10 Static Web Projects for Students

Using HTML, CSS and JavaScript

Djakba Faustin
`djakbafaustin20@gmail.com`

November 6, 2025

Contents

Introduction

This document presents ten static web projects designed for students learning HTML, CSS, and JavaScript. Each project is educational, increases in difficulty, and helps practice fundamental concepts of web development. The projects are static (no backend) and include learning objectives, a suggested structure, and code examples to guide students.

How to use this document:

- Start with the simpler projects and progress to more advanced ones.
- Read the objectives and suggested structure before attempting the code.
- Use the code excerpts as starting points and customize them.

Chapter 1

Project 1: Personal Introduction Page

Description

A simple web page to introduce yourself: your name, hobbies, skills, and a photo.

Learning Objectives

- Use semantic HTML tags (`<header>`, `<section>`, `<footer>`).
- Apply CSS for fonts, colors, and layout.
- Add a simple JavaScript interaction (alert).

Suggested Structure

- **HTML:** Section with an image, heading, paragraph, and button.
- **CSS:** Center the content and style the image (circular border).
- **JavaScript:** Display a welcome alert on button click.

Code Excerpt

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Personal Page</title>
  <style>
    body { text-align: center; font-family: Arial, sans-serif; }
    img { border-radius: 50%; width: 150px; }
  </style>
</head>
<body>
  <h1>I am [Your Name]</h1>
  <img src="photo.jpg" alt="Photo">
  <p>I enjoy programming and learning!</p>
  <button onclick="sayHello()">Discover me</button>

  <script>
```

```
    function sayHello() {
      alert("Welcome to my page!");
    }
  </script>
</body>
</html>
```

Listing 1.1: Personal introduction page

Advanced Task

Add a downloadable CV link.

Chapter 2

Project 2: Shopping List

Description

An application to add and remove items from a shopping list.

Learning Objectives

- Manipulate the DOM (createElement, appendChild).
- Handle events (click, input).
- Style the list and input field.

Suggested Structure

- **HTML:** Text input, add button, and an unordered list (`<ul>`).
- **CSS:** Style the list (bullets, spacing).
- **JavaScript:** Add and remove list items.

Code Excerpt

```
<input type="text" id="item" placeholder="Add an item">
<button onclick="addItem()">Add</button>
<ul id="list"></ul>

<script>
function addItem() {
  let item = document.getElementById("item").value;
  if (item) {
    let li = document.createElement("li");
    li.innerText = item;
    li.onclick = () => li.remove();
    document.getElementById("list").appendChild(li);
    document.getElementById("item").value = "";
  }
}
</script>
```

Listing 2.1: Shopping list

Advanced Task

Add a button to clear the entire list.

Chapter 3

Project 3: Basic Calculator

Description

A calculator that can add, subtract, multiply, and divide.

Learning Objectives

- Use CSS Grid for button layout.
- Handle user input and validation (e.g., avoid division by zero).

Suggested Structure

- **HTML:** Input fields, operation buttons, and result display.
- **CSS:** Use `display: grid` for layout.
- **JavaScript:** Compute results based on the selected operation.

Code Excerpt

```
<input type="number" id="num1" placeholder="Number 1">
<input type="number" id="num2" placeholder="Number 2">
<div class="calc">
  <button onclick="calculate('+')">+</button>
  <button onclick="calculate('-')">-</button>
  <button onclick="calculate('*')">*</button>
  <button onclick="calculate('/')">/</button>
</div>
<p> Result: <span id="result">0</span></p>

<script>
function calculate(op) {
  let num1 = Number(document.getElementById("num1").value);
  let num2 = Number(document.getElementById("num2").value);
  let result;

  if (op === '+') result = num1 + num2;
  else if (op === '-') result = num1 - num2;
  else if (op === '*') result = num1 * num2;
  else if (op === '/') {
    if (num2 === 0) {
      alert("Cannot divide by zero!");
      return;
    }
    result = num1 / num2;
  }
  document.getElementById("result").innerText = result;
}
</script>
```

Listing 3.1: Basic calculator

Advanced Task

Show an error message for invalid inputs.

Chapter 4

Project 4: Photo Gallery

Description

A grid of images with a hover effect (scale or change opacity).

Learning Objectives

- Use CSS Flexbox or Grid.
- Apply transitions on hover (`:hover`, `transform`).
- Optionally display clicked image at a larger size.

Suggested Structure

- **HTML:** A container `<div>` with multiple `<img>` elements.
- **CSS:** `display:flex` and transitions for hover effects.
- **JavaScript:** (Optional) Show image in full-screen/modal.

Code Excerpt

```html
<style>
.gallery { display: flex; flex-wrap: wrap; gap: 10px; }
.gallery img { width: 150px; transition: transform 0.3s; }
.gallery img:hover { transform: scale(1.1); }
</style>

<div class="gallery">
  <img src="photo1.jpg" alt="Photo 1">
  <img src="photo2.jpg" alt="Photo 2">
  <img src="photo3.jpg" alt="Photo 3">
</div>
```

Listing 4.1: Photo gallery

Advanced Task

Add captions under the images.

Chapter 5

Project 5: Character Counter

Description

A tool to count the characters and words in a text input.

Learning Objectives

- Use the `input` event.
- Count words using regular expressions.
- Style the textarea.

Suggested Structure

- **HTML:** `<textarea>` and spans for counters.
- **CSS:** Style the textarea.
- **JavaScript:** Real-time counting.

Code Excerpt

```
<textarea id="text" placeholder="Type your text"></textarea>
<p>Characters: <span id="charCount">0</span></p>
<p>Words: <span id="wordCount">0</span></p>

<script>
document.getElementById("text").addEventListener("input", function () {
  let text = this.value;
  document.getElementById("charCount").innerText = text.length;
  let words = text.trim().split(/\s+/).filter(word => word.length > 0).length;
  document.getElementById("wordCount").innerText = words;
});
</script>
```

Listing 5.1: Character and word counter

Advanced Task

Add a sentence counter.

Chapter 6

Project 6: Digital Clock

Description

A digital clock that displays the current time and updates every second.

Learning Objectives

- Use `setInterval`.
- Work with the JavaScript `Date` object.
- Style the clock using CSS.

Suggested Structure

- **HTML:** A `<div>` for the clock display.
- **CSS:** Big font and color styling.
- **JavaScript:** Update using `toLocaleTimeString`.

Code Excerpt

```
<style>
#clock { font-size: 2em; text-align: center; color: #333; }
</style>

<div id="clock"></div>

<script>
function updateClock() {
  let now = new Date();
  document.getElementById("clock").innerText = now.toLocaleTimeString();
}
setInterval(updateClock, 1000);
updateClock();
</script>
```

Listing 6.1: Digital clock

Advanced Task

Add the date and toggle between 12-hour and 24-hour formats.

Chapter 7

Project 7: Registration Form

Description

A registration form with validation for name, email, and password.

Learning Objectives

- Validate form fields using JavaScript (email pattern, password strength).
- Style form fields and show validation errors.
- Use the `onsubmit` handler to prevent submission on errors.

Suggested Structure

- **HTML:** Form elements for name, email, password.
- **CSS:** Style inputs and display errors in red.
- **JavaScript:** Validate on submit.

Code Excerpt

```
<form onsubmit="validate(event)">
  <input type="text" id="name" placeholder="Name">
  <input type="email" id="email" placeholder="Email">
  <input type="password" id="password" placeholder="Password">
  <button type="submit">Register</button>
</form>
<p id="error" class="error"></p>

<script>
function validate(event) {
  event.preventDefault();
  let name = document.getElementById("name").value;
  let email = document.getElementById("email").value;
  let password = document.getElementById("password").value;
  let errorElem = document.getElementById("error");

  if (!name || !email.includes("@") || password.length < 6) {
    errorElem.innerText = "Invalid fields. Please check your inputs.";
  } else {
    errorElem.innerText = "Registration successful!";
    // Optionally submit form via AJAX or perform other actions
  }
}
</script>
```

Listing 7.1: Registration form with basic validation

Advanced Task

Add password confirmation and stronger password checks.

Chapter 8

Project 8: Simple Matching Game (Pairs)

Description

A simple memory matching game where the user finds matching pairs of cards.

Learning Objectives

- Dynamically create HTML elements.
- Use arrays and event handlers.
- Style card flip effects using CSS transforms.

Suggested Structure

- **HTML:** A game container `<div>` for cards.
- **CSS:** Card styles and flip animation.
- **JavaScript:** Shuffle the cards and check pairs.

Code Excerpt

```
<style>
.card { width: 100px; height: 100px; background: #ccc; margin: 5px; display:
    inline-block; }
.flipped { background: #fff; }
</style>

<div id="game"></div>

<script>
let cards = ["A", "A", "B", "B"];
cards.sort(() => Math.random() - 0.5);
let game = document.getElementById("game");

cards.forEach(card => {
  let div = document.createElement("div");
  div.className = "card";
  div.onclick = () => div.classList.toggle("flipped");
  // Optionally show card content when flipped
  game.appendChild(div);
});
</script>
```

Listing 8.1: Simple pairs game

Advanced Task

Add a counter for the number of attempts and a timer.

Chapter 9

Project 9: Color Generator

Description

A page that generates a random color on button click and applies it to the background.

Learning Objectives

- Generate random numbers and convert them to hex colors.
- Change style properties dynamically via JavaScript.
- Create an attractive interface.

Suggested Structure

- **HTML:** A button and a display area for the color code.
- **CSS:** Center content and style the button.
- **JavaScript:** Generate and apply random hex color.

Code Excerpt

```
<style>
body { text-align: center; font-family: Arial, sans-serif; }
button { padding: 10px 20px; }
</style>

<button onclick="changeColor()">Change color</button>
<p id="code"></p>

<script>
function changeColor() {
  let color = "#" + Math.floor(Math.random() * 16777215).toString(16).padStart
      (6,'0');
  document.body.style.backgroundColor = color;
  document.getElementById("code").innerText = color;
}
</script>
```

Listing 9.1: Random color generator

Advanced Task

Display the hexadecimal color code and provide a button to copy it.

Chapter 10

Project 10: Manual Slideshow

Description

A manual slideshow with previous/next navigation.

Learning Objectives

- Manage an array of images and navigate using index.
- Handle `onclick` for navigation.
- Use CSS transitions for smooth effects.

Suggested Structure

- **HTML:** An `<img>` element and previous/next buttons.
- **CSS:** Size the image and center navigation.
- **JavaScript:** Change the `src` attribute based on index.

Code Excerpt

```
<style>
img { width: 300px; height: 200px; object-fit: cover; }
button { margin: 10px; }
</style>

<img id="slide" src="image1.jpg" alt="Slideshow">
<button onclick="prev()">Previous</button>
<button onclick="next()">Next</button>

<script>
let images = ["image1.jpg", "image2.jpg", "image3.jpg"];
let index = 0;

function next() {
  index = (index + 1) % images.length;
  document.getElementById("slide").src = images[index];
}
function prev() {
  index = (index - 1 + images.length) % images.length;
  document.getElementById("slide").src = images[index];
}
</script>
```

Listing 10.1: Manual slideshow

Advanced Task

Add a smooth transition between slides and an autoplay option.

Tips for Students

- **Progression:** Start with simple projects (e.g., Personal Page, Photo Gallery, Clock) and move to interactive ones.
- **Personalization:** Add your own style—colors, fonts, and animations—to make projects unique.
- **Testing:** Check your projects on different browsers and screen sizes (mobile/responsive).
- **Resources:** Consult MDN Web Docs and other reputable web documentation.
- **Advanced work:** For each project, implement the advanced task to deepen your learning.

Conclusion

These ten projects provide a practical and progressive path to learn HTML, CSS, and JavaScript. They can be adapted and expanded as you grow your skills. Practice often, read documentation, and build personal variations of each project.

Good luck and happy coding!