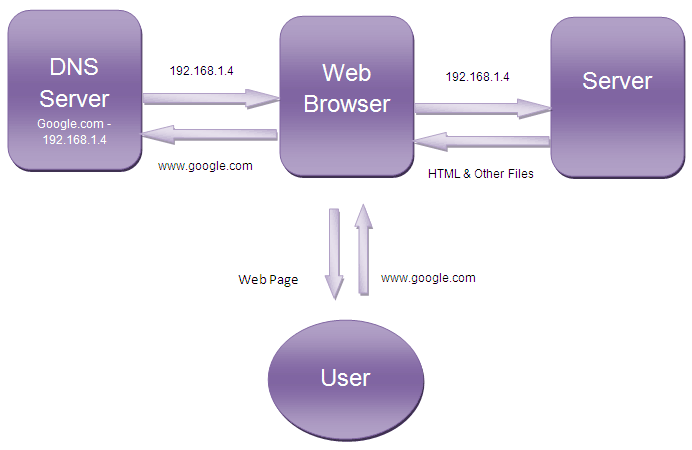
**Define browser**:

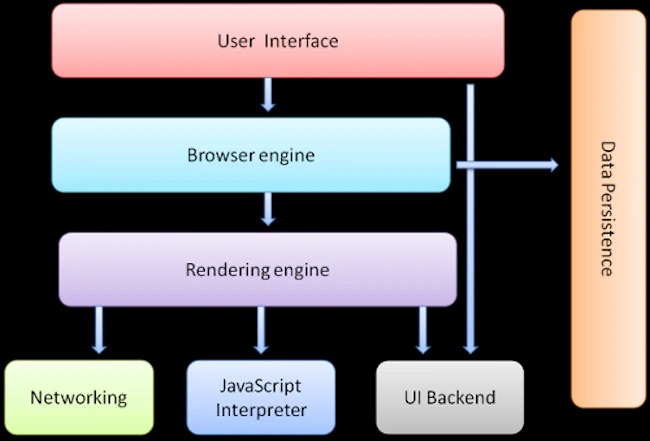
A browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, video and other files. As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

**What happens when URL is entered in browser:**



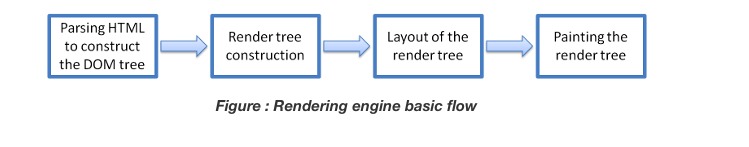
1. Browser checks cache for DNS entry to find the corresponding IP address of website.  
   It looks in the following cache. IT continues checking each cache, one after another:
   * Browser Cache
   * Operating Systems Cache
   * Router Cache
   * ISP Cache
2. If not found in cache, ISP’s (Internet Service Provider) DNS server initiates a DNS query to find IP address of server that hosts the domain name.
3. Browser initiates a TCP (Transfer Control Protocol) connection with the server.
4. Browser sends a HTTP request to the web server. GET or POST request.
5. Server on the host computer handles that request and sends back a response, along with status of the response. It assembles a response in some format like JSON, XML and HTML.
6. Browser displays HTML content.

**High Level Components of browser:**



1. **The User Interface**: The user interface is the space where the user interacts with the browser. It includes the address bar, back and next buttons, home button, refresh and stop, bookmark option, etc. Every other part, except the window where requested web page is displayed, comes under it.
2. **The Browser Engine**: The browser engine works as a bridge between the User interface and the rendering engine. According to the inputs from various user interfaces, it queries and manipulates the rendering engine.
3. **The Rendering Engine**: The rendering engine is responsible for rendering the requested web page on the browser screen. The rendering engine interprets the HTML, XML documents and images that are formatted using CSS and generates the layout that is displayed in the User Interface. However, it can display other types of data also. Different browsers user different rendering engines:  
   \* Internet Explorer: Trident  
   \* Firefox & other Mozilla browsers: Gecko  
   \* Chrome & Opera 15+: Blink  
   \* Chrome (iPhone) & Safari: Webkit
4. **Networking**: Component of the browser which retrieves the URLs using the common internet protocols of HTTP or FTP. The networking component handles all aspects of Internet communication and security. The network component may implement a cache of retrieved documents in order to reduce network traffic.
5. **JavaScript Interpreter:** It is the component of the browser which interprets and executes the JavaScript code embedded in a website. The interpreted results are sent to the rendering engine for display. If the script is external, then first the resource is fetched from the network. Parser keeps on hold until the script is executed.
6. **UI Backend**: UI backend is used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. It underneath uses operating system user interface methods.
7. **Data Persistence/Storage:** This is a persistence layer. Browsers support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem. It is a small database created on the local drive of the computer where the browser is installed. It manages user data such as cache, cookies, bookmarks and preferences.

**Functioning of Rendering Engine:**



The four basic steps include:

* 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.

**Parsing:**

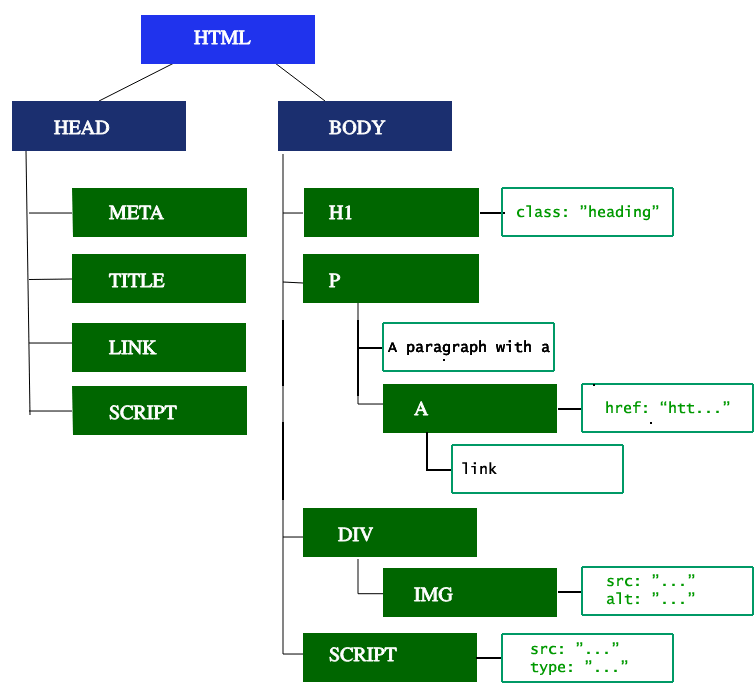
Once the browser receives the first chunk of data, it can begin parsing the information received. Parsing is the step the browser takes to turn the data it receives over the network into the DOM and CSSOM, which is used by the renderer to paint a page to the screen.

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.

**Building DOM Tree:**

The DOM tree describes the content of the document. The <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.

When the parser finds non-blocking resources, such as an image, the browser will request those resources and continue parsing. Parsing can continue when a CSS file is encountered, but <script> tags—particularly those without an async or defer attribute—block rendering and pause the parsing of HTML. Though the browser's preload scanner hastens this process, excessive scripts can still be a significant bottleneck.

.

**Preload Scanner:**

While the browser builds the DOM tree, this process occupies the main thread. As this happens, the *preload scanner* will parse through the content available and request high priority resources like CSS, JavaScript, and web fonts. Thanks to the preload scanner, we don't have to wait until the parser finds a reference to an external resource to request it. It will retrieve resources in the background so that by the time the main HTML parser reaches requested assets, they may already be in flight, or have been downloaded. The optimizations the preload scanner provides reduce blockages. Waiting to obtain CSS doesn't block HTML parsing or downloading, but it does block JavaScript, because JavaScript is often used to query CSS properties' impact on elements.

**Building CSSOM:**

The second step in the critical rendering path is processing CSS and building the CSSOM tree. The CSS object model is similar to the DOM. The DOM and CSSOM are both trees. They are independent data structures. The browser converts the CSS rules into a map of styles it can understand and work with. The browser goes through each rule set in the CSS, creating a tree of nodes with parent, child, and sibling relationships based on the CSS selectors.

**Render:**

The CSSOM and DOM trees created in the parsing step are combined into a render tree which is then used to compute the layout of every visible element, which is then painted to the screen. Rendering steps include style, layout and paint.

* Style: The computed style tree, or render tree, construction starts with the root of the DOM tree, traversing each visible node. Each visible node has its CSSOM rules applied to it. The render tree holds all the visible nodes with content and computed styles — matching up all the relevant styles to every visible node in the DOM tree, and determining, based on the CSS cascade, what the computed styles are for each node.
* Layout: It 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 first time the size and position of nodes are determined is called layout. Subsequent recalculations of node size and locations are called reflows.
* 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. 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.