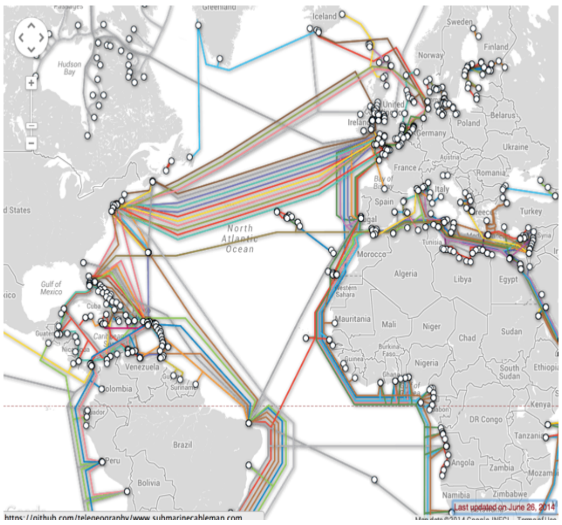
Detailed Report: How the Internet Works – An End-to-End Example of Accessing 'Google.com'.

Abstract: This report provides a comprehensive examination of the intricate processes involved when accessing a website, using "www.google.com" as an example. It delves into each step, from the initial input in the browser to the final rendering of the web page. Key stages include browser actions, DNS resolution, TCP/IP stack operations, HTTP request and response cycles, server processing, data transfer, and browser rendering. Understanding these mechanisms reveals the sophisticated nature of internet communication and highlights the coordinated actions across various layers that enable seamless web interactions.



» What is Internet?

The Internet is a vast global network of interconnected computers and other devices that communicate and exchange data using standardized protocols. It allows for the sharing of information and resources across vast distances, enabling various applications and services.

Keywords: Internet communication , DNS resolution , TCP/IP stack , HTTP request , Browser rendering , Server processing , Data transfer, Web infrastructure , Network routing , Web interaction .

I. **Definitions:**

a) Internet Communication : The exchange of data between devices over a network, allowing users to access information and services online. It encompasses various protocols and technologies that facilitate data transmission.

b) DNS Resolution : The process of converting a human-readable domain name (e.g. www.google.com) into an IP address that computers can use to identify each other on the network.

c) TCP/IP Stack: A suite of communication protocols used to interconnect network devices on the internet. TCP (Transmission Control Protocol) and IP (Internet Protocol) are the core protocols that define how data is transmitted and routed.

d) HTTP Request: A message sent by a client (e.g., a web browser) to a server, requesting specific resources or services. An HTTP GET request is commonly used to fetch web pages.

e) Browser Rendering : The process by which a web browser interprets HTML, CSS, and JavaScript to construct and display a web page visually on the screen.

f) Server Processing : The actions performed by a server upon receiving a request, which may include running scripts, querying databases, and generating content to send back to the client.

g) Data transfer: The movement of data packets between devices across a network. This involves routing through multiple intermediary nodes and reassembling the packets at the destination.

h) Web Infrastructure : The underlying framework that supports internet services and applications. This includes servers, data centres, networking hardware, and software systems that ensure reliable and scalable operations.

i) Network routing: The process of selecting paths in a network along which to send data packets. Routers use algorithms and protocols to determine the most efficient route from source to destination.

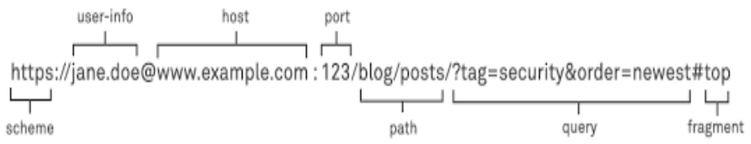
j) Web Interaction : The dynamic exchange between a user and a web application, which can involve navigating pages, submitting forms, and receiving real-time updates. This interaction is facilitated by web technologies and protocols.

II. Literature:

1. Browser Actions

→ **URL Parsing**

When you type "www.google.com" into the browser's address bar and press Enter, the browser parses the URL. This involves breaking down the URL into different components:


user-info
host
port
https://jane.doe@www.example.com:123/blog/posts/?tag=security&order=newest#top
scheme
path
query
fragment

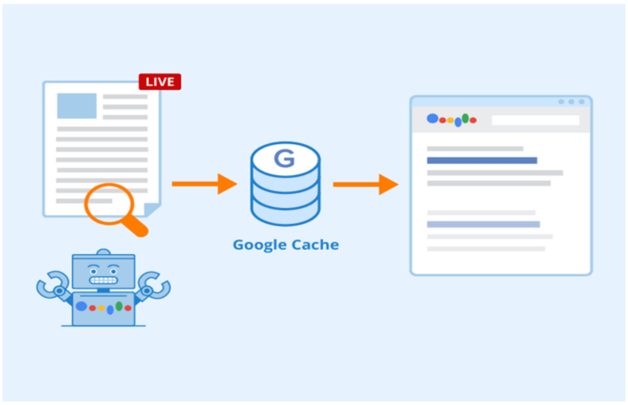
- Scheme: The browser identifies the protocol (HTTP or HTTPS). In this case, it's usually HTTPS, indicating a secure connection.
- Domain Name: The browser extracts "www.google.com" as the domain name.
- Port: Although typically hidden from the user, the browser defaults to port 80 for HTTP and port 443 for HTTPS.
- Path: If any, the path indicates a specific page or resource on the site.
- Query Parameters: These are additional parameters that might be included in the URL to pass data to the server.

→ **Cache Check**

- Cache checking is a process used in computing to determine if a requested piece of data is already stored in a cache memory, thus avoiding the need to retrieve it from a slower storage or compute it again. Thus leading to performance improvement & resource optimization.

- Cache Types: The browser uses multiple caches, such as the disk cache and memory cache. The disk cache is used for long-term storage, while the memory cache is used for quick access to recently accessed resources.

- Cache Validation: The browser may also send a request to the server with headers like If-Modified-Since or If-None-Match to check if the cached content is still valid.

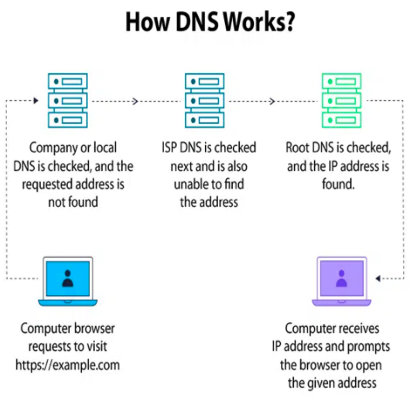


2. DNS Resolution

DNS resolution, also known as Domain Name System resolution, is the process by which a domain name is translated into an IP address (example:192.0.2.1) that computers use to identify each other on a network.

→ **DNS Query**

- Local DNS Cache: The browser first checks the operating system's DNS cache to see if it has recently resolved the domain name.

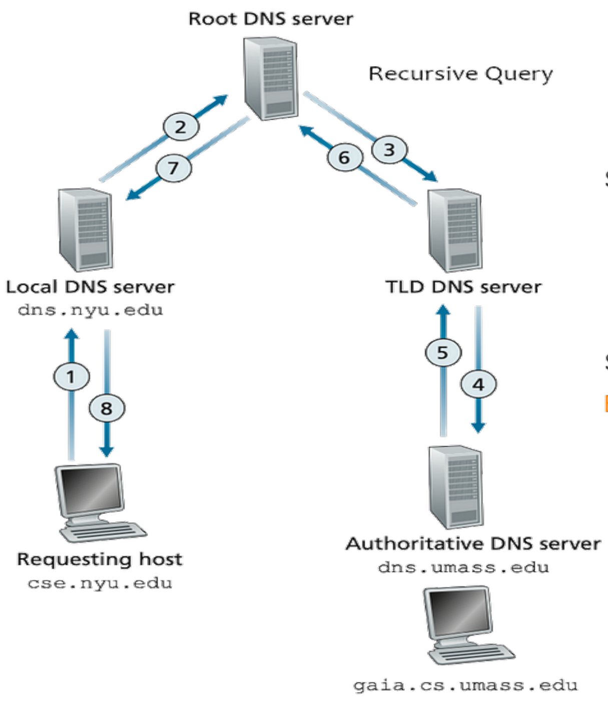

How DNS Works?
Company or local DNS is checked, and the requested address is not found
ISP DNS is checked next and is also unable to find the address
Root DNS is checked, and the IP address is found.
Computer browser requests to visit https://example.com
Computer receives IP address and prompts the browser to open the given address

- DNS Over HTTPS (DoH): Modern browsers may use DNS over HTTPS to encrypt DNS queries, enhancing privacy and security.

→ **Recursive Query**

The DNS resolver performs a recursive query to find the IP address, If the operating system cache also does not have the information, the request is forwarded to a recursive DNS resolver, typically provided by the user's Internet Service Provider (ISP).

- Root DNS Server: There are 13 root



servers worldwide, labelled A through M, which handle queries about TLDs.

- TLD DNS Server: These servers handle queries about domains within their respective top-level domain, such as .com, .net, or .org.
- Authoritative DNS Server: These servers hold the DNS records for specific domains, including A records (IP addresses), CNAME records (aliases), MX records (mail servers), and more.
- Accessing the Website: The browser uses the IP address to establish a connection to the web server hosting the website, and the site is loaded.

- Caching: To optimize future requests, the resolved IP address is cached at various levels (browser, operating system, recursive resolver) according to the time-to-live (TTL) value specified in the DNS records.

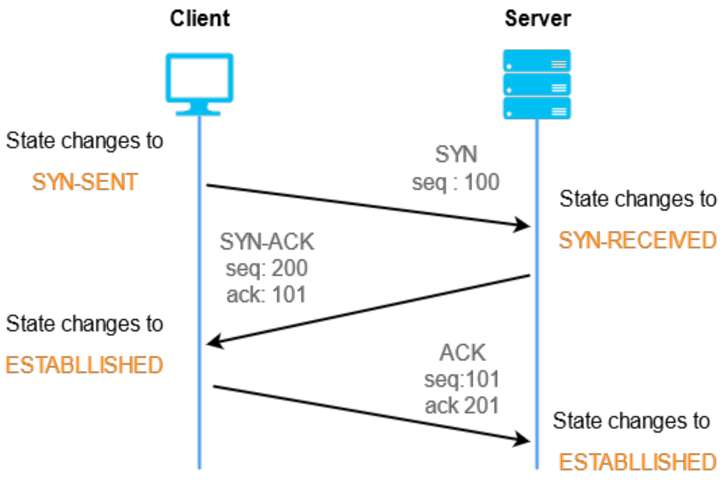
→ **Response**

The DNS resolver returns the IP address to the browser, which uses it to establish a connection with the Google server.

3. TCP/IP Stack Operations

→ **TCP Handshake**

The browser uses the TCP/IP protocol suite to establish a connection:

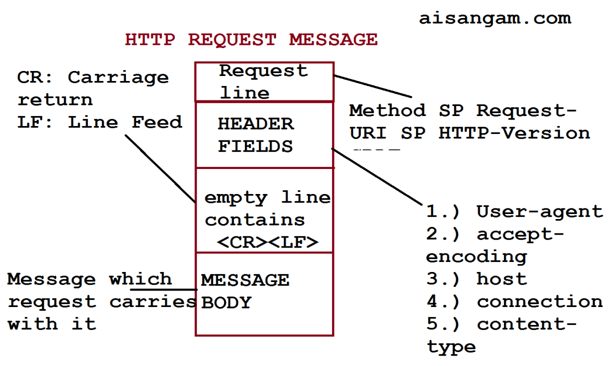


- SYN: The browser sends a SYN (synchronize) packet to the server to initiate the connection.
- SYN-ACK: The server responds with a SYN-ACK (synchronize-acknowledge) packet, acknowledging the request.
- ACK: The browser sends an ACK (acknowledge) packet, establishing a stable connection.
- Sequence Numbers: Each SYN packet includes a sequence number, which is used to track the order of packets and ensure reliable delivery.
- Window Size: The TCP handshake also negotiates the window size, which determines the amount of data that can be sent before receiving an acknowledgment.

4. HTTP Request

→ **Request Headers:** The browser constructs an HTTP GET request for "www.google.com". This request includes headers that provide additional information:

- Host: Specifies the domain name of the server (e.g., "www.google.com").
- User-Agent: Identifies the browser and operating system.
- Accept: Indicates the types of content the browser can handle (e.g., text/html, application/Json).
- Referrer: Indicates the URL of the referring page.



- Connection: Specifies whether the connection should be kept alive or closed after the response.
- Cookies: If the browser has cookies stored for "www.google.com," these are included in the request headers to maintain session state.
- Accept-Encoding: This header specifies the content-encoding methods the client can handle (e.g., gzip, deflate), allowing the server to compress the response.

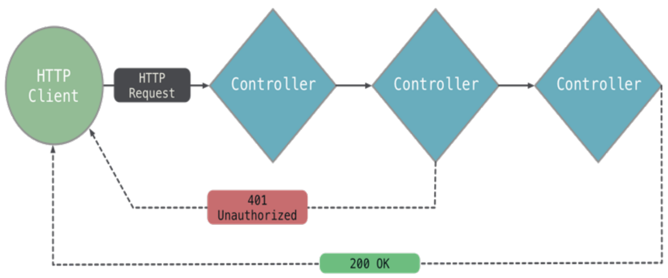
5. Server Processing

Server processing refers to the actions and computations performed by a server in response to client requests. It involves tasks like data retrieval, data storage, computation, content delivery, authentication & authorization, logging & monitoring, network communication & resource management.

→ **Request Handling**

Upon receiving the request, the Google server performs these actions:

- Routing: The server determines which service or application should handle the request.
- Authentication and Authorization: If needed, the server verifies the client's identity and permissions.



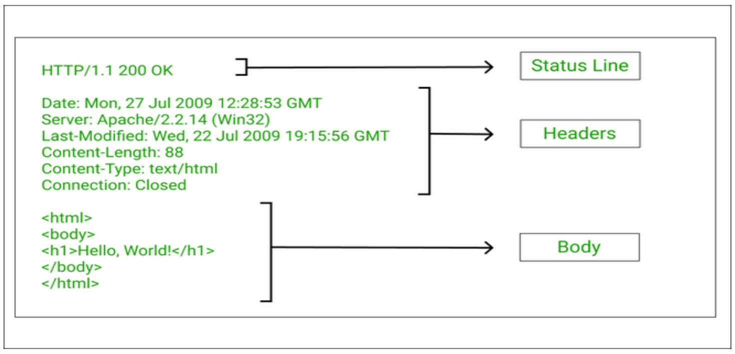
- Data Processing: The server may execute backend scripts, query databases, and perform computations to generate the response.
- Load Balancing: Large websites like Google use load balancers to distribute incoming requests across multiple servers, ensuring high availability and performance.
- Edge Servers: Content delivery networks (CDNs) may serve static content (e.g., images, CSS, JavaScript) from edge servers located close to the user to reduce latency.

→ **Response Generation**

The server generates an HTTP response, which includes:

- Status Line: Indicates the status of the request (e.g., "200 OK" for success).
- Response Headers: Provide metadata about the response (e.g., Content-Type, Content-Length).

- Body: Contains the requested content (e.g., HTML, CSS, JavaScript).



```
HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Status Line
Headers
Body

- Dynamic Content: The server may generate dynamic content using server-side scripting languages (e.g., PHP, Python, Node.js) or frameworks (e.g., Django, Express).
- Database Access: For dynamic pages, the server often queries databases to retrieve or update data. This can involve SQL queries for relational databases or API calls for NoSQL databases.

6. Data Transfer

→ **Packet Transmission**

Packet transmission is the process of sending data across a network in small units called packets. It involves processes like: Data Segmentation, Packet Headers, routing , reassembly & error checking.

- **Data Segmentation**: Large data files are divided into smaller packets for easier and more efficient transmission. Each packet contains a portion of the data, along with control information such as source and destination addresses, and sequencing details. Maximum Transmission Unit is the largest size of a packet or frame, in bytes, that can be sent in a single network transmission. The MTU is an important parameter in networking because it affects the efficiency and performance of data transmission. For Ethernet networks, the standard MTU is 1500 bytes.

- Congestion Control: TCP implements congestion control algorithms (e.g., TCP Reno, TCP Cubic) to prevent network congestion and ensure fair bandwidth distribution.

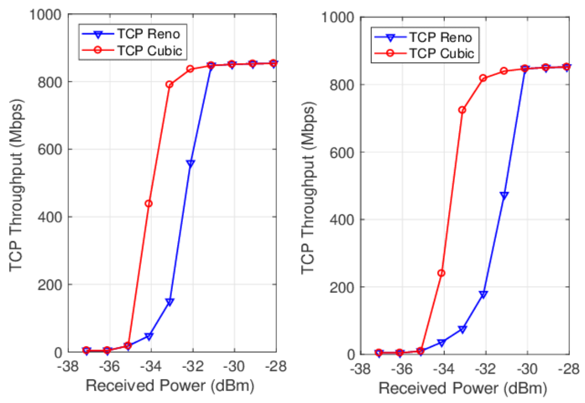


Diagram Interpretation:

- **Low Received Power (-38 dBm to -34 dBm):** Both TCP Reno and TCP Cubic have low throughput.
- **Medium Received Power (-34 dBm to -30 dBm):** TCP Cubic increases throughput faster than TCP Reno, indicating better performance in improving signal conditions.
- **High Received Power (-30 dBm to -28 dBm):** Both algorithms reach maximum throughput, but TCP Cubic achieves this faster and maintains it better than TCP Reno.

  Overall, TCP Cubic demonstrates superior performance by achieving higher throughput more quickly and efficiently as received power improves, making it better suited for environments with fluctuating signal strengths.

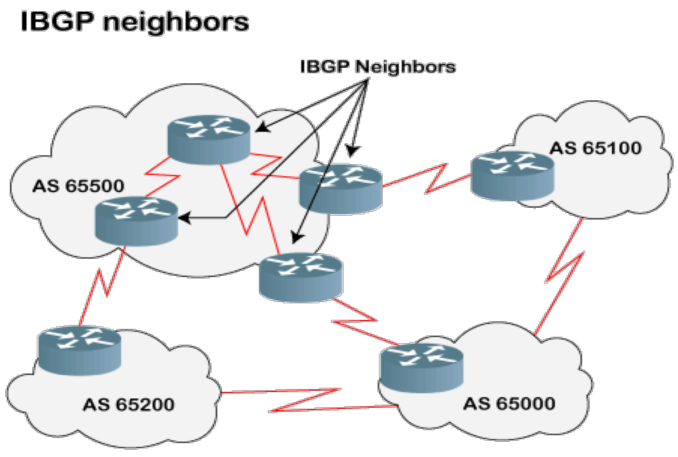
→ **Network Routing**

- Packets are transmitted from the source to the destination through a series of routers and switches. Each router examines the packet header and forwards the packet to the next hop along the path to its destination.

- Routers use algorithms and protocols to determine the most efficient path for each packet from the server to the browser. This path may involve several hops across different networks.

- Routing Protocols: protocols like BGP (Border Gateway Protocol) are used to exchange routing information and determine the best path for data packets. Some pointers are as follows:

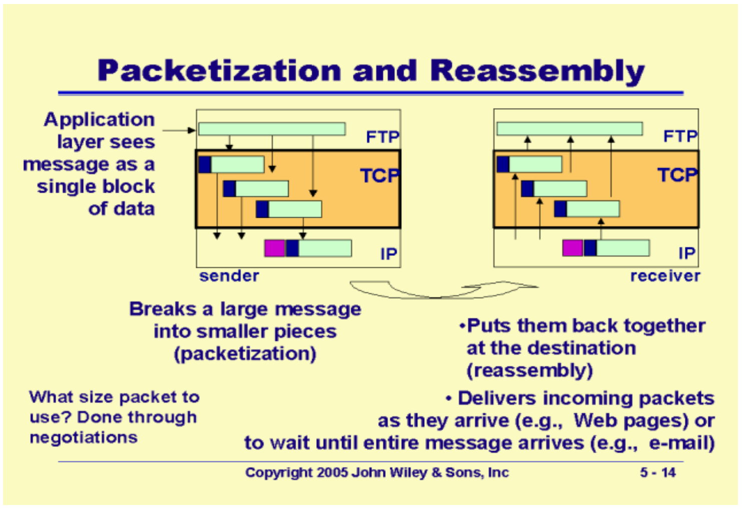
- **Path Vector Protocol**: BGP maintains the path information that gets updated as the network topology changes. This path information is used to make routing decisions based on policies and rule sets rather than just the shortest path.
- **BGP Routing Tables**: BGP routers maintain tables that contain information on available routes, which is used to make forwarding decisions. These tables are constantly updated with new routing information from peer routers.
- **Policy-Based Routing**: Unlike interior gateway protocols that focus on finding the shortest path, BGP is heavily policy-driven. Network administrators can define routing policies that control how BGP selects paths, including preferring certain routes over others.



- Types of BGP: **eBGP (External BGP)**: Used for routing between different autonomous systems. **iBGP (Internal BGP)**: Used for routing within the same autonomous system.

→ **Packet Reassembly**

- Packet reassembly is the process of reconstructing fragmented packets back into their original, complete form at the destination device. This process is essential for ensuring that the transmitted data can be accurately understood and used by the receiving application or system. Key processes in packet reassembly:


Packetization and Reassembly
Application layer sees message as a single block of data
FTP
TCP
IP
sender
FTP
TCP
IP
receiver
Breaks a large message into smaller pieces (packetization)
•Puts them back together at the destination (reassembly)
What size packet to use? Done through negotiations
• Delivers incoming packets as they arrive (e.g., Web pages) or to wait until entire message arrives (e.g., e-mail)
Copyright 2005 John Wiley & Sons, Inc
5 - 14

a. Fragmentation:
- Occurs when packets exceed the Maximum Transmission Unit (MTU) size of a network segment and need to be broken down into smaller fragments for transmission.
- Each fragment is transmitted separately, with necessary headers to ensure correct routing and reassembly.

b. Headers and Identifiers:
- Each fragmented packet includes a header with information such as the source and destination IP addresses, a unique identifier for the fragmented packet, and offset values indicating the position of the fragment in the original packet.
- The More Fragments (MF) flag is set in all but the last fragment to indicate that more fragments are to come.

c. Reassembly Process:
- At the destination, the IP layer receives the fragments and uses the header information to reassemble them in the correct order.
- The unique identifier and offset values ensure that fragments are correctly ordered and placed.
- The reassembly process waits until all fragments have arrived, indicated by the absence of the MF flag in the final fragment.

d. Error Handling:
- If any fragment is lost or corrupted, the entire packet may need to be retransmitted since reassembly cannot complete with missing pieces.

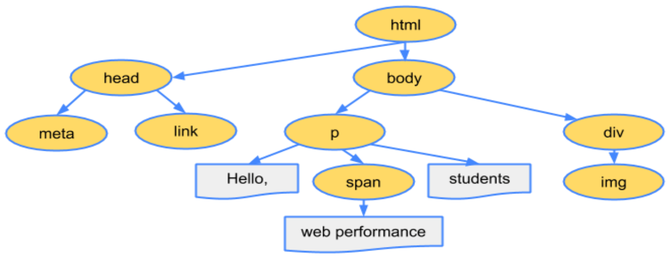
- Error-checking mechanisms, such as checksums in the packet headers, help detect and handle such issues.

- Importance of Packet Reassembly:
  - **Data Integrity**: Ensures that the data received is complete and accurate, preserving the integrity of the transmitted information.
  - **Efficient Communication**: Allows for efficient utilization of network resources by enabling the transmission of large data packets over networks with smaller MTU sizes.
  - **Reliability**: Supports reliable communication, especially in networks where fragmentation is common due to varying MTU sizes across different segments.

7. Browser Rendering

→ Parsing HTML

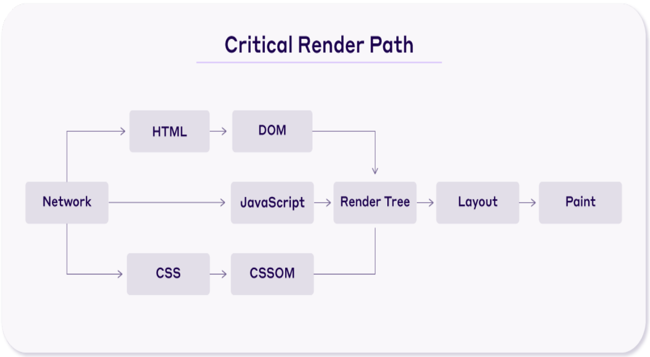
- The browser parses the HTML content to build the Document Object Model (DOM). The DOM represents the structure of the web page as a tree of nodes.



- DOM Construction: The browser builds the DOM tree by parsing the HTML markup. Each element and attribute becomes a node in the tree.

→ Loading Resources

- The browser identifies additional resources specified in the HTML (e.g., CSS, JavaScript, images) and sends separate HTTP requests to fetch them.



- Resource Prioritization: The browser prioritizes loading resources that are critical for rendering the above-the-fold content, such as CSS and critical JavaScript files.

→ Applying CSS

- The browser applies CSS styles to the DOM, determining the layout and appearance of elements.
- CSSOM Construction: The browser constructs the CSS Object Model (CSSOM), a tree-like structure that represents the styles for each element in the DOM.

→ Executing JavaScript

- The browser executes JavaScript code, which can modify the DOM, handle user interactions, and dynamically load additional content.
- JavaScript Engines: Browsers use JavaScript engines (e.g., V8 for Chrome, Spider Monkey for Firefox) to compile and execute JavaScript code.

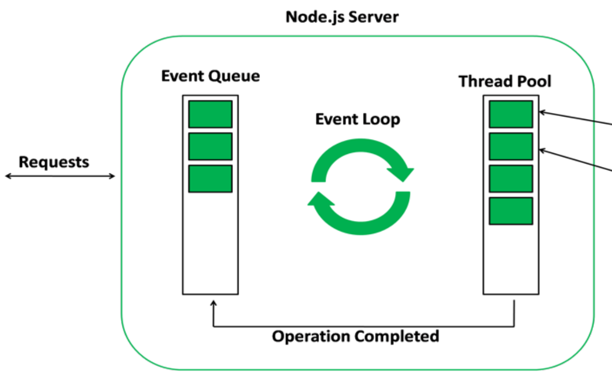
→ Layout and Paint

- The browser calculates the layout of elements based on the CSS and renders the visual representation of the web page on the screen. This involves:
  - Layout: Calculating the position and size of elements.
  - Paint: Drawing pixels on the screen to display the content.
  - Reflow: The browser calculates the layout of elements. Reflow can be an expensive operation if it involves many elements or nested structures.
  - Compositing: The browser uses a compositing process to combine layers and render the final image on the screen. Modern browsers use GPU acceleration to speed up this process.

8. Interactivity

→ Event Handling

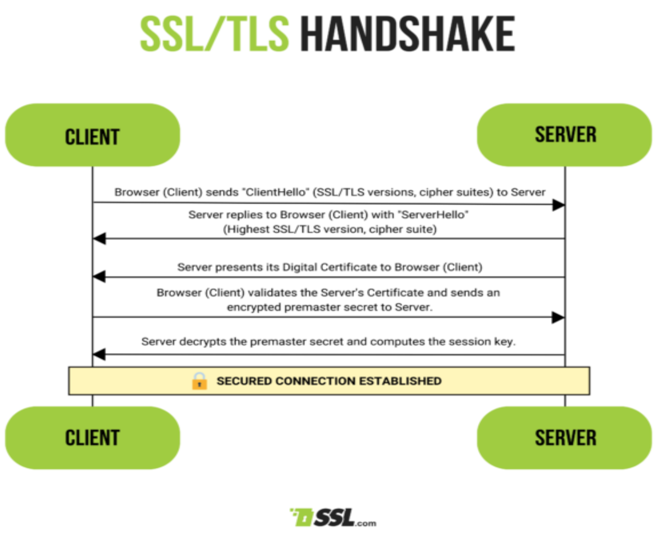
- The browser listens for user interactions (e.g., clicks, keystrokes) and handles events accordingly. JavaScript event listeners can modify the DOM, send additional HTTP requests (e.g., via AJAX), and update the page without requiring a full reload.



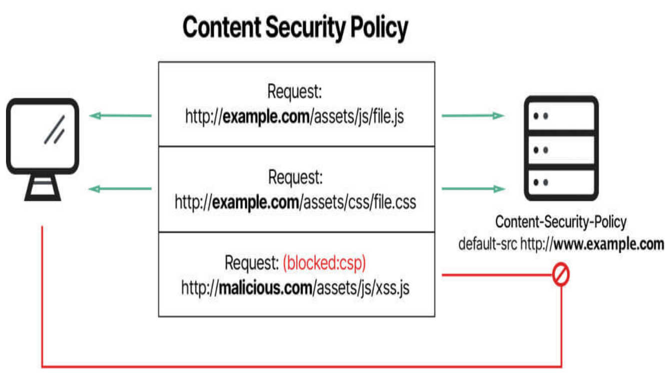
- Event Loop: JavaScript uses an event loop to handle asynchronous operations, such as user interactions and network requests.
- AJAX: Asynchronous JavaScript and XML (AJAX) allows the browser to send HTTP requests and update the page dynamically without reloading it.

9. Security

- SSL/TLS Handshake: For HTTPS connections, the browser and server perform an SSL/TLS handshake to establish an encrypted connection. This involves exchanging certificates and keys.


SSL/TLS HANDSHAKE
CLIENT
SERVER
Browser (Client) sends "ClientHello" (SSL/TLS versions, cipher suites) to Server
Server replies to Browser (Client) with "ServerHello" (Highest SSL/TLS version, cipher suite)
Server presents its Digital Certificate to Browser (Client)
Browser (Client) validates the Server's Certificate and sends an encrypted premaster secret to Server.
Server decrypts the premaster secret and computes the session key.
SECURED CONNECTION ESTABLISHED
CLIENT
SERVER
SSL.com

- Security Headers: The server may include security headers (e.g., Content-Security-Policy, X-Frame-Options) to protect against common vulnerabilities.

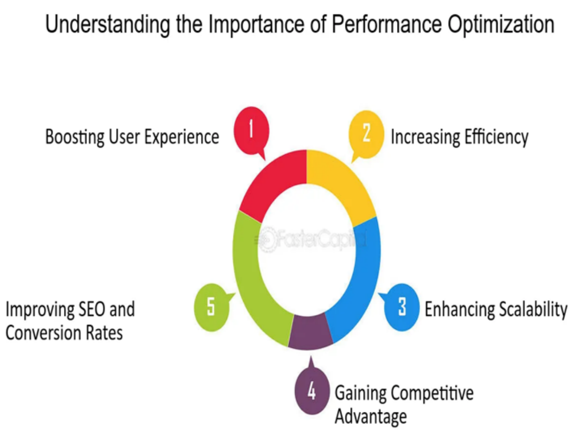

Content Security Policy
Request:
http://example.com/assets/js/file.js
Request:
http://example.com/assets/css/file.css
Request: (blocked:csp)
http://malicious.com/assets/js/xss.js
Content-Security-Policy
default-src http://www.example.com

- **Content-Security-Policy (CSP)** is a security feature that helps prevent various types of attacks, including Cross-Site Scripting (XSS), data injection attacks, and other code execution vulnerabilities. It does so by restricting the sources of content that can be loaded on a web page.

- **X-Frame-Options** is an HTTP response header that provides a simple and effective way to prevent clickjacking attacks by controlling whether a web page can be displayed within an iframe or frame. Clickjacking is a type of attack where a malicious site tricks users into clicking on something different from what they perceive, potentially leading to unauthorized actions or data exposure.

10. Performance Optimization

- Minification: Servers often minify HTML, CSS, and JavaScript to reduce file sizes and improve load times.


Understanding the Importance of Performance Optimization
Boosting User Experience
1
2
Increasing Efficiency
Improving SEO and Conversion Rates
5
3
Enhancing Scalability
4
Gaining Competitive Advantage

- Lazy Loading: The browser can delay the loading of non-critical resources (e.g., images below the fold) until they are needed.

III. References:

- **Google**
- **YouTube**
- **GeekforGeeks**
- **JavaTpoint**