JavaScript Basics — Syntax, Examples, Output, and Detailed Explanation

Run JavaScript Using HTML File

Best for:

- DOM examples
- Buttons
- Forms
- Website interaction

Step-by-Step

Step 1: Open Text Editor

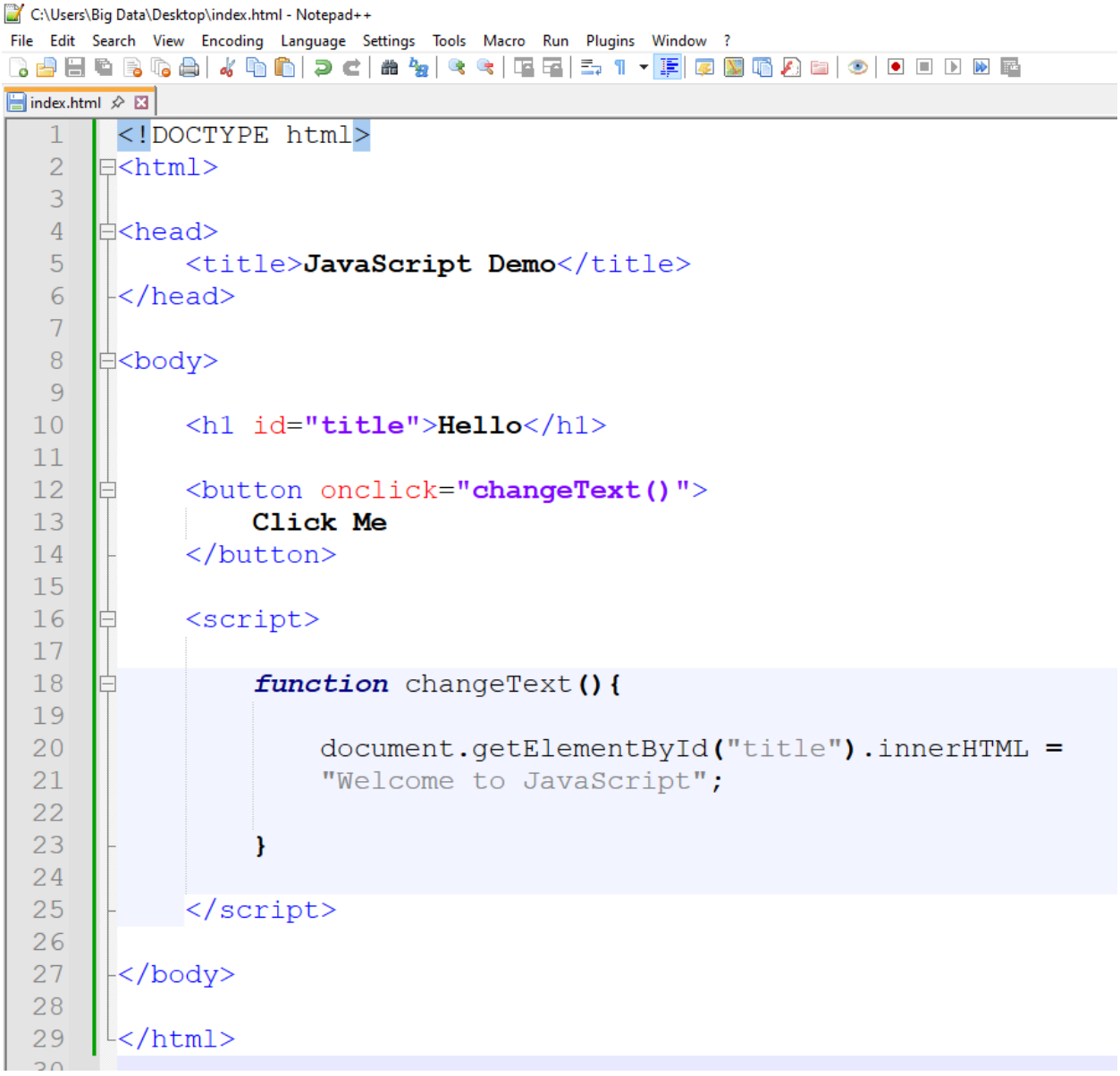
Use:

- Visual Studio Code
- Notepad++
- Sublime Text

Step 2: Create File

Create file:

`index.html (As Shown Below)`



Step 3: Paste This Code

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Demo</title>
</head>

<body>

    <h1 id="title">Hello</h1>

    <button onclick="changeText()">
        Click Me
    </button>

    <script>

        function changeText(){

            document.getElementById("title").innerHTML =
            "Welcome to JavaScript";

        }

    </script>

</body>

</html>
```

Step 4: Save File

Save as:

```
index.html
```

Step 5: Open in Browser

Double-click the file.

OR

Right click → Open With → Chrome

Output

Initially:

```
Hello
```

After button click:

```
Welcome to JavaScript
```

---

2. Run JavaScript Using Node.js

Best for:

- Backend JavaScript
- Practice
- Learning core JavaScript

Install Node.js

Download:

https://nodejs.org/en/download

Install LTS version.

Verify Installation

Open terminal or command prompt:

```
node -v
```

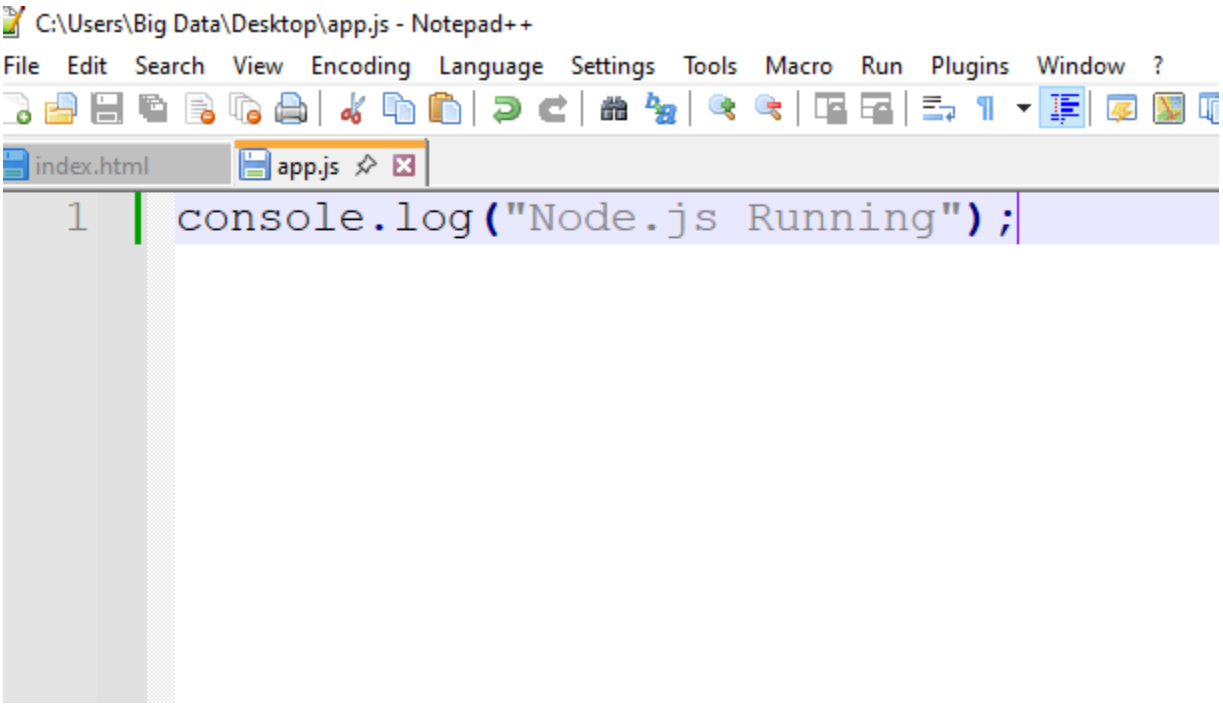
Output Example

v22.0.0

Create JavaScript File((in notepad++ or in vs code)

Example:

app.js
as shown below



Add Code

```
console.log("Node.js Running");
```

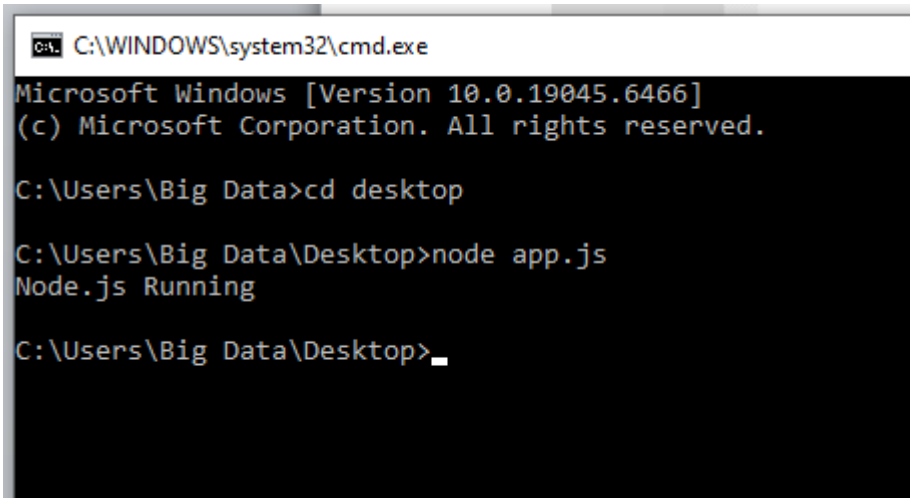
Run File

Open terminal in folder (command prompt in windows pc).

Run:

```
node app.js
```

As shown below



Output

```
Node.js Running
```

---

1. JavaScript Data Types

JavaScript has **Primitive** and **Non-Primitive** data types.

Primitive Data Types

Type	Example
String	`"Hello"`

Type	Example
Number	`100`
Boolean	`true`
Undefined	`undefined`
Null	`null`
BigInt	`123n`
Symbol	`Symbol("id")`

Example

```
let name = "Rahul";       // String
let age = 25;             // Number
let isStudent = true;     // Boolean
let city;                 // Undefined
let emptyValue = null;    // Null

console.log(name);
console.log(age);
console.log(isStudent);
console.log(city);
console.log(emptyValue);
```

Output

```
Rahul
25
true
undefined
null
```

Explanation

- `"Rahul"` → text value → String
- `25` → numeric value → Number
- `true` → logical value → Boolean
- Variable without value → Undefined
- `null` means intentionally empty

Non-Primitive Data Types

Type	Example

Type	Example
Object	`{name:"Rahul"}`
Array	`[1,2,3]`
Function	`function(){}`

---

2. JavaScript Operators

Operators are used to perform operations on values.

A. Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication

Operator	Meaning
/	Division
%	Modulus
**	Power

Example

```
let a = 10;
let b = 3;

console.log(a + b);
console.log(a - b);
console.log(a * b);
console.log(a / b);
console.log(a % b);
console.log(a ** b);
```

Output

```
13
7
30
3.3333333333333335
1
1000
```

B. Comparison Operators

Operator	Meaning
==	Equal
===	Strict Equal
!=	Not Equal
>	Greater
<	Less

Example

```
console.log(5 == "5");
console.log(5 === "5");
console.log(10 > 5);
```

Output

```
true
false
true
```

Explanation

- == checks value only
- === checks value + datatype

C. Logical Operators

Operator	Meaning
&&	AND
`	
!	NOT

Example

```
let age = 20;

console.log(age > 18 && age < 30);
console.log(age < 18 || age > 10);
console.log(!(age > 18));
```

Output

```
true
true
false
```

---

3. if, else if, else

Used for decision making.

Syntax

```
if(condition){
    // code
}
```

```
else if(condition){
    // code
}
else{
    // code
}
```

Example

```
let marks = 75;

if(marks >= 90){
    console.log("Grade A");
}
else if(marks >= 60){
    console.log("Grade B");
}
else{
    console.log("Fail");
}
```

Output

```
Grade B
```

Explanation

- Condition checks from top to bottom
- First true condition executes

---

4. switch Statement

Used when multiple conditions exist.

Syntax

```
switch(expression){
    case value:
        // code
        break;

    default:
        // code
}
```

Example

```
let day = 2;

switch(day){
    case 1:
        console.log("Monday");
        break;

    case 2:
        console.log("Tuesday");
        break;

    case 3:
        console.log("Wednesday");
```

```
        break;

    default:
        console.log("Invalid Day");
}
```

Output

```
Tuesday
```

Explanation

- `break` stops further checking
- `default` runs if no case matches

---

5. Arrays

Array stores multiple values in one variable.

Syntax

```
let arrayName = [value1, value2];
```

Example

```
let fruits = ["Apple", "Banana", "Mango"];

console.log(fruits);
console.log(fruits[0]);
console.log(fruits[2]);
```

Output

```
["Apple", "Banana", "Mango"]
Apple
Mango
```

Array Methods

push()

Adds element at end.

```
fruits.push("Orange");

console.log(fruits);
```

Output

```
["Apple", "Banana", "Mango", "Orange"]
```

---

pop()

Removes last element.

```
fruits.pop();

console.log(fruits);
```

Output

```
["Apple", "Banana", "Mango"]
```

---

6. Functions

Functions are reusable blocks of code.

Syntax

```
function functionName(){
    // code
}
```

Example

```
function greet(){
    console.log("Welcome");
}

greet();
```

Output

```
Welcome
```

Function with Parameters

```
function add(a, b){
    return a + b;
}

console.log(add(5, 3));
```

Output

```
8
```

Explanation

- `a` and `b` are parameters
- `return` sends value back

---

7. Arrow Functions

Shorter syntax for functions.

Syntax

```
const functionName = () => {
    // code
}
```

Example

```
const greet = () => {
    console.log("Hello World");
};

greet();
```

Output

```
Hello World
```

Arrow Function with Parameters

```
const multiply = (a, b) => {
    return a * b;
};

console.log(multiply(4, 5));
```

Output

```
20
```

Short Form

```
const square = n => n * n;

console.log(square(5));
```

Output

```
25
```

---

8. Objects

Objects store data in key-value pairs.

Syntax

```
let objectName = {
    key: value
};
```

Example

```
let student = {
    name: "Rahul",
    age: 21,
    course: "BCA"
};

console.log(student.name);
console.log(student.age);
```

Output

```
Rahul
21
```

Explanation

- `name`, `age`, `course` are properties
- Access using dot `.` operator

---

9. Classes

Classes are templates for creating objects.

Syntax

```
class ClassName{
    constructor(){
    }
}
```

Example

```
class Car {

    constructor(brand, model){
        this.brand = brand;
        this.model = model;
    }

    display(){
        console.log(this.brand + " " + this.model);
    }
}

let car1 = new Car("Toyota", "Fortuner");

car1.display();
```

Output

```
Toyota Fortuner
```

Explanation

constructor()

Automatically runs when object is created.

this keyword

Refers to current object.

new keyword

Creates object from class.

---

10. DOM (Document Object Model)

DOM allows JavaScript to interact with HTML.

Example 1 — Change Text

HTML

```
<h1 id="title">Hello</h1>

<button onclick="changeText()">Click</button>
```

JavaScript

```
function changeText(){
    document.getElementById("title").innerHTML = "Welcome";
}
```

Output

Before Click:

```
Hello
```

After Click:

```
Welcome
```

Explanation

Part	Meaning
`document`	Entire webpage
`getElementById()`	Select HTML element
`innerHTML`	Change content

Example 2 — Change CSS using DOM

HTML

```
<p id="para">JavaScript DOM</p>

<button onclick="changeStyle()">Change Style</button>
```

JavaScript

```
function changeStyle(){

    let element = document.getElementById("para");

    element.style.color = "red";
```

```
    element.style.fontSize = "30px";
}
```

Output

- Text color becomes red
- Font size becomes 30px

Example 3 — Input Box DOM

HTML

```
<input type="text" id="username">

<button onclick="showName()">Show</button>

<p id="result"></p>
```

JavaScript

```
function showName(){

    let name = document.getElementById("username").value;
```

```
    document.getElementById("result").innerHTML = name;
}
```

Output

If user types:

```
Rahul
```

Then output becomes:

```
Rahul
```

Important JavaScript Concepts Summary

Topic	Purpose
Data Types	Store values
Operators	Perform operations
if else	Decision making
switch	Multiple conditions
Arrays	Store multiple values
Functions	Reusable code
Arrow Functions	Short functions
Objects	Key-value data
Classes	Object blueprint
DOM	Control webpage

Complete Mini Example

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Example</title>
</head>

<body>

    <h1 id="heading">Hello</h1>

    <button onclick="changeText()">Click Me</button>

    <script>

        function changeText(){

            let message = "Welcome to JavaScript";

            document.getElementById("heading").innerHTML = message;
        }

    </script>

</body>
</html>
```

Output

- Initially shows `Hello`
- After button click:
  `Welcome to JavaScript`

Best Practices

- Use `let` and `const` instead of `var`
- Use meaningful variable names
- Keep functions small
- Use `===` instead of `==`
- Comment complex code

Difference Between Function and Arrow Function

Normal Function	Arrow Function
Uses `function` keyword	Uses `=>`
Has own `this`	No own `this`
Traditional syntax	Short syntax