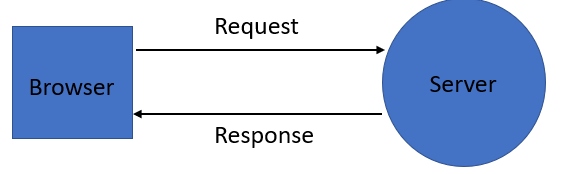
Assignment 1

What is the main functionality of the browser?

Browsers enable us to access the world wide web. To access a website, we need it’s URL (Uniform resource locator).

We are able to send request to the servers for a specific web page and servers will send the requested page as response. We can use browsers as debugging tool as well.

Browsers store the data locally on PC for session management, with each user request required data is sent in the request object to the serves.



High Level Components of a browser

1. **User Interface:**

It is an environment allowing users to use certain features like search bar, refresh button, menu, bookmarks, etc.

1. **Browser Engine:**

The bridge connects the interface and the engine. It monitors the rendition engine while manipulating the inputs coming from multiple user interfaces.

1. **Networking:**

The protocol provides an URL and manages all sorts of safety, privacy and communication.  
In addition, the store network traffic gets saved in retrieved documents.

1. **Data Storage:**

The cookies store information as the data store is an uniform layer that the browsers use. Storage processes like IndexedDB, WebSQL, localStorage, etc works well on browsers.

JavaScript Interpreter:

It allows conversion of JavaScript code in a document and the executes it. Then the engine shows the translation on the screen to the users.

* **Rendering Engine:**

As the name suggests, this component is responsible for rendering a specific web page requested by the user on their screen. It interprets HTML and XML documents along with images that are styled or formatted using CSS, and a final layout is generated, which is displayed on the user interface.

* **Networking:**This component is responsible for managing network calls using standard protocols like HTTP or FTP. It also looks after security issues associated with internet communication.
* **JavaScript Interpreter**: As the name suggests, it is responsible for parsing and executing the JavaScript code embedded in a website. Once the interpreted results are generated, they are forwarded to the rendering engine for displaying on the user interface.
* **UI Backend:**This component uses the user interface methods of the underlying operating system. It is mainly used for drawing basic widgets (windows and combo boxes).
* **Data Storage/Persistence:** It is a persistent layer. A web browser needs to store various types of data locally, for example, cookies. As a result, browsers must be compatible with data storage mechanisms such as WebSQL, IndexedDB, FileSystem, etc.

**Parsers (HTML, CSS, etc)**

The DOM is the internal representation of the markup for the browser. The DOM is also exposed, and can be manipulated through various APIs in JavaScript.

Even if the request page's HTML is larger than the initial 14KB packet, the browser will begin parsing and attempting to render an experience based on the data it has. This is why it's important for web performance optimization to include everything the browser needs to start rendering a page, or at least a template of the page - the CSS and HTML needed for the first render — in the first 14 kilobytes. But before anything is rendered to the screen, the HTML, CSS, and JavaScript have to be parsed.

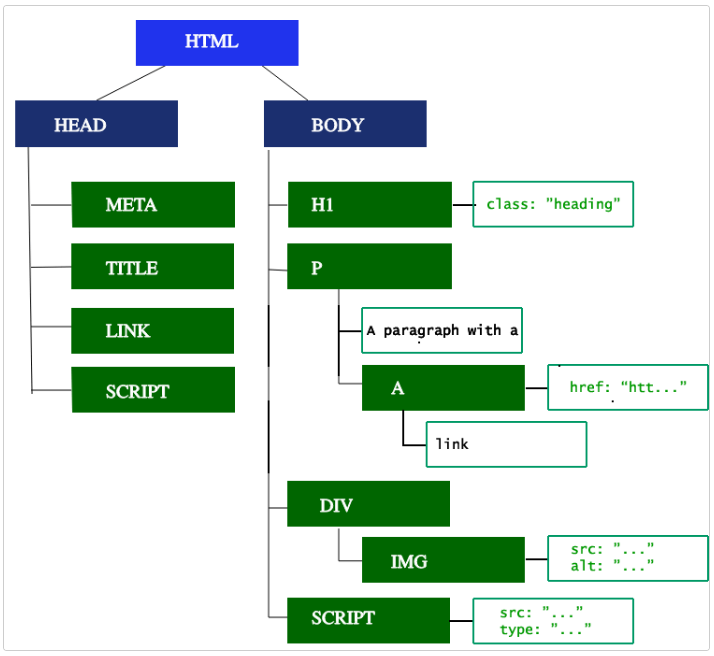
**Script Processors**

The script processor executes Javascript code to process an event. The processor uses a pure Go implementation of ECMAScript 5.1 and has no external dependencies. This can be useful in situations where one of the other processors doesn’t provide the functionality you need to filter events.

The processor can be configured by embedding Javascript in your configuration file or by pointing the processor at external file(s).

**Tree construction**

The DOM tree describes the content of the document. The [<html>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html) element is the first tag and root node of the document tree. The tree reflects the relationships and hierarchies between different tags. Tags nested within other tags are child nodes. The greater the number of DOM nodes, the longer it takes to construct the DOM tree.



**Order of script processing**

1. Loading order: introducing the order in which tags < script /> appear,

The Javascript code on the page is part of the HTML document, so the order in which Javascript is loaded is the order in which the tag < script /> appears, and the external JS in the < script /> tag or introduced through src is executed in the order in which the statement appears, and the execution process is part of the document

**2. Global variables** and functions defined by each script can be invoked by subsequent scripts.

Variable calls must be declared previously, otherwise the value of the variable obtained is undefined.

3. In the same script, function definitions can appear after function calls, but if they are in two separate sections of code and function calls are in the first section of code, an undefined function error will be reported.

**4.document.write()** writes the output to the location of the script document. After the browser parses the content of the document where documemt.write(), it continues to parse the content of the document.write().  
Then proceed to parse the HTML document.

5. **Execution order of JS** function with the same name

**Layout and Painting**

Layout is the process by which the width, height, and location of all the nodes in the render tree are determined, plus the determination of the size and position of each object on the page. *Reflow* is any subsequent size and position determination of any part of the page or the entire document.

Once the render tree is built, layout commences. The render tree identified which nodes are displayed (even if invisible) along with their computed styles, but not the dimensions or location of each node. To determine the exact size and location of each object, the browser starts at the root of the render tree and traverses it.

**Paint:**

The last step in the critical rendering path is painting the individual nodes to the screen, the first occurrence of which is called the [first meaningful paint](https://developer.mozilla.org/en-US/docs/Glossary/first_meaningful_paint). In the painting or rasterization phase, the browser converts each box calculated in the layout phase to actual pixels on the screen. Painting involves drawing every visual part of an element to the screen, including text, colors, borders, shadows, and replaced elements like buttons and images. The browser needs to do this super quickly.