Browsers and how they work?

Notes based on medium article: https://medium.com/@monica1109/how-does-web-browsers-work-c95ad628a509

How does web browsers work?

- A bowser is a software application used to locate, retrieve, and display content on the World Wide Web, including Web pages, images, video and other files
- Client/server model
- Browser is the client run on a computer that contacts the Web server and requests information
- Web server sends the information back to the Web browser which displays the results on the computer
- Today's browsers are fully-functional software suites that can interpret and display HTML Web pages, applications, JavaScript, AJAX and other content hosted on Web servers
- Many browsers offer plug-ins which extend the capabilities of the software
- A browser is a group of structured codes which together performs a series of tasks to display a web page on the screen
- According to the task they perform, these codes are made as different components
- Main components of a web browser:

1. **The User Interface**: The space where 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**: Works as a bridge between the UI and the rendering engine. According to the inputs from the various UI, it queries and manipulates the rendering engine.
3. **The Rendering Engine**: 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 UI. However, using plugins or extensions, it can display other types of data also. Different browsers uses 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 TP or FTP. This component handles all aspects of Internet communication and security.
5. **JavaScript Interpreter**: Component that interprets and executes the JavaScript code embedded in a website. 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**: Used for drawing basic widgets like combo boxes and windows. A generic interface that is not platform specific it uses an OS's UI methods.
7. **Data Persistence/Storage**: A persistence layer. Browsers support storage mechanisms such as localStorage, IdexedDB, 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.

- An important note: In web browsers such as Google Chrome each tab runs in a separate process (multiple instances of rendering engine)
- The networking layer will start sending the content of the requested documents to the rendering engine in chunks of 8KBs

Parsing HTML to construct the DOM tree --> Render tree construction --> Layout of the render tree --> Painting the render tree

- The rendering engine parses the chunks of the HTML document and convert the elements to the DOM nodes in a tree called the "content tree" or the "DOM tree". It also parses both the external CSS files as well in style elements.
- While the DOM tree is being constructed, the browser constructs another tree, the render tree. This tree is of visual elements in the order in which they will be displayed. It is the visual representation of the document. The purpose of this tree is to enable painting the contents in

their correct order. Firefox calls the elements in the render tree "frames". WebKit uses the term renderer or render object.

- After the construction of the render tree, it goes through a "layout process" of the render tree. When the renderer is created and added to the tree, it does not have a position and size. The process of calculating these values is called layout or reflow. This means giving each node the exact coordinates where it should appear on the screen. The position of the root renderer is 0, 0 and its dimensions are the viewport - the visible part of the browser window.
- The next stage is painting. The render tree is traversed and the renderer's "paint()" method is called to display content on the screen. Painting uses the UI backend layer.
- The rendering engine always tries to display the contents on the screen as soon as possible for better user experience. It does not wait for the HTML parsing to complete before starting to build and layout the render tree. It parse and displays the content it has received from the network, while rest of the content still keeps coming from the network.

Other resources:

- https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/#Resources
- https://www.mozilla.org/en-CA/firefox/browsers/what-is-a-browser/#:~:text=How%20does%20a%20web%20browser,are%20transmitted%20on%20the%20web.
- https://www.freecodecamp.org/news/web-application-security-understanding-the-browser-5305ed2f1dac/
- https://learning.oreilly.com/library/view/take-control-of/9781947282360/