Meta Front - End Development Course

Course -1 :Introduction to Front-end Development (Week-1)

In this module, you will learn about the different types of web developers and the roles and responsibilities of front-end, back-end, and full-stack developers. You will take a first look at the core technologies of HTML, CSS, and Javascript and explore the concepts that underpin how the internet works.

Learning Objectives

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

Introduction to the Program

The digital space is a world of connection and opportunity. Take this moment for example, the web has made it possible for you to enroll in this program where you will learn from the personal stories of developers at meta. Isn't that exciting? By the time you have completed this professional certificate, you can become a creator of digital experiences. Connection is evolving and so are you. You might not have a background in tech at all and that's okay. Even if you have no experience, this program can get your job ready within a single year. So, how can this professional certificate prepare you for a job at an organization like meta? The front end developer professional certificate will help you build job ready skills for a front end development role while earning a credential from meta. From meta developers you will learn about how they collaborated, create and test responsive, high performance websites and applications. You'll also discuss interesting topics with other aspiring front end developers and complete a range of coding exercises to improve your skills. It's important to complete all the courses in the certificate in order as each course will build on your skills. Although we have a recommended schedule for each course, the program is entirely self paced, which means your time is your own to manage. As you make your way through the first four courses in the certificate, you'll learn how to code interactive web pages using HTML, CSS and JavaScript. You'll also learn how to use the bootstrap framework before diving into the world of React, a widely used JavaScript library that was created here at meta. Websites built using React are fast, scalable, secure and allow for rich user experience. It's no wonder it's the choice of leading companies. For your final project you will complete a beautiful professional website for your portfolio, complete with interactivity to showcase during your job search. You'll also be ready to collaborate with other developers as you will have learned to use GIT and GIT hub for version control. If you're hoping to operate independently or in a smaller team, not to worry. We've also got you covered when it comes to the basics of user experience. You'll learn how to research a user's needs, create wire frames using popular industry tools like Figma and design interactivity. In the final course, you will prepare for the coding interview. You'll practice your interview skills, refine your resume and tackle some common coding challenges that typically form part of technical job interviews. Once you complete the program, you'll get access to the meta career programs job board. A job search platform that connects you with over 200 employers who have committed to sourcing talent through meta certificate programs. Who knows where you'll end up? Whatever the future of connection looks like, you'll be part of its creation. Let's get started.

Introduction to the course

Many of the activities you do every day can be entirely performed online. You can use applications on your phone, computer, tablet, or other devices to access the web and perform tasks such as shopping, reserving hotels, and chatting with friends and work colleagues. With remote working becoming more popular. You can work, engage with your colleagues and be productive all from the comfort of your home. All of that is possible with a combination of Internet, infrastructure and technologies and skilled professionals who build the apps that you use. Just to mention that when you come across the term app, it could refer to an app on your phone or to a web app that runs on a website or by some other online method. In this course, starting with module one, you will cover an introduction to how the web works, including an exploration of web-pages, web servers, and web browsers. You will learn what each is and what their role is in bringing the Internet to you. You will also get hands-on practice using core Internet technologies like HTML, CSS, and JavaScript. You will learn how developers bring these technologies together and build functional and interactive websites and web applications. As you progress, you will explore some of the tools used by professional developers, and you will learn the fundamentals of coding using best practices and standards. For example, you will learn how to use the web browsers built-in developer tools and code using industry standard software, known as an integrated development environment or IDE. Professionals use IDEs to write code more efficiently. In Module 2, you will begin your coding journey with an introduction to HTML5 and CSS. You will learn the basics of each of these languages and how they compliment each other to layout and style elements on a web page. This includes text, images and multimedia elements like video. Additionally, to make sure that your web-page is accessible to everyone, you will learn how to code for web accessibility. In module three, you will learn how developers use frameworks and libraries. This module will focus on responsive design. You will learn how to implement the Bootstrap library so that web-page is can offer a great browsing experience no matter what type of devices used. You will also learn about user interface or UI design and how to work with common UI components and position them using the flexible Bootstrap grid. Next, you'll be introduced to react, a free and open source JavaScript library that developers use to build user interfaces based on UI components. You will then learn about the difference between static and dynamic content and the benefits of using single-page applications. Speaking of content in

Module 4, you will have an opportunity to put your new skills into practice by editing your very own biographical web-page. In summary, this course provides you with an introduction to web development. It is part of a program of courses that lead you towards a career in software development. There are many videos in your course that will gradually guide you toward that goal. Watch, pause, rewind, and re-watch the videos until you are confident in your skills. Then consolidate your knowledge by consulting the course readings and put your skills into practice during the course exercises. Along the way, you'll encounter several knowledge quizzes where you can self-check your progress. You're not alone in considering a career as a web developer. The course discussion prompts enable you to connect with your classmates, is a great way to share knowledge, discuss difficulties, and make new friends. To be successful in the course, you should try to develop a schedule for your learning regime. Ideally, commit yourself to a regular time slot and duration for your study. You've probably encountered many new technical words and terminology in this video. Don't worry if you don't fully understand all these terms right now, everything will become clearer as you progress through the course.

Front-end, back-end and full-stack developer roles

when you eat at a restaurant, there are often many cooks preparing different parts of your meal. Similarly for the websites and applications you use every day. Many roles are involved in delivering these projects to users. If you were to look up a list of high paying it jobs, web developer roles would certainly feature prominently and with good reason the digital world that we all live in would not be possible without developers creating architect ng and maintaining the technology that we use every day on our devices. But it can be confusing for aspiring developers to understand some of the terminology associated with web development. Finding the right area for you will depend on a greater understanding of web developers roles, responsibilities and technologies for example, suppose you are a visual person in that case you might want to design a stunning website that offers an excellent experience for its end users. Or if you're more analytical, you might be interested in working with the technologies that power, a high performing e commerce site. Likewise, if your interest is mobile devices, your passion may lie in creating the next big social media app. While job roles and titles may vary, web developer roles are usually split into front end, back end and full stack in this video. You will learn about each of these, their differences and the skills required to gain a job in these areas. A front end developer is someone that works on all parts of a website or web app that users will interact with. This can be anything from the style colors, buttons, menus or user interactions as they click swipe and interact with the site. The skills of a front end developer can vary, but they will always focus on three leading technologies. Html CSS and javascript for example, suppose you are a front end developer assigned the task of adding a newsletter, sign up option to the home page of a website. In this case you would use html to build the display elements such as the input area for the user to type their email address and then the button to click to send it. You can then use CSS to position, color and style these elements on the page. And finally, you can use javascript to process the activity when the user clicks the button. This could be something like checking the email address is valid and then sending that email address to the website for storage under newsletter members. While html and CSS skills are essential. The most critical skill is usually javascript. It is the powerhouse of front end technology. This is mainly because of its versatility and the fact that it is paired with powerful libraries and frameworks such as react by meta. These can be used to build rich user interface driven enterprise websites and web apps that are fast, secure and highly scalable. The salary of front end developers are competitive and can vary based on experience. Generally. Front and developer roles will be available for junior intermediate and senior level professionals. As an aspiring developer, this is a great area to get started in entry into the job market. For a junior position is possible with some fundamental demonstrations of core concepts and skills and an eye catching sample portfolio. A back end developer works on the parts of a website or web app that the end users don't see. These activities occur behind the scenes, particularly on the web server in the database or in constructing the architecture. Back end developers are responsible for creating and maintaining functionality when users request information or when the website needs to communicate to another part of the web architecture. For processing for example, performing an account, log in or completing an online purchase using a credit card. A back end developer will facilitate the interaction of the website and the content stored in database. As a result, back end development requires different languages, skills and tools. While these can vary, they generally consist of knowledge relating to back end programming language, database management systems, API's and web servers, salaries are similar to front end developers and depend on experience. Still the salary may be higher in some instances, especially for entry and senior level positions. This is because getting started with back end technologies requires more set up configuration resources and general it structural knowledge. This is in contrast to the front end where you can start learning some elements using only a web browser. The road to back end development is generally long as you must have a proficient understanding of the needs of front end technologies. This can include things like the inner workings of the Internet networks and servers. It's pretty common for aspiring developers to first start with the front end and then move to the back end. Once they have acquired specialist knowledge. A full stack developer is someone equally comfortable working with front end and back end technologies. Full stack developers have skills and knowledge in all areas of the web development project cycle. For example they have relevant expertise in the planning architecture, design, development, deployment and maintenance of the website or web. Full stack developer positions are generally at a more senior level. It can take some time to gain the knowledge, professional experience and skills to become a full stack developer. As a result roles in this area are in high demand and are some of the best paid jobs in the IT Industry.

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:

- Describe the web developer job role.
- Distinguish between front-end, back-end and full-stack developers.
- Explain how data moves through the internet.
- 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:

- Use HTML to create a simple webpage.
- 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:

- Outline the concepts that exist in most UI frameworks.
- Use the Bootstrap CSS framework to create webpages.
- Leverage Bootstrap documentation to reproduce and modify CSS components.
- Use Bootstrap themes.
- 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:

- 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:

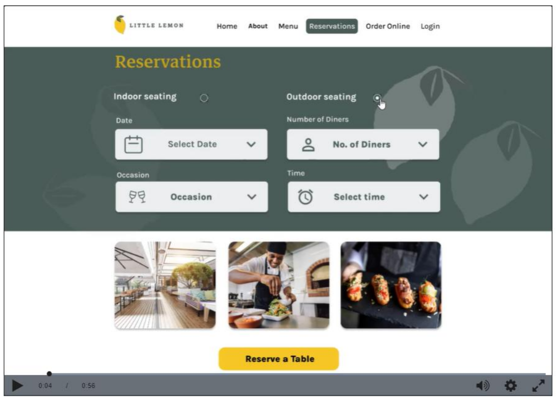
A home screen with information about the restaurant.

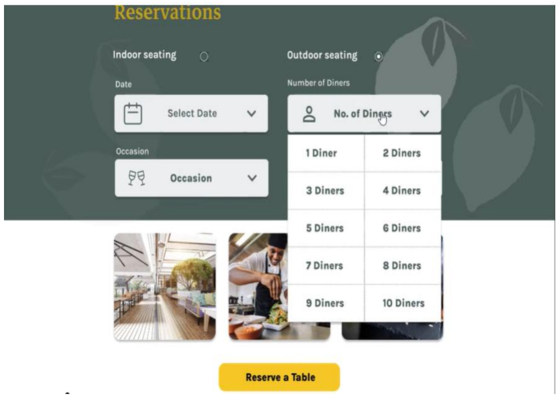
A table reservation system.

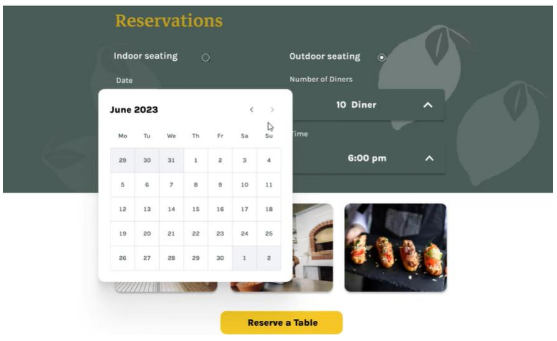
A profile screen for users to enter their personal details.

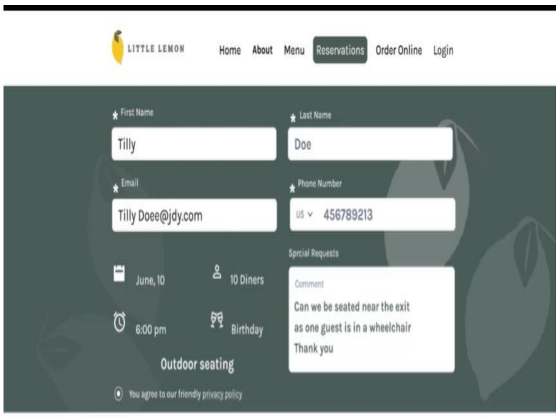
Navigation that enables users to move between parts of the web app.

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.

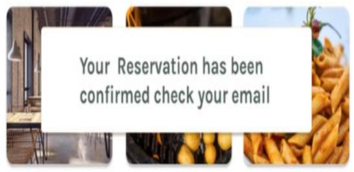


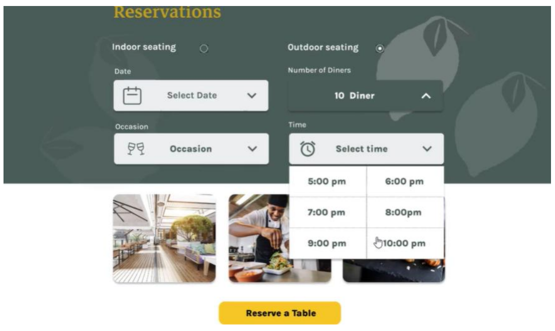






First Name: Tilly
Last Name: Doe
Email: Tilly Doee@jdy.com
Phone Number: US 456789213
June, 10
10 Diners
6:00 pm
Birthday
Outdoor seating
Special Requests
Comment
Can we be seated near the exit as one guest is in a wheelchair Thank you
You agree to our friendly privacy policy


Your Reservation has been confirmed check your email



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.

How the internet works

Hello and welcome. Did you ever find yourself searching for something online? How long does it take to get a response from your search? You search for something and you get a response back in no time. That's really something, isn't it? But did you ever think about how it all happens? In this video, I will guide you through the basics of how the Internet works. Let's start at the beginning. You open your favorite app on your device and you're instantly connected to the world. This is all made possible because two devices connect and communicate via a wired or wireless connection, forming something called a network. You can connect multiple devices to this network. But this becomes very complicated very quickly, as each device needs to connect to every other device to communicate effectively. This problem is solved by something called a network switch that connects multiple devices and allows them to communicate with each other. The network switch can connect to other network switches, and now two networks can connect. These network switches then connect to more network switches until you have something called an interconnected network. This interconnected network has another name that you might be familiar with. It's called the Internet. When we use websites or video streaming services on the Internet, these are provided by computers called servers. Our devices are called clients. This is known as the client-server model. Internet connects the entire world. Have you ever had a video call with someone on another continent? That video data travel through large undersea cables connecting the world's networks. These cables can transfer huge volumes of data per second. There are a lot more technical details that go into making the Internet possible. But this is the main idea. Hopefully, this gave you the big picture of how the Internet works.

What is a web server and how does it work?

A server is a computer that runs applications and services ranging from websites to instant messaging. It's called a server because it provides a service to another computer and its user also known as the client. Is typically stored in something called a data center with hundreds or thousands of other servers, all running different services connected to the internet. There are many different systems in data centers to ensure that servers have continuous power, continuous internet connection and are kept called 24 hours per day. Did you know that there are data centers located all around the world. Many websites use these to allow you to access your content quickly from the data center nearest to you. The data center servers are built based on the service purpose. For example, if the server is only to be used for storing images, it might have a lot of hard drive space. Whereas a server computing complex calculations might need a fast processor and a lot of memory. This is usually referred to as a server hardware. The physical components of a server. Once the server hardware is installed in the data center, a piece of code can now run. The code that runs on the hardware is commonly known as software. One way I like to remember this is to think of hardware as something you can physically touch and is difficult to change as you need to physically replace components. Software is soft or easy to change. You just replace the code running on the server. The surface is a server can run can vary but in this video you will learn about a type of service software known as a web server. A web server has many functions which includes website storage and administration, data storage, security and managing email. Another primary function is to handle something known as a web request. When you open a browser on your device and type the name of the website, it's the job of the web server to send you back to that website's content. This process is known as the request response cycle and you will learn more about it later. Web servers are also designed to respond to thousands of requests from clients per second. In this video, you learned about web servers and how they consist of hardware and software. You also learned that one of the primary roles of a web server is to respond to web requests from the client.

What are websites and webpages?

You may have heard the terms webpages and websites, but what's the difference? A web page is a document that displays images, texts, videos and other content in the web browser, a website is a collection of webpages that link together. You've likely visited many websites this week, in fact you're watching a video on one right now, let's explore an example where you visit your favorite encyclopedia website. When you arrive at the homepage, it contains many links to different articles clicking on one of those links brings you to a new webpage article and that article links to even more articles and other web pages. If all these webpages have a similar address in the web browser's address, it can be safe to assume that they belong to the same website. However, the links on the page do not all have to link to the same website, they can also link to other websites. For example, when you visit your favorite search engine and search for a phrase, the search results are a list of links to other websites. Did you know that as I'm talking to you, thousands of websites are being launched to the internet. Every day, hundreds of thousands of websites are launched, to put things into perspective the Great Pyramid of Giza has 2.3 million blocks of stone. So think about it, in one week there are as many new webpages as stones in the Great Pyramid. With current internet speeds, it would take three million years to download all the webpages on the internet and all of these billions of varied webpages rely on the same core technologies. Most people interact with websites in some form daily, but few know how webpage actually works. I mean, what's it made up of and just how does that webpage get from the web server to what you see on your screen or device? Well, let's explore that now. In its most basic form, a webpage is just a text document, you can open and edit with any text editor, but developers usually use more sophisticated tools for working with webpages. If you want to work with a webpage, you need to know about three technologies and understand how they interact, their HTML, CSS and JavaScript. HTML structures the content you see, CSS controls the colors and style and JavaScript is responsible for the user interaction. As a web developer you will work with these technologies every day and understand how they work together. I always like to imagine it like this, think of a shot, HTML is the actual building, the structure, CSS is the interior decoration and landscaping outside and JavaScript is just like the business, the services offered and the people coming in and out. Let me give you a brief overview of each of these technologies now. HTML stands for hypertext markup language, it works by using something called markup tags. These tags describe the content that is displayed in the browser window, this content can be things like headings, paragraphs, images and even multimedia elements such as audio and video, the way html describes the content is known as markup. CSS is short for cascading style sheets and adds visual enhancements like colors and layout to the web page, this is commonly known as styling. It works by enhancing the HTML elements and telling them how to display. Have you ever noticed a friend changed their color or style of their hair? Well, your friend's hair is like the HTML and the color and style is CSS. The last technology is JavaScript, which is a programming language built into the browser. JavaScript provides web developers with tools for interactivity, data processing, control and action. Have you ever tried to log to a website only to be told that the information you provided was incorrect or browse your favorite video streaming site and seen content update in real time? Well, that's JavaScript in action, JavaScript is the powerhouse of a web page. It has the ability to manipulate the content that you see on the screen as you interact with it. In fact, without JavaScript websites would be kind of boring and very limited in terms of what you can do, okay, you now know about the essential technologies of web page contains. But how exactly does this code get translated to display the content that you see on your screen? When a copy of that webpage is sent from the web server to your browser, each line of code is processed in sequential order from first to last. As each line is interpreted, the browser creates the building blocks, which is the visual representation you see on the screen. This creation process is known as page rendering, the response from the web server must be a complete web page in order to fulfill the request, to show the page in the browser. You will learn more about page rendering from the additional reading at the end of this lesson. And there you have it, you now have joined the exclusive group of developers who know how web page works. You learned about the technologies that make up a web page and how they interact with each other. You also learned the page is rendered by the web browser to display what the end user sees on the screen. In the time since you started watching this video, another few 1000 websites launched, every one of them is another example of modern web developers building and adding to the ever growing internet. Are you ready to become one of them?

What is a web browser and how does it work?

You use a web browser every day on your devices and you probably have one open right now watching this video. But do you really know how they work? In this video, you will learn what a web browser is and describe how it receives and displays content. Let's get started. A web browser, or browser for short, is a software application that you use to browse the World Wide Web. It works by sending a request to a web server and then receives a response containing the content that is to be displayed on the screen of your device. Once the browser is open on your device, there is the address bar where you input the address of the website that you want to visit. The address is commonly known as the Uniform Resource Locator or URL for short. The URL contains the protocol or the HTTP, the domain name, usually the name of the website, and the file path, or the path to the page that is displayed. When you make a request using this URL, the browser and server communicate using a protocol known as the Hypertext Transfer Protocol or HTTP. Once the web browser receives the content, it displays it on the screen of your device. This exchange of information is made possible by something known as the request response cycle. Now, let me demonstrate this using an example many of us are familiar with, searching the web. First, you open a web browser, which is a software application. Next, you type the name of your favorite search engine. The name you type contains something called a domain name. Then when you press Enter, the web browser sends a request across a network and connects to another computer on the Internet called a web server. The web server is a special type of computer that allows other computers to make requests for data. The web server responds by sending a webpage back to the browser. Once the browser receives all the response information, the search engine webpage is made visible. The web page is a coded document that is rendered by the browser and then presented visually to you, the end-user. Now that the search engine webpage is loaded in the browser, you interact with that page to search for what you want to find. For example, type restaurants near me in the search bar and press the search button. Once again, the browser prepares and sends a request to the same search engine web server. This time, the request contains the added instructions to search for the phrase restaurants near me. The search engine web server processes the request by taking the keyword and using it to find the requested data. This data is stored in something called a database, which is connected to the web server. The web server then picks up that data and sends it back to the browser. When the web browser receives the full response from the web server, it again renders a visible webpage with links to some restaurants, a map showing nearby locations and more related information such as reviews, lists, and even reservations. Success, and all made possible by computers having a conversation. You now learned how the request response cycle works. This web requests cycle is very similar for a number of other activities we do online, via chatting with friends, watching our favorite movie, or sharing files with our work colleagues. That is just a brief summary of how the

Internet works. The next time you search for something online, think of all the steps and data that gets exchanged to complete your request.

Web hosting

By now, you know that websites and files are stored on web servers located in datacenters. But what if you wanted to create your own website? Do you really need your own datacenter with specialized hardware and software? Thankfully, the answer is no. Developers can launch websites to the Internet using something known as web hosting. Web hosting is a service where you place your website and files on the hosting companies web server. You're essentially renting the space in return for stable and secure storage. You don't need to be accompanied to use a web host. Individuals can rent space too. In this video, you will learn about the different types of web hosting services that you can use for your websites and web applications. First, let me share with you some of the different hosting options available. These can include shared hosting, virtual private hosting, dedicated hosting, and Cloud hosting. Let's explore each of these hosting types in a little more detail now. The cheapest form of web hosting is known as shared hosting. You pay for a location on a web server containing many web hosting accounts with shared hosting. This means that you also share the service processing power, memory, and bandwidth with other websites that might slow your performance. This option is best for a small website with a small number of visitors. Many developers also use this as a low-cost sandbox environment to practice deploying or hosting their personal websites. Some companies offer free shared hosting, but with limitations and often have advertisements embedded in the webpages. Sites with more considerable demands use virtual private surface or VPS. A VPS is a virtual server with dedicated CPU, memory, and bandwidth resources. It will be running on a hardware server with other VPS instances but as the resources are fixed per VPS instance, your website is unlikely to be impacted by the performance of other VPS instances. A VPS instance will be more expensive than shared hosting. The next option up is to use dedicated hosting. This will be a hardware server that is dedicated to you only. All hardware, CPU, memory, and bandwidth resources are yours to use. Generally, this option is more expensive than a VPS hosting. The last type of web hosting is something you may have heard of. Cloud hosting and the Cloud has grown in popularity over the last decade and is often mentioned in various news and services you use. With Cloud hosting, your website is run in something called a Cloud environment, which spans across multiple physical and virtual servers. If a physical or virtual server fails, your website will run on a different server and stay online. The main advantage of Cloud hosting is that you can use as many resources as you need without hardware limitations. However, you pay based on resource use. For example, if you transfer a file from the Cloud to a web browser, you'll pay for the bandwidth used for that transfer at a fractional cent cost per megabyte. While this can quickly become more expensive, is allows websites and web applications to scale their costs as popularity grows. This is how many of the major web applications operate. In this video, you learned about web hosting and the different hosting options available to individuals and companies.

Soon you will build your very first website. Are you excited to get it hosted so you can share it with the world? For more information on web hosting and Services, please see the additional reading at the end of this lesson.
(Required)

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

Introduction to Internet Protocols

Email is a common communication method that we all know about. But before existed the alternative was to send mail to one another through the postal system. There is a surprising parallel between the postal system and how the computer sends and receives data across the internet every day. Let's compare how they both work. Data sent across the internet is quite an important part of our everyday lives and it wouldn't be possible without Internet Protocol addresses or IP addresses. A useful way to think of IP addresses is that they function much like addresses in a postal system that make it possible for packets of information to be delivered to you. And like with the postal system things can go wrong. But let's first go over how things work. Before we think about how they can go wrong in this video you will learn what IP addresses are and explore how computers send data across the internet. You probably learned how computers connect to each other to form networks and how these networks connect to each other to form the internet. When you send data between computers across the internet, a common way of

understanding that data is needed by the computers and networks that the data travels across. What makes that possible is the Internet Protocol. Version four and version six are currently the two most widely used standards of internet protocol. Think of the old fashioned postal system again when you send a letter to a friend you need their address otherwise they won't receive your letter. Computers work in a similar way. Every computer on a network is assigned an IP address. In protocol version four an IP address contains four octet. It's separated by periods or dots. For example 192.0.2.235. In protocol version six. An IP address contains eight groups of hexadecimal digits separated by a colon. For example 4527:0a00:1567:0200:ff00:0042:8329.
When you send data across a network, you send the data as a series of messages called IP packets. Also known as data grams at a high level IP packets contain a header and a payload or the data. Think of that old fashioned postal system again, when you send a letter. You not only include the recipient's address but also your own address in case a return location is needed. IP packets are the same. They include the destination IP address and source IP address. These addresses are in the header along with some additional information to help deliver the packet. And the payload contains the data of the packet and some of the other protocols which will cover in a moment. Earlier I mentioned that things can go wrong with the postal system. When sending multiple letters to a friend it's possible they may arrive out of order. It's possible that a package will get damaged or if you're really unlucky a letter could get lost. These issues can happen to IP packets too they can arrive out of order, become damaged or corrupted to in transit or be dropped or lost during transit. To solve these problems, the payload part of the packets contains other protocols too. You can think of them as another message inside the payload of the IP packet. The two most common protocols are the Transmission Control Protocol referred to as TCP and the User Datagram Protocol, also known as UDP. TCP can solve all three of the previously mentioned issues but at the cost of a small delay when sending the data. This protocol is used for sending the data that must arrive correctly and in order such as a text or image files. UDP solves the corrupt packet issue but packets can still arrive out of order or not arrive at all. This protocol is used for sending data that can tolerate some data loss such as voice calls or live video streaming. Both of these protocols contain payloads that contain further protocols inside of them and there you have it. The internet uses internet protocols much like an old fashioned postal system. These protocols can help to make sure that data arrives in order does not become corrupted or lost or dropped during transit. Now you're able to explain how IP addresses work and how computers send data across the internet.

Introduction to HTTP

Have you ever noticed the lock icon beside the URL in your web browser? This means that HTTPS, the secure version of HTTP is being used. HTTP is a core operational protocol of the world wide web. It is what enables your web browser to communicate with a web server that hosts a website. HTTP is the communication protocol you use whenever you browse the web. HTTP stands for Hypertext Transfer Protocol. It is a protocol used for transferring web resources such as HTML documents, images, styles, and other files. HTTP is a request response based protocol. A web browser or client sends an HTTP request to a server and the web server sends the HTTP response back to the browser. Next, let's start exploring the makeup of an HTTP request. An HTTP requests consists of a method, path, version and headers. The HTTP method describes the type of action that the client must performed. The primary or the most commonly used HTTP methods are, GET, POST, PUT, and DELETE. The GET method is used to retrieve information from the given server. The POST request is used to send data to the server. The PUT method updates whatever currently exists on the web server with something else. The DELETE method removes the resource. The path is the representation of where the resource is stored on the web server, for example if you wanted to request an image from a page in a website, then the URL in the address bar would need to contain the full path to that particular file on the web server, such as example dot com, forward slash images, forward slash image dot jpg. There are multiple versions of the HTTP protocol. I won't explore these right now, but I want you to be aware that Versions 1.1 and 2.0 are the most used. Finally, there are the headers. Headers contain additional information about the request and the client that is making the request. For certain requests methods, the request will also contain a body of content that the client is sending. Now, let's cover some details about the makeup of an HTTP response. HTTP responses follow a format similar to the request format. Following the header, the response will optionally contain a message body consisting of the response contents such as the HTML document, the image file and so forth. HTTP status codes contained within the header indicate if the HTTP request successfully completed. The code values are in the range of 100-599 and are grouped by purpose. The status message is a text representation of the status code. During your web browsing, have you ever encountered pages that display 404 error not found or 500 errors? Server is not responding. These are HTTP status codes. I want to briefly explain to you about the status codes and their grouping. There are five groups of status codes. They are grouped by the first digit of the error number. Informational is grouped 100-199. Successful responses are grouped from 200-299. Redirection message are 300-399. Client error responses range from 400-499, and server error responses are 500-599. Information responses are provisional responses sent by the server. These responses are interim before the actual response. The most common information response is 100 continue, which indicates that the web client should continue to request or ignore the response if the request is already finished. Successful responses indicate that the request was successfully processed by the web server, with the most common success response being 200 OK. You're receiving these responses every day when you receive content successfully from a website. The meaning of OK, depends on the HTTP method. If the method is GET, it means that the resource is found and is included in the body of the HTTP response. If it's POST, it means that the resource was successfully transmitted to the web server and if it's PUT, the resource was successfully transmitted to the web server. Finally, if the method is DELETE, it means the resource was deleted. Redirection responses indicate to the web client that the requested resource has been moved to a different path. The most common response codes used are 301 moved permanently and 302 found. The difference between the redirection messages 301 and 302 is that 302 indicates a temporary redirection.The resource has been temporarily moved. When web browsers receive these responses, they will automatically submit the request for the resource at the new path. Client error responses indicate that the requests contained bad syntax or content and cannot be processed by the web server. The most common codes used are 400 is used when the web browser or client submitted bad data to the web server, 401 is used to indicate that the user must log into an account before the request can be processed, 403 is used to indicate the request was valid, but that the web server is refusing to process it. This is often used to indicate that a user does not have sufficient permissions to execute an action in a web application, 404 is used to indicate that the request resource was not found on the web server. Server error responses indicate that a failure occurred on the web server while trying to process the request. The most common code used is 500 internal server error, which is a generic error status indicating that the server failed to process the request. Now, have you ever bought something online and needed to enter your credit card information? You wouldn't want someone else to get this information from the HTTP request. This is where HTTPS is involved. HTTPS is the secure version of HTTP. It is used for secure communication between two computers so that nobody else can see the information being sent and received. It does this by using something called encryption. We won't cover encryption right now. Like in HTTP, the requests and responses still behave in the same way and have the same content. The big difference is that before the content is sent, it is turned into a secret code. Only the other computer can turn the secret code back into its original content. If someone else was to look at the code, it wouldn't be understandable.You use HTTPS every day. This is the lock icon you see beside the URL in your web browser. Before I finish, I want to leave you with a brief summary of HTTP. Firstly, it is a protocol used by web clients and web servers. It works to transfer web resources such as HTML files, and is the foundation of any data exchanges on the web. Also, remember that by using HTTPS, we send the information securely. Requests are sent by the client, usually a web browser, and the server replies with responses which may be the return of an image or an HTML page. HTTPS requests have a syntax that includes method, path, versions and headers. HTTP responses follow a similar format to the request. HTTP status codes indicate whether the HTTP requests successfully completed. The status code is a three-digit number that corresponds with groups representing different types of results. Well, now you know how the HTTP protocol request and response cycle works. You know about its methods and the elements that make up an HTTP request. Good job.

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.
PATCH	The client partially updates 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/5
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).

TTP Response Body

ollowing the HTTP response headers is the HTTP response body. This is the main content of the HTTP response.

his 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>
```

Intro to HTML, CSS and Javascript

The web pages you visit every day are based on three core technologies, HTML, CSS, and JavaScript. Together, they enable you to create web pages and applications so you can offer any content you have seen online. In this video, I will demonstrate two examples you can create using these technologies. This is to help you understand how they interact with one another. Don't worry, you won't need to deal with the details of the three languages just yet. In the first example, I'm building a web page that displays a digital clock. It shows the hour, minutes, and seconds. The time is updated every second. I will work with three files, an HTML, a CSS, and a JavaScript file and you will explore the purpose of each one of them. To create the clock element, Let's first define our HTML code in a file called clock.html. The HTML code has an element that describes the content in hours, minutes, and seconds. If I only use the HTML file, the content would be shown without any style positioning or sizing, just simply as a display of six zeros representing time in the format of hours: minutes: seconds. To apply styles to the HTML element, the HTML code references a CSS file called styles.css. The web browser retrieves the styles.css file and processes it. The CSS code provides styling for the clock. It tells the browser the position, size, color, background, and font type, and size of each element on the screen. With that information, the browser updates the content and the sequence of zeros and colons now display a digital clock. Finally, we need to ensure the clock updates with the correct time. This is where JavaScript comes in. The HTML references a JavaScript file called clock.js. The browser will retrieve clock.js and process it. The JavaScript file contains code that every second updates the content of the hour, minute, and second elements. With the three files created and linked together, your clock is fully functional. The clock example demonstrates JavaScript dynamically updating content. JavaScript can also be used for interaction. The next example demonstrates how JavaScript can be used interactively. It's a web page that plays a video. Below the video, there is a button that plays or pauses the video. The button contains a play icon. My HTML page describes the content, which is the video element and the Play button element. The HTML code references a CSS file that the browser retrieves. It applies the styling to the video and button. The code in the HTML file also references a JavaScript file. The browser will retrieve the videoplayer.js file and process it. In our example, the code in the JavaScript file performs four actions. Firstly, it registers a listener on the button that will execute some code when the button is clicked. The second function is that when the code runs, it checks the current state of the video. The result of that check is, if the video is currently stopped, it begins playing the video and changes the buttons icon to a stop icon. Or if the video is currently playing, it stops playing the video and changes the buttons icon to a play icon. Using the three files you create a fully interactive video player. A summary of your functioning video player application is when the user first sees a page, the video will be stopped by default. When they click the Play button, the button will change to a stop icon and the video will begin playing. When they click the button again, it will change back to a play button and the video will stop playing. I hope those examples give you an idea of how the three core files link and work together. In both the clock and video examples the HTML file references the CSS and the JavaScript files. The CSS file is called on to format and style your application, and the code in the JavaScript file implements the core functions of the app and user interactivity.

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.

Webpages, Websites and Web Apps

When you do something online, you are likely to encounter web pages, websites, and web applications. But how do they differ? As a developer, how do you decide which one to create? Let's explore the main features of each one. A typical web page is one single page that consists of HTML, CSS, and JavaScript. It displays images, text, videos, and other content in the web browser. If a web page is one single page then a website is a collection of web pages that link together under one domain name. You've likely visited many websites this week. You know, when you visit your favorite encyclopedia website and the homepage has a lot of links to different articles, clicking on one of those links brings you to an article on a new web page, and that article links to more articles, and other web pages. Well, since all of these web pages exist under the same domain name, they are a website. The technical term for a link, you click on a hyperlink. This is because they link to hypertext content. Remember that HTML is Hypertext Markup Language. However, links themselves, don't have to link to the same website. They can also link to other websites. For example when you go to your favorite search engine and search for a phrase, the search results are a list of links to other websites. You'll explore more about hyperlinks later. By now, you should probably know the difference between a web page and the websites, but what about a web application? This is where the definitions gets a little more blurred.

The terms website and web application are often used interchangeably. The key difference between a website, and web application is the level of interactivity, and dynamic content. The easy way to remember this is that a website is more informative and a web application is more interactive. Think of ordering food online. Let's say you would like to order some food and you go to your favorite site, the browser then displays a web page, you select some food from the menu and submit your order. Compare this to for example, a company website that displays information about themselves and the services they provide. In the food ordering example, the content displayed is specific to your user account and location, the web application displays content based on the user's input and interaction. Whereas with the company website, the user simply views the content and this content is the same for everyone who visits the website. You should now be able to distinguish between web pages, websites, and web applications. You know that web pages at a particular domain make up a website, and that the key difference between websites, and web applications is the level of interactivity, and dynamic content.

Developer tools

If a car breaks down, you can open the hood to examine the engine and to find out what's gone wrong. As a developer, if your front end isn't working as expected, you can also open the hood to check what's going wrong. In fact, it's not just your own code that you can investigate. How about viewing other people's code. By the end of this video you'll be able to access and make use of common web browser developer tools.

Most web browsers come equipped with a set of developer tools that allow developers to inspect their HTML, CSS and Javascript code. Also, to trace http request to the web server, investigate performance issues and review web page security. Let's find out more by exploring the homepage of the Little Lemon Cafe. To begin, I've just opened the Little Lemon Cafes web page on my chrome browser. On the homepage, I can view their menu. But I want to explore the HTML structure of this menu. To do that, I need to open the developer tools. To open the developer tools in chrome, press the F12 key on your keyboard for PC or command option J on Mac. Alternatively, you can right click on the web page and select inspect. There are various tabs on the top row of developer tools that provide different functionality. In this video, I'll give you a high level explanation of some of the most used tools. 1st, let's open the console tab. This tab outputs, javascript logs and errors from your web application. The sources tab shows all content resolved for the current page. It includes HTML, CSS, Javascript, images and videos. With the network tab, you can inspect the timeline and details of http requests and responses for the web page. The performance tab shows what the web browser is doing over time. It is useful if your web application is running slow because you can pinpoint the functions that are taking the most time. The memory tab displays the parts of your code that are consuming the most resources. Finally, let's check the most used tab the elements tab. You can use this tab to inspect the documents, HTML elements and their properties. For example, when I hover over an element in the elements tab, it highlights that element in the browser pane. On the right side of the panel, there are tabs for inspecting the details of the elements further. This panel shows us what CSS

is applied to an element and the result of the element displayed in the browser. We will explore these details in a later lesson. For now, you just need to know that if you click on an HTML element, then the information for that element will appear in the tab. Now, I'm going to demonstrate a fun trick. If you double click the HTML, you can edit it in the web browser. For example, if I select the Our Menu body element, then I can change the first item in the menu from chicken Burger, to Turkey Burger.
This doesn't change the content on the web server. It only updates it for me while the web pages open. If I open the web page again, it will reset.
The Web browser developer tools are a valuable part of your development toolkit to help you investigate and diagnose problems and you now know how to access and make use of them. Great work.

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.

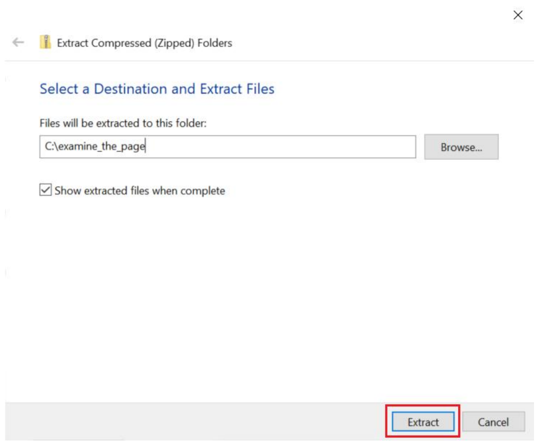
Objectives

- 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


Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:
C:\examine_the_page

☑ Show extracted files when complete

Step 2: Unzip the file.

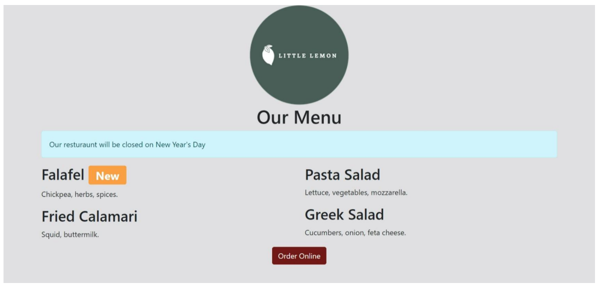
On Windows, open your Downloads folder, right-click the file examine_the_page.zip and select Extract All.

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

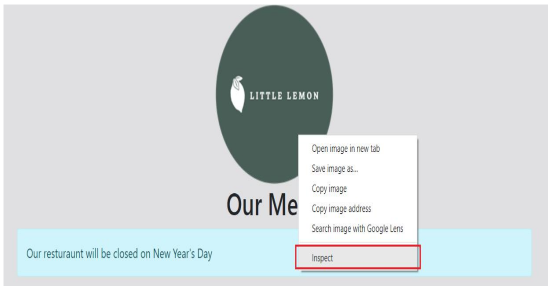
Once unzipped, there will be a

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:

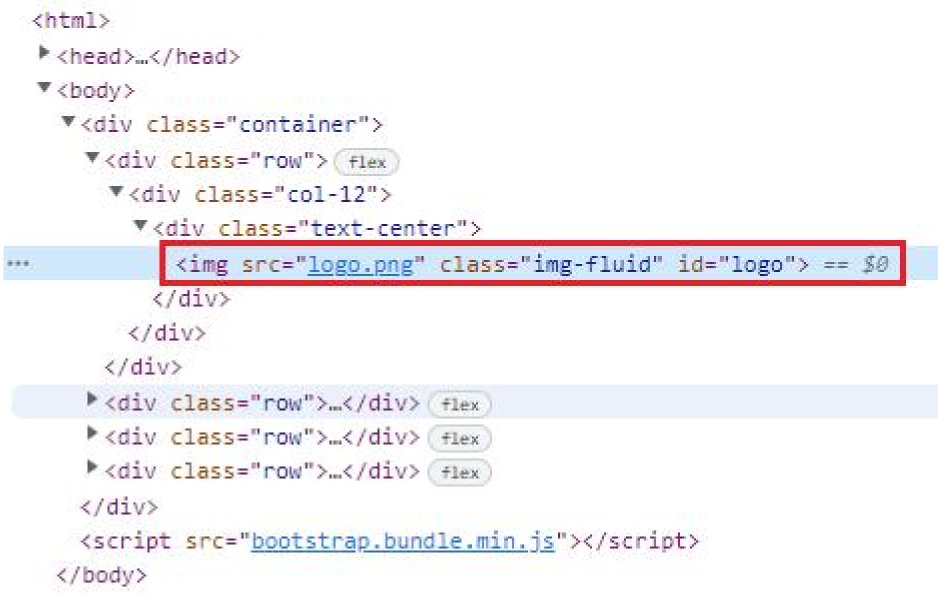


Step 4: Right-click the Little Lemon logo and select Inspect (or Inspect Element)

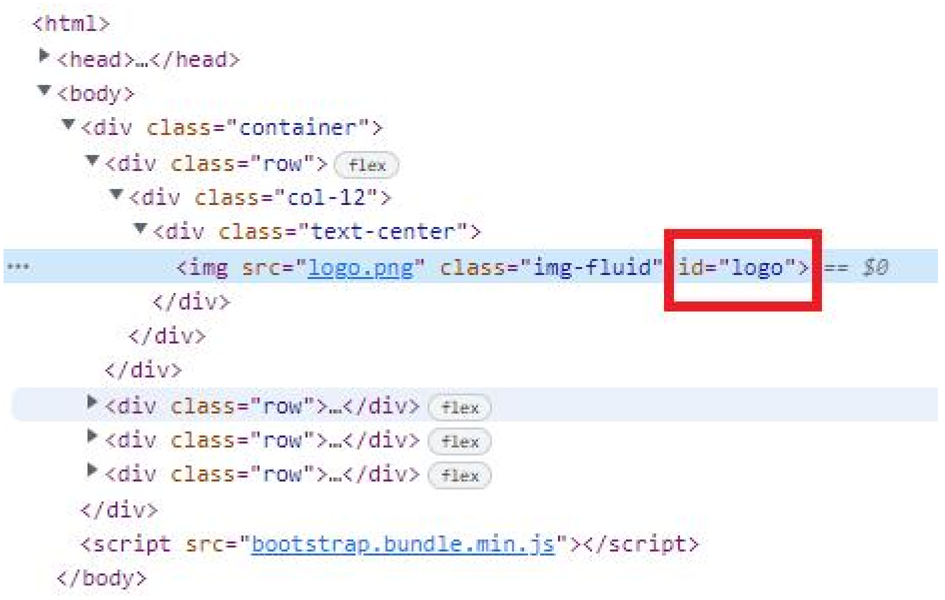


Step 5: Inspect the line in the HTML for the logo in the developer tools panel. The line begins with <img.

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



```
<html>
 ▶<head>…</head>
 ▼<body>
   ▼<div class="container">
     ▼<div class="row"> flex
       ▼<div class="col-12">
         ▼<div class="text-center">
...          <img src="logo.png" class="img-fluid" id="logo"> == $0
           </div>
         </div>
       </div>
     ▶<div class="row">…</div> flex
     ▶<div class="row">…</div> flex
     ▶<div class="row">…</div> flex
   </div>
   <script src="bootstrap.bundle.min.js"></script>
 </body>
```



```
<html>
 ▶<head>…</head>
 ▼<body>
   ▼<div class="container">
     ▼<div class="row"> flex
       ▼<div class="col-12">
         ▼<div class="text-center">
...          <img src="logo.png" class="img-fluid" id="logo"> == $0
           </div>
         </div>
       </div>
     ▶<div class="row">…</div> flex
     ▶<div class="row">…</div> flex
     ▶<div class="row">…</div> flex
     </div>
     <script src="bootstrap.bundle.min.js"></script>
   </body>
```

Frameworks and libraries

You are developing solutions but you need to save some time and build faster. What if some of your build problems have already been solved for you? Well it's true someone has already figured out many key development processes and they're contained in frameworks and libraries that are used every day in software development, so what exactly are frameworks and libraries? Let's say you are not a developer, but instead you work as a carpenter. You make chairs and sell them online. As a carpenter you don't design a new hammer for every chair you make. It makes much more sense to use an existing hammer, but of course you are a developer. As such it's important for you to know that to speed up development, developers use already developed frameworks and libraries in their application development. These might be open source, meaning that the source code is freely-available for anyone to modify and build from. There are thousands of open source libraries and frameworks available or there might be proprietary, ones that are licensed or developed internally. Many developers use the terms framework and library interchangeably, so what's the difference between them? Libraries are reusable pieces of code that can be used by your application. They are purpose-built to provide a specific functionality. To give a more technical example, you're building a small e-commerce website. When a user wants to register they need to provide their email address. Email addresses while easy to read can be complicated to validate. In fact email addresses are defined across several technical specifications. That's a lot of reading just to validate an email. Even if you do read through all the specification, there's no point in spending hours or even days implementing their standards because you have access to so many readily-available libraries to validate email addresses. It is for specific functionality like validating an email address that libraries are useful. A developers simply uses the library to access the functionality they require, as a result they can have more time to continue focusing on the development of their application. Frameworks on the other hand provide a structure for developers to build with. Consider this in the context of our carpenter analogy. As a carpenter you create a lot of different chairs, therefore there would be a blueprint for each chair to speed up building them. You can decide the type of wood to use, but the dimensions and style of the chair are always the same. Frameworks act as a structure where the developer provides their own code that the framework interacts with. For example, there are many frameworks for developing web applications. These frameworks handle functionality that is common to all web applications such as receiving HTTP requests and sending HTTP responses. The developer then adds their own code that implements the functionality of the web application. For instance with the e-commerce website example, a framework would handle receiving HTTP requests. The developer would implement code that processes the request and returns a response from which the framework would send a response over HTTP. Now let's compare how the frameworks relate to libraries. Most frameworks use many libraries. The libraries that the framework uses can be used for your application. If you wish, your application can also use other libraries. You also need to consider when to use a framework and when to use a library. Frameworks are considered opinionated and libraries are considered unopinionated. This is defined as the degree of freedom available to the developer to choose how to code a feature. The opinionatedness will vary between frameworks, but by definition they will always be more opinionated than a library. The benefit of this is that they can replace libraries as needed. For example when new technologies become available frameworks to find the libraries flow and control of an application, whereas with the libraries those are left to the developer to decide. As with everything there are advantages and disadvantages to both. Frameworks are a great way to reduce development time and to enforce a structure on how code is written. They have many best practices already in place and contain most of what is needed to develop an application, however, sometimes you may find that the way you need to code something doesn't fit into the structure of the framework. Other times you may find that some of the libraries the framework uses may conflict with a library that you are required to use and cause compatibility issues. If an application is built without a framework, the developer will need to decide on the set of libraries they wish to use to achieve the functionality they must deliver. They will also need to take care that the selected libraries can work together. The upside to this is that they can replace libraries as needed. For example, if a new better library is released, the developer can replace the usage of the old library. This is much easier than replacing a framework. Frameworks and libraries give you the opportunity to reuse existing web app functions. This can result in faster development, fewer errors, and more time for you to spend on the essential features of your application. Instead of reinventing the wheel, you can use frameworks and libraries that are designed specifically to help your web app development processes.

APIs and services

Every day you access information on your phone, like reading the news, purchasing goods and services or communicating with friends over social media. But how is all this information transferred behind the scenes? Your favorite websites and apps. Probably use API's and as a web developer, you'll discover that API's developer friendly, easily accessible and a very valuable and useful development tool. A PI is the acronym for application programming interface. An API is a set of functions and procedures for creating applications that access the features or data of an operating system, application or other service. If this still sounds a bit vague, just remember that the term API, is intentionally open too many applications and use cases. As a web developer, a lot of the day to day job involves working with API's. Some common API's that web developers work with include Browser, API REST API and Sensor-Based API. Over the next few minutes, you'll explore each of these API types and review a few specific examples. To begin with, here's a brief outline of how a piecewise functions. In Software development, API's are often the bridge between different components or systems. This earns them names like gateway or middleware. The term is used widely to represent many different tools and systems. Let's consider some examples of different API use cases. One common type of API, is Browser or Web APIs, which are built into the browser itself. They extend the functionality of the browser by adding new services and are designed to simplify complex functions and provide easy syntax for building advanced features. A good example, is the DOM API. The DOM API turns the html document into a tree of nodes that are represented as JavaScript objects. Another example, is the geolocation API that returns coordinates of where the browser is located. There are also other API's available for fetching data known as Fetch API drawing, graphics or Canvas API keeping history or history API. And client side storage also known as Web Storage API. Another critical type of API for web development is the RESTful or REST API. This kind of API provides data for popular web and mobile apps. These are also called web servers. Let's explore REST in a bit more detail. REST or representational state transfer, is a set of principles that help build highly efficient API's. One of the main responsibilities of these kinds of API's is sending and receiving data to and from a centralized database. We can query our own REST API or third party ones. One last type of API, that you might encounter as a web developer is a Sensor-Based API. This is what the internet of things also known as IOT is based on. These are actual physical senses that are interconnected with each other. The sensors can communicate through API and track and respond to physical data. Some examples are Philips hue, smart lights and node bots. That's a lot of API to think about. Fortunately, for web developers the most common data API is a RESTtful API which as you've learned is a web server that provides responses to requests. These API web servers are designed to provide the data backbone for a web client like a web page or mobile app. This means that these API's must be able to accomplish things like getting data or get, creating data. Also referred to as post updating data or put and deleting data or delete. API issues, REST principles and good design practices to create discoverable interfaces. This helps us get the exact response expected. But exactly how do they work? Here's a closer description of their activity. These API's use endpoints to specify how different resources can be accessed. The endpoint is built into the URL when accessing the API. Once the endpoint is hit, the API performs whatever service side processing is needed to build the response. Two common forms of response are, full web pages and data form based on JavaScript called Jason. In this video, you explored some API's and as a web developer, you will frequently work with many different types of API's. You will often use API's to extend the abilities of systems or to act as a bridge between different components.

What is a an IDE?

Think of a group of construction workers, every worker has a toolbox that helps them get their job done. As a developer, you'll also use many tools. One of the main tools in your toolbox is the integrated development environment or IDE. By the end of this video you'll be able to identify an IDE and explain the benefits of using an IDE during development. An integrated development environment or IDE is software for building applications. An IDE is just like a text editor except instead of writing documents you're writing code. There are many IDEs available, some are specific to one programming language while others support many languages in one IDE. Let's explore some common IDE features. Here I am working within an IDE. First let's cover syntax highlighting. To improve readability for developers, IEDs have syntax highlighting. What this means, is that special keywords of the programming language are highlighted in different colors so that the developer can quickly differentiate these keywords from other texts. For example, if you're writing JavaScript code without syntax highlighting, it could be harder to identify keywords from other texts. With syntax highlighting, that gets much easier because the JavaScript keywords and variables are colored differently. Now, let's explore error highlighting. Just like checking spelling in a text document, IDEs can highlight mistakes you make in your programming code. For example, if I delete the equal symbol where it's needed, my IDE will highlight the error. Another feature of IDEs is also complete. When you're typing a message on your phone, it suggests words as you type. An IDE's autocomplete is a similar feature. Since programming languages have special keywords, IDEs can offer suggestions to autocomplete words as you start typing them. Additionally, another feature called IntelliSense can make IEDs very smart and even able to understand your code. They can detect variables and functions and offer them as suggestions during autocomplete. For example, if I have a JavaScript function named myFunction defined at the top of the JavaScript file, then as soon as I start typing the letter m my IDE suggest this function as an autocompletion. Then there is refactoring. Since IEDs understand your code, they can help you if you need to change it. To demonstrate how refactoring works, let's continue with the myFunction function that I defined a moment ago. In the code, the function is then called multiple times. It can also be called in the code of other files too. But what if you need to rename this function? You would need to rename it in every file that uses the function ensuring that you update those files to use the new name. This process is known as refactoring, changing the structure of the code without changing the functionality. Doing this manually is very time consuming and prone to error. If you mistyped the new function name in one place, the application will break. Since the IDE understand your code, it can assist with refactoring and automatically update the function name across all files. That saves a lot of time. Let's rename our function now. I just right click on the function and select rename symbol. Then I change it from myFunction to ourFunction. The IED then updates all references of that function name. IDEs come with a lot of other features to help investigate bugs and collaborate with other developers. Many even allow you to extend their functionality using plugins and extensions, but that's beyond the scope of this lesson.

We have explored some features of IDEs in this video. You now know how IDEs operate as part of the developers toolbox to write code more effectively. Well done.

Additional Resources

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

HTTP Overview (Mozilla)

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Introduction to Networking by Dr.Charles R Severance

https://www.amazon.com/Introduction-Networking-How-Internet-Works/dp/1511654945/

Chrome Developer Tools Overview (Google)

https://developer.chrome.com/docs/devtools/overview/

Firefox Developer Tools User Docs (Mozilla)

https://firefox-source-docs.mozilla.org/devtools-user/index.html

Getting Started with Visual Studio Code (Microsoft)

https://code.visualstudio.com/docs