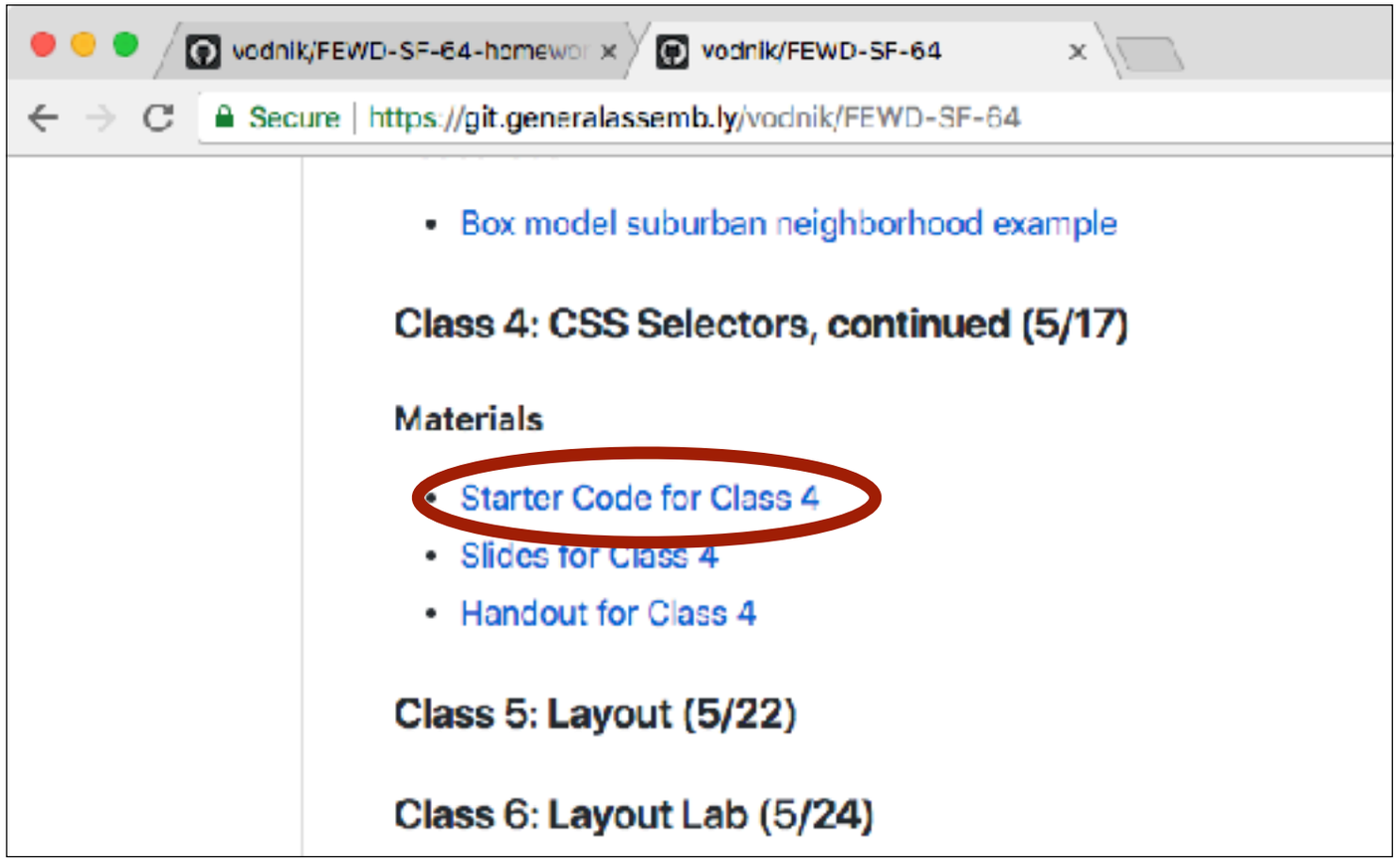
HELLO!

- Go to the class repo on GitHub Enterprise
- Click the **Starter Code for Class 4** link to download the files
- Move the downloaded file to your FEWD folder
- Unzip the files



WEEKLY OVERVIEW

WEEK 2	CSS Selectors / CSS Selectors continued
WEEK 3	Layout / Layout Lab
WEEK 4	Grid Systems & Wireframing / Responsive Design

LEARNING OBJECTIVES

- Add classes and IDs to HTML elements and apply CSS styles to elements based on class and ID.
- Explain when you would use a class and when you would use an ID.
- Apply CSS to elements based on their relationships.
- Describe inheritance in CSS.
- Create and update a repo on GitHub Enterprise

AGENDA

- Review
- Class & ID selectors
- Inheritance
- GitHub
- Lab Time

EXIT TICKET QUESTIONS

1. Why is there so much Latin text in our class examples? :)
2. On Thursdays- can you post the homework assignment in our Homework slack channel?
3. I am confused about how to use the selectors in context.
4. Are there any other ways to control how different html elements appear in relation to each other besides controlling width and height? Any tricks to figuring out how to align various elements?
5. Still a little confused on the difference between padding and margins.
6. Can padding have the same issue as margins when multiple are put together (margin collapse)?
7. A little bit confused about the application on the box-sizing

EXIT TICKET QUESTIONS

8. Review css classes and ID's since we only spent 5 mins of it this class.
9. Confused how sometimes you can not include certain things like header on your nav anchor tag, but it can still work fine without it.

REVIEW:
MORE CSS SELECTORS

SELECTORS — MOST COMMON

- We've been using type selectors… but there are other kinds of selectors too:

SELECTOR:	MEANING:	EXAMPLE:
TYPE	Selects an element	`a {}`
DESCENDANT	Selects an element that is a **descendent** of another element	`p a {}`
UNIVERSAL	Selects **all** elements in a document	`* {}`
MULTIPLE	Selects multiple elements	`h1, h2 {}`

DESCENDANT SELECTORS

SELECTORS

- The **last** element in the selector string is *always* the one that we are styling.
- Try reading right-to-left!


```
<p>Want to get in touch? Send us an <a href="#">email!</a></p>
```

ANCHORS THAT ARE DESCENDANTS OF A PARAGRAPH

SELECTORS

```
<header>
    <h1><a href="">Nested Selectors</a></h1>
    <nav>
        <a href="">Home</a>
        <a href="">About</a>
        <a href="">Contact</a>
        <a href="">Blog</a>
    </nav>
</header>
```

ANCHORS THAT ARE DESCENDANTS OF A NAV.

NOTE: THEY'RE ALSO DESCENDANTS OF A HEADER TOO!

SELECTORS

```
<header>
    <h1><a href="">Nested Selectors</a></h1>
    <nav>
        <a href="">Home</a>
        <a href="">About</a>
        <a href="">Contact</a>
        <a href="">Blog</a>
    </nav>
</header>
```

ANCHORS THAT ARE DESCENDANTS OF A H1

NOTE: THEY'RE ALSO DESCENDANTS OF A HEADER TOO!

TARGETING SPECIFIC ELEMENTS: CLASSES AND IDS

TARGETING SPECIFIC ELEMENTS

- Classes & IDs allow us to add ‘labels’ to elements so we can target them in our CSS.

WHY DO WE USE BAR CODES OR SERIAL NUMBERS?

CLASSES

- Classes are used to group elements together
- Like bar codes or UPCs (universal product codes)

```
<div class="alert">Content</div>
```

```
.alert {
    color: red;
    font-size: 20px;
}
```

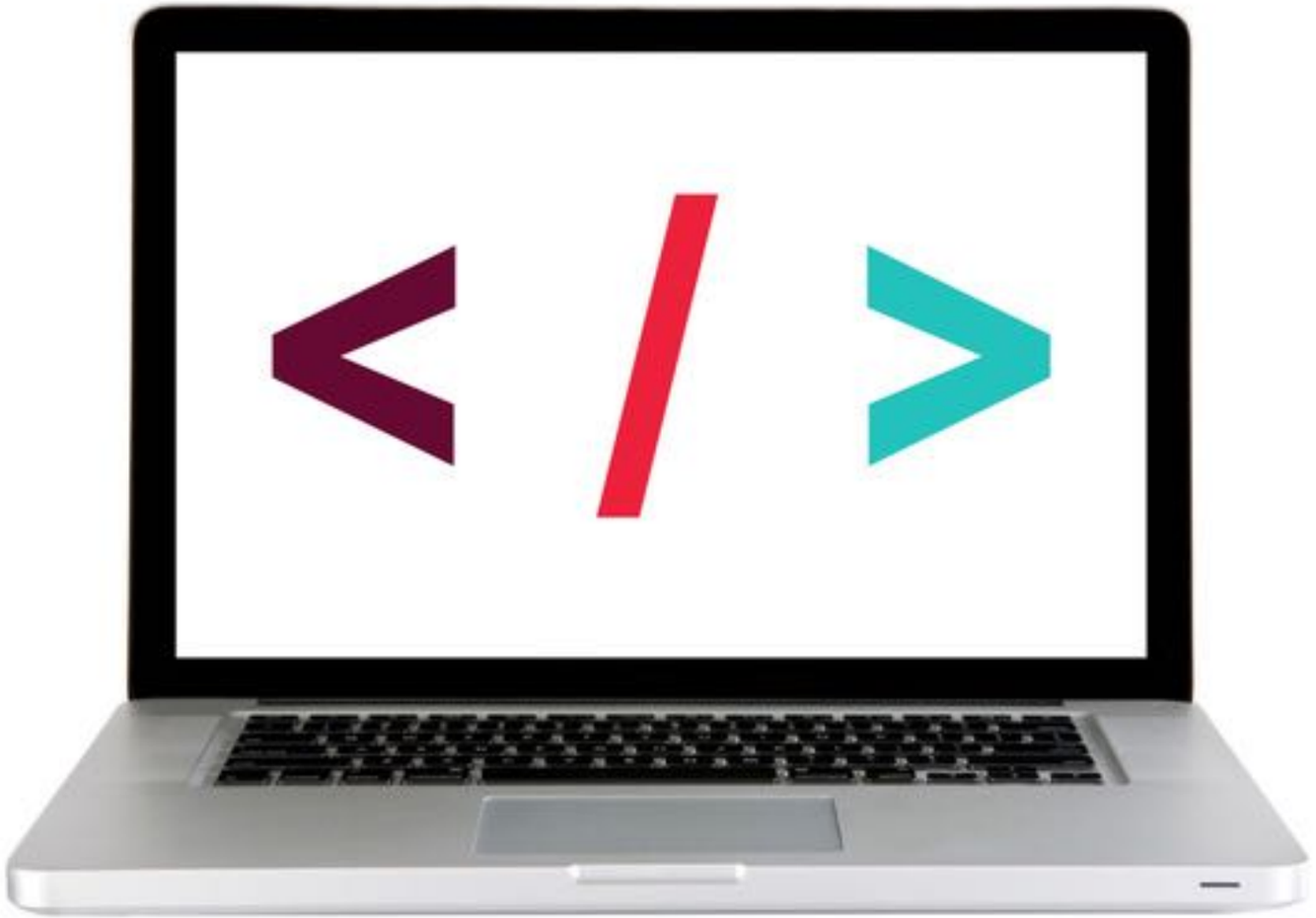
IDS

- Ids are used to target *one specific element*
- Each element can only have one id
- **Important**: two elements on the same page cannot have the same id

```html
<nav id="main-nav">Content</nav>
```

```css
#main-nav {
   text-align: center;
}
```

DEMO: CLASSES AND IDS EXAMPLE



ACTIVITY: CLASSES AND IDS ICE CREAM

EXERCISE

LOCATION

- `starter_code > classes_and_ids`

KEY OBJECTIVE

- Use classes and IDs to target elements

TIMING

1. Look at the image provided. Which ice cream items would you use a class to style? How about an ID?
2. Follow steps 1 - 3 in your CSS file (towards the end)

WHY IS CSS “CASCADING”?

INHERITANCE

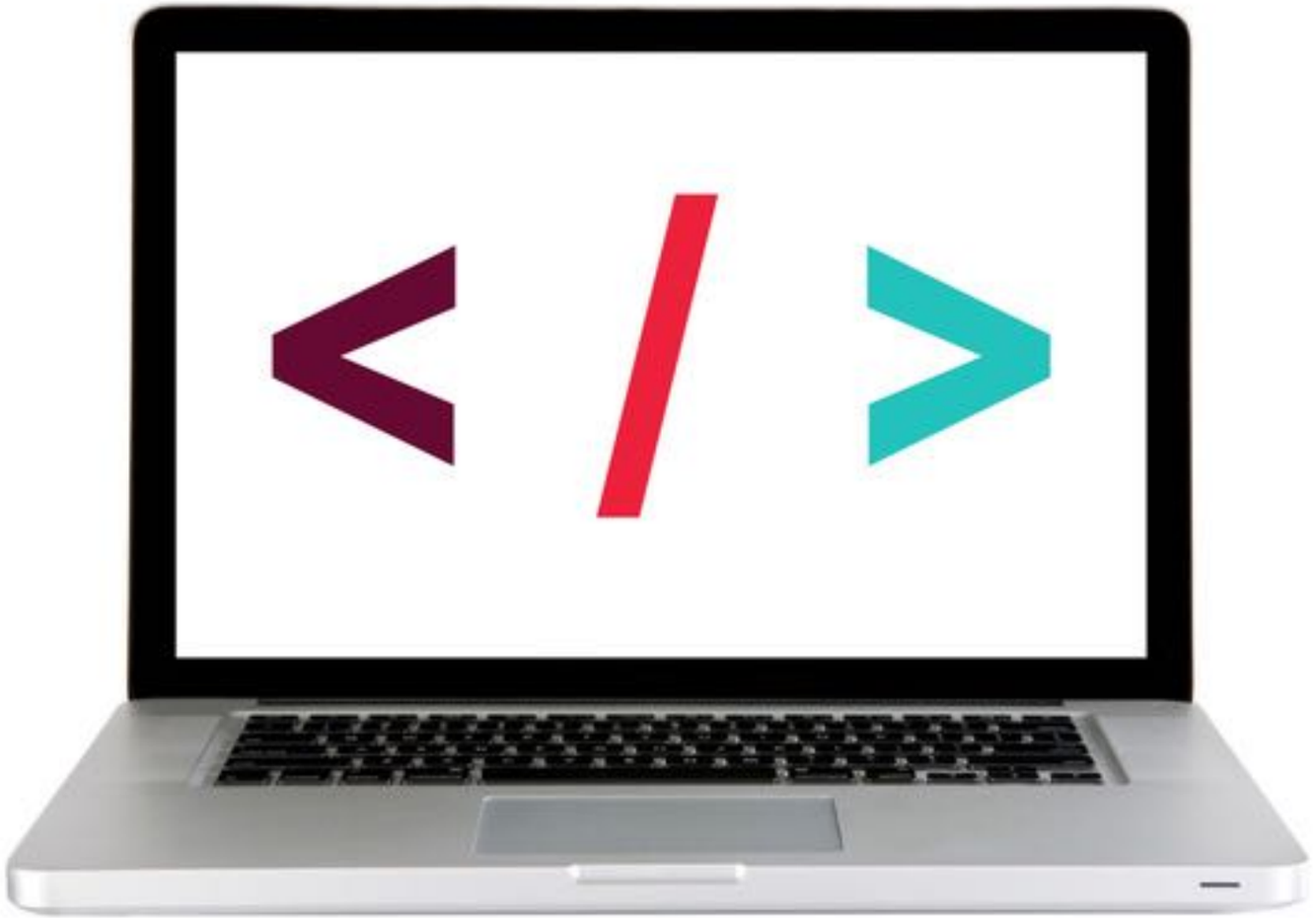
INHERITANCE — SETTING BASE STYLES

- Certain properties are passed on from a parent element down to its children
- If you specify the *font-family* or *color* properties on the `body` element, they will be inherited, or applied, to most child elements **unless there is a more specific rule that applies**.

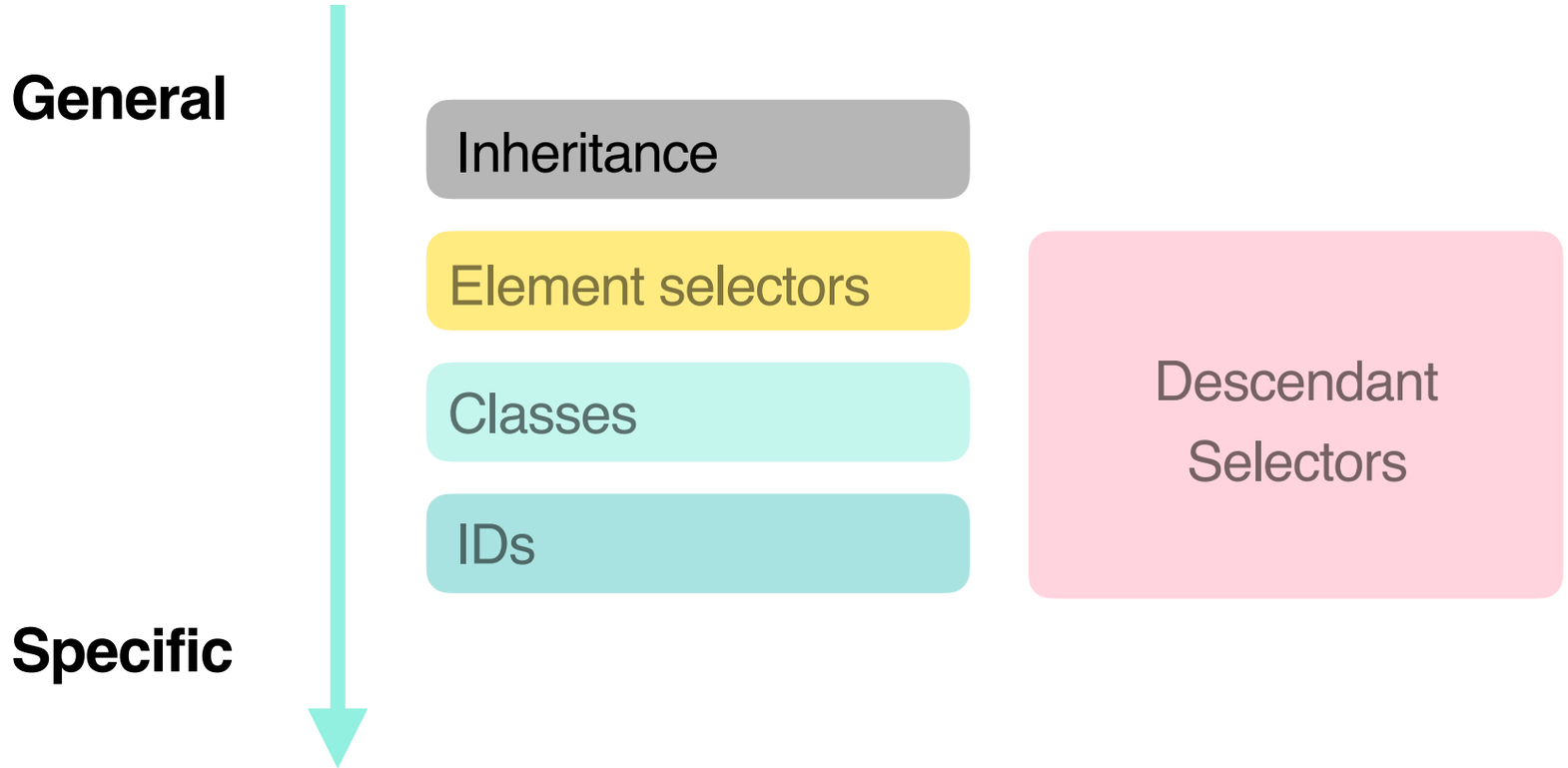
Inherited properties you'll use in this course:

- `color`
- `font-family`
- `font-size`
- `font-style`
- `font-weight`
- `letter-spacing`
- `line-height`
- `list-style`
- `text-align`
- `text-indent`
- `text-transform`
- `word-spacing`

CODEALONG: CASCADING EXAMPLE PART 2



MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC



ACTIVITY: DISCUSS INHERITANCE

KEY OBJECTIVE

- Explain inheritance in CSS. How can you use inheritance in your stylesheets to write less code in the long run?

TYPE OF ACTIVITY

- Turn and Talk

TASKS

1 min	1. Discuss the question with your groups
2 min	2. Pick one person to jot down your thoughts and share them via Slack.

ACTIVITY: SPECIFICITY CACTUS SITE

KEY OBJECTIVE

- Practice using inheritance to set up default styles for a webpage

LOCATION

- Starter Code > Specificity

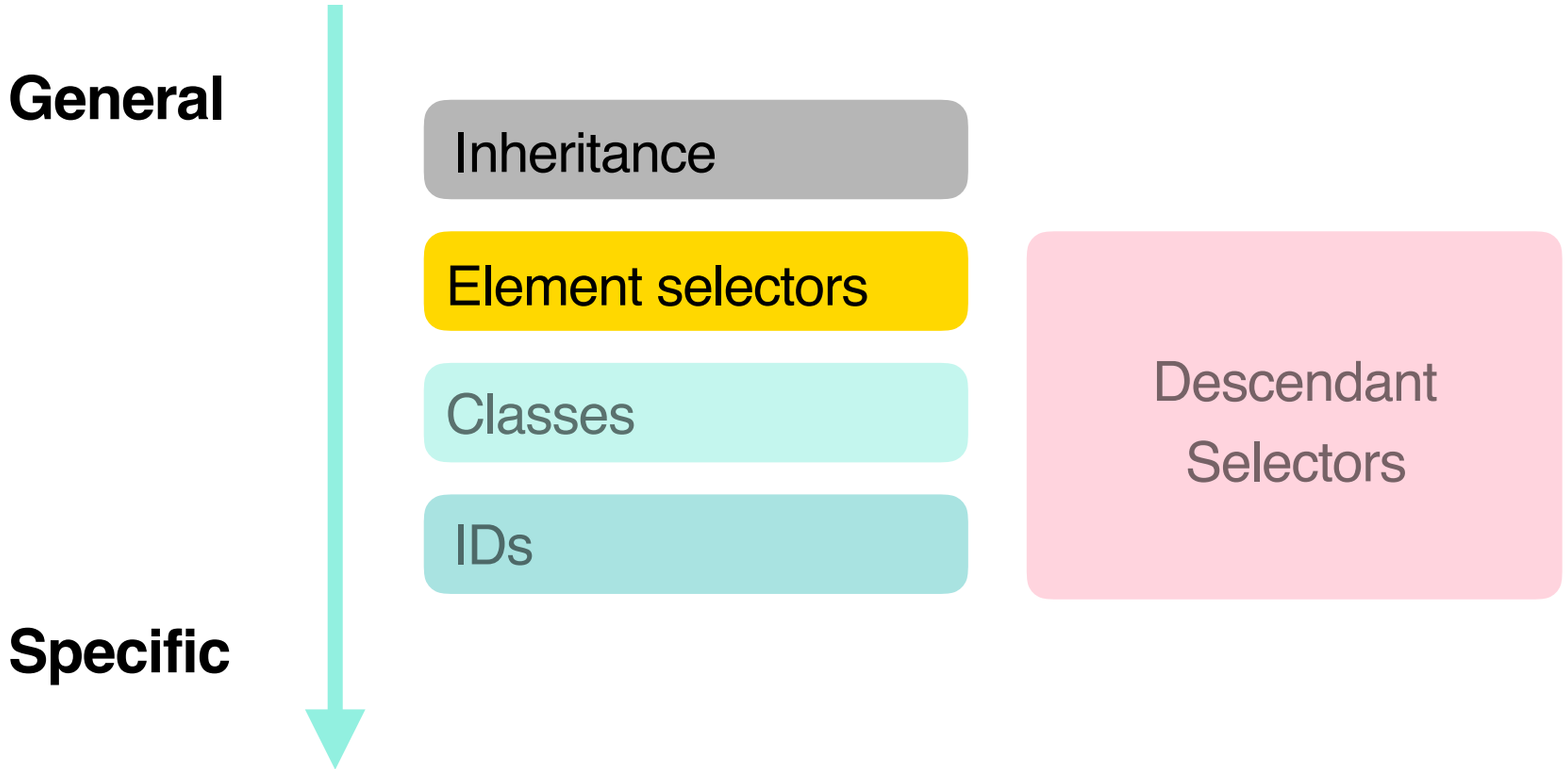
TASKS

1 min

1. Follow instructions under STEP 1

ELEMENT SELECTORS

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC



ACTIVITY: SPECIFICITY CACTUS SITE

KEY OBJECTIVE

- Practice using element selectors to style groups of elements

TYPE OF ACTIVITY

- Starter Code > Specificity

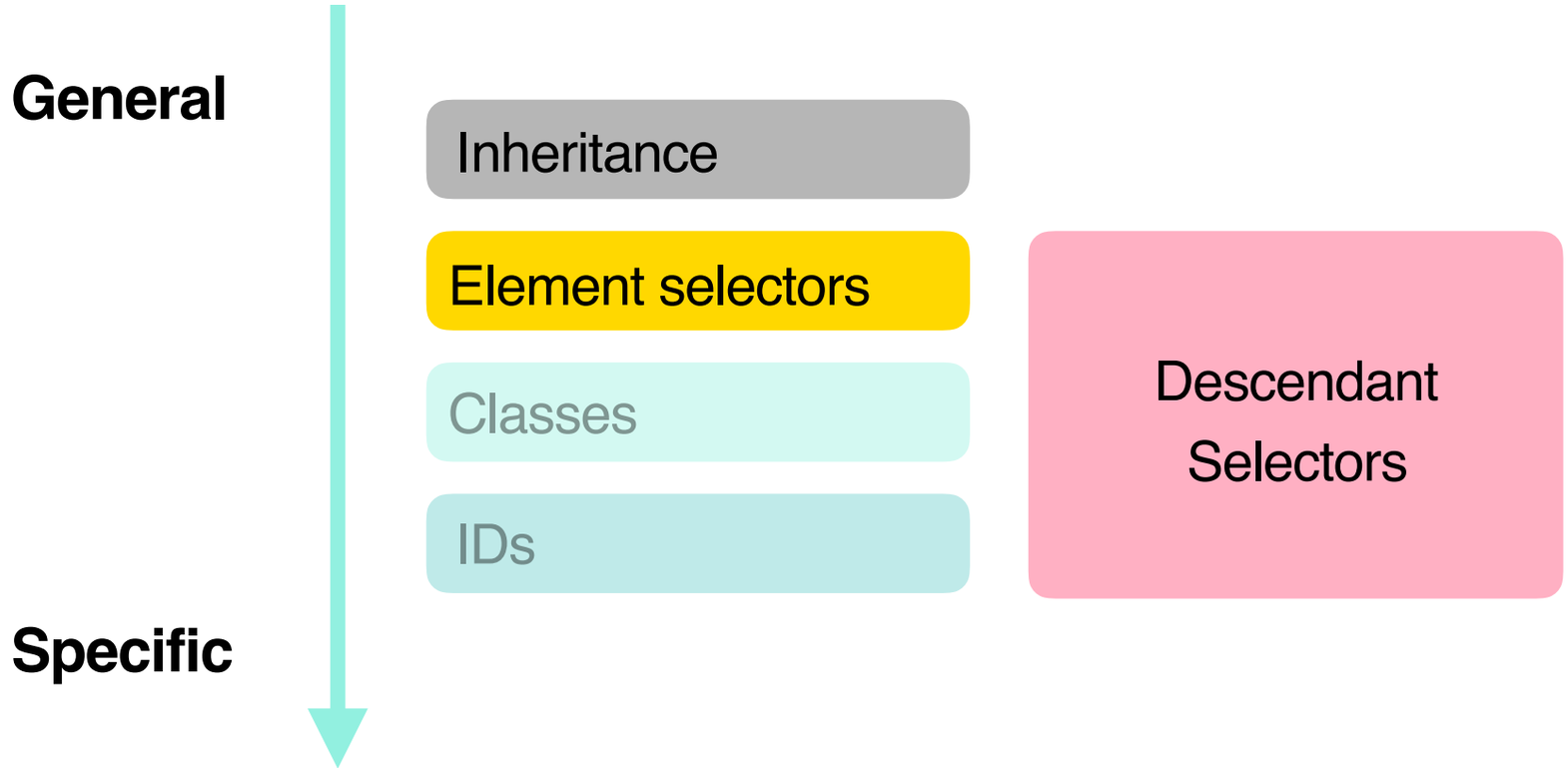
TASKS

1 min

1. Follow the instructions under STEP 2.

DESCENDANT SELECTORS

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC



ACTIVITY: SPECIFICITY CACTUS SITE

EXERCISE

KEY OBJECTIVE

- Practice using nested selectors to style elements based on their relationships.

TYPE OF ACTIVITY

- Starter Code > Specificity

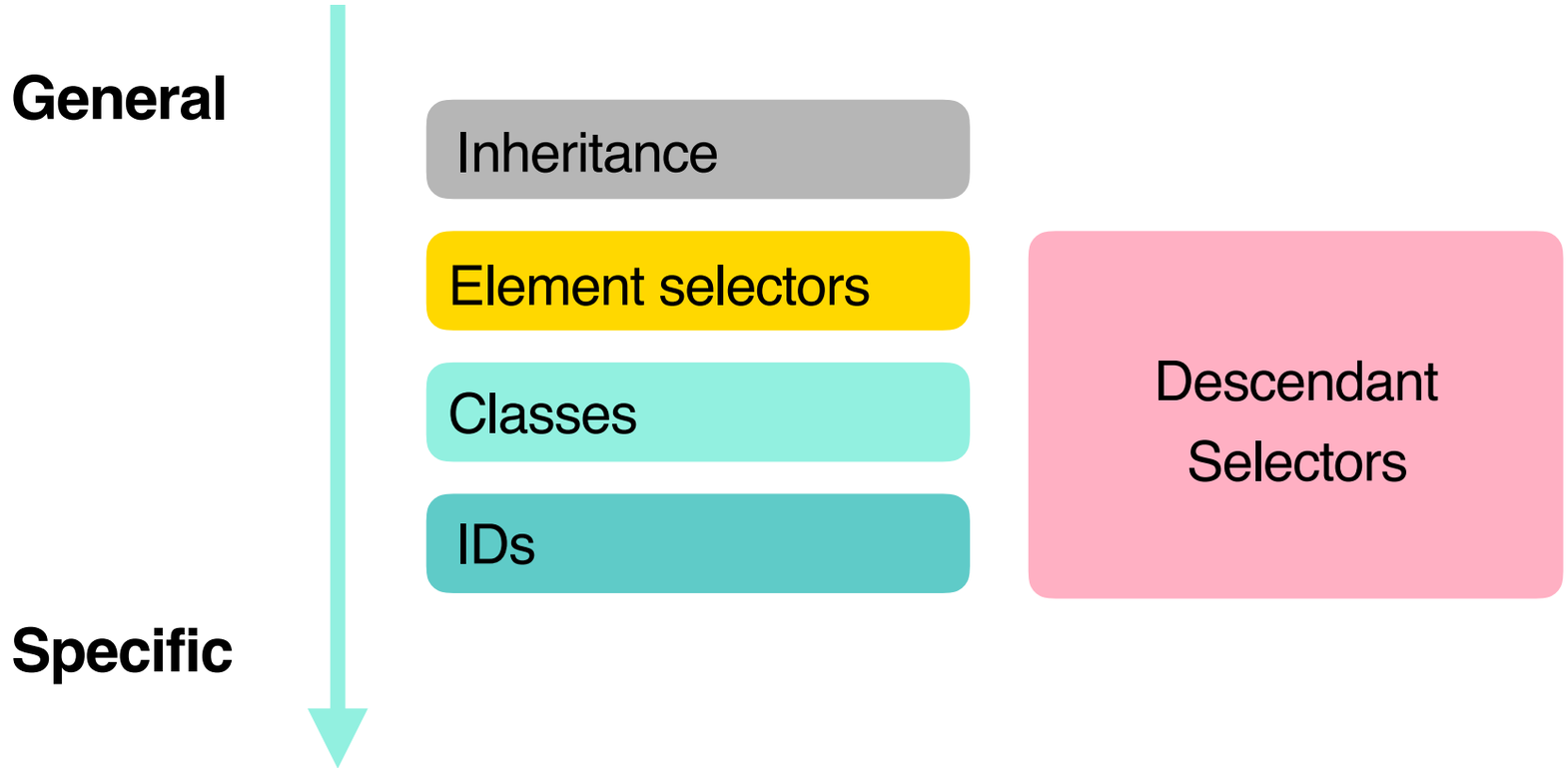
TASKS

1 min

1. Follow the instructions under STEP 3.

CLASSES AND IDS

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC



ACTIVITY: SPECIFICITY CACTUS SITE

KEY OBJECTIVE

- Practice using classes and IDs to style unique elements or groups of elements

TYPE OF ACTIVITY

- Starter Code > Specificity

TASKS

1 min

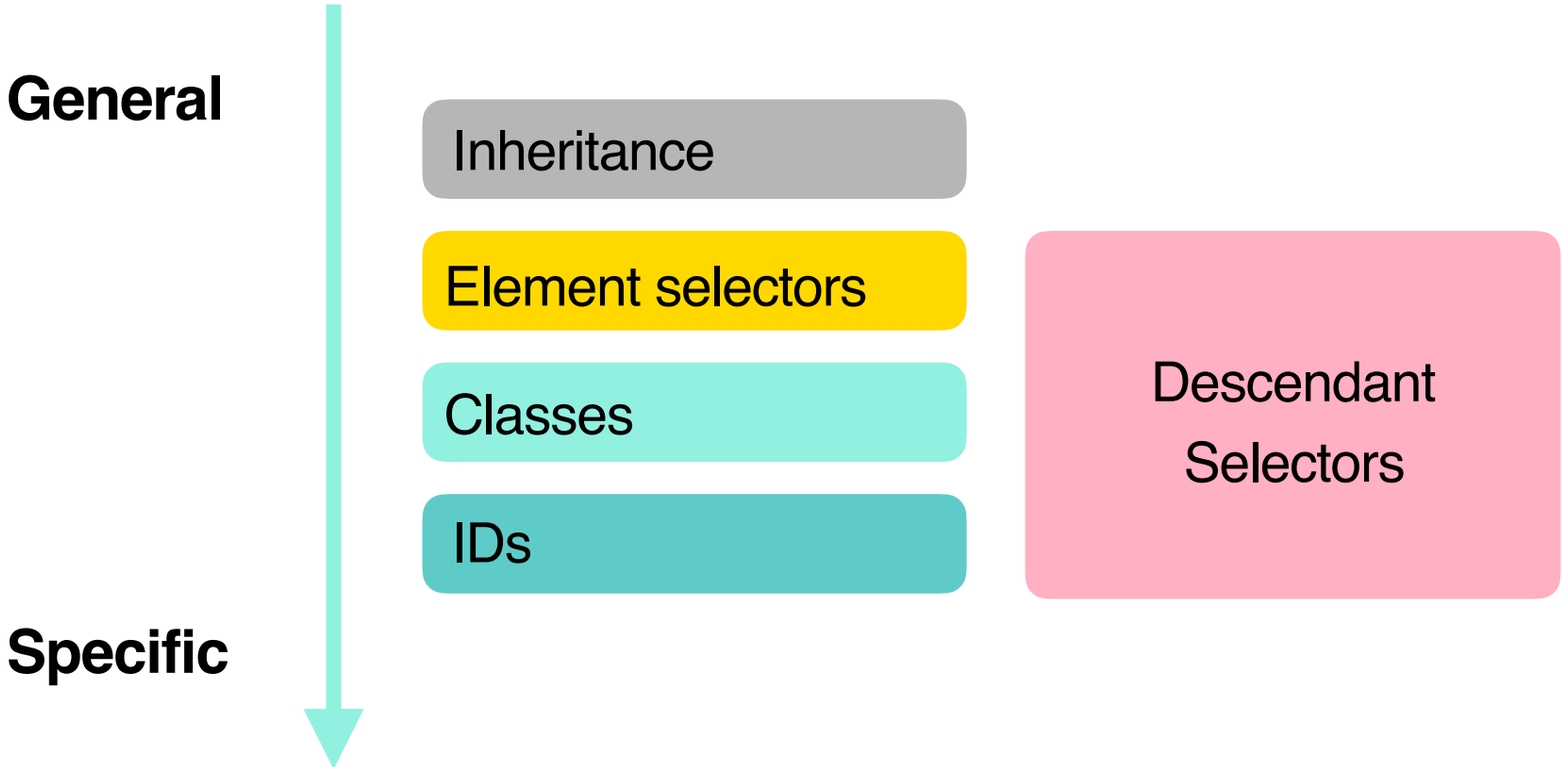
1. Follow the instructions under STEP 4.

SPECIFICITY GAME!!!

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC

SPECIFICITY:

The *more specific rule* will take precedence over the more general rule



CSS CASCADING — THE SPECIFICITY GAME

- If multiple style rules are targeted at the same element, which style will be applied?
- We can calculate the specificity of the selectors to find out!
- The styles for the more specific selector are the styles that will be applied.

```
p a {
    font-size: 50px;
}

a {
    font-size: 30px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

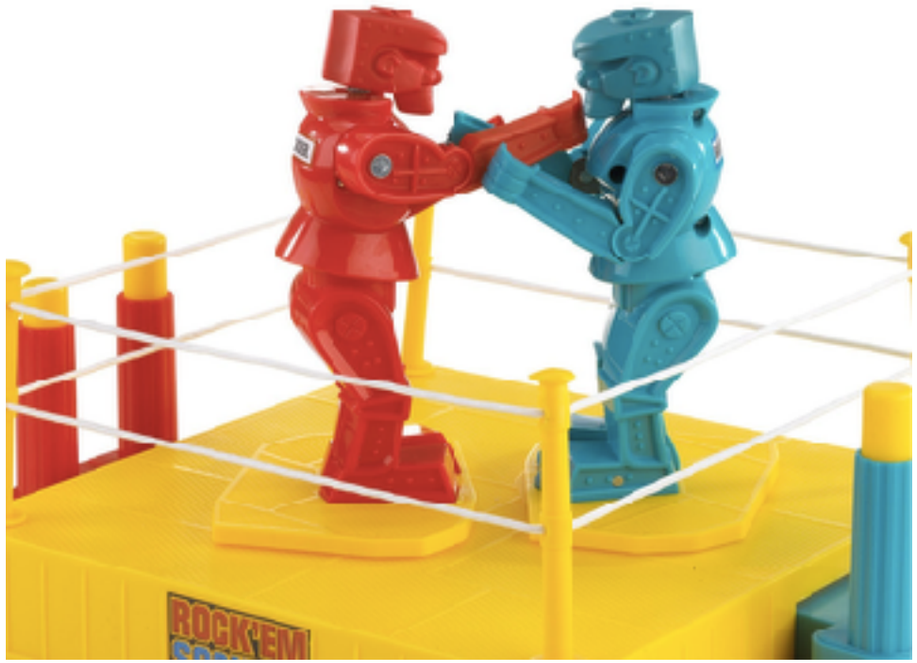
TO CALCULATE SPECIFICITY,
TAKE A LOOK AT THE ENTIRE "SELECTOR CHAIN"

```
p a {
    font-size: 50px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

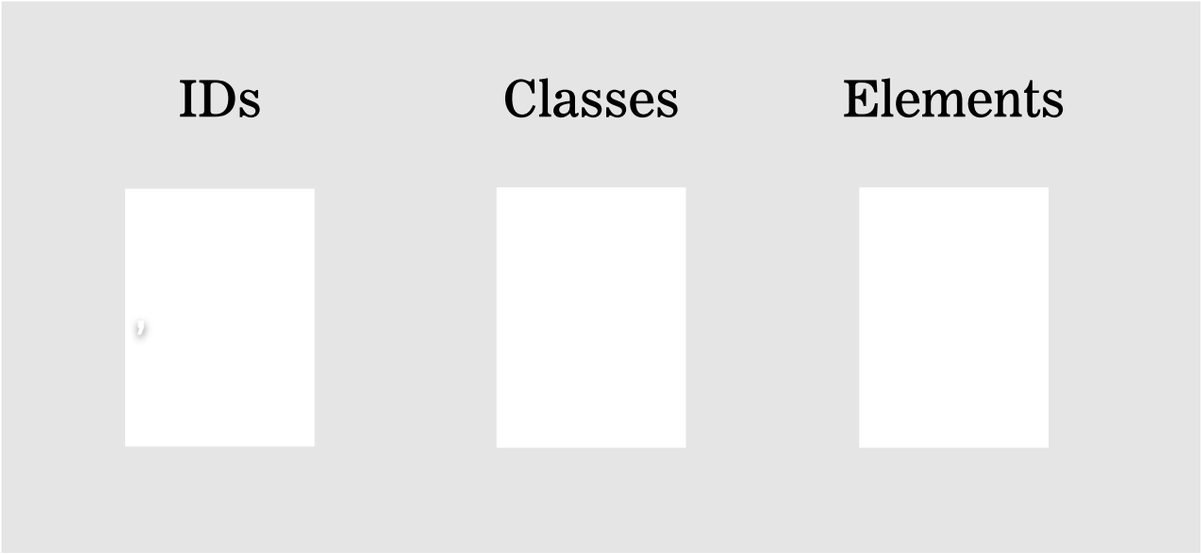
If you have conflicting styles under the “p a” selector and the “a” selector… who will win?

p a a



CSS CASCADING — THE SPECIFICITY GAME

Use a chart like this to calculate the specificity to find out!



CSS CASCADING — THE SPECIFICITY GAME

2 ELEMENTS

```
p a {
    font-size: 50px;
}
```

IDs	Classes	Elements
0	0	2

SCORE: 2

CSS CASCADING — THE SPECIFICITY GAME

1 CLASS

```
.about {
    font-size: 50px;
}
```

IDs	Classes	Elements
0	1	0

SCORE: 10

CSS CASCADING — THE SPECIFICITY GAME

ID ELEMENT

```
#about a {
    font-size: 50px;
}
```

IDs	Classes	Elements
1	0	1

SCORE: 101

CSS CASCADING — THE SPECIFICITY GAME

Write this down!

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the font-size for the anchor be 50px or 30px?*

HTML:

```html
<p>Visit my <a href="#">Website</a></p>
```

CSS:

```css
p a {
    font-size: 50px;
}

a {
    font-size: 30px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

p a

IDs	Classes	Elements
0	0	2

WINNER!

a

IDs	Classes	Elements
0	0	1

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the font-size for the anchor be 50px or 30px?*

HTML:

```html
<p>Visit my <a href="#">Website</a></p>
```

CSS:

```css
p a { /* Score: 2 */
    font-size: 50px;
}

a { /* Score: 1 */
    font-size: 30px;
}
```

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the anchor with the class* home *be pink or blue?*

HTML:

```
<nav id="main-nav">
    <a href="#" class="home">Home</a>
    <a href="#">About</a>
    <a href="#">Resume</a>
</nav>
```

CSS:

```
#main-nav a {
    color: pink;
}

.home {
    color: blue;
}
```

CSS CASCADING

`#main-nav a`

IDs	Classes	Elements
1	0	1

WINNER!

`.home`

IDs	Classes	Elements
0	1	0

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the anchor with the class* home *be pink or blue?*

HTML:

```html
<nav id="main-nav">
    <a href="#" class="home">Home</a>
    <a href="#">About</a>
    <a href="#">Resume</a>
</nav>
```

CSS:

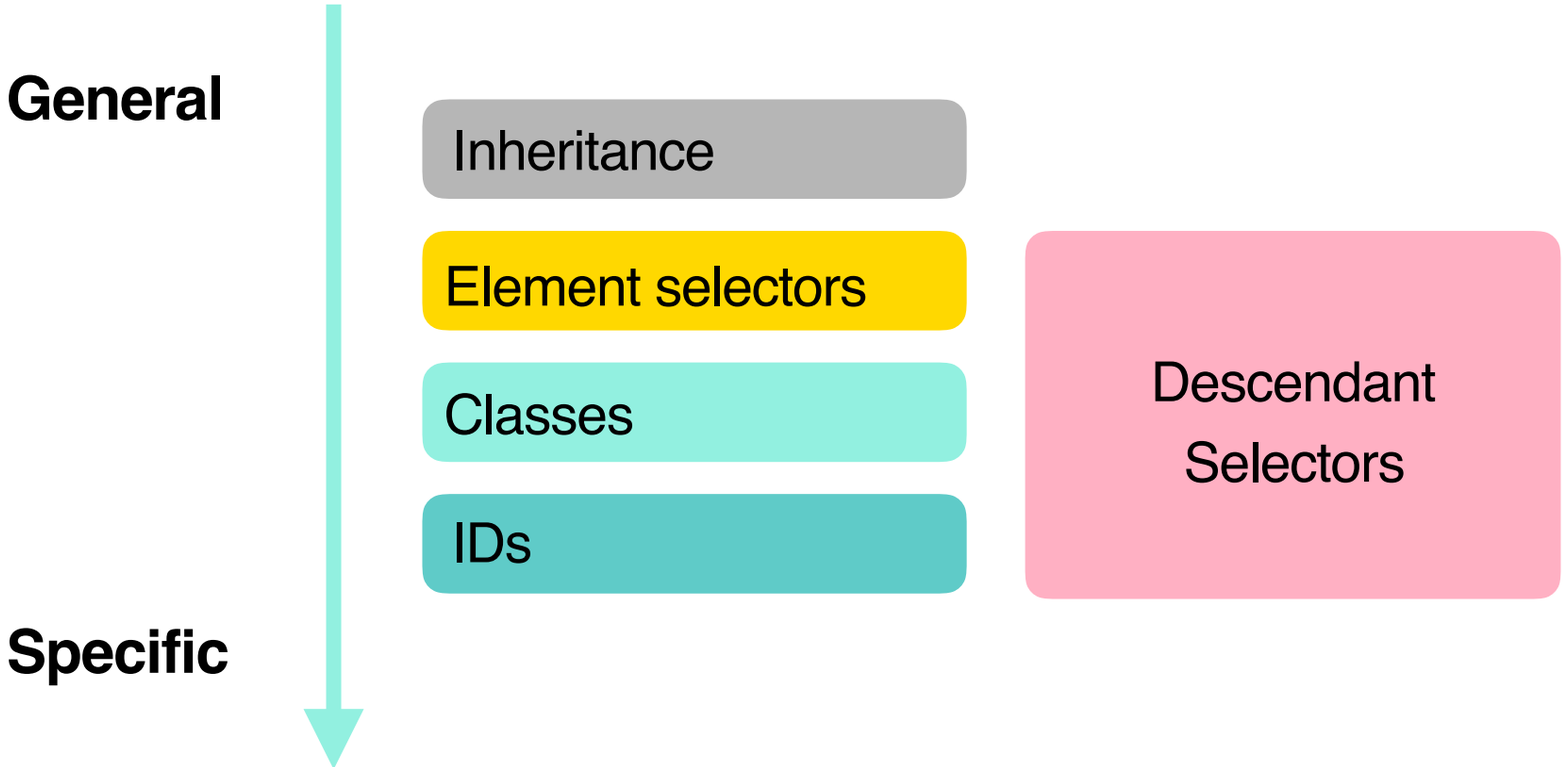
```css
#main-nav a { /* Score: 101 */
    color: pink;
}

.home { /* Score: 10 */
    color: blue;
}
```

MORE ABOUT CASCADING — GENERAL TO MORE SPECIFIC

SPECIFICITY:

The *more specific rule* will take precedence over the more general rule



LAST RULE

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the anchor have an underline or no underline?*

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
li a { /* Score: 2 */
    text-decoration: underline;
}

p a { /* Score: 2 */
    text-decoration: none;
}
```

LAST RULE

`li a`

IDs	Classes	Elements
0	0	2

`p a`

IDs	Classes	Elements
0	0	2

CSS CASCADING — THE SPECIFICITY GAME

SLACK POLL: *Will the anchor have an underline or no underline?*

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
li a { /* Score: 2 */
    text-decoration: underline;
}

p a { /* Score: 2 */
    text-decoration: none;
}
```

TIEBREAKER? LAST RULE WINS!

CSS CASCADING — THE SPECIFICITY GAME

Let's say we switched it…

HTML:

```html
<li>
    <p>Visit this <a href="#">cool</a> site.</p>
</li>
```

CSS:

```css
p a { /* Score: 2 */
    text-decoration: none;
}

li a { /* Score: 2 */
    text-decoration: underline;
}
```

TIEBREAKER? LAST RULE WINS!

ACTIVITY

KEY OBJECTIVE

- When multiple style rules apply to the same element, how can we figure out which one will be applied?
- How can we apply this principle to allow us to write less CSS in the long run?

TYPE OF ACTIVITY

- Turn and Talk

TASKS

5 min

1. Talk with your groups. Have one person write out your responses in Slack.

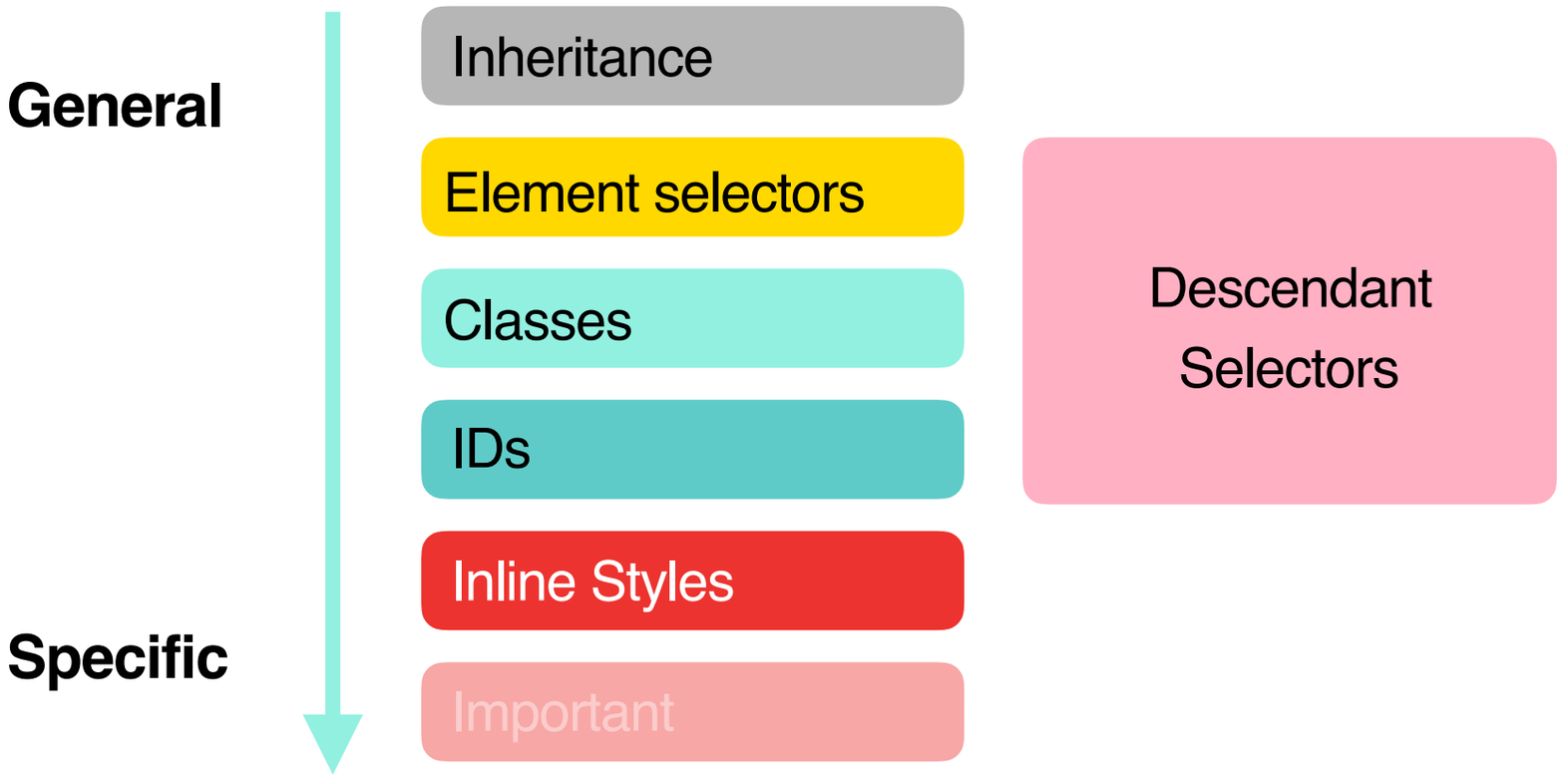
THINGS YOU SHOULD USE IF YOU WANT TO BE

INLINE STYLES

```
<li style="color: red;">Content</li>
```

BAD!!!!

MORE ABOUT CASCADING



CSS IMPORTANCE

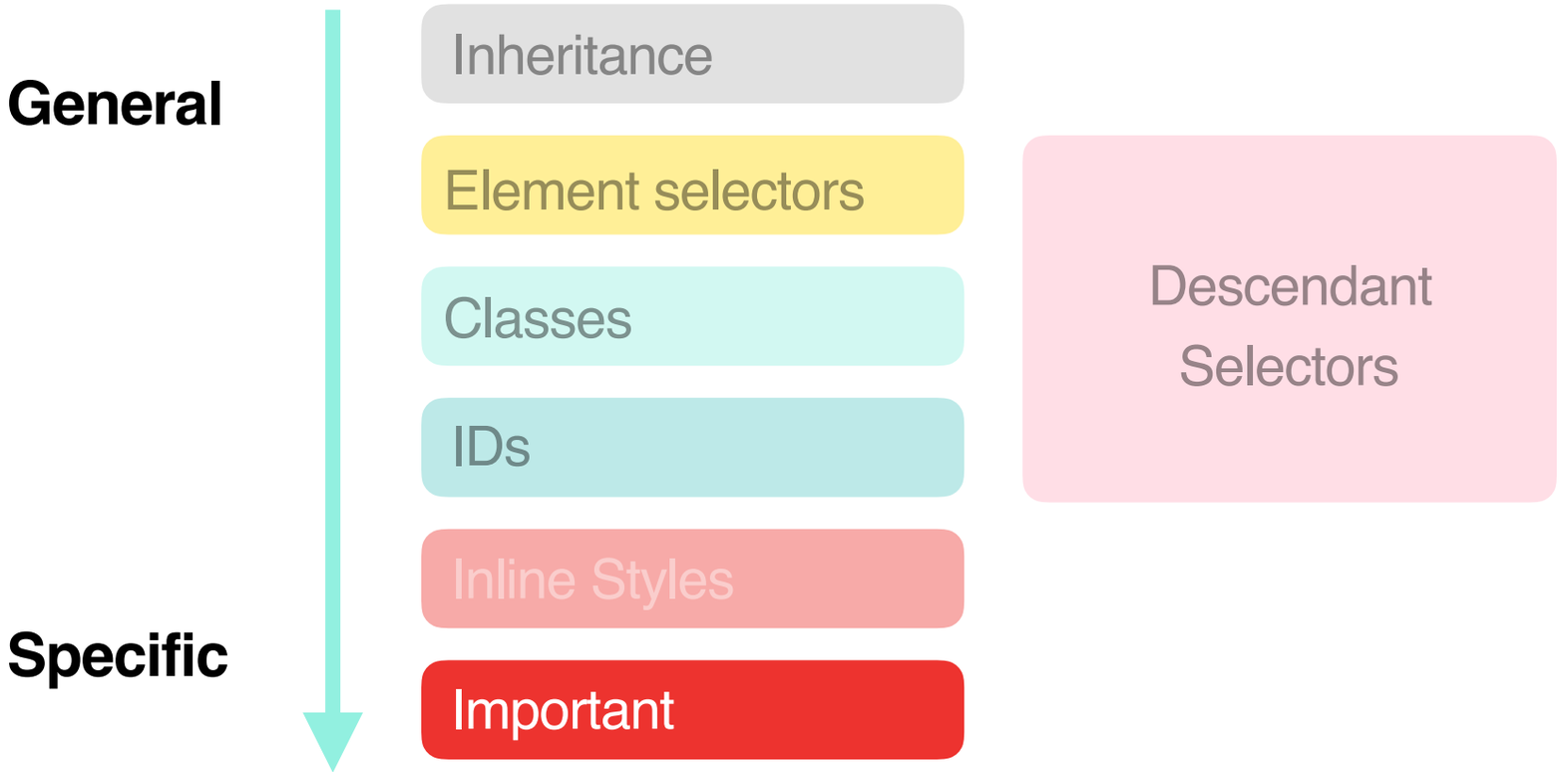
Adding **!important** after any property value indicates that it should be considered *more important than other rules that apply to the same element.*

```
li {
   font-size: 100px !important;
}
```

ONLY WHEN NECESSARY!!!

(Which is almost never)

MORE ABOUT CASCADING



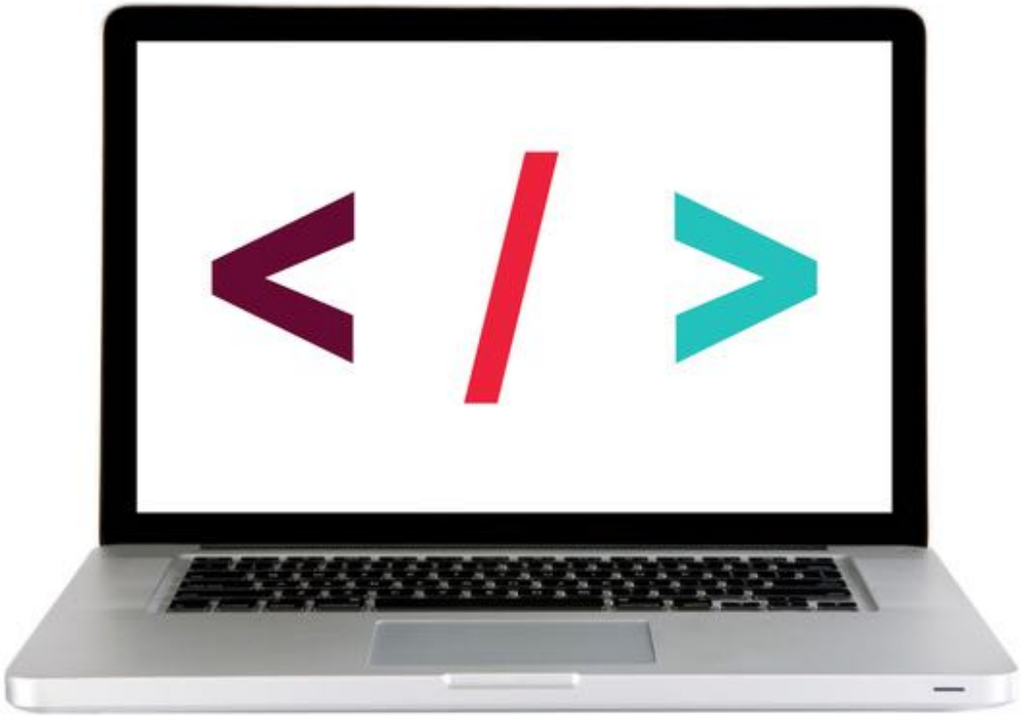
TOOLS: BROWSER DEVELOPER TOOLS

CHROME DEV TOOLS

Right click > Inspect

- WARNING: You can fiddle around with the styles here, but MAKE SURE TO UPDATE (and save) YOUR CSS FILE IN YOUR EDITOR!

DEMO: WORKING WITH CHROME DEV TOOLS



Use Dev Tools to inspect solutions!

TOOLS: VALIDATORS

HTML VALIDATOR

- Online code validators are helpful tools that can help you catch syntax errors as you work.
- An HTML validator can help catch errors such as missing closing tags, missing quotation marks, or misspelled tags.
- All your submitted homework should pass validation.
- A popular option for HTML validation is https://html5.validator.nu/

TO VALIDATE HTML:

1. Go to https://html5.validator.nu/
2. Select "Text Field" from the dropdown
3. Delete the default HTML that is in the text field
4. Copy and paste your entire HTML file into the text field.
5. Click the "Validate" button.

CSS VALIDATOR

- A CSS validator can help us catch errors such as a missing semicolon, missing curly brace, or misspellings.
- All your submitted homework should pass validation.

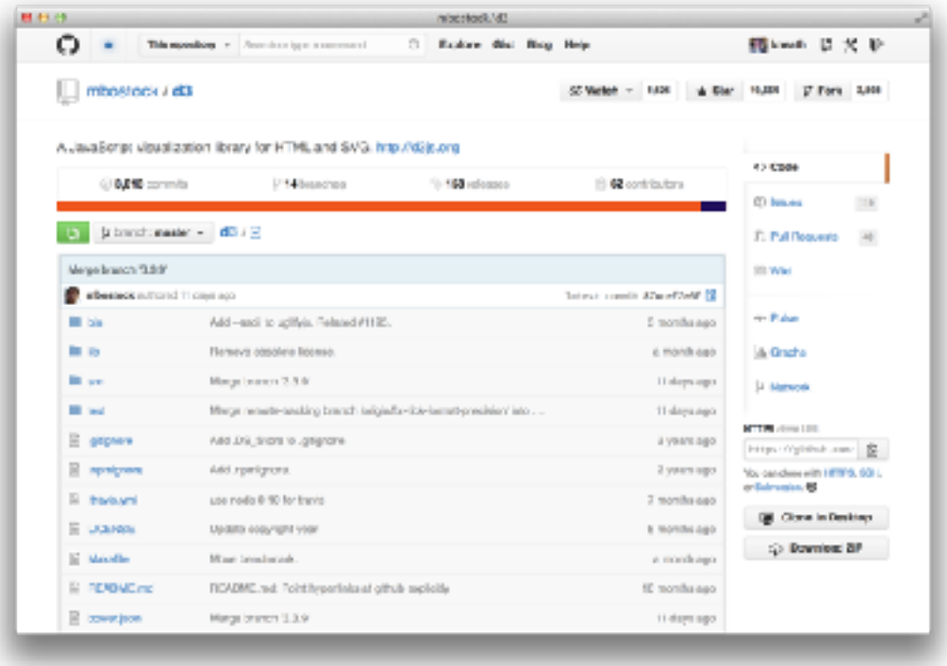
STEPS TO VALIDATE CSS:

1. Go to http://csslint.net/
2. Select the "By direct input" tab
3. Copy and paste your entire CSS file into the text field.
4. Click the "Check" button.

FEWD

INTRODUCTION TO GIT & GITHUB

Github Enterprise

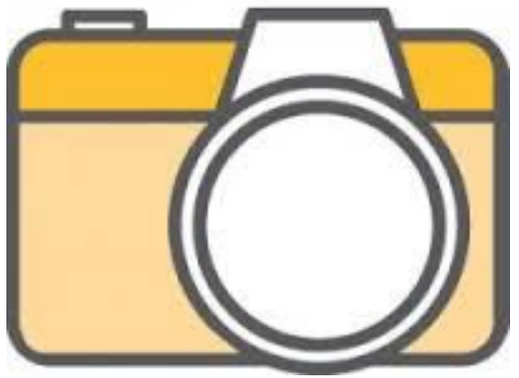


The class repo will have slide decks and handouts, as well as details on the class schedule and homework.
You'll also use GitHub to submit homework starting in Week 2.

GITHUB

GIT

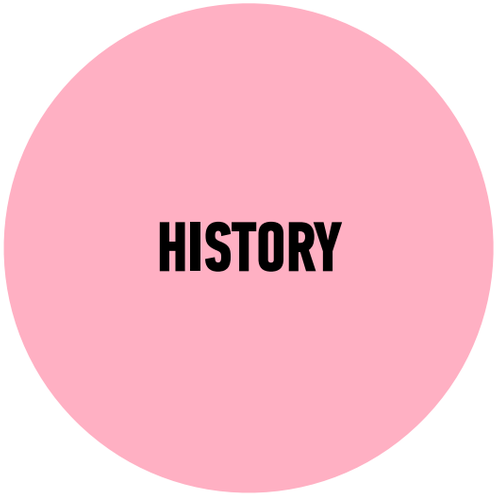
- A **version control** program that saves the state of your project's files and folders
- Basically, it takes a "snapshot" of what all your files look like at a moment and stores a reference to that "snapshot"



GITHUB

- A **web app/platform** that makes it easy to manage git repositories.
- Similar to Dropbox or Google Drive, but for code.
- Stores a history of files and the changes that happen within each changed document.
- Hosts files on the cloud so you can share the finished product with other people.
- **Git** - the technology that Github is based on top of - was designed to allow for multiple engineers to work on the same project.

Why use GitHub?



- Since GitHub stores a history of the code, it allows developers to go back in time if something breaks.

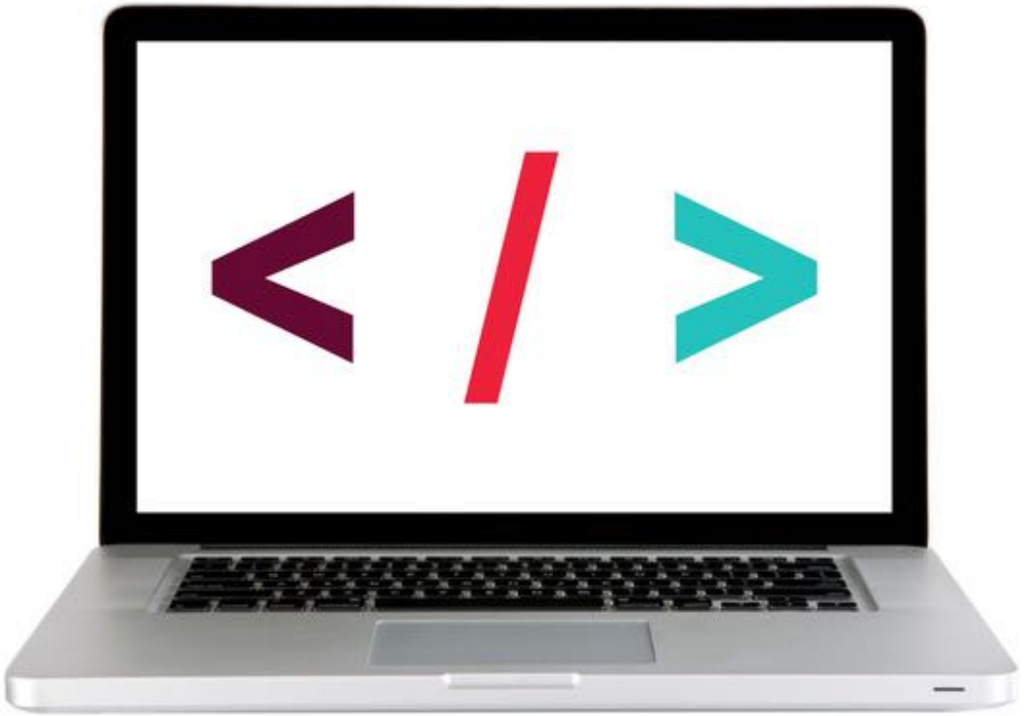
- Allows multiple developers to work on the same project. Much like Google Drive lets multiple people collaborate on the same document, GitHub allows this for code.
- You can see who worked on what.

- GitHub allows for feedback to be given on the code which, hopefully, increases code quality.

What is a repository (repo)?

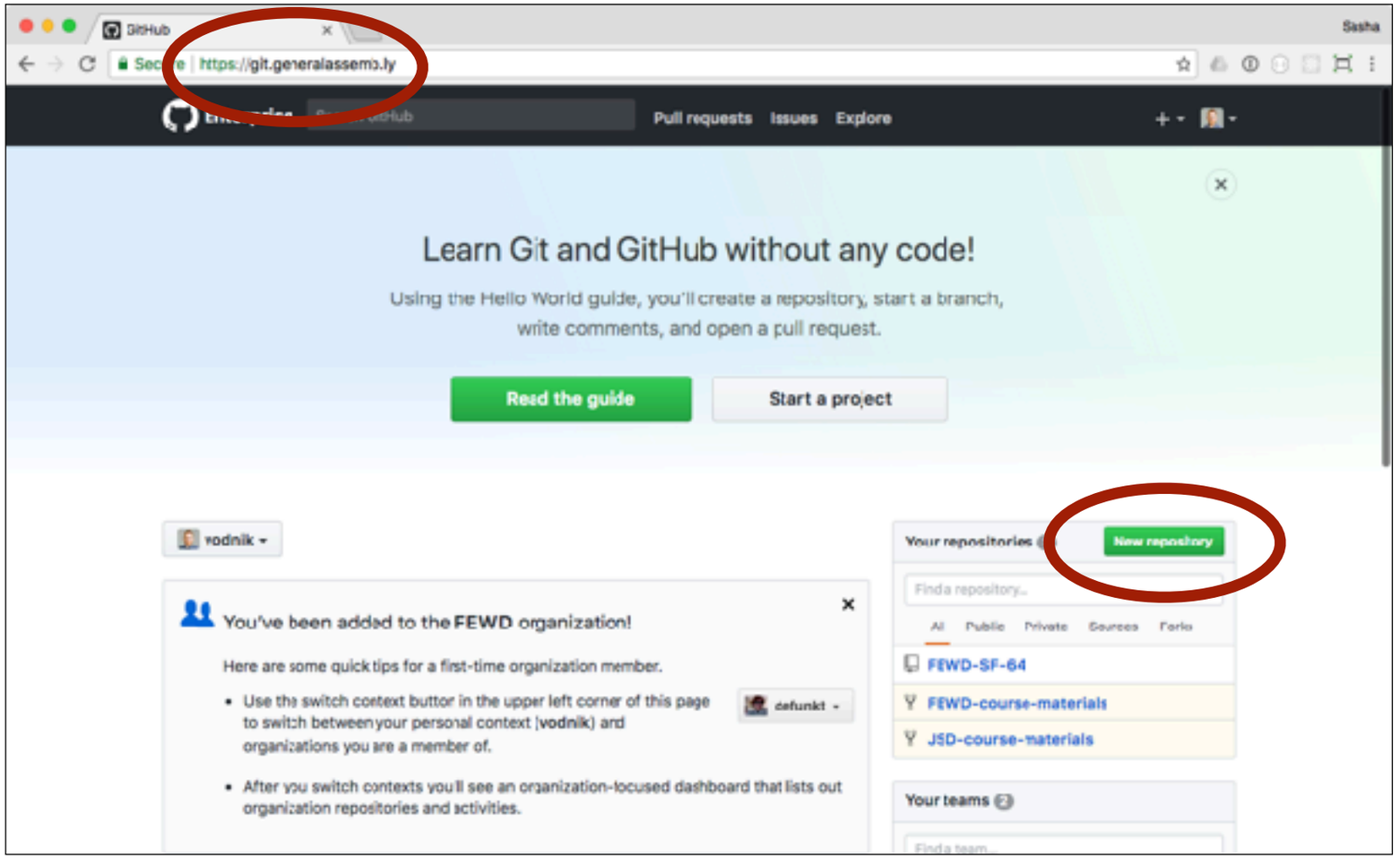
- Basic element of GitHub
- Contains all of a project's files (all the code)
- One or more users can contribute to a single repository
- Repositories are either public or private
- You will submit all your upcoming homework assignments by creating GitHub repos

DEMO: CREATING A NEW GITHUB ENTERPRISE REPO AND ADDING FILES TO IT



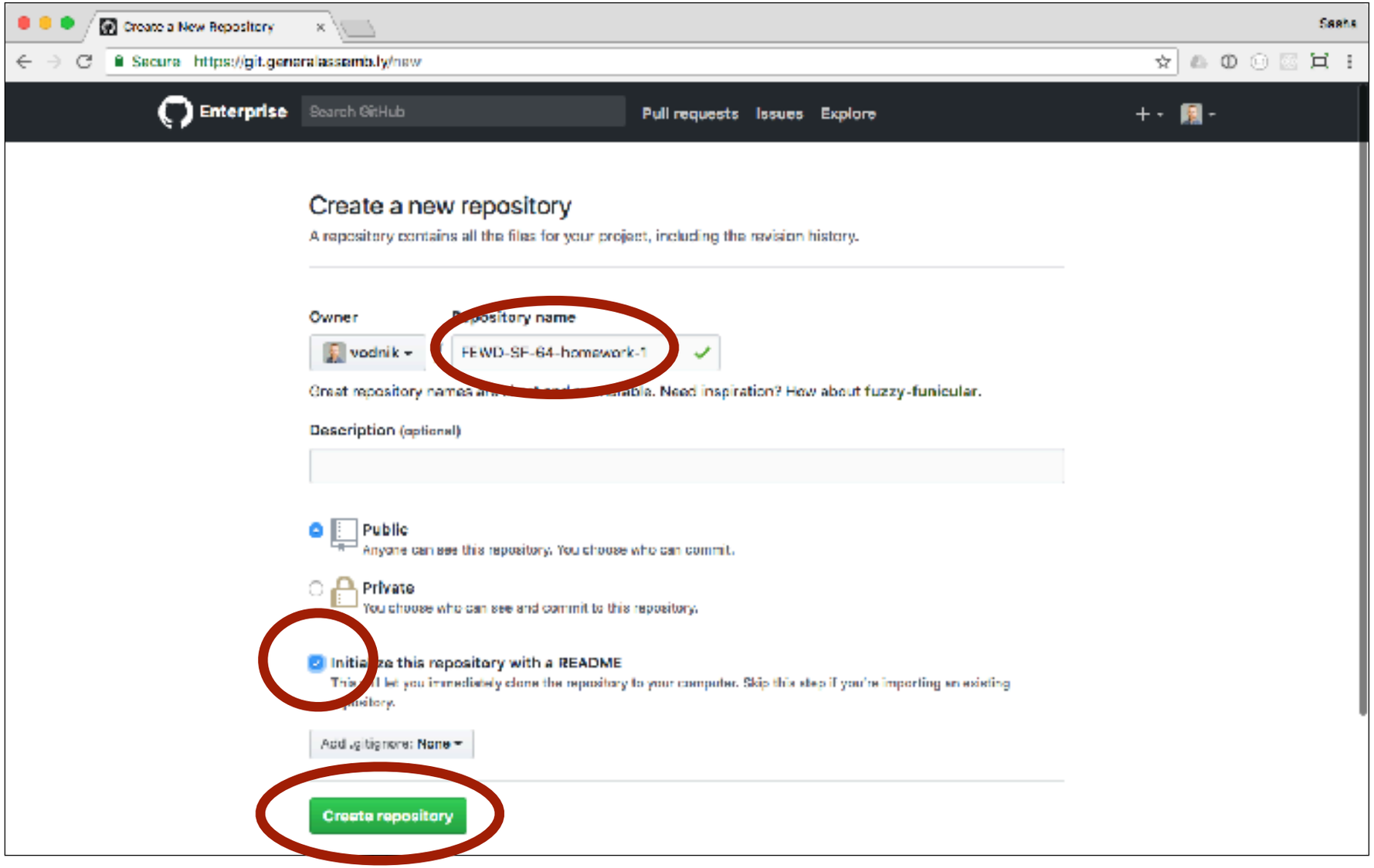
DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

- In Chrome, go to git.generalassemb.ly and log in
- Click the **New repository** button to start creating a new repo



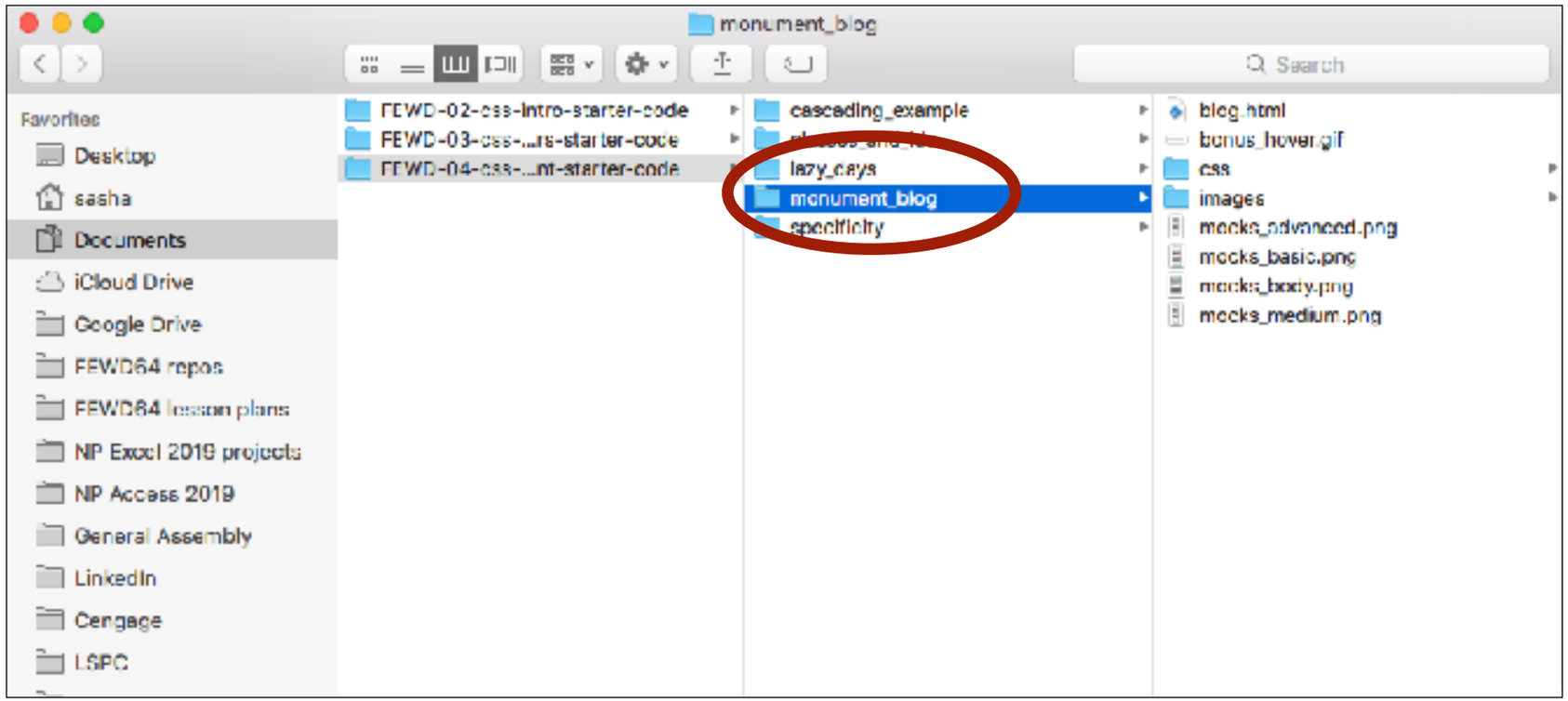
DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

- In the *Repository name* box, type a name for the repo (if you can't think of one, use the one that GitHub suggests on the line below the Repository name box)
- Check the **Initialize this repository with a README** box
- Click the **Create repository** button



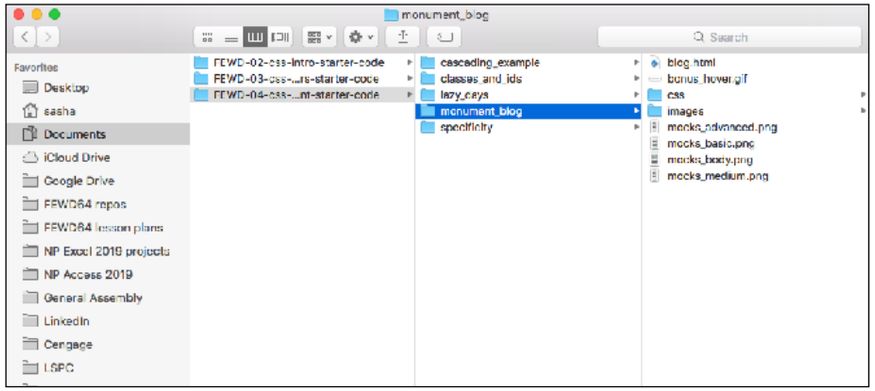
DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

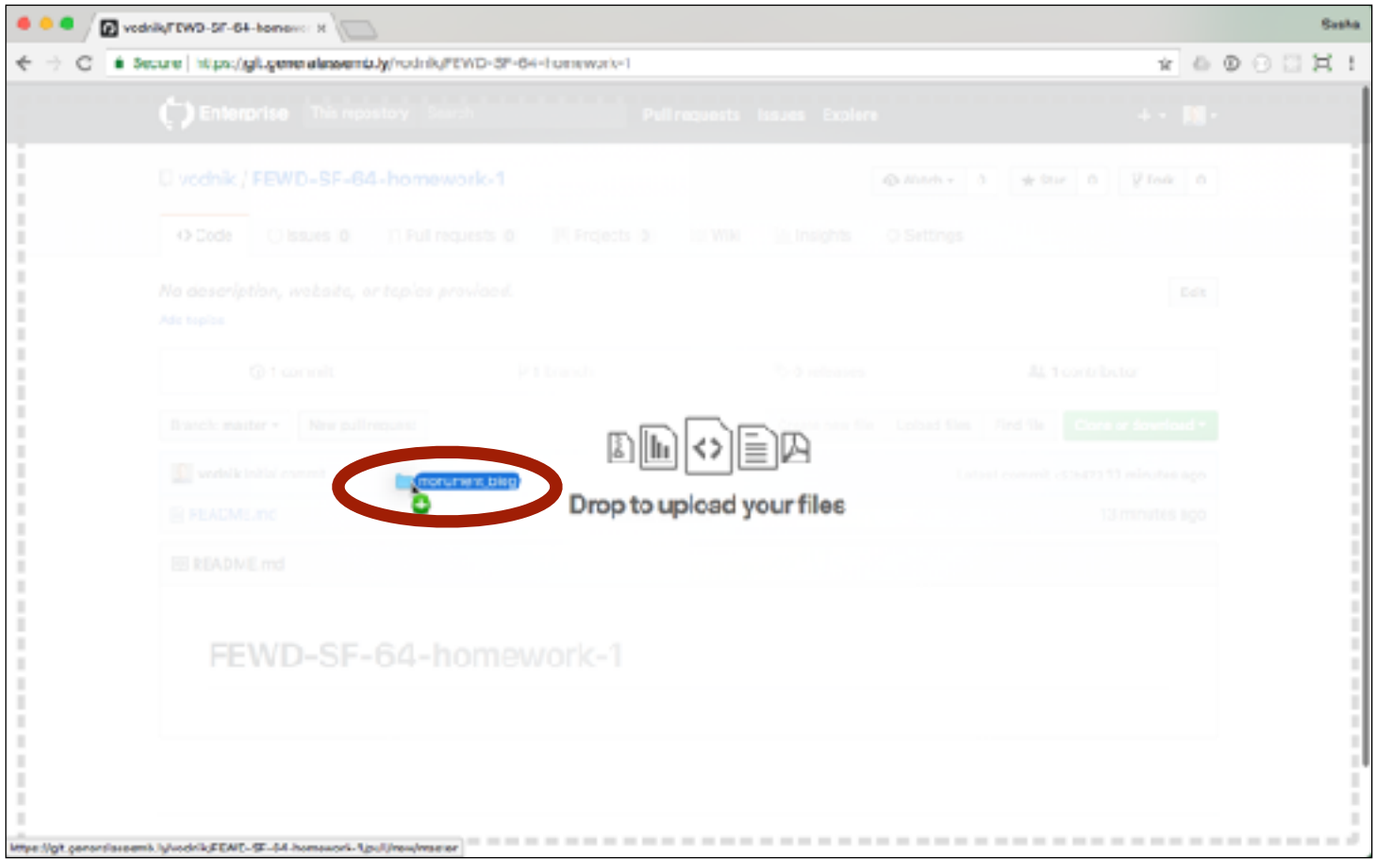
- Switch to the Finder, then navigate to the folder containing the project you want to add to your new repo



DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

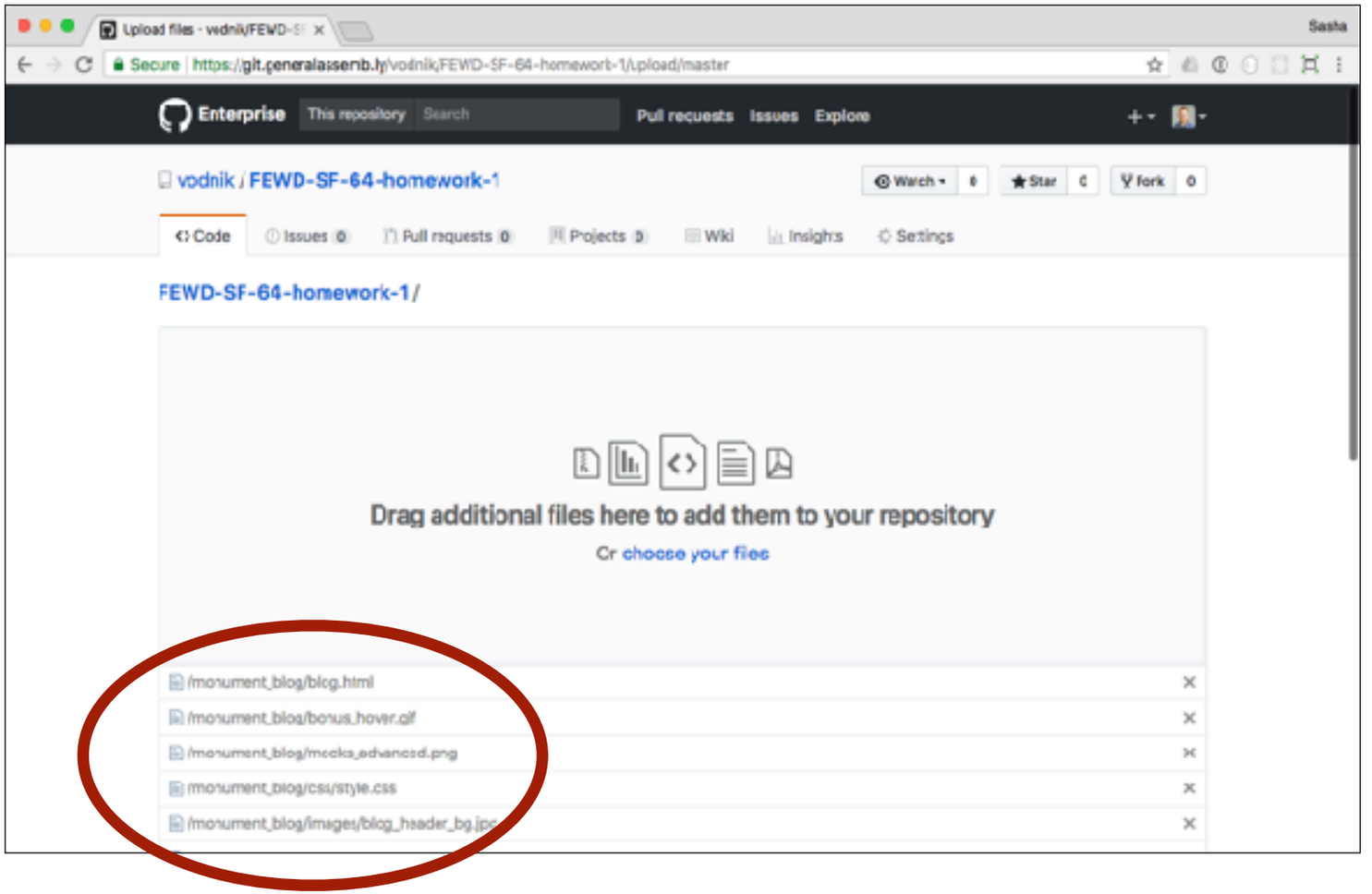
- Drag the entire folder over to the Chrome window displaying your repo





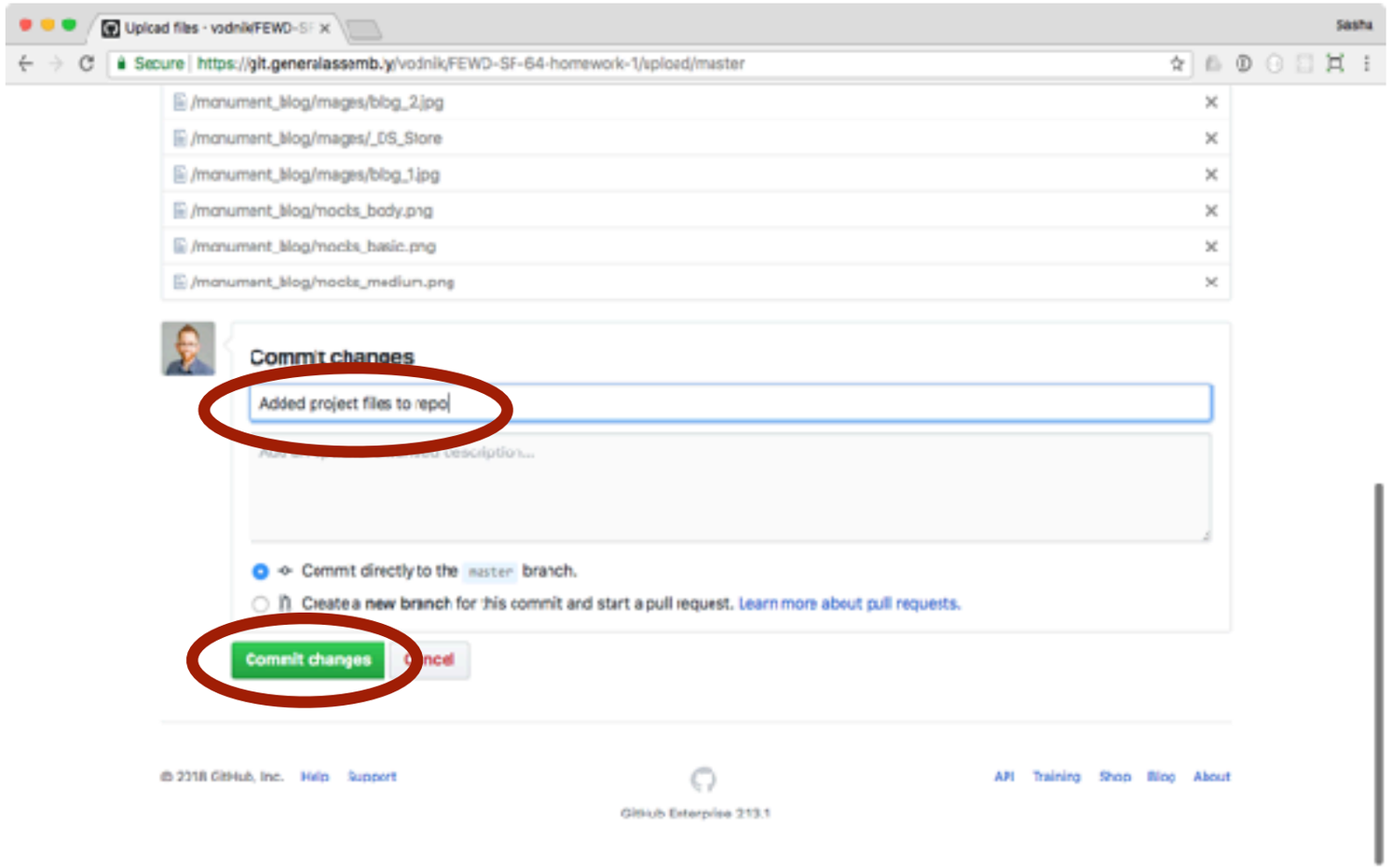
DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

- When copying is finished, examine the list of files to be sure you copied the files you expected to.



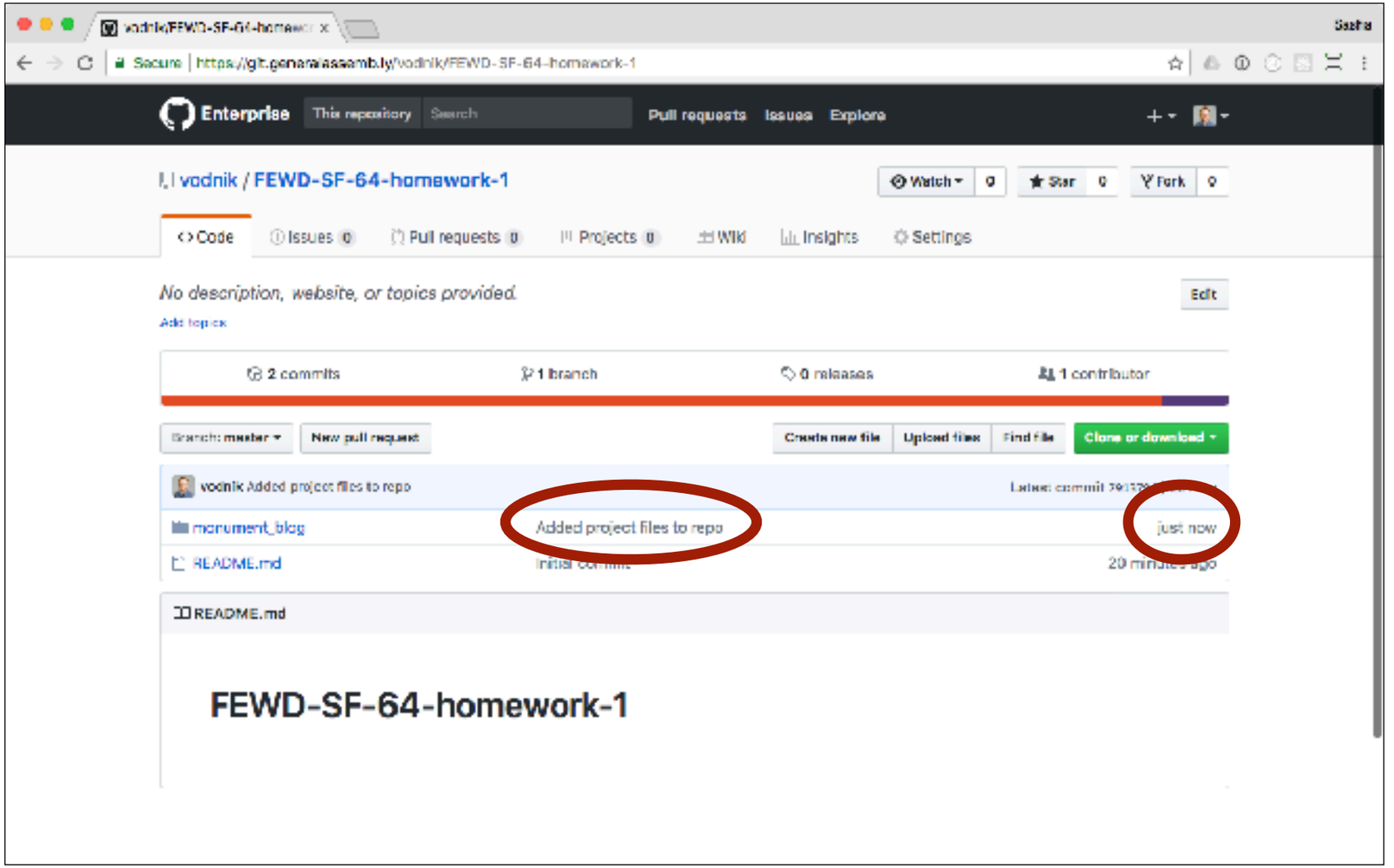
DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

- Scroll to the bottom of the web page
- In the box under the Commit changes heading, type a short description of what changes you're making to the repo
- Click the **Commit changes** button

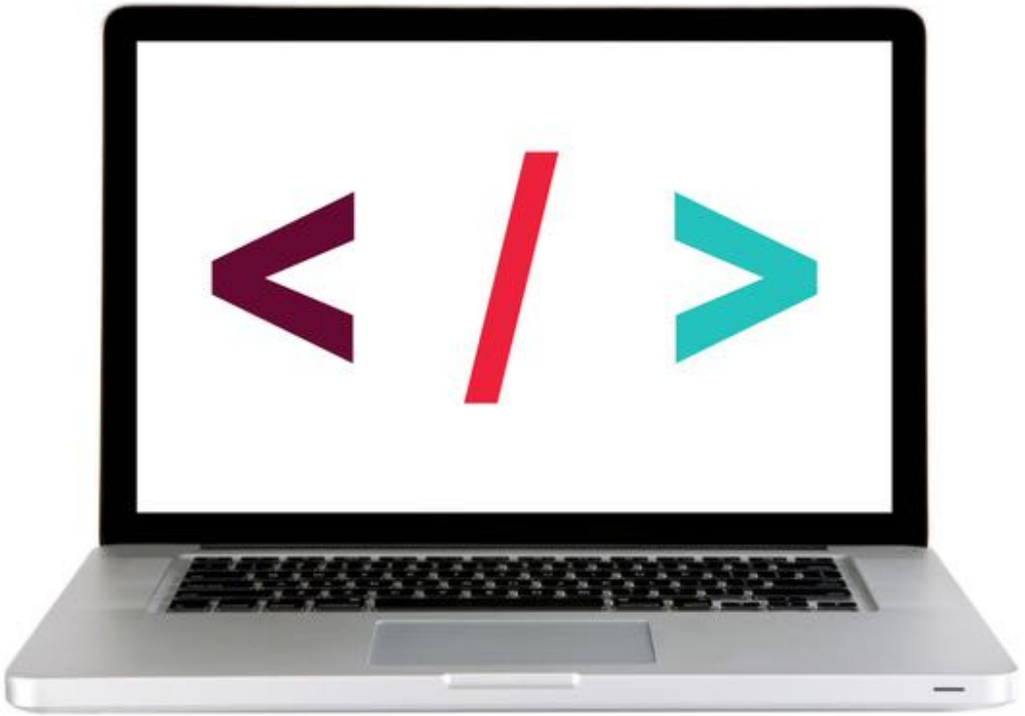


DEMO: CREATING A GITHUB ENTERPRISE REPO AND ADDING FILES TO IT

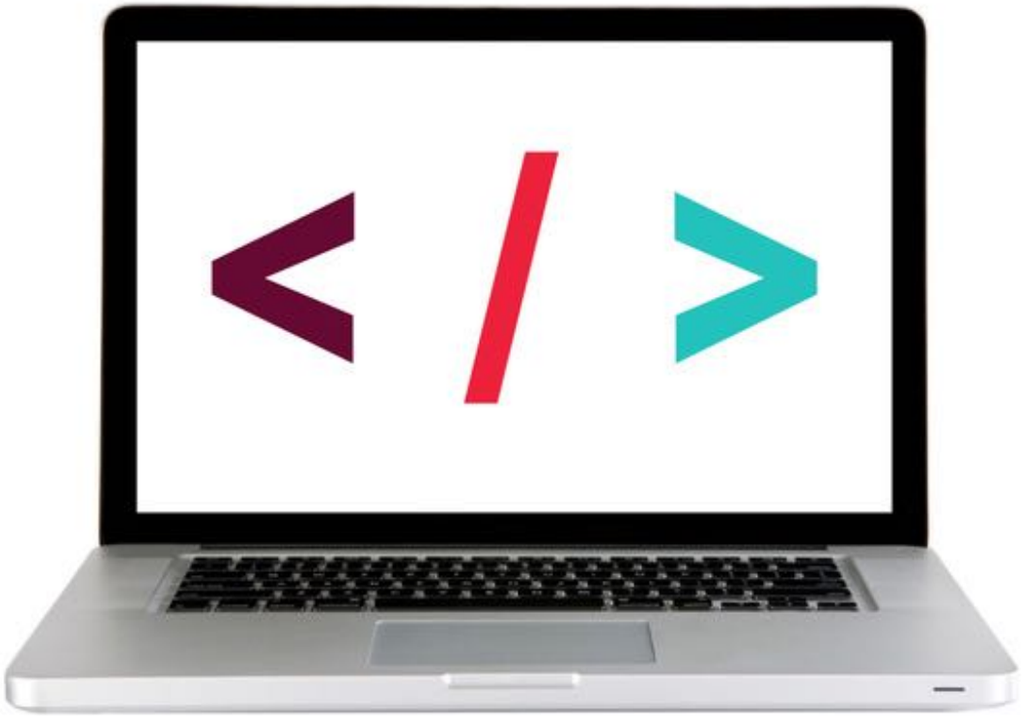
- On the page that opens, the file list should now include the folder you added, showing the comment you specified in the previous step, and indicating that the files were changed recently — SUCCESS!



CODEALONG: CREATE A GITHUB ENTERPRISE REPO AND ADD FILES TO IT

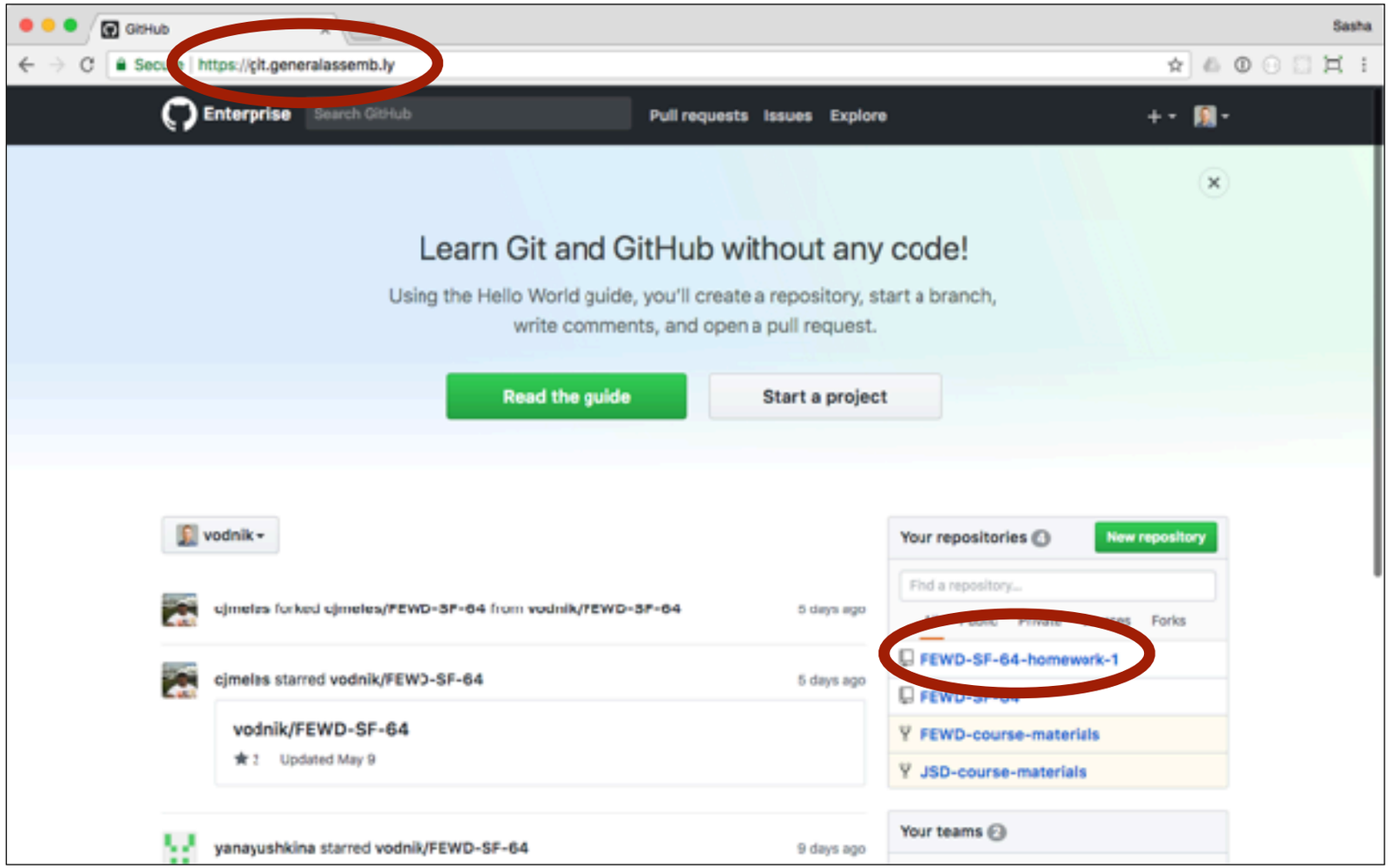


DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO



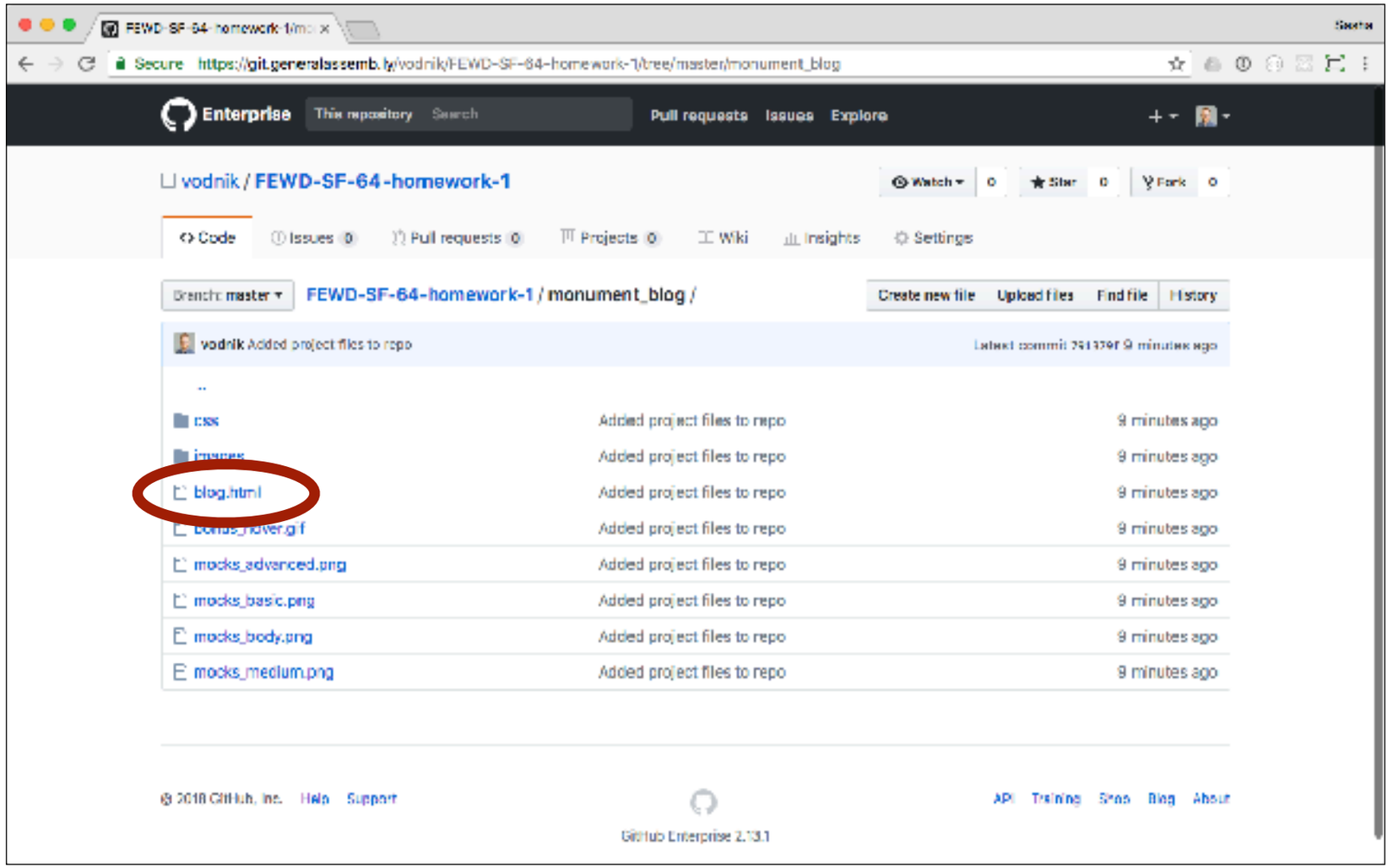
DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

- In Visual Studio Code, save the changes you've made to your code.
- In Chrome, go to git.generalassemb.ly and log in.
- On the right side, click the name of the repo whose files you want to update.



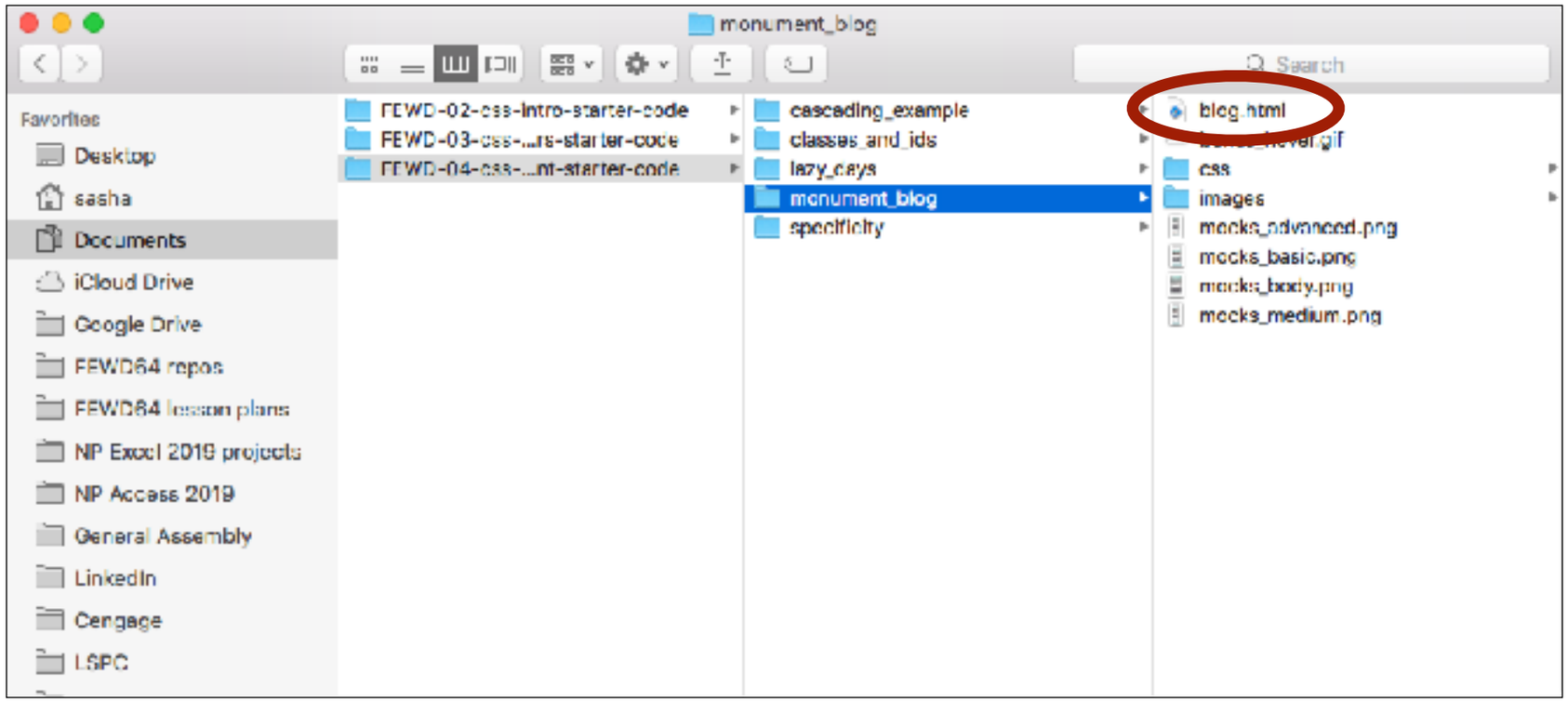
DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

- In your repo, open the folder containing the file you want to update. (To avoid duplicating files, the name of the file you want to update must be visible.)



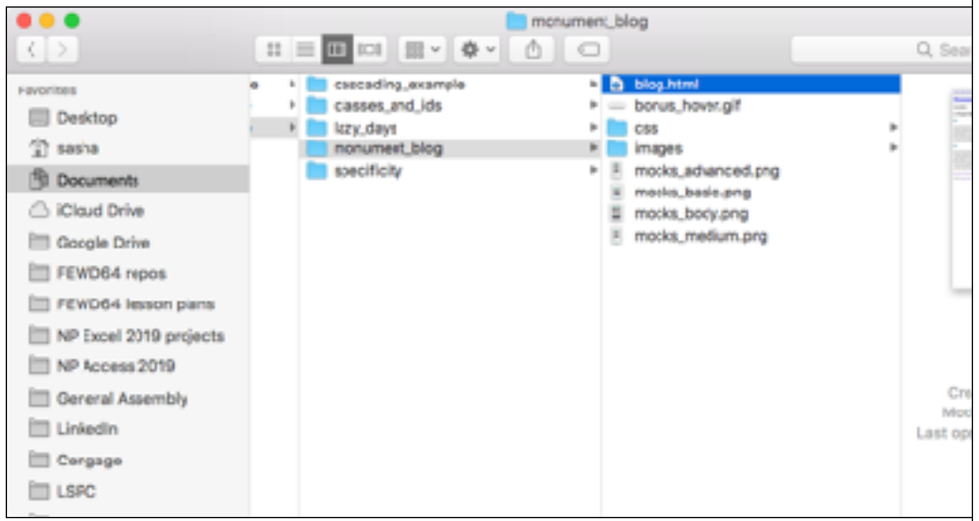
DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

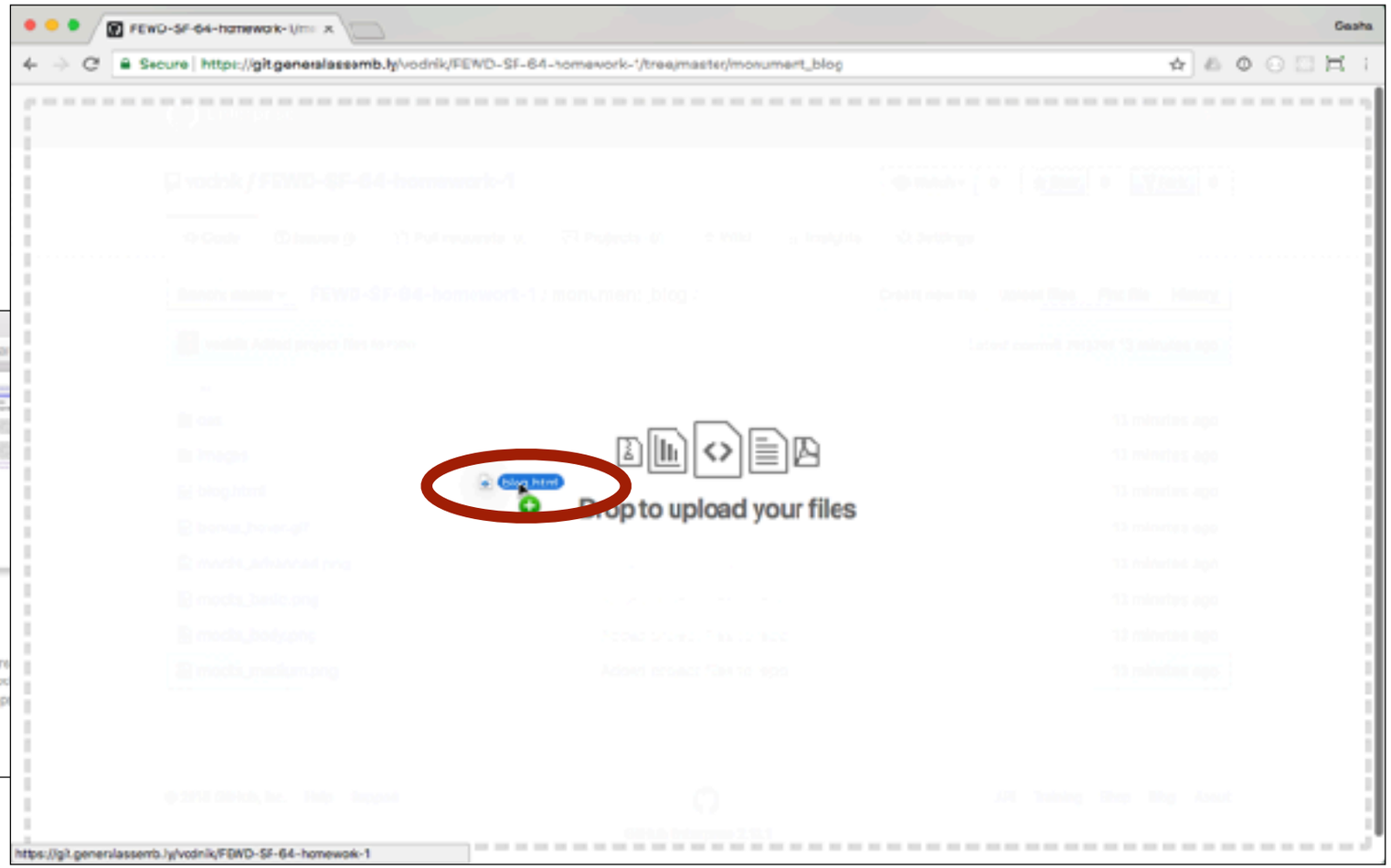
- Switch to the Finder, then navigate to the folder containing a file you want to update



DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

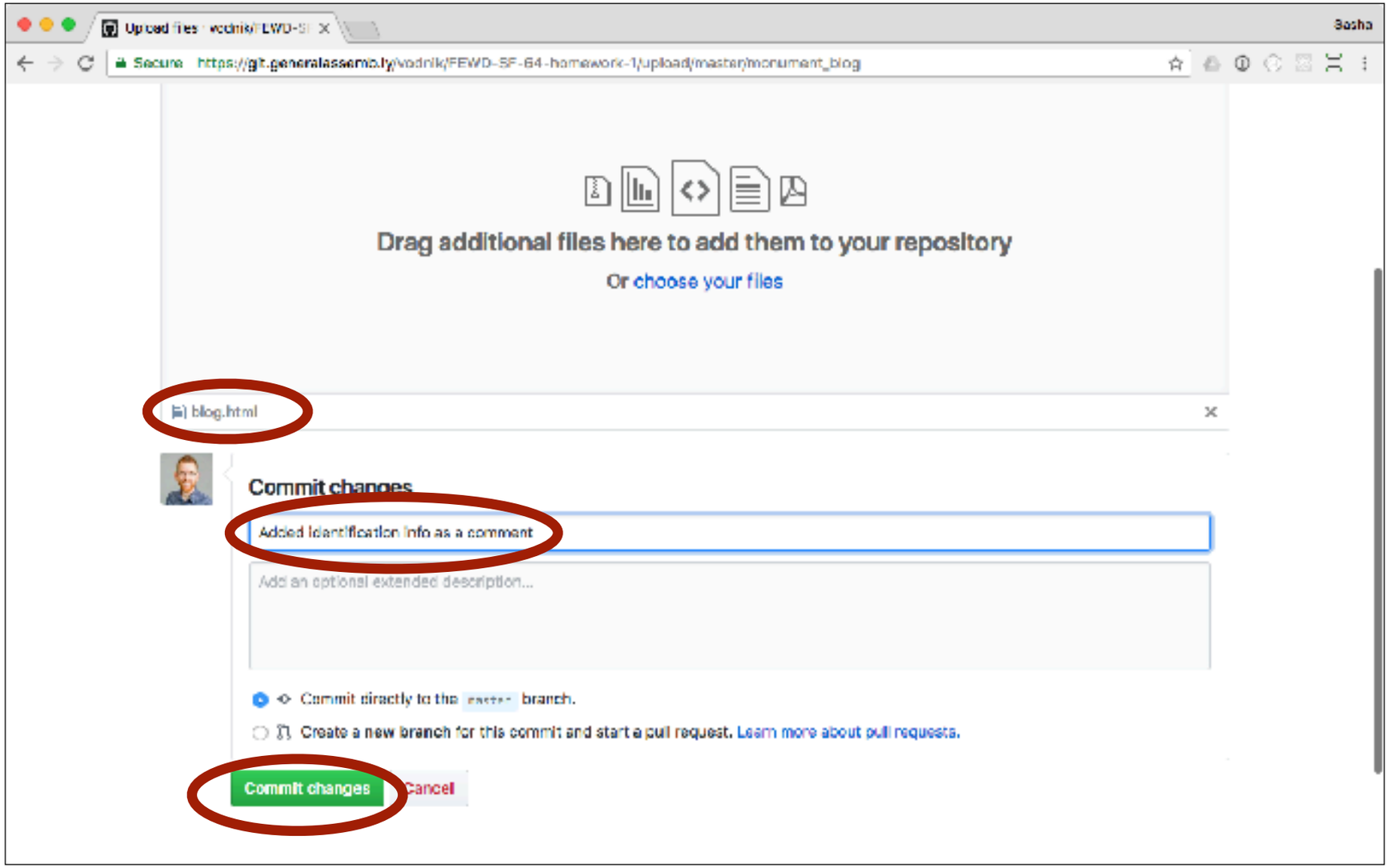
- Drag the changed file over to the Chrome window displaying your repo





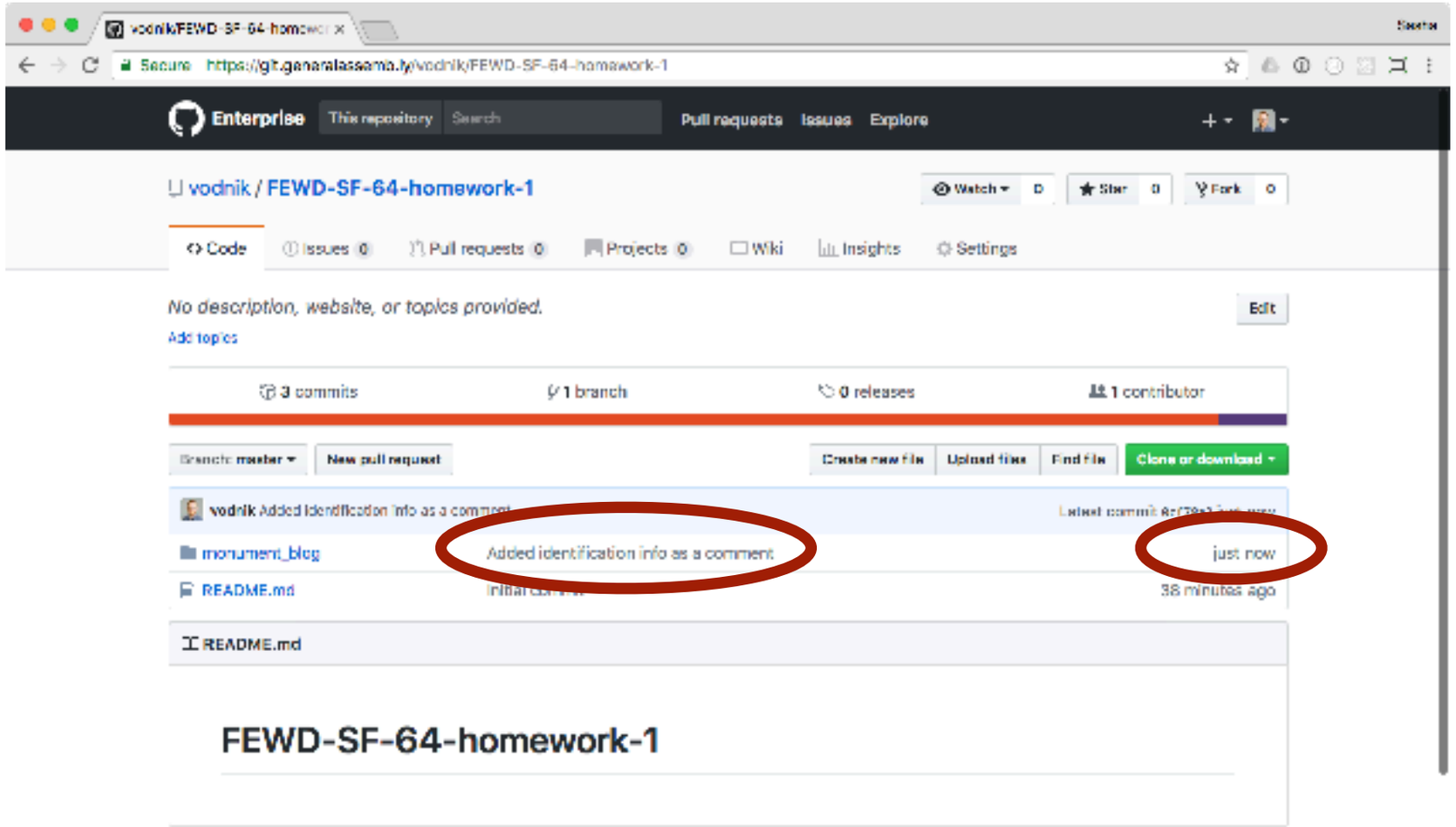
DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

- Verify that the filename you intended to update is displayed.
- Scroll to the bottom, then in the box under the Commit changes heading, type a short description of what changes you're making to the repo
- Click the **Commit changes** button

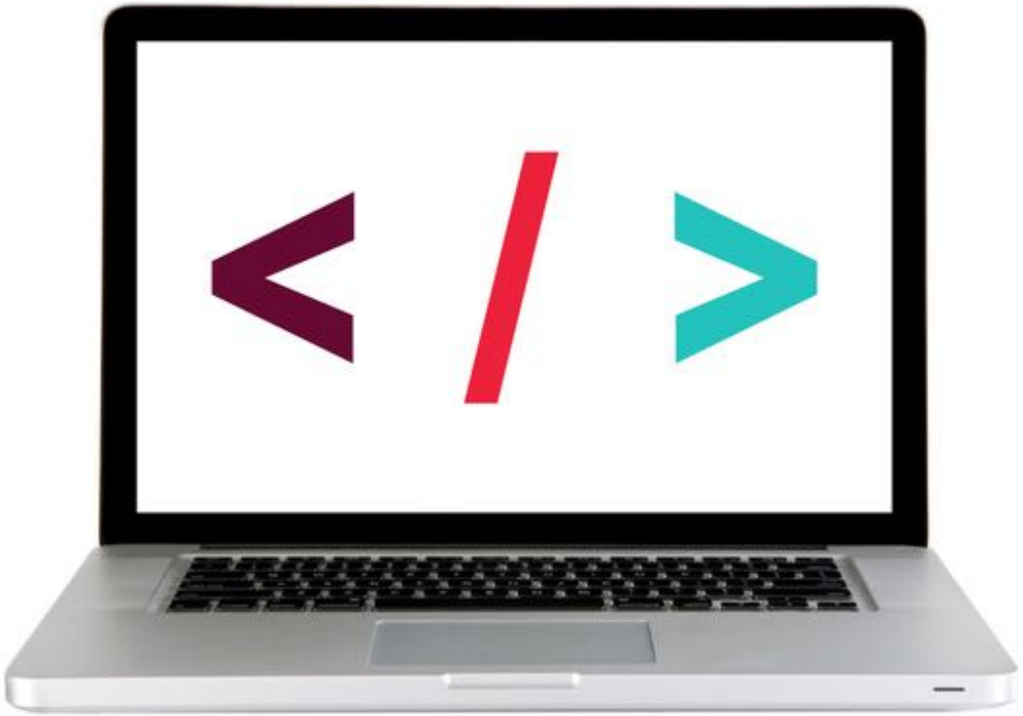


DEMO: UPDATING FILES IN AN EXISTING GITHUB ENTERPRISE REPO

- On the page that opens, the folder you updated should show the comment you specified in the previous step, and indicate that the files were changed recently — SUCCESS!
- You can also open the folder and examine the information for the specific file you updated.



CODEALONG: UPDATE FILES IN AN EXISTING GITHUB ENTERPRISE REPO



LAB: MONUMENT BLOG

OVERVIEW

Congratulations! You have been hired as the developer for Monument Lifestyle Magazine. You are working closely with a designer and have been provided design mockups for the website. Your goal is to replicate these mockups as closely as possible.

You will be adding components to the blog page in the labs/ homework this week and will be working on building the landing page for the site during the lab sessions next week.

ACTIVITY: MONUMENT BLOG PLANNING

KEY OBJECTIVE

- Plan out the Monument Magazine site.

LOCATION

- Starter code > Monument Blog > mocks_basic.png

TASKS

Consider the following questions with your groups:

1. What is the most common font-family?
2. Is there a common color for the text?
3. Is most of the text uppercase? Lowercase?
4. How about the anchors? Are they underlined?
5. Is there a common color for the anchors?

ACTIVITY: MONUMENT BLOG

EXERCISE

KEY OBJECTIVE

- Recreate the Monument Magazine site

TASKS

1. Look through the provided HTML.
2. Start by adding styles that are the most common to the body, using mocks_body.png and mocks_basic.png as a guide
3. **Bonus 1:** Work from mocks_medium.png
4. **Bonus 2:** Work from mocks_advanced.png (You'll need to do some research to implement a background-image in the header and FontAwesome icons in the footer)
5. **Bonus 3:** Work from bonus_hover.gif to add a hover effect to links

LEARNING OBJECTIVES

- Add classes and IDs to HTML elements and apply CSS styles to elements based on class and ID.
- Explain when you would use a class and when you would use an ID.
- Apply CSS to elements based on their relationships.
- Describe inheritance in CSS.
- Create and update a repo on GitHub Enterprise

WEEKLY OVERVIEW

WEEK 2	CSS Selectors / CSS Selectors continued
WEEK 3	Layout / Layout Lab
WEEK 4	Grid Systems & Wireframing / Responsive Design

HOMEWORK

HOMEWORK

- Finish the Monument blog we started in class
- Come up with an idea for your final project (multiple options is fine, too)

EXIT TICKETS