# Intro to HTML and CSS

by Emile Jean Baptiste

## General Overview

This unit is an introduction to HTML and CSS. This unit was designed to be part of a class that covers multiple aspects of Web Development, including JavaScript and SQL, so it does not go too in depth into HTML or CSS as a class focused on those languages might. This class was originally meant to be maybe a senior level class, but I tried to make it a bit more accessible to students of different CS experience, although I still envision this being a class students take after an intro CS class. This unit in particular was designed with project-based learning in mind, with students having a small project made in most lessons while also building towards the goal of a larger-scale project of their own machinations.

## Motivation for Unit

The thing that I love about Computer Science the most is its versatility and potential for creativity. Although I was never a professional, web development is one of the aspects of computer science I like doing the most and one that I think has a lot of potential for intersecting with other fields. One of the central philosophies I has in mind while developing this unit was that I really wanted to not only show off various different applications of what they'd be learning, HTML and CSS, but I also wanted to give students a chance at every opportunity to really make the work in this class their own and be something that matters to them, even if it's just because they got to use an image they liked once. I also wanted this to be a class where students felt like they had a bit of control of the direction the class takes and what they learn from it. The class has a relatively consistent structure of watch me and the slides for a bit, and then do something (hopefully) fun. While I realize in retrospect that there may not be enough time to finish assignments by the end of each class, I also made it so that these assignments were the only HW as to hopefully not be too stressful of a class. I focused on HTML and CSS as a unit because I thought it would be easier to work with what would likely be the first unit in the curriculum, and I wouldn't have to worry too much about thinking what would students theoretically know in the middle of the year.

# Standards Referenced

- 9-12.DL.7
- 9-12.NSD.4
- 9-12.IC.7
- 9-12.DL.5
- 9-12.IC.6
- 9-12.DL.4

# Tools Used

- Google Classroom
- GitHub Classroom
- HTML
- CSS
- Google Fonts
- Computer
- Internet

# Resources

In addition to presenting the slides for each lesson in class, students will also have access to the slides after each class as a resource. Additionally resources used include

- https://www.youtube.com/watch?v=Gn3w1UvTx0A
- https://www.youtube.com/watch?v=U1vCFZmtYUA
- https://www.youtube.com/watch?v=DqjPr7auwdY
- https://www.youtube.com/watch?v=3Zciwq2H7BI
- https://www.thewikipediagame.com/
- https://www.w3schools.com/tags/ref_colornames.asp
- https://www.w3schools.com/colors/colors_hexadecimal.asp
- https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html
- https://www.learnui.design/tools/accessible-color-generator.html
- http://colorsafe.co/
- https://imsdb.com/
- https://www.w3schools.com/cssref/css_websafe_fonts.php
- https://fonts.google.com
- https://www.youtube.com/watch?v=46m_4P492P4
- https://www.w3schools.com/css/css_positioning.asp
- https://coursework.vschool.io/css-layout-and-positioning/
- https://www.forgotten-adventures.net/live-gallery/
- https://www.spriters-resource.com/
- https://blackgirlbytes.dev/
- https://www.nikitaghodke.com/resources
- https://emilejb.github.io/
- https://emilejb.github.io/compass/

# Lessons

Total length: 2.4 Weeks (12 days)

Lesson 00: History of the Web and How the Web Works

Lesson 01: How to use GitHub

Lesson 02: HTML Syntax and Structure of HTML Documents | Activity: Simple About Me Page

Lesson 03: File Paths and the  tag | Activity: My Favorite Things Page

Lesson 04: The tag and Connecting Multiple Webpages | Activity: Trivia/Mini-Quiz Project (Pt. 1)

Lesson 05: CSS Syntax and Selectors | Activity: Trivia/Mini-Quiz Project (Pt. 2)

Lesson 06: Text Styling Using CSS | Activity: Stylizing Dialogue/Scripts

Lesson 07: Positioning and Resizing Using CSS | Activity: Web "Diorama"

Lesson 08: Containers in HTML | Activity: Making a Comic

Lesson 09: Introduction to Personal Website Project | Activity: Working on Personal Website Project

Lesson 10: Working on Personal Website Project (Day 2)

Lesson 11: Working on Personal Website Project (Day 3)

# Assesments

## Formative Assessments

- Various assignments at the end of lessons 02-08 with specific requirements for the students to meet
- In-Class check-ins while students work on their assignments after the presentations

## Summative Assessments

- Part 1 of student's personal website project

# Lesson 00: History of the Web and How the Web Works

# Lesson Structure

## Standards

- 9-12.DL.7
- 9-12.NSD.4
- 9-12.IC.7

## Aim: What is the web?

## Warm-Up (2-3 minutes)

Do Now: What comes to mind when you think about the World Wide Web? What do you use the web for?

Students will be given 2-3 minutes to answer these questions in their notebook (or something equivalent) and potentially discuss their answers with their classmates next to them.

## Warm-Up Review (3 - 5 minutes)

After students are given some time to think of their answers to the Do Now, we'll turn this into a classwide discussion, taking time to hear students' answers to the first question of the Do Now, then the second. It may also make sense to write down their answers on the board to keep in mind throughout the class period.

# Slide Presentation (30-35 minutes)

Present the slides provided in the resources/lesson00/lesson00_slides.pdf

As this lesson in particular is more lecture style, presenting the slides will take up a majority of the time. That being said, there will be several prompts for discussions and asking students questions throughout, such as discussing the difference between "the web" vs the internet or quickly surveying what browsers students use, so that it is more interactive.

If students have access to computers during this lesson, for part of the presentation students may also try to inspect element so that they can see the HTML code underlying websites themselves.

# Exit Ticket / Homework (Rest of Class)

Browse the web and think about

- What’s something you’d want to learn how to do with web development
- What’s something you’d want to know about the web itself Submit your thoughts to Google Classroom.

# Aim: What is the Web?

Do Now: What comes to mind when you think about the World Wide Web? What do you use the web for?

# A Common Misconception: The Internet vs The Web

The Internet

- A network of networks
- Invented in 1983
- Can be considered a form of hardware

The Web

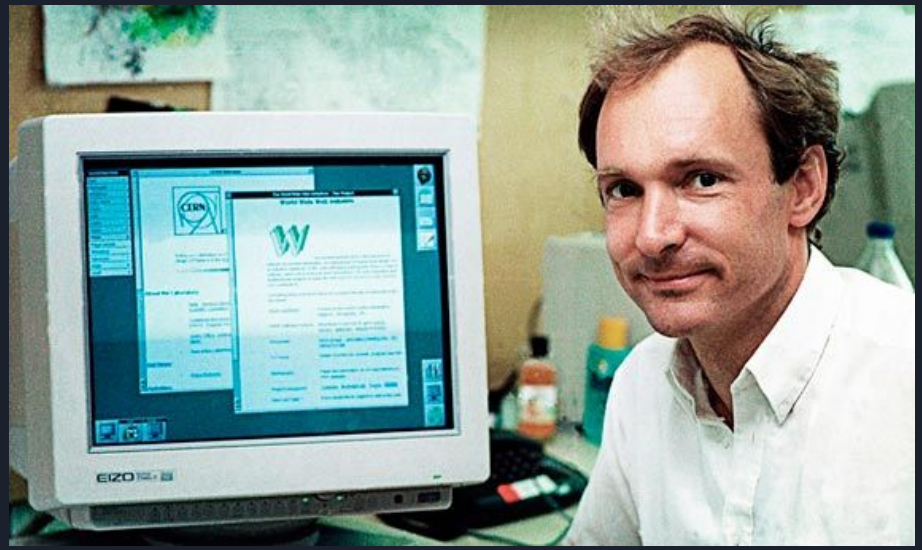
- A network of webpages
- Invented in 1989
- Can be considered a form of software

In other words, the web can be considered a way to use the internet, but it is only one use of the internet.

What other uses of the internet besides the web can you think of?

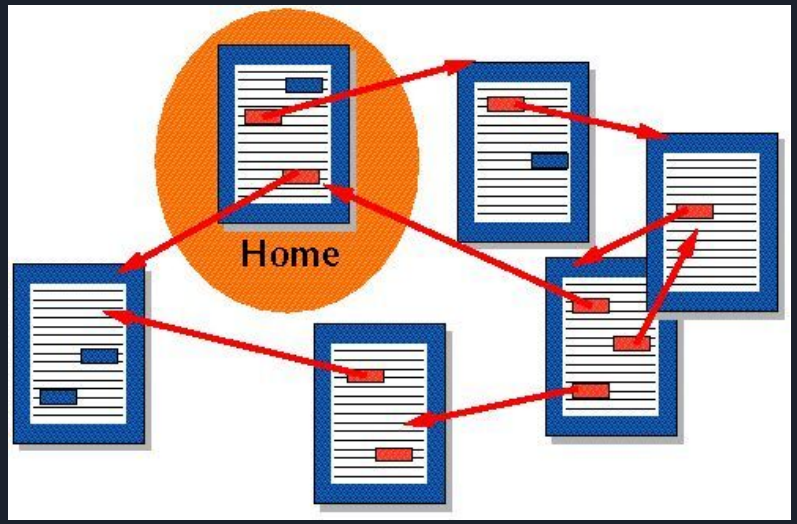
# The Creation of the World Wide Web

- The web is officially known as the World Wide Web
- The World Wide Web was created in 1989 by Tim Berners Lee, a British Computer Scientist, at CERN
- Tim Berners Lee goal for creating the web was to make sharing information between scientists at CERN easier
- Eventually, the internet became available for public use in the early 1990s and spread throughout the world after
- In 1993, Tim Berners Lee and other computer scientists ensured that the World Wide Web would be free to use forever



# How the Web Works: Hypertext

- Hypertext is one of the key technologies that make the web possible.
- Hypertext is text that contains references to other text that can be immediately accessed
- Example: https://en.wikipedia.org/wiki/Computer_science (This link itself is also an example of Hypertext)
- The World Wide Web essentially is a complex web of hypertext documents



# How the Web Works: URL

- URL stands for Uniform Resource Locator
- URLs are unique references that lead to a specific resource on the web
- URLs can be thought of as the address of something on the web, hence their other name of web address
- Like real addresses though, sometimes the thing you want may move to a different address or something else may replace what's at that address
- Nowadays, if you enter something that is not a URL in the URL bar it will search for that term (often through Google), although that was not always the case

# HTTP URL Anatomy

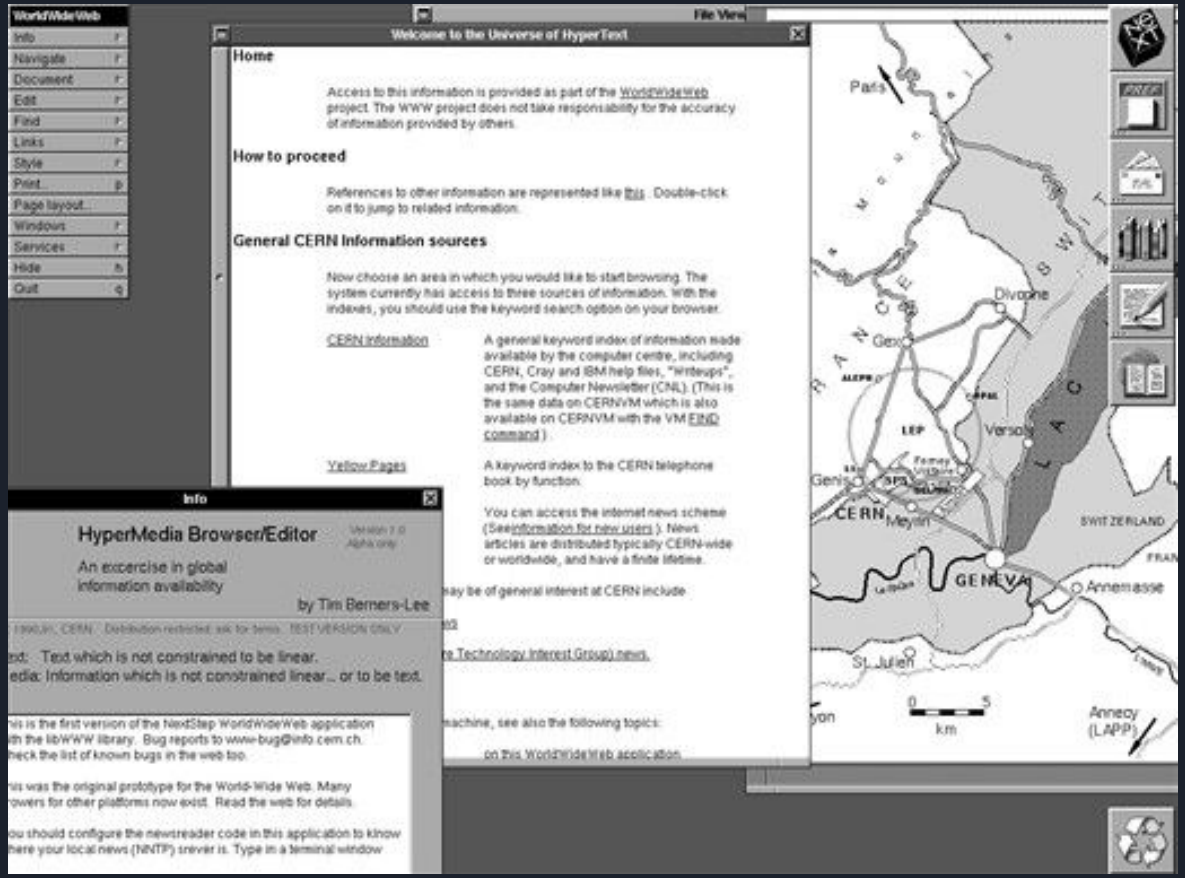
1 2 3 4 5 6 7 8

https://www.example.com:3000/path/resource?id=123#section-id

## Key

1. Scheme – defines how the resource will be obtained.
2. Subdomain – www is most common but not required.
3. Domain – unique value within its top-level domain.
4. Top-level Domain – hundreds of options now exist.
5. Port – if omitted HTTP will connect on port 80, HTTPS on 443.
6. Path – specify and perhaps find requested resource.
7. Query String – data passed to server-side software, if present.
8. Fragment Identifier – a specific place within an HTML document.

# The First Web Browser



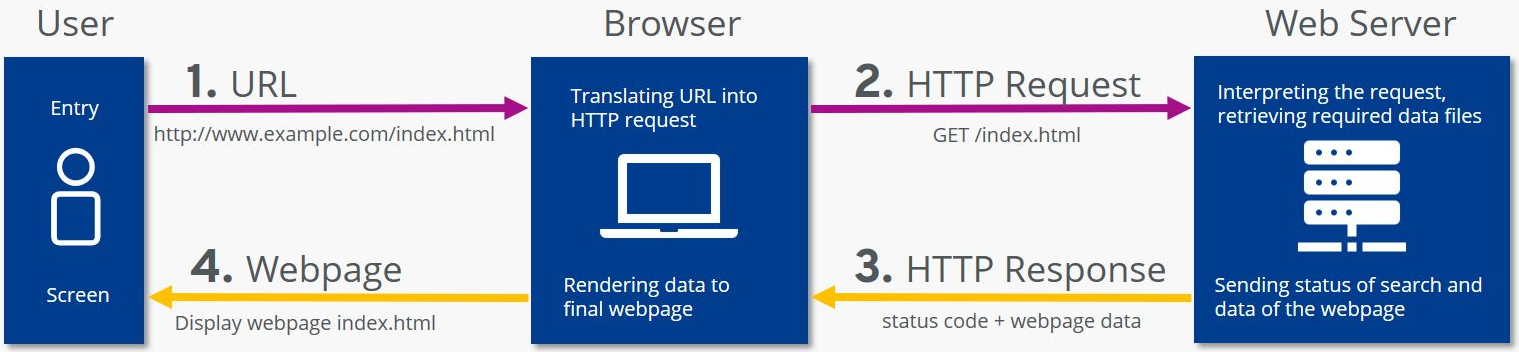
# How the Web Works: Browsers

- Browsers are the program used to actually surf the web and interact with web pages
- What are some browsers that y'all know of or have heard of?
- What browser do you use? Why do you use that browser over others?
- Nowadays, many browsers are built off of Chromium, an open-source browser project by Google
- Although all browsers are capable of surfing the web, there are differences between them
- Some web pages work better on some browsers than others because of that
- To fully understand how browsers work, we need to mention two more parts of the web

# How the Web Works: HTTP

- HTTP stands for HyperText Transfer Protocol
- HTTP is how data is exchanged on the web between clients and servers
- Whenever one accesses a web page or something else on the web, it goes through this process
    1. The browser makes a request to the server containing the data we want
    2. The server receives and processes the request from the browser
    3. The server retrieves the desired data and sends it back to the browser
    4. The client receives the data from the server
    5. The browser displays that data as a web page
- Sometimes we don't always get the data we request on the web, but we'll go into more detail about that and HTTP overall in a future lesson

# Communication process according to HTTP



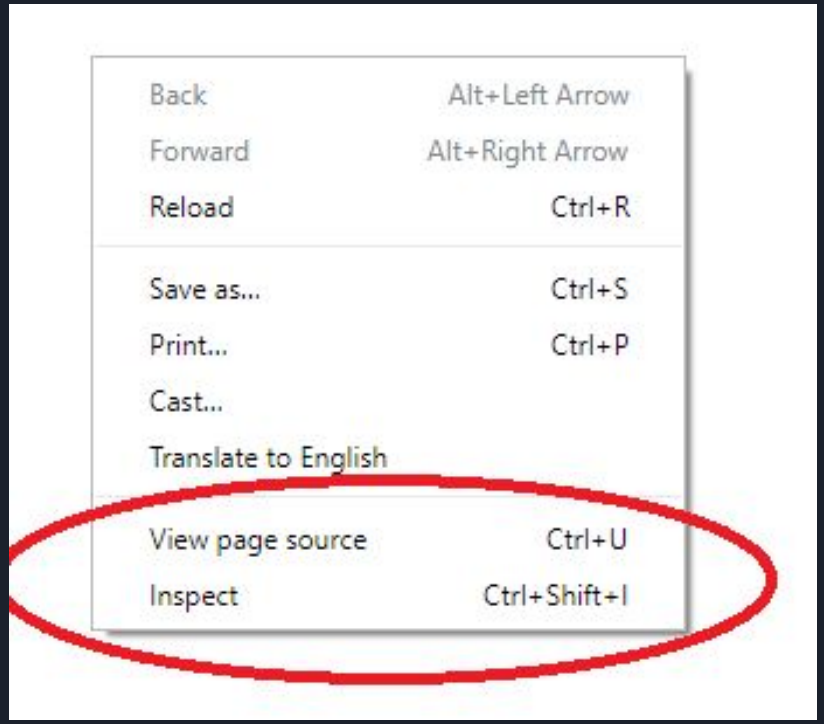

IONOS

# How the Web Works: HTML

- HTML stands for HyperText Markup Language
- HTML is the language that web pages are written in and the first language we'll be learning in this class
  - A website is a collection of web pages under a common domain.
- When we make a request to a server for a web page, we get back an HTML file written in HTML code
- The browser interprets the HTML code into the more visual web pages  that we're familiar with, but an HTML file itself just looks like text
- In fact, we can actually see the HTML code that makes up a web page by doing one of two things

# Viewing the HTML Behind the Scenes

- We can see the HTML that makes up a web page by right clicking and selecting Inspect Element or View Page Source (or by using the appropriate keyboard shortcut)
- The code you see may not make sense now, but hopefully you'll have a better understanding as time goes on



# To Be Continued...

- While we be learning more about different aspects of the web later, this about does it for the basics of the history of the web and how the web works
- The web is very complex however and entire classes could be dedicated to discussing parts of the web, such as networks, culture, history, or development
- For the remainder of class and for homework, browse the web and think about
  - What's something you'd want to learn how to do with web development
  - What's something you'd want to know about the web itself
- Submit your thoughts to Google Classroom.

# Lesson 01:
# How to use GitHub

# Lesson Structure

## Standards

- 9-12.DL.7
- 9-12.DL.5
- 9-12.NSD.4

## Aim: How do we use GitHub?

## Lesson Goals

Throughout the course, students will use GitHub to receive scaffolded code/templates, submit coding assignments, and host their projects on the web. Students may also decide to use GitHub for other purposes outside this class or once the class is over (my own GitHub was made for school). However, GitHub can be quite complex to learn at first and get adjusted to. Furthermore, it is important to be present for extra help or guidance as needed to make sure that students are ready to do the work for future lessons. Thus, this entire class period will be dedicated to trying to make sure every student has their GitHub account ready as well as a bare minimum knowledge of the features important for this class. In particular, students need to

1. Create a GitHub account if they don't have one already
2. Accept an assignment from GitHub Classroom so that they are on the roster
3. Learn how to use the GitHub Developer feature on the web (Technically optional, but potentially easier than alternatives)
4. Commit and push their changes to their repositories
5. Publish their project using GitHub Pages

# Activity

Students will follow the directions given on Google Classroom and shown on the board in class. Those directions are

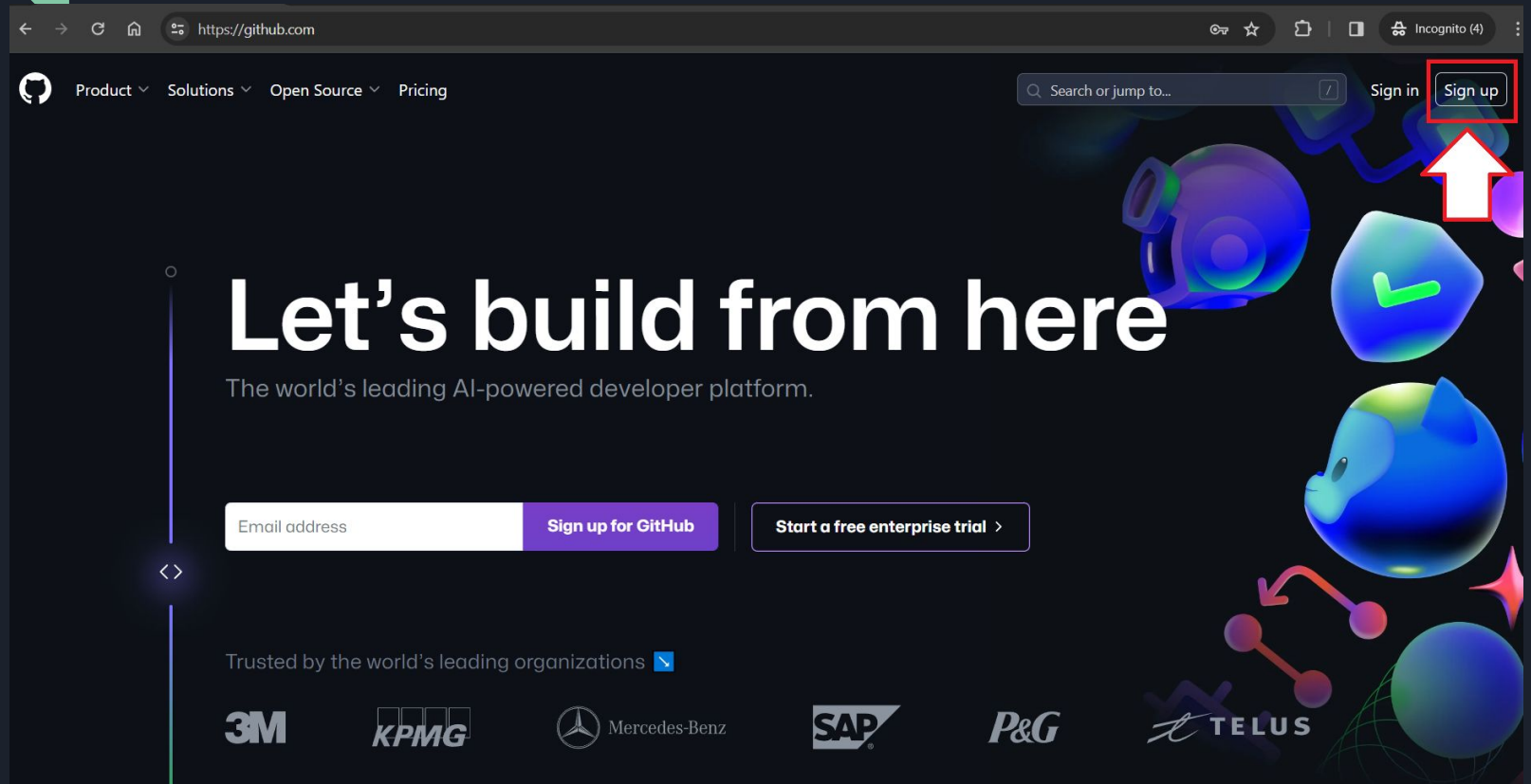
1. If you do not already have a GitHub account, follow the steps to create a new GitHub account
2. Sign into your GitHub account
3. Go to Google Classroom, navigate to the work for today, click on the GitHub classroom link and follow the prompts
4. In your repository, go to Settings -> Pages -> Deploy (from main, root folder)
5. Wait a bit, then refresh the page to see the link to your assignment on the web
6. Afterwards, go back to the code tab and press the "." key
7. Navigate to the file named index.html [provided in code/lesson01/index.html] and answer the questions in the file in the field provided
8. Go to the button on the side named Source Control and then press the Commit & Push button. Make sure to include a message.
9. Go back to the URL for your assignment (if you didn't save or copy it, you can get to the link again by going to the Page part of settings). It may take a bit to see the changes to your page
10. Once you see the changes on your web page that you made, copy the link to your webpage and post it for the Google Classroom assignment for today

# Resources

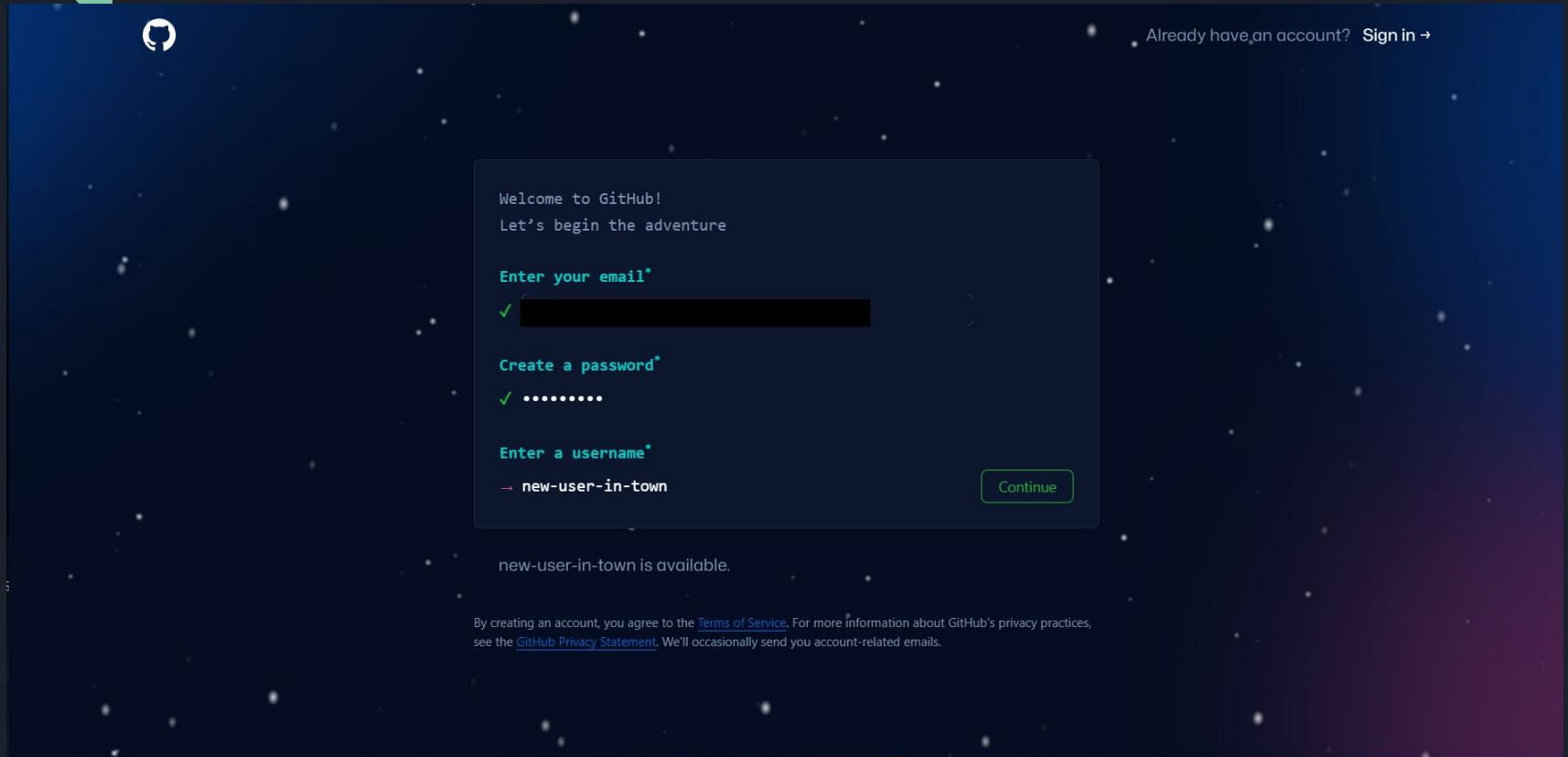
1. Slides (provided in resources/lesson01/lesson01_slides.pdf)
2. https://www.youtube.com/watch?v=Gn3w1UvTx0A
3. https://www.youtube.com/watch?v=U1vCFZmtYUA
4. https://www.youtube.com/watch?v=DqjPr7auwdY
5. https://www.youtube.com/watch?v=3Zciwq2H7BI

# Setting Up GitHub

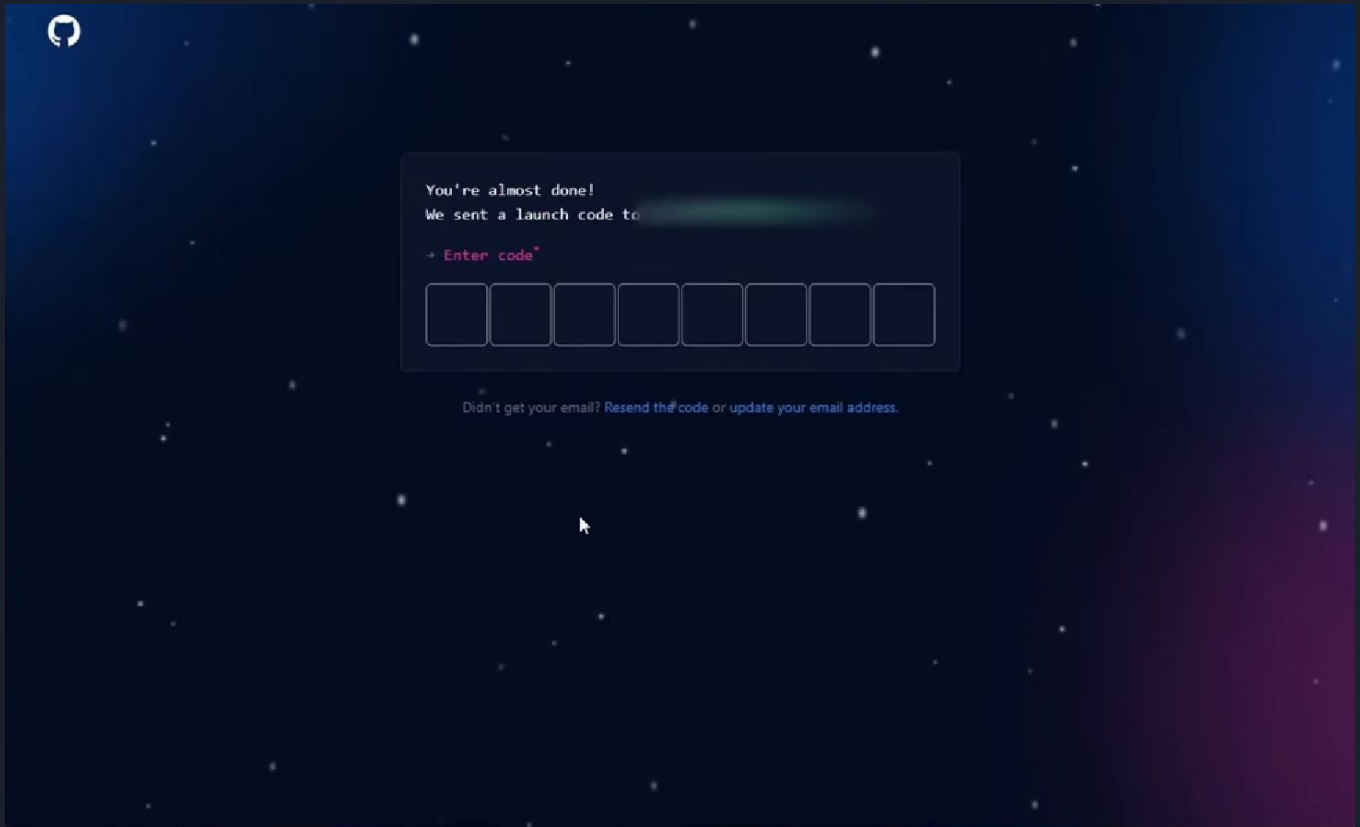
1. Go To github.com and press the Sign Up Button



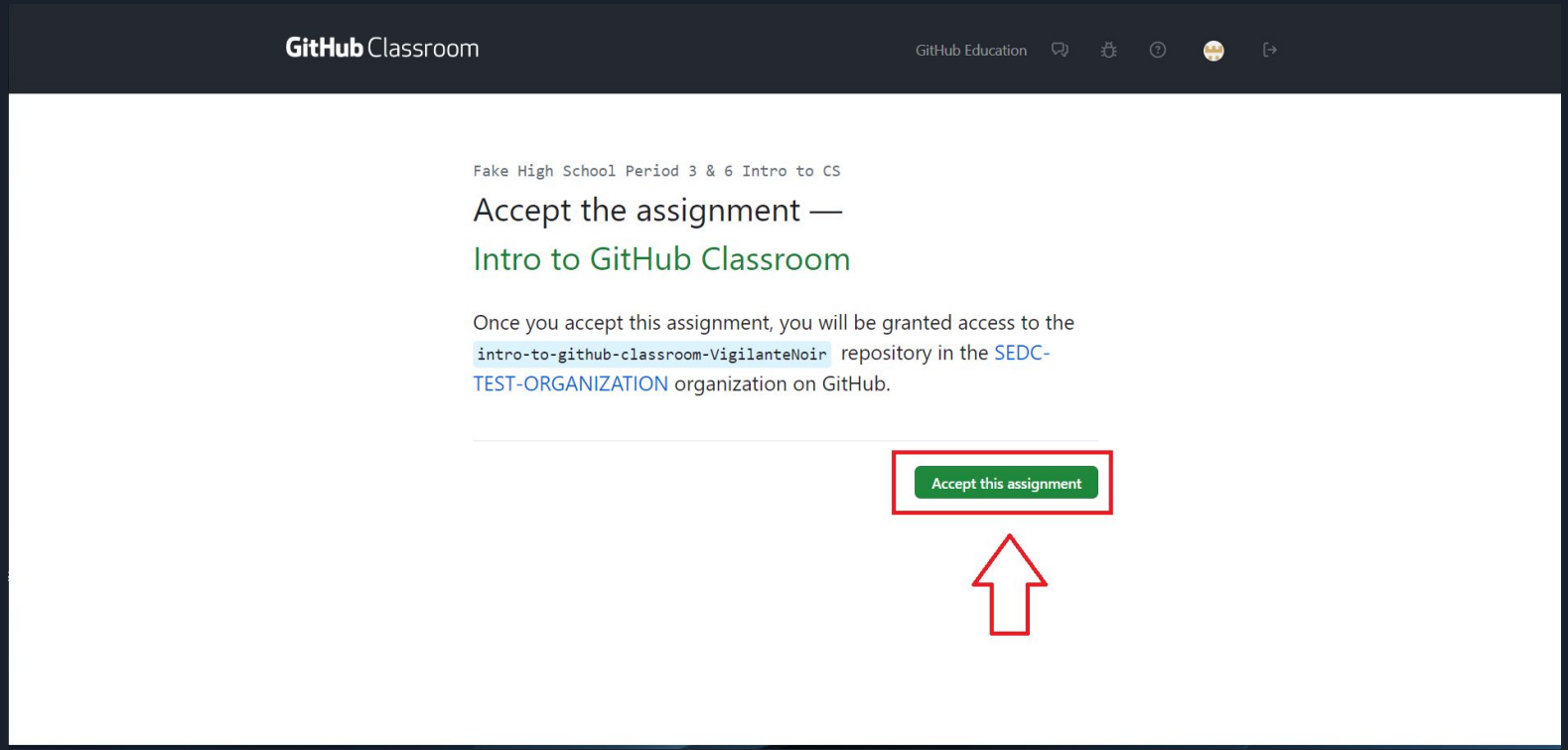
# 2. Enter the information requested by the Sign-Up Process (email, password, and username)



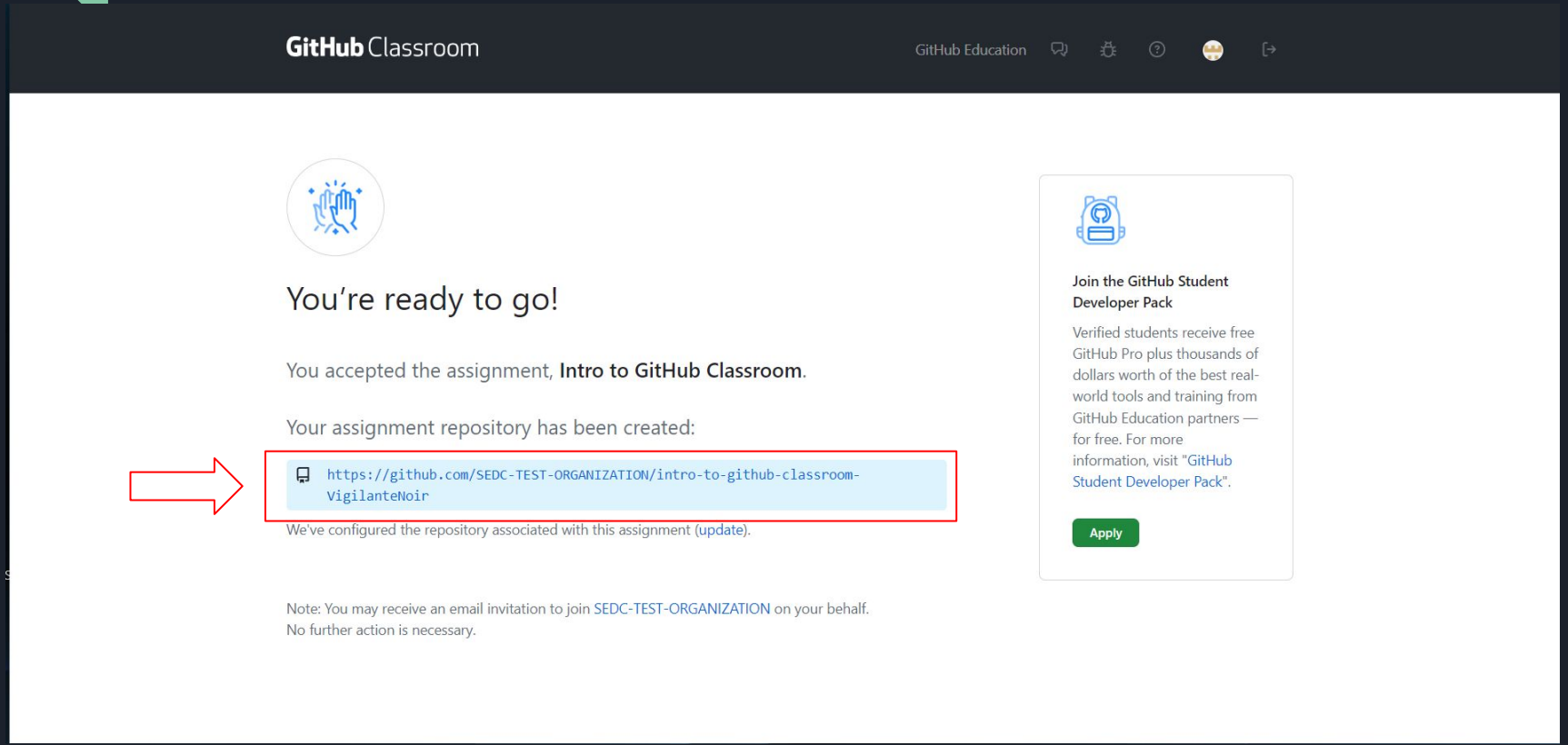
# 3. Enter the verification code sent to your email



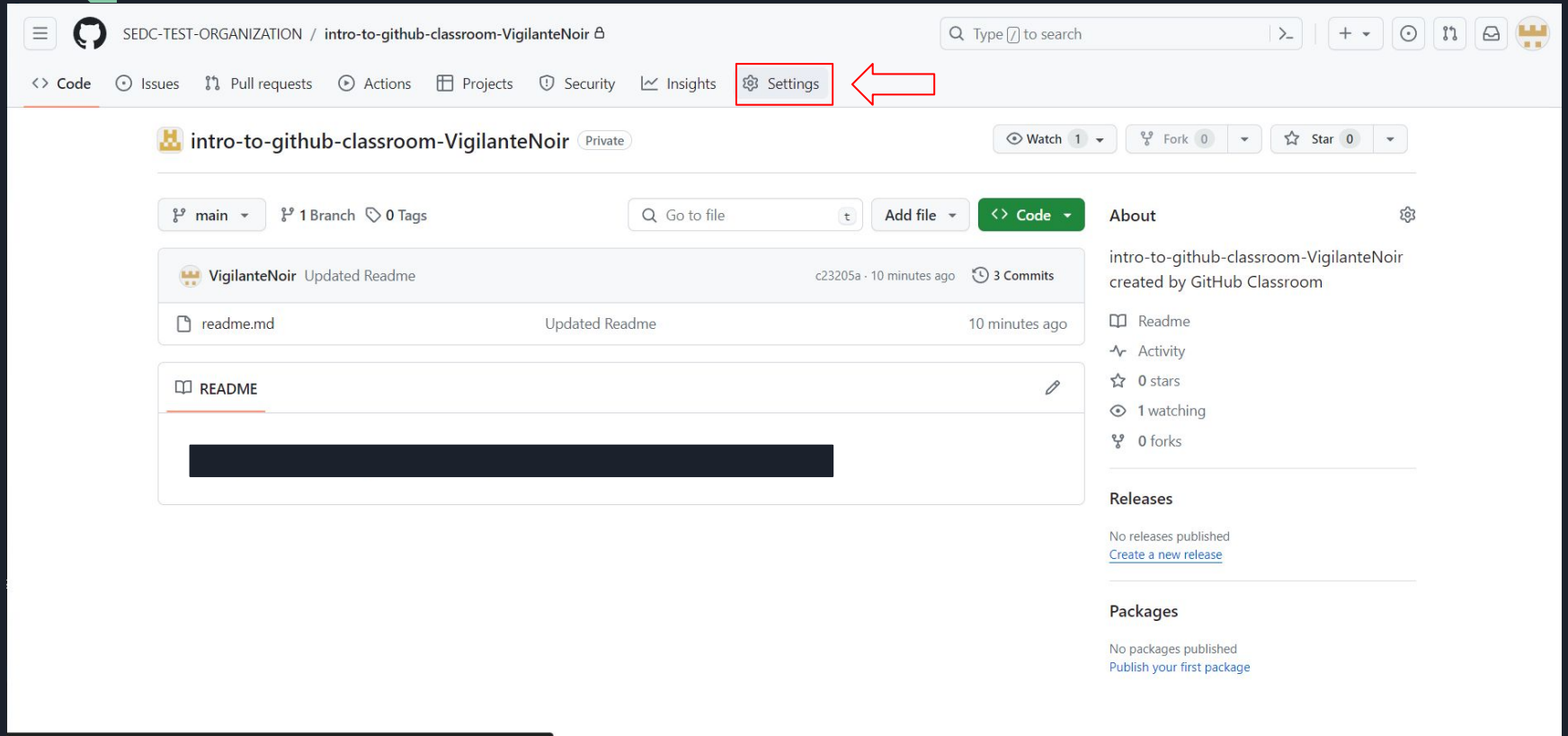
# 4. After logging in, go to Google Classroom, and click on the GitHub Classroom link. Press the Accept this assignment button.



# 5. After waiting a bit and refreshing, click on the link to your assignment repository



# 6. Click on the Settings button in the top right



# 7. Click on the Pages button in the left menu

SEDC-TEST-ORGANIZATION / intro-to-github-classroom-VigilanteNoir

Type / to search

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

General

Access

Collaborators and teams

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Pages

Custom properties Beta

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

## General

Repository name

intro-to-github-classroom-Vigilantel | Rename

Template repository

Template repositories let users generate new repositories with the same directory structure and files. Learn more about template repositories.

Require contributors to sign off on web-based commits

Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for contributors to affirm that their commit complies with the repository's terms, commonly the Developer Certificate of Origin (DCO). Learn more about signing off on commits.

## Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main

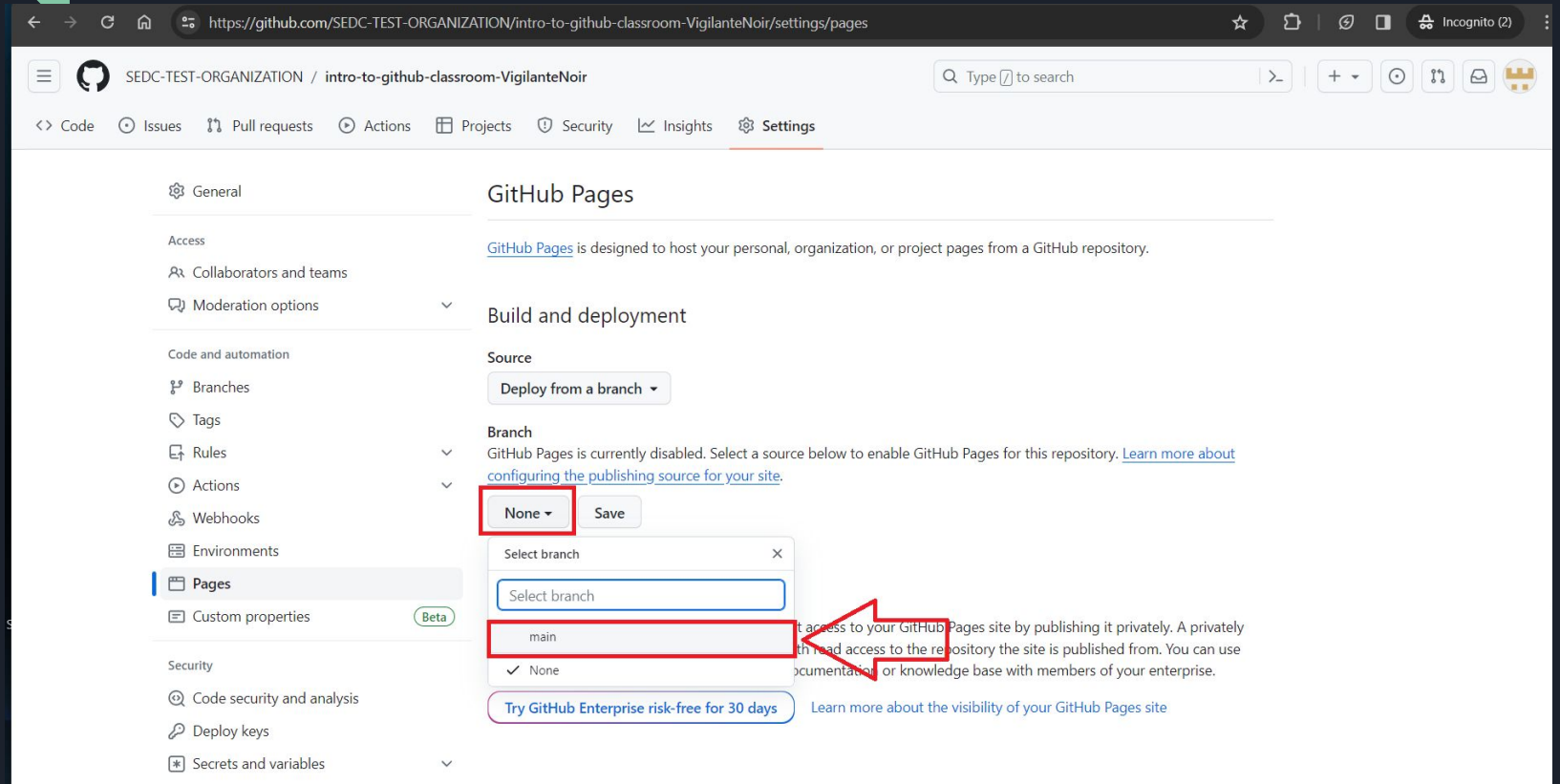
## Features

Wikis

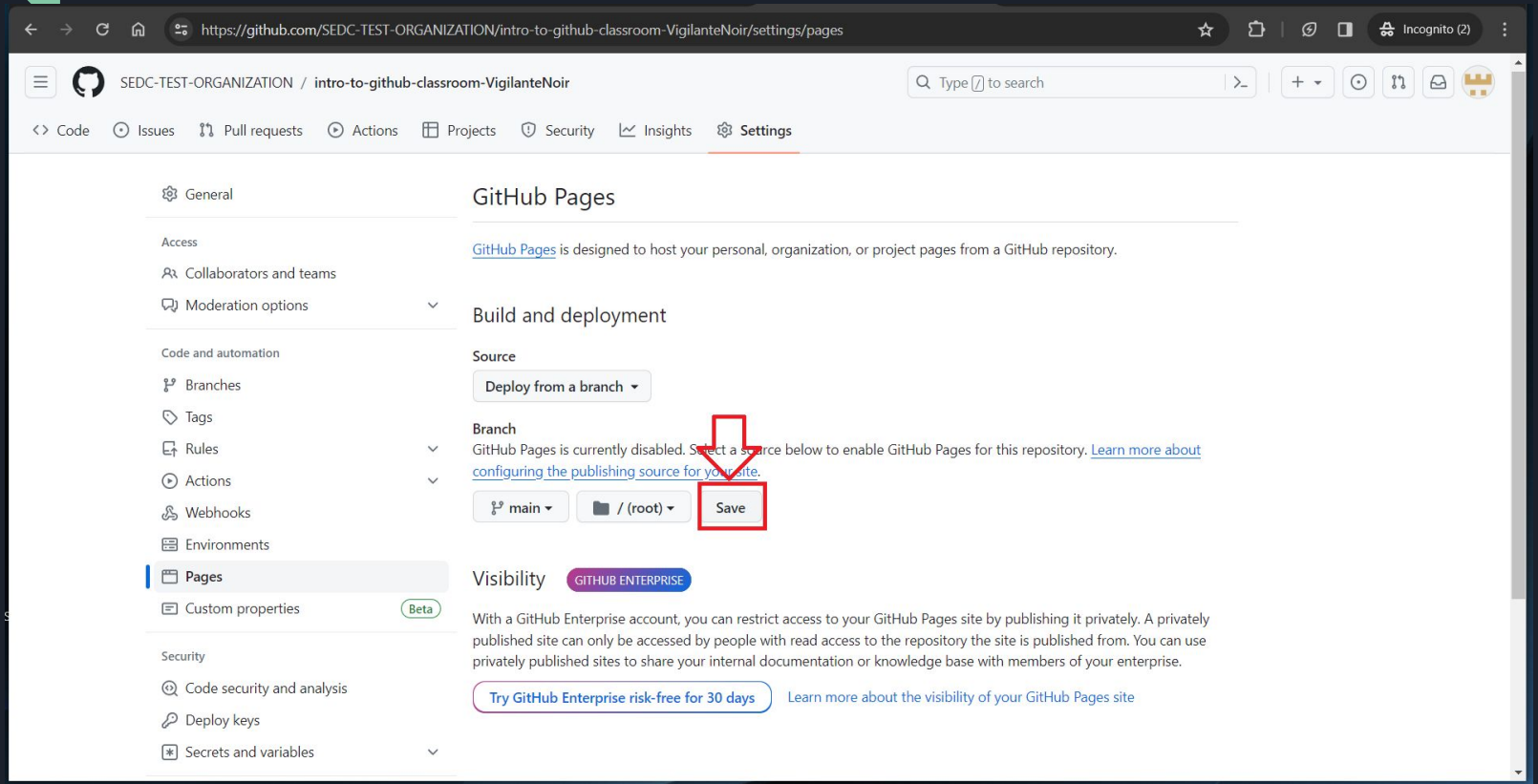
Wikis host documentation for your repository.

https://github.com/SEDC-TEST-ORGANIZATION/intro-to-github-classroom-VigilanteNoir/settings/pages

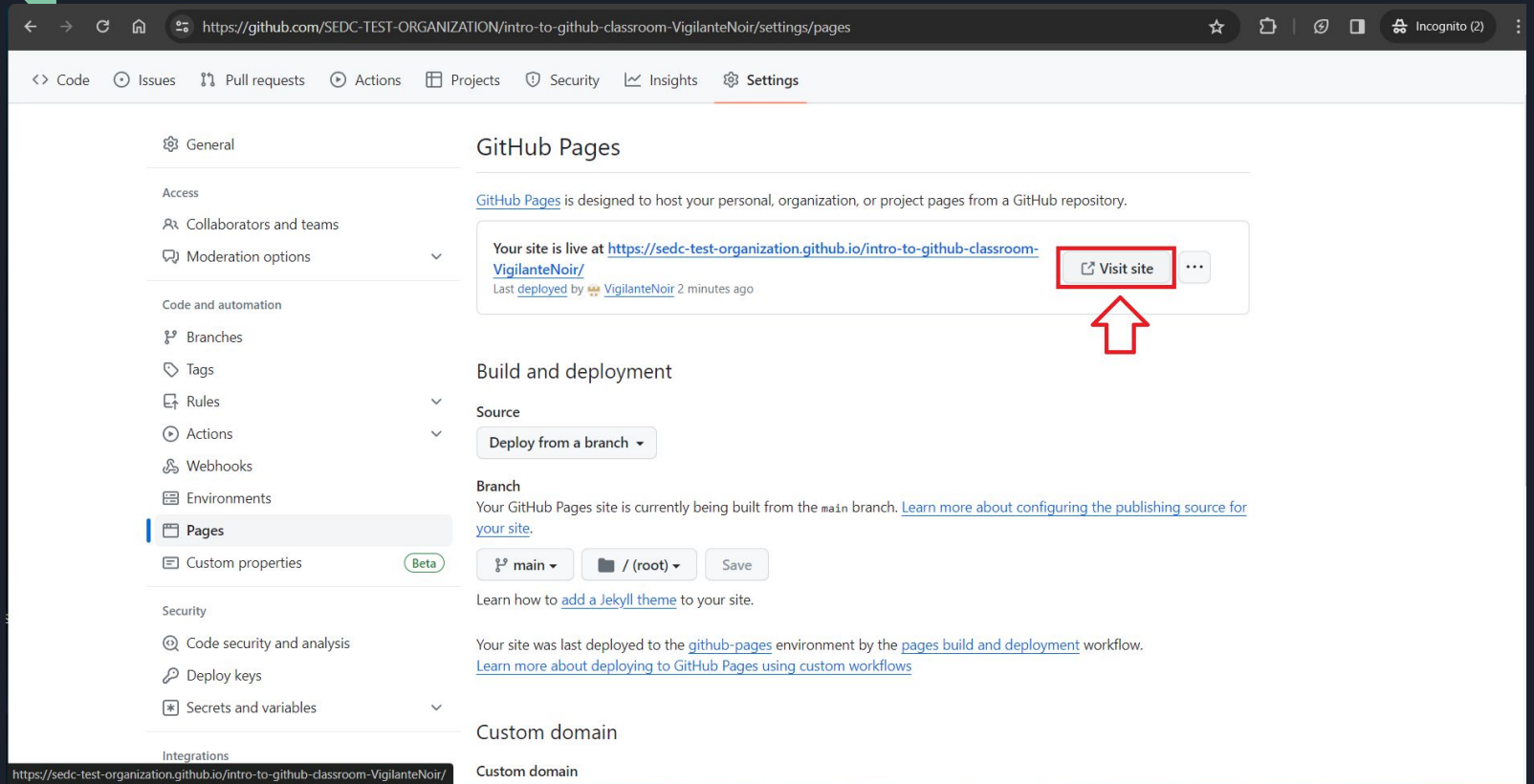
# 8. Click on None under Branch, then select main



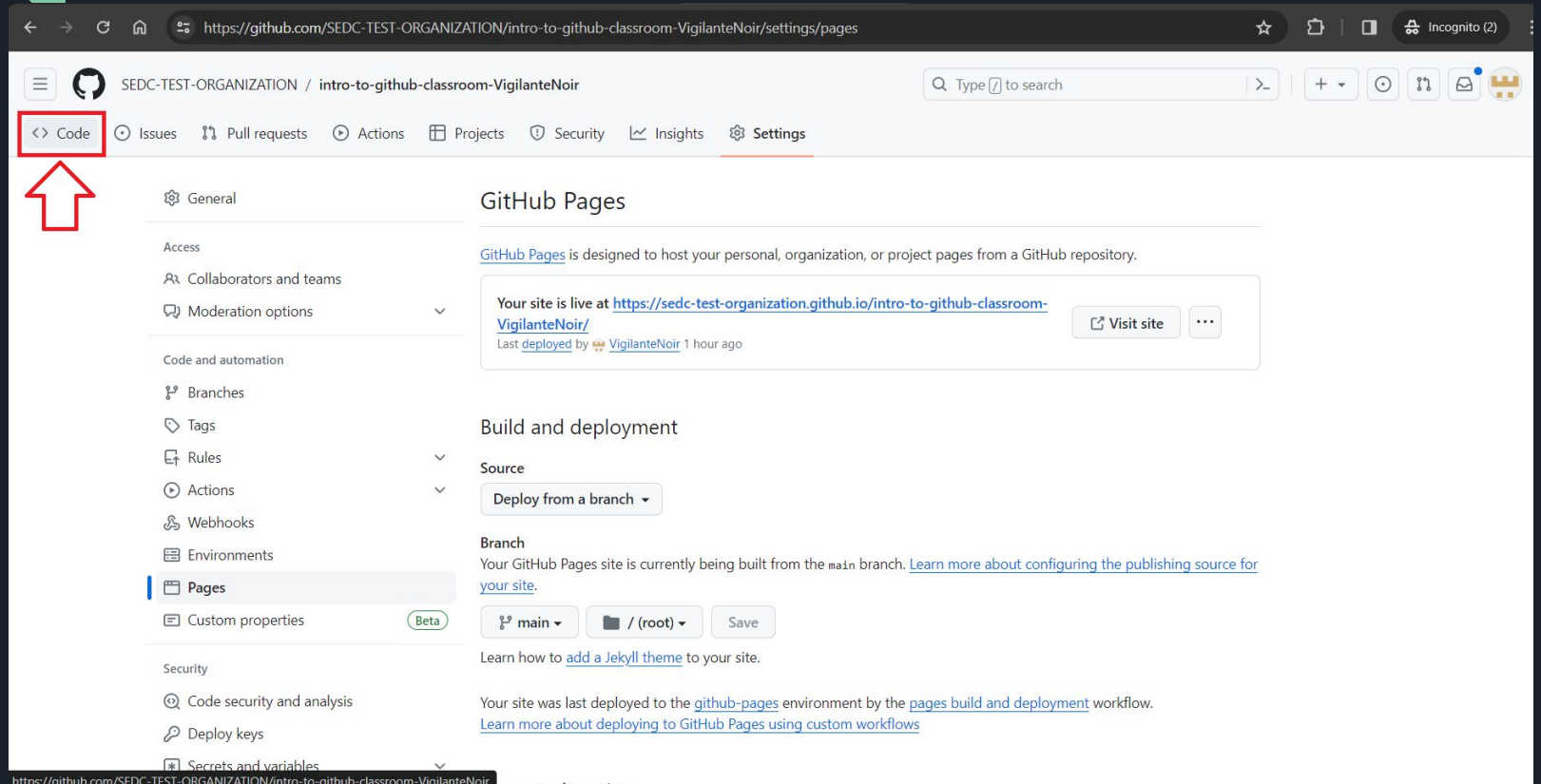
# 9. Press Save to Deploy your Web Page. Then, refresh after waiting a bit.



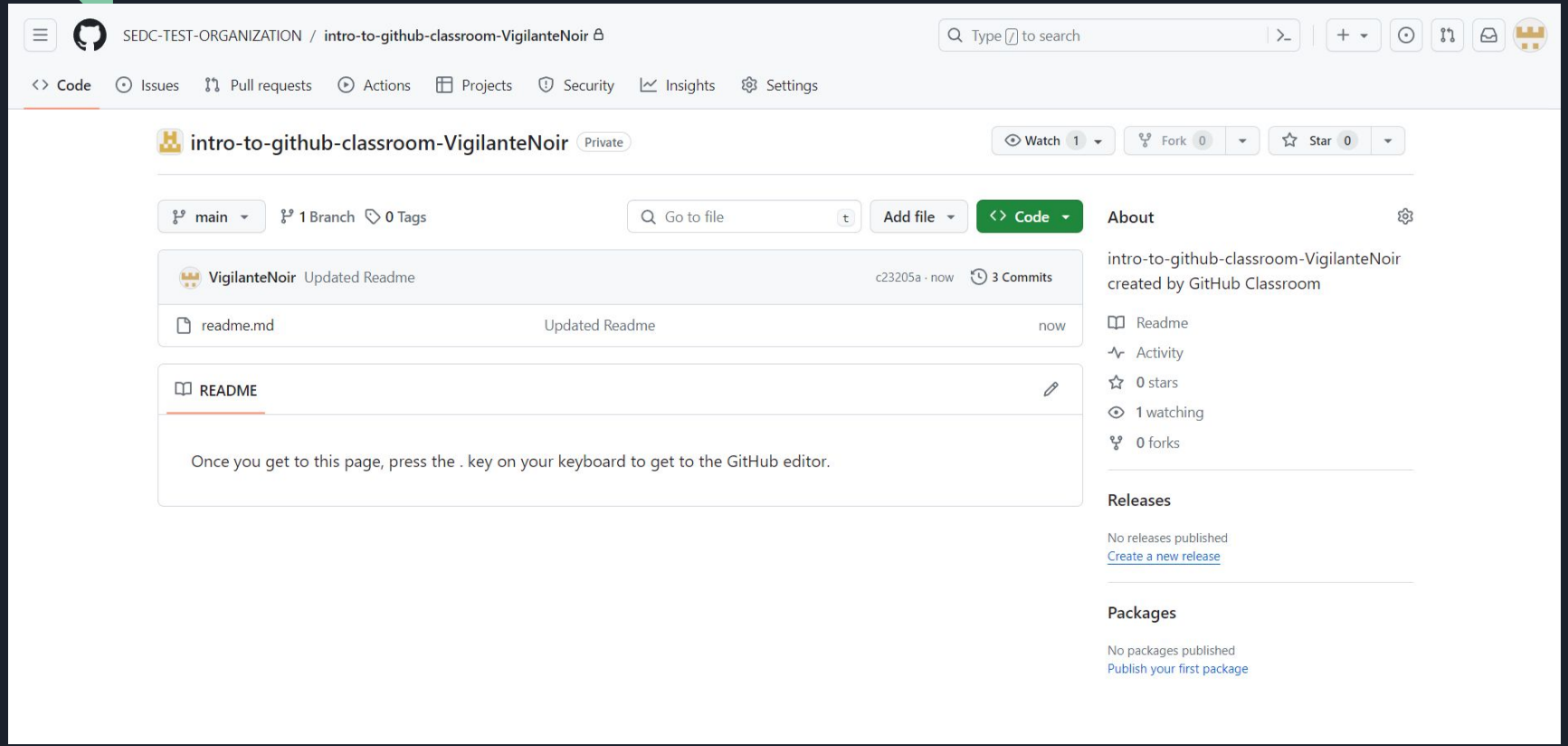
# 10. Click on the Visit Site button to see your Web Page. You may also want to copy the URL for later



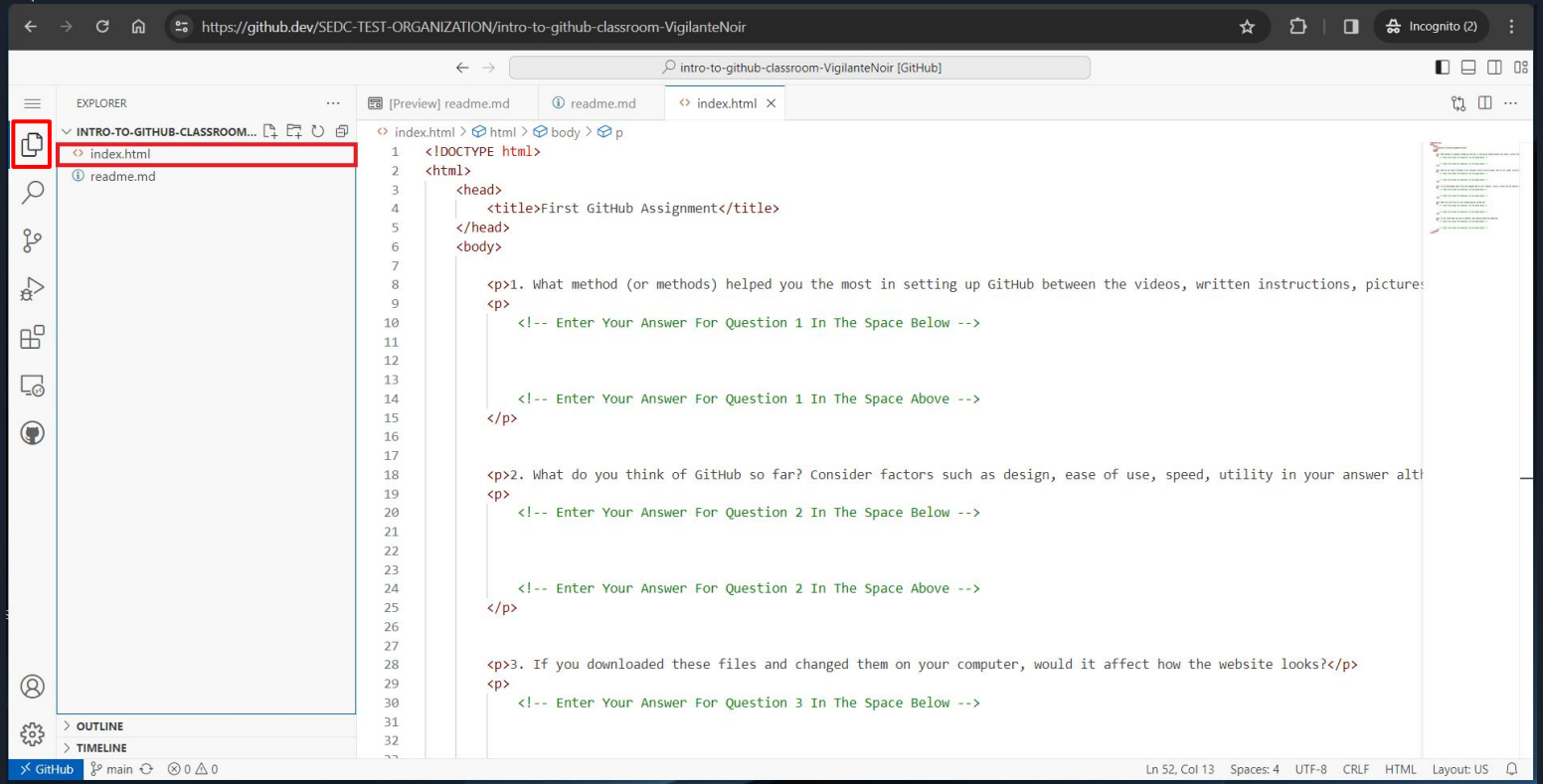
# 11. Press the Code button on the top left



# 12. Press . on your keyboard to get to the GitHub editor



# 13. Click on the Explorer Button (2 pages) in the top left, then click on index.html. Answer the questions in the space provided



# 14. When you're finished answering the questions, press the Source Control button

https://github.dev/SEDC-TEST-ORGANIZATION/intro-to-github-classroom-VigilanteNoir

Incognito (2)

intro-to-github-classroom-VigilanteNoir [GitHub]

EXPLORER

INTRO-TO-GITHUB-CLASSROOM-VIGILANTENOIR [...

index.html M

readme.md

Source Control (Ctrl+Shift+G) - 1 pending changes

[Preview] readme.md | readme.md | index.html M

index.html > html

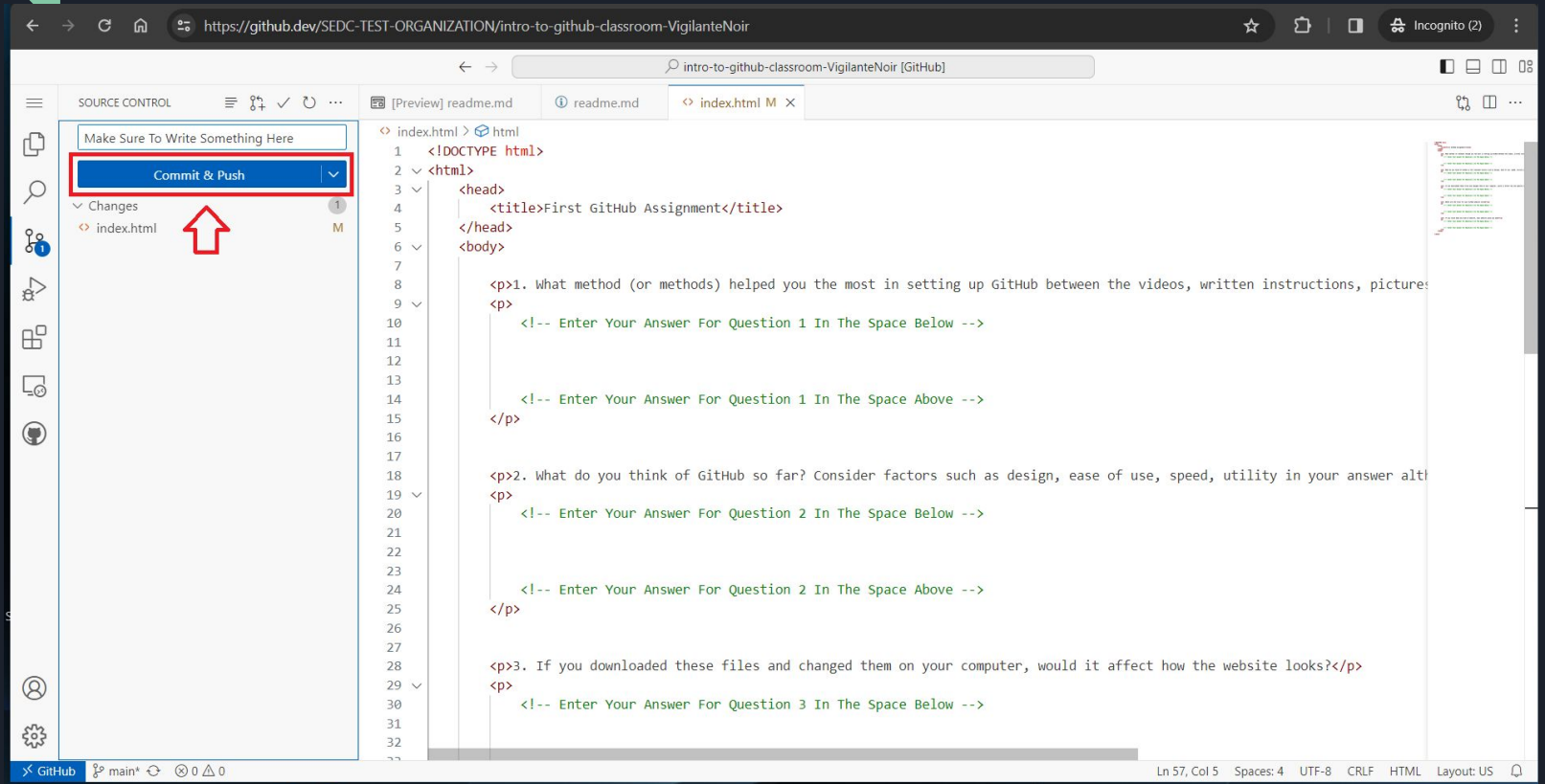
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>First GitHub Assignment</title>
5      </head>
6      <body>
7
8          <p>1. What method (or methods) helped you the most in setting up GitHub between the videos, written instructions, picture
9          <p>
10             <!-- Enter Your Answer For Question 1 In The Space Below -->
11
12
13
14             <!-- Enter Your Answer For Question 1 In The Space Above -->
15         </p>
16
17
18         <p>2. What do you think of GitHub so far? Consider factors such as design, ease of use, speed, utility in your answer alt
19         <p>
20             <!-- Enter Your Answer For Question 2 In The Space Below -->
21
22
23
24             <!-- Enter Your Answer For Question 2 In The Space Above -->
25         </p>
26
27
28         <p>3. If you downloaded these files and changed them on your computer, would it affect how the website looks?</p>
29         <p>
30             <!-- Enter Your Answer For Question 3 In The Space Below -->
31
32
```

OUTLINE

TIMELINE

GitHub | main* | 0 0 | Ln 57, Col 5 | Spaces: 4 | UTF-8 | CRLF | HTML | Layout: US

# 15. Write a message in the space provided and press the Commit & Push button to save your changes



16. Revisit the Pages part of Setting or use the URL you copied earlier to view your web page. It may take a while to update. When you're 100% finished, send the URL for your web page in Google Classroom

# Lesson 02: HTML Syntax and Structure of HTML Documents

# Lesson Structure

```
<u>In</u> case you haven’t noticed, I'm <i>weird</i>. I’m a <i>weirdo</i>. I don't
<b>fit</b> in. And I <b>don't</b> want to fit in. Have you ever seen me without this
<s>stupid</s> hat on? That's <i><u>weird</u></i>.
```

## Standards

- 9-12.IC.6
- 9-12.IC.7

In case you haven’t noticed, I'm *weird*. I’m a *weirdo*. I don't **fit** in. And I **don't** want to fit in. Have you ever seen me without this ~~stupid~~ hat on? That's *weird*.

## Aim: How Do We Write HTML code?

## Warm-Up (2-3 minutes)

Do Now: Compare these two screenshots of nearly identical text. How do these two screenshots of text differ? What connection exists between the two screenshots of text?

(Depicted in the Top Right)

Students will be given 2-3 minutes to review the screenshots and think of their answers before reviewing it as a class

## Warm-Up Review (3 - 5 minutes)

After students are given some time to think of their answers to the Do Now, we'll turn this into a classwide discussion, taking time to hear students' answers to the first question of the Do Now, then the second. If no student mentions it in their answer, ask if they can tell what each tag does/corresponds to. This will act a segue into the slides and lesson for today

# Slide Presentation (15-20 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- A brief comparison of HTML vs Programming Languages
- Syntax of HTML tags
- How to properly nest HTML tags
- The 4 main HTML tags that make up an HTML document (html, head, title, body) as well as DOCTYPE HTML
- A few basic HTML tags for text (h1-h6 tags and the p tag)

In addition, a quick live coding demonstration using Inspect Element to better show what the title tag affects

# Exit Ticket / Homework (Rest of Class)

After presenting the slides, students will spend the rest of class creating a basic personal web page from scratch. The GitHub Classroom starter code will have the file already created for them but will be otherwise empty. For this assignment students will need to

1. Include and properly order the 4 basic structure tags of HTML + DOCTYPE HTML
2. Use one of the heading tags to write their name (or pseudonym if they prefer) inside the body
3. Write another, smaller heading tag that says "About Me" (or another appropriate sub-heading if they prefer) inside the body
4. Use the p tag to write a bit about themselves inside the body
5. Commit/Push when they are finished and if they wish, publish it using GitHub Pages
6. Submit a link to the webpage when they're done on Google Classroom In addition to these instructions being on Google Classroom where they will find the link to the assignment, these instructions will also be in the readme.md with the student's starter code. Anything students do not finish in class will be expected to be done for homework.

# Aim: How Do We Write HTML Code?

Do Now: Compare these two screenshots of nearly identical text. How do these two screenshots of text differ? What connection exists between the two screenshots of text?

```
<u>In</u> case you haven’t noticed, I'm <i>weird</i>. I’m a <i>weirdo</i>. I don't
<b>fit</b> in. And I <b>don't</b> want to fit in. Have you ever seen me without this
<s>stupid</s> hat on? That's <i><u>weird</u></i>.
```

<u>In</u> case you haven’t noticed, I'm *weird*. I’m a *weirdo*. I don't **fit** in. And I **don't** want to fit in. Have you ever seen me without this ~~stupid~~ hat on? That's <u>*weird*</u>.

# HTML vs Other Programming Languages

- Remember that HTML stands for HyperText **Markup Language**, HTML is not a programming language.

- Unlike languages such as Python, Java, C++, etc. HTML doesn't have conditionals, variables, functions, or even math.

- In other words, HTML is more static while other programming languages are more dynamic

# Syntax of HTML Tags

- HTML works based on tags. HTML Tags follow this syntax:

1. HTML tags are surrounded by angular brackets < >
2. Every HTML tag has a name inside the angular brackets. Tags are not displayed on the webpage
3. In between the tags are the content that will actually be displayed on the webpage
4. The first of a pair of HTML tags is known as the start tag or opening tag
5. The second of a pair of HTML tags is known as the end tag or closing tag. It has a / before the first angular bracket

# How to nest HTML tags

- When writing HTML code, you may often have to write HTML tags inside HTML tags
- Like nesting dolls, pairs of HTML tags need to be entirely inside another pair of HTML tags
- Another way to think about it is that the last tag that you open should be the first tag that you close

## Correct

```
...
<body>
    <div>
        <p>This is the right way to
do this</p>
    </div>
</body>
...
```

## Incorrect

```
...
<body>
    <div>
        <p>This is the right way to do
this</body>
    </p>
</div>
...
```

# HTML Structure Tags

- Not all web pages look the same but even still there are a few HTML tags that need to be part of every web page

- These are known as HTML Structure Tags and are as follows:
  - The <html> tag
  - The <head> tag
  - The <title> tag
  - The <body> tag

- In addition, <!DOCTYPE html> should be included as the first line of every HTML document to declare that it is an HTML document

# HTML Structure Tags: <html> tag

- The <html> tag is the outermost tag in an HTML document

- In other words, the opening <html> tag should be the first tag in the document and the closing </html> tag should be the last tag in the document

- As the tag name might imply, all HTML code inside a document should go inside the <html> tag

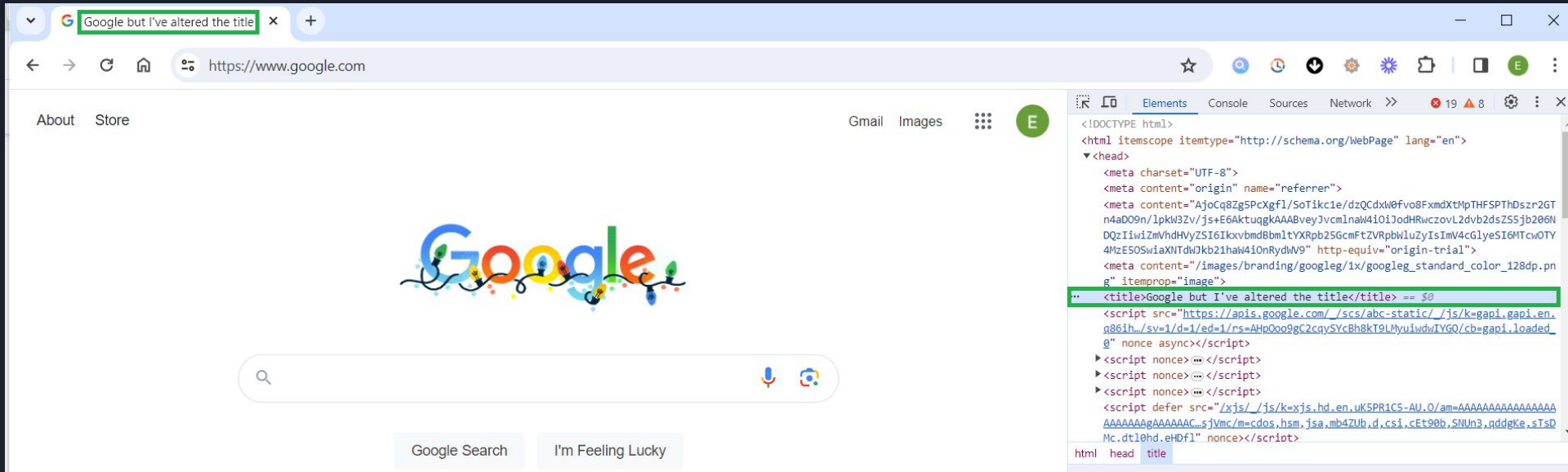
# HTML Structure Tags: <head> tag

- The <head> tag should be the first tag inside the <html> tag

- The <head> tag should be used for a web page's metadata, or in other words information *about* the web page.

# HTML Structure Tags: <title> tag

- The <title> tag is an example of something that would go inside the head tag.
- The <title> tag controls what displays as the tab name for the web page.



# HTML Structure Tags: <body> tag

- The <body> tag goes inside the <html> tag but after, not inside, the <head> tag

- The <body> tag is used for the content that actually appears on a webpage, such as text, images, links, videos, etc.

- Most of the coding we do in HTML will go inside the body tag

# Order of HTML Structure Tags

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Order of HTML Structure Tags</title>
    </head>
    <body>
        . . .
    </body>
</html>
```

# HTML Text Tags

- The <p> tag, which stands for the paragraph tag, is a tag that can be used for writing text in HTML. Like the name implies, it's best suited for paragraphs or longer stretches of text but it doesn't need to be used in that way exclusively.

- The <h1> tag, or the heading tag, is another tag that can be used for writing text in HTML. When compared to the <p> tag, the text of the <h1> tag is bigger and bolder. It is well suited for writing headings in a web page

- In addition to the <h1> tag, there are other tags ranging from <h1> to <h6>. These are also heading tags, but they range in size with h1 being the biggest and h6 being the smallest. These other tags are suitable for sub-headings.

# Lesson 03:
# File Paths and the <img> tag

# Lesson Structure

## Standards

- 9-12.DL.5
- 9-12.NSD.4
- 9-12.IC.6

## Aim: How Can We Use File Paths To Add Images To Our Web Pages?

## Warm-Up (2-3 minutes)

Do Now: Imagine you see some cool art or a funny meme while browsing the web. How would you share that image with others? Try to describe it or write it down as a step-by-step process. Does that process change depending on any factors? (i.e. where you saw the image, if you're on your phone or a computer, who you're sending it to, etc.)

Students will be given 2-3 minutes to think about their answer to the Do Now before going over it as a class

## Warm-Up Review (3-5 minutes)

After students are given some time to think of their answers to the Do Now, we'll turn this into a classwide discussion, taking time to hear students' answers.

If no student mentions it, demonstrate how one way to share images with others is to just send them the image URL (and demonstrate how to get an image URL in case some students don't know). Some platforms even automatically convert the URL into the image, like Discord. Use this to segue into the lesson and slides for the day

## Slide Presentation (20-25 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- The img tag
- explanation of attributes as part of HTML tag syntax
- img src attribute
- img alt attribute
- self-closing tags
- brief discussion/demonstration of how to use file explorer
- a comparison between file paths on computers and URLs (relative vs absolute file paths)
- how to reference other files whether they're in the same, child, or parent folder
- brief discussion of image file types

In addition to the slides, live demo

- Using the img tag with a URL
- The process of downloading an image and finding it in file explorer
- an example of what the activity for today might look like (Code in code/lesson03/teacher folder but a live version also exists here: https://emilejb.github.io/unit-plan-sites/lesson03/)

## Activity / Homework: My Favorite Things (Rest of Class)

After presenting the slides, students will spend the rest of class (finishing it up for HW if they need to) creating a webpage that has images of 6 of their favorite things of categories of their choice, such as favorite movie, favorite book, favorite animal, favorite holiday, etc. Students will be provided with starter code via GitHub Classroom to get started from and follow the comments in the HTML file as well as directions on the board. These instructions will also be in the readme in their starter code. Their repository will have an images folder with a picture showing how to upload to the images folder as well as an index.html with some starter code already present and comments making it clear where they add code to. The repository will also contain a style.css, but that'll be discussed more in the future. Students directions will be as follows

1. In the designated space, add a heading tag that contains the category of one of your favorite things i.e. My Favorite Movie
2. After the heading tag describing the category, add an img tag.
3. Now the fun part, search for an image that represents what your favorite thing in that category is. My favorite movie is Revenge of the Sith, so I might try to find an image of the poster for that move
4. Once you found the image you want to use, download it to your computer, then upload it to your repository. Make sure to put the image in the images folder
5. Set the src attribute for the img tag to the file path for the image you want to use
6. Set the alt attribute for the img tag to a brief description of the image
7. Underneath the img tag, add another smaller heading tag or a p tag saying what the name of the favorite thing is
8. You're not done, but now is a good point to commit/push and deploy the web page to make sure you have everything working properly so far
9. Once you've done that, repeat steps 1-8 for the remaining 5 spaces
10. When you're done with everything, submit a link to the webpage when on Google Classroom so I know that you're finished

# Aim: How Can We Use File Paths To Add Images To Our Web Pages?

Do Now: Imagine you see some cool art or a funny meme while browsing the web. How would you share that image with others? Try to describe it or write it down as a step-by-step process.

Does that process change depending on any factors? (i.e. where you saw the image, if you're on your phone or a computer, who you're sending it to, etc.)

# The <img> tag

- The <img> tag is used to add images to web pages. An example of the syntax for the <img> tag is shown below:

```
<img src = “https://i.imgur.com/FKmX7dt.gif”>
```

**How is this tag different from the some of the tags we viewed in the previous lesson?**

# Self-Closing Tags

- Unlike the tags we learned about previously, the <img> tag doesn't have a start and end tag, it is just made up of one tag

- Tags that are only made up of one tag are called Self-Closing Tags. <img> is just one example of a Self-Closing Tag, but there are other Self-Closing Tags as well

- Because Self-Closing Tags are made up of only a single tag, it's not possible to put other HTML tags between the tags like it is with other tags.

- But it is possible to put something in the tag itself…

# HTML Attributes

- Sometimes HTML tags need extra information to be used properly.
- For example, it's not enough to say we want to add an image to a webpage, we need to specify ***what*** image we want to use
- For cases like this, we use attributes inside the HTML tag, which follow a syntax like this.

```
<tagname attribute="value">
```

- Note that attributes go inside the angular brackets
- A tag is not limited to one attribute and can have multiple attributes
- Attributes are not limited to Self-Closing Tags, but some tags have attributes specific to that tag
- <img> is an example of just one tag that uses attributes, we'll see more later in the class

# Attributes of the <img> tag

- To specify what image to display on a web page with the <img> tag, the src attribute is used
- The src attribute needs to have its value set to the location of an image file
- As seen in the previous slide, this can be done with the URL for an image but there are other ways to do this as well, as we will see soon…

- Another attribute we can add to the img tag is the alt attribute
- The alt attribute is useful for screen reader accessibility or as an alternative to users when a page is loading slowly
- The value of the alt attribute should describe what the image is
- An image tag with an alt attribute might look like this

```
<img src = “https://i.imgur.com/FKmX7dt.gif” alt=”Rick Roll”>
```

# Another way to use src attribute

- While the <img> tag can be used with URLs for images, it can also be used with local images, or images on the same computer / server, as well
- A benefit of using the <img> tag this way is that you don't have to worry about image URLs changing or expiring.
- It is also easier to tell at a glance what image an <img> tag will display when you can see the filename instead of the URL
- Before we go over how to use <img> tags in this way, it's important to become more familiar with how files and folders work on our computers
- We'll do a brief demonstration of how to download an image from the internet and how to find it on your computer

https://store.steampowered.com/app/386940/Ultimate_Chicken_Horse/

Install Steam | login | language

STEAM

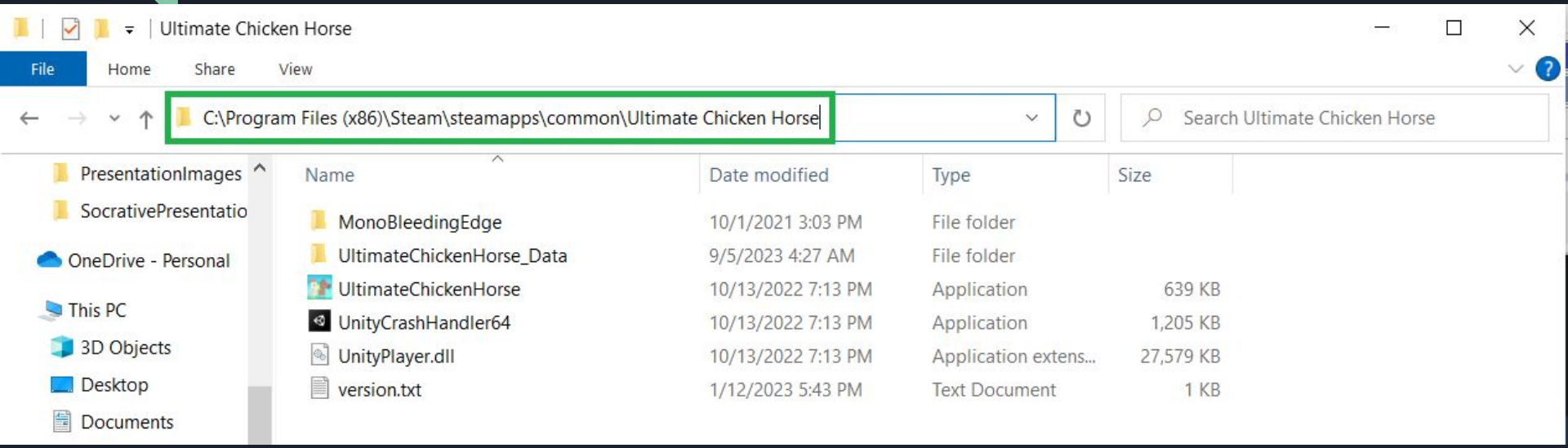
STORE COMMUNITY ABOUT SUPPORT

Your Store | New & Noteworthy | Categories | Points Shop | News | Labs | search

All Games > Indie Games > Ultimate Chicken Horse

# Ultimate Chicken Horse

Community Hub

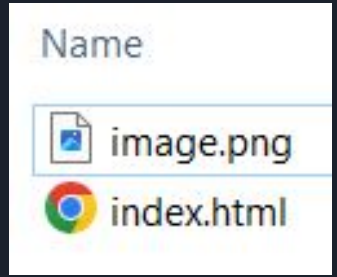

Ultimate Chicken Horse
File
Home
Share
View
C:\Program Files (x86)\Steam\steamapps\common\Ultimate Chicken Horse
Search Ultimate Chicken Horse
PresentationImages
SocrativePresentatio
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Name
Date modified
Type
Size
MonoBleedingEdge
10/1/2021 3:03 PM
File folder
UltimateChickenHorse_Data
9/5/2023 4:27 AM
File folder
UltimateChickenHorse
10/13/2022 7:13 PM
Application
639 KB
UnityCrashHandler64
10/13/2022 7:13 PM
Application
1,205 KB
UnityPlayer.dll
10/13/2022 7:13 PM
Application extens...
27,579 KB
version.txt
1/12/2023 5:43 PM
Text Document
1 KB

# File Paths

- Consider the images on the two previous slides, both relating to the Steam game Ultimate Chicken Horse
- The first image shows the Steam page for game while the second shows the files for the game on my computer
- Both URLs and File Paths can be thought of as directions for the file(s) we're looking for
- In the first image, the URL is telling me to get to the page to buy the game, I need to go to steam website and navigate through the app folder, the unique folder for the game (386940), and then the Ultimate Chicken Horse folder
- In the second image, the file path is telling me to get to the game files I need to go through the Program Files folder, then the Steam folder, next the steamapps folder, after the common folder, and finally the Ultimate Chicken Horse folder

# Relative File Paths (Same Folder)

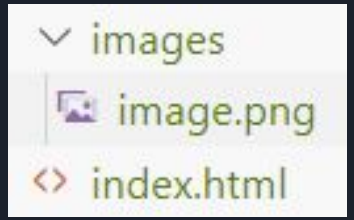
- We can compare file paths to giving directions on how to get to a folder or file(s) we want
- Much like giving directions in real life, it's a lot easier the closer we are to our destination
- Imagine we're writing HTML code in an index.html file inside some folder. Also in that folder is an image, image.png
- If we're starting from index.html, because image.png is in the same folder and so close, the directions to reference the file would be quite simple
- Thus, if we wanted to add an image of image.png, the code for it would look like this




```
<img src="image.png">
```

# Relative File Paths (Child Folder)

- Now imagine that the file that we want, image.png, is inside another folder
- image.png is a bit further away from index.html now, so the directions to get from index.html to image.png are a little bit more complicated
- The code to add an image of image.png in this instance would look like this:

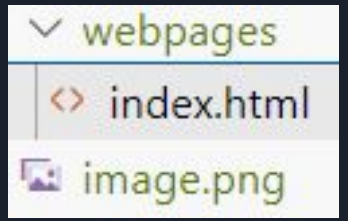



```
<img src="images/image.png">
```

- When a file we want is inside another folder in the same folder our html file is in, we call that a child folder or a subfolder
- When accessing something in a child folder, we have to put the name of the folder(s) it's in and a / before putting the name of the file we want

# Relative File Paths (Parent Folder)

- Now imagine the scenario from the previous slide, but with the positions of index.html and image.png swapped
- In other words, the image file we want is just outside of the folder out html file is in
- The code to add an image of image.png in this instance would look like this:




```
<img src="../image.png">
```

- When a file we want is outside the folder the folder our html file is in, we call that a parent folder
- When accessing something in a parent folder, we need to put ../ before putting the name of the file we want

# File Extensions

- Every file has a file extension at the end that informs what type of file it is and how it should be used
- Our HTML files have a .html extension, such as index.html
- There are actually many different file extensions for images, such as:
  - .jpeg/.jpg: These types of images tend to have smaller file sizes and are better at compressing data, but cannot have transparency
  - .png: Unlike jpegs, pngs can have transparent backgrounds and can be resized without losing too much quality
  - .gif: One of the biggest reasons to use gifs is if you want to include animated images
- As long as it is an image, nearly any image file type can be used with the <img> tag in HTML

# Lesson 04:
# The <a> tag and Connecting Multiple Webpages

# Lesson Structure

## Standards

- 9-12.DL.5
- 9-12.NSD.4
- 9-12.IC.7

## Aim: How Can We Use Links To Create Websites With Multiple Pages

## Warm-Up (3-5 minutes)

Do Now: Students will be asked to play the Wikipedia game, linked here https://www.thewikipediagame.com/

Students will be given 3-5 minute to try to play the game, but they don't need to feel pressured to actually win within the timeframe. This is more of a fun segue into a discussion about how most websites on the internet are made of a collection of web pages, not just a single one.

# Slide Presentation (10-15 minutes)

Present the slides provided in the materials

The slides for this lesson are a bit short as the only new thing being learned is the a tag, although we'll also talk a bit about the logistics of having to deal with multiple web pages as well.

In addition, live demo an example for what the activity for today might look like. Importantly, focus on how the the site structure diagram in the slides corresponds to the experience of using the quiz site and prompt students to pay attention to the URL of each page as we click on links and go from one page to another. (Code for demo is in code/lesson04/ folder but a live version also exists here: https://emilejb.github.io/unit-plan-sites/lesson04/)

# Activity: A Mini-Quiz/Trivia (Rest of Class)

After presenting the slides, students will spend the rest of class (finishing it up for HW if they need to) creating a collection of web pages to make a mini-quiz/trivia game about whatever topic(s) of their choice. Students will be creating all the files and folders themselves from an empty repository (although the instructions below will be included in the readme.md), but they will be given a guide to follow for the organization of their files and presented with a demonstration of what the site should look like and how it should work. Students assignment will also need to meet these requirements

- The game should have an intro page before the first question briefly describing what it will be about
- The intro page should link to the first question of the quiz, which is its own page in a seperate child folder
- Each question page should include a question and at least two answers. At least one of the answers should link to a page that tells the user they answered incorrectly, and at least one of the answers should link to a page that tells the user they answered correctly
- The incorrect page should link back to the intro page for the quiz so that they have to start from the beginning
- The correct page should link to the next question, which will be in it's own child folder of the main folder
- The quiz should be made up of at least two questions, but they can do more if they wish
- Big Hint: This assignment will be much easier if student duplicate files or copy and paste their HTML code and adjust as needed

# Aim: How Can We Use Links To Create Websites With Multiple Pages?

Do Now: Play the Wikipedia Game (https://www.thewikipediagame.com/) and try your best!

# Linking using the <a> tag

- Remember that the HT in HTML stands for Hypertext, which is text that references other text that can immediately be accessed
- Typically, users of the web go from one web page to the next by clicking on links, like in the Wikipedia Game
- Links are one of the most important parts of the web, and we can make our own links by using the <a> tag, also known as the anchor tag
- Below is an example of what the code for an anchor tag may look like as well as what would appear on the web page

```
<a href="https://www.thewikipediagame.com/"> Click Here To Play </a>
```

Click Here To Play

# Syntax of the <a> tag

```
<a href="https://www.thewikipediagame.com/"> Click Here To Play </a>
```

Click Here To Play

- The <a> tag has an attribute named href.
- Like the src attribute for <img>, the value for the href attribute should be the file that we want to redirect to when clicking the link
- Unlike the <img> tag, the <a> tag is not self-closing
- Often when copying and pasting a link in a text or on an app, it will automatically make that link clickable
- However, when coding in HTML we need to add text in between the start and end <a> tags or else there won't be anything to click
- Notice that in this example, "Click Here To Play" is what would be displayed on the web page and what we would need to click to go to The Wikipedia Game site.

# Values for href

- Like we saw with the <img> tag last lesson, we can use a URL as the value for the href attribute or we can use a relative file path
- For example, the code of an <a> tag that links to another HTML file in the same folder may look like this

```
<a href="page2.html"> Go To Next Page </a>
```

- The <a> tag also doesn't have to link to an HTML file, it can link to almost any type of file, including images
- **<u>What would be the difference between the two lines of code below?</u>**

```
1. <a href="image.png">See Image</a>
```

```
2. <img src="image.png">
```

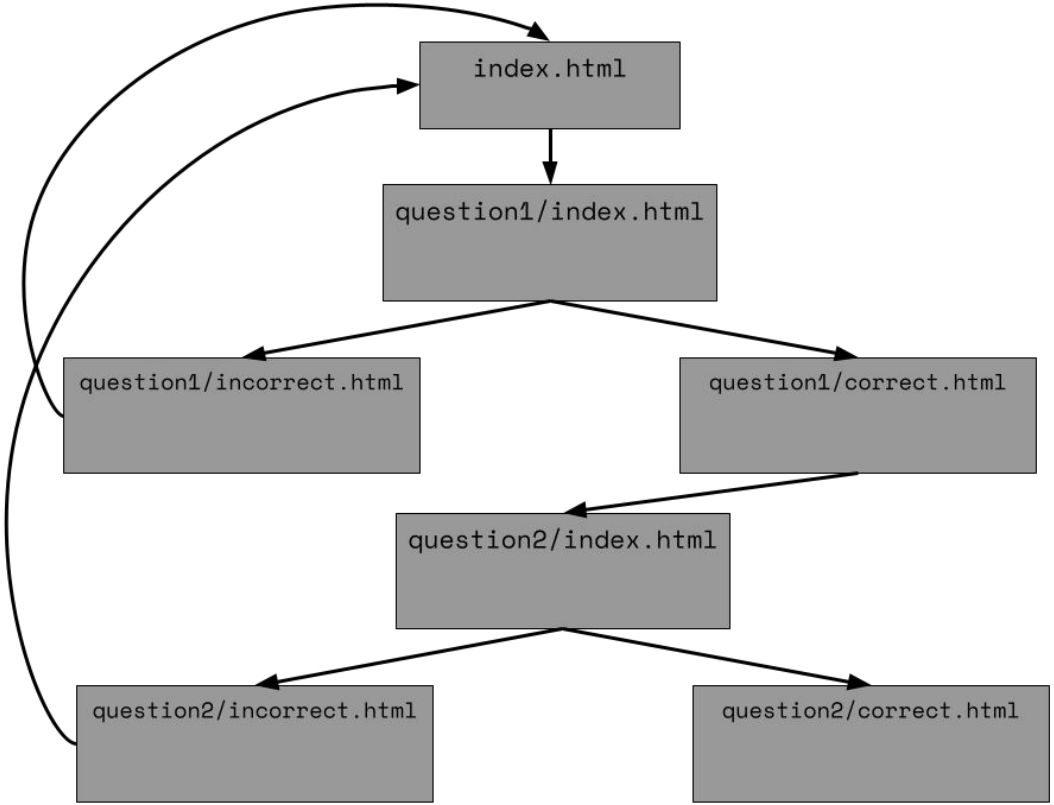
# Website Structure

- So far, we've only worked on making sites that consist of a single web page, but most websites are made up of multiple pages
- Wikipedia for instance, is made up of millions of web pages for each article, but as we saw in the Do Now we can't go from any article on Wikipedia to any other article on Wikipedia in one jump (not without using the search feature anyways)
- When designing a website that consists of multiple pages, it's important to consider how the user navigates through the site and accesses each page
- For example, you probably wouldn't want to work on adding a page to your site that is inaccessible to users because they don't know the URL for the page and no link points to it
- How the pages in a website are connected is the structure of the website and it has a notable effect on the experience of using that site
- It may be helpful to plan or draw out a website's structure before starting to work on it

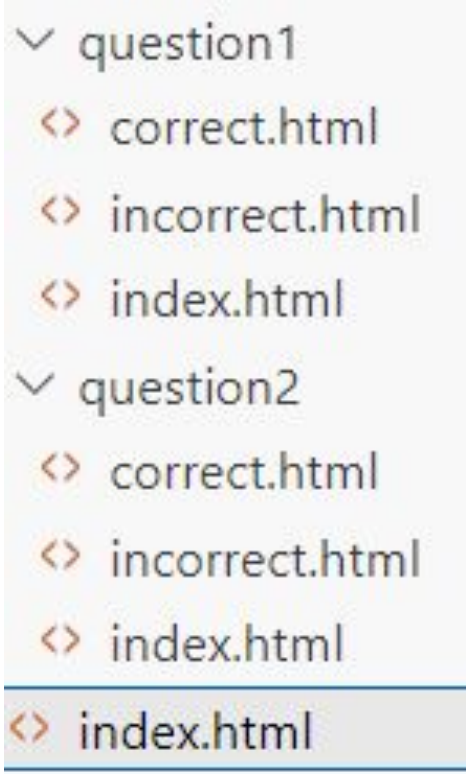
# Structure for Today’s Assignment

- For today’s assignment you’ll be working on making short quizzes
- While there is a guide for you to follow for the website structure of this assignment so you don’t have to come up with a structure on your own, you’ll still need to add the code to actually implement said structure

# Quiz Assignment Structure



# Quiz Assignment Structure (Folder View)



# Lesson 05:
# CSS Syntax and Selectors

# Lesson Structure

## Standards

- 9-12.IC.7
- 9-12.IC.6
- 9-12.DL.5

## Aim: What is CSS and How Can We Use It to Style Our Web Pages?

## Warm-Up (2-3 minutes)

Do Now: Think about some of the websites you've visited or seen in the past. What websites do you like the design of, and why? Or what website designs do you dislike, and why?

Students will be given 2-3 minutes to think about their answer to the Do Now and discuss it with classmates before going over it as a class.

## Warm-Up Review (3-5 minutes)

After students are given some time to think of their answers to the Do Now, we'll turn this into a classwide discussion, taking time to hear students' answers. This will be used to segue into the slides for today's lesson

# Slide Presentation (20-25 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- Brief history of CSS
- Explanation of what CSS is and it's role in the web
- CSS syntax
- CSS selectors by tagname, class, and id
- the class and id attributes
- the color and background-color properties
- live demonstration of CSS in action
- Style tag vs style.css file
- link tag to link style.css
- example of what students will be working on next

In addition to the slides, live demo

- Using the color and background-color properties by using Inspect Element
- The effect of using different selectors with the same rules by using Inspect Element
- The Color Pallette generating links
- an example of what the activity for today might look like (Code in code/lesson05/teacher folder but a live version also exists here: https://emilejb.github.io/unit-plan-sites/lesson05/)
- Demonstrate how Inspect Element can be used to experiment with values on their websites without needing to rewrite, commit, wait, and refresh

# Activity: Mini-Quiz/Trivia Pt. 2 (Rest of Class)

For today's activity students will revisit their quiz that they worked on in the previous lesson and try using a bit of CSS to add some color to it. Students will be encouraged to have fun with it but there will be some requirements for the assignment they'll be expected to meet. Those are

- Use at least two different background colors for different pages on the quiz. For example, you could make the pages for question 1 one color and pages for question 2 another, or you could split it by the correct or incorrect pages.
- The quiz actually has big flaw where it can be easy to tell what all the incorrect answers are by looking at the link colors. Make it so that all links are the same color as to not give anything away
- On the correct pages (not the one where the user picks their answer), style the answers so that the incorrect answer(s) are styled one way (maybe colored red) while the correct answer(s) are styled another way (maybe colored green)
- Consider using the links discussed and shown in class to help pick colors for your quiz if you find it hard to choose what colors to use yourself

If students don't end up finishing in class, they will finish it as Homework

# Resources

- https://www.w3schools.com/tags/ref_colornames.asp
- https://www.w3schools.com/colors/colors_hexadecimal.asp
- https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html
- https://www.learnui.design/tools/accessible-color-generator.html
- http://colorsafe.co/

# Aim: What is CSS and How Can We Use It to Style Our Web Pages?

Do Now: Think about some of the websites you've visited or seen in the past. What websites do you like the design of, and why? Or what website designs do you dislike, and why?

# What is CSS?

- CSS stands for Cascading Style Sheets
- CSS is used to style web pages such as by adding or altering the color, font, layout, size, or more
- A way to think about CSS in relation to HTML is that if web pages are houses, HTML would be the building itself or the structural parts of the house while CSS would be how the house is decorated

# A Brief History of CSS

- CSS was published as a W3C Standard in 1996
- Internet Explorer was actually the first commercial browser to support CSS
- Before CSS was published browsers had their own stylesheets, meaning how web pages looked was browser dependant, much more so than today
- This also meant that web developers had less control over what their web pages looked like

# How CSS works

- Like HTML, CSS is not a programming language like C++, Java, or Python
- However, CSS is also not a Markup Language like HTML either
- CSS essentially works by describing a series of rules for the browser to follow when displaying a web page
- As such, CSS actually has it's own syntax, rules, and structure distinct from HTML
- CSS code generally follows this structure

```
selector {
    property:value;
    property:value;
}
```

# CSS Properties

- CSS properties control various aspects of a web page and the elements within that web page
- If we want to add or change a property of an HTML element, we have to specify what property we want to change and what value that property should be set to
- Notice that we need to but a colon between the property and its value, and a semicolon after its value
- There are many, many CSS properties, some of which we will learn later, but the first CSS properties we will learn about are color and background-color
- The color property changes the text color of an HTML element
- The background-color property changed the background color of an HTML element

# Using colors in CSS

- Both the color and background-color properties take colors as values
- Names of colors can be used as the values for these properties, but CSS doesn't recognize every color name
- The colors that CSS does recognize can be found here: https://www.w3schools.com/tags/ref_colornames.asp
- A wider variety of colors (over 16 million) can be used as values for these properties if using hex codes instead
- Hex codes describe how intense the red, green, and blue values of a color are, but in hexadecimal instead of decimal
- Hex codes are described in more detail here and can also be used to help generate hex codes
- https://www.w3schools.com/colors/colors_hexadecimal.asp

# Tools to pick colors

- While you can free to use any combination of colors you want while designing your websites, some combinations of colors work better than others
- If you want some assistance picking colors for a website, the links below may be helpful
- https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html
- https://www.learnui.design/tools/accessible-color-generator.html
- http://colorsafe.co/
- In addition to generating good looking color schemes, both links are designed to generate color palettes and combinations that meet the guides for accessibility on the web
- Using color palettes and combinations with a sufficient amount of contrast make it easier for those with impaired vision to access those websites

# Selectors in CSS

- While CSS properties describe what style changes are to be made on a web page, the selector is what determines what HTML element(s) those changes are applied to
- There are a variety of ways to use selectors with CSS, but we'll cover only 3 in today's lesson
- Tag name selectors, also called type selectors, apply CSS to all HTML elements that use that tag name
- To specify what CSS rules we want to apply to what selectors, we enclose the CSS property and values we want to change for that element in curly brackets after the selector
- An example of using the tagname selector is shown in the next slide.

# Tagname Selector Example

**CSS**

```
p {
    color:red;
}
```

**HTML**

```
<p>Sentence 1</p>
<p>Sentence 2</p>
<p>Sentence 3</p>
<p>Sentence 4</p>
<h3>Sentence 5</h3>
```

**Output**

Sentence 1

Sentence 2

Sentence 3

Sentence 4

**Sentence 5**

# Ids in HTML and CSS

- The example in the previous slide shows how we can style all <p> tags to be red (while the h3 tag is unchanged)
- However, what if we only wanted to change one of the <p> elements but not all of them?
- An Id is an attribute we can add to an HTML element. It doesn't do anything on its own, but it allows us to specifically target that element with our CSS
- Unlike the src attribute for <img> or href attribute for <a>, the id attribute is not specific to any particular tag.
- In fact, almost any tag can have an id attribute, this is known as a global attribute
- The value for the id can be almost anything, but it can't have spaces or start with a number
- Styling something by Id in CSS is very similar to styling by tag name, except the tag name is replaced with the of what you want to style, preceded by a hashtag (#)

```
#id {
    property:value;
}
```

# Id Selector Example

**CSS**

```
#unlucky {
    color:red;
}
```

**HTML**

```
<p>Sentence 1</p>
<p>Sentence 2</p>
<p>Sentence 3</p>
<p id="unlucky">Sentence 4</p>
<h3>Sentence 5</h3>
```

**Output**

Sentence 1

Sentence 2

Sentence 3

Sentence 4

**Sentence 5**

# Classes in HTML and CSS

- The example in the previous slide shows how we can style a specific <p> element
- However, what if we wanted to style more than one <p> element, but not all of them?
- We could create multiple ids for each thing we want to style, but if we want them to have the same style there's a better way
- A Class is another attribute we can add to an HTML element.
- It functions very similarly to Ids, but where Ids are meant to be unique, multiple elements can have the same class
- Styling something by Class in CSS is similar to styling by id as well, except the id is replaced with the class name of what you want to style and the hashtag is replaced with a period (.)

```
.classname {

    property:value;

}
```

# Class Selector Example

**CSS**

```
.odd {
    color:red;
}
```

**HTML**

```
<p class="odd">Sentence 1</p>
<p>Sentence 2</p>
<p class="odd">Sentence 3</p>
<p id="unlucky">Sentence 4</p>
<h3 class="odd">Sentence 5</h3>
```

**Output**

Sentence 1

Sentence 2

Sentence 3

Sentence 4

**Sentence 5**

# Where to write CSS

- The question remains of where do we actually write the CSS code?
- We'll discuss two ways to write and apply CSS code to our web pages
- One way is to use the <style> tag in the <head> portion of our HTML document
- The <style> tag is unique because although it is an HTML tag, it's designed for CSS code to be written inside
- The CSS code written inside the style tag will apply to the HTML document its in

# Style Tag Example

```
<html>
      <head>
            <title>CSS Example</title>
            <style>
            .odd {
                  color:red;
                  }
            </style>
      </head>
      <body>
<p class=”odd”>Sentence 1</p>
<p>Sentence 2</p>
<p class=”odd”>Sentence 3</p>
      </body>
</head>
```

**Output**

Sentence 1

Sentence 2

Sentence 3

# CSS Files and <link> tag

- CSS can also be written in a separate file. Just as we have .html files, we also have .css files
- If CSS is written in a different file, the HTML document needs to be able access that file somehow
- The <link> tag can do a few things, but the most important is its ability to link CSS files to our HTML documents
- The <link> tag goes in the <head> portion of the HTML document, and might look like this

```
<link href="style.css" rel="stylesheet">
```

- Like the <a> tag, the href attribute here determines the file that is being referenced
- We also need to include rel="stylesheet" to make it clear that the file being linked is a CSS file

When working on a website with multiple similar pages, would it be better to use the <style> tag or to create a CSS file and use the <link> tag to implement CSS?

# Lesson 06:
# Text Styling Using CSS

# Lesson Structure

## Standards

- 9-12.IC.7
- 9-12.DL.5

## Aim: How Can We Use CSS to Style Text?

## Warm-Up (2-3 minutes)

Do Now: What is the difference between the different heading tags? What about the difference between the heading tags and the paragraph tag? Generally, how can you tell whether something is a clickable link or not?

Students will be given 2-3 minutes to write down their answer.

## Warm-Up Review (2-3 minutes)

After students are given some time to do the Do Now, the teacher will call on students to share their answers, which will be used to segue into the lessons and the slides.

# Slide Presentation (15-20 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- Default CSS properties of tags
- font-size CSS property
- font-weight CSS property
- font-style CSS property
- text-decoration CSS property
- text-align CSS property
- font-family CSS property
- how to use Google Fonts for more font choices
- span tag
- example/explanation of what students will be working on next

In addition to the slides, live demo

- The CSS properties being introduced
- How to use Google Fonts
- How using a span tag helps by *not* making a new line
- an example of what the activity for today might look like (Code in code/lesson06/teacher folder but a live version also exists here: https://emilejb.github.io/unit-plan-sites/lesson06/)
- How they might use https://imsdb.com/ as part of their assignment
- Demonstrate again how Inspect Element can be used to experiment with values on their websites without needing to rewrite, commit, wait, and refresh

# Activity: Styling Dialogue (Rest of Class)

For today's activity students will practice using text styling by styling dialogue. The dialogue can come from anywhere. A sample text will be provided for students who want to get to coding right away while https://imsdb.com/ will also be provided as an option for students to browse if they want to take dialogue from a movie they enjoy.Theoretically one could also collaborate with English teachers and pick samples corresponding to those classes. Ultimately, the content and origin of the dialogue doesn't really matter. Students will be using text styling to differentiate the dialogue between different characters as well as also using text-styling with the span tag to emphasize certain parts of the text. Students assignments will need to meet these requirements but are welcome to go beyond them

- Getting at least 8 lines of dialogue split between at least two or more characters
- Using the font-size property to make the text bigger and easier to read on the screen
- Use text-styling to differentiate the text of the different characters
    - Specifically, each character should at least have different font-family and color properties.
    - At least one character should also have left-aligned text and at least one character should have right-aligned text.
    - Other CSS properties can also be used to differentiate between the characters if desired.
- Use the span tag to change the styling of certain parts of the text
    - The font-weight, font-style, and text-decoration property should each be used at least once for this purpose
    - You may also choose to change the color, size, or font of certain parts of the text if you wish If students don't end up finishing in class, they will finish it as Homework

# Resources

- https://imsdb.com/
- https://www.w3schools.com/cssref/css_websafe_fonts.php
- https://fonts.google.com
- https://www.youtube.com/watch?v=46m_4P492P4 (Students who weren't in class or need to go over the steps may watch this)

# Aim: How Can We Use CSS to Style Text?

Do Now: What is the difference between the different heading tags?

What about the difference between the heading tags and the paragraph tag?

Generally, how can you tell whether something is a clickable link or not?

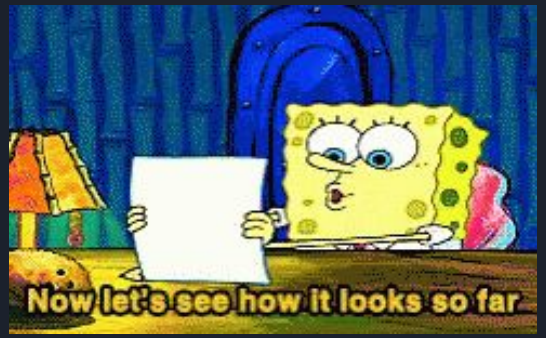
# Default CSS Properties

- Even if we don't add any CSS code to our HTML, our HTML elements still have default CSS properties
- For example, think about the color property we learned about last lesson
- We don't usually have to specify that our text should be black, that just tends to be the default
- However, <a> tags are notably blue (or purple) by default even without specifying their color
- This is because certain HTML tags have certain default properties
- By default, the heading tags are always bolded and <h1> is always bigger than <h6>
- But just like we learned to change the color of our links last lesson, we can change these other default text properties as well

**Before we go over some them, what are some of the different ways that we can style text, or change how text looks?**

# Text Styling

- Many of the text styling options possible on the web are similar to what you can do with text in Microsoft Word, Google Docs, or some forum sites.
- Some of the ways we can style text include changing the font, the size, making it bold, changing the alignment, making it bold or italicized or underlined
- Although not as simple as just pressing a button, we can do similar text styling for our websites by using CSS
- However, to do so we need to learn the corresponding CSS properties and their possible values



# Font Size

- One of the most noticeable ways we can style text is by changing its size
- In CSS, we can change the size of text in that HTML element when displayed on a browser by using the font-size property
- When using the font-size property, we need to set its value to a number, similar to what we may do in Microsoft Word or Google Docs
- However unlike those programs, we need to specify a unit after the number. This is actually common for a lot of CSS properties that take numeric values
- While there are a number of different units that we could potentially use, for this lesson we'll just focus on the unit px, short for pixel

```
h3 {
    font-size:24px;
}
```

# Text Align

- Another way that we can style text is by changing its alignment, or whether its on the left, center, or right side
- In CSS, we can change the alignment of an HTML element when displayed on a browser by using the text-align property
- The text-align property can be set to left, center, or right. It can also be set to justify, which stretches each line except the last to all be the same width, like in newspapers
- Despite the name, the text-align actually affects more than just text, and can change the alignment of images too

```
h1  {
    text-align:center;
}
```

# Font Weight

- Another way we can style text is by altering the weight, or the boldness of it
- In CSS, we can change the boldness of text in that HTML element when displayed on a browser by using the **font-weight** property
- Setting the font-weight property to bold will make the text bold
- Alternatively, we can set the font-weight property to normal if styling an element that is already bold by default, like the header tags

```
p  {
    font-weight:bold;
}
```

```
h6 {
    font-weight:normal;
}
```

# Font Style

- Italicizing is another way that we can style text
- In CSS, we can change the italicize text in that HTML element when displayed on a browser by using the *font-style* property
- Setting the font-style property to italic will italicize the text
- Setting the font-style property to normal will un-italicize the text

```
h4  {
    font-style:italic;
}
```

# Text Decoration

- If you want to underline text in HTML, that can be done using the text-decoration CSS property and setting it to underline
- If you have a tag that is underlined by default, like the <a> tag, you can set text-decoration to none to remove the underline
- text-decoration can also be set to line-through to add a line through the text, or overline to add a line above the text (often useful for math purposes like repeating decimals)

```
a  {
    text-decoration:none;
}
```

```
h2 {
    text-decoration:underline;
}
```

# Font Family

- Choosing the right font for a website (or anything really) can make a big **impact**
- In CSS, we can change the font of text in that HTML element when displayed on a browser by using the `font-family` property
- Unlike the other text-styling properties we saw earlier, the font-family property typically takes two values, separated by a comma
- The first value is the name of font being used
- The second value acts as a backup if the first value doesn't work for some reason, and is a generic font (of which there a 5 types)

# The Five Generic Fonts

- Serif: These fonts contain serifs — small decorative strokes that protrude from the main body of the letter. These tend to be better for printed media
- Sans-Serif: These fonts do not have serifs. These tend to be better for media viewed on screens
- Monospace: These fonts have equal spacing between characters,
- Cursive: These fonts tend to resemble handwriting in some way
- Fantasy: These fonts are highly stylized and decorative

# Font Options

- https://www.w3schools.com/cssref/css_websafe_fonts.php
- The link above contains a few examples of web safe fonts, meaning fonts that can be reasonably expected to work on any browser
- The code for altering the font-family property may look like this:

```
h5 {
    font-family: "Gill Sans", sans-serif;
}
```

- These are not the only fonts useable on the web, but some other fonts may depend on the user's browser or computer
- However, there are ways to access even more font choices for your websites and better ensure that they work on a variety of browsers and computers

# Google Fonts

Google Fonts

#  tag

- What if we want to style a specific word or section of our text differently from the rest?
- Unfortunately, we can't put that section in its own <p> tag or header tag because each new tag creates a new line in our web page
- To solve this problem, we can use the  tag
- Unlike other tags, the span tag doesn't create a new line and can be used in the middle of our text
- The span tag can be styled like any other tag, especially if given an id or a class
- An example use of the span tag might look like this

**CSS**

```
#heat {
    color:red;
    font-weight:bold;
}
```

**HTML**

```
<p>This song is absolutely <span
id="heat">fire!!</span></p>
```

**Output**

This song is absolutely **fire!!**

# Lesson 07: Positioning and Resizing Using CSS

# Lesson Structure

## Standards

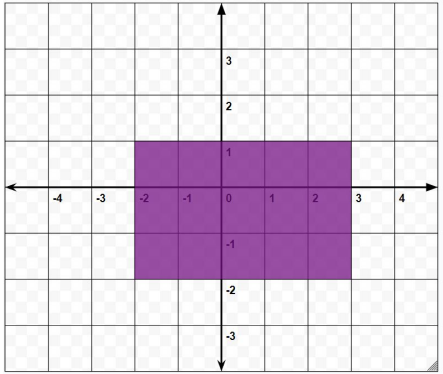
- 9-12.IC.7
- 9-12.DL.5

## Aim: How Can We Use CSS to Resize and Position HTML Elements?

## Warm-Up (2-3 minutes)

Do Now: What are the coordinates of each corner of the rectangle in the image?

Students will be given 2-3 minutes to write down their answers.




## Warm-Up Review (3-5 minutes)

After students are given some time to do the Do Now, the teacher will call on students to share their answers, which will be used to segue into the lessons and the slides.

# Slide Presentation (20-25 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- Coordinates in Math vs Coordinates in Computer Programs
- top,left,bottom,right CSS properties
- position CSS property
- position: static
- position: relative
- position: absolute
- position: fixed
- width, height CSS property
- background-image CSS property
- example/explanation of what students will be working on next

In addition to the slides, live demo

- The values of the position property
- Using left/right/top/bottom to position elements
- Using the width and/or height property for images
- An example of what the activity for today might look like (Code in code/lesson07/teacher folder but a live version also exists here: https://emilejb.github.io/unit-plan-sites/lesson07/)
- Demonstrate again how Inspect Element can be used to experiment with values on their websites without needing to rewrite, commit, wait, and refresh (this is especially important for this lesson as positioning elements how students want might take significant trial and error)

# Activity: Building a Scene (Rest of Class)

For today's activity students will practice using CSS to create a scene composed of multiple images (images will be provided for students to use (located in the resources folder) to make the activity easier to start but they are also welcome to search for and use any of their choice). Students assignments will need to meet these requirements but are welcome to go beyond them

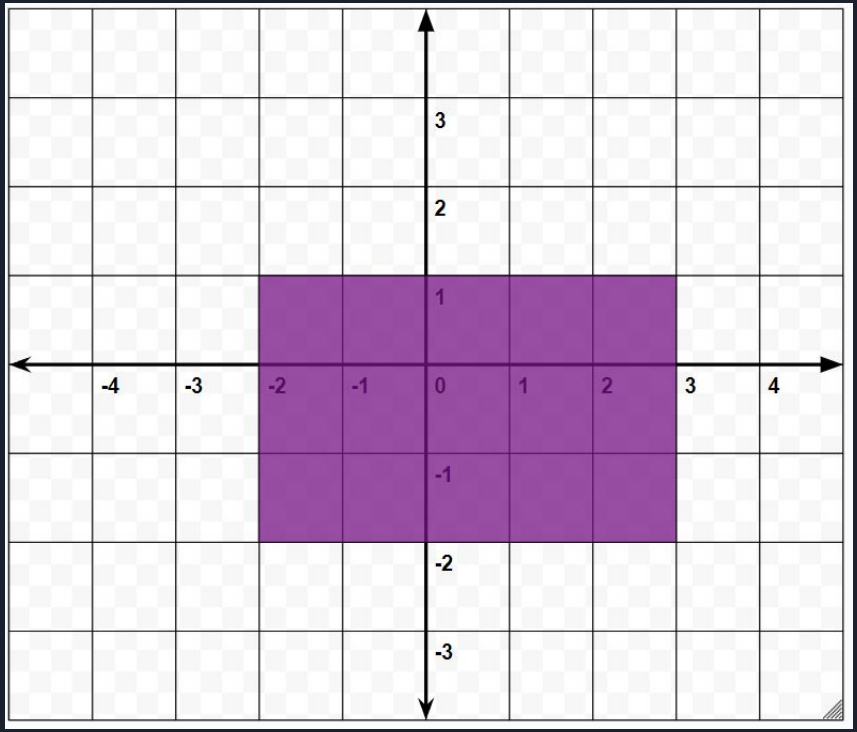
- Creating the background for the scene by setting a background image for the page
- Adding at least 5 images to the page
  - each image should also be positioned using the position and top/left/bottom/right properties
  - each image should also be resized using the width and/or height property If students don't end up finishing in class, they will finish it as Homework

# Resources

- https://www.w3schools.com/css/css_positioning.asp
- https://coursework.vschool.io/css-layout-and-positioning/
- https://www.forgotten-adventures.net/live-gallery/

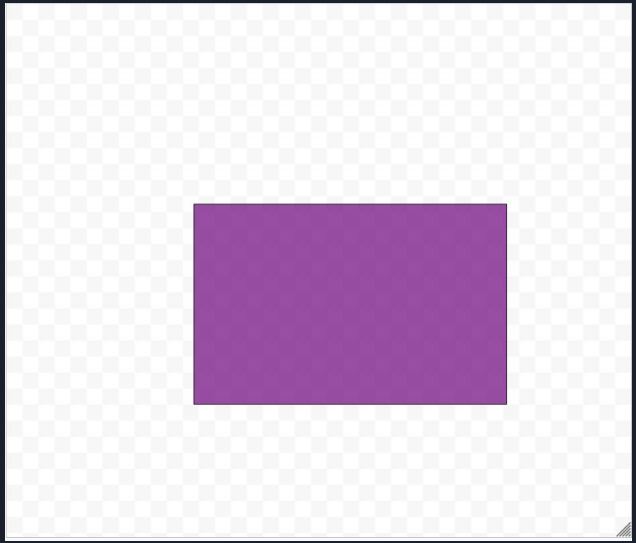
# Aim: How Can We Use CSS to Resize and Position HTML Elements?

Do Now: What are the coordinates of each corner of the rectangle in the image?



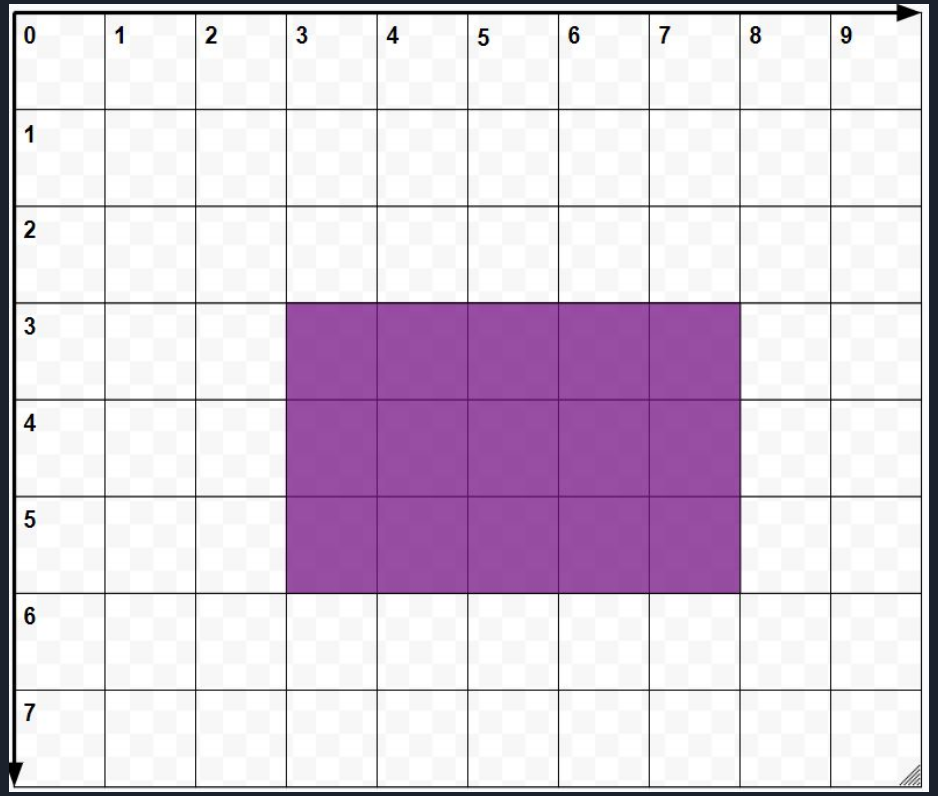
# Unseen Coordinate System

- For the Do Now, we could use the coordinate grid to help us locate the corners of the rectangle
- Our websites work using a coordinate system like in Math as well
- However, the coordinate system used in our websites differs from what you may be more familiar with from Math in two important ways
- The first big difference is that unlike with a Cartesian coordinate grid, we can't actually see the lines or numbers making up the grid when we browse the web
- However, just because we can't see the underlying grid, doesn't mean it's not there



This is the same rectangle as before, but now with a coordinate system that matches what's used by web browsers.

What would be the coordinates of each corner of the rectangle now, using this new coordinate system?



# The Top and Left Properties

- Much like how we can plot a point on a grid in Math given its coordinates, we can do something similar with our HTML elements using CSS
- The top and left CSS properties are analogous to the y and x coordinates in math respectively.
- Both CSS properties take numeric values, like with the font-size property from last lesson.
- There are many units that can be used for the top and left properties as well, but we'll stick to pixels (px) for now
- Although (0,0) is the top left corner in this coordinate system, it's still possible to use negative values for top and left.
- A negative value just means that part or all of that HTML element will be out of bounds

```
#photo {
    left:200px;
    top:400px;
}
```

# The Bottom and Right Properties

- Elements in HTML can also be positioned using the bottom and right CSS properties
- Sometimes it may make more sense to say something should be near the bottom of the screen instead of saying it should be ~700px from the top
- The bottom and right properties work nearly identically to the top and left properties, but use a different frame of reference
- It may be easy to see something like left:50px and interpret that as meaning move something 50px to the left
- Instead, think of it as meaning move something 50px **from** the left-most side
- Or, think of it as moving it in the opposite direction, left:50px means moving 50px to the **right**

# The Position Property

- The top, left, right, and/or bottom properties need to be combined with the position property for them to work properly
- Despite the name, the position property doesn't affect where elements are positioned, but rather how they are positioned, like choosing what set of rules to apply
- There are four values for the position property that we cover today,
- position:static;
- position:relative;
- position:absolute;
- position:fixed;
- We can demo and practice the difference between these values using this site: https://www.w3schools.com/css/css_positioning.asp

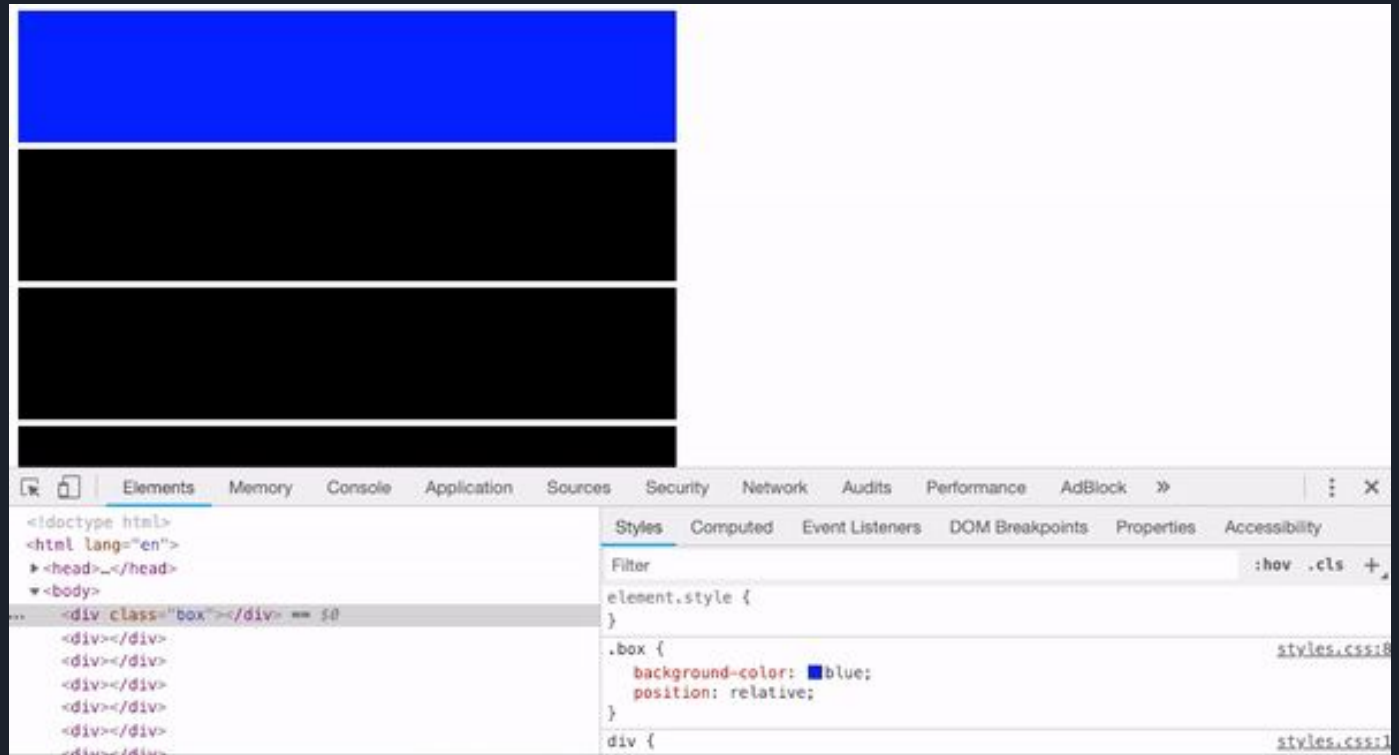
# position: static

- Static is actually the default value for most HTML elements
- An element with position:static actually isn’t affected by the top, left, bottom, or right properties
- Thus, an element with position:static tends to stay at their default position determined by the content and flow of the webpage

# position: relative

- An HTML element with position: relative actually behaves like an element with position: static as long as no value for top, left, bottom, or right properties are given
- However, if an element has position: relative and one of the above properties are applied, the element will move the corresponding distance from it's default position in a direction determined by the properties used
- Changing the position of an element with position: relative does not affect or move the other elements around it, even if it would overlap with something

# position: relative

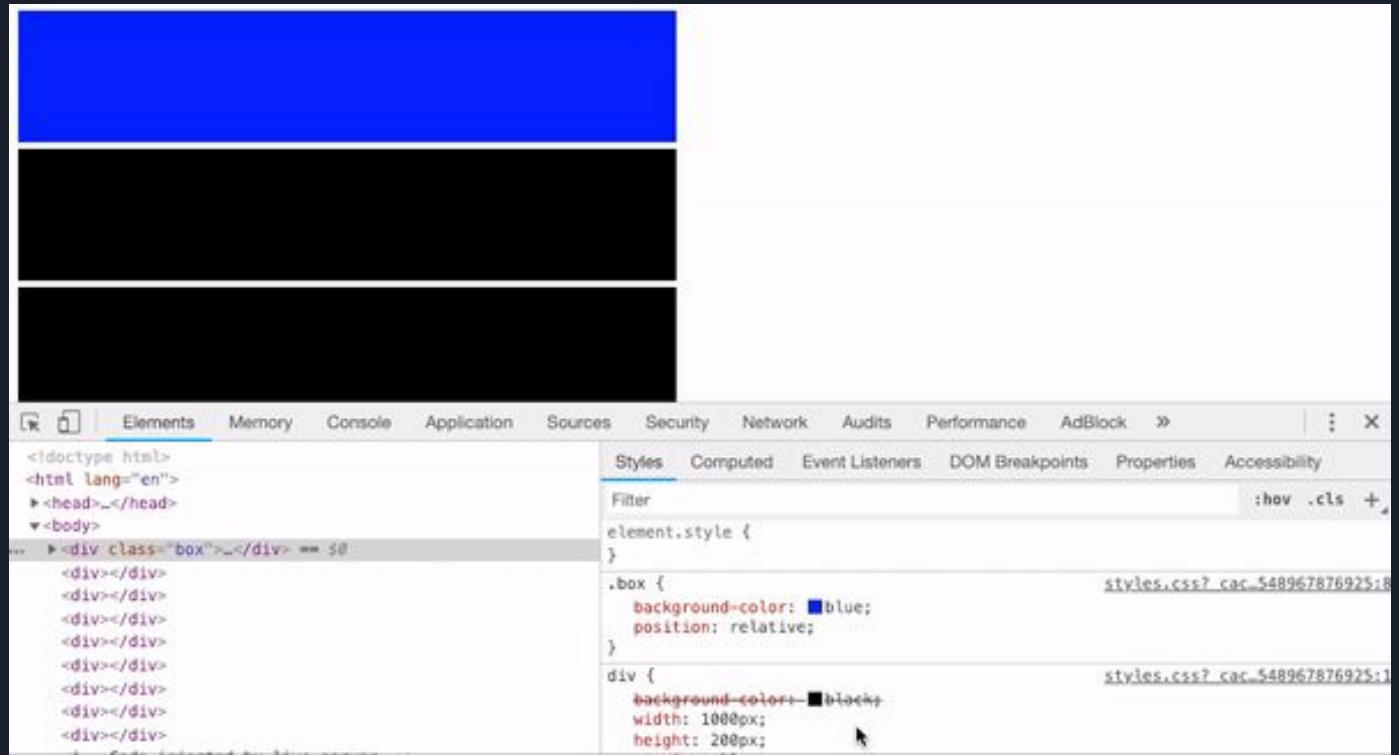


# position: absolute

- An HTML element with position: absolute ignores it's default position dictated by the flow of the web page
- Instead, elements with position: absolute are positioned with respect to the window
- That means that while the below code would not cause any change to an element with position: relative, it would position an element with position:absolute to the top-left corner of the page

```
img {
    top:0px;
    left:0px;
}
```

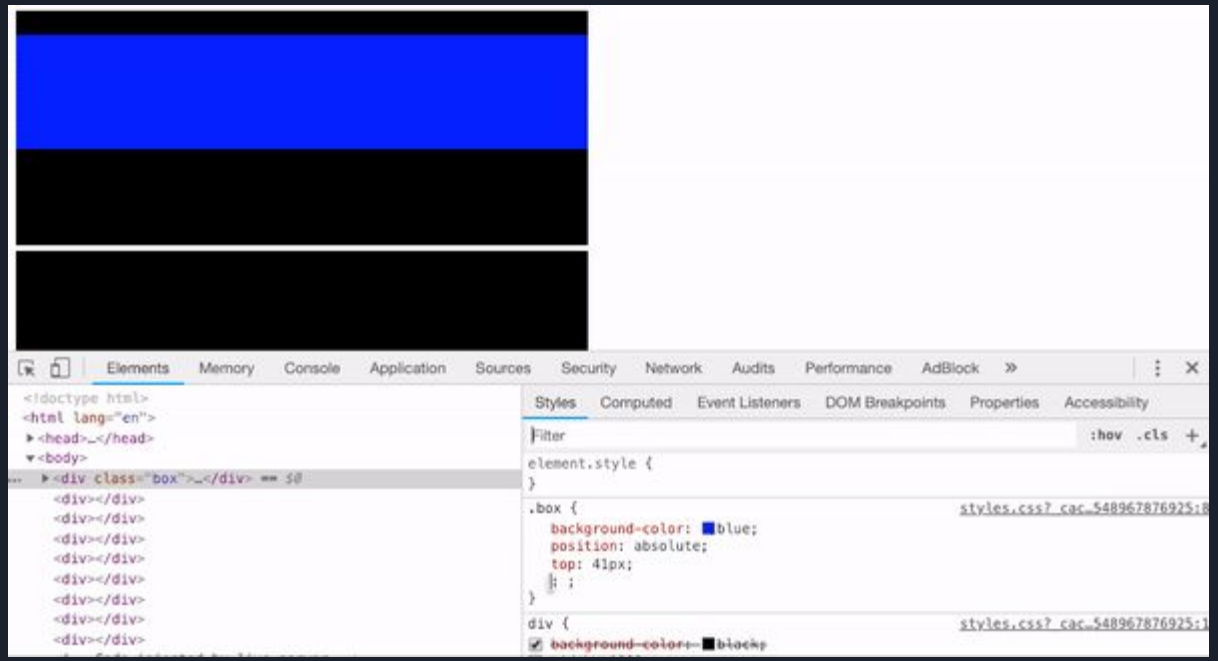
# position: absolute



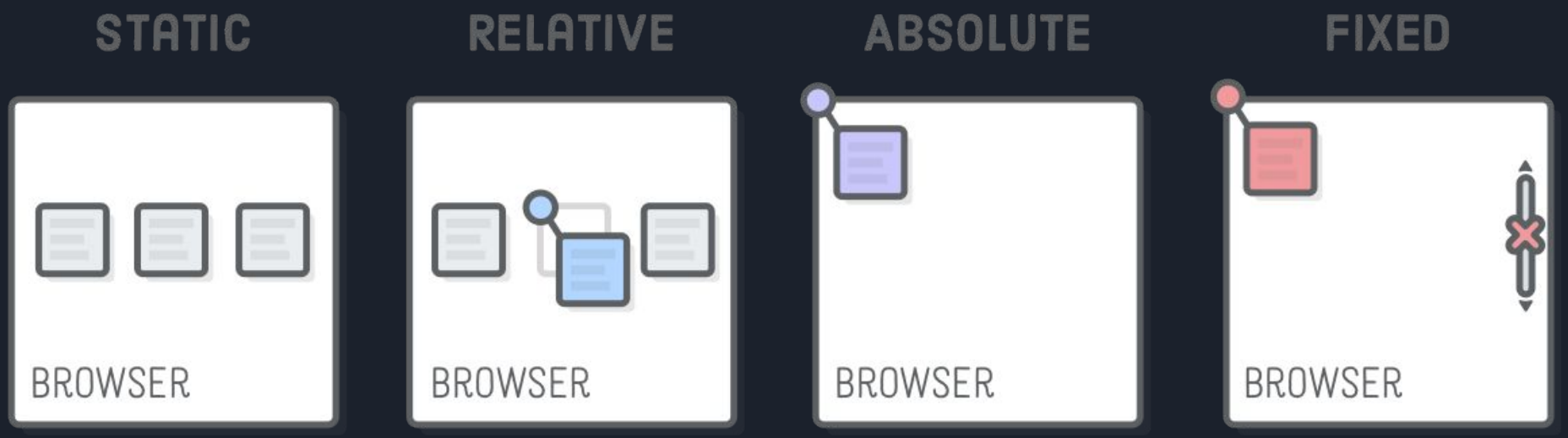
# position: fixed

- An HTML element with position: fixed behaves very similarly to an element with position:absolute;

- However fixed elements are “fixed” in place on the user’s screen, staying in the same position even if the user scrolls up or down



# Position Values Recap



# Width and Height properties

- We can change the size of HTML elements by using the width and height properties
- These properties work especially well with images
- Like with the top, left, bottom, and right properties, the width and height properties use numeric values and we usually use pixels (px) as the unit
- If we use just one of the width or height properties for an image, it will automatically scale the other dimension to maintain the same aspect ratio
- For example, if an image is originally 200px x 400px and we set the width to 600px using CSS, the height will be automatically set to 1200px
- However, if we specify both the width and height for an image, it will not maintain the image's original aspect ratio and may look stretched or distorted

```
img {
    width: 600px;
}
```

# Using a background image

- In a previous lesson, we learned how to change the background-color of an HTML element
- However, we are not just limited to using colors as backgrounds, we can use images as backgrounds as well
- We can set a background image for an HTML element by using the background-image property in CSS
- This is an example of code one might use to use a background image for their web page

```
body {

    background-image: url("images/ocean.png");

}
```

- Notice that the file path or URL for the desired background image needs to be put in quotes and put in between the parentheses of **url( )**

# Lesson 08:
# Containers in HTML

# Lesson Structure

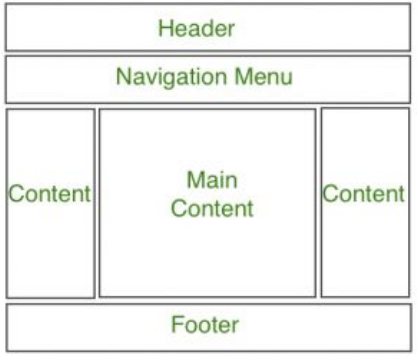
# Standards

- **9-12.IC.7**

# Aim: How Can We Use Divs To Organize Our Web Pages?

# Warm-Up (2-3 minutes)

Do Now: What websites can you think of that have a layout similar to the one depicted here?




Students will be given 2-3 minutes to write down their answers.

# Warm-Up Review (3-5 minutes)

After students are given some time to do the Do Now, we'll turn this into a classwide discussion, taking time to hear students' answers. This will be used to segue into the slides for today's lesson.

# Slide Presentation (20-25 minutes)

Present the slides provided in the materials

The slides for this lesson will cover

- div tag
- HTML Box Model
- padding property
- border property
- margin property
- display property
- block vs inline vs inline-block
- example/explanation of what students will be working on next

In addition to the slides, live demo

- divs with CSS vs divs without CSS
- different parts of the box model, especially border and margin
- an example of what the activity for today might look like

# Activity: Creating a Comic (Rest of Class)

For today's activity students will practice using divs by creating a 4 panel comic using HTML and CSS. Students assignments will need to meet these requirements (which will be in a readme in their starter code) but are welcome to go beyond them

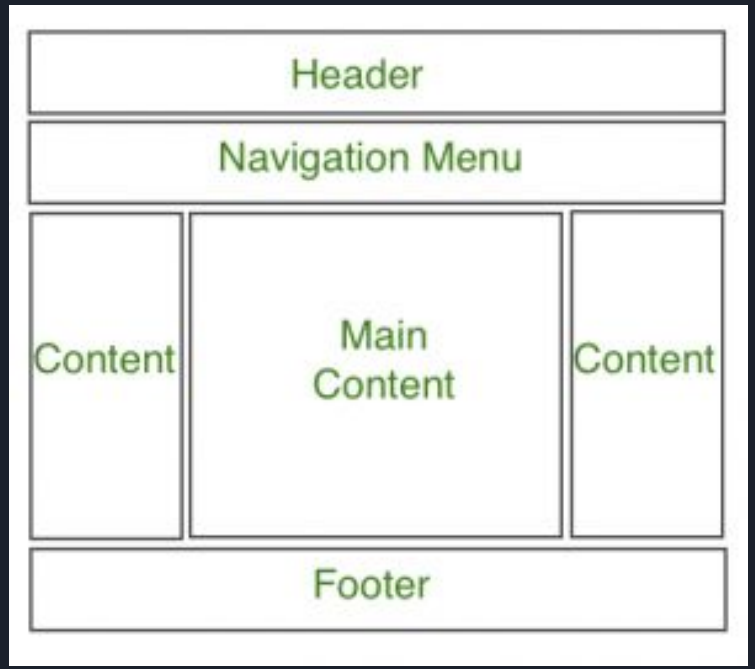
- Creating a div to act as a container for the comic itself
- Adding at least 4 divs inside the container to be the panels
- Style each panel using CSS, giving each one at least width,height,background-color/background-image, and border properties
- Inside each panel, include at least one image and at least one text tag
- Make sure to change at least one of the text, image, or image CSS properties between panels Students are free to use any school appropriate images they want for this project, but some images will also be provided for students who don't want to think too much about it or get right to coding (located in resources). If students don't end up finishing in class, they will finish it as Homework

# Resources

- https://www.spriters-resource.com/

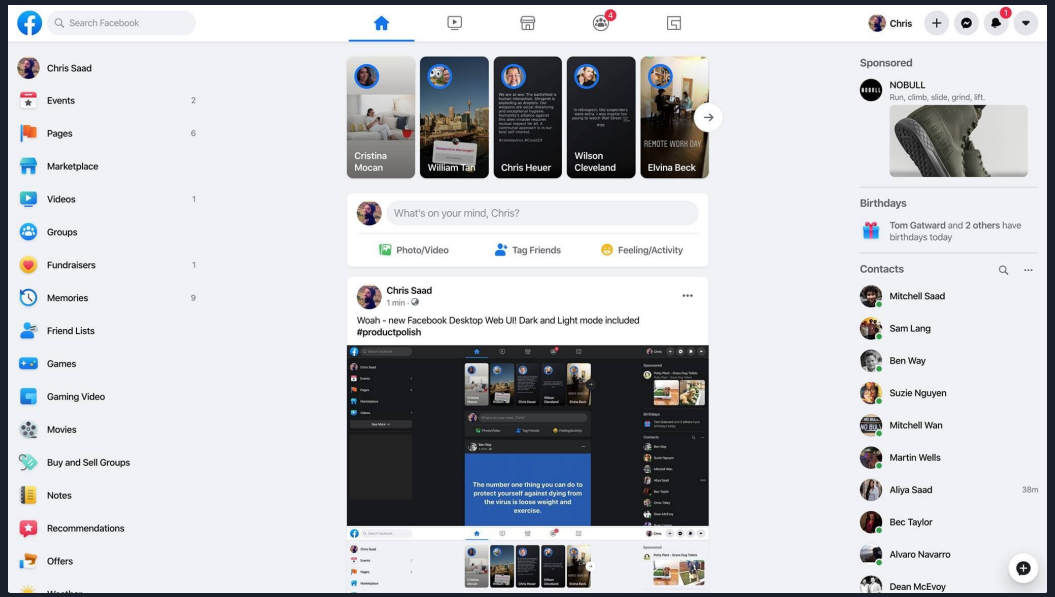
# Aim: How Can We Use Divs To Organize Our Web Pages?

Do Now: What websites can you think of that have a layout similar to the one depicted here?



# Discussing Website Layouts

- Websites are rarely just a single, continuous column of text, images, or other content
- Instead, websites are often divided up into different sections
- For example, on Facebook the middle column shows posts on your timeline, the left column shows the different parts of the site you can navigate to, and the right column shows the friends you can message



# The  tag

- There are a wide range of reasons why we might want to divide our web pages into different parts, or group similar or related things on a web page together
- Whatever the reason, we can do so using the  tag!
- The  tag goes in the body portion of out HTML document and its main purpose is to act as a container
- We can put nearly any content in between our  start and end tags, including other  tags
- However, it is important to note that the  tag by itself does nothing and wouldn't even be noticeable that it was used on a web page
- The true strength of the  tag lies in how it interacts with CSS

# No CSS Or Divs

```
<p>This sentence is not in a div</p>

<p>This sentence is also not in a div</p>
```

This sentence is not in a div

This sentence is also not in a div

# Divs without CSS

```html
<div>
  <p>This sentence is in one div</p>
</div>

  <div>
    <p>This sentence is in another div</p>
  </div>
```

This sentence is in one div

This sentence is in another div

# Divs with CSS

```
1 .demo {
2   background-color: green;
3   text-align:center;
4 }
```

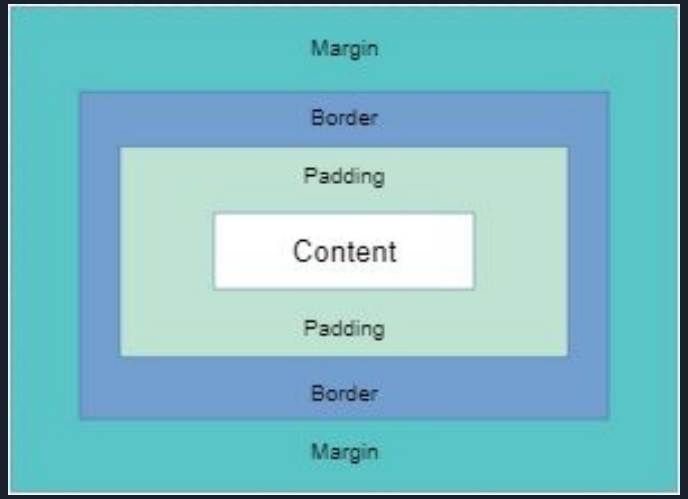
```
<div class='demo'>
  <p>This sentence is in one div</p>
</div>

  <div class='demo'>
    <p>This sentence is in another div</p>
  </div>
```

This sentence is in one div

This sentence is in another div

# The Box Model

- As previously mentioned, the strength of the  tag lies in how it can be used with CSS
- The Box Model is an important facet of CSS and can really help control the size, shape, and color of our divs
- Nearly every HTML element on a web page is actually made up of 4 concentric rectangles, as depicted below



# The Box Model: Content

- Content is the innermost box of the Box Model
- Although we haven't referred to it by name before, this is the box that we've been dealing with in class so far
- The content box is where the content of our HTML elements live, such as images for the <img> tag, text for the <p> tag, and the various things that might go inside a  tag
- When we set the width and/or height of an element using CSS, we're actually only setting the size of the content box

# The Box Model: Padding

- Padding lies in between the Content and Border boxes of the Box Model
- As the name suggests, padding is just extra space around the Content box
- Notably, if we set a background-color for an HTML element, the padding will be the same color
- We can change how much padding an element has by using the padding property
- Alternatively, we can specify how much padding to use on each side by using the padding-top, padding-left, padding-bottom, and padding-right properties
- Below are both valid ways to add padding to an element

```
div {
    padding: 5px;
}
```

```
div {
    padding-top:5px;
    padding-left:10px;
    padding-right:5px;
}
```

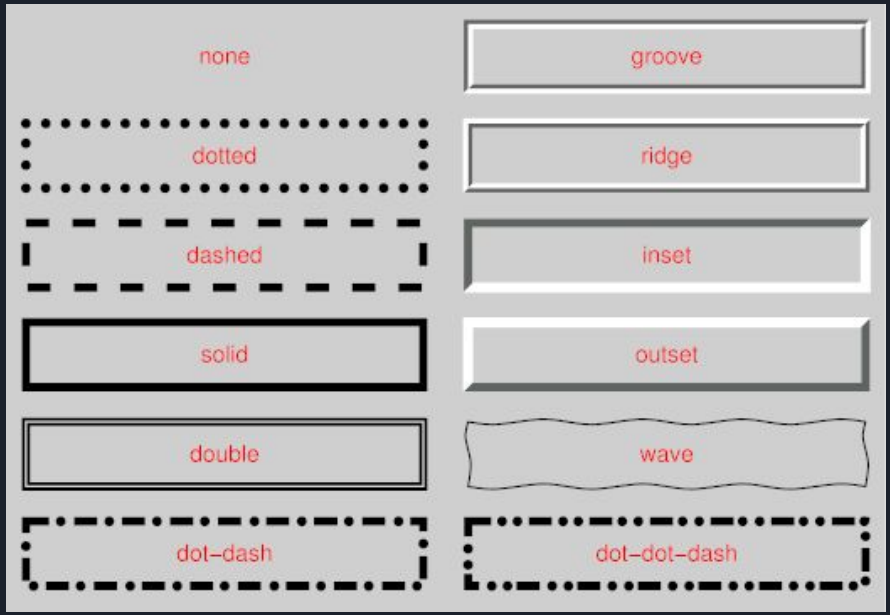
# The Box Model: Border

- Border lies in between the Padding and Margin boxes of the Box Model
- The Border box is the outermost visible box of the box model
- Unlike the Padding box, not only can we specify how thick the border should be, we can also set the color and style of the border
- The border property takes 3 values, size, style, and color
- The value for size works similarly to padding and we can also use color names or hex codes for the color value
- Some of the different styles of border we can use are shown on the next slide
- Also, similar to padding, we can set each side of the border independently using border-top, border-left, border-bottom, or border-right

# The Box Model: Border

🔥 CSS is the language we use to describe how HTML elements should be displayed.

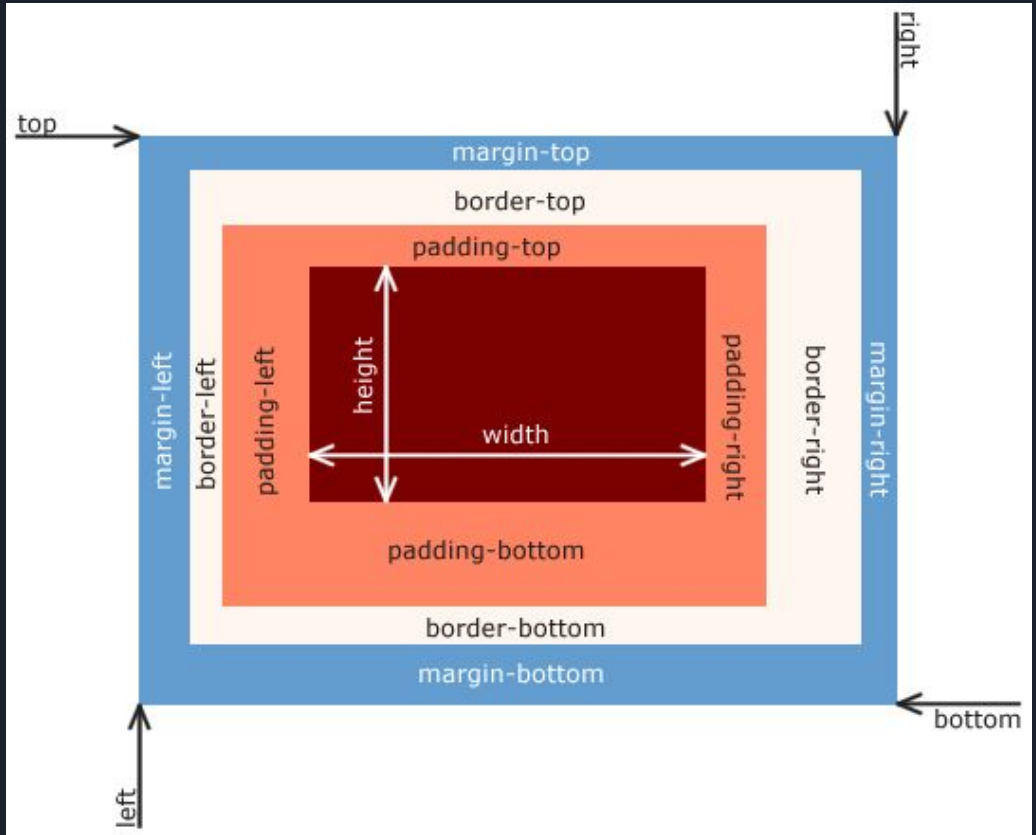
```
div {
  border: 4px solid purple;

}
```



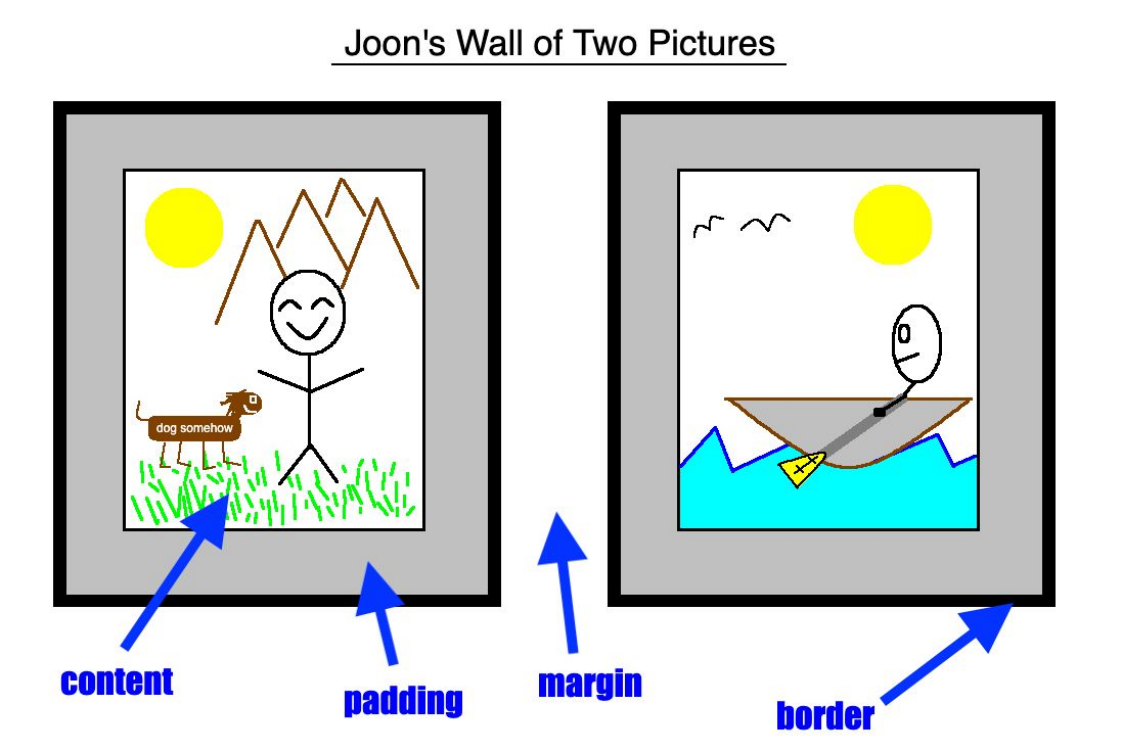
# The Box Model: Margin

- The margin is the outermost box of the box model
- Unlike the other 3 parts of the Box Model, the margin is invisible
- In essence, the margin controls how much space you want between your HTML elements, like needing to keep a 6ft distance around others
- We can use the margin property to control how much space we should have around an HTML element
- Also, similar to padding and border, we can set each side of the margin independently by using margin-top, margin-left, margin-bottom, or margin-right

# Box Model and Corresponding CSS properties



# Example of parts of the Box Model



# The Display Property

- The Box Model can help us control the size, shape, and color of a div (or other HTML elements)
- However, we also need to consider how we should multiple divs should be arranged on a page
- Theoretically, we could give each div position:absolute and position them on a web page manually, but that is not always the best choice and often harder than the alternative
- Instead, we can use the display CSS property to have a bit more control over how multiple HTML elements are arranged
- There are many possible values for the display property, but for today's lesson we will discuss only three
  - display: block
  - display: inline
  - display: inline-block

# display: block

- When an HTML element has display:block, it stretches to take up as much width as possible
- In addition, elements with a block value for display will always start on a new line
- Most of the HTML tags that we've learned about so far actually have a value of block by default, including the  tag
- display: block is useful if you want to arrange your HTML elements vertically

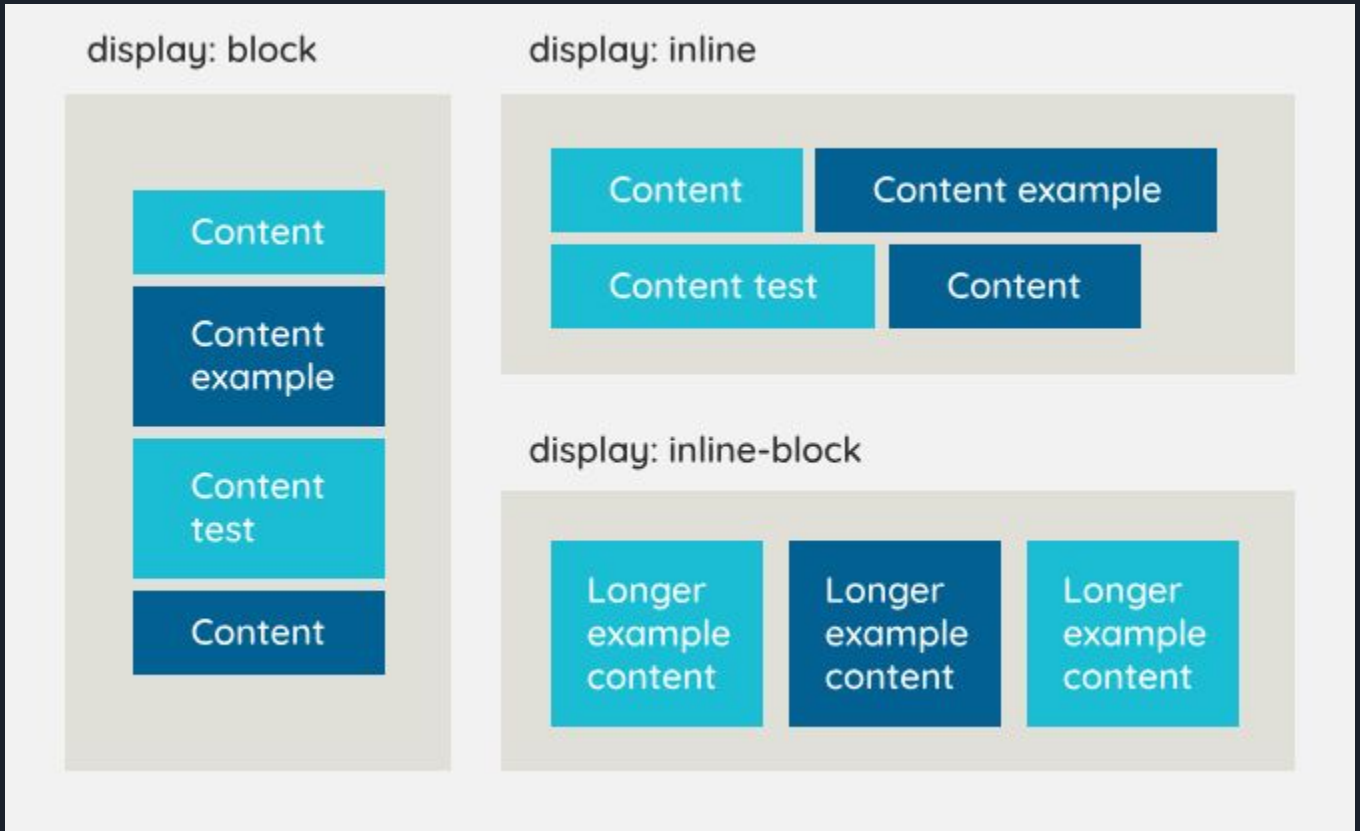
# display: inline

- When an HTML element has display:inline, it only takes up as much space as needed
- In addition, elements with an inline-value for display do not start on a new line
- The  tag and <a> tag are examples of tags that have a value of inline by default
- display: inline is useful if you want to arrange your HTML elements horizontally, but there is another option as well…

# display: inline-block

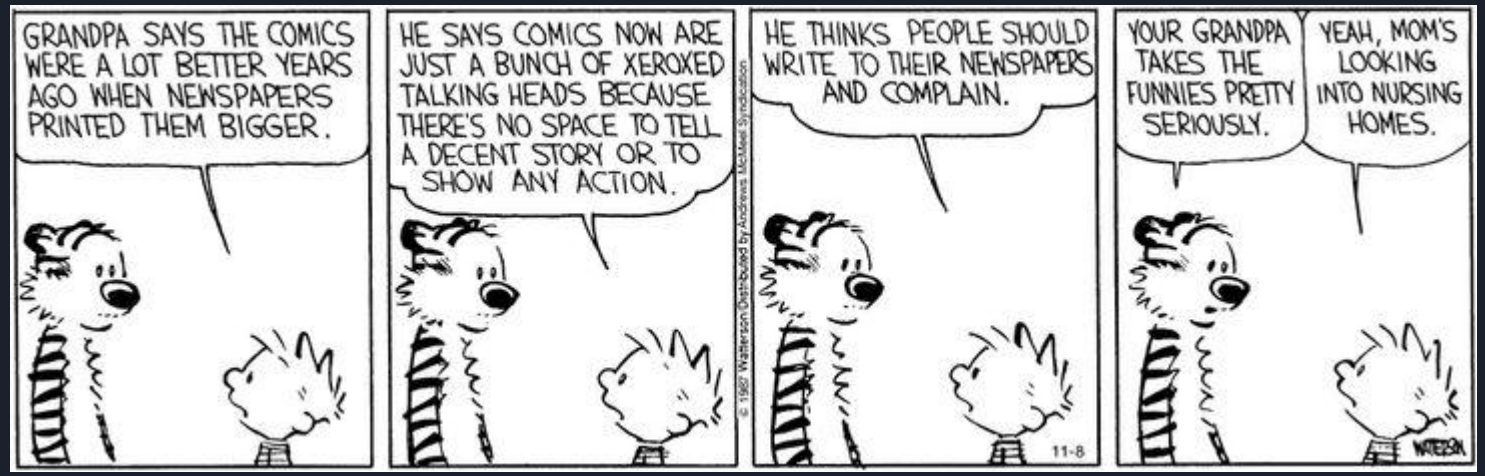
- An HTML element with display: inline-block behaves similarly to inline elements
- However, while inline elements only take up as much space as needed, inline-block elements can have their size set by the width and height property
- display: inline-block is useful if you want to arrange your HTML elements horizontally and have more control over their appearance

# The Display Property Visualized

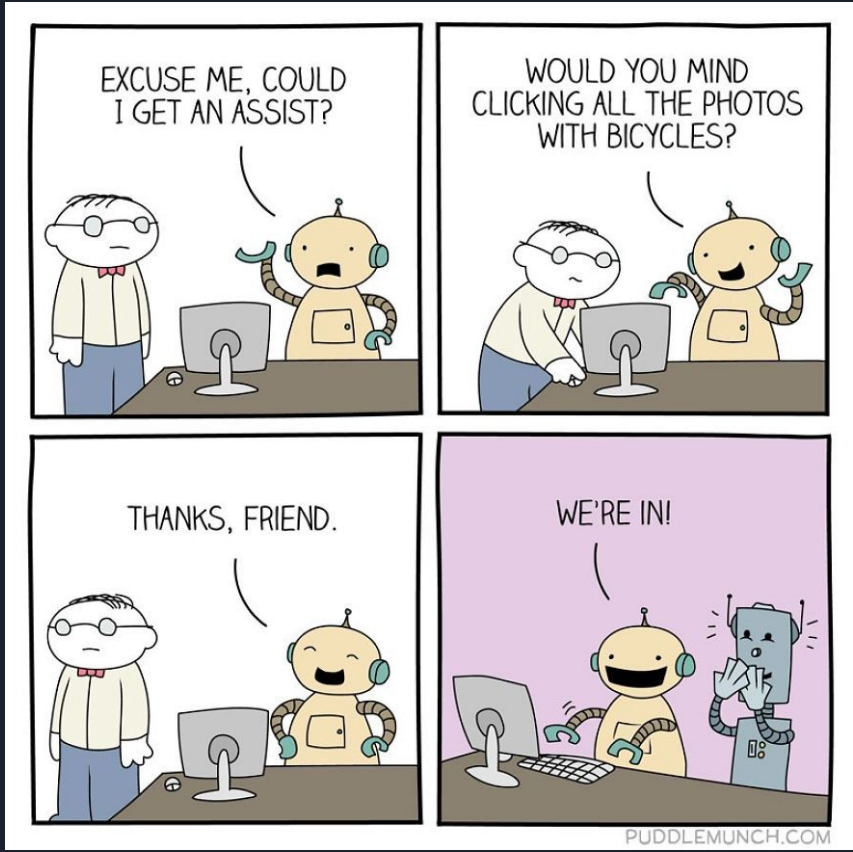


When designing your comic, keep in mind that they can be…

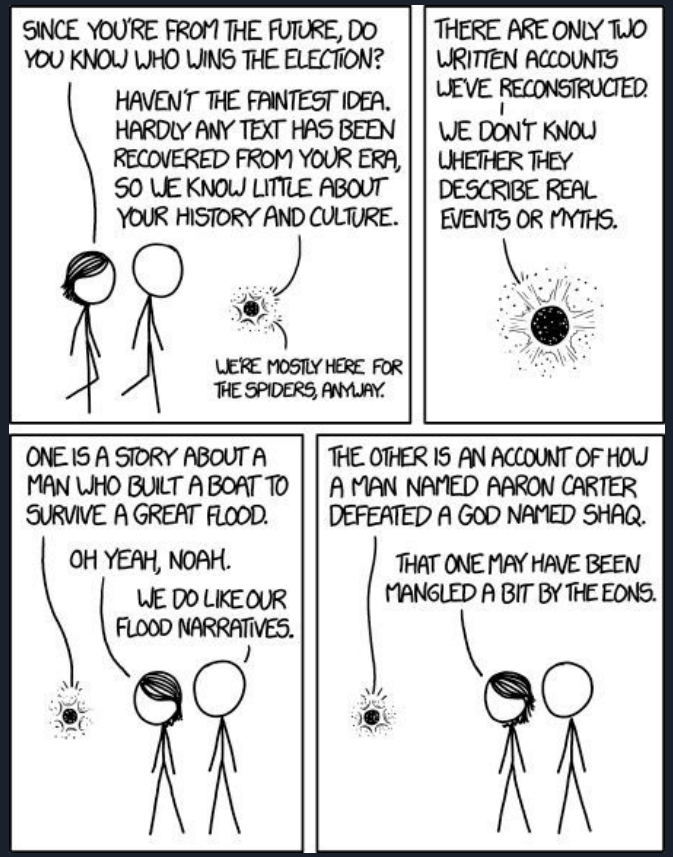
# Horizontal


GRANDPA SAYS THE COMICS WERE A LOT BETTER YEARS AGO WHEN NEWSPAPERS PRINTED THEM BIGGER.
HE SAYS COMICS NOW ARE JUST A BUNCH OF XEROXED TALKING HEADS BECAUSE THERE'S NO SPACE TO TELL A DECENT STORY OR TO SHOW ANY ACTION.
© 1987 Watterson Distributed by Andrews McMeel Syndication
HE THINKS PEOPLE SHOULD WRITE TO THEIR NEWSPAPERS AND COMPLAIN.
11-8
YOUR GRANDPA TAKES THE FUNNIES PRETTY SERIOUSLY.
YEAH, MOM'S LOOKING INTO NURSING HOMES.
WATTERSON

# 2x2


EXCUSE ME, COULD I GET AN ASSIST?
WOULD YOU MIND CLICKING ALL THE PHOTOS WITH BICYCLES?
THANKS, FRIEND.
WE'RE IN!
PUDDLEMUNCH.COM

# Or Even Unequal in Size



# Lesson 09-11: Working on Personal Website Project

## Lesson Structure
### Aim:  Working on Personal Website Project

### Assignment Details
Students will spend 3 days in class (and time out of class if they wish) working on their Personal Website Projects. This will be an ongoing project throughout the course that they will revist at the end of each unit, with this being their first time working on it.

Students will see examples of what their websites could potentially be (examples shown in Resources below and would realistically include prior student work as well if it existed). While they won't be able to recreate all the examples shown right away, this will maybe inspire them to go in certain directions with their websites. Since I don't expect students to know exactly what they will want to do for their websites by this point, the assignment will instead focus on getting them started with an About Me page, A Portfolio page containing links to their previous works, and a home page that for now can be whatever they want it to (the home page at this point is more for navigational purposes)

In additon, students will be expected to turn in a journal of sorts to Google Classroom at the end of the week. The journal doesn't have any specified length, but students should write about their process so far making their websites, such as why they made the design choices they did, what troubles did they encounter, what did they realize they want to add to their website now, etc.

When grading this assignment, I will be checking for
- If the basic structure tags are properly used for each page
- If images are used for the About Me page (it doesn't have to be them, but it should be something they can relate to)
- There are links to go from the home page, about me page, and portfolio page on all three pages
- The portfolio page contains links to the assignments they worked on so far, with each assignment being in its own div
- Students make use of CSS throughout the site, including
    - colors
    - text-styling
    - use of positioning or the display property

### Resources
- https://blackgirlbytes.dev/
- https://www.nikitaghodke.com/resources
- https://emilejb.github.io/
- https://emilejb.github.io/compass/