Vlad Bogdantsev

https://github.com/vladproduction/My-Pet-Projects/tree/main/tax-form-declaration

The purpose of this task is to show how to use HTML, CSS, and JavaScript (if necessary) to design a web page.

Designing a Web Page: “TAX DECLARATION FORM”

**Task completed by following criteria:**

**1.** Functional and non-functional requirements for the solution are provided.

**2.** Web page design complete and unambiguous.

**3.** HTML is semantically, and classes are used.

**4.** CSS stored in external files and maps to HTML using classes.

**5.** JS events is affecting the functionality of a web page.

Link to the project: https://github.com/vladproduction/My-Pet-Projects/tree/main/tax-form-declaration

1. **Functional and non-functional requirements.**

***Functional Requirements:***

1)Personal Information Input:

* Users can enter their full name.
* Users can provide their residential address.
* Users can input their Social Security Number (SSN).

2)Income Details:

* Users can enter their primary income (salary).
* Users can add other sources of income (e.g., freelance work, investments).

3)Expense Deductions (list deductible expenses):

* Medical expenses.
* Educational costs.
* Other allowable deductions.

4)Form Submission:

* Users can submit the form.
* Upon submission, a confirmation message or alert should be displayed.

5)Validation:

* The form should validate that all required fields are filled out.
* Invalid entries (like incorrect formats for SSN) should be flagged with appropriate error messages.
* The user should be informed about any missing or incorrect information upon submission.

6)Reset/Draft Saving:

* Users can reset the form to clear all inputs.
* Users can save their input as a draft (optional feature).

***Non-Functional Requirements:***

1)Performance: webpage must load within 2 seconds for a better user experience.

2)Responsiveness: The web page must display correctly on various devices (desktops, tablets, smartphones) and adjust to different screen sizes.

3)Usability:

* The user interface should be intuitive and easy to navigate.
* Clear instructions should be provided near relevant input areas.

4)Security:

* User data entered in the form must be transmitted securely (e.g., using HTTPS).
* Personal data should not be stored without user consent.

1. **Page design (**Mockup Design for Tax Declaration Form**).**

***- Layout -***

1)Header Section:

* Title: "Tax Declaration Form" (Center-aligned, bold, larger font size)
* Subtitle: "Mockup Tax Declaration Form"

2)Personal Information Section:

* Label: "Personal Information" (Bold, slightly larger font)
* Full Name: [\_\_\_\_\_\_\_\_\_\_]
* Address: [\_\_\_\_\_\_\_\_\_\_]
* Social Security Number: [\_\_\_\_\_\_\_\_\_\_]

(use placeholder text in input fields for guidance, e.g., "John Doe", "123 Main St", "123-45-6789")

3)Income Details Section:

* Label: "Income Details" (Bold, slightly larger font)
* Primary Income (EURO): [\_\_\_\_\_\_\_\_\_\_]
* Additional Income (EURO): [\_\_\_\_\_\_\_\_\_\_]

(use placeholder text in input fields for guidance, e.g., "150")

4)Expense Deductions Section:

* Label: "Expense Deductions" (Bold, slightly larger font)
* Medical Expenses (EURO): [\_\_\_\_\_\_\_\_\_\_]
* Educational Costs (EURO): [\_\_\_\_\_\_\_\_\_\_]
* Other Deductible Expenses (EURO): [\_\_\_\_\_\_\_\_\_\_]

(use placeholder text in input fields for guidance, e.g., "150")

5)Submission Buttons:

* Submit Button: [ Submit ]
* Reset Button: [ Reset ]

(Buttons styled with a contrasting color to stand out)

6)Validation Messages Section:

(Use notifications below each input field to display validation messages in red/alert style)

7)Footer Section:

* Contact Information: "For assistance, contact [support-office@tax.euro.com]"

***- Styling Suggestions -***

* Color Scheme: Use a clean, professional color palette (e.g., shades of blue, grey, and white).
* Typography: Choose legible fonts and maintain hierarchy (bolder and larger for section titles, standard size for input labels).
* Spacing: Ensure adequate padding and margins between input fields and sections for a clean layout.
* Responsive Design: Ensure that the layout adjusts fluidly for different screen sizes, rearranging form elements as necessary for mobile devices.
* Icons: Consider using icons next to form labels (e.g., a user icon next to “Full Name”) to enhance visual appeal.

1. **HTML files and classes.**

**tax-declaration-form.html :**

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Tax Declaration Form</title>  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">  
 <!-- should before custom CSS file -->  
 <link rel="stylesheet" href="styles.css"> <!-- Link to external CSS file -->  
 <script src="script.js" defer></script> <!-- Link to external JS file -->  
</head>  
  
<body>  
  
<header>  
 <h1 class="main-title">Tax Declaration Form</h1> <!-- Main Title -->  
 <h2 class="sub-title">Mockup Tax Declaration Form</h2> <!-- Subtitle -->  
</header>  
  
<form id="tax-form">  
  
 <section class="personal-info">  
 <div id="current-date-time" style="text-align: center; margin-bottom: 20px; font-weight: bold;">  
 <span id="formatted-date-time"></span>  
 </div>  
 <h3><i class="fas fa-user"></i> Personal Information</h3>  
 <label for="full-name">Full Name:</label>  
 <input type="text" id="full-name" placeholder="John Doe" required class="personal-input">  
 <label for="address">Address:</label>  
 <input type="text" id="address" placeholder="123 Main St" required class="personal-input">  
 <label for="ssn">Social Security Number:</label>  
 <input type="text" id="ssn" placeholder="123-45-6789" required>  
 <div class="validation-message" id="ssn-error"></div>  
 </section>  
 <section class="income-details">  
 <h3><i class="fas fa-money-bill-wave"></i> Income Details</h3>  
 <label for="primary-income">Primary Income (EURO):</label>  
 <input type="number" id="primary-income" placeholder="150" required class="money-input">  
 <label for="additional-income">Additional Income (EURO):</label>  
 <input type="number" id="additional-income" placeholder="150" class="money-input">  
 </section>  
 <section class="expense-deductions">  
 <h3><i class="fas fa-file-invoice-dollar"></i> Expense Deductions</h3>  
 <label for="medical-expenses">Medical Expenses (EURO):</label>  
 <input type="number" id="medical-expenses" placeholder="150" class="money-input">  
 <label for="educational-costs">Educational Costs (EURO):</label>  
 <input type="number" id="educational-costs" placeholder="150" class="money-input">  
 <label for="other-expenses">Other Deductible Expenses (EURO):</label>  
 <input type="number" id="other-expenses" placeholder="150" class="money-input">  
 </section>  
 <div class="buttons">  
 <button type="submit" id="submit-btn">Submit</button>  
 <button type="reset" id="reset-btn">Reset</button>  
 </div>  
</form>  
<footer>  
 <p>For assistance, contact: <a href="mailto:support-office@tax.euro.com">support-office@tax.euro.com</a></p>  
</footer>  
</body>  
</html>

Description HTML (tax-declaration-form.html):

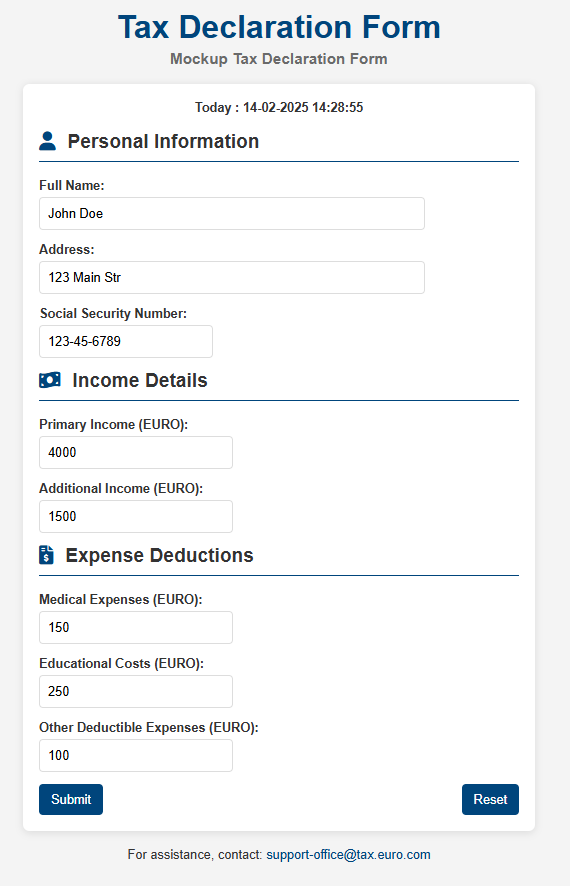
1. Header and Subtitle: Displays the title and subtitle of the form.
2. Current Date and Time: Displays the current date and time at the top of the form in the desired format.
3. Form Sections: Contains sections for personal information, income details, and expense deductions, with appropriate labels and input fields.
4. Validation Messages: Includes a validation area for displaying error messages related to the SSN.
5. Submit and Reset Buttons: Allows for form submission and resetting of the fields.
6. Footer: Provides contact information for assistance.

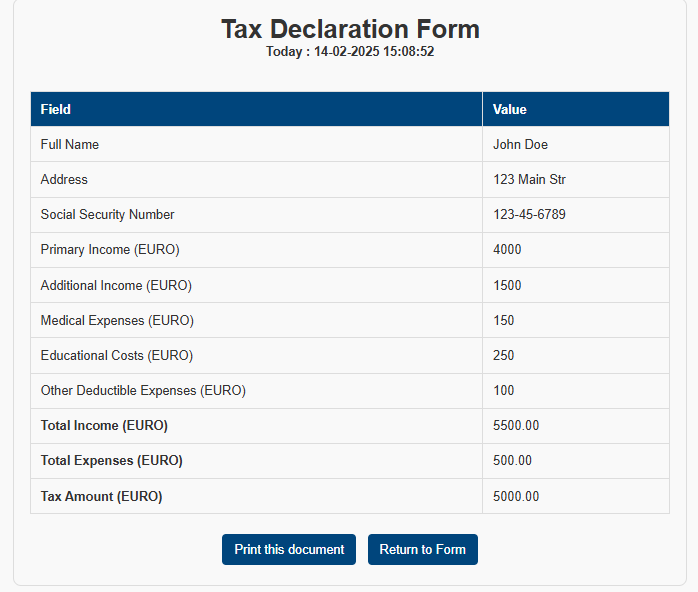
**print-preview.html :**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Print Preview</title>  
 <link rel="stylesheet" href="styles.css"> <!-- CSS styles -->  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 margin: 20px;  
 padding: 20px;  
 background-color: #f9f9f9;  
 border: 1px solid #ddd;  
 border-radius: 10px;  
 max-width: 800px; /\* Ensuring width for A4 format \*/  
 margin: auto; /\* Centering the container \*/  
 }  
 h1 {  
 text-align: center;  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
 th, td {  
 border: 1px solid #ddd;  
 padding: 12px;  
 text-align: left;  
 }  
 th {  
 background-color: #00457c;  
 color: white;  
 }  
 .button-container {  
 margin-top: 20px;  
 text-align: center;  
 }  
 button {  
 background-color: #00457c;  
 color: #fff;  
 border: none;  
 padding: 10px 15px;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 1em;  
 margin: 5px;  
 }  
 button:hover {  
 background-color: #003366;  
 }  
 </style>  
</head>  
<body>  
  
<h1>Tax Declaration Form</h1>  
<div id="current-date-time" style="text-align: center; margin-bottom: 20px; font-weight: bold;">  
 Today: <span id="formatted-date-time"></span>  
</div>  
<table>  
 <tr>  
 <th>Field</th>  
 <th>Value</th>  
 </tr>  
 <tr>  
 <td>Full Name</td>  
 <td id="full-name-preview"></td>  
 </tr>  
 <tr>  
 <td>Address</td>  
 <td id="address-preview"></td>  
 </tr>  
 <tr>  
 <td>Social Security Number</td>  
 <td id="ssn-preview"></td>  
 </tr>  
 <tr>  
 <td>Primary Income (EURO)</td>  
 <td id="primary-income-preview"></td>  
 </tr>  
 <tr>  
 <td>Additional Income (EURO)</td>  
 <td id="additional-income-preview"></td>  
 </tr>  
 <tr>  
 <td>Medical Expenses (EURO)</td>  
 <td id="medical-expenses-preview"></td>  
 </tr>  
 <tr>  
 <td>Educational Costs (EURO)</td>  
 <td id="educational-costs-preview"></td>  
 </tr>  
 <tr>  
 <td>Other Deductible Expenses (EURO)</td>  
 <td id="other-expenses-preview"></td>  
 </tr>  
 <tr>  
 <td><strong>Total Income (EURO)</strong></td>  
 <td id="total-income"></td>  
 </tr>  
 <tr>  
 <td><strong>Total Expenses (EURO)</strong></td>  
 <td id="total-expenses"></td>  
 </tr>  
 <tr>  
 <td><strong>Tax Amount (EURO)</strong></td>  
 <td id="tax-amount"></td>  
 </tr>  
</table>  
  
<div class="button-container">  
 <button id="print-btn">Print this document</button>  
 <button id="return-btn">Return to Form</button>  
</div>  
  
<script>  
 // Fetch URL parameters and populate the content  
 function getQueryParams() {  
 const params = {};  
 *window*.location.search.substr(1).split("&").forEach(function(item) {  
 const parts = item.split("=");  
 params[decodeURIComponent(parts[0])] = decodeURIComponent(parts[1].replace(/\+/g, ' ')); // Replace + with space  
 });  
 return params;  
 }  
  
 // Function to format date and time  
 function getCurrentDateTime() {  
 const now = new *Date*();  
 const day = String(now.getDate()).padStart(2, '0'); // Two-digit day  
 const month = String(now.getMonth() + 1).padStart(2, '0'); // Two-digit month (0-indexed)  
 const year = now.getFullYear();  
 const hours = String(now.getHours()).padStart(2, '0'); // Two-digit hours  
 const minutes = String(now.getMinutes()).padStart(2, '0'); // Two-digit minutes  
 const seconds = String(now.getSeconds()).padStart(2, '0'); // Two-digit seconds  
 return `Today : ${day}-${month}-${year} ${hours}:${minutes}:${seconds}`; // Return formatted date string  
 }  
  
 *document*.addEventListener('DOMContentLoaded', () => {  
 const params = getQueryParams();  
 *document*.getElementById('full-name-preview').textContent = params.fullName || '';  
 *document*.getElementById('address-preview').textContent = params.address || '';  
 *document*.getElementById('ssn-preview').textContent = params.ssn || '';  
 *document*.getElementById('primary-income-preview').textContent = params.primaryIncome || '';  
 *document*.getElementById('additional-income-preview').textContent = params.additionalIncome || '';  
 *document*.getElementById('medical-expenses-preview').textContent = params.medicalExpenses || '';  
 *document*.getElementById('educational-costs-preview').textContent = params.educationalCosts || '';  
 *document*.getElementById('other-expenses-preview').textContent = params.otherExpenses || '';  
  
 // Calculate totals for income and expenses  
 const primaryIncome = parseFloat(params.primaryIncome) || 0;  
 const additionalIncome = parseFloat(params.additionalIncome) || 0;  
 const medicalExpenses = parseFloat(params.medicalExpenses) || 0;  
 const educationalCosts = parseFloat(params.educationalCosts) || 0;  
 const otherExpenses = parseFloat(params.otherExpenses) || 0;  
  
 const totalIncome = primaryIncome + additionalIncome;  
 const totalExpenses = medicalExpenses + educationalCosts + otherExpenses;  
 const taxAmount = totalIncome - totalExpenses; // Difference between income and expenses  
  
 // Display totals  
 *document*.getElementById('total-income').textContent = totalIncome.toFixed(2);  
 *document*.getElementById('total-expenses').textContent = totalExpenses.toFixed(2);  
 *document*.getElementById('tax-amount').textContent = taxAmount.toFixed(2);  
  
 // Set current date and time on the form  
 *document*.getElementById('current-date-time').textContent = getCurrentDateTime();  
  
 // Print function  
 *document*.getElementById('print-btn').onclick = function() {  
 *window*.print();  
 };  
  
 // Return to form button functionality  
 *document*.getElementById('return-btn').onclick = function() {  
 *window*.history.back(); // Go back to the previous page (the declaration form)  
 };  
 });  
</script>

Description HTML (print-preview.html):

1. Header: Displays the title of the document.
2. Current Date and Time: Shows the current date and time in the specified format at the top of the page.
3. Data Table: Structured table presents the user's tax declaration information, including financial data.
4. Calculated Totals: Displays total income, total expenses, and tax amount based on user input.
5. Print and Return Buttons: Provides options to print the document or return to the original tax declaration form for edits.





1. **CSS external files maps to HTML using classes.**

/\* Reset some default styles \*/  
body, h1, h2, h3, p {  
 margin: 0;  
 padding: 0;  
 font-family: Arial, sans-serif;  
}  
  
/\* Body styling \*/  
body {  
 background-color: #f4f4f4;  
 color: #333;  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 padding: 20px;  
}  
  
/\* Header styling \*/  
.main-title {  
 font-size: 2.5em;  
 color: #00457c; /\* Dark blue color \*/  
}  
  
.sub-title {  
 font-size: 1.2em; /\* Smaller font size \*/  
 color: #6a6a6a; /\* Gray color \*/  
 text-align: center; /\* Center the text \*/  
 margin-top: 5px; /\* Space above the subtitle \*/  
 margin-bottom: 20px; /\* Space below the subtitle \*/  
}  
  
/\* Form styling \*/  
#tax-form {  
 background: #fff;  
 padding: 20px;  
 border-radius: 8px;  
 box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
 width: 100%;  
 max-width: 600px; /\* Limiting max width for better readability \*/  
}  
  
/\* Icons for sections \*/  
h3 i {  
 margin-right: 8px; /\* Add space between the icon and the text \*/  
 color: #00457c; /\* Change icon color to match your theme \*/  
}  
  
/\* Section titles \*/  
h3 {  
 font-size: 1.5em;  
 border-bottom: 2px solid #00457c;  
 padding-bottom: 10px;  
 margin-bottom: 20px;  
}  
  
/\* Labels and input fields \*/  
label {  
 display: block; /\* Makes the label take up the full width \*/  
 margin-bottom: 5px;  
 font-weight: bold;  
}  
  
/\* Shorter input fields \*/  
.personal-input {  
 width: calc(80% - 20px); /\* Shorter width for personal info \*/  
}  
  
.money-input {  
 width: calc(40% - 20px); /\* Much smaller width for money inputs \*/  
}  
  
input[type="text"],  
input[type="number"] {  
 padding: 10px;  
 margin-bottom: 15px;  
 border: 1px solid #ddd;  
 border-radius: 5px;  
 font-size: 1em;  
}  
  
/\* Focus state for input fields \*/  
input:focus {  
 border-color: #00457c;  
 outline: none;  
}  
  
/\* Button styling \*/  
.buttons {  
 display: flex;  
 justify-content: space-between;  
}  
  
button {  
 background-color: #00457c;  
 color: #fff;  
 border: none;  
 padding: 10px 15px;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 1em;  
 transition: background-color 0.3s;  
}  
  
button:hover {  
 background-color: #003366;  
}  
  
/\* Validation messages \*/  
.validation-message {  
 color: red;  
 font-size: 0.9em;  
 margin-top: -10px;  
 margin-bottom: 10px;  
}  
  
/\* Footer styling \*/  
footer {  
 margin-top: 20px;  
 text-align: center;  
}  
  
footer a {  
 color: #00457c;  
 text-decoration: none;  
}  
  
footer a:hover {  
 text-decoration: underline;  
}

**Description CSS:**

* **Reset Styles**: Basic resets for margins and paddings to ensure consistency across browsers;
* **Body Styling**: Background color, font family, and center alignment for the entire body;
* **Header Styling**: Specific colors and sizes for the main and subtitle;
* **Form Styling**: Adding padding, a background, border radius, and shadow for the form to enhance appearance;
* **Section Titles**: Styled headers for sections with borders to separate them visually;
* **Label and Input Fields**: Block display for labels, styling input fields for padding, borders, and focus effects;
* **Button Styling**: Styled buttons with hover effects for better interactivity;
* **Validation Messages**: Specific color and styles for validation messages, making them stand out in red;
* **Footer Styling**: Simple styling for footer text and links;

**Relations HTML and CSS:**

Provided CSS code in a file named **styles.css** within the same directory as HTML file;

HTML file correctly links to the styles.css using the <link> tag in the <head> section;

1. **JS events (content of file and description provided).**

*document*.addEventListener('DOMContentLoaded', () => {  
 const form = *document*.getElementById('tax-form');  
 const ssnInput = *document*.getElementById('ssn');  
 const ssnError = *document*.getElementById('ssn-error');  
  
 form.addEventListener('submit', function(event) {  
 ssnError.textContent = '';  
  
 if (!validateSSN(ssnInput.value)) {  
 event.preventDefault(); // Prevent form submission  
 ssnError.textContent = 'Please enter a valid Social Security Number (format: XXX-XX-XXXX).';  
 } else {  
 event.preventDefault(); // Prevent the default form submission  
 showPrintPreview(); // Call the function to show print preview  
 }  
 });  
  
 function validateSSN(ssn) {  
 const ssnPattern = /^\d{3}-\d{2}-\d{4}$/; // Pattern: XXX-XX-XXXX  
 return ssnPattern.test(ssn);  
 }  
  
 function showPrintPreview() {  
 const formData = {  
 fullName: *document*.getElementById('full-name').value,  
 address: *document*.getElementById('address').value,  
 ssn: *document*.getElementById('ssn').value,  
 primaryIncome: *document*.getElementById('primary-income').value,  
 additionalIncome: *document*.getElementById('additional-income').value,  
 medicalExpenses: *document*.getElementById('medical-expenses').value,  
 educationalCosts: *document*.getElementById('educational-costs').value,  
 otherExpenses: *document*.getElementById('other-expenses').value  
 };  
  
 // Redirect to print-preview.html, passing form data via query parameters  
 const queryString = new URLSearchParams(formData).toString();  
 *window*.location.href = `print-preview.html?${queryString}`;  
 }  
  
 // Function to format date and time  
 function getCurrentDateTime() {  
 const now = new *Date*();  
 const day = String(now.getDate()).padStart(2, '0'); // Two-digit day  
 const month = String(now.getMonth() + 1).padStart(2, '0'); // Two-digit month (0-indexed)  
 const year = now.getFullYear();  
 const hours = String(now.getHours()).padStart(2, '0'); // Two-digit hours  
 const minutes = String(now.getMinutes()).padStart(2, '0'); // Two-digit minutes  
 const seconds = String(now.getSeconds()).padStart(2, '0'); // Two-digit seconds  
 return `Today : ${day}-${month}-${year} ${hours}:${minutes}:${seconds}`; // Return formatted date string  
 }  
  
 // Set current date and time  
 *document*.getElementById('current-date-time').textContent = getCurrentDateTime();  
});

Description code of script.js :

1. Event Listener for DOMContentLoaded: Ensures the script runs once the DOM is fully loaded.
2. Form Submission Handling: Validates the SSN and either displays an error message or redirects to the print preview.
3. Validation Function: Checks if the SSN format is correct.
4. Print Preview Redirection: Collects form data and redirects to the print-preview.html page with the data as query parameters.
5. Date and Time Formatting Function: Returns a formatted date string in the specified format.
6. Display Current Date and Time: Sets the current date and time on the main form page.