**When a user enters an URL in the browser, how does the browser fetch the desired result?**

**Answer:**

Steps for what happens when we enter a URL:

1.Browser checks cache for DNS entry to find the corresponding IP address of website.

It looks for following cache. If not found in one, then continues checking to the next until found.

(1) Browser Cache

(2) Operating Systems Cache

(3) Router Cache

(4) ISP Cache

2.If not found in cache, ISP’s (Internet Service Provider) DNS server initiates a DNS query to find IP address of server that hosts the domain name.

The requests are sent using small data packets that contain information content of request and IP address it is destined for.

3.Browser initiates a TCP (Transfer Control Protocol) connection with the server using synchronize (SYN) and acknowledge (ACK) messages.

4.Browser sends an HTTP request to the web server. GET or POST request.

5.Server on the host computer handles that request and sends back a response. It assembles a response in some format like JSON, XML and HTML.

6.Server sends out an HTTP response along with the status of response.

7.Browser displays HTML content

8.Finally, Done.

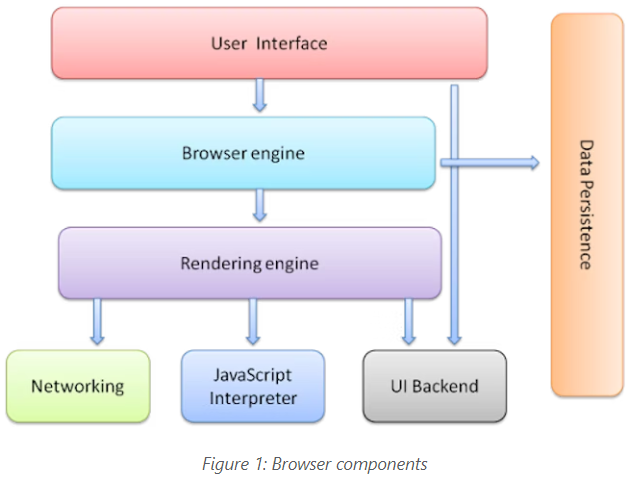
**a) What is the main functionality of the browser?**

**Answer:**

A web browser takes you anywhere on the internet. It retrieves information from other parts of the web and displays it on your desktop or mobile device. The information is transferred using the Hypertext Transfer Protocol, which defines how text, images and video are transmitted on the web.

**b) High Level Components of a browser**

Answer:



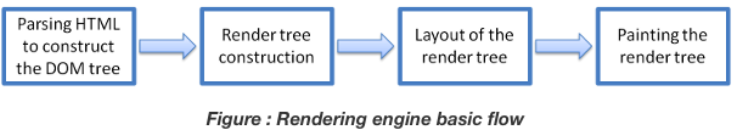
The browser's main components are:

1. **The user interface**: this includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser display except the window where you see the requested page.
2. **The browser engine**: marshals actions between the UI and the rendering engine.
3. **The rendering engine**: responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.
4. **Networking**: for network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.
5. **UI backend**: used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.
6. **JavaScript interpreter**. Used to parse and execute JavaScript code.
7. **Data storage**. This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as local Storage, Indexed DB, Web SQL and File System.

**c) Rendering engine and its use**

**Rendering engine**

A **rendering engine** is software that draws text and images on the screen. The engine draws structured text from a document (often [HTML](https://developer.mozilla.org/en-US/docs/Glossary/HTML)), and formats it properly based on the given style declarations (often given in [CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS)). Examples of layout engines: [Blink](https://developer.mozilla.org/en-US/docs/Glossary/Blink), [Gecko](https://developer.mozilla.org/en-US/docs/Glossary/Gecko), Edge HTML, Web Kit.



**d) Parsers (HTML, CSS, etc)**

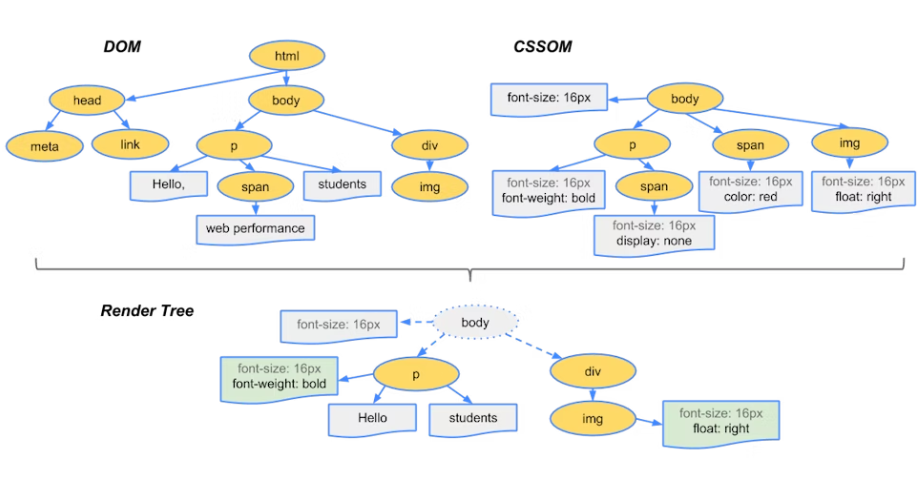
Parsing means analysing and converting a program into an internal format that a runtime environment can actually run, for example the [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript) engine inside browsers.

**e) Script Processors**

The Script processor **allows you to specify your own processor logic for a simple processor using JavaScript or Groovy**. The script is entered as an option on the script processor.

**f) Tree construction**

**The input to the tree construction stage is a sequence of tokens from the tokenization stage**. The tree construction stage is associated with a DOM Document object when a parser is created. The "output" of this stage consists of dynamically modifying or extending that document's DOM tree.



**g) Order of script processing**

Designing your web page using JavaScript requires attention to the order in which your code appears and whether you are encapsulating code into functions or objects, all of which impact the order in which the code runs.

**h) Layout and Painting**

**Layout**

The layout (also called reflow) peace will be in charge to calculate the positions and dimensions of each node on the screen. For instance, if you rotate your phone, or if you resize your browser, the layout peace will be executed.

**Paint**

Finally, now that we know which nodes are visible, and their computed styles and geometry, we can pass this information to the final stage, which converts each node in the render tree to actual pixels on the screen. This step is often referred to as “painting”, “rasterizing.” or “repainting”.

**RESOURES:**

1. <https://developer.mozilla.org/>
2. <https://www.oracle.com/>
3. <https://dev.to/>
4. <https://medium.com/>