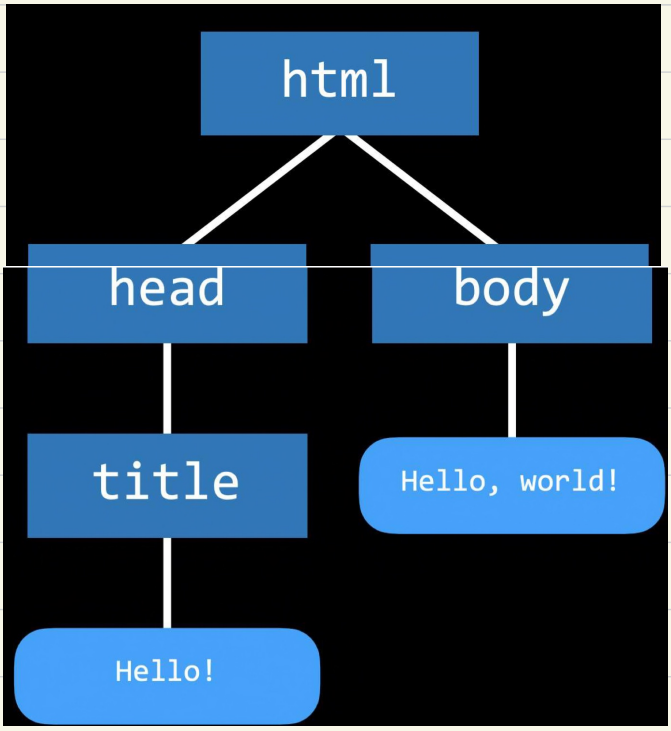
Lecture 1 - HTML, CSS

Part A - HTML5

① HTML 5 (Hypertext Markup Language) is a markup language that defines the structure of a web page. It's interpreted by web browser in order to display content.

② Document Object Model (DOM)



③ HTML 5 elements

- There are many HTML elements you may want to use to customize your page, including headings, lists, and bolded sections. In this next example, we'll see a few of these in action.
- One more thing to note: `<!-- -->` gives us a comment in HTML, so we'll use that below to explain some of the elements.

```
<!DOCTYPE html> ← indicate HTML 5
<html lang="en">← primary language is english
    <head>
        <title>HTML Elements</title>
    </head>
    <body>
        <!-- We can create headings using h1 through h6 as tags. -->
        <h1>A Large Heading</h1>
        <h2>A Smaller Heading</h2>
        <h6>The Smallest Heading</h6>

        <!-- The strong and i tags give us bold and italics respectively. -->
        A <strong>bold</strong> word and an <i>italicized</i> word!

        <!-- We can link to another page (such as cs50's page) using a. -->
        View the <a href="https://cs50.harvard.edu/">CS50 Website</a>!
                 (hyper reference)
        <!-- We used ul for an unordered list and ol for an ordered one. both ordered and unordered lists contain li, or list
        An unordered list: (in dot format)
        <ul>
            <li>foo</li>
            <li>bar</li>
            <li>baz</li>
```

we can link other html as well
e.g. "image.html"

③

```
    </ul>
    An ordered list:
    <ol>
        <li>foo</li>
        <li>bar</li>
        <li>baz</li>
    </ol>

    <!-- Images require a src attribute, which can be either the path to a file on your computer or the link to an image o
    An image:
    <img src="../../images/duck.jpeg" alt="Rubber Duck Picture">
    <!-- We can also see above that for some elements that don't contain other ones, closing tags are not necessary. -->

    <!-- Here, we use a br tag to add white space to the page. -->
    <br/> <br/>

    <!-- A few different tags are necessary to create a table. -->
    <table>
        <thead>
            <th>Ocean</th>
            <th>Average Depth</th>
            <th>Maximum Depth</th>
        </thead>
        <tbody>
            <tr>
                <td>Pacific</td>
                <td>4280 m</td>
                <td>10911 m</td>
            </tr>
            <tr>
                <td>Atlantic</td>
                <td>3646 m</td>
                <td>8486 m</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

(with 1,2,3...)

image is self-closing

"blank row"

constructed row by row

head of the table

"table head"

"table row"

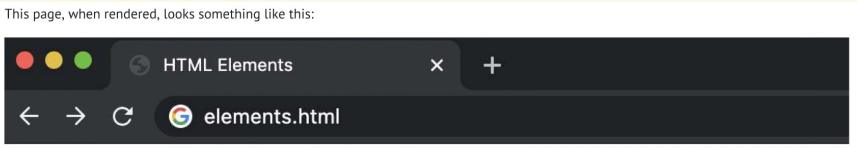
"table data"

closing tags

④ output

output in browsers

This page, when rendered, looks something like this:



A Large Heading

A Smaller Heading

The Smallest Heading

A **bold** word and an *italicized* word! View the CS50 Website! An unordered list:

- foo
- bar
- baz

An ordered list:

1. foo
2. bar
3. baz

An image: 

Ocean	Average Depth	Maximum Depth
Pacific	4280 m	10911 m
Atlantic	3646 m	8486 m

⑤ where to look up?

- In case you're worried about it, know that you'll never have to memorize these elements. It's very easy to simply search something like "image in HTML" to find the `img` tag. One resource that's especially helpful for learning about these elements is W3 Schools (https://www.w3schools.com/html/html_elements.asp).

⑥ Forms: allow users to input information

<input ...> is self-closing

Forms

- Another set of elements that is really important when creating a website is how to collect information from users. You can allow users to enter information using an HTML form, which can contain several different types of input. Later in the course, we'll learn about how to handle information once a form has been submitted.
- Just as with other HTML elements, there's no need to memorize these, and W3 Schools is a great resource for learning about them!

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Forms</title>
</head>
<body>
    <form>
        <input type="text" placeholder="First Name" name="first">
        <input type="password" placeholder="Password" name="password">
        <div>
            Favorite Color:
            <input name="color" type="radio" value="blue"> Blue
            <input name="color" type="radio" value="green"> Green
            <input name="color" type="radio" value="yellow"> Yellow
            <input name="color" type="radio" value="red"> Red
        </div>
        <input type="submit">
    </form>
</body>
</html>
```

self-closing

user ref. (opt.)

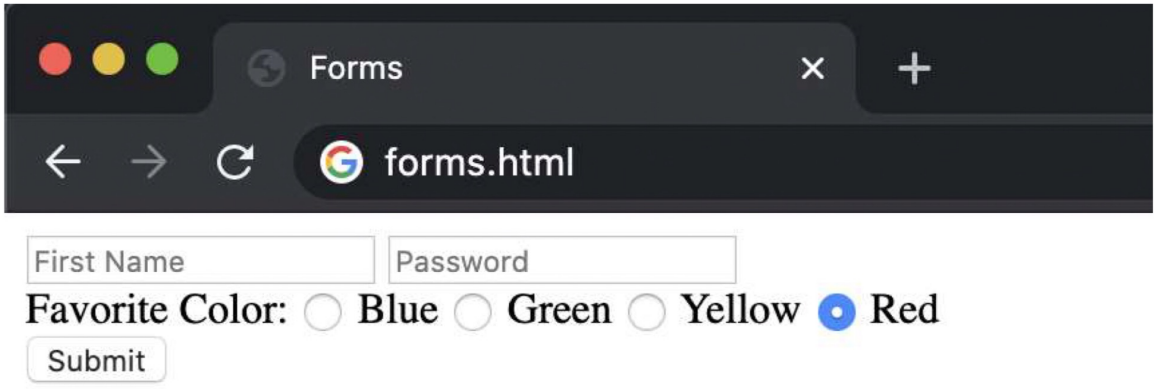
internal ref.

same line

"division" (new nested section)

this type hide inputs

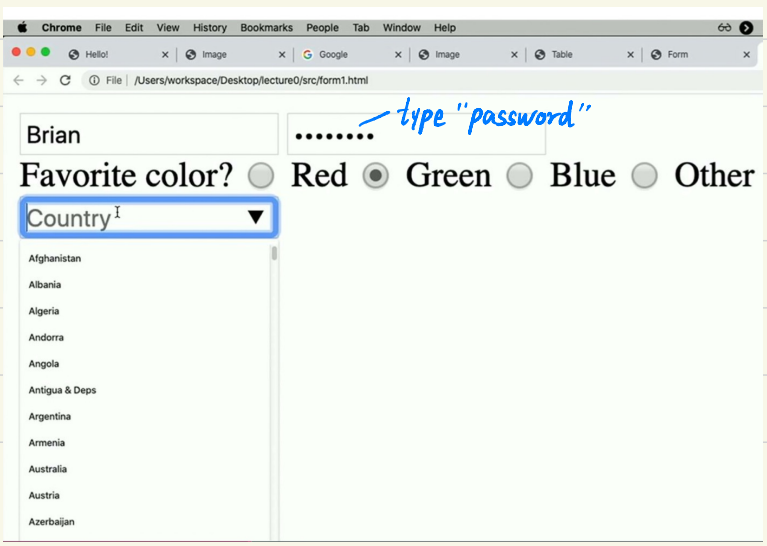
same line



⑥′ Drop-down menu (spinner) within a form

<datalist ...> ... ... </datalist>

```
<div>
    <input name="country" list="countries" placeholder="Country">
    <datalist id="countries">
        <option value="Afghanistan">
        <option value="Albania">
        <option value="Algeria">
        <option value="Andorra">
        <option value="Angola">
        <option value="Antigua & Deps">
        <option value="Argentina">
        <option value="Armenia">
        <option value="Australia">
        ...
    </datalist>
</div>
```



Part B - CSS 3

① CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

(used to customize the appearance of a website)

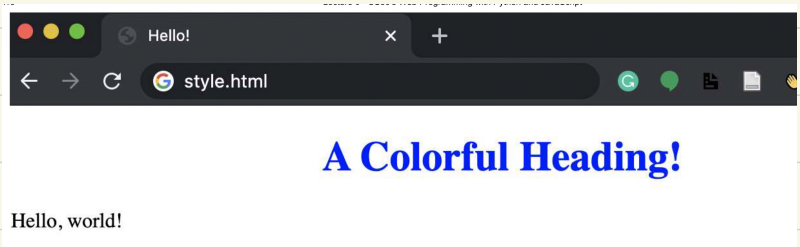
② Colorful heading

<hx style= ...>... </hx>

(1-line method)

- While we're just getting, started, we can add a style attribute to any HTML element in order to apply some CSS to it.
- We change style by altering the CSS properties of an element, writing something like `color: blue` or `text-align: center`
- In this example below, we make a slight change to our very first file to give it a colorful heading:

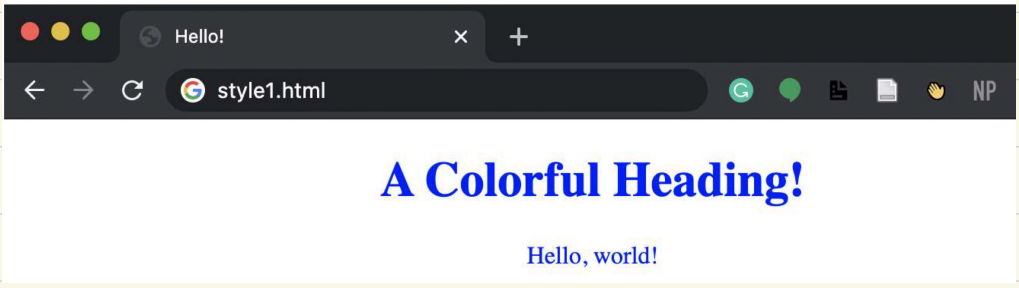
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hello!</title>
    </head>
    <body>
        <h1 style="color: blue; text-align: center;">A Colorful Heading!</h1>
        Hello, world!
    </body>
<html>
```



③ Styling an outer element will let all inner elements take that style (indirectly)

- If we style an outer element, all of the inner elements automatically take on that style. We can see this if we move the styling we just applied from the header tag to the body tag:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hello!</title>
    </head>
    <body style="color: blue; text-align: center;">
        <h1 >A Colorful Heading!</h1>
        Hello, world!
    </body>
<html>
```



④ more fashionable ways: 4.1 add a <style> ... </style> tag for each kind of elements

in <head>

- One way of doing this is to add your styling between `<style>` tags in the `head`. Inside these tags, we write which types of elements we want to be style, and the styling we wish to apply to them. For example:

```
<html lang="en">
<!DOCTYPE html>
<head>
    <title>Hello!</title>
    <style>
        h1 {
            color: blue;
            text-align: center;
        }
    </style>
</head>
<body>
    <h1 >A Colorful Heading!</h1>
    Hello, world!
</body>
</html>
```

curly bracket!

for each h1 tag

4.2 include a <link ... > (self-closing) tag in <head> with a link to styles.css file

- Another way is to include in a `<link>` element in your `head` with a link to a styles.css file that contains some styling. This means the HTML file would look like:

```
<html lang="en">
<!DOCTYPE html>
<head>
    <title>Hello!</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1 >A Colorful Heading!</h1>
    Hello, world!
</body>
</html>
```

"related"

hyperref.

And our file called `styles.css` would look like:

styles.css

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

⑤ CSS properties

- There are far too many CSS properties to go over here, but just like HTML elements, it's typically easy to Google something along the lines of "change font to blue CSS" to get the result. Some of the most common ones though are:
  - `color`: the color of text
  - `text-align`: where elements are placed on the page
  - `background-color`: can be set to any color
  - `width`: in pixels or percent of a page
  - `height`: in pixels or percent of a page
  - `padding`: how much space should be left inside an element
  - `margin`: how much space should be left outside an element
  - `font-family`: type of font for text on page
  - `font-size`: in pixels
  - `border`: size type (solid, dashed, etc) color
- Let's use some of what we just learned to improve upon our oceans table from above. Here's some HTML to start us off:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Nicer Table</title>
    </head>
```

⑤' (Cont'd)

```
    <body>
        <table>
            <thead>
                <th>Ocean</th>
                <th>Average Depth</th>
                <th>Maximum Depth</th>
            </thead>
            <tbody>
                <tr>
                    <td>Pacific</td>
                    <td>4280 m</td>
                    <td>10911 m</td>
                </tr>
                <tr>
                    <td>Atlantic</td>
                    <td>3646 m</td>
                    <td>8486 m</td>
                </tr>
            </tbody>
        </table>
    </body>
<html>
```

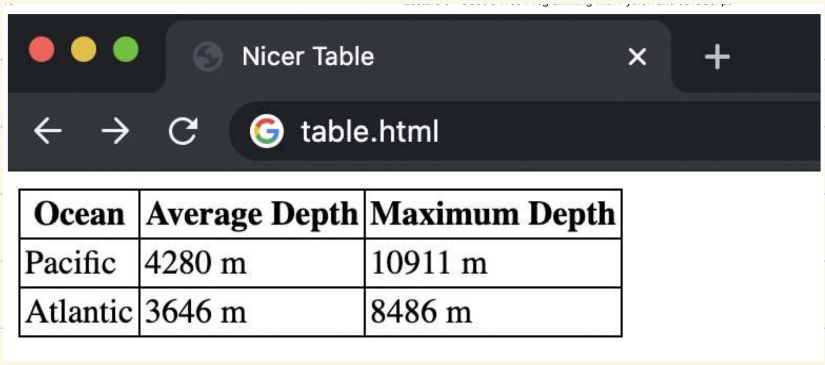
- The above looks a lot like what we had before, but now, either by including a `style` tag or a `link` to a stylesheet in the head element, we add the following css:

```
table {
    border: 1px solid black;
    border-collapse: collapse;
}

td {
    border: 1px solid black;
    padding: 2px;
}

th {
    border: 1px solid black;
    padding: 2px;
}
```

Which leaves us with this nicer-looking table:



Ocean	Average Depth	Maximum Depth
Pacific	4280 m	10911 m
Atlantic	3646 m	8486 m

⑥ Compress duplicated styling codes

- You may already be thinking that there's some needless repetition in our CSS at the moment, as `td` and `th` have the same styling. We can (and should) condense this down to the following code, using a comma to show the styling should apply to more than one element type.

```
table {
    border: 1px solid black;
    border-collapse: collapse;
}

td, th {
    border: 1px solid black;
    padding: 2px;
}
```

curly bracket

same style codes for the 2 types (for each td/th tag)

⑦ Font related styles:

```
xxx {
    font-family: Arial, sans-serif;
    font-size: 28px;
    font-weight: bold;
}
```

the second is a backup font-family

⑧ "border-collapse: collapse" can collapse duplicated borders

(solve the problem caused by ⑥'s styling)

"padding: <No.>px;" will give space inside a table cell

⑨ 3 styling targets

(CSS selectors)

- This is a good introduction into what are known as CSS selectors (https://www.w3schools.com/cssref/css_selectors.asp). There are many ways to determine which HTML elements you are styling, some of which we'll mention here:

9.1 - **element type:** this is what we've been doing so far: styling all elements of the same type.

9.2 - **id:** Another option is to give our HTML elements an id like so: `<h1 id="first-header">Hello!</h1>` and then applying styling using `#first-header{...}` using the hashtag to show that we're searching by id. Importantly, no two elements can have the same id, and no element can have more than one id.

"#"

id is 1 and only 1

9.3 - **class:** This is similar to id, but a class can be shared by more than one element, and a single element can have more than one class. We add classes to an HTML element like this: `<h1 class="page-text muted">Hello!</h1>` (note that we just added two classes to the element: `page-text` and `muted`). We then style based on class using a period instead of a hashtag: `.muted {...}`

period

2 classes once separated by space

⑩ CSS collision: styling precedence

but blue based on its id? CSS has a specificity order that goes:

1. In-line styling
2. id
3. class
4. element type

⑪ more CSS selectors

a, b	Multiple Element Selector
a b	Descendant Selector
a > b	Child Selector
a + b	Adjacent Sibling Selector
<t>[a=b]	Attribute Selector
a:b	Pseudoclass Selector
a::b	Pseudoelement Selector

all (grand) children

only immediate children

⑫ example of "Descendant Selector" select children, grandchildren...

Descendant Selector: Here, we use the descendant selector to only apply styling to list items found within an unordered list:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Using Selectors</title>
        <style>
            ul li {
                color: blue;
```

"ul > li" can also work here because only 1 immediate child

1. foo
2. bar
   - hello
   - goodbye
   - hello
3. baz

```
            }
        </style>
    </head>
    <body>
        <ol>
            <li>foo</li>
            <li> bar
                <ul>
                    <li>hello</li>
                    <li>goodbye</li>
                    <li>hello</li>
                </ul>
            </li>
            <li>baz</li>
        </ol>

    </body>
</html>
```

(13) example of "attribute selector"

Attributes as Selectors: We can also narrow down our selection based on the attributes we assign to HTML elements using brackets. For example, in the following list of links, we choose to only make the link to Amazon red:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Using Selectors</title>
        <style>
            a[href="https://www.amazon.com/"] {
                color: red;
            }
        </style>
    </head>
    <body>
        <ol>
            <li><a href="https://www.google.com/">Google</a></li>
            <li><a href="https://www.amazon.com/">Amazon</a> </li>
            <li><a href="https://www.facebook.com/">Facebook</a></li>
        </ol>

    </body>
<html>
```

attribute of the tag <a>

1. Google
2. Amazon
3. Facebook

(14) CSS pseudoclasses

(https://www.w3schools.com/css/css_pseudo_classes.asp), which provides additional styling during special circumstances. We write this by adding a colon after our selector, and then adding the circumstance after that colon.

- In the case of the button, we would add `:hover` to the button selector to specify the design only when hovering:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Pseudoclasses</title>
        <style>
            button {
                background-color: red;
                width: 200px;
                height: 50px;
                font-size: 24px;
            }

            button:hover {
                background-color: green;
            }
        </style>
    </head>
    <body>
        <button>Button 1</button>
        <button>Button 2</button>
        <button>Button 3</button>

    </body>
<html>
```

Button 1 Button 2 Button 3

the buttons become green when the cursor hovers over them

Part C - CSS Responsive Design (phone/tablet view)

① why Responsive Design ? & "Viewport" method (add the below line in <head>)

Responsive Design

- Today, many people view websites on devices other than computers, such as smartphones and tablets. It's important to make sure your website is readable to people on all devices.
- One way we can achieve this is through knowledge of the **viewport**. The viewport is the part of the screen that is actually visible to the user at any given time. By default, many webpages assume that the viewport is the same on any device, which is what leads to many sites (especially older ones) being difficult to interact with on mobile devices.
- One simple way to improve the appearance of a site on a mobile device is to add the following line in the head of our HTML files. This line tells the mobile device to use a viewport that is the same width as that of the device you're using rather than a much larger one.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

② another method: "Media Queries" example with "@media"

- Another way we can deal with different devices is through media queries (https://www.w3schools.com/cssref/css3_pr_mediaquery.asp). Media queries are ways of changing the style of a page based on how the page is being viewed.
- For an example of a media query, let's try to simply change the color of the screen when it shrinks down to a certain size. We signal a media query by typing `@media` followed by the type of query in parentheses:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Screen Size</title>
        <style>
            @media (min-width: 600px) {
                body {
                    background-color: red;
                }
            }

            @media (max-width: 599px) {
                body {
                    background-color: blue;
                }
            }
        </style>
    </head>
    <body>
        <h1>Welcome to the page!</h1>
    </body>
</html>
```

>= 600px → red

<= 599px → blue

③ a third method: "flexbox" (latest version CSS) solve "overflow" issue properly

- Another way to deal with differing screen size is using a new CSS attribute known as a flexbox (https://www.w3schools.com/css/css3_flexbox.asp). This allows us to easily have elements wrap around to the next line if they don't fit horizontally. We do this by putting all of our elements in a `div` that we'll call our container. We then add some styling to that div specifying that we want to use a flexbox display for the elements inside of it. We've also added some additional styling to the inner divs to better illustrate the wrapping that's occuring here.

display: flex;

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Screen Size</title>
        <style>
            #container {
                display: flex;
                flex-wrap: wrap;
            }

            #container > div {
                background-color: green;
                font-size: 20px;
                margin: 20px;
                padding: 20px;
                width: 200px;
            }
        </style>
    </head>
    <body>
        <div id="container">
            <div>Some text 1!</div>
            <div>Some text 2!</div>
            <div>Some text 3!</div>
            <div>Some text 4!</div>
            <div>Some text 5!</div>
            <div>Some text 6!</div>
            <div>Some text 7!</div>
            <div>Some text 8!</div>
            <div>Some text 9!</div>
            <div>Some text 10!</div>
            <div>Some text 11!</div>
            <div>Some text 12!</div>
        </div>
    </body>
</html>
```

] flexbox

want to adapt responsively

③' (Cont'd) output:

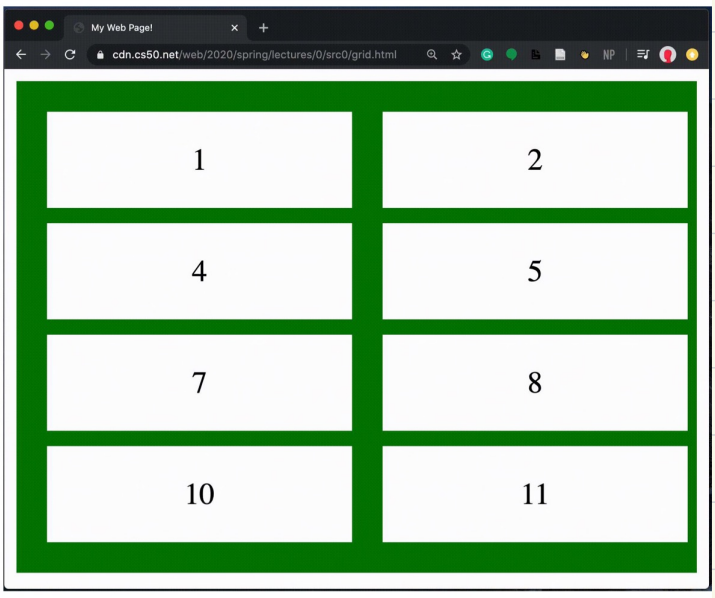
④ a fourth method "HTML grid"

display: grid;

- Another popular way of styling a page is using an HTML grid (https://www.w3schools.com/css/css_grid.asp). In this grid, we can specify style attributes such as column widths and gaps between columns and rows, as demonstrated below. Note that when we specify column widths, we say the third one is `auto`, meaning it should fill the rest of the page.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>My Web Page!</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            .grid {
                background-color: green;
                display: grid;
                padding: 20px;
                grid-column-gap: 20px;
                grid-row-gap: 10px;
                grid-template-columns: 200px 200px auto;
            }

            .grid-item {
                background-color: white;
                font-size: 20px;
                padding: 20px;
                text-align: center;
            }
        </style>
    </head>
    <body>
        <div class="grid">
            <div class="grid-item">1</div>
            <div class="grid-item">2</div>
            <div class="grid-item">3</div>
            <div class="grid-item">4</div>
            <div class="grid-item">5</div>
            <div class="grid-item">6</div>
            <div class="grid-item">7</div>
            <div class="grid-item">8</div>
            <div class="grid-item">9</div>
            <div class="grid-item">10</div>
            <div class="grid-item">11</div>
            <div class="grid-item">12</div>
        </div>
    </body>
</html>
```



⑤ Styling libraries with Bootstrap website

Google "bootstrap" → home page → "Docs" button on top → "introduction" section has a link

copy & paste into <head> → choose left bar's desired sections (e.g. Components)

⑥ Bootstrap grid system with 12 columns

Bootstrap

- It turns out that there are many libraries that other people have already written that can make the styling of a webpage even simpler. One popular library that we'll use throughout the course is known as bootstrap (https://getbootstrap.com/).
- We can include bootstrap in our code by adding a single line to the head of our HTML file:

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2
```

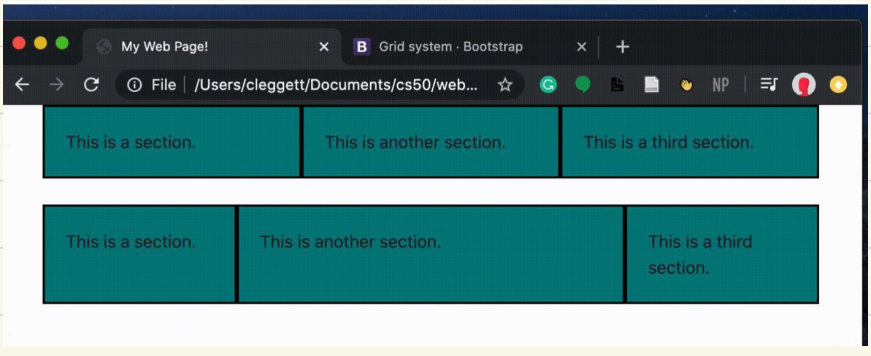
- Next, we can look at some of bootstrap's features by navigating to the documentation (https://getbootstrap.com/docs/4.5/components/) portion of their website. On this page, you'll find many examples of classes you can add to elements that allow them to be styled with bootstrap.
- One popular bootstrap feature is their grid system (https://getbootstrap.com/docs/4.0/layout/grid/). Bootstrap automatically splits a page into 12 columns, and we can decide how many columns an element takes up by adding the class `col-x` where `x` is a number between 1 and 12. For example, in the following page, we have a row of columns of equal width, and then a row where the center column is larger:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>My Web Page!</title>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha3
        <style>
            .row > div {
                padding: 20px;
                background-color: teal;
                border: 2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="row">
                <div class="col-4">
                    This is a section.
                </div>
                <div class="col-4">
                    This is another section.
                </div>
                <div class="col-4">
                    This is a third section.
                </div>
            </div>
        </div>
        <br/>
        <div class="container">
            <div class="row">
                <div class="col-3">
                    This is a section.
                </div>
                <div class="col-6">
                    This is another section.
                </div>
                <div class="col-3">
                    This is a third section.
                </div>
            </div>
        </div>
    </body>
</html>
```

Σ = 12

Σ = 12

output



⑦ Bootstrap allows to specify column sizes that differ depending on the screen size.

- To improve mobile-responsiveness, bootstrap also allows us to specify column sizes that differ depending on the screen size. In the following example, we use `col-lg-3` to show that an element should take up 3 columns on a large screen, and `col-sm-6` to show an element should take up 6 columns when the screen is small:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>My Web Page!</title>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha3
        <style>
            .row > div {
                padding: 20px;
                background-color: teal;
                border: 2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="row">
                <div class="col-lg-3 col-sm-6">
                    This is a section.
                </div>
                <div class="col-lg-3 col-sm-6">
                    This is another section.
                </div>
                <div class="col-lg-3 col-sm-6">
                    This is a third section.
                </div>
                <div class="col-lg-3 col-sm-6">
                    This is a fourth section.
                </div>
            </div>
        </div>
    </body>
</html>
```

two rows for small screen, as two sets of $\Sigma = 12$

Part D – Sass

① Sass (Syntactically Awesome Style Sheet) is a language that allows us to write CSS more efficiently in several ways

- So far, we've found a few ways to eliminate redundancy in CSS such as moving it to separate files or using bootstrap, but there are still quite a few places where we can still make improvements. For example, what if we want several elements to have different styles, but for all of them to be the same color? If we decide later we want to change the color, then we would have to change it within several different elements.
- Sass (https://sass-lang.com/) is a language that allows us to write CSS more efficiently in several ways, one of which is by allowing us to have variables, as in the following example.
- When writing in Sass, we create a new file with the extension `filename.scss`. In this file, we can create a new variable by adding a `$` before a name, then a colon, then a value. For example, we would write `$color: red` to set the variable color to the value red. We then access that variable using `$color`. Here's an example of our variables.scss file:

```
$color: red;
```
← variable

```
ul {
    font-size: 14px;
    color: $color;
}

ol {
    font-size: 18px;
    color: $color;
}
```

② Sass is an extension to CSS and needs to be downloaded and installed.
.scss file supports variables, then we compile it to a .css file to be linked.

- Now, in order to link this styling to our HTML file, we can't just link to the `.scss` file because most web browsers only recognize `.css` files. To deal with this problem, we have to download a program called Sass (https://sass-lang.com/install) onto our computers. Then, in our terminal, we write `sass variables.scss:variables.css` This command will compile a .scss file named `variables.scss` into a .css file named `variables.css`, to which you can add a link in your HTML page.
- To speed up this process, we can use the command `sass --watch variables.scss:variables.css` which automatically changes the `.css` file every time a change is detected in the `.scss` file.
- While using Sass, we can also physically nest our styling rather than use the CSS selectors we talked about earlier. For example, if we want to apply some styling only to paragraphs and unordered lists within a div, we can write the following:

better syntax

```
div {
    font-size: 18px;

    p {
        color: blue;
    }

    ul {
        color: green;
    }
}
```

Once compiled into CSS, we would get a file that looks like:

```
div {
    font-size: 18px;
}

div p {
    color: blue;
}

div ul {
    color: green;
}
```

③ other than "variables" ($) & "nested styling" (avoid CSS selectors), Sass supports "inheritance" (%)

- One more feature that Sass gives us is known as inheritance (https://sass-lang.com/guide). This allows us to create a basic set of styling that can be shared by several different elements. We do this by adding a `%` before a name of a class, adding some styling, and then later adding the line `@extend %classname` to the beginning of some styling. For example, the following code applies the styling within the `message` class to each of the different classes below, resulting in a webpage that looks like the one below.

```
%message {
    font-family: sans-serif;
    font-size: 18px;
    font-weight: bold;
    border: 1px solid black;
    padding: 20px;
    margin: 20px;
}

.success {
    @extend %message;
    background-color: green;
}

.warning {
    @extend %message;
    background-color: orange;
}

.error {
    @extend %message;
    background-color: red;
}
```

curly brackets

This is a success message.

This is a warning message.

This is an error message.

Lecture 2 - Git

Part A - Git & GitHub

① Git is a command line tool that will help us with version control in several ways:

1.1 Keep track of changes to code

1.2 Synchronize code between different people

1.3 Test changes to code without losing the original

1.4 Revert back to old versions of code

② Github

2.1 Create a repository

2.2 Locally, git clone the file in terminal $ git clone <url>

2.3 $ touch <fileName>.html to create a HTML file

2.4 $ code . will open default IDE

2.5 $ git add <file Name>

2.6 $ git commit -m "<string Description>"

2.7 $ git status to see what's going on

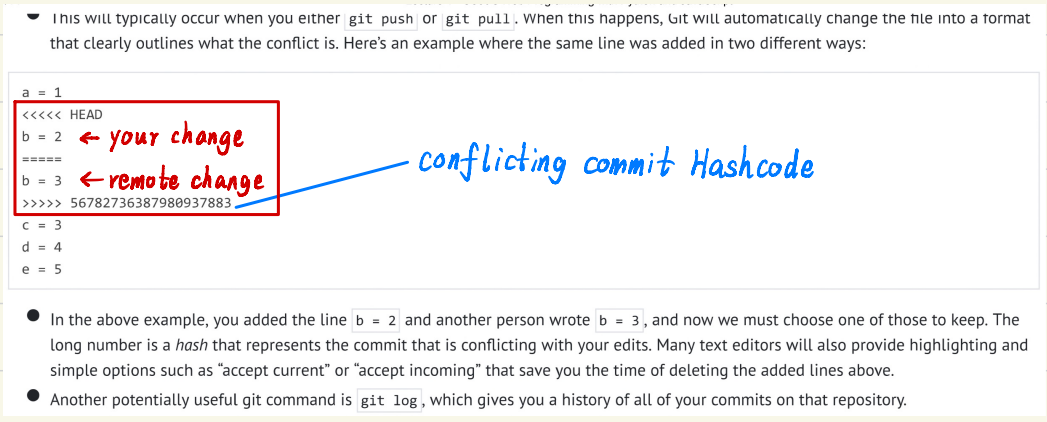
2.8 $ git push to push it online

2.9 make some changes and use $ git status to check status

③ $ git commit -am "<string Description>" combines 2.5 + 2.6

④ $ git pull pull most updated changes to local working fold

⑤ Merge conflicts ※ only AFTER commit may conflicts happen



- This will typically occur when you either `git push` or `git pull`. When this happens, Git will automatically change the file into a format that clearly outlines what the conflict is. Here's an example where the same line was added in two different ways:

```
a = 1
<<<<< HEAD
b = 2   ← your change
=====
b = 3   ← remote change
>>>>> 567827363879809378883
c = 3
d = 4
e = 5
```

conflicting commit Hashcode

- In the above example, you added the line `b = 2` and another person wrote `b = 3`, and now we must choose one of those to keep. The long number is a *hash* that represents the commit that is conflicting with your edits. Many text editors will also provide highlighting and simple options such as "accept current" or "accept incoming" that save you the time of deleting the added lines above.
- Another potentially useful git command is `git log`, which gives you a history of all of your commits on that repository.

when running $ git pull, a conflict will happen, and we can deal with it in IDE.

then $ git commit -am "<string>" (can manually delete & modify)

$ git push again to make the change remotely.

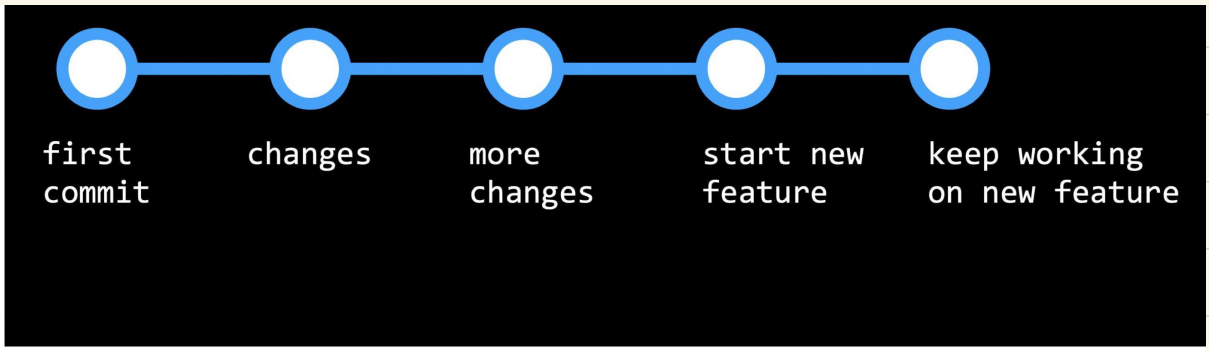
⑥ $ git log gives a history of all commits on the depository

⑦ $ git reset --hard <commitHash> return all back to the hashed commit

$ git reset --hard origin/master return all to current remote version

Part B - Branching

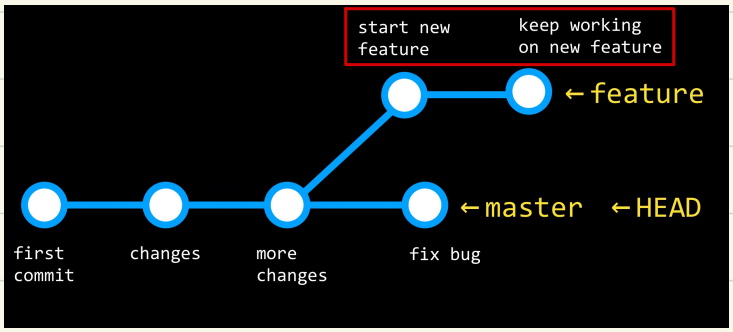
① keep to 1 branch is Linear - can NOT revert without changing wanted new feature

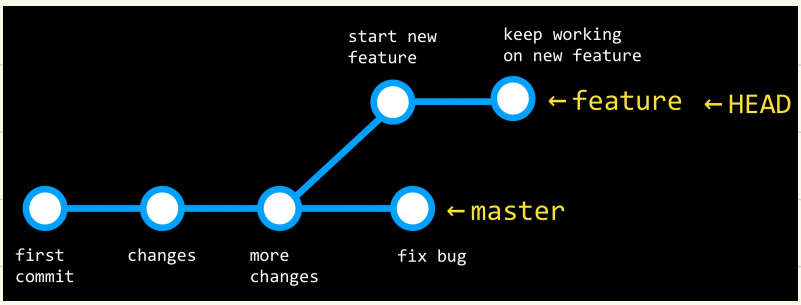


But this could become problematic if we then discover a bug in our original code, and want to revert back without changing the new feature.
This is where branching can become really useful.

② Branching when starting a new feature is a good habit

The "HEAD" can switch between branches



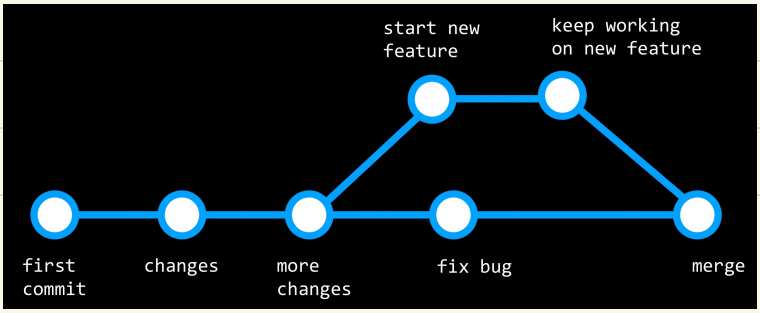


③ review branches $ git branch

Braching $ git checkout <-b> <branchName> (optional) used for creating new branch

④ Merge branch $ git merge <branchName> → Solve conflicts if there is → Commit again

(only commited changes may cause conflicts)



Part C – GitHub unique features

① Forking : make a copy of the repository that you are the owner of.
e.g. fork bootstrap repository

② Pull Requests

- **Forking**: As a GitHub user, you have the ability to *fork* any repository that you have access to, which creates a copy of the repository that you are the owner of. We do this by clicking the "Fork" button in the top-right.
- **Pull Requests**: Once you've forked a repository and made some changes to your version, you may want to request that those changes be added to the main version of the repository. For example, if you wanted to add a new feature to Bootstrap, you could fork the repository, make some changes, and then submit a pull request. This pull request could then be evaluated and possibly accepted by the people who run the Bootsrap repository. This process of people making a few edits and then requesting that they be merged into a main repository is vital for what is known as *open source software*, or software that created by contributions from a number of developers.

③ GitHub Pages : a simple way to publish a static site to web

- **GitHub Pages**: GitHub Pages is a simple way to publish a static site to the web. (We'll learn later about static vs dynamic sites.) In order to do this:
  1. Create a new GitHub repository.
  2. Clone the repository and make changes locally, making sure to include an `index.html` file which will be the landing page for your website.
  3. Push those changes to GitHub.
  4. Navigate to the Settings page of your repository, scroll down to GitHub Pages, and choose the master branch in the dropdown menu.
  5. Scroll back down to the GitHub Pages part of the settings page, and after a few minutes, you should see a notification that "Your site is published at: ..." including a URL where you can find your site!

xxx.github.io

Lecture 3 - Python (Supplements)

Part A - Functional Programming

① Functional Programming Paradigm: Functions are treated as values just like any other variables.

② Decorators (use with @): higher-order functions that can modify other functions.

Decorators

One thing made possible by functional programming is the idea of a decorator, which is a higher-order function that can modify another function. For example, let's write a decorator that announces when a function is about to begin, and when it ends. We can then apply this decorator using an `@` symbol.

```
def announce(f):
    def wrapper():
        print("About to run the function")
        f()
        print("Done with the function")
    return wrapper

@announce
def hello():
    print("Hello, world!")

hello()

""" Output:
About to run the function
Hello, world!
Done with the function
"""
```

the parameter is a function (function as a variable)

return "type" must be a function

③ Lambda Functions (lambda): useful when we don't want to write a whole separate function for a single, small use (a quick way to create a function)

input output

```
square = lambda x: x * x
```

Where the input is to the left of the `:` and the output is on the right.

This can be useful when we don't want to write a whole separate function for a single, small use. For example, if we want to sort some objects where it's not clear at first how to sort them. Imagine we have a list of people, but with names and houses instead of just names that we wish to sort:

```
people = [
    {"name": "Harry", "house": "Gryffindor"},
    {"name": "Cho", "house": "Ravenclaw"},
    {"name": "Draco", "house": "Slytherin"}
]

people.sort()

print(people)
```

This, however, leaves us with the error:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'dict' and 'dict'
```

③' Lambda Functions - e.x. of sorting dictionaries

We can solve this problem by including a `key` argument to the sort function, which specifies which part of the dictionary we wish to use to sort:

```
people = [
    {"name": "Harry", "house": "Gryffindor"},
    {"name": "Cho", "house": "Ravenclaw"},
    {"name": "Draco", "house": "Slytherin"}
]

def f(person):
    return person["name"]

people.sort(key=f)

print(people)

""" Output:
[{'name': 'Cho', 'house': 'Ravenclaw'}, {'name': 'Draco', 'house': 'Slytherin'}, {'name': 'Harry', 'house': 'Gryffindor'}]
"""
```

without vs. with lambda function

While this does work, we've had to write an entire function that we're only using once, we can make our code more readable by using a lambda function:

```
people = [
    {"name": "Harry", "house": "Gryffindor"},
    {"name": "Cho", "house": "Ravenclaw"},
    {"name": "Draco", "house": "Slytherin"}
]

people.sort(key=lambda person: person["name"])
print(people)

""" Output:
[{'name': 'Cho', 'house': 'Ravenclaw'}, {'name': 'Draco', 'house': 'Slytherin'}, {'name': 'Harry', 'house': 'Gryffindor'}]
"""
```

Part B - Exceptions

① Exception Handling (with try ... except ...)

```
import sys

try:
    x = int(input("x: "))
    y = int(input("y: "))
except ValueError:
    print("Error: Invalid input")
    sys.exit(1)

try:
    result = x / y
except ZeroDivisionError:
    print("Error: Cannot divide by 0.")
    # Exit the program
    sys.exit(1)

print(f"{x} / {y} = {result}")
```