Week 13: Special Topic – Responsive Web Design

Agenda

- Overview
    - Previous Participation Questions/Responses
    - What's Due?
- Responsive Web Design (RWD - Lecture/Class Activity)
    - What is it?
    - How do we use it?
- Responsive Fact Page (CW/HW)
- Participation Questions (4)
- Next Week

Overview

Last week, we did a final review of JavaScript and learned a little bit about JQuery and how it can be used on a page. This week, we will cover Responsive Web Design (RWD) and learn methods for implementing it so that our pages work well on a wide variety of screen sizes and devices.

Previous Participation Questions/Responses
https://forms.gle/hQ8u8b9tWKaMWm4q9

This link has the responses to the fun questions from previous weeks. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

What's Due?

- Participation Questions (4)
- Responsive Fact Page (CW/HW)

Responsive Web Design (Lecture/Class Activity)

What is it?
Responsive web design (RWD) is an approach to web design that incorporates a variety of techniques to help pages render on numerous devices depending on screen sizes, orientation, browser, etc. Ideally, the page layout would change based on the previously mentioned factors without forcing any particular view on the user.

How do we use it?
Responsive design is implemented primarily through CSS. By using certain properties and values, we can make the content on our pages enjoyable on any device or screen. Below is a list of properties and methods for using them on your page to increase its responsiveness.

Width
You can use several types of units with the width property to create interesting layouts. However, percentages are the easiest to use when it comes to container widths. If you have a two-column layout, 50% widths on each container will ensure that the content from each container only takes up half of the screen.

Using pixel-based widths isn't recommended. This is because containers with pixel-based widths will not adjust to the size of the screen and will always have the same screen size. If the screen size is smaller than the width of the container, a horizontal scrollbar will appear, and this is bad for usability.

Max-width
This property is great when paired with width for containers. You can give a container a width of 100% so it's always 100% of its container or the screen. Adding a max-width of 900px would make it so that the content is responsive but won't be wider than 900px at any point.

For images, this property is incredibly useful. You can use img {max-width: 100%} and this will almost guarantee that all images on your page are responsive. They won't go beyond the width of their container and will shrink as the page shrinks as well.

Font Styles
The font on a page may initially be very large. However, as the screen size decreases, it's important to resize your fonts. Font-size is best for altering the size when a page increases or decreases. It's a good idea to do your font sizes in ems since that is a relative unit. Pixels are okay, but relative units scale depending on the structure of your HTML.

Line-height and letter-spacing can also be useful for making fonts more readable on various screen sizes. A good value for line-height is 1.3 or 1.4, but you can play around with values until you see something you like.

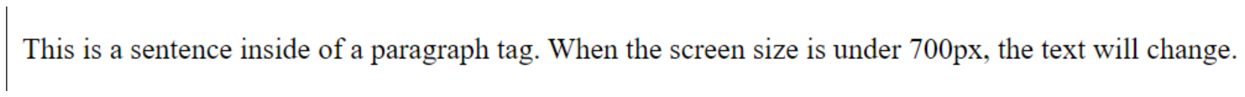
@media()
Media queries are an important part of responsive design. More modern properties such as flex and grid make it so that they aren't entirely necessary, but most responsive websites make usage of media queries. Media queries are rules with styles we can place in our CSS that allow us to change elements on our page. For example, you can have a two-column layout that changes to a one-column layout when there isn't enough room on the screen.

To use media queries, you need to place the following tag in the head tag of your HTML document. It's known as the viewport meta tag and without it, your web pages wouldn't look so good depending on the device/browser being used.

```
<meta name="viewport" content="width=device-width, initial-
scale=1">
```

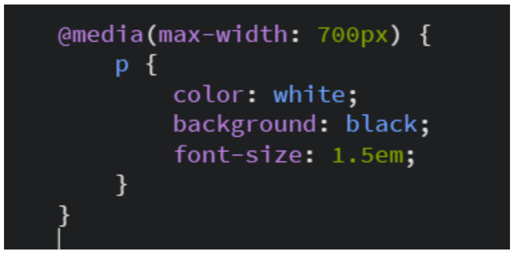
Without this tag, we cannot use **media queries**. Media queries are useful for changing styles on your page as necessary. This can include hiding content, making content appear, or altering content that's already visible.

In the image below, we have a basic p tag with some content inside of it. With a media query, we can change how this paragraph looks based on a screen size we choose.

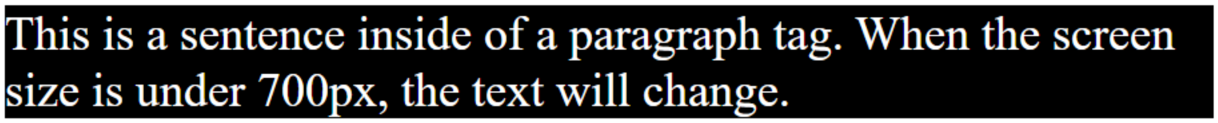

This is a sentence inside of a paragraph tag. When the screen size is under 700px, the text will change.

The syntax for media queries is: @media(some rule) {styles in here}

We replace "some rule" for something like max-width, min-width, etc. and then provide a screen size which will act as a trigger for the styles inside. In the screenshot below, we have a media query set to max-width of 700px. If the screen width is 700px or below, the styles inside of the curly brackets will execute.


```
@media(max-width: 700px) {
    p {
        color: white;
        background: black;
        font-size: 1.5em;
    }
}
```

Once my screen width gets below 700px, the content will look as it is shown below.


This is a sentence inside of a paragraph tag. When the screen size is under 700px, the text will change.

The "700px" portion of my code is known as my "breakpoint." This is a point in the page where things typically begin to break in terms of appearance. We can then apply styles to fix the look of the page. Determining your breakpoints shouldn't be hard. All you have to do is resize your page in a browser and look for times where content becomes squished or inaccessible. You can then add a breakpoint near that width and add yours styles.

To find the current width of a browser screen, you can inspect the page and resize it. If you're in Google Chrome, the size of the screen should appear in the top right of the page. You can use those values in your media queries.

In the examples above, I used "max-width" with my media query. Max-width is typically used for when you're going from a desktop screen to a smaller screen. It essentially says, from 0 up until this screen size, apply these styles. Min-width works from smaller screens to larger ones. It is like saying, from this minimum width and higher, apply these styles.

I've provided an example link below the "Online Resources" with some basic examples of responsive design using floats and inline-block (inline-block can also be used for positioning). Please view the page source or inspect the page for comments in the HTML/CSS with explanations of what's going on.

Class Example & Online Resources
This link shows how to use some basic media queries to change the structure of a page.

https://www.albany.edu/~cv762525/cinf201/examples/basic-responsive-example.html

To view all other class examples, visit this link and click whatever you want to see. The js folder contains the various JS files used for the examples.

https://www.albany.edu/~cv762525/cinf201/examples/

You **do not** have to read through every single link below. They are to be used as needed if you want additional coded examples. JavaScript has many features, and we will only scratch the very surface of them in this course. However, digging deeper is good if you want to implement advanced features on your pages.

Online Resources

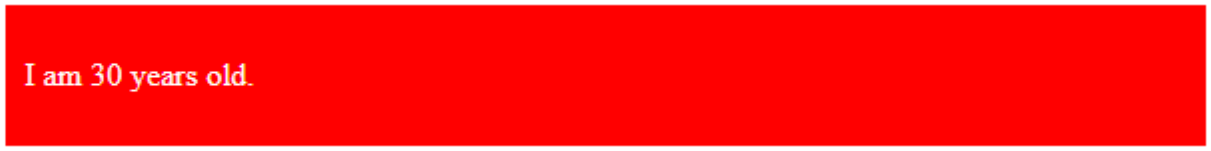
- https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag (Viewport)
- https://developers.google.com/web/fundamentals/design-and-ux/responsive (General)
- https://css-tricks.com/logic-in-media-queries/ (Media Query Logic)
- https://www.smashingmagazine.com/2018/02/media-queries-responsive-design-2018/ (Alternative to floats/media queries)
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox (Flexbox)
- https://joshnh.com/weblog/why-you-should-use-inline-block-when-positioning-elements/ (Argument for inline-block vs floats)

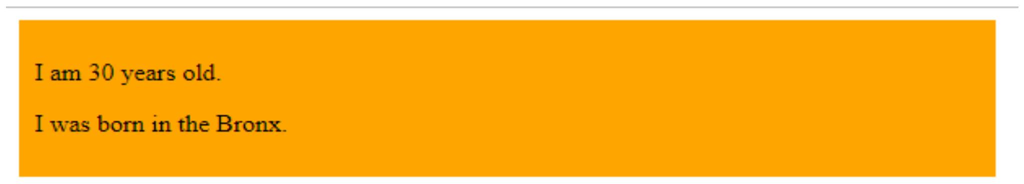
Responsive Fact Page (CW/HW)

Create a web page that will display **6 total facts** about you after the screen is resized. **No web page is being provided to you for this assignment, you must create one yourself.** Initially, only one fact should be displayed. As the page gets larger or smaller, a new fact should be displayed until all 6 facts are present. The CSS for this assignment can be internally or externally placed.

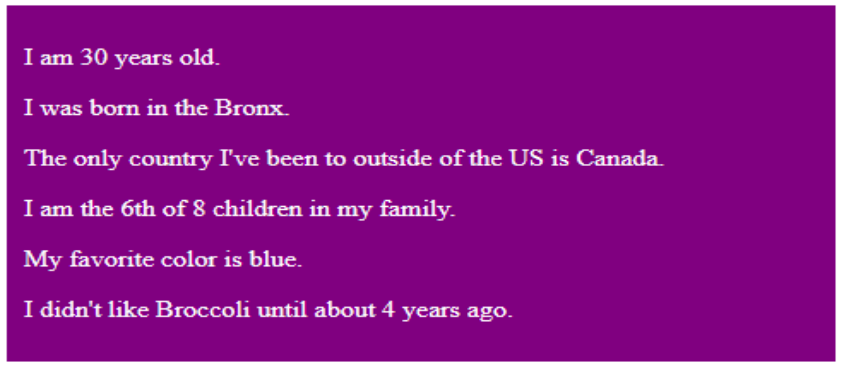
The page can start off full-width with only one fact being displayed. As the page gets smaller, a new fact should appear until all 6 facts are displayed. Each time a new fact is displayed, the font color of the text should change. Also, the background color of either the body or newly displayed fact should be changed as well. **You can also start with a small screen and add facts as it gets bigger.** For this, you would have to reverse your media query logic (use min-width for small to big screen transitions instead of max-width).

The images below depict what part of the page should look like initially, after one media query, and what it should look like at the end. The colors are entirely up to you but try to use white or black for the font, so I can read things more easily. In my version, I decided to use six paragraphs wrapped in a div. Every time a media query was triggered, I would change the paragraph font color and the container color. As long as new facts appear, and some color changes are made, you'll receive credit for your work.


I am 30 years old.


I am 30 years old.

I was born in the Bronx.


I am 30 years old.

I was born in the Bronx.

The only country I've been to outside of the US is Canada.

I am the 6th of 8 children in my family.

My favorite color is blue.

I didn't like Broccoli until about 4 years ago.

<u>Responsive Fact Page Tips</u>

- Start your media queries from 1200px and then space them out by 150px (1200, 1050, 900, 750, and 600). You can have your "breakpoints" occur whenever you want but these can be a good starting point. If you choose to use these numbers, max-width will be the way to go.
- Have the last 5 facts set to display: none at first and then display them one at a time in each media query (display: block).

You don't have to use personal facts if you don't want to, list any interesting facts if you'd like

But Professor, I Don't Know Where to Start

If you're not sure where to start at all, this section was made for you! I've listed out basic steps that you can take below for your convenience. Follow them carefully and you should be able to get the assignment started.

- Go to Blackboard and download the "Template Files" zip folder from Course Materials → Additional Resources
- Take the files inside of this zip file and place them in another folder you can find.
  - Create a "Week-13" folder in your "Documents" folder or some other folder you can access easily
  - Place your html file, css folder, and css file in this newly created "Week-13" folder
- Open the "html-template.html" and "style.css" files with a text editor of your choice
- Add 6 facts to your HTML file (6 p tags could work, but the structure is up to you)
- Use display: none on the 2nd through 5th facts so they are hidden (style.css file)
- Add styles for the first paragraph so that it has a different color/background color
- Add 5 media query blocks (one for each fact that will be displayed)

- Inside of these media query blocks, you'll add styles for each paragraph (display: block and change the color/background properties)

Your web page is **due December 2nd at 6pm.** To submit the work for this exercise, go to Blackboard → Course Materials → Lecture Notes for this week's class. You can also visit the "Assignments" folder and the submission area will be titled "Responsive Fact Page" **You must submit a live link to your web page**. The work submitted will be evaluated based on the rubric explained below.

Responsive Fact Page – 10pts

- Live link submitted – .5pt
- HTML is valid – .75pt
- CSS is neat/organized – .75pt
- Each fact displays eventually and has a color change – 8pts or ~1.33pt for each fact

Participation Questions 11/18

To receive credit for participation in today's class, please answer all the questions in the Google Form linked below. If you don't want to answer the final question (fun), simply put N/A (not applicable) for your answer and that will count.

This Week's Participation Questions:
https://forms.gle/vqR48dAnByeGtPjP8

Previous Participation Questions/Responses
https://forms.gle/hQ8u8b9tWKaMWm4q9

This link has the responses to the fun question from previous weeks. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

Next Week

We will have our Final Project Reviews in our final class. Each of you will do a brief presentation on your website. The presentation should only be about 2-3 minutes long and cover your website's topic, why you choose it, and other things as mentioned in the Final Project Description on Blackboard.