***Ques:-***

When a user enters an URL in the browser, how does the browser fetch the desired result ? Explain this with the below in mind and demonstrate this by drawing a diagram for the same.(2-3hours)

a. What is the main functionality of the browser?

b. High Level Components of a browser.

c. Rendering engine and its use.

d. Parsers (HTML, CSS, etc)

e. Script Processors

f. Tree construction

g. Order of script processing

h. Layout and Painting

***Ans:-***

A browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, video and other files. As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

## **The browser's high-level structure**

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.



## **Rendering engines**

Different browsers use different rendering engines: Internet Explorer uses Trident, Firefox uses Gecko, Safari uses Web Kit. Chrome and Opera (from version 15) use Blink, a fork of Web Kit.

Web Kit is an open-source rendering engine which started as an engine for the Linux platform and was modified by Apple to support Mac and Windows.

## **The main flow**

The rendering engine will start getting the contents of the requested document from the networking layer. This will usually be done in 8kB chunks.

After that, this is the basic flow of the rendering engine:

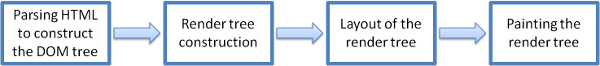


Figure : Rendering engine basic flow

The rendering engine will start parsing the HTML document and convert elements to DOM nodes in a tree called the "content tree". The engine will parse the style data, both in external CSS files and in style elements. Styling information together with visual instructions in the HTML will be used to create another tree: the render tree.

The render tree contains rectangles with visual attributes like color and dimensions. The rectangles are in the right order to be displayed on the screen.

After the construction of the render tree, it goes through a layout process. This means giving each node the exact coordinates where it should appear on the screen. The next stage is [painting](https://web.dev/howbrowserswork/#Painting) - the render tree will be traversed, and each node will be painted using the UI backend layer.

It's important to understand that this is a gradual process. For better user experience, the rendering engine will try to display contents on the screen as soon as possible. It will not wait until all HTML is parsed before starting to build and layout the render tree. Parts of the content will be parsed and displayed, while the process continues with the rest of the contents that keeps coming from the network

### **Parsing - general**

Parsing a document means translating it to a structure the code can use. The result of parsing is usually a tree of nodes that represent the structure of the document. This is called a parse tree or a syntax tree.

#### **Tree construction algorithm**

When the parser is created the Document object is created. During the tree construction stage the DOM tree with the Document in its root will be modified and elements will be added to it. Each node emitted by the tokenizer will be processed by the tree constructor. For each token the specification defines which DOM element is relevant to it and will be created for this token. The element is added to the DOM tree, and also the stack of open elements. This stack is used to correct nesting mismatches and unclosed tags. The algorithm is also described as a state machine. The states are called "insertion modes".

## The order of processing scripts

### Scripts

### Speculative parsing

### Style sheets

## Render tree construction

### Style Computation

### Sharing style data

## Layout

When the renderer is created and added to the tree, it does not have a position and size. Calculating these values is called layout or reflow.

Steps for what happens when we enter a URL

1. When we enter URL, browser checks cache for DNS record to find the corresponding IP address of URL.

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

* Browser Cache
* Operating Systems Cache
* Router Cache
* ISP Cache

1. If the requested URL is not in the cache, ISP’s DNS server initiates a DNS query to find the IP address of the server that hosts the website which we are looking for. The requests are sent using small data packets that contain information content of request and IP address it is destined for.
2. Once the browser receives the correct IP address, it will build a connection with the server that matches the IP address to transfer information. Browser initiates a [TCP (Transfer Control Protocol)](https://www.geeksforgeeks.org/tcp-and-udp-in-transport-layer/) connection with the server using synchronize(SYN) and acknowledge(ACK) messages.
3. Once the TCP connection is established, it is time to start transferring data! Browser sends an [HTTP](https://www.geeksforgeeks.org/http-non-persistent-persistent-connection/) request to the web server. GET or POST request. The browser will send a GET request asking for web page. If you’re entering credentials or submitting a form, this could be a POST request.
4. After that, the server on the host computer handles the corresponding request and sends back a response. It assembles a response in some format like JSON, [XML](https://www.geeksforgeeks.org/xml-basics/) and HTML.
5. Server sends out an HTTP response along with the status of response. The server response contains the web page you requested as well as the status code, compression type (Content-Encoding), how to cache the page (Cache-Control), any cookies to set, privacy information, etc.
6. And finally, the browser displays the HTML content in phases. First, it will render the bare bone HTML skeleton. Then it will check the HTML tags and send out GET requests for additional elements on the web page, such as images, CSS stylesheets, JavaScript files, etc. These static files are cached by the browser, so it doesn’t have to fetch them again the next time you visit the page.