Course syllabus for Introduction to Front-End Development

This course is the first of a series that aims to help you learn more about web development and prepares you for using Bootstrap on a biographical page you will create. By the end of this course, you'll be able to:

- Describe the front-end developer role
- Explain the core and underlying technologies that power the internet
- Use HTML to create a simple webpage
- Use CSS to control the appearance of a simple webpage
- Explain what React is
- Describe the applications and characteristics of the most popular UI frameworks

In this course, you will explore the following:

Module 1: Get started with web development

In this module, you are introduced to web development. You'll learn about the different types of web developer roles and the responsibilities of front-end, back-end and full-stack developers. You will get a streamlined overview of the core technologies of HTML, CSS and JavaScript and explore the concepts that underpin how the internet works. Furthermore, you will be able to access hands-on exercises to edit a website.

After completing this module, you will be able to:

1. Describe the web developer job role.
2. Distinguish between front-end, back-end and full-stack developers.
3. Explain how data moves through the internet.
4. Describe the technologies that underpin the internet.

Module 2: Introduction to HTML5 and CSS

Here you'll learn about HTML5 and CSS. You'll also examine how to construct HTML documents and add basic styling and layout using CSS.

After completing this module, you will be able to:

1. Use HTML to create a simple webpage.
2. Use CSS to define the style of a simple webpage.

Module 3: UI Frameworks

In this module, you'll learn about UI frameworks. In addition, you will learn how to use the Bootstrap framework to build responsive interfaces. You'll explore the benefits of working with UI frameworks.

After completing this module, you will be able to:

1. Outline the concepts that exist in most UI frameworks.
2. Use the Bootstrap CSS framework to create webpages.
3. Leverage Bootstrap documentation to reproduce and modify CSS components.
4. Use Bootstrap themes.
5. Describe the basics of React in relation to other frameworks and web technologies.

Module 4: Graded Assessment

Here you'll learn about the graded assessment. After you complete the individual units in this module, you'll synthesize the skills from the course to create and style a biographical page. You'll also have the opportunity to reflect on the course content and the learning path that lies ahead.
After completing this module, you will be able to:

1. Create and style a biographical page.

Capstone project demo: Little Lemon website

You've just begun your coding journey on the Meta front-end developer program. By the end of this program, you'll put your new skills to work by completing a real-world portfolio project, where you'll create your own dynamic front-end web application. Completing this project will help you to validate the knowledge and skills that you have gained.

What you will be able to develop

You will build this web app yourself in React and use all of the excellent tools available. Putting your newly acquired skills into practice, you will demonstrate how to build and program part of a responsive web app.
It includes the following elements:

- A home screen with information about the restaurant.
- A table reservation system.
- A profile screen for users to enter their personal details.
- Navigation that enables users to move between parts of the web app.

Project preview

In the video below, you can take a trip into the future and get a preview of what you'll be able to create with your new set of skills at the end of this program.

Demo

That's what you'll be able to accomplish by the end of this Front-end development program. Pretty cool, right? It's a modern front-end application for the Little Lemon restaurant and the best thing about this project is that it's part of your portfolio and you'll be able to showcase it to any prospective employer as proof of your abilities. It shows the multiple skills you will learn during this program.

Conclusion

In this reading, you explored the front-end application for the Little Lemon restaurant that demonstrates the kind of project you will develop towards the end of the program.
Best of luck in your coding journey.

Additional Resources

Learn more Here is a list of resources that may be helpful as you continue your learning journey.

What is a Web Server? (NGINX)
https://www.nginx.com/resources/glossary/web-server/
What is a Web Browser? (Mozilla)
https://www.mozilla.org/en-US/firefox/browsers/what-is-a-browser/
Who invented the Internet? And why? (Kurzgesagt)
https://youtu.be/21eFwbb48sE
What is Cloud Computing? (Amazon)
https://youtu.be/mxT233EdY5c
Browser Engines (Wikipedia)
https://en.wikipedia.org/wiki/Browser_engine

HTTP examples

This reading explores the contents of HTTP requests and responses in more depth.

Request Line

Every HTTP request begins with the request line.
This consists of the HTTP method, the requested resource and the HTTP protocol version.

```
GET /home.html HTTP/1.1
```

In this example, `GET` is the HTTP method, `/home.html` is the resource requested and HTTP 1.1 is the protocol used.

HTTP Methods

HTTP methods indicate the action that the client wishes to perform on the web server resource.
Common HTTP methods are:

HTTP Method	Description
GET	The client requests a resource on the web server.
POST	The client submits data to a resource on the web server.
PUT	The client replaces a resource on the web server.
DELETE	The client deletes a resource on the web server.

HTTP Request Headers

After the request line, the HTTP headers are followed by a line break.
There are various possibilities when including an HTTP header in the HTTP request. A header is a case-insensitive name followed by a `:` and then followed by a value.
Common headers are:

```
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: */*
Accept-Language: en
Content-type: text/json
```

- The **`Host`** header specifies the host of the server and indicates where the resource is requested from.
- The **`User-Agent`** header informs the web server of the application that is making the request. It often includes the operating system (Windows, Mac, Linux), version and application vendor.
- The **`Accept`** header informs the web server what type of content the client will accept as the response.
- The **`Accept-Language`** header indicates the language and optionally the locale that the client prefers.
- The **`Content-type`** header indicates the type of content being transmitted in the request body.

HTTP Request Body

HTTP requests can optionally include a request body. A request body is often included when using the HTTP POST and PUT methods to transmit data.

```
POST /users HTTP/1.1
Host: example.com

{
 "key1":"value1",
 "key2":"value2",
 "array1":["value3","value4"]
}
```

```
PUT /users/1 HTTP/1.1
Host: example.com
Content-type: text/json

{"key1":"value1"}
```

HTTP Responses

When the web server is finished processing the HTTP request, it will send back an HTTP response. The first line of the response is the status line. This line shows the client if the request was successful or if an error occurred.

`HTTP/1.1 200 OK`

The line begins with the HTTP protocol version, followed by the status code and a reason phrase. The reason phrase is a textual representation of the status code.

HTTP Status Codes

The first digit of an HTTP status code indicates the category of the response: Information, Successful, Redirection, Client Error or Server Error.
The common status codes you'll encounter for each category are:

1XX Informational

Status Code	Reason Phrase	Description
100	Continue	The server received the request headers and should continue to send the request body.
101	Switching Protocols	The client has requested the server to switch protocols and the server has agreed to do so.

2XX Successful

Status Code	Reason Phrase	Description
200	OK	Standard response returned by the server to indicate it successfully processed the request.
201	Created	The server successfully processed the request and a resource was created.
202	Accepted	The server accepted the request for processing but the processing has not yet been completed.
204	No Content	The server successfully processed the request but is not returning any content.

3XX Redirection

Status Code	Reason Phrase	Description
301	Moved Permanently	This request and all future requests should be sent to the returned location.
302	Found	This request should be sent to the returned location.

4XX Client Error

Status Code	Reason Phrase	Description
400	Bad Request	The server cannot process the request due to a client error, e.g., invalid request or transmitted data is too large.
401	Unauthorized	The client making the request is unauthorized and should authenticate.

403	Forbidden	The request was valid but the server is refusing to process it. This is usually returned due to the client having insufficient permissions for the website, e.g., requesting an administrator action but the user is not an administrator.
404	Not Found	The server did not find the requested resource.
405	Method Not Allowed	The web server does not support the HTTP method used.

5XX Server Error

Status Code	Reason Phrase	Description
500	Internal Server Error	A generic error status code given when an unexpected error or condition occurred while processing the request.
502	Bad Gateway	The web server received an invalid response from the Application Server.
503	Service Unavailable	The web server cannot process the request.

HTTP Response Headers

Following the status line, there are optional HTTP response headers followed by a line break.
Similar to the request headers, there are many possible HTTP headers that can be included in the HTTP response.
Common response headers are:

```
Date: Fri, 11 Feb 2022 15:00:00 GMT+2
Server: Apache/2.2.14 (Linux)
Content-Length: 84
Content-Type: text/html
```

- The **`Date`** header specifies the date and time the HTTP response was generated.
- The **`Server`** header describes the web server software used to generate the response.
- The **`Content-Length`** header describes the length of the response.
- The **`Content-Type`** header describes the media type of the resource returned (e.g. HTML document, image, video).

HTTP Response Body

Following the HTTP response headers is the HTTP response body. This is the main content of the HTTP response.
This can contain images, video, HTML documents and other media types.

```
HTTP/1.1 200 OK
Date: Fri, 11 Feb 2022 15:00:00 GMT+2
Server: Apache/2.2.14 (Linux)
Content-Length: 84
Content-Type: text/html

<html>
  <head><title>Test</title></head>
  <body>Test HTML page.</body>
</html>
```

Other Internet Protocols

Hypertext Transfer Protocols (HTTP) are used on top of Transmission Control Protocol (TCP) to transfer webpages and other content from websites. This reading explores other protocols commonly used on the Internet.

Dynamic Host Configuration Protocol (DHCP)

You've learned that computers need IP addresses to communicate with each other. When your computer connects to a network, the Dynamic Host Configuration Protocol or DHCP as it is commonly known, is used to assign your computer an IP address. Your computer communicates over User Datagram Protocol (UDP) using the protocol with a type of server called a DHCP server. The server keeps track of computers on the network and their IP addresses. It will assign your computer an IP address and respond over the protocol to let it know which IP address to use. Once your computer has an IP address, it can communicate with other computers on the network.

Domain Name System Protocol (DNS)

Your computer needs a way to know with which IP address to communicate when you visit a website in your web browser, for example, **`meta.com`**. The Domain Name System Protocol, commonly known as DNS, provides this function. Your computer then checks with the DNS server associated with the domain name and then returns the correct IP address.

Internet Message Access Protocol (IMAP)

Do you check your emails on your mobile or tablet device? Or maybe you use an email application on your computer? Your device needs a way to download emails and manage your mailbox on the server storing your emails. This is the purpose of the Internet Message Access Protocol or IMAP.

Simple Mail Transfer Protocol (SMTP)

Now that your emails are on your device, you need a way to send emails. The Simple Mail Transfer Protocol, or SMTP, is used. It allows email clients to submit emails for sending via an SMTP server. You can also use it to receive emails from an email client, but IMAP is more commonly used.

Post Office Protocol (POP)

The Post Office Protocol (POP) is an older protocol used to download emails to an email client. The main difference in using POP instead of IMAP is that POP will delete the emails on the server once they have been downloaded to your local device. Although it is no longer commonly used in email clients, developers often use it to implement email automation as it is a more straightforward protocol than IMAP.

File Transfer Protocol (FTP)

When running your websites and web applications on the Internet, you'll need a way to transfer the files from your local computer to the server they'll run on. The standard protocol used for this is the File Transfer Protocol or FTP. FTP allows you to list, send, receive and delete files on a server. Your server must run an FTP Server and you will need an FTP Client on your local machine. You'll learn more about these in a later course.

Secure Shell Protocol (SSH)

When you start working with servers, you'll also need a way to log in and interact with the computer remotely. The most common method of doing this is using the Secure Shell Protocol, commonly referred to as SSH. Using an SSH client allows you to connect to an SSH server running on a server to perform commands on the remote computer. All data sent over SSH is encrypted. This means that third parties cannot understand the data transmitted. Only the sending and receiving computers can understand the data.

SSH File Transfer Protocol (SFTP)

The data is transmitted insecurely when using the File Transfer Protocol. This means that third parties may understand the data that you are sending. This is not right if you transmit company files such as software and databases. To solve this, the SSH File Transfer Protocol, alternatively called the Secure File Transfer Protocol, can be used to transfer files over the SSH protocol. This ensures that the data is transmitted securely. Most FTP clients also support the SFTP protocol.

Exercise: Examine a web page

Introduction

In this exercise, you will practice examining an HTML page using the developer tools.

Goal

- Inspect the HTML document using the developer tools in your browser.

Objectives

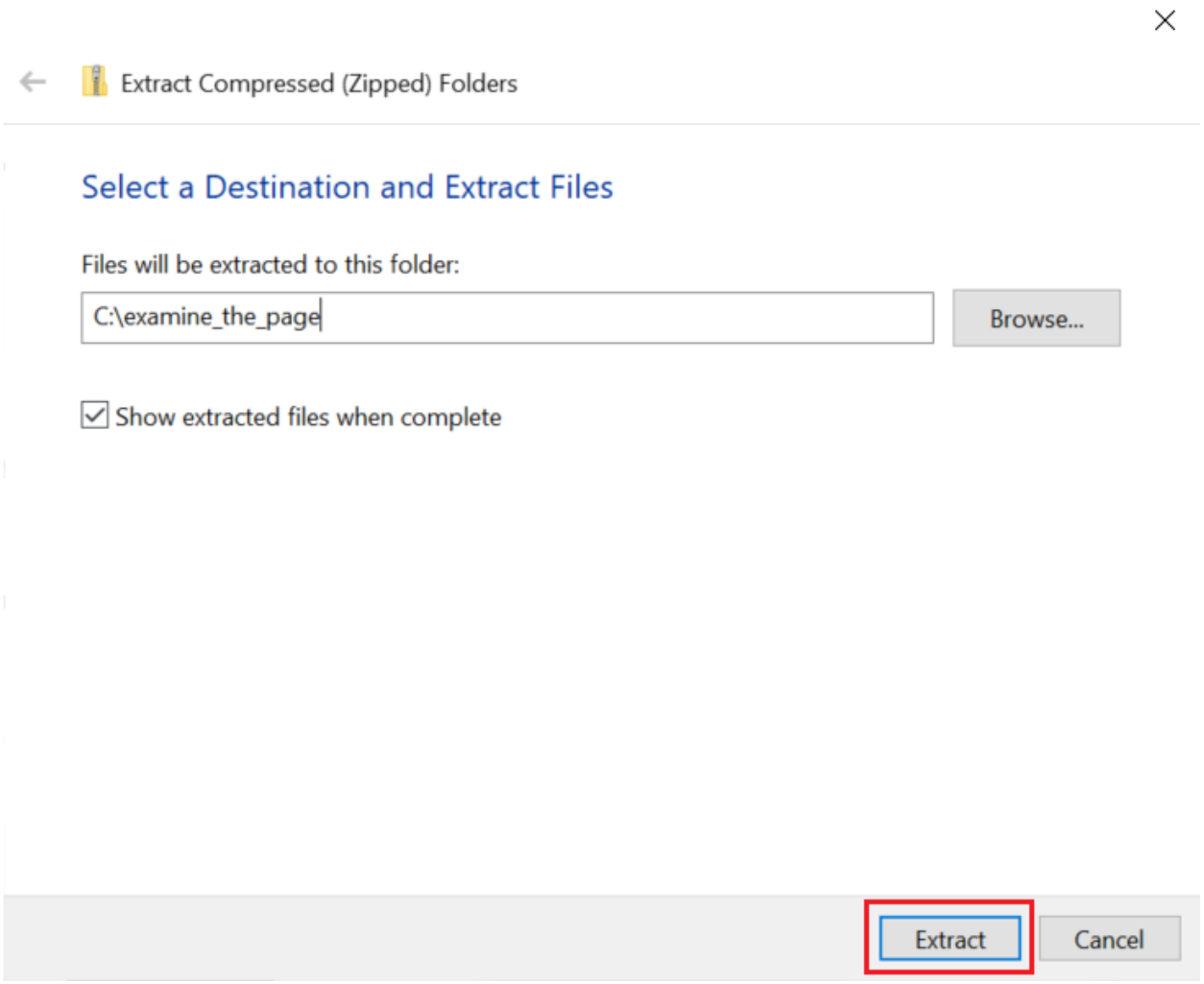
- Find the HTML ID of the Little Lemon logo.

Instructions

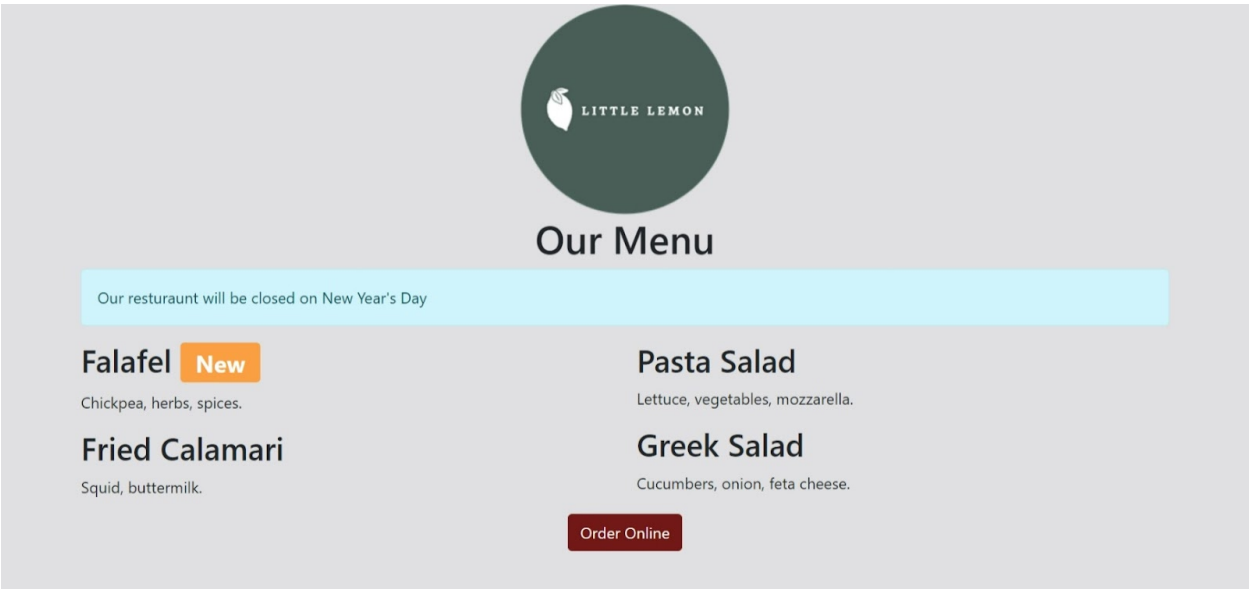
Step 1: Download the following file on your local system.

examine_the_page
ZIP File

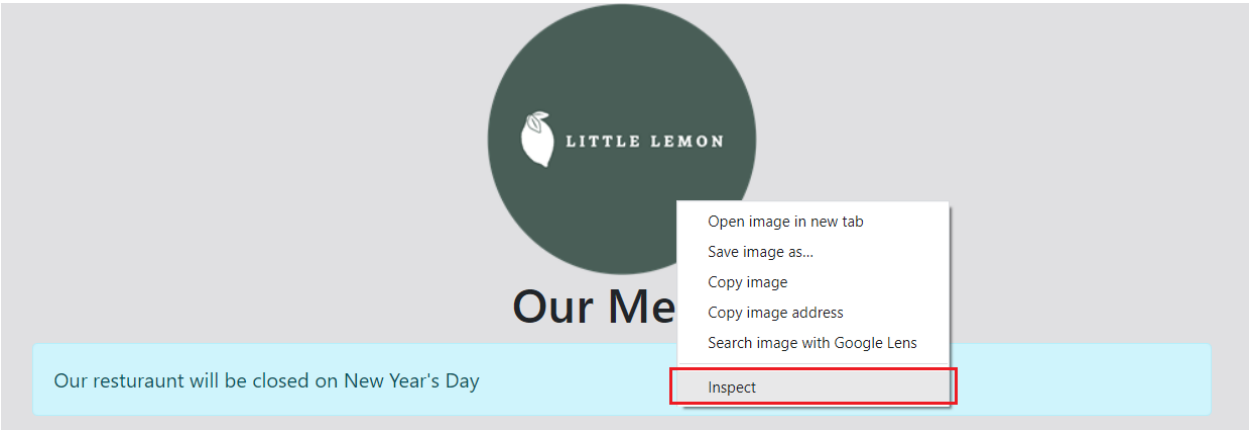
Step 2: Unzip the file.
On Windows, open your Downloads folder, right-click the file *examine_the_page.zip* and select Extract All.



On Mac, open your Downloads folder and double click the file *examine_the_page.zip*.
Once unzipped, there will be a folder named *examine_the_page*.
Step 3: Open the folder and double click index.html to view the file in your local web browser. Verify that it looks like this:



Step 4: Right-click the Little Lemon logo and select Inspect (or Inspect Element)



Step 5: Inspect the line in the HTML for the logo in the developer tools panel. The line begins with <img.

Note: In the line, there is an ID in the following format id="???". Note the value that the ID is equal to.

Tips

- If you get stuck, close the developer tools and start over.
- Review the lesson *Developer Tools*.

Exercise: Edit a website using a browser developer tools

Introduction

In this exercise, you will practice editing an HTML page using the developer tools.

Goal

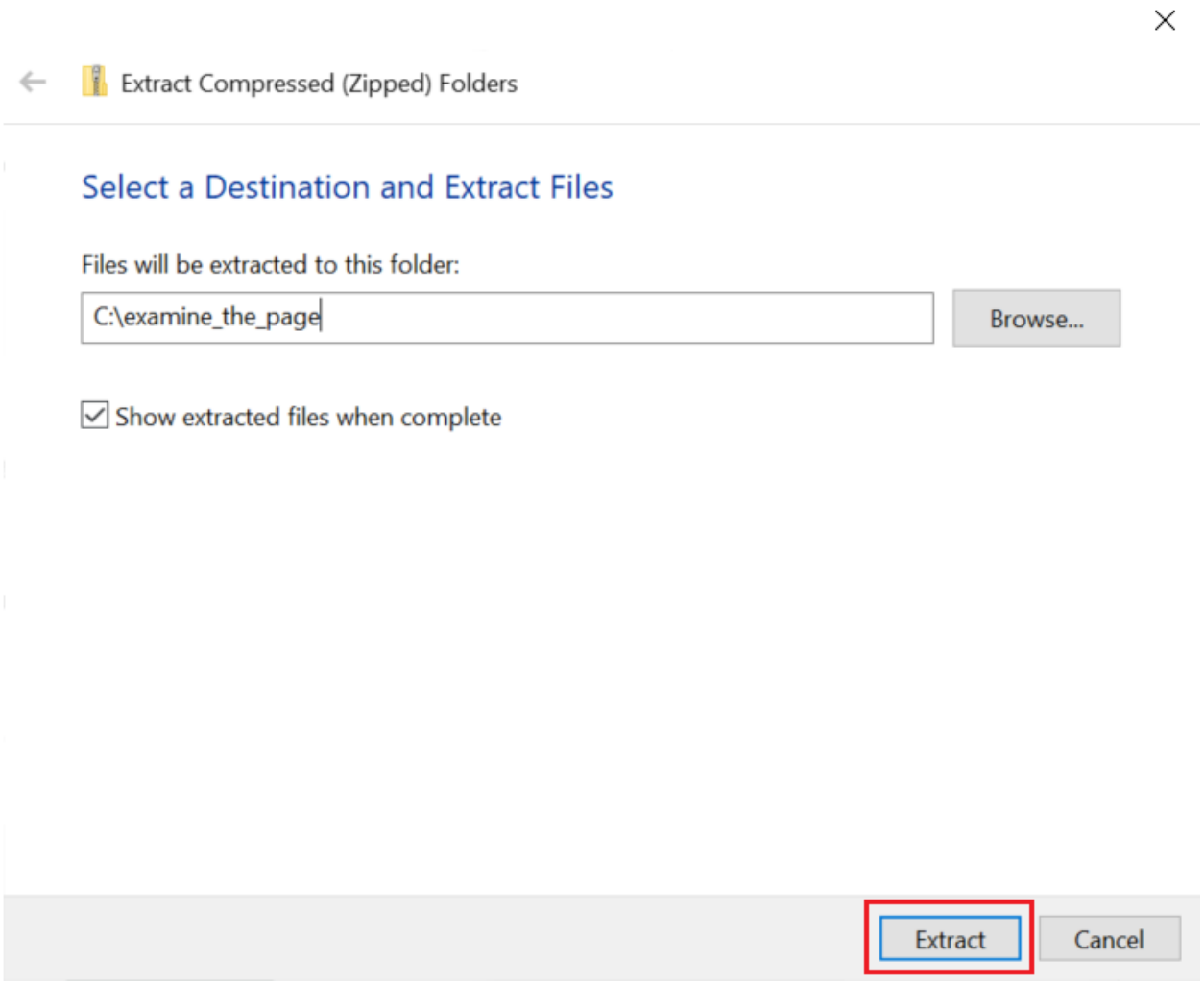
- Edit the HTML document using the developer tools in your browser.

Objectives

- Change the text of Our Menu to Little Lemon Menu.

Instructions

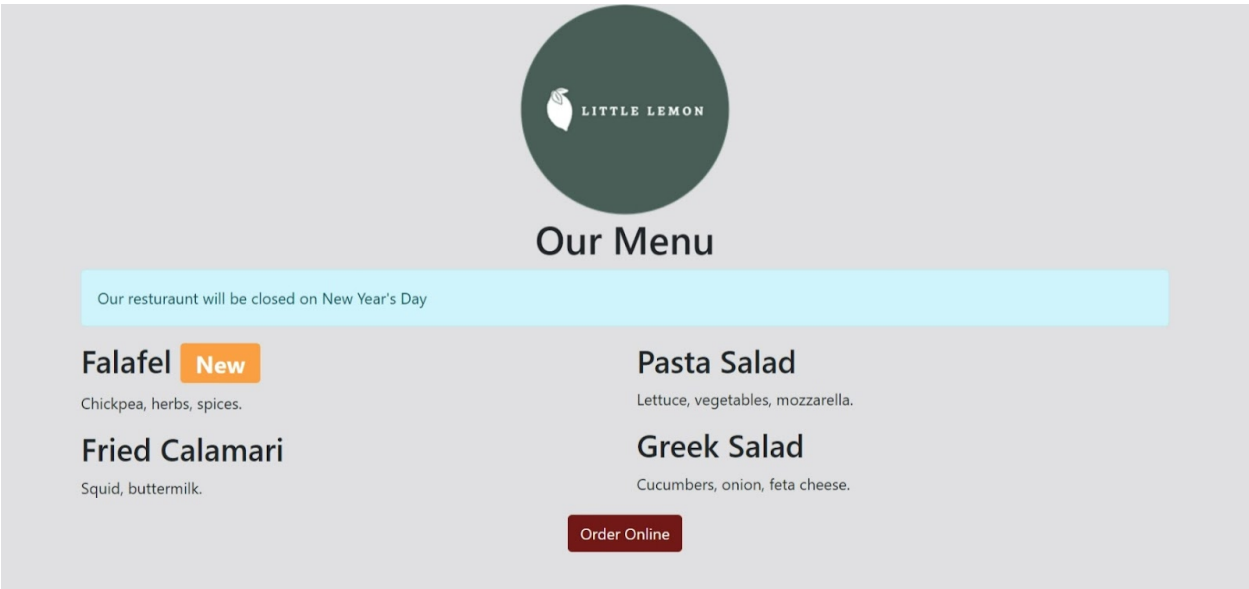
Step 1: Download the following file on your local system.
Note: If you have completed the exercise Examine the Page, you can skip steps 1 and 2 as the file contains the same assets from that exercise.
examine_the_page
ZIP File
Step 2: Unzip the file.
On Windows, open your Downloads folder, right-click the file *examine_the_page.zip* and select Extract All.



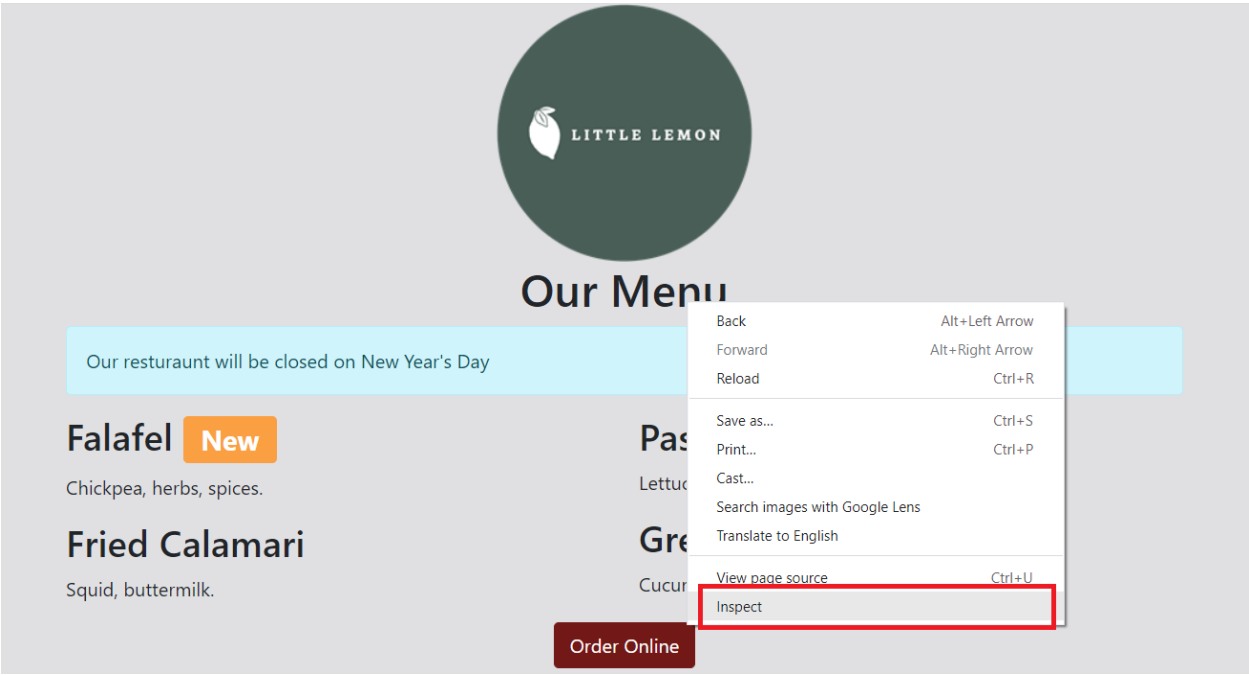
On Mac, open your Downloads folder and double click the file *examine_the_page.zip*.

Once unzipped, there will be a folder named *examine_the_page*.

Step 3: Open the index.html file in your local browser for preview. Verify that it looks like this:



Step 4: Right-click the **`Our Menu`** text and select **`Inspect`** or **`Inspect Element.`**



Step 5: Double-click the **`Our Menu`** text in the Elements tab of the developer tools panel.

Step 6: Change the text to **`Little Lemon Menu`**.

Step 7: Close the developer tools.

Step 8: Verify that the text has changed on the web page.

Tips

- If you get stuck, close the developer tools and start from the beginning.
- Review the lesson *Developer Tools*.

Setting up your local development environment

This reading walks you through the steps to set up an Integrated Development Environment, or IDE, on a Windows and on a Mac (further down below).

The IDE you'll be using in the course is Visual Studio Code, which Microsoft provides for free and comes with a wealth of plugins and extensions to make your life as a developer easier.

You have two options for using Visual Studio Code to complete your course activities:

Option 1: Use Visual Studio Code in-browser with Coursera Labs

Coursera's platform offers an in-browser version of Visual Studio Code which is preconfigured and requires no local setup. You can access the Visual Studio Code environment through the "Lab" items included in this course. Your work and files will be saved and available within this in-browser lab while you have course access.

Option 2: Work on your local device

You can also choose to complete your work on your local machine if you prefer. This will require a few steps of set up in advance.

First, you need to download the IDE from Microsoft's website - https://code.visualstudio.com/download.

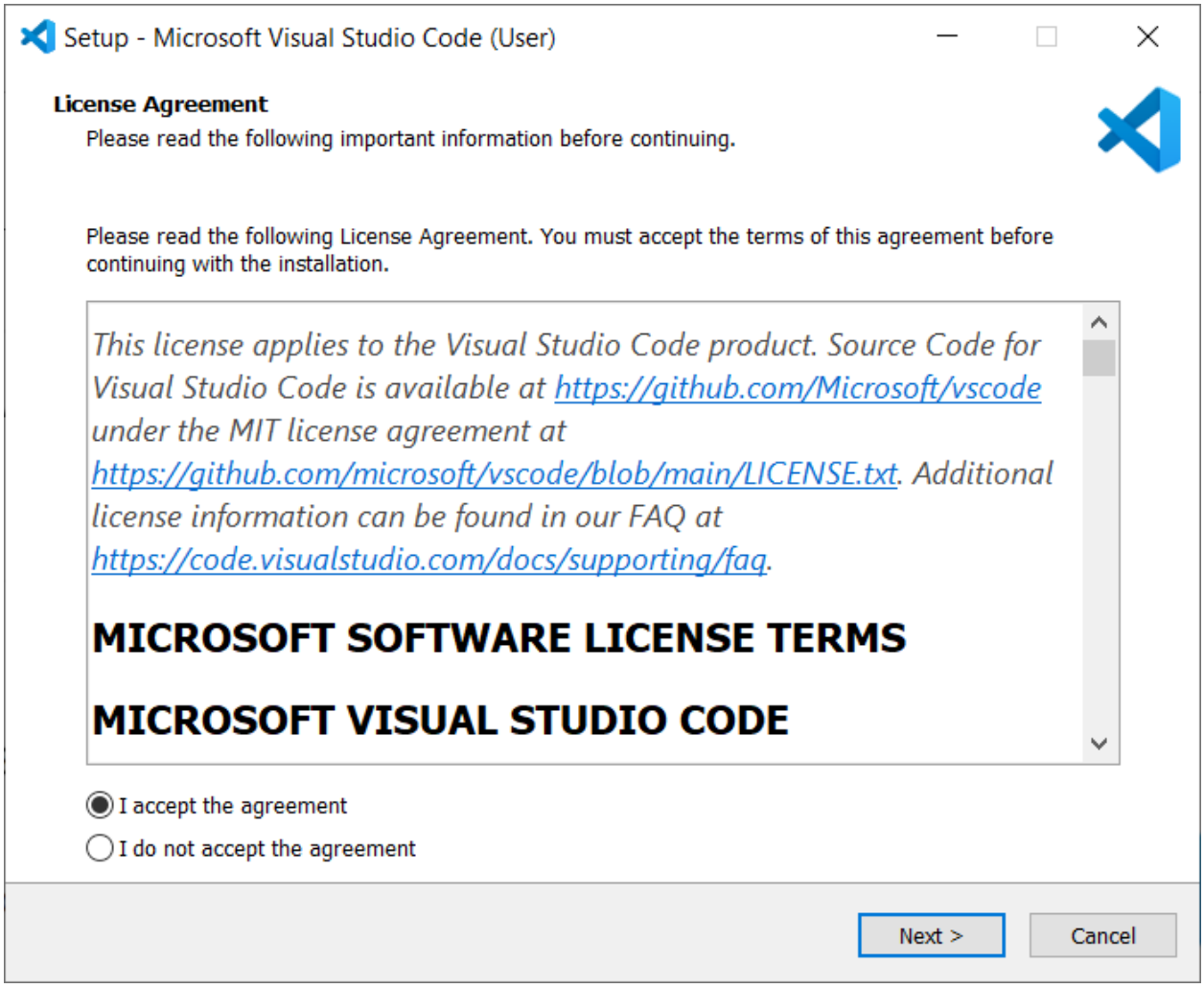
Select the download based on your operating system.

Windows

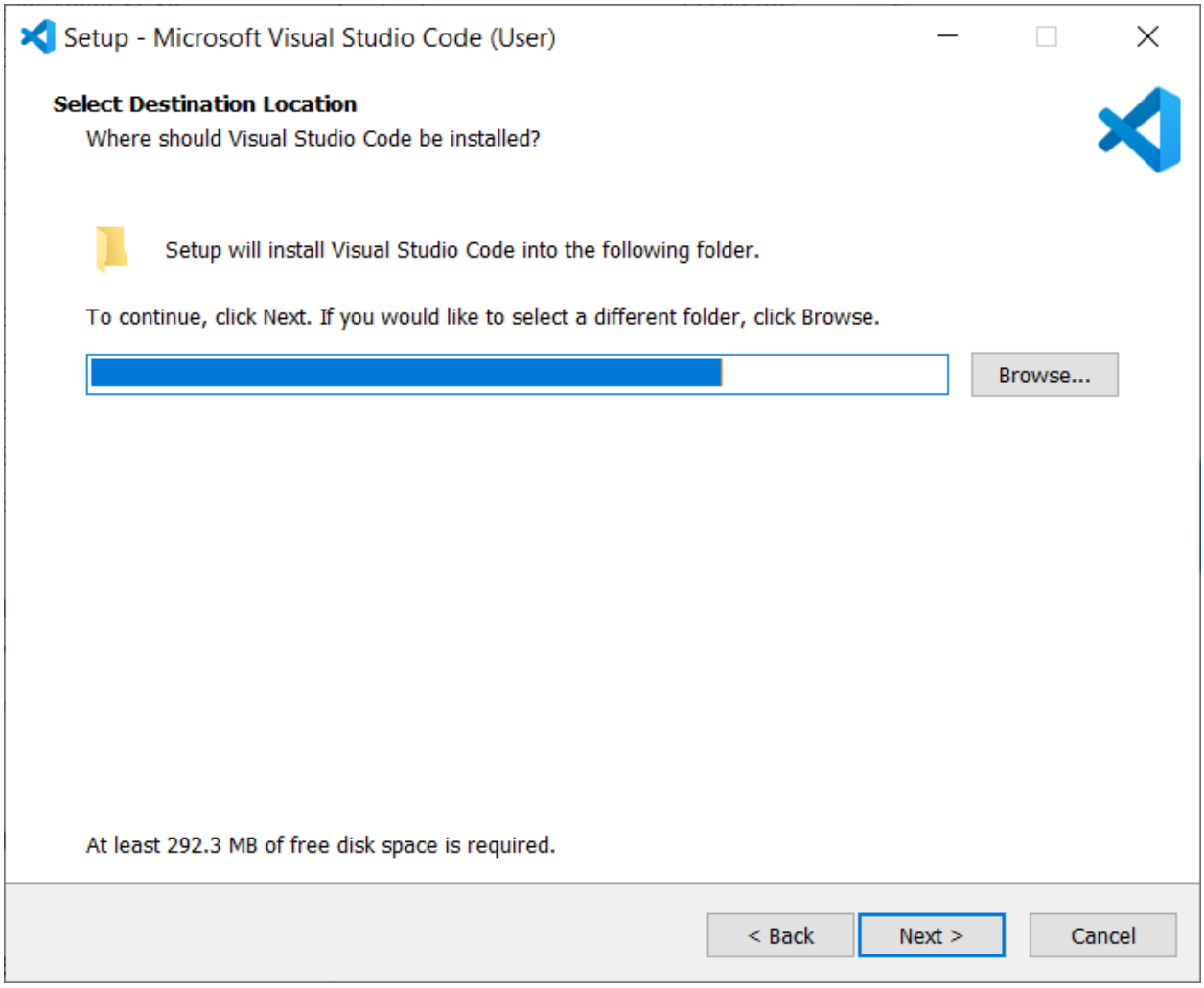
Step 1: Download the Windows installer.

Step 2: Open the file to install it once the download is complete.

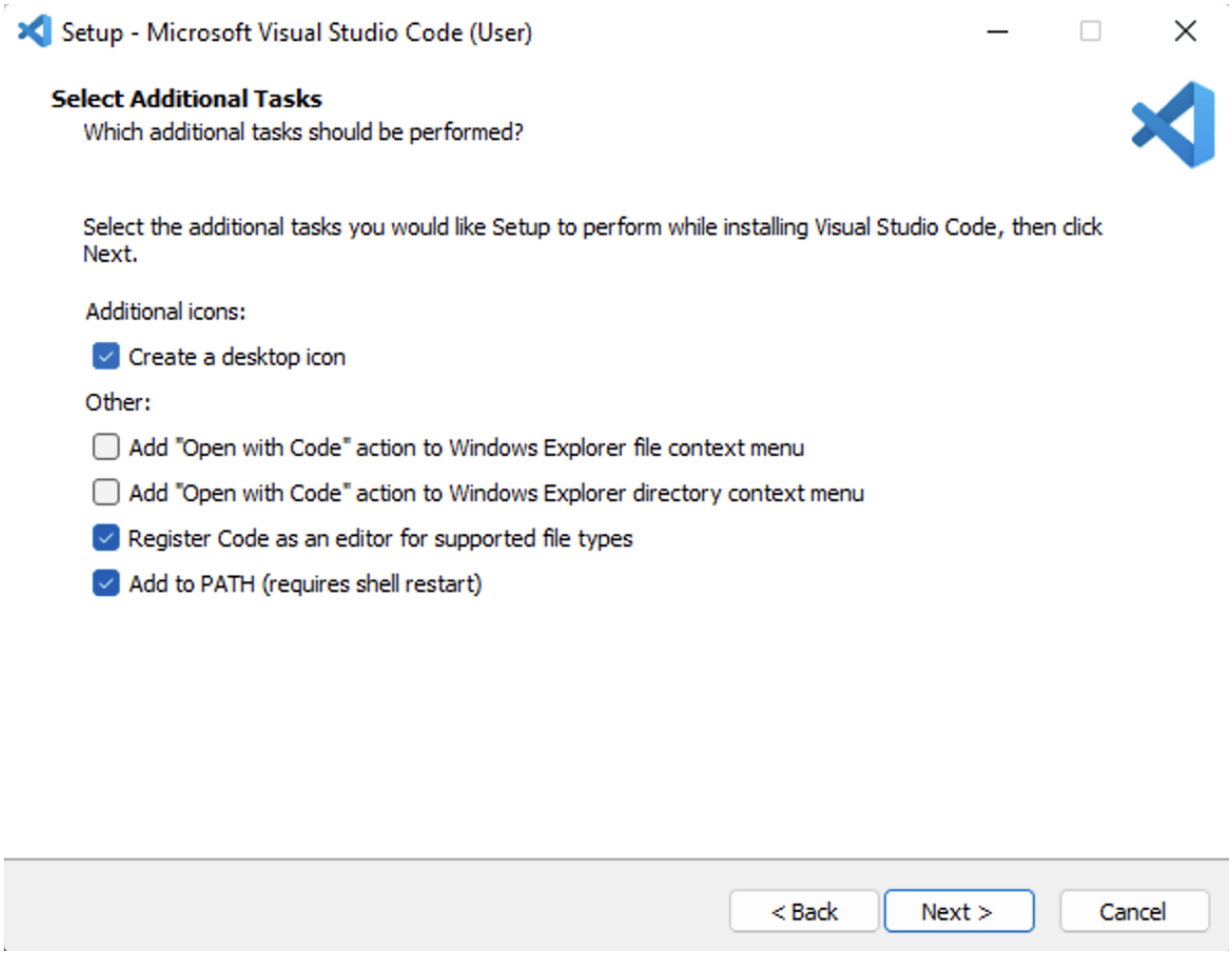
Step 3: Review and accept the license agreement, then click Next.



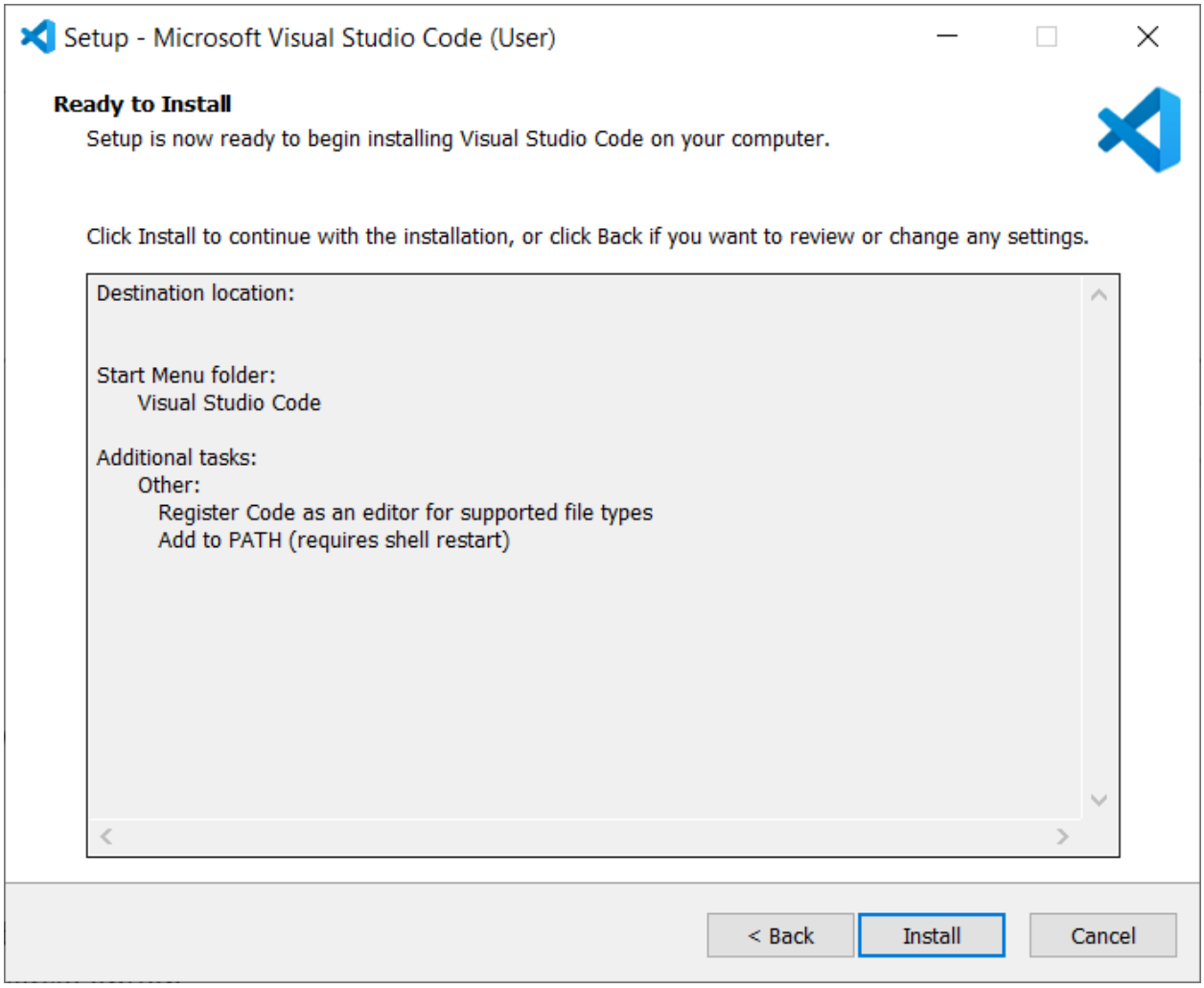
Step 4: Keep the default value when prompted for the destination location and click next.



Step 5: On the additional tasks view, make sure that **Add to PATH** is selected.



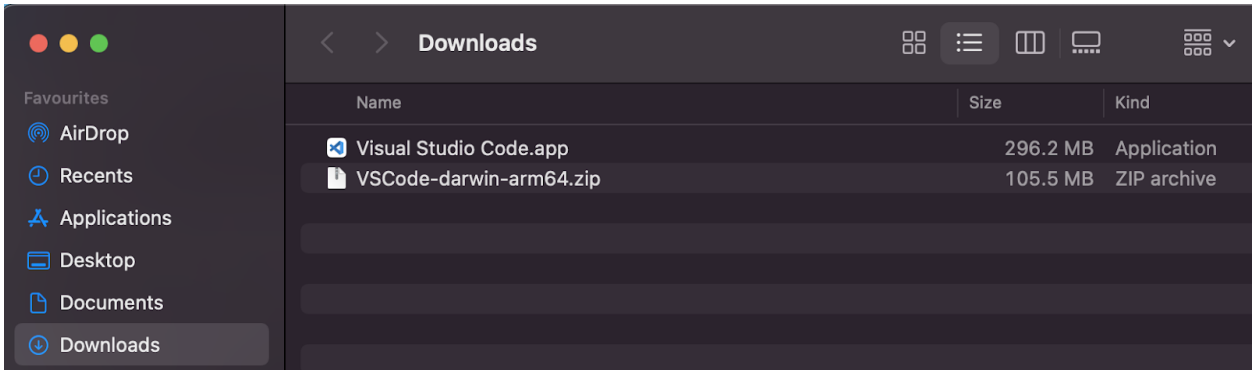
Step 6: Click next.
Step 7: Click install when the ready to install page appears.



Step 8: Click finish once completed, and the application will launch.

Mac

Step 1: Download the application based on the chipset you have. M1 macs use Apple Silicon, and older Macs use Intel. If you are not sure, choose the Universal option.
Step 2: Go to your Downloads folder once the download is complete.
Step 3: Double-click the zip file to extract the contents.
Step 4: Drag and drop the .app file to the application link in Finder below.



Step 5: Open the app.

Linux

Please refer to the official Linux installation guide for Visual Studio Code.

Selecting a working directory

Now that you have Visual Studio Code set up create a folder on your device that you'll use to do course exercises.

Open Visual Studio Code, go to `File` and select `Open Folder`. Using the file browser, select the folder you just created.

Congratulations, you're set up now to begin writing some code.

Visual Studio Code on Coursera

In addition to having Visual Studio Code installed on your own computer, in this course and throughout this program, you'll have the opportunity to work in Visual Studio Code right here on Coursera!

As you progress through the course, you'll be able to write code in hands-on activities called **Labs**. In these labs you'll be able to open Visual Studio Code and start writing code without ever leaving the course.

You'll have plenty of opportunities to see Labs in action later in the course, but for now, use the images below as a visual guide to how Labs will look and operate in your browser.

Lab: Creating an HTML Document

The Labs contain instructions explaining the coding task.

Creating an HTML Document

Open Lab

Instructions

Introduction

In this ungraded lab you will practice creating a simple HTML document.

Goal

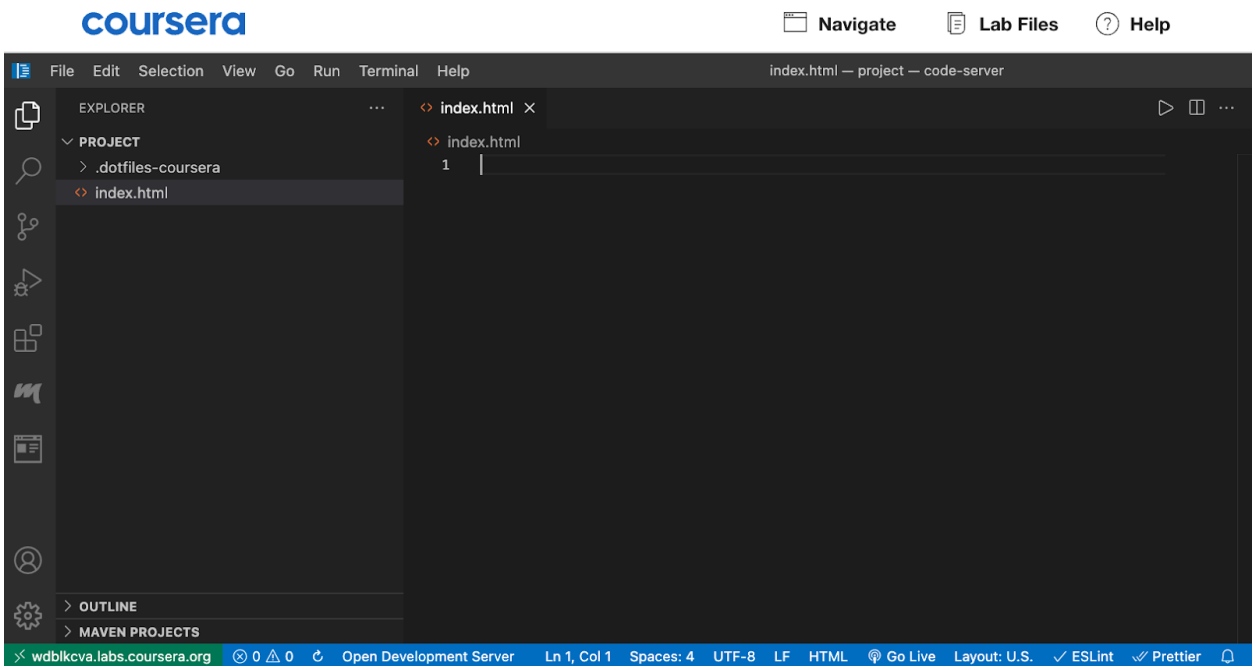
- Create a valid HTML document that displays a piece of text.

Objectives

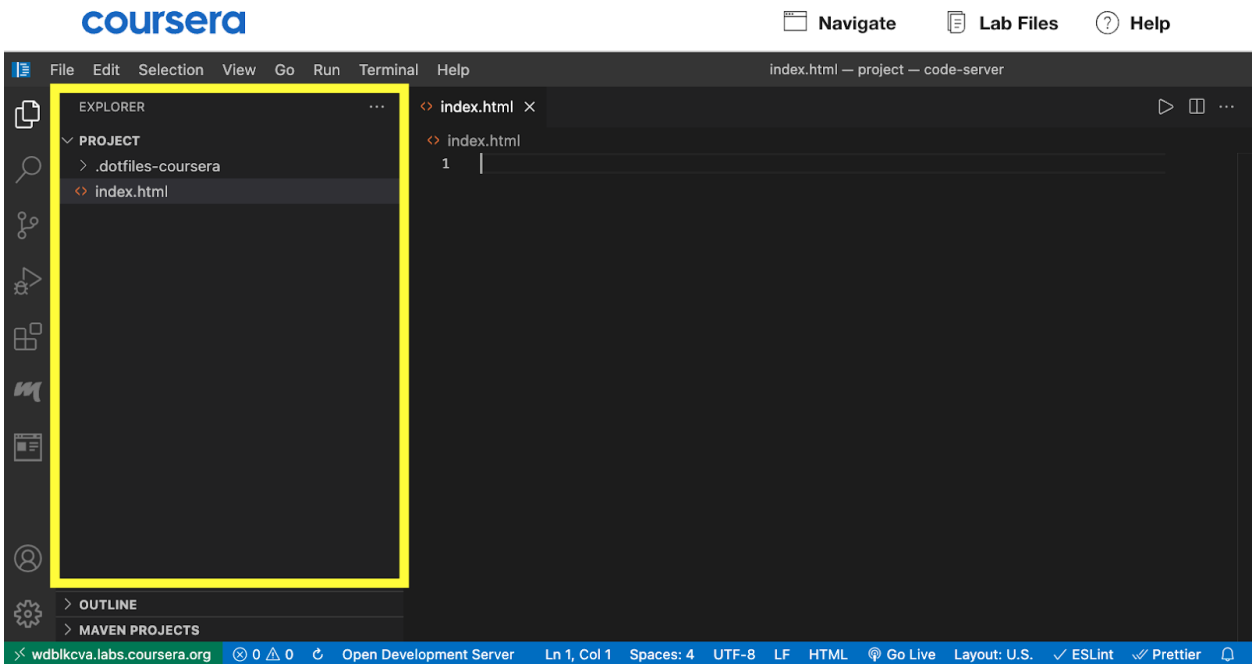
- Add the DOCTYPE.
- Add the HTML, head and body elements.
- Add the title element.
- Add the text to the body element.

Instructions

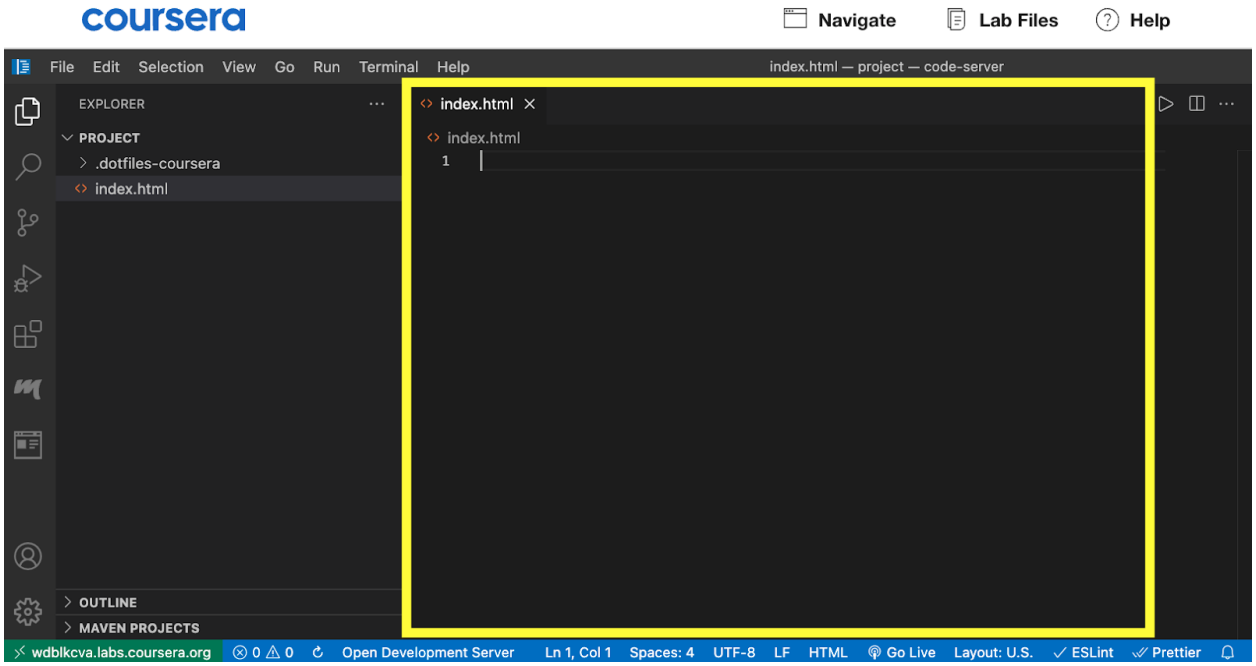
When you click the button to open the lab, a new tab will open with Visual Studio Code already setup and ready for you to start writing code!



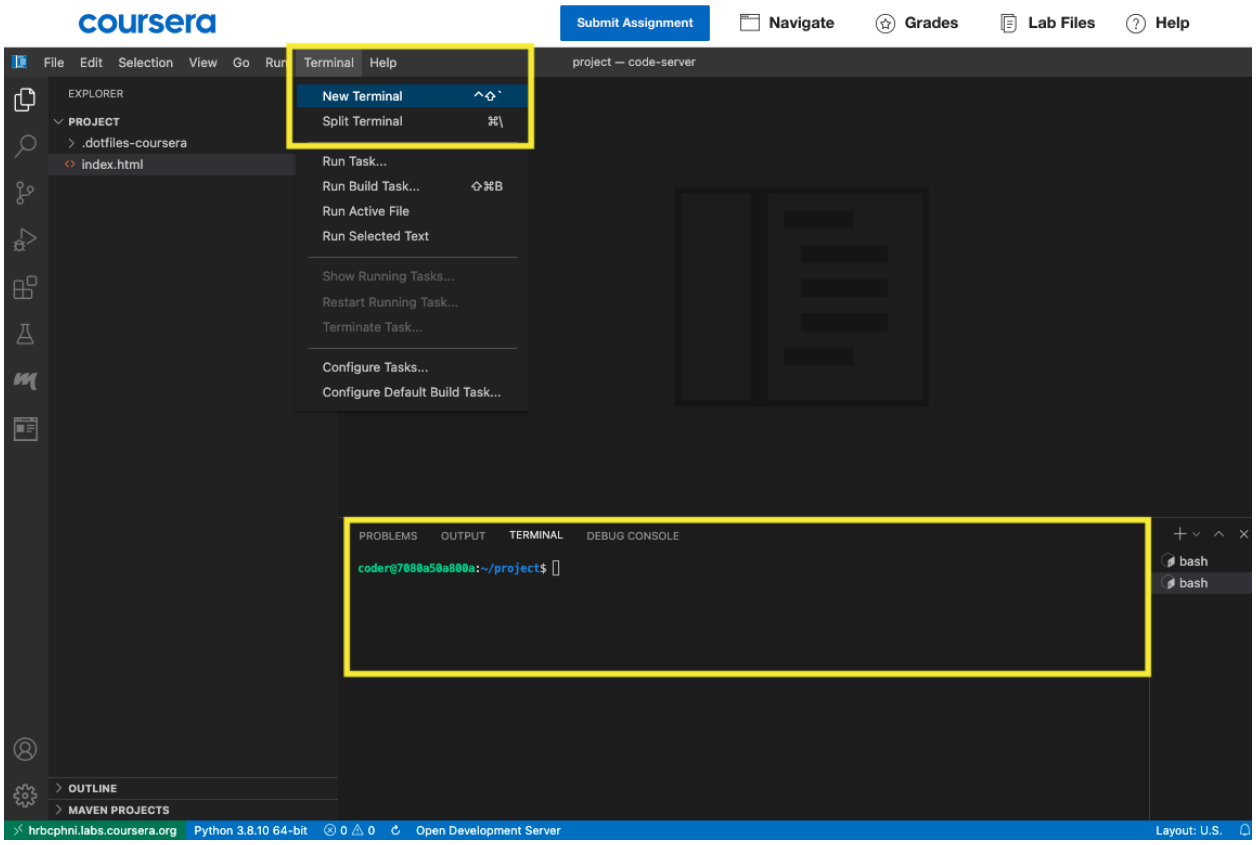
You'll see all the files for the lab in the Project folder in the left sidebar.



And the large editor area where you write your code for the lab.

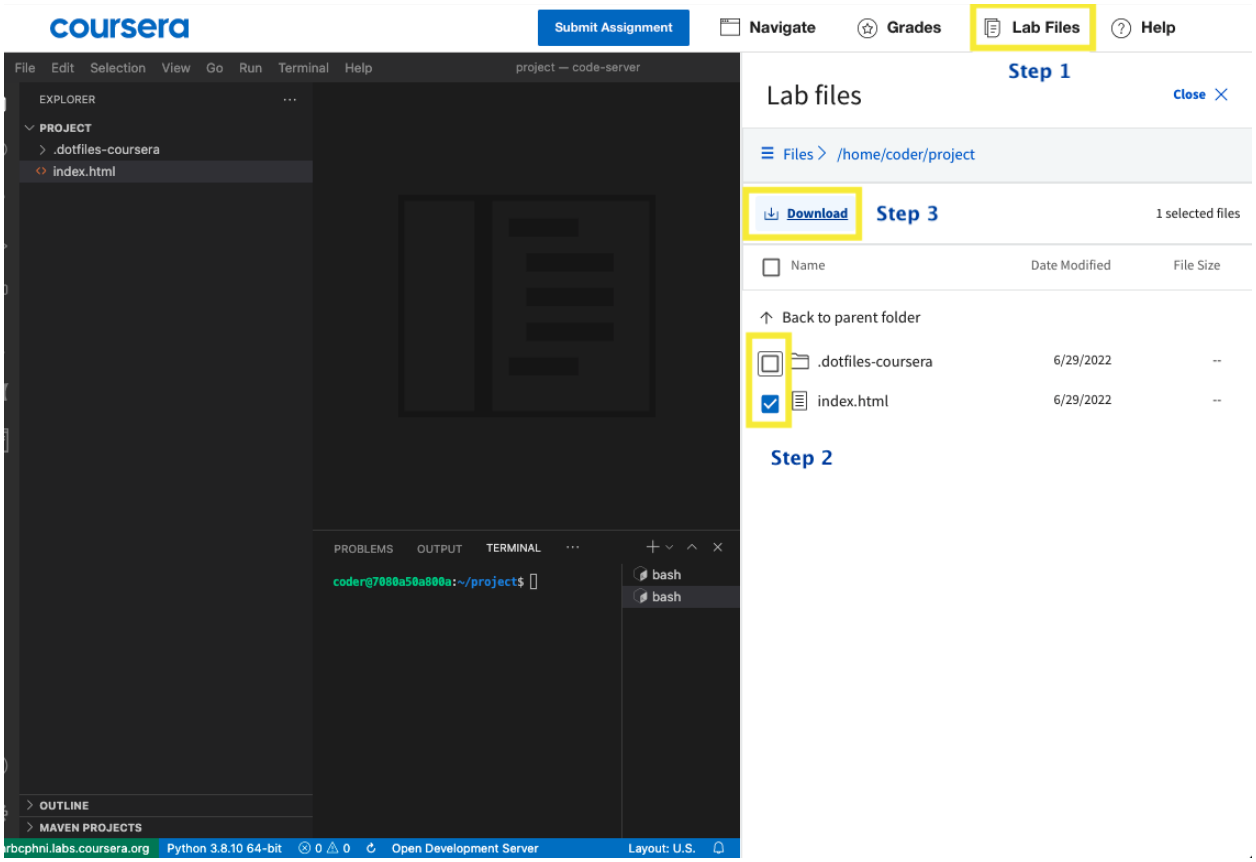


You may need to use a tool called the Terminal from time to time to complete course activities. You can open this by selecting the **Terminal** option in the upper Visual Studio Code toolbar.



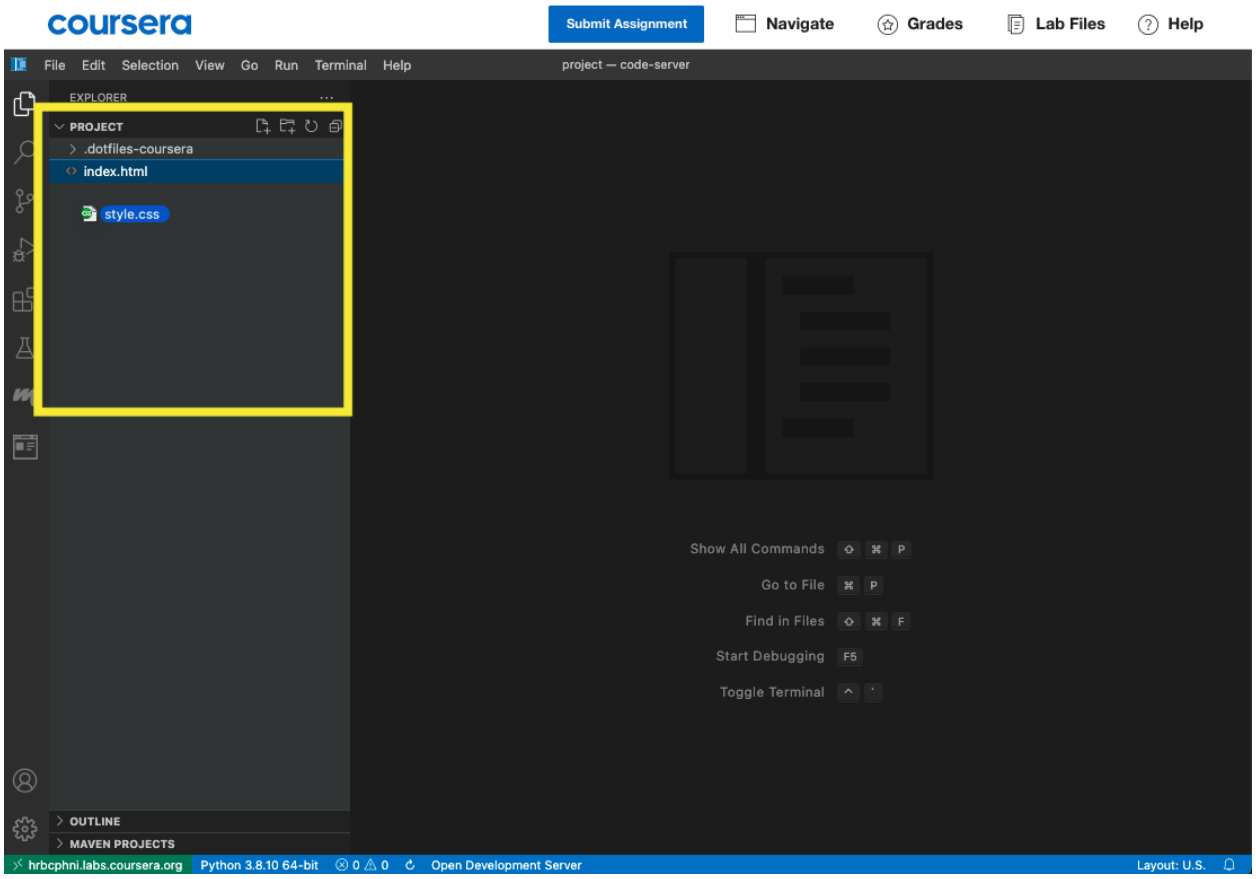
How to download files from your Visual Studio Code Lab to your local device

1. Select the **Lab Files** button in your Lab Toolbar.
2. You'll be able to download your full workspace, specific folders, or individual files through the checkbox selection tool.
3. After you've selected these files, use the **Download** link to download your files to your local device.



How to upload local files to your Visual Studio Code Lab

If you'd like to upload your course files from your local device to your Visual Studio Code lab, **drag and drop** your file from your local device into the Visual Studio Code file tree.

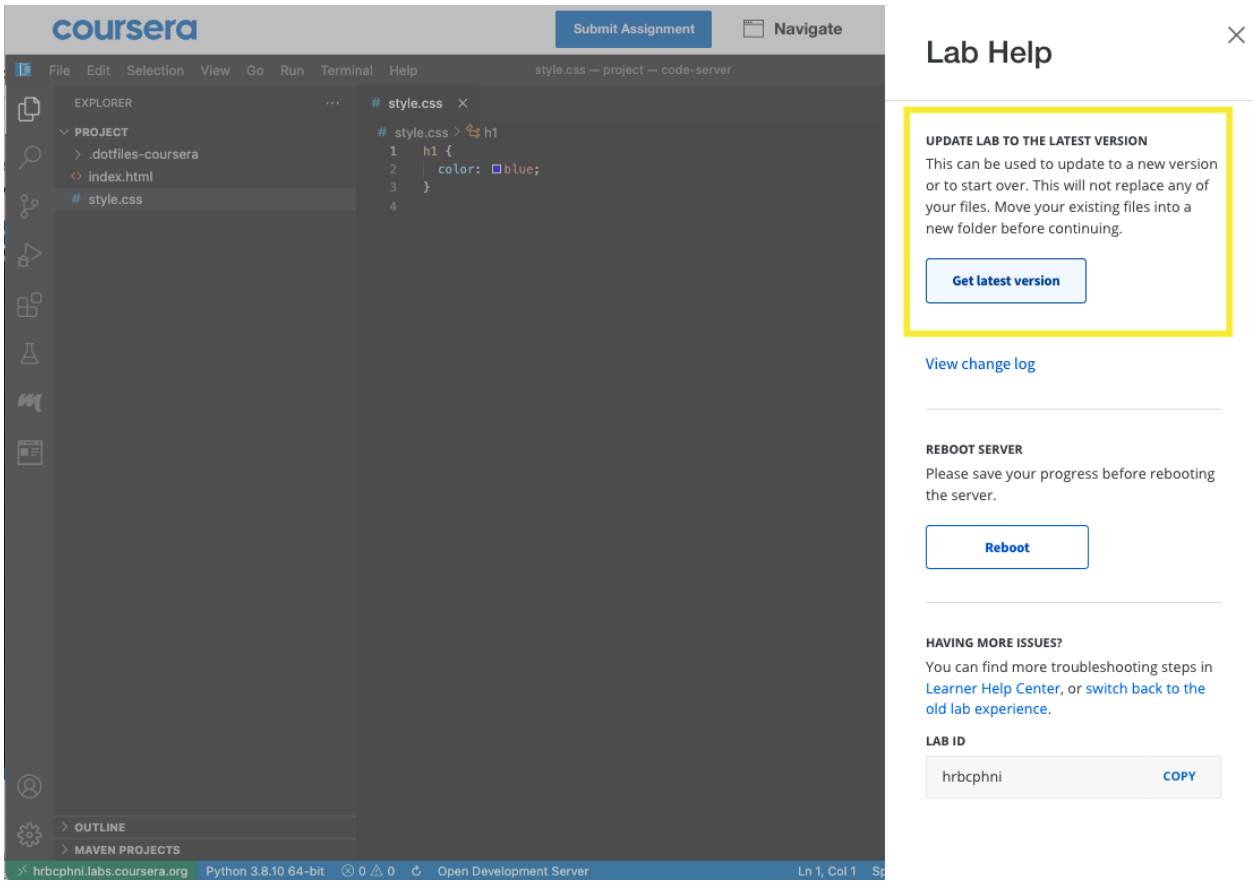


How to get a fresh copy of course-provided starter files

Your work will be saved and persist within your Visual Studio Code lab while you are enrolled in the course. If you'd like to get a fresh copy of the original instructor-provided files at any time, you can do this through the **Lab Help** option in your Lab Toolbar. Don't worry - your original work and files will still remain in your lab until you personally remove or delete them, even when refreshing your files through the steps below.

1. First rename your original files to something like *[yourfilename] [original].[your file extension]*`. You can do this by right-clicking on your file in the Visual Studio Code file tree, selecting **Rename**, and providing a new file name.
    - For example for index.html, this could be renamed to *`index [original].html`*

2. Select **Lab Help** from your Lab Toolbar and then select **Get latest version.**



3. You should now see a fresh copy of the original instructor-provided files in your lab, in addition to your own (renamed) files.

Simple HTML tags

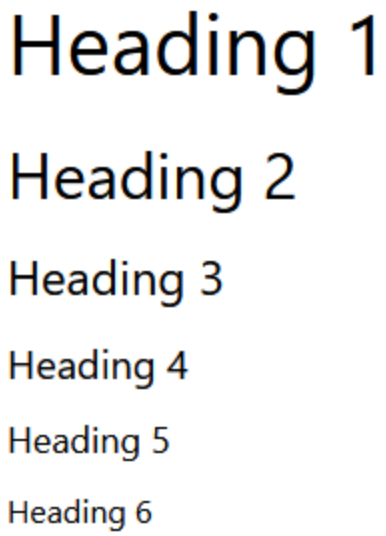
There are many tags available in HTML. Here you will learn about common tags that you'll use as a developer.

Headings

Headings allow you to display titles and subtitles on your webpage.

```
<body>
  <h1>Heading 1</h1>
  <h2>Heading 2</h2>
  <h3>Heading 3</h3>
  <h4>Heading 4</h4>
  <h5>Heading 5</h5>
  <h6>Heading 6</h6>
</body>
```

The following displays in the web browser:



Paragraphs

Paragraphs contain text content.

```
<p>
   This paragraph
   contains a lot of lines
   but they are ignored.
</p>
```

The following displays in the web browser:

This paragraph contains a lot of lines but they are ignored.

Note that putting content on a new line is ignored by the web browser.

Line Breaks

As you've learned, line breaks in the paragraph tag line are ignored by HTML. Instead, they must be specified using the **
** tag. The **
** tag does not need a closing tag.

```
<p>
   This paragraph<br>
   contains a lot of lines<br>
   and they are displayed.
</p>
```

The following displays in the web browser:

This paragraph
contains a lot of lines
and they are displayed.

Strong

Strong tags can be used to indicate that a range of text has importance.

```
<p>
   No matter how much the dog barks: <strong>don't feed him
chocolate</strong>.
</p>
```

The following displays in the web browser:

No matter how much the dog barks: **don't feed him chocolate**.

Bold

Bold tags can be used to draw the reader's attention to a range of text.

```
<p>
   The primary colors are <b>red</b>, <b>yellow</b> and <b>blue</b>.
</p>
```

The following displays in the web browser:

The primary colors are **red**, **yellow** and **blue**.

The following displays in the web browser:

In case of emergency, **push the red button**.

Bold tags should be used to draw attention but not to indicate that something is more important. Consider the following example:

```
The three core technologies of the Internet are <b>HTML</b>, <b>CSS</b>
and <b>Javascript</b>.
```

The following displays in the web browser:

The three core technologies of the Internet are **HTML**, **CSS** and **Javascript**.

Emphasis

Emphasis tags can be used to add emphasis to text.

```
<p>
   Wake up <em>now</em>!
</p>
```

The following displays in the web browser:

Wake up *now*!

Italics

Italics tags can be used to offset a range of text.

```
<p>
   The term <i>HTML</i> stands for HyperText Markup Language.
</p>
```

The following displays in the web browser:

The term *HTML* stands for HyperText Markup Language.

Emphasis vs. Italics

By default both tags will have the same visual effect in the web browser. The only difference is the meaning.
Emphasis tags stress the text contained in them. Let's explore the following example:

```
I <em>really</em> want ice cream.
```

The following displays in the web browser:

I *really* want ice cream.

Italics represent off-set text and should be used for technical terms, titles, a thought or a phrase from another language, for example:

```
My favourite book is <i>Dracula</i>.
```

The following displays in the web browser:

My favourite book is *Dracula*.

Screen readers will not announce any difference if an *italics* tag is used.

Lists

You can add lists to your web pages. There are two types of lists in HTML.
Lists can be unordered using the **<ul>** tag. List items are specified using the **<li>** tag, for example:

```
<ul>
   <li>Tea</li>
   <li>Sugar</li>
   <li>Milk</li>
</ul>
```

This displays in the web browser as:

- Tea
- Sugar
- Milk

Lists can also be ordered using the **<ol>** tag. Again, list items are specified using the **<li>** tag.

```
<ol>
   <li>Rocky</li>
   <li>Rocky II</li>
   <li>Rocky III</li>
</ol>
```

This displays as the following in the web browser.

1. Rocky
2. Rocky II
3. Rocky III

Div tags

A ** ** tag defines a content division in a HTML document. It acts as a generic container and has no effect on the content unless it is styled by CSS.
The following example shows a ** ** element that contains a paragraph element:

```
<div>
   <p>This is a paragraph inside a div</p>
</div>
```

This displays as the following in the web browser.

This is a paragraph inside a div

It can be nested inside other elements, for example:

```
<div>
```

```
    <div>
        <p>This is a paragraph inside a div that's inside another div</p>
    </div>
</div>
```

This displays in the web browser as:

This is a paragraph inside a div that's inside another div

As mentioned, the div has no impact on content unless it is styled by CSS. Let's add a small CSS rule that styles all divs on the page.
Don't worry about the meaning of the CSS just yet, you'll explore CSS further in a later lesson. In summary, you're applying a rule that adds a border and some visual spacing to the element.

```
<style>
    div {
        border: 1px solid black;
        padding: 2px;
    }
</style>
<div>
    <div>
        <p>This is a paragraph inside stylized divs</p>
    </div>
</div>
```

This displays in the web browser as:

This is a paragraph inside stylized divs

Div elements are an important part of building webpages. More advanced usage of div elements will be explored in another course.

Comments

If you want to leave a comment in the code for other developers, it can be added as:
`<!-- This is a comment -->`
The comment will not be displayed in the web browser.

Additional Resources

Learn more Here is a list of resources that may be helpful as you continue your learning journey.
HTML Elements Reference (Mozilla)
https://developer.mozilla.org/en-US/docs/Web/HTML/Element
The Form Element (Mozilla)

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form
What is the Document Object Model? (W3C)
https://www.w3.org/TR/WD-DOM/introduction.html
ARIA in HTML (W3C via Github)
https://w3c.github.io/html-aria/
ARIA Authoring Practices (W3C)
https://www.w3.org/TR/wai-aria-practices-1.2/

Different types of selectors

When styling a web page, there are many types of selectors available that allow developers to be as broad or as specific as they need to be when selecting HTML elements to apply CSS rules to. Here you will learn about some of the common CSS selectors that you will use as a developer.

Element Selectors

The element selector allows developers to select HTML elements based on their element type. For example, if you use `p` as the selector, the rule will apply to all `p` elements on the webpage.

HTML

```
<p>Once upon a time...</p>
<p>In a hidden land...</p>
```

CSS

```
p {
  color: blue;
}
```

ID Selectors

The ID selector uses the id attribute of an HTML element. Since the id is unique within a webpage, it allows the developer to select a specific element for styling. ID selectors are prefixed with a `#` character.

HTML

```
<span id="latest">New!</span>
```

CSS

```
#latest {
  background-color: purple;
}
```

Class Selectors

Elements can also be selected based on the class attribute applied to them. The CSS rule has been applied to all elements with the specified class name. The class selector is prefixed with a `.` character.

In the following example, the CSS rule applies to both elements as they have the `navigation` *CSS class applied to them.*

HTML

```
<a class="navigation">Go Back</a>
```

```
<p class="navigation">Go Forward</p>
```

CSS

```
.navigation {
  margin: 2px;
}
```

Element with Class Selector

A more specific method for selecting HTML elements is by first selecting the HTML element, then selecting the CSS class or ID.
The example below selects all **`p`** *elements that have the CSS class* **`introduction`** *applied to them.*
HTML

```
<p class="introduction"></a>
```

CSS

```
p.introduction {
  margin: 2px;
}
```

Descendant Selectors

Descendant selectors are useful if you need to select HTML elements that are contained within another selector.
Let's explore an example.
You have the following HTML structure and CSS rule.
HTML

```
<div id="blog">
  <h1>Latest News</h1>
  <div>
    <h1>Today's Weather</h1>
    <p>The weather will be sunny</p>
  </div>
  <p>Subscribe for more news</p>
</div>
<div>
  <h1>Archives</h1>
</div>
```

CSS

```
#blog h1 {
  color: blue;
}
```

The CSS rule will select all `h1` *elements that are contained within the element with the ID* `blog`*. The CSS rule will not apply to the* `h1` *element containing the text* `Archives`*.*
The structure of a descendant selector is a CSS selector, followed by a single space character, followed by another CSS selector.
Multiple descendants can also be selected. For example, to select all `h1` *elements that are descendants of* `div` *elements which are descendants of the* `blog` *element, the selector is specified as follows.*
CSS

```
#blog div h1 {
  color: blue;
}
```

Child Selectors

Child selectors are more specific than descendant selectors. They only select elements that are immediate descendants (children) of a selector (the parent).
For example, you have the following HTML structure:
HTML

```
<div id="blog">
  <h1>Latest News</h1>
  <div>
    <h1>Today's Weather</h1>
    <p>The weather will be sunny</p>
  </div>
  <p>Subscribe for more news</p>
</div>
```

If you wanted to style the `h1` *element containing the text* `Latest News`*, you can use the following child selector:*
CSS

```
#blog > h1 {
  color: blue;
}
```

This will select the element with the ID `blog` *(the parent), then it will select all* `h1` *elements that are contained directly in that element (the children). The structure of the child selector is a CSS selector followed by the child combinator symbol* `>` *followed by another CSS selector.*
***Note** that this will not go beyond a single depth level. Therefore, the CSS rule will **not** be applied to the* `h1` *element containing the text* `Today's Weather`*.*

:hover Pseudo-Class

A special keyword called a pseudo-class allows developers to select elements based on their state. Don't worry too much about what that means right now. For now, let's look at how the hover pseudo-class allows you to style an element when the mouse cursor hovers over the element.
The simplest example of this is changing the color of a hyperlink when it is hovered over. To do this, you add the `:hover` *pseudo-class to the end of the selector. In the following example, adding* `:hover` *to the* `a` *element will change the color of the hyperlink to orange when it is hovered over.*
CSS

```
a:hover {
  color: orange;
}
```

This pseudo-class is very useful for creating visual effects based on user interaction.

Other Selectors

There are many other CSS selectors available to style your webpage.

Text and color in CSS

As you design websites, you'll be working a lot with colors and text. There are many different ways to display text and equally as many ways to define colors.
This reading covers how text and color work in CSS.

Color

Colors are used in many CSS properties, for example:

```
p {
  color: blue;
}
```

From CSS Version 3, there are five main ways to reference a color.

- *By RGB value,*
- *By RGBA value,*
- *By HSL value,*
- *By hex value and*
- *By predefined color names.*

RGB value

RGB is a color model that adds the colors red (R), green (G) and blue (B) together to create colors. This is based on how the human eye sees colors.
Each value is defined as a number between `0` *and* `255`*, representing the intensity of that color.*
For example, the color red would have the RGB value of `255,0,0` *since the intensity of the red color would be 100% while blue and green would be 0%.*
The color black then would be `0,0,0` *and the color white* `255,255,255`*.*

When using RGB values in CSS, they can be defined using the `rgb` *keyword:*

```
p {
  color: rgb(255, 0, 0);
}
```

RGBA value

RGBA is an extension of RGB that add an alpha (A) channel. The alpha channel represents the opacity, or transparency, of the color.

Similar to RGB, this is specified in CSS using the `rgba` *keyword:*

```
p {
  color: rgba(255, 0, 0, 128);
}
```

```
1
2
3
```

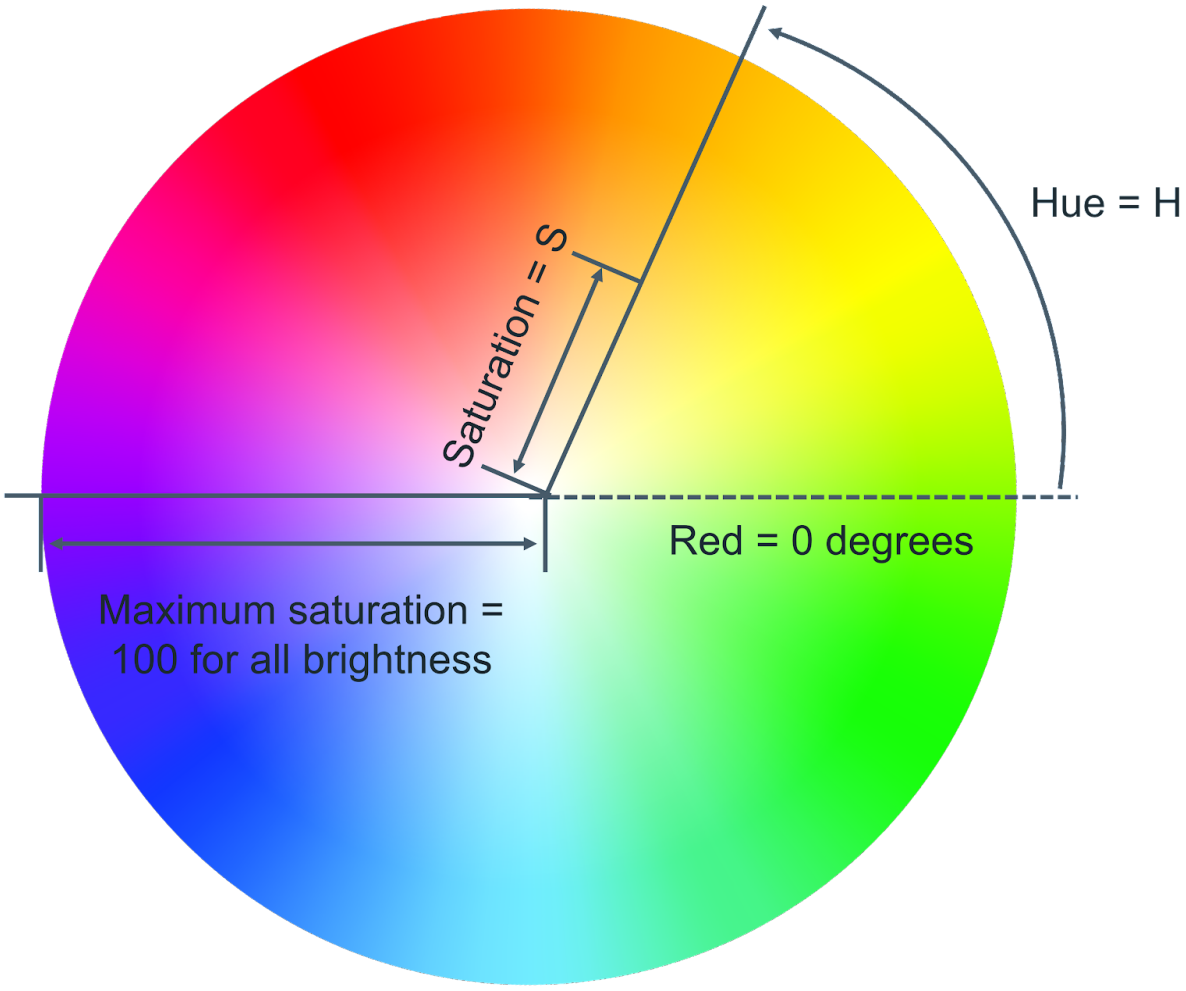
```
p {

  color: rgba(255, 0, 0, 128);

}
```

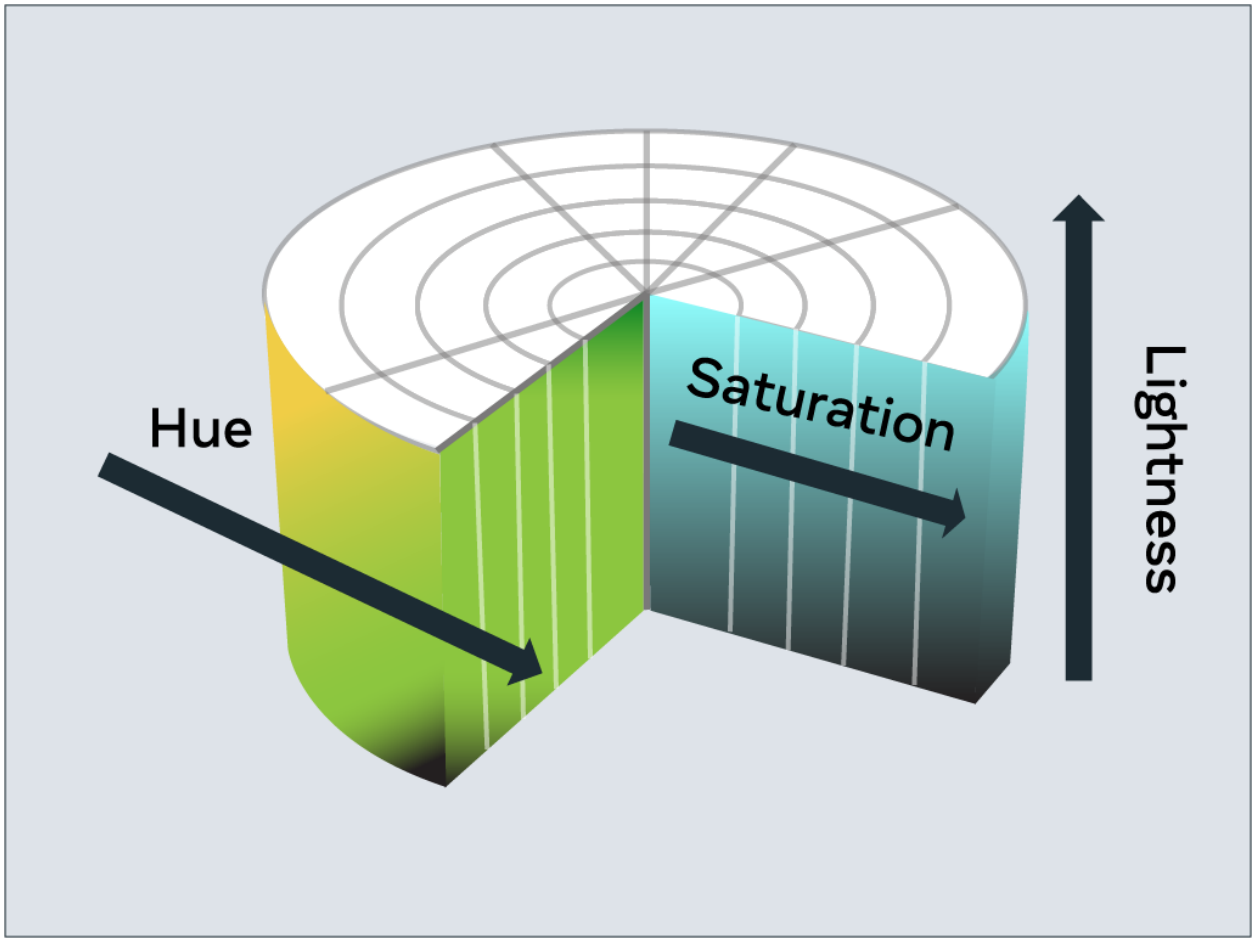
HSL value

HSL is a newer color model defined as Hue (H), Saturation (S) and Lightness (L). The aim of the model is to simplify mental visualization of the color that the value represents. Think of a rainbow that has been turned into a full circle. This represents the Hue. The Hue value is the degree value on this circle, from 0 degrees to 360 degrees. 0 is red, 120 is green and 240 is blue.



Saturation is the distance from the center of the circle to its edge. The saturation value is represented by a percentage from 0% to 100% where 0% is the center of the circle and 100% is its edge. For example, 0% will mean that the color is more grey and 100% represents the full color.

Lightness is the third element of this color model. Think of it as turning the circle into a 3D cylinder where the bottom of the cylinder is more black and toward the top is more white. Therefore, lightness is the distance from the bottom of the cylinder to the top. Again, lightness is represented by a percentage from 0% to 100% where 0% is the bottom of the cylinder and 100% is its top. In other words, 0% will mean that the color is more black and 100% is white.



In CSS, you use the `hsl` *keyword to define a color with HSL.*

```
p {
  color: hsl(0, 100%, 50%);
}
```

Hex value

Colors can be specified using a hexadecimal value. If you're unfamiliar with hexadecimal, think of it as a different number set.

Decimal is what you use every day. Digits range from `0` *to* `9` *before tens and hundreds are used.*

Hexadecimal is similar, except it has 16 digits. This is counted as `0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F`.

In fact, you can convert between decimal and hexadecimal. Decimal `10` *is equal to hexadecimal* `A`. *Hexadecimal* `F` *is equal to decimal* `15`.

Hexadecimal can also go to tens and hundreds. For example, decimal `16` *is equal to hexadecimal* `10`, *with* `10` *being the next number after* `F`.

It can be a little confusing at first but don't worry, there are plenty of converters available if you get stuck.

Colors specified using hexadecimal are prefixed with a `#` symbol followed by the RGB value in hexadecimal format.
For example, the color red which is RGB `255,0,0` would be written as hexadecimal `#FF0000`.
Again don't worry if you get stuck, there are plenty of converters available for this too!

Predefined color names

Modern web browsers support 140 predefined color names. These color names are for convenience purposes and can be mapped to equivalent hex/RGB/HSL values.
Some common color names available are listed below.
black
silver
gray
white
maroon
red
purple
fuchsia
green
lime
olive
yellow
navy
blue
teal
aqua

Text

With CSS there are many ways to change how text is displayed. In this section, you'll learn the most common text manipulation CSS properties.

Text Color

The `color` property sets the color of text. The following CSS sets the text color for all paragraph elements to red.

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

Text Font and Size

There are many different fonts to display text on your computer. In simple terms, a font is a collection of text characters written in a specific style and size.
If you've used a word processor before, you're probably familiar with the fonts Times New Roman and Calibri.

To set the font used by text in CSS you use the **font-family** property.

```
p {
  font-family: "Courier New", monospace;
}
```

Since computers vary in what fonts they have installed, it is recommended to include several fonts when using the **font-family** property. These are specified in a fallback order, meaning that if the first font is not available, it will check for the second font. If the second font is not available, then it will check for the third font and so on. If none of the fonts are available, it will use the browser's default font.

To set the size of the font, the **font-size** property is used.

```
p {
  font-family: "Courier New", monospace;
  font-size: 12px;
}
```

Text Transformation

Text transformation is useful if you want to ensure the correct capitalization of the text content. In the example below, the CSS rule will change all text in paragraph elements to uppercase using the **text-transform** property:

```
p {
  text-transform: uppercase;
}
```

The most commonly used values for the **text-transform** property are: **uppercase**, **lowercase**, **capitalize** and **none**. The default value used is **none**, which means the text displays as it was written in the HTML document.

Text Decoration

The **text-decoration** property is useful to apply additional decoration to text such as underlining and line-through (strikethrough).

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

It is possible to set the color, thickness and styling of the decoration too. In the example below, the underline will be a solid red line that is 5 pixels thick.

```
p {
  text-decoration: underline red solid 5px;
}
```

If this is confusing, don't worry. These properties can be individually set using the **text-decoration-line**, **text-decoration-color**, **text-decoration-style** and **text-decoration-thickness** properties. Let's use the same example again and define it using the individual properties:

```
p {
  text-decoration-line: underline;
```

```
  text-decoration-color: red;
  text-decoration-style: solid;
  text-decoration-thickness: 5px;
}
```

The most common **`text-decoration-line`** *values used are:* **`underline`**, **`overline`**, **`line-through`** *and* **`none`**. *None is the default value to use no text decoration.*
There are many styles available for the **`text-decoration-style`** *property;* **`solid`**, **`double`**, **`dotted`**, **`dashed`** *and* **`wavy`**. *The* **`text-decoration-style`** *property requires the decoration line to be defined. If the decoration style is not specified,* **`solid`** *will be used.*

lignment basics

Let's explore how to align text and HTML elements using CSS.
Let's first focus on horizontal alignment. Vertical alignment is more difficult so you'll explore that later on.

Text Alignment

Aligning text within an HTML element is very simple. To do this, you use the **`text-align`** *CSS property. In the following example, the CSS rule is setting the text of all paragraph elements to be center aligned.*

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

Text alignment can be set to **`left`**, **`right`**, **`center`** *and* **`justify`**.
The **`justify`** *alignment spreads the text out so that every line of the text has the same width.*
The default alignment is **`left`** *for languages that are left-to-right such as English. For right-to-left languages such as Arabic, the default alignment is* **`right`**.

HTML Element Alignment

HTML element alignment is more complicated than text alignment. To align HTML elements, you must consider the box model and document flow from previous lessons. Aligning an HTML element is done by changing the properties of its box model and how it impacts the document flow.

HTML Element Center Alignment

To center align an element, you set a width on the element and push its margins out to fill the remaining available space of the parent element as in the following HTML structure:

```
<div class="parent">
  <div class="child">
  </div>
</div>
```

In your CSS, you'll set the **`parent`** *element to have a red border to visualize the space it occupies:*

```
.parent {
  border: 4px solid red;
}
```

The **`child`** *element will have a width equal to 50% of the* **`parent`** *element with a padding of 20 pixels. Note that* **`padding: 20px`** *is shorthand for setting the padding top, bottom, left and right to* **`20px`***. To visualize the space it occupies, set the border to green:*

```
.child {
  width: 50%;
  padding: 20px;
  border: 4px solid green;
}
```

To align the element to the center, set its **`margin`** *property to* **`auto`***. The* **`auto`** *will tell the browser to calculate the margin automatically based on the space available.*

```
.child {
  width: 50%;
  padding: 20px;
  border: 4px solid green;
  margin: auto;
}
```

The result is the **`child`** *element is centered within the* **`parent`** *element:*

It is important to note that this works because the **`div`** *element is a block-level element. If you want to align an inline element like* **`img`***, you will need to change it to a block-level element. Similar to the* **`div`** *example, you add the* **`img`** *to a parent element:*

```
<div class="parent">
  <img src="photo.png" class="child">
</div>
```

The CSS rule then changes the **`img`** *element to a block-level element and sets its margin to* **`auto`**:

```
.child {
  display: block;
  width: 50%;
  margin: auto;
}
```

To be more precise, in CSS you can set only the left and right margins to auto. This allows you to set the top and bottom margins to specific values if needed.

```
.child {
  display: block;
  width: 50%;
  margin-left: auto;
  margin-right: auto;
}
```

HTML Element Left / Right Alignment

The two most common ways to left and right align elements are to use the **`float`** *property and the* **`position`** *property.*

The **`position`** *property has several value options that impact how the element displays in the document flow. You'll explore how to use the* **`position`** *property later on. For now, let's focus on the* **`float`** *property.*

The **`float`** *property sets an element's position relative to the text content within a parent element. Text will wrap around the element.*

In the following example, the image will be aligned to the right of the **`div`** *element. The text content will wrap around the image:*

HTML

```
<div class="parent">
  <img src="photo.png" class="child"> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur eu odio eget leo auctor porta sit amet sit amet justo. Donec fermentum quam in diam volutpat, at lacinia diam placerat. Aenean quis feugiat sem. Suspendisse a dui massa. Phasellus scelerisque, mi vestibulum iaculis tristique, orci tellus gravida nisi, in pellentesque elit massa ut lorem. Sed elementum ornare nunc vel cursus. Duis sed enim in nulla efficitur convallis sed eget dolor. Curabitur scelerisque eros erat, in vulputate dolor consequat vel. Praesent ac sapien condimentum, ultricies libero at, auctor orci. Curabitur ut augue ac massa convallis faucibus sed in magna. Phasellus scelerisque auctor est a auctor. Nam laoreet sem sapien, porta imperdiet urna laoreet eu. Morbi dolor turpis, congue id bibendum eget, viverra et risus. Quisque vitae erat id tortor ullamcorper maximus.
</div>
```

CSS

```
.child {
  float: right;
}
```

The following displays in the web browser:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur eu odio eget leo auctor porta sit amet sit amet justo. Donec fermentum quam in diam volutpat, at lacinia diam placerat. Aenean quis feugiat sem. Suspendisse a dui massa. Phasellus scelerisque, mi vestibulum iaculis tristique, orci tellus gravida nisi, in pellentesque elit massa ut lorem. Sed elementum ornare nunc vel cursus. Duis sed enim in nulla efficitur convallis sed eget dolor. Curabitur scelerisque eros erat, in vulputate dolor consequat vel. Praesent ac sapien condimentum, ultricies libero at, auctor orci. Curabitur ut augue ac massa convallis faucibus sed in magna. Phasellus scelerisque auctor est a auctor. Nam laoreet sem sapien, porta imperdiet urna laoreet eu. Morbi dolor turpis, congue id bibendum eget, viverra et risus. Quisque vitae erat id tortor ullamcorper maximus.

Additional resources

Learn more *Here is a list of resources that may be helpful as you continue your learning journey.*
CSS Reference (Mozilla)
https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
HTML and CSS: Design and build websites by Jon Duckett
https://www.amazon.com/HTML-CSS-Design-Build-Websites/dp/1118008189/
CSS Definitive Guide by Eric Meyer
https://www.amazon.com/CSS-Definitive-Guide-Visual-Presentation/dp/1449393195/
Mark as completed
Like
Dislike
Report an issue

Bootstrap

Bootstrap is often described as a way to "build fast, responsive sites" and it is a "feature-packed, powerful, and extensible frontend toolkit".
Some people refer to it as a "front-end" framework, and some are trying to be more specific by referring to it as a "CSS framework" or a “CSS library”.
So, what is Bootstrap?
Simply put, Bootstrap is a library of CSS and JavaScript code that you can combine to quickly build visually appealing websites.
Modern web development is all about **components**. Small pieces of reusable code that allow you to build websites quickly. Bootstrap comes with multiple components for very fast construction of multiple components, or parts of components.

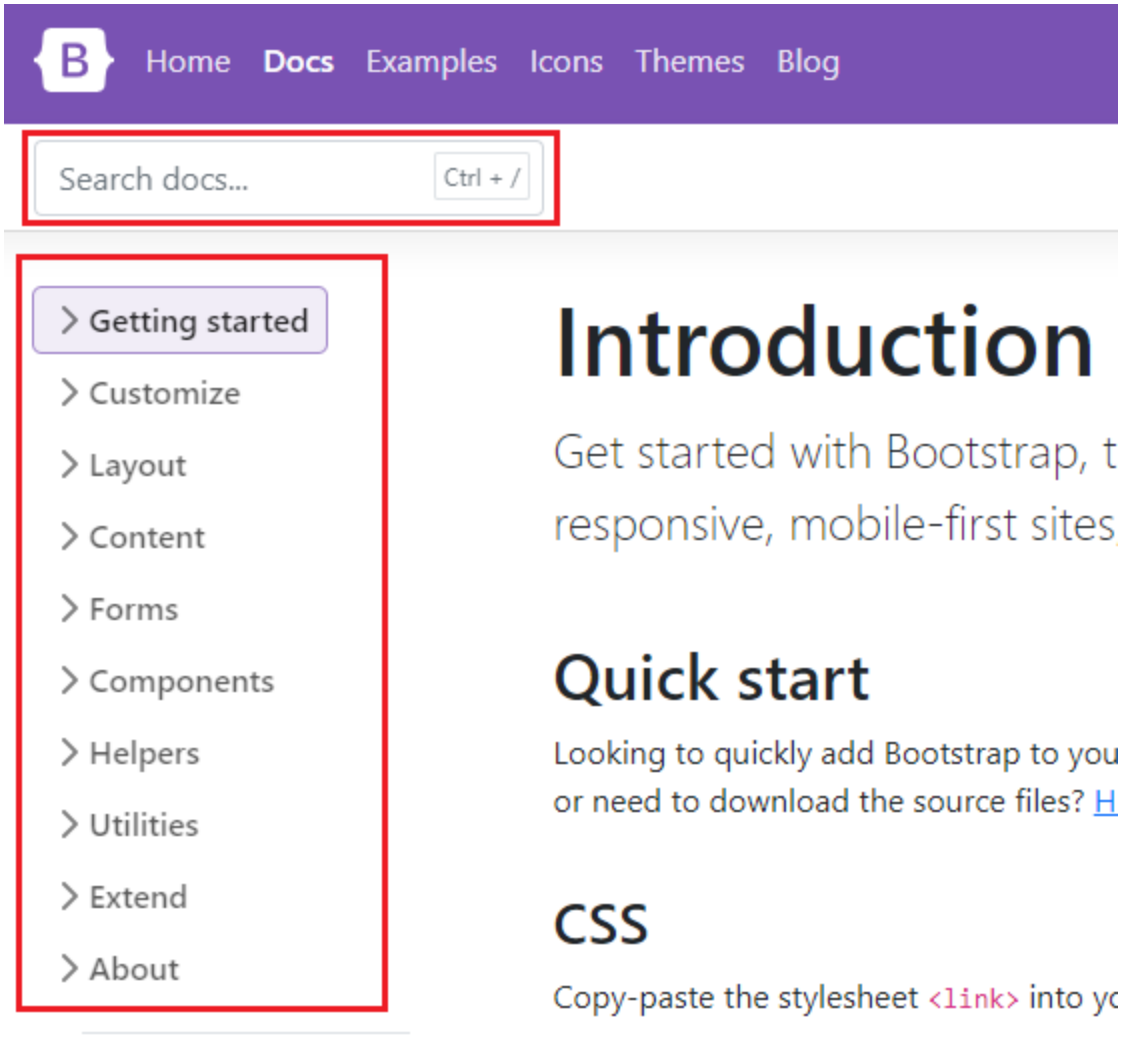
Another important aspect of modern development is **responsive grids** which allow web pages to adapt their layout and content depending on the device in which they are viewed. Bootstrap comes with a pre-made set of CSS rules for building a responsive grid.
Bootstrap is very popular amongst developers as it saves development time and provides a way for developers to build visually appealing prototypes and websites.
Bootstrap saves significant time because all the CSS code that styles its grid and pre-built components is already written. Instead of having to have a high level of expertise in various CSS concepts, you can just use the existing Bootstrap CSS classes to produce nicely-looking websites. This is indispensable when you need to quickly iterate on website layouts.
Once you know how Bootstrap works, you'll have enough knowledge to tweak its styling and a whole new world of development opens up to you.
Since Bootstrap is so popular, understanding how to work with it is a prerequisite in many web development companies. Additionally, you can be safe in knowing that both you and your team members have a common design system and you don't have to spend time deciding how to build one. You are free to jump from team to team, from project to project, even from one company to another, and you don't need to re-learn "their way of doing things".
All of these points make investing time to learn Bootstrap a great way to boost your web development skills. In this lesson, you'll be introduced to the core concepts of Bootstrap and learn how to build web pages using it.

Using Bootstrap documentation

Bootstrap comes with detailed documentation on setting up and using the features available in its library. The documentation is clear and has many code examples to help you get started.
In this reading, you'll explore the frequently used documentation sections.
The documentation for Bootstrap is currently available at the following link.
https://getbootstrap.com/docs

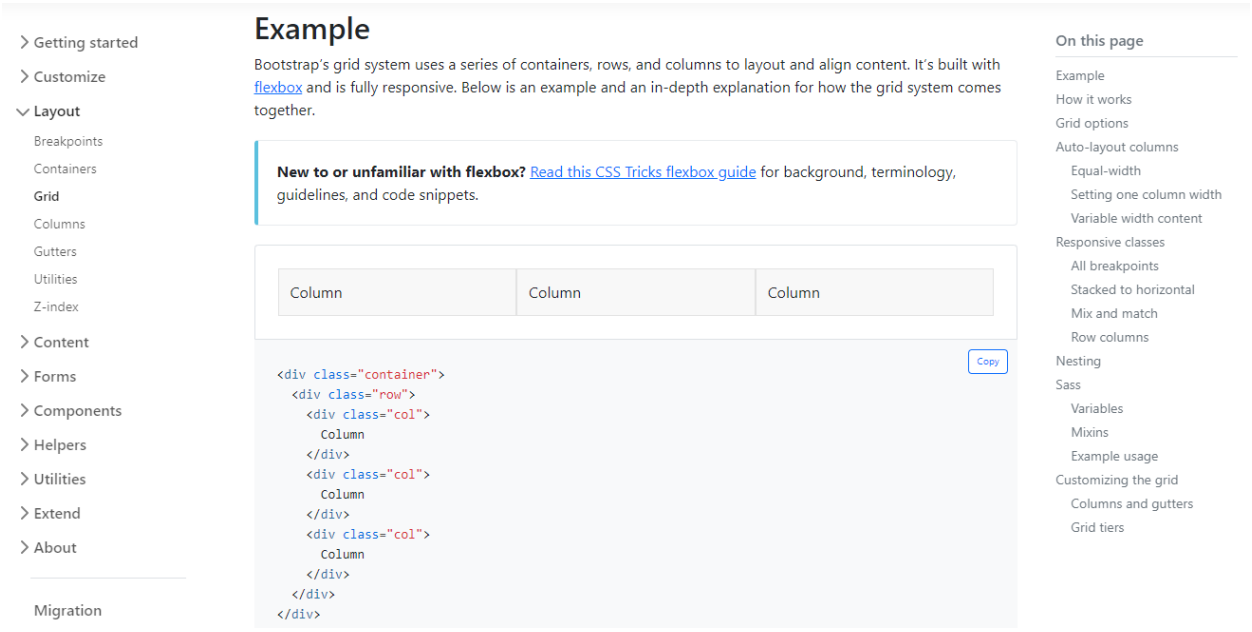
Navigating the documentation

The sidebar on the webpage allows you to navigate through the different sections of the documentation. There is also a search box if you need to search for a specific piece of information.



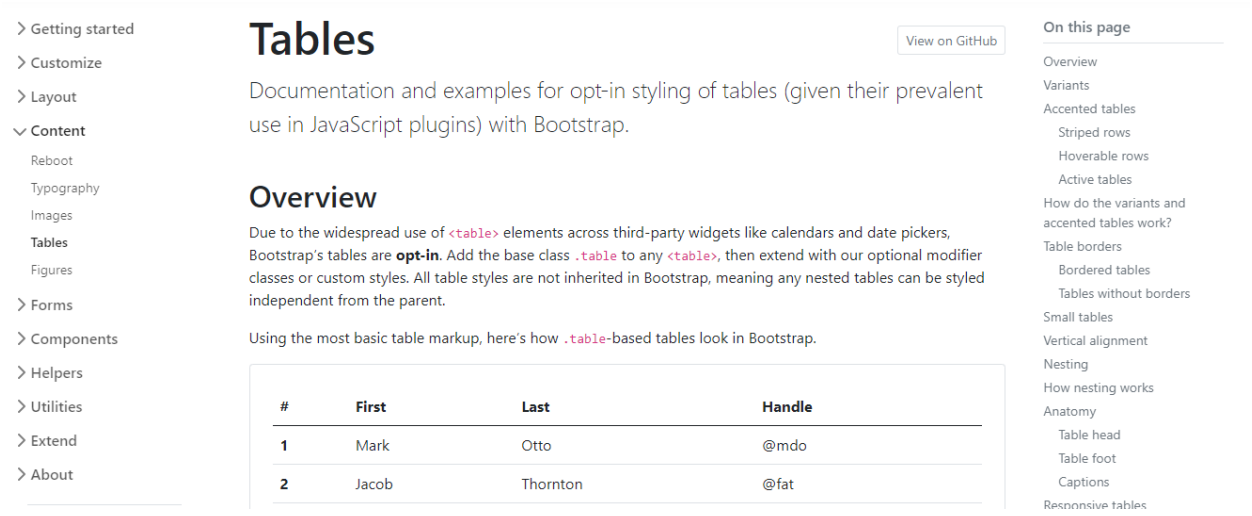
Layout

The layout section of the documentation describes how to use the grid system of Bootstrap. This covers what you've learned so far and includes more advanced usage such as offsets, column alignment, auto-layout and variable width columns.



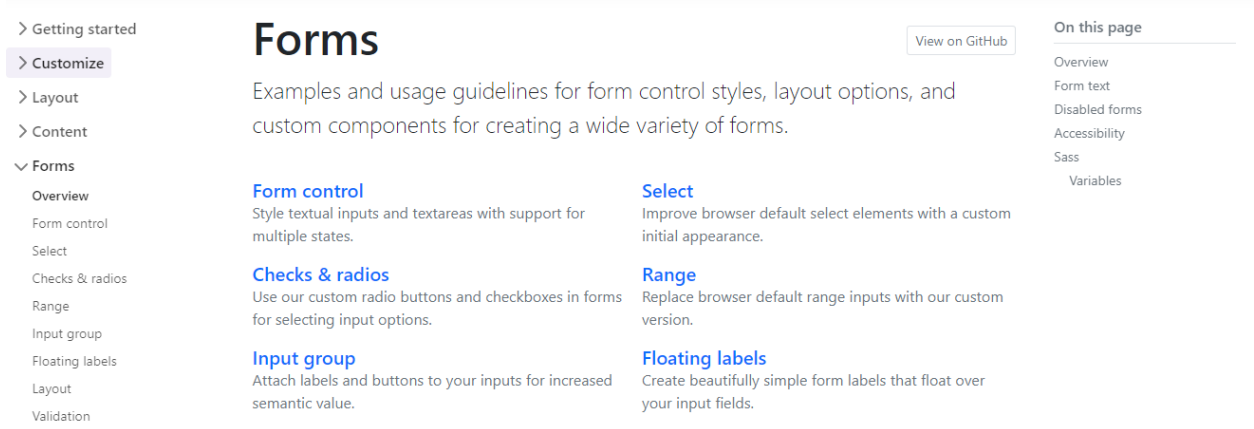
Content

The content section of the documentation describes Bootstrap's default text styling and how to use responsive images and tables. You've learned the basics of these earlier on and this section goes into further detail.



Forms

The forms section of the documentation describes how to build forms using Bootstrap's styles. The library has many CSS rules to improve your form's user interface and experience. Below are some features you'll frequently use as a developer:



Form Styling

Bootstrap includes CSS rules to improve the visual style of input elements.
For example:

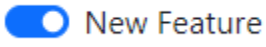
This table outlines the different HTML form elements and which Bootstrap CSS class should be used for them.

Form Element	CSS class
`input`	`form-control`
`input type="checkbox"`	`form-check-input`
`input type="radio"`	`form-check-input`
`input type="range"`	`form-range`
`select`	`form-select`

Using these CSS classes will style the elements appropriately for different input types, sizings and states. More information is available on the Forms documentation page.

Switches

If you've used an app on your mobile device, you're probably familiar with the switch input type.



Bootstrap includes CSS rules to style checkbox input elements as switches.
To do this:

1. Add the `input` to a `div` element.
2. On the `div` element, apply the `form-check` and `form-switch` CSS classes.
3. On the `input` element, add the `form-check-input` CSS class.

```
<div class="form-check form-switch">
  <input class="form-check-input" type="checkbox">
</div>
```

More information is available in the Switches section of the documentation.

Input Groups

Input groups are useful for providing additional content to the input field. For example, if you wanted to request the user to input a US dollar amount, you can use an input group to show the dollar symbol and cents amount.

To do this:

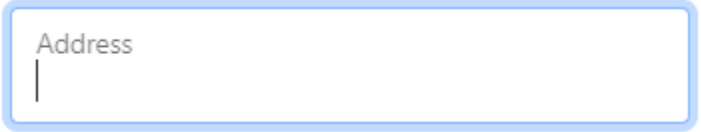
1. Add the **`input`** to a **`div`** element.
2. Apply the **`input-group`** CSS classes on the **`div`** element.
3. Add a **`span`** element before and/or after the **`input`** element and apply the **`input-group-text`** CSS class to it. The text content is then added inside the **`span`** element.

```
<div class="input-group">
  <span class="input-group-text">$</span>
  <input type="text" class="form-control">
  <span class="input-group-text">.00</span>
</div>
```

More information is available on the Input Groups documentation page.

Floating Labels

Floating labels help provide form information to the user as part of the input itself. These are different from regular form placeholders. The information stays visible if the user is interacting with the element or if the element has content.

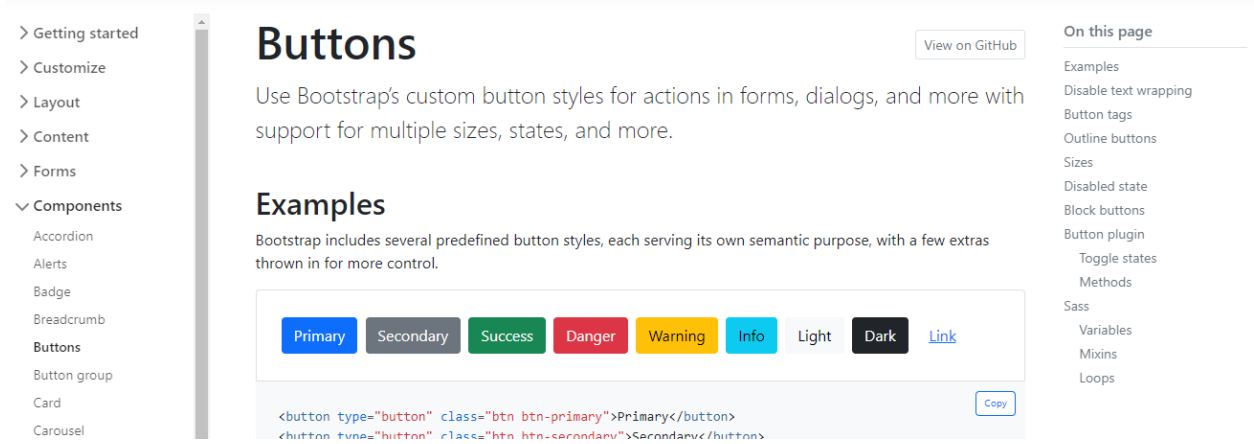


To do this, add the **`input`** to a **`div`** element. On the **`div`** element, apply the **`form-floating`** CSS classes.

```
<div class="form-floating">
  <input type="email" class="form-control" id="addressInput"
placeholder="Address">
  <label for="addressInput">Address</label>
</div>
```

Components

As you have learned, Bootstrap comes with many pre-made UI elements and styles to help speed up your development.
Some of these components require Javascript to work, while others only require CSS classes applied to HTML elements. The Components section of the documentation explains these requirements on each component page and provides many code examples.



Conclusion

Now that you are familiar with how to use the Bootstrap documentation, maybe try some new components and styles on a webpage that you've previously built.

Other CSS frameworks and libraries

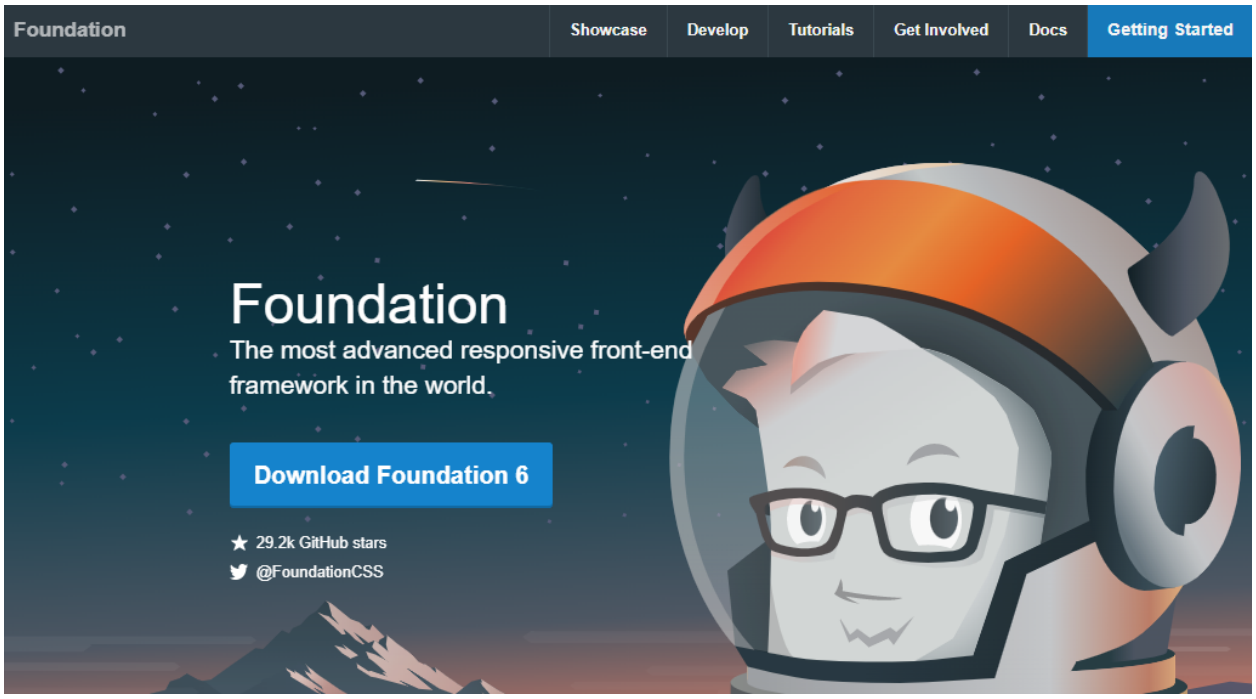
Other CSS frameworks and libraries

As a developer, you'll use many CSS libraries and frameworks throughout your career. As you move on to different projects and as technologies advance, knowing what solutions are available is critical. While Bootstrap is one of the most popular CSS libraries, many others are available, each with different purposes, designs and technical approaches. This reading will introduce you to other popular CSS libraries and frameworks.

Foundation

Official Website

Foundation is a framework for building user interfaces similar to Bootstrap. It is used by many large companies such as Pixar, Polar and Sonos. One prominent feature of Foundation is that it can be used to style content for sending via email.



Pure.css

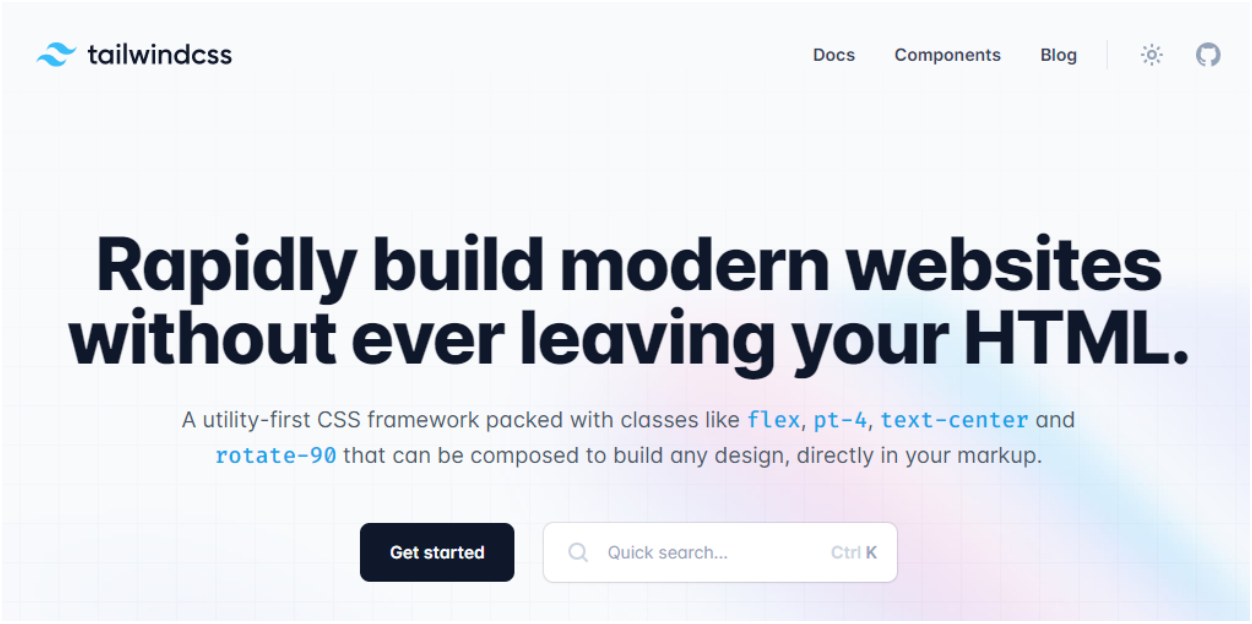
Official Website

Pure.css is another library for building user interfaces. While it doesn't have as many features as Bootstrap, it is designed to be minimal in file size. Smaller file sizes improve loading times for web pages as there is less data to transfer from the web server. If your next project is focused on minimal loading time, this library is worth considering.

Tailwind CSS

Official Website

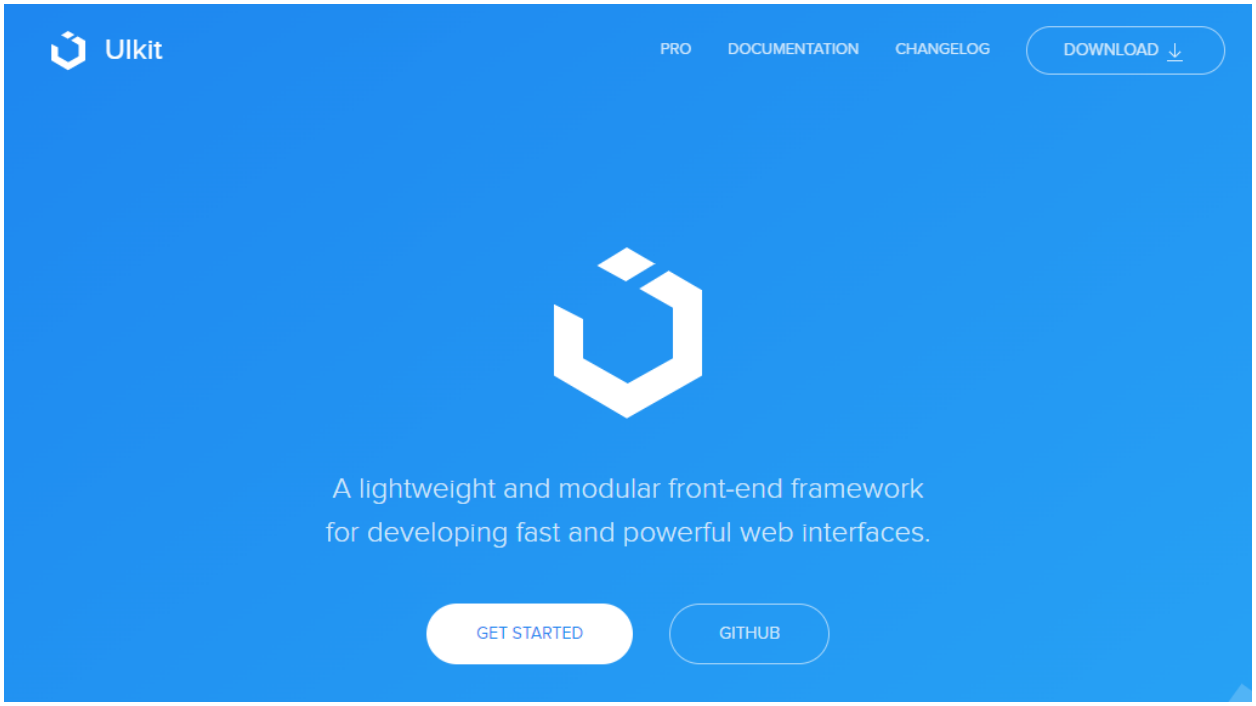
Tailwind CSS is a CSS framework that uses a utility-based approach for its CSS rules. This means that the framework provides many CSS classes with a single purpose. For example, the CSS class pt-6 sets the padding-top CSS property to 6 pixels. This means that you can be precise in applying styling to your HTML without writing CSS. The advantage to this is that it is more flexible for customizing your webpage's design using the framework. However, the disadvantage is that if multiple developers are working on a project, it could lead to inconsistent design if the team is not strict on its design rules.



UIKit

Official Website

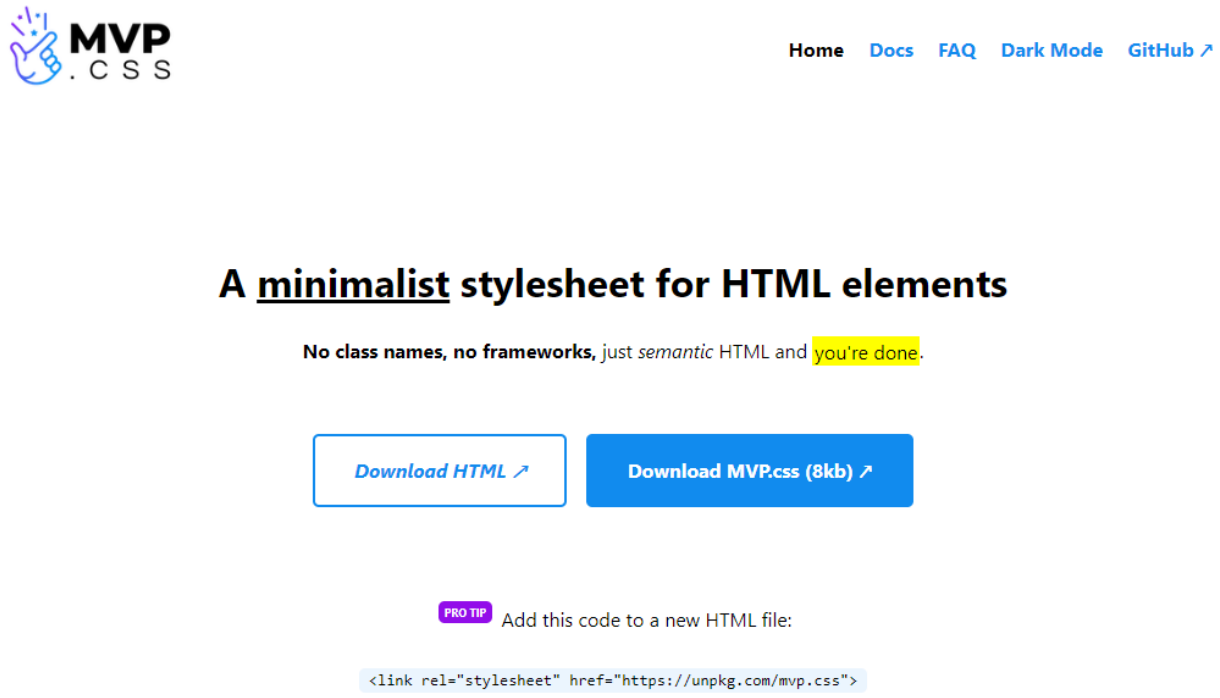
UIKit is a lightweight CSS framework featuring a small set of responsive components. Its simple design allows developers to easily customize the style rules and visuals.



MVP.css

Official Website

MVP.css is a small CSS library that automatically styles HTML elements without needing to apply CSS classes to them. The library aims to allow a developer to quickly prototype a user interface without worrying about the final design, while still being visually appealing. MVP comes from the technical term Minimal Viable Product, a product with sufficient features to demo to customers or other business stakeholders.



Conclusion

If you're curious to learn more about these frameworks, their websites feature set up guides, tutorials and documentation to get started. It is a good exercise to compare and contrast different libraries and frameworks to understand different workflows available to you as a developer.

Additional Resources

Bootstrap Official Website
https://getbootstrap.com/
Bootstrap 5 Foundations by Daniel Foreman
https://www.amazon.com/Bootstrap-Foundations-Mr-Daniel-Foreman/dp/B0948GRS8W/
Responsive Web Design with HTML5 and CSS by Ben Frain
https://www.amazon.com/Responsive-Web-Design-HTML5-CSS/dp/1839211563/
Bootstrap Themes
https://themes.getbootstrap.com/

Case Study: Why did Facebook engineers create React?

There are a lot of JavaScript Model-View-Controller (MVC) frameworks out there. Why did we build React and why would you want to use it?

React isn't an MVC framework.

React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time.

React doesn’t use templates.

Traditionally, web application UIs are built using templates or HTML directives. These templates dictate the full set of abstractions that you are allowed to use to build your UI.
React approaches building user interfaces differently by breaking them into **components**. This means React uses a real, full-featured programming language to render views, which we see as an advantage over templates for a few reasons:

- **JavaScript is a flexible, powerful programming language** with the ability to build abstractions. This is incredibly important in large applications.
- By unifying your markup with its corresponding view logic, React can actually make views **easier to extend and maintain**.
- By baking an understanding of markup and content into JavaScript, there’s **no manual string concatenation** and therefore less surface area for XSS vulnerabilities.

We’ve also created JSX, an optional syntax extension, in case you prefer the readability of HTML to raw JavaScript.

React updates are dead simple.

React really shines when your data changes over time.
In a traditional JavaScript application, you need to look at what data changed and imperatively make changes to the DOM to keep it up-to-date. Even AngularJS, which provides a declarative interface via directives and data binding requires a linking function to manually update DOM nodes.
React takes a different approach.
When your component is first initialized, the `render` method is called, generating a lightweight representation of your view. From that representation, a string of markup is produced and injected into the document. When your data changes, the `render` method is called again. In order to perform updates as efficiently as possible, we diff the return value from the previous call to `render` with the new one and generate a minimal set of changes to be applied to the DOM.
The data returned from `render` is neither a string nor a DOM node — it’s a lightweight description of what the DOM should look like.
We call this process **reconciliation**. Check out this jsFiddle to see an example of reconciliation in action.
Because this re-render is so fast (around 1ms for TodoMVC), the developer doesn’t need to explicitly specify data bindings. We’ve found this approach makes it easier to build apps.

HTML is just the beginning.

Because React has its own lightweight representation of the document, we can do some pretty cool things with it:

- Facebook has dynamic charts that render to `<canvas>` instead of HTML.
- Instagram is a “single page” web app built entirely with React and `Backbone.Router`. Designers regularly contribute React code with JSX.
- We’ve built internal prototypes that run React apps in a web worker and use React to drive **native iOS views** via an Objective-C bridge.
- You can run React on the server for SEO, performance, code sharing and overall flexibility.

- Events behave in a consistent, standards-compliant way in all browsers (including IE8) and automatically use event delegation.

Head on over to https://reactjs.org to check out what we have built.

The Virtual DOM

React builds a representation of the browser Document Object Model or DOM in memory called the virtual DOM. As components are updated, React checks to see if the component's HTML code in the virtual DOM matches the browser DOM. If a change is required, the browser DOM is updated. If nothing has changed, then no update is performed.

As you know, this is called the **reconciliation** process and can be broken down into the following steps:

Step 1: The virtual DOM is updated.

Step 2: The virtual DOM is compared to the previous version of the virtual DOM and checks which elements have changed.

Step 3: The changed elements are updated in the browser DOM.

Step 4: The displayed webpage updates to match the browser DOM.

As updating the browser DOM can be a slow operation, this process helps to reduce the number of updates to the browser DOM by only updating when it is necessary.

But even with this process, if a lot of elements are updated by an event, pushing the update to the browser DOM can still be expensive and cause slow performance in the web application.

The React team invested many years of research into solving this problem. The outcome of that research is what's known as the React Fiber Architecture.

The Fiber Architecture allows React to incrementally render the web page. What this means is that instead of immediately updating the browser DOM with all virtual DOM changes, React can spread the update over time. But what does "over time" mean?

Imagine a really long web page in the web browser. If the user scrolls to the bottom, the top of the web page is no longer visible. The user then clicks a button on the bottom of the web page that updates some text on the top of the web page.

But the top of the page isn't visible. Therefore, why update it immediately?

Perhaps there is text currently displayed on the bottom of the page that also updates when the button is clicked. Wouldn't that be a higher priority to update than the non-visible text?

This is the principle of the React Fiber Architecture. React can optimize when and where updates occur to the browser DOM to significantly improve application performance and responsiveness to user input. Think of it as a priority system. The highest priority changes, the elements visible to the user, are updated first. While lower priority changes, the elements not currently displayed, are updated later.

While you're unlikely to interact with the virtual DOM and Fiber Architecture yourself, it's good to know what's going on if issues occur during the development of your web application.

There are many tools available to help you investigate how React is processing your webpage. The official React Developer Tools web browser plugin developed by Meta will be one of the key tools in your developer toolbox. So, if you do have to look deeper into the code, you'll have the right toolbox available to help you. These tools will be explored later on.

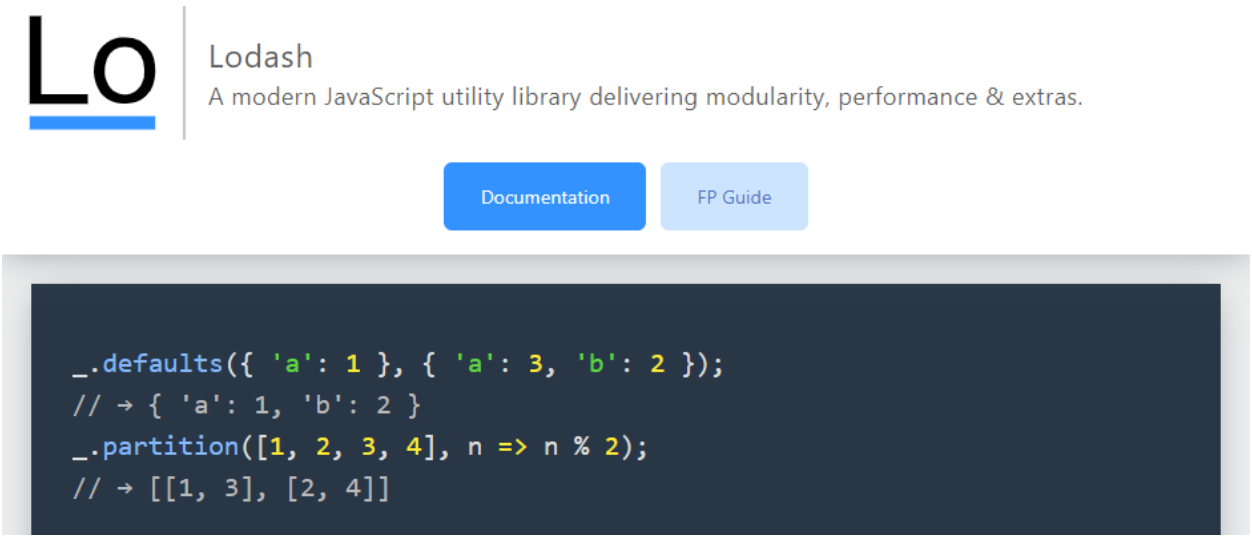
Alternatives to React

React is a library and not a framework. This means you'll often use other JavaScript libraries with it to build your application. In this reading, you will be briefly introduced to some JavaScript libraries commonly used with React.

Lodash

Official Website

As a developer, there's a lot of logic you'll commonly write across applications. For example, you might need to sort a list of items or round a number such as `3.14` to `3`. Lodash provides common logic such as these as a utility library to save you time as a developer.

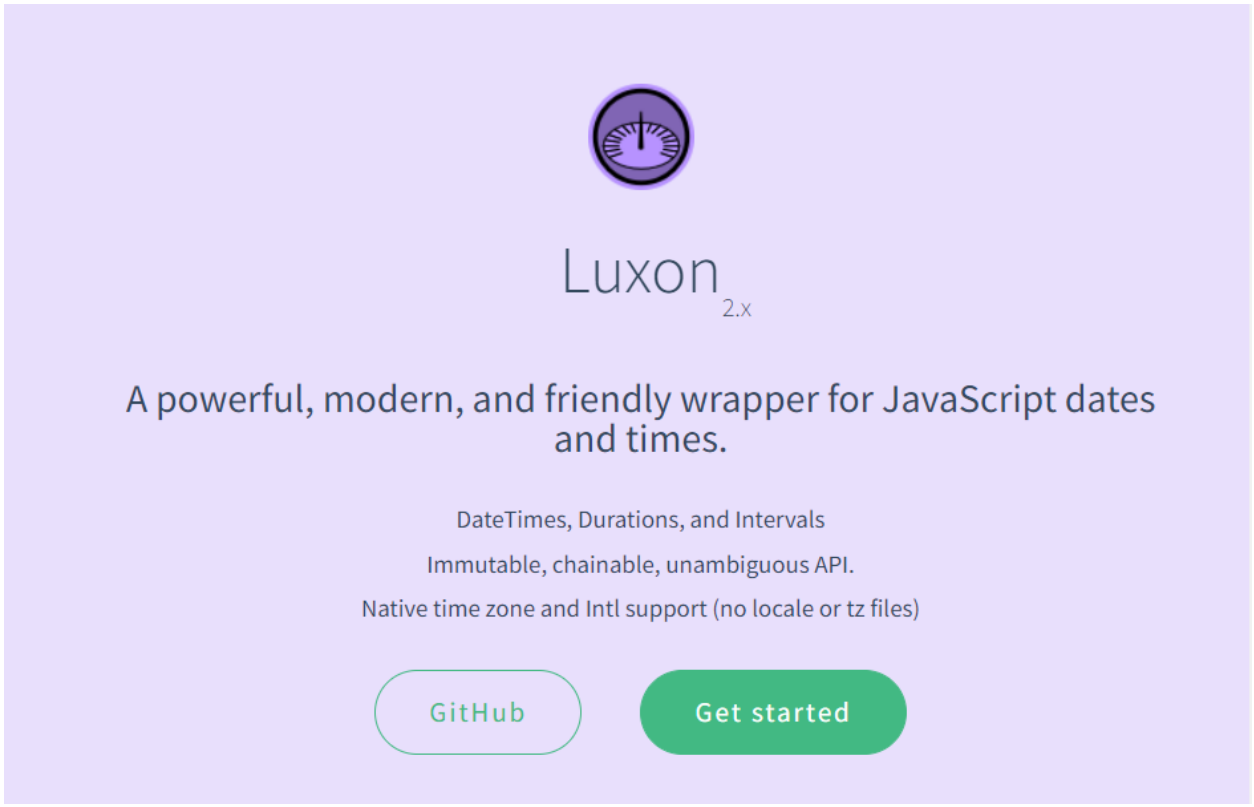


Luxon

Official Website

You'll be working with dates and times often as a developer. Think of viewing a list of orders and when they were placed, or displaying a calendar schedule for an event. Dates and times are everywhere.

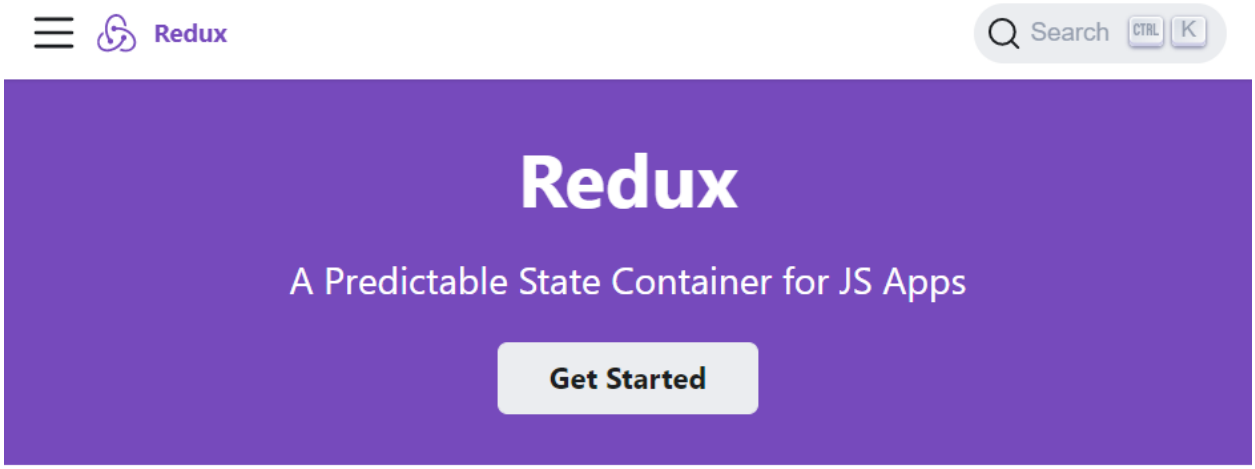
Luxon helps you work with dates and times by providing functions to manipulate and display them. For example, think of how dates are formatted in different countries. In the United States the format is **`Month Day Year`** but in Europe it is **`Day Month Year`**. This is one area where Luxon can help you display the date in the user's local format.



Redux

[Official Website]

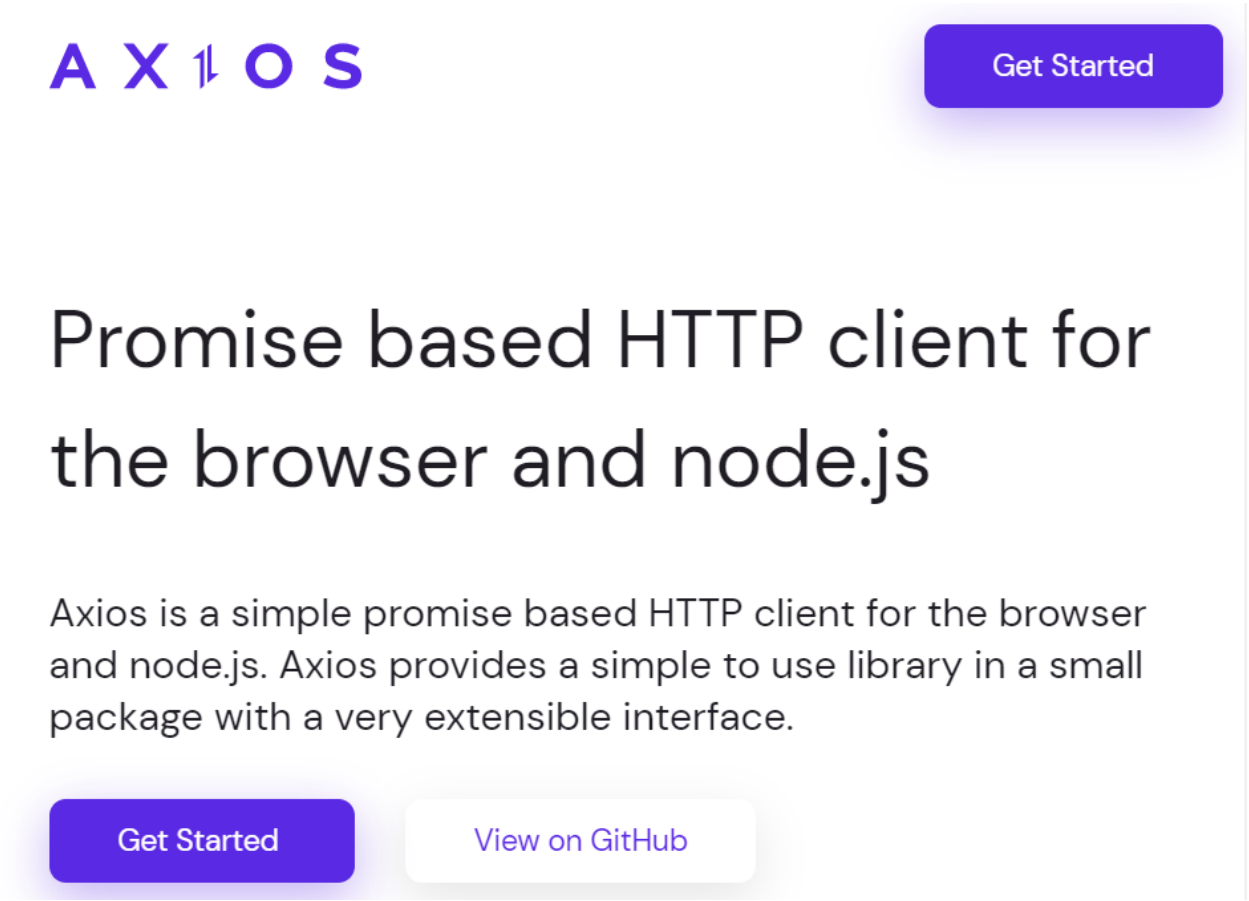
When building a web application, you'll need to keep track of its state. Think of when you shop online. The web application tracks items currently in your shopping cart. When you remove an item from the cart, the application needs to update what displays on the screen. This is where Redux comes in. It helps you manage your application state and even has advanced features such as undo and redo.



Axios

Official Website

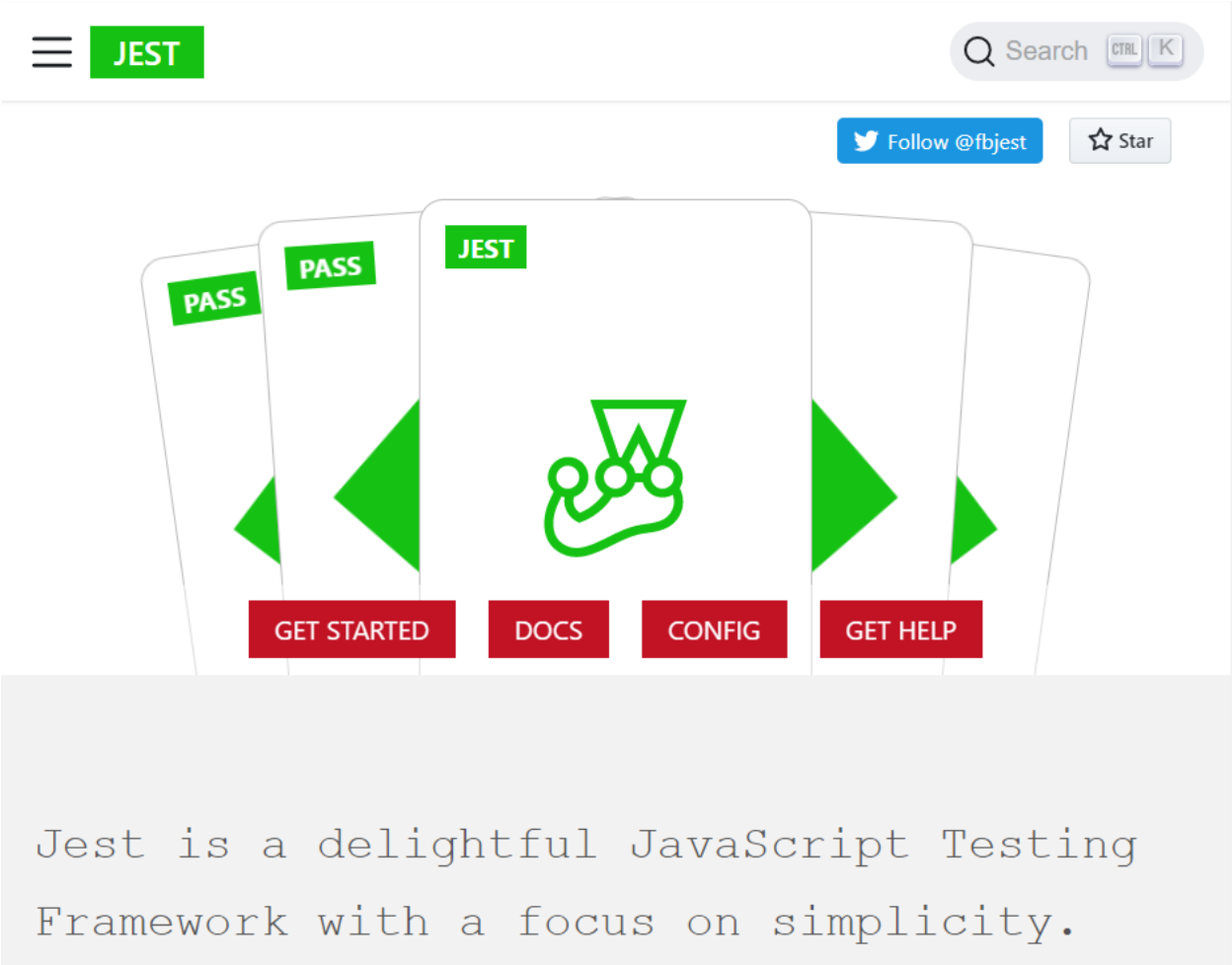
As a developer you'll be communicating with APIs over HTTP frequently. The Axios library helps to simplify sending HTTP requests and processing the response. It also provides advanced features allowing you to cancel requests and to change data received from the web server before your application uses the data.



Jest

Official Website

It is good practice to write automated tests for your code as a professional developer. The jest library helps you to do this and works with many libraries and frameworks. It also provides reporting utilities such as providing information on how much of your code is tested by your automated tests.



Conclusion

If you're curious to learn more about these libraries, their websites feature setup guides, tutorials and documentation to get started. These libraries will be covered later on.

Additional Resources

Learn more Here is a list of resources that may be helpful as you continue your learning journey.

React Official Website

https://reactjs.org/

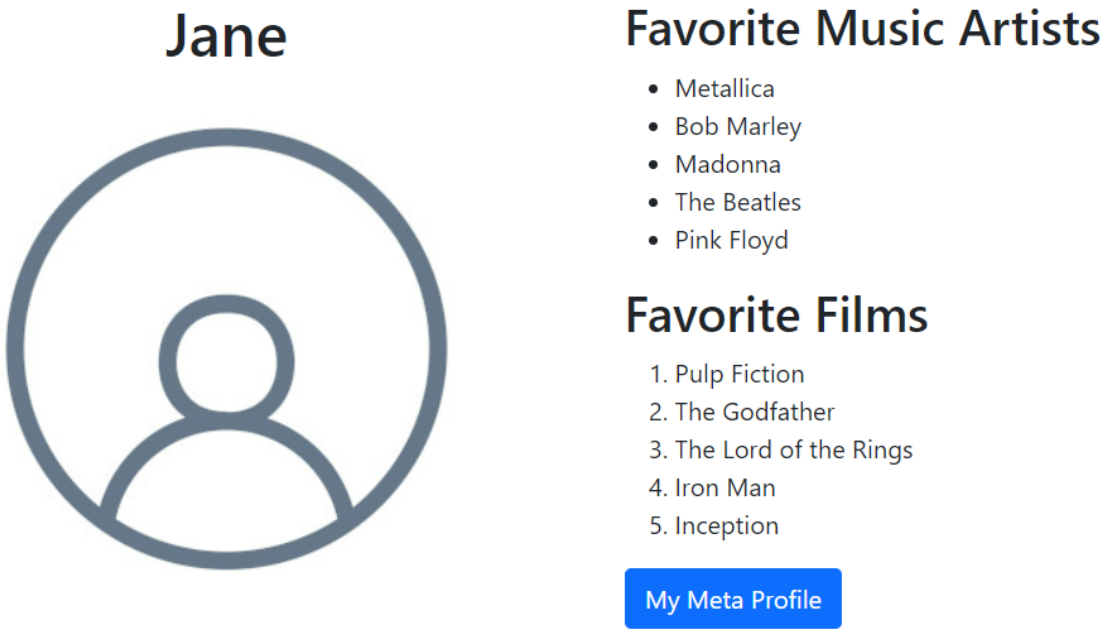
Choosing between Traditional Web Apps and Single Page Apps (Microsoft)
https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps

React Source Code (Github)
https://github.com/facebook/react

Introduction to React.js
The original video recorded at Facebook in 2013.
https://youtu.be/XxVg_s8xAms

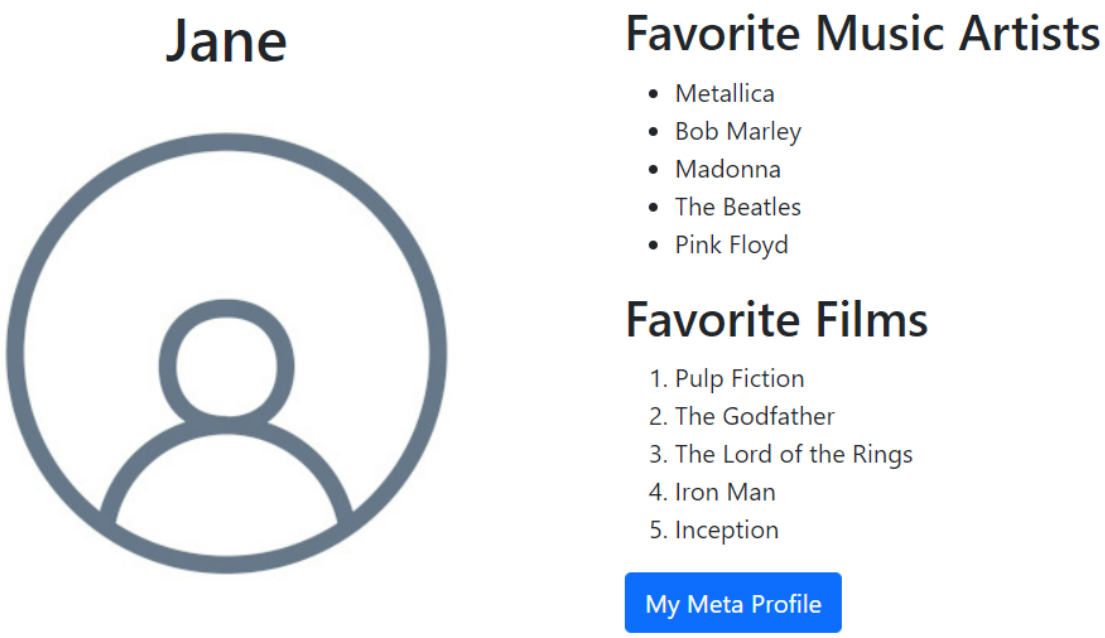
About the Ungraded Lab: Improve your Bio page with Bootstrap

In this Ungraded Lab, you will update your biographical page from Week 2 - Introduction to HTML5 and CSS to use Bootstrap.
The expected outcome is a two-column biographical page with your name and a photo in the left column and your favorite music artists and films in the right column.
The image below shows how your page should look once you finish the assessment.



Exemplar

By updating your biographical page to use Bootstrap, your new page should be similar to the image below.



Your HTML file structure and content should be similar to the snippet below. Note where the Bootstrap CSS classes were used in the different HTML elements.

```
<!DOCTYPE html>
<html>
<head>
    <title>My Bio Page</title>
    <link href="bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <div class="row">
            <div id="bio" class="col-12 col-lg-6 text-center">
                <h1>Jane</h1>
                <img src="photo.jpg" class="img-fluid" />
            </div>
            <div id="more" class="col-12 col-lg-6">
                <h2>Favorite Music Artists</h2>
                <ul>
                    <li>Metallica</li>
                    <li>Bob Marley</li>
                    <li>Madonna</li>
                    <li>The Beatles</li>
                    <li>Pink Floyd</li>
                </ul>
```

```
                <h2>Favorite Films</h2>
                <ol>
                    <li>Pulp Fiction</li>
                    <li>The Godfather</li>
                    <li>The Lord of the Rings</li>
                    <li>Iron Man</li>
                    <li>Inception</li>
                </ol>
                <a href="https://www.meta.com/user/123" class="btn
btn-primary">My Meta Profile</a>
            </div>
        </div>
    </div>
    <script src="bootstrap.bundle.min.js"></script>
</body>
</html>
```

As a developer, Bootstrap is one of the many libraries in your toolkit to help build web applications. Bootstrap has many utilities and components. We encourage you to read their documentation and experiment further with styling your webpage.

Mark as completed

Like

Dislike

Report an issue

Next steps after Introduction to Web Development

Congratulations! You've completed this course and taken another step toward improving your knowledge, skills and qualifications.

In the next course, you'll learn about JavaScript. JavaScript is the programming language that powers the modern web. You will learn the basic concepts of web development with JavaScript. You will work with functions, objects, arrays, variables, data types, the HTML DOM and much more. You will learn how to use JavaScript within the React framework and discover interactive possibilities with modern JavaScript technologies. Finally, you will learn about the practice of testing code and how to write a unit test using Jest.

Mark as completed