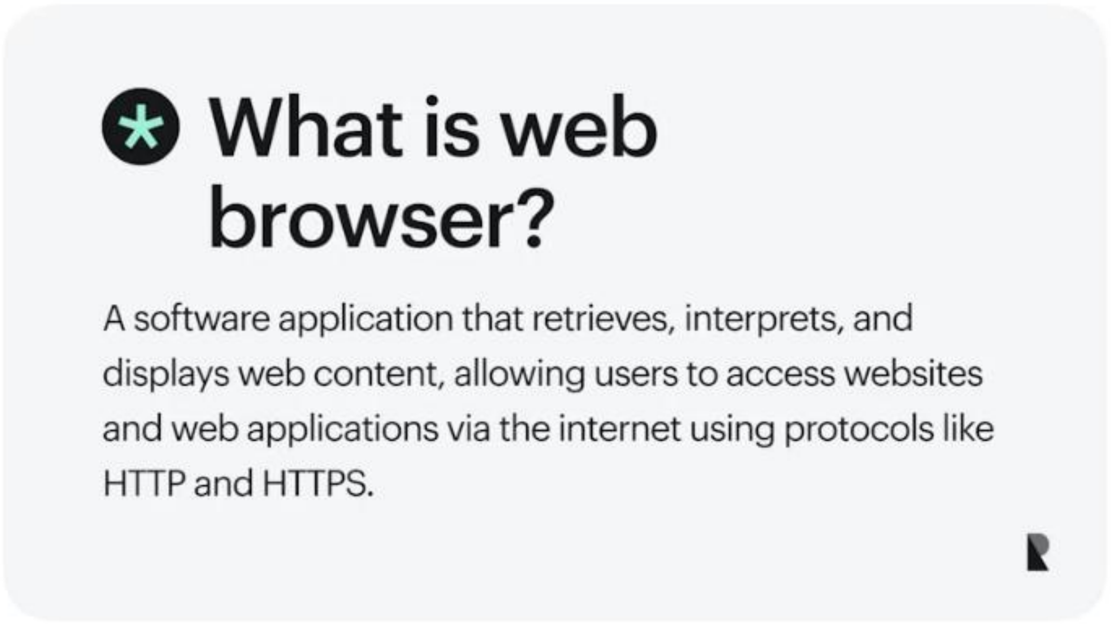
Browsers

Web browser interpret HTML, CSS, and JavaScrip to render web pages, modern browsers use rendering engine (Blink, Geckim Webkit) and JavaScript engine, offering features like tabs, bookmarks, extension, and security through sandboxing and HTTPS enforcement.

Introduction



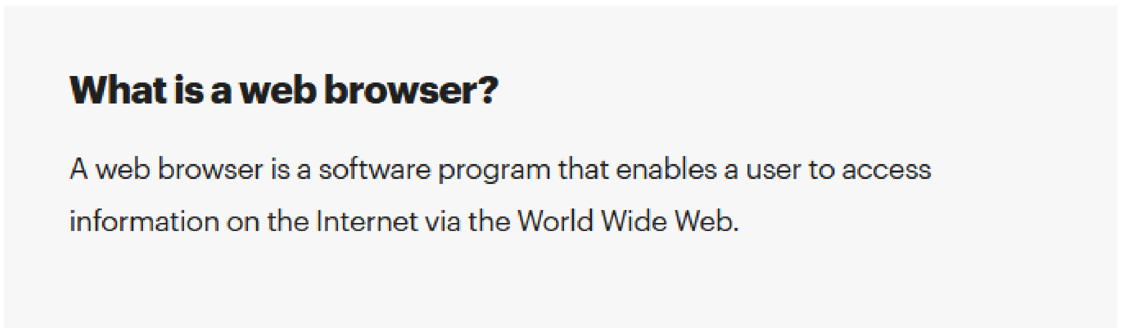
What is web browser?

A software application that retrieves, interprets, and displays web content, allowing users to access websites and web applications via the internet using protocols like HTTP and HTTPS.

Witch the rise of the internet, digital media has become essential for business to build their online presence and engage with customers. Web browsers play a key role in this process, serving as vital tools for companies to establish themselves in the competitive market. The article discusses the fundamentals of web browsers, how they function, their features, and traces their historical development. It emphasizes the importance of understanding web browsers in today1s digital landscapes.

Defining Web Browsers

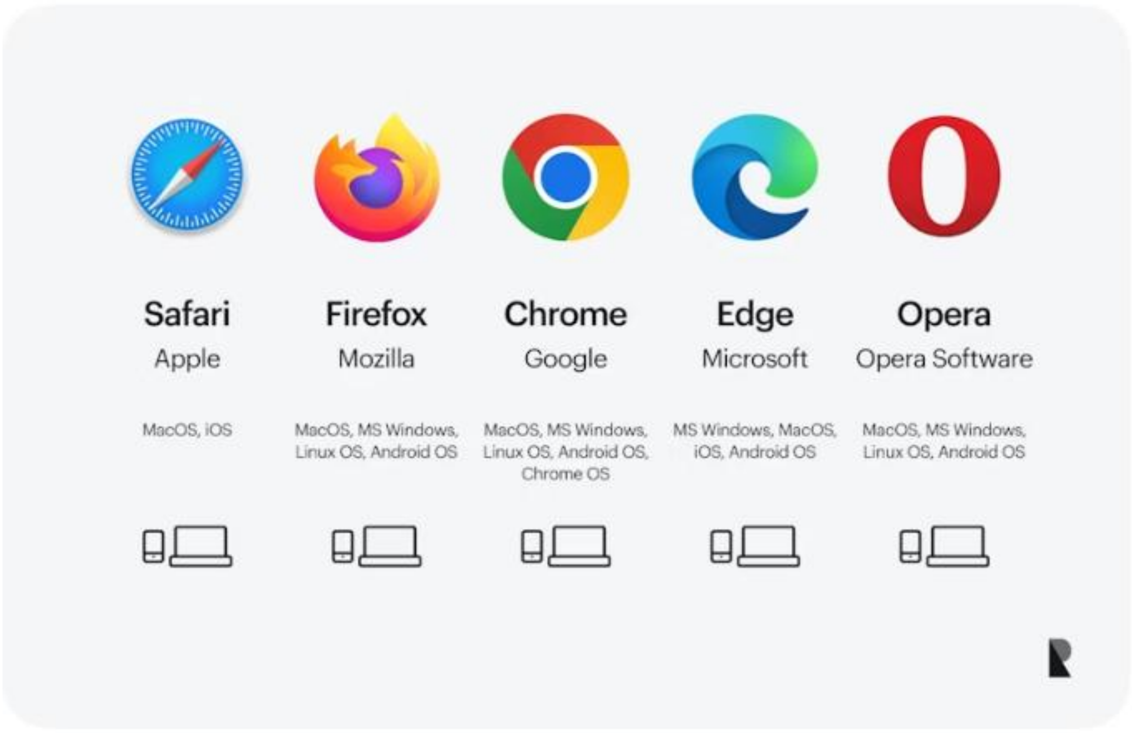
Web browsers are essential tools for businesses and applications to establish their presence on the internet. They server as gateways to the digital world, allowing users to access and interact with online content. Web development heavily relies on browsers, leading companies to hire or outsource to maintain and improve them.



What is a web browser?

A web browser is a software program that enables a user to access information on the Internet via the World Wide Web.

A web browser is a software application that retrieves and displays web pages from servers, enabling users to view text, images, and videos online. By entering URL, the browser fetches data from the corresponding servers and presents it a un a user-friendly format.


Safari
Apple
MacOS, iOS
Firefox
Mozilla
MacOS, MS Windows, Linux OS, Android OS
Chrome
Google
MacOS, MS Windows, Linux OS, Android OS, Chrome OS
Edge
Microsoft
MS Windows, MacOS, iOS, Android OS
Opera
Opera Software
MacOS, MS Windows, Linux OS, Android OS

Popular browsers today include Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera. The article also aims to explore how browsers function and their evolution over time.

Overview

Fast-loading websites are crucial for delivering good experiences, as users expect quick and smooth interactions. Two major challenges in web performance are latency and the single-threaded nature of browsers.

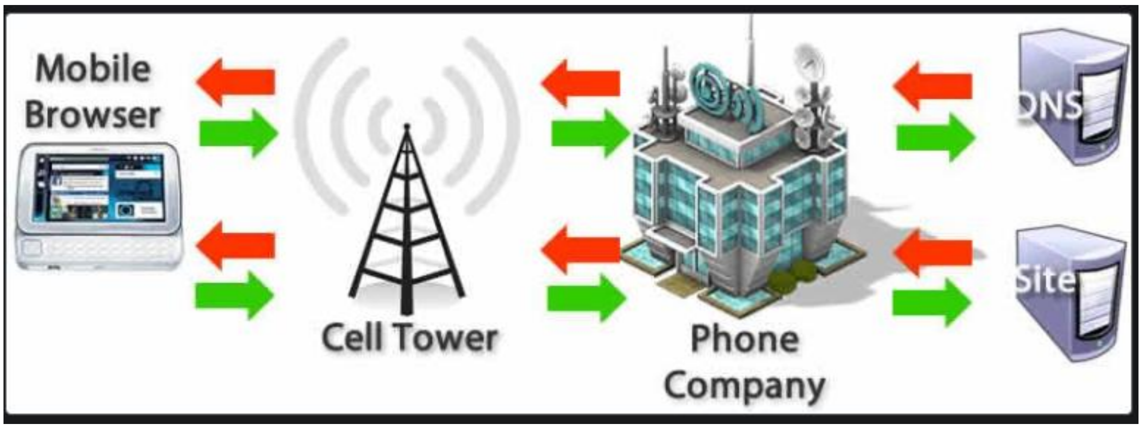
Latency refers to the delay in transmitting data over the network, which affects how quickly a page loads. Developers aim to reduce this to make appear faster.
Single-threaded browsers process tasks one at a time, which can slow down interactions. To improve performance, developers must optimize rendering and minimize the workload on the main thread to ensure smooth scrolling and responsiveness.

Navigation

Navigation stars when a user requests a web page through like entering a URL or clicking a link. Web performance aims to reduce the times process takes, but latency and bandwidth can slow it down.

DNS Lookup

To locate a web page, the browser performs a DNS lookup to translate the domain name into an IP address. This process is usually quick and cached for future use, but each unique hostname on a page requires its own lookup. On mobile networks, DNS lookups can be slower due to additional routing steps, increasing latency.

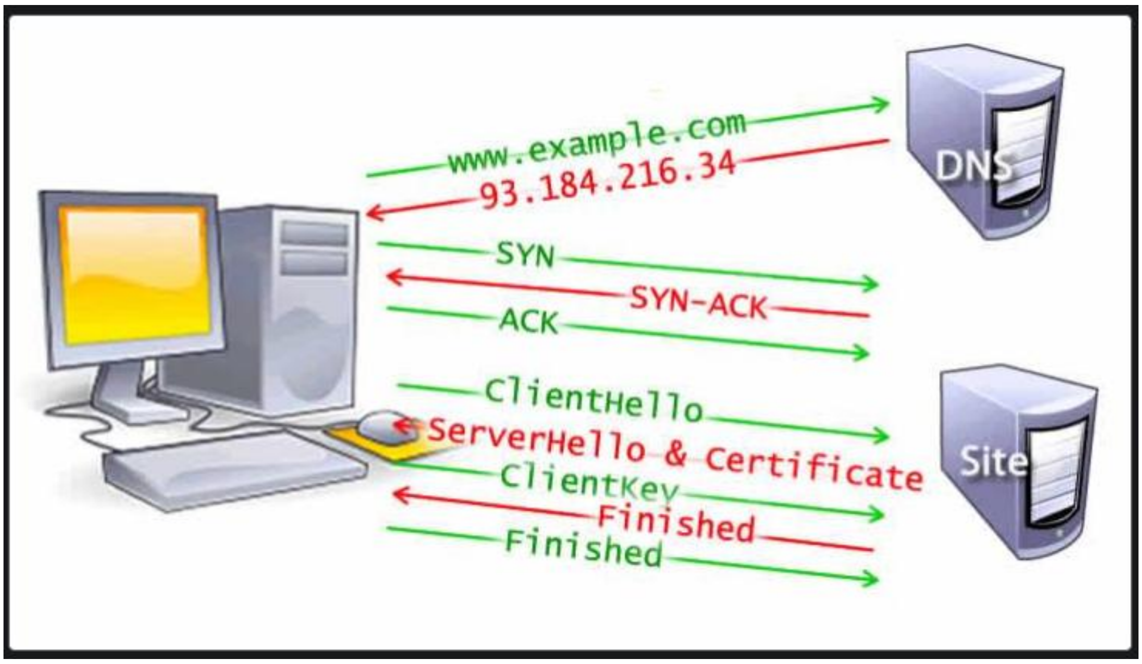


TCP Handshake

After obtaining the IP address, the browser initiates a TCP handshake to establish a connection with the server. This involves three message exchanges (SYN, SYN-ACK, ACK) to negotiate communication parameters before data transfer begins.

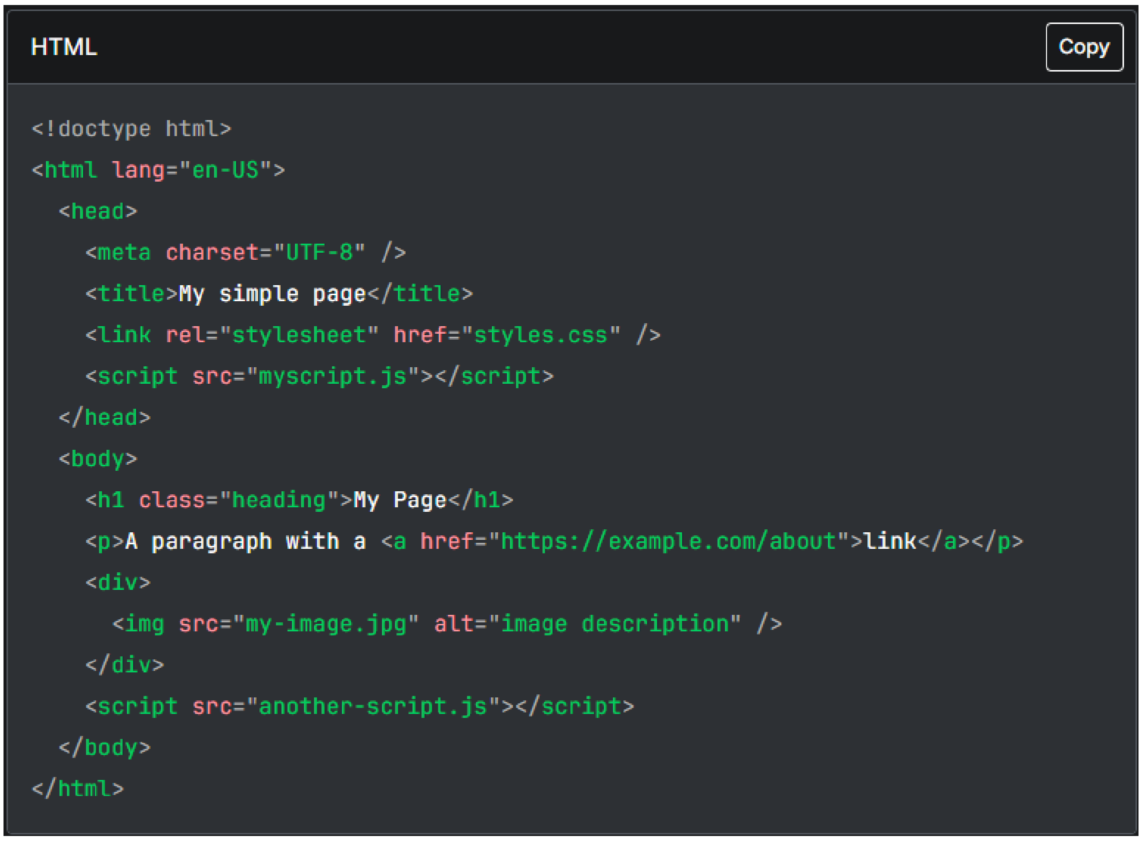
TLS Negotiation

For secure HTTPS connection, a TLS handshake is performed to set up encryption and verify the server. This process involves five additional message exchanges. Although it adds latency, it ensures secure data transmission. Only after completing DNS, TCP, and TLS steps can the browser send the actual content request.



Response

Once a connection to a web servers is established, the browser sends an HTTP GET request, typically for an HTML file. The server responds witch headers and the HTML content. The time to First Byte (TTFB) is the time between the users request and the arrival of the first byte of HTML data, usually around 14KB. Linked resources like CSS, JavaScript, and images are only requested as browser parses the HTML.



```html
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8" />
    <title>My simple page</title>
    <link rel="stylesheet" href="styles.css" />
    <script src="myscript.js"></script>
  </head>
  <body>
    <h1 class="heading">My Page</h1>
    <p>A paragraph with a <a href="https://example.com/about">link</a></p>
    <div>
      <img src="my-image.jpg" alt="image description" />
    </div>
    <script src="another-script.js"></script>
  </body>
</html>
```

Congestion control / TCP slow start

TCP splits data into segments and requires acknowledgement (ACKs) from the client to confirm receipts sending too few segments slows transmission, while sending too many can overwhelm the network. To menage this, TCP uses the slow start algorithm, which gradually increases the amount of data sent based on ACKs received. This process is controlled by the congestion window (CWND), which starts small and doubles with each ack, but halves if ACKs are missing- ensuring efficient and balanced data flow.

Parsing

Once the browser receives the initial data (even just the first 14KB), it begins parsing the HTML to build the DOM (Document Object model) and CSOM (CSS object Model). These structures are essential for rendering the page. Parsing transforms raw HTML, CSS, and JavaScript into Structured object the browser can understand and manipulate.

Building the DOM Tree

The browser tokenizes the HTML and constructs the DOM tree, representing the document's structure. Each HTML element became a node, witch nested elements forming parent-child relationships. A larger number while non-blocking resources like images are fetched in parallel

Preload Scanner

While the main thread parses HTML, the preload scanner scans ahead to identify and request high-priority resources (CSS, JS, Fonts). This allows the browser to begin downloading assets before the parser reaches them, reducing delays. Scripts marked with async or defer help avoid blocking the parser.

Building the CSSOM Tree

The browser processes CSS to build the CSSOM tree, a structure similar to the DOM but for styles. It applies rules hierarchically, starting witch general one and typically not a performance bottleneck.

Other Processes

> JavaScript Compilation: JavaScript is parsed into abstract syntax trees, compiled into bytecode, and interpreted. Most of this happens on the main thread unless using web workers.
>
> Accessibility Tree: The browser builds an accessibility tree (AOM), a semantic version of the DOM used by assistive technologies. It updates alongside the DOM but cannot be modified by external tools.

Render

After parsing, the browser combines the DOM and CSSOM into a render tree. This tree is used to calculate the layout of visible elements and paint them to the screen. Some elements may be promoted to their own layers and rendered via the GPU, improving performance and freeing the main thread.

Style

The render tree is built by traversing visible nodes in the DOM and applying CSS rules from the CSSOM. Elements not meant to be displayed (like <head> or those with display: none) are excluded. Nodes with visibility: hidden are included because they still occupy space. The result is a tree of visible elements with computed styles.

Layout

The browser calculates the size and position of each node in the render tree, considering the viewport and box model properties. This initial calculation is called layout. If content changes later (like an image loading), the layout is recalculated in a process called reflow.

Paint

The browser converts layout boxes into pixels, drawing visual elements like text, borders, and images. To maintain performance, painting is often split into layers. Elements with certain properties (e.g., opacity, transform, will-change) are painted on separate layers, which can be rendered by the GPU.

Compositing

When multiple layers are used, compositing arranges them correctly on the screen. If a reflow occurs (e.g., when an image loads), it may trigger a repaint and re-composite. Defining dimensions for elements can help avoid unnecessary reflows and improve rendering efficiency.

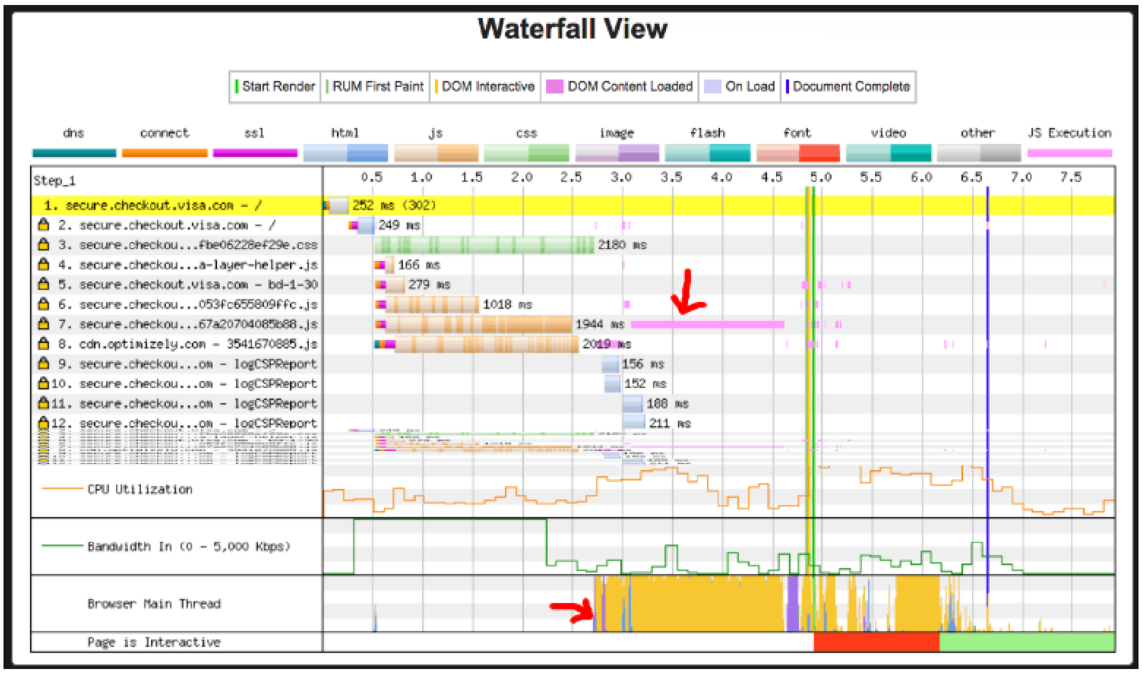
Interactivity

Even after the page is visually painted, it might not be fully interactive. If JavaScript is deferred and runs after the onload event, it can still occupy the main thread, delaying responsiveness to user actions like scrolling or tapping.

Time to Interactive (TTI) measures how long it takes from the initial request to the moment the page responds to user input within 50ms. If the main thread is busy with JavaScript tasks (parsing, compiling, executing), the page may appear loaded but still feel sluggish or unresponsive.

For example, if a large script takes time to download and execute, the user might see the page quickly but experience lag when trying to interact with it. To ensure a smooth experience, it's crucial to avoid blocking the main thread unnecessarily.

In this example, JavaScript execution took over 1.5 seconds, and the main thread was fully occupied that entire time, unresponsive to click events or screen taps.


Waterfall View
Start Render
RUM First Paint
DOM Interactive
DOM Content Loaded
On Load
Document Complete
dns
connect
ssl
html
js
css
image
flash
font
video
other
JS Execution
Step_1
1. secure.checkout.visa.com - / 252 ms (302)
2. secure.checkout.visa.com - / 249 ms
3. secure.checkou...fbe06228ef29e.css 2180 ms
4. secure.checkou...a-layer-helper.js 166 ms
5. secure.checkout.visa.com - bd-1-30 279 ms
6. secure.checkou...053fc655809ffc.js 1018 ms
7. secure.checkou...67a20704085b88.js 1944 ms
8. cdn.optimizely.com - 3541670885.js 2019 ms
9. secure.checkou...om - logCSPReport 156 ms
10. secure.checkou...om - logCSPReport 152 ms
11. secure.checkou...om - logCSPReport 188 ms
12. secure.checkou...om - logCSPReport 211 ms
CPU Utilization
Bandwidth In (0 - 5,000 Kbps)
Browser Main Thread
Page is Interactive