A high-level overview of the JavaScript language

Generic definition:

- JavaScript is a high level object-oriented, multi-paradigm programming language.

Monster definition:

- JavaScript is a **high-level**, **prototype-based object-oriented**, **multi-paradigm**, interpreted or just-in-time compiled, **dynamic, single-threaded**, garbage-collected programming language **with first-class functions** and a **non-blocking event loop concurrency model**.

  (although this definition is just for fun, its always better to get the bigger picture before diving deep into the concepts)

  Deconstructing the Monster Definition

Deconstructing the Monster Definition

High level :

- Every computer program needs resource in order to run (like memory & cpu).
- Low level languages like "C" requires for us to manage these recourse **manually.** For example - by asking the computer/cpu memory in order to create a variable.
- However, high level languages like **JavaScript & Python** does NOT require users to manage those resources at all. These languages have some abstraction that takes such work (memory management) away from us. This makes the language easier to learn & use.
    - but they will never be as fast & as optimised as the C & similar low level level languages
- One of the tools that takes away garbage collection away from us is - **garbage collection.**

Garbage-Collected :

- It an algorithm in the JavaScript engine that removes all the unnecessary objects from the computer memory in order not to clog it up with unnecessary stuff.
    - Its like JavaScript having a cleaning guy that cleans the code on its own so that we don't have to do it manually🧹

Interpreter or just-in-time-compiled

- Computer only understands machine language i.e. 0s & 1s - but it is not possible and very complex for humans to break down every single instruction into 0s & 1s.
- For this, we have human readable programming languages which is an abstraction over the machine code (0s & 1s). These programming

languages need to be either **compiled or interpreted** for the machine to understand.

- In JavaScript, this translation happens inside the JavaScript engine.

Multi-paradigm

- JavaScript is called a **multi-paradigm** programming language because it allows different ways of thinking about and organizing code. Imagine it like a set of tools—you can choose the best tool for the job depending on the situation.

  Paradigm: An approach and mindset of structuring code, which will direct your coding style and technique.

  The main approaches (or paradigms) JavaScript supports are:

  1. **Procedural programming** – This is like following a set of instructions step by step, similar to a cooking recipe. It focuses on doing tasks in a specific order.
  2. **Object-oriented programming (OOP)** – This approach organizes code into "objects," which are like real-world things that have properties and behaviors. It's useful for structuring complex applications in a way that's easy to manage.
  3. **Functional programming** – This focuses on using functions to process data in a predictable way. Instead of changing things directly, it prefers to take input and return a new result, making code easier to understand and debug.

  Since JavaScript supports all these approaches, developers can choose the best one (or even combine them) based on their needs. This flexibility makes JavaScript a powerful language for different types of applications, from simple websites to complex web apps. 🚀

Protype-based object-oriented

Prototype-Based Object-Oriented Programming (OOP) (click)

First-class functions

- In a language with **first-class functions,** functions are simply treated as **variables.** We can pass them into other functions and return them from functions.

- Allows us to do functional programming.

Dynamic:

- Essentially means that we can type its expressions dynamically and change values.
- Also we don't have to mention datatypes. JavaScript assigns datatype dynamically on its own.
    - Example:-

```
let x = 23;
let y = 49;
x = 'sahil';  //here we change the value of x dynamically
console.log(`my name is ${x}`) // output: - My name is Sahil
                                              // here x dynamically entered usin
                                              // template literal
```

Single threaded & Non-blocking event loop concurrency model

- Concurrency model: Just a fancy term that means how the JavaScript engine handles multiple tasks happening at the same time.

  why do we need that?

  Because: JavaScript runs in one **single thread,** so it can do only one thing at a time.

  Therefore, we need a way of handling the multiple things that happen at the same time.

- In computing, **thread** is like a set of instruction that is executed in the computers cpu.
    - Basically, a thread is where our code is executed in the processor.

  So what about a long running task?

    - Apparently it would block the single thread. However, we want **non-blocking behaviour.**

How do we achieve that?

- By using an **event loop.**
    - It takes the **long running tasks** & executes them in the background and puts them back in the main thread once they are finished.

In a nutshell, this is JavaScript’s Non-blocking event-loop concurrency model with a single thread.

Prototype-Based Object-Oriented Programming (OOP)

Prototype-based OOP is a way of structuring code where **objects inherit from other objects**, rather than from rigid class structures.

How It Works:

1. **Objects are the primary building blocks** – Instead of defining classes first, you create objects directly.
2. **Prototypes act as a template** – Each object can have a prototype, which is another object it inherits properties and methods from.
3. **Prototype Chain** – If an object doesn't have a property or method, JavaScript looks up the prototype chain to find it.

Key Features of Prototype-Based OOP:

- **No need for classes** – Objects are created from other objects, making the system more flexible.
- **Dynamic inheritance** – You can modify an object or its prototype at runtime, changing behavior on the fly.
- **Memory efficiency** – Instead of each object having its own copy of methods, they share methods through the prototype.

Why JavaScript Uses This Model?

JavaScript was designed to be lightweight and dynamic for web development. The prototype-based system allows for greater flexibility and simpler inheritance compared to class-based systems like C++ or Java. Even though JavaScript now supports `class` syntax, it still works under the hood using prototypes.