"Web Design DeCal" Hands-On Session 8 (Design)
Responsive Web Design

In this week's hands-on session, you will be using CSS Media Query and JQuery IScroll to create layouts that render differently depending on the browser's width.

I. Media Query

Take a look at the CSS of **media.html** via web inspector. You can see that we have a cover on the left and the content on the right, implemented via position: absolute. cover has a width 30% and the content has margin-left:30% to make room for the cover. For smaller browser width, this will not work. (Try it!)

For smaller browser width, we would have to ditch the horizontal stack layout for a vertical stack layout.

A. First, we declare a CSS media query so that we write the new CSS rules, they only get applied to specified browser width. We want to apply these new CSS rules when the browser width is small, so write:

   @media (max-width: 1000px) {

   }

   This will make sure that all CSS rules inside this declaration will work for browser widths up to 1000px.

B. Recall that to make a vertical stack, you simply stack divs using position: static (default). Thus, we write "position: static; width: 100%; height: 300px;" for #cover and "position:static" for #answers, inside the media query block. Recall that the latter CSS rule overwrites the former CSS rule, so this new CSS rule will overwrite the original CSS rules defined.

   Question: what would happen if we declare the media query in the beginning of the CSS document?

C. Now try decreasing the browser width. You will see that the cover will stack vertically, since the media query rule is applied. If you increase the browser width, the media query will be removed, so the layout reverts back to a horizontal stack.

II. Developing a full mobile optimized site via JQuery iOSSlider

CSS Media Query has its limitations – it works fine as long as you stick to the original layout of the site. That is, as long as you are willing to not change the HTML layout, it

works perfectly. However, for some cases, you need to make a full-fledged mobile layout that requires an entirely new HTML layout.

In this tutorial, you will be making a paginated mobile website using JQuery iOSSlider plugin. You will also learn how to use external plugins and reading documentations, which is essential in front-end web development.

In assets folder, we have included JQuery iOSSlider plugin. The documentation is available on this website: https://iosscripts.com/iosslider-vertical/

A. Open up **mobile.html** in your web browser. It works fine as is, but we want to make it look more like a mobile app by making it paginated. (Refer to mobile_solution folder for a demo.) Most of the CSS styles are already completed for you. Here, we will be following the iOSSlider installation guide from the documentation. The first step is to include the link to the plugin by adding this code right above script.js

   <script src="assets/jquery.iosslider-vertical.js"></script>

   What would happen if you place this code above JQuery import?

B. In the documentation, you structure the HTML so that you have a viewport (with class iosSliderVertical), the "slider" inside a viewport, and each page represented as "slides." In our HTML, we have a similar layout, except that we do not have the matching class attributes as what's required.

   Add iosSliderVertical class to #viewport, slider class to #page-slider, and add slide class to .page. Remember that you can have multiple classes by having class="page slide"

C. Following the documentation, add the following CSS code from the documentation to our CSS code (**style_mobile.css**). Change <slide width> and <slide height> to 100%, since the size of our slides is full-screen.

D. Finally, add the following JQuery code to script.js

```
$('.iosSliderVertical'). iosSliderVertical({

  snapToChildren: true,
  desktopClickDrag: true,
  elasticPullResistance: 0.0,
  elasticFrictionCoefficient: 0.3

});
```

Now, try running the index.html and dragging around the pages!