Course: Web Application Programming

Week 1: Introduction to Web Applications and Front-End Development

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Course Overview
2. Introduction to Web Applications and Front-End Development (Introduction to web technologies and architecture and
3. HTML and CSS basics for structuring and styling web content)

Course Overview

1. Course Description
2. Course Objectives
3. Course Learning outcomes
4. A glance Course Content
5. Course requirements
6. Course length
7. Text Books and References

Course Overview-Course Description

This course introduces students to the fundamental concepts and techniques of web application programming. Students will learn how to design, develop, and deploy interactive and dynamic web applications using modern web technologies.

The course covers both client-side and server-side programming, along with an emphasis on best practices and industry standards.

Course Overview-Course Objectives

This Course is intended to enable learners to:

1. Develop deep Understanding of the architecture of web applications and the role of client-side and server-side components.
2. Develop interactive user interfaces using HTML, CSS, and JavaScript.
3. Implement client-side scripting to enhance user experience and interactivity.
4. Create dynamic web pages using server-side scripting languages.
5. Design and manage databases to support web applications.
6. Implement security measures and best practices for web application development.
7. Deploy and maintain web applications on hosting platforms.

Course Overview-Course Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Design and develop responsive web interfaces.
2. Utilize JavaScript to create interactive web elements.
3. Implement data validation and manipulation on the client-side.
4. Develop server-side scripts to process and respond to user requests.
5. Create and manage databases for web applications.
6. Apply security principles to safeguard web applications against common vulnerabilities.
7. Deploy a functional web application to a hosting environment.

Course Overview- Course Content

As in the syllabus shared, one can traverse through the content of this course as needed.

Course Overview-Course Requirements

For faster learning, it is recommended that the attendees of this course should have basics of HTML, CSS, JavaScritpt, SQL etc. However, although it might be challenging, one may be able to cope up with the course with the concrete prior knowledge in the above mentioned areas, since lecture materials shared attempts to cover the said basics.

Course Overview-Text Books

JavaScript: The Definitive Guide. Flanagan, D. O'Reilly Media. (2020).

HTML and CSS: Design and Build Websites. Duckett, J. Wiley. (2014).

Head First HTML and CSS. Freeman, A., & Robson, E. O'Reilly Media. (2020).

PHP and MySQL Web Development. Welling, L., & Thomson, L. Addison-Wesley Professional. (2016).

Node.js For Embedded Systems. Holzner, S. Apress. (2016).

Introduction to Web Applications and Front-End Development

1. What Is a Web Application?

A web application, often referred to simply as a "web app," is a software application or program that runs on web browsers over the internet. Unlike traditional desktop applications, web applications do not need to be installed on a user's device and are accessed through a web browser. They are hosted on web servers and are typically designed to provide a specific set of functionnalities or services to users.

Key characteristics and features of web applications include:

1. **Access via Web Browsers:** Web applications are accessed through popular web browsers such as Chrome, Firefox, Safari, and Edge. Users do not need to download or install any software locally; they can simply visit a website to use the application.
2. **Platform Independence:** Web applications are platform-independent, meaning they can be used on various operating systems, including Windows, macOS, Linux, and mobile platforms like Android and iOS.

Introduction to Web Applications and Front-End Development

Key characteristics and features of web applications include:

3. **Internet Connectivity:** Web applications require an internet connection to function. Some web apps may have limited offline functionality, but they primarily rely on a connection to the web server.
4. **Centralized Data Storage:** Data for web applications is typically stored on a remote server, which allows for centralized data management and access from multiple devices.
5. **User Authentication:** Many web applications require users to create accounts and log in to access personalized features, data, or services. User authentication is essential for security and data privacy.
6. **Interactivity:** Web applications often offer interactive features, such as forms, user

Introduction to Web Applications and Front-End Development

Key characteristics and features of web applications include:

7. **Updates and Maintenance:** Web applications can be updated and maintained on the server-side, ensuring that users always have access to the latest features and security enhancements without needing to update the software locally.
8. **Scalability:** Web applications can easily scale to accommodate a growing user base by adding server resources or optimizing the codebase.
9. **Security:** Web applications need to implement security measures to protect user data, prevent unauthorized access, and defend against web-based vulnerabilities like cross-site scripting (XSS) and SQL injection.

Introduction to Web Applications and Front-End Development

2. What's the Difference Between a Web App and a Website?

Web apps are designed to be interactive whereas a website’s primary purpose is to present information. However most websites embeds web Applications in them.

Introduction to Web Applications and Front-End Development+

3. What Are Progressive Web Apps (PWAs)?

Progressive Web Apps (PWAs) are a type of web application that combines the best features of both web and native mobile applications. They are designed to provide a reliable and engaging user experience, similar to that of native apps, while still being accessible through a web browser. Here are the **key characteristics and features of PWAs:**

1. **Progressive Enhancement:** PWAs are built using web technologies (HTML, CSS, and JavaScript) and are designed to work on any platform with a standards-compliant browser. They are "progressive" in the sense that they can be accessed and used by users on older browsers but can also take advantage of modern browser features for enhanced performance and functionality.
2. **Responsive Design:** PWAs are responsive by nature, adapting their layout and content to fit various screen sizes and orientations, including desktops, tablets, and smartphones.
3. **Offline Functionality:** One of the standout features of PWAs is their ability to work offline or in low-network conditions. PWAs can cache resources and data, allowing users to access content and features even when they are not connected to the internet. This is achieved through **service workers**, which are *scripts that run in the background* and can intercept network requests.
4. **App-Like Experience:** PWAs offer a user experience similar to native mobile apps. They can be launched from the home screen, have their own app icon, and run in a standalone window without the browser's address bar, providing a more immersive feel.

Introduction to Web Applications and Front-End Development+

3. **Key characteristics and features of PWAs:+**

5. **Push Notifications:** PWAs can send push notifications to users' devices, even when the PWA is not actively open in the browser. This feature allows for real-time updates and engagement with users.
6. **Secure:** PWAs are served over HTTPS, ensuring data privacy and security. This is especially important for apps that handle sensitive information.
7. **Discoverability:** PWAs can be discovered through search engines, making them easily findable by users through web searches.
8. **Easy Updates:** Updates to PWAs are seamless for users since they don't need to go through app stores to download and install updates. Updates can be applied in the background, ensuring users have access to the latest features and security patches.

Introduction to Web Applications and Front-End Development+

3. **Key characteristics and features of PWAs:++**

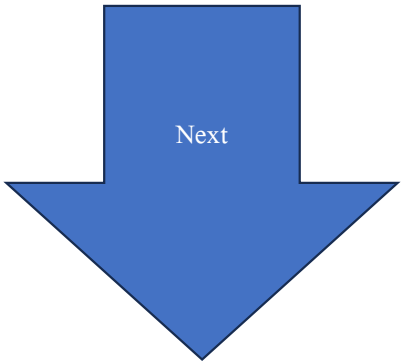
8. **Linkable:** PWAs can be shared and accessed through URLs, just like traditional websites. This makes them easy to share and distribute.
9. **Cross-Platform Compatibility:** PWAs are designed to work across various platforms and devices, reducing the need for separate development efforts for different operating systems.
10. **Reduced Data Usage:** PWAs are often optimized for efficient data usage, which can be particularly beneficial in regions with limited or expensive internet connectivity.
11. **Improved Performance:** PWAs can provide fast load times and smooth interactions, making them more engaging for users.

Introduction to Web Applications and

Front-End Development++

4. What Are the Advantages and Disadvantages of Web Applications?

Web applications have become an integral part of our digital lives, offering numerous advantages and some disadvantages. Here's a breakdown of both:



Introduction to Web Applications and Front-End Development++

Advantages of Web Apps

1. **Accessibility:** Web applications are accessible from any device with a web browser and an internet connection. This flexibility allows users to access the application from various platforms, including desktops, laptops, tablets, and smartphones.
2. **Cross-Platform Compatibility:** Web applications are typically built using standard web technologies (HTML, CSS, JavaScript), making them compatible with different operating systems, such as Windows, macOS, and Linux.
3. **No Installation Required:** Unlike native applications that need to be downloaded and installed, web applications can be used instantly, saving users time and storage space.
4. **Easy Updates:** Web applications can be updated on the server-side, ensuring that all users have access to the latest features and security patches without requiring manual updates. (OpenAI. 2021).

Introduction to Web Applications and Front-End Development++

Advantages of Web Apps+

5. **Cost-Effective Development:** Developing web applications can be more cost-effective than creating native applications for multiple platforms, as you can often use a single codebase for all users.
6. **Scalability:** Web applications can easily scale to accommodate a growing user base by upgrading server infrastructure, without the need for users to download and install updates.
7. **Centralized Data:** Data is stored on the server, which ensures consistency and enables collaborative work among users from different locations.
8. **Security:** Web applications can benefit from centralized security measures and continuous monitoring, reducing the risk of data breaches and unauthorized access.
9. **Search Engine Optimization (SEO):** Web applications can be optimized for search engines, making it easier for users to find your application through web searches. (OpenAI. 2021).

Introduction to Web Applications and Front-End Development+++

Disadvantages of Web Apps

Of course, a pro and con list would be futile if there were only pros. Naturally, web app development has disadvantages too.

1. **Dependency on Internet Connection:** Web applications require an internet connection to function. Offline functionality can be limited, and if the user has a slow or unreliable internet connection, it can lead to a poor user experience.
2. **Performance:** Web applications may not perform as well as native applications, especially for tasks that require heavy graphics processing or complex computations.
3. **Limited Access to Device Features:** Web applications have limited access to device-specific features like GPS, camera, and sensors compared to native apps.
4. **Security Concerns:** While web applications can be secure, they are susceptible to certain web-based vulnerabilities, such as cross-site scripting (XSS) and cross-site request forgery (CSRF). (OpenAI. 2021).

Introduction to Web Applications and Front-End Development+++

Disadvantages of Web Apps+

5. **Browser Compatibility:** Ensuring that a web application functions consistently across different web browsers can be challenging due to variations in browser standards and rendering engines.
6. **Less Intuitive User Experience:** Web applications may not provide the same level of user experience and responsiveness as native applications, leading to potential user dissatisfaction.
7. **Limited Offline Functionality:** Although some web applications can work offline to some extent using technologies like Progressive Web Apps (PWAs), their offline capabilities are typically more limited compared to native apps.
8. **Data Privacy Concerns:** Users may have concerns about data privacy when using web applications, as data is stored on remote servers, and they may not have full control over their personal information. (OpenAI. 2021).

Introduction to Web Applications and Front-End Development++++

5. Types of Web Applications

Web applications come in various types, each designed to fulfill specific purposes and meet the needs of different users. Here are some common types of web applications:

1. **Content Management Systems (CMS):** CMS web applications are used to create, manage, and publish digital content, such as websites and blogs. Popular CMS platforms include WordPress, Drupal, and Joomla.
2. **E-Commerce Applications:** E-commerce web applications enable online shopping and transactions. They allow users to browse products, add them to a cart, make payments, and manage their accounts. Examples include Amazon, eBay, and Shopify.
3. **Social Media Platforms:** Social media web applications facilitate communication and interaction among users. They allow people to create profiles, share content, connect with others, and engage in various online activities. Examples include Facebook, Twitter, and Instagram.
4. **Online Banking and Finance Apps:** These web applications provide users with access to their bank accounts, financial information, and online transactions. Users can check balances, transfer funds, pay bills, and manage investments. Examples include PayPal and Mint.

Introduction to Web Applications and Front-End Development++++

5. Types of Web Applications+

5. **Webmail and Email Clients:** Web-based email applications allow users to send, receive, and manage emails directly from a web browser. Examples include Gmail, Yahoo Mail, and Outlook.com.
6. **Online Learning and Education:** Web-based learning management systems (LMS) and e-learning platforms provide educational content, courses, and tools for online teaching and learning. Examples include Moodle and Coursera.
7. **Project Management and Collaboration Tools:** These web applications help teams and organizations collaborate on projects, manage tasks, share documents, and communicate. Examples include Trello, Asana, and Slack.
8. **Customer Relationship Management (CRM):** CRM web applications help businesses manage customer interactions, track sales leads, and streamline marketing efforts. Examples include Salesforce and HubSpot.
9. **File Storage and Sharing Services:** These applications allow users to store files in the cloud, share them with others, and access them from various devices. Examples include Google Drive, Dropbox, and OneDrive.

Introduction to Web Applications and Front-End Development

5. Types of Web Applications++

10. **Video Conferencing and Communication Tools:** Web-based video conferencing platforms enable online meetings, webinars, and virtual communication. Examples include Zoom and Microsoft Teams.
11. **Booking and Reservation Systems:** These web applications facilitate booking appointments, reservations, and tickets for various services, such as hotels, flights, restaurants, and events. Examples include Booking.com and OpenTable.
12. **Healthcare and Telemedicine Apps:** Web applications in the healthcare sector provide patients with access to medical records, online consultations, prescription management, and health information. Examples include Teladoc and WebMD.
13. **Gaming and Entertainment:** Online gaming platforms and streaming services offer games, videos, music, and other forms of entertainment through web browsers. Examples include Steam and Netflix.
14. **Weather and Mapping Apps:** Web-based weather services provide real-time weather forecasts and mapping applications for navigation and location-based services. Examples include Weather.com and Google Maps.
15. **Government and Civic Engagement:** Government websites and civic engagement platforms offer citizens access to government services, information, and opportunities for participation in public affairs.

These are just a few examples of the many types of web applications available today. Web applications continue to evolve and adapt to meet the changing needs of users and businesses across various industries.

Introduction to Web Applications and

Front-End Development++++

6. Web Apps Frameworks

Web application frameworks are pre-built libraries, tools, and templates that provide a structured way to develop web applications. They streamline the development process by offering a foundation for developers to build upon, reducing the need to write repetitive code. Here are some popular web application frameworks in various programming languages:

1. **JavaScript Frameworks:**
2. **Ruby Frameworks**
3. **Java Frameworks**
4. **PHP Frameworks**
5. **C#/.NET Frameworks**
6. **Go Frameworks**
7. **Node.js Frameworks**

Introduction to Web Applications and Front-End Development+++++

6. Web Apps Frameworks +

1. JavaScript Frameworks:

i. **React:** A JavaScript library for building user interfaces, particularly for single-page applications (SPAs). It's commonly used with other libraries and tools to create complete web applications.

ii. **Angular:** A comprehensive front-end framework developed by Google. It provides a full set of tools for building dynamic and complex web applications.

iii. **Vue.js:** A progressive JavaScript framework that is often compared to React and Angular. It is known for its simplicity and flexibility.

Introduction to Web Applications and Front-End Development

6. Web Apps Frameworks ++

2. Python Frameworks:

i. **Django:** A high-level Python web framework that emphasizes rapid development, clean and pragmatic code, and the "batteries-included" philosophy, offering many built-in features.

ii. **Flask:** A lightweight and micro web framework for Python. It provides the essentials for web development but allows developers to choose and integrate additional libraries as needed.

iii. **FastAPI:** A modern Python web framework for building APIs quickly and efficiently. It is known for its speed and automatic generation of documentation.

Introduction to Web Applications and Front-End Development

6. Web Apps Frameworks +++

3. Ruby Frameworks:

i. **Ruby on Rails (Rails):** A full-stack web framework for Ruby that follows the convention over configuration **(CoC)** and don't repeat yourself **(DRY)** principles. It is well-suited for building web applications rapidly.

Introduction to Web Applications and Front-End Development

6. Web Apps Frameworks ++++

4. Java Frameworks:

i. **Spring Boot:** A Java-based framework that simplifies the development of Java applications, including web applications. It offers a wide range of modules and tools.

ii. **JavaServer Faces (JSF):** A component-based Java web framework for building user interfaces for web applications. It is part of the Java EE (Enterprise Edition) platform.

Introduction to Web Applications and Front-End Development

6. Web Apps Frameworks +++++

5. PHP Frameworks:

i. **Laravel:** A PHP web application framework known for its elegant syntax and developer-friendly features. It simplifies tasks like authentication, routing, and database interactions.

ii. **Symfony:** A high-performance PHP framework that offers a wide range of reusable components and tools for building robust web applications.

Introduction to Web Applications and Front-End Development

6. Web Apps Frameworks++++++

6. C#/.NET Frameworks:

i. **ASP.NET Core:** A cross-platform, high-performance framework for building modern, cloud-based, and internet-connected applications using C#.

7. Go Frameworks:

i. **Gin:** A minimalistic and fast web framework for the Go programming language. It is designed for building efficient and scalable APIs.

Introduction to Web Applications and Front-End Development++++++

6. Web Apps Frameworks +++++++

8. Node.js Frameworks:

i. **Express.js:** A minimalist Node.js web application framework that provides a simple, unopinionated way to build web applications and APIs.

ii. **NestJS:** A TypeScript-based framework for building scalable and maintainable server-side applications using Node.js. It is often used for building APIs.

In summary, we can say these are few examples of web application frameworks available in different programming languages. The choice of framework often depends on the ***specific requirements*** *of the project*, the programming language ***expertise*** of the development team, and other factors such as *scalability*, *performance*, and *community support*.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications

Developing web applications involves a series of steps that encompass planning, designing, coding, testing, and deployment. Here's a general overview of the steps involved in the web application development process:

1. **Define the Purpose and Goals:**
    i. Clearly define the purpose of the web application.
    ii. Identify specific goals and objectives that the application should achieve.
2. **Market Research and Analysis:**
    i. Research the target audience and their needs.
    ii. Analyze competitors and similar applications in the market.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications+

3. **Planning and Requirements Gathering:**
    i. Create a project plan outlining timelines, resources, and milestones.
    ii. Gather detailed requirements for the web application's functionality and features.
4. **Wireframing and Prototyping:**
    i. Create wireframes and prototypes to visualize the application's layout and user interface (UI).
    ii. Discuss and refine the design with stakeholders.
5. **Design and UI/UX Development:**
    i. Develop the visual design, including colors, typography, and graphics.
    ii. Design the user interface (UI) and user experience (UX) to ensure usability and user-friendliness.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications++

6. **Back-End Development:**
   i. Set up the server environment and database infrastructure.
   ii. Develop the server-side logic and functionality using a chosen programming language and framework.
   iii. Implement user authentication, data storage, and API endpoints.
7. **Front-End Development:**
   i. Write HTML, CSS, and JavaScript code to create the client-side user interface.
   ii. Ensure the application is responsive and accessible on various devices and browsers.
   iii. Implement interactive features and user interactions.
8. **Integration of Third-Party Services:**
   i. Integrate third-party services or APIs (Application Programming Interfaces) as needed for functionality such as payments, social media sharing, or geolocation.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications+++

9. Testing:

i. Perform thorough testing, including functional testing, usability testing, and performance testing.

ii. Identify and fix bugs, issues, and compatibility problems.

10. Security Measures:

i. Implement security best practices to protect user data and the application from common web vulnerabilities.

ii. Conduct security audits and penetration testing if necessary.

11. Documentation:

i. Create documentation for developers and users, including code documentation and user manuals.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications++++

12. **User Acceptance Testing (UAT):**
    i. Allow stakeholders or end-users to test the application and provide feedback.
    ii. Make necessary refinements based on user feedback.
13. **Deployment:**
    i. Choose a hosting environment or cloud platform for deployment.
    ii. Set up the production server and configure domain settings.
    iii. Deploy the application to the production server.
14. **Monitoring and Maintenance:**
    i. Implement monitoring tools to track application performance and server health.
    ii. Regularly update the application with bug fixes, feature enhancements, and security patches.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications+++++

15. **Launch and Marketing:**
    i. Launch the web application to the public or a targeted user group.
    ii. Implement marketing and promotional strategies to attract users.
16. **Post-Launch Support:**
    i. Provide ongoing support to address user issues and inquiries.
    ii. Continue to improve and update the application based on user feedback and evolving requirements.
17. **Scaling and Optimization:**
    i. Monitor application usage and scale resources as needed to accommodate increased traffic.
    ii. Optimize performance, database queries, and code for efficiency.

Introduction to Web Applications and Front-End Development++++++

7. Steps To Developing Web Applications++++++

18. Data Analytics and Reporting:

i. Implement analytics tools to track user behavior and gather insights for future improvements.

Web application development is an iterative process, and ongoing updates and improvements are often necessary to meet changing user needs and maintain competitiveness in the market. Collaboration between developers, designers, project managers, and stakeholders is crucial throughout the entire development lifecycle.

Introduction to Web Applications and Front-End Development+++++++

8. Examples of Web Applications

Web applications are diverse and can be found in various domains. Here are some examples of popular web applications:

1. **Google Workspace (formerly G Suite):** Google Workspace includes web applications like Gmail (email), Google Docs (word processing), Google Sheets (spreadsheets), Google Slides (presentation software), and Google Drive (cloud storage and file sharing).
2. **Facebook:** A social media web application that allows users to create profiles, connect with friends, share updates, photos, and videos, and engage in various online activities.
3. **Amazon:** An e-commerce web application where users can browse, search for, and purchase products, as well as manage their accounts.
4. **Twitter:** A microblogging platform that enables users to post short messages (tweets), follow other users, and engage in real-time conversations.
5. **YouTube:** A video-sharing platform where users can upload, view, and share videos on a wide range of topics.
6. **Netflix:** A subscription-based streaming service that delivers movies, TV shows, and documentaries to users over the internet.

Introduction to Web Applications and Front-End Development

8. Examples of Web Applications+

Web applications are diverse and can be found in various domains. Here are some examples of popular web applications:

7. **Dropbox:** A cloud storage and file-sharing web application that allows users to store, synchronize, and share files and folders.
8. **Trello:** A project management and collaboration tool that helps teams organize tasks and projects on visual boards.
9. **Slack:** A team communication and collaboration platform that combines messaging, file sharing, and integrations with other productivity tools.
10. **Salesforce:** A customer relationship management (CRM) web application that helps businesses manage customer data, sales, and marketing efforts.
11. **Moodle:** An open-source learning management system (LMS) used for creating and managing online courses and educational content.
12. **WordPress:** A content management system (CMS) that allows users to create and manage websites and blogs with a wide range of plugins and themes.
13. **LinkedIn:** A professional networking web application that connects professionals, facilitates job searching, and offers career-related content.

Introduction to Web Applications and Front-End Development

8. Examples of Web Applications++

Web applications are diverse and can be found in various domains. Here are some examples of popular web applications:

14. **GitHub:** A web-based platform for version control and collaborative software development, used by developers to manage and share code.
15. **Zoom:** A video conferencing and online meeting platform that enables users to host and join virtual meetings, webinars, and conferences.
16. **OpenTable:** A restaurant reservation web application that allows users to search for and book tables at various restaurants.
17. **Airbnb:** An online marketplace that enables users to rent or book accommodations, including homes, apartments, and vacation rentals.
18. **Evernote:** A note-taking and organization web application that helps users capture and manage notes, documents, and ideas.
19. **Pinterest:** A social media platform focused on visual discovery and sharing, where users can "pin" and organize images and content.
20. **WebMD:** A healthcare web application that provides medical information, symptom checking, and health-related resources.

These are just a few examples of the many web applications available, each catering to specific needs and interests of users in various domains. Web applications continue to evolve and expand, offering a wide range of functionality and services.

What is Front-End Development?

Introduction to Web Applications and Front-End Development

What is Front-End Development?

Web application front-end development, often referred to simply as "front-end development," is the process of creating the user interface (UI) and user experience (UX) of a web application. Front-end development focuses on the visual and interactive aspects of a website or web application that users interact with directly through their web browsers.

Key components and responsibilities of web application front-end development include:

Introduction to Web Applications and Front-End Development

Key components and responsibilities of web application front-end development include:

1. **HTML** (Hypertext Markup Language): Front-end developers use HTML to structure the content of a web page. HTML defines the layout, headings, paragraphs, forms, and other structural elements of a web page.
2. **CSS** (Cascading Style Sheets): CSS is used to style the HTML content, making it visually appealing and user-friendly. Front-end developers use CSS to control the colors, fonts, spacing, layout, and overall visual presentation of the web application.
3. **JavaScript:** JavaScript is a programming language that adds interactivity and dynamic behavior to web pages. Front-end developers use JavaScript to create features such as interactive forms, animations, real-time updates, and client-side validation.
4. **Responsive Web Design:** Front-end developers ensure that web applications are responsive, meaning they adapt and display correctly on various screen sizes and devices, including desktops, laptops, tablets, and smartphones.

Introduction to Web Applications and Front-End Development

Key components and responsibilities of web application front-end development include:

5. **Frameworks and Libraries:** Front-end developers often leverage frameworks and libraries such as React, Angular, Vue.js, and jQuery to streamline development and build interactive web applications more efficiently.
6. **Cross-Browser Compatibility:** Front-end developers must ensure that the web application functions consistently and looks good on different web browsers (e.g., Chrome, Firefox, Safari, Edge) and browser versions.
7. **Accessibility:** Making web applications accessible to individuals with disabilities is a critical aspect of front-end development. Developers should adhere to accessibility standards (e.g., WCAG- Web Content Accessibility Guidelines) and incorporate features like alternative text for images and keyboard navigation.
8. **Performance Optimization:** Front-end developers optimize web applications for speed and performance by minimizing page load times, reducing HTTP requests, and optimizing assets like images and scripts.

Introduction to Web Applications and Front-End Development

Key components and responsibilities of web application front-end development include:

9. **Testing and Debugging:** Front-end developers test their code across various browsers and devices to identify and fix any issues or bugs. They may use browser developer tools for debugging.
10. **Collaboration:** Front-end developers collaborate closely with back-end developers, designers, and other team members to integrate front-end code with server-side logic, database interactions, and design concepts.
11. **User Experience (UX) Design:** While not all front-end developers are UX designers, they often work closely with designers to implement the user interface and ensure that the user experience is intuitive and user-friendly.
12. **Version Control:** Front-end developers use version control systems (e.g., Git) to manage and track changes to the codebase, collaborate with others, and maintain code history.

Overall, web application front-end development is crucial for creating visually appealing, interactive, and responsive user interfaces that deliver a positive user experience and engage users effectively. It involves a combination of technical skills, design principles, and collaboration with other members of the development team.(OpenAI, 2021).

Introduction to Web Applications and Front-End Development-End

Summary

1. What Is a Web Application?
2. What's the Difference Between a Web App and a Wesbite?
3. What Are Progressive Web Apps (PWAs)?
4. What Are the Advantages and Disadvantages of Web Applications?
5. Types of Web Applications
6. Web Apps Frameworks & Other Technologies
7. Steps To Developing Web Applications
8. Examples of Web Applications
9. What is Front-End Development?

HTML basics for structuring and styling web content)

What is HTML?

i. HTML stands for Hyper Text Markup Language

ii. HTML is the standard markup language for creating Web pages

iii. HTML describes the structure of a Web page

iv. HTML consists of a series of elements

v. HTML elements tell the browser how to display the content

vi. HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

A Simple Structure of HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title Here </title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Example Explained

1. The <!DOCTYPE html> declaration defines that this document is an HTML5 document
2. The <html> </html> element is the root element of an HTML page
3. The <head> </head> element contains meta information about the HTML page
4. The <title> <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
5. The <body> <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
6. The <h1> </h1> element defines a large heading
7. The   element defines a paragraph

A Simple Structure of HTML Document

HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Note: The content inside the <body> section will be displayed in a browser. The content inside the <title> element will be shown in the browser's title bar or in the page's tab.

What is an HTML Element?

The HTML **element** is everything from the start tag to the end tag:

<tagname> Content goes here... </tagname>

E.g.

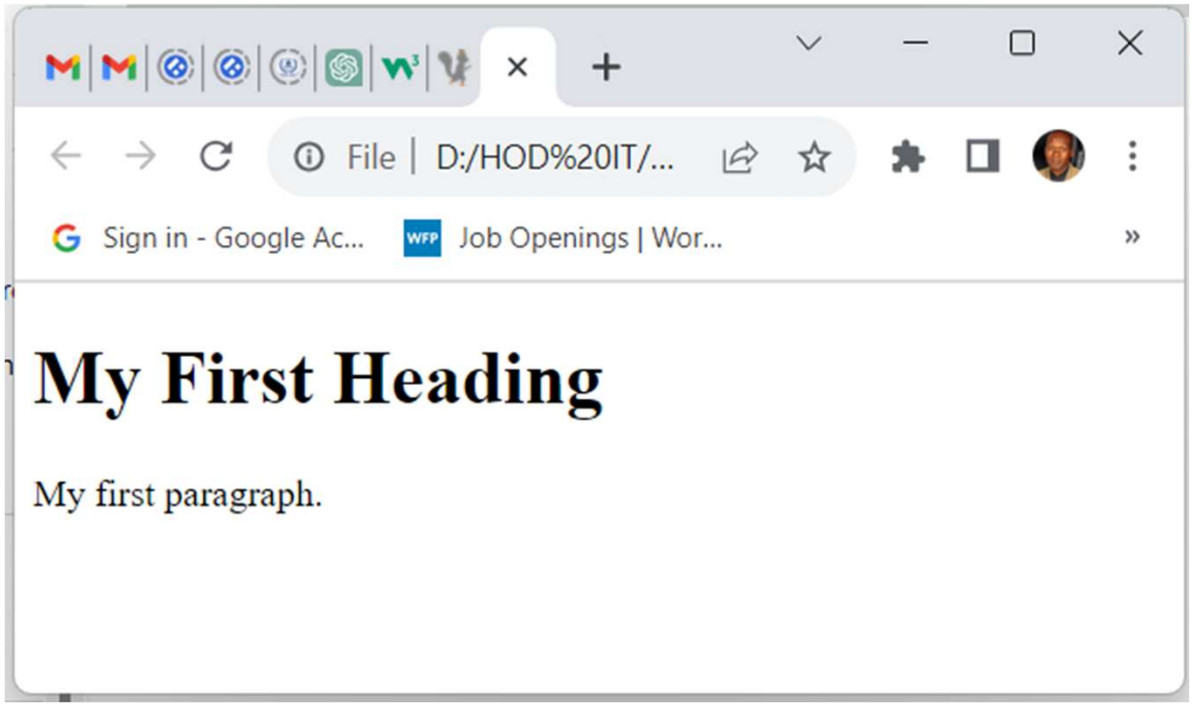
<h1>My First Heading</h1>

My first paragraph.

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

https://www.w3schools.com/html/html_intro.asp

A Simple Structure of HTML Document+Output on run



Note!

More about HTML will be covered in this course proceedings. However one can learn more about HTML in advance through https://www.w3schools.com/html/ link.

Introduction to CSS

Introduction to CSS

What is CSS?

i. CSS stands for Cascading Style Sheets

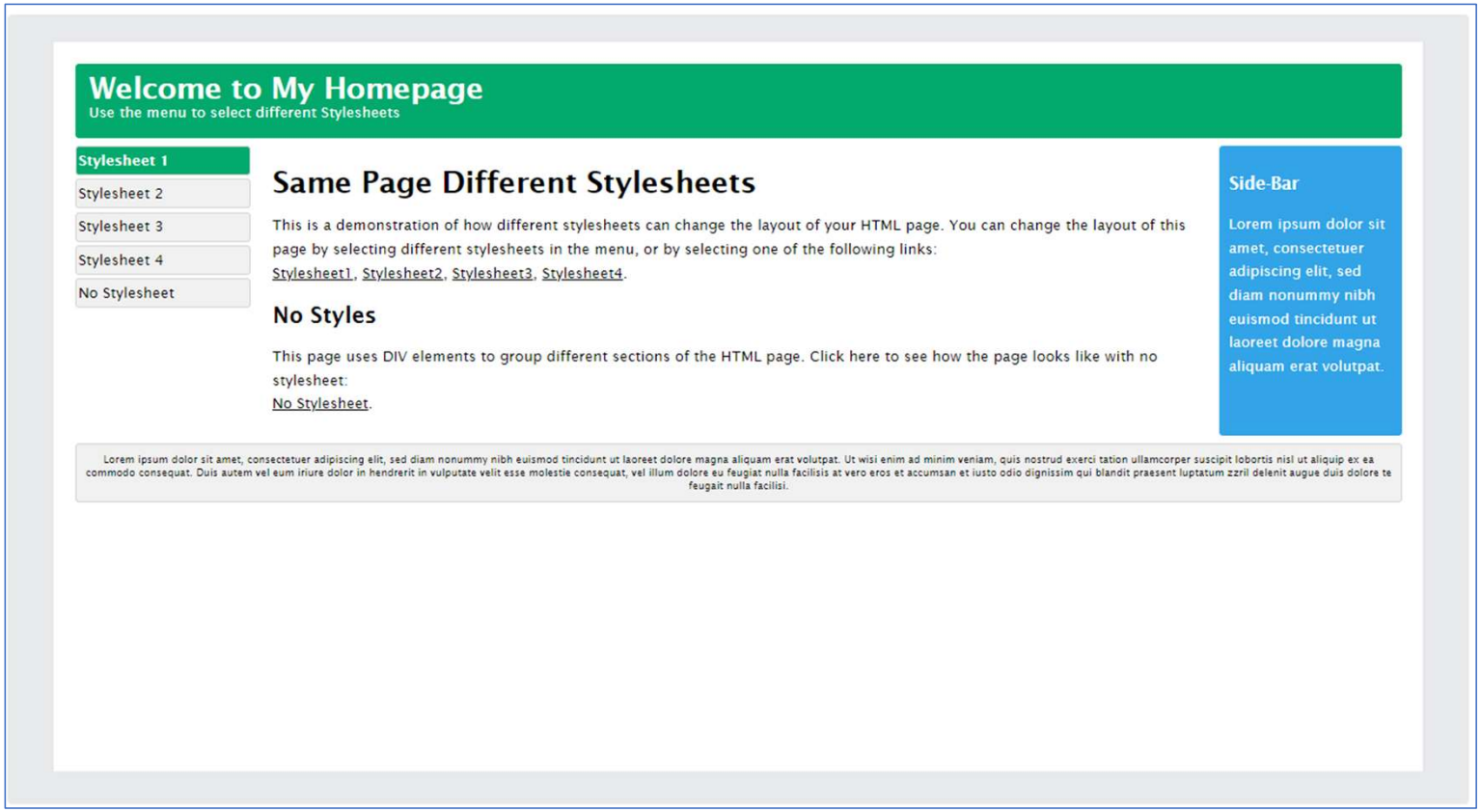
ii. CSS describes how HTML elements are to be displayed on screen, paper, or in other media

iii. CSS saves a lot of work. It can control the layout of multiple web pages all at once

iv. External stylesheets are stored in CSS files

CSS in Action - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" links below to see the different styles



(https://www.w3schools.com/css/css_intro.asp)

Why use CSS?

1. **CSS is used to define styles for your web pages**, including the design, layout and variations in display for different devices and screen sizes.

2. **CSS Solved a Big Problem**

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like:

<h1>This is a heading</h1>

This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page!

Why use CSS?+

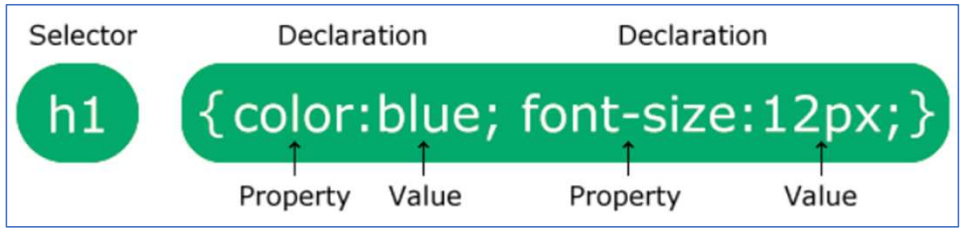
3. **CSS Saves a Lot of Work!**

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file!

CSS Syntax

A CSS rule consists of a selector and a declaration block.



(https://www.w3schools.com/css/css_syntax.asp)

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

CSS Syntax Example

In this example all <p> elements will be center-aligned, with a red text color:

```css
p {
  color: red;
  text-align: center;
}
```

Example Explained

•**p** is a selector in CSS (it points to the HTML element you want to style: <p>).

•color is a property, and red is the property value

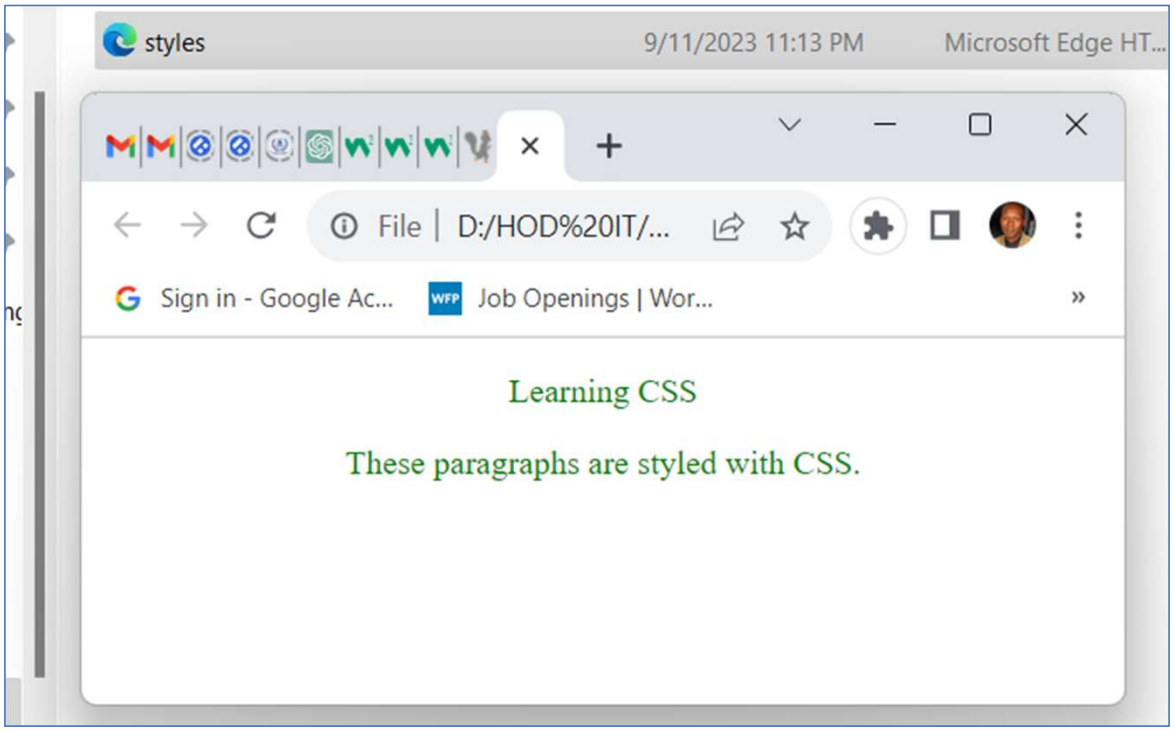
•text-align is a property, and center is the property value

Styled html page

```html
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: green;
  text-align: center;

}
</style>
</head>
<body>

<p>Learning CSS</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

CSS Syntax Example-output

Note that the output of the paragraphed text in the styles.html file above is centered and colored green.



CSS Selectors

CSS (Cascading Style Sheets) selectors are patterns or rules used to select and target specific HTML elements within a web page. They allow you to apply styles (such as colors, fonts, margins, and more) to those selected elements. CSS selectors are a fundamental part of styling web pages and are essential for creating visually appealing and consistent designs. Here are some common CSS selectors:

1. **Element Selector** (p): This selector targets all <p> (paragraph) elements on the page and sets their text color to blue.
2. **Class Selector** (highlight): This selector targets elements with the highlight class and gives them a yellow background color. In the example, it's applied to the second <p> element.
3. **ID Selector** (header): This selector targets the element with the header ID and increases its font size to 24 pixels. In the example, it's applied to the <h1> element.
4. **Universal Selector** (*): This selector applies styles to all elements on the page. In the example, it sets the margin and padding of all elements to zero, effectively resetting them.

CSS Selectors

CSS (Cascading Style Sheets) selectors are patterns or rules used to select and target specific HTML elements within a web page. They allow you to apply styles (such as colors, fonts, margins, and more) to those selected elements. CSS selectors are a fundamental part of styling web pages and are essential for creating visually appealing and consistent designs. Here are some common CSS selectors:

5. **Descendant Selector** (ul li): This selector targets all <li> elements that are descendants of a <ul> element and gives them square list bullets. It's applied to the list items within the <ul>.
6. **Adjacent Sibling Selector** (h2 + p): This selector targets a <p> element that immediately follows an <h2> element and makes its text bold. In the example, it's applied to the second <p> element.
7. **Attribute Selector** (input[type="text"]): This selector targets <input> elements with the attribute type="text" and adds a gray border to them. In the example, it's applied to the text input field.

CSS Selectors + Example code

1. Simple or Element Selector: Selects HTML elements by their element type. For example:

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

This selects all <p> (paragraph) elements and makes their text color blue.

2. Class Selector: Selects elements with a specific class attribute. Classes are prefixed with a period (dot). For example:

```
.highlight {
  background-color: yellow;
}
```

This selects all elements with class="highlight" and gives them a yellow background color.

CSS Selectors ++

3. ID Selector: Selects a single element with a specific ID attribute. IDs are prefixed with a hash (#). For example:

```
#header {
  font-size: 24px;
}
```

This selects the element with id="header" and sets its font size to 24 pixels.

4. Universal Selector: Selects all elements on the page. It's denoted by an asterisk (*). For example:

```
* {
  margin: 0;
  padding: 0;
}
```

This sets the margin and padding of all elements to zero.

CSS Selectors +++

5. Descendant Selector: Selects an element that is a descendant of another element. It uses whitespace to separate the ancestor and descendant. For example:

```
ul li {
  list-style-type: square;
}
```

This selects all <li> elements that are descendants of a <ul> element and gives them square list bullets.

6. Adjacent Sibling Selector: Selects an element that is immediately preceded by another element. It uses the plus sign (+) to separate the two selectors. For example:

```
h2 + p {
  font-weight: bold;
}
```

This selects a <p> element that directly follows an <h2> element and makes its text bold.

CSS Selectors ++++

7 Attribute Selector: Selects elements based on their attributes. It can match elements with a specific attribute or attribute value. For example:

```
input[type="text"] {
  border: 1px solid #ccc;
}
```

This selects all <input> elements with type="text" and adds a gray border.

These are some of the most commonly used CSS selectors. By combining and nesting selectors, you can precisely target and style elements within your HTML documents, creating the desired visual presentation for your web pages.

HTML page with CSS selectors above applied

In this example, we will use the file called **selectorsApplied.html** to show how the above selectors work.

latest-news.php | news-and-events.php | hosts | omara.txt | index.html | new 1 | lab2.c | styles.html | SelectorsApplied.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Selectors Example</title>
    <style>
        /* Element Selector */
        p {
            color: blue;
        }

        /* Class Selector */
        .highlight {
            background-color: yellow;
        }

        /* ID Selector */
        #header {
            font-size: 24px;
        }

        /* Universal Selector */
        * {
            margin: 0;
            padding: 0;
        }
```

HTML page with CSS selectors above applied+

In this example, we will use the file called **selectorsApplied.html** to show how the above selectors work.

```
28
29          /* Descendant Selector */
30          ul li {
31              list-style-type: square;
32          }
33
34          /* Adjacent Sibling Selector */
35          h2 + p {
36              font-weight: bold;
37          }
38
39          /* Attribute Selector */
40          input[type="text"] {
41              border: 1px solid #ccc;
42          }
43      </style>
```

HTML page with CSS selectors above applied++

In this example, we will use the file called **selectorsApplied.html** to show how the above selectors work.

```
43      </style>
44  </head>
45  <body>
46      <h1 id="header">CSS Selectors Example</h1>
47      <p>This is a simple HTML page with CSS styling using various selectors.</p>
48      <p class="highlight">This paragraph has a yellow background color due to the class selector.</p>
49      <ul>
50          <li>List item 1</li>
51          <li>List item 2</li>
52          <li>List item 3</li>
53      </ul>
54      <h2>Heading 2</h2>
55      <p>This paragraph is bold because it follows an H2 element.</p>
56      <input type="text" placeholder="Text input field">
57  </body>
58  </html>
```

End of the page

HTML page with CSS selectors above applied+++ Explained

In this example:

The **<p>** elements are styled with the element selector to have blue text.

The second **<p>** element has a class of **"highlight"** and is styled with the class selector to have a yellow background color.

The **<h1>** element has an ID of **"header"** and is styled with the **ID selector** to have a font size of 24 pixels.

The **universal selector *** is used to reset the margin and padding of all elements to zero.

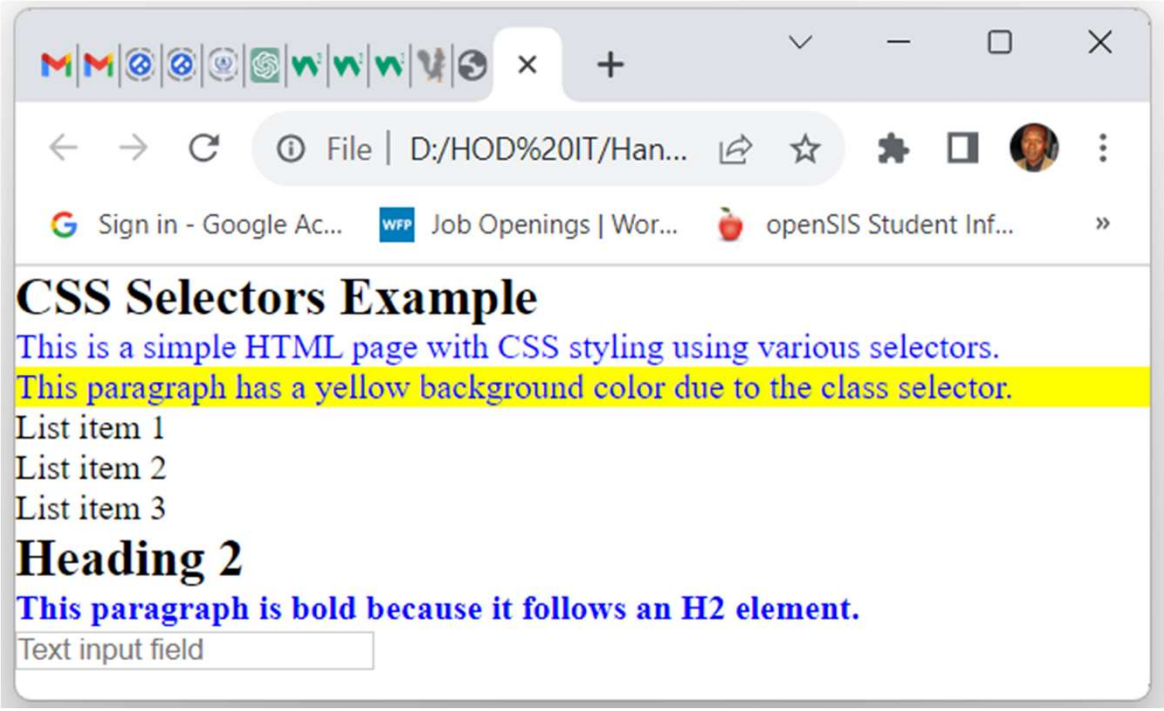
The **<ul>** and its **<li>** children are styled using the descendant selector to have square list bullets.

The <p> element immediately following an **<h2>** element is styled with the adjacent sibling selector to have bold text.

The **<input>** element with a type of **"text"** is styled with the attribute selector to have a **gray border**.

You can type the same kind ofcode into an HTML file and open it in a web browser to see the styling in action.

HTML page with CSS selectors above applied++++Output



This is just the beginning of CSS. Expect more on this in the subsequent lectures. Or keep learning through https://www.w3schools.com/css/

Menus

CSS Selectors Summary

In this section, we said CSS selectors are patterns or rules used to select and target specific HTML elements within a web page, giving examples as being; **Element Selector** (p), **Class Selector** (.), **ID Selector** (#), **Universal Selector** (*), **Descendant Selector** (ul li), **Adjacent Sibling Selector** (h2 + p), and **Attribute Selector** (input[type="text"]).

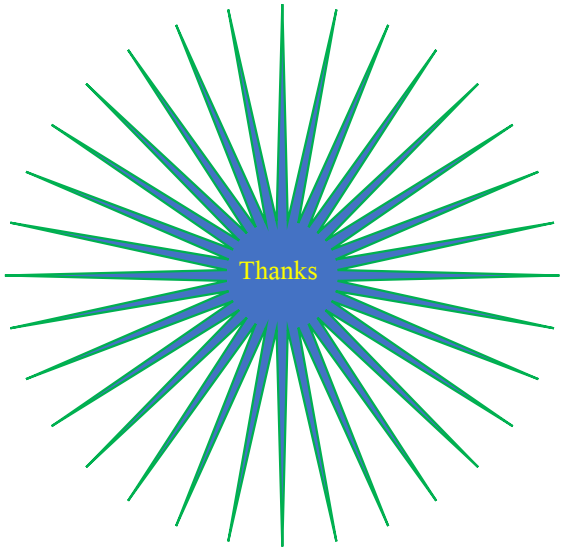
More about CSS selectors will be interfaced in the subsequent lectures

Summary

General Summary

1. Course Overview
2. Introduction to Web Applications and Front-End Development (Introduction to web technologies and architecture and
3. HTML and CSS basics for structuring and styling web content)

Thank you for Listening



References

Brewster, C. (2023a, July 17). Web app development in 2023: Everything you need to know. Trio. https://www.trio.dev/front-end/resources/web-app-development

OpenAI. (2021, September 11). Advantages and disadvantages of Web applications. OpenAI Knowledge Base.

HTML introduction. Introduction to HTML. (n.d.). https://www.w3schools.com/html/html_intro.asp

CSS introduction. (n.d.). https://www.w3schools.com/css/css_intro.asp