Complete JavaScript Course: Learn by Building Projects + Interactive Workbook

Course Overview

Master JavaScript through hands-on projects with built-in practice exercises

Learning Path: Fundamentals ’ DOM Manipulation ’ Async Programming ’ Real APIs
Approach: Learn a concept ’ Practice it ’ Build a project
Portfolio Projects: 8 complete applications

MODULE 1: JAVASCRIPT FUNDAMENTALS + WORKBOOK

Step 1: What is JavaScript?

The Web Trinity:

- **HTML** = Structure (skeleton)
- **CSS** = Style (clothes/makeup)
- **JavaScript** = Behavior (brain/movement)

Real-World Examples:

- Show/hide menus
- Form validation
- Animations & slideshows
- Full applications (games, shopping carts)

Fun fact: JavaScript runs inside your browser, so you don't need extra software to try it out. Just a browser lik
e Chrome, Firefox, or Edge.

Step 2: Your First JavaScript Code

```
html

<!DOCTYPE html>
<html>
```

```
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <button onclick="showMessage()">Click Me!</button>

  <script>
    function showMessage() {
      alert('Hello! You clicked the button!');
    }
  </script>
</body>
</html>
```

What's happening here?

1. The `<button>` creates a clickable button
2. The `onclick="showMessage()"` attribute tells the browser: when this button is clicked, run the function called showMessage
3. The `<script>` tag holds the JavaScript code. Inside, we created a function that shows a popup alert

👉 **You can also put your JavaScript in an external file (script.js) and include it with:**

html

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

This keeps your code cleaner.

Step 3: Variables – Storing Information

Think of variables as labeled boxes where you store information.

javascript

```
let studentName = "Maria";  // text (string)
const studentAge = 15;      // number
let testScore = 95;         // number
```

- `let` – use when the value might change
- `const` – use when the value should never change
- `var` – old way, avoid it (can cause bugs)

✅ Practice:

```
let favoriteColor = "blue";
let favoriteNumber = 7;
let isStudent = true;
```

```
console.log(favoriteColor);
console.log(favoriteNumber);
console.log(isStudent);
```

Tip:

- Strings go in quotes " "
- Numbers don't need quotes
- true and false are special boolean values

Step 4: Basic Math Operations

JavaScript can be your calculator:

```
javascript

let num1 = 10;
let num2 = 5;

let sum = num1 + num2;       // 15
let difference = num1 - num2; // 5
let product = num1 * num2;   // 50
let quotient = num1 / num2;  // 2

console.log("Sum:", sum);
console.log("Difference:", difference);
console.log("Product:", product);
console.log("Quotient:", quotient);
```

Other operators:

- ** = exponentiation (e.g., 2 ** 3 = 8)
- % = remainder (e.g., 10 % 3 = 1)

PROJECT 1: SIMPLE CALCULATOR

This project combines HTML inputs + JavaScript functions.

You'll learn:

- How to get values from the page

- How to convert text to numbers
- How to update text dynamically
- How to handle errors (like dividing by 0)

Key new concept:

```javascript
parseFloat("3.14"); // turns text into a number
```

Without parseFloat, numbers from input fields are treated as text, and "5" + "5" would give "55" instead of 10.

WORKBOOK SECTION 1: FUNDAMENTALS PRACTICE

🔹 Quick Recap

Module	Key Topics
Basics	Variables (let, const), math operators, console.log

Practice Questions

Fill in the Blank

1. The keyword used to declare a variable that cannot change is _____.
2. To check if two values are equal in value and type, we use _____.
3. The array index always starts at _____.

Multiple Choice

1. Which symbol is used for "not equal"?
   a) !=
   b) !==
   c) <>
   d) ~
2. What will this code log?

javascript

```
let x = 5;
let y = "5";
console.log(x === y);
```

a) true
b) false
c) 5
d) "5"

True/False

1. var is the recommended way to declare variables in modern JavaScript. (T/F)

2. JavaScript can modify HTML and CSS dynamically. (T/F)

Coding Challenges

1. Write a program that asks for a user's age and prints:

- "Too young" if under 13
- "Teenager" if between 13-19
- "Adult" otherwise

2. Debug this code:

javascript

```
let fruits = ["apple", "banana"];
console.log(fruit[0]);  // fix this
```

MODULE 2: MAKING DECISIONS IN YOUR CODE + WORKBOOK

Step 1: If Statements – Making Choices

If statements help programs decide what to do:

javascript

```
let temperature = 25;

if (temperature > 30) {
```

```
    console.log("It's hot outside!");
} else if (temperature > 20) {
    console.log("The weather is nice!");
} else {
    console.log("It's cold outside!");
}
```

👉 **Code runs top to bottom. Once a condition is true, the rest are skipped.**

Step 2: Comparison Operators

- `>` greater than
- `<` less than
- `>=` greater than or equal
- `<=` less than or equal
- `===` equal in value and type
- `!==` not equal

Example:

javascript

```
let age = 16;

console.log(age > 18);  // false
console.log(age === 16); // true
console.log(age !== 16); // false
```

💡 **Always use === instead of == (safer, avoids type confusion).**

Step 3: Logical Operators (AND, OR, NOT)

- `&&` ′n AND (both must be true)
- `||` ′n OR (at least one true)
- `!` ′n NOT (opposite)

javascript

```
let hasTicket = true;
let hasMoney = false;
```

```
if (hasTicket && hasMoney) {
    console.log("You can go to the movie!");
} else if (hasTicket || hasMoney) {
    console.log("You might be able to go!");
} else {
    console.log("You need a ticket and money!");
}
```

PROJECT 2: GRADE CALCULATOR

Here you'll learn input validation: making sure users enter valid numbers (0-100).

You'll also use if/else to decide grades, and `.style.color` to change text color.

💡 **Why use .style.color?**
Because JavaScript can directly modify CSS. This makes your web page dynamic.

WORKBOOK SECTION 2: DECISIONS PRACTICE

Quick Recap

Module	Key Topics
Decisions	if/else, comparison operators, logical operators

Mini Projects

Project 2: Grade Calculator

- User enters score (0-100)
- Show "A" for 90+, "B" for 80-89, "C" for 70-79, "Fail" otherwise
- Make the text green for pass and red for fail

🔹 Coding Challenges

1. Create an array of 5 numbers. Use a loop to print only even numbers.

2. Write an object called car with properties: brand, model, year. Print the brand.

MODULE 3: WORKING WITH LISTS AND REPETITION + WORKBOOK

Step 1: Arrays – Storing Multiple Values

Arrays = boxes that hold multiple items:

javascript

```
let fruits = ["apple", "banana", "orange"];
let testScores = [95, 87, 92, 78];
```

👉 Items are accessed by index (starting at 0):

javascript

```
console.log(fruits[0]); // apple
console.log(testScores[2]); // 92
```

Step 2: Looping Through Arrays

Loops help repeat tasks without writing code many times.

javascript

```
let classmates = ["Sarah", "Tom", "Jessica", "Mike"];

// For loop
for (let i = 0; i < classmates.length; i++) {
    console.log("Hello " + classmates[i]);
}

// forEach loop
classmates.forEach(function(classmate) {
    console.log("Hi " + classmate);
});
```

Step 3: Array Methods

JavaScript arrays come with powerful methods:

javascript

```
let numbers = [1, 2, 3, 4, 5];

numbers.push(6);  // add to end
numbers.pop();    // remove from end
numbers.shift();  // remove first
numbers.unshift(0); // add to start

let doubled = numbers.map(num => num * 2); // transform
let evenNumbers = numbers.filter(num => num % 2 === 0); // filter
```

PROJECT 3: SHOPPING LIST APP

Here you'll:

- Use an array to store shopping items
- Add/remove items with buttons
- Use Date.now() to give each item a unique ID
- Use keypress event to allow pressing Enter as a shortcut

💡 **New concept: Event delegation ḥ Instead of attaching separate event listeners to every element, you can attach one to a parent and check which child triggered it. (Useful in bigger apps.)**

WORKBOOK SECTION 3: LISTS & LOOPS PRACTICE

🔹 Quick Recap

Module	Key Topics
Loops	Arrays, for loop, forEach, array methods

🔹 Mini Projects

Project 3: Shopping List

- Input box + "Add Item" button
- List items below with delete buttons
- Use an array to manage items

MODULE 4: WORKING WITH OBJECTS AND DATA + WORKBOOK

Step 1: Objects – Grouping Related Data

Objects store key 'n value pairs:

javascript

```
let student = {
    name: "John Smith",
    age: 16,
    grade: 10,
    subjects: ["Math", "Science", "English"],
    isActive: true
};
```

Access data with dot `student.name` **or bracket** `student["age"]`.

Step 2: Multiple Objects

Arrays of objects let you handle real-world data:

javascript

```
let students = [
    {name: "Alice", grade: 95, subject: "Math" },
    {name: "Bob", grade: 87, subject: "Science"},
    {name: "Charlie", grade: 92, subject: "English" }
];
```

PROJECT 4: STUDENT MANAGER

Here you'll:

- Store students in an array of objects

- Dynamically generate a table with student info
- Use .reduce() to calculate averages
- Conditionally style rows (high-score class for top students)

💡 **This is your first step toward CRUD apps (Create, Read, Update, Delete).**

WORKBOOK SECTION 4: OBJECTS PRACTICE

Quick Recap

Module	Key Topics
Objects	Objects, arrays of objects

Mini Projects

Project 4: Student Manager

- Store student data in objects (name, grade, subject)
- Show them in a table
- Highlight students with grade > 90

MODULE 5: MAKING WEBSITES INTERACTIVE + WORKBOOK

Step 1: Event Listeners

Instead of inline onclick=..., best practice is to use addEventListener:

javascript

```
document.getElementById('myButton').addEventListener('click', function() {
  alert('Button was clicked!');
});
```

Why better?

- Keeps HTML clean
- Allows multiple events on the same element
- Easier to maintain

Step 2: Creating Elements Dynamically

javascript

```
let newDiv = document.createElement('div');
newDiv.textContent = 'This is a new div!';
newDiv.className = 'my-class';
document.body.appendChild(newDiv);
```

This is the power of JavaScript: building HTML with code.

PROJECT 5: INTERACTIVE QUIZ APP

Here you'll:

- Store questions in an array of objects
- Dynamically generate HTML elements (div, span, button)
- Keep track of application state (current question, score)
- Provide feedback (correct/incorrect)
- Show final results and allow restart

This project ties together:

- Arrays & objects (questions)
- Loops (options)
- Conditionals (correct answer)
- Events (button clicks)
- DOM manipulation (show/hide screens)

WORKBOOK SECTION 5: DOM MANIPULATION PRACTICE

Quick Recap

Module	Key Topics
DOM	Event listeners, DOM manipulation

🔹 Mini Projects

Project 5: Quiz App

- Show one question at a time with multiple choices
- Track score
- Show results at the end with "Restart" button

MODULE 6: ASYNC JAVASCRIPT & APIs + WORKBOOK

Step 1: Synchronous vs Asynchronous

- **Synchronous** ́n Code runs line by line, waiting for each task to finish
- **Asynchronous** ́n Some tasks (like network requests, timers) run in the background so the rest of the cod e can keep running

Example:

javascript

```
console.log("Start");

setTimeout(() => {
  console.log("Inside setTimeout");
}, 2000);
```

```
console.log('End');
```

Output:

```
text

Start
End
Inside setTimeout
```

(The delay happens, but the rest of the code continues.)

Step 2: Promises

Promises represent values that will be available later.

```
javascript

let promise = new Promise((resolve, reject) => {
  let success = true;
  if (success) {
    resolve("Task finished successfully!");
  } else {
    reject("Something went wrong.");
  }
});

promise
  .then(result => console.log(result))
  .catch(error => console.log(error));
```

Step 3: Async/Await

A cleaner way to handle promises:

```
javascript

async function fetchData() {
  try {
    let response = await fetch("https://jsonplaceholder.typicode.com/posts/1");
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.log("Error:", error);
  }
}

fetchData();
```

PROJECT 6: WEATHER APP

- User enters a city
- App fetches live weather data from an API
- Displays temperature, description, and weather icon
- Handles errors (e.g., invalid city name)

💡 **New concept: API keys (many APIs require signing up for free access)**

WORKBOOK SECTION 6: ASYNC JAVASCRIPT PRACTICE

Quick Recap

Module	Key Topics
Async JS	setTimeout, promises, async/await, fetch API

Advanced Projects

Project 6: Weather App

- User types a city ′n fetch weather API ′n show temperature + condition
- Handle invalid city names with error messages

MODULE 7: LOCAL STORAGE & STATE MANAGEMENT + WORKBOOK

Step 1: What is Local Storage?

Local storage lets you save data inside the browser. Even if the page is refreshed, data stays until manually cleared.

javascript

```javascript
// Save
localStorage.setItem("username", "Alice");

// Retrieve
let name = localStorage.getItem("username");
console.log(name);

// Remove
localStorage.removeItem("username");
```

Step 2: JSON (JavaScript Object Notation)

When saving complex data (arrays, objects), you must use JSON.stringify() and JSON.parse().

javascript

```javascript
let tasks = ["Buy milk", "Do homework"];

// Save
localStorage.setItem("tasks", JSON.stringify(tasks));

// Retrieve
let storedTasks = JSON.parse(localStorage.getItem("tasks"));
console.log(storedTasks);
```

PROJECT 7: PERSISTENT TO-DO LIST

- User adds tasks ′n saved in local storage
- Refresh page ′n tasks remain
- Add "Clear All" button to reset

💡 **This is a real-world app pattern: storing user data locally without a database.**

WORKBOOK SECTION 7: LOCAL STORAGE PRACTICE

Quick Recap

Module	Key Topics
Local Storage	Saving/retrieving user data, JSON

Advanced Projects

Project 7: Persistent To-Do List

- Input box for tasks
- Save tasks in local storage
- Reload page 'n tasks remain
- "Clear All" button to reset

Practice Questions

Fill in the Blank

4. The method JSON._____ turns a JavaScript object into a string for storage.

5. setTimeout is an example of _____ JavaScript.

Multiple Choice

3. Which method removes the last item from an array?
   a) shift()
   b) pop()
   c) unshift()
   d) slice()

True/False

3. localStorage clears automatically when you refresh a page. (T/F)

MODULE 8: FINAL PORTFOLIO PROJECT + WORKBOOK

Step 1: Bringing It All Together

By now, you know:

- DOM manipulation
- Events
- Loops & conditionals
- Objects & arrays
- Async JavaScript (fetch API)
- Local storage

👉 **Time to combine them all into one capstone project.**

FINAL PROJECT: MOVIE SEARCH APP

Features:

1. Search for a movie using the OMDb API
2. Display results (poster, title, year, rating)
3. Add movies to "Favorites" 'n saved in local storage
4. Favorites page 'n list of saved movies with option to remove
5. Error handling (invalid movie, no results)

Skills learned here:

- Fetching API data (real-world APIs)
- Handling user input (forms, events)
- Displaying data dynamically (cards, lists)
- Saving state (favorites stored in local storage)
- Code organization (splitting logic into functions)

WORKBOOK SECTION 8: CAPSTONE PROJECT

Capstone Project

Movie Search App

Requirements:

- Search input for movies
- Fetch results from OMDb API
- Display posters + title + year
- Add to Favorites ‘n stored in localStorage
- Favorites page ‘n show saved movies with "Remove" option
- Handle invalid searches