ISCG6420 Semester 1 2024

Exercise: Creating a Multi-page Form

This exercise will take you through creating a user input form with multiple sub-forms. The learning outcomes will prepare you for creating the deliverables of Assignment 1 Part 3.

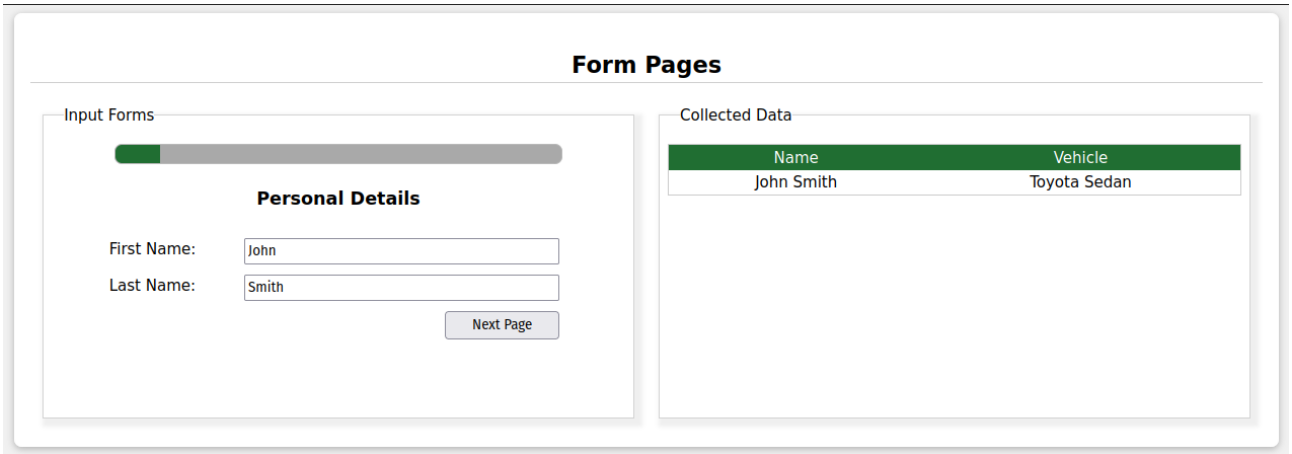
Company A has a fleet of vehicles for use by its employees. The company wants a form system to collect employee name and vehicle preferences to better assign appropriate vehicles each month.

You have been tasked with creating a frontend prototype for the form system. The prototype scope has been limited to three form pages (including summary), and collecting and displaying form data locally. API communication and backend infrastructure will be prototyped separately.

Form structure

The first form page will collect the user's first and last name.
The second form page will collect vehicle preference data.
The third page will display a summary of the collected data.

Page mockup



Create a new webpage

Create a new website project or open an existing project in your code editor. In the root directory create a new “userform.html” file and open it for editing.
Insert a website base template (VSCode shortcut = “html:5”).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

In the body, create a section with an <h2> section title, two card elements, and a <script> linking an external JS file:

```
<body>
    <section>
        <h2 class="underline">Form Pages</h2>
        <div class="container-half card">…
        </div>
        <div class="container-half card">…
        </div>
        <script src="formPaging.js"></script>
    </section>
</body>
```

Form pages

The first card div will contain the form pages. Inside the first card div, create a subtitle, a progress (value: 10, max: 100), and 3 elements.
Give each div an ID with sequential numbers to identify each as a separate form page. Apply and inline style to the divs with “display: block” for the first, and “display: none” for the rest. This will show the first page and hide the others by default.

```
        <div class="container-half card">
            <subtitle class="card-title">Input Forms</subtitle>
            <progress value="10" max="100"></progress>
            <div id="form-page1" class="form-page" style="display: block">…
            </div>
            <div id="form-page2" class="form-page" style="display: none">…
            </div>
            <div id="form-page3" class="form-page" style="display: none;">…
            </div>
        </div>
```

Form structure

In each form page, create 3 elements. These will contain the page title, page input fields, and page navigation buttons:

```
<div id="form-page1" class="form-page" style="display: block">
    <div> <!-- page title -->

    </div>
    <div> <!-- page input fields -->

    </div>
    <div> <!-- page nav buttons -->

    </div>
</div>
```

Populate page 1

In form page 1, add an <h3> to the page title .

```
<div> <!-- page title -->
    <h3>Personal Details</h3>
</div>
```

In the input fields , create two input fields with labels, one for First Name, and one for Last Name. Wrap each input field in the label for styling simplicity:

```
<div> <!-- page input fields -->
    <label for="firstName" class="row">
        First Name:
        <input type="text" id="name-first" />
    </label>
    <label for="lastName" class="row">
        Last Name:
        <input type="text" id="name-last" />
    </label>
</div>
```

In the nav buttons , create a button with type: button and a click event to run function “showPage(2)”.

```
<div> <!-- page nav buttons -->
    <button class="next" type="button" onclick="showPage(2)">
        Next Page
    </button>
</div>
```

Note: the number parameter (2) indicates a form page number we want to display on click.

Populate page 2

In form page 2, add an <h3> to the page title .

```
<div> <!-- page title -->
    <h3>Vehicle Details</h3>
</div>
```

The inputs are a bit more complicated for page 2. Create two <divs> to separate the radio button input from the select input:

```
<div> <!-- page input fields -->
    <div> <!-- radio button input -->
        <label for="vehicle-sedan">
            <input
                type="radio"
                id="vehicle-sedan"
                name="vehicle"
                value="Sedan"
                checked="true"
            />
            Van
        </label>
        <label for="vehicle-SUV">
            <input
                type="radio"
                id="vehicle-suv"
                name="vehicle"
                value="SUV"
            />
            SUV
        </label>
    </div>
    <div> <!-- select box input -->
        <label for="vehicle-make">
            Make:
            <select>
                <option value="Toyota">Toyota</option>
                <option value="Mazda">Mazda</option>
                <option value="Honda">Honda</option>
            </select>
        </label>
    </div>
</div>
```

In the nav buttons , create two buttons. One for Next, and one for Previous:

```
<div> <!-- page nav buttons -->
    <button class="next" type="button" onclick="showPage(3)">
        Next Page
    </button>
    <button class="previous" type="button" onclick="showPage(1)">
        Previous Page
    </button>
</div>
```

Populate page 3

In form page 3, add an <h3> to the page title .

```
<div> <!-- page title -->
    <h3>Summary</h3>
</div>
```

The input fields are replaced by data display fields for the summary page. Create two elements with child elements. The span innerHTML will be updated in JS.

```
<div> <!-- page data fields -->
    <div class="row">
        Name: <span id="summary-name"></span>
    </div>
    <div class="row">
        Vehicle: <span id="summary-vehicle"></span>
    </div>
</div>
```

In the nav buttons , create two buttons. One for previous, and one for submitting.

```
<div> <!-- page nav buttons -->
    <button class="previous" type="button" onclick="showPage(2)">
        Previous Page
    </button>
    <button class="next" type="button" onclick="submitData()">
        Submit
    </button>
</div>
```

Input form card complete.

Data display card

The second card will contain a table displaying the data collected by the forms in card 1.

Add a <subtitle> and <table> to the second card . In the table, create a <thead> heading row with two <td> elements: one for “Name”, one for “Vehicle”:

```
<div class="container-half card">
    <subtitle class="card-title">Collected Data</subtitle>
    <table id="collected-data">
        <thead>
            <td>Name</td>
            <td>Vehicle</td>
        </thead>
    </table>
</div>
```

HTML complete.

Add stylesheet

Add the provided stylesheet “form-pages.css” to your web page.

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Multi form</title>
    <link rel="stylesheet" href="./form-pages.css" />
</head>
```

CSS complete.

JavaScript

We need the following functionality for the form system:

1. Ability to navigate between form pages on button click
2. Display form progress
3. Collect and display form input in the summary page
4. Create and add an instance of form data to the data collection

This can be achieved with some variables and four functions.

Create 7 block-level element reference variables:

- 3 for refererring to each form page
- 1 for referring to the progress bar
- 2 for referring to the summary text fields
- 1 for referring to the data table

```
const formPage1 = document.getElementById("form-page1");
const formPage2 = document.getElementById("form-page2");
const formPage3 = document.getElementById("form-page3");

const progressBar = document.querySelector("progress");

const summaryName = document.getElementById("summary-name");
const summaryVehicle = document.getElementById("summary-vehicle");

const formData = document.getElementById("collected-data");
```

Page navigation and progress display

Create a function named “showPage” and provide an input parameter “pageNumber”. Inside the function create a switch-case block with pageNumber as the input, and a case for the values: 1, 2, 3.

```
function showPage(pageNumber) {
    switch (pageNumber) {
        case 1:

            break;

        case 2:

            break;

        case 3:

            break;

        default:
            break;
    }
}
```

In each case:

- set the “formPageX.style.display = block or none” such that only the page being navigated to will be displayed.
- Set the “progressBar.value” to a percentage that represents the current form completion progress.

```
        case 1:
            formPage1.style.display = 'block';
            formPage2.style.display = 'none';
            formPage3.style.display = 'none';
            progressBar.value=10;
            break;

        case 2:
            formPage1.style.display = 'none';
            formPage2.style.display = 'block';
            formPage3.style.display = 'none';
            progressBar.value=50;
            break;

        case 3:
            formPage1.style.display = 'none';
            formPage2.style.display = 'none';
            formPage3.style.display = 'block';
            progressBar.value=100;
            break;
```

Collate form data

Create a new function and name it “getFormData”.

```
function getFormData() {

}
```

Inside the function create a variable for each form input. Fetch the input field value and save it to the variables:

```
function getFormData() {
    // example using ID-based fetch
    const fName = document.getElementById("name-first").value;
    const lName = document.getElementById("name-last").value;

    // example using CSS selectors within formPage2
    const vType = formPage2.querySelector('input[name=vehicle]:checked').value;
    const vMake = formPage2.querySelector('select').value;
```

Create an object and populate it with the form data. Return the object from the function:

```
    return data = {
        name: fName + " " + lName,
        vehicle: vMake + " " + vType
    };
}
```

This function will be used by any code that uses the form data to simplify the fetching process.

Update Summary fields

Create a new function called “updateSummary”. Inside the function, call the getFormData() function, and set the summaryName and summaryVehicle contents to the object values:

```
function updateSummary() {
    const data = getFormData();

    summaryName.innerHTML = data.name;
    summaryVehicle.innerHTML = data.vehicle;
}
```

Submitting data

Create a new function called “submitData”. Inside the function, create three elements:

- Table row (tr)
- Table data cell (td) -> child of tr
- Table data cell (td) -> child of tr

```
function submitData() {
    const dataRow = document.createElement("tr");
    const cellName = document.createElement("td");
    const cellVehicle = document.createElement("td");

    dataRow.appendChild(cellName);
    dataRow.appendChild(cellVehicle);

}
```

Inside the function, call the getFormData() function, and set cellName and cellVehicle contents to the object values. Add dataRow as a child to formData:

```
    dataRow.appendChild(cellVehicle);

    const data = getFormData();

    cellName.innerHTML = data.name;
    cellVehicle.innerHTML = data.vehicle;

    formData.appendChild(dataRow);
}
```

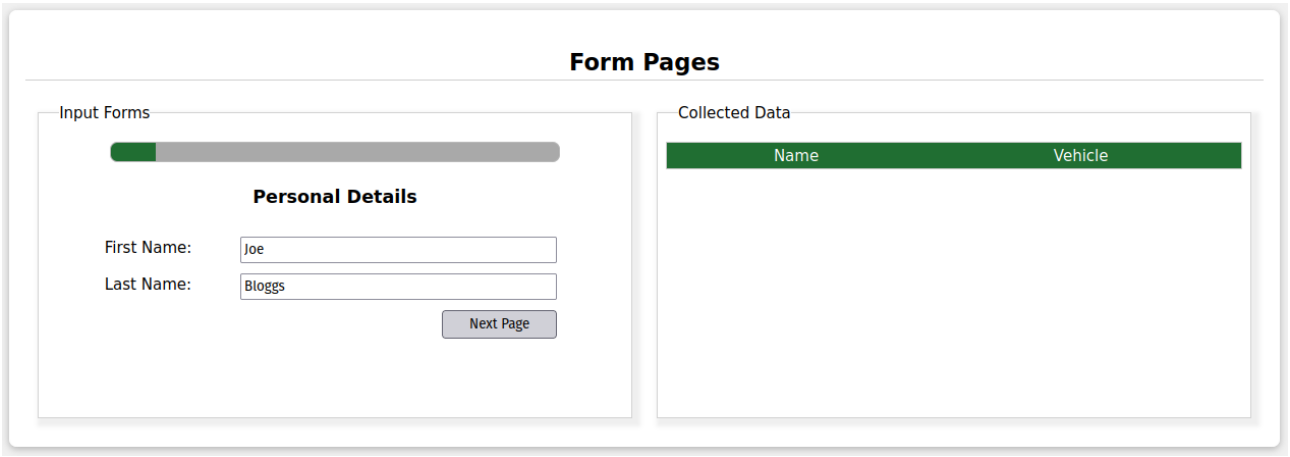
Submitting data

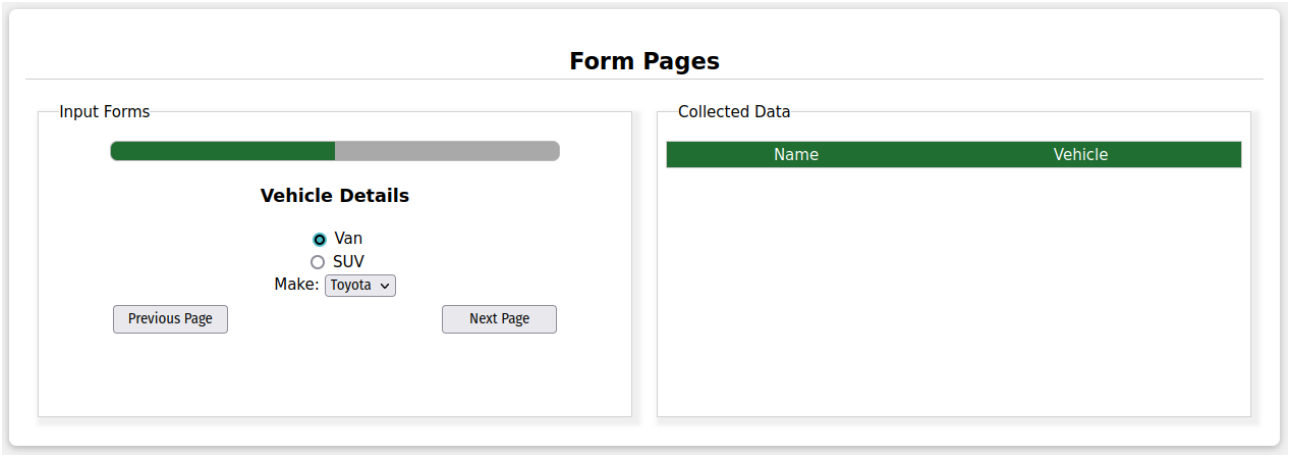
At the end of case 3 in the showPage() function, add a call to the updateSummary() function. This will make sure the summary data fields get populated when the user navigates to the summary page:

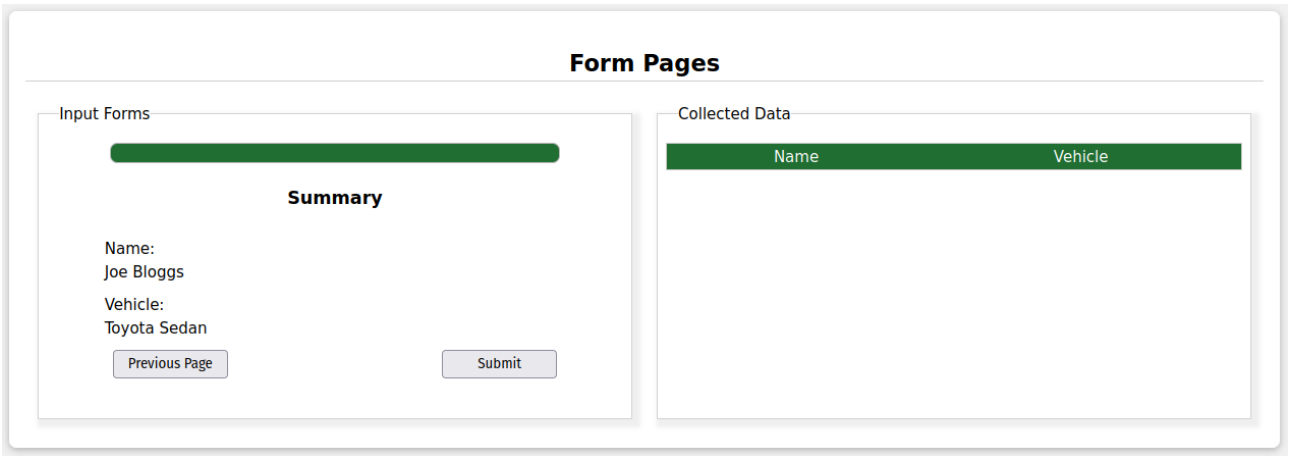
```
        case 3:
            formPage1.style.display = 'none';
            formPage2.style.display = 'none';
            formPage3.style.display = 'block';
            progressBar.value=100;
            updateSummary();
            break;
```

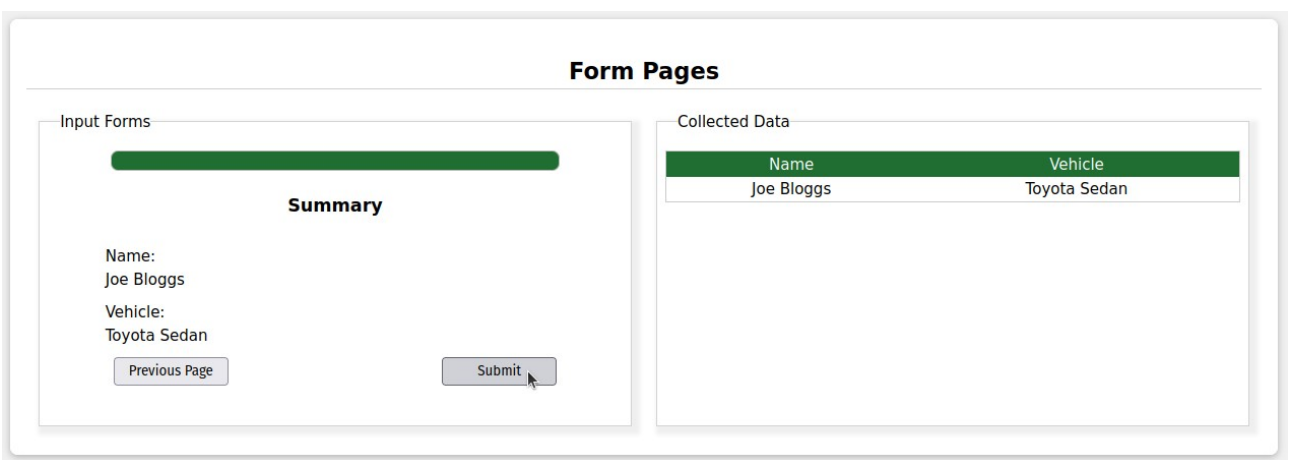
Test

Test your multi-page form system by populating the input fields with dummy data and submitting. Your web page should match the following behaviour:









Challenge

Form pages 1 and 2 have not been styled in accordance with the design applied to page 1 and the rest of the web page.

- Use CSS to set the height of each form page to match.
- Align the elements of page 2 and page 3 to match page 1.
- Change the style and position of the progress bar to blend in to the form design.

Wrapping up

Save your work, commit and push your changes to your git repository.

Exercise complete.