due to the way browsers handle sub-pixel rounding differently, layouts using calc() expressions may have unexpected results.

1. Partial support in Android Browser 4.4 refers to the browser lacking the ability to multiply and divide values.
Manually adjusting line-height with media queries for optimum readability across vast number of screen sizes can be very hard. What makes it even harder, is, that instead of the screen width, the line-height should be relative to its container’s width and its font settings in order to achieve proper readability and appropriate spacing.

Thanks to @Wilto, there has been a jQuery plugin called Molten Leading around for quite some time already which makes it possible to automate this process and define a minimum width at which the adjustment starts, a maximum element width where it stops, and a minimum and maximum line-height to adjust through.
And here's what a CSS lock looks like in code:

```css
line-height: calc(1.3em + (1.5 - 1.3) * ((100vw - 21em)/(35 - 21)));
```

To understand how the formula works within `calc()`, we're going to work through it backwards.

1. See the very last part? $35-21$. That gives us the *full range of our paragraph's width*. It resolves to 14, because `14em` is the difference between our paragraph's width at its most narrow and most wide.

2. To the left of that, we've got `100vw-21em`. Because of the way CSS `calc` works, this resolves to an em-based value — and gives us a numerator to place above the 14em we already figured out. So, for example, let's say the viewport width (`100vw`) is equivalent to 34em. $34em-21em = 13em$. *Note that the viewport unit in this step is our secret sauce. The fact that this value can change dynamically with browser window width is what makes a dynamic line-height value possible.*

3. So the whole expression to the right of the multiplication sign gets distilled down to this: $13em / 14em$, or $0.928571429em$. Think of this as how close we are to the "upper gate" of our lock. If it's near zero, we're close to the lower gate. If it's near one, we're close to the upper gate.

4. Moving to the left of the multiplication sign, we compute the difference between our maximum and minimum line heights. $1.5-1.3 = 0.2$. This gives us the *full range of our fluid line height*.

5. Now we multiply the full range of our fluid line height (step 4) by how far along we are toward the upper gate of our lock (step 3):

   $0.2 * 0.928571429em = 0.185714286em$. 
By and by, an old friend of mine, a miner, came down from one of the decayed mining camps of Tuolumne, California, and I went back with him. We lived in a small cabin on a verdant hillside, and there were not five other cabins in view over the wide expanse of hill and forest. Yet a flourishing city of two or three thousand population had occupied this grassy dead solitude during the flush times of twelve or fifteen years before, and where our cabin stood had once been the heart of the teeming hive, the centre of the city. When the mines gave out the town fell into decay, and in a few years wholly disappeared—streets, dwellings, shops, everything—and left no sign. The grassy slopes were as green and smooth and desolate of life as if they had never been disturbed. The mere handful of miners still remaining, had seen the town spring up spread, grow and flourish in its pride; and they had seen it sicken and die, and pass away like a dream. With it their hopes had died, and their zest of life. They had long ago resigned themselves to their exile, and ceased to
How do you make sure that in a multi-column table, *both* a row and a column are *highlighted* on hover and on tap? Highlighting the current row is easy, but what about the column?

— Matt Walton
<table>
<thead>
<tr>
<th></th>
<th>50kg</th>
<th>55kg</th>
<th>60kg</th>
<th>65kg</th>
<th>70kg</th>
</tr>
</thead>
<tbody>
<tr>
<td>160cm</td>
<td>20</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>165cm</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td>170cm</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>175cm</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
</tr>
</tbody>
</table>
We create *tall pseudo elements* on `<td>`’s with a negative `top`-value of half of that value. Then we hide these pseudo elements with `overflow: hidden`, and use `negative z-index` to keep it below the content. Then we make all cells focusable and focus them on `touchstart`.

— @simurai
CSS:

table {
    overflow: hidden;
}

td, th {
    position: relative;
}

tr:hover {
    background-color: #ffa;
}

td:hover::after {
    content: "";
    position: absolute;
    width: 100%;
    height: 10000px;
    left: 0;
    top: -5000px;
    background-color: currentColor;
    z-index: -1;
}
### CSS:

```css
table { overflow: hidden; }
tr:hover { background-color: #ffa; }
td:hover::after, th:hover::after {
  content: "";
  position: absolute;
  width: 100%;
  height: 10000px;
  left: 0;
  top: -5000px;
  background-color: currentColor;
  z-index: -1;
}
```
Highlighting rows of a table is pretty darn easy in CSS. `tr:hover { background: yellow; }` does well there. But highlighting columns has always been a little trickier, because there is no single HTML element that is parent to table cells in a column. A dash of JavaScript can handle it easily, but Andrew Howe recently emailed me to share a little trick he found on StackOverflow, posted by Matt Walton.

It was a few years old, so I thought I'd just clean it up and post it here.

The trick is using huge pseudo elements on the `<td>`s, hidden by the table
Email clients, primarily *Gmail*, don’t support media queries. To build responsive email layouts, we have to use table-layouts. But is there a *better way?*
Dreamforce Summer Savings

Dreamforce. It's a smart investment. An astounding 96% of attendees surveyed said their business improvements outweighed the cost of going to Dreamforce, and 98% would recommend it to others. Justify your trip.

NEW to Dreamforce: Get the vision and roadmap for the products you care most about.

Summer Special
Save $300 off the on-site price of $1,299. Register for $999 until July 31.

Justify your trip: Read our blog about how to convince your boss to let you come to Dreamforce.
• **Content Stacking**

```html
<table> /* “Desktop” width = 600px = 300*2 */
    <tr>
        <td class="col" width="300">...</td>
        <td class="col" width="300">...</td>
    </tr>
</table>
```
• **Content Stacking**

```css
@media only screen and (max-width: 600px) {
  table, tr, td {
    display: block; /* table-cell -> block */
    width: 100%;
  }
}
```
- **Column Switching**

```html
<table> /* “Desktop” width = 600px = 300*2 */
  <tr>
    <td class="sub-col" width="300">...</td>
    <td class="main-col" width="300">...</td>
  </tr>
</table>
```
• **Column Switching**

```css
@media only screen and (max-width: 500px) {
    table, tr, td {
        display: block; /* table-cell -> block */
        width: 100%;
    }
}

    td[class=main-col] { display: table-header-group; }
    td[class=sub-col] { display: table-footer-group; }
```
• **Image Shifter**

```
<table> /* “Desktop” width = 600px */
  <tr>
    <td class="image" width="100">...</td>
    <td class="content">
      <div class="header">...</div>
      <div class="description">...</div>
    </td>
  </tr>
</table>
```
• **Image Shifter**

```css
@media only screen and (max-width: 500px) {
  table, tr, td { display: block; }
  td[class=image] { float: left; }
  .description { clear: both; }
}
```
• **Order and Re-order**

```html
<td class="wrapper"> /* Nested tables, oh my... */
  <table class="header">Header</table>
  <table class="nav">Navigation</table>
  <table class="content">Content</table>
  <table class="footer">Footer</table>
</td>
```
• **Order and Re-order**

@media only screen and (max-width: 500px) {
  table[class=wrapper] { display: table; }
  table[class=header] { display: table-caption; }
  table[class=nav] { display: block; }
  table[class=content] { display: table-header-group; }
  table[class=footer] { display: table-footer-group; }
}


calc() & width & min-width & max-width.
CSS:

```
.box {
  width: 320px;
  min-width: 480px;
  max-width: 160px;
}
```

If the `width` value is greater than the `max-width` value, `max-width` wins.
• **CSS:**

```css
.box {
    width: 320px;
    min-width: 480px;
    max-width: 160px;
}
```

If the *min-width* value is greater than the *width* or *max-width* values, then *min-width* wins.
Let’s build a 2-col-layout that stacks and grows below 480px. No media queries allowed.

- **CSS:**

```css
.box {
  display: inline-block;
  min-width: 50%; // basically 2-col-desktop version
  max-width: 100%; // basically 1-col-mobile version
  width: calc((480px - 100%) * 480);
  /* 480px = the breakpoint, 100% = width of the parent
  Goal: create a value bigger than our max-width or smaller than our min-width, so that either one of those property is applied instead. */
}
```
with a parent of 500px

.block {
    display:inline-block;
    min-width:50%; 250px winner
    max-width:100%; 500px
    width:calc((480px - 100%) * 480); -9600px
}

with a parent of 400px

.block {
    display:inline-block;
    min-width:50%;200px
    max-width:100%;400px winner
    width:calc((480px - 100%) * 480);38400px
}

The Fab Four

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Live Twitter Feed

Tweet with #TEDC15 and refresh to see your tweet in the stream!

Amanda Soehnlen @asoehnlen • 3m
this is just about the best thing (minus #tedc15 emails) i have read today
https://t.co/dx95dnr0kx

Jason Tropp @tropp • 4m
hey #TEDC15 the messages you hid in that email code are AMAZING! you like us you really like us :D #hiddenmessages

Jon Woodrow @jonwoodrow • 5m
This is very cool @connect_agency #TEDC15 http://t.co/cqy6RyGbt2

Marc Hedlund @Marc_everything • 6m
Live twitter feed in an email 0_0 #TEDC15

Brittany R @britties_86 • 6m
Litmus' live twitter feed - that EVEN works in Outlook - is wizardry! Where's platform 9 3/4? I need that knowledge #TEDC15

Tweet #TEDC15 to Join the Feed
Save the dates!
We’re going worldwide.

Tickets—and more details—are coming soon.
Here’s the HTML:

```html
<div class="tweet" id="tweet-1">
  <div class="tweet-avatar-wrapper">
    <div class="avatar"></div>
  </div>
  <div class="tweet-wrapper">
    <span class="name"></span> <span class="handle"></span> <span class="timestamp"></span> <span class="copy"></span>
  </div>
</div>
```

To dynamically overwrite the CSS, we relied on an external stylesheet that was updated every 10 seconds and included in our email like so:

```html
<link href="http://assets.insights.litmus.com/campaigns/t" rel="stylesheet">
```
Here's the corresponding CSS, with the tweet information in the content property:

```css
#tweet-1 .name::before {
    content: "Kevin Mandeville";
}

#tweet-1 .handle::after {
    content: "@KevinMandeville";
}

#tweet-1 .copy::before {
    content: "I'm excited for TEDC15! Who's going?";
}

#tweet-1 .timestamp::after {
    content: "1m";
}
```
We created a simple web page of just the Twitter feed using the same HTML and CSS from the email. We simply took a screenshot of the feed at the same dimensions of 600×902 through the command line utility `wkhtmltoimage` and dynamically updated that same image every 10 seconds.
MAY 26, 2015

How to Code A Live Dynamic Twitter Feed in HTML Email

BY KEVIN MANDEVILLE

As we started nailing down logistics for The Email Design Conference 2015, conversations about making our launch email bigger and better than last year’s began. Due to its unique HTML5 video background technique, last year’s email generated some serious buzz. How could we top that?

Our goals were twofold: generate interest in the conference and use an innovative—and inspiring—technique in the email to do so. After many brainstorming sessions, we decided to use the common approach of dynamic content, but with a twist. We used a unique implementation to create a live Twitter feed in an email. Yes, a Twitter feed in an email.
Bulletproof email buttons

Design gorgeous buttons using progressively enhanced VML and CSS. You can also create rock-solid background images in emails too.

Button text

Show me the button!

Background image

http://i.imgur.com/b7tepK9.png

Host your own image or use a free service like imgur to (use “Direct Link” URL).

Background color

#99c739

Font color

#ffffff

Button width

200 px

Button height

38 px

Show me the button!
Bulletproof background images

Use rock-solid background images in your HTML email with some help from VML and CSS. Why not do the same with our bulletproof button generator?

Background Image
http://i.imgur.com/XCnBXwP.png

Host your own image or use a free service like imgur (use "Direct Link" URL).

Fallback color
#7bceeb

Shown if the background image isn't loaded, and behind images that have transparency.

Apply background image to:

- Full email body
  Tile the background image in the full email window.
Responsive Email Patterns

A collection of patterns & modules for responsive emails

Submit A Pattern

Layout
- Column Drop
- 2 Equal-Width Columns

Navigation
- Top Links
- Stacked Split Links

Lists
- List w/ Thumbnails
- List w/ Thumbnails v2
The Fab Four technique to create Responsive Emails without Media Queries

I think I found a new way to create responsive emails, without media queries. The solution involves the CSS `calc()` function and the three `width`, `min-width` and `max-width` properties.

Or as I like to call them all together: the Fab Four (in CSS).
You want to add a background to inline text for headings, but the text should be padded along both the left and right edge of each line. The left and right padding will only apply to the very first and very last line.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fugiat suscipit ut odit animi consequatur error numquam perspiciatis voluptas quod eum.
Fabien Doiron's box-shadow Method

Turns out you can use zero-spread box-shadow on an inline element on only the x-axis to pad each line. Essentially:

```css
.padded-multi-line {
    display: inline;
    background: orange;
    box-shadow: 10px 0 0 orange, -10px 0 0 orange;
}
```

Here is the original and then my fork to show how it works:

DEFAULT: THIS IS HOW HIGHLIGHTED TEXT USUALLY WRAPS. IT GETS TIGHT ON THE LEFT AND RIGHT EDGES.
The `box-decoration-break` in CSS specifies element’s appearance if the box for the element is fragmented, i.e. when an inline box wraps onto multiple lines, or when a block spans more than one column inside a column layout container.
Adam Campbell's box-decoration-break Method

During a discussion that popped up over this, Adam pointed out there is a new CSS property that is (as I understand it) specifically for this. This removes the need for three elements. Technically you only need one, the inline element, but it's likely you'll be doing this on a header so you'll probably end up with a block-parent anyway, which is best for spacing.

Here is the original and my stripped down demo:

![How do I add padding to subsequent lines of an inline text element?](image)

This is working in Chrome and Safari, but not Firefox now working in Firefox 32+, in my quick tests. Chrome and Safari require it to be webkit-box-decoration-break.
**CSS box-decoration-break** - WD

Controls whether the box's margins, borders, padding, and other decorations wrap the broken edges of the box fragments (when the box is split by a break (page/column/region/line)).

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Edge</td>
<td>Chrome</td>
</tr>
<tr>
<td>8</td>
<td>43</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>45</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
<td>50</td>
</tr>
</tbody>
</table>

Notes: Known issues (0) Resources (4) Feedback

1 Partial support refers to working for inline elements but not across column or page breaks.
CHECK OUT REACT FOR BEGINNERS

I LOVE TO LEARN!

I SURE DO

ARE YOU A SUBLIME TEXT POWER USER?

LOVE FLEXBOX!
a {
  font-size: 4rem;
  text-decoration: none;
  color: #1e1f23;
  text-shadow: 1px 1px 0 rgba(255,255,255,0.4);
}
.fun-hover {
  background-image: -webkit-linear-gradient(left, #25b0a9 50%, #fee603 50%);
  background-image: linear-gradient(to right, #25b0a9 50%, #fee603 50%);
  background-position: 0;
  background-size: 200%;
  -webkit-transition: all 0.4s;
  transition: all 0.4s;
}
.fun-hover:hover {
  background-position: -100%;
}
CSS Highlight on Hover Animation

Jan 20, 2016

Check out React for beginners

Are you a Sublime Text power user?

I love to learn!

I sure do love Flexbox!

I saw this great little highlight hover animation on Web Designer Depot the other day and I wondered how it worked. Turns out it’s just a few lines of CSS.
Images make up a large portion of bandwidth payload. Is there any way to optimize images beyond good ol’ image optimization? What if a hero image has to render fast, e.g. on landing pages?

— Tobias Baldauf
The original photo has 1600px width, 971 Kb. Quality 60 brings the size down to 213 Kb.
• Blurring unimportant parts of the photo brings the size down to 147 Kb.
Sequential JPEG  Progressive JPEG

Images taken from http://www.pixeltech.net/article/1374757887-Use-progressive-JPEG-to-improve-user-experience
Default Scan Levels

# Initial DC scan for Y,Cb,Cr (lowest bit not sent)
0,1,2: 0-0, 0, 1 ;

# First AC scan: send first 5 Y AC coefficients, minus 2 lowest bits:
0: 1-5, 0, 2 ;

# Send all Cr,Cb AC coefficients, minus lowest bit:
# (chroma data is usually too small to be worth subdividing further;
# but note we send Cr first since eye is least sensitive to Cb)
2: 1-63, 0, 0 ;
1: 1-63, 0, 0 ;

# Send remaining Y AC coefficients, minus 2 lowest bits:
0: 6-63, 0, 2 ;

# Send next-to-lowest bit of all Y AC coefficients:
0: 1-63, 2, 1 ;

# At this point we've sent all but the lowest bit of all coefficients.
# Send lowest bit of DC coefficients
0,1,2: 0-0, 1, 0 ;

# Send lowest bit of AC coefficients
2: 1-63, 1, 0 ;
1: 1-63, 1, 0 ;

# Y AC lowest bit scan is last; it's usually the largest scan
0: 1-63, 1, 0 ;
# Initial DC scan for Y,Cb,Cr (lowest bit not sent)
0,1,2: 0-0, 0, 1;

# First AC scan: send first 5 Y AC coefficients, minus 2 lowest bits:
0: 1-5, 0, 2;

# Send all Cr,Cb AC coefficients, minus lowest bit:
# (chroma data is usually too small to be worth subdividing further;
# but note we send Cr first since eye is least sensitive to Cb)
8: 1-63, 0, 0;
8: 1-63, 0, 0;

# Send remaining Y AC coefficients, minus 2 lowest bits:
0: 6-63, 0, 2;

# Send next-to-lowest bit of all Y AC coefficients:
0: 1-63, 2, 1;

# At this point we've sent all but the lowest bit of all coefficients.
# Send lowest bit of DC coefficients
0,1,2: 0-0, 1, 0;

# Send lowest bit of AC coefficients
2: 1-63, 1, 0;
8: 1-63, 1, 0;

# Y AC lowest bit scan is last; it's usually the largest scan
6: 1-63, 1, 0;
1st Scan Layer Has Small Byte Size

Ships Fast

&

Shows Soon
# Interleaved DC scan for Y, Cb, Cr:

0, 1, 2: 0-0, 0, 1;

Initial DC for all channels

# AC scans:

0: 1-27, 0, 0;
Half of all brighter values

2: 1-63, 0, 0;
All remaining color channel values

3/4: 1-63, 0, 0;

# Remaining Y coefficients

0: 28-63, 0, 0;

2nd half of brighter channel
### Progressive JPEGs via HTTP2

<table>
<thead>
<tr>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>Visually Complete</th>
<th>Speed Index</th>
<th>DOM Elements</th>
<th>Result (error code)</th>
<th>Document Complete</th>
<th>Fully Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.180s</td>
<td>0.118s</td>
<td>0.436s</td>
<td>3.500s</td>
<td>1537</td>
<td>75</td>
<td>0</td>
<td>3.180s</td>
<td>3.213s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,741 KB</td>
<td>1,741 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RUM First Paint</th>
<th>domContentLoaded</th>
<th>loadEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.822s</td>
<td>0.646s - 0.646s (0.000s)</td>
<td>3.468s - 3.470s (0.002s)</td>
</tr>
</tbody>
</table>

### Optimized Progressive JPEGs via HTTP2

<table>
<thead>
<tr>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>Visually Complete</th>
<th>Speed Index</th>
<th>DOM Elements</th>
<th>Result (error code)</th>
<th>Document Complete</th>
<th>Fully Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.486s</td>
<td>0.120s</td>
<td>0.471s</td>
<td>3.800s</td>
<td>1445</td>
<td>75</td>
<td>0</td>
<td>3.486s</td>
<td>3.590s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,822 KB</td>
<td>1,822 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RUM First Paint</th>
<th>domContentLoaded</th>
<th>loadEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.735s</td>
<td>0.648s - 0.648s (0.000s)</td>
<td>3.766s - 3.768s (0.002s)</td>
</tr>
</tbody>
</table>
A Bash script to automate adaptive JPEG compression using common CLI tools

- **69** commits
- **1** branch
- **0** releases
- **5** contributors

Branch: **master**

**technopagan** Updating README

- **images**: Updating the description and info to reflect the newest changes, update...
- **unitests**: Switch to mozjpeg as default encoder, optimizing tile size selection, ...
- **README.md**: Updating README
- **adept.sh**: Switch to mozjpeg as default encoder, optimizing tile size selection, ...

**Adept - the adaptive JPG Compressor**
Improved JPEG encoder.
By default, tables are quite unpredictable and spongy, and if you don’t know how lengthy the content inside cells will be, some columns can be *unpredictably wide*, destroying the layout. What to do?

— Louis Lazaris
<table>
<thead>
<tr>
<th></th>
<th>1-1</th>
<th>1-2</th>
<th>1-3</th>
<th>1-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Example text goes here. Example text goes here.</td>
<td>2-2</td>
<td>2-3</td>
<td>2-4</td>
</tr>
<tr>
<td>3-1</td>
<td></td>
<td>3-2</td>
<td>3-3</td>
<td>3-4</td>
</tr>
<tr>
<td>4-1</td>
<td></td>
<td>4-2</td>
<td>4-3</td>
<td>4-4</td>
</tr>
</tbody>
</table>
Default Tables are Weird

<table>
<thead>
<tr>
<th>I'm a table and stuff</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm as wide as the cells are</td>
</tr>
<tr>
<td>I can easily become 100% wide. and I'll stop there if I possibly can.</td>
</tr>
<tr>
<td>If you don't specify it, the cells will be kind of arbitrarily wide depending on the content inside. This one is way wider right? (^{(°_o)}/)</td>
</tr>
<tr>
<td>If you do specify it, that will be respected if it's possible to do so. These should be 50% wide.</td>
</tr>
<tr>
<td>But things can get weird. I told both of these cells to be 1000px wide and it's like NOPE.</td>
</tr>
<tr>
<td>It's kinda like a calculation. I'm told to be 2000px wide, thus I'm 66.66% wide (2/3 of total 3000px). And I'm told to be 1000px wide, thus I'm 33.33% wide (1/3 of total 3000px).</td>
</tr>
<tr>
<td>An image that's too wide though? That'll bust out like nobody's business.</td>
</tr>
<tr>
<td>You can try hiding the overflow but that doesn't work.</td>
</tr>
<tr>
<td>Or some non-wrapping text? That'll blast out a table in short order like this.</td>
</tr>
</tbody>
</table>
But seriously, as always, a live example will help. In the following demo, the table has `table-layout: fixed` added in the CSS. Click the toggle button to toggle it off, then on, etc.

### Using the `table-layout` property

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>2-1</td>
<td>1-2</td>
<td>1-3</td>
</tr>
<tr>
<td>2-2</td>
<td>3-1</td>
<td>2-3</td>
<td>1-4</td>
</tr>
<tr>
<td>3-2</td>
<td>4-1</td>
<td>3-3</td>
<td>2-4</td>
</tr>
<tr>
<td>4-2</td>
<td>4-3</td>
<td>3-4</td>
<td>4-4</td>
</tr>
</tbody>
</table>

You can see in this example the advantage of using `table-layout: fixed`, as opposed to the default of `auto`. This won’t always be the best choice and it won’t always be necessary, but it’s a nice one to keep in mind when dealing with tables that have cells with variable-width data.

Chris Coyier did a great write-up on this property last year, so if you want a much more comprehensive discussion, that’s your best bet.
With `table-layout: fixed;` the layout is fixed based on the first row. Set the width of those, and the rest of the table follows.

— Chris Coyier
With `table-layout: fixed;`

Things get a lot sturdier and more predictable with property/value in place.

The layout is fixed based on the first row. Set the width of those, and the rest of the table follows.

It's a little more complicated, but not much. Here's an exploration:

```html
<table>
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Data 1</td>
      <td>Data 2</td>
    </tr>
    <tr>
      <td>Data 3</td>
      <td>Data 4</td>
    </tr>
  </tbody>
</table>
```
### HTML
```html
<table class="users">
  <thead>
    <tr>
      <th class="row-ID">ID</th>
    </tr>
  </thead>
</table>
```

### CSS
```css
.users {
  table-layout: fixed;
  width: 100%;
  white-space: nowrap;
}
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Job</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Johnny Five</td>
<td>Robotin'</td>
<td><a href="mailto:need@input.com">need@input.com</a></td>
</tr>
<tr>
<td>0002</td>
<td>Super Superlonglastname_smith</td>
<td>Doin' stuff</td>
<td><a href="mailto:doing@stuff.com">doing@stuff.com</a></td>
</tr>
<tr>
<td>0003</td>
<td>Roger Wilco</td>
<td>Truckdrivin'</td>
<td><a href="mailto:roger@wilco.com">roger@wilco.com</a></td>
</tr>
<tr>
<td>0004</td>
<td>Mad Hatter</td>
<td>Hat Makin'</td>
<td>loves@mercury....</td>
</tr>
</tbody>
</table>
Whenever the content is *dynamically injected* into the DOM or when “above-the-fold” CSS is used or font changes, users see content jumping around. That’s particularly annoying on small screens. Can we fix it?
If we know the expected height of the block, we can use `min-height` to “reserve” space for it, or calculate it with JavaScript on `window` early on and write styles dynamically.
Alternatively, we could use a smooth *transition* from the initial (fixed) height to the final (fixed) height. Downside: transitions don’t work on *min-height*. 
Give it a whirl.

We’ve created a sleek, ready-made payment UI for accepting cards -- or even PayPal -- in your app or website. Once completed, your client code obtains a payment method nonce for use on your server.

Credit Card:
CC# - 4111 1111 1111 1111
EXP - 01/20

PayPal:
EMAIL - test@test.com
PASS - test

Drop-in UI integration

A few lines of code and you’re open for business.
• CSS:

```css
.ad-wrapper {
  height: 0;
  overflow: hidden;
  transition: height 0.8s ease-out;
}

.ad-wrapper.loaded {
  height: 400px;
}
```
Content Jumping (and How To Avoid It)

The following is a guest post by Brandon Smith (CodePen). Brandon takes a look at one of the more eye-twisting behaviors of web sites and some possibilities for avoiding it.

Few things are as annoying on the web as having the page layout unexpectedly change or shift while you're trying to view or interact with it. Whether you're attempting to read an article as it wriggles around in front of you, or you try to click a link only to have another one push it out of the way and take you off to somewhere unexpected, it's always frustrating.
What’s the deal with emoji? Can we style them with CSS, or change them with JavaScript? Or even better, can they be an alternative to SVG and icon fonts?
Clockwise Rightwards and Leftwards Open Circle Arrows With Circled One Overlay
(also known as U+1F502, &#x1f502; or \u1f502).
Person Raising Both Hands in Celebration (also known as Festivus Miracle Emoji, U+1F64C, &#1f64c; or \u1f64c).
Reimagining single-page applications with *progressive enhancement*

What’s the difference between a *web page* and a *web application*? Though we tend to identify documents with reading and applications with interaction, most web-based applications are of the blended variety.

Users can consume information and perform tasks in the same place. Regardless, the way we approach building web applications usually dispenses with some of the simple virtues of the readable web. Single-page applications tend to take the form of runtimes, JavaScript executables deployed like popup shops into vacant...
Is the internet killing creativity?

The internet is a wonderful place (mostly). An unprecedented revolution in communication, it continues to empower more people to publish and share their knowledge than any other phenomenon in history. It is a limitless playground of ideas and unbridled creativity. Or is it?

Posted in Coding / CSS / Design / HTML

Free Photoshop tools for web designers

High-impact, minimal-effort, cross-browser testing
Is the internet killing creativity?

The internet is a wonderful place (mostly). An unprecedented revolution in communication, it continues to empower more people to publish and share their knowledge than any other phenomenon in history. It is...
Is the internet killing creativity?

The internet is a wonderful place (mostly). An unprecedented revolution in communication, it continues to empower more people to publish and share their knowledge than any other phenomenon in history. It is a limitless playground of ideas and unbridled creativity. Or is it?

Posted in Coding
CSS Design HTML
Is the internet killing creativity?

The internet is a wonderful place (mostly). An unprecedented revolution in communication, it continues to empower more people to publish and share their knowledge than any other phenomenon in history. It is a limit playground of ideas and unbridled creativity. Or is it?

Posted in Coding
Is the internet killing creativity?

The internet is a wonderful place (mostly). An unprecedented revolution in communication, it continues to empower more people to publish and share their knowledge than any other phenomenon in history. It is a limitless playground of ideas and unbridled creativity. Or is it?

Posted in Coding
My name is
Glen Maddern
and I do rad web stuff

ARTICLES

CSS Modules
Welcome to the Future - 2015-08-19

Interoperable CSS
A CSS standard for the Loader Age - 2015-06-21

JavaScript in 2015
*Emoji* are coloured glyphs added into Unicode 6.0 in 2010. They are depicted in the spec, but the exact appearance *isn’t defined* and varies between fonts, just like normal typefaces display letters differently.
These are all the same emoji!
This is what the “grinning face with smiling eyes” emoji looks like on devices for each of these platforms:

Apple  Google  Microsoft  Samsung  LG  HTC  Twitter  Facebook  Mozilla  Emoji One
People's Emotion/Sentiment Rankings for 😅

Number of People that Gave Each Ranking

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Negative</th>
<th>Neutral</th>
<th>Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>
“Blissfully happy” or “ready to fight”: Varying Interpretations of Emoji

Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, Brent Hecht
GroupLens Research, University of Minnesota
Minneapolis, MN 55455, USA
{hml13, thebault, schang, ijohnson, terveen, bhecht}@cs.umn.edu

Abstract
Emoji are commonly used in modern text communication. However, as graphics with nuanced details, emoji may be open to interpretation. Emoji also render differently on different viewing platforms (e.g., Apple’s iPhone vs. Google’s Nexus phone), potentially leading to communication errors. We explore whether emoji renderings or differences across platforms give rise to diverse interpretations of emoji. Through an online survey, we solicit people’s interpretations of a sample of the most popular emoji characters, each rendered for multiple platforms. Both in terms of sentiment and semantics, we analyze the variance in interpretation of the emoji, quantifying which emoji are most (and least) likely to be misinterpreted. In cases in which participants rated

Most commonly-used emoji are encoded in the Unicode standard for indexing characters. There are currently 1,282 emoji in the Unicode standard, and for each of these, the Unicode Consortium provides a code and name (e.g., U+1F600 for “grinning face”) but not the actual graphic. This is the same as is the case for Unicode text characters: for example, the Unicode character U+0041 indexes the Latin capital letter ‘A’, but it does not indicate specifically how the ‘A’ should look. Instead, a font renders the Unicode characters a particular way: the appearance of this text that you are reading is dictated by the Times New Roman font.
Because *emoji* are Unicode code points, we can create a font with the emoji that we need in our interface and **override** platform-specific designs to avoid inconsistencies.
<table>
<thead>
<tr>
<th>Native</th>
<th>Apple</th>
<th>Android</th>
<th>Android</th>
<th>Symbola</th>
<th>Twitter</th>
<th>Phantom</th>
<th>Unicode</th>
<th>Bytes (UTF-8)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>😃</td>
<td>😃</td>
<td>😃</td>
<td>😃</td>
<td>😄</td>
<td>😄</td>
<td>😄</td>
<td>U+1F601</td>
<td>\xE0\x9F\x98\x81</td>
<td>grinning face with smiling eyes</td>
</tr>
<tr>
<td>😭</td>
<td>😭</td>
<td>😭</td>
<td>😭</td>
<td>😭</td>
<td>😭</td>
<td>😭</td>
<td>U+1F602</td>
<td>\xE0\x9F\x98\x82</td>
<td>face with tears of joy</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>U+1F603</td>
<td>\xE0\x9F\x98\x83</td>
<td>smiling face with open mouth</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>U+1F604</td>
<td>\xE0\x9F\x98\x84</td>
<td>smiling face with open mouth and smiling eyes</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>U+1F605</td>
<td>\xE0\x9F\x98\x85</td>
<td>smiling face with open mouth and cold sweat</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>U+1F606</td>
<td>\xE0\x9F\x98\x86</td>
<td>smiling face with open mouth and tightly-closed eyes</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>U+1F609</td>
<td>\xE0\x9F\x98\x89</td>
<td>winking face</td>
</tr>
</tbody>
</table>
Internally strings are represented in **UTF-16**, so each code point can be represented by one or more 16-bit code units. Some emojis use only 1 code unit, others 2 and more.
Not all *emoji* are created equal. They can be modified with emoji modifiers, so some are *surrogates* which is why sometimes icons are rendered incorrectly.
"👍".length // returns 4
"毖".length // returns 4
"👨‍👩‍👧‍👦".length // returns 11
• HTML:

```html
<p style="font-family: 'Sentinel', sans-serif;">%</p>
```
• **HTML:**

```html
<p style="font-family: 'Sentinel', sans-serif;">🪦</p>
```

• **The browser will:**

  — Look up the glyph in the *Sentinel* font,
  — If it can’t find the glyph, it will fall back to the *fallback* font,
  — In this case, fallback is *sans-serif* (Helvetica/Arial),
  — The fallback doesn’t have the glyph either,
  — Browser will try to figure out the glyph type,
  — Eventually it will look up in a locally installed Emoji font (e.g. *AppleColorEmoji*),
  — The browser will render the icon.
Emoji: how do you get from U+1F355 to 🍕?

April 4, 2016

You know that scene in The Rock where Nicolas Cage is super dreamy (like he is) and decides his life mission is to look for VX poison gas and save San Francisco (like he would)? That’s baaaasically me, if by “look for VX poison gas” you mean “nerd out on emoji”, and by “save San Francisco” you mean “and tell everyone about it”. I mean, you clicked on this link, what did you think was going to happen?

飲み How did we get so lucky?

An emoji is a coloured glyph. They appeared around 1999 in Japan, where each mobile carrier implemented their own variants, and people were sending them around in text messages. This was a bit of a mess, as you can imagine proprietary formats interacting with other proprietary formats to be, so in 2000 there was a proposal to standardize them. It wasn’t until 2009, though, that emoji got specced in Unicode 5.2 #blessed.

Spec trivia: each emoji has a design guideline and name, which is a description/suggestion of what the emoji should look like. This is why 😞, for example, often gets in trouble for being labelled as Information Desk Person, but is actually just a sassy lady: it’s the implementation of the emoji that doesn’t match its original description, not the other way around. If you take sassy lady away from me though, there will be words.
Styleguides

Git Commit Messages

- Use the present tense ("Add feature" not "Added feature")
- Use the imperative mood ("Move cursor to..." not "Moves cursor to...")
- Limit the first line to 72 characters or less
- Reference issues and pull requests liberally
- When only changing documentation, include \[ci skip\] in the commit description
- Consider starting the commit message with an applicable emoji:
  - 🎨:art: when improving the format/structure of the code
  - 🏎️:racehorse: when improving performance
  - 🚚:non-potable_water: when plugging memory leaks
  - 📝:memo: when writing docs
  - 🐧:penguin: when fixing something on Linux
  - 🍎:apple: when fixing something on macOS
  - 🕵️‍♂️:checkered_flag: when fixing something on Windows
  - 🐛:bug: when fixing a bug
  - ⚠️:fire: when removing code or files
  - ❤️:green_heart: when fixing the CI build
  - ✅:white_check_mark: when adding tests
  - ⛔️:lock: when dealing with security
  - ⬆️:arrow_up: when upgrading dependencies
  - ⬇️:arrow_down: when downgrading dependencies
  - 🚴‍♂️:shirt: when removing linter warnings

JavaScript Styleguide
Emoji Toggles

A Pen by Chris Coyier

**HTML**
```
1. `<div class="emoji-toggle emoji-happy">
2.   <input type="checkbox" id="toggle1" class="toggle">
3.   `<div class="emoji">
```

**CSS (SCSS)**
```
1. // Base class
2. .emoji-toggle {
3.   position: relative;
4.   width: 60px;
5.   margin: 40px auto;
6.   .well {
```

**Emoji Toggles**

- Happy 😊 Mad
- Plane ✈️ Train
- Burger 🍔 Pizza
- Thumbs Up 👍 Thumbs Down
- Loves Me 💘 Loves Me Not
**View Emoji**

iOS 6, iOS 7 and iOS 8 all support the full unicode emoji set of characters, as well as the backwards-compatible emoji from iOS 5 and earlier which were based on the proprietary format used in Japan.

All native iOS apps can view emoji.

<table>
<thead>
<tr>
<th>App</th>
<th>9</th>
<th>8–6</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safari</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Chrome</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Twitter</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Facebook</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Instagram</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Snapchat</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Whatsapp</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Emoji Keyboard**

iOS has supported emoji since iOS 4, but first became available as a keyboard option outside of Japan in iOS 5.

<table>
<thead>
<tr>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Android

View Emoji

Android supports emoji in version 4.1 Jelly Bean and above. Android 4.4 KitKat and above is required to display color emoji.

<table>
<thead>
<tr>
<th>App</th>
<th>5.0</th>
<th>4.4</th>
<th>4.3</th>
<th>4.2/4.1</th>
<th>4.0 +</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Browser</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Twitter</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Facebook</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Instagram</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Snapchat</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Whatsapp</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Hangouts</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Emoji Keyboard

Android 4.3 Jelly Bean was the first release to include a built-in keyboard (iWnn IME), and Android 4.4 KitKat is the first to include a color emoji keyboard.

Versions of Android prior to 4.3 require the use of a third-party keyboard app to insert...
Windows

View Emoji

Windows supports emoji on Windows 7 or above, including Windows RT, Windows 8 and Windows 10. Windows Vista does not support emoji, nor does any version prior to it.

To view color emoji, Windows 8.1 and above is required, and only in Internet Explorer or Edge. Chrome does not support native emoji on any version of Windows.

<table>
<thead>
<tr>
<th>App</th>
<th>10</th>
<th>8</th>
<th>7</th>
<th>Vista</th>
<th>XP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Chrome*</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Firefox</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

*Chrome doesn't display emoji (yet), but addons exist to provide a workaround.

Emoji Keyboard

Windows 10, Windows 8 and Windows RT have a touch-keyboard that can be enabled for direct emoji input. Windows 7 requires the use of copy and paste.
How Emoji Can Improve Your Code—Seriously

Who doesn’t love emoji? As a massive emoji user on messaging and email apps, I’ve started to experiment with how I could bring my love of emoji into my day-to-day software development.

And although this started as a bit of a joke, I’ve found emoji to be legitimately valuable in my development work in some cases. Why?

We, as developers, routinely look at large amounts of text—whether it’s code, production logs, commit messages, documentation, or whatever—and emoji inherently stand out in what is normally a wall of text. It’s far easier to pick an emoji out of a list than a random string, and that skimmability can lead to real productivity gains. Plus, even if there is no actual productivity gain—using emoji is just plain fun. Here are a few of the things I’ve been playing with.

Emoji in comments

Suppose you need to add a warning to your code, something like this:
WARNING:
Changing this code breaks the build

Because emoji are far easier to read at a glance, I could see them being valuable if you adopt standards on your team for marking up certain common things in your comments.

TODOS:
- Unit Test
- Performance profiling

And again, even if you don’t have a practical use for emoji in comments, they’re still an entertaining way to interact with coworkers.

Why does this cause the app to crash?

// O(n²) complexity

And since the emoji are just in comments, there’s little risk of breaking
Emoji in commit messages

Commit messages are usually boring. But with emojis you can make them slightly less boring. I’ve been trying a one-emoji-per-commit policy lately and I haven’t been disappointed so far.

- Better delete icons
- Show an error if the user attempts to register without a connection
- Remove some files no longer used on this branch
- A couple of README tweaks
- Merge branch 'angular'

Although this is a bit of a silly example, I would again make the argument that the emoji improve the readability here. If you’re looking at a giant list of commit messages, which you often are in big projects, having an emoji in the text makes picking a single commit out of a list a heck of a lot easier. The Atom editor by GitHub even has an emoji guide in their [git commit message styleguide](https://github.com/atom/atom-editor/blob/master/STYLE GUIDE.md).

Emoji in code
So you’re building a responsive *multi-lingual* website that aims to support over 30 languages. How do you architect a system to support this kind of complexity?
The crucial asset of longevity is building "neutral", configurable components which can be easily extended and adjusted.
// english.json
{
  serviceName: 'english';
  language: 'en';
  textDirection: 'ltr';
  socialMediaButtons: ['twitter', 'facebook', 'reddit'];
}

// russian.json
{
  serviceName: 'russian';
  language: 'ru';
  textDirection: 'ltr';
  textLength: 'verbose';
  socialMediaButtons: ['twitter', 'facebook', 'vk'];
}
config/english.json
    /russian.json

css/english.css
    /russian.css

sass/english.scss
    /russian.scss
    /mixins/_textDirection.scss
    /mixins/_textLength.scss
    /mixins/_socialMediaButtons.scss

index.en.html
index.ru.html
With a *templating* language, we can then plug data from config files and hence customize HTML output for every language.
// english.scss
$english = true;
$script = 'latin';
$direction = 'left';
@include(mixins/directions);
@include(mainstyles);

// arabic.scss
$arabic = true;
$script = 'arabic';
$direction = 'right';
@include(mixins/directions);
@include(mainstyles);

@if $arabic {
  // apply styling only to Arabic version
}
// directions.scss
$margin-left: margin-left;
if $direction == 'right' {
    $margin-left: margin-right;
}

$padding-left: padding-left;
if $direction == 'right' {
    $padding-left: padding-right;
}

$left: left;
if $direction == 'right' {
    $left: right;
}
// directions.scss
$margin-left: margin-left;
if $direction == 'right' {
  $margin-left: margin-right;
}

$padding-left: padding-left;
if $direction == 'right' {
  $padding-left: padding-right;
}

$left: left;
if $direction == 'right' {
  $left: right;
}

$margin-right: margin-right;
if $direction == 'right' {
  $margin-right: margin-left;
}

$padding-right: padding-right;
if $direction == 'right' {
  $padding-right: padding-left;
}

$right: right;
if $direction == 'right' {
  $right: left;
}
// global.scss
.nav-element {
    #{$margin-left}: 10px;
    #{$padding-right}: 10px;
    #{$left}: 10px;
}

// english.css
.nav-element {
    margin-left: 10px;
    padding-right: 10px;
    left: 10px;
}

// arabic.css
.nav-element {
    margin-right: 10px;
    padding-left: 10px;
    right: 10px;
}
// global.scss
.nav-element {
    float: flip(left, right);
    padding: flip(10px 10px 0 0, 10px 0 0 10px);
    line-height: get-script-value(latin 1.3, arabic 1.6);
}

// english.css
.nav-element {
    float: left;
    padding: 10px 10px 0 0;
    line-height: 1.3em;
}

// arabic.css
.nav-element {
    float: right;
    padding: 10px 0 0 10px;
    line-height: 1.6em;
}
Uhuru asaba Afrika kwikura ku mfashanyo

Umukuru w'igihugu wa Kenya, Uhuru Kenyatta, yahamagariye baganzina biwe b'abanye Afrika guhagarika kwakira imashanyo y’igihugu bikize.

Ubukarugendo burabangamiwe muri Kenya
Igiti giri c’abukarugendo bagendera igihu hucu Kenya cagabanukese 25%.

Avuga ko ari umwirabura tariwe
Abavye y’umugore aheranium agateka k’abirabura muri Amerika, bavuga ko amaze imyaka abesho ko ari umwiraburwa.

Mu Burundi imiyerekano "itumwa n’abamenyeshamakuru";leta
Ubutegeti mu Burundi bwemeza ko ata imiyerekano kirangwa i Bujumbura.

BBC itanga inyigisho kw’itangazamakuru

11 Ukwu gatandatu 2015 | Amakuru
Украина: Порошенко отправил в отставку главу минобороны

Президент Украины Петр Порошенко отправил в отставку министра обороны страны Валерия Гелетя. Гелетя занимал этот пост с начала июля 2014 года.

Британец Хэмилтон выиграл первый Гран-при России в Сочи

Британский гонщик Льюис Хэмилтон из команды "Мерседес" стал победителем первого в истории "Формулы-1" Гран-при России, который прошел в Сочи.

Глава Гонконга: протестующие не изменят позицию Китая

Глава администрации Гонконга Лян Чжэнъин заявил, что у участников уличных акций протеста нулевой шанс изменить позицию Пекина по поводу формата выборов 2017 года.
Египет: полиция разогнала студенческие протесты в Каире

В Египте студенты нескольких крупных университетов устроили в Каире антиправительственные протесты.

Они требовали от властей освободить задержанных ранее участников студенческих демонстраций.

Полиция при поддержке нескольких бронемашин разогнала протесты, несколько студентов было арестовано.

Сообщается, что протесты были организованы сторонниками свергнутого президента Мохаммеда Мурси.

Европейский Чемпионат по футболу 2016: Россия не смогла обыграть Молдавию

На севере Ливии возобновились бои
Sometimes web fonts might be out of question, so consider generating *full page screenshots* and sending a mix of HTML + images to the users.
Responsive News

Where BBC News developers blog about responsive design.
Opinions expressed on this blog are those of the individual contributors, and are not necessarily those of the BBC as a whole.

13 tips for making responsive web design multi-lingual

Responsive Web Design (RWD) builds on the primary design principle underlying the web’s core usefulness and growth: universality. A content out approach that is device agnostic makes your responsive website future friendly as it will in theory work on any device. The web wins the more viewable your website is. By adapting our responsive websites to work with multiple languages we can further increase the number of users who are able to use our content.

The BBC News responsive codebase is responsible for the rendering of 28 different
What if you want to use a full-width element in a *fixed-width container*? E.g. when you want *some* content to extend beyond the boundaries of the container?

— Tyler Sticka
**HTML:**

```html
<div class="u-containProse">
    <p>...</p>
    <p>...</p>
</div>
```

**CSS:**

```
.u-containProse {
    margin: 0 auto;
    max-width: 40em;
}
```
• **HTML:**

```html
<div class="u--containProse">
  <p>...</p>
</div>

<img src="..." alt="..." />

<div class="u--containProse">
  <p>...</p>
</div>
```

• **CSS:**

```css
.u--containProse { 
  margin: 0 auto;
  max-width: 40em;
}
```
To release our child element from its container, we need to know *how much space* there is between the container edge and the viewport edge.
What’s this space exactly? Well, it’s half the viewport width minus half the container width. \texttt{calc()} to the rescue!
• HTML:

```html
<div class="u-containProse">
  <p>...</p>
  <img class="u-release" src="..." />
  <p>...</p>
</div>
```

• CSS:

```css
.u-release {
  margin-left: calc(-50vw + 50%);
  margin-right: calc(-50vw + 50%);
}
```
When the height or width of the initial containing block is changed, they are scaled accordingly. Note that the initial containing block’s size is affected by the presence of scrollbars on the viewport.
• **HTML:**

```html
<div class="u-containProse">
  <p>...</p>
  <img class="u-release" src="..." />
  <p>...</p>
</div>
```

• **CSS:**

```css
.u-release {
  margin-left: calc(-50vw + 50%);
  margin-right: calc(-50vw + 50%);
}
```
**HTML:**

```html
<div class="u-containProse">
  <p>...</p>
  <img class="u-release" src="..." />
  <p>...</p>
</div>
```

**CSS:**

```css
.u-release {
  margin-left: calc(-50vw + 50%);
  margin-right: calc(-50vw + 50%);
}

html, body {
  overflow-x: hidden;
}
```
We push the container to the exact **middle** of the browser window with `left: 50%`, then pull it back to the left edge with `-50vw` margin (h/t Sven Wolfermann).

**CSS:**
```
.u-release {
  width: 100vw;
  position: relative;
  left: 50%;
  right: 50%;
  margin-left: -50vw;
  margin-right: -50vw;
}
```
Breaking Out With Viewport Units and Calc

Written by Tyler Sticka on May 26, 2016

While iterating on a new article layout for the impending Cloud Four redesign, I encountered an old CSS layout problem.

For long-form content, it's usually a good idea to limit line lengths for readability. The most straightforward way to do that is to wrap the post content in a containing element:

```css
.u-containProse {
    max-width: 40em;
    margin-left: auto;
    margin-right: auto;
}
```
What if you want to build fluid horizontal lines around centered content for an art-directed article? How exactly would you build it?

— Jack Brewer
Example Title

Fusce dapibus, tellus ac cursus commodo, tortor mauris

or

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus donec sed odio dui.

Action here
Example Title

Fusce dapibus, tellus ac cursus commodo, tortor mauris

or

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus donec sed odio dui.

Action here

• CSS:

```css
.has-lines {
    display: flex;
    justify-content: center; // horizontal alignment
    align-items: center; // replace stretch / v. alignment
}
```
• **CSS:**

```css
.has-lines {
    display: flex;
    justify-content: center; // horizontal alignment
    align-items: center; // replace stretch / v. alignment
}

.has-lines:before, .has-lines:after {
    content: ''; 
    display: inline-block;
    vertical-align: middle;
    flex-grow: 1; // take all the space you can
    height: 1px;
    background: #ccc;
    min-width: 20px;
}
```
• **CSS:**

```css
.has-lines {
  display: flex;
  justify-content: center; // horizontal alignment
  align-items: center; // replace stretch / v. alignment
}

.has-lines:before, .has-lines:after {
  content: ''; 
  display: inline-block;
  vertical-align: middle;
  flex-grow: 1; // take all the space you can
  height: 1px;
  background: #ccc;
  min-width: 20px;
}

.has-lines:before { margin-right: 20px; }
.has-lines:after { margin-left: 20px; }
```
Example Title

Fusce dapibus, tellus ac cursus commodo, tortor mauris

or

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus donec sed odio dui.

Action here

Uses flexbox to avoid the need for nested elements, or using overflow hidden to mask super-long absolutely-positioned lines.

This technique works in all decent browsers, including IE11+. IE10 requires some legacy flexbox syntax and some extra tweaks but also displays as expected.

IE9 and earlier show partial lines (at the pseudo-element’s min-width), but still looks passable. If content wraps to multiple lines, the after
So you want to implement a baseline rhythm in CSS. Insert two differently formatted elements next to each other and they’ll seem out of phase. How do we bring them under control?

— Jan Dudek
Success usually comes to those who are too busy to be looking for it. —J.H. Thoreau
Often we define a common *line height* value (or its multiple) that’s used for all elements, including their *paddings* and *margins*, occasionally taking *border widths* into the equation.
What if we _align the baseline_ instead? So that all type—regardless of its size—lies on the same _grid line_? We just need to calculate the _offset_ and then shift the content by that offset.
Success usually comes to those who are too busy to be looking for it. —J.H. Thoreau
By default, browsers center the cap height between grid lines (the height of a capital letter above the baseline). So we shift it by half the difference between line height and cap height.
To determine the *cap height*, we fiddle with *offset values* until the type is properly aligned with the grid.
vertical-rhythm.scss:

$line-height: 24px;

$font-stacks: (s: $font-stack-text,
               m: $font-stack-text,
               l: $font-stack-display,
               xl: $font-stack-display);

$font-sizes: (s: 13px, m: 15px, l: 19px, xl: 27px);
$cap-heights: (s: 0.8, m: 0.8, l: 0.68, xl: 0.68);
$line-height: 24px;

$font-stack-text: (  
  s: $font-stack-text,  
  m: $font-stack-text,  
  l: $font-stack-display,  
  xl: $font-stack-display
);

/font-sizes: (s: 13px, m: 15px, l: 19px, xl: 27px);
$cap-heights: (s: 0.8, m: 0.8, l: 0.68, xl: 0.68);

@function rhythm-shift($size-name) {
  $font-size: map-get($font-sizes, $size-name);
  $cap-height: map-get($cap-heights, $size-name);
  $offset: ($line-height - $cap-height * $font-size) / 2;
  return round($offset);
}
Now we just need to apply the offset, and do so reliably. We can combine positive top margin and negative bottom margin to make it work.
$offset: rhythm-shift(m);

.rhythm-m {
    margin-top: $offset;
    margin-bottom: -1 * $offset;
}
• **Collapsing** works differently with positive and negative margins:

  • *Two positive margins*
    The bigger one wins.
  
  • *Two negative margins*
    The lower (i.e. the *more* negative) wins.
  
  • *One positive, one negative margin*
    The margins *sum up.*
If an element doesn’t have a `border nor padding`, and its first child has a margin, that margin will `flow out` of the parent. Use `overflow: hidden`. 
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque gravida eget massa a tincidunt.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi interdum elit at sapien vestibulum sodales.

Morbi efficitur et ipsum sit amet ullamcorper.

Curabitur vel est eget sapien scelerisque efficitur a sit amet risus. Sed scelerisque tortor mi, vel venenatis odio tempor eu. Vestibulum a aliquet nibh.

Phasellus mattis vulputate felis, eget porta lectus maximus ac.

Proin sed mi vitae diam tempus pharetra.
Implementing baseline rhythm in CSS

Written by Jan Dudek

15 June 2016

Vertical rhythm is a typographic concept that's so often misunderstood by front-end engineers.

By aligning type to a vertical grid, designers can make their work look harmonious and clean. And by having the same visual rhythm implemented correctly, front-end architects can achieve consistent, good-looking results more easily and in shorter time. All that without the need for designers’ input in the process.
What’s the best way to *encode and compress* a 1×1px image? It could be useful as a placeholder or default image, for example.

— Jon Sneyers
Uncompressed pixel is just one bit to four bytes – depending on how we define it: as black & white (1 bit), grayscale (1 byte), grayscale + alpha (2 bytes), RGB (3 bytes), or RGBA (4 bytes).
Every image format specifies how to interpret the data: the width and height of the image, and the number of bits or bytes per pixel (headers).
• **Image formats** contain headers with some metadata:

  • *Magic number*
    A fixed identifier, e.g. GIF87A/GIF89a, JFIF, Exif, PNG.

  • *Decoding details*
    Color profiles, orientation, gamma, or dots-per-pixel.

  • *Arbitrary metadata*
    Timestamps, copyright notices, or GPS coordinates.

  • *“Overhead” stuff*
    Markers and checksums or paddings for robustness.
Here is a summary table that gives the (optimal) file sizes for various one-pixel images:

<table>
<thead>
<tr>
<th>Format</th>
<th>white</th>
<th>black</th>
<th>gray</th>
<th>yellow #FFFF00</th>
<th>transparent</th>
<th>semitransparent #1337BABE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNG</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>69</td>
<td>68</td>
<td>70</td>
</tr>
<tr>
<td>GIF</td>
<td>35</td>
<td>35</td>
<td>43</td>
<td>35</td>
<td>43</td>
<td>/</td>
</tr>
<tr>
<td>JPEG</td>
<td>160</td>
<td>160</td>
<td>159</td>
<td>288</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Lossy WebP</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>64</td>
<td>82</td>
<td>92</td>
</tr>
<tr>
<td>Lossless WebP</td>
<td>38</td>
<td>34</td>
<td>38</td>
<td>36</td>
<td>34</td>
<td>38</td>
</tr>
<tr>
<td>Lossy BPG</td>
<td>31</td>
<td>31</td>
<td>29</td>
<td>36</td>
<td>57</td>
<td>62</td>
</tr>
<tr>
<td>Lossless BPG</td>
<td>59</td>
<td>59</td>
<td>37</td>
<td>124</td>
<td>113</td>
<td>160</td>
</tr>
<tr>
<td>FLIF</td>
<td>15</td>
<td>14</td>
<td>15</td>
<td>18</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>PNM/PAM</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>14</td>
<td>67</td>
<td>69</td>
</tr>
</tbody>
</table>
And here are the numbers for a huge 268-megapixel black square:

<table>
<thead>
<tr>
<th>Format</th>
<th>Bytes</th>
<th>Pixels per bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBM</td>
<td>33,552,399</td>
<td>1.00</td>
</tr>
<tr>
<td>PNG</td>
<td>32,645</td>
<td>1,027.73</td>
</tr>
<tr>
<td>GIF</td>
<td>182,225</td>
<td>184.11</td>
</tr>
<tr>
<td>JPEG</td>
<td>1,048,737</td>
<td>31.99</td>
</tr>
<tr>
<td>Lossy WebP</td>
<td>477,334</td>
<td>70.29</td>
</tr>
<tr>
<td>Lossless WebP</td>
<td>10,366</td>
<td>3,236.57</td>
</tr>
<tr>
<td>Lossy BPG</td>
<td>16,734</td>
<td>2,004.90</td>
</tr>
<tr>
<td>Lossless BPG</td>
<td>18,032</td>
<td>1,860.60</td>
</tr>
<tr>
<td>FLIF</td>
<td>15</td>
<td>2,236,689</td>
</tr>
</tbody>
</table>
One pixel is worth three thousand words

By Jon Sneyers  •  Jul 20, 2016

A couple of months ago while taking a break from implementing cool new features like q_auto and g_auto, I was joking in our team chat about how well various image formats “compress” one-pixel images. In response, Orly — who runs the blog — asked me if I’d write a post about single-pixel images. I said: “Sure, why not. But it will be a very short blog post. After all, there’s not much you can say about a single pixel.”

Looks like I was wrong. Very wrong.
What if you have to translate an interface into *many languages*? The length of words is unpredictable. How do you manage it across devices?

— Michael Scharnagl
The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.
Rindfleischetikettierungsüberwachungsauflagenübertragungsgesetz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.
The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜÄÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.
Rindfleischetikettierun...

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAUG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.
Conclusion

I tested all the examples above and combinations of them in IE7, IE8, IE9, IE10, IE11, Edge, Firefox 39 (Windows, Linux, Mac), Chrome 44 (Windows, Linux, Mac), Opera 30 (Windows, Mac), Safari 8 (Mac), Safari 5.1 (Windows), Android 5 (Nexus 6), Android 4.4 (Nexus 5), Android 2.3 (Galaxy S2), iOS 8.3 (iPhone 6), iOS 7 (iPhone 5S), iOS 6 (iPhone5), Opera Mini (Android 5), Opera Classic (Android 5), Opera Mobile (Android 5) and Windows Phone 8.1 (Lumia 930) using real devices and BrowserStack – Here is a list of all 26 browsers and their results.

When searching the web you probably will find the following solution:

```css
.hyphenate {
    -ms-word-break: break-all;
    word-break: break-all;
    word-break: break-word;
    -webkit-hyphens: auto;
    -moz-hyphens: auto;
    hyphens: auto;
}
```

While this works great in most cases, I found out that in Firefox hyphens won’t work (although I will be working on this with the Firefox team in the near future) and have been having success with CodeMirror.
# CSS3 Overflow-wrap

Allows lines to be broken within words if an otherwise unbreakable string is too long to fit. Currently mostly supported using the `word-wrap` property.

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Edge</td>
<td>Firefox</td>
<td>Chrome</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>41</td>
<td>46</td>
</tr>
<tr>
<td>10</td>
<td>42</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>40</td>
<td>45</td>
</tr>
</tbody>
</table>

Partial support refers to requiring the legacy name "word-wrap" (rather than "overflow-wrap") to work.
Dealing with long words in CSS

July 31, 2015

The web consists of content, content consists of words and words can be long, very long. Everyone involved with the web will sooner or later have to deal with long words.

Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.
Dealing with "above-the-fold" CSS can be annoying — it has to be maintained and can’t be cached properly. *Is there a better way*?
Deploying Critical CSS

• With **HTTP/2**, HTTP-requests are *cheap*. Use the “scout” approach with *many small files*!

• **Inlining critical CSS** is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*
Deploying Critical CSS

- **Inlining critical CSS** is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*

- We’ve moved away from...

```html
<head>
  <link rel="stylesheet" href="combined.min.css">
</head>
<body>
  ...content...
</body>
```
Deploying Critical CSS

• *Inlining critical CSS* is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*

• ...towards:

```html
<head>
    <meta name="fullcss" content="full.css">
    <!-- #if expr="$HTTP_COOKIE=/fullcss\=true/" -->
    <link rel="stylesheet" href="full.css">
    <!-- #else -->
    <style> /* Critical CSS styles */ </style>
    <noscript><link rel="stylesheet" href="full.css"></noscript>
    <!-- #endif -->
</head>
```
Deploying Critical CSS

...and then:

```html
<head>
<!-- #if expr="$HTTP_COOKIE=/fullcss\=true/" -->
<link rel="stylesheet" href="full.css">
<!-- #else -->
<style>
/* Critical CSS styles, plus: */
article, .comments, aside, footer { display: none; }
</style>
<script>
loadCSS("full.css"); /* or rest.css + critical.css */
</script>
<noscript><link rel="stylesheet" href="full.css"></noscript>
<!-- #endif -->
</head>
```
Deploying Critical CSS

- A simple, “recommended” “HTTP/2” way:

  `<head>
      <link rel="stylesheet" href="site-header.css">
      <link rel="stylesheet" href="article.css">
      <link rel="stylesheet" href="comments.css">
      <link rel="stylesheet" href="sidebar.css">
      <link rel="stylesheet" href="site-footer.css">
  </head>

  <body>
      ...content...
  </body>`
Deploying Critical CSS

• But “progressive CSS” way is even better:

```html
<head>...</head>
<body>
  <!-- HTTP/2 push critical or inline; whatever faster -->
  <link rel="stylesheet" href="site-header.css">
  <header>...</header>

  <link rel="stylesheet" href="article.css">
  <main>...</main>

  <link rel="stylesheet" href="comments.css">
  <section class="comments">...</section>
  ...content...
</body>
```
Deploying Critical CSS

- *Multi-Stage CSS loading* removes the need for critical CSS, and provides *sequential* rendering.

- *Browser behavior* supports the technique; browsers block rendering when necessary.
The future of loading CSS

Posted 11 February 2016 - totally overshadowed by some waving gravity thing. Thanks Einstein.

Chrome is intending to **change the behaviour of `<link rel="stylesheet">`**, which will be noticeable when it appears within `<body>`.

The impact and benefits of this aren't clear from the blink-dev post, so I wanted to go into detail here.

The current state of loading CSS

```html
<head>
  <link rel="stylesheet" href="/all-of-my-styles.css">
</head>
<body>
  ...content...
</body>
```

CSS blocks rendering, leaving the user staring at a white screen until `all-of-my-styles.css` fully downloads.

It's common to bundle all of a site's CSS into one or two resources, meaning the user downloads a large number of rules that don't apply to the current page. This is because...
You’ve built a *perfect grid* with *perfectly sized* thumbnail images (e.g. squared size), but when the client uploads images with incorrect dimensions, they are squished into the rectangle, incorrectly resized. How do you manage this issue?
original images
Base style: img { width: 200px; height: 200px; border: 1px solid; background: #eee; }

Images squished
For `img src`, we can use `object-fit` property in CSS to "letterbox" the images, preserving the ratio, or crop the images inside the block. For background images, we can apply `background-size` exactly the same way.
Base style + img { object-fit: contain }

Images letterboxed, correct aspect ratio maintained
Base style + img { object-fit: cover; }

Images expand to cover width and height, correct aspect ratio maintained
Base style + img { object-fit: none; }

Images expand to actual size, cropped inside set width and height, correct aspect ratio maintained
Method of specifying how an object (image or video) should fit inside its box.

<table>
<thead>
<tr>
<th>Browser</th>
<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></td>
<td>8</td>
<td>33</td>
<td>31</td>
<td>35</td>
<td>35</td>
<td>31</td>
<td>31</td>
<td>4.1</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>34</td>
<td>37</td>
<td>35</td>
<td>35</td>
<td>37</td>
<td>37</td>
<td>4.3</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>35</td>
<td>39</td>
<td>36</td>
<td>39</td>
<td>36</td>
<td>36</td>
<td>4.4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>36</td>
<td>40</td>
<td>36</td>
<td>40</td>
<td>36</td>
<td>36</td>
<td>4.4</td>
<td>40</td>
</tr>
</tbody>
</table>

Notes: Known issues (0) Resources (6) Feedback
Polyfill for CSS object-fit property to fill-in/fit-in images into containers.

v0.3.7

- **dist**: Update usage of indexed style properties regarding
- **src**: Update usage of indexed style properties regarding
- **tests**: Add new info for local testing.
- **.bowerrc**: Initial commit
- **.editorconfig**: add pkg things
- **.gitattributes**: Initial commit
- **.gitignore**: add html classes, fix tests
- **.jshintrc**: add pkg things
- **CHANGELOG.md**: Update README with latest browser developments
- **CONTRIBUTING.md**: Initial commit
- **Gruntfile.js**: Typo, whitespace & updated main array
The Widescreen Web: Using CSS object-fit

Estimated reading time: 4 minutes, 15 seconds

Cet article est également disponible en français

Most everyone is familiar with the fact that movies are letterboxed, rescaled or cropped as they move from the theater to televisions and tablets. As a web developer, you’ll also be aware of the different ways background images can be made responsive on web pages, stretching to cover the viewport, or being cropped as the browser window resizes.

An obvious gap in web design is the lack of the same intelligent and automatic resizing rules for images and videos. That’s the role of CSS object-fit.
The CSS3 object-fit and object-position Properties

Introduction

A common problem in CSS concerns how to control the aspect ratio of replaced elements, such as `<img>` or `<video>` elements. For example, you might want to have all images occupy the same space on a page, but not distort and lose their aspect ratio when someone uses an image file that isn’t quite the right size. Resizing and letter-boxing the image slightly to conserve the aspect ratio is...
What if you wanted the color of the SVG icon to *inherit* the color property of a button in which it resides? Can we use CSS alone (no SASS/LESS) to establish this relationship?

— Osvaldas Valutis
Settings

Settings

Settings

normal

:hover, :focus

:active
This is a fancy link styling.
Klaipėda is a city in Lithuania situated at the mouth of the Danė River where it flows into the Baltic Sea. It is the third largest city in Lithuania and the capital of Klaipėda County. The city has a complex recorded history, partially due to the combined regional importance of the Port of Klaipėda, a usually ice-free port on the Baltic Sea, and the Akmena – Danė River.

It has been controlled by the Teutonic Knights, the Duchy of Prussia, the Kingdom of Prussia, the German Empire, the Entente States immediately after World War I, Lithuania as a result of the 1923 Klaipėda Revolt, and the Third Reich following the 1939 German ultimatum to Lithuania. The city was incorporated into Lithuania during its tenure as a Soviet Socialist Republic and has remained within Lithuania following its re-establishment as an independent state.
SVG

This is my favorite. Take a very common example on the Web — a button with SVG icon and a title in it. I have these here on my website too:

Of course, you are a very responsible web designer, and you style :hover, :focus, :active states of the button for a better interaction with a user. This is how your code usually looks like:
Of course, you are a very responsible web designer, and you style :hover, :focus, :active states of the button for a better interaction with a user. This is how your code usually looks like:

```css
.button
{
  color: #000;
  border: 2px solid #000;
}
.button:hover, .button:focus
{
  color: #333;
  border-color: #333;
}
.button:active
{
  color: #666;
  border-color: #666;
}
.button svg
{
  fill: #000;
}
.button:hover svg, .button:focus svg
{
  fill: #333;
}
.button:active svg
{
  fill: #666;
}
```
Currently I am writing a front-end code for client’s e-commerce website which has a few different button designs. Moreover, there are anchors that have `:visited` state styled in addition. And there are many more similar SVG usage cases (toolbars, etc.) where SVG has to have the color of the text. `currentColor` helps to reduce the code twice:

```css
/* put this in your reset-normalize-defaults.css file */

svg
{
    fill: currentColor;

}

/* now you don't have to style SVG and border-color at all */

.button
{
    color: #000;
    border: 2px solid currentColor;
}

.button:hover,
.button:focus
{
    color: #333;
}

.button:active
{
    color: #666;
}
```
Keeping CSS short with `currentColor`

27 Nov

Turns out `currentColor` has been here for quite some time now, but I heard about it only a few months ago when I read Dudley Storey’s post. He states that it is supported very well across the browsers (IE9+). This was enough for me to start using it in production. I was quite surprised how useful the keyword is: it helps to keep CSS code shorter and smarter.

Before diving into practical usage examples, here is a short theory course. This is how MDN describes `currentColor`:

>The `currentColor` keyword represents the calculated value of the element’s `color` property. It allows to make the color properties inherited by properties or child’s element properties that do not inherit it by default.

SVG

This is my favorite. Take a very common example on the Web — a button with SVG icon and a title in it. I have these here on my website too:
# CSS currentColor value

A CSS value that will apply the existing color value to other properties like background-color, etc.

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Firefox</td>
<td>Chrome</td>
<td>Safari</td>
</tr>
<tr>
<td>31</td>
<td>36</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>40</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Opera</td>
<td>iOS Safari</td>
</tr>
<tr>
<td>27</td>
<td>7.1</td>
<td>8.1</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
</tbody>
</table>

Notes: No notes

Known issues: 0

Resources: 3

Feedback:
HTTP cache is unreliable. Is there any way to keep assets such as fonts, sprites or CSS/JS files in the cache and improving performance for subsequent visits in general?
Service Workers: Foundation

- **Service worker** is a background process running JS. It behaves like a proxy server on a user’s device.
  - Websites *register a service worker*; it then intercepts all network traffic between the site and the outside world.
  - Once installed, browsers *dispatch events* to the service worker first, for each network request.
- **Goal**: a reliable “native” proxy that would deliver content when users are both online and *offline*. 

```javascript
var size = window.getComputedStyle(document.body, ':after').getPropertyValue('content');
if (size == 'desktop') {
  // Load some more content.
}
```
Offline Strategy

• Depending on *complexity*, we might use offline caching, background processes, push notifications...

• A common offline strategy is quite *simple*:
  — *Explicitely cache* resources like CSS, JS, fonts, images;
  — *Cache the homepage* to display it when network fails;
  — For other pages, have a “fallback” offline page.

• We need to *register a Service worker* first.
Offline Strategy

• We need to register a Service worker first.

• JavaScript:

```javascript
if (navigator.serviceWorker) {
    navigator.serviceWorker.register(
        '/sw.js', {
            scope: '/'
        });
}
```

• sw.js sits in the root to act on any request; otherwise it acts on requests to a directory in which it resides.
Offline Strategy

- JavaScript:
  ```javascript
  if (navigator.serviceWorker) {
    navigator.serviceWorker.register(
      '/sw.js', {
        scope: '/'
      }
    );
  }
  ```

- `sw.js` sits in the root to act on *any* request; otherwise it acts on requests to a directory in which it resides.

- Caching of `sw.js` is capped for 24h via Cache-Control. Browsers will review the file when accessing the site.
Offline Strategy

- `sw.js` sits in the root to act on *any* request; otherwise it acts on requests to a directory in which it resides.

- Next, we need to *install* the *registered service worker*.

```javascript
self.addEventListener('install', function (event) {
    event.waitUntil(updateStaticCache());
});

// Update 'version' if you need to refresh the cache
var staticCacheName='static';
var version='v1::';
```
Offline Strategy

• Next, we need to install the registered service worker.

```javascript
self.addEventListener('install', function (event) {
  event.waitUntil(updateStaticCache());
});

// Update 'version' if you need to refresh the cache
var staticCacheName='static';
var version='v1::';
```

• By adding a version number (version), we can easily update the cache later just by updating the version number.
Offline Strategy

- We’ll populate `cache` with `updateStaticCache()`.

```javascript
// Store core files in a cache (incl. 'offline' page)
function updateStaticCache() {
    return caches.open(version + staticCacheName)
        .then(function (cache) {
            return cache.addAll(['js/scripts.js',
                                'css/styles.css',
                                'images/logo.svg',
                                'fonts/webfont.woff',
                                '/
                                '/offline'])
        });
}
```
Offline Strategy

• Now we can *activate* the service worker. We’ll clean up any outdated caches (checking the version number).

```javascript
self.addEventListener('activate', function (event) {
  event.waitUntil(
    caches.keys()
      .then(function (keys) {
        // Remove caches whose name is no longer valid
        return Promise.all(keys
          .filter(function (key) {
            return key.indexOf(version) !== 0;
          })
          .map(function (key) {
            return caches.delete(key);
          })
      ));
});
});
```
Offline Strategy

- **Register** a service worker ✓
- **Install** a service worker ✓
- **Versioning** setup for a service worker ✓
- **Cache population** setup for a service worker ✓
- **Activate** a service worker ✓
- **Managing cache** for a service worker ✓
- **Intercepting requests** with a service worker.
Offline Strategy

• The `fetch` event is fired every time the browser is going to request a file. We can intercept that request:

```javascript
self.addEventListener('fetch', function (event) {
  var request = event.request;
  // Always fetch non-GET requests from network
  if (request.method !== 'GET') {
    event.respondWith(
      fetch(request)
        .catch(function () {
          return caches.match('offline.html');
        }));
  }
  return;
});
```
Offline Strategy

- For *HTML requests*, try the *network* first. If it fails, try to fetch *cache*. If all fails, show the *offline* page.

- For *file requests*, try to fetch files from *cache* first. If it fails, make a *network* request. If it fails, use *fallback*. 
if (request.headers.get('Accept').indexOf('text/html') !== -1) {
    event.respondWith(
        fetch(request)
            .then(function (response) {
                // Put a copy of this page in the cache
                var copy = response.clone();
                caches.open(version + staticCacheName)
                    .then(function (cache) {
                        cache.put(request, copy);
                    });
                return response;
            })
            .catch(function () {
                return caches.match(request)
                    .then(function (response) {
                        return response || caches.match('/offline.html');
                    });
            });
    return;
}
if (request.headers.get('Accept').indexOf('text/html') !== -1) {
    event.respondWith(
        fetch(request)
        .then(function (response) {
            // Put a copy of this page in the cache
            var copy = response.clone();
            caches.open(version + staticCacheName)
                .then(function (cache) {
                    cache.put(request, copy);
                });
            return response;
        })
        .catch(function () {
            return caches.match(request)
                .then(function (response) {
                    return response || caches.match('/offline.html');
                });
        })
    );
    return;
}
Offline Strategy

• For *HTML requests*, try the *network* first. If it fails, try to fetch *cache*. If all fails, show the *offline* page.

• For *file requests*, try to fetch files from *cache* first. If it fails, make a *network* request. If it fails, use *fallback*.
event.respondWith(
  caches.match(request)
    .then(function (response) {
      return response || fetch(request)
        .catch(function () {
          // If request is img, show offline placeholder
          if (request.headers.get('Accept').indexOf('image') !== -1) {
            return new Response(
              '<svg.. />', { headers: { 'Content-Type': 'image/svg+xml' } }
            );
          }
        });
    })
);
Offline Strategy

- **Register** a service worker ✓
- **Install** a service worker ✓
- **Versioning** setup for a service worker ✓
- **Cache population** setup for a service worker ✓
- **Activate** a service worker ✓
- **Managing cache** for a service worker ✓
- **Intercepting requests** with a service worker ✓
Service Workers Gotchas

- Service workers *require* a secure connection (HTTPS) and are based on ES6 Promises.

- *Cache API* is completely separate from HTTP cache. *Response* is a stream; to cache it, you need to copy it first.

- A service worker can have *multiple caches* and can be *registered* many times; the browser will figure it out.
Service Workers Gotchas

• If a service worker is *broken*, browsers will skip the code and fall back to the network.

• Service workers use only *asynchronous APIs*. E.g. they can’t work with localStorage (*synchronous*).
A promise represents the eventual result of an asynchronous operation.

<table>
<thead>
<tr>
<th>Browser</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Edge</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Firefox</td>
<td>40</td>
<td>42</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>Chrome</td>
<td>43</td>
<td>46</td>
<td>46</td>
<td>49</td>
</tr>
<tr>
<td>Safari</td>
<td>44</td>
<td>45</td>
<td>47</td>
<td>49</td>
</tr>
<tr>
<td>Opera</td>
<td>8</td>
<td>9</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>8.4</td>
<td>9.1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>4.1</td>
<td>4.3</td>
<td>4.3</td>
<td>4.4</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.1</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4</td>
</tr>
</tbody>
</table>

*Global usage share statistics based on data from StatCounter GlobalStats for October, 2015. See the browser usage table for usage by browser version.*

By @Pyrd, design by @Lensco. Location detection provided by ipinfo.io.
A polyfill for ES6-style Promises

- 434 commits
- 3 branches
- 12 releases
- 51 contributors

Latest commit 7d90241 28 days ago by stefanpenner

- bin
- config
- dist
- lib
- server
- test
- vendor
- .gitignore
- .jshintrc
- .release.json
- .spmignore
- .travis.yml
- Broccoli.js
- CHANGELOG.md

- sync with RSVP.
- sync with RSVP.
- release v3.0.2 & bump both bower and package.json
- fix typo: replace fulfillment/vertex to fulfillment/Vertex
- fix, simplify and cleanup the build tooling.
- reduce timeout
- update es6 module transpiler
- 2.1.0
- sync with RSVP.
- sync with RSVP.
- support npm
- Use container based infrastructure on travis-ci
- Mangle and compress the minified version
- Fill in changelog from 2.1.0 -> 2.3.0

You can clone with HTTPS or Subversion.
**Lightweight promise polyfill for the browser and node. A+ Compliant**

<table>
<thead>
<tr>
<th>Branch: <strong>master</strong></th>
<th><strong>promise-polyfill</strong> / +</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tests</strong></td>
<td>Removed unnecessary test requires</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Added A+ Test suite to dev dependencies</td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Updated travis.yml to node .10</td>
</tr>
<tr>
<td>Gruntfile.js</td>
<td>Added grunt for minify. Removed setImmediate polyfill. Updated readme.</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Added license file</td>
</tr>
<tr>
<td>Promise.js</td>
<td>Updated version to 2.1.0</td>
</tr>
<tr>
<td>Promise.min.js</td>
<td>Updated version to 2.1.0</td>
</tr>
<tr>
<td>README.md</td>
<td>Updated version to 2.1.0</td>
</tr>
<tr>
<td>bower.json</td>
<td>Updated version to 2.1.0</td>
</tr>
<tr>
<td>jasmine.json</td>
<td>Fixed Promise.immediateFn</td>
</tr>
<tr>
<td>package.json</td>
<td>Updated version to 2.1.0</td>
</tr>
<tr>
<td><strong>51 commits</strong></td>
<td><strong>1 branch</strong></td>
</tr>
<tr>
<td>9 releases</td>
<td>2 contributors</td>
</tr>
</tbody>
</table>

Latest commit 7ee8e28 on Jun 25

HTTPS clone URL: [https://github.com/taylorhakes/promise-polyfill](https://github.com/taylorhakes/promise-polyfill)
You’ve gone incognito

Pages you view in Incognito tabs won’t stick around in your browser’s history, cookie store, or search history after you’ve closed all of your incognito tabs. Any files you download or bookmarks you create will be kept.

However, you aren’t invisible. Going Incognito doesn’t hide your browsing from your employer, your internet service provider, or the websites you visit.
Service Workers — WD

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari</td>
<td>48</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Opera</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS Safari</td>
<td>50</td>
<td>48</td>
<td>8</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>49</td>
<td></td>
<td>9.2</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4, 4.4</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

Notes: Known issues (0) Resources (8) Feedback

Details on partial support can be found on is ServiceWorker Ready?

MS Edge status: In Development
WebKit status: Under Consideration

1 Partial support can be enabled in Firefox with the `dom.serviceWorkers.enabled` flag.
My first Service Worker

November 7th, 2015

I've made no secret of the fact that I'm really excited about Service Workers. I'm not alone. At the Coldfront conference in Copenhagen, pretty much every talk mentioned Service Workers.

Obviously I'm excited about what Service Workers enable: offline caching, background processes, push notifications, and all sorts of other goodies that allow the web to compete with native. But more than that, I'm really excited about the way that the Service Worker spec has been designed. Instead of being an all-or-nothing technology that you have to bet the farm on, it has been deliberately crafted to be used as an enhancement on top of existing sites (oh, how I wish that web components would follow a similar path).

I've got plenty of ideas on how Service Workers could be used to enhance a community site like The Session or the kind of events sites that we produce at Cleft, but to begin with, I figured it would make sense to use my own personal site as a playground.

To start with, I've already conquered the first hurdle: serving my site over HTTPS. Service Workers require a secure connection. But you can play around with running a Service Worker locally if you run a copy of your site on localhost.

That's how I started experimenting with Service Workers: serving on localhost, and stopping and starting my local Apache server with `apachectl stop` and `apachectl start` on the command line.
What is a Service Worker?

Rich offline experiences, periodic background syncs, push notifications—functionality that would normally require a native application—are coming to the web. Service workers provide the technical foundation that all these features will rely on.

What is a Service Worker?

A service worker is a script that is run by your browser in the background, separate from a web page, opening the door to features which don’t need a web
A node module to generate service worker code that will precache specific resources.

### Repository Details

- **375 commits**
- **7 branches**
- **4 releases**
- **14 contributors**

### Commits

- **jeffposnick** Added comments to sw-precache config.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>app-shell-demo</td>
<td>Added comments to sw-precache config.</td>
<td>8 days ago</td>
</tr>
<tr>
<td>demo</td>
<td>Properly read in runtimeCaching option from external config file (#110)</td>
<td>2 months ago</td>
</tr>
<tr>
<td>lib</td>
<td>Adds a dontCacheBustUrlsMatching option.</td>
<td>12 days ago</td>
</tr>
<tr>
<td>test</td>
<td>Adds a dontCacheBustUrlsMatching option.</td>
<td>12 days ago</td>
</tr>
<tr>
<td>.eslintrc</td>
<td>Switching over to google linting rules</td>
<td>7 months ago</td>
</tr>
<tr>
<td>.eslintrc</td>
<td>Switching over to google linting rules</td>
<td>7 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Updated to use paths under demo/</td>
<td>2 years ago</td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Run Travis against node 0.12 and the latest stable release (#130)</td>
<td>17 days ago</td>
</tr>
<tr>
<td>GettingStarted.md</td>
<td>User regexp literals instead of new RegExp</td>
<td>5 months ago</td>
</tr>
</tbody>
</table>

- 219 commits
- 10 branches
- 13 releases
- 11 contributors

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
<th>Latest commit</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>doc-pages: Fixing short link</td>
<td>79ade01</td>
<td>3 days ago</td>
</tr>
<tr>
<td></td>
<td>lib: Follow up to #101. Ensure that the <code>RegExp</code> origin always exists.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>project: Fixes #143</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>recipes/cache-expiration-options: Adding support for auto-building docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>test: automated-suite: minor comment updates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.eslintrc: Move to the Google ESLint config, and fix the linting errors.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.gitignore: Updating for new sw-testing-helpers (#128)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>npmignore: Don’t exclude the library itself from npm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.travis.yml: Updating of tests</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introduction

The Service Worker Cookbook is a collection of working, practical examples of using service workers in modern web apps.

Tip: Open your Developer Tools console to view `fetch` events and informative messages about what each recipe's service worker is doing!

Attribution

The Service Worker Cookbook was created by Mozilla with contributions from developers like you. All source code is available on GitHub. Contributions and requests welcome.

Recipes

Immediate Claim

This recipe shows how to have the service worker immediately take control of the page without waiting for a navigation event.

Message Relay

This recipe shows how to communicate between the service worker and a page and shows how to use a service worker to relay messages between pages.
The offline cookbook

Update: Together with Udacity I created a free offline-first interactive course. It involves taking an online-only site to full offline-first glory. Many of the patterns in this article are used.

When AppCache arrived on the scene it gave us a couple of patterns to make content work offline. If those were the patterns you needed, congratulations, you won the AppCache lottery (the jackpot remains unclaimed), but the rest of us were left huddled in a corner rocking back & forth.

With ServiceWorker (intro) we gave up trying to solve offline, and gave developers the moving parts to go solve it themselves. It gives you control over caching and how requests are handled. That means you get to create your own patterns. Let's take a look at a few possible patterns in isolation, but in practice you’ll likely use many of them in tandem depending on URL & context.

All code examples work today in Chrome & Firefox, unless otherwise noted. For full details on service worker support, see "Is Service Worker Ready?".
is ServiceWorker ready?

Status * Spec * Intro * Resources * GitHub

ServiceWorker enthusiasm

The first thing any implementation needs.

Safari: No public commitment
Edge: Under consideration, but positive signals.

Promises

Not ServiceWorker-specific, but required by ServiceWorker. Spec.

Safari: Promise.resolve(promise) creates a new Promise instance (fixed in the nightlies)
Edge: Preview build.
Is there any way to isolate expensive components, similar to lazy loading, and paint important content faster using CSS alone?

— Michael Scharnagl
The *contain* property is a primitive for isolating style, layout and paint. It allows us to limit a specific DOM sub-tree and the rest of the document with *native boundaries*. 
With the `contain` property, we can define priorities for loading and painting CSS components.

- **Third-party widgets**
  Delay expensive layout with `contain: strict`.

- **Off-screen modules**
  Delay expensive paints with `contain: paint`.

- **Container queries**
  Focus on the local scope with `contain: strict`. 
• With the `contain` property, we can define priorities for loading and painting CSS components.

  • *Third-party widgets*
    Delay expensive layout with `contain: strict`.

  • *Off-screen modules*
    Delay expensive paints with `contain: paint`.

  • *Container queries*
    Focus on the local scope with `contain: strict`.

• Browser support is coming.
  Enabled by default in Chrome 52.
CSS containment

April 5, 2016
by Michael Scharnagl

I haven’t heard about the contain property until some weeks ago when I asked about use cases for container queries on twitter and David Baron mentioned it in a response saying that contain: strict; can avoid many of the theoretical problems of container queries. Since then I read the specification and everything I found about it trying to understand it; Here is what I learned.

The contain property

It’s a primitive for isolating style, layout, and paint. The contain property allows developers to limit a specific DOM sub–tree and the rest of the document; You can think of it like an iframe. Much like an iframe, this boundary establishes a new layout root, ensuring that DOM changes in the sub–tree never trigger refloas in the parent document.
The CSS `contain` property lets developers limit the scope of the browser's styles, layout and paint work for faster and more efficient rendering.

### Browser Support

<table>
<thead>
<tr>
<th>Browser</th>
<th>IE</th>
<th>Edge</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></td>
<td>8</td>
<td>45</td>
<td>46</td>
<td>50</td>
<td>29</td>
<td>45</td>
<td>48</td>
<td>8.4</td>
<td>4.3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>13</td>
<td>47</td>
<td>51</td>
<td>9.1</td>
<td>1</td>
<td>49</td>
<td>8.4</td>
<td>4.4</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>14</td>
<td>48</td>
<td>52</td>
<td>10</td>
<td>1</td>
<td>49</td>
<td>8.4</td>
<td>4.4.4</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>53</td>
<td>TP</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes

- Enabled via the "Experimental Web Platform features" flag

**MS Edge status:** Under Consideration
When using *web fonts*, we want users to be able to access the content as fast as possible, yet also avoid irritating repaints and reflows in the browser. What would be the best *font loading strategy* today?
Elegant Overflow with CSS Ellipsis

text-overflow: ellipsis

overflow: hidden
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

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!
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** 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**.

- **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:**

  ```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** 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_Reg' 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.
<table>
<thead>
<tr>
<th></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>

- **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.
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 += ' fonts-ready';
  }).catch(function () {
    var docEl = document.documentElement;
    docEl.className += ' fonts-failed';
  });
```
• **JavaScript:**

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

• **CSS:**

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

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

• CSS:

```css
.fonts-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 += ' fonts-ready';
}).catch(function () {
    var docEl = document.documentElement;
    docEl.className += ' fonts-failed';
});
```

• CSS:

```css
.fonts-loaded h1 {
    font-family: "Elena Regular";
    // font-rendering: "block 0s swap infinite"; // FOUT
    font-rendering: "block 3s swap 3s"; // FOIT, at most 3sec
}
```
CSS Font Loading

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>Browser</th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opera</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS Safari</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opera Mini</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android Browser</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: Can be enabled in Firefox using the layout.css.font-loading-api.enabled flag. Enabled by default in Firefox 41. See this bug

Known issues (1)

Resources (4)

Feedback

*Global usage share statistics based on data from StatCounter GlobalStats for May, 2016. See the browser usage table for usage by browser version.*
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

`bramstein` 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:bramstein/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
What if you have a large photo that requires a transparent shadow? PNG is too large in file size, and JPEG isn’t good enough in quality. 

*Trick*: create a regular non-transparent JPG and an 8-bit PNG (alpha mask) and load both images *inside an SVG container*. 
• hero-image.svg:

```xml
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 560 1388">
  <defs>
    <mask id="canTopMask">
      <image width="560" height="1388" xlink:href="can-top-alpha.png">
      </image>
    </mask>
  </defs>
  <image mask="url(#canTopMask)" id="canTop" width="560" height="1388"
    xlink:href="can-top.jpg"></image>
</svg>
```
• **hero-image.svg:**

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     viewBox="0 0 560 1388">
  <defs>
    <mask id="canTopMask"/>
    <image width="560" height="1388" xlink:href="can-top-alpha.png"/>
  </mask>
</defs>

<image mask="url(#canTopMask)" id="canTop" width="560" height="1388"
       xlink:href="can-top.jpg"></image>
</svg>
```

• **HTML/CSS:**

```html
<img src="hero-image.svg" />, background: url("hero-image.svg")
```
Applying Alpha Channels to JPGs

In this article I will show how to combine the advantages of JPG's high compression rate with the flexibility of having alpha channels like we use them in PNG. Open the demo to see this technique in action, a semi transparent JPG in front of HTML text:

Demo of JPG with alpha channel.

As all modern browsers support images with alpha channels\(^1\), it has become fairly easy for designers to place semi transparent images in front of irregular backgrounds.

The weapon of choice will be PNG as it is the only widespread
JPG+PNG to SVG Mask

Combine the transparency of a PNG with the compression of a JPEG. Based on the idea from Using SVG to Shrink Your PNGs, but adapted to use data URIs instead of external images. Include on your page as inline SVG, using an `<img src="image.svg"/>` tag, or as a background-image.

Tested in the latest versions of Chrome, Firefox and Safari. This SVG technique’s compatibility via an `<img />` tag or as a background-image may not be perfect. See this pen to test on your browser. Inline seems to be the best option for compatibility, in which case you should use external assets so that they can be cached. Please fork or comment to improve.

To get started, upload two images:

- One as your primary image, named whatever (Try this one:)
- One as a mask *(a black and white PNG is best, just like this:)* with `-mask` or `-alpha` in the filename.

**Upload:**

Images:  Choose Files  No file chosen

Make sure the mask has `-mask` or `-alpha` in the filename.

**Example:**

![Image](image.png) ![Mask](mask.png) ![Output](output.png)
How do you *efficiently* scale up/down any UI component (e.g. a slider or calendar) and keep all the proportions intact—without fiddling with width, height or border-radius manually?

— @simurai
By sneaking a *Trojan horse* into your components. We use *rem* for components “root” and *em* for sub-parts of the components. Then, by adjusting the *font-size of the root*, we adjust *all* size-related CSS properties of a component at once.

— @simuraj
Let me show you in an example: For every CSS property that has a direct impact on the component’s size, you use the **EM** unit.

```
.Calendar {
    width: 5em;
    height: 2em;
    border-radius: .5em;
    border: 1px solid gold;
}
```

Note that the border is set to **1px** since it should stay always like that, unrelated to size changes.

In some cases you need to override the font-size that comes from the UA style sheet. For example when you use a `<button>` or `<input>` element. You can add a font-size of **100%**, **1em** or **inherit** to make it inherit from its parent. Or use something like [normalize.css](https://necolas.github.io/normalize.css) which already takes care of...
rem (root em) units

Type of unit similar to "em", but relative only to the root element, not any parent element. Thus compounding does not occur as it does with "em" units.

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Firefox</td>
<td>Chrome</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td>11</td>
<td>33</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>41</td>
</tr>
</tbody>
</table>

Notes: No notes

Known issues: (4)

Resources: (3)

Feedback:
What if you wanted to style a block with a light background color differently than if the same block has a dark background image? Or reverse text color based on bg color automatically in CSS?

— Osvaldas Valutis
```html
1. <h2>Sally Ride</h2> Normal Heading
```

```css
1. h2{
   background-size: cover;
   padding: 1em;
   line-height: 1;
   border: 1em solid #000;
   display: inline-block;
   vertical-align: top;
}
2. h2{style="url('http'){
   padding: 6em 1em .75em;
   color: #fff;
```

Sally Ride Normal Heading

Valentina Tereshkova Heading with image
Sally Ride

Normal Heading

Valentina Tereshkova

Heading with image
<h2>Sally Ride</h2>Normal Heading

Valentina Tereshkova Heading with image
BackgroundCheck

Automatically switch to a darker or a lighter version of an element depending on the brightness of images behind it.

Drag and Release elements over images
BackgroundCheck

Automatically switch to a darker or a lighter version of an element depending on the brightness of images behind it.

Drag and Release elements over images
An experimental festival celebrating independently produced art and technology. Read More.

September 10–13, 2015
Revolution Hall, Portland, Oregon

Sold Out
The `mix-blend-mode` property defines how an element’s content should blend with its background. Use `mix-blend-mode: difference` or `mix-blend-mode: darken` to have backgrounds, shapes, and text all interact in subtle/interesting ways.

— Robin Rundle
An experimental festival celebrating independently produced art and technology. Read More.

September 10–13, 2015
Revolution Hall, Portland, Oregon

Sold Out

**CSS:**

```css
.letterbox { background-color: #fff; }
.letterbox-letter { fill: #f7ff33;
  mix-blend-mode: darken;
/* The background is replaced with the color that is darker, otherwise it is left as it was. */ }
```
What if you want all links to have an underline except the ones you specify? Or you want all `<li>`’s in the navigation to have a right border, *except the last one*. Normally you would use `:last-child` (or extra class) to overwrite a default CSS rule.

— Ire Aderinokun
// instead of putting it on
border-right: 1px solid #424242;
 &:last-child {
  border-right: 0; // and then taking it off
}

// use CSS :not() to only apply to the elements you want
 &:not(:last-child) {
  border-right: 1px solid #424242;
}

🔥 Protip: Use CSS :not() instead of applying and unapplying borders on navigations. Supported wherever last-child is
For example, I may want all links on my site to have an underline, except ones which I specify. Normally, I would write -

```css
a {
    text-decoration: underline;
}

a.no-underline {
    text-decoration: none;
}
```

Doing this means that the links with the class .no-underline have the default styling unnecessarily applied to them. Using the :not selector, I can avoid this extra declaration -

```css
a:not(.no-underline) {
    text-decoration: underline;
}

a.no-underline {
    text-decoration: none;
}
```
You’ve built an alert message box. To be resilient to failure, how can we make sure that the box will be hidden when there is no content within it?

— Ire Aderinokun
The :empty pseudo-class is useful for elements which are styled in such a way that they appear regardless of whether they have content in them or not.

An example could be alert message boxes. We only want them to appear under certain circumstances, and the text content within them may be dynamically inserted and removed depending on the context. Using the :empty pseudo-class, we can make sure that the box will be hidden when there is no content within it.

```css
.alert {
    background-color: beige;
    border: 2px solid rgb(150, 150, 150);
    border-radius: 5px;
    padding: 5px 10px;
    display: inline-block;
}
.alert:empty {
    display: none;
}
```
CSS3 selectors - REC

Advanced element selection using selectors including:

- [foo^="bar"]
- [foo$="bar"]
- [foo*="bar"]
- :root
- :nth-child()
- :nth-last-child()
- nth-of-type
- nth-last-of-type()
- :last-child
- :first-of-type
- :last-of-type
- :only-child
- :only-of-type
- :empty
- :target
- :enabled
- :disabled
- :checked
- :not()
- ~ (general sibling)

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Edge</td>
<td>Firefox</td>
<td>Chrome</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>39</td>
<td>44</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td>11</td>
<td>41</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>42</td>
<td>46</td>
</tr>
<tr>
<td>13</td>
<td>43</td>
<td>43</td>
<td>47</td>
</tr>
<tr>
<td>14</td>
<td>44</td>
<td>44</td>
<td>48</td>
</tr>
</tbody>
</table>
Mobile browsers will wait approx. 300ms from the time that you tap the button to fire the click event. The reason for this is that the browser is waiting to see if you are actually performing a double tap.
Polyfill to remove click delays on browsers with touch UIs

- 395 commits
- 7 branches
- 21 releases
- 41 contributors

Branch: master

- examples
  - Fix typo in example test page: s/instead/instead/

- lib
  - Merge remote-tracking branch 'origin/master' into pr-257

- tests
  - bug fix for: ftlabs/fastclick#312

- .gitignore
  - Add script to build minified library (fix #66)

- .npmignore
  - Ignore examples, tests, etc

- LICENSE
  - Update license and maintainers

- Makefile
  - Add test task

- README.md
  - Merge pull request #314 from Rowno/patch-1

- bower.json
  - Version in bower.json file redundant

- component.json
  - Update the other package manager's version numbers

- package.json
  - 1.0.6

Latest commit 9977a91 on Jan 26, 2015
The *touch-action* property in CSS can be used to disable this behaviour. With *touch-action: manipulation*, the user agent may consider touches that begin on the element only for the purposes of scrolling and continuous zooming.
**CSS touch-action property**

touch-action is a CSS property that controls filtering of gesture events, providing developers with a declarative mechanism to selectively disable touch scrolling (in one or both axes), pinch-zooming or double-tap-zooming.

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Edge</td>
<td>Firefox</td>
</tr>
<tr>
<td></td>
<td>Chrome</td>
<td>Safari</td>
</tr>
<tr>
<td></td>
<td>Opera</td>
<td>iOS Safari</td>
</tr>
<tr>
<td></td>
<td>Android Browser</td>
<td>Chrome for Android</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>49</td>
<td>8.4</td>
</tr>
<tr>
<td>14</td>
<td>45</td>
<td>47</td>
</tr>
<tr>
<td>46</td>
<td>9.1</td>
<td>36</td>
</tr>
<tr>
<td>47</td>
<td>37</td>
<td>9.3</td>
</tr>
<tr>
<td>48</td>
<td>49</td>
<td></td>
</tr>
</tbody>
</table>

---

1. Supported in Firefox behind the `layout.css.touch_action.enabled` flag. Firefox for Windows 8 Touch ('Metro') enabled by default.
What if you have 3 containers (thumbnails gallery), each 33% width, but they also have padding defined in \textit{em} values, and border defined in \textit{px} values? The layout might \textit{break} because width doesn’t consider those values.
| .col — Width: 33%  
| Padding: 1em  
| Border: 1px |
| .col — Width: 33%  
| Padding: 1em  
| Border: 1px |
| .col — Width: 33%  
| Padding: 1em  
| Border: 1px |
We can apply the good ol’ IE Box Model to ensure that whenever we define width (or height), it always includes the padding and the border as well. We do that by applying `box-sizing: border-box` to pretty much everything.
CSS3 Box-sizing

Method of specifying whether or not an element's borders and padding should be included in size units

<table>
<thead>
<tr>
<th>Platform</th>
<th>Usage</th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>12%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unprefixed</td>
<td>10.86%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netherlands</td>
<td>10.99%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unprefixed</td>
<td>10.42%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Browser</th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: Known issues (4) Resources (6) Feedback
One of my least favorite parts about layout with CSS is the relationship of width and padding. You’re busy defining widths to match your grid or general column proportions, then down the line you start to add in text, which necessitates defining padding for those boxes. And ‘lo and behold, you now are subtracting pixels from your original width so the box doesn’t expand.

Ugh. If I say the width is 200px, gosh darn it, it’s gonna be a 200px wide box even if I have 20px of padding. So as you know, this is NOT how the box model has worked for the past ten years. Wikipedia has a great history of this box model. Jeff Kaufman also dove into the history

Anyway, I have a recommendation for your CSS going forward:
Inheriting box-sizing Probably Slightly Better Best-Practice

Published July 15, 2014 by Chris Coyier

I'm a big fan of resetting box-sizing to border-box, so much that we have a special day of the year around here. But there is a little adjustment to setting it that seems like a pretty good idea.

Here's the adjusted version:

```css
html {
    box-sizing: border-box;
}
*,:before,:after {
    box-sizing: inherit;
}
```

Credit on the inheritance idea to Jon Neal here, who says:

“This will give you the same result, and make it easier to change the box-sizing in plugins or other components that leverage other behavior.”
FITTER, HAPPIER, MORE PRODUCTIVE

COMFORTABLE

NOT DRINKING TOO MUCH

REGULAR EXERCISE AT THE GYM

(THREE DAYS A WEEK)

GETTING ON BETTER WITH YOUR ASSOCIATE EMPLOYEE CONTEMPORARIES

AT EASE
What if you want to *create some sort of visual distortion*? E.g. for headlines or hero banners or prominent boxes. Can we achieve it with CSS/HTML alone?
GLITCH
There are many *little details* that often make an experience just annoying. E.g. an age prompt once you try to access an age-restricted website. *Can we do better?*
EXPLORE THE WORLD'S WIDEST COLLECTION OF SINGLE CASK WHISKIES
By entering this site, I confirm I am over the legal drinking age in my country of residence.

DATE OF BIRTH
32  43  1547

SELECT COUNTRY
UNITED KINGDOM

Use a cookie to remember me. Only check this box if you are not using a shared computer.

ENTER SITE ➤

You must have cookies enabled to use this website
For further information on deleting or controlling cookies, please visit www.aboutcookies.org.
By entering this site you agree to our Terms and Conditions and Privacy and Cookies notice.
Welcome to Carlsberg Global
Enter the year of your birth

1 9

By submitting this form, you agree to the Cookie and Privacy policy of the Carlsberg Group. To learn more, please read our cookie and privacy policy...
Redesigning the Country Selector

Turning standard drop-downs into advanced auto-complete fields

During our recent checkout study we found several usability issues when using a drop-down for your country selector: a lack of overview, unclear sorting, scrolling issues, inconsistent UIs, a lack of context on mobile devices, and finally, they break the user’s tab-flow.

So we took it upon ourselves to redesign the country selector.

Using principles of progressive enhancement we turn a standard drop-down into an advanced auto-complete field. This means the drop-down remains accessible, while providing a much better experience in modern browsers – handling typos, multiple spelling sequences, synonyms and prioritization.

You can read the more about the design process and usability aspects in our article on Smashing Magazine: Redesigning the Country Selector.

To be notified about future improvements, follow the project on GitHub.
"I'm just copying it over because I think it's tiresome to write it all in again – it's quite tricky to write email addresses on a phone", a subject explains while copying his email address in the email confirmation fields at Foot Locker.
Email verification is unnecessary

60% of users consistently copy/paste their email when asked to verify it in the checkout.
• **Email verification is unnecessary**
Remove the email verification field altogether, or use *inline autocomplete* or “review” page instead.
Adaptive Map

A map experience that defaults to a text link to Google Maps, loads in a static map image for small screens and an iframe map for larger screens.

Read more about Adaptive Maps

More Responsive Patterns

Edit this Pen  Report Abuse
Live election results: track the votes in Michigan, Mississippi, Idaho and Hawaii

Data via Associated Press. Last checked 7:51

Explore more coverage of the 2016 presidential election

<table>
<thead>
<tr>
<th>HI</th>
<th>ID</th>
<th>MI</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>Sanders</td>
<td>Clinton</td>
</tr>
<tr>
<td>Trump</td>
<td>Cruz</td>
<td>Trump</td>
<td>Trump</td>
</tr>
</tbody>
</table>

Hawaii

Statewide results
45/45 precincts (100%) reporting

- Donald Trump: 5,677 (42.4%)
- Ted Cruz: 4,379 (32.7%)
- Marco Rubio: 1,759 (13.1%)
March 12 Primary Results

Keep up with the First Draft newsletter.

<table>
<thead>
<tr>
<th>Republicans</th>
<th>D.C. 19 del.</th>
<th>Wyo. 29</th>
<th>Delegates March 12</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trump</td>
<td>14%</td>
<td>7%</td>
<td>1</td>
<td>460</td>
</tr>
<tr>
<td>Cruz</td>
<td>12%</td>
<td>66%</td>
<td>9</td>
<td>369</td>
</tr>
<tr>
<td>Rubio</td>
<td><strong>37%</strong></td>
<td>20%</td>
<td>11</td>
<td>163</td>
</tr>
<tr>
<td>Kasich</td>
<td>36%</td>
<td>0%</td>
<td>9</td>
<td>63</td>
</tr>
<tr>
<td>Uncommitted</td>
<td>—</td>
<td>7%</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Reporting:
- 100% Lead
- 100% Win

Lead Win Needed: 1,237 to win
2016 election center

hillary clinton

CNN delegate estimate

total delegates: 4765
Look Ma! I broke that HTML!

SHOW MODAL
Responsive Icons
YOU'VE ARRIVED
THE MOMENT
YOU BOARD

Fly with us

Whether it's your first flight or simply your latest, we work to anticipate your every need.

The Emirates Experience

Fly to Vangon, Myanmar and Hanoi, Vietnam

From 3 August 2016, we will begin a daily circular service from Dubai to Vangon and Hanoi.
114 000 м²
ТРЦ на Хорошевском шоссе, Москва
Открытие в 2016 году
12,7 млн посетителей в год, 250 магазинов

137 000 м²
ТРЦ «Океания», Москва
Открытие в 2016 году
18,5 млн посетителей в год
300 магазинов

145 000 м²
ТРЦ «Галерея Краснодар», Краснодар
WELCOME TO THE MUSEUM!

Discover some of the highlights and exciting exhibits on show at the new home of football history.

THE F2 FREESTYLERS TAKE ON THE MUSEUM

What happens when you invite two of the world’s best freestylers to the FIFA World Football Museum?

UNCOVER FOOTBALL
Never worry about the quality of typefaces, the sources, licenses. The font you want, anywhere, anytime.

BROWSE ALL FONTS
Outbound flight
Zurich → Copenhagen

Wed 13/05/2015
from CHF 379

7:10 ZRH → 08:55 CPH
Travel time: 1h 45m
Operated by SWISS
LX 1288

<table>
<thead>
<tr>
<th>Flight Information</th>
<th>Economy Saver</th>
<th>Economy</th>
<th>Economy Flex</th>
<th>Business Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:10 ZRH → 08:55 CPH</td>
<td>--</td>
<td>--</td>
<td>CHF 494</td>
<td>--</td>
</tr>
<tr>
<td>Operated by SWISS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LX 1288</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:20 ZRH → 12:05 CPH</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Operated by SAS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCANDINAVIAN AIRLINES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LX 4700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:20 ZRH → 14:05 CPH</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Operated by SWISS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LX 1270</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Grand total per adult
Mileage calculator
Distagon T* FE 35mm F1.4 ZA
SEL35F14Z

★★★★☆ (5)
RRP £1,509.00 (incl. VAT)

Sonnar T* E 24mm F1.8 ZA
SEL24F18Z

★★★★☆ (2)
RRP £839.00 (incl. VAT)

Sonnar T* FE 35mm F2.8 ZA
SEL35F28Z

★★★★☆ (14)
RRP £699.00 (incl. VAT)

Distagon T* 24mm F2 ZA SSM
SAL24F20Z

★★★★☆ (1)
RRP £1,889.00 (incl. VAT)
α Lenses

See upcoming lenses

67 Products
Filter

SORT BY:
Featured

G MASTER
Tomorrow's lenses
BMW'NİZİ SEÇİN

1. BMW 1 Serisi 5-Kapı

2. BMW 2 Serisi Coupé, BMW Serisi Active Tourer, BMW Serisi Gran Tourer

3. BMW 3 Serisi Sedan, BMW 3 Serisi Gran Turismo

4. BMW X1 Serisi, BMW X2 Serisi, BMW X3 Serisi, BMW X4 Serisi, BMW X5 Serisi, BMW X6 Serisi

Filtrele 

01 Modeller
02 Motor Seçenekleri
03 Dış Görünüm
04 İç Görünüm
05 Paketler
06 Donanım Seçenekleri
07 Özet

Konfigürатор Asistanı

© BMW Türkiye 2016
Last week, you may have noticed that we released a facelift for our interactive maps. Our Deal Score markers have finally been brought up to 2014 design standards to match the Omnibox. However, what may not be as apparent is that our maps are now between 10 and 100 times faster, depending on the device.

**Background**

This blog post from March gives a good overview of how our maps used to work. Our maps consisted of three different layers: an image tile layer, an SVG layer, and a Leaflet marker layer.

This is how our map used to look. The actual stadium is an image tile, the blue section outline is an SVG layer, and the green dot is a Leaflet marker, an HTML element containing an image. There are a couple drawbacks to this approach...
Billy Joel

Great Deals

- **74** Section floor b
  - 7 tickets
  - $916

- **70** Madison club suit...
  - 2 e-tickets
  - $272

Good Deals
High Performance Map Interactions Using HTML5 Canvas

AUG 18TH, 2014

ChairNerd
Code, Design & Growth at SeatGeek
SeatGeek
StubHub app uses VR to show how good (or bad) your seats are

Maybe you should upgrade to Club Level after all.

Nicole Lee, @nicole
03.31.16 in Personal Computing
Auld Lang Syne

Guitar:

`A6/9`:
- 5 7 7 6 7 7
- X 9 7 9 10 X

`F#m7`:
- 7 X 7 9 7 7
- 7 7 7 7 7

`Bm7`:
- 7 7 7 7 7
- 7 7 7 7 7

`E9`:
- 5 7 7 6 7 7
- 7 7 7 7 7

`A6/9`:
- 5 7 7 6 7 7
- 7 7 7 7 7

`A7`:
- 5 4 5 6 X X
- 6 6 6 6

`D6/9`:
- 5 5 5 5 5
- X 5 4 5 5

`Ebdim`:
- 5 6 7 6
- 7 6 5 7 X

`A6/9` (bottom):
- 5 7 7 6 7 7
- 7 7 7 7 7

Music notation:

- `C`:
- `D`:
- `E`:
- `F`:
- `G`:
- `A`:
- `B`:
An example from Struik's *Lectures on Classical Differential Geometry*

This procedure is simply a generalization of the method used in Sects. 1-3 and 1-4 to obtain the equations of the osculating plane and the osculating circle. Let $f(u)$ near $P(u = u_0)$ have finite derivatives $f^{(i)}(u_0)$, $i = 1, 2, \ldots, n + 1$. Then if we take $u = u_1$ at $A$ and write $h = u_1 - u_0$, then there exists a Taylor development of $f(u)$ of the form (compare Eq. (1-5)):

$$f(u_1) = f(u_0) + hf'(u_0) + \frac{h^2}{2!}f''(u_0) + \cdots + \frac{h^{n+1}}{(n + 1)!}f^{(n+1)}(u_0) + o(h^{n+1}).$$

Here, $f(u_0) = 0$ since $P$ lies on $\Sigma_2$, and $h$ is of order $AP$ (see theorem Sec. 1-2); $f(u_1)$ is of order $AD$. Hence necessary and sufficient conditions that the surface has a contact of order $n$ at $P$ with the curve are that at $P$ the relations hold:

$$f(u) = f'(u) = f''(u) = \cdots = f^{(n)}(u) = 0;\quad f^{(n+1)}(u) \neq 0.$$
Concert to celebrate 70th Victory anniversary

Vladimir Putin attended a concert The Roads of Great Victory on Red Square that concluded the gala events to celebrate the 70th anniversary of Victory in the Great Patriotic War of 1941–1945.

May 9, 2015, 22:10
Red Square, Moscow
"We pay tribute to all those who fought to the bitter for every street, every house and every frontier of our Motherland."

Transcripts, May 9, 2015

Parade to mark the 70th anniversary of the Great Victory

3 days ago

Concert to celebrate 70th Victory anniversary

Vladimir Putin attended a concert The Roads of Great Victory on Red Square that concluded the gala events to celebrate the 70th anniversary of Victory in the Great Patriotic War of 1941–1945.

May 9, 2015 22:10 Red Square, Moscow
Simpler tariffs, easier choices

All energy tariffs have changed to a simpler structure and each customer will only have to choose from four per supplier. This will make it easier to compare tariffs and chose the one that is best for you.

Interactive energy bill

Our interactive energy bill helps to explain each part of an energy bill, including the new tariff information you will be seeing on bills in the coming months. Follow the orange light bulbs to see descriptions of each area. You can also download a copy of the bill, which includes explanations of each area (PDF 2MB).
THOU SHALT NOT DEFINETH
THAT LINE-HEIGHT IN PIXELS!
THOU SHALT NOT BASETH THY BREAKPOINTS ON DEVICE SIZES!
THOU SHALT NOT TAKE THE NAME OF THE LORD, PERFORMANCE, IN VAIN!
THOU SHALT
BEGINNETH WITH
MOBILE
Thou shalt not covet the menu of hamburgers!
THOU SHALT NOT PUT IMPORTANT SHIT BELOW THE FOLD!
THOU SHALT NOT WORSHIP THE CAROUSEL OF FALSE IDOLS!
California state Sen. LIZ FIGUEROA was upset that people who weren't using Gmail—but were corresponding with people who were—would also have THEIR emails scanned, effectively being forced into the keyword-scanning system.

Would you say that was an invasion of privacy?

Yes, that is intrusive!

Would [it] be able to read my emotions? Would [it] know that I was sad? Happy? Withdrawn?

Yes, and it can do a lot more than that!

Google co-founders Sergey Brin and Larry Page scoffed at Figueroa's fears that the company was retaining data over time.

I knew then that his definition of privacy and my definition of privacy were VERY different.

Figuer as was in the "penumbra of fear" because she was afraid of something the Google execs said they WEREN'T doing—collecting a comprehensive digital dossier.

We were trying to alert people where technology was heading and let them participate in the conversation.

And at least give their thumbs up or thumbs down, not just [have it] forced on us and say, "This is the way it's going to be."

"...and started reading your private materials—like your IRS statements, your diary or your private correspondence..."

"...but then imploded—and all that went away."

"And AL GORE played the negotiator—saying that the bill would go away."

Gore was a senior adviser to Google. He was a fellow Dem, former VP... Can you imagine going in? Here I am, and these two rock stars—plus Al?!

*Al Gore did not respond to repeated requests for comment. Google asked to comment for this story.
WORK & CO creates digital products and services that define great brands.

Forrester Research — “Work & Co set out to do something different with their new company by having senior focused teams — each project team includes at least one company partner who is actively engaged doing the work.”

Download the full report

01 Expertise

All Of Our Secrets

We only make digital products & services. Product Strategy, User Research, Design, Technology, Analytics & Optimization: everything you need to make a successful digital product or service, and nothing more.

Prototypes, not presentations.

In our experience, the biggest ideas usually come from hard work during iterative design and development. Forrester Research called our way of working “a model to follow.”

One team: Client + Work & Co.

We ask our clients to participate directly with us in an intensely collaborative process based on specific measures of success, rapid prototyping, and continuous testing and improvement. The result is
Bloomberg Businessweek Design 2016

April 11, 2016
Yerba Buena Center
San Francisco

#BWDESIGN2016

About
Speakers
Schedule
Location

Featuring:
and more!

BUY TICKETS

CONTACT US / PRESS RELEASE
Ok, I'm gonna pay $600 of it

Virgin America

Your booking is confirmed! Here are the details.

Booking Number: VAFQT2

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Departure</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr 15, Sat</td>
<td>9:10am</td>
<td>Seattle</td>
<td>New York</td>
</tr>
<tr>
<td>SEA</td>
<td>1 stop</td>
<td>JFK</td>
<td></td>
</tr>
<tr>
<td>Apr 27, Sat</td>
<td>08:10am</td>
<td>New York</td>
<td>Seattle</td>
</tr>
<tr>
<td>JFK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Passengers:
- John Davidson
- Kathy Clark

Total: $874.00

View Details
Total $54.20

PLACE ORDER

Get real-time shipping updates on your phone
Send to Messenger

SHIPPING OPTIONS

- Standard Shipping (3-5 days)
- Express Shipping (+ $15) (2 days)
A Robotic Home That Knows When You’re Hungover

A company is developing apartment buildings with sensors, automated appliances, and the ability to learn an owner’s habits.

by Will Knight

01 The Effects of Fukushima Linger after Five Years, but Not from Radiation
While hundreds died in the evacuation, none perished as a result of exposure to radiation.
by Richard Martin

02 The Danger of the Universal Basic Income
Giving everyone a “basic income” is the latest trendy idea sweeping Silicon Valley. It’s a terrible solution to a real problem.
Summary

— apply pseudo-elements on broken images
— contain: strict reduces painting costs
— fluid type with font-size: calc (1em + 1vw)
— nested links with `<object>`
— highlight row/column with pseudo-elements
— control table-layout with table-layout: fixed
— padded lines with the box-shadow trick
— consider critical progressive CSS (HTTP/2)
— use object-fit: cover to letterbox images
— use native variables with CSS currentColor
Summary

- service workers can boost performance a lot
- test scan levels to improve image delivery
- early fetch of critical resources with preload
- add interface screens to pattern libraries
- use the Font Loading API to load web fonts
- replace age prompt with a year prompt
- replace country selector with autosuggest
- replace email verification with email review
- load maps/lightboxes conditionally
- experiment and produce something creative.
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/))

- **Sections illustrations**: “bisous les copains”,
  by Guillaume Kurkdjian ([http://bisouslesescopains.tumblr.com/](http://bisouslesescopains.tumblr.com/))

- **Techniques**: by (tremendous) web design community.