# JavaScript Basics

*A Comprehensive 7-Page Guide for Beginners*

## Page 1: Introduction to JavaScript

### What is JavaScript?

JavaScript is a high-level, interpreted programming language that conforms to the ECMAScript specification. It's primarily known as the scripting language for web pages but is also used in many non-browser environments.

### History and Evolution

- **1995**: Created by Brendan Eich at Netscape in just 10 days
- **1997**: Standardized as ECMAScript
- **2009**: ECMAScript 5 (ES5) released with significant improvements
- **2015**: ECMAScript 2015 (ES6) introduced major enhancements
- **Present**: Annual releases with new features

### Why Learn JavaScript?

1. **Ubiquity**: Runs on nearly all web browsers
2. **Versatility**: Used for front-end, back-end (Node.js), mobile apps, and more
3. **Large Community**: Extensive resources and libraries available
4. **High Demand**: One of the most sought-after programming skills

### Setting Up Your Environment

To start with JavaScript, you need:

1. A text editor (VS Code, Sublime Text, Atom)
2. A web browser (Chrome, Firefox, Safari)
3. Browser Developer Tools (F12 or Right-click → Inspect)

## Page 2: JavaScript Fundamentals

### Variables and Data Types

#### Declaring Variables

javascript Copy

```javascript
// Modern ways to declare variables
let name = "John";      // Block-scoped, can be reassigned
const age = 30;         // Block-scoped, cannot be reassigned

// Older way (avoid if possible)
var occupation = "Developer";  // Function-scoped
```

### Primitive Data Types

- **String**: Text values

  javascript Copy

  ```javascript
  let greeting = "Hello, World!";
  ```

- **Number**: Numeric values (integers and floating-point)

  javascript Copy

  ```javascript
  let count = 42;
  let price = 19.99;
  ```

- **Boolean**: True or false values

  javascript Copy

  ```javascript
  let isActive = true;
  ```

- **Undefined**: Variable declared but not assigned a value

  javascript Copy

  ```javascript
  let user;  // Value is undefined
  ```

- **Null**: Intentional absence of any value

  javascript Copy

  ```javascript
  let data = null;
  ```

- **Symbol**: Unique and immutable values (ES6)

  javascript Copy

  ```javascript
  let id = Symbol("id");
  ```

- **BigInt**: For integers larger than Number can handle (ES2020)

javascript

Copy

```javascript
let bigNumber = 9007199254740991n;
```

## Page 3: Operators and Expressions

### Arithmetic Operators

javascript

Copy

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

let sum = a + b;         // Addition: 13
let difference = a - b;  // Subtraction: 7
let product = a * b;     // Multiplication: 30
let quotient = a / b;    // Division: 3.333...
let remainder = a % b;   // Modulus (remainder): 1
let power = a ** b;      // Exponentiation: 1000
```

### Comparison Operators

javascript

Copy

```javascript
a > b    // Greater than: true
a < b    // Less than: false
a >= b   // Greater than or equal to: true
a <= b   // Less than or equal to: false
a == b   // Equal to (value): false
a === b  // Strict equal to (value and type): false
a != b   // Not equal to: true
a !== b  // Strict not equal to: true
```

### Logical Operators

javascript

Copy

```javascript
let x = true, y = false;

x && y  // Logical AND: false
x || y  // Logical OR: true
!x      // Logical NOT: false
```

### Assignment Operators

javascript Copy

```javascript
let c = 5;

c += 2;   // c = c + 2: 7
c -= 1;   // c = c - 1: 6
c *= 3;   // c = c * 3: 18
c /= 2;   // c = c / 2: 9
c %= 4;   // c = c % 4: 1
```

### Template Literals (ES6)

javascript Copy

```javascript
let name = "Alice";
let greeting = `Hello, ${name}!`;  // "Hello, Alice!"
```

## Page 4: Control Flow

### Conditional Statements

#### if-else Statement

javascript Copy

```javascript
let hour = 14;

if (hour < 12) {
    console.log("Good morning!");
} else if (hour < 18) {
    console.log("Good afternoon!");
} else {
    console.log("Good evening!");
}
```

#### Switch Statement

javascript

```javascript
let day = "Monday";

switch (day) {
    case "Monday":
        console.log("Start of work week");
        break;
    case "Friday":
        console.log("End of work week");
        break;
    case "Saturday":
    case "Sunday":
        console.log("Weekend!");
        break;
    default:
        console.log("Midweek");
}
```

### Ternary Operator

javascript

```javascript
let age = 20;
let canVote = age >= 18 ? "Yes" : "No";
```

## Loops

### for Loop

javascript

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

### while Loop

javascript

```javascript
let count = 0;
while (count < 5) {
    console.log(`Count: ${count}`);
    count++;
}
```

#### do-while Loop

javascript Copy

```javascript
let i = 0;
do {
    console.log(`Do-while: ${i}`);
    i++;
} while (i < 5);
```

#### for...of Loop (ES6, for iterables)

javascript Copy

```javascript
let colors = ["red", "green", "blue"];
for (let color of colors) {
    console.log(color);
}
```

#### for...in Loop (for object properties)

javascript Copy

```javascript
let person = {name: "John", age: 30};
for (let key in person) {
    console.log(`${key}: ${person[key]}`);
}
```

## Page 5: Functions

### Defining Functions

#### Function Declaration

javascript Copy

```javascript
function greet(name) {
    return `Hello, ${name}!`;
}
```

#### Function Expression

javascript Copy

```javascript
const greet = function(name) {
    return `Hello, ${name}!`;
};
```

### Arrow Functions (ES6)

javascript Copy

```javascript
const greet = (name) => `Hello, ${name}!`;

// Multiple parameters
const add = (a, b) => a + b;

// No parameters
const sayHi = () => "Hi there!";

// Multiple statements
const process = (num) => {
    const result = num * 2;
    return result + 10;
};
```

## Parameters and Arguments

javascript Copy

```javascript
// Default parameters
function greet(name = "Guest") {
    return `Hello, ${name}!`;
}

greet();          // "Hello, Guest!"
greet("Alice");  // "Hello, Alice!"

// Rest parameters
function sum(...numbers) {
    return numbers.reduce((total, num) => total + num, 0);
}

sum(1, 2, 3, 4);  // 10
```

## Scope and Closures

### Scope

javascript

```javascript
let globalVar = "I'm global";

function example() {
    let localVar = "I'm local";
    console.log(globalVar);  // Accessible
    console.log(localVar);   // Accessible
}

console.log(globalVar);      // Accessible
console.log(localVar);       // Error: not defined
```

### Closures

javascript

```javascript
function createCounter() {
    let count = 0;
    return function() {
        return ++count;
    };
}

const counter = createCounter();
console.log(counter());  // 1
console.log(counter());  // 2
console.log(counter());  // 3
```

## Page 6: Arrays and Objects

### Arrays

#### Creating Arrays

javascript

```javascript
// Array literal
let fruits = ["apple", "banana", "orange"];

// Array constructor
let numbers = new Array(1, 2, 3, 4);

// Empty array
let empty = [];
```

### Accessing Elements

javascript

Copy

```javascript
let first = fruits[0];       // "apple"
let last = fruits[fruits.length - 1];  // "orange"
```

### Common Array Methods

javascript

Copy

```javascript
// Adding/removing elements
fruits.push("grape");          // Add to end
fruits.pop();                  // Remove from end
fruits.unshift("strawberry"); // Add to beginning
fruits.shift();                // Remove from beginning
fruits.splice(1, 1, "kiwi"); // Remove 1 element at index 1, add "kiwi"

// Finding elements
fruits.indexOf("banana");    // 1 (or -1 if not found)
fruits.includes("apple");    // true

// Transforming arrays
let doubled = numbers.map(num => num * 2);      // [2, 4, 6, 8]
let evens = numbers.filter(num => num % 2 === 0); // [2, 4]
let sum = numbers.reduce((total, num) => total + num, 0); // 10

// Iterating
fruits.forEach(fruit => console.log(fruit));
```

## Objects

### Creating Objects

javascript Copy

```javascript
// Object literal
let person = {
    name: "John",
    age: 30,
    isEmployed: true
};

// Object constructor
let user = new Object();
user.name = "Alice";
user.age = 25;
```

### Accessing Properties

javascript Copy

```javascript
// Dot notation
let name = person.name;  // "John"

// Bracket notation
let age = person["age"]; // 30
```

### Object Methods

javascript Copy

```javascript
let calculator = {
    add: function(a, b) {
        return a + b;
    },
    // Shorthand method (ES6)
    subtract(a, b) {
        return a - b;
    }
};

calculator.add(5, 3);      // 8
calculator.subtract(10, 4); // 6
```

### Object Destructuring (ES6)

javascript Copy

```javascript
let {name, age} = person;
console.log(name);  // "John"
console.log(age);   // 30
```

## Page 7: DOM Manipulation and Events

### Accessing DOM Elements

javascript Copy

```javascript
// By ID
const header = document.getElementById("header");

// By class name
const items = document.getElementsByClassName("item");

// By tag name
const paragraphs = document.getElementsByTagName("p");

// Query selectors (returns first match)
const container = document.querySelector(".container");

// Query selectors (returns all matches)
const buttons = document.querySelectorAll("button");
```

### Modifying DOM Elements

javascript

Copy

```javascript
// Changing content
element.textContent = "New text";  // Text only
element.innerHTML = "<span>HTML content</span>";  // HTML

// Modifying attributes
element.setAttribute("src", "image.jpg");
element.getAttribute("href");
element.removeAttribute("disabled");

// Modifying styles
element.style.color = "blue";
element.style.fontSize = "16px";

// Modifying classes
element.classList.add("active");
element.classList.remove("disabled");
element.classList.toggle("highlighted");
element.classList.contains("selected");  // Check if class exists
```

### Creating and Removing Elements

javascript

Copy

```javascript
// Create element
const newDiv = document.createElement("div");
newDiv.textContent = "I'm a new div";

// Append to document
document.body.appendChild(newDiv);

// Insert before another element
parentElement.insertBefore(newDiv, referenceElement);

// Remove element
element.remove();  // Modern way
parentElement.removeChild(childElement);  // Traditional way
```

### Event Handling

javascript

```javascript
// Using addEventListener
const button = document.querySelector("button");

button.addEventListener("click", function(event) {
    console.log("Button clicked!");
    console.log(event);  // Event object with details
});

// Multiple events
element.addEventListener("mouseover", handleMouseOver);
element.addEventListener("mouseout", handleMouseOut);

function handleMouseOver() {
    console.log("Mouse over!");
}

function handleMouseOut() {
    console.log("Mouse out!");
}

// Removing event listeners
element.removeEventListener("click", handlerFunction);
```

### Common Events

- `click`: When an element is clicked
- `dblclick`: When an element is double-clicked
- `mouseover`/`mouseout`: When the mouse enters/leaves an element
- `keydown`/`keyup`: When a key is pressed/released
- `submit`: When a form is submitted
- `load`: When a page or image loads
- `resize`: When the window is resized
- `scroll`: When the user scrolls

### Event Delegation

javascript

```javascript
// Instead of attaching events to multiple child elements
document.getElementById("parent-list").addEventListener("click", function(event) {
    if (event.target.matches("li")) {
        console.log("List item clicked:", event.target.textContent);
    }
});
```

Happy coding with JavaScript! To continue your learning journey, explore topics like asynchronous JavaScript (Promises, async/await), modules, and popular frameworks like React, Vue, or Angular.