**Mobile Web Development Takeaway**

When I began learning about Web Development last semester, my mind was a clean slate. I had no prior knowledge, no expectations, and no understanding of how things work ‘on the web’. From then to now, I’ve learned so much useful information about developing websites. As this reflection is about what I’ve learned in Mobile Web Development, I’ll stray away from what I learned during Internet Technologies with you, but in both, I’ve learned a ton.

Firstly, this semester in Mobile Web Development, I learned what the “mobile first” approach is. In essence, the mobile first approach is beginning our sites layout and design with the mobile user in mind. We start with styling for the little screen users and work our way up to the big screen users. Aside from sizing for the mobile first approach, we also use a similar process for features. This process is called ‘progressive enhancement’. In progressive enhancement, we build our site up with styles and features that all users can access, rather than strip things off at the end to suit most users (graceful degradation). While the two approaches are debated about, I prefer the progressive enhancement approach rather than graceful degradation, because it seems much more intuitive to add supported features as I go, instead of strip things off at the end that aren’t supported.

Secondly, I learned about media queries this semester. Media queries are probably one of the most useful things a web developer could learn since they allow you to layout your content in different ways based on specific properties. For example, in a mobile first approach we can start off designing our main styles to be viewed on a mobile phone screen, then use media queries based on a ‘min-width’ query to change the style to be more appropriate for a tablet or a desktop computer. The most powerful thing about media queries is the range of things that can be queried. You can query off height, width, pixel density, orientation, and much more.

Thirdly, I learned about Flexbox and CSS Grid. Flexbox is a display style, mainly focused on elements in a row, that allows you to change the horizontal and vertical alignment based on the parent container. CSS Grid is like Flexbox, but better suited for several rows of content. While you can accomplish virtually the same things with Flexbox, it’s much easier and more convenient to use CSS Grid to set the layout for your content if there is more than one row to be styled. One downside to CSS Grid is that it’s not as widely supported as Flexbox, however most browsers at this point support it with the obvious exception of IE.

Another thing I learned this semester is what microdata is and how to implement it. Microdata is specifically styled content in the form of Schema or JSON LD that can be parsed by search engine ‘crawlers’ to better piece together the content of your site. An easy way to see how microdata is implemented is by Googling something and looking at the top right corner of the search results page. It’s common today to see microdata being used on almost everything I Google. Schema microdata is implemented via inline styling in the HTML document whereas JSON LD is implemented in the form of a script (usually linked to but can be embedded as well). The advantage to using Schema is that it’s easier to see what content is being tagged, but it clutters the document a bit, while JSON LD can be linked to in an external JavaScript document. Having external microdata can be nice for working on building out your site, but it can easily be overlooked by others (non-crawlers).

Finally, I learned about responsive images, HTML5 native video, and HTML5 native audio. Responsive images are accomplished by setting images max-width to 100% so that they are fluid, then using the picture element along with media queries to tell the browser to display a specifically sized image based on the queried condition. This implementation requires using srcset’s and the standard img tag inside the picture tags. One downside to having responsive images on your site is that it requires several copies of the image at different sizes which can be a bit taxing on browser performance, but it’s definitely worth having responsive images! Another small problem with responsive images is that not all browsers provide support for them yet, but this can usually be overcame pretty easily with ‘picturefill’. HTML5 native audio and video are very similar, so I’ll speak about them both as one ‘thing’. The plus to native audio/video is that by using the ‘controls’ tag in the element, most browsers will display a default player. In the chance that a browser doesn’t support HTML5 native audio/video, the swf player can easily be used as a polyfill. Swf player is a fallback player that has been created by adobe flash player and requires the user to allow/have flash player installed.

In conclusion, the above are just a few of the numerous things I have learned from Mobile Web Development. My understanding for how things work, and why we do things the way we do this semester has skyrocketed. I’ve really enjoyed learning more about Web Development and taking your classes has been a tremendous help! Thank you for another great semester!!