

# INFO/CS 1300: Lab 5 (9/22)

By Sharon Jeong

Due at the end of your lab section. Demo your work in front of a TA to get your participation credit.

## Overview

In this lab, we will begin by discussing file size restrictions, and why you want to resize your image files. Then we will explore responsive design. Responsive CSS enables your website to look professional on both personal computers and mobile devices.

## What You Need

Bring your laptop, **with Firefox installed**, to your lab section.

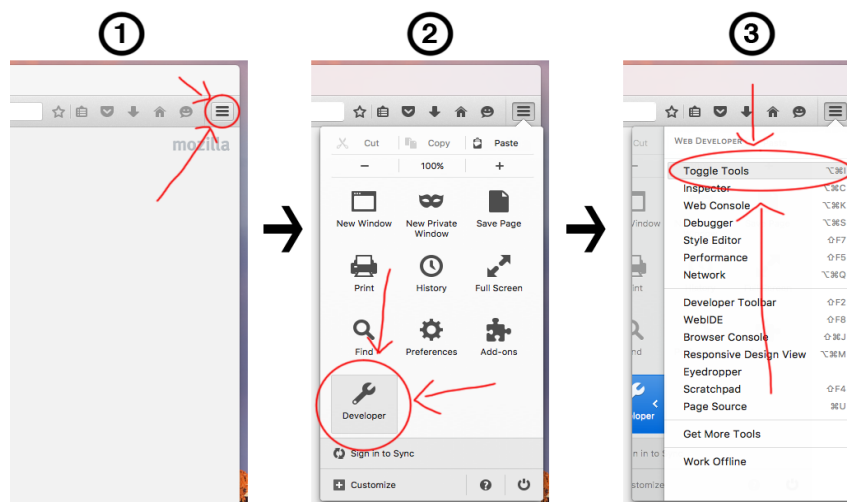
If you do not have Firefox installed, use the lab's computers or see the directions from Lab 1.

## Part 1: File Size

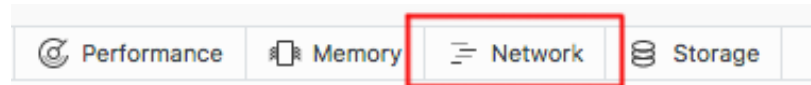
To explore why file size matters, we'll be using Firefox's Developer Tools. We are going to learn a little bit more about the capabilities of the network tab and why it is important to keep your files small.

**Do not use Chrome for this lab.** Chrome also has its own developer tools, however this lab was specifically written for Firefox.

Open up Firefox and open the Developer Tools. First (1), click the menu item in the upper right corner. Second (2), click the “Developer” button, which opens the Web Developer menu. Third (3), in this menu, click “Toggle Tools” to open up the developer tools.



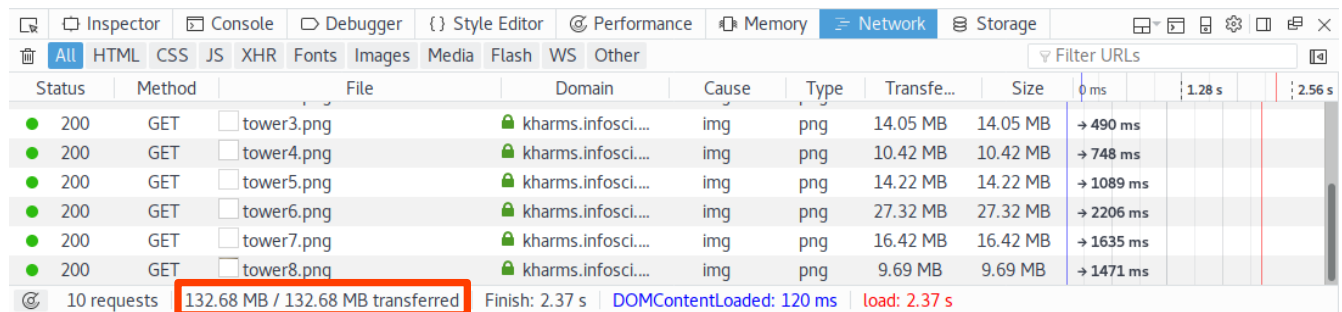
Now, navigate to the “Network” tab in the developer tools (see image below). Recall how we discussed GET requests in lecture. The network tab shows the HTTP requests between your browser and the the web server.



## CSS “Shrunk” Images

Now, navigate to <https://kharms.infosci.cornell.edu/teaching/info-1300/lab-5/large-files/>

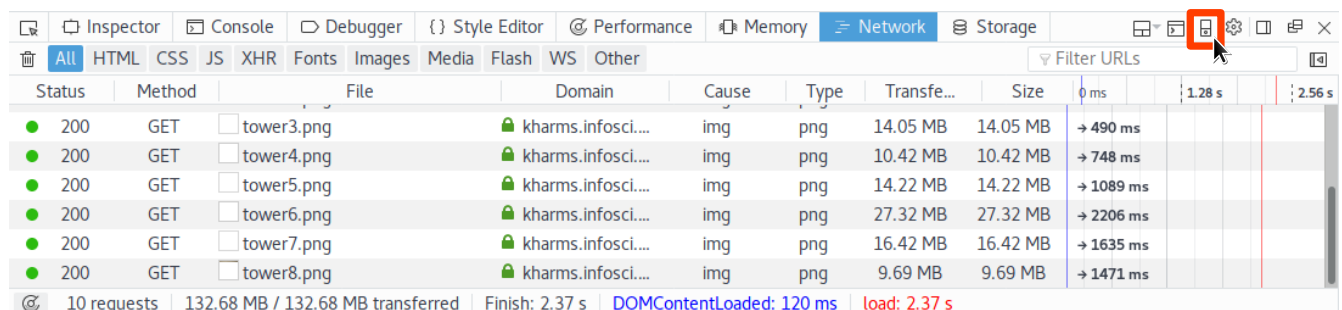
On this webpage we have collected some images of Cornell’s Clock Tower. Look at the very bottom of the network tab to find the amount of data transferred. It’s 133 MB! This is because all of those thumbnails are actually full-sized PNG files that were “shrunk” using CSS.



Status	Method	File	Domain	Cause	Type	Transfe...	Size	0 ms	1.28 s	2.56 s
200	GET	tower3.png	kharms.infosci...	img	png	14.05 MB	14.05 MB	→ 490 ms		
200	GET	tower4.png	kharms.infosci...	img	png	10.42 MB	10.42 MB	→ 748 ms		
200	GET	tower5.png	kharms.infosci...	img	png	14.22 MB	14.22 MB	→ 1089 ms		
200	GET	tower6.png	kharms.infosci...	img	png	27.32 MB	27.32 MB	→ 2206 ms		
200	GET	tower7.png	kharms.infosci...	img	png	16.42 MB	16.42 MB	→ 1635 ms		
200	GET	tower8.png	kharms.infosci...	img	png	9.69 MB	9.69 MB	→ 1471 ms		
10 requests   132.68 MB / 132.68 MB transferred   Finish: 2.37 s   DOMContentLoaded: 120 ms   load: 2.37 s										

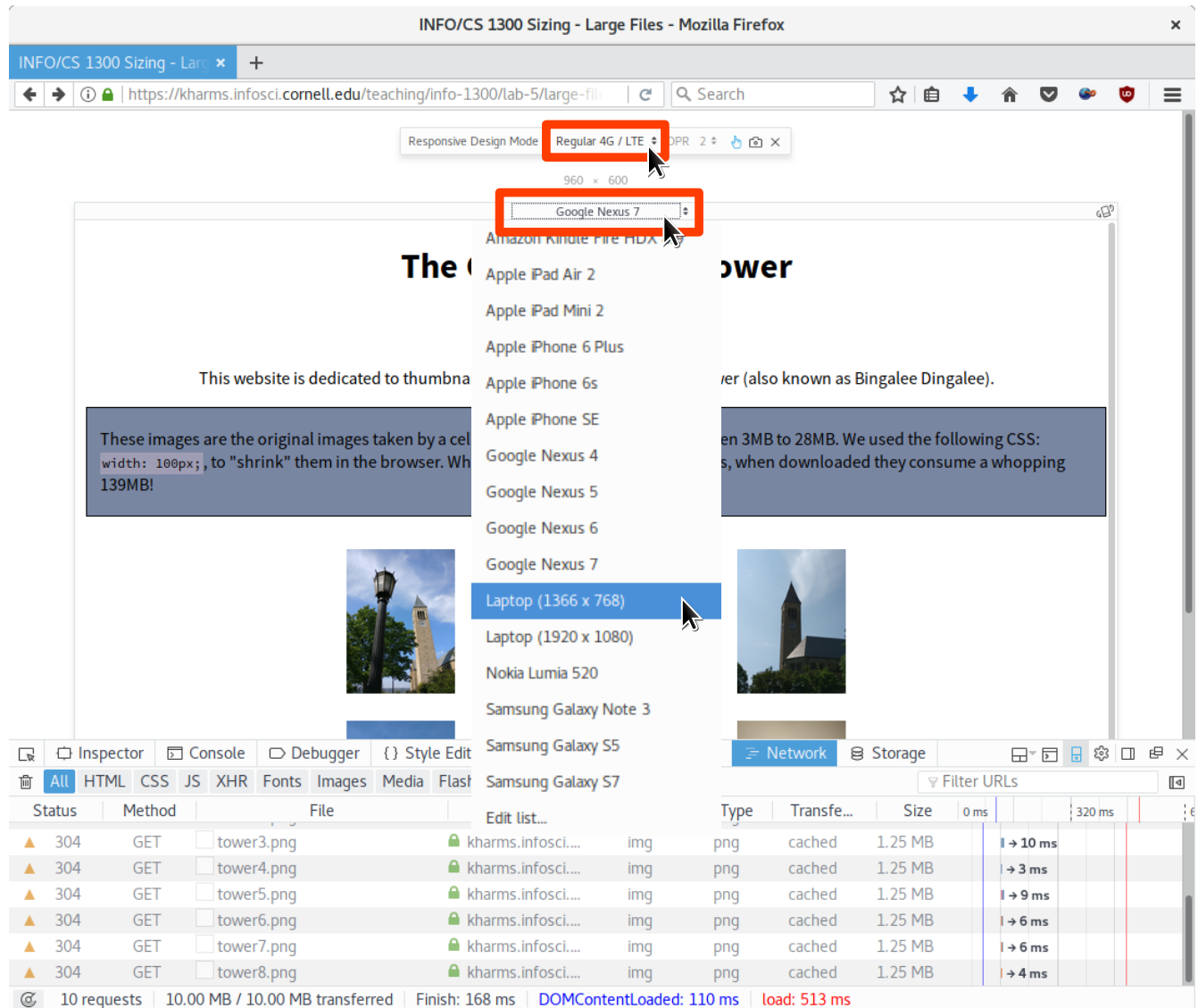
This is a great example of why it is important to scale your image files to the proper size using a program like [GIMP](#) rather than using CSS. If you only had one gigabyte (1GB) of data per month on your cellphone plan, this would have used up 13% of your data in a single web visit!

But let’s say you have unlimited data, so you don’t really care about data limits (even though your users might). Let’s use the Firefox developer tools to test what this site would be like if you were trying to view this image on your phone, using your 4G connection rather than Cornell WiFi. At the top right of the developer tools area, look for this button:



Which activates “responsive design mode.” Suddenly, the website is in a tiny *viewport*, simulating what this website looks like on a device with a smaller screen. Let’s return it to a laptop sized view, by clicking the “no device selected” menu and selecting the “Laptop (1366 x 768)” item (see the image below).

Now, let's test what a 4G connection would look like. At the very top of the page is a menu with “no throttling” as the default. Change this to “Regular 4G / LTE” (see the image below). Then, **hard refresh** the page (Shift+Control+R on Windows; Shift+Command+R on Mac).



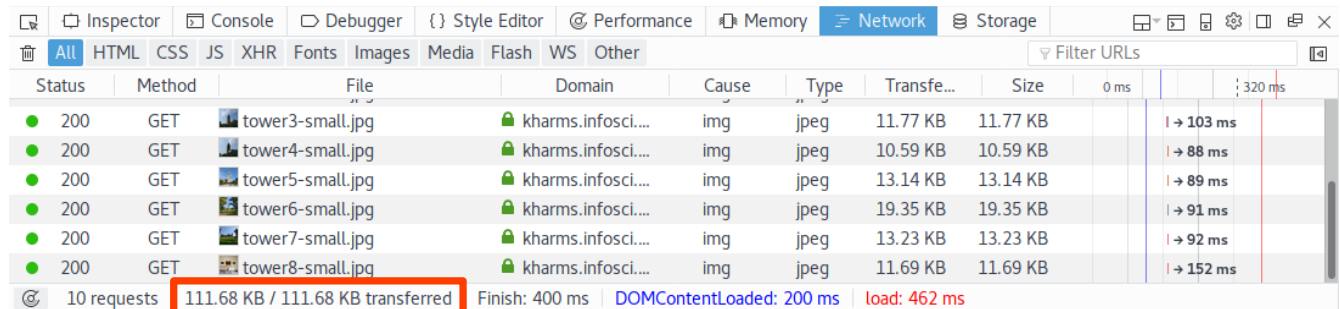
You should see the images loading at a glacial pace. You don't need to wait for them to finish loading; it will be a while. Now imagine you needed to see a map very quickly, and you were stuck outside Ithaca where you had a 2G connection. Large images would take ages to load! This is why the files of your web pages (i.e. images) need to have reasonable file sizes, otherwise your web page will take forever to download on slow internet connections.

## Scaled Image Files

Now, open the following URL in the same tab: <https://kharms.infosci.cornell.edu/teaching/info-1300/lab-5/small-files/>

Make sure you are in responsive design mode with 4G throttling. Hard refresh the page if necessary.

In this version of the webpage, we scaled the size of the image files to 100 pixels wide using GIMP's **Image > Scale Image** feature. You will notice that the page takes only moments to load. You can even switch to Regular 2G speed and see it load quite quickly (make sure you hard refresh!). If you check the networks tab, you will see it used only 112 KB, which is about 0.1% of the monthly data total we discussed above.



Status	Method	File	Domain	Cause	Type	Transfe...	Size	0 ms	320 ms
200	GET	tower3-small.jpg	kharms.infosci...	img	jpeg	11.77 KB	11.77 KB		103 ms
200	GET	tower4-small.jpg	kharms.infosci...	img	jpeg	10.59 KB	10.59 KB		88 ms
200	GET	tower5-small.jpg	kharms.infosci...	img	jpeg	13.14 KB	13.14 KB		89 ms
200	GET	tower6-small.jpg	kharms.infosci...	img	jpeg	19.35 KB	19.35 KB		91 ms
200	GET	tower7-small.jpg	kharms.infosci...	img	jpeg	13.23 KB	13.23 KB		92 ms
200	GET	tower8-small.jpg	kharms.infosci...	img	jpeg	11.69 KB	11.69 KB		152 ms

10 requests 111.68 KB / 111.68 KB transferred Finish: 400 ms DOMContentLoaded: 200 ms load: 462 ms

Now, try to keep the lessons you learned in this section in mind for Project 2. You'll be working with a lot of images, and some of them may be very large (the clock tower images were taken with a smartphone camera and were multiple megabytes). Make sure you scale them down with an image manipulation program rather than CSS! Also, consider saving the images in a smaller compression format like JPG rather than PNG.

## Part 2: Responsive CSS for Mobile Devices

Create a folder called **lab5-NetID** (example: lab5-sj632).

Download the lab5.zip file from CMS and unzip the contents to your folder. We have created a website which contains a zoo website showcasing a gallery with the many animals that appear at the zoo. You will be modifying the CSS to make the page *responsive*, so that it looks good on many of different devices.

### Initial Website

Open the **index.html** in Firefox. Now resize the window. If you shrink or enlarge the window, you will notice its structure stays the same, but scroll bars appear if necessary to display all of the content. However, this is only what it looks like on your computer. Let's turn on Responsive Design mode again and examine how the website would look on a popular mobile device: an **iPhone 6S**.

Notice that in this "mobile simulation" that this website doesn't look so good; the sidebar takes up far too much space, you have to scroll all over the place to see the zoo, and it is generally a terrible user experience.

**Note: Your Project 2 should work on other mobile devices besides the iPhone.** iPhones constitute around 15% of the current market share; Android phones currently have around 85% market share. The android phones listed in Firefox's responsive mode are the Google Nexus 4, 5, 6 (7 is a tablet) and the Samsung Galaxy phones. The Nokia Lumia 520 is a Windows Mobile phone.

## Creating a Mobile-Friendly Website

Go into **all.css** and start adding styles at the bottom of the file (where the comments note it). **Do not modify anything above the final comment in all.css** (the reason for this will become clear in Part 3).

For now, if you want to modify a style that already has an existing selector in all.css, just add a new one at the bottom (which will have *priority* over any earlier selectors in that CSS file).

Let's start by moving the entire zoo into one column. Modify the width of the #content div (again, **at the bottom of all.css**) to be 375px (the width of the iPhone 6S). Tip: For your Project 2 you can use relative dimensions so that it will layout properly on other phones besides the iPhone 6s. (i.e. width: 100%; vs width: 375px;).

If you refresh the page, you'll see a slightly better view. Everything is now in one column and easy to scroll through. However, it is a little small and stuck on the left.

Now, complete the following tasks by adding CSS to the end of all.css:

- Give the title at the top a margin of 5 pixels so that it isn't right against the edges (hint: you may have to change the font-size to around 34px).
- In the previous version, we set the height of the sidebar to 820px to make it line up with the rest of the zoo, but now this extra height is not useful. Use CSS to reset the height to be the default value (hint: check out the *inherit* and *auto* values for the height property).
- Make all of the blocks 350 pixels wide and centered. Recall how we discussed centering block elements in class. (Tip: Don't forget that you can use relative widths for your Project 2!)
- Set the default text size for the page to 18 pixels and the text size for the names of animals to 24 pixels so that mobile viewers have an easier time reading the text.
- Make all of the animal images 350 pixels wide, and then adjust the heights so that none of the content overflows (that is, make sure none of the animal names appear outside of the border).

When finished, your web page should look like the figure to the right.



## Part 3: Responsive Design

You've now authored some CSS so that the web page is mobile-friendly for the iPhone 6s. Now, switch back to the laptop view of your website. You've ruined the original webpage! Even though it now looks good on mobile, it looks terrible on desktop browsers. This isn't good responsive design.

Let's fix it! Wrap the new CSS you wrote at the bottom of all.css (to make the website mobile friendly) inside the following block:

```
@media screen and (max-width: 700px) {  
  /* Your CSS code from Part 2 */  
}
```

This tells the browser to only use that CSS when the browser window is 700 pixels or smaller (that is, a device with a small screen, like the iPhone 6s).

Refresh your page, and use Firefox's responsive design mode to switch between the laptop view and the iPhone 6s view. Now, the design magically flips between your two styles! You can also simulate this effect by just viewing the page normally and resizing the browser, but this method is less precise than using responsive design mode.

Congratulations, you made your first responsive web page!

## Extra: Responsive Tablet Design

If you still have time left, try designing a third version of your webpage, this time for tablets (e.g. the Google Nexus 7). You can figure out how you want the page to be laid out, but make sure it is proper for a tablet-sized view.

Hint: You'll probably need to take advantage of the fact that @media can take both a max and a min width:

```
@media screen and (min-width: ###px) and (max-width: ###px)
```

## Credit

Once you have finished this task, show your page to your TA to make sure you have done everything correctly and get checked off for participation in the lab.

If you still have time left, you should continue working on Project 2.