LEVEL 1
LOCALIZATION
It’s your lucky day. You grow, and your company expands to foreign markets. Your site has to support 7 languages. How do you architect CSS/JS to support it?
The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle.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 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;
}
<table>
<thead>
<tr>
<th>Language</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burmese</td>
<td>5 လိုဏ်ဂါလ 2014</td>
</tr>
<tr>
<td>Sinhala</td>
<td>5 මා අමාර්තමල් 2014</td>
</tr>
<tr>
<td>Chinese</td>
<td>2014年 10月 5日</td>
</tr>
<tr>
<td>Persian</td>
<td>5 أكتوبر 2014 - 13 مهر 1393</td>
</tr>
<tr>
<td>Pashto</td>
<td>5 ِاکتوبر 2014 - 13 تہ 1393</td>
</tr>
<tr>
<td>Arabic</td>
<td>5 أكتوبر/ تشرين الأول 2014</td>
</tr>
</tbody>
</table>
Uhuru asaba Afrika kwikura ku mfashanyo

Umukuru w'igihu wa Kenya, Uhuru Kenyatta, yahamagariye bagenzi b'iwowe b'abanye Afrika guhagarika kwakira imfashanyo yi'igihu bikize.

Haolye amasaha 4 Amakuru

Ubukerarugendo burabangamiwe muri Kenya
Igitiri c'abakerarugendo bagendera igihugu ca Kenya cagabunyiseko 25%.
Haolye amasaha 8 Amakuru

Avuga ko ari umwirabura tari we
Abavuyeyi b'umugora aharanira agateka k'abirabura muri Amerika, bavuga ko amaze imyiaka abesha ko ari umwirabura.

Mu Burundi imyiyereko "itumwa n'abamenyeshamakuru":leta
Ubutegetsi mu Burundi twemeye ko ata myiyereko ikirangwa i Bujumbura.
11 Ukwa gatandatu 2015 Amakuru
Украина: Порошенко отправил в отставку главу минобороны

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

Британец Хэймилтон выиграл первый Гран-при России в Сочи

Британский гонщик Льюис Хэймилтон 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 Чжэйсян заявил, что у участников уличных акций протеста нулевой шанс изменить позицию Пекина по поводу формата выборов 2017 года.
Лента новостей

39 минут назад
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.

1 час назад
Евро-2016: Россия не смогла обыграть Молдавию

1 час назад
На севере Ливии возобновились бои
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
You have to build in **fluid, flexible type**, and designers want you to implement perfect *modular scale*. The proportions have to stay consistent across screens.
<table>
<thead>
<tr>
<th>Bases</th>
<th>Text</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>16px</td>
<td>121.5px</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>7.594em</td>
<td>7.594em</td>
</tr>
<tr>
<td></td>
<td>7.594em @ 16</td>
<td></td>
</tr>
<tr>
<td>81px</td>
<td>5.063em</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>5.063em</td>
<td>5.063em</td>
</tr>
<tr>
<td></td>
<td>5.063em @ 16</td>
<td></td>
</tr>
<tr>
<td>54px</td>
<td>3.375em</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>3.375em</td>
<td>3.375em</td>
</tr>
<tr>
<td></td>
<td>3.375em @ 16</td>
<td></td>
</tr>
<tr>
<td>36px</td>
<td>2.25em</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>2.25em</td>
<td>2.25em</td>
</tr>
<tr>
<td></td>
<td>2.25em @ 16</td>
<td></td>
</tr>
<tr>
<td>24px</td>
<td>1.5em</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>1.5em</td>
<td>1.5em</td>
</tr>
<tr>
<td></td>
<td>1.5em @ 16</td>
<td></td>
</tr>
<tr>
<td>16px</td>
<td>1em</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>1em</td>
<td>1em</td>
</tr>
<tr>
<td></td>
<td>1em @ 16</td>
<td></td>
</tr>
</tbody>
</table>

The quick brown fox jumps over the lazy dog
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
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.

— @simurai
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.

```css
.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** which already takes care of
Default (100%/1em)

```
.Component {
    font-size: 75%;
}
```

```
.Component {
    font-size: x-large;
}
```

Example used: Digit components
**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>
</table>

<table>
<thead>
<tr>
<th></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>8</td>
<td></td>
<td>35</td>
<td>5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>36</td>
<td>6.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>37</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>38</td>
<td>7.1</td>
<td></td>
<td>24</td>
<td></td>
<td>7</td>
<td>4.3</td>
<td>38</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>39</td>
<td>8</td>
<td></td>
<td>25</td>
<td></td>
<td>4.4</td>
<td>4.4.3</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>40</td>
<td></td>
<td></td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes | Known issues (4) | Resources (3) | Feedback
---|------------------|---------------|----------
No notes
With media queries, we can target specific screen width ranges and adjust type by just manipulating the font-size `rem` value of the article’s `container`. 
CSS Architecture

• *Main CSS* contains default type styles:

```css
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54 */
h1 { font-size: 3.375rem }
h2 { font-size: 2.25rem }
h3 { font-size: 1.5rem }
h4 { font-size: 1rem }
caption { font-size: 0.667rem }
small { font-size: 0.444rem }
```
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54 */
h1 { font-size: 3.375rem }
h2 { font-size: 2.25rem }
h3 { font-size: 1.5rem }
h4 { font-size: 1rem }
caption { font-size: 0.667rem }
small { font-size: 0.444rem }

/* Ideal line length: 66 ch; => max-width: 33em */
article { max-width: 33em; }
:lang(de) article { max-width: 40em; }
p, ul, ol, dl, table { margin-bottom: 1.45rem; }
CSS Architecture

/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54 */
h1 { font-size: 54px; font-size: 3.375rem }
h2 { font-size: 36px; font-size: 2.25rem }
h3 { font-size: 16px; font-size: 1rem; }
h4 { font-size: 24px; font-size: 1.5rem }
caption { font-size: 7px; font-size: 0.667rem }
small { font-size: 11px; font-size: 0.444rem }

/* Ideal line length: 66 ch; => max-width: 33em */
article { max-width: 33em; }
:lang(de) article { max-width: 40em; }
p, ul, ol, dl, table { margin-bottom: 1.45rem; }
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
\[ \text{html} \{ \text{font-size: calc}(1em + 1vw); \}\]
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.
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 \textit{min} and \textit{max} font-size and the \textit{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
**calc() as CSS unit value**

Method of allowing calculated values for length units, i.e. width:

```
calc(100% - 3em)
```

<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></td>
<td>Chrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Safari</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Opera</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iOS Safari</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Opera Mini</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Android Browser</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chrome for Android</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>45</td>
<td>49</td>
<td>48</td>
<td>8.4</td>
<td>9.2</td>
<td>4.3</td>
<td>4.4</td>
<td>4.4.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</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>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>48</td>
<td>52</td>
<td>10</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>53</td>
<td>TP</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes**

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.
Truly Fluid Typography With vh And vw Units

By Michael Riethmuller

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

28 Comments

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.
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.
min width
21 em

line-height lock
1.3 - 1.5

max width
35 em

min line-height
1.3

max line-height
1.5
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.`
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.
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 rest of life. They had long ago resigned themselves to their exile, and ceased to...
The math of CSS locks

September 2016

A CSS lock is a Responsive Web Design technique that lets you transition smoothly between two values, depending on the current viewport size, rather than jump straight from one value to the other.

This concept, and one implementation, were proposed by Tim Brown in Flexible typography with CSS locks.

When I tried wrapping my head around Tim's implementation, and creating variants of it, I had a hard time figuring out what was going on exactly. I did a lot of back-of-the-envelope calculations, and I thought it would be useful to share a mathematical explanation.

I'll describe the technique, its limitations, and the math that make it work. But don't worry about the math: it's basically addition and multiplication, and I've broken down the steps as much as possible—also there will be nice graphs.

Table of contents

1 What's a CSS lock?
   Viewport-relative sizes
Flexible typography with CSS locks

In early 2012, I shared a formula for “molten leading”, or fluid line spacing. Experienced typographers know that long lines of text need more line spacing, but line spacing can be tighter for short lines of text. The problem is that on the web, our texts are flexible. So our line spacing needs to flex, too:

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 @whi, there has been a jQuery plugin called Molten Leading around for quite a while. It tackles the problem with a simple Javascript solution that I’ve used in the past for my personal site. I’ve updated it to fix a bug and make it cross-browser compatible:

http://whi.me/molten-leading
We've barely scratched the surface of typography on the web and the reason why will shock you to your core. (Just kidding about that last bit). The web is so different to print, yet we are stuck in the ways of designing around our knowledge of print media. We need better tools to experiment, design, and build typography that is native to the web medium. Now's the time for us to start to push forwards with the future of typography on the web, and to forge its own language and best practices.
You’ve designed sophisticated **hover-effects** *(e.g. on images)* but what if hover isn’t available? Is there a way to *reliably* check for interaction feature support and adjust the experience accordingly?
In the past, to handle *hover-based interactions*, we relied on JavaScript workarounds to detect hover or tap. With *CSS Level 4 Media Queries*, we can detect if *hover* is available.
(Mouse isn’t the only device input that can “hover” (game consoles can “hover”, too). Also, a device might have multiple inputs, but the media query looks up the primary input.)
— Create 2 illustrations: *low-fi SVG, high-fi SVG.*
— Wrap a link around embedded *low-fi SVG* first,
— *Set the high-fi SVG as background image* on that link,
— *By default* → set **opacity: 0** on *low-fi SVG*,
— *Hover supported* → set **opacity: 1** instead, animate to **0** on hover.
• CSS:
  ```css
  .source__image {
    opacity: .75;
    transition: opacity 400ms;
    background: url("high-fidelity.svg");
  }

  .source__image:hover {
    opacity: 1;
  }

  .source__image svg {
    opacity: 0;
  }
  ```

• CSS:
  ```css
  @media (hover) {
    .source__image svg {
      opacity: 1;
      transition: opacity 400ms;
    }
    .source__image:hover svg {
      opacity: 0;
    }
  }
  ```
CSS:

```css
.page-wrap::after {
    display: block;
    content: ''; 
    position: fixed;
    bottom: 0;
    left: 0;
    width: 100%;
    height: 10em;
    background: linear-gradient(rgba(0,0,0,0), rgba(0,0,0,1));
}
```
Beware: *any* content that falls underneath the shadow we just created will not be *selectable* or otherwise available for interaction.
We don’t want *any* user interaction, and we don’t want to *block interactivity* of elements falling beneath the fade pseudo element. Meet *pointer-events*. 
CSS:

```
.page-wrap::after {
  display: block;
  content: ''; 
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 10em;
  background: linear-gradient(rgba(0,0,0,0), rgba(0,0,0,1));
}
```
CSS:

@supports (pointer-events: none)
    .page-wrap::after {
        display: block;
        content: '';  
        position: fixed;
        bottom: 0;
        left: 0;
        width: 100%;
        height: 10em;
        background: linear-gradient(rgba(0,0,0,0), rgba(0,0,0,1));
        pointer-events: none;
    }
Media Queries: interaction media features

Allows a media query to be set based on the presence and accuracy of the user's pointing device, and whether they have the ability to hover over elements on the page. This includes the pointer, any-pointer, hover, and any-hover media features.

<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>Current</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>51</td>
<td>56</td>
<td>49</td>
<td>10</td>
<td>43</td>
<td>9.3</td>
<td>10.2</td>
<td>4.4</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>52</td>
<td>53</td>
<td>55</td>
<td>44</td>
<td>4.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Known issues</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resources</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CSS Feature Queries allow authors to condition rules based on whether particular property declarations are supported in CSS using the @supports at rule.

<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></td>
<td>14</td>
<td>51</td>
<td>55</td>
<td>49</td>
<td>10</td>
<td>43</td>
<td>9.3</td>
<td>10.2</td>
<td>4.4</td>
<td>4.4, 4.4</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>15</td>
<td>53</td>
<td>56</td>
<td>10.1</td>
<td>44</td>
<td>10.3</td>
<td>all</td>
<td>56</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>58</td>
<td>TP</td>
<td>45</td>
<td>46</td>
<td>44</td>
<td>10.3</td>
<td>all</td>
<td>56</td>
<td>57</td>
</tr>
<tr>
<td>Notes</td>
<td>Known issues (2)</td>
<td>Resources (7)</td>
<td>Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CSS pointer-events (for HTML)

This CSS property, when set to "none" allows elements to not receive hover/click events, instead the event will occur on anything behind it.

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date 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>11</td>
<td>14</td>
<td>52</td>
<td>51</td>
</tr>
<tr>
<td>56</td>
<td>55</td>
<td>56</td>
<td>10</td>
</tr>
<tr>
<td>54</td>
<td>58</td>
<td>TP</td>
<td>45</td>
</tr>
</tbody>
</table>

Notes | Known issues (4) | Resources (5) | Feedback

[Source: caniuse.com/feat(pointer-events)]
New side project: highlights minisite, and checking for interaction feature support

Posted March 20, 2017

Tagged new work personal project web design web development css

I read quite a bit, and wanted somewhere to keep highlights from books and articles. That “somewhere” is highlights.melanie-richards.com, a simple Jekyll site hosted on Github Pages. I’m not doing anything fancy to get the highlights into the project (for various reasons I won’t bore you with), just using some YAML data files.

I did, however, come across a couple neat little use cases for interaction feature queries.

Handling a hover-dependent interaction
What if you’ve designed a rich **visual experience** for your website or app, but it makes the website hardly usable on **low battery**? How do you make sure that your websiter remains “responsive” then?
Working With The **JavaScript** BATTERY API
The *Battery Status API* talks to the device’s hardware and provides accurate data about the device’s *charging* state. Accessible via `navigator.getBattery()`.
• JavaScript:

```javascript
if (navigator.getBattery) {
    // Battery API available.
} else {
    // No Battery API support.
    // Handle error accordingly.
}
```

• JavaScript:

```javascript
navigator.getBattery().then(
    function(batteryManager) {
        // Get current charge in percentages.
        var level = batteryManager.level * 100;
    }
).catch(
    function(e) {
        console.error(e);
    });
```
getBattery() returns a promise and resolves with a batteryManager object containing information about the current status of the hardware.
- `batteryManager.level` returns the current charge, a float between 0 and 1.

- `batteryManager.charging` returns if the device is on power supply (true/false).

- `batteryManager.chargingTime` returns remaining time in sec till fully charged.

- `batteryManager.dischargingTime` returns remaining time until battery is dead.
The *Battery Status API* also provides events that can be used to monitor changes in above properties.

- `batteryManager.onlevelchange`
- `batteryManager.onchargingchange`
- `batteryManager.onchargingtimechange`
- `batteryManager.ondischargingtimechange`
getBattery() returns a promise and resolves with a `batteryManager` object containing information about the current status of the hardware.
JavaScript:

navigator.getBattery()
  .then(function(battery) {
    // Switch to Power Save for non-charging, low battery.
    battery.onlevelchange = function() {
      if (battery.level < 0.3 && !battery.charging) {
        powerSavingMode = true;
        // Remove parallax, web fonts, map embeds, video, JS
        // Save user’s data, inform them about low battery
      }
    }
  }
  .catch(function(e) {
    console.error(e);
  });
Working with the JavaScript Battery API

Demo
POWERSAVING IS OFF

Lorem ipsum dolor sit amet, consectetur, adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Advert

Control how much battery the virtual device has:

Drain Battery
# Battery Status API

Method to provide information about the battery status of the hosting device.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>14</td>
<td>51</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>Edge</td>
<td>15</td>
<td>52</td>
<td>57</td>
<td>15</td>
</tr>
<tr>
<td>Firefox</td>
<td>51</td>
<td>56</td>
<td>10.1</td>
<td>14</td>
</tr>
<tr>
<td>Chrome</td>
<td>55</td>
<td>56</td>
<td>10.1</td>
<td>15</td>
</tr>
<tr>
<td>Safari</td>
<td>43</td>
<td>9.3</td>
<td>44</td>
<td>54</td>
</tr>
<tr>
<td>Opera</td>
<td>44</td>
<td>10.3</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>44</td>
<td>10.3</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.4</td>
<td>4.4</td>
<td>4.4</td>
<td>4.4</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

Global: 63.54% + 9.94% = 73.47%
unprefixed: 63.54% + 9.88% = 73.42%

Notes: MS Edge status: Under Consideration
WebKit status: Removed
Often we want to target *specific children* in the DOM, but the parent might have unknown number of children. So usually we would style all children first and then *overwrite the styles*.
// 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
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
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;
}
```
On :not and Specificity

Mar 15, 2016  css

The negation pseudo-class, :not, can be incredibly useful. It allows us to target elements based on what attributes they don’t have, rather than what they do. This helps us avoid writing extra, increasingly specific, rules in an attempt to override previous ones.

A common example of this is when we want to apply a style to all list items, except the last one. For example -

```css
/* Without :not */
li { border-right: 1px solid #000; }
li:last-child { border-right: none; }

/* Using :not */
li:not(:last-child) { border-right: 1px solid #000; }
```
CSS3 selectors

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)
What if you want a **tidy grid** with fine and consistent **line endings**? Sometimes you might end up with **not enough space** to display all content blocks in a row, or **not enough items** to properly fill a row.

— Patrick Clancey
A **quantity selector** is a CSS selector that allows styles to be applied to elements based on the number of siblings.
Build a query

WHICH ELEMENT WILL BE COUNTED

ui

TYPE OF QUERY

At-Least

AMOUNT OF ITEMS

# of items to count

CREATE QUERY

Your Code

Copy and paste the code below into your styles

```
ul

// Build a query on the left
```

Try it out

Your quantity query will be reflected on the items below by a change in colour. Add and remove items to see the styling be applied when the query matches.

ADD ITEM  REMOVE ITEM
General sibling selector \(~\) separates two selectors and matches the second element only if it is preceded by the first, and both share a common parent.
`li:nth-last-child(6):first-child`
• **CSS:**

```css
li:nth-last-child(6):first-child,
li:nth-last-child(6):first-child ~ li {
  color: green;
}
```
• CSS:

```css
li:nth-child(n+6) {
  color: green;
}
```
li:nth-last-child(n+6) {
    color: green;
}

7 items

6 items

5 items
  color: green;
}
Selector support

All of the CSS2.1 and CSS3 selectors used in this article are supported in Internet Explorer 9 and above, including all reasonably recent mobile/handheld stock browsers.

Internet Explorer 8 support is good for most selector types, but technically partial, so you might want to consider a JavaScript polyfill. Alternately, you could pair the selectors for the “safer” of the layout strategies with IE9-specific classes. In the case of the navigation menu, the safer option is the one catering to more items, using `inline-block`. The declaration block would look something like this:

```
nav li:nth-last-child(n+6),
nav li:nth-last-child(n+6) ~ li,

.lt-ie9 nav li {
  display: inline-block;
  /* etc */
}
```
To create a **perfect grid**, we’ll need to define layout for *any* number of items with *specific quantity selectors* within media queries.
• “Mod query selector” in CSS:

```css
li:nth-last-child(3n):first-child,
li:nth-last-child(3n):first-child ~ li {
  /* … styles for list items in a list divisible by 3 … */
}
```
— Select all following siblings (~ li) which follow after
— The first child (first li in the list here), (:first-child) that also is
— Divisible by 3, starting from the end (:nth-last-child(3n)).

li:nth-last-child(3n):first-child,  
li:nth-last-child(3n):first-child ~ li {
  /* ... styles for list items in a list divisible by 3 ... */
}
“Range selector” in CSS:

```css
li:nth-child(n+3):nth-child(-n+5) {
  /* ... styles for list items from 3 to 5 ... */
}
```

— Select all the items up to and including the fifth item, then
— Select all the items from the third item onwards.
<table>
<thead>
<tr>
<th>Product title</th>
<th>Product detail text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product title</td>
<td>Product detail text</td>
</tr>
<tr>
<td>Product title</td>
<td>Product detail text</td>
</tr>
<tr>
<td>Product title</td>
<td>Product detail text</td>
</tr>
</tbody>
</table>
We use a **mod query** to check if the number of items is divisible by 3. Then we use a **range selector** to style items differently, e.g. apply one styling to *first three*, another styling to the fourth *through* ninth, and another to 10th *onwards*. Voilà!
“Mod query selector” in CSS:

```css
li:nth-last-child(3n):first-child /* mod query */
~ li:nth-child(n+4):nth-child(-n+6) { /* range selector */
    /* … styles for 4th to 6th elements, in a list divisible by 3 … */
}
```
Tidy list using Mod Queries and Range Selectors

This demo is based on my short article about CSS Mod Queries and Range Selectors. The CSS query matches a Mod and styles the remainder accordingly.

The queries will style the items near the top of the list so the last line is always tidy and you’ll never get orphans at the end of the list.

Use the form below to set the mod and add/remove list items. Check the generated CSS to see how the list works.

Select the mod base for your list

Mod 3

Add remove list items

Add Item

Remove Item

The queries generated are as follows:

First two items:
First item is always 100% wide, the second child is 50% if there are only two items

Remainder 0:
As first item is 100%, the second row items have a width of 100%/mod-1

Remainder 1:
No action needed, the default width for all items plus the first item

Remainder 2:
Second row has two 50% width items, third row has width of 100%/mod-1

Remainder 3:
Second row has three 33.33% width items, third row has width of 100%/mod-1

*/

/* mod 3 */
/* first two, common to all queries */

li:first-child {
    width: 100%;
}

li:nth-child(2):last-child {
    margin-left: 25%;
    width: 50%;
}

/* styles for less than 3 items */

li:nth-last-child(3):first-child ~ li { width: 50%;
}

li:nth-last-child(3n):first-child ~ li { width: 33.333%;
}

li:nth-last-child(3n+2):first-child ~ li { width: 50%;
}

li:nth-last-child(3n+1):first-child ~ li { width: 50%;
}

li:nth-last-child(3n):first-child ~ li { width: 50%;
}

li:nth-last-child(3n+2):first-child ~ li { width: 50%;
}

li:first-child message + /

li:nth-last-child(3n+3):first-child:before { content: 'X';

X;
Using CSS Mod Queries with Range Selectors

by Patrick Clancy · October 11, 2016

Recently, I was asked to build a simple list that would display in a grid—one that could start with a single element and grow throughout the day, yet always be tidy regardless of the length. So, as you do sometimes when you’re busy with one thing and asked if you can do something completely different, I tried to think of any reason why it couldn’t be done, came up blank, and distractedly said, “Yes.”

At the time, I was working on a London-based news organization’s website. We’d spent the previous year migrating their CMS to the Adobe AEM platform while simultaneously implementing a responsive UI—both big improvements. Since that phase was complete, we were starting to focus on finessing the UI and building new features. The development project was divided into a number of small semiautonomous teams. My team was focusing on hub pages, and I was leading the UI effort.

Each hub page is essentially a list of lists, simply there to help readers find content that interests them. As you can imagine, a news website is almost exclusively made of content lists! A page full of generic vertical lists would be unhelpful and unappealing; we wanted readers to enjoy browsing the content related to their sphere of interest...
Quantity Queries for CSS

by Heydon Pickering - March 03, 2015
Published in CSS, HTML

Don’t you just hate documentaries that don’t deliver? They have enticing names like In Search of the Giant Squid, and tease you with shots of murky underwater shapes and excited scientists pointing far out to sea. You settle down to watch, eyes narrowed with suspicion, thinking, “I better see some squid or I’m writing an angry letter to the network.”

Sure enough, 90 minutes of interviews with bored-looking fishermen later, the presenter is forced to conclude, “No... no, we didn’t find any big squids. But maybe one day [majestic orchestral flourish].” Great. You wanted Finding Nemo and got Not Finding Nemo instead.

I wouldn’t do that to you, friends. This is your guide to creating style breakpoints for quantities of HTML elements, much as you already do with @media queries for viewport dimensions. I’m not pointing at some blurry specification in the distance or a twinkle in an implementer’s eye. We’re going to do this today, with CSS that’s already...
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*?
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:

![SVG button](image)

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:

```
.button
```
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:

```
.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
.fill: $666;

/* 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>IE</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Android Browser</th>
<th>Chrome for Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>31</td>
<td>36</td>
<td>37</td>
<td></td>
<td></td>
<td>4.1</td>
<td>4.3</td>
</tr>
<tr>
<td>9</td>
<td>35</td>
<td>38</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td></td>
<td>7.1</td>
<td>7.1</td>
<td></td>
<td></td>
<td></td>
<td>4.4.4</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>41</td>
<td>8</td>
<td>8.1</td>
<td>8</td>
<td></td>
<td>37</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td>37</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

|   |   |   |   |   |   |   |   |   |   |
| Notes | Known issues (0) | Resources (3) | Feedback |

No notes
LEVEL 6
OUTBREAK
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?
• HTML:

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

• CSS:

```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, we just need to subtract *half the container width* from *half the viewport width*. `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:**
```
<div class="u-containProse">
  <p>...</p>
  <img class="u-release" src="..." />
  <p>...</p>
</div>
```

**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:

```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;
}
```
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?
...Given two identical images that are displayed at the same size on a website, one can be dramatically smaller than the other in file size if it’s highly compressed and dramatically larger in dimensions than it is displayed in.

— Daan Jobsis
600×400px file, 0% JPEG quality, displayed in 600×400 (file size 7 Kb)
600×400px file, 0% JPEG quality,
displayed in 300×200 (file size 7 Kb)
300×200px file (21 Kb)
80% JPEG quality
displayed in 300×200

600×400px file (7 Kb)
% JPEG quality
displayed in 300×200
<table>
<thead>
<tr>
<th>Device</th>
<th>PPI</th>
<th>Tested</th>
<th>Working</th>
<th>Browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iPad 3</td>
<td>264</td>
<td>Yes</td>
<td>Yes</td>
<td>Safari, Chrome</td>
</tr>
<tr>
<td>Apple iPhone 4 / 4S</td>
<td>325</td>
<td>Yes</td>
<td>Yes</td>
<td>Safari, Chrome</td>
</tr>
<tr>
<td>Apple MacBook Pro 15 &quot;Retina Display&quot;</td>
<td>220</td>
<td>Yes</td>
<td>Yes</td>
<td>Safari</td>
</tr>
<tr>
<td>Archos 10.1 G9</td>
<td>149</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTC ChaCha</td>
<td>222</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTC Desire S</td>
<td>252</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTC One V</td>
<td>252</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Nokia Lumia 800</td>
<td>252</td>
<td>Yes</td>
<td>Yes</td>
<td>Mobile Internet Explorer 9</td>
</tr>
<tr>
<td>Nokia Lumia 900</td>
<td>217</td>
<td>Yes</td>
<td>No</td>
<td>Mobile Internet Explorer 9</td>
</tr>
<tr>
<td>Samsung Galaxy Ace</td>
<td>164</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Nexus</td>
<td>316</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy S Advance</td>
<td>233</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy SIII</td>
<td>305</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Tab 2 10.1</td>
<td>149</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Xcover</td>
<td>158</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Samsung Note</td>
<td>285</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Retina revolution
Geplaatst op 27 juli 2012 door Dean Jobsis

The devil is in the details
Detail is probably one of the most important values for a designer, an eye for detail should be in our DNA. As a perfectionist I like my designs to be pixel perfect. I am allergic for “aggies” and ugly compressed artifacts in icons and images on websites. Apple’s Retina revolution is an interesting evolution that is turning the design world upside down. The Retina display has a high enough pixel density to prevent pixelation to be noticable to the human eye. Therefore a Retina display is a lot sharper and more pleasant to look at. Apple has doubled the amount of horizontal and vertical pixels on the iphone, The New iPad, and now also on the new MacBook Pro. The Retina revolution is irreversible, and other companies have already started or will also start implementing this new Retina technology.

Nowadays pixel perfection can be obtained with techniques like @font-face and CSS3. Making fonts, borders, shadows, and gradients sparkle on your screen. These elements are based on vectors or mathematical expressions which allows them to be scaled to enormous sizes without creating distortion. This does not count for rasterized images which consist of pixels. An image that looks good on a normal display will appear blurry on a Retina display. The Retina display blows up the image, it doubles the amount of pixels. There is not enough data for the image to be displayed...
Aftonbladet’s Images Strategy

• Design specification defined *main requirements*:
  
  • Optimization of the mobile version,
  • The pages should be easy to cache,
  • A single HTML file to be served to all users,
  • All images on a content delivery network (CDN),
  • No complexity in the image-serving logic,
  • Serving different image versions to different devices.

• *Solution*: Loading images with JavaScript after HTML and CSS have fully loaded.
• 30% JPEG quality: bright-red areas don’t compress well.

• Editors can select compression rates, but aggressive compression is a default.
• The homepage on a mobile device has 40 images.

• On average, the “large” screen has 650 Kb, “medium” — 570 Kb, “small” — 450 Kb.
• The original photo has 1600px width, \textbf{971 Kb.}
  Quality 60 brings the size down to \textbf{213 Kb.}
• Blurring unimportant parts of the photo brings the size down to 147 Kb.
Sequential JPEG  Progressive JPEG

Images taken from http://www.pixelstech.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: 0-63, 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)
2a:  1-63, 0, 0;
3a:  1-63, 0, 0;

# Send remaining Y AC coefficients, minus 2 lowest bits:
6a:  6-63, 0, 2;
15 # Send next-to-lowest bit of all Y AC coefficients:
8a:  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
2a:  1-63, 1, 0;
3a:  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

1: 1-63, 0, 0;

# Remaining Y coefficients

5: 28-63, 0, 0;  
Second half of brightness 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>3.18s</td>
<td>0.118s</td>
<td>0.436s</td>
<td>3.500s</td>
<td>1537</td>
<td>75</td>
<td>0</td>
<td>3.18s 21</td>
<td>3.213s 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 21</td>
<td>3.590s 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

Adept - the adaptive JPG Compressor
Improved JPEG encoder.

- 3,486 commits
- 8 branches
- 7 releases
- 24 contributors

Branch: master

<table>
<thead>
<tr>
<th>Author</th>
<th>Message</th>
<th>Files</th>
<th>Add</th>
<th>Delete</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>pomek</td>
<td>Merge pull request #207 from mozilla/jpeg-yuv-dashup</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmake</td>
<td>Win: Enable testing cross-compiled builds</td>
<td></td>
<td></td>
<td></td>
<td>5 months ago</td>
</tr>
<tr>
<td>doohml</td>
<td>Bump TurboJPEG C API revision to 1.5</td>
<td></td>
<td></td>
<td></td>
<td>4 months ago</td>
</tr>
<tr>
<td>java</td>
<td>Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo</td>
<td></td>
<td></td>
<td></td>
<td>2 months ago</td>
</tr>
<tr>
<td>mview</td>
<td>Merge libjpeg-turbo r1390</td>
<td></td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>release</td>
<td>Merge libjpeg-turbo r1390</td>
<td></td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>sharedlib</td>
<td>Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo</td>
<td></td>
<td></td>
<td></td>
<td>2 months ago</td>
</tr>
<tr>
<td>simd</td>
<td>Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo</td>
<td></td>
<td></td>
<td></td>
<td>2 months ago</td>
</tr>
<tr>
<td>testsimages</td>
<td>12-bit JPEG support</td>
<td></td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>win</td>
<td>Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo</td>
<td></td>
<td></td>
<td></td>
<td>2 months ago</td>
</tr>
<tr>
<td>.gitauthors</td>
<td>Script for git-svn reinitialization</td>
<td></td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
</tbody>
</table>
What if you have a large photo that requires a transparent shadow? PNG is way too large in file size, and JPEG isn’t good enough in quality because of the gradient in the background. What do you do?
• 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>
```
• 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"/>
</svg>
```

• HTML/CSS:

```html
<img src="hero-image.svg" />, background: url("hero-image.svg")
```
September 7, 2014

USING SVG TO SHRINK YOUR PNGS

Wouldn’t it be great if you could get the compression of a JPEG and keep the transparency of a PNG? Well, you can, sort of. Here’s a little trick that I discovered while working on the new Sapporo Beer website.

Notice how the beer can on the Sapporo website has a transparent area (it’s hard to notice but there’s a video playing behind it). As a PNG, the beer can doesn’t show the transparency of SVG (sort of).
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:
Text compression matters. What’s the best strategy to compress assets/content these days? Essentially, we want to minimize bandwidth + speed up delivery.
gzip is the most common compression format on the web; its most common implementation is zlib, and it uses a combination of LZ77 and Huffman encoding algorithms (called deflate).
Each compression library (like zlib) has *preset quality settings*, ranging from *fast* compression (levels 1–3) to *slow* compression (levels 4–9).
As developers, we care about the transferred file size and compression/decompression speed — for both static and dynamic web content.
Zopfli can be thought of as a way to do a “very good, but slow, deflate or zlib compression”. High compression ratio at the cost of a higher overhead for compressing. *Backwards-compatible* for browsers that support only gzip.

— Cody Ray Hoeft

https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli
"Brotli is a whole new compression and decompression format. For Brotli, browser support has to be built into the browser. *Future-compatible* with the next generation of browsers.

— Cody Ray Hoeft

https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli
Brotni Accept-Encoding/Content-Encoding

More effective lossless compression algorithm than gzip and deflate.

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date 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>11</td>
<td>14</td>
<td>51</td>
<td>56</td>
</tr>
<tr>
<td>15</td>
<td>52</td>
<td>57</td>
<td>10.1</td>
</tr>
<tr>
<td>53</td>
<td>58</td>
<td>TP</td>
<td>44</td>
</tr>
<tr>
<td>54</td>
<td>59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes

MS Edge status: Preview Release
1. Supported in Chrome and Opera behind the 'Brotni Content-Encoding' flag
2. Enabled since 27 May 2016
“Brotli is a whole new lossless compression and decompression format. For Brotli, browser support has to be built into the browser. *Future-compatible* with the next generation of browsers.

— Cody Ray Hoeft

https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli
Brotli and Zopfli

• Compared to gzip, Brotli is significantly slower at compressing data, but provides much better savings.
  — Brotli is an open-sourced, lossless compression format,
  — Brotli shows significant improvements for static content,
  — Brotli’s decompression is fast: comparable to zlib,
  — Brotli has an advantage for large files on slow connections,
  — Expect 14-39% file savings on text-based assets (level 4),
  — Ideal for HTML, CSS, JavaScript, SVG — anything text-based.
  — Brotli support is restricted to HTTPS connections.
Capable browsers advertise their ability to accept Brotli-compressed resources in the `Accept-Encoding` request header.

```
Accept-Encoding: gzip, deflate, sdch, br
```

If both browser and server support Brotli, the server uses Brotli compression and sets the `Content-Encoding` header of the response to `br` accordingly.

```
Content-Encoding: br
```
brotni_static on;

Enabling this will cause Brotni to look for files that have already been compressed and serve those files directly without performing any compression itself. What we also did, was to remove the brotni_types directive that instructs the server to compress all file types. This makes Brotni fall back to its default of only compressing the default mime type text/html. brotni_static then makes sure that Nginx returns the statically compressed version for any resource that it can match with a brotnified counterpart.

dist/assets/css/
    main-7bca136736.css
    main-7bca136736.css.gz
    main-7bca136736.css.br
The relevant bit of Nginx configuration now looks like this:

```
http {
    # ...truncated...
    gzip on;
    gzip_static on;
    gzip_vary on;

    brotli on;
    brotli_comp_level 4;
    brotli_static on;
    # ...truncated...
}
```
Brotli and Zopfli

- Compared to gzip, Brotli is significantly slower at compressing data, but provides much better savings.
  - Browsers advertise support via Accept-Encoding request header:
    Accept-Encoding: gzip, deflate, sdch, br
  - Servers can choose to use Brotli and serve Content-Encoding: br
  - You might need to recompile your server to include a Brotli module (available for Apache, Nginx, IIS).
  - Zopfli often not applicable for on-the-fly compression, but a good alternative for one-time compression of static content.
Brotli/Zopfli Compression Strategy

- Compared to gzip, Brotli is significantly slower at compressing data, but provides much better savings.
  - Pre-compress static assets with Brotli+Gzip at the highest level,
  - Compress (dynamic) HTML on the fly with Brotli at level 1–4.
  - Check for Brotli support on CDNs (KeyCDN, CDN77, Fastly).
  - Server handles content negotiation for Brotli or gzip.
  - Use Zopfli if you can’t install/maintain Brotli on the server.

“Results of experimenting with Brotli for dynamic web content”, https://blog.cloudflare.com/results-experimenting-brotli/
Perceived performance matters. The more invisible the loading of assets is, the faster the overall experience is. How can we speed up delivery *effortlessly*?
Resource hints allow developers to provide some hints to the browser to prompt the download of assets, or rendering, silently in the background.
Resource hints allow developers to provide some hints to the browser to prompt the download of assets, or rendering, silently in the background.

- `<link rel="prefetch" href="(url)">` tells browsers to fetch a resource that will probably be needed for the next navigation (low priority).
- `<link rel="prefetch" href="(url)"`  
  tells browsers to fetch a resource that will *probably* be needed for the next navigation *(low priority)*.

- `<link rel="prerender" href="(url)"`  
  tells browsers to render the specified page in the background *(low priority)*.
- `<link rel="prerender" href="(url)">`
tells browsers to render the specified page in the background (**low priority**).

- `<link rel="dns-prefetch" href="(url)">`
gives a hint to the browser to perform a DNS lookup in the background (**low priority**).
- `<link rel="dns-prefetch" href="(url)">` gives a hint to the browser to perform a DNS lookup in the background (*low priority*).

- `<link rel="preconnect" href="(url)">` gives a hint to the browser to initiate the connection handshake (DNS, TCP, TLS) in the background (*low priority*).
- `<link rel="preconnect" href="(url)">` gives a hint to the browser to initiate the connection handshake (DNS, TCP, TLS) in the background (low priority).

- `<link rel="preload" href="(url)" as="(type)">` gives a hint to the browser to prefetch resources and set the right resource priority for loading assets.
The basic use case for `preload` is loading of *late-discovered critical resources*. If we omit the `as` attribute, it’s just an XHR request, fetching with a fairly low priority.
<link rel="preload"
    href="late-discovered.js"
    as="script">

The `as` attribute tells the browser what it is downloading. E.g. audio, font, image, script, style, track, video, document.
E.g. you could include `<preload>` directives for web fonts that you know you’ll need for rendering of the page.

```html
<link rel="preload" href="font.woff2" type="font/woff2" crossorigin as="font">
```
var preload = document.createElement("link");
link.href="myscript.js"
link.rel="preload";
link.as="script";
document.head.appendChild(link);

E.g. you could *request* the fetching of a resource because you know you’ll need it, but you don’t want to execute it yet.
E.g. you could *request* the fetching of a resource because you know you’ll need it, but you don’t want to execute it yet.
E.g. you could load assets *conditionally* (e.g. a static map on smaller screens, and an interactive map on large screens).
Resource Hints: prefetch

Informs the browsers that a given resource should be prefetched so it can be loaded more quickly. This is indicated using `<link rel="prefetch" href="(url)">`
Resource Hints: prerender

Gives a hint to the browser to render the specified page in the background, speeding up page load if the user navigates to it. This is indicated using `<link rel="prerender" href="(url)"`.

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>14</td>
<td>51</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>Edge</td>
<td>15</td>
<td>53</td>
<td>58</td>
<td>15</td>
</tr>
<tr>
<td>Firefox</td>
<td>57</td>
<td>10.1</td>
<td>44</td>
<td>56</td>
</tr>
<tr>
<td>Chrome</td>
<td>10</td>
<td>44</td>
<td>44</td>
<td>56</td>
</tr>
<tr>
<td>Safari</td>
<td>43</td>
<td>4.4</td>
<td>4.4</td>
<td>57</td>
</tr>
<tr>
<td>Opera</td>
<td>43</td>
<td>4.4</td>
<td>4.4</td>
<td>57</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>9.3</td>
<td>56</td>
<td>56</td>
<td>57</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>56</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

Notes: No notes

Known issues: 1
Resources: 2
Feedback:
Resource Hints: dns-prefetch

Gives a hint to the browser to perform a DNS lookup in the background to improve performance. This is indicated using `<link rel="dns-prefetch" href="http://example-domain.com/">

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date 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>11</td>
<td>15</td>
<td>53</td>
<td>58</td>
</tr>
<tr>
<td>54</td>
<td>59</td>
<td>TP</td>
<td>45</td>
</tr>
<tr>
<td>55</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Resource Hints: preconnect

Gives a hint to the browser to begin the connection handshake (DNS, TCP, TLS) in the background to improve performance. This is indicated using `<link rel="preconnect" href="https://example-domain.com/">`

<table>
<thead>
<tr>
<th></th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox</td>
<td>51</td>
<td>56</td>
<td>9.3</td>
<td>4.4</td>
</tr>
<tr>
<td>Chrome</td>
<td>49</td>
<td>57</td>
<td>10</td>
<td>44</td>
</tr>
<tr>
<td>Safari</td>
<td>10.1</td>
<td>10.3</td>
<td>all</td>
<td>56</td>
</tr>
<tr>
<td>Opera</td>
<td>43</td>
<td>44</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>iOS Safari</td>
<td>9.3</td>
<td>10.2</td>
<td>all</td>
<td>57</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>4.4</td>
<td>10.3</td>
<td>all</td>
<td>56</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.4</td>
<td>4.4</td>
<td>all</td>
<td>57</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>4.4</td>
<td>57</td>
<td>all</td>
<td>57</td>
</tr>
</tbody>
</table>

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

MS Edge status: Under Consideration
Resource Hints: preload

Using `<link rel="preload">`, browsers can be informed to prefetch resources without having to execute them, allowing fine-grained control over when and how resources are loaded.

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date 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>11</td>
<td>51</td>
<td>53</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>55</td>
<td>58</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>51</td>
<td>56</td>
<td>43</td>
</tr>
<tr>
<td>51</td>
<td>57</td>
<td>10.1</td>
<td>10.2</td>
</tr>
<tr>
<td>52</td>
<td>56</td>
<td>44</td>
<td>all</td>
</tr>
<tr>
<td>53</td>
<td>58</td>
<td>44</td>
<td>56</td>
</tr>
<tr>
<td>54</td>
<td>59</td>
<td>TP</td>
<td>45</td>
</tr>
<tr>
<td>55</td>
<td>60</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>56</td>
<td>61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: Known issues (0) Resources (6) Feedback

MS Edge status: Under Consideration