Week 4: JavaScript Objects and AJAX (fetch)

FH University of Applied Sciences
TECHNIKUM WIEN

Week 4: JavaScript Objects and AJAX (fetch)

This week we'll cover:

- **JavaScript Objects**: Grouping related data and functionality.
- **Looping Through Object Arrays**: Iterating over objects efficiently.
- **AJAX with Fetch API and XMLHttpRequest**: Loading and processing JSON data asynchronously.
- **Enhancing the Piano Game**: Using objects and AJAX to fetch notes dynamically.

Introduction to JavaScript Objects

What are JavaScript Objects?

- **Objects** are collections of related data and functions, stored as properties and methods.
- They allow better organization and encapsulation of data.
- Objects are used frequently in JavaScript for structuring data, managing state, and defining behavior.

Object Basics

- Objects are created using curly braces `{}`.
- Store data as key-value pairs, where each key is a property of the object.

Example:

```
const pianoKey = {
    note: "C",
    key: "a"
};
console.log(pianoKey.note); // Output: C
```

Explanation: This object stores data about a piano key, including its `note` and associated `key` (keyboard key).

Why Use Objects?

- Objects make code more readable and structured.
- They enable easy data manipulation and retrieval.
- Used frequently in APIs and AJAX responses.

Looping Through Object Arrays

When working with multiple objects, we often store them in an array and loop through them.

Using `forEach()`

The `forEach()` method allows us to iterate over an array of objects easily.

```
const keys = [
    { note: "C", key: "a" },
    { note: "D", key: "s" },
    { note: "E", key: "d" }
];

keys.forEach(key => {
    console.log(`Key: ${key.key}, Note: ${key.note}`);
});
```

Using `map()`

The `map()` method is useful when we need to transform object data into a new array.

```
const notes = keys.map(key => key.note);
console.log(notes); // Output: ["C", "D", "E"]
```

Using `for...of`

Another way to loop through object arrays is with `for...of`.

```
for (let key of keys) {
    console.log(`Playing note: ${key.note}`);
}
```

Try It: Create an array of objects and use `forEach()`, `map()`, and `for...of` to iterate over them.

Introduction to AJAX (Asynchronous JavaScript and XML)

What is AJAX?

- AJAX allows web pages to retrieve and send data asynchronously without reloading the page.
- It is commonly used to fetch JSON data from a server or API.
- Improves performance by updating only necessary parts of the webpage.

XMLHttpRequest (XHR) vs Fetch API

Before Fetch API, **XMLHttpRequest (XHR)** was the standard way to make AJAX requests.

XMLHttpRequest Basics

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", true);
xhr.onload = function() {
    if (xhr.status === 200) {
        const data = JSON.parse(xhr.responseText);
        console.log(data);
    }
};
xhr.onerror = function() {
    console.error("Error fetching data");
};
xhr.send();
```

Fetch API Basics

The Fetch API provides a modern and cleaner way to make HTTP requests in JavaScript compared to `XMLHttpRequest`.

```
fetch('data.json')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error('Error:', error));
```

Comparison:

Feature	XMLHttpRequest	Fetch API
Syntax	More complex	Simpler & cleaner
Promises	No	Yes
Error Handling	Manual	Built-in with `.catch()`
Streaming	Limited	Supports streaming

Why Use Fetch API?

- **Asynchronous**: Non-blocking requests improve responsiveness.
- **Promise-based**: More readable and manageable than callbacks.
- **Handles JSON data** efficiently.

Enhancing the Piano Game with Objects and AJAX

We'll use:

1. **Objects** to store information for each piano key.
2. **AJAX (Fetch API)** to dynamically load piano notes from a JSON file.
3. **User Interaction** to trigger sound playback.

Fetching Piano Notes from a JSON File

Instead of hardcoding the notes, we will fetch them from a JSON file.

Example JSON File (`notes.json`):

```
{
  "song": "Moonlight Sonata",
  "notes": ["C4", "E4", "G4", "C5", "E5", "G5"]
}
```

Fetch Notes Using Fetch API

```
async function loadNotes() {
    try {
        const response = await fetch("notes.json");
        const data = await response.json();
        console.log("Loaded Notes:", data.notes);
    } catch (error) {
        console.error("Error loading notes:", error);
    }
}
```

Summary

1. **Define Piano Keys as Objects**.
2. **Loop Through Object Arrays** using `forEach()`, `map()`, and `for...of`.
3. **Fetch Notes Dynamically** from a JSON file.
4. **Understand XMLHttpRequest vs Fetch API**.
5. **Use Fetch API for Asynchronous Data Retrieval**.
6. **Implement User Interaction** to play notes.

Challenge: Extend the functionality by allowing users to select different songs stored in multiple JSON files.

Weekly Assignment Breakdown

1. **Create Piano Key Objects**: Define properties for note and key.
2. **Loop Through Objects**: Use different methods to iterate over data.
3. **Fetch JSON Data Using Fetch API**: Retrieve notes dynamically.
4. **Compare Fetch API and XMLHttpRequest**.
5. **Enable User Interaction**: Play sound on key press or click.
6. **Enhance User Experience**: Allow selection of multiple song files.

Q&A: Ask questions about objects, AJAX, or event handling.