Exercise 3.1

Lazy Loading Images

# Objective

* Your job is to:
  + Find at least 5 hi-res images from the internet and optimize them for a webpage.
  + Create a webpage that contains your 5+ images
  + Observe how page load times change when using lazy loading vs. not using lazy loading.

# Setup

Like in exercise 1.1, we will need to run a server in order to see the results we want in the browser. Instead of giving you a script file to double click on, this time I’ll show you how you can run the server yourself using VS Code.

1. From the Downloads folder where you downloaded this exercise, drag the exercise folder *3.1-Lazy-Loading-Images* into a fresh VS Code window.
   1. If when you open VS Code there are still documents open from the last time you were working, be sure to close them all, then drag the exercise folder into VS Code.
2. Press Ctrl+` - this will bring up the terminal at the bottom of your VS Code window.
   1. ` (called a backtick) is the key to the left of the ‘1’ key and under the escape key.
3. In the terminal, you should see ~/Downloads/3.1-Lazy-Loading-Images-Starter
   1. This is assuming you followed the instructions and started from your Downloads folder. If not, your path might look different than the one listed above.
4. Check if node.js is installed on your machine:
   1. Type node -v into the terminal and hit enter.
   2. If you see a version pop up, you’re good to go.
   3. If you see something like command node not found, then you’ll have to go to <https://nodejs.org/en/> and download + install node before you can continue.
5. Install the http-server package:
   1. Type npm install http-server -g into the terminal and hit enter.
6. Spin up a server by running http-server . you should see:

Starting up http-server, serving .

Available on:

http://10.0.75.1:8080

http://192.168.1.9:8080

http://127.0.0.1:8080

http://172.17.104.209:8080

Hit CTRL-C to stop the server

1. Success! You’re ready to start.

# Directions

1. Find some hi-res images from the web that you’d like to use for this exercise.
   1. Make sure they are more than 1MB each in size to really see results.
   2. Save them into the img folder inside of the exercise folder.
2. For every image you have, add an image tag to the index.html file that is also included in the folder.
   1. Should be something like <img src=”img/<filename>.jpg”/>
3. Open Chrome:
   1. Go to localhost:8080/index.html
   2. Open the DevTools
4. Click on the “Network” tab along the top. If you don’t see it, you may have to increase the width of the DevTools or click on the icon of the double-arrow on the far right to see additional tabs.
5. Make sure the “Disable cache” checkbox is checked and refresh the page.
6. You should see the DevTools fill up with rows of data as all your page assets get loaded.
   1. Take note of the “Load” time in the bottom right-hand corner of the DevTools.
   2. Write that time in the table below in the Online / No Lazy Loading cell.
7. To the right of the “Disable cache” checkbox, you’ll see a drop-down menu currently with the value of “Online”.
   1. Click that drop-down and change the value to “Fast 3G”.
   2. Reload the page.
   3. Write the Load time in the table below in the Fast 3G / No Lazy Loading cell.
   4. Repeat with Slow 3G.
8. Now that we have some data for No Lazy Loading, let’s try With Lazy Loading. Change all your image tags from

<img src=”img/1.jpg”/>

To

<img class=”lazy” data-src=”img/<filename>.jpg” width=”500” height=”500”/>

* + Set an explicit height and width to your images. 500 is only here has a placeholder.
  + We need to change src to data-src because that is what prevents the browser from automatically loading the image. data-src is also what the jQuery Lazy plugin looks for when looking at what it should load next.
  + We also must specify a height and width. This is so that the browser knows how much real estate to reserve on the page for the image once it eventually loads. This prevents the content around our image from moving around when the image gets loaded.

1. Repeat steps 5-7, filling in the appropriate cells in the table, but now With Lazy Loading.

# Data

Total size of all your images added up together: \_\_\_\_ MB

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Page Load Time (s)**  **No Lazy Loading [A]** | **Page Load Time (s)**  **With Lazy Loading [B]** | **Efficiency Increase (%)**  **B / A \* 100** |
| **Online** | 489 ms | 59 ms | 828.81% |
| **Fast 3G** | 1.9 min | 2.71 s | 5313.65% |
| **Slow 3G** | 7.0 min | 9.88 s | 4048.58% |

# Observations

From the data collected, I noticed that:

There was a drastic spike in the effiiciency percentage from no lazy loading to with lazy loading.

# Bonus

Try compressing your images using tools like <https://tinyjpg.com/> and <https://tinypng.com/> to see how much more time you can shave off of your page load times!

# Submission

1. Rename this file *3.1-Lazy-Loading-Images-FIRSTNAME-LASTNAME.docx*
2. Submit on Moodle.