**HOW BROWSERS WORK.**

The main function of *browsers* is to present the web recourse chosen, by requesting it from the server and displaying it in the browser window. Resource can be HTML document PDF, image, etc…

The location of the resource is specified by the user using a URI (Uniform Resource Identifier).

**The five major browsers on Desktop today are:**

1. Chrome
2. Firefox
3. Internet Explorer
4. Safari
5. Opera

**On mobile, the main are:**

1. Android browser
2. iPhone
3. Opera mini & Opera mobile
4. UC browser
5. Chrome…etc

All except **Opera** browsers are based on Webkit.

Browser user interfaces have a lot in common with each other. Among the common user interface elements are:

* Address bar for inserting a URI
* Back and forward buttons
* Bookmarking options
* Refresh and stop buttons for refreshing or stopping the loading of current documents
* Home button that takes you to your home page

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 localStorage, IndexedDB, WebSQL and FileSystem.

**Rendering engines**

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.

A **DOM** is the object presentation of the HTML document and the interface of HTML elements to the outside world like JavaScript.

A **Render tree** is a tree used to enable painting the contents in their correct order.

**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.

HTML cannot be parsed using the regular top down or bottom up parsers.

The reasons are:

1. The forgiving nature of the language.
2. The fact that browsers have traditional error tolerance to support well known cases of invalid HTML.
3. The parsing process is reentrant. For other languages, the source doesn't change during parsing, but in HTML, dynamic code (*such as script elements containing document.write() calls*) can add extra tokens, so the parsing process actually modifies the input.

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

**Lexical analysis** is the process of breaking the input into tokens.

**Syntax analysis** is the applying of the language syntax rules.

**Translation**

Parsing is often used in *translation:* transforming the input document to another format. An example is **compilation**. The compiler that compiles *source code* into *machine code* first parses it into a parse tree and then translates the tree into a machine code document.