Week 4 codify academy

**Week 4**

**In Class Quiz**

1. What do media queries do?
2. In relation to media queries, what does max-width do?
3. In relation to media queries, what does min-width do?
4. What does the meta viewport tag do?
5. Why is responsive design important?

**Group Discussion Questions**

1. Has everybody started adding coding skills to your LinkedIn profiles?
2. Has everybody followed at least four new companies on LinkedIn this week?
3. Why is responsive design important?
4. What are the three types of devices on which consumers visit websites?
5. At least one of these devices has two different viewing modes. What are they?
6. What does the meta viewport element do?
7. How do you view Chrome’s mobile developer tools?
8. What is the difference between the max-width CSS property and max-width in a media query?
9. What is the difference between min-width and max-width in a media query?
10. When should a developer choose to use min-width instead of max-width in a media query, or vice versa?
11. What does display: none do? Why might we use it inside of media queries?
12. What is the opposite of display: none? Why might we use it inside of media queries?
13. What is the maximum number of media queries you should use per website?
14. Should you create a separate CSS file for media queries? Where should you write your media queries?
15. What type of code goes inside of media queries?
16. If you write a media query for your website and find that your code is not responding the way you expect it to, what is the very first thing you should check when troubleshooting? What is the second? What is the third?
17. How do we include animate.css in our HTML file?
18. How do we animate an element using animate.css?
19. What additional class can we add to an element to make an animate.css animation loop forever?

**Exercises**

***Important: Make sure to add comments to your code.***

1. Create a copy of your template folder and rename it to week4exercise.
2. Drag this new folder into your 'week4' folder inside your 'codify' folder.
3. Open Github and your terminal/Git Bash. Push your empty week3exercise project folder to your new Github repository before we begin. Reference [this guide](https://files.slack.com/files-pri/T039N7TM0-F8JSX6CAV/new_github_repository_walkthrough.txt" \t "_blank) for pushing your project folder to your new repository.
4. In HTML: Create 4 div elements. Each with a class of box and a second class. The second class can be any color you want - we will set the background-color of each div to the color of the second class.  
   ex: <div class="box red"></div>
5. In CSS: Select the body element and add text-align center as a property and value.
6. In CSS: Select the class of box and add 3 properties display, height & width (Hint: we want our boxes to sit next to each other, all in one line, so think carefully about what values you should use!).
7. In CSS: Select the second class of each div (they should be different class names) and add a background color to each selector each with a different color.  
   ex: .red { background-color: red; }
8. In CSS: Using media queries make it so that the 2nd and 3rd div elements disappear when you SHRINK the screen size below 600px.
9. In CSS: Using media queries make it so that all of the divs have their width extended to 100% of the screen when you INCREASE the screen size beyond 1000px.
10. Connect animate.css to your index.html if it is not already.
11. Add a new div with a class of container, and give it a height and width of 200px, and a background-color.
12. Give this new div a class of “animated fadeOut” and refresh your page. Watch the div fade out. If the animation is not running, make sure animate.css is connected to your HTML either via CDN, or with a local file path to the animate.css folder inside your project folder.
13. Now open your inspector, click on the “select an element” button in the top left corner, hover over the space where the div had been visible until its outline is highlighted, and click on it. You should see “<div class=”animated fadeOut”></div>” highlighted in your elements tab.
14. Examine your styles tab and try to see how/why the div disappeared. In other words, what CSS property was applied to make the div disappear?
15. Create an H1 below your animated div, with the text content of the property and value which made the div disappear.
16. How is this property different than display: none? Highlight the animated div with your inspector again, after you have added the h1, and add “display: none” at the top of your styles tab just under where it says “element.style {“. What's different?
17. Refer to [this guide](https://files.slack.com/files-pri/T039N7TM0-F8KGXRZEH/updating_github_repository_walkthrough.txt" \t "_blank) to update your Github repository with your completed week 3 exercises when you’re done!

***Once you have completed these exercises add them to Github.***

[Download Completed Week 4 Exercises](http://codifyacademy.com/wk4/week4exercises.zip)

**Debugging Exercises**

**For these exercises, see the following images for an idea of what the web page SHOULD look like at different screen sizes:**

* [Large monitor screen size](http://codifyacademy.com/wk4/img/IronGiantScreenshot_LargeMonitor.PNG" \t "_blank)
* [Laptop screen size](http://codifyacademy.com/wk4/img/IronGiantScreenshot_Laptop.PNG" \t "_blank)
* [Tablet screen size](http://codifyacademy.com/wk4/img/IronGiantScreenshot_Tablet.PNG" \t "_blank)
* [Mobile screen size](http://codifyacademy.com/wk4/img/IronGiantScreenshot_Mobile.PNG" \t "_blank)

Now download each version of this project below, one at a time, and open them in your text editor.  
  
Be sure to open the page in your browser AND CHANGE YOUR WINDOW SIZE to see what is and isn't currently working at different sizes, and then examine the code to find the problems.

* [Debug max-width media queries](http://codifyacademy.com/wk4/debug/Debug_maxWidth.zip)
* [Debug min-width media queries](http://codifyacademy.com/wk4/debug/Debug_minWidth.zip)

The final, completed SOLUTION code can be found [here (max-width)](http://codifyacademy.com/wk4/debug/SOLUTION_CODE_maxWidth.zip" \t "_blank) and [here (min-width)](http://codifyacademy.com/wk4/debug/SOLUTION_CODE_minWidth.zip" \t "_blank).

**Project**

***Important: Make sure to add comments to your code.***

This week we will focus on adding media queries to our previous projects so they look good on any screen size. Take your PSD replica from Week 3 and make it responsive.

**Steps For Making a Website Responsive**

1. Begin by adding the meta viewport tag into the head of the document, otherwise your media queries won't work. Refer back to Week 3 in Litmos to remind yourself how to write it out or Google it! Once you have your meta viewport tag in your head you can begin to add in media queries to your CSS.
2. Before you start adding media queries, PLAN HOW YOU WANT YOUR SITE TO LOOK ON DIFFERENT SCREEN SIZES! You will want to display your information differently on a mobile screen from how it appears on a large, high-res monitor. Consider how you want your layout to change and when, and what CSS you will need to add or overwrite in order to effect these changes.
3. Inspect your site in browser and use the Google Responsive Developer Tool to resize the screen, when elements start to look bad at any screen size that's where you will add a media query.
4. When you're happy with how the website looks from 400px - 1200px then validate your CSS code to make sure there are no new errors.
5. Upload your updated project to Github following the steps in "How to Update a Existing Project on Github" from week 2.
6. Once your done check off in Litmos Week 4 that the in class project is complete.

**Working Project Examples**

[Google Responsive Example](http://codifyacademy.com/litmos/wk04/google-responsive" \t "_blank) [Salespage Responsive Example](http://codifyacademy.com/litmos/wk04/salespage-responsive" \t "_blank)

**Jobs Program**

One of the most important factors which potential employers look at to determine whether or not a job applicant would be a good hire is their Github profile. Whenever you make a push to Github, your profile logs the push.  
  
Employers will examine your profile to see how actively you push code to Github. An active Github account can indicate an active coder, an active collaborator, or someone who actively contributes to open source code; and can serve as a mark of professionalism.  
  
From here on out, commit to pushing a change to Github every single day. The changes you push can be minor, or even insignificant, but make a change to one of your projects or set of exercises, and update the corresponding Github repository at least once a day to begin building an active coding track-record.  
  
As your projects become more involved, or as you engage in collaborative coding, this will be a vital habit to take advantage of.