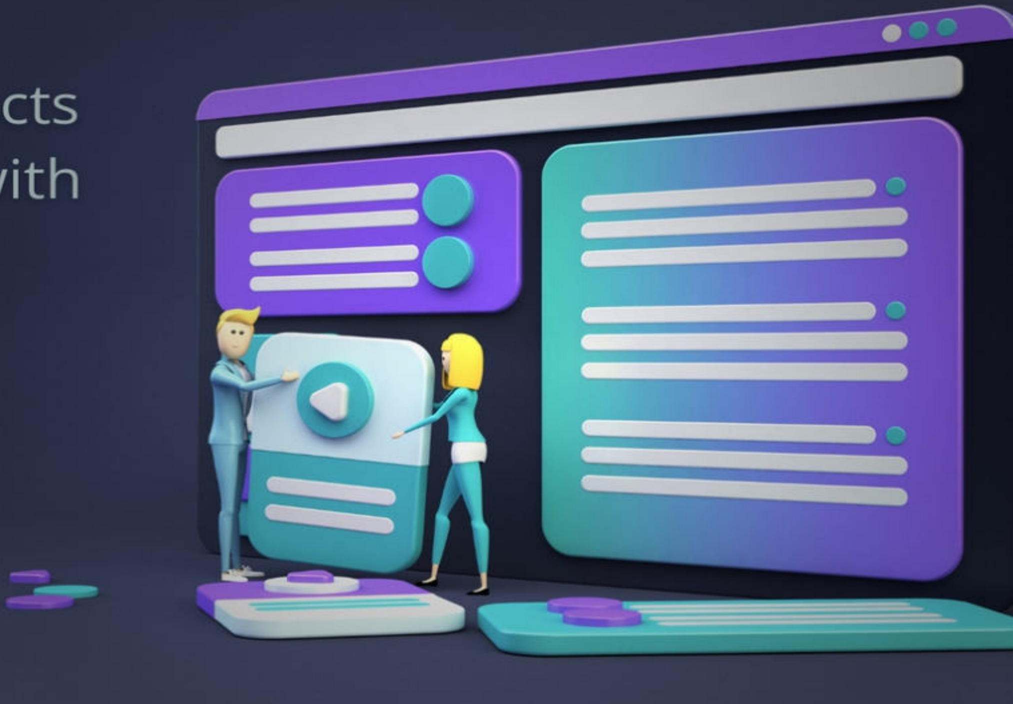
15 Web Projects With Vanilla JavaScript

Build 15 mini frontend projects from scratch with HTML5, CSS & JavaScript.



Contents

Section 1: Introduction

Chapter 1: Welcome To The Course

Chapter 2: Getting Setup

Section 2: Project 1 - Form Validator | Intro Project

Chapter 3: Project Intro

Chapter 4: Project HTML

Chapter 5: Project CSS

Chapter 6: Adding Simple Validation

Chapter 7: Check Required & Refactor

Chapter 8: Check Length, Email & Password Match

Section 3: Project 2 - Movie Seat Booking | DOM & Local Storage

Chapter 9: Project Intro

Chapter 10: Project HTML

Chapter 11: Project CSS

Chapter 12: Selecting Movie & Seats From UI

Chapter 13: Save Data To Local Storage

Chapter 14: Populate UI With Saved Data

Section 4: Project 3 - Custom Video Player | HTML5 Video API

Chapter 15: Project Intro

Chapter 16: Project HTML

Chapter 17: Project CSS

Chapter 18: Play, Pause & Stop

Chapter 19: Video Progress Bar & Timestamp

Section 5: Project 4 - Exchange Rate Calculator | Fetch & JSON Intro

Chapter 20: Project Intro

Chapter 21: Project HTML

Chapter 22: Project CSS

Chapter 23: A Look at JSON & Fetch

Chapter 24: Fetch Rates & Update DOM

Section 6: Project 5 - DOM Array Methods | forEach, map, filter, sort, reduce

Chapter 25: Project Intro

Chapter 26: Project UI

Chapter 27: Generate Random Users - Fetch w/ Async/Await

Chapter 28: Output Users - forEach() & DOM Methods

Chapter 29: Double Money - map()

Chapter 30: Sort By Richest - sort()

Chapter 31: Show Millionaires - filter()

Chapter 32: Calculate Wealth - reduce()

Section 7: Project 6 - Menu Slider & Modal | DOM & CSS

Chapter 33: Project Intro

Chapter 34: Project HTML

Chapter 35: Navbar Styling

Chapter 36: Header & Modal Styling

Chapter 37: Menu & Modal Toggle

Section 8: Project 7 - Hangman Game | DOM, SVG, Events

Chapter 38: Project Intro

Chapter 39: Draw Hangman With SVG

Chapter 40: Main Styling

Chapter 41: Popup & Notification Styling

Chapter 42: Display Words Function

Chapter 43: Letter Press Event Handler

Chapter 44: Wrong Letters & Play Again

Section 9: Project 8 - Meal Finder | Fetch & MealDB API

Chapter 45: Project Intro

Chapter 46: Project HTML & Base CSS

Chapter 47: Search & Display Meals From API

Chapter 48: Show Single Meal Page

Chapter 49: Display Random Meal & Single Meal CSS

Section 10: Project 9 - Expense Tracker | Array Methods & Local Storage

Chapter 50: Project Intro

Chapter 51: Project HTML

Chapter 52: Project CSS

Chapter 53: Show Transaction Items

Chapter 54: Display Balance, Income & Expense

Chapter 55: Add & Delete Transactions

Chapter 56: Persist To Local Storage

Section 11: Project 10 - Infinite Scroll Posts | Fetch, Async/Await, CSS Loader

Chapter 57: Project Intro

Chapter 58: Project HTML

Chapter 59: Project CSS & Loader Animation

Chapter 60: Get Initial Posts From API

Chapter 61: Add Infinite Scrolling

Chapter 62: Filter Fetched Posts

Section 12: Project 11 - Speech Text Reader | Speech Synthesis

Chapter 63: Project Intro

Chapter 64: HTML & Output Speech Boxes

Chapter 65: Project CSS

Chapter 66: Get Speech Voices

Chapter 67: Speech Buttons

Chapter 68: Change Voice & Custom Text Box

Section 13: Project 12 - Relaxer App | CSS Animations, setTimeout

Chapter 69: Project Intro

Chapter 70: Creating The Large Circle

Chapter 71: Creating & Animating The Pointer

Chapter 72: Breath Animation With JS Trigger

Section 14: Project 13 - New Year Countdown | DOM, Date & Time

Chapter 73: Project Intro

Chapter 74: Landing Page HTML & Styling

Chapter 75: Create The Countdown

Chapter 76: Dynamic Year & Spinner

Section 15: Project 14 - Sortable List | Drag & Drop API

Chapter 77: Project Intro

Chapter 78: Insert List Items Into DOM

Chapter 79: Scramble List Items

Chapter 80: Core CSS

Chapter 81: Drag & Drop Functionality

Chapter 82: Check Order

Section 16: Project 15 - Breakout Game | HTML5 Canvas API

Chapter 83: Project Intro

Chapter 84: Creating & Styling The Page

Chapter 85: Canvas Plan Outline

Chapter 86: Draw Ball, Paddle & Score

Chapter 87: Creating The Bricks

Chapter 88: Move Paddle

Chapter 89: Move Ball & Break Bricks

Chapter 90: Lose & Reset Game

~ Conclusion

Section 1:

Introduction

Welcome To The Course

First and foremost, let me express my sincere gratitude for picking up “15 Web Projects With Vanilla JavaScript.” This coursebook is your gateway to stepping into the world of web development, where you’ll be sculpting projects using the fundamental tools of the web: HTML5, CSS, and JavaScript. As you turn the pages of this coursebook, you will embark on an immersive journey, one that promises not only to equip you with the necessary skills but also to offer an engaging hands-on experience.

What Awaits You?

Over the next several sections, we will delve deep into a series of projects, each curated to provide a balanced mixture of theory and practice. From creating form validators and custom video players to concocting games like Hangman and Breakout, every project is designed with two primary objectives:

1. Skill Acquisition: Ensuring you gain a concrete understanding of core web development concepts.

2. Skill Application: Enabling you to apply the acquired knowledge in real-world scenarios.

Why “Vanilla” JavaScript?

In the vast ocean of web development, numerous frameworks and libraries promise faster development and nifty features. While these are excellent tools for seasoned developers, it’s crucial for beginners to grasp the fundamental, raw power of plain JavaScript. By focusing on “vanilla” JavaScript, we ensure that you understand the core of web scripting, making it easier for you to adapt to any library or framework in the future.

Who Is This Coursebook For?

This coursebook caters to a wide audience:

- Beginners who have dabbled a bit in HTML, CSS, and JS and are eager to build projects.

- Intermediate developers looking to solidify their understanding and gain more hands-on experience.

- Educators and Instructors seeking a structured curriculum for teaching web development.

How To Get The Most Out Of This Coursebook

1. Follow Along Actively: As this is a project-based coursebook, always have your development environment ready. Type out the code snippets, experiment with them, break them, and fix them.

2. Practice Makes Perfect: At the end of some chapters, I’ve included optional exercises. Engage with them to cement your understanding.

3. Stay Curious: Whenever a new concept is introduced, take a moment to explore it further. The more you challenge yourself, the better you'll grasp the nuances of web development.

Prerequisites

While this coursebook is designed to be comprehensive, having a foundational understanding of HTML, CSS, and JS will be beneficial. If you're an absolute beginner, I'd recommend starting with my "Modern HTML/CSS From The Beginning" and "Modern JS From The Beginning" courses on Udemy. This will give you a solid footing to tackle the projects in this coursebook.

Wrapping Up

Remember, every coder, no matter how skilled, started from scratch. With patience, persistence, and the right resources, you too can carve out your niche in the world of web development.

So, as we stand at the threshold of this exciting journey, take a deep breath, roll up your sleeves, and let's dive into the world of web projects with vanilla JavaScript!

Getting Setup

Welcome to the first step of our exciting journey through the world of web development! Before diving into the various projects we'll tackle, it's essential to ensure our development environment is correctly set up. This chapter will guide you through the installation and configuration of all the tools and software required to follow along with the projects in this book.

2.1. System Requirements

Before we start, ensure you have a computer with:

- An operating system like Windows, macOS, or Linux.

- A minimum of 4GB RAM (8GB recommended for smoother performance).

- A stable internet connection.

2.2. Text Editor

Every developer needs a reliable text editor. Here are some popular choices:

- Visual Studio Code (VS Code): This free, open-source editor from Microsoft is widely adopted because of its ease of use, extensive feature set, and vast library of extensions. [Download here.](https://code.visualstudio.com/)

- Sublime Text: Another favorite among developers, Sublime Text offers a clean interface and robust performance. [Download here.](https://www.sublimetext.com/)

- Atom: Developed by GitHub, Atom is a free, open-source editor that’s customizable and beginner-friendly. [Download here.](https://atom.io/)

Recommended Extensions for VS Code: If you choose VS Code, consider installing extensions like `Live Server`, `Prettier`, and `ESLint` for an enhanced development experience.

2.3. Web Browsers

To view and test our projects, you’ll need a modern web browser. While most browsers will work, the following are recommended due to their extensive developer tools:

- Google Chrome [Download here.](https://www.google.com/chrome/)

- Mozilla Firefox [Download here.](https://www.mozilla.org/en-US/firefox/new/)

- Microsoft Edge (Chromium version) [Download here.](https://www.microsoft.com/edge)

2.4. Browser Developer Tools

Modern browsers come equipped with developer tools, allowing you to inspect, debug, and profile your web projects. Familiarize yourself with the developer tools of your chosen browser. They’re essential for diagnosing issues and understanding how your code operates within the browser.

2.5. Node.js and npm

Some projects in this book will require the use of Node.js and npm (node package manager). Node.js allows you to run JavaScript outside the browser, while npm is the world’s largest software registry.

1. Installing Node.js: Visit the [official Node.js website](https://nodejs.org/) and download the LTS (Long Term Support) version. The installation process is straightforward.

2. Verifying Installation: Once installed, open your terminal or command prompt and run the following commands to ensure both Node.js and npm are installed:

“`bash

```
node -v

npm -v
```

“`

2.6. Version Control with Git

Git is a distributed version control system that helps track changes in your code. It’s highly recommended for every developer.

1. Installing Git: Download and install Git from the [official website](https://git-scm.com/).

2. Verifying Installation: In your terminal or command prompt, type:

“`bash

git —version

“`

2.7. Directory Structure

Maintain a clean directory structure for your projects. Here’s a suggested structure:

“`

```
/web-projects
    /project-1-form-validator
        /css
        /js
        index.html
    /project-2-movie-seat-booking
        /css
        /js
        index.html
    … and so on
```

“`

By keeping your files organized, you ensure that as your projects grow, you won’t get lost in a maze of files and folders.

2.8. Wrapping Up

Congratulations! You’ve set up your development environment. As you proceed through the book, you’ll gain hands-on experience and deepen your understanding of the tools we've discussed in this chapter. Remember, web development is a journey, and every project will enhance your skills and knowledge.

Now that we're all set, let's dive into our first project in the next chapter!

Section 2: Project 1 - Form Validator | Intro Project

Project Intro

Welcome to our very first project of this exciting journey: The Form Validator! Forms are an integral part of the web, and no matter how the landscape of web development changes, form validations will always be a necessary skill for web developers. Whether you're building a sign-up page, a login form, or any other input system, ensuring the accuracy and correctness of data is vital. This introductory project is designed to provide a strong foundation for those validations.

Why Form Validation?

In the vast world of web development, why did we choose to start with form validation? Here are a few reasons:

1. Universality: Nearly every website you visit has some form - be it a search bar, a login page, or a feedback form.

2. User Experience: A good form validation system provides immediate feedback to users, ensuring they provide the right information.

3. Security: Ensuring data is validated on the front-end can also be a first line of defense against malicious inputs.

What We’ll Build

We’re going to design a simple form that asks users for a username, email, and password. This form will have the following features:

- Fields will be checked to ensure they aren’t left empty.

- Email addresses will be validated to ensure they’re in a proper format.

- Passwords will be validated for length and to ensure a confirm password matches.

Technologies & Techniques

Although the main focus here is on JavaScript, HTML and CSS will play crucial roles. Here’s a glimpse of what we’ll use:

1. HTML: Structure our form and inputs.

2. CSS: Stylish visual cues to indicate errors or successful inputs.

3. JavaScript: The heart of our validation logic. We’ll be using vanilla JavaScript, without any libraries or frameworks.

Prerequisites

You should have a basic understanding of HTML, CSS, and JavaScript. If you’re completely new or need a refresher, it’s advisable to first check out the “Modern HTML/CSS From The Beginning” and “Modern JS From The Beginning” courses on Udemy.

What You’ll Gain

By the end of this project, you’ll:

1. Understand how to intercept form submissions using JavaScript.

2. Learn how to traverse and manipulate the DOM to highlight errors.

3. Be familiar with basic string methods and regular expressions for validation.

4. Have a foundational project in your portfolio to showcase basic front-end validation skills.

Let’s Get Started!

Excited? You should be! This foundational project will set the pace for the other, more complex projects we’ll tackle in this course. So, buckle up, and let’s dive into the world of form validation!

In the next chapter, we’ll begin with the structure of our form by creating its HTML layout. Let’s begin this amazing journey together!

Project HTML

Welcome to Chapter 4, where we’re going to dive into the heart of our first project: the Form Validator. Before we begin styling and adding dynamic functionality, we need a solid foundation. That foundation is our HTML structure.

Understanding the Project:

Before diving into the code, let’s grasp the core concept of our project. A Form Validator ensures that the data entered by users meets specific criteria. This is important not only for user experience but also for security reasons.

Our form will require:

- A username

- An email address

- A password

- A password confirmation

Each of these fields will have its own validation criteria, which we’ll implement in the subsequent chapters.

Setting Up the Base HTML:

Begin with the foundational structure of an HTML document:

“`html

```
<!DOCTYPE html>
<html lang=“en”>
<head>
    <meta charset=“UTF-8”>
    <meta name=“viewport” content=“width=device-width,
initial-scale=1.0”>
    <title>Form Validator</title>
    <!— Your styles will be linked here later —>
</head>
<body>
</body>
</html>
```

“`

Building the Form:

Inside the `<body>`, we’ll structure our form:

“`html

```
<div class=“container”>
    <form id=“form” class=“form”>
        <h2>Register With Us</h2>
        <!— Username Field —>
        <div class="form-control">
         <label for="username">Username</label>
         <input type="text" id="username"
placeholder="Enter username">
         <!— Error message will be displayed here —>
         <small>Error message</small>
        </div>
        <!— Email Field —>
        <div class="form-control">
         <label for="email">Email</label>
         <input type="email" id="email" placeholder="Enter
email">
         <small>Error message</small>
        </div>
        <!— Password Field —>
        <div class="form-control">
         <label for="password">Password</label>
         <input type="password" id="password"
placeholder="Enter password">
         <small>Error message</small>
        </div>
        <!— Password Check Field —>
        <div class="form-control">
         <label for="password2">Confirm
Password</label>
         <input type="password" id="password2"
placeholder="Enter password again">
         <small>Error message</small>
        </div>
        <!— Submit Button —>
        <button type="submit">Submit</button>
    </form>
</div>
“`
```

Here’s what we’ve done:

1. Container: All our form elements are wrapped within a `<div>` container to help with styling and centering the form on the page.

2. Form Controls: Each input field and its label are enclosed within a `form-control` div. This setup will help us in styling and showing error messages related to each field.

3. Error Messages: A `<small>` tag is used to display error messages. Initially, it contains a generic error message, but this will change dynamically as we validate the form.

4. Submit Button: This button will be used to trigger our validation. When the user presses it, the form will either submit (if everything is valid) or show relevant error messages.

Conclusion:

The HTML structure for our Form Validator is set up. As simple as it might look now, this structure is the backbone of our dynamic functionality.

Always remember, the key to creating effective web projects is to maintain a clean and understandable structure in the HTML, which makes styling and scripting a lot smoother. Onward to the next chapter!

Project CSS

Welcome to the styling portion of our first project: the Form Validator. As you may recall, the aim of this project is to introduce you to the basics of web development, and one of the essential aspects of building a visually appealing web application is styling. CSS (Cascading Style Sheets) allows us to apply styles to our HTML documents, making them more aesthetically pleasing and user-friendly.

For our form validator, we’re going to focus on creating a neat, simple, and intuitive design that guides users through the process of inputting their details. The goal here is not just to make things look nice but also to use styles as a tool for better usability.

---

1. Base Styles:

Let’s begin by adding some general styles to our form:

“`css

```css
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    background-color: f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
.container {
    background-color: fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
    width: 400px;
```

}
“`

These base styles will provide a pleasant gray background, center our form on the page, and give the form container a nice shadow and rounded corners.

---

2. Form Styles:

Now, let’s style the form elements.

“`css

```
form {
    width: 100%;
}
form label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}
form input[type=“text”],
form input[type=“password”],
form input[type=“email”] {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid ccc;
    border-radius: 4px;
}
form button {
    display: block;
    width: 100%;
    padding: 10px;
    border: none;
    border-radius: 4px;
    background-color: 333;
    color: fff;
    cursor: pointer;
}
```

“`

These styles make our form inputs stretch to take up the full width of the container, making it easier for users to click into them. They also apply some padding for better visual spacing and a neat appearance.

---

3. Validation Feedback Styles:

To provide immediate feedback to users, let’s add some styles to highlight invalid and valid fields:

“`css

```
.invalid {
    border-color: red;
    background-color: ffe6e6;
}
.valid {
    border-color: green;
    background-color: e6ffe6;
}
.error-message {
    color: red;
    font-weight: bold;
    font-size: 0.8rem;
    margin-top: -10px;
    margin-bottom: 10px;
}
“`
```

The `.invalid` and `.valid` classes provide a visual cue about the input field status by changing border colors. The `.error-message` class will be used for displaying specific error messages next to our input fields.

---

4. Responsive Design:

Considering the growing number of mobile users, our form should be responsive:

```
“`css
@media screen and (max-width: 420px) {
    .container {
        width: 90%;
    }
}
“`
```

This media query ensures that on smaller screens, our form takes up most of the viewport width, leaving a small margin on the sides.

Conclusion:

CSS is a powerful tool that goes beyond just making things look good. With the styles we’ve added, our form is now not only visually appealing but also user-friendly. The visual cues from our validation feedback styles will guide users through the form, making the entire process intuitive and efficient.

Adding Simple Validation

In the world of web development, one thing you’ll come across often is the need to validate forms. Whether it’s a sign-up form, a login page, or a settings update page, it’s essential to ensure that the data users provide is correct and secure. In this chapter, we will delve deep into creating simple form validation for our introductory project using vanilla JavaScript.

Why is validation important?

Before diving into the code, let’s understand why validation is crucial:

1. User Experience (UX): Proper validation ensures users fill out forms correctly, reducing errors and misunderstandings.

2. Security: Validating inputs helps prevent malicious users from sending harmful data to your server.

3. Data Integrity: By validating forms, you ensure that the data sent to the server adheres to the expected format.

Getting Started with Simple Validation

For our intro project, we have a basic form that we want users to fill out. Let’s assume it contains fields like `username`, `email`, and `password`. Our task is to validate these fields using simple checks.

HTML Structure:

“`html

```
<form id="userForm">
    <input type="text" id="username" placeholder="Username">
    <input type="email" id="email" placeholder="Email">
    <input type="password" id="password" placeholder="Password">
    <button type="submit">Submit</button>
    <p id=“error-message”></p>
</form>
```

“`

The `error-message` paragraph is where we will display any validation errors to the user.

JavaScript Validation:

Begin by selecting our form and adding an event listener for the `submit` event.

“`javascript

```
const form = document.getElementById(‘userForm’);
form.addEventListener(‘submit’, function(e) {
    e.preventDefault(); // Prevents the form from submitting
    validateForm();
});
```

“`

Next, we’ll define our `validateForm` function:

“`javascript

```
function validateForm() {
    const username = document.getElementById(‘username’).value;
    const email = document.getElementById(‘email’).value;
    const password = document.getElementById(‘password’).value;
    const errorMessage = document.getElementById(‘error-message’);
    // Resetting the error message
    errorMessage.textContent = ”;
    // Validate Username
    if(username === ” || username.length < 3) {
        errorMessage.textContent += ‘Username must be at least 3 characters long.\n’;
        return;
    }
    // Validate Email
    const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,6}$/;
    if(!emailPattern.test(email)) {
        errorMessage.textContent += ‘Please enter a valid email address.\n’;
        return;
    }
    // Validate Password
    if(password === ” || password.length < 8) {
        errorMessage.textContent += ‘Password must be at least 8 characters long.\n’;
        return;
    }
}
“`
```

Here’s a brief overview of our validation checks:

- Username: It shouldn’t be empty, and its length should be at least 3 characters.

- Email: We use a regular expression (regex) to ensure the email is in the correct format.

- Password: It shouldn’t be empty, and its length should be at least 8 characters.
