**01-Class-Content/02-css-bootstrap/01-Activities/01-Hack-Websites/**

**File**

* None

**Instructions**

* For the next 15 minutes, take a website that you commonly use (Amazon, Google, Huff Po, etc.) and heavily modify it using the Google Developer Tools.
* Be sure to at least modify:
  + Content (Change words)
  + Colors
  + Spacing
  + Etc.
* Slack out a screenshot to the class when you’re done.

**01-Class-Content/02-css-bootstrap/01-Activities/02-ModifyOwnWebsites/**

File

* None

Instructions

* For the next 10 minutes, take a website that you yourself worked on (ex: In-Class activities, homework assignments, etc.), and utilize the Google Developer Tools to help you test changes in real-time.

NOTE: Focus on getting more comfortable using the Developer Tools. Trust us. You will WANT to use these Developer Tools as you proceed in this course.

**01-Class-Content/02-css-bootstrap/01-Activities/04-ResetCSS/**

**### File**

\* [`example`](example.html)

**### Instructions**

\* Follow the instructions in the HTML document to incorporate a reset.css file into a basic HTML file.

\* Student is to find a reset.css from Google and put it into their directory

\* \_Note the impact the reset file makes after its inclusion.\_

\*\*\* How can you determine the styles that the browser is using by default (Without the reset being used)?\*\*

\* Discuss with a partner: Why are resets important?

\*\*\* HINT:\*\* You should be using the CSS made available on this link: <http://meyerweb.com/eric/tools/css/reset/>

\* \*\*BONUS:\*\* Incorporate a CSS reset into any page you've worked on to date.

**01-Class-Content/02-css-bootstrap/01-Activities/05-GithubPagesPersonal**

**Activity 5 - GitHub Pages Personal Bio**

Now that we have gone over how to create a personal website using GitHub Pages, it is time to publish your own personal website! See the guide below as a reminder of what steps you need to take.

**Instructions**

* Build a basic HTML page and deploy it to github pages.
* Once your personal webpage is live, slack out the link to it so that everyone can see what you have made.

**Guide**

1. Create a new repo that is named <<YOUR USERNAME>>.github.io where <> is your GitHub username. Double check that you use exactly your username. (For example, janedoe.github.io would be the GitHub pages repo name for the GitHub user "janedoe")
2. Navigate into a folder and clone the repo into it
3. Add an HTML file named index.html and code out a basic webpage (or use a previous page)
4. Add, commit, and push your changes into the repository
5. Navigate to <<YOUR\_USERNAME>>.github.io and you will find that your new web page has gone live! (again, where <> is your GitHub username. - we just use that as a placeholder.)

**Bonuses**

* Spruce up your personal website page: add extra tags and flesh your page out some more - give it some pop
* Push updates to your personal website multiple times. You'll see it update on the live URL!
* Add a few new pages to your website
* Add CSS to your deployed webpage, including a CSS Reset

**01-Class-Content/02-css-bootstrap/01-Activities/06-GithubPagesProject/**

**# Activity 6 - GitHub Pages Project Webpage**

In this activity you'll practice creating github pages with projects. You'll build a website about your favorite type of animal!

**### Instructions**

Make a simple website about your favorite type of animal.

\* Create a header that will act as a title for the page

\* Create a few short paragraphs describing the animal's best characteristics.

\* Include at least one picture of the animal

\* Once you have created your HTML add, commit, and push your files up to GitHub Pages.

\* Make sure to navigate to your page in the web browser to ensure your website is live

**### Guide**

1. Create a new repository on your GitHub account. \*You can name this repository whatever you would like.\*

2. Once inside of the repository, create a new file and name it `index.html`

3. Add your HTML into this file, save it, and then navigate into your repository's `Settings` tab.

4. Scroll down to the GitHub Pages section and then, in the section labeled `Source`, select that you would like to use the `master` branch as your source.

5. Navigate to `<<YOUR USERNAME>>.github.io/<<REPO NAME YOU CHOSE>>` and you will find that your new web page has gone live! (For example, if your GitHub username is `johndoe` and the project is `cssDemo`, your URL would be `johndoe.github.io/cssDemo`)

**### Bonuses**

\* See if you can use your personal site from the previous activity as a portfolio that can be used to show off all of the work you have done in the past. Update it to link to this activity's site.

\* Try to create a navigation system which links from your personal website to this new animal site and vice-versa.

\* Have some fun! Test your HTML and CSS skills by going above and beyond with your page!

**01-Class-Content/02-css-bootstrap/01-Activities/07-CSSTypography/**

**# CSS Typography**

In this activity, you'll be modifying CSS properties to style your text in different ways.

**## Instructions**

\* Using the `html-template` file provided to you, create an external CSS stylesheet to modify the typography of each of the body paragraphs.

\* Be sure to modify the following CSS properties for each paragraph of your page:

\* font-family

\* font-size

\* font-weight

\* line-height

\* Hint: You will need to do three things to accomplish this task.

\* Link to a CSS page that you'll make

\* Create classes for each paragraph

\* Add in styles for each class

**## Bonus**

\* If you finish early, read about [TypePlate - http://typeplate.com/](http://typeplate.com/) and see if you can figure out how to incorporate it into your site.

\* If you still finish early, spend some time trying to actually make the page look nice. Think about what's aesthetically pleasing and push yourself to create it.

**01-Class-Content/02-css-bootstrap/01-Activities/08-GoogleFontDemo/**

**# Google Fonts**

This activity demonstrates how to incorporate new fonts into a web page.

**## Instructions**

\* Create (or re-use) an HTML Document of your choosing and then incorporate 2-3 Custom Google Fonts into the page.

**## Bonus**

\* If you finish early, read about Font Awesome icons and see if you can figure out how to incorporate one onto your page.

**01-Class-Content/02-css-bootstrap/01-Activities/09-Pseudoclass/**

**# :Pseudo-classes**

In this activity you'll use psuedo-classes to style elements based on their state.

**## Instructions**

\* Create a series of links and modify the pseudo-classes associated with their Default, Active, Hover, and Focus States.

\* Hint: Nope. No starter code. Gotta do this from scratch :-)

**## Bonus**

\* If you finish early, incorporate a pseudo-class we haven’t covered in your page. Then read about “pseudo-elements” and try to incorporate one as well.

\* Slack out the code and screenshot if you end up completing the bonus.

\* \*\*Extra\*\* - Learn about css transitions to add a fading hover effect to a link ([\\&lt;a](file:///\\&lt;a)>)

**01-Class-Content/02-css-bootstrap/01-Activities/10-WheresCSS/bootstrap-powered.html**

**01-Class-Content/02-css-bootstrap/01-Activities/11-BootstrapDemo/**

**# Bootstrap Demo**

In this activity, we'll get some experience working with Bootstrap components.

**## Instructions**

\* Working with someone closest to you:

\* Navigate to the Twitter Bootstrap website.

\* Then copy the link to the Bootstrap CSS file into one of your old HTML files.

\* Look through the Bootstrap CSS or Components list and incorporate at least three Bootstrap elements onto your page.

\* Slack a screenshot of your page to your section’s channel when you are done.

\* Hint: First include a link to Bootstrap’s CSS. Then visit the Bootstrap page on CSS or Components.

\* If you finish early, let the TAs know so you can be sent to help others around you.

**01-Class-Content/02-css-bootstrap/01-Activities/12-WarmupBootstrap/**

**### File**

\* \*None\*

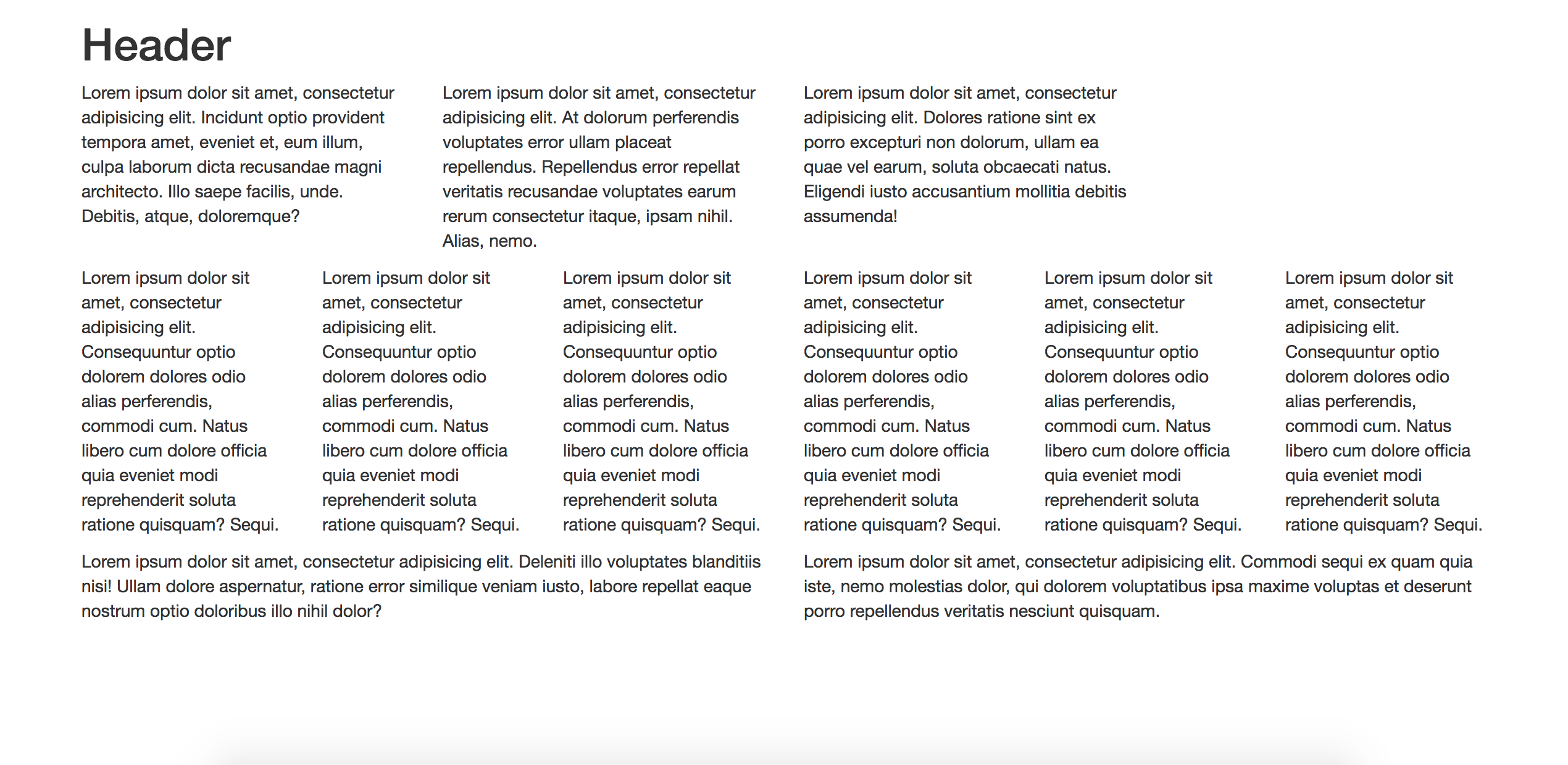
**### Instructions**

\* As a repeat of last class, quickly create an HTML web page.

\* Then add in a Bootstrap component of your choosing.

\* \*\*HINT:\*\* Copy the Bootstrap CSS link. Then copy the code from any Bootstrap element or CSS.

**01-Class-Content/02-css-bootstrap/01-Activities/14-TextLayout/Images/text-layout.png**



**01-Class-Content/02-css-bootstrap/01-Activities/14-TextLayout/**

**### File**

\* \*None\*

**### Instruction**

\* Using Twitter bootstrap, create a layout that looks like:

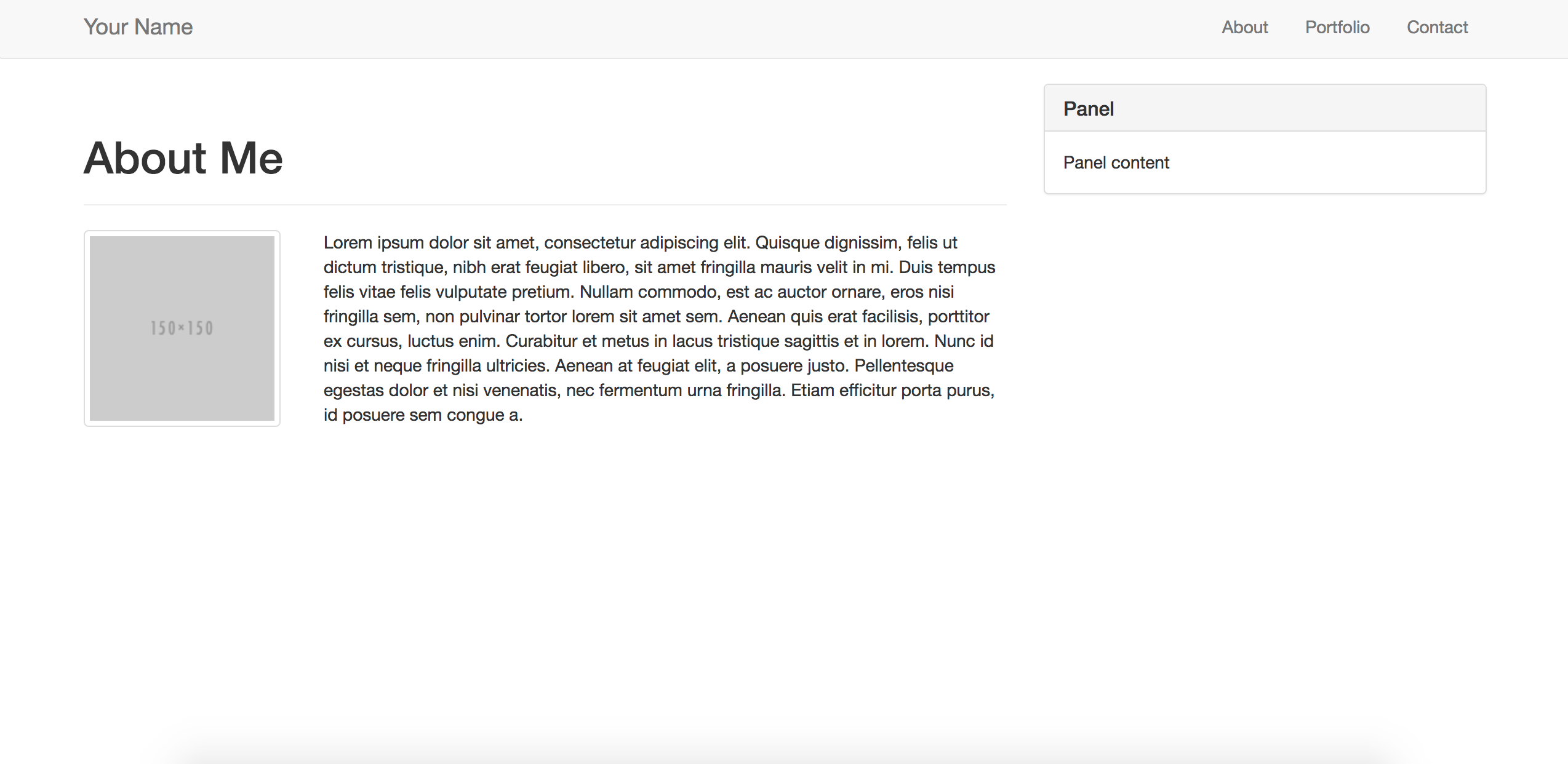
![text-layout activity design](Images/text-layout.png)

\* \*\*HINT:\*\* Spend some time prior to coding on drawing out the grid layout.

\* \*\*HINT:\*\* Count the number of rows and columns.

\* \*\*HINT:\*\* Then use the bootstrap grid syntax to code it out.

**01-Class-Content/02-css-bootstrap/01-Activities/16-PanelLayout/Images/panel-layout.png**



**01-Class-Content/02-css-bootstrap/01-Activities/16-PanelLayout**

**### File**

\* \*None\*

**### Instructions**

\* Working in pairs and using Twitter Bootstrap make a page that looks like the following image:

![panel-layout design](Images/panel-layout.png)

\* Be sure to note the:

\* Grid Layout

\* Navbar

\* Sidebar panel

\* Thumbnail

**01-Class-Content/02-css-bootstrap/01-Activities/18-StudentMedia**

**### File**

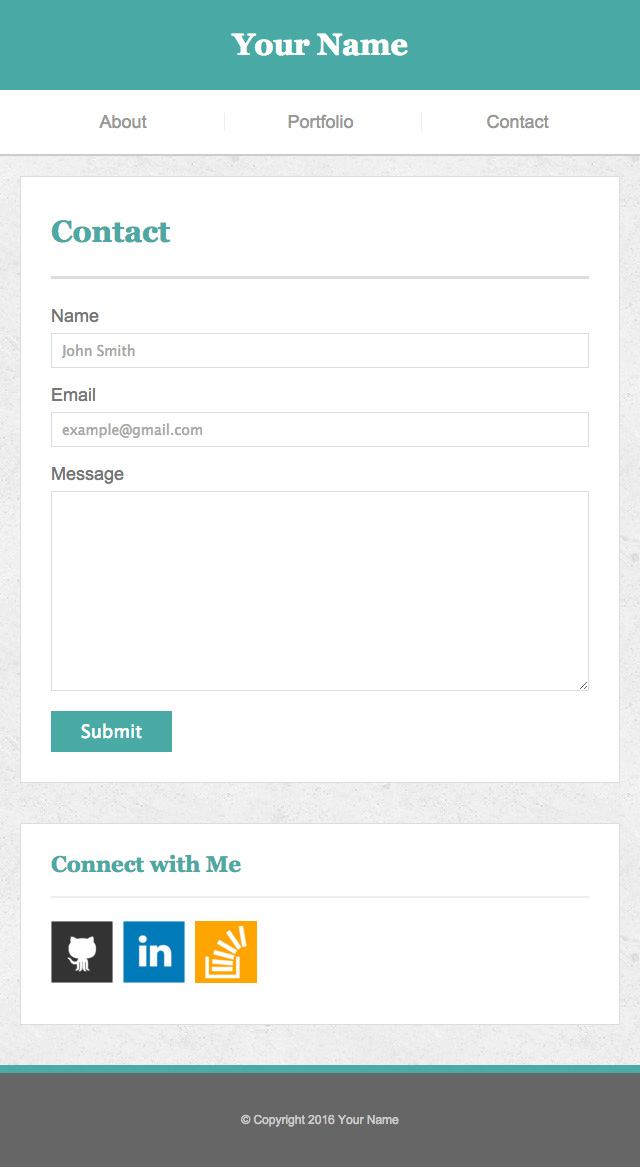
\* [`media-unsolved`](Unsolved/media-unsolved.html)

**### Instructions**

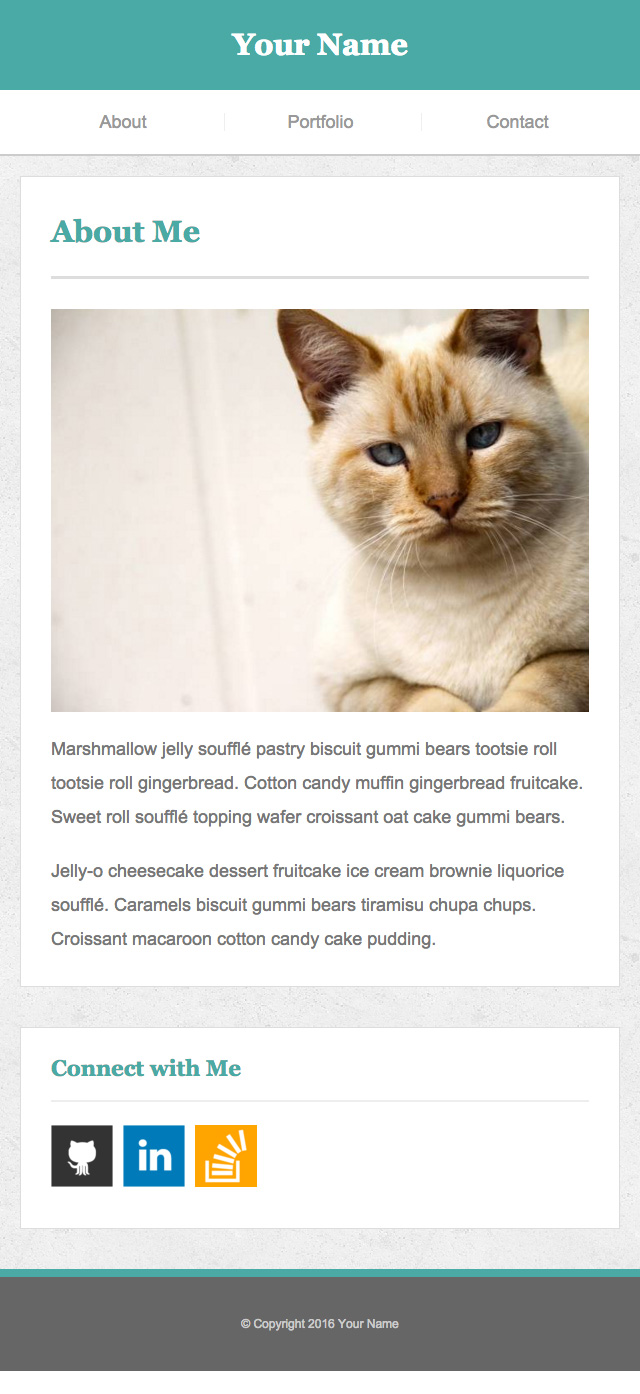
\* Add more styling to the CSS file such that the `@media screen` code creates padding and margins for the two boxes when the page is below 768px.

\* \*\*NOTE:\*\* This isn't easy. Just give it your best shot. Even if you don't get the exact solution, but are able to better understand how media queries work, that's a solid takeaway.

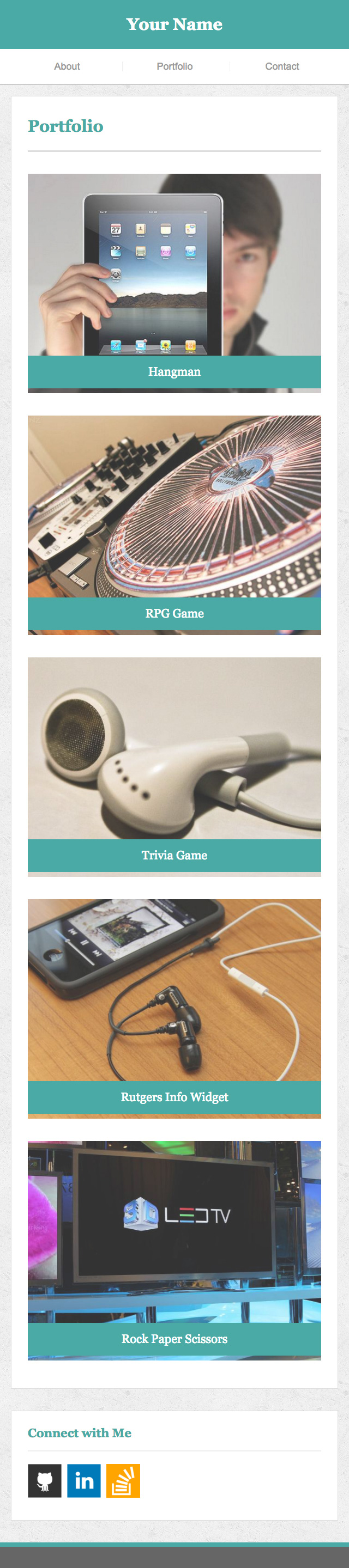
**01-Class-Content/02-css-bootstrap/02-Homework/Instructions/Images/640-contact.jpg**



**01-Class-Content/02-css-bootstrap/02-Homework/Instructions/Images/640-index.jpg**



**01-Class-Content/02-css-bootstrap/02-Homework/Instructions/Images/640-portfolio.jpg**



**01-Class-Content/02-css-bootstrap/02-Homework/Instructions/homework-instructions.md**

**Responsiveness Assignment**

Overview

In this assignment, you'll create two different portfolios. The first will be an update of the one you made last week. You'll enhance the portfolio you made last week with a mobile-responsive layout. For your second portfolio, you’ll build a layout using the Bootstrap CSS framework.

Before You Begin

* You've learned a ton of material these past two weeks: HTML, CSS, GitHub, GitHub Pages, and Bootstrap. If you feel like you're falling behind, there's no need to panic. You'll have 22 weeks to digest and master this material.
* We're diving into JavaScript next week, and HTML/CSS will start receiving less focus. Still, you'll find that a basic knowledge of HTML/CSS will help you understand JavaScript, especially when we use it to manipulate web pages.

Instructions

1. Create two new GitHub repositories and name them Responsive-Portfolio and Bootstrap-Portfolio.
2. Clone these repositories to your computer.
3. Copy the contents of Basic-Portfolio (your first homework solution) and paste the mentioned files into Responsive-Portfolio.
4. Note: Be sure not to include any dot files (e.g. .git, .gitignore) from the Basic-Portfolio repo.
5. If you chose the Skeleton exercise for your first homework assignment, contact a TA, who will provide you with a template for your portfolio.

Assignment One Instructions - (No Bootstrap)

1. Inside your Responsive-Portfolio folder, find your styles.css file. You will write your media queries at the bottom of styles.css.
2. Use three @media screen tags, each with one of these max-widths: 980px, 768px and 640px.
   1. You use 980px because you never want any of the content to be cut off. Since the desktop layout is about 960px wide, you want the media queries to kick in before your content gets cut off.
   2. 768px is about the width of a tablet and 640px is about the width of a phone in landscape.

Make the layout match the following screenshots:

1. index.html: 980px, 768px, 640px
2. portfolio.html: 980px, 768px, 640px
3. contact.html: 980px, 768px, 640px
4. Make the position of the header static (the default positioning) when the screen is 640px wide. The header design takes up a lot of room; you don't want it to stick to the top of a small screen and leave no room for the rest of your site.
5. Be sure to include the viewport tag in all your HTML files, otherwise your media-queries won't function as expected on mobile devices. (Hint: You won't need to use exact pixels for anything other than the container)
6. **Protip**: Use the Chrome extensions Window Resizer and Browser Width to see the browser dimensions in Chrome.
7. Deploy your new portfolio (now with media queries!) to GitHub Pages.

**Assignment Two Instructions (Bootstrap)**

1. Inside your Bootstrap-Portfolio repo, create index.html, portfolio.html and contact.html.
2. Using Bootstrap, recreate your portfolio site with the following items:
   1. A navbar
   2. A responsive layout (remember the grid, rows and columns are your friends)
      1. eg. On xs and sm screens, each section should take up the entire grid. On m and larger screens, each section should take up 2/3 of the grid and the sidebar should take up 1/3 of the grid
   3. Responsive images

**BONUS**

Using Bootstrap, make a sticky footer and use sub-rows and sub-solumns on your portfolio site (Hint: Check out the Bootstrap documentation)

1. Your Bootstrap solution should minimize use of media queries.
2. Deploy your new Bootstrap-powered portfolio to GitHub Pages.

**Submitting Your Work on BootcampSpot.com**



**BONUS**

Incorporate CSS animations in your portfolio. [More info here.](http://www.w3schools.com/css/css3_animations.asp)