Students notes

Intro to JS

How does the computer work?

Computers function through a series of processes involving hardware and software. The primary components include the central processing unit (CPU), memory, storage, and input/output devices. When a computer is turned on, the CPU executes instructions from the operating system, applications, and user inputs, manipulating data stored in memory.

What is a computer program?

A computer program is a set of instructions written in a programming language to perform specific tasks. To create and execute these programs, developers use a coding environment. For JavaScript, the coding environment includes a JavaScript engine. In web development, browsers handle the JavaScript engine, and each browser has its own engine.

JavaScript Compilation Process

1. **Compiler/Parser:**
    - Checks each line of code for syntax errors.
    - Enforces rules; errors result in the code not running.
2. **Converting to Machine Code:**
    - If the code passes checks, it is converted to machine code.
3. **Different JavaScript Engines:**
    - Browsers use distinct engines (e.g., V8 in Chrome, SpiderMonkey in Firefox).

- Mobile phones may have different JavaScript engines, affecting code execution speed.

JavaScript

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.

While HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

What is a scripting language?

> *Scripting language is like a sub-category of programming languages which will connect one language to another which is crucial for developing websites. Difference between scripting and programming languages is the role of a compiler. Programming languages will be compiled in one go whereas scripting languages will be interpreted line by line.*

JavaScript Engine:

The JavaScript code that we write can not be understood by the computer

A JavaScript Engine is a program that converts JavaScript code into machine code that allows a computer to perform a specific task.

The JavaScript engines are developed by web browser vendors.

JS Engine	Developed By
V8	Google for Chrome (open-source)
SpiderMonkey	Mozilla Firefox
JavaScriptCore	Apple for Safari (open-source)
Chakra	Microsoft Edge(the latest version of Edge uses V8)

Let's look at the 2 ways of executing js code

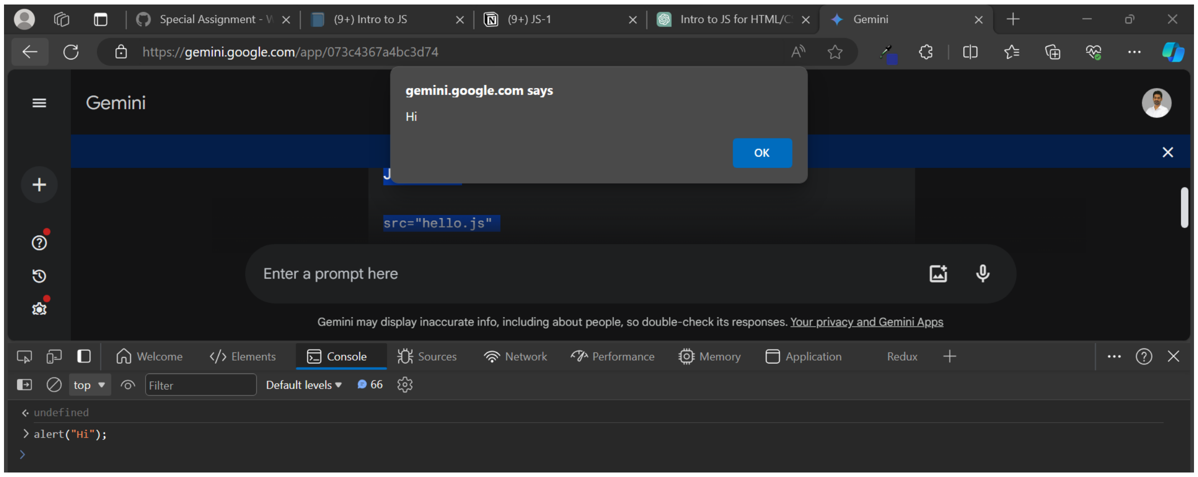
In a Web Browser

When you run JavaScript in a web browser:

- *The browser has a built-in JavaScript engine (like V8 in Chrome or SpiderMonkey in Firefox).*
- *You can write your code in a separate `.js` file and link it to your HTML, or directly paste it into the browser console.*

1. **Create a file with .js extension. Write Your JavaScript Code** that displays an alert message:

```
alert("Hello, world!");
```



https://gemini.google.com/app/073c4367a4bc3d74

gemini.google.com says

Hi

OK

src="hello.js"

Enter a prompt here

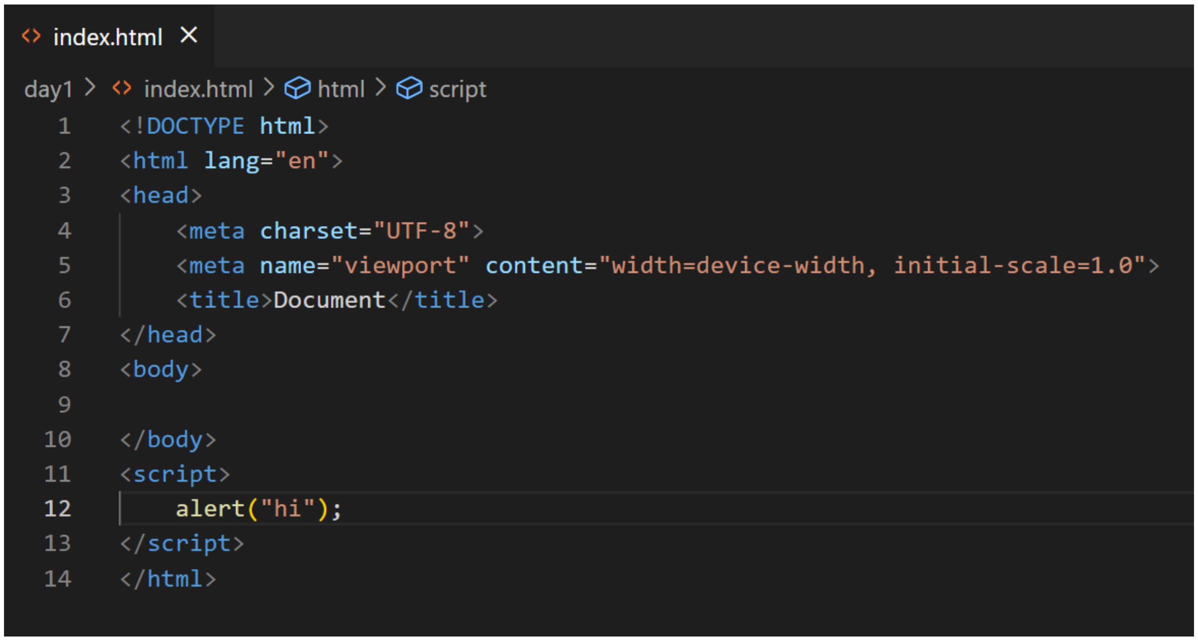
Gemini may display inaccurate info, including about people, so double-check its responses. Your privacy and Gemini Apps

```
< undefined
> alert("Hi");
>
```

1. Create an HTML File (Optional):
Keep your JS code separate in an HTML file for better organization and maintainability.

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
</head>
<body>
    <h1>Welcome!</h1>
    <script src="hello.js"></script>  </body>
</html>
```

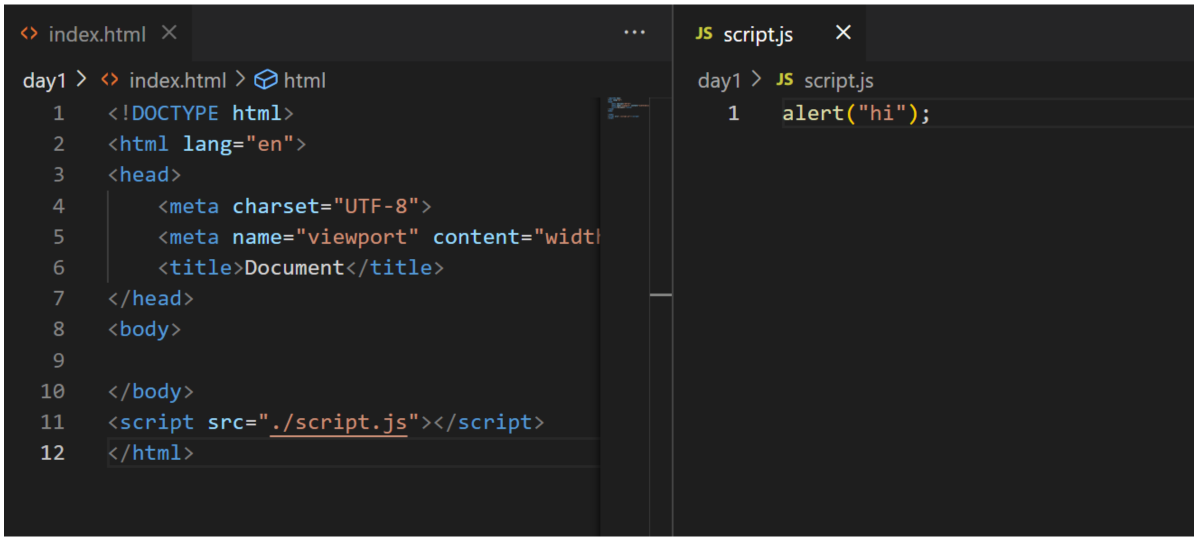
2. **Run the Code in a Browser:** Create an HTML file, simply open it in your web browser. The browser will automatically parse the HTML and execute any linked JavaScript code.
   Add your js inside the script tag.



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
<script>
    alert("hi");
</script>
</html>
```

You can also create a different .js file and add it using `src` attribute in the script tag.



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="widt
    <title>Document</title>
</head>
<body>

</body>
<script src="./script.js"></script>
</html>
```

```js
alert("hi");
```

Using Node.js

When you run JavaScript using Node.js:

Node.js provides a runtime environment for executing JavaScript code outside the browser.

JavaScript Runtime:

- *JavaScript runtime is an environment that provides all the necessary components to run a JavaScript program.*
- *Every browser has a JavaScript Engine*
- *A JavaScript Engine is one component in the JavaScript Runtime*

Node.js is an open-source, cross-platform JavaScript runtime environment. that helps us to run JavaScript outside the browser. It is not a language, it is not a framework. It can execute not only the standard ECMAScript but also new features made available through C++ binding using the V8 engine.

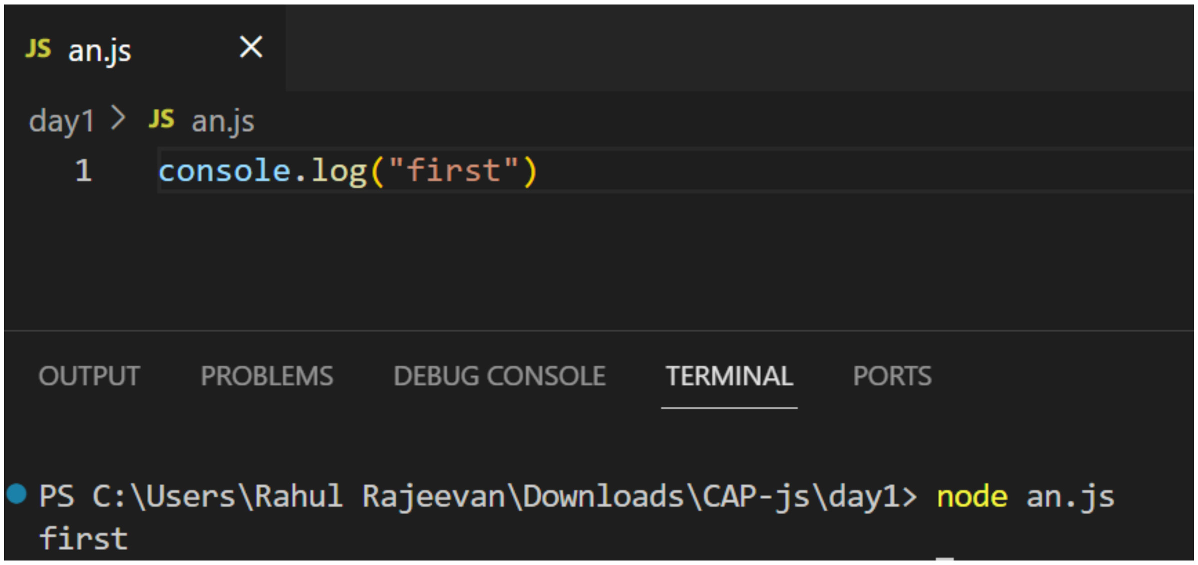
Open-source: source code is publically available for sharing and modification.

Cross-platform: available for Mac, Windows, and Linux.

1. **Install Node.js:** If you don't have Node.js installed, download and install it from the official Node.js website https://nodejs.org/en. Node.js includes an environment for running JavaScript code outside of a web browser.
2. **Write Your JavaScript Code:** Similar to the web browser approach, create a new file in your code editor and write your JavaScript code.
3. **Save the File:** Save your JavaScript code with a `.js` extension (e.g., `myscript.js`).
4. **Run the Code from the Command Line:** Open a terminal or command prompt window and navigate to the directory where you saved your JavaScript file.

```
//to run the code

node myscript.js
```



```
console.log("first")
```

```
PS C:\Users\Rahul Rajeevan\Downloads\CAP-js\day1> node an.js
first
```

Variables

What is a variable?

A variable is a storage container with an assigned name that holds data. It represents a value that can vary or change during program execution.

Purpose of Variables

Variables serve as storage locations with assigned names, holding data for future use. The syntax for declaring and using variables in JavaScript includes keywords like `let`, `const`, and `var`.

Variable Declaration Examples

```
// Using Var
var a;          // Declaring a variable.
a = 10;         // Assigning a value to the variable.
console.log(a); // Running the code.

// Shorthand for declaration and assignment
var a = 10;
console.log(a);
```

Data Types

1. **Number:**
    - Represents numeric values.
2. **String:**
    - Represents textual data. Always enclosed in quotes (`""` or `''`).

    ```
    var name = "gohan";
    console.log(name);
    ```

Checking Data Types

```
var x = 12;
console.log(typeof(x)); // Output: number

var y = "12";
console.log(typeof(y)); // Output: string
```

Understanding these concepts is fundamental for anyone diving into JavaScript and programming in general.

- variables rules
    - A variable name cannot start with a digit
        - ex 123
    - a variable cannot start with any symbol
        - ex @#
            - exception for $name and _name
    - variable names should be self-explanatory.
    - A variable name must start with a letter or an underscore character (_)
    - A variable name can only contain alpha-numeric characters and underscores (`a-z, A-Z`, `0-9`, and `_`)
        - ex- abc12 or ab12c.
    - Variable names are case-sensitive (age, Age and AGE are three different variables)

- variable **Declaration, Assigning, and Printing**.

```
var x; //declaration

x=10; //assigning

console.log(x); //printing to console/shell.

//I can also do
```

```
var x=10; //This is basically declaration and assigning togethe
```

Camel Case

Each word, except the first, starts with a capital letter:

myVariableName = "John"

Snake Case

Each word is separated by an underscore character:

my_variable_name = "John"

Student Task

- create a variable of their name, age, and gender.

Mathematical Operators

The common operators in Javascript are :

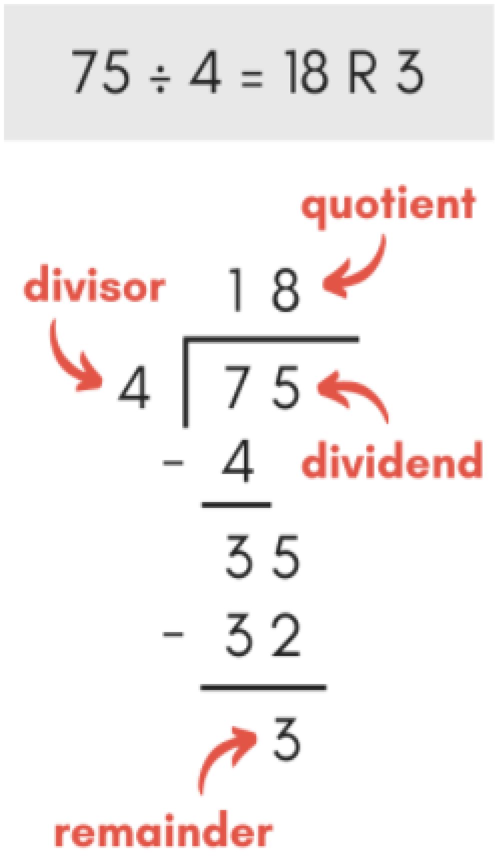
- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

few examples.

```
var a=4;
var b=7;

console.log("The sum is ", a+b);
console.log("The Difference is ", a-b);
console.log("The Multiplication is ", a*b);
console.log("The Division is ", a/b);
```

Modulo Operator (%)



- When we divide a number by another, there are 4 terms that we need to understand
    - quotient
    - divisor
    - dividend
    - remainder
- Looking at the above picture, you will get clear about the above terms
- The remainder represents by %.
    - For Eg : var result = 75 % 4 [Read as 75 Modulus 4]

Exponentiation Operator (**)

- This basically calculates the **Power** of something.

```
var a=2;
var b=3;

console.log(a**b); // 8

var x=2;
var y=2;
var z=x**y;

console.log(z); // 4
```

String Concatenation (Joining)

- Concatenation means a series of interconnected things.
- Use + to join two or more string

Example 1 :

```
var name = "Akhila";
console.log("Hello "+name+" Welcome To chitkara");
```

Example 2 :

```
var a = "Masai School";
var line = a+" "+b;
console.log(line);
```

```
var a="Hello";
var b="World";

var x=a+b;
console.log(x); // HelloWorld

console.log(a+" "+b) //Hello World
```

Boolean Datatype

- It can only contain **true** or **false.**

```
var x=true;

console.log(x); // true
console.log(typeof(x));  // boolean
```

Type Coercion:

Type coercion refers to the **automatic conversion of values from one data type to another, typically performed during operations or comparisons involving different data types**.

string + number= string

number +number= number

string + string = string.

```
// The Number 10 is converted to
// string '10' and then '+'
// concatenates both strings
let x = 10 + '20';
let y = '20' + 10;

// The Boolean value true is converted
// to string 'true' and then '+'
// concatenates both the strings
let z = true + '10';

console.log(x);  // 1020
console.log(y);  // 2010
console.log(z);  // true10
```

Expression

Anything that evaluates to a value.

```
4>5 //false
3*4 //12
`$(5>6] is an expression`
```

Statement

Assigned to a variable.

```
var t=`$(5>6] is answer`;
let t=10+5;
```

Relational Operators or Comparison Operators

- There are 8 types of relational operators.

S.No	Operator	Description
1	>	Greater than
2	≥	Greater than or equal to
3	<	Less than
4	≤	Less than or equal to

- These take Two inputs and print the output as Boolean.

1. **> (Greater than)**

    - It will give me **true** if the **first value is strictly greater than the second one**.
    - It will give you **false** when the **first value is less than or equal to the second value**.

```
// check if Goku height is greater than vegeta or not.

var goku_height= 6;
var vegeta_height= 5;
console.log(shiva_height>jai_height); // true ;
```

2. **≥ (Greater than equal to)**

- It will give you **true** if the first value is **greater than or equal to the second value**.
- It will give me **false** if the first value is **less than the second value**.

```
// case 1
6>=5 // true
4>=4 // true

//case 2
3>=10 // false

//Goku Example
var goku_marks=35;
var passing_marks=35;
var is_passed=goku_marks>=passing_marks;

console.log(is_passed); // true
```

- **inform them when to use > and when to use ≥.**

3. **< (Less than)**

- It will give you **true** if the first value is **strictly less than the second value.**
- It will give you **false** if the first value is **greater than or equal to the second value.**

```
console.log(5<5); // false
console.log(3<5); //true
console.log(15<8); // false
```

4. **≤ (Less than equal to)**

- It will give you **true** if the first value is **strictly less than or equal to the second value.**
- It will give you **false** if the first value is **greater than the second value.**

```
// case 1
3<=4 // true
5<=5 // true
```

```
//case 2
5<=1 // false
```

S.no	operator	Description
5	==	Double Equal to
6	!=	Not Equal to
7	===	Triple equal to
8	!==	Not double equal to

- **They can be applied to both numbers and strings.**
- **the output will be Boolean**

  == (Double Equal to)

  - It gives **true** if the **first value is equal to the second value.**
  - It gives a **false** if the **first value is not equal to the second value.**

  ```
  console.log(2==2); // true
  console.log("chunnu"=="chunnu"); // true
  console.log(2==5); // false
  console.log("Chunnu"=="chunnu"); //false
  ```

  != (Not Equal to)

  - Opposite of ==.
  - It gives **true** if both the values are **not equal**.
  - It gives **false** if both the values are **equal**.

  ```
  console.log(2!=2); // false
  console.log("chunnu"!="chunnu"); // false
  ```

```
console.log(2!=5); // true
console.log("Chunnu"!="Chunnu"); // true
```

===(Triple Equal to)

- It gives **true** if the **first value is equal to the second value and also their datatypes are the same.**
- It gives a **false** if the **first value is not equal to the second value or their datatypes are different.**
- **If we check string "2" and number 2 with "==", it will give true as the output, and to overcome this we use "===".**
- **It also checks the data types of the operands.**

```
// if i use ==

console.log("2"==2); // true (not checking the datatype, only

// if i use ===

console.log("2"===2); // false (also checking the datatype al
```

!== (Not Double Equal to)

- Opposite of ===.
- It gives **true** if both the values or datatypes are **not equal**.
- It gives **false** if both the values and datatypes are **equal**.

```
// if i use !==

console.log("2"!==2); // true (also checking the datatype alo

consoele.log("3"!=="2") // true;
```

```
console.log("2"!=="2")// false
```