A browser helps you to travel anywhere on the internet. It mainly retrieves information or data from other parts of the web and displays it on your device. Using Hypertext Transfer Protocol, which describes how text, images and the videos are transmitted from the internet. This information needs to be shared and displayed in a consistent format so that people using any browser, anywhere in the world can see the information. not all browser makers choose to interpret the format in the same way. For users, this means that a website can look and function differently. Creating consistency between browsers, so that any user can enjoy the internet, regardless of the browser they choose, is called web standards.Whenever the web browser fetches data from an internet connected server it uses a piece of software called a rendering engine to translate that data into text and images. This data is written in Hypertext Markup Language (HTML) and web browsers read this code to create what we see, hear and experience on the internet.Hyperlinks allow users to follow a path to other pages or sites on the web. Every webpage, image and video has its own unique Uniform Resource Locator (URL), which is also known as a web address. When a browser visits a server for data, the web address tells the browser where to look for each item that is described in the html, which then tells the browser where it goes on the web page.
The user interface: this includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser displays except the window where you see the requested page.

1. The browser engine: marshals actions between the UI and the rendering engine.
2. 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.
3. Networking: for network calls such as HTTP requests, using different implementations for different platforms behind a platform-independent interface.
4. 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.
5. JavaScript interpreter. Used to parse and execute JavaScript code.
6. 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

A rendering engine is software that draws text and images on the screen. The engine draws structured text from a document (often HTML), and formats it properly based on the given style declarations (often given in CSS). Examples of layout engines: Blink, Gecko, EdgeHTML, WebKit.
So we have HTML content at the beginning which goes through a process called tokenization. Tokenization is a common process in almost every programming language where code is split into several tokens which are easier to understand while parsing. This is where the HTML's parser understands which is the start and which is the end of the tag, which tag it is and what is inside the tag.
Now we know, html tag starts at the top and then the head tag starts before the html ends so we can figure out that the head is inside html and create a tree out of it. Thus we then get something called a parse tree which eventually becomes a DOM tree.The process of creating and embedding scripts in a web page is known as web-scripting. A script or a computer-script is a list of commands that are embedded in a web-page normally and are interpreted and executed by a certain program or scripting engine.Scripts may be written for a variety of purposes such as for automating processes on a local-computer or to generate web pages.The programming languages in which scripts are written are called scripting

languages, there are many scripting languages available today.Common scripting languages are VBScript, JavaScript, ASP, PHP, PERL, JSP etc.There are Two types of scripts. Client Side, Server Side. The CSSOM and DOM trees are combined into a render tree, which is then used to compute the layout of each visible element and serves as an input to the paint process that renders the pixels to screen. Optimizing each of these steps is critical to achieving optimal rendering performance.In the previous section on constructing the object model, we built the DOM and the CSSOM trees based on the HTML and CSS input. However, both of these are independent objects that capture different aspects of the document: one describes the content, and the other describes the style rules that need to be applied to the document.

- The DOM and CSSOM trees are combined to form the render tree.
- Render tree contains only the nodes required to render the page.
- Layout computes the exact position and size of each object.
- The last step is paint, which takes in the final render tree and renders the pixels to the screen.

To construct the render tree, the browser roughly does the following:

1. Starting at the root of the DOM tree, traverse each visible node.
    - Some nodes are not visible (for example, script tags, meta tags, and so on), and are omitted since they are not reflected in the rendered output.
    - Some nodes are hidden via CSS and are also omitted from the render tree; for example, the span node---in the example above---is missing from the render tree because we have an explicit rule that sets the "display: none" property on it.
2. For each visible node, find the appropriate matching CSSOM rules and apply them.
3. Emit visible nodes with content and their computed styles.

Executable objects go through four execution stages. The second one is the generation stage. Scripts are generated during this stage. The time at which the script is generated depends on object attributes. The order in which scripts are processed in an object depends on which Process pages the scripts are on.
This page includes the following:

- Execution Stages
- Time of Processing
- Order of Processing
- Processing In Scripts

JavaScript: The use of JavaScript is to achieve visual changes. For example, to add an animation or DOM element.
Style: Matches each DOM element with the corresponding CSS style based on the CSS selector. After this step, the CSS style rule for each DOM element gets determined.
Layout: Calculates the size and position of each DOM element to be displayed on the screen. The layout of elements on a Web page is relative, which means a single element can affect others. For example, in case there is a change in the width of the element, the widths of its child and grandchild elements get affected. Therefore, the layout process often gets involved for the browser.
Paint: Essentially, painting is the process of filling in pixels. It requires painting of text, colors, images, borders, shadows, and other visual effects of a DOM element. In general, the painting gets completed in multiple layers.
Composite: As mentioned above, the painting of DOM elements gets done at numerous layers on the page. Once it is complete, the browser combines all the layers into one layer in the correct order and displays them on the screen. This process is especially important for

pages with overlapping elements as the incorrect layer composition order may result in an abnormal display of the elements.