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

A web browser is application software for accessing the World Wide Web or a local website. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device.

**High Level Components of a browser?**

**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.

**The browser engine:** marshals actions between the UI and the rendering engine.

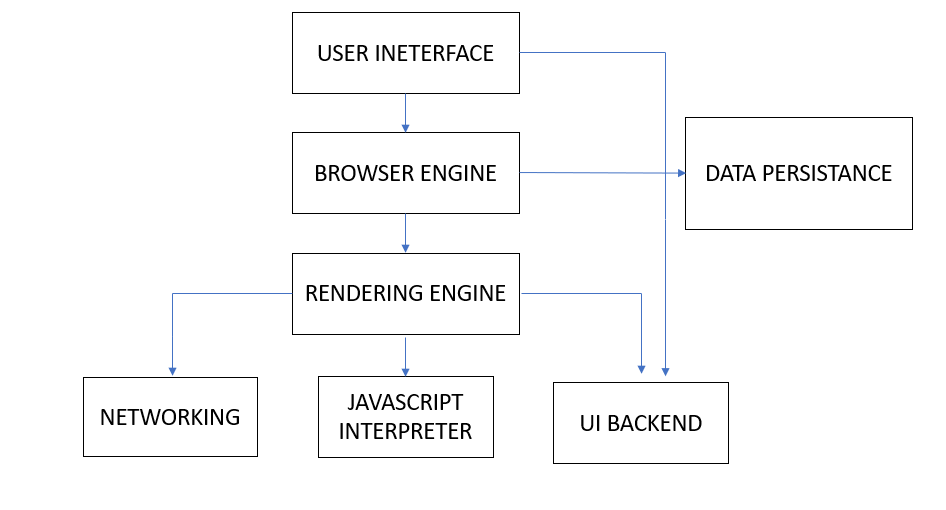
**The rendering engine:** responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.

**Networking:** for network calls such as HTTP requests, using different implementations for different platforms behind a platform-independent interface.

**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.

**JavaScript interpreter**. Used to parse and execute JavaScript code.

**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 localStorage, IndexedDB, WebSQL and FileSystem.



**Rendering engine and its use.**

Once a user requests a particular document, the rendering engine starts fetching the content of the requested document. This is done via the networking layer. The rendering engine starts receiving the content of that specific document in chunks of 8 KBs from the networking layer. After this, the basic flow of the rendering engine begins

* The requested HTML page is parsed in chunks, including the external CSS files and in style elements, by the rendering engine. The HTML elements are then converted into DOM nodes to form a “content tree” or “DOM tree.”
* Simultaneously, the browser also creates a render tree. This tree includes both the styling information as well as the visual instructions that define the order in which the elements will be displayed. The render tree ensures that the content is displayed in the desired order.
* Further, the render tree goes through the layout process. When a render tree is created, the position or size values are not assigned. The entire process of calculating values for evaluating the desired position is called a layout process. In this process, every node is assigned the exact coordinates. This ensures that every node appears at an accurate position on the screen.
* The final step is to paint the screen, wherein the render tree is traversed, and the renderer’s paint() method is invoked, which paints each node on the screen using the UI backend layer.

**The order of processing scripts and style sheets**

The model of the web is synchronous. Authors expect scripts to be parsed and executed immediately when the parser reaches

a <script> tag. The parsing of the document halts until the script has been executed. If the script is external then the

resource must first be fetched from the network–this is also done synchronously, and parsing halts until the resource is fetched.

This was the model for many years and is also specified in HTML4 and 5 specifications. Authors can add the "defer" attribute to a

script, in which case it will not halt document parsing and will execute after the document is parsed. HTML5 adds an option to mark

the script as asynchronous so it will be parsed and executed by a different thread.