Module 1: JavaScript Foundations

Weeks 1-2

Learning Objectives

By the end of this module, you will:

- Understand JavaScript's role in web development
- Set up a proper development environment
- Master variable declaration and data types
- Apply operators and understand expressions
- Work with type conversion and coercion

What is JavaScript?

JavaScript is:

- A high-level, interpreted programming language
- The programming language of the web
- Dynamic and weakly typed
- Prototype-based object-oriented
- Multi-paradigm (procedural, OOP, functional)

Where JavaScript Runs:

- **Client-side**: In web browsers
- **Server-side**: Node.js environments
- **Mobile apps**: React Native, Ionic
- **Desktop apps**: Electron framework

JavaScript Engines

Popular JavaScript Engines:

- **V8** (Chrome, Node.js)

- **SpiderMonkey** (Firefox)
- **JavaScriptCore** (Safari)
- **Chakra** (Edge)

Execution Context:

```javascript
// Global execution context
var globalVar = "I'm global";

function exampleFunction() {
  // Function execution context
  var localVar = "I'm local";
  console.log(globalVar); // Accessible
}
```

Development Environment Setup

Required Tools:

1. **Code Editor**: VS Code
2. **Browser**: Chrome or Firefox
3. **Extensions**: JavaScript (ES6) code snippets, Live Server

VS Code Setup:

```javascript
// Install useful extensions:
// - JavaScript (ES6) code snippets
// - Live Server
// - Prettier - Code formatter
// - ESLint
```

Including JavaScript in HTML

Three Methods:

1. Inline JavaScript:

```html
<button onclick="alert('Hello!')">Click me</button>
```

2. Internal JavaScript:

```html
<script>
  console.log("Hello, JavaScript!");
</script>
```

3. External JavaScript:

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

Best Practice: Use external files for maintainability

Basic Syntax and Structure

Code Structure:

```javascript
// Single-line comment
/*
   Multi-line comment
   Spans multiple lines
*/

// Statements end with semicolons (optional but recommended)
console.log("Hello, World!");

// Code blocks use curly braces
if (true) {
    console.log("This is a code block");
}
```

Case Sensitivity:

```javascript
var myVariable = "Hello";
var MyVariable = "World";  // Different variable!
var MYVARIABLE = "Test";   // Also different!
```

Variables and Data Types

Variable Declaration:

`var` (Function-scoped):

```javascript
var studentName = "Alice";
var studentName = "Bob"; // Can redeclare
```

`let` (Block-scoped):

```javascript
let courseName = "JavaScript Mastery";
// let courseName = "React"; // Error: Cannot redeclare
```

`const` (Block-scoped, immutable binding):

```javascript
const currentYear = 2024;
// currentYear = 2025; // Error: Cannot reassign
```

Primitive Data Types

Number:

```javascript
```

```
let age = 25;                // Integer
let gpa = 3.75;              // Decimal
let scientificNotation = 2e3;   // 2000
let infinity = Infinity;
let notANumber = NaN;
```

String:

```
javascript

let singleQuotes = 'Hello';
let doubleQuotes = "World";
let templateLiterals = `Hello, ${name}!`;
let multiLine = `This is a
multi-line string`;
```

Boolean:

```
javascript

let isStudent = true;
let hasGraduated = false;
```

Special Values

Null and Undefined:

```
javascript

let score = null;       // Intentional absence of value
let grade;              // Undefined (declared but not assigned)

console.log(grade);     // undefined
console.log(score);     // null
```

Symbol (ES6):

```
javascript
```

```
let uniqueId = Symbol('id');
let anotherId = Symbol('id');
console.log(uniqueId === anotherId); // false
```

BigInt (ES2020):

```
javascript

let bigNumber = 123456789012345678901234567890123456789012345678901234567890n;
```

Type Checking

Using `typeof`:

```
javascript

console.log(typeof 42);         // "number"
console.log(typeof "Hello");    // "string"
console.log(typeof true);       // "boolean"
console.log(typeof undefined);  // "undefined"
console.log(typeof null);       // "object" (known quirk!)
console.log(typeof {});         // "object"
console.log(typeof []);         // "object"
console.log(typeof function(){}); // "function"
```

Type Conversion

Explicit Conversion:

```
javascript
```

```javascript
// To String
let num = 42;
let strNum = String(num);       // "42"
let strNum2 = num.toString();   // "42"

// To Number
let str = "42";
let numStr = Number(str);       // 42
let numStr2 = parseInt(str);    // 42
let numStr3 = parseFloat("3.14"); // 3.14

// To Boolean
let truthyValue = Boolean(1);   // true
let falsyValue = Boolean(0);    // false
```

Type Coercion (Implicit)

Automatic Type Conversion:

javascript

```javascript
// String concatenation
console.log("5" + 3);    // "53"
console.log(5 + "3");    // "53"

// Arithmetic operations
console.log("5" - 3);    // 2
console.log("5" * "2");  // 10
console.log("5" / "2");  // 2.5

// Comparison
console.log("5" == 5);   // true (loose equality)
console.log("5" === 5);  // false (strict equality)
```

Operators

Arithmetic Operators:

javascript

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

console.log(a + b);  // 13 (Addition)
console.log(a - b);  // 7  (Subtraction)
console.log(a * b);  // 30 (Multiplication)
console.log(a / b);  // 3.333... (Division)
console.log(a % b);  // 1  (Remainder/Modulus)
console.log(a ** b); // 1000 (Exponentiation - ES2016)
```

Assignment Operators:

javascript

```javascript
let x = 5;
x += 3;  // x = x + 3; (8)
x -= 2;  // x = x - 2; (6)
x *= 4;  // x = x * 4; (24)
x /= 2;  // x = x / 2; (12)
x %= 5;  // x = x % 5; (2)
```

Comparison Operators

javascript

```javascript
let score1 = 85;
let score2 = "85";

// Equality operators
console.log(score1 == score2);   // true (loose equality)
console.log(score1 === score2);  // false (strict equality)
console.log(score1 != score2);   // false (loose inequality)
console.log(score1 !== score2);  // true (strict inequality)

// Relational operators
console.log(score1 > 80);      // true
console.log(score1 < 90);      // true
console.log(score1 >= 85);     // true
console.log(score1 <= 85);     // true
```

Logical Operators

```javascript
let isLoggedIn = true;
let hasPermission = false;
let isAdmin = true;

// AND (&&)
console.log(isLoggedIn && hasPermission); // false
console.log(isLoggedIn && isAdmin);       // true

// OR (||)
console.log(hasPermission || isAdmin);    // true
console.log(hasPermission || false);      // false

// NOT (!)
console.log(!isLoggedIn);                 // false
console.log(!hasPermission);              // true

// Short-circuit evaluation
let result = isLoggedIn && getUser();     // getUser() only runs if isLoggedIn is true
```

Ternary Operator

Conditional Assignment:

```javascript
let score = 85;
let grade = score >= 90 ? 'A' :
            score >= 80 ? 'B' :
            score >= 70 ? 'C' :
            score >= 60 ? 'D' : 'F';

console.log(grade); // 'B'

// Simple example
let status = isLoggedIn ? 'Welcome!' : 'Please log in';
```

Operator Precedence

Order of Operations:

javascript

```javascript
let result = 3 + 4 * 5;       // 23 (not 35)
let result2 = (3 + 4) * 5;    // 35

// Precedence levels (high to low):
// 1. Parentheses ()
// 2. Exponentiation **
// 3. Unary operators !, -, +
// 4. Multiplication *, Division /, Modulus %
// 5. Addition +, Subtraction -
// 6. Comparison <, >, <=, >=
// 7. Equality ==, !=, ===, !==
// 8. Logical AND &&
// 9. Logical OR ||
// 10. Ternary ?:
// 11. Assignment =, +=, -=, etc.
```

Practical Example: Student Information System

javascript

```javascript
// Student Information System
let studentName = "Alice Johnson";
let studentEmail = "alice@university.edu";
let currentYear = 2024;
let studentId = 12345;
let gpa = 3.85;
let isFullTime = true;
let graduationYear = null; // Not yet determined

// Display student information
console.log("=== Student Information ===");
console.log("Name: " + studentName);
console.log("Email: " + studentEmail);
console.log("ID: " + studentId);
console.log("GPA: " + gpa.toFixed(2));
console.log("Status: " + (isFullTime ? "Full-time" : "Part-time"));

// Type checking
console.log("\n=== Data Types ===");
console.log("Name type: " + typeof studentName);
console.log("GPA type: " + typeof gpa);
console.log("Status type: " + typeof isFullTime);
console.log("Graduation year type: " + typeof graduationYear);

// GPA calculation example
let totalPoints = 87.5;
let totalCredits = 24;
let calculatedGPA = totalPoints / totalCredits;
console.log("\nCalculated GPA: " + calculatedGPA.toFixed(2));
```

Common Pitfalls and Best Practices

Avoid These Mistakes:

javascript

```javascript
// DON'T: Mixing data types unexpectedly
let confusing = "5" + 3 - 2; // "53" - 2 = 51

// DON'T: Using var unnecessarily
var oldStyle = "avoid"; // Use let or const

// DON'T: Comparing with ==
if ("0" == false) { } // true, but confusing

// DO: Use strict equality
if ("0" === false) { } // false, as expected
```

Best Practices:

```javascript
// Use meaningful variable names
let studentGradePointAverage = 3.75; // Good
let sgpa = 3.75; // Avoid abbreviations

// Use const for values that don't change
const MAX_STUDENTS = 30;
const UNIVERSITY_NAME = "Tech University";

// Use template literals for string interpolation
let welcomeMessage = `Welcome, ${studentName}!`;
```

Debugging and Development Tools

Browser Developer Tools:

- **Console**: View output and errors
- **Sources**: Set breakpoints and step through code
- **Elements**: Inspect HTML and CSS
- **Network**: Monitor API calls

Console Methods:

```javascript
```

```
console.log("Basic logging");
console.warn("Warning message");
console.error("Error message");
console.info("Information message");
console.table([{name: "Alice", gpa: 3.8}, {name: "Bob", gpa: 3.6}]);
```

Assignment 1: Student Information System

Requirements:

Create an HTML page with JavaScript that:

1. **Declares variables** for student information:
   - Name, email, major, year, GPA, enrollment status
2. **Displays information** using different output methods:
   - console.log(), alert(), document.write()
3. **Demonstrates type conversion**:
   - Convert strings to numbers for calculations
   - Convert numbers to strings for display
4. **Calculates derived values**:
   - Years until graduation
   - GPA category (Excellent, Good, Average, etc.)

Example Structure:

html

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Information System</title>
</head>
<body>
  <h1>Student Information System</h1>
  <script src="student-info.js"></script>
</body>
</html>
```

Next Module Preview

Module 2: Control Structures

- Conditional statements (if/else, switch)
- Loops (for, while, do-while)
- Loop control and optimization
- Building a grade calculator

Preparation:

- Practice variable declarations
- Experiment with different data types
- Get comfortable with operators
- Set up your development environment

Questions and Discussion

Think About:

1. When would you use `let` vs `const` vs `var`?
2. What are the implications of JavaScript's type coercion?
3. How can understanding operator precedence prevent bugs?
4. What debugging strategies work best for you?

Resources:

- MDN JavaScript Guide: Variables
- JavaScript.info: Data types
- VS Code JavaScript debugging guide