Week 3: JavaScript Arrays and Loops

FH University of Applied Sciences TECHNIKUM WIEN

Week 3: JavaScript Arrays and Loops

This week we'll cover:

- **JavaScript Arrays**: Storing and accessing multiple values.
- **Loops**: Efficiently handling repeated tasks.
- **Enhancing the Memory Game**: Applying arrays and loops.

Introduction to Arrays

- **Array**: A data structure that stores multiple values.
- Useful for managing groups of related data, like cards in a memory game.

Array Basics

- Created with square brackets `[]`.
- Access elements by index, starting from `0`.

Example:

```
let cards = ['🍎', '🍌', '🍇', '🍒'];
console.log(cards[0]); // Output: 🍎
```

Adding and Modifying Array Elements

- **Accessing Elements**: `cards[0]` returns the first element.
- **Modifying Elements**: Set a new value at a specific index.

Example:

```
let cards = ['🍎', '🍌', '🍇', '🍒'];
cards[2] = '🍉';
console.log(cards); // Output: ['🍎', '🍌', '🍉', '🍒']
```

Try It: Modify one of the elements in an array and log the result.

Common Array Methods

- Arrays have built-in methods to add, remove, or modify elements.

Useful Methods

- `push()`: Adds an element to the end.
- `pop()`: Removes the last element.
- `shift()`: Removes the first element.
- `unshift()`: Adds an element to the beginning.

Example:

```
let fruits = ['apple', 'banana'];
fruits.push('cherry');
console.log(fruits); // Output: ['apple', 'banana', 'cherry']
```

Looping Through Arrays with `forEach`

- `forEach` **Method**: Runs a function for each element in an array.

Syntax:

```
array.forEach(element => {
    // code to run on each element
});
```

Example:

```
let fruits = ['apple', 'banana', 'cherry'];
fruits.forEach(fruit => {
    console.log(fruit);
});
// Output:
// apple
// banana
// cherry
```

Try It: Loop through an array and log each element.

`for` Loops

- `for` **Loop**: Runs a set number of times, ideal for iterating over arrays.

Syntax:

```
for (let i = 0; i < array.length; i++) {
    console.log(array[i]);
}
```

Example:

```
let colors = ['red', 'green', 'blue'];
for (let i = 0; i < colors.length; i++) {
    console.log(colors[i]);
}
```

Explanation: This loop logs each color in the `colors` array.

`while` Loops

- `while` **Loop**: Runs as long as a condition is true, useful for flexible repetition.

Syntax:

```
while (condition) {
    // code to run while condition is true
}
```

Example:

```
let count = 0;
while (count < 3) {
    console.log(count);
    count++;
}
// Output: 0, 1, 2
```

Enhancing the Memory Game with Arrays and Loops

We'll use **arrays** to manage card data and **loops** to render and shuffle cards.

Goals:

1. Store cards in an array.
2. Use loops to render and shuffle cards.
3. Implement matching logic with arrays.

Creating the Card Array

1. **Define an Array of Card Symbols**:
   - Use an array to store each card's symbol.
2. **Example**:

```
let cards = ['🍎', '🍎', '🍌', '🍌', '🍇', '🍇', '🍒', '🍒'];
```

3. **Shuffle the Array**:
   - Use a `shuffle` function to randomize the array.

```
function shuffle(array) {
    array.sort(() => Math.random() - 0.5);
}
shuffle(cards);
```

Try It: Test the shuffle function to see if the cards are randomized.

Rendering Cards with a Loop

Use a loop to dynamically create and render each card on the game board.

Example Code:

```
const gameBoard = document.getElementById('game-board');
cards.forEach(symbol => {
    const card = document.createElement('div');
    card.classList.add('card');
    card.dataset.symbol = symbol;
    card.addEventListener('click', flipCard);
    gameBoard.appendChild(card);
});
```

Explanation: This code creates a `div` for each card and adds it to the game board.

Flip Logic with Arrays

1. **Track Flipped Cards**:
   - Use an array to store flipped cards temporarily.
2. **Example**:

```
let flippedCards = [];

function flipCard(event) {
    const card = event.target;
    card.classList.add('flipped');
    card.textContent = card.dataset.symbol;
    flippedCards.push(card);
}
```

Try It: Test the flip logic and verify that two cards are tracked in `flippedCards`.

Matching Logic with Arrays

1. **Check for Matches**:
    - Compare the symbols of two flipped cards to see if they match.
2. **Example**:

```
function checkForMatch() {
    const [card1, card2] = flippedCards;
    if (HOW CAN YOU COMPARE CARDS?) {
        flippedCards = []; // Reset the array if there’s a match
    } else {
        setTimeout(() => {
            //Wait 1 second and get cards back into game ...
        }, 1000);
    }
}
```

Explanation: This code resets the flipped cards if they match or flips them back if they don’t.

Game Reset and Shuffle Function

1. **Adding a Reset Button**:
   - Use a button to reshuffle and restart the game.
2. **Example**:

```
const resetButton = document.getElementById('reset-button');
resetButton.addEventListener('click', () => {
    gameBoard.innerHTML = '';
    ...
});
```

3. **Explanation**: This clears the board, reshuffles the cards, and re-renders the game.

Putting It All Together

1. **Define the Card Array**.
2. **Shuffle and Render Cards**.
3. **Flip and Match Cards**.
4. **Add Reset Functionality**.

Exercise for the Week: Enhanced Memory Game

Objective: Use arrays and loops to enhance the Memory Game with:

1. **

Shuffling and Rendering**: Randomize and display cards.
2. **Flip and Match Logic**: Track flipped cards and check for matches.
3. **Reset Functionality**: Restart the game with reshuffled cards.

Extra Challenge: Add visual effects for matched cards.

Weekly Assignment Breakdown

1. **Set Up and Shuffle Cards**: Store and randomize card symbols in an array.
2. **Render with Loops**: Use a loop to display cards on the game board.
3. **Implement Flip and Match Logic**: Manage flipped cards and check for matches.
4. **Add Reset Feature**: Create a button to reshuffle and reset the game.

Tips: Use `console.log()` to track card symbols and matches.

Summary and Q&A

- **JavaScript Arrays**: Storing and managing multiple values.
- **Loops**: Automating repetitive tasks.
- **Project Recap**: Enhancing the Memory Game with arrays and loops.

Q&A: Open session for questions about arrays, loops, or the assignment.