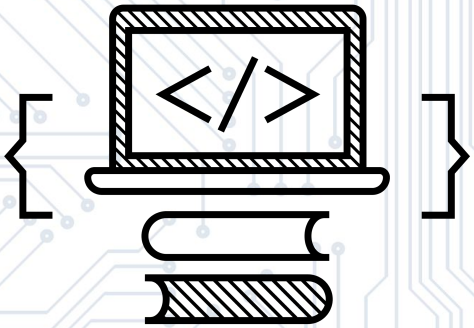


Going Live

Web Development Boot Camp

Lesson 2.1

Objectives

In today's class, we will:

01 Recap the basic topics covered to date.

02 Offer a conceptual introduction to CSS reset and downloadable style sheets.

03 Work with Chrome DevTools for site inspection.

04 Utilize GitHub Pages for website deployment.

Checking In

How’s it going?

Instructor Feedback

Seriously? Mind. Blown.

Instructor Feedback

Things I've noticed people doing incredibly well:

- All of you are handling an enormous volume of information.
- All of you are asking the right questions.
- You notice the right details.
- You all help each other out.
- And, most importantly, you are figuring out things on your own.

A Few Admin Things

Homework

Remember, Homework 1 is due on Saturday, July 18, 2020 @ 11:59 PM.

- Homework link: https://bootcampspot.com/coursework
- Remember to submit homework via GitHub and GitHub Pages: github.com/[username]/[repository_name]
- And seriously, submit whatever you have, even if you don't like what you've made. Not submitting your assignment will result in a 0.

Office Hours and Additional Help

Remember:

- Office hours are held 45 minutes before class and 30 minutes after class.
- Review in-class material (activities and slides): https://oregon.bootcampcontent.com/Oregon_Coding_Bootcamp/uofo-por-fsf-pt-07-2020-u-c
- Re-watch class videos: https://bootcampspot.com/sessions -> choose session -> at top will be link to recorded videos

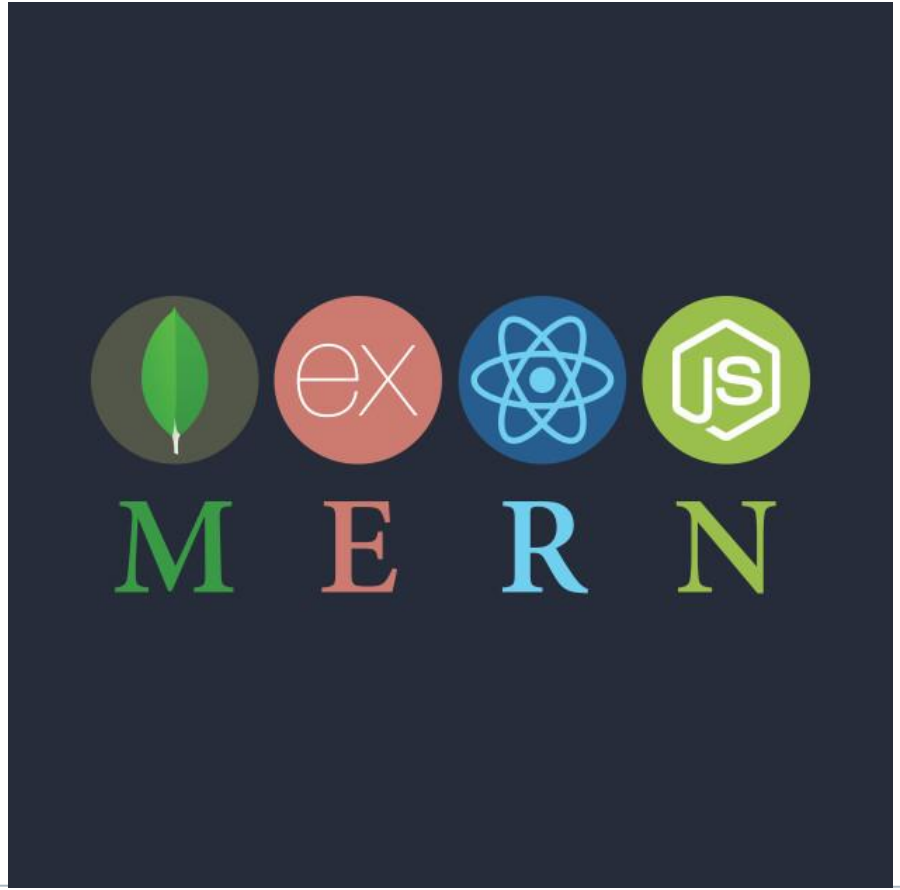
Recap

Recap

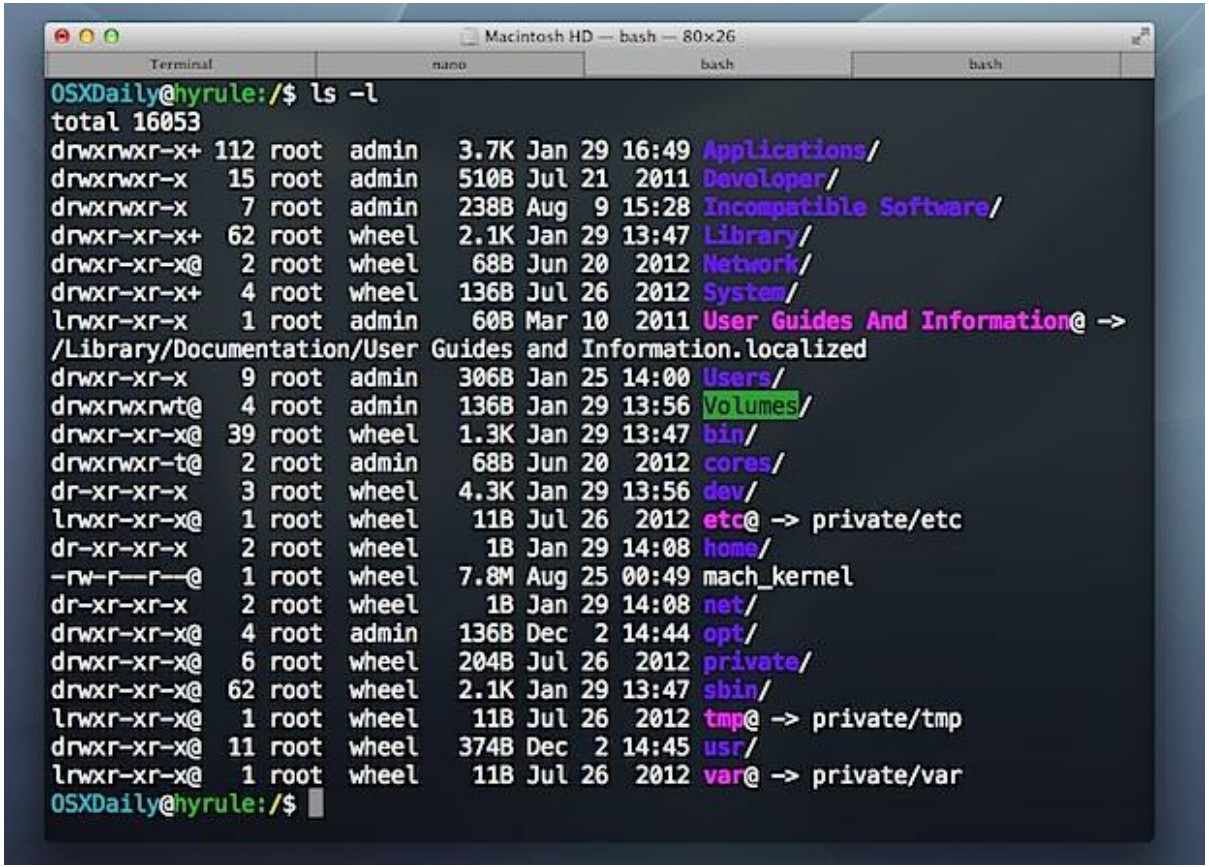
In just one whirlwind week we've covered:

- Basics of full-stack development
- Terminal and Git Bash
- HTML syntax
- Git concepts and commands
- CSS purpose, syntax, and styles
- Floating
- Positioning
- Box model
- Chrome DevTools
- **How to learn on your own!**

What Is Full-Stack Development?



> Intro to Console

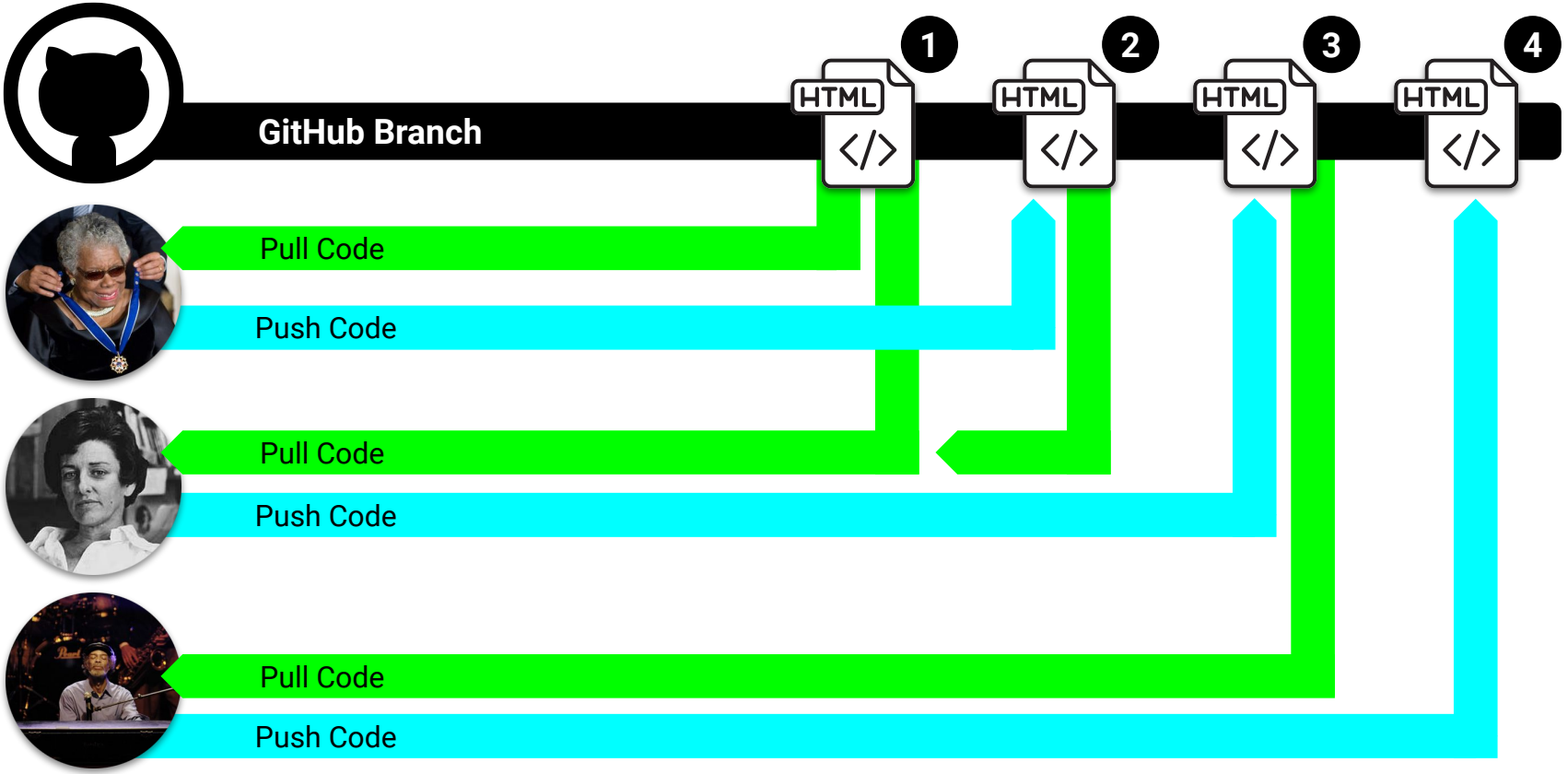


<title> Intro to HTML </title>

HTML is one of the three base languages behind every website.

It defines all of the basic content as well as a bit of formatting.

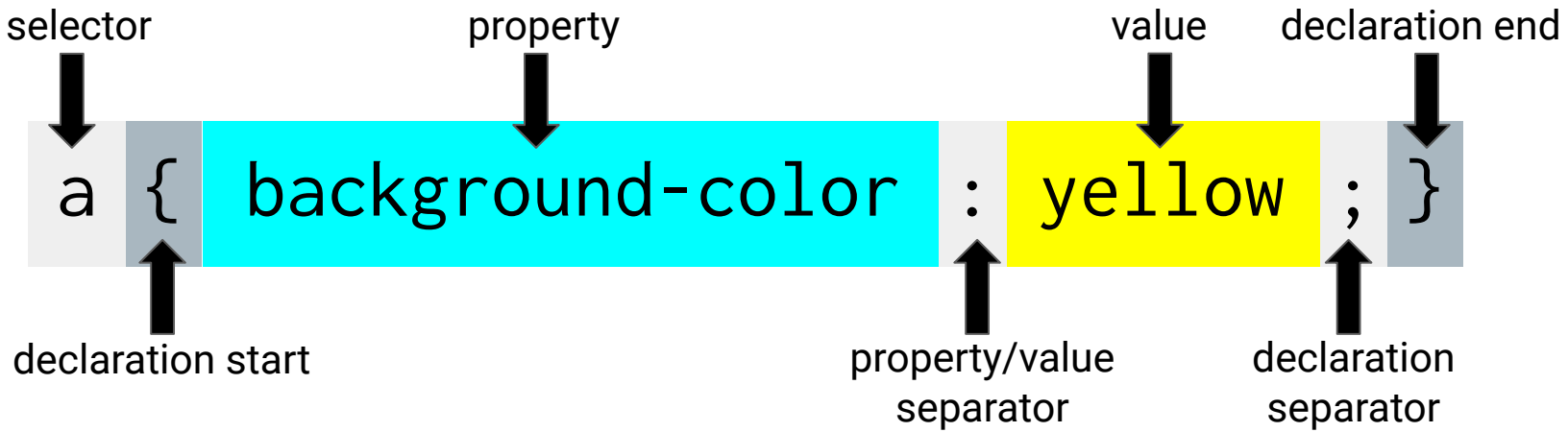
Pushing and Pulling to GitHub



CSS Syntax

CSS works by hooking onto **selectors** added into HTML using **classes** and **identifiers**.

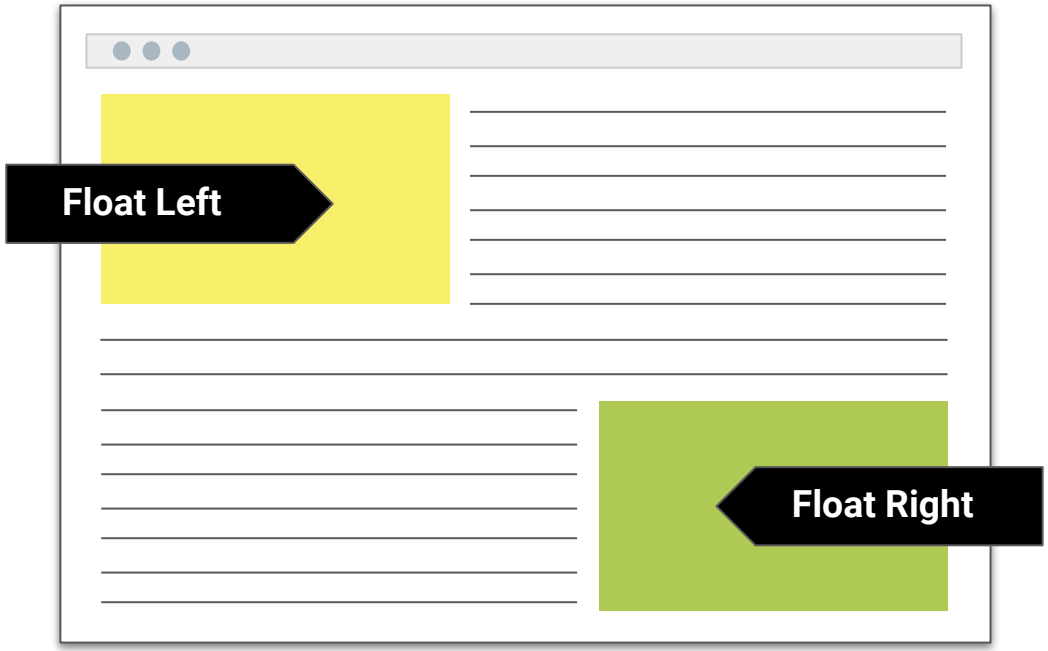
Once hooked, we apply **styles** to those HTML elements using CSS.



The Concept of Flow

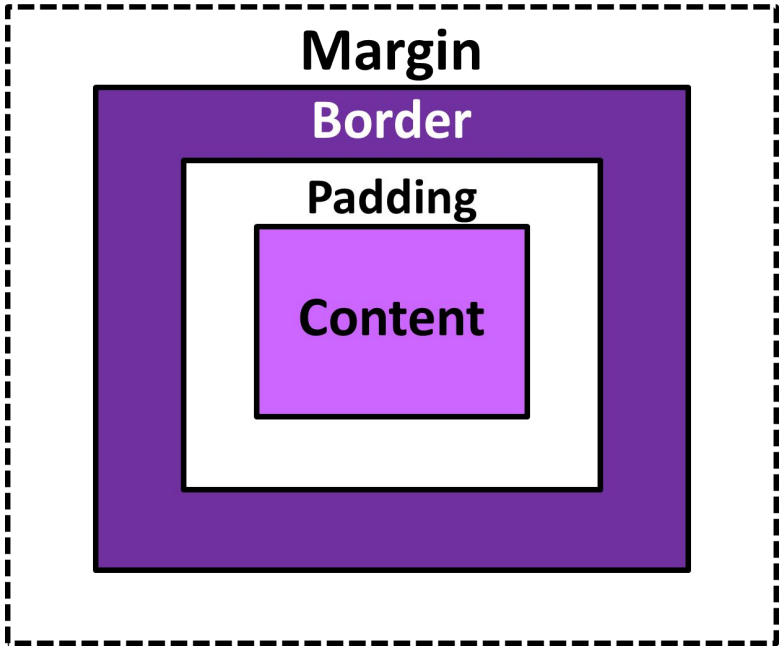
By default, every HTML element displayed in the browser is governed by a concept called **flow**.

This means that HTML elements force their adjacent elements to flow around them.



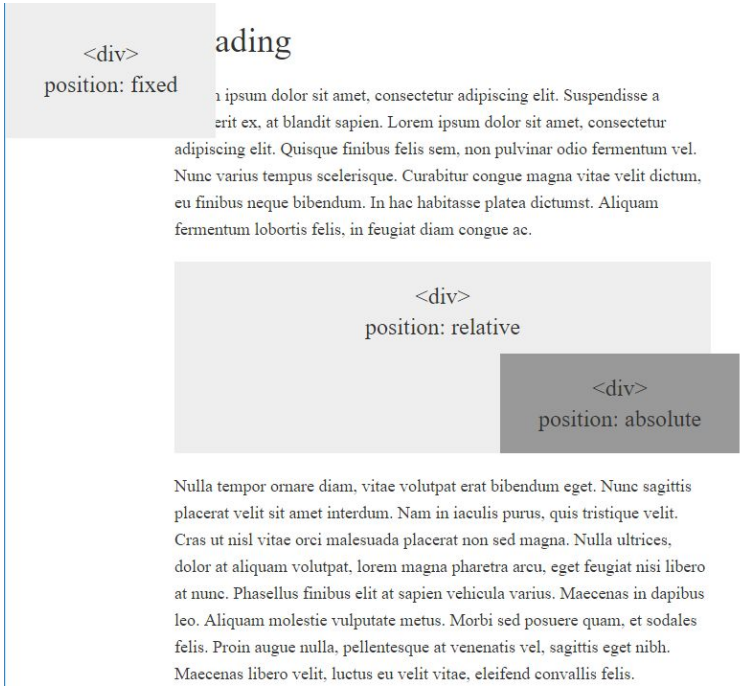
The Box Model

The CSS **box model** wraps every HTML element in a box. This box consists of padding, border, margin, and content, and allows developers to modify spacing styles.



CSS Positioning

We can orient our HTML elements in relation to space with CSS positioning (static, relative, fixed, absolute).



Resources

- stackoverflow.com
- w3schools.com
- designshack.net
- css-tricks.com
- smashingmagazine.com
- sitepoint.com
- developer.mozilla.org
- cssnewbie.com

General Questions/Issues?

Double Take

Divs and Sections

stackoverflow.com

What is the difference between `<section>` and `<div>`?

What is the purpose of new HTML5 elements like `<section>` and `<article>`?

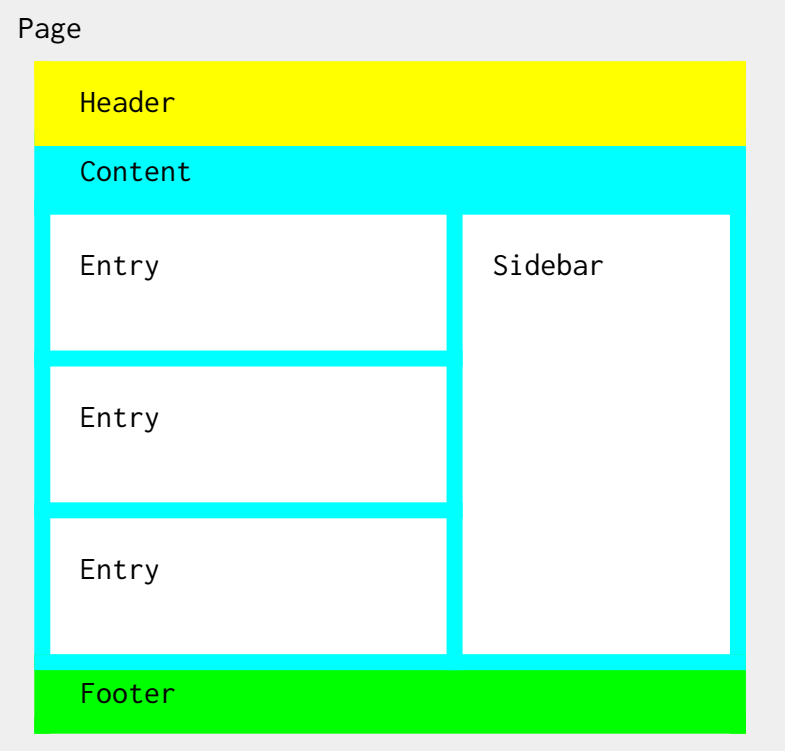
Why would you use an HTML5 semantic tag instead of `<div>`?

When do you use a `<div>` element, and when do you use a `<section>` element?

Which should you use for semantic markup and page layout: `<section>` or `<div>`?

Divs and Sections

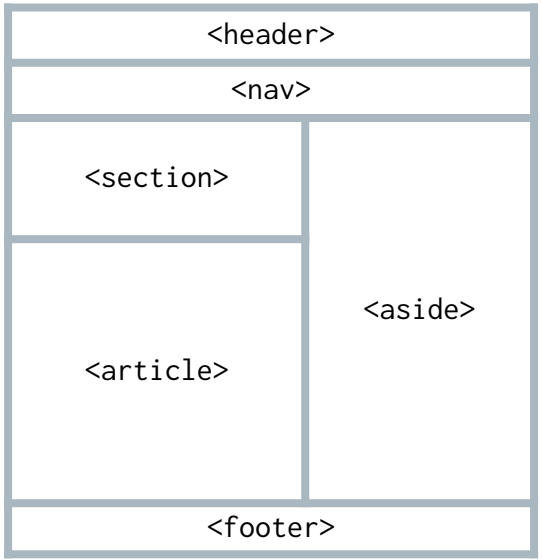
All web layouts are made up of containers, traditionally called **divs**.



Divs and Sections

HTML5 introduced the concept of **semantic layouts,** meaning divs could be given more meaningful names. In theory, this helps with organization and search engine optimization.

Website Layout Using HTML5:
HTML5 offers new semantic elements that define different parts of a webpage:



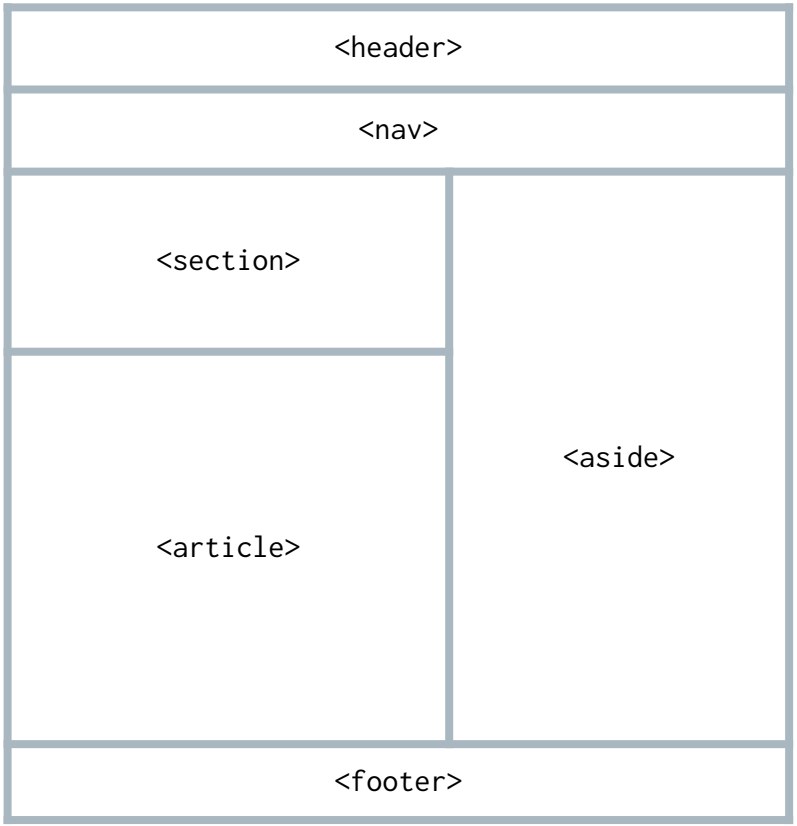
`<header>`	Defines a header for a document or a section
`<nav>`	Defines a container for navigation links
`<section>`	Defines a section in a document
`<article>`	Defines an independent self-contained article
`<footer>`	Defines a footer for a document or a section
`<details>`	Defines additional details
`<summary>`	Defines a heading for the `<details>` element

Divs and Sections

That said, it seems many, if not most, websites still use basic divs.

There are reasons for this that we'll showcase in later lessons.

Additionally, it's possible to include semantics by using ID names and classes.



Divs and Sections

Bottom line: Follow the homework instructions. And when you're out in the real world, follow the convention of where you work!

`div?`

`section?`

Classes vs. IDs

When choosing between a CSS ID and a CSS class, follow the convention:

Classes (.classname) are used if the same style will be used on multiple HTML elements.

IDs (#idname) are used if a style is unique to a certain HTML element.

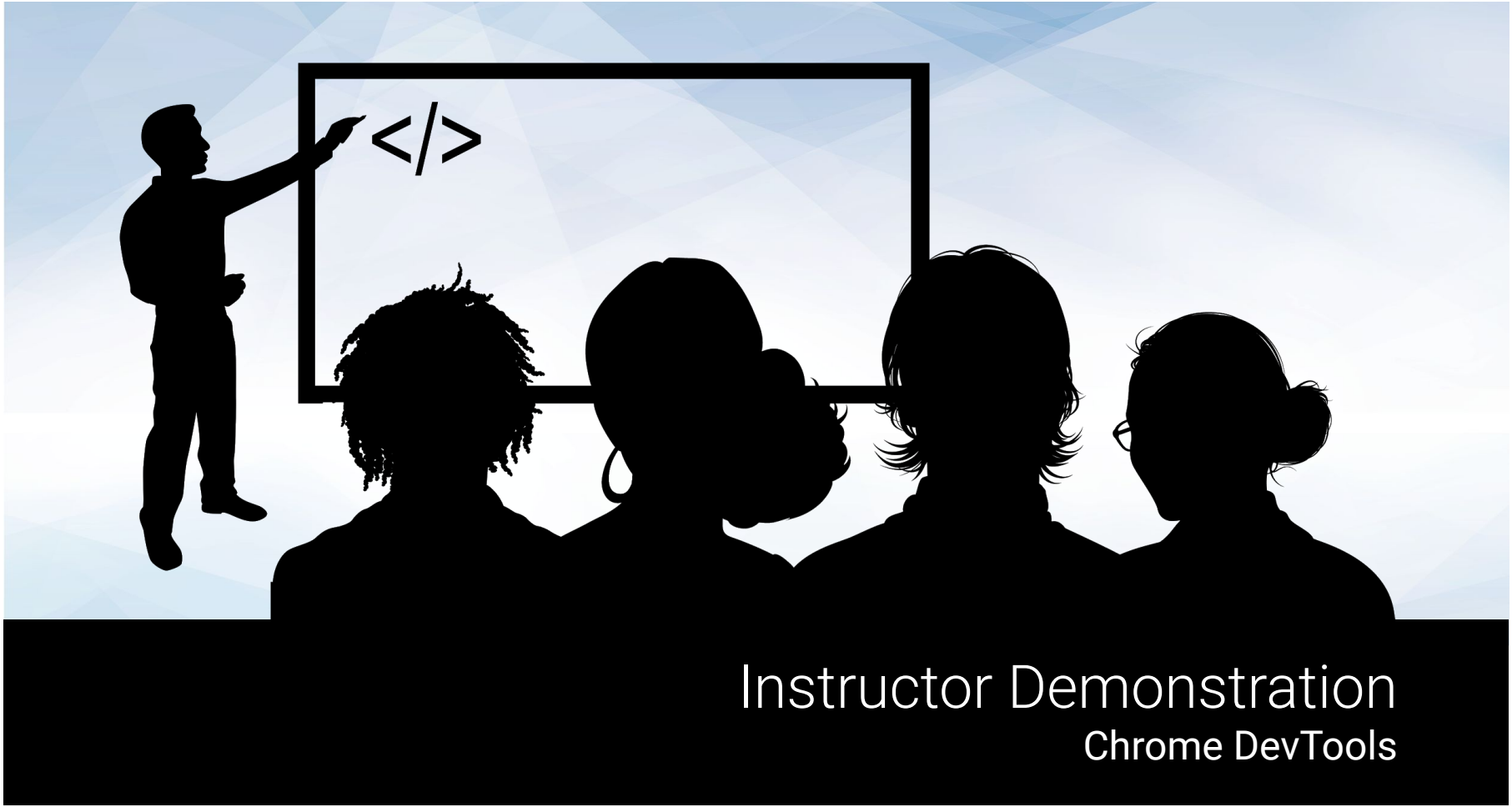
Chrome DevTools (Inspector)

This is one of the web development tools you will use most frequently. It allows you to investigate how and why HTML is rendering.

Start using it!

Modifying Sites

You can edit any page's HTML and CSS with Chrome DevTools.
Plus, you'll see your results instantly.



Instructor Demonstration

Chrome DevTools

Activity: Modify a Website

For the next 15 minutes, take a website you commonly use (Amazon, Google, HuffPo, etc.) and heavily modify it using Chrome DevTools.

Be sure to modify the following:

- Content (change words)
- Colors
- Spacing

Send a screenshot to the class's Slack channel when you're done.

Suggested Time: 15 minutes

Practice Through Frustration

Keep practicing. It gets better!

Activity: Modify Your Own Website

For the next 10 minutes, edit any site that you've been working on—in class or for homework—with Chrome DevTools.

Be sure to at least modify:

- Content (change words)
- Colors
- Spacing

Suggested Time: 10 minutes

Take a Break!

CSS Resets

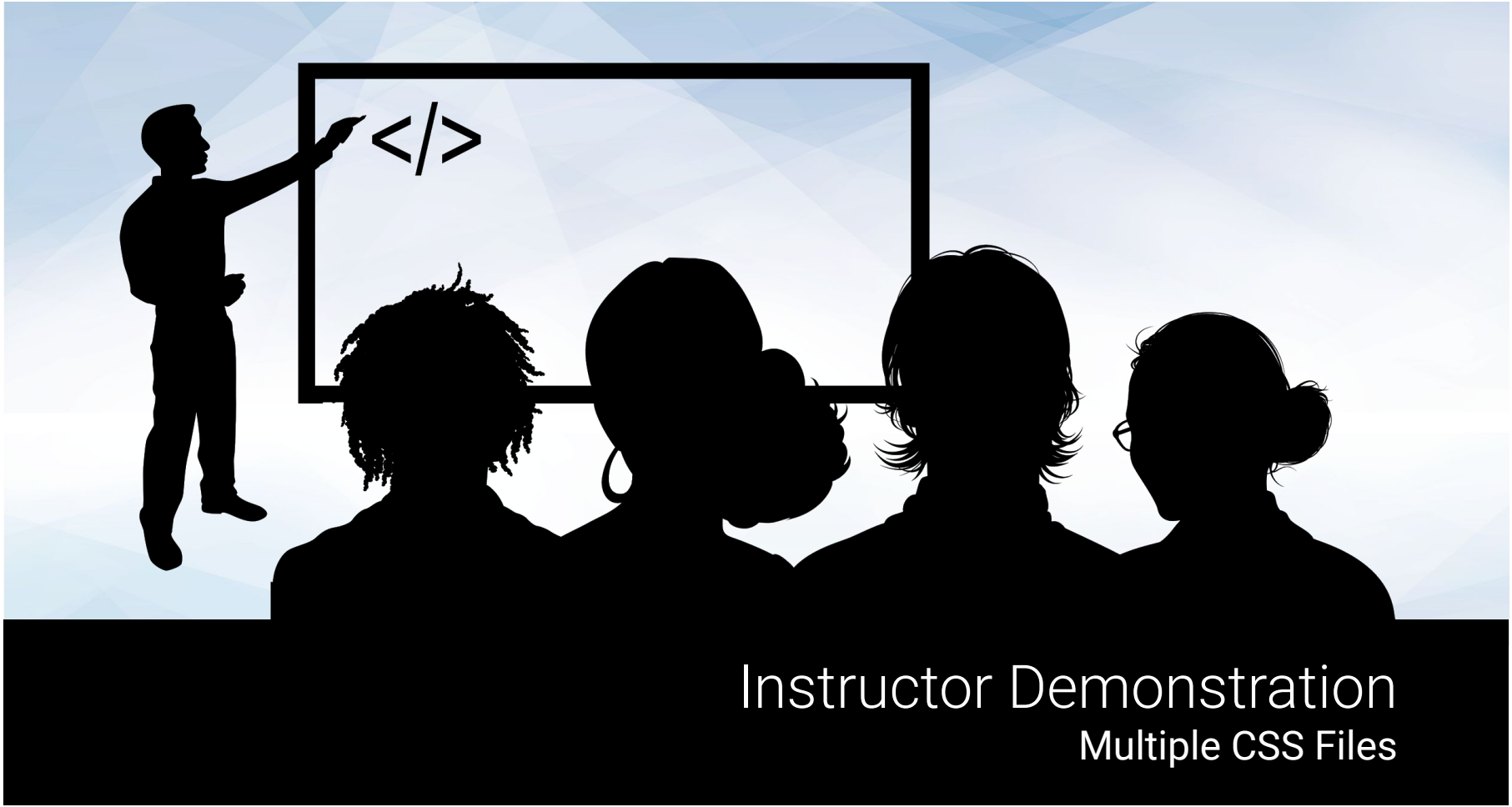
Loading Multiple CSS Files (**Important!**)

Deploying multiple CSS files simultaneously is a powerful technique.

This technique allows developers to create complex designs made up of abounding design elements.

Remember, the loading order matters!

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Multiple CSS Files!!</title>
5      <link rel="stylesheet" href="assets/style1.css">
6      <link rel="stylesheet" href="assets/style2.css">
7      <link rel="stylesheet" href="assets/style3.css">
8  </head>
9  <body>
10     <header>
```



Instructor Demonstration

Multiple CSS Files

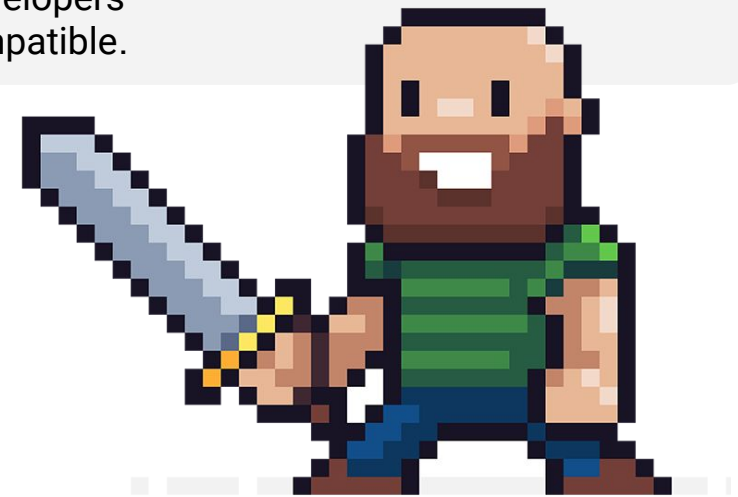
By a show of hands,
which browser do you use?

Battle of the Browsers

Under the hood, web browsers often render webpages differently than their competition.

These disparities could mean that the HTML/CSS displays differently in each web client.

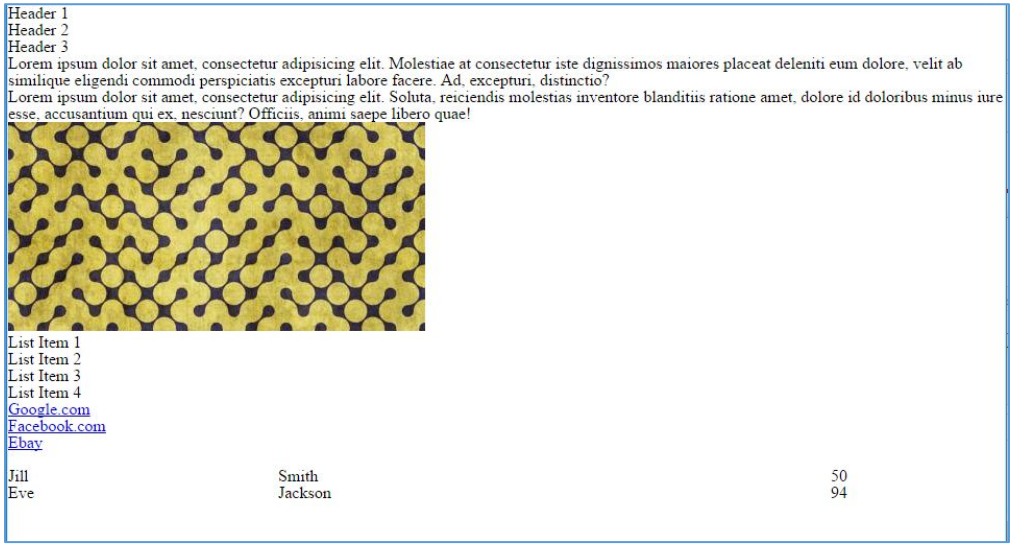
Because of these potential differences, web developers need to make their websites cross-browser compatible.

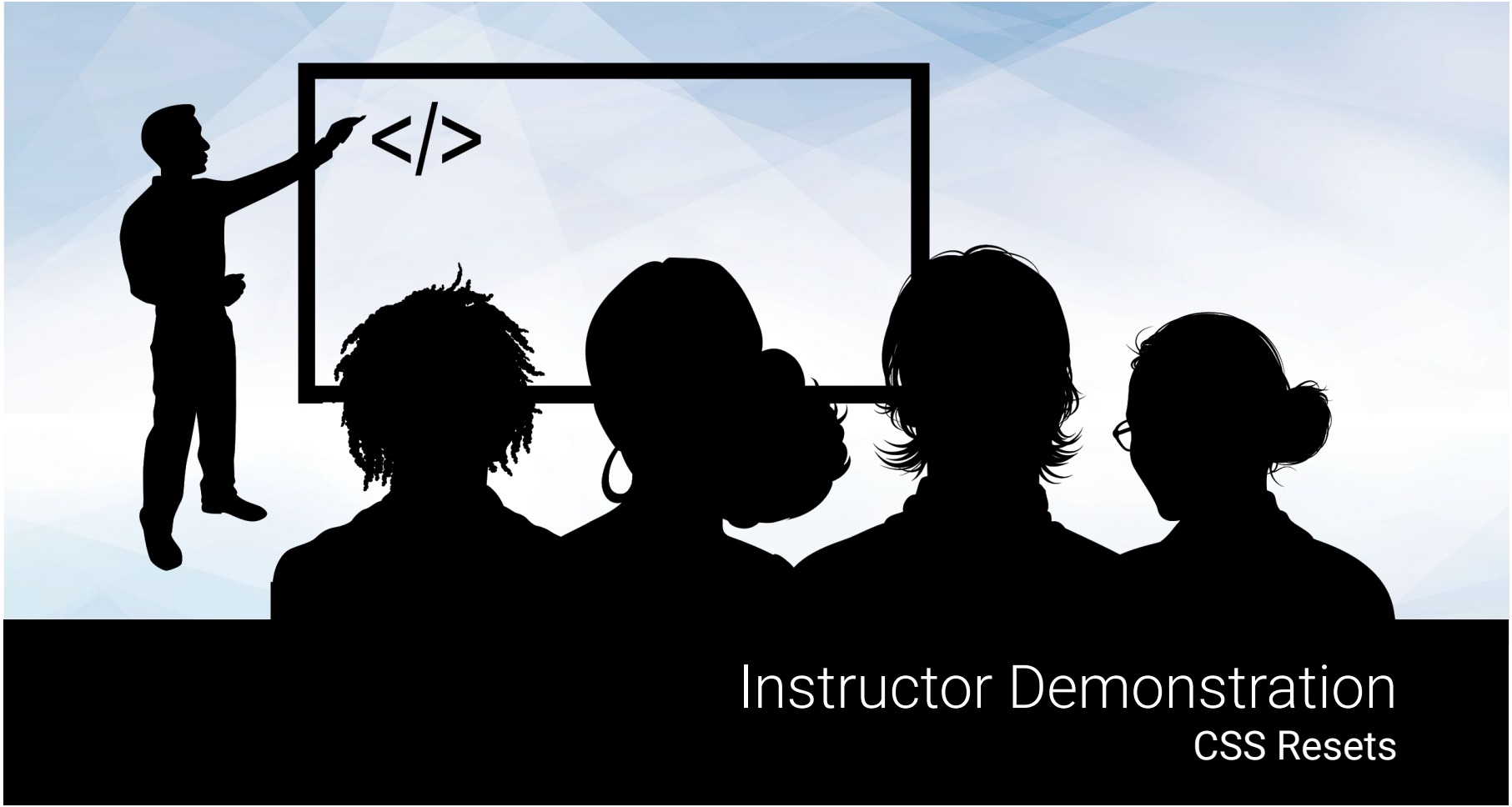


Reset.css (or Normalize.css)

Reset.css will “reset” all browser-specific CSS. This means your site will appear the same in all browsers.

However, you will have to restyle everything yourself.





Instructor Demonstration

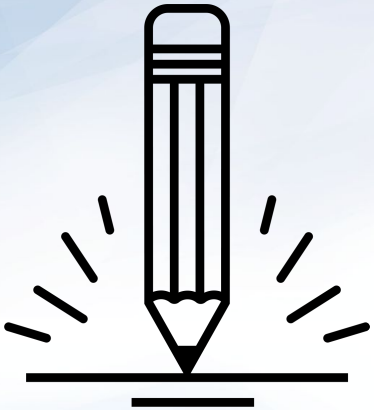
CSS Resets

Why CSS Resets Matter

01 They are important for creating browser-compatible websites.

02 They are an example of using someone else's CSS in your website.

03 They are a common topic for front-end developer interview questions.



Activity:
CSS Resets

Suggested Time:
10 minutes

Activity: CSS Resets

Follow the instructions provided via Slack to incorporate a reset.css file into a basic HTML file.

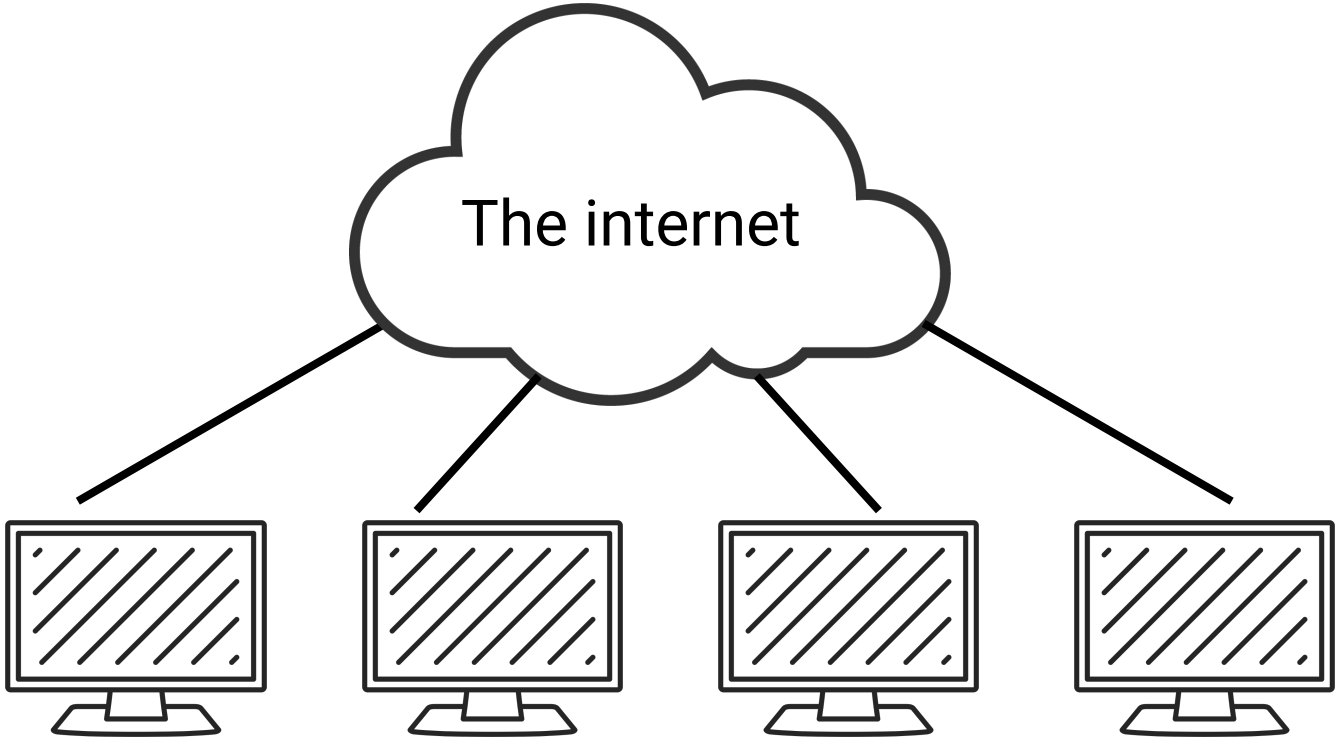
Note the impact the reset file makes after its inclusion.

Suggested Time: 10 minutes

To the Web with GitHub!

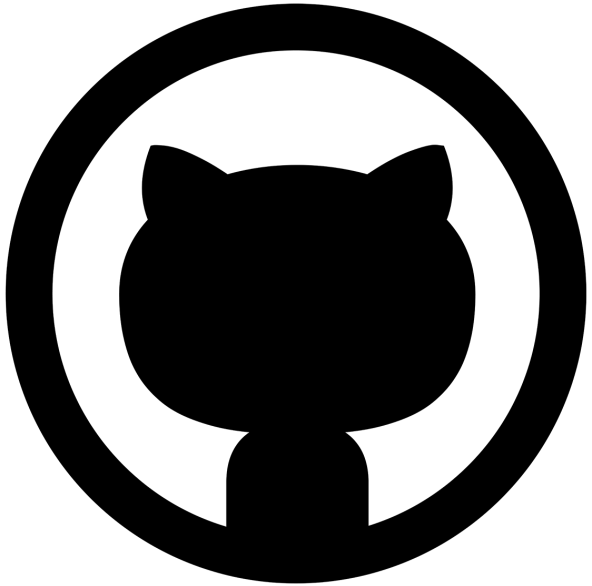
The Internet

Here is a deep and complex diagram illustrating how the internet works.

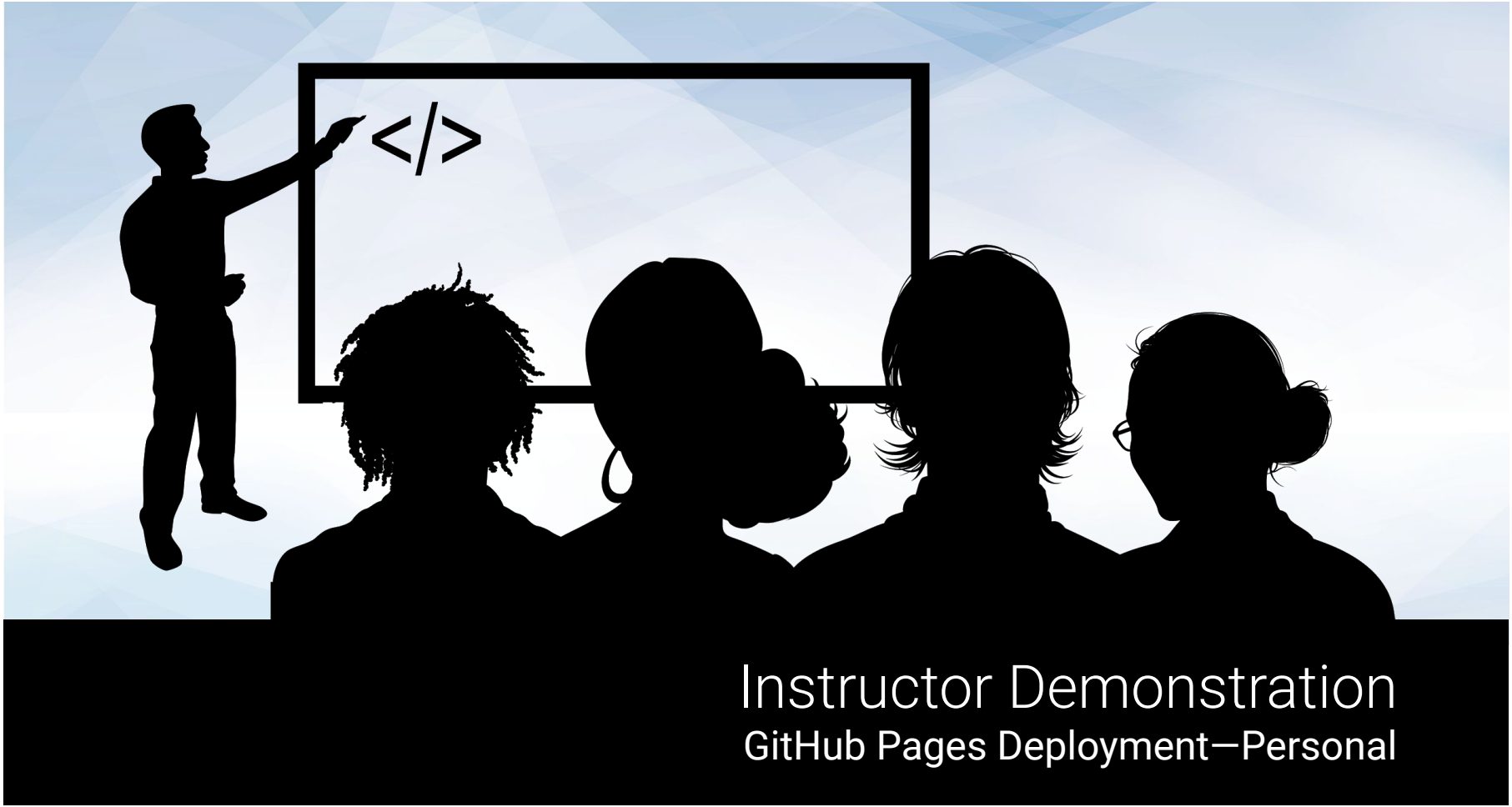


The World Will See Our Greatness!

GitHub provides hosting for static websites, which means we can deploy our websites and applications onto their servers for the world to see.



Let’s All Log In to GitHub



Instructor Demonstration

GitHub Pages Deployment—Personal

Deploying a Static Personal Site to GitHub Pages

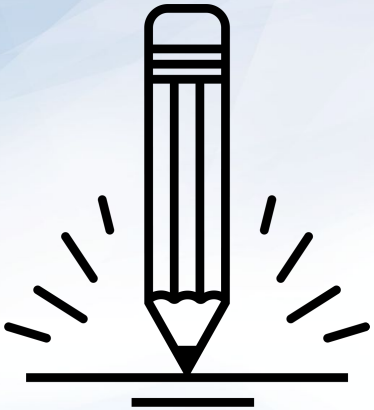
Follow these basic steps:

01 Create a new repo named _username_.github.io.

02 Navigate to a folder on your computer and clone the repo into it.

03 Build your files.

04 Add, commit, and push your changes to the repository.

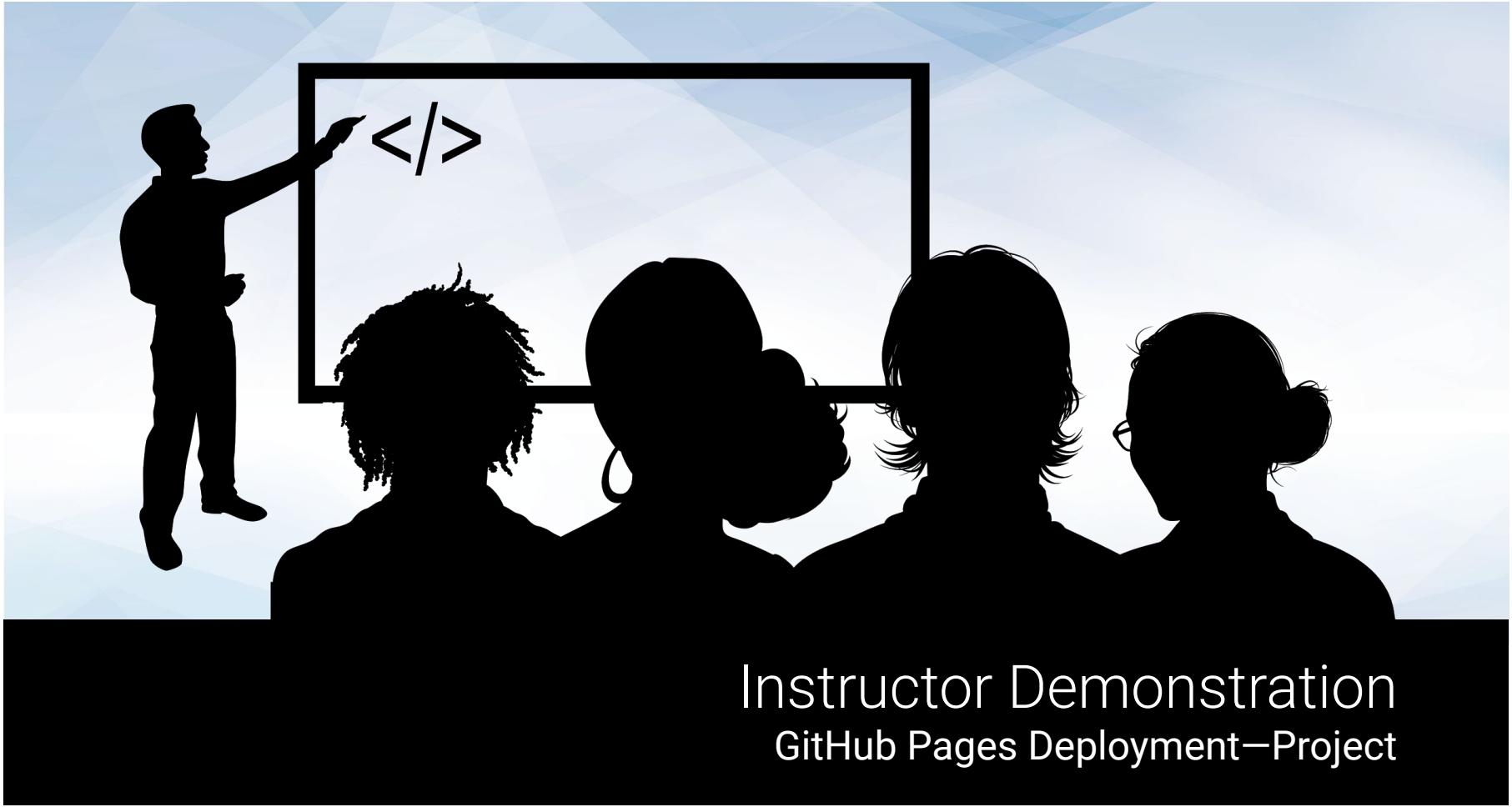


Activity:
Deploying Bio to GitHub Pages

Time to take your newfangled website and deploy it to the cloud—in this case, GitHub Pages.

Additional instructions will be sent via Slack.

Suggested Time:
15 minutes



Instructor Demonstration

GitHub Pages Deployment—Project

Deploying a Static Project Site to GitHub Pages

Follow these steps:

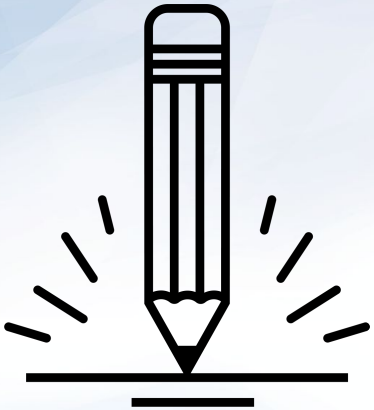
01 Create a new repository in your GitHub account. You can name this repository whatever you like.

02 In the repository, create a new file and name it `index.html`.

03 Add some basic HTML to this file, save it, and then navigate to your repository's Settings tab.

04 Scroll down to the GitHub Pages section. In the Source section, select the master branch as your source

05 Navigate to `<username>.github.io/<repositoryname>`; you will find that your new webpage has gone live!



Activity: Creating a Project Site

Build a newfangled website, and then deploy it to GitHub Pages as a project instead of a personal site.

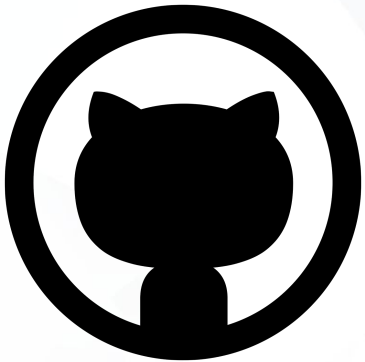
Additional instructions will be sent via Slack.

Suggested Time:
15 minutes

Questions?

Homework 1 Help

Extra Material



And Back to Git…