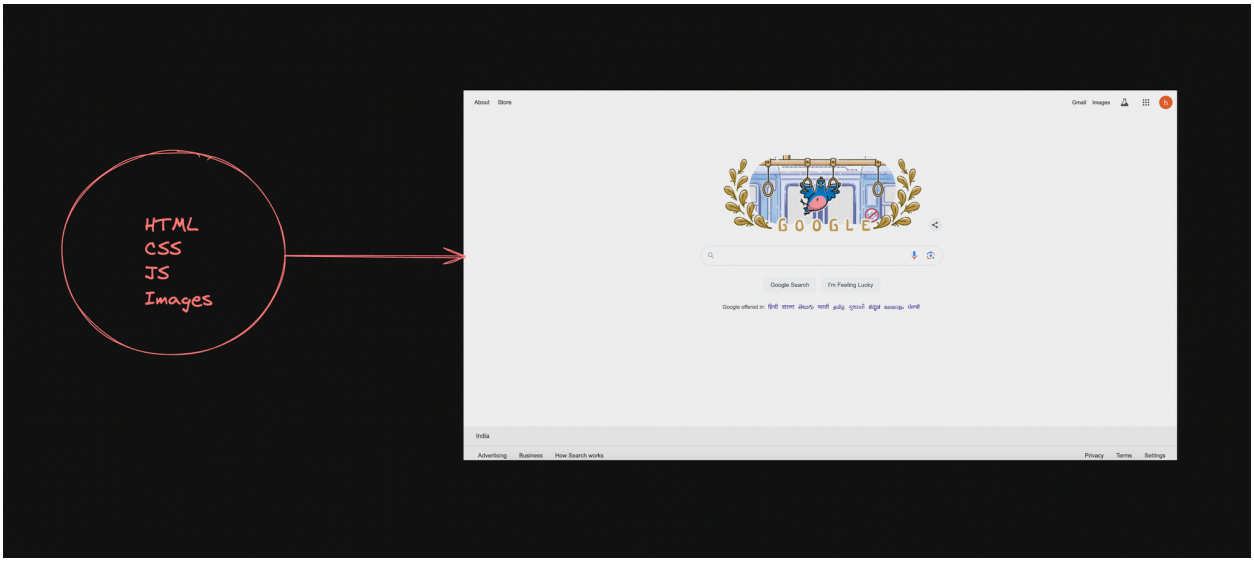
Javascript - The basics

Web development

Web development involves writing a lot of HTML, CSS and JS code.

Historically (and even today to some extend), browsers could only understand HTML, CSS and JS

Any website that you see, is a bunch of HTML, CSS and JS files along with some assets (images, videos etc)



Facts/Callouts

1. React, NextJS are `frameworks` . They compile down to HTML, CSS, JS in the end. That is what your browser understands.
2. When you run your C++ code on `leetcode` , it does not run on your browser/machine. It runs somewhere else. Your browser can't (almost) compile and run C++ code.

Before we proceed, do one of the following -

1. Install Node.js locally
2. Keep your browser console open for testing locally

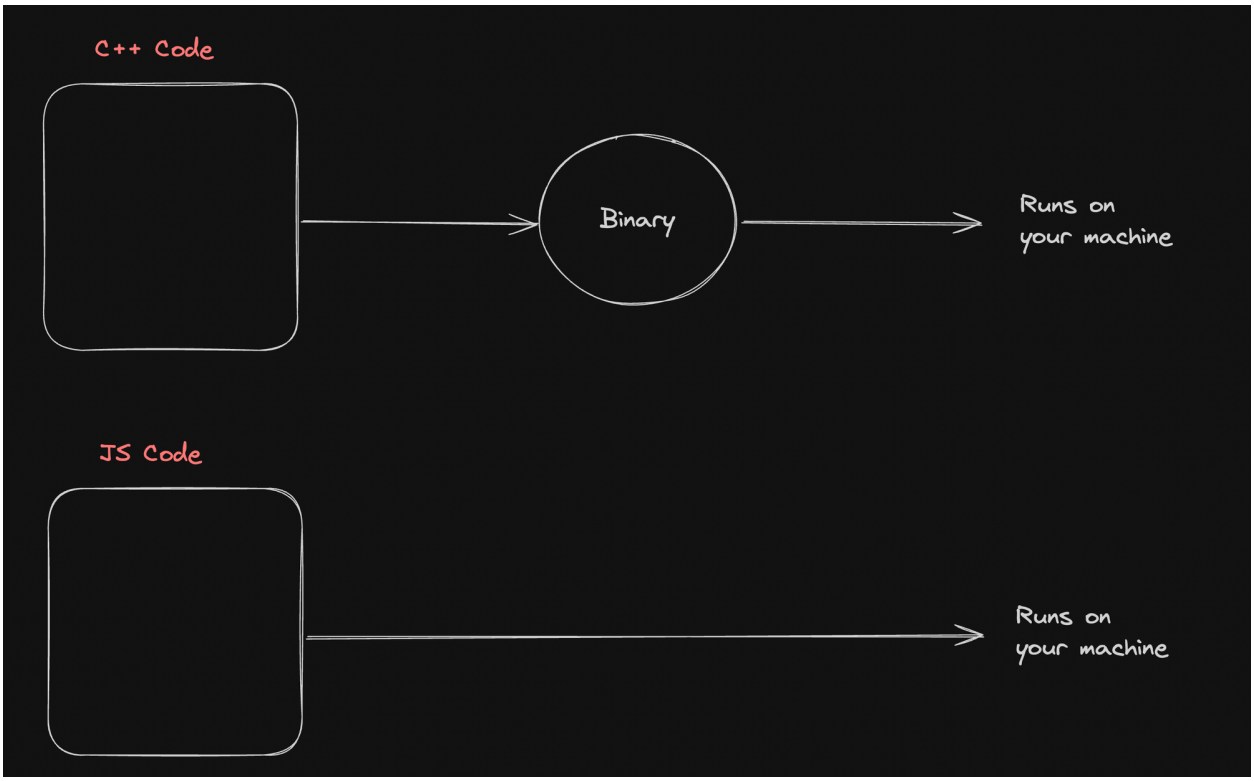
Properties of JS

Every language comes with it’s unique set of features.

Javascript has the following -

1. Interpreted

JavaScript is an interpreted language, meaning it's executed line-by-line at runtime by the JavaScript engine in the browser or server environment, rather than being compiled into machine code beforehand.



Upsides -

1. There is one less step to do before running your code

Downsides -

1. Performance Overhead:
2. More prone to runtime errors

2. Dynamically Typed

Variables in JavaScript are not bound to a specific data type. Types are determined at runtime and can change as the program executes

C++ Code (won't compile)

```
#include <iostream>

int main() {
  int a = 1;
  a = "hello";
  a = true;
}
```

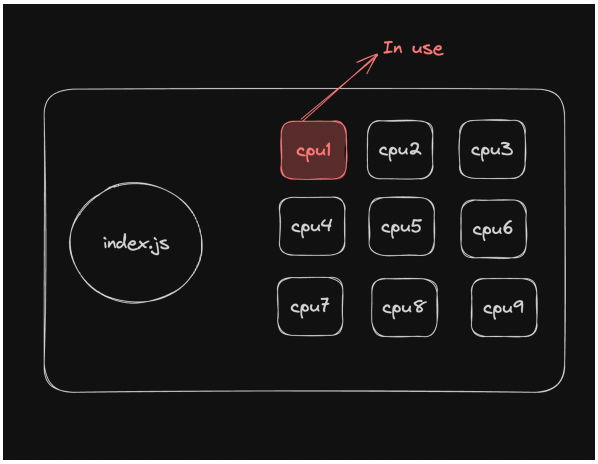
JS Code (will compile)

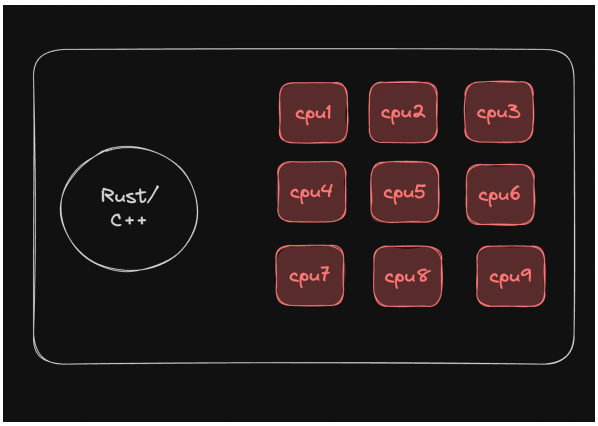
```
var a = 1;
a = "hello";
a = true;

console.log(a)
```

3. Single threaded

JavaScript executes code in a single-threaded environment, meaning it processes one task at a time.





4. Garbage collected

JavaScript automatically manages memory allocation and deallocation through garbage collection, which helps prevent memory leaks by automatically reclaiming memory used by objects no longer in use.

Syntax of Javascript

1. Variables

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

```
let name = "John";     // Variable that can be reassigned
const age = 30;        // Constant variable that cannot be re
assigned
var isStudent = true;  // Older way to declare variables, fun
ction-scoped
```

2. Data types

```
let number = 42;              // Number
let string = "Hello World";   // String
```

```
let isActive = false;        // Boolean
let numbers = [1, 2, 3];     // Array
```

3. Operators

```
let sum = 10 + 5;          // Arithmetic operator
let isEqual = (10 === 10); // Comparison operator
let isTrue = (true && false); // Logical operator
```

4. Functions

```
// Function declaration
function greet(name) {
    return "Hello, " + name;
}

// Function call
let message = greet("John"); // "Hello, John"
```

Primitive types

1. number
2. string
3. boolean

Complex types

1. Arrays
2. Objects

Arrays

Arrays let you group data together

```
const users = ["alice", "bob", "david"];
const tatalUsers = users.length;
```

```
const firstUser = users[0];
```

💡 map, filter, reduce

Objects

An object in JavaScript is a collection of `key-value pairs`, where each `key` is a string and each `value` can be any valid JavaScript data type, including another object.

```
let user = {
    name: "Harry",
    age: 19
}

console.log("Harry's age is " + user.age);
```

Object of Objects

We can have an even more complex object (object of objects)

```
const user1 = {
    name: "harry",
    age: 19,
    address: {
        city: "San Jose",
        country: "US",
        address: "1122 Minor Ave"
    }
}

const city = user1.address.city;
```

Classes

In JavaScript, classes are a way to define blueprints for creating objects (these objects are different from the objects defined in the last section).

For example

```
class Rectangle {
   constructor(width, height, color) {
        this.width = width;
        this.height = height;
        this.color = color;
   }

   area() {
       const area = this.width * this.height;
         return area;
   }

   paint() {
            console.log(`Painting with color ${this.color}`);
   }

}

const rect = new Rectangle(2, 4)
const area = rect.area();
console.log(area)
```

Key Concepts

1. **Class Declaration**:
   - You declare a class using the `class` keyword.
   - Inside a class, you define properties (variables) and methods (functions) that will belong to the objects created from this class.
2. **Constructor**:

- A special method inside the class that is called when you create an instance (an object) of the class.
- It's used to initialize the properties of the object.

3. **Methods**:
   - Functions that are defined inside the class and can be used by all instances of the class.
4. **Inheritance**:
   - Classes can inherit properties and methods from other classes, allowing you to create a new class based on an existing one.
5. **Static Methods**:
   - Methods that belong to the class itself, not to instances of the class. You call them directly on the class.
6. **Getters and Setters**:
   - Special methods that allow you to define how properties are accessed and modified.

Inheritance in classes

Inheritance in JavaScript classes allows one class to inherit properties and methods from another class. This mechanism enables code reuse, making it easier to create new classes that are based on existing ones, without having to duplicate code.

Create a Circle class

```
class Circle {
   constructor(radius, color) {
       this.radius = radius;
       this.color = color;
   }

   area() {
```

```
        const area = this.radius * this.radius * Math.PI;
         return area;
    }

    paint() {
            console.log(`Painting with color ${this.color}`);
    }

}

const circle = new Circle(2, "red")
const area = circle.area();
console.log(area)
```

💡 Can you see there is code repetition here and in the Rectangle class?

Create a base shape class

Base class

```
class Shape {
    constructor(color) {
        this.color = color;
    }

    paint() {
            console.log(`Painting with color ${this.color}`);
    }

    area() {
        throw new Error('The area method must be implemented
in the subclass');
    }
```

```
    getDescription() {
        return `A shape with color ${this.color}`;
    }
}
```

Rectangle class

```
class Rectangle extends Shape {
    constructor(width, height, color) {
        super(color);  // Call the parent class constructor t
o set the color
        this.width = width;
        this.height = height;
    }

    area() {
        return this.width * this.height;
    }

    getDescription() {
        return `A rectangle with width ${this.width}, height
${this.height}, and color ${this.color}`;
    }
}
```

Circle class

```
class Circle extends Shape {
    constructor(radius, color) {
        super(color);  // Call the parent class constructor t
o set the color
        this.radius = radius;
    }

    area() {
        return Math.PI * this.radius * this.radius;
```

```
    }

    getDescription() {
        return `A circle with radius ${this.radius} and color
${this.color}`;
    }
}
```

Some more classes

Date

```
const now = new Date(); // Current date and time
console.log(now.toISOString()); // Outputs the date in ISO fo
rmat
```

Maps

```
const map = new Map();
map.set('name', 'Alice');
map.set('age', 30);
console.log(map.get('name'));
```