

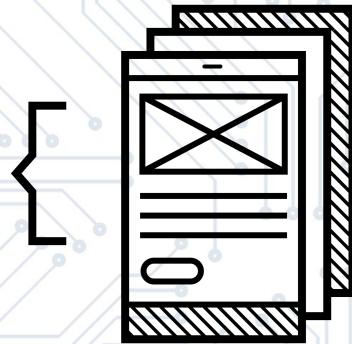
# Positioning Elements on the Web

UX/UI Design Boot Camp  
Lesson 17.3



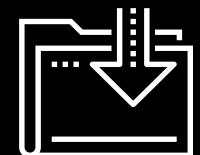


**45 minutes**



# Positioning Elements on the Web

UX/UI Design Boot Camp  
Lesson 17.3



# Today's Objectives

---

By the end of class today, you will:



Position HTML elements using CSS floats and automatic margins.



Position inline, block, and inline-block HTML elements to build webpages.



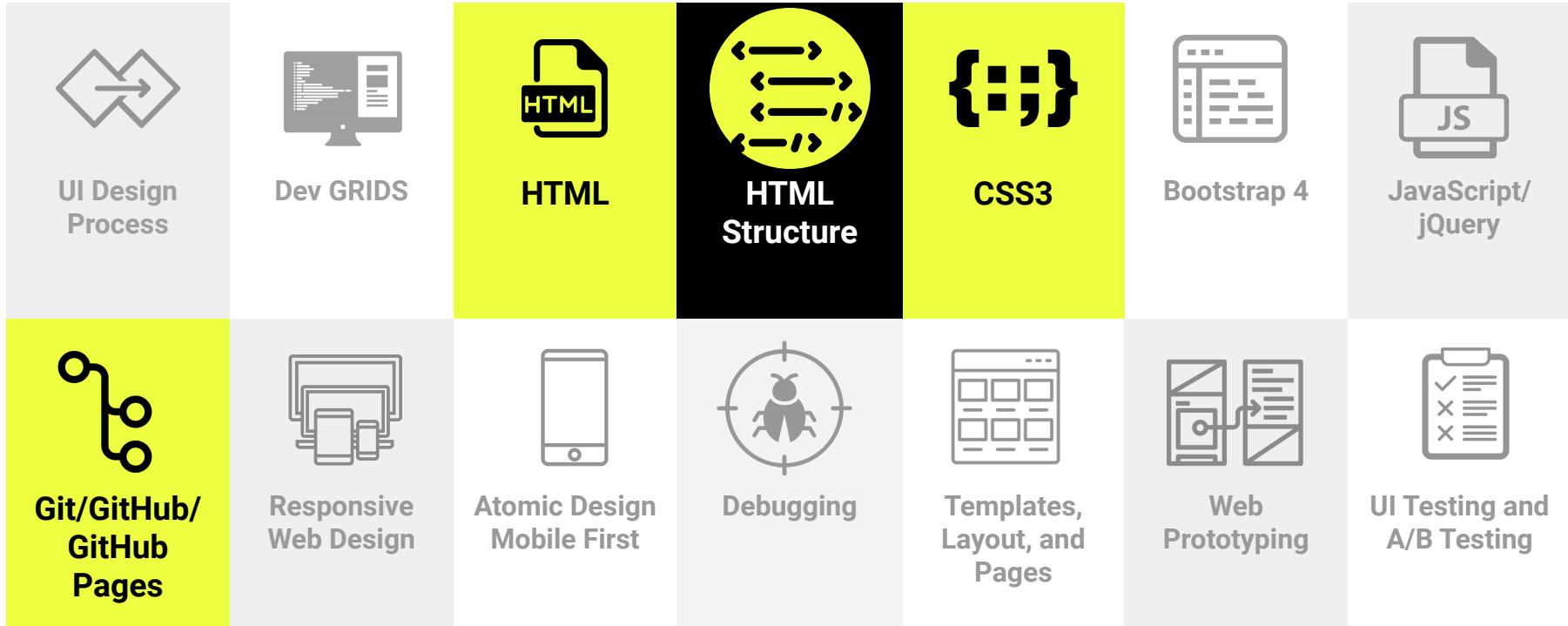
Create HTML divs that overlap by positioning elements absolutely with CSS.



Code your first webpage and host it via GitHub Pages to become your first website.

# Front-End Development Units

Today we'll continue with the basics of front-end development.



# CSS Float and Display Properties

# CSS Float Property

# Float CSS



# Float CSS

---

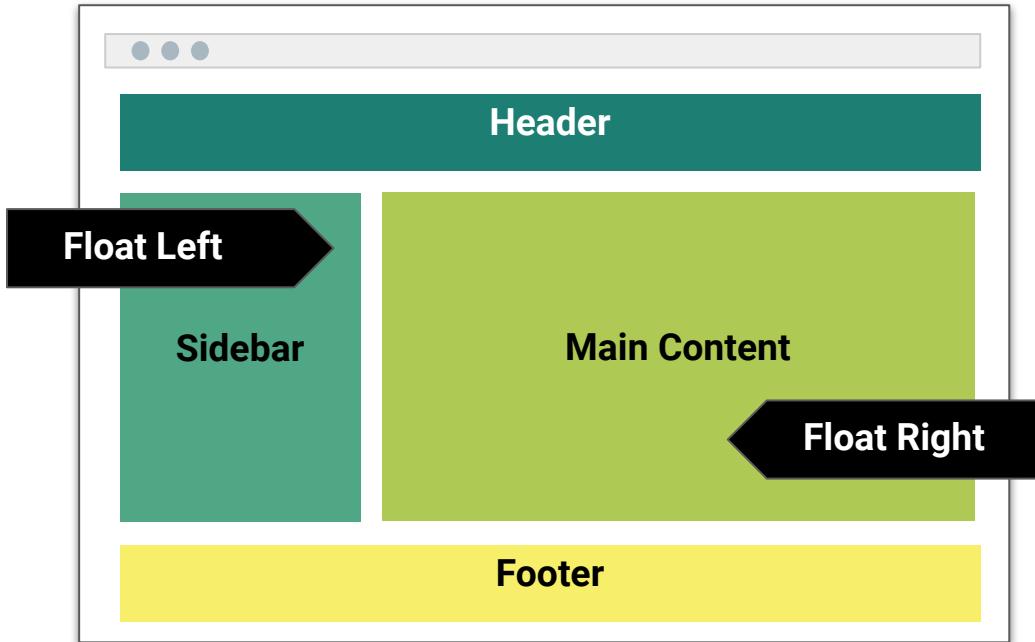
The **float CSS** property places an element on the left or right side of its container, allowing text and inline elements to wrap around it.

Floats were designed to wrap text around images, but can also be used to make an element move to the farthest left or right in its container.



# Float CSS

Float can be used to build containers that line up next each other but it can be tricky.



## CSS

```
#sidebar {  
    float: left;  
}  
  
#main-content {  
    float: right;  
}
```



## Time For a Quick Video

---

[Floats in CSS](#)

# CSS Display Property

# CSS Display Property

The `display` property is the most important CSS property for controlling layout.

As a beginner, **most problems you encounter** early on will be caused by the `display` property and trying to position elements in and around each other.

The `display` property specifies if/how an element is displayed.

Each behaves differently from others and understanding the differences will save you both time and frustration.

There are many different display properties. Here are the basic ones we will be working with first.

We will be covering the more advanced display properties next week.

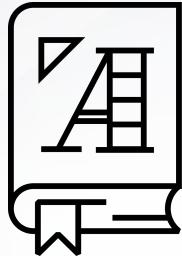
```
display: block; // Sets the element to display block.
```

```
display: inline; // Sets the element to display as a inline element.
```

```
display: none; // hides a element on your html page.
```

```
display: inline-block; // sets the element to display as a inline-block element.
```

# **HTML: Block-Level Elements**



**HTML: Block-level elements** always start on a new line and take up the full width of its parent container (stretches out to the left and right as far as it can).

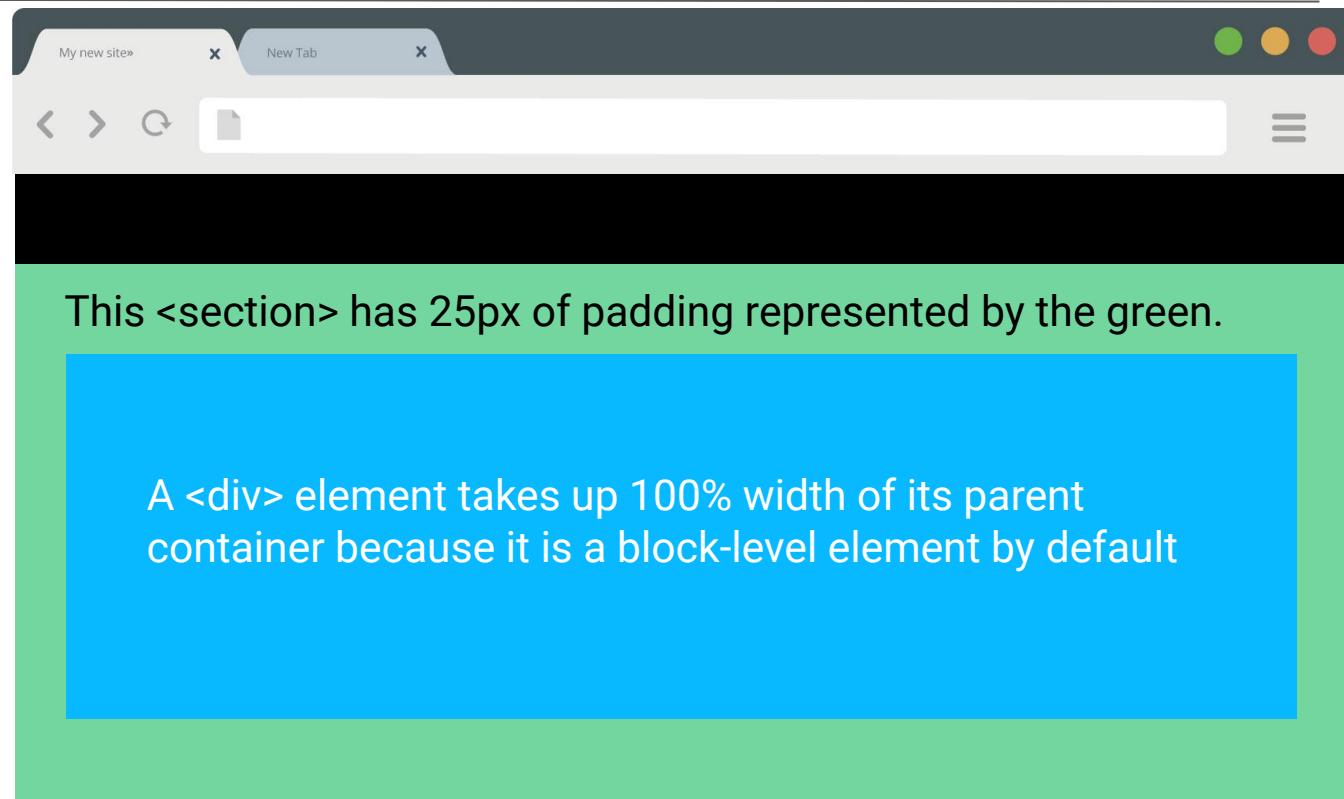
Block-level elements = `<div>`, `<section>`, `<p>`, `<h1-h6>`, `<ol>`, `<ul>`, `<form>`, `<blockquote>`, etc.

List of block-level elements: [https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level\\_elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements)

# Block-Level Elements

Block-level elements respect the margin and padding of their parent containers.

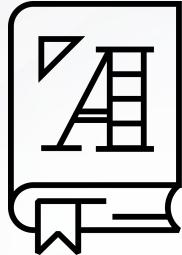
Block-level elements are used to contain inline elements.



This <section> has 25px of padding represented by the green.

A <div> element takes up 100% width of its parent container because it is a block-level element by default

# **HTML: Inline Elements**



**HTML: Inline elements** only occupy the space bounded by the tags **defining the element**, instead of breaking the flow of the content.

Inline HTML elements examples = `<button>`, `<input>`, `<label>`, `<img>`, `<textarea>`, `<span>`, etc.  
List of inline elements: [https://developer.mozilla.org/en-US/docs/Web/HTML/Inline\\_elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements)

# Inline Elements

Inline elements only take up as much space as needed and will sit beside other inline elements. Below, we see an inline element, a span, side by side with text (also inline).

The following span is an `inline element`; its background has been colored to display both the beginning and end of the inline element's tag.

```
HTML
1 <div>The following span is an <span class="highlight">inline
element</span>;
2 its background has been colored to display both the beginning
and end of
3 the inline element's tag.</div>
```

```
CSS
1 body {
2   margin: 0;
3   padding: 4px;
4   margin-top: 15px;
5   border: 1px solid #333;
6 }
7
8 .highlight {
9   background-color:#ee3;
10 }
```

The following span is an `inline element`; its background has been colored to display both the beginning and end of the inline element's tag.

# **HTML and CSS: Inline-Block Elements**



**display: inline-block;** sets an HTML element to display as an inline-block container via CSS class or ID.

The element itself is formatted as an inline element, but you can apply height and width values.

# display: inline-block Elements

inline-block elements  
display as inline so they  
line up next to each other.

This could be a navigation  
bar or a series of cards  
containing content.

The screenshot shows a code editor interface with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <div id="container">
2   <div class="redBlock">
3     I'm a inline block element
4   </div>
5   <div class="redBlock">
6     I'm a inline block element
7   </div>
8   <div class="redBlock">
9     I'm a inline block element
10  </div>
11 </div>
```

The right pane is labeled "CSS" and contains the following code:

```
1 #container {
2   background-color: #DDDDDD;
3   height: 250px;
4 }
5 .redBlock {
6   display: inline-block;
7   background-color: red;
8   padding: 50px;
9   color: white;
10  font-family: arial;
11 }
```

Three red arrows point from the three "redBlock" div elements in the HTML code to the "display: inline-block;" line in the CSS code. Below the code editor, there is a visual representation of three red rectangular boxes arranged horizontally, each containing the text "I'm a inline block element".

# Inline-Block Elements

Inline-block elements can be used to create two-column layouts.

Be warned! With this method, you will not be able to have the two elements fill 100% of a screen.

There will always be a small gap between.

The screenshot shows a code editor interface with three panels: HTML, CSS, and JS. The HTML panel contains the following code:

```
<section id="container">
  <div class="containerBlock left">
    Container: Left
  </div>
  <div class="containerBlock right">
    Container: Right
  </div>
</section>
```

The CSS panel contains the following code:

```
.containerBlock {
  display: inline-block;
  height: 500px;
  font-size: 18px;
  padding: 5px;
}
.left {
  width: 40%;
  background-color: green;
  color: white;
}
.right {
  width: 58%;
  background-color: red;
}
```

The preview area below the editor shows two columns: "Container: Left" (green) and "Container: Right" (red), separated by a thin white gap.

# CSS Display Review

# CSS Display Recap

---

## HTML

**Block-level** elements fill 100% of the width of their containers.

Block Element

Block Element

Block Element

## HTML

**Inline elements** only take up the space they need and line up next to other inline elements. You cannot give inline elements height or width.

Inline elements display next to each other. This **span** sits next to other elements and doesn't take up space.

## HTML/CSS

**Inline-block** elements line up next to each other like inline elements.

You can give inline-block elements heights and widths.

Inline-Block

Inline-Block

Inline-Block

# CSS Width-Based Percentages

# CSS Width-Based Percentages Using Relative Units

We have referenced content that fills 100% of the width of its container. We can apply percentage-based widths and margins to block-level elements.

The screenshot shows a code editor interface with two panes. The left pane is labeled 'HTML' and contains the following code:

```
<div class="width">100% Width Issues</div>
```

The right pane is labeled 'CSS' and contains the following code:

```
div {  
  width: 100%;  
  margin: 5%;  
  padding: 1%;  
  background-color: red;  
  height: 250px;  
  color: white;  
  font-family: arial, helvetica;  
  font-size: 22px;  
}
```

It's worth noting, however, how the box model calculates widths on a page, as setting them correctly is necessary for successful layouts.



# CSS Width-Based Percentages Explained

---

Our content is breaking out of its container because we have it set to 100% width and 5% margin (left and right) and 1% padding.

```
div {  
    width: 100%;  
    margin: 5%;  
    padding: 1%;  
    background-color: red;  
    height: 250px;  
    color: white;  
    font-family: arial, helvetica;  
    font-size: 22px;  
}
```

# CSS Width-Based Percentages Explained

---

Margin and padding applies itself to the left, right, top, and bottom of an element.

Our 5% margin really takes up 5% on the left and 5% on the right (10% total), causing our div to break. The same applies for padding.

```
div {  
    width: 100%;  
    margin: 5%;  
    padding: 1%;  
    background-color: red;  
    height: 250px;  
    color: white;  
    font-family: arial, helvetica;  
    font-size: 22px;  
}
```

# CSS Width-Based Percentages Explained

To get our layout to fill up 100% of its width, we need to make sure our box model equals 100%.

We set our width to take up 88% of our layout.

We set our margin to be 5%. Because the margin includes the left and the right side, a margin of 5% is really equal to 10% of the total width.

Lastly, we have 1% padding set as well. The left and the right of our div take up 2% of the space with padding.

$$88\% \text{ width} + 10\% \text{ margin} + 2\% \text{ padding} = 100\% \text{ width}$$

The screenshot shows a browser's developer tools with two panels: 'HTML' and 'CSS'. In the 'HTML' panel, there is a single div element with the class 'width' containing the text '100% Width Issues'. In the 'CSS' panel, the following styles are defined for the div:

```
div {  
    width: 88%;  
    margin: 5%;  
    padding: 1%;  
    background-color: red;  
    height: 250px;  
    color: white;  
    font-family: arial, helvetica;  
    font-size: 22px;  
}
```

Red arrows point from each of the three text blocks above to the corresponding CSS properties: 'width: 88%', 'margin: 5%', and 'padding: 1%'.

A large red box at the bottom contains the text '100% Width Issues'.

# Let's Review: Layout Positioning

---



What is an HTML inline element?



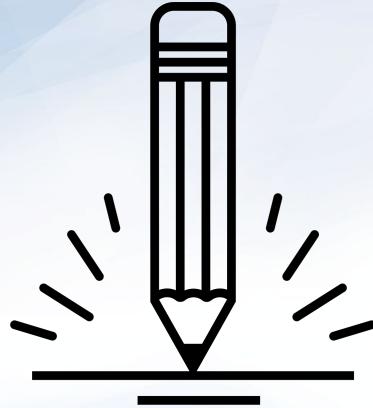
What is a HTML block element?



What is an HTML/CSS inline-block element?



What is a CSS float used for?



# Activity:

## Practice with CSS Positioning

(Instructions sent via Slack)

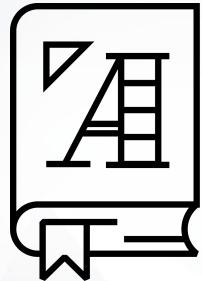
Suggested Time:  
20 minutes





Time's Up! Let's Review.

# CSS Position: Five Properties



In CSS, the **position property** is used to create complex layouts where you need items placed in a specific area.

# CSS – Position: Static

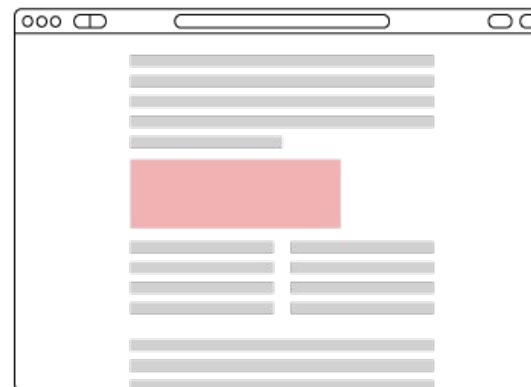
We position the element according to the normal flow of the document. Top, right, bottom, left and z-index have no effect on elements that are not positioned.

Static elements are the default value of all elements.

position: static



RESPECTS CONTENT FLOW



CAN'T BE POSITIONED OUTSIDE  
NATURAL POSITION



SCROLLS WITH CONTENT

# CSS – Position: Relative

---

Relative positioning allows you to control the position of your element using top, left, right, and bottom.

More important, relative positioning is used to contain absolutely positioned elements.

*(Absolutely positioned elements are positioned according to their nearest positioned container.  
More on this later.)*

```
position: relative;  
height: 250px;  
Width: 250px;
```

```
position:  
absolute;  
top: 30%;  
right: -25px;
```

# CSS – Position: Fixed

The element is removed from the normal document flow. It is positioned to the viewport of the browser and will always stay where you put it even if the user scrolls down the page.

## Fixed Element

It is so easy to get started



1 Minute Setup



Start Chatting



Track Progress



# CSS – Position: Sticky

---

This is the newest position property.

An element is positioned according to the normal flow of the document, and then sticks to the top of browser when a certain scroll height is met.

## P R E S E N T A T I O N

*Sliding content with sticky tab nav*

ES6

Flexbox

React

Angular

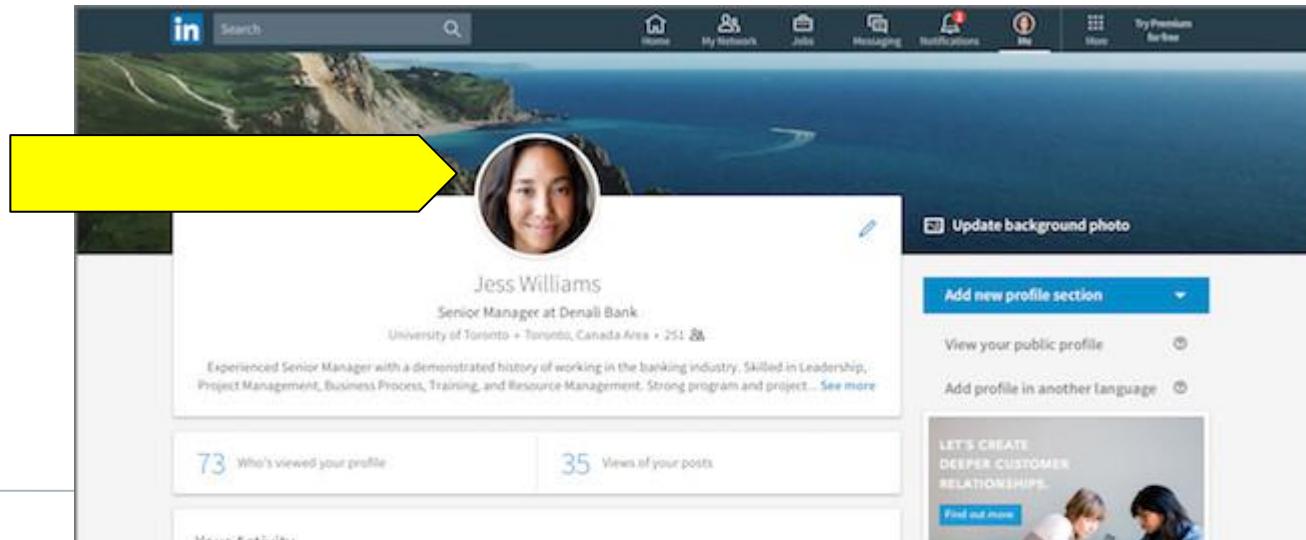
Other

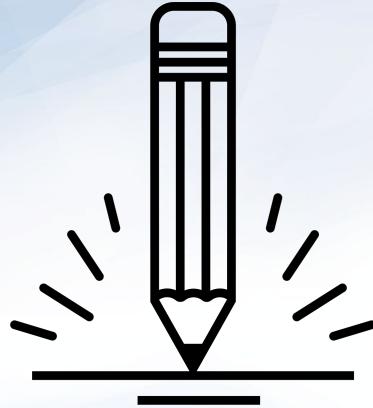
# CSS – Position: Absolute

Absolute elements are removed from the document flow and are positioned relative to their nearest-positioned ancestor.

If a parent container has any type of position property applied to it (except static), then absolutely positioned elements will be contained and positioned inside it. If no element has a stated position, then it will be positioned to the browser viewport or window.

This element is absolutely positioned, so it can be pushed above the div to create this look.





# Activity:

## CSS Positioning With Relative, Absolute, and Fixed Positioning

(Instructions sent via Slack)

Suggested Time:  
20 minutes





Time's Up! Let's Review.

*Break*





## Challenge:

### Build a One-Page Website

You will practice building a basic webpage and then upload it to GitHub Pages.

(Instructions sent via Slack)

Suggested Time:  
50 minutes





Time's Up! Let's Review.

# **Webpage vs Website**

# Webpage vs. Website



VS.

All

Hello, Sign in  
Account & Lists

Returns & Orders Try Prime

Cart

Alexa, drop in on the kids room.

echo dot

Amazon's response to COVID-19

Bargain tech accessories

Kindle book deals

The Marvelous Mrs. Maisel

WATCH NOW | prime video

Sign in for the best experience

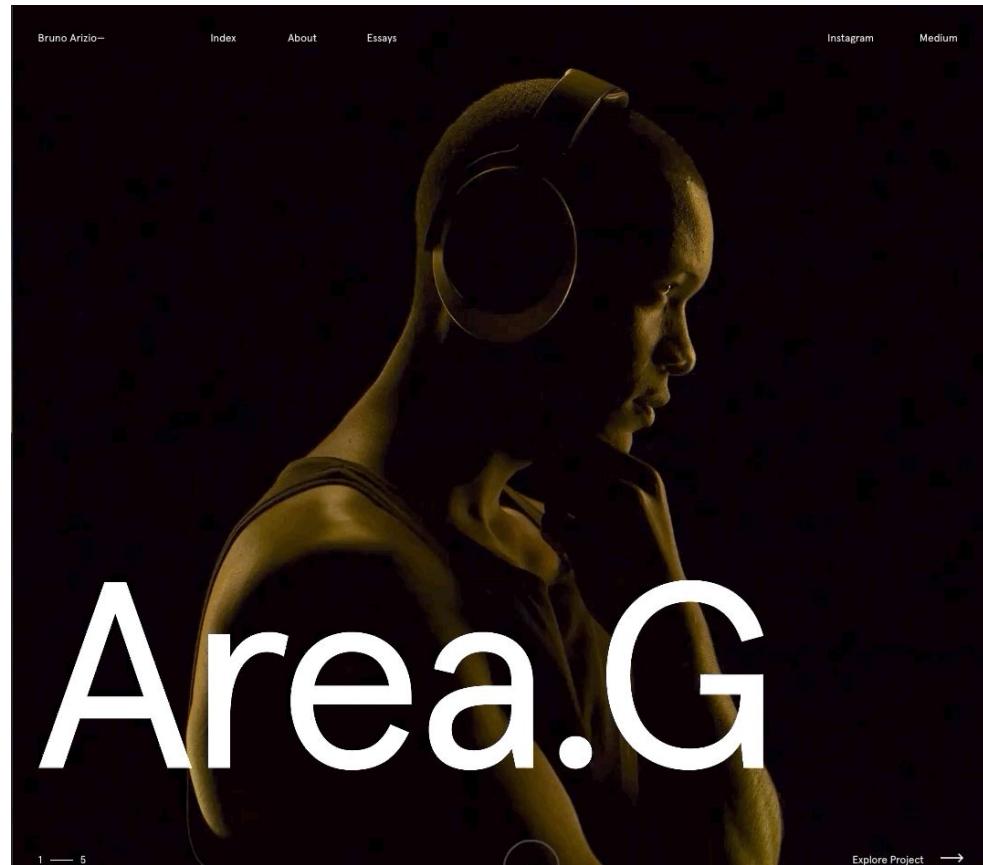
Sign in securely

# A Webpage Is ...

A domain that is connected to the internet that displays a single HTML document that can be viewed in a web browser.

<https://brunoarizio.com/> > homepage  
(index.html)

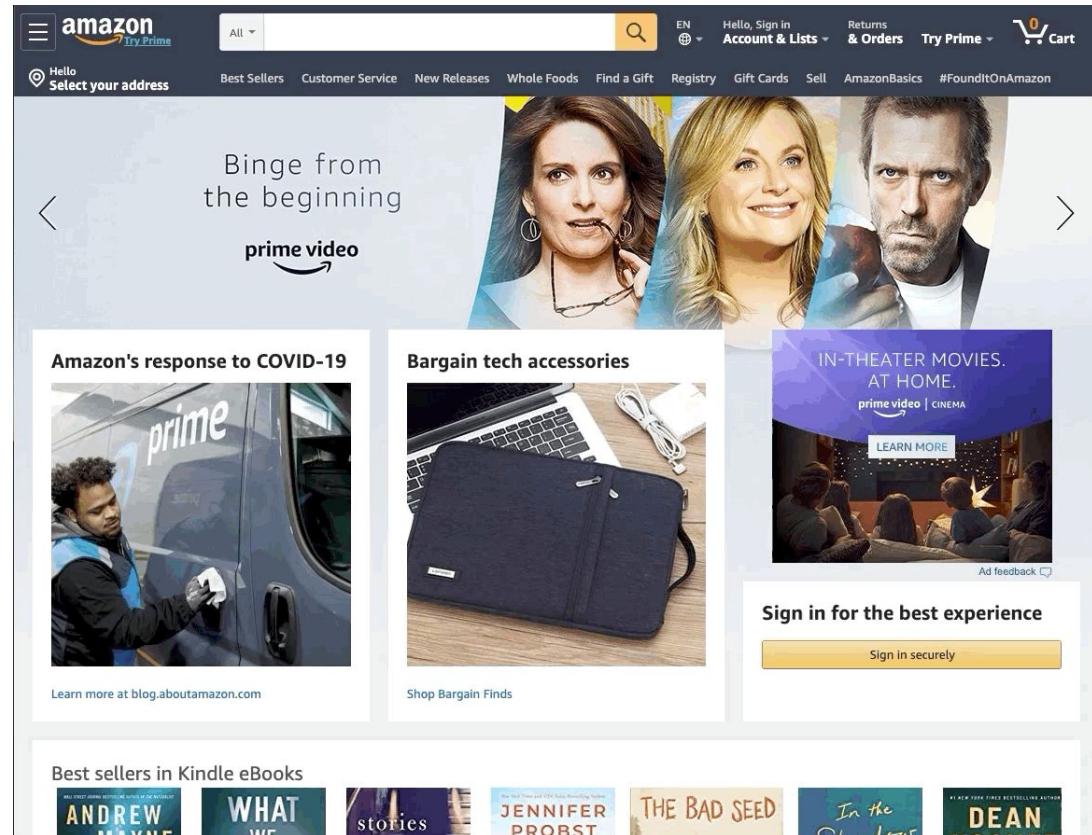
- <https://brunoarizio.com/about> > about page  
(about.html)
- <https://brunoarizio.com/essays> > essays page  
(essays.html)



# A Website is ...

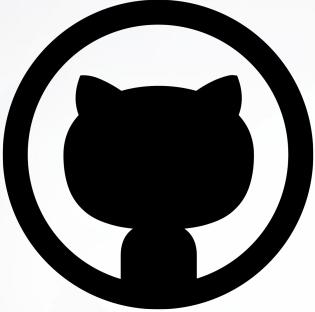
A domain connected to the internet that maintains one or more pages on the World Wide Web.

For example,  
Amazon's website has 1000's  
of webpages.





# GitHub and Version Control



**Git/GitHub** is a distributed **version control** system for tracking changes in source code during software development.

It is designed for coordinating work among programmers, but it can be used to track changes in any set of files.

# Why Git?

It is a version control system that allows team members (developers, UX, UI) to collaborate without overriding each other's work.

Git keeps a timeline and allows your code repo to be protected and fixed quickly.

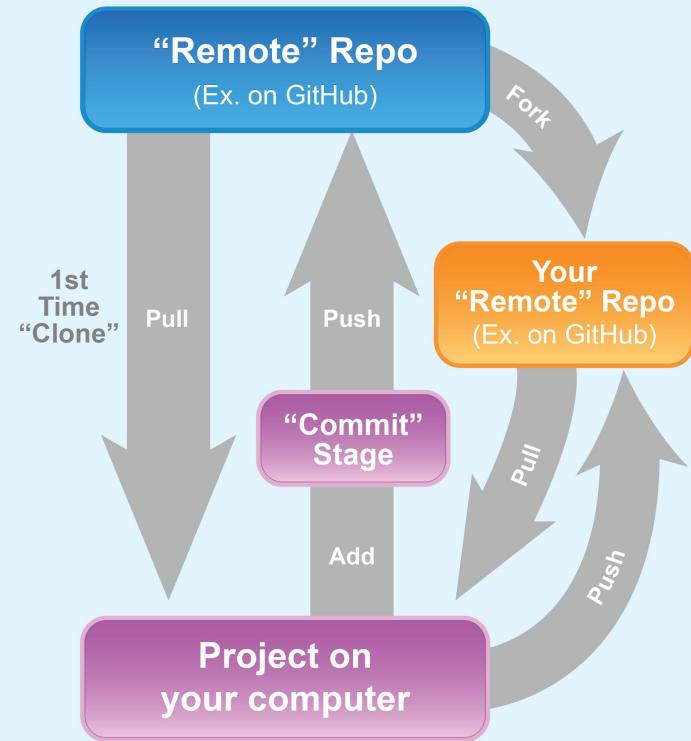
<https://git-scm.com/book/en/v2/Git-Internals-Git-References>

<https://www.c-sharpcorner.com/article/what-is-version-control-git-vs-tfs/>

<https://medium.com/shyp-design/managing-style-guides-at-shyp-c217116c8126>

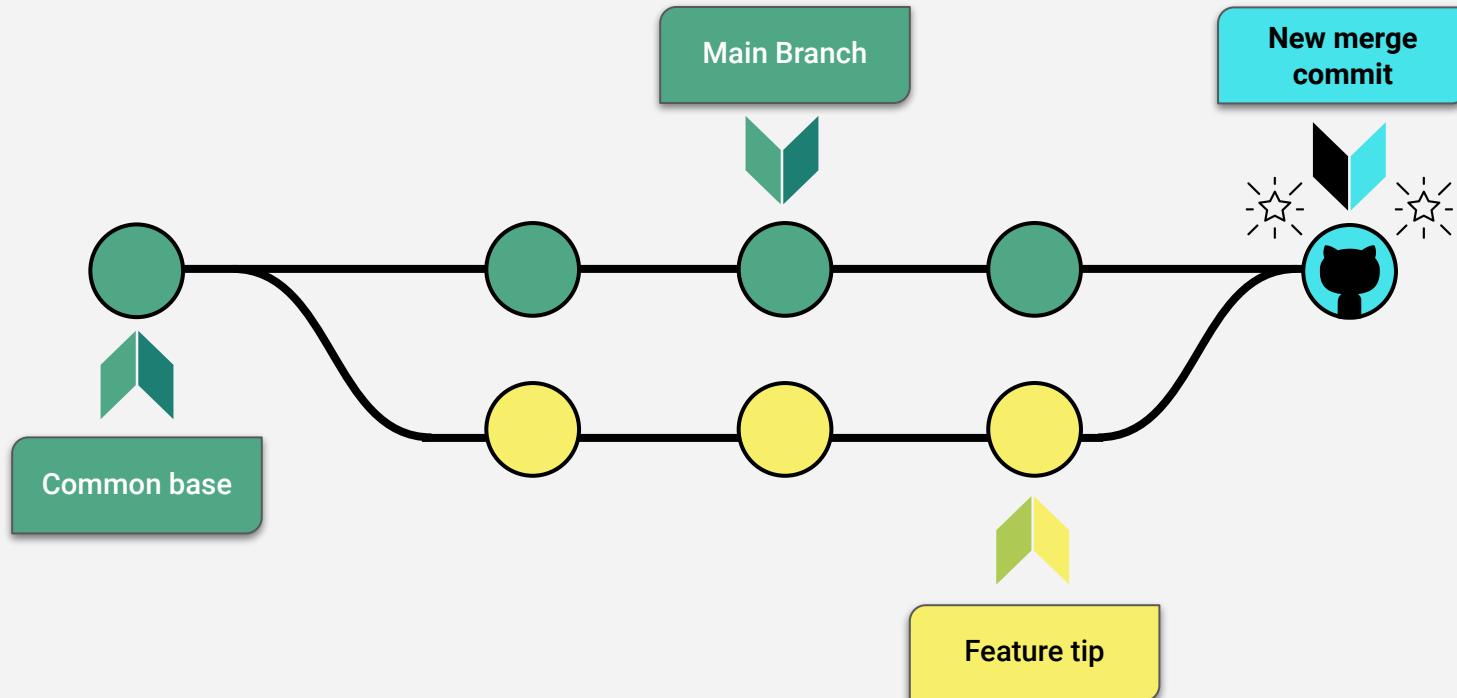
<https://www.atlassian.com/git/tutorials/why-git>

## Git for Non-Developers (in a Tiny Nutshell)



# What Is Version Control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.



# What Is Version Control?

---

Git tracks changes to any file that is being tracked by Git.

Saved versions of these changes are called commits. They represent individual changes in a file that you then pushed to GitHub.

Commits are tracked with a number

- Commits on Jun 29, 2019

sanity check	 Alan Huber committed yesterday	 7e047ed 
minor edits	 Alan Huber committed yesterday	 1bb3490 
merge complete	 Alan Huber committed yesterday	 12eba76 
Merge branch '16.2-lp-edits' of https://github.com/coding-boot-camp/u...	 Alan Huber committed yesterday	 806747f 

# What Is Version Control?

GitHub/GitHub Desktop tracks individual changes in your file and allows you to roll back to previous commits if you make a mistake.

Git is an industry standard for any kind of development job. But it is gaining popularity in other industries as well.

When **commits** are created, they are given descriptions so you know what you changed.

Give them descriptive names!

The image shows a screenshot of GitHub Desktop. At the top, there's a navigation bar with 'Changes 2' (highlighted), 'History', and a file tab labeled 'js/index.js'. Below this, a sidebar lists three files: 'index.html' and 'js/index.js' (both checked) and 'js/index.js' (selected). The main area displays a code diff for 'js/index.js'. The left column shows line numbers (1-28) and the right column shows the corresponding code. A red arrow points from the word 'Summary' in the commit dialog below to the 'Summary (required)' field in the diff view. Another red arrow points from the word 'Description' in the commit dialog to the 'Description' input field in the diff view. The commit dialog at the bottom has fields for 'Summary (required)' and 'Description', and a blue 'Commit to master' button.

```
@@ -1,10 +1,3 @@
-var toggle = document.getElementsByClassName("toggle")
-
-const toggleContent = function() {
-  var clickTargetData = this.getAttribute("data");
-  slideCheck(clickTargetData);
-};

function slideCheck(clickTarget) {
  var code = document.getElementById('code');
  var design = document.getElementById('design');

@@ -23,6 +16,12 @@
  //do nothing
}

+var toggle = document.getElementsByClassName("toggle")
+
+const toggleContent = function() {
+  var clickTargetData = this.getAttribute("data");
+  slideCheck(clickTargetData);
+};

for (var i = 0; i < toggle.length; i++) {
  toggle[i].addEventListener('click', toggleContent,
```

# UI Developer

Web Advanced—Irvine CA, 92618

\$70,000–89,000 (DOE) a year Full-time,

**Apply Now**

## Required

- Expert in Adobe XD and Photoshop
- Expert in at least one wireframing/prototyping software such as InVision
- 4 years experience in providing UX across large-scale websites
- Excellent writing and communication skills
- Conceptual understanding of the technical implications of UX/UI decisions
- Proficient in Google Analytics

## Desired

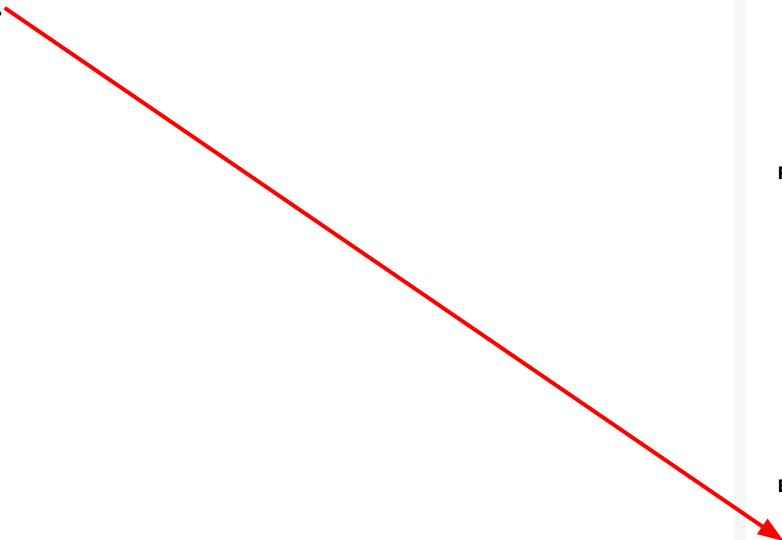
- Expertise with Git + GitHub Version control.
- Experience with tools such as HotJar, UserTesting.co, Visual Website Optimizer
- Experience working with clients on a variety of projects
- Experience as part of a team that including front and back end developers
- Proficiency in design review softwares such as Zeplin, and project management softwares such as Jira.
- Some HTML/CSS knowledge/coding ability

**Development  
Software  
Experience**

# Why Learn GitHub?

Let's look at a few more job postings.

This is a UX/UI designer position in Santa Monica. It's a big plus to know GitHub.



## UX/UI Designer

StackCommerce - Santa Monica, CA

Apply Now



- Passionate. You live and breathe design. You regularly stay up to date on current design trends and enjoy learning new skills as a designer. You have a passion for the user's experience and the ability to execute that with meaningful and elegant design. Love what you do and have fun in the process.
- Analytical. You have deep knowledge of user-centered design and usability best practices. You understand and respond to performance data (metric tests, usability studies), and steer the direction of future creative designs. You are able to backup your design decisions with meaningful data or reasoning.
- Experienced. Experience in managing workload against resources and competing priorities, successfully working with leadership and cross-functional teams. You have a great UX/UI portfolio, demonstrating outstanding design, execution, and problem solving skills.
- Collaborative. Have conviction to champion your perspective but able to successfully collaborate with other points of view. Effortlessly collaborate with a multidisciplinary team to provide concept visualizations, prototypes, specifications, and ultimately high-resolution assets for beautiful user interfaces in a fast-paced environment.

### Requirements:

- Experience designing for large-scale ecommerce or SaaS platform
- An online portfolio demonstrating conceptual thinking and creative problem solving
- Ability to translate wireframes into outstanding visual/UI design
- Advanced understanding of CSS, HTML5
- BA/BS degree in Design or similar background or equivalent practical experience
- Minimum of 5 years of experience in UX/UI design
- Experience managing multiple projects and the ability to meet aggressive deadlines
- Ability to work cross-team and synthesize feedback from multiple teams/owners
- Proficiency with design tools such as Adobe Creative Suite, Sketch, Invision
- Experience designing for responsive platforms

### Big Plus:

- Ability to prototype interactions/designs
- Experience using analytical tools such as Google Analytics and/or Looker
- Experience using GitHub Issues

# Why Learn GitHub?

Here is a senior UX designer role.  
Once again, they specifically ask for  
knowledge of HTML, CSS, and use  
of GitHub.

Many UX jobs are for websites,  
making understanding the tools  
essential to excelling at the job.

## Senior UX Designer

Sumo ★★★★★ 13 reviews - Austin, TX

Apply Now



- Conducting user research, testing usability, and explaining your findings
- Deciding what level of prototyping a new feature requires
- Collaborating with the Sumo product team to understand customer and company goals
- Creating wireframes, storyboards, sitemaps, screen flows, and prototypes
- Synthesizing and presenting your findings
- Working with our UI team to explain what you've built and designed
- Exporting and annotating your design to engineers
- Continuously pushing design projects to the next phase
- Participating in brainstorming sessions and providing feedback on ideas

### Role Requirements:

- 3+ years of experience
- Most recent portfolio demonstrating both design skill and process
- Experience in creating easy-to-use designs that display holistic thinking
- Amazing user research experience
- Acute attention to detail
- Experience with and proficiency in design and prototyping software (Sketch, Invision, Adobe...etc.)
- Stellar presentation skills
- Strong communication skills
- Must love dogs and tacos!

### Software Experience:

- Sketch software
- Prototyping & wireframing software
- HTML
- CSS
- Github

# Git: Client Graphical Software

Git Client Software options:

A Git Client GUI that offers a visual representation of your repositories.

UX/UI Boot Camp will use:



Fork

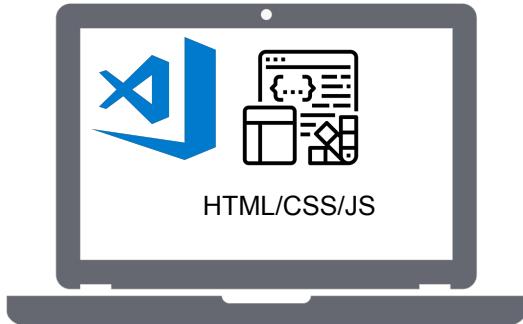


GitHub Desktop

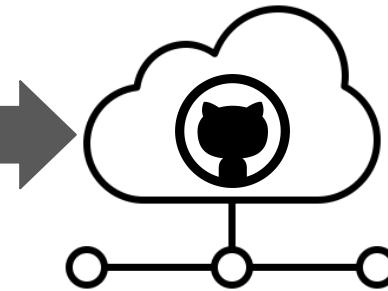
# Code Environment

---

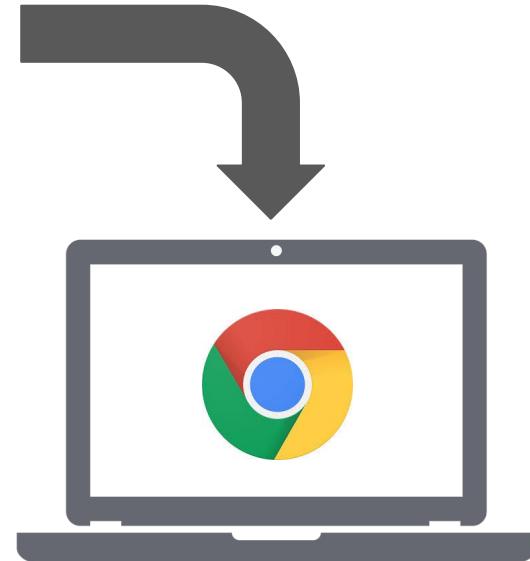
Save code  
versions via  
Git Client  
(GitHub Desktop)  
to GitHub



Code in Visual Studio  
Code  
(Local)



GitHub repo hosts  
webpage via  
GitHub Pages



Review/test via Chrome  
developer tools  
(Online)

# GitHub Desktop Functions

# GitHub Desktop Functions

## New Repository:

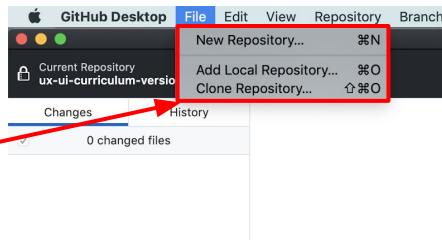
Button creates a new repo on your desktop.

## Add Local Repository:

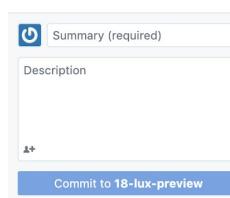
Adds an already existing git repo to GitHub Desktop.

## Git Clone:

Clones a git repository onto the local file system.



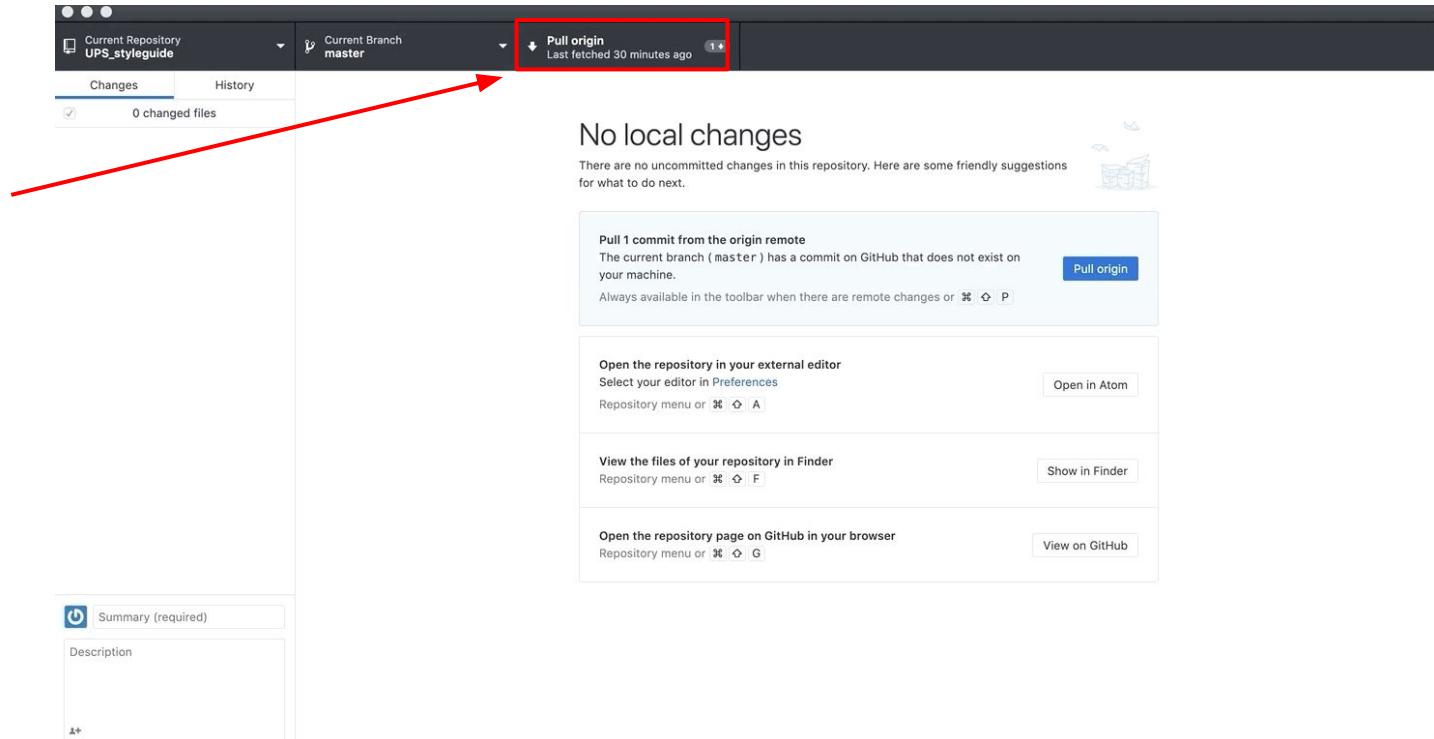
The screenshot shows the GitHub Desktop application interface. The 'File' menu is open, displaying three options: 'New Repository...', 'Add Local Repository...', and 'Clone Repository...'. A red arrow points from the text 'Clone Repository...' to the 'Clone Repository...' option in the menu. To the right of the menu, there is a main workspace area with the title 'Current Repository: ux-ui-curriculum-version'. Below the title, it says 'Changes' and 'History', with '0 changed files'. On the right side of the workspace, there is a sidebar with several sections: 'No local changes' (with a note about no uncommitted changes), 'Publish your branch' (with a 'Publish branch' button), 'Open the repository in your external editor' (with an 'Open in Atom' button), 'View the files of your repository in Finder' (with a 'Show in Finder' button), and 'Open the repository page on GitHub in your browser' (with a 'View on GitHub' button). There is also a small illustration of a person working at a desk.



# GitHub Desktop Functions

## Git Pull:

Used to update your local repository with changes to local branches.

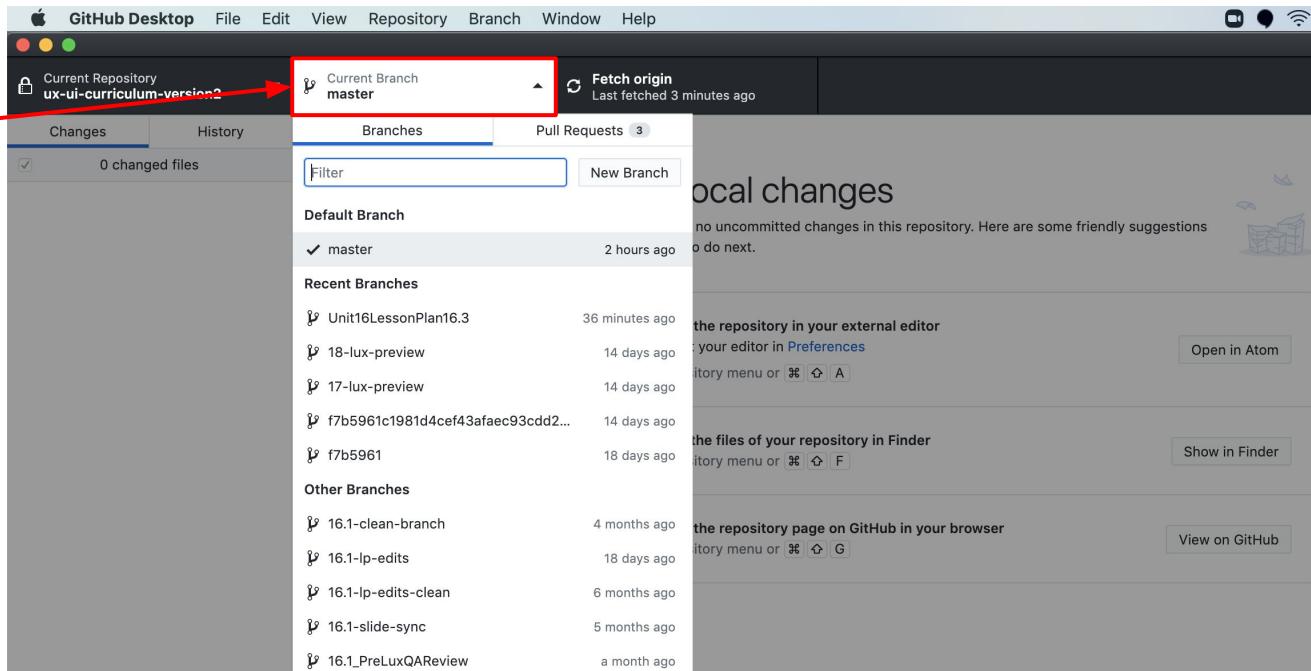


# GitHub Desktop Functions

## Branches:

You can create and manage branches via GitHub Desktop.

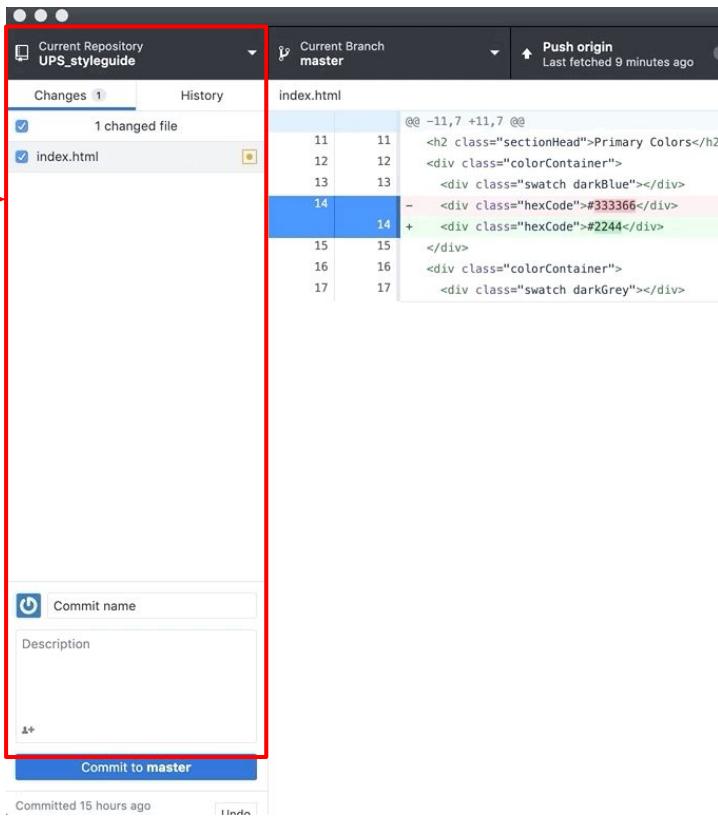
Branches are local copies of repositories that allow you to work separately from the main repository.



# GitHub Desktop Functions

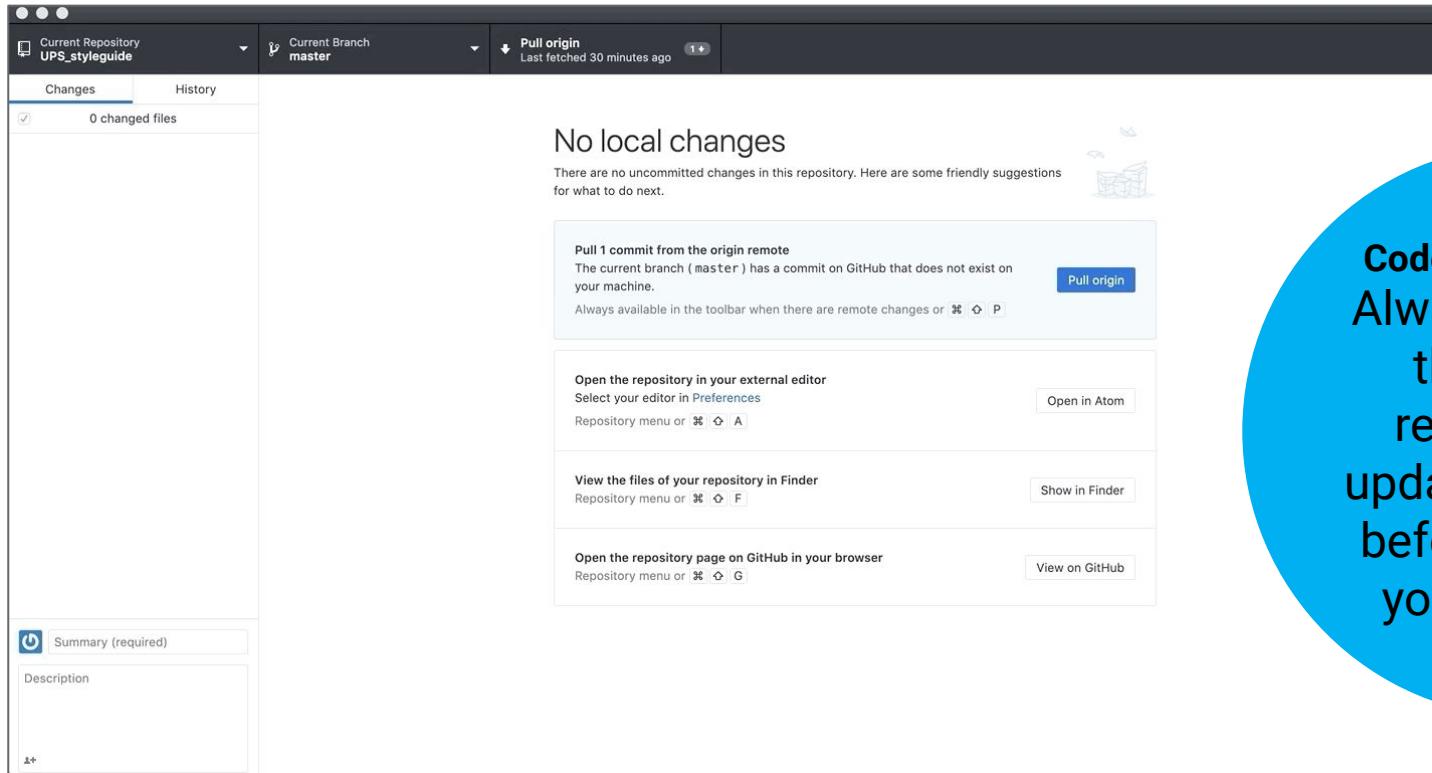
## Git Push:

When changes are made to your local Git repository, GitHub Desktop tracks your changes and allows you to update the cloud repository.



# Common GitHub Desktop Workflows

# Workflow 1: Pull From a Repository



**Code Best Practice:**  
**Always pull from**  
**the remote**  
**repository to**  
**update your code**  
**before you start**  
**your workday.**

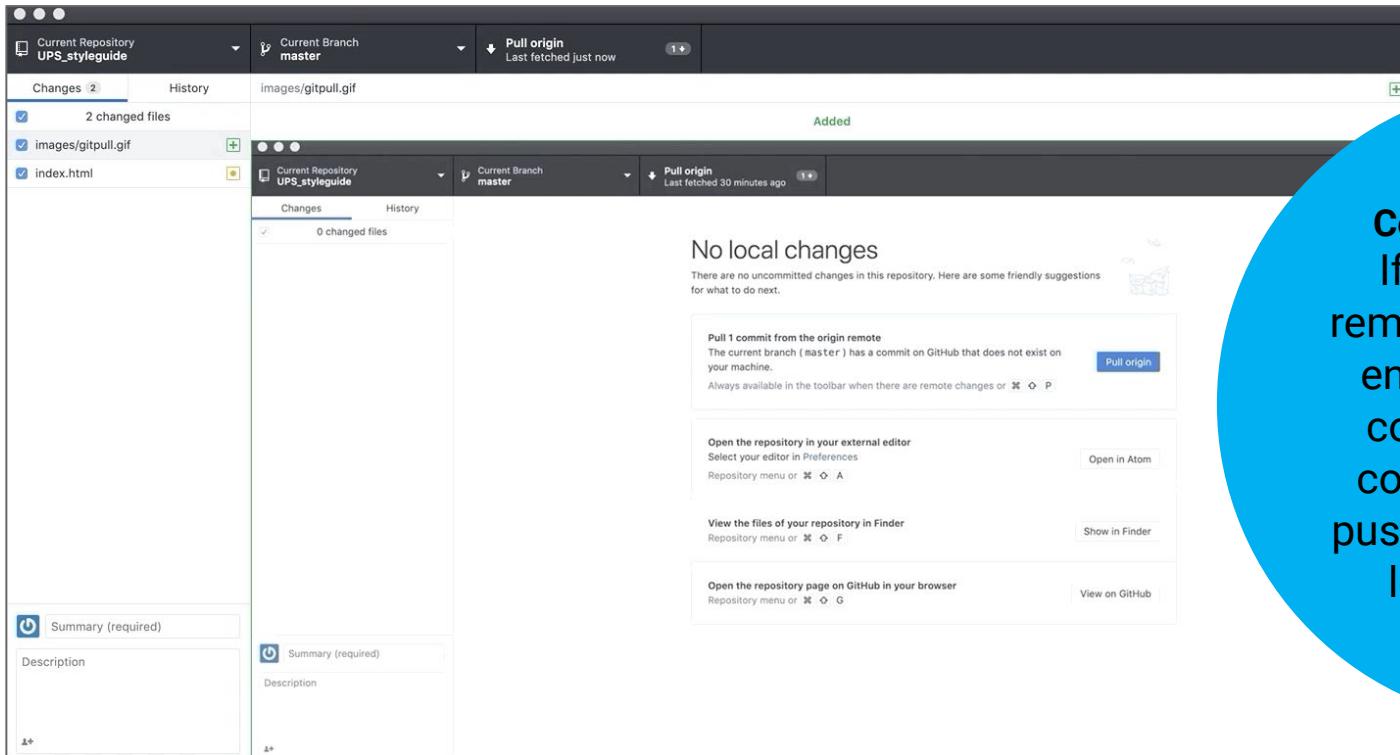
# Workflow 2: Push Local Changes

The screenshot shows the GitHub Desktop application interface. At the top, there are three dropdown menus: 'Current Repository' set to 'UPS\_styleguide', 'Current Branch' set to 'master', and 'Push origin' with a note 'Last fetched 9 minutes ago'. Below these are two tabs: 'Changes' (selected) and 'History'. Under 'Changes', it says '1 changed file' and lists 'index.html'. The main area shows a diff of 'index.html' with line numbers 11-17. Line 14 is highlighted with a blue selection bar, showing a deletion of a hex code and an addition of another. At the bottom left, there's a 'Commit name' field with a placeholder 'Commit name', a 'Description' text area, and a 'Commit to master' button. A status bar at the bottom indicates 'Committed 15 hours ago' and has an 'Undo' button.

## Code Best Practice:

1. Make changes to a document in your repo and then add it as a commit.
2. Push the commit up to your repository to update it.

# Workflow 3: Resolve Merge Conflicts



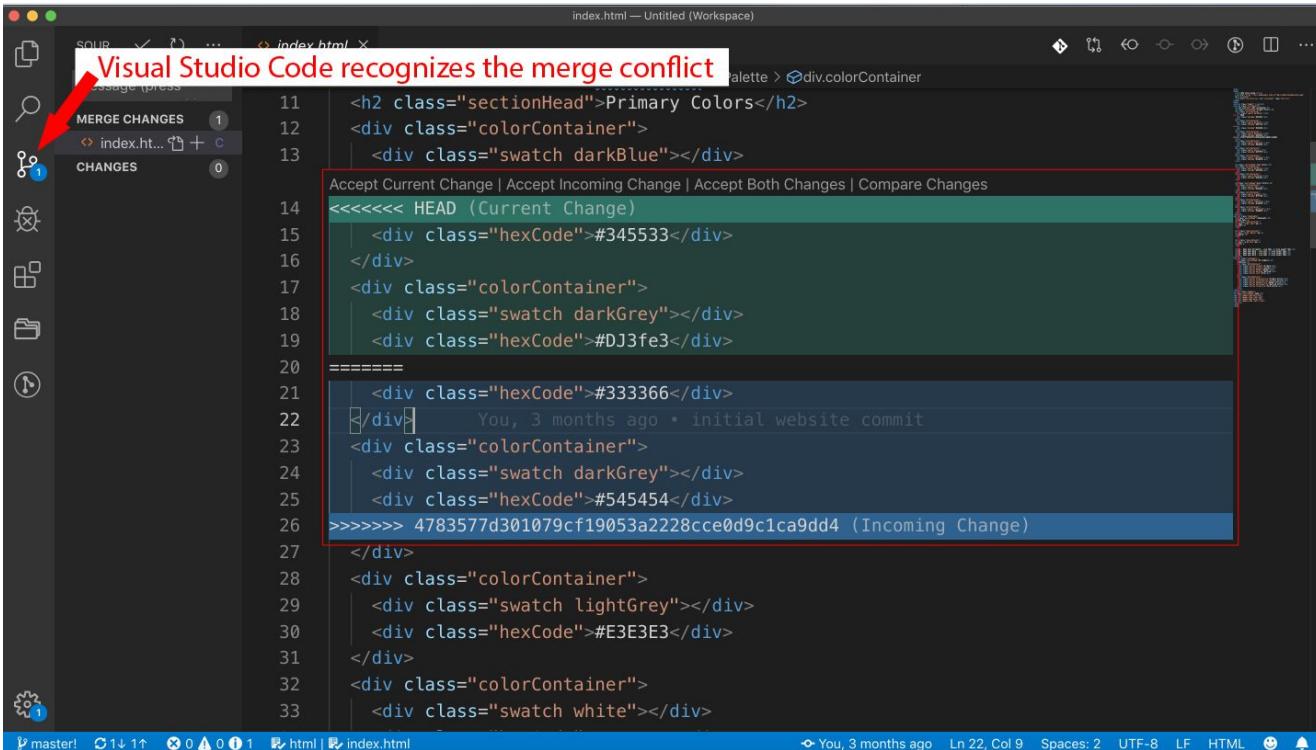
**Code Best Practice:**  
If you pull from a  
remote repository and  
encounter a merge  
conflict, select the  
correct version and  
push the merge to the  
local repository.

# Workflow 3.1: Resolve Merge Conflicts

The next step is to open the conflicting file in Visual Studio Code.

Make sure you have this plugin installed for Visual Studio Code:

[Visual Studio Code Git Package](#)



Visual Studio Code recognizes the merge conflict

```
index.html — Untitled (Workspace)
```

MERGE CHANGES 1

CHANGES 0

```
<h2 class="sectionHead">Primary Colors</h2>
<div class="colorContainer">
  <div class="swatch darkBlue"></div>
<div class="hexCode">#345533</div>
</div>
<div class="colorContainer">
  <div class="swatch darkGrey"></div>
  <div class="hexCode">#DJ3fe3</div>
<div class="hexCode">#333366</div>
</div>
You, 3 months ago * initial website commit
<div class="colorContainer">
  <div class="swatch darkGrey"></div>
  <div class="hexCode">#545454</div>
<div class="hexCode">4783577d301079cf19053a2228cce0d9c1ca9dd4 (Incoming Change)</div>
</div>
<div class="colorContainer">
  <div class="swatch lightGrey"></div>
  <div class="hexCode">#E3E3E3</div>
</div>
<div class="colorContainer">
  <div class="swatch white"></div>
```

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

=====

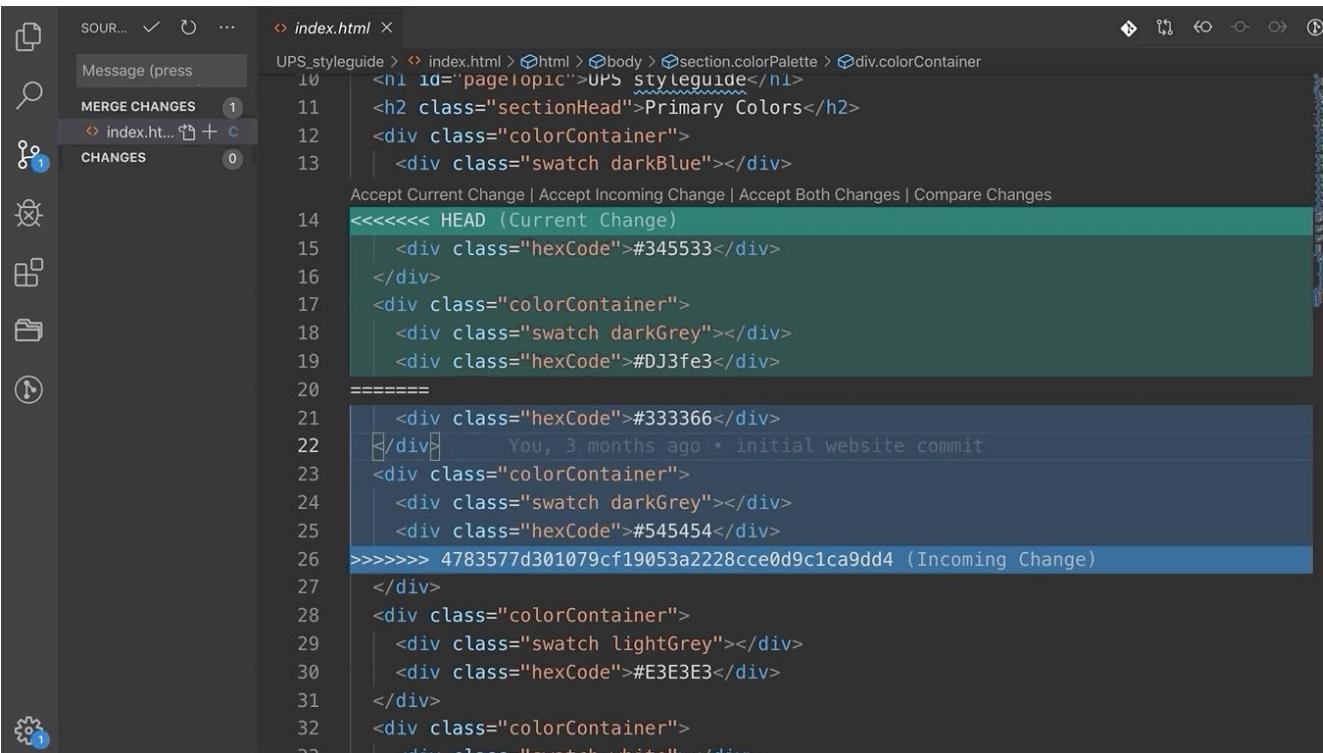
=====  
You, 3 months ago \* initial website commit  
=====

=====  
4783577d301079cf19053a2228cce0d9c1ca9dd4 (Incoming Change)  
=====

master 1 11 0 0 1 1 html index.html You, 3 months ago Ln 22, Col 9 Spaces: 2 UTF-8 LF HTML

# Workflow 3.2: Resolve Merge Conflicts

Select which change is the most current and up-to-date by clicking accept current change or accept incoming change.



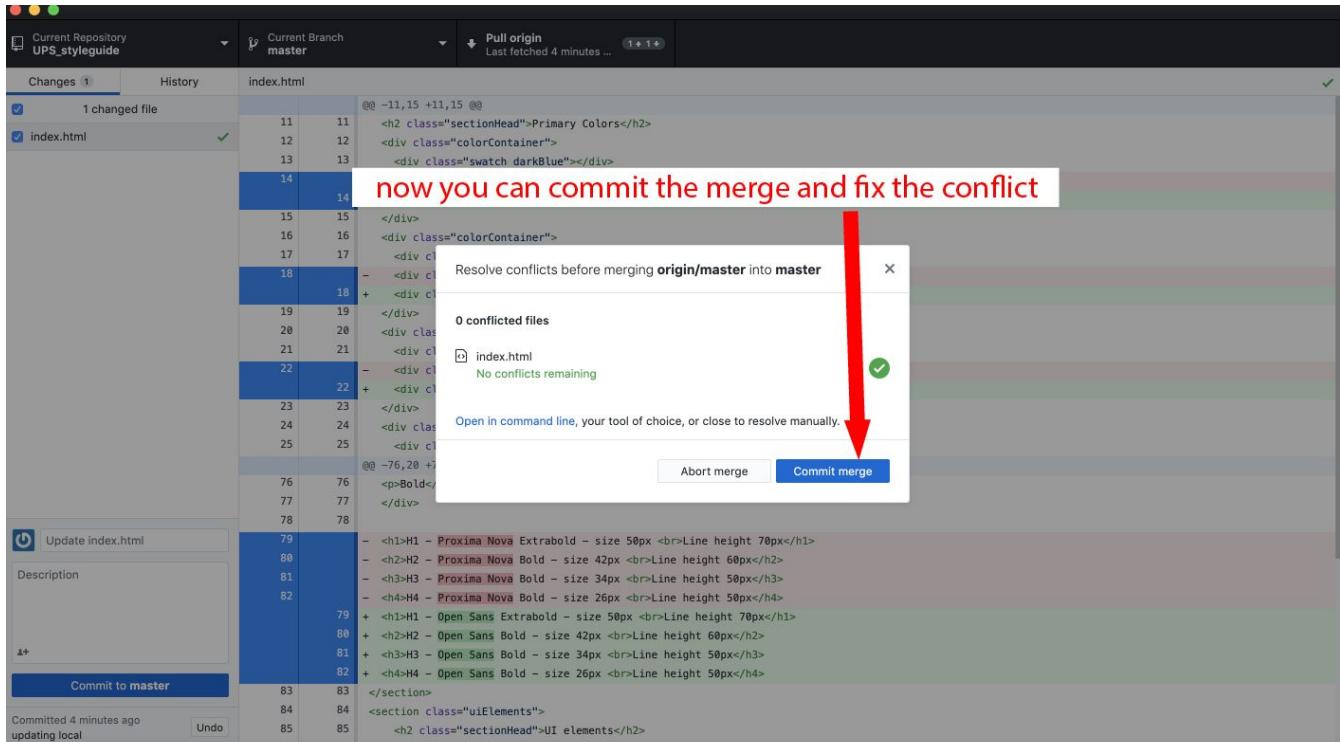
The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with icons for file operations like copy, paste, search, and refresh. The main area is titled "index.html" and displays the following code:

```
<n1 id="pageTitle">UPS Styleguide</n1>
<h2 class="sectionHead">Primary Colors</h2>
<div class="colorContainer">
    <div class="swatch darkBlue"></div>
    <div class="hexCode">#345533</div>
    <div class="colorContainer">
        <div class="swatch darkGrey"></div>
        <div class="hexCode">#D3fe3</div>
    </div>
    <div class="hexCode">#333366</div>
    </div> You, 3 months ago * initial website commit
    <div class="colorContainer">
        <div class="swatch darkGrey"></div>
        <div class="hexCode">#545454</div>
    </div>
    <div class="hexCode">#4783577d301079cf19053a2228cce0d9c1ca9dd4 (Incoming Change)
    </div>
    <div class="colorContainer">
        <div class="swatch lightGrey"></div>
        <div class="hexCode">#E3E3E3</div>
    </div>
    <div class="colorContainer">
        <div class="swatch white"></div>
    </div>
</div>
```

The code editor highlights specific sections with different colors: the first section from line 14 to line 20 is highlighted in green, and the second section from line 21 to line 33 is highlighted in blue. A tooltip at the bottom of the green-highlighted area reads "Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes". The status bar at the bottom of the editor shows the commit message "You, 3 months ago \* initial website commit".

# Workflow 3.3: Resolve Merge Conflicts

After all the conflicting files have been fixed, we can merge our code and push it back to the repo, all in GitHub Desktop.



# Hosting a Website With GitHub Pages



**GitHub Pages** is a static site-hosting service designed to host your personal, organization, or project pages directly from a GitHub repository.

# GitHub Pages

A GitHub Page will host your web content for turning in assignments.

The screenshot shows a web browser window with the URL <https://github.build.ge.com/212574846/GitHub-Pages/>. The page title is "GitHub Pages". The main content area says "GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository." Below this, there is a "Source" section with a "None" dropdown menu open. The "master branch" option is selected, highlighted in blue, and the sub-option "Use the master branch for GitHub Pages." is visible. Other options shown are "master branch /docs folder" and "None". A "Save" button is located next to the dropdown. At the bottom, there is a note about publishing manually via a gh-pages branch.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

**Source**

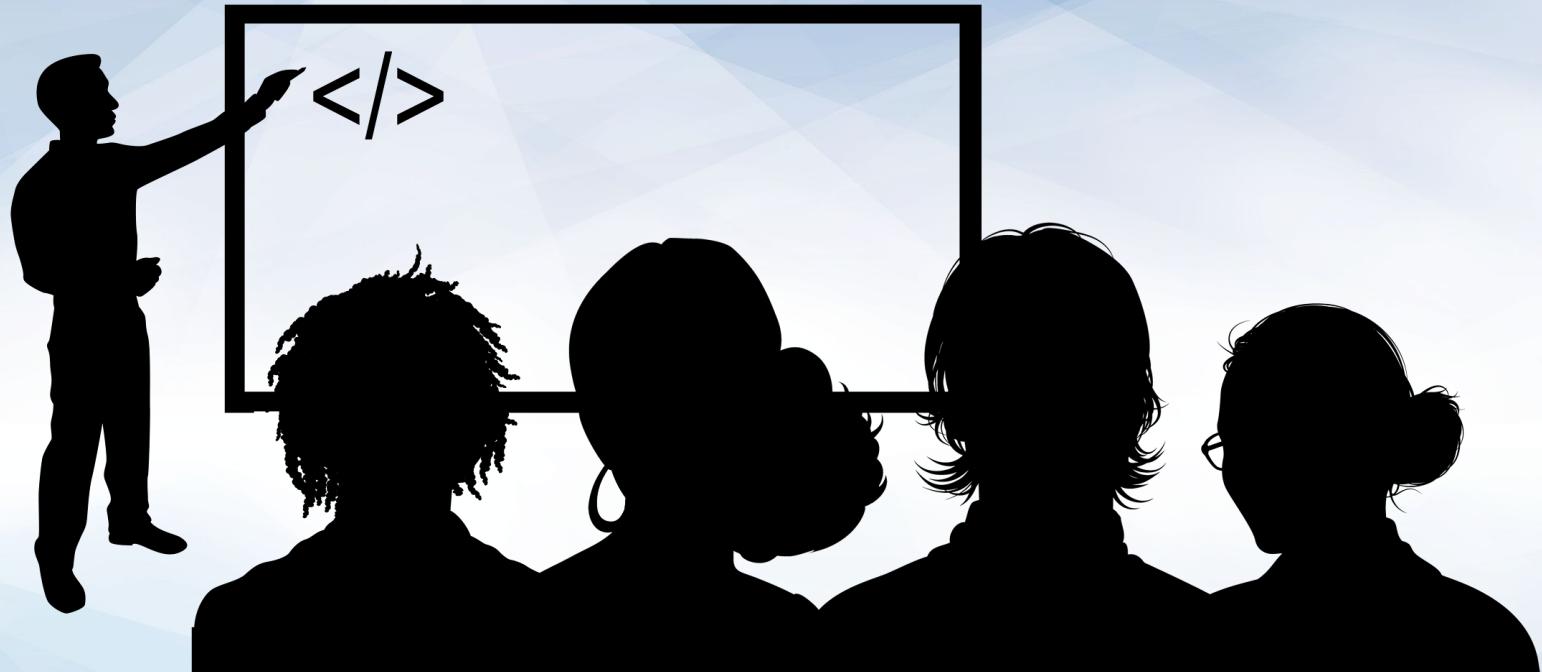
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more](#).

**None**

Select source

- master branch**  
Use the master branch for GitHub Pages.
- master branch /docs folder  
Use only the /docs folder for GitHub Pages.
- None**  
Disable GitHub Pages.

To publish a page manually, push an HTML or [Jekyll](#) site to a branch named gh-pages. Read the [Pages help article](#) for more information.



Instructor Demonstration  
GitHub Pages



# **Challenge:**

## Upload Your First Webpage to the GitHub Cloud!

(Instructions sent via Slack)

**Suggested Time:**  
20 minutes





Time's Up! Let's Review.

# Congratulations! Recap

---

Today we learned:

01

## HTML Positioning of Elements

How to code html markup for webpage design.



02

## HTML Structure and CSS Classes and IDs

How to code html and CSS together for element positioning.



03

## GitHub and GitHub Pages

How to push pages to a version control cloud and publish webpages.





# Questions?

*The  
End*



**30 minutes**