Master JavaScript from the ground up with this comprehensive Core JavaScript course. Designed for developers looking to build a strong foundation, this course covers everything from basic syntax to advanced concepts like classes, functional programming, and asynchronous code.

Dive deep into the essentials of browser-based JavaScript, including DOM manipulation, event handling, and modern ES features. This course also introduces testing practices and prepares you for frameworks like Angular, React, Vue, and Svelte.

What you will learn:

- Understand modern JavaScript syntax, features, and best practices.
- Organize and structure your code effectively with modern **ES Modules**.
- Write robust object-oriented and functional JavaScript.
- Manage asynchronous workflows with promises and async/await.
- Manipulate and interact with the DOM using modern APIs.
- Leverage JavaScript modules for structured and maintainable code.
- Test JavaScript code with tools like Jest and Vite-Test.
- Prepare for advanced JavaScript concepts and frameworks.

Core JavaScript/ECMAScript

Introduction

- Introduce **JavaScript** and its standard, **ECMAScript**.
- Explore the evolution (**ECMAScript versions**, ES5, ES6/ES2015, and beyond).
- Discuss **browser and runtime support** for modern ECMAScript features.
- Explain language basics: **literals**, **identifiers**, **operators**, and **reserved words**.
- Introduce the **JavaScript Type System** and its dynamic nature.
- Highlight the differences between **primitive types** and **objects**.

Node.js and Babel for front-end development

- Introduce **Node.js** (for front-end development)
- Setting up a new **project**
- Understand package.json
- Managing dependencies using **npm** and/or **yarn**
- Introduce package registries (public and private)
- Use linters such as **JSHint**
- Introduce **Babel**
- Setting up a babel project
- Appreciate babel **plugins** and **presets**
- Use and configure the **preset-env**
- Introduce **Browserlist**
- Defining browserlist **queries** to define your **target** (e.g, browsers)

Core Syntax

- Understand **equality** (== vs ===).
- Explain **falsy** and **truthy** values.
- Appreciate **strict mode**
- Declare variables with **let** and **const**, and the legacy **var**.
- Discuss the benefits of **immutable values**.
- Explore **global objects** in various environments.
- Understand **type coercion** and its pitfalls.
- Define and use **named functions**.
- Introduce modern string literals with **template strings**.
- Use different types of **expressions** (arithmetic, logical, etc.).
- Apply **control structures** such as if, for, and while.
- Discuss the **advantages of expressions** over statements.
- Understand and use **exceptions** for error handling.
- Using **tr/catch** to handle exceptions

Modules and Code Organization

- Understand **modules**.
- Explore **ES Modules** (import/export) and their compatibility with older systems.
- Compare **modules** with older patterns like **immediately invoked function expressions (IIFE)**.
- Learn various import and export techniques (default, named, and re-exports).
- Use **dynamic imports** for code splitting and lazy loading.
- Organize large codebases with **barrel files** for cleaner imports.

Objects, Classes and Object-oriented JavaScript

Objects introduction

- Create objects as **object literals**.
- Define object **properties** and **methods**.
- Use getters and setters **accessors**.
- Apply **shorthand property names** for conciseness.
- Use **destructuring assignments** for objects.
- Leverage destructuring with **immutable objects**.
- Define and use **constructors** to initialize objects.
- Discuss limitations of constructors compared to **class syntax** (explained later).

Core Objects and Utilities

- Declare and manipulate **arrays**.
- Use mutable array methods (**pop**, **push**, **(un)shift**, **slice**, **splice**, etc.).
- Work with iterables using the **for...of** loop and **spread syntax**.
- Work with core object types: **Date** and **String**.

- Understand and use **Map** and **Set** for efficient data storage.
- Explore **WeakMap** and **WeakSet** for memory-managed key-value pairs.
- Work with the **Math** object for mathematical operations.
- Introduce the **JSON** object for parsing and stringifying data.

Dealing with Null and Undefined

- Understand the **difference** between null and undefined.
- Recognize the problems caused by **nullish values** (null and undefined).
- **Protect** against nullish values using safe patterns.
- Use logical operators (**&&**, **||**) for fallback logic.
- Apply **optional chaining** (?.) to safely access nested properties (ES2020).
- Use the **nullish coalescing operator** (??) to handle default values (ES2020).
- Combine **optional chaining** and **nullish coalescing** for safer expressions.
- Explore **default destructuring** to provide fallback values when destructuring objects or arrays.

Classes

- Explain **prototypical inheritance**.
- Define and use **ES classes**.
- Add **methods** to classes to add behaviors.
- Define and use **constructor functions** to initialize objects.
- **Override** methods and constructors.
- Define **fields**, including **private fields** (ES2020+).
- Add **static members** to classes for shared properties or methods.
- Implement **class inheritance** with the extends keyword.
- Discuss the differences between **class syntax** and traditional prototypes.

Functions

More on Functions

- Use **default parameters** to assign fallback values.
- Apply **rest/variadic parameters**.
- Use **destructured parameters** to improve readability.
- Combine **default values** with destructured parameters for cleaner function signatures.
- Explore the **arguments** object and its limitations in modern JavaScript.
- Understand **function scope** and the role of the **this** keyword.

Functional JavaScript

- Understand the principles of **Functional Programming**.
- Explore Functional Programming in **JavaScript**.
- Deal with **state** and avoid **mutability**.
- Use **functions as values** for dynamic programming.

- Write and use **higher-order functions** to pass and return functions.
- Using **clojures**
- Define **lambda expressions** for inline function behavior.
- Use **arrow functions** and understand their differences from traditional functions.
- Discuss the importance of **pure functions** and avoiding side effects.
- Combine functions using the **pipeline operator** (|>) for cleaner composition
- Implement **currying** to break down functions into smaller, single-argument functions.
- Use **recursion** to solve iterative problems in a functional style.
- Introduce popular **functional libraries**

Functional Style Arrays and Iterables

- Recap **arrays** and their role as iterables in JavaScript.
- Use array **higher-order functions** for clean and functional operations.
- Transform arrays using **map** as a functor.
- **Filter** arrays to extract elements based on conditions.
- Apply **predicate logic** to arrays for testing values.
- Find elements in arrays using **find** and **findIndex**.
- Combine array methods (**map**, **filter**, **reduce**).

Asynchronous Programming

- Use **Promise** for handling asynchronous code.
- Define and manage **promises**.
- Handle promise **errors** with .catch and finally.
- Combine multiple promises with **Promise.all/any/race**.
- Use **async/await** for cleaner asynchronous code.
- Handle errors in **async/await** with try/catch.

Testing / BDD

- Introduce the importance of **testing** in software development.
- Discuss different testing techniques (**unit, BDD, e2e**).
- Introduce and use modern tools like **Jest** and **Vite-Test**.
- Write **Suites** and **Specs** for structured test cases.
- Test **asynchronous code** effectively.
- Use **Mocking** and **Stubbing** with tools like **Spies**.
- Mock **time** for testing time-sensitive code.

JavaScript in the Browser

Introduction

- Add JavaScript to pages: **unobtrusive**, **inline**, and **external** scripts.
- Explore ways of loading external scripts: **async** and **defer** attributes.

JavaScript Modules in the Browser

- Introduce **ES Modules** (<script type="module">) for modern browsers.
- Compare **module scripts** and traditional scripts.
- Use **import** and **export** to structure code across files.
- Understand the benefits of **deferred execution** with modules.
- Learn about **dynamic imports** for lazy loading (import()).
- Discuss **module caching** and the role of the browser cache.
- Use tools like **Vite** or **ESBuild** to optimize module bundling.

DOM Manipulation and Events

- Introduce the **Window** and **Document** objects
- Understand the structure of a **DOM**
- **Navigating** the DOM
- **Adding and removing** elements
- **Changing** element properties (inner text, attributes, properties)
- Understand the **event loop**, the **call stack**, and how JavaScript executes in the browser.
- Working with **events**
- Writing **event handlers**
- Understand **event bubbling**
- **Cancelling** default behaviour
- Using **XHR** to interact with servers
- Using the **fetch API**
- Using the **history** API