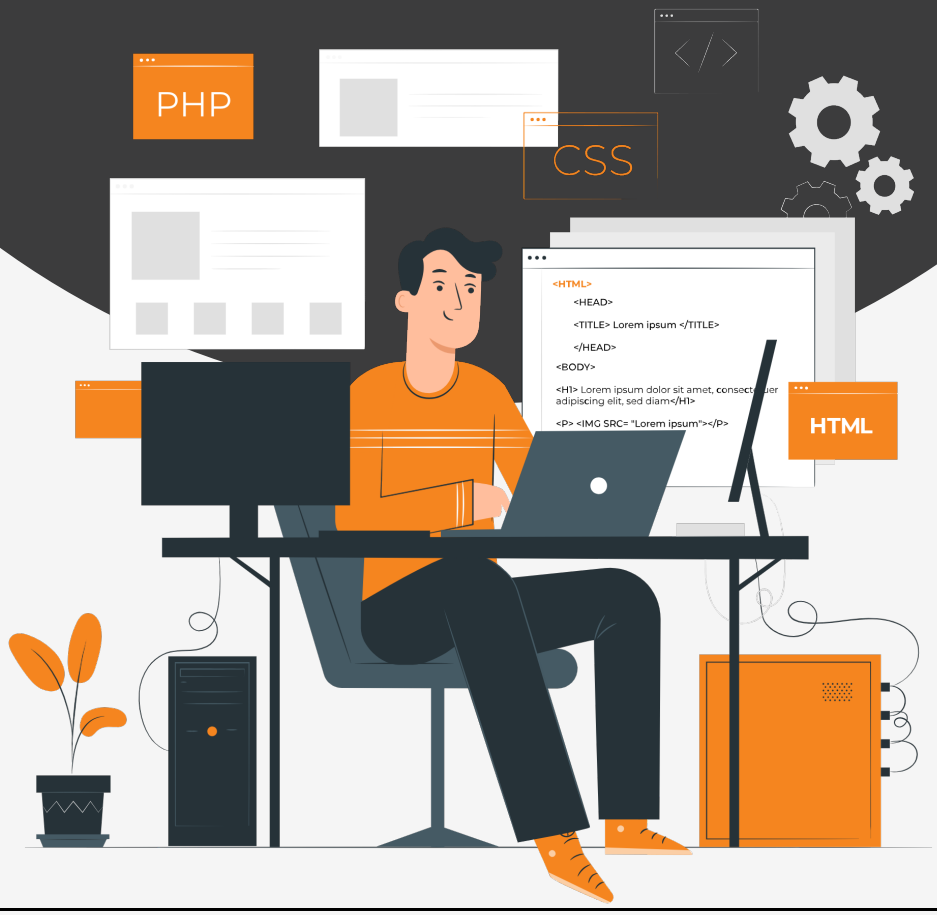
Lesson:

Web Browser



PW SKILLS

Topics to be covered in

1. Introduction to the web browsers
2. History of web browsers
3. Features of web browsers
4. Introduction to browser engine
5. Responsibilities of browser engine
6. Browser rendering process
7. Common browser compatibility issues

Introduction to web browsers

A web browser is a software application that enables users to access and browse the World Wide Web. It provides a graphical user interface for displaying and interacting with web pages and allows users to navigate between pages, view multimedia content, and interact with web-based applications. There are a variety of web browsers available, each with its own unique features and capabilities.

History of web browsers

The first web browser, called WorldWideWeb, was developed by Tim Berners-Lee in 1990. It was a simple text-based browser that ran on the NeXTSTEP operating system. In 1993, the Mosaic browser was released, which was the first graphical web browser to gain widespread popularity. Mosaic was later replaced by the Netscape Navigator browser, which dominated the market throughout the mid-1990s. In the late 1990s, Microsoft released its own web browser, Internet Explorer, which quickly became the dominant browser on the Windows platform. Today, there are many web browsers available, including Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera.

Features of web browsers

Web browsers offer a wide range of features and capabilities to enhance the user's browsing experience. Here are some common features found in most web browsers:

1. **Tabbed browsing:** Most modern web browsers allow users to open multiple web pages in separate tabs within a single window, making it easier to switch between them.
2. **Bookmarks:** Browsers allow users to save links to their favourite web pages for easy access later.
3. **History:** Web browsers keep track of the user's browsing history, allowing them to quickly revisit sites they have previously visited.
4. **Privacy and Security:** Browsers include features such as private browsing mode, which prevents the browser from saving any data about the user's browsing session. Additionally, browsers include security features such as phishing and malware protection.
5. **Extensions and Plugins:** Browsers offer various extensions and plugins, allowing users to customize their browsers and add new features or functionality.
6. **Developer Tools:** Most browsers include a suite of tools for web developers, including the ability to inspect and debug web pages, view page source code, and manipulate the Document Object Model (DOM).

Introduction to Browser Engine

A browser engine, also known as a rendering engine, is a software component that processes HTML, CSS, and JavaScript code to display web pages on a user's screen. When a user requests a web page, the browser engine retrieves the page's code from the server and processes it to create a visual representation of the page. Overall, the purpose of a browser engine is to enable users to browse the web and interact with web pages in a fast, efficient, and reliable manner, while also providing developers with the tools they need to build and maintain high-quality web applications.

Some major responsibilities of the browser engine are as follows:-

1. **Rendering web pages:** The primary purpose of a browser engine is to render web pages on a user's screen. When a user visits a web page, the browser engine processes the HTML, CSS, and JavaScript code to create a visual representation of the page. This process includes parsing the code, determining the layout of the page, and rendering the content on the screen.
2. **Handling user interactions:** In addition to rendering web pages, browser engines are also responsible for handling user interactions with those pages. This includes responding to user clicks, handling form submissions, and updating the page in real time in response to user actions.
3. **Optimizing page load times:** Browser engines play a critical role in optimizing page load times. They use techniques such as preloading and caching to ensure that pages load as quickly as possible. For example, a browser engine may preload resources that are likely to be needed next to reduce the amount of time it takes to load a page.
4. **Supporting web standards:** Browser engines are responsible for supporting web standards and ensuring that web pages are displayed correctly across different browsers and devices. This includes implementing the latest HTML, CSS, and JavaScript standards, as well as supporting older legacy technologies that are still in use.
5. **Providing developer tools:** Finally, browser engines provide a set of developer tools that allow web developers to debug and inspect web pages as they're being rendered. These tools can be used to identify and fix issues with web pages, optimize performance, and test for compatibility across different browsers and devices.

How does a browser display a website from a url?

The process of displaying a web page from url in a browser is followed by these steps -

1. Domain Name System (DNS)
   a. The browser begins by performing a DNS lookup to translate the website's domain name (e.g., www.pwskills.com) into an IP address.
   b. It sends a request to DNS servers to retrieve the IP address associated with the domain.

2. Establishing a TCP Connection
   a. The browser establishes a TCP (Transmission Control Protocol) connection with the web server identified by the IP address obtained from the DNS lookup.
   b. TCP ensures reliable communication by dividing data into packets and managing their transmission.

3. Sending an HTTP Request
    a. The browser sends an HTTP (Hypertext Transfer Protocol) request to the web server.
    b. The request includes the HTTP method (such as GET or POST) and the requested resource's URL.

4. Server Processing
    a. The web server receives the HTTP request and processes it.
    b. It may execute server-side code, access databases, or perform other necessary operations.

5. Generating the HTTP Response
    a. The web server generates an HTTP response, which includes the requested resource (such as an HTML file) and associated metadata.
    b. The response is typically encoded in HTML, but it can also include CSS, JavaScript, images, or other resources.

6. Transmitting the HTTP Response
    a. The web server sends the HTTP response back to the browser over the established TCP connection.
    b. The response is divided into packets, and the TCP protocol ensures their reliable delivery.

7. Rendering the HTML
    a. The browser receives the HTTP response and begins rendering the HTML content.
    b. It parses the HTML document from top to bottom, creating a Document Object Model (DOM) representation of the webpage's structure.

8. Fetching External Resources
    a. While parsing the HTML, the browser encounters external resources such as CSS stylesheets, JavaScript files, or images.
    b. It sends additional requests to retrieve these resources, allowing them to be rendered and applied to the webpage.

9. Executing JavaScript
    a. If the HTML includes JavaScript code, the browser executes it.
    b. JavaScript can dynamically modify the webpage's structure, content, and behavior.

10. Rendering the Webpage
    a. The browser combines the parsed HTML, applied CSS styles, and executed JavaScript to render the webpage visually.
    b. The rendering process involves laying out elements, applying styles, and displaying the final result on the screen.

11. User Interaction and Event Handling
    a. The rendered web page becomes interactive, allowing users to interact with buttons, forms, links, and other elements.
    b. The browser handles user events (such as clicks or form submissions) and triggers appropriate actions.

13. Ongoing Interaction
    c. The browser maintains the established TCP connection with the web server for potential future requests or updates.
    d. It continues to process user interactions, handle additional HTTP requests, and update the webpage dynamically as needed.

Common browser compatibility issues

Here is a list of some common browser compatibility issues, which is generally faced by the users -

1. CSS rendering differences

Variation in how browsers interpret and display CSS styles, leading to differences in layout, positioning, and visual appearance of elements.

2. JavaScript Compatibility

Differences in JavaScript implementations and support for various ECMAScript versions across different browsers, resulting in inconsistent behavior or errors.

3. HTML Markup Interpretation

Variations in how browsers parse and interpret HTML markup, potentially causing differences in rendering and functionality.

4. Cross-Browser CSS3 Features

Limited or inconsistent support for CSS3 features, requiring fallback options or alternative approaches to ensure consistent rendering across different browsers.

Note: We can see the compatibility table of any properties or functions that can be used in different browsers on platforms like MDN.

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android
form	✓ 1	✓ 12	✓ 1	✓ 15	✓ 4	✓ 18	✓ 4	✓ 14	✓ 3.2	✓ 1.0	✓ 4.4
accept	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4
accept-charset	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4
action	✓ 1	✓ 12	✓ 1	✓ 15	✓ 4	✓ 18	✓ 4	✓ 14	✓ 3.2	✓ 1.0	✓ 4.4
autocapitalize	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✓ Yes	✗ No	✗ No
autocomplete	✓ 14 *	✓ 12	✓ 4	✓ 15 *	✓ 6	✓ 18 *	✓ 4	✓ 14 *	✓ 6	✓ 1.0 *	✓ 4.4 *
enctype	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4
method	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4
name	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4
novalidate	✓ 10	✓ 12	✓ 4	✓ 15	✓ 10.1	✓ 18	✓ 4	✓ 14	✓ 10.3	✓ 1.0	✓ 37
rel	✓ 108	✓ 108	✓ 111	✓ 94	✓ 15.4	✓ 108	✓ 111	✓ 73	✓ 15.4	✓ 21.0	✓ 108
target	✓ 1	✓ 12	✓ 1	✓ 15	✓ 3	✓ 18	✓ 4	✓ 14	✓ 2	✓ 1.0	✓ 4.4