JavaScript Essentials - The Complete Guide by Arun Raj

Arun Raj - The Complete Guide to JS

---

1. Introduction to JavaScript

1.1 What is JavaScript?

- JavaScript is a scripting language used to create dynamic and interactive content on websites.
- It runs on the browser and works alongside HTML and CSS.

Example:
html

```
<script>
  alert("Hello, World!");
</script>
```

-

1.2 Role of JavaScript in Web Development

- Adds interactivity (e.g., form validation, animations).
- Updates and manipulates HTML and CSS dynamically.
- Handles asynchronous operations (e.g., fetching data from servers).

1.3 Setting Up the Environment

- Use **Browser Console** (e.g., Chrome Developer Tools).
- Use a **Code Editor** like Visual Studio Code.

Example (Browser Console):

1. Open DevTools (F12 in Chrome).

Type:
javascript

```
console.log("Welcome to JavaScript!");
```

1.4 JavaScript Syntax and Structure

- JavaScript statements end with a semicolon (`;`).
- Code blocks are defined using `{ }`.

Example:
javascript
```
if (true) {
  console.log("This is a code block.");
}
```

---

2. Basics of JavaScript

2.1 Variables and Constants

- `var`: Global or function-scoped.
- `let`: Block-scoped.
- `const`: Block-scoped, cannot be reassigned.

Example:
javascript
```
let name = "John";
const age = 25;
var country = "India";
```

2.2 Data Types

- **Primitive**: String, Number, Boolean, Undefined, Null.
- **Non-Primitive**: Object, Array, Function.

Example:
javascript
```
let isStudent = true; // Boolean
let score = 95; // Number
let user = { name: "Alice", age: 20 }; // Object
```

2.3 Operators

Arithmetic: `+`, `-`, `*`, `/`, `%`.
javascript
Copy code
```
let sum = 5 + 10; // 15
```

Assignment: `=`, `+=`, `-=`.
javascript
```
let x = 10;
x += 5; // 15
```

Comparison: `==`, `===`, `<`, `>`.
javascript
```
console.log(10 === "10"); // false
```

Logical: `&&`, `||`, `!`.
javascript
```
console.log(true && false); // false
```

2.4 Conditional Statements

`if`, `else`, `else if`:
javascript

```
let age = 18;
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

-

`switch`:
javascript
```
let color = "red";
switch (color) {
  case "red":
    console.log("Stop!");
    break;
  default:
    console.log("Go!");
}
```

2.5 Loops

`for`:
javascript
```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```
`while`:
javascript
```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

---

3. Functions in JavaScript

3.1 Defining and Calling Functions

Example:
javascript
```
function greet() {
  return "Hello!";
}
console.log(greet());
```

3.2 Function Parameters and Return Values

Example:
javascript
```
function add(a, b) {
  return a + b;
}
console.log(add(5, 3)); // 8
```

3.3 Arrow Functions

Example:
javascript
```
const multiply = (x, y) => x * y;
console.log(multiply(4, 5)); // 20
```

3.4 Callback Functions

```
function processInput(input, callback) {
  callback(input.toUpperCase());
}
processInput("hello", console.log);
```

3.5 IIFE (Immediately Invoked Function Expression)

Example:
javascript
```
(function () {
  console.log("This runs immediately!");
})();
```

---

4. JavaScript Objects

4.1 Introduction to Objects

- Objects store key-value pairs.

Example:
javascript
```
let car = { brand: "Tesla", model: "Model X" };
```

4.2 Object Properties and Methods

Example:
javascript
```
let person = {
  name: "Alice",
  greet: function () {
    return `Hello, ${this.name}`;
  },
};
console.log(person.greet());
```

4.3 Object Destructuring

Example:
javascript
```
let { brand, model } = car;
console.log(brand); // Tesla
```

4.4 Object-Oriented Programming Concepts

4.4.1 Classes and Constructors

Example:
javascript
```
class Animal {
  constructor(name) {
    this.name = name;
  }
}
let dog = new Animal("Buddy");
```

4.4.2 Prototypes and Inheritance

Example:
javascript
```
function Animal(name) {
  this.name = name;
}
Animal.prototype.speak = function () {
  return `${this.name} makes a sound.`;
};

let dog = new Animal("Buddy");
```

4.3 Object Destructuring

Object destructuring allows you to extract properties from objects into variables.

Example:
javascript
```
const user = { name: "Arun", age: 25, location: "India" };
const { name, age } = user; // Destructuring
console.log(name); // Output: Arun
console.log(age);  // Output: 25
```

4.4 Object-Oriented Programming Concepts

4.4.1 Classes and Constructors

Classes are blueprints for creating objects. The `constructor` method is used to initialize properties.

Example:
javascript
```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  greet() {
    console.log(`Hi, I'm ${this.name}`);
  }
}
const person = new Person("Arun", 25);
person.greet(); // Output: Hi, I'm Arun
```

4.4.2 Prototypes and Inheritance

Prototypes allow objects to inherit properties and methods.

Example:
javascript

```
function Animal(name) {
  this.name = name;
}
Animal.prototype.speak = function () {
  console.log(`${this.name} makes a sound`);
};
const dog = new Animal("Dog");
dog.speak(); // Output: Dog makes a sound
```

---

5. Arrays and Array Methods

5.1 Creating and Accessing Arrays

Example:
javascript

```
const fruits = ["Apple", "Banana", "Cherry"];
console.log(fruits[0]); // Output: Apple
```

5.2 Common Array Methods

- **push**: Add to the end, **pop**: Remove from the end.

shift: Remove from the start, **unshift**: Add to the start.
javascript
Copy code

```
let arr = [1, 2, 3];
arr.push(4); // [1, 2, 3, 4]
arr.pop();   // [1, 2, 3]
```

5.3 Advanced Array Methods

map: Create a new array by applying a function to each element.
javascript

```
const nums = [1, 2, 3];
const squares = nums.map((num) => num * num); // [1, 4, 9]
```

filter: Create a new array with elements that pass a condition.
javascript
Copy code

```
const even = nums.filter((num) => num % 2 === 0); // [2]
```

reduce: Accumulate values.
javascript

```
const sum = nums.reduce((total, num) => total + num, 0); // 6
```

5.4 Multidimensional Arrays

Example:
javascript

```
const matrix = [[1, 2], [3, 4]];
console.log(matrix[1][0]); // Output: 3
```

---

6. DOM Manipulation

6.1 What is the DOM?

The Document Object Model represents HTML elements as objects.

6.2 Selecting Elements

Example:
javascript
```
const heading = document.getElementById("title");
console.log(heading.innerText);
```

6.3 Changing Element Content and Styles

Example:
javascript
```
heading.innerText = "Hello, World!";
heading.style.color = "blue";
```

6.4 Adding and Removing Elements

Example:
javascript
```
const div = document.createElement("div");
div.innerText = "New Element";
document.body.appendChild(div);
```

6.5 Event Handling

Example:
javascript
```
button.addEventListener("click", () => {
  alert("Button clicked!");
});
```

---

7. JavaScript Events

7.1 Mouse Events

Examples: `click`, `dblclick`, `mouseover`.
javascript
```
element.addEventListener("click", () => console.log("Clicked!"));
```

7.2 Keyboard Events

Examples: `keydown`, `keyup`.
javascript
```
document.addEventListener("keydown", (event) =>
console.log(event.key));
```

7.3 Form Events

Examples: `submit`, `change`.
javascript
Copy code
```
form.addEventListener("submit", (e) => e.preventDefault());
```

---

8. JavaScript Error Handling

8.1 Understanding Errors

- **SyntaxError**, **ReferenceError**, etc.

8.2 try, catch, finally Blocks

Example:
javascript
```
try {
  console.log(undeclaredVar);
} catch (error) {
  console.log("Error:", error.message);
} finally {
  console.log("Execution completed.");
}
```

8.3 Throwing Custom Errors

Example:
javascript
```
function validateAge(age) {
  if (age < 18) throw new Error("Age must be 18 or above");}
```

---

9. Advanced JavaScript Concepts

9.1 Closures and Scope

Example:
javascript
```
function outer() {
  let count = 0;
  return function inner() {
    count++;
    console.log(count);
  };
}
const counter = outer();
counter(); // 1
counter(); // 2
```

9.2 Asynchronous JavaScript

Promises:
javascript
```
fetch("https://api.example.com").then((response) =>
console.log(response));
```

---

10. JavaScript APIs

10.1 Browser APIs

Example:
javascript
```
navigator.geolocation.getCurrentPosition((pos) => console.log(pos));
```

10.2 Working with JSON

Example:
javascript
```
const jsonData = '{"name": "Arun", "age": 25}';
const obj = JSON.parse(jsonData); console.log(obj.name); // Arun
```