**Arrays**

In JavaScript, an array is a data structure that can hold multiple values at once.

Arrays are used to store a collection of data in a single variable.

Each value in an array is called an element, and each element has a numeric index starting from 0.

**Creating Arrays**

You can create arrays in JavaScript using different methods:

1. **Using square brackets:**

let fruits = ["Apple", "Banana", "Cherry"];

1. **Using the Array constructor:**

let fruits = new Array("Apple", "Banana", "Cherry");

**Accessing Elements**

You can access elements in an array using their index:

let fruits = ["Apple", "Banana", "Cherry"];

console.log(fruits[0]); // Output: Apple

console.log(fruits[1]); // Output: Banana

console.log(fruits[2]); // Output: Cherry

**Modifying Elements**

You can modify elements in an array by assigning new values to specific indices:

let fruits = ["Apple", "Banana", "Cherry"];

fruits[1] = "Blueberry";

console.log(fruits); // Output: ["Apple", "Blueberry", "Cherry"]

**Adding and Removing Elements**

JavaScript arrays come with many built-in methods for adding and removing elements.

**Adding elements:**

1. **Using push:** Adds one or more elements to the end of an array.

let fruits = ["Apple", "Banana"];

fruits.push("Cherry");

console.log(fruits); // Output: ["Apple", "Banana", "Cherry"]

1. **Using unshift:** Adds one or more elements to the beginning of an array.

let fruits = ["Apple", "Banana"];

fruits.unshift("Cherry");

console.log(fruits); // Output: ["Cherry", "Apple", "Banana"]

**Removing elements:**

1. **Using pop:** Removes the last element from an array.

let fruits = ["Apple", "Banana", "Cherry"];

fruits.pop();

console.log(fruits); // Output: ["Apple", "Banana"]

1. **Using shift:** Removes the first element from an array.

let fruits = ["Apple", "Banana", "Cherry"];

fruits.shift();

console.log(fruits); // Output: ["Banana", "Cherry"]

**Array Properties and Methods**

* **length:** Returns the number of elements in an array.

let fruits = ["Apple", "Banana", "Cherry"];

console.log(fruits.length); // Output: 3

* **concat:** Merges two or more arrays.

let fruits = ["Apple", "Banana"];

let moreFruits = ["Cherry", "Date"];

let allFruits = fruits.concat(moreFruits);

console.log(allFruits); // Output: ["Apple", "Banana", "Cherry", "Date"]

* **slice:** Returns a shallow copy of a portion of an array into a new array.

let fruits = ["Apple", "Banana", "Cherry", "Date"];

let someFruits = fruits.slice(1, 3);

console.log(someFruits); // Output: ["Banana", "Cherry"]

* **splice:** Adds/removes items to/from an array.

let fruits = ["Apple", "Banana", "Cherry"];

fruits.splice(1, 1, "Blueberry", "Cantaloupe");

console.log(fruits); // Output: ["Apple", "Blueberry", "Cantaloupe", "Cherry"]

* **forEach:** Executes a provided function once for each array element.

let fruits = ["Apple", "Banana", "Cherry"];

fruits.forEach(function(fruit) {

console.log(fruit);

});

// Output:

// Apple

// Banana

// Cherry

**Multidimensional Arrays**

You can create arrays of arrays, which are known as multidimensional arrays:

let matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

console.log(matrix[1][2]); // Output: 6

**Summary**

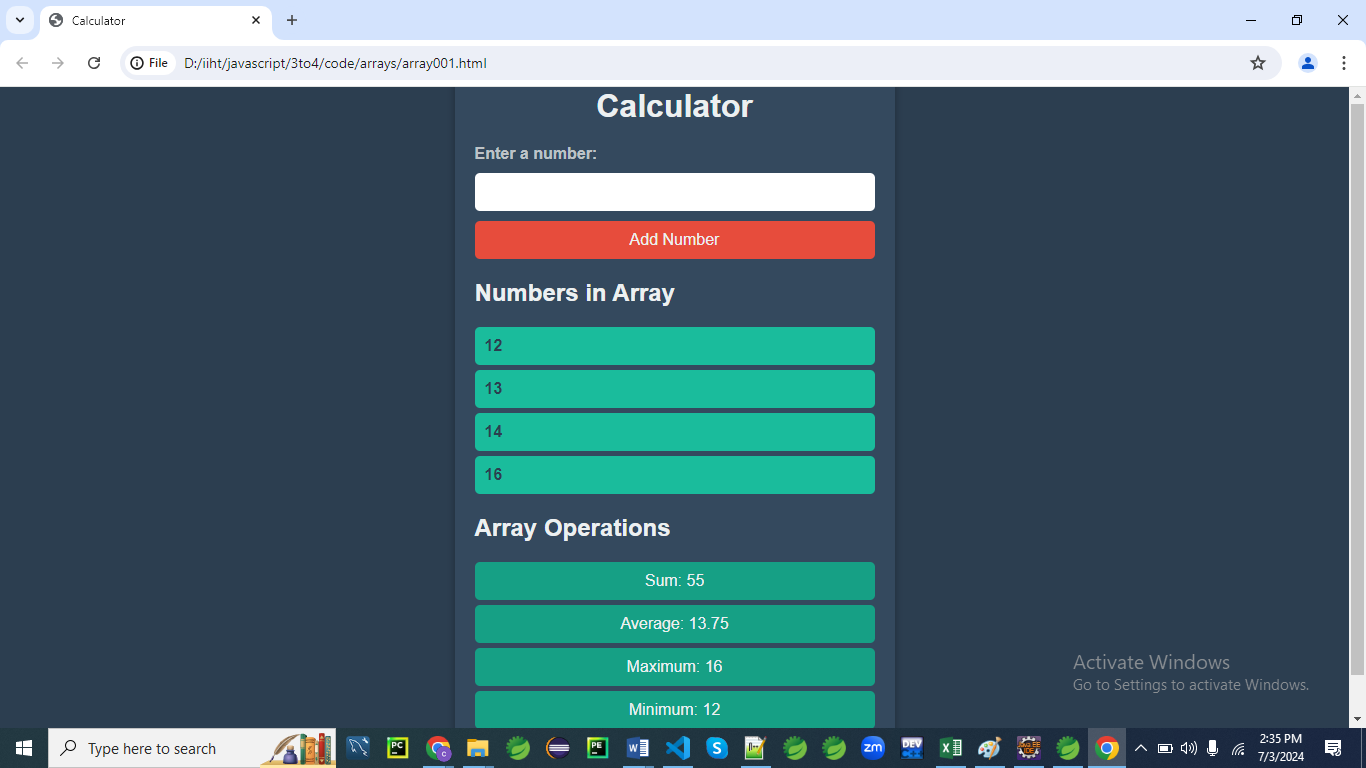
Arrays in JavaScript are versatile and powerful data structures used to store multiple values in a single variable.

They provide various methods and properties to manipulate and traverse the data efficiently.

**Example 001**

Let's create a simple application where users can input numbers into an array, and the program will display various operations performed on the array, such as calculating the sum, average, maximum, and minimum values.

This example will include HTML, CSS, and JavaScript.



**HTML Structure**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Calculator</title>

<link rel="stylesheet" href="styles001.css">

</head>

<body>

<div class="container">

<h1>Mathematic Operations</h1>

<!-- Form to add numbers to the array -->

<div class="form-section">

<form id="addNumberForm">

<label for="numberInput">Enter a number:</label>

<input type="number" id="numberInput" required>

<button type="submit">Add Number</button>

</form>

</div>

<div class="form-section">

<h2>Numbers</h2>

<ul id="numberList"></ul>

</div>

<div class="form-section">

<h2>Array Operations</h2>

<p id="sum">Sum: 0</p>

<p id="average">Average: 0</p>

<p id="maximum">Maximum: 0</p>

<p id="minimum">Minimum: 0</p>

</div>

</div>

<script src="script001.js"></script>

</body>

</html>

**Explanation:**

* **DOCTYPE and <html> Tag:** The document type declaration (<!DOCTYPE html>) tells the browser that this is an HTML5 document. The <html> tag encloses the entire HTML document and specifies the language as English (lang="en").
* **Head Section (<head>):**
  + <meta charset="UTF-8"> specifies the character encoding for the document.
  + <meta name="viewport" content="width=device-width, initial-scale=1.0"> ensures the page is responsive and adjusts correctly on different devices.
  + <title>Calculator</title> sets the title of the web page that appears in the browser tab.
  + <link rel="stylesheet" href="styles001.css"> links the external CSS file for styling.
* **Body Section (<body>):**
  + <div class="container"> serves as the main wrapper for the content.
  + <h1>Calculator</h1> is a heading for the page.
  + **Form Section (<div class="form-section">):** Contains a form with an ID of addNumberForm for adding numbers to the array.
    - <label for="numberInput">Enter a number:</label> provides a label for the input field.
    - <input type="number" id="numberInput" required> is a numeric input field with an ID of numberInput that is required.
    - <button type="submit">Add Number</button> is a button to submit the form.
  + **Numbers Section (<div class="form-section">):** Displays the list of numbers in an unordered list with an ID of numberList.
  + **Operations Section (<div class="form-section">):** Displays the results of mathematical operations:
    - <p id="sum">Sum: 0</p> shows the sum of the numbers.
    - <p id="average">Average: 0</p> shows the average of the numbers.
    - <p id="maximum">Maximum: 0</p> shows the maximum number.
    - <p id="minimum">Minimum: 0</p> shows the minimum number.
* **JavaScript File (<script src="script001.js"></script>):** Links the external JavaScript file that contains the logic for handling the form submission and performing the calculations.

**CSS Styles**

/\* styles001.css \*/

body {

    font-family: 'Arial', sans-serif;

    background: #2c3e50;

    color: #ecf0f1;

    margin: 0;

    padding: 0;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    background: #34495e;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

    max-width: 400px;

    width: 100%;

}

h1 {

    text-align: center;

    color: #ecf0f1;

    margin-bottom: 20px;

}

.form-section {

    margin-bottom: 20px;

}

form {

    display: flex;

    flex-direction: column;

    gap: 10px;

}

label {

    font-weight: bold;

    color: #bdc3c7;

}

input[type="number"] {

    padding: 10px;

    border: none;

    border-radius: 5px;

    outline: none;

    font-size: 16px;

}

button {

    padding: 10px;

    border: none;

    border-radius: 5px;

    background: #e74c3c;

    color: #ecf0f1;

    cursor: pointer;

    font-size: 16px;

    transition: background 0.3s;

}

button:hover {

    background: #c0392b;

}

ul {

    list-style-type: none;

    padding: 0;

}

li {

    padding: 10px;

    margin: 5px 0;

    background: #1abc9c;

    border-radius: 5px;

    color: #2c3e50;

    font-weight: bold;

    display: flex;

    justify-content: space-between;

}

.form-section p {

    margin: 5px 0;

    padding: 10px;

    background: #16a085;

    border-radius: 5px;

    text-align: center;

}

**Explanation:**

**Body Element Styles**

**body {**

**font-family: 'Arial', sans-serif;**

**background: #2c3e50;**

**color: #ecf0f1;**

**margin: 0;**

**padding: 0;**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 100vh;**

**}**

**Explanation:**

* font-family: 'Arial', sans-serif;: Sets the font family of the text to Arial, with a fallback to sans-serif.
* background: #2c3e50;: Sets the background color of the body to a dark blue-gray color.
* color: #ecf0f1;: Sets the text color to a light gray.
* margin: 0;: Removes the default margin around the body.
* padding: 0;: Removes the default padding inside the body.
* display: flex;: Uses flexbox layout for the body, making it a flex container.
* justify-content: center;: Horizontally centers the flex items (in this case, the container).
* align-items: center;: Vertically centers the flex items.
* height: 100vh;: Sets the height of the body to 100% of the viewport height, making it full-screen height.

**Container Element Styles**

**.container {**

**background: #34495e;**

**padding: 20px;**

**border-radius: 10px;**

**box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);**

**max-width: 400px;**

**width: 100%;**

**}**

**Explanation:**

* background: #34495e;: Sets the background color of the container to a slightly darker shade.
* padding: 20px;: Adds 20 pixels of padding inside the container.
* border-radius: 10px;: Rounds the corners of the container with a 10-pixel radius.
* box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);: Adds a shadow below the container to give a 3D effect.
* max-width: 400px;: Sets the maximum width of the container to 400 pixels.
* width: 100%;: Ensures the container takes up 100% of the width of its parent (in this case, the body).

**Heading Styles**

**h1 {**

**text-align: center;**

**color: #ecf0f1;**

**margin-bottom: 20px;**

**}**

**Explanation:**

* text-align: center;: Centers the text horizontally.
* color: #ecf0f1;: Sets the color of the heading text to a light gray.
* margin-bottom: 20px;: Adds 20 pixels of margin below the heading to create space between it and the next element.

**Form Section Styles**

**.form-section {**

**margin-bottom: 20px;**

**}**

**Explanation:**

* margin-bottom: 20px;: Adds 20 pixels of margin below each form section to create space between sections.

**Form Styles**

**form {**

**display: flex;**

**flex-direction: column;**

**gap: 10px;**

**}**

**Explanation:**

* display: flex;: Uses flexbox layout for the form, making it a flex container.
* flex-direction: column;: Stacks the flex items (form elements) vertically.
* gap: 10px;: Adds 10 pixels of space between the flex items.

**Label Styles**

**label {**

**font-weight: bold;**

**color: #bdc3c7;**

**}**

**Explanation:**

* font-weight: bold;: Makes the text bold.
* color: #bdc3c7;: Sets the color of the label text to a light gray.

**Input Field Styles**

**input[type="number"] {**

**padding: 10px;**

**border: none;**

**border-radius: 5px;**

**outline: none;**

**font-size: 16px;**

**}**

**Explanation:**

* padding: 10px;: Adds 10 pixels of padding inside the input field.
* border: none;: Removes the default border of the input field.
* border-radius: 5px;: Rounds the corners of the input field with a 5-pixel radius.
* outline: none;: Removes the outline that typically appears when the input field is focused.
* font-size: 16px;: Sets the font size of the text inside the input field to 16 pixels.

**Button Styles**

**button {**

**padding: 10px;**

**border: none;**

**border-radius: 5px;**

**background: #e74c3c;**

**color: #ecf0f1;**

**cursor: pointer;**

**font-size: 16px;**

**transition: background 0.3s;**

**}**

**Explanation:**

* padding: 10px;: Adds 10 pixels of padding inside the button.
* border: none;: Removes the default border of the button.
* border-radius: 5px;: Rounds the corners of the button with a 5-pixel radius.
* background: #e74c3c;: Sets the background color of the button to a red shade.
* color: #ecf0f1;: Sets the text color of the button to a light gray.
* cursor: pointer;: Changes the cursor to a pointer when hovering over the button.
* font-size: 16px;: Sets the font size of the button text to 16 pixels.
* transition: background 0.3s;: Adds a transition effect to the background color change.

**Button Hover Styles**

**button:hover {**

**background: #c0392b;**

**}**

**Explanation:**

* background: #c0392b;: Changes the background color of the button to a darker red shade when hovered over.

**List Styles**

**ul {**

**list-style-type: none;**

**padding: 0;**

**}**

**Explanation:**

* list-style-type: none;: Removes the default bullet points from the list items.
* padding: 0;: Removes the default padding of the unordered list.

**List Item Styles**

**li {**

**padding: 10px;**

**margin: 5px 0;**

**background: #1abc9c;**

**border-radius: 5px;**

**color: #2c3e50;**

**font-weight: bold;**

**display: flex;**

**justify-content: space-between;**

**}**

**Explanation:**

* padding: 10px;: Adds 10 pixels of padding inside each list item.
* margin: 5px 0;: Adds 5 pixels of margin above and below each list item.
* background: #1abc9c;: Sets the background color of each list item to a teal shade.
* border-radius: 5px;: Rounds the corners of each list item with a 5-pixel radius.
* color: #2c3e50;: Sets the text color of each list item to a dark blue-gray.
* font-weight: bold;: Makes the text bold.
* display: flex;: Uses flexbox layout for the list items.
* justify-content: space-between;: Distributes space between the child elements of each list item, pushing them to opposite ends.

**Form Section Paragraph Styles**

**.form-section p {**

**margin: 5px 0;**

**padding: 10px;**

**background: #16a085;**

**border-radius: 5px;**

**text-align: center;**

**}**

**Explanation:**

* margin: 5px 0;: Adds 5 pixels of margin above and below each paragraph.
* padding: 10px;: Adds 10 pixels of padding inside each paragraph.
* background: #16a085;: Sets the background color of each paragraph to a greenish shade.
* border-radius: 5px;: Rounds the corners of each paragraph with a 5-pixel radius.
* text-align: center;: Centers the text inside each paragraph.

These styles combine to create a visually appealing and modern-looking calculator application, with clearly defined sections, interactive form elements, and responsive design principles.

**JavaScript Code**

//script001.js

document.addEventListener('DOMContentLoaded', function() {

const addNumberForm = document.getElementById('addNumberForm');

const numberList = document.getElementById('numberList');

const sumElement = document.getElementById('sum');

const averageElement = document.getElementById('average');

const maximumElement = document.getElementById('maximum');

const minimumElement = document.getElementById('minimum');

let numbers = [];

addNumberForm.addEventListener('submit', function(event) {

event.preventDefault(); // Prevent the form from submitting the traditional way

const numberInput = document.getElementById('numberInput').value;

const number = parseFloat(numberInput);

if (!isNaN(number)) {

// Add new number to the numbers array

numbers.push(number);

// Clear the form field

addNumberForm.reset();

// Render the numbers and perform operations

renderNumbers();

performOperations();

}

});

function renderNumbers() {

numberList.innerHTML = ''; // Clear existing numbers

// Loop through numbers array and create list elements

numbers.forEach((number) => {

const li = document.createElement('li');

li.textContent = number;

numberList.appendChild(li);

});

}

function performOperations() {

if (numbers.length > 0) {

const sum = numbers.reduce((acc, curr) => acc + curr, 0);

const average = sum / numbers.length;

const maximum = Math.max(...numbers);

const minimum = Math.min(...numbers);

sumElement.textContent = `Sum: ${sum}`;

averageElement.textContent = `Average: ${average}`;

maximumElement.textContent = `Maximum: ${maximum}`;

minimumElement.textContent = `Minimum: ${minimum}`;

}

else {

sumElement.textContent = 'Sum: 0';

averageElement.textContent = 'Average: 0';

maximumElement.textContent = 'Maximum: 0';

minimumElement.textContent = 'Minimum: 0';

}

}

});

**Explanation:**

Let's break down the JavaScript code step by step to understand how it works.

**DOMContentLoaded Event Listener**

**document.addEventListener('DOMContentLoaded', function() {**

**// All code inside this function will run once the DOM is fully loaded**

**});**

**Explanation:**

* This line ensures that the JavaScript code runs only after the entire HTML document has been completely loaded and parsed. This prevents issues with accessing DOM elements before they are available.

**Variable Declarations**

**const addNumberForm = document.getElementById('addNumberForm');**

**const numberList = document.getElementById('numberList');**

**const sumElement = document.getElementById('sum');**

**const averageElement = document.getElementById('average');**

**const maximumElement = document.getElementById('maximum');**

**const minimumElement = document.getElementById('minimum');**

**let numbers = [];**

**Explanation:**

* The addNumberForm variable stores the form element used to add numbers to the array.
* The numberList variable stores the <ul> element where the numbers will be displayed.
* The sumElement, averageElement, maximumElement, and minimumElement variables store the <p> elements where the results of the array operations (sum, average, maximum, and minimum) will be displayed.
* The numbers array is initialized as an empty array and will store the numbers added by the user.

**Form Submission Event Listener**

**addNumberForm.addEventListener('submit', function(event) {**

**event.preventDefault(); // Prevent the form from submitting the traditional way**

**const numberInput = document.getElementById('numberInput').value;**

**const number = parseFloat(numberInput);**

**if (!isNaN(number)) {**

**// Add new number to the numbers array**

**numbers.push(number);**

**// Clear the form field**

**addNumberForm.reset();**

**// Render the numbers and perform operations**

**renderNumbers();**

**performOperations();**

**}**

**});**

**Explanation:**

* Adds an event listener for the form's submit event.
* event.preventDefault(); prevents the form from submitting in the traditional way, which would reload the page.
* Retrieves the value from the number input field, converts it to a floating-point number using parseFloat.
* If the input is a valid number (!isNaN(number)), it is added to the numbers array.
* The form is reset to clear the input field.
* Calls renderNumbers() to update the displayed list of numbers.
* Calls performOperations() to update the displayed results (sum, average, maximum, minimum).

**Render Numbers Function**

**function renderNumbers() {**

**numberList.innerHTML = ''; // Clear existing numbers**

**// Loop through numbers array and create list elements**

**numbers.forEach((number) => {**

**const li = document.createElement('li');**

**li.textContent = number;**

**numberList.appendChild(li);**

**});**

**}**

**Explanation:**

* Clears the current contents of the number list (numberList.innerHTML = '';).
* Loops through each number in the numbers array.
* For each number, creates a new <li> element, sets its text content to the number, and appends it to the numberList <ul> element.

**Perform Operations Function**

**function performOperations() {**

**if (numbers.length > 0) {**

**const sum = numbers.reduce((acc, curr) => acc + curr, 0);**

**const average = sum / numbers.length;**

**const maximum = Math.max(...numbers);**

**const minimum = Math.min(...numbers);**

**sumElement.textContent = `Sum: ${sum}`;**

**averageElement.textContent = `Average: ${average}`;**

**maximumElement.textContent = `Maximum: ${maximum}`;**

**minimumElement.textContent = `Minimum: ${minimum}`;**

**}**

**else {**

**sumElement.textContent = 'Sum: 0';**

**averageElement.textContent = 'Average: 0';**

**maximumElement.textContent = 'Maximum: 0';**

**minimumElement.textContent = 'Minimum: 0';**

**}**

**}**

**Explanation:**

* Checks if the numbers array has any elements (numbers.length > 0).
* If the array has elements:
  + Calculates the sum of the numbers using reduce method.
  + Calculates the average by dividing the sum by the number of elements in the array.
  + Finds the maximum number using Math.max with the spread operator (...).
  + Finds the minimum number using Math.min with the spread operator.
  + Updates the text content of the respective elements (sumElement, averageElement, maximumElement, minimumElement) to display the calculated values.
* If the array is empty, sets the text content of the respective elements to 0.

**Summary**

The program creates a simple calculator that allows users to add numbers to an array through a form, displays the numbers, and performs basic arithmetic operations (sum, average, maximum, minimum) on the array.

The results are updated dynamically whenever a new number is added.

The DOMContentLoaded event ensures the script runs only after the HTML is fully loaded, preventing errors due to accessing DOM elements too early.

The CSS file styles the elements to create a visually appealing layout.

**Explanation**

1. **HTML Structure:**
   * The HTML includes a form to add numbers and a section to display the list of numbers.
   * Another section displays the results of various operations performed on the array, such as sum, average, maximum, and minimum.
2. **CSS Styles:**
   * The CSS styles create a clean and modern design for the array operations application.
   * It includes styles for the body, container, headings, forms, labels, input fields, buttons, and list items.
3. **JavaScript Code:**
   * **Event Listener for Adding Numbers:**
     + An event listener on the addNumberForm handles form submission, preventing the default form submission behavior.
     + The number is retrieved from the form field and added to the numbers array.
     + The form field is cleared, and the renderNumbers and performOperations functions are called to update the list of numbers and perform the array operations.
   * **Render Numbers Function:**
     + This function loops through the numbers array and creates list elements for each number.
     + It dynamically creates li elements with the number and appends them to the numberList.
   * **Perform Operations Function:**
     + This function calculates the sum, average, maximum, and minimum of the numbers in the array.
     + It updates the corresponding elements with the results of these operations.

This example demonstrates how to use arrays in JavaScript to store and manipulate data, perform various operations on the data, and dynamically update the DOM to display the results.

The use of CSS3 styling enhances the visual appeal and user experience.