Week 5: RWD Part 2

Agenda

- Overview
  - What's Due?
- RWD Part 2
  - Responsive Design Recap
  - More Tips/Methods
    - Graceful Degradation
    - Progressive Enhancement
  - Class Examples
- Second RWD Exercise
- Responsive Page Evaluation (Two Posts)
- Next Week

Overview

Last week, we learned about responsive design and how to implement it on our pages. The viewport meta tag and media queries were introduced as components to aid us in making our pages perform well on multiple screen sizes and devices. This week, we will learn more about responsive design and some approaches to it.

What's Due

- Second RWD Exercise
- Responsive Page Evaluation (Two Posts)

Responsive Web Design Part 2

Recap

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, and other condition. It is typically implemented with CSS. By using certain properties and values, we can alter the looks of our pages so that the information is accessible in any media.

Width

The width property determines how wide an item will be on a page. For containers, percentages are ideal since they are based on the width of the browser. If you utilize the box-sizing property with a border-box value, widths and margins become the primary properties to consider for multi-column layouts. Using pixels for widths can result in your page being static and not fluid.

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

Font-size, line-height, and letter-spacing are useful properties for keeping your text content looking presentable. Pixels and ems are good units to use for the values you provide. The goal is to always make your font readable no matter the screen size or device.

@media()

Media queries are an important part of responsive design. They allow you to apply styles to the page depending on the conditions you establish. A condition might be a certain page width, screen orientation, or even when the page is being printed.

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 content starts to become squished or disorganized in terms of its appearance. This may also be a time when you want to alter the way a page looks to provide a different layout.

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.

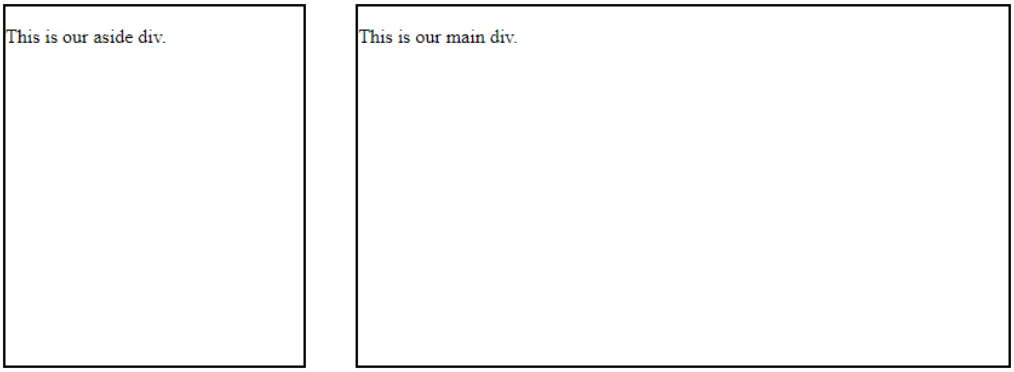
Flexbox Properties

Flexbox works by using “display: flex;” on the parent container of content you want to appear in one row. Creating a two-column layout on a page would look something like what is shown in the screen shots below. Flex-wrap will wrap our content onto the next line automatically if it’s needed. When using percentages, this won’t have an effect. However, if you used pixel-based widths, content would move to the next line automatically if there isn’t enough space.

```html
<div class="flex">
    <div class="aside">
        <p>This is our aside div.</p>
    </div>
    <div class="main">
        <p>This is our main div.</p>
    </div>
</div>
```

```css
* {box-sizing: border-box;}
.flex {
    display: flex;
    flex-flow: row wrap;
    width: 100%;
    max-width: 900px;
    margin: 0 auto;
}
.aside {
    width: 30%;
    margin-right: 5%;
    height: 300px;
    border: 2px solid black;
}
.main {
    width: 65%;
    height: 300px;
    border: 2px solid black;
}
```

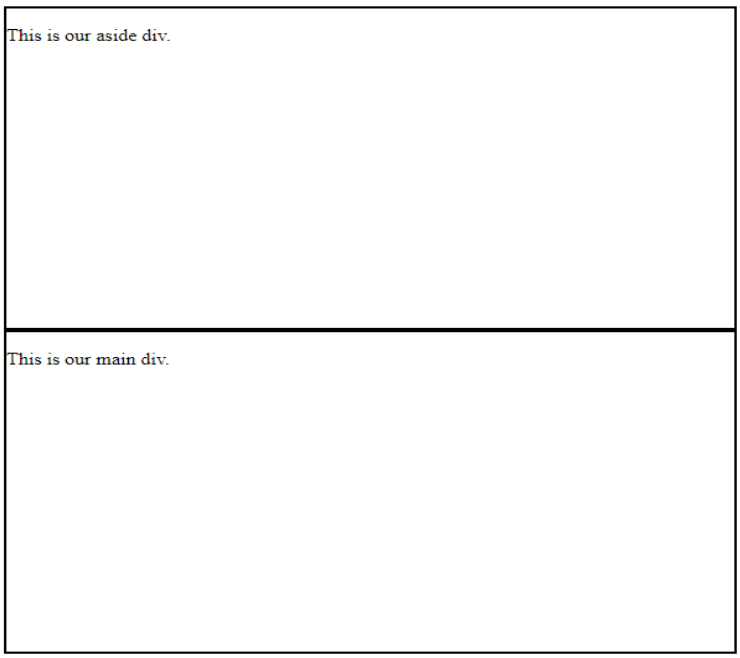
That code would produce the following layout on our page



If we wanted to turn this into a one-column layout when we got to a smaller screen size, we could use a media query combined with the flex-direction and width properties as demonstrated in the screenshot below.

```css
@media(max-width: 700px) {
    .flex {
        flex-direction: column;
    }
    .main, .aside {
        width: 100%;
    }
}
```

In the media query above, we change the flex direction to column. By default, flex box makes items fit in a horizontal row. By changing the direction to column, we make the boxes stack on top of each other. The only issue is that the old widths remain, so we use the next declaration to change the widths to make our containers full-width (100%). You can use less width and apply margins to alter the look as you'd like. The code above would result in the following layout once our page became smaller than 700px.



Other properties for flexbox include "order" and "justify-content." Order allows you to change the order of content on the page and justify content allows you to alter the spacing of your elements.

Floats

The same concepts in flexbox apply when using floats. When you have multiple items floated side-by-side as is done in this week's assignment, you would have to alter the float property. If something is floated left or right, you could change it to "float: none;" which removes the float on that item. You would then alter the width as we did above for flexbox. This can be seen in the class example titled "basic-responsive-example."

More Tips/Methods

With RWD, our focus must be on the screen's "real estate" which is the area we are able to work with when displaying our content. Users will visit websites on any number of devices: smart phones, tablets, desktops, etc. It is important to take all these devices into account.

However, we should **never** build our website for any specific device. The best thing to do is to build our website and then determine where to add breakpoints to ensure a smooth transition from larger screens to smaller ones, or vice-versa. Here are some additional tips for making your website more responsive. Most of them are related to usability/accessibility, which we will cover in more detail later this semester.

- Keep the most important content on the screen, remove anything extra

- Make font readable on smaller screens
- Give tappable buttons/links enough space (padding and margin)
- Minimize your navigation menus
- Remove unnecessary images on smaller screens
- Use percentages and relative units such as em in your page

There are two main approaches to RWD which are often expressed as "progressive enhancement" or "graceful degradation." These approaches are also used in the development cycle for other types of applications. They are separate approaches, but the goal of making web pages accessible across various browsers or devices is the same.

Progressive Enhancement
Progressively enhanced websites start simple, and then add features as they become available. HTML is typically the backbone for this kind of development. If the HTML is structured in a way that is understandable, then the presentation/features on top of it are easier to build.

After HTML, CSS will be added for the presentation of the page. It may be developed for mobile devices first and then higher resolutions are targeted after. This allows you to start small and then add features as larger screens can fit them. You might start with the core features/information and then add things like image carousels, modals, etc. for larger screens when the space becomes available.

The media query you would use for this is typically min-width. It would look like the line of code below which is equivalent to saying: from this minimum width and higher, apply these styles.

@media(min-width: 400px) {styles in here}

Progressive enhancement is useful for having a baseline level of functionality for all devices and screen sizes. Then, more advanced features are added as larger screen sizes/newer browsers are developed. This is a great approach if you can start on a website with fresh code or don't have too much content to overhaul/restructure.

Graceful Degradation
Graceful degradation is useful when you want your website to have very modern features that older browsers might not support. The enhanced version of your website would be the one with the best available features on it. As the browsers/technologies being used to access your content get older, some of those features might become unavailable or have reduced functionality. In terms of responsiveness, you might remove features that only work on desktops as opposed to mobile devices.

This is typically used for websites/applications already in existence. These websites already have content in them and as the screen gets smaller, you remove features and other things that aren't entirely necessary for the design. This could be removing images, extra text, etc. so that your core features are more accessible for users. The line of code for this is below and looks very similar to the one above. The difference is switching max for min. This is like saying: from this maximum width and below, apply these styles.

@media(max-width: 1000px) {styles in here}

With this approach, you are reducing the number of accessible features, but doing so in a manner that doesn't limit website functionality.

Which Approach Do I Use?
The approach you use is entirely up to you. However, I would personally utilize concepts from both. No matter what you do, your content should be accessible from any browser, browser version, screen size, screen orientation, etc.

You should know what content is the most important and the primary method of access for that content. Most people are now utilizing mobile devices to access internet content. I would build my own website from a mobile perspective first. Focus on the content that should be available on every device/screen size and make sure it's present/understandable from the smallest screen to the largest ones.

If you understand which content is the most important, you can decide what features to build around it. Some of these features may be available only for desktops or certain views/devices. Using those features is not a bad idea. However, you should include alternatives for older browsers and devices (depending on the scenario).

In this sense, we are using the progressive enhancement method by focusing on important content first. However, we also consider what features can be helpful to users and building alternatives for devices/browsers with lower capabilities to embrace graceful degradation.

The links below cover some best practices and have some comparisons of different strategies. Overall, you'll notice a pattern in each guide in that the focus is on the real estate of the screen and how much room we give to the user for what they're trying to accomplish.

Online Resources (Not required, but can be helpful for guidance)

- https://uxtricks.design/blogs/ux-design/responsive-design/
- https://medium.com/level-up-web/best-practices-of-responsive-web-design-6da8578f65c4 (Links to many other tutorials)
- https://www.toptal.com/designers/responsive/responsive-design-best-practices
- https://www.interaction-design.org/literature/topics/responsive-design#:~:text=Responsive%20design%20is%20a%20graphic,ensure%20content%20consistency%20across%20devices.
- https://www.w3.org/wiki/Graceful_degradation_versus_progressive_enhancement
- https://www.mavenecommerce.com/2017/10/31/progressive-enhancement-vs-graceful-degradation/

Class Examples
All the links below point to pages that I created to give you an example of the concepts mentioned above. Feel free to right-click the page and view the source. You can then click the link to the CSS file to see my code which has comments placed inside of it.

This link shows how to use some basic media queries to change the structure of a page.

https://www.albany.edu/~cv762525/cinf362/examples/basic-responsive-example.html

Here's a link to all course examples:
https://www.albany.edu/~cv762525/cinf362/examples/ (coded examples)
https://www.albany.edu/~cv762525/cinf362/videos/ (videos)

There is also content on Blackboard to help you with understanding/debugging CSS code. These items are located on Blackboard → Resources → CSS and will help further explain CSS and demonstrate some best practices you should adopt moving forward.

Before completing the assignment(s) for this week, please read the "Viewing Your Web Pages.docx" file on Blackboard. You will not be able to submit anything for the assignment without completing that portion first. It is located directly in the Lecture Notes folder.

Second RWD Exercise

Last week, you had to take a static page and make it fluid/responsive. This meant changing pixel-based widths for major containers to percentages and adding media queries to alter the page layout at various screen sizes. Your task this week will be to take a page I have built and use media queries to make it fully responsive.

Due February 27th at midnight
Download the "Second-RWD-Exercise.zip" folder from Blackboard under this week's Lecture Notes or the Assignment folder. Extract everything from inside of this zip folder to some place where you will be able to find them easily. There will be an HTML file, 3 images, a css folder, and a CSS file.

The 3 images depict my page at 3 different stages. In total, I used 4 media queries to structure the page as I did in the 3rd image. Please note that you don't have to style the page exactly as I did. You can improve on it if you'd like as my styles may not be the most aesthetically pleasing. The screenshots also don't have all of the content present as some things are cut out.

Open the page in a browser and try to resize it. You'll notice that the content is fluid, and no horizontal scrollbars are created at any size. This is the power of using percentages and relative units while creating styles for your page. In my CSS, I only use percentages for widths and ems for font sizes, padding, and margins. I would recommend doing the same for your own work and especially the Final Project.

The page is fluid and looks okay overall, but even though no horizontal scrollbar is created, some content becomes squished and difficult to read or use. Like last week, your task is to add or alter CSS so that this page becomes responsive down to a size of 400px. This means that the font sizes, margin, padding, positioning, etc. will have to be changed so that all the content is present on the screen at smaller sizes. There should be no squished content on the page or too much white space at any size. The transition from a large screen to a smaller one should be as smooth as possible.

You can change the HTML/CSS as needed to complete this assignment. However, all content in terms of text/images should still be present.

There are a few ways to fix this page and I've provided a short list of tips to help get you started.

General Tips

- Your smallest media query shouldn't be below 400px and your largest one shouldn't be above 1200px at least for this assignment
- Remove floats from floated items
- Change the values for widths
- Change the flexbox values (row to column)
- Change the order of columns
- Decrease or increase font sizes at different sizes
- The media queries I used were for 1050, 700, 600, and 500px

If your CSS isn't working at all, it's probably because the href value you've provided in the link tag is incorrect. If your CSS is working, but some styles are not appearing as you expect, check for missing semicolons, and make sure you aren't overriding your style in some other declaration that is more specific.

Your webpage is **due on Sunday, February 27th at midnight.** To submit the work for this exercise, visit Blackboard → Course Materials → Lectures Notes for this week's class. You can also go into the "Assignments" folder and the submission area will be titled "Second RWD Exercise." You should be submitting a live link to me; it can be through the UAlbany server or 000webhost. The work submitted will be evaluated based on the rubric explained below.

Second RWD Exercise Rubric – 10pts

- External CSS file is used – 1pt
- Live link submitted – 1pt
- CSS is organized/easy to read – 2pts
- Content is responsive down to 400px – 4pts
- Content transitions are smooth (avoid having squished content, unreadable text, and horizontal scrollbars when the page gets smaller) – 2pts

Responsive Page Evaluation (Two Posts)

Initial Post due Thursday, February 24th

Visit a few websites and discuss the general responsiveness of a few pages on those websites. You don't have to cover each page of every website but write enough to demonstrate that you at least went through a few pages and tried to resize them. If you aren't sure what websites to look at, I've provided a list of a few which span different fields. Feel free to use some of the links for your discussion. **Make sure to provide links to some of the websites or pages you evaluated.**

- http://nycourts.gov/reporter/
- https://www.wish.com/
- https://www.albany.edu/
- https://www.nasa.gov/
- https://www.whitehouse.gov/
- https://www.nike.com/

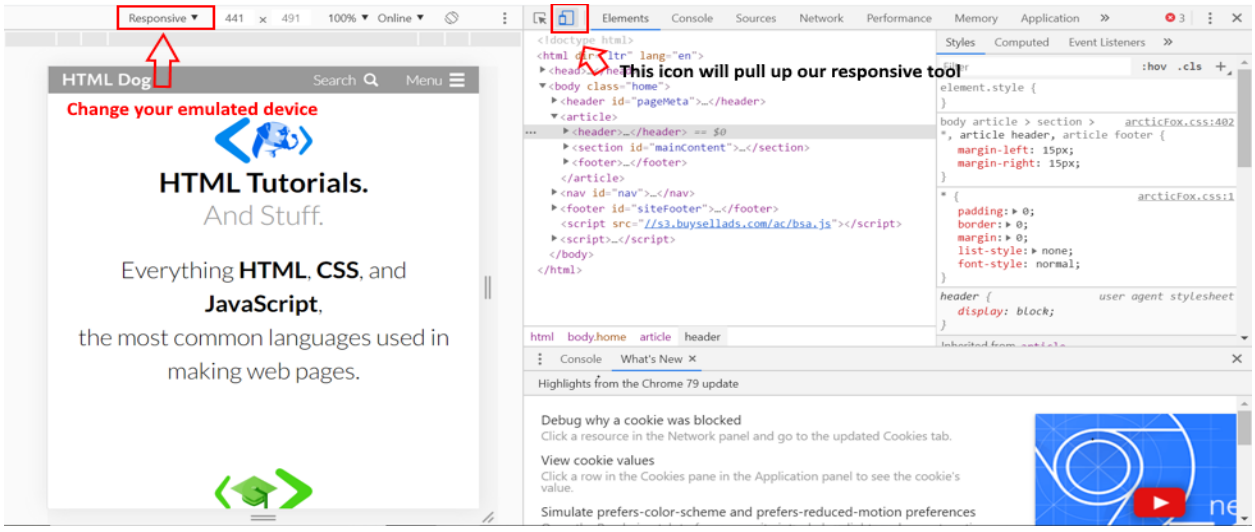
- https://www.rocketleague.com/
- https://www.homedepot.com/

The responsiveness of a website can be defined as how this website's structure responds on different devices or resolutions. Some website's appearances will change when viewed on a desktop, a tablet, or even a phone. Here are some questions to guide your discussion. **Do not answer the questions in a question/answer format. Please write complete sentences in about two paragraphs.**

- Are the pages responsive on different devices?
- What components of the websites are done well in terms of mobile design?
- What areas could be improved?
- What media queries did they use? CTRL or CMD + F in their css style sheet can help you find them
- Is a horizontal scrollbar present? Is content hidden because you must scroll to see it?
- Are buttons and other interactive elements sufficiently sized and easy to click/tap?
- Do you notice any common components or features across the websites on any platform? (Hamburger menus, content disappearing, etc.)
- How does the page perform when the URL is plugged into the Google mobile-friendly test (https://search.google.com/test/mobile-friendly)?

To test each website's responsiveness, view the websites on different devices (phone, desktop, tablet, laptop, etc.) or use Google Chrome's web tool to emulate what it would look like on various platforms. To use Google Chrome's web tool while using a PC, press F12. After this, press Ctrl + Shift + M. This should pull up the responsive tool. You can also right-click the page and select "Inspect Element" to view the web tools. Afterward, you would click the little cellphone/tablet icon (Toggle Device Toolbar) in the top left of the web tool window.

I've attached a screenshot below to highlight the button. It is surrounded by a red box and has black text next to it. The other button surrounded by a red box with red text next to it (left side of image) will allow you to change the device you are emulating and viewing the page on. Try to view pages on a few different devices.



This link also provides explanations to some of the tools depicted within this screenshot:

https://developers.google.com/web/tools/chrome-devtools/device-mode

The initial post is **due Thursday, February 24th at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "Responsive Page Evaluation" where you can post your initial post. You can also visit the Discussion Board area directly from the Course Materials folder.

Response Post due Sunday, February 27th
In your response post, compare the websites mentioned by other students to the ones you evaluated. Do you agree with their overall assessment? Were their websites responsive? Were there any similar features mentioned? Did you notice anything that could be improved for the website? These aren't required questions but just some to give you an idea of what I'd like to see in your posts.

The response post is **due Sunday, February 27th at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "Responsive Page Evaluation" where you can post your response post. You can also visit the Discussion Board area directly from the Course Materials folder.

Responsive Page Evaluation Rubric – 2pts
The initial post is worth 1.5pts and the response post is worth .5pt for a total of 2 points. I will be evaluating your posts based on the following criteria:

- Did you provide links to the websites you evaluated?
- Were specific items about the websites mentioned?
- Did your response post contribute to the original post?
    - Avoid summarizing the other person's post or simply saying that it was a good post.

Next Week

Next week, we will do our third/final bootcamp which covers JavaScript and how to add it to our pages to give them more interactive components. Feel free to read through the links below for a brief introduction to JavaScript.

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.htmldog.com/guides/javascript/
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript