Front-end best practices for web development

GOALS
maintenance, consistency, quality, education

HTML

HTML is used to organize the dom. Stray away from mixing HTML with CSS. It is best practice to not style HTML markup, with exception of headings. One way of looking at it, HTML is the skeleton (structural layer),CSS is the skin (presentation layer).

META TAGS
Always use the following meta tags in the HTML <head>
```
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="HandheldFriendly" content="True">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Use the MobileOptimized meta tag for if your site links to a separate mobile site.
```
<meta name="MobileOptimized" content="width">
```

LAYOUT TAGS
```
<header><nav><main><section><aside><article><figure><picture><footer>
```

Keep markup semantic using HTML5 tags to increase readability throughout the code.

For older browsers, such as IE8, use normalize.css (https://necolas.github.io/normalize.css/) which will render all elements appropriately.

MEDIA TAGS
`<audio>` and `<video>` older IE browsers do not support these tags so you should consider applying polyfills for such browsers.

Modern browsers support MP4 filetype, for those that do not, use OGG and and WEBM as fallbacks.

HEADINGS TAGS

<h1> <h2> <h3> <h4> <h5> <h6>

Be mindful of SEO's use of HTML headings as these tags are used to enhance search results.

CSS

BEST PRACTICES

1. Avoid targeting HTML elements unless it's headings or nested span tags
2. Never target grid classes ex: col-sm-12
3. Classes must be semantic, informative and clear.
4. Avoid using ID's whenever possible
5. Avoid !importants. A constant need of !important is an indicator of poor selector handling.
6. Follow Naming conventions
7. Always apply fallbacks (preferably through mixins)
8. Utilize inline-block whenever possible, avoid floats (except of course when they're applied by a grid framework). If float is a must, make sure to CLEAR the float before and after it's usage.
9. Reset your CSS to avoid browser default CSS
10. Recommended to have unitless line heights such as line-height: 1;

NAMING CONVENTIONS

.block-element__modifier

BEM (which stands for Block-Element-Modifier) is a naming convention commonly used to keep classes semantic, consistent, reusable, and to help reduce specificity. There are other naming conventions such as SMACS or OOCSS to achieve this technique; the only difference is the naming structure.

Prefix CSS classes that are utilized through JS with js-

SPECIFICITY

Avoid specific targeting in CSS, unless absolutely necessary, such as
body .layout .element {}
body > .layout > .element

CSS PREPROCESSORS

CSS preprocessors are specifically important and helpful when developing a large-scale site. However, it is not a requirement.

MEDIA QUERIES FOR MOBILE/HIGH RESOLUTION

Stick to 3 media queries that represent values close to most small, medium and large displays.

$xsmall-screen = 380px
$small-screen = 480px
$medium-screen = 768px
$large-screen = 1080px
$xlarge-screen = 2200px
These values can change depending on project

SASS or LESS

SILENT HELPERS

Strongly recommended to use silent helpers as it supports reuse.

RULES

1. Never nest #ID's
2. Avoid heavy nesting, 3 layers max
3. Extend typography styles using silent classes
4. Make use of mixins and silent classes when code is repeated.
5. All media breakpoints should be variables.
6. REM > EM > PX

ORGANIZE

1. Organize files based on silent helpers, core, components, detection (browser/device/feature detection)
2. Keep all global variables inside _var.scss sheet.
3. Keep unique variables at very top of its component-specific sheet and include component name first,
   ex: in blocks/_teasers.scss
   $teaser_left-padding: 20px;

Javascript

THIRD PARTY LIBRARIES
jQuery, underscore

NAMING CONVENTION
Prefix functions with ini_ when initializing.

COMMENTS
Keep comments styles consistent and avoid writing more than 2 lines of comments other than those that are at the root of the file.

Accessibility

The goal of Web Accessibility is to ensure the site's content and it's functionality can be accessible to the general public, including those with disabilities.

WCAG PRINCIPLES

- **Perceivable**: Can users perceive the content? This helps us keep in mind that just because something is perceivable with one sense, such as sight, that doesn't mean that all users can perceive it.
- **Operable**: Can users use UI components and navigate the content? For example, something that requires a hover interaction cannot be operated by someone who can't use a mouse or touch screen.
- **Understandable**: Can users understand the content? Can users understand the interface and is it consistent enough to avoid confusion?
- **Robust**: Can the content be consumed by a wide variety of user agents (browsers)? Does it work with assistive technology?

TYPES OF IMPAIRMENTS

- Visual
- Motor
- Hearing
- Cognitive

PRACTICAL WAYS OF INCORPORATING WEB ACCESSIBILITY

- **Styling**: Incorporating enough white space, creating a hierarchy in color and typography to make content more readable in chunks.

- **Focus**: include focus styling for input fields, include variation in states (disabled, active, focus, visited), use breadcrumbs to help guide user

Browser Support

INTERNET EXPLORER

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
Prevent IE from switching modes. This tag helps override IE compatibility mode.

caniuse.com

Great resource to always reference as you're applying unfamiliar HTML/CSS code. Only use code that is supported by the browser that you must develop for. Keep in mind project contracts that support may support specific projects.

FEATURE DETECTION

A good practice is to use feature detection to fix any inconsistency among browsers. Tools such as Modernizr (modernizr.com) does the job for you, Modernizr tells you what HTML, CSS and JavaScript features the user's browser has to offer. The most important detection would be *touch/no-touch* to determine whether the browser is being rendered through a touchscreen.

Load Time

REDUCE HTTP REQUESTS

When a user requests a webpage, the browser has to download and perform code (specifically Javascript code) to render the page. This process requires the browser to load not only the DOM, but also retrieve its assets such as images, css and javascript file, and other external sources embedded in the head or footer of the DOM.

There are a number of ways to help reduce the amount of calls the browser has to make to render the webpage you've requested.

Minify CSS & JS files – Minified files not only weight less than un-minified versions but they also reduce the number of requests, given that you combined separate files into 1.

Sprites – Combining multiple images into a single image file is called a Sprite. This practice is used to help with performance, as the browser only has to render 1 image as opposed to multiple. You can manipulate the image crop as an <img> by setting an overflow hidden to it's parent wrapper, or as a background image by targeting the background-position.

Images

ALT TEXT

Important for SEO and accessibility, always add an alt attribute to images.

SVG's

The benefit of using SVG images is to retain the quality of vector based images. Use SVG's for LOGO's (with PNG image as fallback for older browsers).

Embeded SVG – Embedding SVG images into HTML requires opening up the SVG image file using a text editor. You'll notice XML code, which is capable of getting rendered through a browser to display the image. That code can be placed inside your HTML file and referenced via XML.

Best practice is to combine all SVG's into 1 file, properly segmenting them into their own <symbol>. Apply a unique ID to each symbol and reference them in your HTML markup as so:

Embedding is the best way of using utilizing SVG's as you have the capability of changing the color of the SVG using CSS property *fill*.

<img> & Background-image SVG

SVG's may also be applied, you can still stretch out the image while retaining the quality but will not have the ability to change color. SVG is not practical for photographs but is ideal for logos, diagrams and charts. The primary drawback is a lack of support in IE8 and below but you could always provide a PNG fallback or use a shim such as Raphaël or svgweb.