

Lab Exercise 5: Responsive Layouts and Media Queries

Tomoko Okochi

Goals: To create a web page that will respond to any device by arranging itself into an appropriate layout based on the width of the screen.

In order to attain this goal, you will create a simple web page about rocky planets that has three separate layouts that roughly correspond to mobile phone, tablet, and desktop/laptop devices.

Task 1: Setup

In the starter files, I've included some screenshots of what the site should look like when it's done, as well as a starter HTML document and CSS stylesheet. Open the folder on VS Code or an IDE of your choice.

I've provided most of the HTML for the page, as well as a little bit of CSS. As you can see, I've added a background to the <header> element using the **background** property. Using this property, I've added a background image and a linear gradient to darken the bottom of the header slightly. (note that we can add more than one background property by separating each property by a comma.) I've also added a background color to the <footer> at the bottom of the page.

You are welcome to add HTML tags to efficiently design the page.

Task 2: Mobile Layout

We're going to use the **Mobile First** methodology to write our CSS. This means that we'll start by designing the mobile layout (or small screen layout) and then move on to layouts for larger devices after that. *Why do you think we do it this way? What are the advantages and disadvantages to using Mobile First Methodology?*

Use the screenshot image **mobile-screenshot.png** to guide you. As you work, resize your browser so that it is narrow, like a mobile phone.

Some guidelines:

- Use the Google Font family **Lato** for h1 and h2 headings; use the **Open Sans** Google Font family for everything else, with the **italic** style for menu items;
- Use the **rgba()** function to set a partially-transparent background color for the nav menu;
- Use the four-parameter shorthand for **padding** and **margin** values that allow you to set the top, right, bottom, and left values at the same time:
https://www.w3schools.com/css/tryit.asp?filename=trycss_margin_shorthand_4val

- Set all other colors, styles, borders, and box model properties to match the screenshots. Try to get as close as you can to the screenshots, but don't worry about matching the values perfectly.

Task 3: Tablet Layout

Next, we're going to create a layout that will take effect for screens or browser windows wider than 600 pixels. The number 600 doesn't correspond to any particular device; remember that we should design our layouts based on the most effective arrangement of the content at every given viewport size, not based on specific device sizes. *Why would we take this approach? Why not just find the size of the most common mobile device and design for that?*

Use the screenshot image **tablet-screenshot.png** to guide you.

For the tablet layout, implement the following:

- **header:** find a way to make the header have a height that is approximately proportional to the height of the window by using the **vh** unit:
<https://css-tricks.com/fun-viewport-units/>
 - o **Hint:** sometimes it's easier to set the padding-top and padding-bottom of an element's children than it is to set its height.
- Have the menu items align horizontally to the right;
- Have the articles laid out in a two-column grid;
- Have the two sections of the footer laid out in a two-column grid;
- Place vertical borders on the left and right side of the appropriate elements, as in the screenshots;
- Implement any other stylistic differences that you see between the mobile and tablet layouts.

Task 4: Desktop Layout

For screens wider than 1100 pixels, have the articles arranged in a three-column grid.

Task 5: Load Speed and Performance

Since page load speed is important for UX, SEO, and resource usage, let's analyze our own site. In **Firefox**, open the **Inspect** panel and navigate to the **Network** tab (there is a similar interface in **Chrome** if that's your preferred browser). Hold the **shift** key on the keyboard and reload the page. (This forces the browser to download all resources from the server once again instead of using the local cached copies.) You should see the load time and total page size at the bottom of the panel.

Do some research to find out if the load time and page size are acceptable. Figure out a way to improve these numbers without adversely affecting the quality of the page contents. Which items on the page do you think have the largest file size? How can you reduce their file size? Hint: this might have something to do with lessons from your graphics classes.

Hand In

Rename your working folder containing index.html, styles.css and image files to **4815-a5-firstname-lastname**. Make sure the work you're submitting does not contain any unnecessary files, such as the screenshots from the starter files. Create a Zip archive from the **4815-a5-firstname-lastname** folder and hand it in to Brightspace.

Grading

- [5] Mobile styles;
- [5] Tablet layout;
- [2] Desktop layout;
- [2] Content centered with background extended to edges;
- [2] Page size and load speed reduced.

Total: 16

Note that up to -5 may be deducted for improper hand in, mis-named files, disorganized workspace, etc.

Resources:

- Basic media query guide:
 - o https://www.w3schools.com/cssref/css3_pr_mediaquery.asp
- Units that are proportional to viewport:
 - o <https://css-tricks.com/fun-viewport-units/>
- Viewport meta tag:
 - o https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag
- General philosophy of mobile-first design:
 - o <https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00>