# Front-end developers code standards

This document contains guidelines for web applications built by the Creative Technology (front end engineering) practice of DIGITAS France. It is to be readily available to anyone who wishes to check the iterative progress of our best practices.

This document's primary motivation is two-fold:

- Code consistency
- Best practices

All code in any code base should look like a single person typed it, no matter how many people contributed. This means strictly enforcing these agreed upon guidelines at all times.

By maintaining consistency in coding styles and conventions, we can ease the burden of legacy code maintenance, and mitigate risk of breakage in the future. By adhering to best practices, we ensure optimized page loading, performance and maintainable code.

# General Guidelines

## Pillars of Front-end Development

- Separation of presentation, content, and behavior.
- Markup should be well-formed, semantically correct (http://www.bbc.co.uk/guidelines/futuremedia/technical/semantic_markup.shtml) and generally valid.
- Javascript should progressively enhance the experience (http://icant.co.uk/articles/pragmatic-progressive-enhancement/).

## General Practices

### Indentation

For all code languages, we require indentation to be done via soft tabs (using the space character). Hitting *Tab* in your text editor shall be equivalent to **two spaces**.

### Readability vs Compression

We prefer readability over file-size savings when it comes to maintaining existing files. Plenty of whitespace is encouraged, along with ASCII art, where appropriate. There is no need for any developer to purposefully compress HTML or CSS, nor obfuscate JavaScript.

We will use our build process to automatically minify and gzip all static client-side files, such as CSS and javascript.

### Commenting

The requirement to comment code obsessively was pioneered by managers, team leaders and other people that interact with code infrequently. It is sought merely as a check box for an employee's KPI (Key Performance Indicators)s, and provides little return for the time spent doing so.

If a best-practice oriented developer follows the guidelines established in the following documents, their code should become so readable and obvious that the need to comment what it is doing is embarassingly redundant. Consider the following example. In this: booleans are posed as questions, and functions are named intuitively.

```
if (user.hasPermission) { editPage(); }
```

Commenting, in this scenario at least, is completely unnecessary.

When you comment your code, do it using Markdown. We use Docco (http://jashkenas.github.io/docco/) and StyleDocco (http://jacobrask.github.io/styledocco/) to automatically generate the docs.

#### SITUATIONS WHERE COMMENTING IS IMPORTANT

Some parts of a project will never be easy to scan and understand. Consider a complicated regular expression, or a math function calculating angles or switching between degrees and radians. Without the comment above, beginner and intermediate readers will be fairly clueless to the scripts meaning.

```
// RegEx for validating US phone numbers,
// can be (XXX) XXX-XXXX (with or without dashes, spaces or brackets)
var phoneRegEx = /^\(?(\d{3})\)?[- ]?(\d{3})[- ]?(\d{4})$/;
```

# Markup

## HTML5

HTML5 (http://www.ibm.com/developerworks/library/wa-webstandards/index.html) is a new version of HTML and XHTML. The HTML5 draft (http://www.w3.org/TR/html5/) specification defines a single language that can be written in HTML and XML. It attempts to solve issues found in previous iterations of HTML and addresses the needs of *web applications*, an area previously not adequately covered by HTML. (source (http://html5.org/)).

We will use the HTML5 Doctype and HTML5 features when appropriate.

## What's Up, DOCTYPE?

### The absence of a DOCTYPE is a crime punishable by death (https://github.digitas.fr/pages/html/code-standards/ikillyou.png).

You may have relied on the following DOCTYPE in the past

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

In HTML5, this has been superseded by a leaner and meaner snippet.

```
<!DOCTYPE html>
```

It's supported in all modern browsers, and throws IE6 and IE7 into standards mode. Source (http://ejohn.org/blog/html5-doctype/).

## Write Valid Semantic Markup

Writing websites with clean, semantic HTML is something we wish we could always do. Sometimes we find ourselves limited by the way pages were setup by our predecessors, or sometimes we're coding an HTML email. The validity of the HTML should never be compromised, even if to solve a browser specific bug.

Headings should be heirarchically created from `<h1>` onwards, paragraphs should always be in `<p>` tags and so on and so forth. If you write semantic HTML, the resultant page will be cleaner, lighter and easily parsed by search engine spiders. This is one of the simplest SEO fixes you can undertake.

**Which do you think looks cleaner, this?:**

```
<span class="sectionHeading">A Heading</span>
<br /> <br /> Lorem ipsum dolor sit amet. ... <br /> <br />
```

**Or this?**

```
<h2>A Heading</h2>
<p> Lorem ipsum dolor sit amet. ... </p>
```

We will test our markup against the W3C validator (http://validator.w3.org/), to ensure that the markup is well formed. 100% valid code is not a goal, but validation certainly helps to write more maintainable sites as well as debugging code.

## Character Encoding

All markup should be delivered as UTF-8, as its the most friendly for internationalization. It should be designated in the head of the document.

Setting the character set using `<meta>` tags.

In HTML5, just do:

```
<meta charset="utf-8" />
```

## Structure

Elements such as `header, footer, section, article`, and others, can be used multiple times on a page, and more than once within one or the other even. In cases where there will be more than one of the same element, that is used for the same thing such as using `article` for each blog post of a page and each article having a `header` and `footer`, adding a class to them can help separate them from any other places you may use these elements on the page. That way you can target the elements more specifically in your CSS. However this is not always necessary, depending on the structure of the page and the way you define your selectors.

```
<header>
  <h1>Our Blog</h1>
</header>

<article>
  <header>
    <h1>This is the best blog title ever</h1>
  </header>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
</article>
<article>
  <header>
    <h1>This is the second best blog title</h1>
  </header>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
</article>
```

Structuring pages cleanly allows for simplified selectors which keep the specificity level low making it easier to override selectors later on as the stylesheet grows.

```
header h1 { color: red; }
article header h1 { color: blue; }
```

## General Markup Guidelines

The following are general guidelines for structuring your HTML markup. Authors are reminded to always use markup which represents the semantics of the content in the document being created.

- Use actual `p` elements for paragraph delimiters as opposed to multiple `br` tags.
- Make use of `dl` (definition lists) and `blockquote`, when appropriate.
- Items in list form should always be housed in a `ul`, `ol`, or `dl`, never a `div` or a `p`.
- Use `label` fields to label each form field, the `for` attribute should associate itself with the input field, so users can click the labels. `cursor:pointer;` on the label is wise, as well. note 1 (http://www.accessifyforum.com/viewtopic.php?t=1926#14178) note 2 (http://www.accessifyforum.com/viewtopic.php?t=6738)
- Do not use the size attribute on your `input` fields. The size attribute is relative to the font-size of the text inside the input. Instead use css width.
- Place an html comment on some closing div tags to indicate what element you're closing. It will help when there is lots of nesting and indentation.

- Tables shouldn't be used for page layout.
- Use microformats (http://en.wikipedia.org/wiki/Microformat) and/or Microdata where appropriate, specifically hCard and adr.
- Make use of `thead`, `tbody`, and `th` tags (and the `scope` attribute) when appropriate.

```
<table summary="This is a chart of year-end returns for 2005.">
  <thead>
    <tr>
      <th scope="col">Table header 1</th>
      <th scope="col">Table header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Table data 1</td>
      <td>Table data 2</td>
    </tr>
  </tbody>
</table>
```

- Always use title-case for headers and titles. Do not use all caps or all lowercase titles in markup, instead apply the CSS property `text-transform:uppercase/lowercase`.

### Quoting Attributes

The HTML5 specification defines quotes around attributes as optional. For consistency with attributes that accept whitespace, **all attributes should be double quoted**.

```
<p class="line note" data-attribute="106">This is my paragraph of special text.</p>
```

# Stylesheets

## General Coding Principles

- Add CSS through external files, minimizing the # of files, if possible. It should always be in the HEAD of the document.
- Use the `link` tag to include, never the @import. (http://blog.amodernfable.com/2008/01/21/thoughts-on-linking-to-stylesheets/)
- Don't include styles inline in the document, either in a style tag or on the elements. It's harder to track down style rules.
- Use normalize.css (http://necolas.github.com/normalize.css/) to make rendering more consistent across browsers.
- Documentation should be written using Markdown.
- Elements that occur only once inside a document should use IDs, otherwise, use classes.
- Understand cascading and selector specificity (http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html) so you can write very terse and effective code.
- Write selectors that are optimized for speed. Where possible, avoid expensive CSS selectors. For example, avoid the * wildcard selector and don't qualify ID selectors (e.g. `div#myid`) or class selectors (e.g. `table.results`.) This is especially important with web applications where speed is paramount and there can be thousands or even tens of thousands of DOM elements. More on writing efficient CSS on the MDC (https://developer.mozilla.org/en/Writing_Efficient_CSS).

### Understanding the Box Model is Key

The "box model" is a key determining factor in how a browser renders your page. A healthy understanding of it's intricacies will make your job so indescribably easier. The box model denotes the way in which the physical dimensions of a HTML element are calculated. If a block element has a fixed width of say, 100px, then how should the padding, border and margin be placed?

Plenty of websites offer in depth descriptions, but put simply: the standards compliant implementation places the border and padding outside of the specified width.

There is a fix at hand, and it involves a CSS property called `box-sizing`, and it works in **IE8 and above**. It allows you to choose the exact way in which an elements dimensions are calculated, and its a lifesaver. Parameter support varies and vendor prefixes apply, so consult caniuse (http://caniuse.com/css3-boxsizing) for specifics.

```
/* the old way (178 + 20 + 2 = 200) */
.foo {
  width: 178px;
  padding: 10px;
  border: 1px;
}

/* a better way */
.foo {
  width: 200px;
  padding: 10px;
  border: 1px;
  box-sizing: border-box;
}
```

While it was always possible to mentally calculate widths by removing pixel units from each other (as per the first method), it was never entirely clear how to do so with variable width units like percentages and EMs. There was no other solution at this point besides wrapping elements in parent elements to ensure widths and `padding/margin/borders` could all be separate.

### CSS Validation

We typically don't use the W3C validator (http://jigsaw.w3.org/css-validator/).

### CSS Formatting

At minimum, format CSS with selectors on one line and each property on its own line.

As an enhancement to that style, related or child styles and additional 2 spaces. That allows for hierarchical scanning and organization and makes (for some people) an easier-to-read style sheet.

Also, if you specify multiple selectors, it's a good idea to start each on new line. This prevents lines from growing long and improves readability as well as version control workflow.

```
.post-list li a{
  color:#A8A8A8;
  background-color:#FFF;
}
  .post-list li a:hover{
    color:#000;
    text-decoration:none;
  }
  .post-list li .author a,
  .post-list li .author a:hover{
    color:#F30;
    text-transform:uppercase;
  }
```

For multiple author environments, single line CSS should be avoided because it can cause issues with version control.

```
.post-list li span{ background:red; } /* bad */
```

### Alphabetize

If you're performance obsessed alphabetizing CSS properties increases the odds of larger repeatable patterns being present to aid in GZIP compression (http://www.barryvan.com.au/2009/08/css-minifier-and-alphabetiser/).

## Classes vs. IDs

You should only give elements an ID attribute if they are unique. They should be applied to that element only and nothing else. Classes can be applied to multiple elements that share the same style properties. Things that should look and work in the same way can have the same class name.

```
<ul id="categories">
  <li class="item">Category 1</li>
  <li class="item">Category 2</li>
  <li class="item">Category 3</li>
</ul>
```

## Naming Conventions for Selectors

All `id` and `class` declarations in CSS shall be written in only lowercase. Words should be separated by dashes.

It is always preferable to name something, be it an ID or a class, by the nature of **what it is** rather than by **what it looks like**. For instance, a class name of `big-blue-text` for a special note on a page is quite meaningless if it has been changed to have a small red text color. Using a more intelligent convention such as `note-text` is better because when the visual style changes it still makes sense.

## Selectors

The CSS Selectors Level 3 (http://www.w3.org/TR/2009/PR-css3-selectors-20091215/) specification introduces a whole new set of CSS Selectors (http://www.w3.org/TR/css3-selectors/#selectors) that are extremely useful for better selection of elements.

### Pseudo-classes

Pseudo-classes (http://www.w3.org/TR/css3-selectors/#pseudo-classes) enable you to dynamically style content. Some pseudo-classes have existed since CSS1 ( `:visited, :hover` , etc.) and CSS2 ( `:first-child, :lang` ). As of CSS3, 16 new pseudo-classes have been added to the list and are especially useful for styling dynamic content. Learn how to use pseudo-classes in-depth (http://www.smashingmagazine.com/2011/03/30/how-to-use-css3-pseudo-classes/).

### Combinators & Attribute Selectors

Combinators (http://www.w3.org/TR/css3-selectors/#combinators) provide shortcuts for selecting elements that are a descendant element, a child element, or an element's sibling.

Attribute Selectors (http://www.w3.org/TR/css3-selectors/#attribute-selectors) are great for finding elements that have a specific attribute and/or specific value. Knowledge of regular expressions helps with attribute selectors.

### Specificity

Browsers calculate a selector's specificity (http://www.w3.org/TR/2009/PR-css3-selectors-20091215/) to determine which CSS rule should apply. If two selectors apply to the same element, the one with the **higher specificity wins**.

IDs have a higher specificity than attribute selectors do, and class selectors have higher specificity than any number of element selectors. Always try to use IDs to increase the specificity. There are times when we may try to apply a CSS rule to an element and it does not work no matter what we try. This is likely because the specificity of the selector used is lower than another one and the properties of the higher one are taking precedence over those you want to apply. This is more common in working with larger more complex stylesheets. It isn't a big issue with smaller projects usually.

#### Calculating specificity

When working with a large and complex stylesheet it helps to know how to calculate the value of a selector's specificity, to save you time and to make your selectors more efficient.

- Specificity Calculator (http://specificity.keegan.st/)
- Some things you should know about specificity (http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/)
- IE Specificity bugs (http://www.brunildo.org/test/IEASpec.html#a)

Using `!important` overrides all specificity no matter how high it is. We like to avoid using it for this reason. Most of the time it is not necessary. Even if you need to override a selector in

a stylesheet you don't have access to, there are usually ways to override it without using !important. Avoid using it if possible.

### Pixels vs. Ems

We use the `em` unit of measurement to define `font-size`.

### Know when to Float, and when to Position

Floats are great for sucking elements out of the DOM and forcing them hard up against a left or a right edge. They became the bread and butter of the post table layout stage in front end dev, possibly because of the poor browser support of `display: inline` and `inline-block`, as well as `z-index` bugs stemming from position support. These days there really is no excuse. Inline-block is fairly well supported, and a quick hack will get it working in IE7.

The arguments that previously held back absolutely positioning elements with CSS have thankfully died down. In theory, positioning allows you to place elements on a page (or within any container for that matter) with Xs and Ys in a straightforward manner that should be familiar to people like Flash developers.

### Internet Explorer Bugs

Inevitably, when all other browsers appear to be working correctly, any and all versions of Internet Explorer will introduce a few nonsensical bugs, delaying time to deployment. While we encourage troubleshooting and building code that will work in all browsers without special modifications, sometimes it is necessary to use conditional `if IE` comments for CSS hooks we can use in our stylesheets. Read more on paulirish.com (http://paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/)

Fixing IE

```
<!--[if lt IE 7]> <html class="no-js ie old-ie" lang="en"> <![endif]-->
<!--[if IE 7]>    <html class="no-js ie old-ie ie7" lang="en"> <![endif]-->
<!--[if IE 8]>    <html class="no-js ie old-ie ie8" lang="en"> <![endif]-->
<!--[if IE 9]>    <html class="no-js ie ie9" lang="en"> <![endif]-->
<!--[if gt IE 9]><!--><html class="no-js" lang="en"><!--<![endif]-->
```

### Shorthand

In general, CSS shorthand is preferred because of its terseness, and the ability to later go back and add in values that are already present, such as the case with margin and padding. Developers should be aware of the TRBL (Top Right Bottom Left) acronym, denoting the order in which the sides of an element are defined, in a clock-wise manner: Top, Right, Bottom, Left. If *bottom* is undefined, it inherits its value from *top*. Likewise, if *left* is undefined, it inherits its value from *right*. If only the *top* value is defined, all sides inherit from that one declaration.

For more on reducing stylesheet code redundancy, and using CSS shorthand in general:

- (http://qrayg.com/journal/news/css-background-shorthand)http://qrayg.com/journal/news/css-background-shorthand (http://qrayg.com/journal/news/css-background-shorthand)
- (http://sonspring.com/journal/css-redundancy)http://sonspring.com/journal/css-redundancy (http://sonspring.com/journal/css-redundancy)
- (http://dustindiaz.com/css-shorthand)http://dustindiaz.com/css-shorthand (http://dustindiaz.com/css-shorthand)

### Images

- For repeating images, use something larger than 1x1 pixels (http://www.iandevlin.com/blog/2010/03/webdev/fading-issue-with-repeating-background-transparent-image-in-internet-explorer)
- You should never be using spacer images.
- Use CSS sprites generously. They make hover states easy, improve page load time, and reduce carbon dioxide emissions.
- Typically, all images should be sliced with a transparent background (PNG8). All should be cropped tightly to the image boundaries.
- However, the logo should always have a background matte and have padding before the crop. (so other people can hotlink to the file)

### Color Management

- Verify that all members on your team have consistent color management settings.
- Any given two monitors most likely display the colors differently, but sRGB color profile must be your default.
- When you open files in Photoshop, pay attention to Color Profile warnings and notify team members when Photoshop is suggesting to convert an image to a different profile.* Never embed a color profile in your images.
- When you Save for Web and Devices from Photoshop, be sure to uncheck "Embed Color Profile."

### General Text and Font Styling

#### Headings

- Define default styling for `h1-h6` headings including headings as links. It's helpful to declare these at the top of your CSS document, and modify them with as necessary for consistency across the site.
- Headings should show a hierarchy indicating different levels of importance from the top down starting with h1 having the largest font size.
- SEO: To get a rough idea of how your page hierarchy is organized and read, use your Developer Toolbar to disable CSS. You'll end up with a text-based view of all your `h1-h6` tags, `strong`, `em`, etc.

#### Links

- Default styles for links should be declared and different from the main text styling, and with differing styles for hover state.
- When styling links with underlines use `border-bottom` and some padding with `text-decoration: none;`. This just looks better.

### Vertical & Horizontal Centering

Centering elements horizontally is not exactly rocket science, and I'm sure most of you are familiar with the following snippet:

```
.class { width: 960px; margin: 0 auto; }
```

Front end devs have been using this snippet for a long time, without fully understanding why it didn't work vertically. From my understanding, it's important to remember that the parent element will generally have a `height: auto;` on it, and that there is no 100% height in which to vertically center the element. Applying the `position: absolute;` effectively moves the element out into position mode and responds to the pushing and pulling of auto margins and no specific location.

```
.exactMiddle {
  width: 100px;
  height: 100px;
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  margin: auto;
}
```

The downsides of this method include its lack of support in IE6 and IE7, and the absence of a scroll bar if the browser is resized to be smaller than the centered object. There are more methods on this page (http://blog.themeforest.net/tutorials/vertical-centering-with-css/) (this is Method 4), but this is by far the best.

The vertical centering of text in an element is also straightforward. If the text is on a single line, like a horizontal navigation item, you can set the `line-height` to be that of the element's physical height.

```
#horizNav li {
  height: 32px;
  line-height: 32px;
}
```

### Feature Sniff, Don't Browser Sniff

In the earlier discusison of JavaScript feature detection, applying properties if a browser is *any version* of IE is increasingly problematic. Man-of-steel Paul Irish pioneered the use of IE version sniffing (http://paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/) to address this problem, but Modernizr (http://www.modernizr.com) has since come to the rescue. Modernizr places classes on the root `<html>` element specifying whether features are supported. Bleeding edge styles can then easily cascade from (or be removed from) these classes.

```
.my_elem {
  -webkit-box-shadow: 0 1px 2px rgba(0,0,0,0.25);
  -moz-box-shadow: 0 1px 2px rgba(0,0,0,0.25);
  box-shadow: 0 1px 2px rgba(0,0,0,0.25);
}

/* when box shadow isn't supported, use borders instead */
.no-boxshadow .my_elem {
  border: 1px solid #666;
  border-bottom-width: 2px;
}
```

### You're Not !important

A reliance upon the `!important` tag is a dangerous thing. The cases that warrant its use are rare and specific. They revolve around the necessity to override another stylesheet which you do not have access or permission to edit. Another scenario is hard coding an element's styles to prevent inline JavaScript styles from taking precedence. Instead `!important` is used as a lazy shortcut to set the priority of your style over another, causing headaches further down the line.

The use of the `!important` tag can be mostly avoided via the better understanding of CSS selector precedence, and how to better target elements. The more specific the selector, the more likely it will be accepted as the applicable style. The following example from vanseodesign demonstrates the specificity at work.

```
p { font-size: 12px; }
p.bio { font-size: 14px; }
```

Their article (http://www.vanseodesign.com/css/css-specificity-inheritance-cascaade/) on style precedence does a better job explaining inheritence than I ever could, so please give it a go.

### Aggressive Degradation

It's worth noting that this is a personal opinion, and best suited to very specific situations. The stance of aggressive degradation will not be well received in large commercial projects or enterprise solutions relying upon older browsers.

Aggressive degradation dictates that if a particular (older) browser cannot render a certain effect, it should simply be omitted. A CSS3 button is a good example. Effects such as `border-radius`, `box-shadow`, `text-shadow` and `gradients` will be displayed in cutting edge browsers. A graceful fallback of a `.PNG` would be provided for slightly older browsers, and the most graceful of all solutions would include a PNG-Fix for IE6 or the use of `filter` arguments to replicate gradients and shadows. However, aggressive degradation in this situation instructs you to neglect the older browsers and present them with a flat, satisfactory object.

Put simply, aggressive degradation boils down to: **if your browser can't render a gradient or a box shadow, tough luck**.

While not ideal for every situation, it ensures the timely delivery of projects and that the root product is still usable and not reliant on (validation breaking) hacks.

## Web Typography

The use of custom fonts and typefaces on the web has been growing more popular the past few years. with native browser support on the rise and several supporting services and APIs now available there is real momentum in this space. Each of these approaches have their own pros and cons. Before a project kicks off it's best to do research into technology and licensing limitations to choose the proper approach for the specific project.

All of these approaches have drawbacks in code overhead, development time and performance (clock and perceived). Familiarizing yourself with these issues and communicating them to the other members of the team and to the client will save significant problems later on in the project.

Listed here are some popular methods of embed custom fonts, list in the order of our preference for implementation.

## @font-face

The @font-face at-rule (http://www.w3.org/TR/2011/WD-css3-fonts-20110324/#font-face-rule) allows you to define custom fonts. It was first defined in the CSS2 specification, but was removed from CSS2.1. Currently, it's a draft recommendation for CSS3.

Our first and most preferred choice for customizing fonts on the web is @font-face, simply because it is part of the CSS Fonts Module working draft which means it will continue to grow in popularity as browser support grows, and ease of use for it improves as it becomes more stable.

### Use and Abuse

Even though you can transform just about any font into a web font file, you should still make sure it is legally okay for you to do so. Therefor inspect the EULA and check if web embedding is allowed.

Foundries are understandably reluctant to allow designers and developers the ability to place font files directly on a server which can then be copied by a savvy end user. Particular foundries also prohibit the embedding of particular file types, such as `.TTF` and `.OTF`. Many foundries have updated their conditions to specify how their fonts can be used on the web. View Font Licensing and Protection Details (http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=UNESCO_Font_Lic) for more information.

If, after careful consideration, you believe the desired font is web embeddable: head on over to the Font Squirrel @font-face Generator (http://www.fontsquirrel.com/fontface/generator). It utilises Fontspring's bulletproof @font-face structure (http://www.fontspring.com/blog/further-hardening-of-the-bulletproof-syntax) and automatically generates all the required file formats.

### Bulletproof @font-face

For now, when using `@font-face` it's recommended to declare the source for each font format. This is important if you want it to work in the most number of browsers, though it is not a requirement for use.

The font formats included in the specification are:

- **woff**: WOFF (Web Open Font Format)
- **ttf**: TrueType
- **ttf**, **otf**: OpenType
- **eot**: Embedded OpenType
- **svg**, **svgz**: SVG Font

For full cross-browser compatibility use Fontsprings' new bulletproof @font-face syntax (http://www.fontspring.com/blog/further-hardening-of-the-bulletproof-syntax) (*latest version as of 2/21/11*).

```
@font-face {
  font-family: 'MyFontFamily';
  src: url('myfont-webfont.eot');           /* IE9 Compat Modes */
  src: url('myfont-webfont.eot?iefix') format('eot'), /* IE6-IE8 */
     url('myfont-webfont.woff') format('woff'),   /* Modern Browsers */
     url('myfont-webfont.ttf') format('truetype'),    /* Safari, Android, iOS */
     url('myfont-webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
  font-weight: <font-weight>;
  font-style: <font-style>;
  // etc.
}
```

Here's a working demo (http://www.thecssninja.com/demo/css_fontface/) using this version of implementation.

### Prevent Compatibility Mode

Sometimes IE (Internet Explorer) can have a mind of its own and will switch to compatibility mode without you knowing. Include the following in the site `<head>` to prevent your site from defaulting to compatibility mode:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

### Tips for @font-face

- IE 6–8 will only accept a TrueType font packaged as an EOT.
- font-weight and font-style have different meanings within `@font-face`. Declarations where `font-weight:bold;` means this is the bold version of this typeface, rather than apply bold to this text
- @font-face gotchas (http://paulirish.com/2010/font-face-gotchas/)

**Pros**

- Easy to implement
- Large variety of APIs
- Customizable
- Easy to add to elements
- Nothing required besides CSS
- Is currently part of the working draft of CSS Fonts Module 3

**Cons**

- Limited browser support if used improperly
- Some older versions of modern browsers (Chrome, Opera) don't always render well. Text can have rough edges. ***I have not been able to confirm whether this is still an issue now or not.*

## Google WebFonts API & Font Loader

There are two options available with Google Webfonts (https://code.google.com/apis/webfonts/). Both options have their downsides of course but they can be just as good to use as `@font-face`, it all depends on a projects needs.

### Webfonts API

Google's Webfonts API (https://code.google.com/apis/webfonts/docs/getting_started.html) essentially does the same thing as `@font-face`, it just does all the hard work for you, providing wider browser support.The major drawback to this method is the very small font library it uses. To make it work all you need to do is include the stylesheet + the font name.

```
<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Font+Name">
```

Then define a style for the selector you want to apply the font to:

```
css-selector {
  font-family: 'Font Name', serif;
}
```

## Webfont Loader

Another option Google offers is the Webfont Loader (https://code.google.com/apis/webfonts/docs/webfont_loader.html) which is a JavaScript library that allows for more control than the font API does. You can also use multiple webfont providers like Typekit. To use it include the script in your page:

```
<script type="text/javascript">
  WebFontConfig = { google: { families: [ 'Tangerine', 'Cantarell' ]} };
  (function() {
    var wf = document.createElement('script');
    wf.src = ('https:' == document.location.protocol ? 'https' : 'http') +
          '://ajax.googleapis.com/ajax/libs/webfont/1/webfont.js';
    wf.type = 'text/javascript';
    wf.async = 'true';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(wf, s);
  })();
</script>
```

Including the webfont.js file this way is faster if not already using the Ajax APIs. Otherwise you should use this:

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
  google.load("webfont", "1");
  google.setOnLoadCallback(function() {
    WebFont.load({ google: { families: [ 'Tangerine', 'Cantarell' ]} });
  });
</script>
```

By using the Webfont Loader you have more ability to customize things including the use of more fonts, not just those in the Google Webfont library which is not large. However, it then requires you to load JavaScript, sacrificing one thing for another.

**Pros**

- Very easy to implement
- Wide browser support
- Can be combined with Typekit
- Customizable when using the font loader
- API does the same thing as `@font-face`

**Cons**

- Very small font library if using the font API
- Using the Webfont Loader requires the use of JavaScript to work
- Most browsers will load the rest of the page first, leaving a blank space where the text would be, or otherwise show the fallback option if one exists, until the page fully loads.
- Some fonts in the webfont library render poorly on Windows

## Typekit

Using Typekit (https://typekit.com) has its advantages and shouldn't be completely disregarded when choosing which method to use for adding custom fonts to a web site. It has strong platform integration and is a scalable and popular service. It can be used with Google Webfonts and is easily added to WordPress, Posterous, Typepad, and other similar CMS powered sites.

However, full use of Typekit doesn't come without a cost (https://typekit.com/plans). If you need to use it on more than 2 sites or on a site that gets a high amount of pageviews you will need to pay an annual cost of $49.99, and for sites with a million+ pageviews it costs twice as much. Though, you probably have the money to cover the cost if you're getting over a million pageviews. If not then you may want to rethink your business model.

**Pros**

- Large font library, including Adobe fonts
- Easy implementation
- Google Webfont API and blogging platform integration
- Free plan has limits but doesn't expire

**Cons**

- Requires JavaScript to use
- Limited font library access without paying
- Free and cheapest plans only allow use on 1-2 web sites and 2-5 fonts per site
- You have to pay to use it on more than 1 site

### Specifications & Font File Formats

- CSS 2 Fonts (http://www.w3.org/TR/1998/REC-CSS2-19980512/fonts.html#font-descriptions) – May 1998 (Obsolete)
- CSS 3 Fonts (http://www.w3.org/TR/css3-fonts/) – Working Draft 2009
- CSS Fonts Module (http://www.w3.org/TR/css3-fonts/) – W3C Working Draft March 2011
- WOFF Font Format (http://www.w3.org/TR/WOFF/) – Working Draft 2010
- SVG Font Format (http://www.w3.org/TR/SVG/fonts.html)
- Embedded Open Type (EOT) File Format (http://www.w3.org/Submission/EOT/)
- Microsoft Open Type Specification (http://www.microsoft.com/typography/otspec/)
- OpenType Feature File Specification (http://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html#9.e)
- Apple True Type Reference (http://developer.apple.com/fonts/TTRefMan/)

### Use CSS3

You can do a lot of new things with the added features in the CSS3 spec, many of which are not yet fully supported by all the major browsers. That doesn't mean they can't be used today. For the things that aren't supported there are fallback libraries or other 'polyfills' which you can use to fill in the holes where browsers may be lacking some support of a new feature.

# Javascripts

## JavaScript Libraries

We primarily develop new applications in jQuery (http://api.jquery.com/).

## General Coding Principles

- 99% of code should be housed in external javascript files. They should be included at the END of the BODY tag for maximum page performance.
- Don't rely on the user-agent string. Do proper feature detection. (More at Dive Into HTML5: Detection (http://diveintohtml5.info/detect.html) & jQuery.support docs (http://api.jquery.com/jQuery.support/))
- Don't use document.write().
- All Boolean variables should start with "is".
- Test for positive conditions

```
isValid = (test.value >= 4 && test.success);
```

- Name variables and functions logically: For example: `popUpWindowForAd` rather than `myWindow`.
- Don't manually minify. With the exception of the traditional `i`, etc. for `for` loops, variables should be long enough to be meaningful.
- Documentation should be written using Markdown.
- Constants or configuration variables (like animation durations, etc.) should be at the top of the file.
- Strive to create functions which can be generalized, take parameters, and return values. This allows for substantial code reuse and, when combined with includes or external scripts, can reduce the overhead when scripts need to change. For example, instead of hard coding a pop-window with window size, options, and url, consider creating a function which takes size, url, and options as variables.
- Comment your code *using markdown*! It helps reduce time spent troubleshooting JavaScript functions.
- Don't waste your time with `<![CDATA[ ]]>` or `<!-- -->` comments surrounding your inline javascript.
- Organize your code as an Object Literal/Singleton (http://kaijaeger.com/articles/the-singleton-design-pattern-in-javascript.html), in the Module Pattern (http://www.yuiblog.com/blog/2007/06/12/module-pattern/), or as an Object with constructors (http://mckoss.com/jscript/object.htm).
- Minimize global variables - the less globals you create, the better. Generally one, for your application namespace, is a good number.
- When specifying any global variable, clearly identify it

```
window.globalVar = { ... }
```

### White-space

Any discussion about formatting, whitespacing and the placement of braces is going to be hotly debated. I guess the simplest rule is that, unless you're willing to completely format a whole document, **respect and maintain the formatting of an existing document**. That means: see same-line braces throughout a JS file, continue to write code with same-line braces. Your code should fail the code review process if it doesn't maintain consistency with the rest of the document.

Consistent formatting makes code more readable, and also means the code can be easily modified with find and replace commands. The coding habits we have picked up are thankfully very similar to what jQuery officially encourages. There are a few minor discrepencies, but again, these are personal issues or things that we think cannot be maintained. Further Reading

(http://docs.jquery.com/JQuery_Core_Style_Guidelines)

In general, the use of whitespace should follow longstanding English reading conventions. Such that, there will be one space after each comma and colon (and semi-colon where applicable), but no spaces immediately inside the right and left sides of parenthesis. In short, we advocate readability within reason. Additionally, braces should always appear on the same line as their preceding argument.

Consider the following examples of a JavaScript for-loop...

**Correct**

```
for (var i = 0, j = arr.length; i < j; i++) {
  // Do something.
}
```

**Incorrect**

```
for ( var i = 0, j = arr.length; i < j; i++ )
{
  // Do something.
}
```

**Also incorrect**

```
for(var i=0,j=arr.length;i< j;i++){
  // Do something.
}
```

## Variables, ID & Class

All JavaScript variables shall be written in either completely lowercase letter or camelCase. The two exceptions to this are Constructor functions, which are capitalized, and Constants which are all uppercase.

## Event Delegation

When assigning unobtrusive event listeners, it is typically acceptable to assign the event listener directly to the element(s) which will trigger some resulting action. However, occasionally there may be multiple elements which match the criteria for which you are checking, and attaching event listeners to each one might negatively impact performance. In such cases you should use event delegation instead.

jQuery's on() delegation (http://api.jquery.com/on/#direct-and-delegated-events) is usually used.

## Always Use === Comparison

The use of the == equality operator allows for frustrating bugs to slip through almost undetected. It allows for weak typing that is best explained by JavaScript Garden (http://bonsaiden.github.com/JavaScript-Garden/#types.equality). The use of the strict equality operator === does not run type coercion and therefore strictly evaluates the difference between two objects. Again, consult JavaScript Garden (http://bonsaiden.github.com/JavaScript-Garden/#types.equality) for more information

```
var zeroAsAString = "0";
if (zeroAsAString == 0) { // gets in here lolwut }
if (zeroAsAString === 0) { // never gets in here }
```

### The Exception

Double equals comparison is allowed when comparing to null, because it will detect both null or undefined properties. If you don't fully understand this, I still suggest you use triple equals.

```
var foo = null;

// foo is null, but bar is undefined as it has not been declared

if (foo == null && bar == null) { // still got in here }
```

## Always Specify the Second 'radix' Parameter When Using .parseInt()

When parsing a string to an integer, it is considered good practice to specify the second 'radix' parameter - which determines to what base the string should be converted to. The default setting will trigger a radix of 16 whenever the string is lead by a `0`. Most beginner and intermediate users are only ever going to be using a radix of `10`. Thanks to João Moreno for logging the correction (https://github.com/taitems/Front-End-Development-Guidelines/issues/23).

```
console.log( parseInt("08") ); // logs: 2
console.log( parseInt("08", 10) ); // logs: 8
```

## Avoid Comparing to true and false

Direct comparison to the values of true and false is unnecessary. Sometimes it might be good for clarity, but it's just extra code.

```
if (foo === true) { // good that they're using triple equals, bad as it's redundant }
if (foo) { // yay! }
if (!bar) { // the opposite }
```

## Avoid Polluting the Global Namespace

An over-reliance on global variables is something all of us, myself especially, are guilty of. Arguments as to why globals are bad are fairly straight forward: the chance of script and variable conflicts is increased, and both the source file and the namespace itself become littered with countless ambiguously named variables.

Douglas Crockford (http://yuiblog.com/) believes that the quality of a JavaScript application can be assessed by the number of global variables it uses; the less the better. Given that not everything can be a local (but let's be honest, that one you're thinking about right now, it can, don't be lazy) you need to find a way of structuring your variables to prevent clashes and minimise the bloat. The easiest way is to employ a single variable or a minimal amount of modules on which the variables are set. Crockford mentions that YUI uses a single global, `YAHOO`. He discusses this in more detail in his blog post "Global Domination" (http://yuiblog.com/blog/2006/06/01/global-domination/).

Considering that, in the case of small web apps, globals are generally used to store application-wide settings: it's generally better to namespace your project or settings as objects.

```
// polluted global name space
var settingA = true;
var settingB = false;
var settingC = "test";

// a settings namespace
var settings = {
  settingA: true,
  settingB: false,
  settingC: "test"
};
```

But if we're avoiding globals to reduce the chance of conflicts, isn't standardising the namespaces to be the same going to increase chance of one app's settings overwriting anothers? Well, it would make sense. It is instead suggested that you namespace your globals to your own specific app name, or reassign your namespace much in the same way that jQuery uses `$.noConflict()` mode.

```
var App = { settings: { settingA: true } };

// accessed as
App.settings.settingA; // true
```

### Camel Case Variables

The camel casing (or *camelCasing*) of JavaScript variables is accepted as the standard in most coding environments. The only exception that was raised in the comment section is the use of uppercase and underscores to denote contants.

```
var X_Position = obj.scrollLeft; // bad
var xPosition = obj.scrollLeft; // good
SCENE_GRAVITY = 1; // constant // good
```

### Loop Performance - Cache Array Length

Looping is arguably the most important part of JavaScript performance to get right. Shave a millisecond or two off inside of a loop, potentially gain seconds overall. One such way is to cache the length of an array so it doesnt have to be calculated every time the loop is iterated through.

```
var toLoop = new Array(1000);

for (var i = 0; i < toLoop.length; i++) {
  // BAD - the length has to be evaluated 1000 times
}

for (var i = 0, len = toLoop.length; i < len; i++) {
  // GOOD - the length is only looked up once and then cached
}
```

#### The Exception

If you're looping through an array to find and remove a particular item, this will alter the array length. Any time you change the array length by either adding or removing items from inside the loop, you will get yourself into trouble. Consider either re-setting the length or avoid caching it for this particular situation.

### Loop Performance - Use 'break;' & 'continue;'

The ability to step over and out of loops is really useful in avoiding costly loop cycles.

If you're looking for something inside of a loop, what do you do once you find it? Say the condition you're looking for is matched halfway through a 1000 item loop. Do you execute whatever you intend to do, and allow the loop to continue to iterate over the remaining 500 items, knowing that there's no chance it will hit an if statement? Nope! You break out of your loop, literally!

```
var bigArray = new Array(1000);
for (var i = 0, len = bigArray.length; i < len; i++) {
  if (i === 500) { break; }
  console.log(i); // will only log out 0 - 499
}
```

Another problem is skipping over a particular iteration and then continuing on with the loop. While things like odds and evens are better managed by replacing `i++` with `i + 2`, some conditions need to be specifically listened for, to then trigger the skip. Anything that prevent's running through an entire iteration is pretty handy.

```
var bigArray = new Array(1000);
for (var i = 0, len = bigArray.length; i < len; i++) {
  if (condition) { continue; }
  doCostlyStuff();
}
```

### Don't Send Too Many Function Parameters

This is a pretty bad idea, more for readability than anything:

```
function greet(name, language, age, gender, hairColour, eyeColour) {
  alert(name);
}
```

It's a much better idea to construct an object before-hand or to pass the object inline

```
function greet(user) {
  alert(user.name);
}

greet({ name: "Bob", gender: "male" });
```

### Remap 'this' to 'self'

When writing object-oriented (OO) JavaScript, the scope of `this` must be understood. Regardless of what design pattern you choose to structure your pseudo-classes, a reference to `this` is generally the easiest way to refer back to an instance. The moment you begin integrating jQuery helper methods with your pseudo-classes is the moment you notice the changing scope of `this`.

```
Bob.findFriend("Barry");

Person.prototype.findFriend = function(toFind) {
  // this = Bob
  $(this.friends).each(function() {
    // this = Bob.friends[i]
    if (this.name === toFind) {
      // this = Barry
      return this;
    }
  });
}
```

In the above example, `this` has changed from a reference to `Bob`, to his friend `Barry`. It's important to understand what happened to the value of `this` over time. Inside of the prototyped function, `this` refers to the instance of the pseudo-class (in this case `Bob`). Once we step inside the `$.each()` loop, `this` is then re-mapped to be item `i` in the parsed array.

The solution is to remap the value of `this` to either `self` or `_self`. While `self` (sans underscore) is not exactly a reserved keyword (https://developer.mozilla.org/en/JavaScript/Reference/Reserved_Words), it *is* a property of the `window` object. Although my use of `self` was picked up from the jQuery source code, they have realised their mistake and are attempting to rectify the situation (http://bugs.jqueryui.com/ticket/5404) and instead use `_self`. Personally, I prefer the use of `self` for the sheer cleanliness - but it can throw some pretty confusing bugs for people. Tread carefully.

In the following example I will better utilise the parameters made available with the `$.each()` helper, as well as re-mapping the value of `this`.

```
Bob.findFriend("Barry");

Person.prototype.findFriend = function(toFind) {
  // the only time "this" is used
  var _self = this;
  $(_self.friends).each(function(i,item) {
    if (item.name === toFind) { return item; }
  });
}
```

### CanIHaz Boolean?

Booleans should be easily identifiable by the way they are named. Use prefixes like `is`, `can` or `has` to propose a question.

```
isEditing = true;
obj.canEdit = true;
user.hasPermission = true;
```

### Minimizing Repaints & Reflows

Repaints and reflows relate to the process of re-rendering the DOM when particular properties or elements are altered. Repaints are triggered when an element's look is changed without altering its layout. Nicole Sullivan describes these changes in a thorough blog post (http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/) as style changes such as visibility or background-color. Reflows are the more costly alternative, caused by changes that alter the layout of the page. Examples include the addition or removal of elements, changes to an element's width or height, and even resizing the browser window. Worst yet is the domino effect of reflows that cause ancestor, sibling and child elements to reflow.

There is no doubt that both reflows and repaints should be avoided if possible, but how?

#### A Reflow Example

It's not that the following snippet is "bad code" exactly. But let's assume that the array `arr` has 10 items.

```
var myList = document.getElementById("myList");
for (var i = 0, len = arr.length; i < len; i++) {
  myList.innerHTML += "<li>" + arr[i].title + "</li>"; //reflow - appending to element
}
```

In the above `for` loop, a reflow will be triggered for every iteration of the loop. 10 iterations cause 10 reflows.

Now consider the following:

```
var constructedHTML = "";
for (var i = 0, len = arr.length; i < len; i++) {
  constructedHTML += "<li>" + arr[i].title + "</li>"; //no reflow - appending to string
}
document.getElementById("myList").innerHTML = constructedHTML; //reflow
```

In this scenario, the elements are being constructed within a string. Not a single reflow is created by the loop, as the DOM is not being altered. Only once the array has been completely looped through is the string then applied as the `innerHTML` of an object, causing the only reflow of the function.

There are endless types of reflows and repaints that can be avoided, and lucky you gets to go on an read about them. Reading material on the subject matter is plentiful, but most of it is

linked to from the excellent starting point that is Nicole Sullivan's blog post (http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/). There are important lessons to be taken away from this when it comes to a multitude of technologies synonymous with "web 3.0" and HTML5. The lesson above can be directly applied to writing jQuery. It's also important to consider when fiddling with `canvas`, and trying to keep a frame rate in the 30-60 range.

### Don't Use Milliseconds to Generate Unique IDs

There is a method for generating unique IDs that has hung around since the early days of web dev. It involved appending the elapsed milliseconds since January 1, 1970 to your static ID by way of:

```
var myID = "static" + new Date().getTime();
```

This was a fairly foolproof method originally, because even if two of the above lines were performed one after the other, a few millisecondsnormally separated their execution. New browsers brought with them newJavaScript engines, coupled with ever increasing clock speed. Thesedays it's more likely that your milliseconds match than are slightlyincremented.

This leads to bugs that are near impossible to debug by conventionalmeans. Because your DOM is created on the fly, traditional validatio nof the page source won't identify multiple IDs as an error. JavaScript and iQuery error handling dictates that the first match for the ID will be utilised and other matches ignored. So it doesn't even throw a JS error!

No, the only real method to debug it is line by line breakpointing and logging - but "pause" at the wrong line and your milliseconds will no longer clash!

The good thing is that there are plenty of alternatives. To be pedantic, it's worth noting that a computer's random function is not truly random as it is seeded by system time - but the probability of clashes is rather minuscule.

```
var myID = "static" + Math.round(Math.random() * 10000);
```

Personally, I'm partial to a bit of faux GUID generation. Technically a GUID is generated according to your hardware, but this JavaScript function does the next best thing. The following is a handy function I've pinched from a stack overflow post (http://stackoverflow.com/questions/105034/how-to-create-a-guid-uuid-in-javascript).

```
function S4() {
  return (((1+Math.random())*0x10000)|0).toString(16).substring(1);
}
function guid() {
  return (S4()+S4()+"-"+S4()+"-"+S4()+"-"+S4()+"-"+S4()+S4()+S4());
}
var myID = 'static' + '-' + guid();
```

### Feature Sniff, Don't Browser Sniff

Does the client's browser support geolocation? Does the client's browser support web workers? HTML5 video? HTML5 audio? The answer used to be:

```
if ($.browser.msie) { // no it doesn't }
```

But things are rapidly changing. The latest version of IE is almost a modern browser, but as usual it's making front end development a pain. Earlier versions of IE were generally as equally sucky as their predecessors, so it enabled lazy JavaScript developers to simply detect `if (ie)` and execute some proprietary Microsoft slops syntax. Now IE9 has done away with these functions, but that old `if (ie)` chestnut is throwing a spanner in the works.

So what if you could detect support for individual features without sniffing the (unreliable and cloakable) user-agent?

If you answered *"that would be ghetto"*, then you are correct.

In steps Modernizr (http://www.modernizr.com), a JavaScript library developed in part by industry dream-boat Paul Irish. With wide adoption, tiny file-size and plenty of documentation (http://www.modernizr.com/docs/#s1): implementing it is a no-brainer. It creates a `Modernizr` object that contains the results of its detection tests, so checking feature support is as simple as the following:

```
// old way of detecting canvas support
if (!!document.createElement('canvas').getContext) { ... }

// with Modernizr
if (Modernizr.canvas) { ... }
```

### Readable Milliseconds

A handy way of writing milliseconds in a readable format. Great for beginners, but mostly a gimmick.

```
// is this 3, 30 or 300 seconds?
var timeout = 30000;

// an extra calculation, but easier to read and modify
var timeout = 30 * 1000;
```

## jQuery Specific

### Chain Like a Mad Dog

One of the best parts of jQuery is its function chaining. You've probably used it a bit, maybe a few simple calls one after another... but have you ever traversed the DOM like a mad dog? Take some time to familiarise yourself with the .end() (http://api.jquery.com/end/) function. It is critical for when you begin stepping up and down the DOM tree from your original selector.

```
  $(".quote")
    .hide()
    .find("a").text("Click here").bind("click",doStuff).end()
    .parent().removeClass().addClass("testimonial").draggable().end()
    .fadeIn("slow");
```

In the example above, the .end() (http://api.jquery.com/end/) function is used once we have finished doing things with a particular DOM object and want to traverse back up the DOM to the original object we called. We then load back up and dive back into the DOM.

### Using data-* Attributes

Those of you who have been writing JavaScript (and not jQuery) for a good length of time are most likely familiar with attributes. Setting them. Getting them. Abusing `rel` and `title` instead...

So when *isn't* HTML5 or jQuery coming the rescue? New specs allow the use of `data-` prefixes on HTML elements to indicate attributes which can hold data, and jQuery does an awesome job of converting the designated string into the correct JavaScript type. It's a beautiful partnership. Let's create a `DIV` with some data attributes.

```
<div id="test" data-is-bool="true" data-some-number="123"></div>
```

Now even though our values are wrapped in quotation marks, they won't be handled as strings:

```
typeof $("#test").data("isBool"); // boolean
typeof $("#test").data("someNumber"); // number
```

### Special Casing

It's also important to notice the lower casing required to get these snippets to work. Like many places in JavaScript, a preceding hyphen signifies camel casing of the next letter. The following camel casing of the HTML attribute **does not work** and the same JavaScript used above will return `undefined`.

**Does not work :(**

```
<div id="test" data-isBool="true" data-someNumber="123"></div>
```

**Does work :)**

```
<div id="test" data-is-bool="true" data-some-number="123"></div>
```

## Optimise Your Selectors

jQuery is pretty chill. It can do pretty much everything but make you coffee. One thing you have to be careful about is abusing the power that is the sizzleJS (http://sizzlejs.com/) selector engine. There are two strategies to overcome this: *caching the selector results* and *using efficient selectors*.

### Caching Selector Results

Do a costly DOM query every time you want to change something, or store a reference to the element? Pretty clear choice.

```
// before
$(".quote a").bind("click", doStuff); // DOM query eww

// now
$(".quote a").addClass("quoteLink"); // DOM query eww

// later
$(".quote a").fadeIn("slow"); // DOM query eww
```

Ignoring chaining, this is better:

```
// the only DOM query
var $quoteLinks = $(".quote a");

// before
$quoteLinks.bind("click", doStuff);

// now
$quoteLinks.addClass("quoteLink");

// later
$quoteLinks.fadeIn("slow");
```

### Using Efficient Selectors

So jQuery/sizzleJS can use CSS3 selectors like a boss, but what's the real cost? Behind the scenes the browser is hopefully using `document.querySelector()`, but there's also a fair chance it will be breaking down your selector string and querying the DOM manually.

```
// an ID search is the quickest possible query, then it just takes a list of the childNodes and matches the class
$("#quoteList").children(".quotes");

// looks for the "foo" class only in the pre-defined bar element
$(".foo", bar);
```

## A 'for' Loop is Always Quicker Than a 'each()' Loop

No matter what happens in the next few years of browser development, a native `for` loop will always be quicker than a jQuery `$.each()` loop. When you think of what jQuery really is (a library wrapped around native JS functions) you begin to realise that the native underlying JavaScript is always going to be quicker. It's a tradeoff of run speed versus authoring speed.

It is vital that a native `for` loop is always used for performance critical functions that could fire potentially hundreds of times per second. Examples include:

- Mouse movement
- Timer intervals
- Loops within loops

## Debugging

Even with the best of validators, inevitably browser quirks will cause issues. There are several invaluable tools which will help to refine code integrity and loading speed. It is important that you have all of these tools available to you, despite the browser you primarily use for development. We recommend developing for Firefox and Safari first, then Google Chrome and Opera, with additional tweaks via conditional comments just for Internet Explorer. The following is a list of helpful debuggers and speed analyzers...

- **Google Chrome**: Developer Tools (http://google.com/chrome/intl/en/webmasters-faq.html#tools)
- **Safari**: Web Inspector (http://developer.apple.com/safari/library/documentation/AppleApplications/Conceptual/Safari_Developer_Guide/UsingtheWebInspector/UsingtheWebInspector.html)
- **Firefox**: Firebug (http://getfirebug.com/), Page Speed (http://code.google.com/speed/page-speed/), YSlow (http://developer.yahoo.com/yslow/)
- **Opera**: Dragonfly (http://opera.com/dragonfly/)
- **Internet Explorer 6-7**: Developer Toolbar (http://microsoft.com/downloads/details.aspx?familyid=E59C3964-672D-4511-BB3E-2D5E1DB91038)
- **Internet Explorer 8-10**: Developer Tools (http://msdn.microsoft.com/en-us/library/dd565625(v=VS.85).aspx)

### Patterns for better JavaScript

- Writing Maintainable Code
- Single var Pattern
- Hoisting: A Problem with Scattered vars
- (Not) Augmenting Built-in Prototypes
- Avoiding Implied Typecasting
- Avoiding eval()
- Number Conversions with parseInt()
- Opening Brace Location
- Capitalizing Constructors
- Writing Comments
- Avoid void
- Avoid with Statement
- Avoid continue Statement
- Avoid Bitwise Operators if possible

Stoyan Stefanov covers these and more in detail here (http://net.tutsplus.com/tutorials/javascript-ajax/the-essentials-of-writing-high-quality-javascript/).

# Accessibility

- Section 508 Standards for intranet and internet information and applications (http://www.section508.gov/index.cfm?FuseAction=Content&ID=12#Web)
- W3C checklist of checkpoints for accessibility (http://www.w3.org/TR/WCAG10/full-checklist.html).
- The WCAG 1.0 Guidelines (http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/#Guidelines)

## Fallbacks for Middle Mouse Clicks

One of the most frustrating accessibility and usability flaws of the modern web stems from the remapping of hyperlink click functions. Elements that appear to be hyperlinks may have their single click functionality remapped via JavaScript, breaking middle mouse click (open in new tab) functionality. If they can be opened in a new tab, their href of a single hash sends you back to the same page.

A modern example of a popular website that is contributing to this problem is the Twitter web app. Middle mouse clicking of names or user avatars yields completely different results throughout the web app.

```
<!-- The old way, breaking the web -->
<a href="#"></a>

<!-- If you can't deliver a page on mouse click, it's not a hyperlink -->
<span class="link" role="link"></span>
```

Another alternative is the use of "hashbangs", that remap normal URLs to hash links and fetch pages via AJAX. Libraries that provide hashbang functionality should be able to display the page normally when middle mouse clicked, or load the content from that page into a designated area when clicked normally. But tread carefully, there are plenty of people who believe hashbangs are breaking the web (http://isolani.co.uk/blog/javascript/BreakingTheWebWithHashBangs).

## Use Microformats

Microformats are a way of making contact information machine readable. hCard classes (not vCard) are used to define the type of content contained within elements. These are then extracted or highlighted by the browser.

```
<span class="tel"> <span class="type">home</span>: <span class="value">+1.415.555.1212</span> </span>
```

If you were to navigate to a page that uses this, you would notice that a program like Skype will easily detect what numbers on the page are phone numbers. Mobile Safari does something similar on iOS devices.

For more information: (http://microformats.org/wiki/hcard)http://microformats.org/wiki/hcard (http://microformats.org/wiki/hcard)

## Images Need 'Alt' Text

The `<img>` tag requires `alt` text to both validate and meet accessibility guidelines. The text in the `alt` attribute should be descriptive of what the image shows, or is trying to achieve, unless of course the image is not critical.

If the image is of a list bullet or other trivial icons, it is recommended to simply leave the `alt` attribute empty, but still present. A screenreader will then ignore it, as opposed to having to read out "bullet" 20 times.

```
<img src="dog.gif" alt="Fido and I at the park!" /> <!-- good, descriptive -->
<img src="bullet.gif" alt="bullet" /> <!-- bad, as silly as it seems -->
<img src="bullet.gif" alt="" /> <!-- good -->
```

## Use Tables for Tabular Data Only

Tables should only ever be used for the presentation of tabular data. The only exception is when composing HTML email, in which a table is almost the only thing supported by soul crushing email clients.

For accessibility, table headers should always be presented using `<th>` elements. Remember to also set `cellpadding`, `cellspacing` and `border` values to `0` as these are more consistently controlled by CSS.

```
<table cellpadding="0" cellspacing="0" border="0">
  <thead>
    <tr> <th> Cell Header </th> </tr>
  </thead>
  <tbody>
    <tr> <td> Cell Item </td> </tr>
  </tbody>
</table>
```

# Performance

## Optimize Delivery of CSS and JavaScript

There are many optimizations that should be done (as a front-edn developer) for serving CSS and javascript in Production:

- Follow the Yahoo Performance Guidelines (http://developer.yahoo.com/performance/)
- Smush images using smush.it (http://developer.yahoo.com/yslow/smushit/). Also using YSlow (http://developer.yahoo.com/yslow/) can autosmush all your images for you.
- Avoid inline `<script>` blocks within the HTML. They block rendering and are quite devastating to page load time.
- CSS should be located in the `<head>` of the document, Javascript should be right before the `</body>` tag.
- Both CSS and JavaScript should be served minified. Most people use the YUI Compressor (http://developer.yahoo.com/yui/compressor/) for this. We use grunt's uglify (https://github.com/gruntjs/grunt-contrib-uglify).
- Both should be served using gzip on the wire (http://code.google.com/speed/page-speed/docs/payload.html#GzipCompression)
- CSS should be concatenated together. Obviously this can only be done for files that share the same media type (e.g. screen or print). The general goal here is to reduce the number of HTTP connections to dependencies during the loading of the page.
- Concatenation and minification typically occur during an automated build process, when packaging the code for deployment on stage or production. Many use tools like Ant (http://ant.apache.org/) or Maven (http://maven.apache.org/) for this. We use grunt (http://gruntjs.com/) :metal:

## Optimize JavaScript execution

During a page load, there is typically a lot of script waiting to execute, but you can prioritize it. This order prioritizes based on user response:

1. Script that changes the visible nature of the page needs to fire absolutely first. This includes any font adjustment, box layout, or IE6 pngfixes.
2. Page content initialization
3. Page header, topnav and footer initialization
4. Attaching event handlers
5. Google Analytics, Omniture, Doubleclick, and other 3rd party scripts

## Leverage CSS Sprites and Font Icons

### CSS Sprites

CSS Sprites basically take a number of image assets and merge them together into a singular image file. Each part of it is revealed using CSS background-position. Some typical CSS would look like:

```
a.expandbox {
  display:block;
  width: 75px;
  height: 15px;
  text-indent: -999px;
  overflow:hidden;
  background: url(../img/sprite.png) no-repeat -50px -5px;
}
```

It's quite common to leverage sprites for :hover states. In that case you'll see something like:

```
a.expandbox:hover { background-position: -50px -20px; }
```

Using sprites reduces total page weight and reduces HTTP connections which speeds up page load. More on the general technique and overview at css-tricks.com (http://css-tricks.com/css-sprites/). It is just an overall best practice to make use of CSS sprite images whenever you can.

Many developers use a vertically-oriented sprite in addition to the primary sprite. This vertical sprite will be **less than or equal to 100px wide** (and tall) and contain icons that are typically placed next to text, such as list item bullets or call to action links and buttons. Yahoo uses a few such as this one (http://l.yimg.com/a/i/ww/sp/pa-icons3.gif).

The one consideration is to not make sprites too large, something over 1000px in either direction will end up using a sizeable amount of memory. Read more on when to use sprites and memory usage here (http://blog.vlad1.com/2009/06/22/to-sprite-or-not-to-sprite/), and for more general tips and techniques on creating sprites check out the Mozilla Dev Blog (http://blog.mozilla.com/webdev/2009/03/27/css-spriting-tips/).

Some general tips and techniques (http://blog.mozilla.com/webdev/2009/03/27/css-spriting-tips/) in sprite creation can be found on the Mozilla Dev Blog (http://blog.mozilla.com/webdev/2009/03/27/css-spriting-tips/).

### Font Icons

Read the HTML For Icon Font usage (http://css-tricks.com/html-for-icon-font-usage/) article.

We use

## Image formats

There are four main image formats that should be used, detailed here:

1. **JPEG.** - This will cover all photography-based images, such as homepage and category page promo images, or anything with a very high color count.
2. **PNG24.** This format, easily accessible in Photoshop, outputs high-color count imagery and fully supports graded opacity by pixel. Relatively, it's quite a heavy format as far as kilobyte weight.
3. **PNG8.** - A surprising diversity of color can be captured inside 256 colors, so it's worth trying PNG before heading JPG. PNG also is a lot more compressible than GIF (using tools like pngcrush and pngquant). This format allows graded opacity in nearly all browsers.

   Photoshop cannot output these semi-opaque files correctly but Fireworks can. More detail here: (http://www.sitepoint.com/png8-the-clear-winner/)http://www.sitepoint.com/png8-the-clear-winner/ (http://www.sitepoint.com/png8-the-clear-winner/)
4. **Transparent GIF 89a.** - GIF 89a offers the flexibility of transparency and wide browser support, but the constraints of no graded opacity nor a color depth above 256. In our experience, color depths of 64 still provide very good quality thumbnails, and keep the file size comparably very small.

   All low-color count imagery such as icons or thematic graphics should be done in PNG8, as it's the most size-efficient of these four. PNG8 is our primary recommendation for most site graphics.

Detailed information on the PNG format, browser support, and the pros and cons of each can be found in this excellent article (http://calendar.perfplanet.com/2010/png-that-works/).

For further optimization all of these formats, taking them through Yahoo's Smush.It (http://developer.yahoo.com/yslow/smushit/) will reveal how they can be smaller.

## Measure performance during QA

QA teams should also prioritize performance related tickets alongside visual, functional, and usability issues. Developers and QA should determine how that priority will be assigned. During QA, the success metrics should be tested regularly.

**Tools to test with**

- YSlow (http://developer.yahoo.com/yslow/)
- Page Speed (http://code.google.com/speed/page-speed/)
- Hammerhead (http://stevesouders.com/hammerhead/)
- MSFast (http://msfast.myspace.com/)
- PageTest (http://www.webpagetest.org/)

**When performance goals aren't met, there are three options:**

1. *Redevelop the code* - profile, discover bottlenecks, refactor code, optimize to target faster execution in the browser
2. *Drop the feature* - turn it off for slower browsers only
3. *Redesign approach of the UI* - perhaps the design could use a tweak which would bypass the issue entirely

With this approach, we think all parties have a better chance of having aligned expectations heading in as well as a more sensible workflow in dealing with performance challenges.

# Browser Testing and Support

## What we support

**DIGITAS France supports all recent browsers**.

We will strive to support any other mission critical browsers mandated by the client (IE 7 8 9, Opera, Safari 3 on PC, etc), though we cannot guarantee all functionality is possible.

## How we test

Using browserstack.com (http://browserstack.com)!

## Browser Resolution

Along with catering to browsers, developers must stay conscious of the screen resolutions of their audience. As monitor screens get larger, the breadth of resolutions grows as well. Below are some guidelines to help you in working with resolutions.

**1024px resolution**

- Fold estimated at 570px.
- Optimal width: 960px - Has comfortable padding on both sides, is divisible by many numbers, and also plays well with IAB ad standard widths (http://www.iab.net/iab_products_and_industry_services/1421/1443/1452)
- Larger width: 970px - Still has some padding on both sides in most browsers. The math plays well with the Golden Ratio (http://en.wikipedia.org/wiki/Golden_ratio)
- Maximum width: 996px - Without incurring any horizontal scrollbars in the major browsers. Based on the research here (http://www.nealgrosskopf.com/tech/thread.php?pid=43) the maximum is 1003 px wide if you don't want a horizontal scrollbar.

**Current stats on window sizes**

- Authentic Boredom - Optimal width for 1024px resolution? (http://www.cameronmoll.com/archives/001220.html)
- Market share for browsers, operating systems and search engines (http://marketshare.hitslink.com/report.aspx?qprid=17&qpmr=100&qpdt=1&qpct=3&qptimeframe=M)
- Global Web Stats (http://www.w3counter.com/globalstats.php)

**System resolution is not, however, the same as browser size**

- Browser size does matter - Actual numbers | mentalized (http://mentalized.net/journal/2006/10/24/browser_size_does_matter_actual_numbers/)
- What size do i need to support | baekdal (http://www.baekdal.com/reports/actual-browser-sizes/abs-sizes/)
- Poll results: 50.4% of respondents maximise windows (http://www.456bereastreet.com/archive/200704/poll_results_504_of_respondents_maximise_windows/)
- Screen Resolution and Page Layout (http://www.useit.com/alertbox/screen_resolution.html)

# Search Engine Optimization

## Be aware of SEO best practices

- Section 508 (http://www.section508.gov)
- WCAG 2.0 (http://www.w3.org/TR/WCAG20/)
- Print CSS best practices (http://24ways.org/2007/back-to-the-future-of-print)

### Indexability

We must use semantic markup that's readable and logical when JavaScript and CSS are off. All page content must be in HTML; we don't want to use iframes or JavaScript for loading initial indexable content.

**For accessibility or SEO; All links should be to HTML destinations.**

```
<a href="/shelf/jordan/page/2">
```

**Instead of**

```
<a href="javascript:loadPage(2);">
```

This lets the page get indexed correctly by search engines as well as allows users to open in new tabs and windows.

### URL Optimization

The title tag should feature target keywords for the unique page. The titles should be unique to each page. Headings (h1,h2,etc) should form an outline of the document and represent the most important keywords for that page. URLs should be human-readable with primary target keywords present in them:

```
http://domain.com/mens-shoes/basketball/jordan/jordan-mens-ajf-6-basketball-shoe/
```

**vs**

```
http://domain.com/ecomm.cfm?view=prod&prodId=23425
```

### Flash and Image Replacement

Always use backup HTML content for flash. All promo images should use CSS-based image replacement instead of alt tags.

Fallback non-flash version:

```
<a href="/nike/morethanagame/" id="morethan">
  <h4>Nike: More Than A Game</h4>
  <h5>Experience the movement and view apparel</h5>
</a>
```

```
#morethan {
  background:url(/promos/nikegame.jpg) no-repeat;
  width: 200px;
  height: 100px;
  text-indent: -999px;
  overflow:hidden;
  display:block;
}
```

# Git

## Must Have

1. Git-scm.com (http://git-scm.com/book/) - Your number one `git` reference.
2. Git-convenience (https://github.com/jakearchibald/git-convenience) - Basic `git` shortcuts ( `gc` for `git commit` , `gs` for `git status` , etc..).
3. Git-extras (https://github.com/visionmedia/git-extras) - Advanced `git` shortcuts ( `git-create-branch` to create a local and a remote branch, etc..).

## Key branches

- `master` is the latest, deployed version. The master branch needs to always be clean, lean and stable. **Very stable**.. If we want to show or deploy the project to someone, we show them the master branch or one of its tags.
- `gh-pages` is the hosted built version of the app (not to be used for pull requests).
- `*-wip` is the official work in progress branch for the next release.
  - `yourname-wip` is your general development branch.
  - `featurename-wip` are feature specific branches.

## Commits

- **Commits are per 1 subject.** As an example, if you fix a bug and along the way you made some function a little faster, then those should be 2 different commits.
- **Commit often**, with each commit only changing a file or two, rather than changing a ton of files and commiting it all at once. *This will make it easier to revert a commit if it introduces a bug. It also makes it easier to merge, rebase and offer feedback.*
- You can use `git add -p` to interactively select which changes get staged before you commit.
- When you then commit, you need to provide a one line short commit message and **then** a paragraph or two explaining the change you made in details.
- There should be one blank line between the short and long description.
- If it's a small (obvious and trivial), 1 or 2 line change, then only the short description is sufficient.
- In short: Make sure your commit messages are clear and descriptive. *Good*: `Added initial draft of hello world tutorial` *Bad*: `working on docs`
- It takes time to write a commit message. You **must** take your time to write a clear message.
- **After every commit, the whole project must at least build and run.**

## Pull Requests

1. Create a new branch, please don't work in your `master` branch directly.
2. Fix stuff.
3. Run `grunt` to see if it still builds. Repeat steps 2-3 until done.
4. Update the documentation to reflect any changes.
5. Push to your fork and submit a pull request. (Try to submit pull requests against the latest `*-wip` branch for easier merging)

## Reporting Issues

We only accept issues that are bug reports. Bugs must be isolated and reproducible problems that we can fix within the Application core. Please read the following guidelines before opening any issue.

1. **Search for existing issues.** We get a lot of duplicate issues, and you'd help us out a lot by first checking if someone else has reported the same issue. Moreover, the issue may have already been resolved with a fix available.
2. **Include a test case.** Supply the page name and the concerned zone.
3. **Share as much information as possible.** Include operating system and version, browser and version. Also include steps to reproduce the bug.

# Code Reviews

A code review is the cornerstone of the formal process for ensuring the quality of user experiences developed by the Creative Technology team. Simply put, we encourage conducting code reviews to keep our tools sharp and clean.

**Why should I have a code review?**

Code reviews are a strategic investment of time to mitigate risk on a project.

**Code reviews increase the overall level of knowledge across projects**

Since code reviews involve members from within and outside a project team, techniques and best practices are easily shared throughout the entire Creative Technology team.

**Code reviews eliminate bugs before they are reproduced from a few templates into multiple pages**

Ideally, code reviews are conducted early in the development process, before full production of pages begins. When templates are reviewed by the team and run through multiple validation tools and browsers, bugs can and will appear. This is the ideal time for bugs to be fixed.

**Code reviews give a set of fresh eyes an opportunity to spot issues with code**

Reviewers from outside a project team can spot issues with code more easily than authors of markup, who have been working with code for a longer amount of time.

# Node.js

When you code, sometimes you find yourself in need of a server. Back in 2000 we used APACHE, today we use Node.js (http://nodejs.org/).

- Node.js is an event-driven, server-side javascript environment.
- Node runs javascript using the V8 engine developed by Google for use in their Chrome web browser. Leveraging V8 allows Node to provide a server-side runtime environment that compiles and executes javascript at lightning speeds. The major speed increase is due to the fact that V8 compiles javascript into native machine code, instead of interpreting it or executing it as bytecode.
- Node is open source, and cross-platform, running on Mac OSX, Windows, and Linux.

# Grunt

Grunt (http://gruntjs.com/) is a javascript task runner.

## Why use a task runner?

- In one word: automation.
- The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes.
- After you've configured it, a task runner can do most of that mundane work for you—and your team—with basically zero effort.

## Why use grunt?

- We use it to automate our build process.
- For instance, while you are writing LESS code, grunt watches your files and compiles them to CSS. The same goes for Jade, Coffescript and any other templating/preprocessor you can think of.
- Other than compiling languages, grunt checks our javascript agaisnt jshint (http://www.jshint.com/) recommendations, concatenate and uglify them.
- We also use grunt to deploy the compiled version of our app on github and to generate documentations.
- It also optimizes our images, creates our sprites, generates our font icons, etc….
- The Grunt ecosystem is huge and it's growing every day.
- With literally hundreds of plugins to choose from, you can use Grunt to automate just about anything with a minimum of effort.
- If someone hasn't already built what you need, authoring and publishing your own Grunt plugin to npm is a breeze.

## Sample Gruntfile

```
module.exports = function (grunt) {

  grunt.initConfig({

    pkg: grunt.file.readJSON('package.json'),

    concat: {
      options: {
        separator: ';'
      },
      dist: {
        src: ['src/**/*.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },

    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("dd-mm-yyyy") %> */\n'
      },
      dist: {
        files: {
          'dist/<%= pkg.name %>.min.js': ['<%= concat.dist.dest %>']
        }
      }
    },

    qunit: {
      files: ['test/**/*.html']
    },

    jshint: {
      files: ['gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        // options here to override JSHint defaults
        globals: {
          jQuery: true,
          console: true,
          module: true,
          document: true
        }
      }
    },

    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint', 'qunit']
    }

  });

  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-qunit');
  grunt.loadNpmTasks('grunt-contrib-watch');
  grunt.loadNpmTasks('grunt-contrib-concat');

  // this would be run by typing "grunt test" on the command line
  grunt.registerTask('test', ['jshint', 'qunit']);

  // the default task can be run just by typing "grunt" on the command line
  grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);
};
```

# Code Editors

A great code editor can spark productivity in exceptional ways. Many developers prefer rudimentary text editors, others prefer powerful integraded development environments (IDEs). What follows is a general listing of some of the more well-known tools, it would be impossible to list them all.

- **Sublime Text**: A relatively new editor, Sublime Text is a new approach in editors. "Open Anything" searches through file names and file contents, with remarkable efficiency. Incredibly powerful selection controls allow editing text in multiple locations at once and the "Minimap" gives you a bird's eye view of the open file so you can find your place easily. Actively being developed, new features are being added and the community around the editor is rapidly expanding. Macros, auto-complete, snippets, build tools, the list of features goes on and on. Supports Linux and Mac starting with version 2. Sublime Text (http://www.sublimetext.com/). [Linux, Mac, Windows]
- **Aptana**: Aptana Studio is a powerful, free and open source Ajax development environment which runs stand-alone or within Eclipse. Aptana Studio offers tooling for Ajax including HTML, CSS, DOM, and JavaScript editing and debugging, plus support via additional free plugins for PHP, Ruby on Rails, Adobe AIR, Apple iPhone development. It also features full SVN repository integration for committing, branching, tagging, merging and repository browsing. Aptana (http://aptana.com/). [Linux, Mac, Windows]
- **Geany**: Geany is a text editor using the GTK2 toolkit with basic features of an integrated development environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. It supports many filetypes and has some nice features. Geany (http://www.geany.org/). [Linux, Mac, Windows]
- **Notepad ++**: Notepad++ is a free (free as in "free speech", but also as in "free beer") source code editor and Notepad replacement, which supports several programming languages, running under the MS Windows environment. Notepad ++ (http://notepad-plus.sourceforge.net/uk/about.php). [Windows]
- **e TextEditor**: E is a new text editor for Windows, with powerful editing features and quite a few unique abilities. It makes manipulating text fast and easy, and lets you focus on your writing by automating all the manual work. You can extend it in any language, and by supporting TextMate bundles, it allows you to tap into a huge and active community. e TextEditor (http://www.e-texteditor.com/). [Windows]
- **Edit Plus**: EditPlus is a text editor, HTML editor and programmers editor for Windows. While it can serve as a good Notepad replacement, it also offers many powerful features for Web page authors and programmers. EditPlus (http://www.editplus.com/). [Windows]

- **Homesite**: HomeSite 5.5 provides a lean, code-only editor for web development. Advanced coding features enable you to instantly create and modify HTML, CFML, JSP, and XHTML tags, while enhanced productivity tools allow you to validate, reuse, navigate, and format code more easily. Configure Adobe (formerly Macromedia) HomeSite to fit your needs by extending its functionality and customizing the interface. Homesite (http://www.adobe.com/products/homesite/). [Windows]
- **TextMate**: TextMate claims to be the "Missing Editor" for Mac OS X. A general purpose editor with a sparse interface, the real power is in it's extensibility. Features column selections, recordable macros, snippets, auto-pairing of brackets and other characters, clipboard history, code folding, tab-triggers, tabbed placeholders and mirror typing. And that's just for starters. Anything that can be done via scripts through the Mac command line can be done through custom commands, allowing an extremely high degree of customization and expansion of the feature set. TextMate's "bundle" format has been adapted by many other code editors including the aforementioned e TextEditor. TextMate (http://www.macromates.com). [Mac]
- **Espresso**: Espresso was created by the same fellow that created the innovative CSSEdit CSS editor. Espresso features syntax highlighting, code folding, code completion, document outliner/navigators, projects, powerful find features, and built-in file transfer publishing capabilities. Finally, it has a powerful "Sugar" feature set which allows the creation of custom commands and plugins. Espresso (http://macrabbit.com/espresso/). [Mac]
- **BBEdit**: BBEdit is the grand-daddy of Mac code editors for web development. Features syntax highlighting, exceptionally powerful text manipulation tools, multi-file searches, a scriptable API, text clippings, and extensive Mac Automator features. BBEdit (http://www.barebones.com/products/bbedit/). [Mac]
- **TextWrangler**: The free "little brother" of BBEdit, it is a powerful raw text editor with a massive text manipulation feature set. Searches, regular expressions, text transformations, syntax highlighting and code navigation tools for a variety of different language environments. TextWrangler (http://www.barebones.com/products/textwrangler/). [Mac]
- **Coda**: Coda from Panic software is a powerful IDE with code editing, terminal, remote file management, and help documentation all built into one UI. Aiming to be a one stop shop for your web development workflow, it also features SVN integration and a new plug-in builder with powerful scripting support and TextMate bundle importing. Finally, code clips and live multi-user editing are also supported. Coda (http://www.panic.com/coda/). [Mac]
- **UltraEdit**: Another editor that's been around for ages, this is an immensely robust and powerful text editor, able to open files limited only by the amount of memory on your computer. The feature list is virtually too much to list, with a massive list of text manipulation features, project support, powerful search and replace, hex editing, function lists, a massive list of languages supported (600+) remote file ftp, telnet, ssh, file comparison, scriptable macros, tools and compiler support, and much, much more. UltraEdit (http://www.ultraedit.com/products/ultraedit/). [Linux, Mac, Windows]
- **Vim**: If you have to ask, it's probably not for you. Vim (http://www.vim.org/) [Linux, Mac, Windows]

# Tools

## Validators

- W3C CSS Validation Service (http://jigsaw.w3.org/css-validator/)
- HTML Validation firefox extension (https://addons.mozilla.org/en-US/firefox/addon/249)
- CSS validator (http://jigsaw.w3.org/css-validator/)
- Accessibility - Bobby Validation Service (http://bobby.watchfire.com/)
- Tests individual pages for accessibility against either the W3C or Section 508 standards.
- Accessibility - Cynthia Says Portal (http://www.cynthiasays.com/)
- Similar to Bobby, tests individual pages for accessibility against either the W3C or Section 508 standards.

## Tools

- Browsershots (http://browsershots.org/) makes screenshots of your web design in different operating systems and browsers
- Browserling (https://browserling.com/) is an interactive cross-browser testing site. Fully interactive sessions, not static screenshots.
- Accessibility - Evaluation, Repair, and Transformation Tools for Web Content Accessibility (http://www.w3.org/WAI/ER/existingtools.html) - lists validation tools for validating, testing and fixing web content to make it more accessible.

### Google Chrome Extensions

- **Developer Tools**: Not actually an extension for Chrome, but built right in (shares much with Safari's Web Inspector, both being derived from WebKit.) This suite of tools features a DOM inspector, basic JavaScript debugger, profiling tools, network loading inspector and timelines, page resources inspectors, and more. Developer Tools (http://code.google.com/chrome/devtools/docs/overview.html).
- **code cola**: A pop-up panel with CSS editing tools for examining and modifying the styles on a given page. code cola (https://chrome.google.com/webstore/detail/lomkpheldlbkkfiifcbfifipaofnmnkn).
- **Firebug Lite for Google Chrome**: You really don't need to install an extension to use Firebug Lite with Chrome, though the extension is nice because it enables one-click application of the Firebug Lite script to any page you are working with. Not the full Firebug feature set, but close. Firebug Lite (https://chrome.google.com/webstore/detail/bmagokdooijbeehmkpknfglimnifench).
- **HTML5 Outliner**: The HTML5 Outliner adds a pop-up with a generated HTML5 outline of the current page's header hierarchy. Helps for checking your pages' organization against the new HTML5 header outlining algorithms. HTML5 Outliner (https://chrome.google.com/webstore/detail/afoibpobokebhgfnknfndkgemglggomo).
- **Pendule**: Nice set of tools for showing data about and interacting with the current web page. Very similar to the original web developer toolbar for Firefox (http://chrispederick.com/work/web-developer/). Pendule (https://chrome.google.com/webstore/detail/gbkffbkamcejhkcaocmkdeiiccpmjfdi).
- **Web Developer for Chrome**: Chris Pederick, the original developer of the original Web Developer toolbar for Firefox has ported the majority of it over to Chrome. There you have it. Web Developer (http://chrispederick.com/work/web-developer/).

### Firefox Plugins

- **FireFTP**: FireFTP is a free, secure, cross-platform FTP client for Mozilla Firefox which provides easy and intuitive access to FTP servers. FireFTP (http://fireftp.mozdev.org/).
- **Firebug**: Firebug integrates with Firefox to put a wealth of development tools at your fingertips while you browse. You can edit, debug, and monitor CSS, HTML, and JavaScript live in any web page. Firebug (https://addons.mozilla.org/en-US/firefox/addon/1843).

- **Firequery**: FireQuery is a collection of Firebug enhancements for jQuery integrated into Firebug. Firequery (https://addons.mozilla.org/en-us/firefox/addon/firequery/).
- **Firecookie**: Firecookie adds cookie viewing, editing, and deletion to Firebug. Firecookie (https://addons.mozilla.org/en-US/firefox/addon/firecookie/).
- **CSS Usage**: CSS Coverage is an extension for Firebug which allows you to scan multiple pages of your site to see which CSS rules are actually used in your site. CSS Usage (https://addons.mozilla.org/en-us/firefox/addon/css-usage/).
- **Greasemonkey**: Allows you to customize the way a web page displays using small bits of JavaScript. GreaseMonkey (https://addons.mozilla.org/en-US/firefox/addon/748).
- **Web Developer Toolbar**: Adds a menu and a toolbar with various web developer tools. Web Developer Toolbar (https://addons.mozilla.org/en-US/firefox/addon/60).
- **JSView**: Adds an item in the status bar that displays all external JavaScript and CSS files loaded on a given page. Allows you to click on and view the files and things like their URLs. Great way to pull file URLs to put into Charles for remote debugging. JSView (https://addons.mozilla.org/en-US/firefox/addon/2076).
- **Live HTTP headers**: When running, captures all HTTP traffic from the browser, which enables you to see what files are being requested as well as information about the requests and server responses. Live HTTP Headers (https://addons.mozilla.org/en-US/firefox/addon/3829).
- **Quick Locale Switcher**: A tremendous help when doing internationalization, Quick Locale Switcher allows you to change the locale sent along in the browser's user-agent HTTP header, telling servers to display content for you in other locales. Quick Locale Switcher (https://addons.mozilla.org/en-US/firefox/addon/1333).
- **Screengrab**: Screengrab sits in the Firefox status bar, allowing you to capture and copy or save screen shots of everything from selections of a web page to the entire page, even parts displayed "below the fold." Screengrab (https://addons.mozilla.org/en-US/firefox/addon/1146).
- **Total Validator**: Enables one-click access to sending your page through a markup validator. No better way to quickly check for missing or mismatched tags! Also available as a standalone application. Total Validator (http://www.totalvalidator.com/tool/extension.html).

### Opera Extensions

Dragonfly (http://www.opera.com/dragonfly/) is Opera's developer tool similar to Firebug.

### IE Plugins

**CompanionJS, DebugBar, IE8 Dev tools**.

### Charles Proxy

Charles (http://www.charlesproxy.com/) watches all requests and can tell you a lot of information about them. Also supremely useful is **Map Local** which lets you use a local file in place of a given URL (good for replacing a minified js with a full one).

### Fiddler

From the site: Fiddler (http://www.fiddler2.com/fiddler2/)is a Web Debugging Proxy which logs all HTTP(S) traffic between your computer and the Internet. Fiddler allows you to inspect traffic, set breakpoints, and "fiddle" with incoming or outgoing data. Fiddler includes a powerful event-based scripting subsystem, and can be extended using any .NET language.

### Speedlimit

Speedlimit (http://mschrag.github.com/) is a Leopard (works in snow leopard) preference pane for limiting your network bandwidth to one of a couple different speeds—768k DSL, Edge, 3G, and Dialup. Good for testing your lowest supported speeds or when you want to know how your app will function in real world speeds.

### Tutorials & Tools:

- CSS Cheatsheet (http://lesliefranke.com/files/reference/csscheatsheet.html)
- CSS Links (http://www.dezwozhere.com/links.html)
- Clean CSS (http://www.cleancss.com/)
- Fluid Grids Code Generator (http://csswizardry.com/fluid-grids/)

### Icons

- Famfamfam silk icons (http://www.famfamfam.com/lab/icons/silk/)
- Sweetie (http://sublink.ca/icons/)
- Paul's bookmarks for more icons (http://delicious.com/paul.irish/icons)
- Fugue Icons - 3,100 icons in PNG format. (http://p.yusukekamiyamane.com/)

## Resources

- ECMA 262 5th Edition December 2009 (http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf)
- Opera's The Web Standards Curriculum (http://dev.opera.com/articles/view/1-introduction-to-the-web-standards-cur/#toc) - basic articles about building with web standards.
- 10 Principles of the CSS Masters (http://nettuts.com/html-css-techniques/10-principles-of-the-css-masters/) - all of this is smart advice.
- Google (http://www.google.com) - when in doubt, google it.
- Too lazy? (http://lmgtfy.com/?q=front-end+development+coding+standards&l=1) - let me google that for you then...
- Yahoo's Exceptional Performance (http://developer.yahoo.com/performance/)team has maintained one of the best summaries of performance advice.
- Google now has a Speed site (http://code.google.com/speed/page-speed/docs/rules_intro.html) with excellent detail. (Click All best practices).
- There are many more excellent resources on page optimization (http://delicious.com/paul.irish/optimization) and javascript performance (http://delicious.com/paul.irish/performance).
- Mozilla's coding standards (http://wiki.mozilla.org/WebDev:FrontendCodeStandards).
- Nokia's JavaScript Performance Best Practices (http://wiki.forum.nokia.com/index.php/JavaScript_Performance_Best_Practices).

# Bookmarks

## Guides

- Hack Design (http://hackdesign.org/courses/)
- Designer School (http://designer-school.com/)
- TheExpressiveWeb (http://beta.theexpressiveweb.com/)
- Talks To Help You Become A Better Front-End Engineer In 2013 (http://www.smashingmagazine.com/2012/12/22/talks-to-help-you-become-a-better-front-end-engineer-in-2013/)
- Web Development Teaching Materials (http://www.teaching-materials.org/)
- Learn HTML5, CSS3, and Responsive WebSite Design in One Go (http://javascriptissexy.com/learn-html5-css3-and-responsive-web-site-design-in-one-go/)
- Codeacademy (http://codeacademy.com) Good source for beginners
- Codeschool (http://www.codeschool.com/) - Check out tryruby, trygit and tryjquery courses. Also have many useful courses.

## Architecture

- BEM (http://bem.info/method/): Methodology aimed at achieving fast to develop long-lived projects, team scalability, and code reuse.
- Atomic Design (http://bradfrostweb.com/blog/post/atomic-web-design/)
  - GitHub (https://github.com/bradfrost/patternlab)
  - Video (http://www.besquare.me/session/atomic-design/) + Slides (http://de.slideshare.net/bradfrostweb/atomic-design)
  - Atomic Design: Some Thoughts and One Example (http://15four.com/2013/05/29/atomic-design-some-thoughts-and-one-example/)
  - Atomic Design Makes Me Feel Like a Chemist (http://notebookandpenguin.com/atomic-design-makes-me-feel-like-a-chemist/)
- Polymer Project (http://www.polymer-project.org/): Polymer is a new type of library for the web, built on top of Web Components, and designed to leverage the evolving web platform on modern browsers.
  - Video: Web Components: A Tectonic Shift for Web Development (https://developers.google.com/events/io/sessions/318907648) + Slides (http://www.webcomponentsshift.com/)
  - Video: Web Components in Action (https://developers.google.com/events/io/sessions/324149970)
- Aura (http://aurajs.github.io/aura/) is an event-driven architecture for developing scalable applications using reusable widgets.
- Hydra (http://tcorral.github.io/Hydra.js/) is an easy-to-use framework that provides you with the necessary tools to create scalable applications using modules and widgets.
- Terrific.js (http://terrifically.org/) provides you a Scalable Javascript Architecture, that helps you to modularize your jQuery/Zepto Code in a very intuitive and natural way
- Patterns For Large-Scale JavaScript Application Architecture (http://addyosmani.com/largescalejavascript/)
- Video: Nicholas Zakas: Scalable JavaScript Application Architecture (http://www.youtube.com/watch?v=vXjVFPosQHw)
- Book: Learning JavaScript Design Patterns (http://www.addyosmani.com/resources/essentialjsdesignpatterns/book/)
- Book: Single page apps in depth (http://singlepageappbook.com/index.html)
- Book: Scalable and Modular Architecture for CSS (http://smacss.com/book/)
- jQuery Application Architecture Chart (http://addyosmani.com/resources/toolschart/chart.pdf)
- How To Manage Large jQuery Apps (http://www.slideshare.net/SlexAxton/how-to-manage-large-jquery-apps)
- Comparison between different Observer Pattern implementations (https://github.com/millermedeiros/js-signals/wiki/Comparison-between-different-Observer-Pattern-implementations)

## Workflow

- Video: Javascript Development Workflow of 2013 by Paul Irish (http://www.youtube.com/watch?v=f7AU2Ozu8eo) + Slides (http://dl.dropboxusercontent.com/u/39519/talks/fluent/index.html)
- Yeoman (http://yeoman.io/) is a robust and opinionated set of tools, libraries, and a workflow that can help developers quickly build beautiful, compelling web apps.
- Grunt (http://gruntjs.com/) is a task-based command line build tool for JavaScript projects.
  - Web development is getting complex. Let's go shopping. (http://ruudud.github.com/2012/12/22/grunt/)
  - GruntStart (https://github.com/tsvensen/gruntstart): A Grunt-enabled head-start with the H5BP, jQuery, Modernizr, and Respond. The building blocks to quickly get started with Grunt to create an optimized website.
  - A beginner's guide to Grunt (http://blog.mattbailey.co/post/45519082789/a-beginners-guide-to-grunt)
  - Grunt - Synchronised Testing Between Browsers/Devices (http://blog.mattbailey.co/post/50337824984/grunt-synchronised-testing-between-browsers-devices)
- Brunch (http://brunch.io/) is an assembler for HTML5 applications. It's agnostic to frameworks, libraries, programming, stylesheet & templating languages and backend technology.
- Front-end Process - Flat Builds and Automation
  - Introduction (http://www.gpmd.co.uk/blog/front-end-process-flat-builds-and-automation-part-1-introduction/)
  - Environment Setup & Yeoman (http://www.gpmd.co.uk/blog/front-end-process-flat-builds-and-automation-part-2-environment-setup-and-yeoman/)
  - Grunt Tasks (http://www.gpmd.co.uk/blog/front-end-process-flat-builds-and-automation-part-3-grunt-tasks/)
  - CSS Framework (Inuit) (http://www.gpmd.co.uk/blog/front-end-process-flat-builds-and-automation-part-4-css-framework/)

- CSSCSS (http://zmoazeni.github.io/csscss/): A CSS redundancy analyzer that analyzes redundancy.
- Helium (https://github.com/geuis/helium-css): JavaScript tool to scan your site and show unused CSS.
- JSLint (http://jslint.com/) for detecting errors or problems by static analysis of JavaScript programs.
- JSHint (http://jshint.com/) for more flexible static analysis of JavaScript programs.
- JSLint Error Explanations (http://jslinterrors.com) for explanations of the warnings given by JSLint and JSHint.
- ImageOptim-CLI (https://github.com/JamieMason/ImageOptim-CLI): Make lossless optimisation of images part of your automated build process.
- jpegoptim (https://github.com/tjko/jpegoptim): Utility to optimize/compress JPEG files.
- Pngcrush (http://pmt.sourceforge.net/pngcrush/) is an optimizer for PNG (Portable Network Graphics) files.
- Glue (https://github.com/jorgebastida/glue) is a simple command line tool to generate CSS sprites.
- Roots (http://roots.cx/) A light, super fast, and intuitive build system meant for rapid advanced front end development.
- Sparky.js (http://sparkyjs.com/) is a client-side application scaffold which helps those who want to have organized structure in their app, but don't want to subscribe to a particular client-side MVC framework.
- Mimosa (http://mimosajs.com) is a batteries included web development workflow tool that will get you coding in seconds rather than hunting down plugins and wrangling config for hours.
- Automaton (http://indigounited.com/automaton/): Task automation tool built in JavaScript.
- Cartero (https://github.com/rotundasoftware/cartero): A modular client side asset manager.
- Mod (https://github.com/modulejs/modjs) is a task-based workflow tooling for web, it help developers quickly build robust and high-performance web applications.
- Wraith (https://github.com/BBC-News/wraith) is a screenshot comparison tool.

## Weekly & Daily Resources

- HTML5 Weekly (http://html5weekly.com/)
- HTML5 Bookmarks (http://html5bookmarks.com/)
- CSS Weekly (http://css-weekly.com/)
- JavaScript Weekly (http://javascriptweekly.com/)
- Sidebar.io (http://sidebar.io/) The 5 best design links, everyday
- Web Design Weekly (http://web-design-weekly.com/)
- Web Development Reading List (http://wdrl.helloanselm.com/)
- Webdev Newsletter (http://www.d.umn.edu/itss/training/online/webdesign/webdev_listserv.html)
- Open Web Platform Daily Digest (http://webplatformdaily.org)
- Responsive Design Newsletter (http://responsivedesignweekly.com/)
- echo.js (http://www.echojs.com/): Reddit-like Website for JavaScript Resources
- Web Tools Weekly (http://webtoolsweekly.com/)
- A Drip of JavaScript (http://designpepper.com/a-drip-of-javascript)

## Programming & Markup Languages

### CSS

- Understanding border-image (http://css-tricks.com/understanding-border-image/)
- What No One Told You About Z-Index (http://philipwalton.com/articles/what-no-one-told-you-about-z-index/)
- Principles of writing consistent, idiomatic CSS (https://github.com/necolas/idiomatic-css)
- Github CSS Styleguide (https://github.com/styleguide/css)
- Wordpress CSS Coding Standards (http://make.wordpress.org/core/handbook/coding-standards/css/)
- Wizardry CSS Guidelines (https://github.com/csswizardry/CSS-Guidelines)

### JavaScript

- Book: Eloquent JavaScript (http://eloquentjavascript.net/contents.html)
- JavaScript Study Guide (http://shiflett.org/blog/2012/apr/javascript-study-guide)
- Javascript Closures (http://jibbering.com/faq/notes/closures/)
- OOP In JavaScript: What You NEED to Know (http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/)
- Combinator Recipes for Working With Objects in JavaScript (https://github.com/raganwald/homoiconic/blob/master/2012/12/combinators_1.md)
    - Part II (https://github.com/raganwald/homoiconic/blob/master/2012/12/combinators_2.md)
- Prototypes and Inheritance in JavaScript (http://msdn.microsoft.com/en-us/magazine/ff852808.aspx)

- ECMA 262-3 in detail (http://dmitrysoshnikov.com/tag/ecma-262-3/)
- JavaScript Garden (http://bonsaiden.github.com/JavaScript-Garden/)
- Google JavaScript Style Guide (http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml)
- Github JavaScript Styleguide (https://github.com/styleguide/javascript)
- Wordpress JavaScript Coding Standards (http://make.wordpress.org/core/handbook/coding-standards/javascript/)
- Extensions
  - RubyJS (http://rubyjs.org/) is a JavaScript implementation of all methods from Ruby classes like Array, String, Numbers, Time and more.
  - Mout (http://moutjs.com/) is a collection of modular JavaScript utilities that can be used in the browser as AMD modules or on node.js (without any overhead).
  - bacon.js (https://github.com/raimohanska/bacon.js): A small functional reactive programming lib for JavaScript.
- Flow Control
  - Coroutine Event Loops in Javascript (http://syzygy.st/javascript-coroutines/)
  - How To Node Article on promises describing both sides: node.js and browser (http://howtonode.org/promises)
  - Video by Douglas Crockford on monads which is touching Promises (http://www.youtube.com/watch?v=dkZFtimgAcM)
  - Не надо давать обещания, или Promises наоборот (http://habrahabr.ru/post/166419/)
  - Promises are the monad of asynchronous programming (http://blog.jcoglan.com/2011/03/11/promises-are-the-monad-of-asynchronous-programming/)
  - A Study on Solving Callbacks with JavaScript Generators (http://jlongster.com/A-Study-on-Solving-Callbacks-with-JavaScript-Generators)
  - What's The Point Of Promises? (http://www.kendoui.com/blogs/teamblog/posts/13-03-28/what-is-the-point-of-promises.aspx)
  - Promises/A+ Spec (http://promises-aplus.github.com/promises-spec/)
  - Callbacks vs Coroutines (https://medium.com/code-adventures/174f1fe66127)
  - This document is intended to explain how promises work (https://raw.github.com/kriskowal/q/master/design/README.js)
  - Video: Monads and Gonads (YUIConf Evening Keynote) (http://youtu.be/dkZFtimgAcM)
  - Libraries
    - Standalone-Deferred (https://github.com/warpdesign/Standalone-Deferred)
    - Standalone-Deferred (https://github.com/Mumakil/Standalone-Deferred)
    - Flowy (https://github.com/geeqie/node-flowy)
    - Step (https://github.com/creationix/step)
    - kew (https://github.com/Obvious/kew) is a lightweight promise library optimized for node.js
    - jQuery Timing (http://creativecouple.github.com/jquery-timing/)
    - RSVP.js (https://github.com/tildeio/rsvp.js)
    - q (https://github.com/kriskowal/q)
    - Watch.js (https://github.com/melanke/Watch.JS)

### HTML

- Book: Dive into HTML5 (http://diveintohtml5.info/)
- Principles of writing consistent, idiomatic HTML (https://github.com/necolas/idiomatic-html)
- Google HTML/CSS Style Guide (http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml)
- Wordpress HTML Coding Standards (http://make.wordpress.org/core/handbook/coding-standards/html/)

### Higher Level Languages

- Dart (http://www.dartlang.org/): Typed language compiled to JavaScript. By Google.
- Sass (http://sass-lang.com/) is an extension of CSS3, adding nested rules, variables, mixins, selector inheritance, and more. It's translated to well-formatted, standard CSS using the command line tool or a web-framework plugin.
- Less (http://lesscss.org/) extends CSS with dynamic behavior such as variables, mixins, operations and functions.
- Roole (http://roole.org/) is a language that compiles to CSS.
- Stylus (http://learnboost.github.io/stylus/): Expressive, dynamic, robust CSS
- TypeScript (http://www.typescriptlang.org/) is a typed superset of JavaScript that compiles to plain JavaScript (by Microsoft)

## Frameworks

### JavaScript

- Choosing a framework (http://todomvc.com/)
- Angular (http://angularjs.org/)
  - Github (https://github.com/angular/angular.js)

    - Learning
        - AngularJS-Learning (https://github.com/jmcunningham/AngularJS-Learning): huge list of Angular learning resources
        - AngularJS Screencasts (http://www.egghead.io/)
        - Building Huuuuuge Apps with AngularJS (http://briantford.com/blog/huuuuuge-angular-apps.html)
        - What are the nuances of scope prototypal / prototypical inheritance in AngularJS? (http://stackoverflow.com/questions/14049480/what-are-the-nuances-of-scope-prototypal-prototypical-inheritance-in-angularjs)
        - AngularJS from Basics to Dependency Injection (http://suhairhassan.com/2013/06/07/angularjs-in-depth-part-1.html)
        - AngularJS To Do List App (http://www.dreamfactory.com/angularjs-example)
        - AngularJS vs Ember (http://eviltrout.com/2013/06/15/ember-vs-angular.html)
        - The Hitchhiker's Guide to the Directive (http://amitgharat.wordpress.com/2013/06/08/the-hitchhikers-guide-to-the-directive/)
        - Frontend Workflows with Grunt and Angular JS (http://youtu.be/fSAgFxjFSqY)
    - Integration
        - angular-requirejs-seed (https://github.com/tnajdek/angular-requirejs-seed): this is a fork of Angular Seed but with changes needed for requireJS support.
        - Writing Reusable AngularJS Components with Bower (http://briantford.com/blog/angular-bower.html)
        - Native AngularJS directives for Twitter's Bootstrap. (http://angular-ui.github.io/bootstrap/)
        - Automating AngularJS With Yeoman, Grunt & Bower (http://newtriks.com/2013/06/11/automating-angularjs-with-yeoman-grunt-and-bower/)
        - Optimizing Angular Templates with Grunt on Heroku (http://www.codelord.net/2013/06/16/optimizing-angular-templates-with-grunt-on-heroku/)
        - Building Offline Applications With AngularJS and PouchDB (http://mircozeiss.com/building-offline-applications-with-angularjs-and-pouchdb/)
- jQuery (http://jquery.com/)
    - Github (https://github.com/jquery/jquery)
    - jQuery Wiki Page "Plugins/Authoring" (http://docs.jquery.com/Plugins/Authoring)
    - 30 Days to Learn jQuery (http://freecourses.tutsplus.com/30-days-to-learn-jquery/index.html) - A free Tut+ Premium course
    - jQuery Plugin Development Boilerplate (http://www.websanova.com/tutorials/jquery/jquery-plugin-development-boilerplate)
    - The Ultimate Guide to Writing jQuery Plugins (http://www.websanova.com/tutorials/jquery/the-ultimate-guide-to-writing-jquery-plugins)
    - Book: jQuery Fundamentals (http://jqfundamentals.com/)
    - Pragmatic jQuery Style (https://github.com/modulejs/pragmatic-jquery)
    - Handpicked jQuery Plugins (http://iwantaneff.in/repo/)
- dojo (http://dojotoolkit.org/)
    - Github (https://github.com/dojo/dojo)
    - Documentation (http://dojotoolkit.org/documentation/)
    - Writing Your Own Widget (http://dojotoolkit.org/reference-guide/1.9/quickstart/writingWidgets.html)
    - 10 Reasons Why Your Projects Should Use the Dojo Toolkit (http://net.tutsplus.com/tutorials/javascript-ajax/10-reasons-why-your-projects-should-use-the-dojo-toolkit/) by David Walsh
- Backbone (http://backbonejs.org/)
    - Github (https://github.com/documentcloud/backbone)
    - Book: Developing Backbone.js Applications (http://addyosmani.github.com/backbone-fundamentals/)
    - Unsuck your backbone (https://speakerdeck.com/ammeep/unsuck-your-backbone)
    - A Visual Guide to Marionette.js Views (http://blog.artlogic.com/2013/03/26/a-visual-guide-to-marionette-js-views/)
    - Building a Backbone.js app (http://dailyjs.com/2012/11/29/backbone-tutorial-1/) tutorial series.
    - Migrating an existing app to Backbone (http://www.ofbrooklyn.com/2012/11/13/backbonification-migrating-javascript-to-backbone/)
    - Chaplin.js (http://chaplinjs.org/): An architecture for JavaScript applications using the Backbone.js library.
- Meteor (http://meteor.com/): An open-source platform for building top-quality web apps in a fraction of the time.
    - Github (https://github.com/meteor/meteor)
    - Best Learning Resources for Meteor.js (http://yauh.de/articles/376/best-learning-resources-for-meteorjs)
- Zepto.js (http://zeptojs.com/): A minimalist JavaScript library for modern browsers with a largely jQuery-compatible API.
    - Github (https://github.com/madrobby/zepto)
- Minified.js (http://minifiedjs.com/): Minified.js is a client-side JavaScript library, comparable to jQuery and MooTools in scope. Its features include DOM manipulation, animation, events, cookies and HTTP requests.
    - Github (https://github.com/timjansen/minified.js/)
- React (http://facebook.github.io/react/) by Facebook
    - Github (https://github.com/facebook/react)

- Qooxdoo (http://qooxdoo.org/) is a universal JavaScript framework for building rich, interactive application, native-like apps for mobile devices, light-weight single-page oriented web applications or even applications to run outside the browser.
- jQWidgets (http://www.jqwidgets.com/) is jQuery based framework and set of widgets for building web-based applications that work on PC, Touch and Mobile devices
- Flight (http://twitter.github.io/flight/) is an event-driven web framework, from Twitter
- Singool.js (http://fahad19.github.com/singool/) Lightweight JavaScript framework for developing single-page web applications
- Knockout (http://knockoutjs.com/): Simplify dynamic JavaScript UIs by applying the Model-View-View Model (MVVM) pattern
- Sammy.js (http://sammyjs.org/) is a tiny JavaScript framework developed to ease the pain and provide a basic structure for developing JavaScript applications.
- Ember.js (http://emberjs.com/): A framework for creating ambitious web applications.
    - Let's Learn Ember (http://freecourses.tutsplus.com/lets-learn-ember/) - A free Tut+ Premium course
- Maria (https://github.com/petermichaux/maria): The MVC framework for JavaScript applications. The real MVC. The Smalltalk MVC. The Gang of Four MVC.
- Terrific Composer (http://terrifically.org/composer/) is a Frontend Development Framework specifically designed for building deluxe frontends based on the Terrific concept (http://terrifically.org/)
- Rivets.js (http://rivetsjs.com/): Lightweight and powerful data binding + templating solution for building modern web applications.
- Synapse (http://bruth.github.com/synapse/docs/): Data Binding For The Rest Of Us
- A Comprehensive Collection Of Javascript Application Frameworks (http://www.designyourway.net/blog/resources/a-comprehensive-collection-of-javascript-application-frameworks-28-examples/)
- JavaScript Data Binding Frameworks (http://weblogs.asp.net/dwahlin/archive/2012/07/08/javascript-data-binding-frameworks.aspx)
- Underscore.js (http://underscorejs.org/)

### CSS

- A collection of best frameworks (http://usablica.github.io/front-end-frameworks/compare.html?v=2.0)
- Responsive Grid System (http://responsive.gs/)
- Responsive Grid System (2) (http://www.responsivegridsystem.com/)
- Golden Grid System (http://goldengridsystem.com/)
- Compass (http://compass-style.org/): An open-source CSS Authoring Framework.
- Pondasee (http://tokokoo.github.com/pondasee/): Front-end starter kit combined with the power of SCSS & Compass.
- Centurion (http://jhough10.github.com/Centurion/): A responsive web framework that scales with your device.
- uikit (http://getuikit.com/): A lightweight and modular front-end framework for developing fast and powerful web interfaces.
- Foundation by Zurb (http://foundation.zurb.com/): The most advanced responsive front-end framework in the world.
    - Foundation 3 Documentation (http://foundation.zurb.com/old-docs/f3/)
    - Foundation 4 Documentation (http://foundation.zurb.com/docs/)
    - Stylus based port (https://github.com/blai/foundation)
- Gumby (http://gumbyframework.com/)
- ProfoundGrid (http://www.profoundgrid.com/)
- Groundwork (http://groundwork.sidereel.com/): GroundworkCSS's flexible grid system enables you to do rapid development and scales to fit any screen size, from handheld devices to large format televisions.
- skelJS (http://skeljs.org/): skelJS is a lightweight frontend framework for building responsive sites and apps.
- Ink (http://ink.sapo.pt): Ink is a set of tools for quick development of web interfaces.
- neat (http://neat.bourbon.io/): A lightweight semantic grid framework for Sass and Bourbon.
- Kube (http://imperavi.com/kube/)
- rwdgrid (http://rwdgrid.com/)
- Simple Grid (http://thisisdallas.github.com/Simple-Grid/)
- One% CSS Grid (http://onepcssgrid.mattimling.com/)
- Workless (http://workless.ikreativ.com/)
- inuit.css (http://inuitcss.com/) - a powerful, scalable, Sass-based, BEM, OOCSS framework.
- Kraken (https://github.com/cferdinandi/kraken) is a lightweight, mobile-first boilerplate for front-end web developers.
- Axis (http://roots.cx/axis/) - clean Stylus-based foundation to build your styles upon
- Jeet (http://jeetframework.com/) next gen responsive grid
- Susy (http://susy.oddbird.net/) - Responsive grids for Compass
- Pure (http://purecss.io/) - Minimal, flat, responsive CSS modules.
- Topcoat (https://github.com/topcoat/topcoat) is a CSS library evolved from the Adobe design language developed for Brackets, Edge Reflow, and feedback from the PhoneGap app

developer community.

### Bootstrap

- Home Page (http://http://getbootstrap.com/)
- Github (https://github.com/twitter/bootstrap/)
- Bootstrap Hero List (http://www.bootstraphero.com/the-big-badass-list-of-twitter-bootstrap-resources): The big badass list of bootstrap resources

#### Bootstrap - Widgets

- Color and Date Picker (http://www.eyecon.ro/colorpicker-and-datepicker-for-twitter-bootstrap.htm)
- WYSIHTML5 RTE for Bootstrap (http://jhollingworth.github.com/bootstrap-wysihtml5/)
- Bootstrap Image Gallery (http://blueimp.github.com/Bootstrap-Image-Gallery/)
- jQuery UI Bootstrap (http://addyosmani.github.com/jquery-ui-bootstrap/)
- Pines Notify (http://pinesframework.org/pnotify/): JavaScript notifications for Bootstrap or jQuery UI.
- fuelUX (http://exacttarget.github.com/fuelux/): Fuel UX extends Twitter Bootstrap with additional lightweight JavaScript controls.
- A date range picker for Twitter Bootstrap (http://www.dangrossman.info/2012/08/20/a-date-range-picker-for-twitter-bootstrap/)
- Bootbox.js (http://bootboxjs.com/): alert, confirm and flexible dialogs for Twitter's Bootstrap framework
- Time Picker (http://jdewit.github.com/bootstrap-timepicker/)
- Date/Time Picker (http://tarruda.github.com/bootstrap-datetimepicker/)
- ReCaptcha (http://jsfiddle.net/hqv27/)
- Bootstrap Modal (https://github.com/jschr/bootstrap-modal): Extends Bootstrap's native modals to provide additional functionality.
- SelectBoxIt (http://gregfranko.com/jquery.selectBoxIt.js)
- File Upload (http://jasny.github.com/bootstrap/javascript.html#fileupload)
- BootPag (http://botmonster.com/jquery-bootpag/): BootPag - dynamic pagination jQuery plugin for twitter bootstrap
- Bootstrap Arrows (http://bootstrap-arrows.iarfhlaith.com/): A simple jQuery plugin and add-on to the popular Twitter Bootstrap framework to include the use of arrows at any angle in your UI designs.
- X-editable (http://vitalets.github.com/x-editable/): In-place editing with Twitter Bootstrap, jQuery UI or pure jQuery
- Pagination (https://github.com/markbates/jquery-bootstrap-pagination)
- Slider (http://ghusse.github.com/jQRangeSlider/)
- Application Wizard (https://github.com/amoffat/bootstrap-application-wizard)

#### Bootstrap - Themes

- Bootswatch (http://bootswatch.com/): Theme catalog.
- Jumpstart UI (http://jumpstartui.com/): Another catalog for paid themes
- Darkstrap (http://danneu.com/posts/4-darkstrap-css-a-dark-theme-for-twitter-bootstrap-2): A dark theme.
- jQuery Mobile Boostrap (https://github.com/commadelimited/jQuery-Mobile-Bootstrap-Theme): A jQuery Mobile theme based on Twitter Bootstrap
- Wrapbootstrap (http://wrapbootstrap.com/)
- Flat UI (http://designmodo.github.io/Flat-UI/)Free Web User Interface Kit that Rocks!

#### Bootstrap - Themes - Wordpress

- BootstrapWP (http://rachelbaker.me/bootstrapwp/)
- Roots Theme (http://www.rootstheme.com/)
- Bootstrap (http://digitalnature.eu/themes/bootstrap/)
- Bootstrap for Wordpress (http://wpbootstrap.iubenda.com/)
- Bootstrap for WP (http://envexlabs.com/2011/09/bootstrap-for-wordpress/)
- StrapPress (http://braginteractive.com/bootstrap/)
- WPBS (http://320press.com/wpbs/)

#### Bootstrap - Themes - Misc

- CSS3 Microsoft Modern Buttons (http://ace-subido.github.com/css3-microsoft-metro-buttons/index.html)
- BootMetro (http://aozora.github.com/bootmetro/): Metro style web framework
- Cosmo (http://bootswatch.com/cosmo/): Windows 8 inspired theme
- Inspiritas (http://littke.com/2012/11/06/inspiritas-bootstrap-theme-by-ripple.html)
- Facebook (http://ckrack.github.com/fbootstrapp/)

- Google+ (http://xbreaker.github.com/plusstrap/index.html)
- Flatstrap (http://littlesparkvt.com/flatstrap/index.html)
- Pinstrap (http://bragthemes.com/theme/pinstrap-pinterest-meets-bootstrap/)
    - Boilerstrap (https://github.com/emorikawa/boilerstrap): A blank slate for the modern web. Just add creativity.
    - Bootstrap GUI PSD (http://gui.repixdesign.com/#freebies): a toolkit designed to kickstart webdesign of webapps and sites.
    - Font Awesome (http://fortawesome.github.com/Font-Awesome/): The iconic font designed for use with Twitter Bootstrap
    - Fontello (https://github.com/fontello/fontello/): Icon fonts generator to get only the required fonts.
    - Typo3 Extension (http://typo3.org/extensions/repository/view/mn_twitter_bootstrap)
    - Bootstrap CDN (http://www.bootstrapcdn.com/) Free CDN for Twitter Bootstrap
    - Bootstrap Tour (http://bootstraptour.com/) Quick and easy way to build your product tours with Twitter Bootstrap Popovers.
    - Bootsnipp (http://bootsnipp.com/): Design elements and code snippets for Bootstrap HTML/CSS/JS framework
    - Form Builder (http://bootsnipp.com/forms)
    - PaintStrap (http://paintstrap.com/): Generate beautiful Twitter Bootstrap themes using the Adobe kuler / COLOURlovers color scheme
    - TODC (http://todc.github.com/todc-bootstrap/index.html): A Google-styled theme for Twitter Bootstrap.
    - Layoutit! (http://www.layoutit.com/)

## Cross Browser

- Normalize.css (http://necolas.github.io/normalize.css/)
- HTML5 Cross Browser Polyfills (https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills)
- Viewport Component (https://github.com/pazguille/viewport)
- Can I Use (http://caniuse.com/) - Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers.
- HTML5 Please (http://html5please.com/) - Collective recommendations for polyfills and implementation of HTML5 and CSS3 features.

## Cross Device

### Responsive

- Responsive Boilerplate (http://responsiveboilerplate.com/): A starting point to responsive web design!
- Video: Responsive Design Workflow by Stephen Hay (http://vimeo.com/45915667) + Slides (http://de.slideshare.net/stephenhay/mobilism2012)
- Responsive Patterns (http://bradfrost.github.io/this-is-responsive/patterns.html)
    - Responsive Navigation Patterns (http://bradfrostweb.com/blog/web/responsive-nav-patterns/)
    - Complex Navigation Patterns for Responsive Design (http://bradfrostweb.com/blog/web/complex-navigation-patterns-for-responsive-design/)
- Responsinator (http://www.responsinator.com/)
- How to make a Responsive Newspaper-like layout (http://www.newnet-soft.com/blog/responsive-multi-column)
- The State Of Responsive Web Design (http://mobile.smashingmagazine.com/2013/05/29/the-state-of-responsive-web-design/)
- Facing The Challenge: Building A Responsive Web Application (http://mobile.smashingmagazine.com/2013/06/12/building-a-responsive-web-application/)
    - This repo contains techniques and concepts to build responsive & hybrid web applications. (https://github.com/webpro/responsive-web-apps)

#### Responsive - Tables

- Responsive Data Tables (http://css-tricks.com/responsive-data-tables/)
- stacktable.js (http://johnpolacek.github.com/stacktable.js/)
- FooTable (http://themergency.com/footable/)

#### Responsive - Events

- Breakpoints.js (http://xoxco.com/projects/code/breakpoints/)
- Harvey.js (http://harvesthq.github.com/harvey/)
- Enquire.js (https://github.com/WickyNilliams/enquire.js)

#### Responsive - Images

- Fluid Images (http://unstoppablerobotninja.com/entry/fluid-images/)
- How to Use Responsive Images to Make Your Site Shine on Any Platform (http://www.shutterstock.com/blog/2013/05/how-to-use-responsive-images-to-make-your-site-shine-on-any-platform/)
- Adaptive Images (http://adaptive-images.com/)
- Why We Need Responsive Images (http://timkadlec.com/2013/06/why-we-need-responsive-images/)

- Riloadr (https://github.com/tubalmartin/riloadr): A cross-browser framework-independent responsive images loader.
- jQuery Picture (http://jquerypicture.com/)
- Picturefill (https://github.com/scottjehl/picturefill)
- Clown Car (https://github.com/estelle/clowncar) Technique for Responsive Images
- imgLiquid (https://github.com/karacas/imgLiquid) jQuery plugin to resize images to fit in a container.
- lazy load images (http://ezyz.github.io/Lazy-Load-Images-without-jQuery/) lazy load images with out jquery.

#### Responsive - Text

- FitText (http://fittextjs.com/): A jQuery plugin for inflating web type
- Out Of Words! (http://starburst1977.github.io/out-of-words/): The responsive typography framework behind Words App
- Responsive Font Sizing (http://www.newnet-soft.com/blog/responsivefontsizing): Making your font size respond to your screen size, easy & maintainable.
- Responsive Measure (http://jbrewer.github.com/Responsive-Measure/)

### E-Mail

- Responsive Email Design (http://www.campaignmonitor.com/guides/mobile/): A great guide to create Responsive Email Design
- Responsive Email Templates (http://zurb.com/playground/responsive-email-templates): Some responsive templates for email
- HTML Email Boilerplate (http://htmlemailboilerplate.com/)
- CSS Support in Email (http://www.campaignmonitor.com/css/)

### Mobile

#### Mobile - Frameworks

- jQuery Mobile (http://jquerymobile.com/): Touch-Optimized Web Framework for Smartphones & Tablets
- jQTouch (http://jqtouch.com/) is a Zepto/jQuery plugin for mobile web development on the iPhone, Android, iPod Touch, and other forward-thinking devices.
- Junior (http://justspamjustin.github.com/junior/)

#### Mobile - Emulators

- thumbs.js (http://mwbrooks.github.com/thumbs.js/)
- Phantom Limb (http://viewinglens.com/phantom-limb/)
- Touché (https://github.com/davidcalhoun/touche)

#### Mobile - Scrolling

- jSwipeKinetic (http://jswipekinetic.codeplex.com/) is a jQuery plugin that enables you to add kinetic scrolling on your touch optimized projects. jSwipeKinetic is build on top of jGestures (jgestues.codeplex.com).
- jQuery.pep.js (http://pep.briangonzalez.org/): A lightweight plugin for kinetic-drag on mobile/desktop.
- Overscroll (http://www.azoffdesign.com/overscroll) is a jQuery plug-in that mimics the iphone/ipad scrolling experience in a browser.
- Zynga Scroller (https://github.com/zynga/scroller)
- pull-to-refresh.js (https://github.com/dantipa/pull-to-refresh-js)
- Overthrow (https://github.com/filamentgroup/Overthrow)
- Antiscroll (http://learnboost.github.com/antiscroll/)
- iscroll (http://cubiq.org/iscroll-4)

#### Mobile - Gestures

- jGestures (http://jgestures.codeplex.com/) jQuery plugin enables you to add gesture events such as ‘pinch’, ‘rotate’, ‘swipe’, ‘tap’ and ‘orientationchange’ just like native jQuery events.
- hammer.js (http://eightmedia.github.com/hammer.js/) is a focused, standalone JavaScript library, only for multi-touch gestures.
- Touchy (http://touchyjs.org/): is a highly configurable jQuery plugin. It exposes event data (velocity, etc.) for longpress, drag, pinch, rotate and swipe.
- TouchSwipe (http://labs.skinkers.com/content/touchSwipe/)
- TipTap (https://github.com/marcbourlon/TipTap)
- jQuery.event.swipe (http://stephband.info/jquery.event.swipe/)
- toe.js (https://github.com/dantipa/toe.js) is a tiny library based on jQuery to enable sophisticated gestures on touch devices.
- Jester (https://github.com/plainview/Jester)
- JS Touch Layer (https://github.com/andyjamesdavies/jsTouchLayer)
- Touch Events and Abstractions (http://yuilibrary.com/yui/docs/event/touch.html)

#### Mobile - DOM Objects Manipulation

- jQuery.event.move (http://stephband.info/jquery.event.move/)
- WKTouch (https://github.com/alexgibson/WKTouch): A JavaScript plugin for touch-capable devices, enabling multi-touch drag, scale and rotate on HTML elements.

#### Mobile - Tap Acceleration

- Energize (https://github.com/davidcalhoun/energize.js)
- Tappable (http://cheeaun.github.com/tappable/) is a simple, standalone library to invoke the tap event for touch-friendly web browsers.
- Fastclick (https://github.com/ftlabs/fastclick) is a simple, easy-to-use library for eliminating the 300ms delay between a physical tap and the firing of a click event on mobile browsers.
- Lightning Touch (https://github.com/ucsf-ckm/LightningTouch)
- Creating Fast Buttons for Mobile Web Applications (https://developers.google.com/mobile/articles/fast_buttons)

#### Mobile - Layout

- Snap.js (https://github.com/jakiestfu/Snap.js) A Javascript Library for creating beautiful mobile shelfs in Javascript (Facebook and Path style side menus)
- Flickable.js (https://github.com/tomlongo/Flickable.js) allows you to make any element touchable; useful for flicking between sections, or sliding elements around the page.
- PageSlide (http://srobbin.com/jquery-plugins/pageslide/) is a jQuery plugin which slides a webpage over to reveal an additional interaction pane.
- Swipe (http://swipejs.com/) is a lightweight mobile slider with 1:1 touch movement, resistant bounds, scroll prevention, and completely library agnostic.
- Swiper (http://www.idangero.us/sliders/swiper/): Mobile touch slider with hardware accelerated transitions.
- jQuery Mobile Pagination Plugin (http://filamentgroup.com/lab/jquery_mobile_pagination_plugin/)
- SwipeSlide (https://github.com/max-power/swipeslide): A Zepto Plugin for iOS like swipe navigation.
- stackable.js (http://johnpolacek.github.com/stacktable.js/) is an invaluable jQuery plugin that stacks your tables for small screens. It's a huge advantage in terms of usability on mobile devices.

#### Mobile - Reacting to Device Sensors

- lenticular.js (http://lenticular.attasi.com/) is a jQuery plugin for creating image animations that response to tilting or mouse events.
- This End Up: Using Device Orientation (http://www.html5rocks.com/en/tutorials/device/orientation/)

#### Mobile - iOS

- Safari Image Delivery Best Practices (http://developer.apple.com/library/safari/#documentation/NetworkingInternet/Conceptual/SafariImageDeliveryBestPractices/Introduction/Introduction.html#//apple_ref/doc/uid/TP40012449)
- Safari Graphics, Media, and Visual Effects Coding How-To's (http://developer.apple.com/library/safari/#codinghowtos/Desktop/GraphicsMediaAndVisualEffects/_index.html#//apple_ref/doc/uid/DTS40008251)
- Safari CSS Visual Effects Guide (http://developer.apple.com/library/safari/#documentation/InternetWeb/Conceptual/SafariVisualEffectsProgGuide/Introduction.html#//apple_ref/doc/uid/TP40008032)
- Safari Web Content Guide (http://developer.apple.com/library/safari/#documentation/AppleApplications/Reference/SafariWebContent/Introduction/Introduction.html#//apple_ref/doc/uid/TP40002051)
- Getting Started with iOS Web Apps (http://developer.apple.com/library/safari/#referencelibrary/GettingStarted/GS_iPhoneWebApp/_index.html#//apple_ref/doc/uid/TP40008134)
- The iPad Web Design & Development Toolbox (http://iphone.appstorm.net/roundups/developer/the-ipad-web-design-development-toolbox/)
- Targeting the iPhone 4 Retina Display with CSS3 Media Queries (http://blog.iwalt.com/2010/06/targeting-the-iphone-4-retina-display-with-css3-media-queries.html)
- How do I lock the orientation to portrait mode in a iPhone Web Application? (http://stackoverflow.com/questions/1207008/how-do-i-lock-the-orientation-to-portrait-mode-in-a-iphone-web-application)
- jQuery Retina Display Plugin (http://troymcilvena.com/post/998277515/jquery-retina)
- retina.js (http://retinajs.com/)
- Retina Images (http://retina-images.complexcompulsions.com/)

#### Printers

- Tips And Tricks For Print Style Sheets (http://coding.smashingmagazine.com/2013/03/08/tips-tricks-print-style-sheets/)

## Patterns & Snippets

- Responsive Patterns (http://bradfrost.github.io/this-is-responsive/patterns.html)
- CSS3 Code Snippets (http://www.webinterfacelab.com/snippets/)
- The Blueprints (http://tympanus.net/codrops/category/blueprints/) are a collection of basic and minimal website concepts, components, plugins and layouts with minimal style for easy adaption and usage, or simply for inspiration.

## DOM Manipulation

- xui (http://xuijs.com/): a super micro tiny dom library for authoring html5 mobile web applications.
- Tire (http://tirejs.com/) offers a more lightweight alternative to libraries such as jQuery, Prototype and Zepto.

## Typography

- Quick guide to webfonts via @font-face (http://www.html5rocks.com/en/tutorials/webfonts/quick/)
- How To Achieve Cross-Browser @font-face Support (http://blog.themeforest.net/tutorials/how-to-achieve-cross-browser-font-face-support/)
- Google Fonts (http://www.google.com/fonts/)
- Adobe Edge Web Fonts (http://html.adobe.com/edge/webfonts/): Get started with free web fonts.
  - Edge Web Fonts Website (https://edgewebfonts.adobe.com/fonts)
- Typekit (https://typekit.com/): easy way to use commercial web font on your site.
- Matrix of fonts bundled with Mac and Windows operating systems, Microsoft Office and Adobe Creative Suite (http://media.24ways.org/2007/17/fontmatrix.html)
- Typeset.css (http://joshuarudd.github.com/typeset.css/)
- typecast (http://beta.typecastapp.com/experiment-with-type)
- CSSTypography (http://tilomitra.github.io/csstypography/)
- SO: @font-face fonts only work on their own domain (http://stackoverflow.com/questions/2892691/font-face-fonts-only-work-on-their-own-domain)
- FitText (http://fittextjs.com/): A jQuery plugin for inflating web type
- TypeButter (http://typebutter.com/)
- slabText (https://github.com/freqDec/slabText/)
- Baseline.js (http://daneden.me/baseline/)
- jKerny (http://clearideaz.com/jkerny/)
- Lettering.js (http://letteringjs.com/)
- Trunk8 (http://jrvis.com/trunk8/) is an intelligent text truncation plugin to jQuery. It will cut off just enough text to prevent it from spilling over
- bacon (http://baconforme.com/) is a jQuery plugin that allows you to wrap text around a bezier curve or a line.
- CSS Typography cheat sheet (http://www.newnet-soft.com/blog/csstypography): Small roundup on CSS features that will enhance your web typography.

## Services (Free & Commercial)

- TheToolbox (http://www.thetoolbox.cc/)
- colourco (http://colourco.de/)
- Color Scheme Designer (http://colorschemedesigner.com): find resonate colors for a great design.
- Adobe Kuler (http://kuler.adobe.com): create and share color schemes.
- HTML Entity Character Lookup (http://leftlogic.com/projects/entity-lookup)
- SpritePad (http://spritepad.wearekiss.com/)
- Responsinator (http://www.responsinator.com/)
- HTML Shell (http://htmlshell.com/)
- Form Builder (http://livetools.uiparade.com/form-builder.html)
- Zen BG (http://mudcu.be/bg/)
- What Font Is (http://www.whatfontis.com/) identifies fonts from an image.
- Prepros (http://alphapixels.com/prepros/)
- Browser Shots (http://browsershots.org/) tests compatibility in almost every browser.
- CSS3 Media Query Builder (http://arcsec.ca/media-query-builder/)
- CSSArrowPlease (http://cssarrowplease.com) Simple tooltip arrows with CSS
- jsFiddle (http://jsfiddle.net)
- codepen (http://codepen.io/) is an HTML, CSS, and JavaScript code editor right in your browser with instant previews of the code you see and write.
- site44 (https://www.site44.com/) turns Dropbox folders to websites
- Backlift (https://www.backlift.com/): Work in your Dropbox using your favorite editor. When you save your files, your website will automatically refresh.
- Smush.it (http://www.smushit.com/ysmush.it/) uses optimization techniques specific to image format to remove unnecessary bytes from image files.
- JPEGmini Photo Server (http://www.jpegmini.com/server) is an image optimization server which runs on Amazon EC2 (commercial).
- TinyPNG (http://tinypng.org/): Advanced lossy compression for PNG images that preserves full alpha transparency.
- Website Builders (http://en.wikipedia.org/wiki/Website_builder)
  - Squarespace (http://www.squarespace.com/)
  - BaseKit (http://www.basekit.com/)
  - Doomby (http://www.doomby.com/)
  - Edicy (http://www.edicy.com/)

- IM Creator (http://imcreator.com/)
- Jimdo (http://www.jimdo.com/)
- Moonfruit (http://www.moonfruit.com/)
- uCoz (http://www.ucoz.com/)
- Webnode (http://www.webnode.com)
- Webs (http://www.webs.com/)
- Weebly (http://www.weebly.com/)
- Wix (http://www.wix.com)
- Wordpress.com (http://wordpress.com/)
- Yola (https://www.yola.com/)
- 1&1 Homepage (http://homepage.1und1.de/)

## Animation

- Stylie (http://jeremyckahn.github.com/stylie/)
- animate.less (https://github.com/machito/animate.less), is a bunch of cool, fun, and cross-browser animations converted into LESS for you to use in your Bootstrap projects. Great for emphasis, home pages, sliders, and general just-add-water-awesomeness.
- Canvas Advanced Animation Toolkit (http://hyperandroid.github.com/CAAT/) is an scene graph director-based animation framework for javascript based in the concept of a timeline.
- tween.js (https://github.com/sole/tween.js): Super simple, fast and easy to use tweening engine which incorporates optimised Robert Penner's equations.
- Janis (https://github.com/MikeMcTiernan/Janis) is a lightweight Javascript framework that provides simple animations via CSS transitions for modern browsers on the web as well as mobile devices.
- Rekapi (http://rekapi.com/): A keyframe animation library for JavaScript
- CanvasScript3 (http://www.arahaya.com/canvasscript3/) is a Javascript library for the new HTML5 Canvas with an interface similar to ActionScript3. This library enables Sprite Groups, Layers, Mouse Events, Keyboard Events, Bitmap Effects, Tween Animations etc.
- Shifty (http://jeremyckahn.github.com/shifty/) is a tweening engine built in JavaScript. It is designed to fit any number of tweening needs.
- emile.js (https://github.com/madrobby/emile) is a no-frills stand-alone CSS animation JavaScript framework.
- Firmin (http://extralogical.net/projects/firmin/): a JavaScript animation library using CSS transforms and transitions.
- $fx() (https://github.com/agilemd/Fx): A compact, lightweight Javascript Library for animation.
- Keanu (https://github.com/wambotron/Keanu) is a micro-lib for animation on Canvas/JS.
- jsAnim (http://jsanim.com/) is a powerful, yet easy to use library for adding impressive animations to websites, without sacrificing standards or accessibility. Weighing in at just under 25 kilobytes, jsAnim packs a lot of punch for such a little application.
- GreenSock Animation Platform (http://www.greensock.com/get-started-js/) is a suite of tools for scripted animation.
  - Learning Resources (http://www.greensock.com/learning/)
  - Examples (http://ahrengot.com/tutorials/greensock-javascript-animation/)
  - Codepen Repository (http://codepen.io/GreenSock/): Codepen repository with examples of Greensock usage and code
- scripty2 (http://scripty2.com/) is an open source JavaScript framework for advanced HTML-based user interfaces. Or simply put, scripty2 helps you build a more delicious web.
- Animator.js (http://berniesumption.com/software/animator/): JavaScript animation library.
- Processing.js (http://processingjs.org/): is the sister project of the popular Processing visual programming language, designed for the web. Processing.js makes your data visualizations, digital art, interactive animations, educational graphs, video games, etc. work using web standards and without any plug-ins.
- jQuery Transit (http://ricostacruz.com/jquery.transit/): Super-smooth CSS3 transformations and transitions for jQuery.
- Move.js (http://visionmedia.github.com/move.js/) is a small JavaScript library making CSS3 backed animation extremely simple and elegant.
- Collie (http://jindo.dev.naver.com/collie/) is a Javascript library that helps to create highly optimized animations and games using HTML 5.
- Year Of Moo (http://www.yearofmoo.com/): Angular.js Animations
- animate.css (http://daneden.me/animate/) is a bunch of cool, fun, and cross-browser animations for you to use in your projects. Great for emphasis, home pages, sliders, and general just-add-water-awesomeness.
- Approach (http://srobbin.com/jquery-plugins/approach/)
- Magic (https://github.com/miniMAC/magic) is a CSS3 framework with many animations, it's simple to use and many animations are cross-browser compatible.

## Widgets

### Widgets - Kits

- AlloyUI (http://liferay.github.com/alloyui.com/)
- Kendo UI (http://www.kendoui.com/)
- jQuery UI Bootstrap (http://addyosmani.github.com/jquery-ui-bootstrap/)

- Zebra (http://zebra.gravitysoft.org/): JavaScript library that follows easy OOP concept, provides HTML5 Canvas based Rich UI and includes Java to JavaScript converter tool

### Widgets - File Upload

- jQuery File Upload (http://blueimp.github.com/jQuery-File-Upload/)
- tus fileupload (http://www.tus.io/)
- Fine Uploader (http://fineuploader.com/)

### Widgets - Sitemap

- SlickMap CSS (http://astuteo.com/slickmap)

### Widgets - Table Of Contents

- Tocify (http://gregfranko.com/jquery.tocify.js)
- MagicNav.js (http://johnpolacek.github.com/MagicNav.js/)
- jQuery TOC (http://projects.jga.me/toc/)

### Widgets - Modals

- Avgrund (http://lab.hakim.se/avgrund/)
- Timeout Dialog (http://rigoneri.github.com/timeout-dialog.js/)
- Magnific Popup (http://dimsemenov.com/plugins/magnific-popup/)

### Widgets - Notifications

- Pines Notify (http://pinesframework.org/pnotify/)
- notifier.js (https://github.com/Srirangan/notifer.js)
- noty (http://needim.github.com/noty/)
- Toastr (http://codeseven.github.com/toastr/)
- alertify.js (http://fabien-d.github.com/alertify.js/)
- Apprise (http://thrivingkings.com/read/Apprise-v2-The-new-and-improved-attractive-alert-alternative-for-jQuery)

### Widgets - Gallery Sliders

- Sequence (http://www.sequencejs.com/)
- Rhinoslider (http://rhinoslider.com/)
- Awkward Showcase (http://www.awkwardgroup.com/sandbox/awkward-showcase-a-jquery-plugin/)
- Revolver.js (http://revolverjs.com/)
- Responsive CSS3 Slider (http://csscience.com/responsiveslidercss3/)
- Roundabout (http://fredhq.com/projects/roundabout/)
- Slid.es (http://slid.es/)
- Galleria (http://galleria.io/)

### Widgets - Pagination

- jPages (http://luis-almeida.github.com/jPages/)

### Widgets - Selects

- jQuery Recurrence Input (https://github.com/collective/jquery.recurrenceinput.js)
- Select2 (http://ivaynberg.github.com/select2/)
- jquery.selectable.js (http://endel.github.com/jquery.selectable.js/)
- chosen (https://github.com/harvesthq/chosen) is a library for making long, unwieldy select boxes more friendly.

### Widgets - Progress Bars

- Percentage Loader (http://widgets.better2web.com/loader/)
- progress.js (https://github.com/mdix/progress.js)

### Widgets - Trees

- Nestable (http://dbushell.com/2012/06/17/nestable-jquery-plugin/)

### Widgets - Navigation

- jPanelMenu (http://jpanelmenu.com)

### Widgets - Tooltips

- opentip (http://www.opentip.org/)

### Widgets - Misc

- jQuery Knob (http://anthonyterrien.com/knob/)
- DopelessRotate (http://www.dopeless-design.de/dopeless-rotate-jquery-plugin-360-degrees-product-viewer.html)
- jQuery Addresspicker (http://mngscl-10.s3-website-us-east-1.amazonaws.com/jquery-addresspicker-bootstrap/demos/index.html)
- Fangle (http://jotux.github.io/fangle/): create interactive documents from plain text.

## Visualization

- Photon (https://github.com/thomasxiii/photon)
- CSS3 shapes (http://www.css3shapes.com/)
- Morris.js (http://oesmith.github.com/morris.js/)
- Cube (http://square.github.com/cube/)
- Cubism.js (http://square.github.com/cubism/)
- D3.js (http://d3js.org/)
    - Introduction to D3 (http://www.janwillemtulp.com/2011/03/20/tutorial-introduction-to-d3/)
    - Interactive D3 Tutorial (http://vogievetsky.github.com/IntroD3/)
    - NVD3 (http://nvd3.org/)
- Crossfilter (http://square.github.com/crossfilter/)
- Datavisualization.ch (http://selection.datavisualization.ch/)
- jQuery.Gantt (http://taitems.github.com/jQuery.Gantt/)
- easy pie chart (http://rendro.github.com/easy-pie-chart/)
- JointJS JavaScript Diagramming Library (http://jointjs.com/)
- Gauge.js (http://bernii.github.com/gauge.js/)
- Google Chart Tools (https://developers.google.com/chart/)
- Piecon (http://lipka.github.com/piecon/)
- Viskit.js (http://wso2.github.com/viskit/)
- jStat (http://www.jstat.org/)
- CHAP Links Library (http://almende.github.com/chap-links-library/index.html)
- bonsai (http://bonsaijs.org/)
- Smoothie Charts (http://smoothiecharts.org/)
- DataMaps (http://datamaps.github.com/)
- mxgraph (http://jgraph.github.io/mxgraph/)
- Paper.js (http://paperjs.org) is an open source vector graphics scripting framework that runs on top of the HTML5 Canvas.
- Timeline.js (https://github.com/VeriteCo/TimelineJS): A Storytelling Timeline built in JavaScript.
- xCharts (http://tenxer.github.com/xcharts/) is a JS library for creating attractive, custom data-driven chart visualizations. It uses HTML, CSS, and SVG to create dynamic, fluid charts that can be easily customized and integrated.
- jQuery Gantt editor (http://roberto.open-lab.com/2013/01/28/jquery-gantt-editor-include-today/)
- amCharts (http://www.amcharts.com/) is a robust charting tool that will suit any dataviz need.
- Responsive vertical timeline (http://christian-fei.com/tutorials/simple-vertical-timeline-with-css/)
- Raphael.js (http://raphaeljs.com/) Easy vector graphics for the web.
- three.js (http://threejs.org) Javascript 3D library.
- Highcharts (http://www.highcharts.com/): Interactive JavaScript Charts for your web projects.
- Flot (http://www.flotcharts.org/): Attractive JavaScript plotting for jQuery
- Chart.js (http://www.chartjs.org/): Easy, object oriented client side graphs for designers and developers

## Validation & Forms

- Mention.js (https://github.com/jakiestfu/Mention.js) Lightweight wrapper for adding @user mention functionality to Twitter Bootstraps Typeahead plugin
- ALAJAX (http://www.alajax.com/)
- Parsley.js (http://parsleyjs.org): Validate your forms, frontend, without writing a single line of javascript!

- mailcheck.js (https://github.com/Kicksend/mailcheck): A jQuery plugin that suggests a right domain when your users misspell it in an email address.
- one-validation (https://github.com/One-com/one-validation) is a collection of regular expressions for general validation purposes.
- nextVal (http://jukebox42.github.com/nextVal/) is an easy-to-use, flexible and robust form validation plugin for jQuery.
- Fields.js (http://schneiderik.github.com/fields/) creates collections of fields. Each field is constantly evaluated for validity, and is accessible through the collection.
- IV.js (http://dadleyy.github.com/IV.js/) was created to provide a intutive way to provide validation filters that are useful in processing user input.
- Ladda (http://lab.hakim.se/ladda/): A UI concept which merges loading indicators into the action that invoked them.
- jQuery Super Labels (http://remy.bach.me.uk/superlabels_demo/)

## Transitions

- Quicksand (http://razorjack.net/quicksand/)
- Isotope (http://isotope.metafizzy.co/docs/introduction.html)
- flippant.js (http://mintchaos.github.io/flippant.js/)
- Meaningful Transitions (http://www.ui-transitions.com/)

## Numbers

- numeral.js (http://numeraljs.com/): A javascript library for formatting and manipulating numbers.
- accounting.js (http://josscrowcroft.github.com/accounting.js/)
- numbers.js (https://github.com/sjkaliski/numbers.js)

## Time and Dates

- moment.js (http://momentjs.com/): A 5.5kb javascript date library for parsing, validating, manipulating, and formatting dates.
- XDate (http://arshaw.com/xdate/)
- Pikaday (https://github.com/dbushell/Pikaday)

## Search

- Fullproof (http://reyesr.github.com/fullproof/)
- lunr.js (http://lunrjs.com/) is a simple full text search engine for your client side applications.

## Testing

- FiveUI (http://galoisinc.github.io/FiveUI/) Extensible/open browser extension & headless system for testing UI Consistency guidelines (or anything else you can test with injected JS).
- Writing Testable JavaScript (http://alistapart.com/article/writing-testable-javascript)
- Writing Testable Frontend Javascript Part 1 – Anti-patterns and their fixes (https://shanetomlinson.com/2013/testing-javascript-frontend-part-1-anti-patterns-and-fixes/)
- Introduction to BDD (http://dannorth.net/introducing-bdd/)
- cucumber (https://github.com/cucumber/cucumber/wiki)
    - cucumber.js (https://github.com/cucumber/cucumber-js)
    - Gherkin (https://github.com/cucumber/cucumber/wiki/Gherkin) is the language that Cucumber understands.
        - Feature Introduction (https://github.com/cucumber/cucumber/wiki/Feature-Introduction)
        - Given when then (https://github.com/cucumber/cucumber/wiki/Given-When-Then)
- Jasmine (http://pivotal.github.io/jasmine/) - JavaScript BDD testing framework with Rspec-like syntax
- QUnit (http://qunitjs.com/) - JavaScript unit testing framework
- PhantomCSS (https://github.com/Huddle/PhantomCSS) - for automating visual regression testing of website styling to support refactoring of CSS
- Karma (http://karma-runner.github.io/) - Spectacular Test Runner for JavaScript
- Chai Assertion Library (http://chaijs.com/) - Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.
- Sinon.JS (http://sinonjs.org/) - Standalone test spies, stubs and mocks for JavaScript.
- sinon-chai (https://github.com/domenic/sinon-chai) - Extends Chai with assertions for the Sinon.JS mocking framework.
- PhantomJS (http://phantomjs.org/) - PhantomJS is a headless WebKit, scriptable with a JavaScript API
- SlimerJS (http://slimerjs.org/) - A scriptable browser for Web developers that runs on Gecko

## Template Engines

- ICanHaz.js (http://icanhazjs.com/)
- Hogan.js (http://twitter.github.com/hogan.js/)

- Handlebars.js (http://handlebarsjs.com/): Minimal Templating on Steroids
- Mustache (http://mustache.github.io/): Logic-less templates
- Transparency (http://leonidas.github.com/transparency/)
- doT.js (http://olado.github.com/doT/)
- Walrus (http://documentup.com/jeremyruppel/walrus/)
- Chibi (https://github.com/kylebarrow/chibi)
- Templayed.js (http://archan937.github.com/templayed.js/)
- ECT (http://ectjs.com/): Fastest JavaScript template engine with CoffeeScript syntax
- pithy (https://github.com/caolan/pithy): An internal DSL for generating HTML in JavaScript
- T (https://github.com/gcao/T.js): T.js is a template engine that uses simple Javascript data structure to represent html/xml data.
- Nunjucks (http://nunjucks.jlongster.com/) is a full featured templating engine for javascript.
- Jade (http://jade-lang.com/)
- Linked.in Dust.js (http://linkedin.github.com/dustjs/)
    - The client-side templating throwdown: mustache, handlebars, dust.js, and more (http://engineering.linkedin.com/frontend/client-side-templating-throwdown-mustache-handlebars-dustjs-and-more)
    - Leaving JSPs in the dust: moving LinkedIn to dust.js client-side templates (http://engineering.linkedin.com/frontend/leaving-jsps-dust-moving-linkedin-dustjs-client-side-templates)
    - Tutorial (https://github.com/linkedin/dustjs/wiki/Dust-Tutorial)
    - Testing Console (http://linkedin.github.com/dustjs/test/test.html)

## Routing And URLs

- Crossroads.js (http://millermedeiros.github.com/crossroads.js/)
- Hash.js (http://jonnystromberg.com/hash-js/)
- director (https://github.com/flatiron/director)
- Davis.js (http://davisjs.com/)
- path.js (http://mtrpcic.github.com/pathjs/)
- URI.js (http://medialize.github.com/URI.js/)
- Roadcrew.js (http://grobmeier.github.com/Roadcrew.js/)
- jQuery Address (http://www.asual.com/jquery/address/)
- page.js (https://github.com/visionmedia/page.js)
- speakingurl (https://github.com/pid/speakingurl): generate a so called "static", "Clean URL", "Pretty URL" or "slug" from a string.

## Rich Text Editors

- Mercury (http://jejacks0n.github.io/mercury/)
- Redactor (http://imperavi.com/redactor/)
- WYSIHTML5 (http://xing.github.com/wysihtml5/)
- MarkItUp (http://markitup.jaysalvat.com/)
- Hallo (http://hallojs.org/)
- Substance Composer (https://github.com/substance/composer)
- Proper (https://github.com/michael/proper)
- SnapEditor (http://snapeditor.com/)
- Create.js (http://createjs.org/)
- MDMagick (http://fguillen.github.com/MDMagick/)
- X-editable (http://vitalets.github.com/x-editable/)

## Code Viewers & Editors

- SublimeText (http://sublimetext.com/)
- Rainbow.js (http://craig.is/making/rainbows)
- Intelligist (http://srobbin.com/jquery-plugins/intelligist/)
- Prism.js (http://prismjs.com/)
- Brackets (https://github.com/adobe/brackets): An open source code editor for the web, written in JavaScript, HTML and CSS by Adobe.

- CodeMirror (http://codemirror.net/)
- Scripted (https://github.com/scripted-editor/scripted)
- tabIndent.js (http://julianlam.github.com/tabIndent.js/)
- Behave.js (http://jakiestfu.github.io/Behave.js/) is a lightweight library for adding IDE style behaviors to plain text areas, making it much more enjoyable to write code in.

## Refactoring

- Refactoring Javascript with kratko.js (http://perfectionkills.com/refactoring-javascript-with-kratko-js/)

## Performance

### Performance - General Information

- Video: DOM, HTML5, & CSS3 Performance (http://paulirish.com/2011/dom-html5-css3-performance/) by Paul Irish
- Video: High Performance Javascript (http://vimeo.com/16241085) by Nicholas Zakas
- Video: Building A Performant HTML5 App (http://youtu.be/ft9R72R7TII) with Trunal Bhanse and Akhilesh Gupta (LinkedIn)
- Writing Fast, Memory-Efficient JavaScript (http://coding.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript/)
- JavaScript Performance Best Practices (http://www.developer.nokia.com/Community/Wiki/JavaScript_Performance_Best_Practices) by Nokia
- Improving the performance of your HTML5 App (http://www.html5rocks.com/en/tutorials/speed/html5/)
- Best Practices for a Faster Web App with HTML5 (http://www.html5rocks.com/en/tutorials/speed/quick/)
- Front-end performance for web designers and front-end developers (http://csswizardry.com/2013/01/front-end-performance-for-web-designers-and-front-end-developers/)
- Let's Make the Web Jank-free! (http://jankfree.org/)

### Performance - Memory

- Memory 101 (https://developers.google.com/chrome-developer-tools/docs/memory-analysis-101)
- Memory leak patterns in JavaScript (https://www.ibm.com/developerworks/library/wa-memleak/)
- Understanding and Solving Internet Explorer Leak Patterns (http://msdn.microsoft.com/en-us/library/bb250448.aspx)
- Finding memory leaks (http://gent.ilcore.com/2011/08/finding-memory-leaks.html)
- How to write low garbage real-time Javascript (https://www.scirra.com/blog/76/how-to-write-low-garbage-real-time-javascript)
- JScript Memory Leaks (http://javascript.crockford.com/memory/leak.html)
- Tracking Down Memory Leaks in Node.js (https://hacks.mozilla.org/2012/11/tracking-down-memory-leaks-in-node-js-a-node-js-holiday-season/)
- Effectively Managing Memory at Gmail scale (http://www.html5rocks.com/en/tutorials/memory/effectivemanagement/)
- Writing Fast, Memory-Efficient JavaScript (http://coding.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript/)

### Performance - Tools

- jsPerf, the JavaScript performance playground (http://jsperf.com/)
- Chrome Developer Tools (https://developers.google.com/chrome-developer-tools/)
  - Heap Profiling (https://developers.google.com/chrome-developer-tools/docs/heap-profiling)
  - Profiles Panel (https://developers.google.com/chrome-developer-tools/docs/profiles)
  - JavaScript Profiling With The Chrome Developer Tools (http://coding.smashingmagazine.com/2012/06/12/javascript-profiling-chrome-developer-tools/)
  - Introduction to Chrome Developer Tools, Part One (http://www.html5rocks.com/en/tutorials/developertools/part1/)
  - Revolutions 2013 (http://www.html5rocks.com/en/tutorials/developertools/revolutions2013/)
- V8
  - Optimizing for V8 (http://floitsch.blogspot.co.uk/search/label/V8) - very technical series on the V8 engine
  - I-want-to-optimize-my-JS-application-on-V8 checklist (http://mrale.ph/blog/2011/12/18/v8-optimization-checklist.html)
  - Performance tips for JavaScript in V8 (http://www.html5rocks.com/en/tutorials/speed/v8/)
- JavaScript Leak Finder (http://code.google.com/p/leak-finder-for-javascript/)
  - Introduction Article (http://google-opensource.blogspot.de/2012/08/leak-finder-new-tool-for-javascript.html)
- Navigation Timing (http://www.html5rocks.com/en/tutorials/webperformance/basics/)
- Firebug Paint Events (https://addons.mozilla.org/en-US/firefox/addon/firebug-paint-events/)
- Locache (https://github.com/d0ugal/locache): JavaScript caching framework for client side caching in the browser using localStorage
- Caliper (http://caliper.io/) is a web service for performance monitoring of your web app.

### Performance - DOM Manipulation

- Rendering: repaint, reflow/relayout, restyle (http://www.phpied.com/rendering-repaint-reflowrelayout-restyle/)

- Reflows & Repaints: CSS Performance making your JavaScript slow? (http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/)
- The new game show: "Will it reflow?" (http://calendar.perfplanet.com/2009/the-new-game-show-will-it-reflow/)
- When does reflow happen in a DOM environment? (http://stackoverflow.com/questions/510213/when-does-reflow-happen-in-a-dom-environment)
- Speeding up JavaScript: Working with the DOM (https://developers.google.com/speed/articles/javascript-dom)
- Efficient JavaScript (http://dev.opera.com/articles/view/efficient-javascript/) on dev.opera.com
- When Does JavaScript trigger reflows and rendering (http://mir.aculo.us/2010/08/17/when-does-javascript-trigger-reflows-and-rendering/)
- How (not) to trigger a layout in WebKit (http://gent.ilcore.com/2011/03/how-not-to-trigger-layout-in-webkit.html)

### Performance - Animation

- requestAnimationFrame for smart animating (http://paulirish.com/2011/requestanimationframe-for-smart-animating/)
- Leaner, Meaner, Faster Animations with requestAnimationFrame (http://www.html5rocks.com/en/tutorials/speed/animations/)
- Collie - high performance Animation library (http://jindo.dev.naver.com/collie/)
- Using requestAnimationFrame to Optimize Dragging Events (http://blog.digitalbackcountry.com/2012/05/using-requestanimationframe-to-optimize-dragging-events/)
- requestAnimationFrame API: now with Sub-Millisecond Precision (http://updates.html5rocks.com/2012/05/requestAnimationFrame-API-now-with-sub-millisecond-precision)
- Why moving elements with translate() is better than pos:abs top/left (http://paulirish.com/2012/why-moving-elements-with-translate-is-better-than-posabs-topleft/)

### Performance - Hardware Acceleration

- Visualizing WebKit's hardware acceleration (http://mir.aculo.us/2011/02/08/visualizing-webkits-hardware-acceleration/)
- List of Chromium Command Line Switches (http://peter.sh/experiments/chromium-command-line-switches/)

### Performance - Browser Internals

- Video: Google I/O 2012 - Breaking the JavaScript Speed Limit with V8 (http://www.youtube.com/watch?v=UJPdhx5zTaw) by Daniel Clifford
- Video: Faster HTML and CSS: Layout Engine Internals for Web Developers (http://www.youtube.com/watch?v=a2_6bGNZ7bA): by David Baron of Mozilla
- How Browsers Work: Behind the scenes of modern web browsers (http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/)
- How Browsers Work - Part 1 - Architecture (http://www.vineetgupta.com/2010/11/how-browsers-work-part-1-architecture/)
- Know Your Engines at O'Reilly Velocity 2011 (https://blog.mozilla.org/dmandelin/2011/06/16/know-your-engines-at-oreilly-velocity-2011/)
- Video: GDC 2012: From Console to Chrome (http://www.youtube.com/watch?v=XAqIpGU8ZZk)
- Fast CSS: How Browsers Lay Out Web Pages (http://dbaron.org/talks/2012-03-11-sxsw/master.xhtml)
- Video: What Browsers Really Think of Your App (http://dayofjs.com/videos/22158462/web-browsers_alex-russel)
- WebKit blog five-part series on rendering (https://www.webkit.org/blog/114/webcore-rendering-i-the-basics/)
- Video: Alex Russell - Life Of A Button Element (http://vimeo.com/32364192)
- How a web page loads (http://gent.ilcore.com/2011/05/how-web-page-loads.html)
- Video: The Fundamentals, Primitives and History of HTML5 (http://paulirish.com/2011/primitives-html5-video/)
- The JavaScript engine family tree (http://creativejs.com/2013/06/the-race-for-speed-part-1-the-javascript-engine-family-tree/)
- How JavaScript compilers work (http://creativejs.com/2013/06/the-race-for-speed-part-2-how-javascript-compilers-work/)
- JavaScript compiler strategies (http://creativejs.com/2013/06/the-race-for-speed-part-3-javascript-compiler-strategies/)
- The future for JavaScript (http://creativejs.com/2013/06/the-race-for-speed-part-4-the-future-for-javascript/)

### Performance - Images

- CSS-only: Load images on demand (http://podlipensky.com/2013/06/css-only-load-images-on-demand/)
- Clever JPEG Optimization Techniques (http://www.smashingmagazine.com/2009/07/01/clever-jpeg-optimization-techniques/)
- Clever PNG Optimization Techniques (http://www.smashingmagazine.com/2009/07/15/clever-png-optimization-techniques/)
- How to Optimize PNG and JPEG without Quality Loss (http://www.splashnology.com/article/how-to-optimize-png-and-jpeg-without-quality-loss-part-1/2071/) + Part 2 (http://www.splashnology.com/article/how-to-optimize-png-and-jpeg-without-quality-loss-part-2/2568/)

### Performance - HTTP Transport

- Fantastic front-end performance: Concatenate, Compress & Cache (https://hacks.mozilla.org/2012/12/fantastic-front-end-performance-part-1-concatenate-compress-cache-a-node-js-holiday-season-part-4/)
- Video: Cache Is King! (http://www.youtube.com/watch?v=HKNZ-tQQnSY) by Steve Souders
- Best Practices for Speeding Up Your Web Site (http://developer.yahoo.com/performance/rules.html) by Yahoo

## Modularity & Loaders

- Lazy Module Declaration (http://lmdjs.org/): JavaScript Module-Assembler for building better web applications

- UMD (Universal Module Definition) patterns (https://github.com/umdjs/umd)
- Browserify (http://browserify.org/)
- require.js (http://requirejs.org/)
  - Plugins (https://github.com/jrburke/requirejs/wiki/Plugins)
- sea.js (http://seajs.org/) A Module Loader for the Web
- CrapLoader (https://github.com/gregersrygg/crapLoader)
- Writing Modular JavaScript With AMD, CommonJS & ES Harmony (http://addyosmani.com/writing-modular-js/)
- Путь JavaScript модуля (http://habrahabr.ru/post/181536/)
- Modular JavaScript with RequireJS (http://blog.credera.com/topic/technology-solutions/java/modular-javascript-with-requirejs/)
- LABjs & RequireJS: Loading JavaScript Resources the Fun Way (http://msdn.microsoft.com/en-us/magazine//ff943568.aspx)
- JavaScript Inheritance and Public/Private members (https://go-left.com/blog/2011/08/js-inheritance/)
- Deep dive into the murky waters of script loading (http://www.html5rocks.com/en/tutorials/speed/script-loading/)
- Terminology: Modules (http://dailyjs.com/2013/05/20/terminology-modules/)

## Package Management

- bower (http://bower.io/)
- volo (https://github.com/volojs/volo)
- parcel (https://github.com/sifteo/parcel) is easy package management using a file server and path conventions, with built in support for Amazon S3. It is designed to encode package metadata, including name, version and OS within a path. The conventions allow this metadata to be queried, without the need for a database.
- jam.js (http://jamjs.org/)
- component (http://component.io/)

## Image Manipulation

- Paintbrush.js (https://github.com/mezzoblue/PaintbrushJS)
- Obscura (https://github.com/OiNutter/Obscura)
- jCrop (http://deepliquid.com/content/Jcrop.html)
- Pixastic (http://www.pixastic.com/)

## HTTP

- Specification (http://www.w3.org/Protocols/rfc2616/rfc2616.html)
- Using CORS (http://www.html5rocks.com/en/tutorials/cors/)
- Cross-domain Ajax with Cross-Origin Resource Sharing (http://www.nczonline.net/blog/2010/05/25/cross-domain-ajax-with-cross-origin-resource-sharing/)

## WebSocket

- Specification (http://tools.ietf.org/html/rfc6455)
- socket.io (http://socket.io/) aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms.
- engine.io (https://github.com/LearnBoost/engine.io) is the implementation of transport-based cross-browser/cross-device bi-directional communication layer for Socket.IO.
- SockJS (https://github.com/sockjs/sockjs-client) is a browser JavaScript library that provides a WebSocket-like object. SockJS gives you a coherent, cross-browser, Javascript API which creates a low latency, full duplex, cross-domain communication channel between the browser and the web server.

## Error Handling & Debugging

- Tracing.js (https://github.com/ebobby/tracing.js) - Javascript function tracing.
- JavaScript error handling anti-pattern (http://www.nczonline.net/blog/2009/04/28/javascript-error-handling-anti-pattern/)
- Tattletale (https://github.com/vimeo/tattletale)

## Documentation

- DevDocs (http://devdocs.io/) is an all-in-one API documentation reader with a fast, organized, and consistent interface.
- dexy (http://www.dexy.it/) is a free-form literate documentation tool for writing any kind of technical document incorporating code.
- docco (http://jashkenas.github.com/docco/) is a quick-and-dirty, hundred-line-long, literate-programming-style documentation generator.
- styledocco (http://jacobrask.github.io/styledocco/) generates documentation and style guide documents from your stylesheets.
- Ronn (https://github.com/rtomayko/ronn) builds manuals. It converts simple, human readable textfiles to roff for terminal display, and also to HTML for the web.
- dox (https://github.com/visionmedia/dox) is a JavaScript documentation generator written with node. Dox no longer generates an opinionated structure or style for your docs, it simply

gives you a JSON representation, allowing you to use markdown and JSDoc-style tags.

- YUIDoc (http://yui.github.com/yuidoc/) is a Node.js application that generates API documentation from comments in source, using a syntax similar to tools like Javadoc and Doxygen.
- coddoc (http://doug-martin.github.com/coddoc/) is a jsdoc parsing library. Coddoc is different in that it is easily extensible by allowing users to add tag and code parsers through the use of coddoc.addTagHandler and coddoc.addCodeHandler. coddoc also parses source code to be used in APIs.
- sphinx (http://sphinx-doc.org/) a tool that makes it easy to create intelligent and beautiful documentation
- Using JSDoc (http://usejsdoc.org/)

## Windows 8 Style

- Metro UI CSS (http://metroui.org.ua/)
- BootMetro (http://aozora.github.com/bootmetro/)
- metro-bootstrap (http://talkslab.github.com/metro-bootstrap/)
- Metro UI Template (http://metro-webdesign.info/)
- Droptiles – Metro-Style Web Dashboard (http://www.codeproject.com/Articles/421601/Droptiles-Metro-style-Live-Tiles-powered-Web-Dashb)
- Metro JS (http://www.drewgreenwell.com/projects/metrojs)
- Windows Metro tiles with Javascript and CSS3 (https://github.com/tholman/tileJs)
- Cosmo (http://bootswatch.com/cosmo/)
- Video: What Web Developers Need to Know When Building Metro style Apps (http://channel9.msdn.com/Events/TechEd/NorthAmerica/2012/DEV352)
- CSS3 Microsoft Modern Buttons (http://ace-subido.github.com/css3-microsoft-metro-buttons/index.html)
- Visual Studio 2012 Image Library - 5,000+ Images Downloadable Now (http://dotnet.dzone.com/articles/visual-studio-2012-image)
- Metro User Interface Implementation and Resources (http://www.queness.com/post/13493/metro-user-interface-implementation-and-resources)
- Windows-8-like animations with CSS3 and jQuery (http://sarasoueidan.com/blog/windows8-animations): a demo made by Sara Soueidan about creating a Modern UI design like in Windows 8.

## Video & Audio

- SublimeVideo (http://sublimevideo.net/): Modular Video Player Framework.
- Video.js (http://www.videojs.com/): The open source HTML5 video player.
- FlowPlayer (http://flowplayer.org/): The video player for the web.
- longtailVideo (http://www.longtailvideo.com/): dual-mode HTML5 & Flash player.
- howler.js (http://goldfirestudios.com/blog/104/howler.js-Modern-Web-Audio-Javascript-Library)
- Stratus 2 (http://stratus.sc/) soundcloud player
- jPlayer (http://jplayer.org/) is the completely free and open source (MIT) media library written in JavaScript.
- MediaElement.js (http://mediaelementjs.com/): HTML5 audio and video players in pure HTML and CSS.
- audio.js (http://kolber.github.io/audiojs/) is a drop-in javascript library that allows HTML5's "audio" tag to be used anywhere.

## Library Collections

- Unheap (http://www.unheap.com/): A tidy repository of jQuery plugins
- jster (http://jster.net/)
- MicroJS (http://microjs.com/) helps you discover the most compact-but-powerful microframeworks.
- JSPkg (http://jspkg.com/): reference library and package manager
- jquer.in (http://jquer.in/) is a curated collection of jQuery plugins.

## Other Front End Collections

- pineapple.io (http://pineapple.io/): A central hub of Tutorials, Tools and Assets for developers and designers
- setapp Front End (http://setapp.me/expertise/development-front_end): Discover apps, tools, news and tips
- Frontdesk (https://github.com/miripiruni/frontdesk): Community driven list of useful things for Front End Developers
- Front-end Code Standards & Best Practices (http://isobar-idev.github.io/code-standards/) by Roundarch Isobar
- Devlinks (http://iwantaneff.in/link/) by David Higgins

## Chat

- XMPP using JSON (https://xmpp-ftw.jit.su/)
- Converse.js (http://conversejs.org/) is an open source, webchat client, that runs in the browser and can be integrated into any website.

## Prototyping

- RoughDraft.js (https://github.com/ndreckshage/roughdraft.js)
- Video: Style guides are the new Photoshop by Stephen Hay (http://vimeo.com/52851510) + Slides (http://de.slideshare.net/stephenhay/style-guides-are-the-new-photoshop-fronteers-2012)
- fixie (https://github.com/ryhan/fixie)
- Holder.js (http://imsky.github.com/holder/)

## JSON

- JSON Query Language (http://jsoniq.org/) is a query and processing language specifically designed for the popular JSON data model.
- Oboe.js (https://github.com/jimhigson/oboe.js) makes it really easy to start using json from a response before the ajax request completes; or even if it never completes.

## Presentations

- Ancesor.js (http://kirkas.ch/ascensor/)
- Slideshow.html (http://dmfrancisco.github.com/slideshow.html/)
- HTML5 Slideshow (http://www.ravelrumba.com/blog/html5-slideshow/)
- jQuery Screen (https://github.com/xiam/jquery-screen)
- reveal.js (https://github.com/hakimel/reveal.js)
- html5 Slides by Google (https://code.google.com/p/io-2012-slides/)
- Slides (https://github.com/briancavalier/slides)
- CSSS (http://lea.verou.me/csss/sample-slideshow.html)
- deck.js (http://imakewebthings.com/deck.js/)
- controldeck.js (http://controldeck.aws.af.cm/)
- impress.js (http://bartaz.github.io/impress.js/)

## Scrolling

- AppScroll.js (https://github.com/jakiestfu/AppScroll.js) AppScroll is a tiny (< 100 lines) JS library that fixes the "website dragging" issue caused on iDevices
- Scrollpath (http://joelb.me/scrollpath/)
- jQuery NiceScroll (http://code.google.com/p/jquery-nicescroll/)
- FoldScroll (http://soulwire.github.com/FoldScroll/)
- Scrollorama (http://johnpolacek.github.com/scrollorama/)

## Keyboard

- Mousetrap (http://craig.is/killing/mice)
- Keypress (http://dmauro.github.com/Keypress/)
- konami.js (http://mckamey.github.com/konami-js/)
- Keys.js (https://github.com/bitwalker/keys.js)
- keymage.js (https://github.com/piranha/keymage)

## Tables & DataGrids

- Tablecloth.js (http://tableclothjs.com/)
- Responsive Data Tables (http://css-tricks.com/responsive-data-tables/)
- stacktable.js (http://johnpolacek.github.com/stacktable.js/)
- FooTable (http://themergency.com/footable/)
- Handsontable (http://warpech.github.com/jquery-handsontable/index.html)
- Backgrid.js (http://wyuenho.github.com/backgrid)
- Gridster (http://gridster.net/)
- Flex (http://jsonenglish.com/projects/flex/)

## Security

- oAuth Consumer Library (http://oauth.googlecode.com/svn/code/javascript/) by John Kristian
- ohauth (https://github.com/tmcw/ohauth)

    - How iD Logs in to OpenStreetMap (http://mapbox.com/osmdev/2013/01/15/oauth-in-javascript/)
- JSO - a Javascript OAuth Library (https://github.com/andreassolberg/jso)
- xdoauthproxy (https://code.google.com/p/xdoauthproxy/)
- JavaScript Cryptography (http://disattention.com/13/javascript-cryptography/)
- Javascript Crypto Library (http://www.clipperz.com/open_source/javascript_crypto_library)
- Stanford Javascript Crypto Library (http://crypto.stanford.edu/sjcl/)
- mozilla / jwcrypto (https://github.com/mozilla/jwcrypto)
- Critic
    - Javascript Cryptography Considered Harmful (http://www.matasano.com/articles/javascript-cryptography/)
    - Final post on Javascript crypto (http://rdist.root.org/2010/11/29/final-post-on-javascript-crypto/)

## Layout

- Shapeshift (https://github.com/McPants/jquery.shapeshift) is a plugin which will dynamically arrange a collection of elements into a column grid system similar to Pinterest.
- CollagePlus (http://ed-lea.github.io/jquery-collagePlus/): This plugin for jQuery will arrange your images to fit exactly within a container.
- Masonry (http://masonry.desandro.com/) JavaScript Cascading grid layout library

## Jobs & Hiring

- GitHub Jobs (https://jobs.github.com/)
- Authentic Jobs (http://www.authenticjobs.com/)
- 37signals Job Board (http://jobs.37signals.com/)
- Smashing Jobs (http://jobs.smashingmagazine.com/)
- Front-end Developer Interview Questions (https://github.com/darcyclarke/Front-end-Developer-Interview-Questions)
- Interview This! (https://github.com/ChiperSoft/InterviewThis): A list of questions to ask employers.
- Code problems (https://github.com/blakeembrey/code-problems) -- Common code problems solved using JavaScript

## Misc

- zip.js (http://gildas-lormeau.github.com/zip.js/)
- simpleCart.js (http://simplecartjs.org/)
- Aware.js (http://xoxco.com/projects/code/aware/) is a simple jQuery plugin that allows a site to customize and personalize the display of content based on a reader's behavior without requiring login, authentication, or any server-side processing.
- How to directly upload files to Amazon S3 from your client side web app (http://codeartists.com/post/36892733572/how-to-directly-upload-files-to-amazon-s3-from-your)
- jIO (https://www.j-io.org/) is a client-side JavaScript library to manage documents across multiple storages.
- Lo-Dash (http://lodash.com/) - A low-level utility library delivering consistency, customization, performance, and extra features.

# Sources

- isobar's code standards (http://isobar-idev.github.io/code-standards/)
- taitems's front end development guidelines (http://taitems.github.io/Front-End-Development-Guidelines/)
- bootstrap's contributing guidelines (https://github.com/twbs/bootstrap/blob/master/CONTRIBUTING.md)
- dypsilon's frontend dev bookmarks (https://github.com/dypsilon/frontend-dev-bookmarks/)

# Authors & Contributors

- Paul Irish (https://twitter.com/paul_irish)
- Nick Cooley (https://twitter.com/nickcooley)
- Adam McIntyre (https://twitter.com/_ajm)
- Rob Larsen (http://htmlcssjavascript.com/)
- Joel Oliviera (https://twitter.com/jayroh)
- Jared Williams (https://twitter.com/jaredwilli)
- Annette Arabasz (https://twitter.com/annettish)
- Rob Cherny (https://twitter.com/rcherny)
- Roundarch Isobar (http://www.roundarchisobar.com/)
- Tait Brown (https://twitter.com/taitems)

- Dimi Navrotskyy (https://twitter.com/dypsilon)
- Pierre Cholhot (https://twitter.com/pierrecholhot)