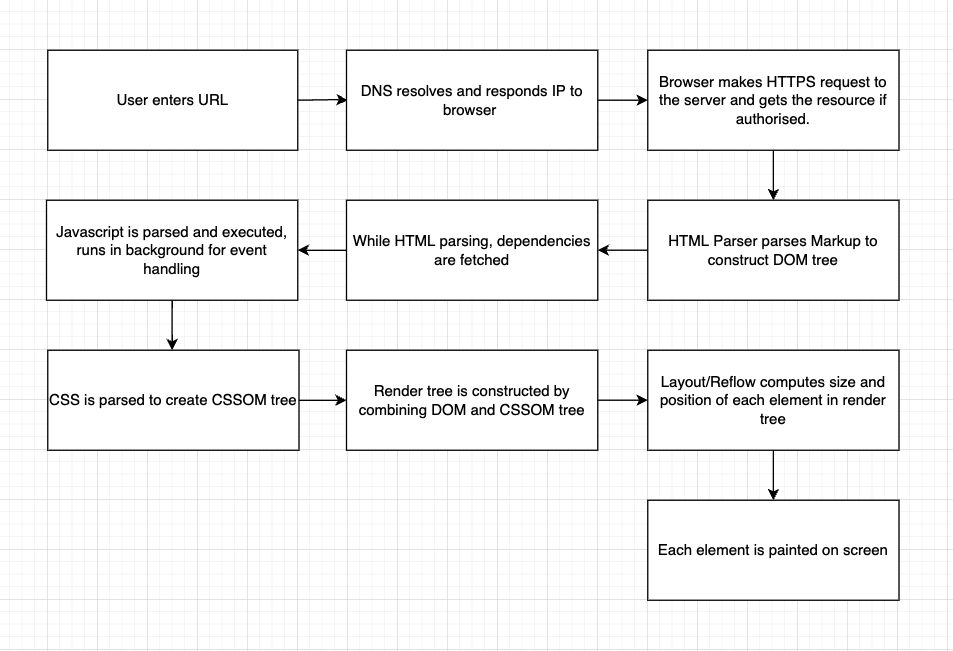
**What happens when user enter the URL in the browser:**



When user enter any URL in the browser, one of the main concepts of networking is used called DNS (Domain Name System) which lets you connect to the requested web page or a resource like an image by providing its IP address. As under the hood IP addresses are used for communication between client and server in TCP/IP which is common Internet communication protocol, browser cannot use example.com to communicate with the google servers. It must know the IP address of example.com before sending any request.

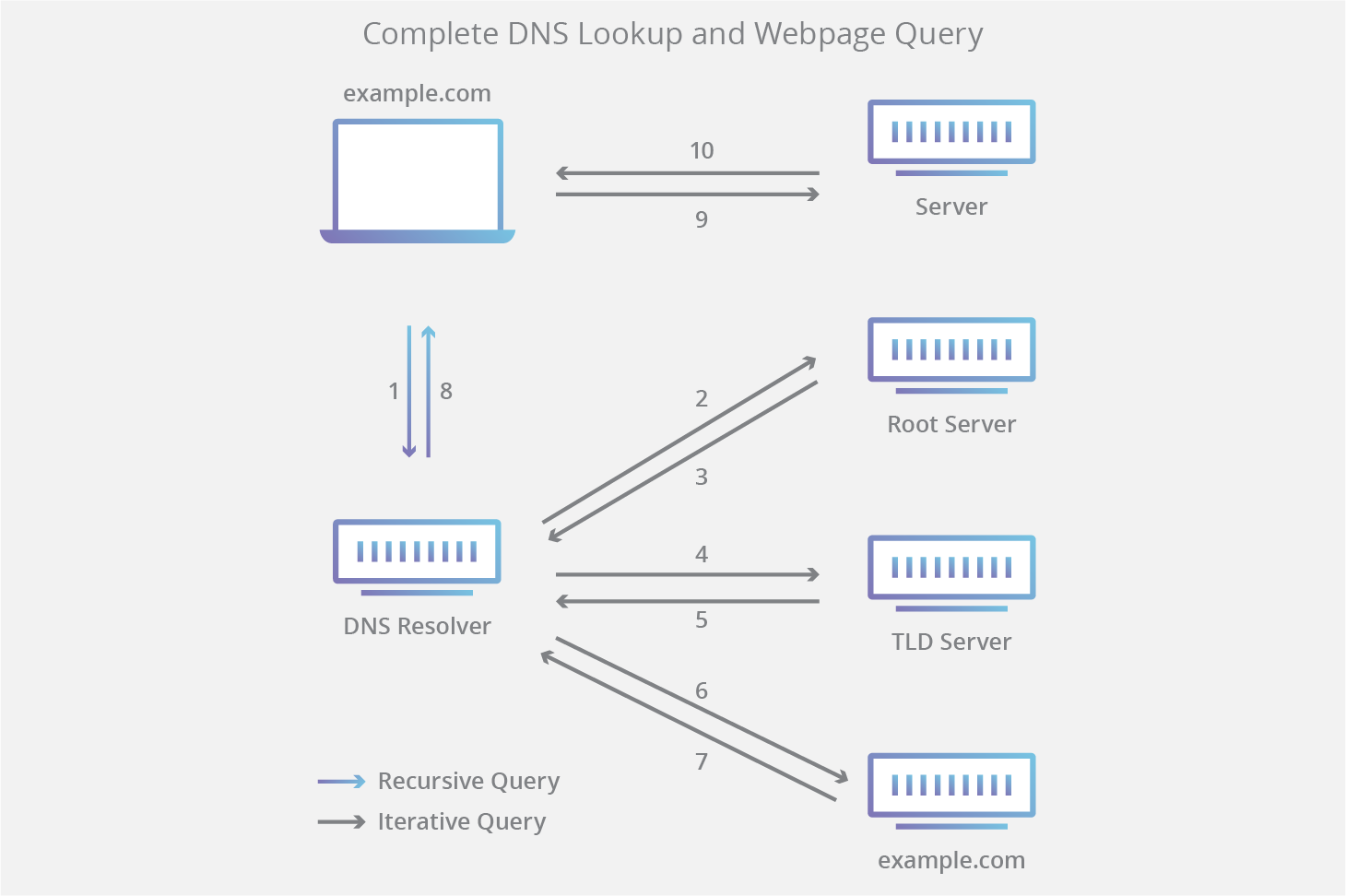
**Following steps will illustrate How DNS works:**

1. When an URL is entered, first that DNS query is sent to the DNS resolver.
2. The resolver then queries a DNS root nameserver which can be considered as the index book for the IPs
3. Then the root server checks with Top Level Domain (TLD) DNS server (such as .com or .net), which stores the information for its domains. When searching for example.com, our request is pointed toward the .com TLD.
4. The DNS resolver then makes a request to the .com TLD.
5. The TLD server then responds with the IP address of the domain’s nameserver, example.com.
6. Then resolver sends a query to the domain’s nameserver.
7. The IP address for example.com is then returned to the resolver from the nameserver.
8. The DNS resolver then responds to the web browser with the IP address of the domain requested initially.

Once the 8 steps of the DNS lookup have returned the IP address for example.com, the browser is able to make the request for the web page:

1. The browser makes a HTTP request to the IP address.
2. The server at that IP returns the webpage to be rendered in the browser.

To save this hassle, modern web browser and operating systems uses DNS caching to store these DNS records locally which is checked before making a DNS query.



**How does browser fetch the desired result:**

* Once browser gets the IP address of the server (website), it uses fetch/XHR HTTPS request for the required resource.
* Browser receives required data in small chunks in data packets which is unit used over the Internet to carry any information.
* The browser then assembles these data packets and creates a web page (any other resource).
* Then browser parses the HTML and loads any dependencies which can be css, javascript files or images and also parses them.
* Finally, browser renders the page on the screen.

As explained above the main functionality of the browser is to find, access and display websites. It does it in the way explained above but lot of other moving parts are not yet discussed. Following will explain the components involved in web browser.

**High Level Components of a browser:**

Following are main components of the browser

1. User Interface: This includes common user interface elements like address bar, back/forward buttons, refresh button, bookmark menu/tab, etc. Browser user interface changes from browser to browser like chrome and safari has different appearance.
2. Browser Engine: It sits between User Interface and Rendering Engine as a communicator
3. Rendering Engine: This is where the magic happens. Its main purpose is to display the web page. One of the most complex steps like parsing HTML and CSS, computing layout, painting, etc. are handled by the rendering engine.
4. Networking: This is an interface which makes HTTPS request over the network.
5. UI Backend: This is browsers interface which communicates with Operating System. Browsers interfaces are written in language which OS understands like C++.
6. Javascript Interpreter: This parses and executes the javascript.
7. Data Storage: This helps to store the data locally for functionalities like cookies, localStorage, IndexedDB, etc.



**Parsing, render tree and layout**

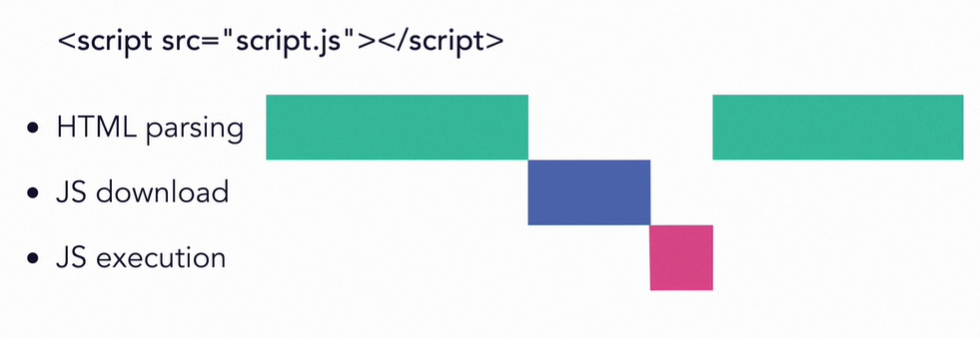
* HTML Parser created DOM (Document Object Model) tree and CSS Parser parses CSS into CSSOM (Cascaded Style Sheets Object Model).
* Then render tree is generated by combining DOM tree and CSSOM tree. Layout which is also called as Reflow is then computed on render tree which finally paints the content on the screen.
* Layout involves mainly calculating position and size of the nodes in the render tree. The coordinate system is relative to the root frame. Top and left coordinates are used.
* HTML uses a flow-based layout model, meaning that most of the time it is possible to compute the geometry in a single pass. Elements later ``in the flow'' typically do not affect the geometry of elements that are earlier ``in the flow'', so layout can proceed left-to-right, top-to-bottom through the document. There are exceptions: for example, HTML tables may require more than one pass.

**Script Processing**

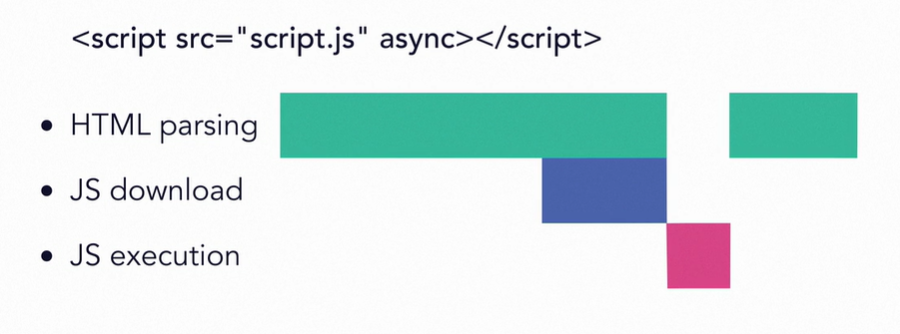
* First HTML is pulled down and indexed then loads javascript and css and runs javascript and changes HTML then applies css and runs javascript in background for event handling.
* As javascript is single-threaded, it faces the problem of Render Blocking in which browser holds off rendering until javascript is loaded and executed.

Javascript can be loaded in three ways

HTML Parsing : default, stops rendering and load and executes script



Async : async, download along with html but while executing it does not download html



Deferred : defer, same as async only executes the script after dom is loaded

