Front-end development Notes

# Week 1

**Web Development Trends**

Web development is a constantly evolving field. Developers and businesses must stay up to date with the latest trends and technologies. Here are some key developments:

**AR and Web Development**

Augmented Reality (AR) is a technology that overlays digital information, such as images, videos, 3D models, or other interactive content, into the real-world environment. AR blends the virtual and physical worlds, enhancing a user's perception of reality by adding computer-generated elements to their view of the real world.

In web development, developers integrate AR to bring augmented reality experiences directly to web browsers. They accomplish this through WebAR technologies, enabling users to access AR content without requiring dedicated AR apps or installations.

For instance, a furniture retail website might incorporate AR functionality, allowing users to visualize how a piece of furniture would appear in their own home through their smartphone camera. Similarly, educational websites might utilize AR to offer interactive and engaging lessons by overlaying 3D models onto textbooks or other physical objects.

Web-based AR provides accessibility and convenience, reaching a broader audience without mandating that users download additional apps. It heightens user engagement, interactivity, and the overall user experience, rendering it an exciting and innovative frontier in web development.

**VR and Web Development**

Virtual Reality (VR) is a technology that immerses users in a computer-generated, three-dimensional, interactive environment, often using specialized equipment like VR headsets or goggles. Web developers incorporate VR into web applications and websites to provide immersive and interactive experiences. Here are some ways in which web developers are using VR:

* VR Websites: Web developers are creating entire websites that are VR experiences. Users can explore these websites using VR headsets, immersing themselves in a 3D environment. These VR websites are helpful in showcasing products, real estate tours, or artistic projects.
* 360-Degree Videos and Photos: Web developers use VR to present 360-degree videos and photos. These immersive visuals are used for virtual tours of museums, tourist attractions, and real estate properties.
* Virtual Reality Games: Web-based VR games are gaining popularity. Web developers use technologies like WebVR and WebXR to build and distribute VR games.
* Training and Simulations: VR helps in training simulations in various industries. Web developers create web-based VR training modules that allow users to practice skills in a safe and controlled virtual environment.

**VUIs Integration in Web Development:**

Voice interfaces, or voice user interfaces (VUIs), allow users to interact with digital devices and applications using spoken language commands. Developers implement voice interfaces in web development using voice recognition and natural language processing (NLP) technologies. These technologies convert spoken language into text, analyze user intent, and trigger appropriate responses or actions within the web application.

Web developers incorporate VR into web applications and websites to provide immersive and interactive experiences. Some key applications of VR in web development include:

* VR Websites: Developers create entire websites as VR experiences, allowing users to explore 3D environments with VR headsets. These are useful for product showcases, real estate tours, and artistic projects.
* 360-Degree Videos and Photos: VR presents 360-degree visuals, enabling virtual tours of museums, tourist attractions, and real estate properties.
* Interactive Storytelling: VR enables the creation of interactive narratives where users become part of the story. It's useful in education, marketing, and entertainment.
* Training and Simulations: VR is employed for training simulations across industries. Web developers build web-based VR training modules for users to practice skills in a controlled virtual environment.

**Responsive Design**

Responsive design in web development is an approach to designing and building websites that ensure they look and function well on various devices and screen sizes. Responsive design aims to create a consistent and user-friendly website experience on a desktop computer, laptop, tablet, smartphone, or any other device with internet capabilities.

The core principles of responsive design encompass:

* Flexible Layouts: Responsive websites use fluid grid layouts that seamlessly adapt to various screen sizes, often favoring relative units like percentages, Ems, and rems for layout elements instead of rigid pixel measurements.
* Media Queries: Utilizing CSS media queries, developers can apply distinct styles to a webpage based on device characteristics like screen width, height, and orientation (portrait or landscape).
* Fluid Images and Videos: Developers design images and videos to scale and maintain proportionality, preventing cutoffs or extremes in size when viewed on diverse devices.
* Mobile-First Approach: Many responsive designs start with a mobile-first approach, prioritizing development for smaller screens before enhancing for larger devices.

**Progressive Web Apps**

Progressive Web Apps (PWAs) are a modern approach to web development that combines the best of web and mobile app technologies. They are web applications that offer a native app-like experience to users right in their web browsers without requiring installation from an app store. Key characteristics of PWAs include:

* Responsiveness: PWAs adapt to various screen sizes and orientations, ensuring a consistent device experience.
* Offline Capabilities: Service workers enable PWAs to cache content, allowing them to function even when the device is offline or has a poor internet connection.
* App-Like Interactions: PWAs offer smooth and responsive interactions, making users feel like they are using a native app.
* Security: PWAs are served over HTTPS to ensure data security and privacy.

**Conclusion**

In summary, the ever-evolving field of web development offers exciting prospects. Augmented Reality (AR) and Virtual Reality (VR) enhance user experiences, Voice User Interfaces (VUIs) enrich interactions, and responsive design ensures adaptability across devices. Progressive Web Apps (PWAs) combine the best of web and mobile apps for seamless experiences. Staying updated on these trends is crucial for creating compelling web experiences in this dynamic landscape.

# Module 1 Summary: Introduction to Web and Front-End Development

Congratulations! You have completed this lesson. At this point in the course, you know:

* Front-end development is creating a website or app’s user interface and experience.
* Front-end developers focus on visual appeal, usability, accessibility, brand consistency, and website performance optimization.
* Full stake development includes proficiency in front-end, back-end, server infrastructure, deployment, configuration, and version control.
* The web development process involves requirements analysis, design, development, testing, deployment, and maintenance.
* Web developers perform various f tasks, including prioritizing daily work, collaborating with clients, conducting research, coding, and programming, utilizing version control systems, troubleshooting, and conducting extensive testing to ensure optimal website performance.
* Effective communication with team members is crucial for smooth collaboration and an efficient workflow.
* Web development is evolving,  focusing on AR and VR integration and Responsive Design.
* UI/UX design process involves understanding user needs and goals, creating wireframes and prototypes, designing visual elements and branding, and conducting interaction design and usability testing.

**Module 1 Glossary: Introduction to Web and Front-End Development**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Accessibility | Design digital products usable by people with disabilities, ensuring equal access and inclusion. |
|  |  |
| Artificial Intelligence (AI) | Integration of intelligent behaviors into web development to personalize user experiences. |
|  |  |
| Augmented Reality (AR) | Technology that overlays digital information onto the real world, often through mobile devices. |
|  |  |
| Caching | Store static assets locally to speed up website loading by retrieving files from local storage instead of the server. |
|  |  |
| Cross-Browser Compatibility | Ensuring a website works consistently across different web browsers. |
|  |  |
| DOM (Document Object Model) | A programming interface for web documents that represents a webpage's structure as a hierarchical tree. |
|  |  |
| Event Handlers | Code that responds to user interactions, like clicking a button or submitting a form. |
|  |  |
| Full-Stack Developer | A developer with skills in both front-end and back-end development. |
|  |  |
| Lighthouse | A performance testing tool that generates reports to evaluate website performance and suggest improvements. |
|  |  |
| Minification | The process of removing unnecessary characters from code to reduce file size and improve loading times. |
|  |  |
| Mobile-First Approach | Designing websites starting from a mobile perspective and scaling up for larger screens. |
|  |  |
| Mobile-First Design | Designing for mobile devices first and then scaling up for larger screens. |
|  |  |
| Progressive Web Apps (PWAs) | Websites that offer an app-like experience with offline access and enhanced performance. |
|  |  |
| Prototyping | Develop interactive models of the final product to test usability and functionality. |
|  |  |
| Responsive Design | Design websites to adapt to different screen sizes and devices. |
|  |  |
| Server-Side Logic | The code that runs on the server manages data and performs operations. |
|  |  |
| UI (User Interface) | The visual elements and layout of a digital product that users interact with. |
|  |  |
|  |  |
| Usability Testing | Observing users as they interact with a product to identify issues and gather feedback. |
|  |  |
| User Personas | Fictional representations of target audience characteristics, motivations, and goals. |
|  |  |
| UX (User Experience) | A user's overall experience while interacting with a digital product focuses on ease of use, satisfaction, and efficiency. |
|  |  |
| Version Control System (VCS) | Software that tracks code changes and enables collaboration among developers. |
|  |  |
| Virtual Reality (VR) | Immersive technology creates a computer-generated environment, typically experienced through headsets. |
|  |  |
| Wireframing | Create simple visual representations of a website's layout and structure. |

# Intro 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.

# Intro to HTTP

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.

**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:

1

2

3

4

5

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.

1

2

3

4

5

6

7

8

POST /users HTTP/1.1

Host: example.com

{

 "key1":"value1",

 "key2":"value2",

 "array1":["value3","value4"]

}

1

2

3

4

5

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:

1

2

3

4

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.

1

2

3

4

5

6

7

8

9

10

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, JS

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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

Play video starting at ::26 and follow transcript0:26

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.

# API and Servers

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.

# 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.

# What is an IDE?

hink 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.

​