JavaScript - Basics to Advance

Introduction to JS and its working

JavaScript is a high-level, interpreted programming language primarily used for client-side web development. It is used to create dynamic content for websites.

Why Interpreted?

JavaScript code is executed line by line by an interpreter at runtime, rather than being compiled into machine code ahead of time. This allows for more flexibility and easier debugging.

Platform Independence: Because it is interpreted, JavaScript can run on any platform that has a compatible interpreter (e.g., web browsers). This makes it highly portable and versatile.

Dynamic Typing: JavaScript doesn’t require you to declare variable types explicitly, which is a common feature of interpreted languages. This adds to its flexibility and ease of use.

Client-side web development refers to the creation and management of the part of a web application or website that runs in the user's web browser. JS helps in adding interactivity and dynamic content to web pages.

JavaScript is often referred to as an interpreted programming language because it is typically executed line by line, or in a statement-by-statement manner, by an interpreter directly without prior compilation into machine code. This is in contrast to languages like C or C++, which are typically compiled before execution into machine code that can be directly executed by the computer's hardware.

In the case of JavaScript, the code is interpreted by the JavaScript engine of the web browser or runtime environment (like Node.js). When a JavaScript file is loaded in a browser, for example, the browser's JavaScript engine reads the code, interprets it, and executes it on-the-fly. This process doesn't involve a separate compilation step into machine code. Instead, the interpreter translates the JavaScript code into machine code instructions as it runs.

Javascript Engines

A JavaScript engine is a program or component that executes JavaScript code. It's essentially the runtime environment for JavaScript, responsible for interpreting and executing JavaScript code. JavaScript engines are typically found within web browsers to handle JavaScript code embedded in web pages, but they can also be standalone environments like Node.js, which allows JavaScript to be run on servers.

E.g. of JS Engines: Popular JavaScript engines include V8 (used in Google Chrome and Node.js), SpiderMonkey (used in Firefox), JavaScriptCore (used in Safari), and Chakra (used in older versions of Microsoft Edge).

What are Browser Tools?

Browser tools, also known as developer tools or web developer tools, are built-in features in web browsers that allow developers to inspect and debug web pages, monitor network activity, analyze performance, and test code. These tools are indispensable for web developers and designers during the development and debugging process. Here are some common browser tools and their functionalities:

Console: The Console tool is a JavaScript console that allows developers to execute JavaScript code directly in the browser, log messages, errors, and warnings, and debug JavaScript code. Developers can use the console for testing and debugging purposes, as well as for logging information during development.

Logging: The primary function of the console is to log messages, errors, and warnings generated by JavaScript code. Developers can use console.log(), console.error(), console.warn(), and console.info() to output information to the console.

```
console.log("This is a log message");
console.error("This is an error message");
console.warn("This is a warning message");
console.info("This is an info message");
```

Debugging: Developers can use the console for debugging purposes by inserting console.log() statements in their code to inspect the values of variables, objects, and expressions at different points during execution.

```
var x = 10;
console.log("The value of x is: ", x);
```

1. Syntax

Syntax refers to the set of rules that define how JavaScript code is written and interpreted. Here are some basic syntax rules:

- **Statements**: JavaScript code is made up of statements, usually ending with a semicolon (`;`).
- **Comments**: Use `//` for single-line comments and `/* */` for multi-line comments.

2. Variable declaration

Variables are used to store information. The information can be anything like cost of an item, username etc.

A variable is a data container or storage which holds a value in it.

Variables are used to store data values. In JavaScript, you can declare variables using var, let, or const.

- **var**: Function-scoped or globally scoped.
- **let**: Block-scoped.
- **const**: Block-scoped and cannot be reassigned.

Let userName = ‘Mohan’;

Once a variable is created you can assign some value to it and you can also reassign a value(sometimes).

let useName =”Mohan” // Value for this variable can be reassigned.
userName =”Ankit” // It is accepted for this variable.
const newName =”Mohan” // Value for this variable can not be reassigned.
newName = “Sohan” //can’t be done as it is a constant variable hence it cannot be changed once assigned a value.