Basics of Web

TechRacine
Integrations Simplified

Table of Contents

1. What is Web?

The web, short for the World Wide Web (WWW), is a system of interconnected documents and resources, primarily accessed via the internet, that allows people to share and access information globally.

Key Concepts of the Web:

1. Web Pages: These are documents written in HTML (HyperText Markup Language) that are displayed in a web browser (such as Chrome or Firefox). Web pages can contain text, images, videos, and other multimedia.

2. Web Browser: A software application (like Chrome, Firefox, Safari) that lets users access and view web pages. When you type a URL or click on a link, the browser fetches the web page from the server and displays it.

3. Hyperlinks: Links embedded in web pages that allow users to navigate from one web page to another. This linking of pages is what makes the web a "web" of information.

4. Web Servers: Computers that store and serve web pages to clients (web browsers) via the internet. When you request a page (e.g., by typing a URL), the server sends the requested content back to your browser.

5. HTTP/HTTPS: These are the protocols (rules) that dictate how web pages are requested and transmitted between clients (browsers) and servers. HTTP (HyperText Transfer Protocol) is the foundation of data communication for the web, and HTTPS is its secure version.

6. URL (Uniform Resource Locator): The address you type into the browser to access a specific web page. It usually starts with "http://" or "https://".

Difference Between the Internet and the Web:
- The internet is the global network of interconnected computers.
- The web is one of the services that runs on the internet, allowing users to access websites, documents, and multimedia.

In summary, the web is a collection of websites and information that users access through browsers over the internet, using protocols like HTTP or HTTPS to transfer data between servers and clients.

2. How web works?

To explain how the web works, let's break it down into basic steps with a diagram to help visualize the process.

Key Components of the Web

Client: The device (computer, smartphone, etc.) used by a user to access a website. The client runs a browser (like Chrome, Firefox, etc.).

Server: A computer where the website's files (HTML, CSS, JavaScript, etc.) and resources are stored.

Internet: The network that connects clients and servers, allowing them to communicate.

Request: When a user wants to visit a website, the browser sends a request to the server.

Response: The server processes the request and sends back the requested files (web pages, images, etc.) to the browser.

How the Web Works: Step-by-Step

User enters a website address (URL): A user types a URL (e.g., www.example.com) into the browser.

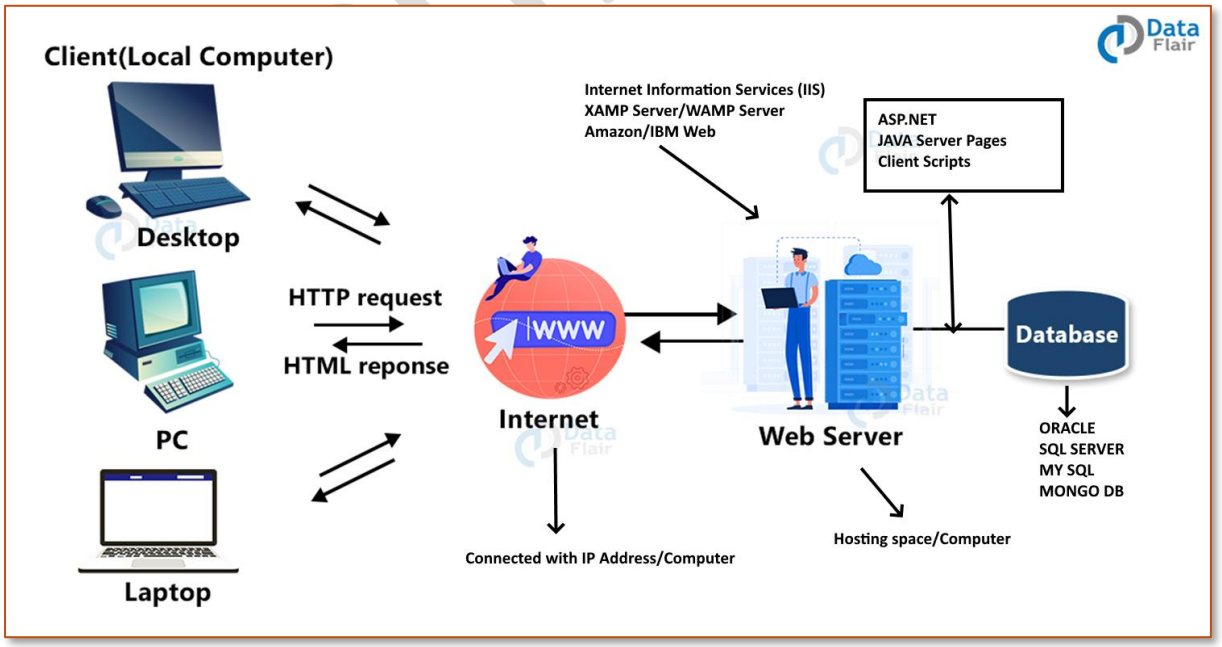
DNS Lookup: The browser sends a request to a DNS (Domain Name System) server to find the IP address of the website's server. The DNS translates the URL (like www.example.com) into the IP address (e.g., 192.168.1.1) of the web server.

Request sent to server: The browser sends an HTTP or HTTPS request to the server that hosts the website, asking for the web page (e.g., index.html).

Server processes request: The server processes the request, prepares the content (HTML, CSS, JavaScript, etc.), and sends a response back to the client.

Browser renders the webpage: The browser receives the HTML file, interprets it, loads any necessary CSS and JavaScript, and renders the page for the user to view and interact with.

3. Diagram of How web works



4. Explanation of Terms

1. **Website:**
   A collection of related web pages under a single domain (like "www.example.com"). Websites can be simple static pages or dynamic applications with interactive elements.

2. **Web Application:**
   A web-based software application that allows users to perform tasks or interact with data, such as Gmail, Facebook, or Google Docs. Unlike simple websites, web applications have more complex functionality and often involve back-end databases and logic.

3. **Front-end:**
   This refers to everything that a user interacts with directly on the web, such as the layout, design, and user interface of a website or web application. Technologies like HTML, CSS, and JavaScript are used to create the front-end.

4. **Back-end:**
   The part of a website or web application that is not visible to users. The back-end handles server-side logic, databases, and application functionality. Back-end technologies often include programming languages like PHP, Python, Node.js, Ruby, and C#, along with databases like MySQL or MongoDB.

5. **HTTP (HyperText Transfer Protocol):**
   The protocol used by the web for communication between the browser (client) and the server. It defines how messages are formatted and transmitted, and how web servers respond to commands.

6. **HTTPS (HyperText Transfer Protocol Secure):**
   An extension of HTTP that adds a layer of security through encryption. HTTPS ensures that data sent between the client and the server is encrypted and secure, making it harder for hackers to intercept.

7. **CSS (Cascading Style Sheets):**
   A language used to describe the style and appearance of web pages. It controls the layout, colors, fonts, and overall look of the page.

8. **JavaScript:**
   A programming language that adds interactivity to websites. It allows developers to create dynamic elements like animations, form validation, and real-time updates without needing to reload the page.

9. **AJAX (Asynchronous JavaScript and XML):**
   A technique that allows web applications to update content dynamically without refreshing the entire web page. For example, when you send a message in a chat app, the page doesn't refresh, but the message appears instantly.

10. **Domain Name:**
    The human-readable address of a website, such as "www.google.com". It maps to an IP address that identifies the server where the website is hosted.

11. **IP Address (Internet Protocol Address):**
    A unique set of numbers that identifies each device connected to the internet. Servers and clients (computers, phones) use IP addresses to locate and communicate with each other over the web.

12. **Hosting:**
    The service that provides storage space and access for websites on the internet. Web hosting companies rent out servers where websites are stored and made accessible to users.

13. **Cookies:**
    Small files stored on a user's computer by websites to remember information, such as login details, shopping cart items, or preferences. Cookies enable the site to "remember" a user between visits.

14. **SEO (Search Engine Optimization):**
    The process of optimizing a website to rank higher in search engine results (like Google). SEO involves improving various aspects of the site, including keywords, loading speed, mobile-friendliness, and backlinks.

15. **URL (Uniform Resource Locator):**
    The address of a specific resource on the web, such as a web page, image, or video. It typically includes the protocol ("http" or "https"), domain name, and file path (e.g., "https://www.example.com/about.html").

16. **API (Application Programming Interface):**
    A set of rules and tools that allows different software applications to communicate with each other. For example, websites may use APIs to retrieve data from external sources, such as displaying weather information or integrating with social media.

17. **DNS (Domain Name System):**
    A system that translates human-readable domain names (like "www.example.com") into IP addresses (like "192.168.1.1"), so browsers can locate the servers hosting the requested websites.

18. **Firewall:**
    A security system that monitors and controls incoming and outgoing network traffic, blocking unauthorized access to or from a network, such as a web server.

19. **SSL Certificate (Secure Sockets Layer):**
    A digital certificate is used to establish a secure, encrypted connection between a web server and a browser. Websites with SSL certificates use HTTPS and display a padlock icon in the browser's address bar, indicating that the connection is secure.

20. **CDN (Content Delivery Network):**
    A network of servers distributed across various locations that helps deliver web content faster by caching copies of the website's data at various geographic points. This reduces the time it takes for the data to travel from the server to the user.

21. **Responsive Design:**

A web design approach that ensures websites work well on various devices, including desktops, tablets, and smartphones. Using techniques like media queries and flexible layouts, responsive design adapts the content to different screen sizes.

22. **UI/UX (User Interface / User Experience):**
    UI (User Interface) refers to the design and layout of a website or app, focusing on the visual elements and interaction points.
    UX (User Experience) refers to the overall experience a user has when navigating and interacting with a website or application, aiming for ease of use and satisfaction.

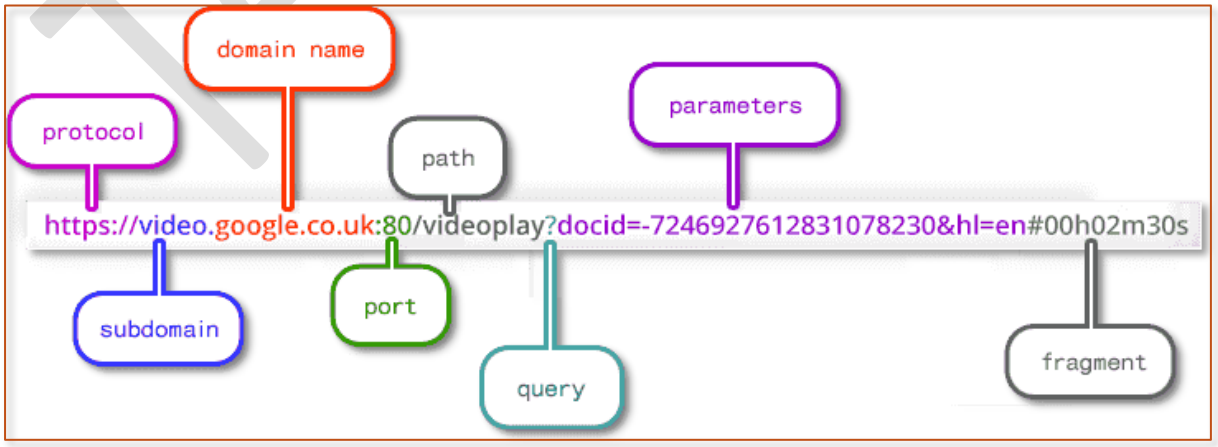
5. URL

URL (Uniform Resource Locator)
A URL is the web address you enter into your browser to access a specific resource (like a web page, an image, or a file) on the internet. It not only identifies the resource but also provides the information on how to access it, including the protocol (e.g., HTTP or HTTPS) and the location (the domain name or IP address).

URL Components:
- Protocol: The method used to access the resource (e.g., "http://" or "https://").
- Domain Name: The address of the server where the resource is hosted (e.g., "www.example.com").
- Path: The specific location of the resource on the server (e.g., "/about.html").
- Optional Parameters: Additional data can be sent in the form of query parameters (e.g., "?id=123").

Example of a URL:
https://www.example.com/about.html?user=123
- "https": Protocol
- "www.example.com": Domain name
- "/about.html": Path
- "?user=123": Query parameters

Example diagram



6. HTTP vs HTTPS

Aspect	HTTP (HyperText Transfer Protocol)	HTTPS (HyperText Transfer Protocol Secure)
Security	HTTP is **not secure**. Data sent between the client (browser) and the server is in plain text, making it vulnerable to interception (man-in-the-middle attacks).	HTTPS is **secure**. It encrypts data using SSL/TLS (Secure Sockets Layer / Transport Layer Security), making it harder for attackers to intercept or tamper with the communication.
Data Encryption	No encryption. Data is transmitted in readable text, which can be easily accessed if intercepted.	Encrypted. Data is scrambled using SSL/TLS encryption, making it unreadable without the decryption key.
Port Number	By default, HTTP uses **port 80** for communication.	HTTPS uses **port 443** by default for secure communication.
SSL/TLS Certificate	Does not require an SSL/TLS certificate for the website to operate.	Requires an **SSL/TLS certificate** issued by a Certificate Authority (CA) to authenticate the website and enable encryption.
SEO Ranking	Websites using HTTP are less favored by search engines like Google in terms of ranking.	Websites using HTTPS are given a **higher SEO ranking** by search engines, as HTTPS is considered a security and trust factor.
Visual Indicators	Browsers typically show no special indicators (some browsers may show a warning for non-secure HTTP sites on sensitive forms).	HTTPS shows a **padlock icon** in the address bar, indicating the connection is secure. In some cases, websites with extended validation certificates show the organization's name next to the padlock.
Performance	Generally faster because there's no encryption overhead, but lacks security.	HTTPS may have slight overhead due to encryption, but modern improvements (like HTTP/2) have minimized this impact, making HTTPS sites almost as fast or faster than HTTP in some cases.
Use Case	Suitable for non-sensitive information (though it's increasingly discouraged for any website).	Necessary for handling sensitive information, such as **credit card numbers**, personal data, and login credentials. It's recommended for **all websites** for better security and trustworthiness.

- **HTTP** is an **insecure** protocol used for transferring web pages and data in plain text, while **HTTPS** is the **secure** version that encrypts the data between the browser and the server using SSL/TLS.

- Websites using **HTTPS** are more secure, improve trust with users, and offer better SEO rankings compared to sites using HTTP

7. How Browsers renders?

Web browsers use rendering engines (also called layout or browser engines) to interpret and convert HTML, CSS, and JavaScript into the visual web pages you see on the screen. Each browser has its own rendering engine, which is responsible for transforming the code into something users can interact with. Here's a breakdown of how browsers interpret HTML using their engines:

1. HTML Rendering Process in Browsers
When you enter a URL or load a web page, the browser follows these steps:

a. HTTP Request
- The browser sends an HTTP/HTTPS request to the web server to retrieve the HTML document and associated resources (like images, CSS, and JavaScript files).

b. Receiving and Parsing the HTML
- Once the server responds, the browser starts parsing the HTML document, line by line, into a DOM (Document Object Model) tree. This DOM is a structured representation of the HTML elements.

c. Parsing CSS and JavaScript
- The browser fetches any CSS files and JavaScript scripts linked within the HTML.
- The CSS is parsed into a CSSOM (CSS Object Model) tree, which defines the styles of HTML elements.
- JavaScript is interpreted and executed (blocking rendering if it's not async or deferred).

d. Rendering Process
- The browser combines the DOM and CSSOM to create a render tree, which determines the layout of elements on the screen. Each node in the DOM corresponds to an element on the page.
- The browser then calculates the layout of these elements (their size, position, etc.), a process called layout or reflow.

e. Painting
- After the layout is calculated, the browser proceeds to paint the pixels onto the screen. This step involves drawing text, colors, images, borders, and other visual elements.

f. Compositing
- Finally, different layers (such as for animations or 3D transformations) are combined in a process called compositing, which results in the final visible page.

2. Rendering Engines in Browsers
Each browser uses a specific rendering engine responsible for carrying out the above tasks:

a. Blink (Google Chrome, Microsoft Edge, Opera)
- Blink is an open-source rendering engine developed by Google and used in Chrome, Microsoft Edge (from version 79 onwards), Opera, and several other browsers.
- Blink is based on WebKit, the engine used in Safari, but it has since diverged and been optimized by Google.
- It is known for fast rendering and support for the latest web standards.

b. WebKit (Safari)
- WebKit is the rendering engine used by Apple Safari on macOS and iOS.
- It was initially developed as part of the KDE project and is known for its excellent integration with macOS and iOS devices.
- WebKit is efficient in handling animations and media on Apple devices and adheres closely to web standards.

c. Gecko (Mozilla Firefox)
- Gecko is the rendering engine developed by the Mozilla Foundation and is used in Firefox and other Mozilla-based browsers.
- Gecko emphasizes open web standards and is highly customizable, allowing developers to experiment with new features.
- It is known for its flexibility and ability to render complex, standards-compliant content.

d. Trident (Internet Explorer) and EdgeHTML (Legacy Microsoft Edge)
- Trident is the engine used in older versions of Internet Explorer (IE 11 and earlier).
- EdgeHTML was a fork of Trident used in the legacy version of Microsoft Edge (pre-Chromium version).
- Both engines had limitations in rendering modern web standards, which led to Microsoft adopting Blink in newer versions of Edge.

3. JavaScript Engines
Each rendering engine is paired with a JavaScript engine that executes the JavaScript code embedded in HTML pages:

- V8: Used by Chrome, Edge, and Opera. V8 is known for its fast execution of JavaScript code and powers both browser environments and server environments like Node.js.
- SpiderMonkey: Used by Firefox. SpiderMonkey is optimized for Firefox and adheres closely to JavaScript standards.
- JavaScriptCore (Nitro): Used by Safari. It's the JavaScript engine integrated with WebKit.
- Chakra: Used by the old Microsoft Edge (pre-Blink version).

4. How Rendering Engines Work
Rendering engines follow a process called the Critical Rendering Path, which includes the following steps:

1. Building the DOM Tree:
- The engine converts the HTML into the DOM tree (a representation of HTML structure).

2. Building the CSSOM Tree:
- CSS is parsed into a CSSOM tree to represent how styles are applied to each element.

3. Creating the Render Tree:
- The DOM and CSSOM trees are combined to form the render tree, which contains only the elements that need to be visible on the page (e.g., hidden elements like `display: none` are excluded).

4. Layout:
- The engine calculates the size and position of each element in the render tree, determining the page layout.

5. Painting:
- The engine paints each element to the screen, determining its visual appearance.

6. Compositing:
- For more complex layouts (e.g., involving 3D transforms, videos, or animations), the engine combines different layers into the final image.

5. Browser Optimization Techniques
- Lazy Loading: Delays loading images or content until needed, speeding up page rendering.
- Minification: Reduces the size of HTML, CSS, and JavaScript files to improve load time.
- Caching: Stores previously loaded resources to avoid downloading them again.