Week 2: JavaScript Functions and Event Handling

FH University of Applied Sciences
TECHNIKUM WIEN

Week 2: JavaScript Functions and Event Handling

In this session, we'll cover:

- **Advanced Functions**: Parameters and return values.
- **Event Handling**: Using click and keyboard events.
- **Hands-On Project**: Building a virtual piano with JavaScript.

Review: JavaScript Functions

- **Functions** are blocks of code that perform a specific task.
- **Reusable**: You can call a function multiple times.
- **Takes Parameters**: Pass data into functions to customize behavior.

Example Function:

```
function greet(name) {
    console.log("Hello, " + name);
}
greet("Alice"); // Output: Hello, Alice
```

Function Parameters and Arguments

- **Parameters** are placeholders for values.
- **Arguments** are the actual values you pass when calling the function.

Example:

```
function playSound(note) {
    console.log("Playing note:", note);
}
playSound("C"); // Output: Playing note: C
playSound("D"); // Output: Playing note: D
```

Exercise: Try creating a function that takes two numbers and logs their sum.

Return Values

- Functions can **return** values using the `return` keyword.
- Return values can be stored in variables or used in other functions.

Example:

```
function add(a, b) {
    return a + b;
}
let result = add(5, 10);
console.log(result); // Output: 15
```

Important: Once a function returns a value, it stops executing further code.

JavaScript Events

- **Events** are actions that occur, like clicks or key presses.
- **Event-Driven Programming**: JavaScript waits for events and reacts to them.

Types of Events:

- **Click Events**: Triggered by mouse clicks.
- **Keyboard Events**: Triggered by key presses, like `keydown` or `keyup`.

Example: We can listen for a button click and trigger a function.

Using `addEventListener`

- `addEventListener` attaches a function to an element's event.
- Syntax:

```
element.addEventListener("event", function);
```

Example:

```
document.getElementById("myButton").addEventListener("click", function() {
    console.log("Button clicked!");
});
```

Keyboard Events

- **Keyboard events** can detect key presses, useful for game controls.
- Common events:
  - `keydown`: When a key is pressed.
  - `keyup`: When a key is released.

Example:

```
document.addEventListener("keydown", function(event) {
    console.log("Key pressed:", event.key);
});
```

This will log the key pressed to the console.

Event Object in JavaScript

- The **event object** is automatically passed to event-handling functions.
- Provides information about the event, like `event.key` or `event.type`.

Example:

```
document.addEventListener("click", function(event) {
    console.log("Mouse clicked at:", event.clientX, event.clientY);
});
```

Try It: Log the key pressed by the user using `event.key`.

Hands-On: Virtual Piano Setup

Goal: Create a simple piano interface using HTML, CSS, and JavaScript.

1. **HTML Layout**: Each piano key is a button with an ID.
2. **CSS Styling**: Styles are provided to create a piano look.
3. **JavaScript**: Attach event listeners to each key.

HTML Example:

```
<div id="piano">
    <button class="key" id="keyC">C</button>
    <button class="key" id="keyD">D</button>
    <button class="key" id="keyE">E</button>
</div>
```

Playing Sounds with Functions

1. **Define a `playSound` Function**:
    - This function takes a note (e.g., "C") and plays a corresponding sound file.
2. **Example `playSound` Function**:

```
function playSound(note) {
    console.log("Playing:", note);
    let audio = new Audio(`sounds/${note}.mp3`);
    audio.play();
}
```

Explanation: `Audio` loads and plays a sound file for the specified note.

Adding Click Event Listeners

- Attach a **click event** listener to each key.
- When clicked, each key will call `playSound` with the correct note.

Example Code:

```
document.getElementById("keyC").addEventListener("click", function() {
    playSound("C");
});
```

Test: Click each key and check the Console output.

Adding Keyboard Events

- Allow users to press keyboard keys to play sounds.
- Map keys (e.g., "a" for "C", "s" for "D") to notes.

Example:

```
document.addEventListener("keydown", function(event) {
    if (event.key === "a") {
        playSound("C");
    } else ...
});
```

Try It: Press "a" or "s" and observe the sound played.

Putting It All Together

1. **Set up the HTML structure** for the piano keys.
2. **Define** `playSound` **function** to play sound files.
3. **Attach click and keydown events** to control the piano with clicks and keyboard.

Exercise for the Week: Virtual Piano Game

Objective: Create a fully functional virtual piano that plays notes with both mouse clicks and keyboard keys.

1. **Set Up HTML and CSS**: Arrange the layout for piano keys.
2. **Implement `playSound` Function**: Load and play audio files for each note.
3. **Add Event Listeners**: Attach click and keydown events to each piano key.

Weekly Assignment Breakdown

1. **HTML Structure**: Set up HTML with unique IDs for each key (1 hour).
2. **JavaScript Functions**:
    - Create `playSound` to play specific sounds (1 hour).
3. **Event Handling**:
    - Attach click events to each key (1 hour).
    - Attach keyboard events to play sounds via keys (2 hours).
4. **Debugging**: Use `console.log()` to check sound triggers (1 hour).

Bonus: Add color effects when keys are clicked or pressed.

Debugging Tips

1. **Use** `console.log()` to check which key triggers the sound.
2. **Test Keyboard Mappings**:
    - Ensure each keyboard key triggers the correct note.
3. **Check the Console for Errors**:
    - Look for any `Audio` file loading errors.

Tip: Test each function in isolation before combining them.

Summary and Q&A

- **JavaScript Functions**: Parameters and return values.
- **Events in JavaScript**: `click` and `keydown` event listeners.
- **Project Recap**: Building a virtual piano with interactive controls.

Q&A: Questions on functions, events, or the weekly assignment?