**HOW DOES THE INTERNET WORKS WHAT HAPPENS WHEN YOU TYPE A URL**

When you type a URL like "[https://www.google.com](https://www.google.com/)" into a web browser and hit your enter key, there are a lot of things that go on before you finally get some output on your browser.

Before I take the individual steps involved and explain them in detail, let me give you a general overview of everything that goes on within those few microseconds.

1. Your computer sends a request to the domain name system (DNS) server which serves as an address book for all domain names. This then sends back the exact IP address of the server which [https://www.google.com](https://www.google.com/) points to.
2. Knowing this IP, your computer then establishes a connection with the server through the IP address. The type of this connection is known as Transmission Control Protocol (TCP) and your computer is able to establish this connection through the Internet Protocol (IP). This whole process is known as a "handshake".
3. If your computer is behind a firewall, the firewall checks to ensure that the particular request you are making is allowed before permitting it. Also, if the server you are trying to access is also behind a firewall, a similar check will be done before you are finally able to connect to the server.
4. After establishing the connection, your browser now sends a request for the web page using an encryption protocol like Secure Sockets Layer (SSL) or Transport Layer Security (TLS) in order to encrypt the data that will be shared between your computer and the server. This type of encryption is what is responsible for the "s" in "**https**" which also implies that the connection is secure.
5. Companies like Google with high traffic maintain a host of servers and for that matter, they have a load balancer that receives most of the requests and sends them to a particular server. The request from your browser will therefore hit the load balancer first which will forward it to a specific server depending on the algorithm used by the load balancer.
6. The server that receives the request then sends a response back to the load balancer which also forwards the response back to your browser. This response will mostly include HTML, CSS, and JavaScript files that makeup Google's homepage.
7. The HTML files returned to tell the browser how to render the content of the page. The CSS file tells the browser how to style the content while the JavaScript file adds interactivity to the page.
8. If there is a need for some dynamic content such as Google search results, then the web server will make a request to the application server, which in turn may make a request to a database server to get some data and send it back to the web server. The web server will then include these in the response it sends back to the browser.
9. Finally, the browser will render the page and display it to you.

**DNS Request**

Anytime you use your browser to access any website or domain name, the browser stores information concerning that domain name (DNS record) in its cache.

So, anytime you type a domain name like "[google.com](http://google.com/)" into your web browser, the browser first checks its cache to see if it has a recent copy of the DNS record for that domain.

If there is a recent copy of the DNS records for that domain, it will use the IP address in the cache to send a request to the server. This speeds up the process of resolving the domain name to an IP address because it avoids the need to send a request to the DNS server.

If the browser cache does not contain a recent copy of the DNS record, or if the DNS record has changed since the last time it was cached, the browser will send a request to the **DNS server** to resolve the domain name to an IP address.

This is also a whole process on its own, so let's explore the various steps involved.

**DNS lookup process**

Here's how the DNS lookup process works in more detail:

1. The browser sends a request to the local DNS resolver, which is usually provided by the internet service provider (ISP).
2. The local DNS resolver checks its cache to see if it has a recent copy of the DNS record for the domain. If it does, it sends the IP address back to the browser.
3. If the local DNS resolver does not have a recent copy of the DNS record, it sends a request to a root name server.
4. The root name-server responds with the address of a top-level domain (TLD) name-server, such as .com or .org.
5. The local DNS resolver sends a request to the TLD name server.
6. The TLD name server responds with the address of the authoritative name server for the domain.
7. The local DNS resolver sends a request to the authoritative name server.
8. The authoritative name server responds with the IP address for the domain.
9. The local DNS resolver sends the IP address back to the browser.
10. The browser sends a request to the server at the IP address to retrieve the web page.

This process may involve additional steps if the DNS record is not found at any of the name servers, or if the DNS record has been configured to use a service such as DNS load balancing or content delivery networks (CDN).

Once the IP address has been resolved, it is cached by the local DNS resolver and the browser so that future requests for the same domain name can be resolved more quickly.

The length of time that the DNS record is cached (the "TTL" or Time To Live) is determined by the authoritative name server and can be configured by the domain owner.

**TCP/IP connection**

TCP (Transmission Control Protocol) and IP (Internet Protocol) are two of the main protocols that make up the Internet.

They work together to establish a connection between a client and a server and facilitate the transmission of data between them.

When you enter "[google.com](http://google.com/)" into a web browser, the browser uses TCP/IP to establish a connection with the server that hosts the website.

Here's what happens in more detail:

1. The browser sends a request to the server using IP to establish a connection.
2. The server receives the request and sends back a message acknowledging the request to establish a connection. This is the handshake process.
3. Once the handshake is complete, the browser can send a request for the web page it wants to access (in this case, the homepage of [google.com](http://google.com/)). This request is sent using TCP, which ensures that the request is transmitted reliably and in the correct order.
4. The server receives the request and sends back the HTML code for the homepage of [google.com](http://google.com/) to the browser. This response is also sent using TCP to ensure reliable transmission.
5. The browser receives the HTML code and uses it to render the web page on your screen. Any resources (such as images) that the web page needs are also requested and received using TCP/IP.

**Firewall**

A firewall is a security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules. Its primary purpose is to protect a network from external threats, such as hackers and malware.

When you type a URL like "[google.com](https://www.google.com/)" into your browser, the request that your browser makes to Google's server passes through the firewall on its way. The firewall checks the incoming request to make sure it is allowed based on its security rules.

There are two main types of security rules that a firewall uses to check incoming requests:

1. Rules that allow or block **traffic based on the source and destination** of the request. For example, a firewall may be configured to block all traffic from certain countries or to allow only certain IP addresses to access the network.
2. Rules that allow or block **traffic based on the type of traffic**. For example, a firewall may be configured to block all traffic on certain ports (such as those used by malware) or to allow only certain types of traffic (such as HTTP or HTTPS).

If the incoming request meets the security rules set by the firewall in front of Google's server, it is allowed through, and the browser is able to access the website.

However, if the request does not meet the security rules, it is blocked, and the browser is unable to access the website.

**HTTPS/SSL**

HTTPS (Hypertext Transfer Protocol Secure) is a secure version of the HTTP protocol used to transmit data on the Internet. It is used to encrypt the data transmitted between your browser and Google's server.

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are encryption protocols that are used to secure the data transmitted over HTTPS.

When your browser establishes a connection with Google's server using HTTPS, your browser and Google's server first agree on the version of SSL/TLS to use and then create a secure, encrypted channel for transmitting the data.

Let me use an analogy to explain what is going on here.

HTTPS is like a locked box that is used to send messages over the internet. When you want to send a message using HTTPS, you put the message in the locked box and send it to the person you want to receive the message. Only the person you are sending the message to has the key to unlock the box and read the message.

SSL/TLS are like special codes that are used to lock and unlock the box. When you want to send a message using HTTPS, you and the person you are sending the message to agree on the code to use to lock and unlock the box. This way, only you and the person you are sending the message to know the code and can read the message.

When you type "[**google.com**](https://www.google.com/)" into your browser, the browser is like the person sending the message. The server that hosts [google.com](http://google.com/) is like the person receiving the message. The browser sends a request for the web page using HTTPS, which is like putting the request in a locked box and sending it to the server. The server then sends the web page back to the browser using HTTPS, which is like putting the web page in a locked box and sending it back to the browser.

**Load balancer**

A load balancer is a device that distributes incoming network traffic across a group of servers or resources.

Its primary function is to ensure that the traffic is distributed evenly across the servers in order to avoid overloading any single server and to increase the overall capacity and reliability of the system.

A company like Google, which receives billions of website visitors a day, will need a lot of servers to serve all these users. Therefore, there will be a need for them to set up a load balancer to ensure that some of the servers are not overburdened while others are being underutilized.

In the case of a browser trying to access [google.com](http://google.com/), the load balancer would receive the incoming request from the browser and then forward it to one of the servers in the Google server network. The particular server chosen will depend on the type of load-balancing algorithm implemented.

**Web server**

A web server is a computer program that is responsible for handling requests for web pages from clients (such as a browser trying to access [google.com](http://google.com/)). When a client sends a request for a web page to a web server, the server processes the **request** and returns the appropriate **response** to the client.

This means that when trying to access [google.com](http://google.com/), Google's server will receive a request from the load balancer.

The web server would then process the request and generate a response, typically including the HTML, CSS, and JavaScript files that make up the web page.

The web server would then send this response back to the load balancer, which would forward it to the browser. The browser would then use the HTML, CSS, and JavaScript files to render the web page for the user.

**Application server and database (if applicable)**

Unlike the web server, the application server handles dynamic content. When using "[**google.com**](http://google.com/)", the application server will be responsible for generating the search results (which change based on the query you put into the search engine).

When you submit a search query to Google, the request is first sent to the load balancer, which forwards it to one of the web servers in the Google server network. The web server then sends the request to the application server, which processes the request and generates the search results.

Depending on the complexity of the search query, the application server may need to make a request to a **database** in order to retrieve the necessary data.

For example, if you are searching for a specific product on an e-commerce website, the application server may need to retrieve information about the product from a database.

Once the application server has obtained the necessary data, it sends it back to the web server, which includes it in the response that is sent back to the browser. The browser then uses this information to display the search results to you.

**Rendering the page**

When a browser receives a response from a web server, it processes the HTML, CSS, and JavaScript files that are included in the response in order to render the web page.

The rendering process involves interpreting the HTML and CSS code, rendering any images or other media that are included on the page, and executing any JavaScript code that is present on the page.

In your case, your browser would receive the response from the web server, which includes the HTML, CSS, and JavaScript files that make up the Google web page.

The browser would then use these files to render the page and display it to you. This process typically involves the following:

* displaying the text and images on the page in the appropriate positions
* formatting the text and layout according to the CSS styles
* executing any JavaScript code that is present on the page

Once the page has been fully rendered, you can now interact with it by clicking links, entering text, or interacting with other elements on the page.