# Week-1 Assignment

Exercise 1.1:

A web browser is an application for accessing website. 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. 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.

The browser’s main components are:

1. **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:** marshal actions between the UI and the rendering engine.
3. **The rendering engine:** responsible for displaying requested content. For example, if the requested content in HTML, the rendering engine parsers 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 localStorage, IndexedDB, WebSQL and FileSystem.



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

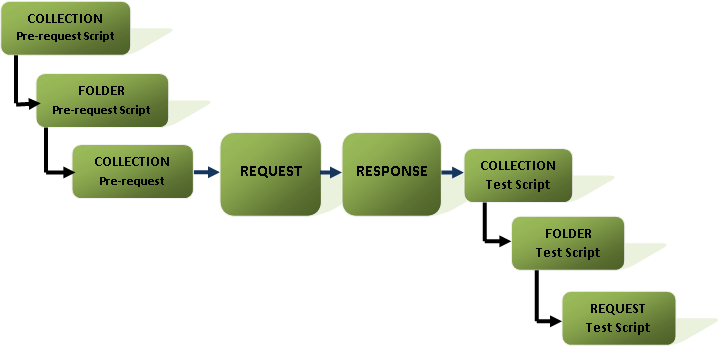
The rendering engine will start parsing the HTML document and convert elements to DOM nodes in a tree called the "content tree". The engine will parse the style data, both in external CSS files and in style elements. Styling information together with visual instructions in the HTML will be used to create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

Parsing can be separated into two sub processes: lexical analysis and syntax analysis.

Lexical analysis is the process of breaking the input into tokens. Tokens are the language vocabulary: the collection of valid building blocks. In human language it will consist of all the words that appear in the dictionary for that language. Syntax analysis is the applying of the language syntax rules.

Parsers usually divide the work between two components: the lexer (sometimes called tokenizer) that is responsible for breaking the input into valid tokens, and the parser that is responsible for constructing the parse tree by analysing the document structure according to the language syntax rules.

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.

The browsers use algorithms for parsing, The Parsing Algorithm and the tree construction algorithm and the tokenization algorithms.

The following steps are implemented when URL is entered in the browser.

1. Browser looks for the protocol that tells the browser to make a connection using Transport Layer Security.
2. The Domain specified points the server IP address with respect to the path where the scripts and been stored.
3. The browsers obtain those files and generates the static content.
4. The JavaScript files contains events that are used to obtain and modify data. For obtaining data it triggers the request methods such as GET, POST for gathering the data.
5. These data can be obtained in the form of JavaScript, XML etc... These data are further displayed as per the instructions provided in the HTML and JavaScript code.