REPORT

Q. What happens when you type google.com in your browser and hit enter?

Contents

Introduction:

When we type"google.com" and press enter then it initiates a series of actions and processes, involving multiple layers of technology and communication protocols. This detailed analysis into each step, covering everything from the initial key press to the final rendering of the Google's homepage.

1. Key Press Events

"g" Key is Pressed

- When a key is pressed then the series of steps and working of internal component are involved in getting that letter to be displayed.
- On the very first key stroke, the computer starts looking for history in the cache and other places and autocompletes the word typed if found.

The "Enter" Key is Pressed

- **Physical Action:** We press the "Enter" key.
- **Electrical Signal:** An electrical circuit specific to the "Enter" key is closed, sending a signal.
- **Debouncing:** The signal is debounced by the keyboard controller.
- **Key Code Generation:** The controller generates a keycode (usually 13 for Enter) and sends it tothe computer.
- **OS Handling:** The OS processes this keycode and sends it to the active application.

2. URL Parsing

Determining the

Input

- **Input Evaluation:** The browser examines the entered text to determine whether it is a valid URLor a search term.
    - **URL:** If the input contains a protocol (e.g., "http://", "https://"), a domain name, or otherURL-specific components, it is treated as a URL.
    - **Search Term:** If the input doesn't meet URL criteria, it is treated as a search query andsent to the default search engine (e.g., Google, Bing).

Handling Unicode

- **Internationalized Domain Names (IDN):** If the URL contains non-ASCII characters, the browserconverts it to Punycode, ensuring compatibility with the DNS system.

3. HTTP or HTTPS?

HSTS (HTTP Strict Transport Security) Check

- **Preloaded HSTS List:** The browser checks if "google.com" is in its preloaded HSTS list.
    - **HSTS Policy:** If present, the browser enforces HTTPS for all connections to this domain.
    - **Default Protocol:** If not in the HSTS list, the browser may initially try HTTP and then beredirected to HTTPS by the server.

DNS ResolutionChecking Cache

- **Local DNS Cache:** The browser first checks its local cache for the IP address of "google.com" toreduce latency.

DNS Query

- **Hosts File:** If the address is not cached, the browser checks the local hosts file for a manual entryof the IP address.
- **DNS Server:** If not found in the hosts file, the browser sends a DNS query to the configured DNSresolver (often provided by our ISP or manually set).

DNS Query Process

- **Recursive DNS Query:**
    - **Root Servers:** The DNS resolver queries the root DNS servers to find the authoritativeservers for the top-level domain (TLD) ".com".
    - **TLD Servers:** The root servers respond with the addresses of the TLD servers for ".com".
    - **Authoritative DNS Servers:** The DNS resolver queries the TLD servers, which respondwith the authoritative DNS servers for "google.com".
    - **Final Query:** The resolver queries the authoritative servers to get the IP address for"google.com".

ARP (Address Resolution Protocol)

- **MAC Address Resolution:** If the DNS resolver is on a different subnet, an ARP request is sent todetermine the MAC address of the router or gateway.

DNS Response

- **IP Address:** The DNS resolver returns the IP address (e.g., 8.8.8.8) of "google.com" to the browser.

4. Establishing Connection

Opening a Socket

- **TCP Socket:** The browser requests a TCP socket connection to the IP address and appropriateport (443 for HTTPS, 80 for HTTP).

TCP Handshake

- **Three-Way Handshake:**
    - **SYN Packet:**
        - The client (browser) sends a SYN packet to the google server.
        - The packet includes an initial sequence number (ISN), a randomly chosennumber serving as the starting point for the sequence of bytes.
        - The SYN flag in the TCP header is set to indicate that this packet is for initiating aconnection.
    - **SYN-ACK Packet:**
        - The Google server responds with a SYN-ACK packet upon receiving the SYNpacket.
        - This packet acknowledges the client's SYN packet with an acknowledgmentnumber one more than the client's sequence number.
        - It synchronizes the server's sequence numbers with the client by including theserver's own initial sequence number.
        - Both the SYN and ACK flags in the TCP header are set.
    - **ACK Packet:**
        - The client responds to the server’s SYN-ACK with an ACK packet.
        - This packet acknowledges the server’s SYN-ACK by setting the acknowledgmentnumber to one more than the server's sequence number.
        - The ACK flag is set in the TCP header, and the SYN flag is not set, indicating thatthe connection is now established.

TLS Handshake (for HTTPS)

- **ClientHello:** Client sends a ClientHello message to the server, specifying supported TLS versions and cipher suites.
- **ServerHello:** The server responds with a ServerHello message, selecting the TLS version andcipher suite, and sends its digital certificate.
- **Certificate Verification:** Client verifies the server's certificate using a chain of trust up to a trustedCertificate Authority (CA).
- **Key Exchange:** Browser and the server exchange keys to establish a secure session.
- **Session Establishment:** Both parties agree on session keys, completing the TLS handshake.

5. Sending the HTTP

Request Crafting

- **HTTP GET Request:** We construct an HTTP GET request for "google.com".

Sending the Request

- **Packet Segmentation:** The HTTP request is broken into TCP segments and sent to the server.

6. Server-Side

ProcessingReceiving

the Request

- **HTTP Server:** The server receives the request through its HTTP server software (e.g., Nginx,Apache).
- **Virtual Host:** The server determines the appropriate virtual host (google.com) based on the"Host" header.

Request Handling

- **Method Verification:** The server checks that the HTTP method (GET) is supported.
- **Security Checks:** The server performs security checks, such as IP authorization, rate limiting, and access control.
- **Rewriting Rules:** The server applies URL rewriting rules if configured (e.g., redirecting HTTP toHTTPS).

Content Retrieval

- **Static Content:** The server retrieves the requested static file (e.g., index.html) from the filesystem.
- **Dynamic Content:** For dynamic content, the server processes scripts, or applications (e.g., PHP,Python, Node.js) to generate the HTML response.

7. Sending the Response

Crafting the Response

- **HTTP Response:** The server constructs an HTTP response with the requested content.

Transporting the Response

- **TCP Segmentation:** The response is segmented into TCP packets and sent to browser.
- **ACKs:** Client acknowledges each received segment to ensure reliable delivery.

8. Rendering the Page

Receiving the

Response

- **Reassembly:** Client reassemble the TCP segments to form the complete HTTP response.

Parsing and Rendering

- **HTML Parsing:** The browser parses the HTML content to construct the Document Object Model(DOM).
- **CSS Parsing:** The browser fetches and parses CSS files to construct the CSS Object Model(CSSOM).
- **JavaScript Execution:** The browser fetches and executes JavaScript files.
- **Render Tree:** The browser constructs the render tree by combining the DOM and CSSOM.
- **Layout:** The browser calculates the layout, determining the position and size of elements on thepage.
- **Painting:** The browser paints the pixels on the screen based on the layout.

9. Additional Requests

Resource Loading

- **Embedded Resources:** The HTML may reference additional resources (e.g., images, CSS, JS).
    - **Concurrent Requests:** The browser makes concurrent HTTP requests to fetch these resources.
- **Resource Parsing:** The browser parses and processes these resources as they are received.

Caching

- **Browser Cache:** The browser caches resources (e.g., images, CSS, JS) for future use to reduceload times and bandwidth usage.
- **Cache-Control Headers:** The server may provide cache-control headers to instruct the browseron how to cache resources.

Summary

When we type "google.com" and press Enter, a series of events unfold involving hardware interactions, software processes, network communications, and security protocols. From the initial keypress to the final rendering of the Google homepage, this process highlights the complexity and efficiency of modern web browsing, involving multiple systems and layers working seamlessly together.

References:

1) GitHub - alex/what-happens-when: An attempt to answer the age old interview question "What happens when you type google.com into your browser and press enter?"
2) https://chatgpt.com/
3) https://en.wikipedia.org/wiki/Punycode