# JavaScript Fundamentals - Session 4: JSON, APIs & Data Storage

## Table of Contents

1. [JSON Deep Dive](#json)

2. [AJAX & APIs](#ajax-apis)

3. [Data Storage](#data-storage)

4. [Modern JavaScript Features](#modern-js)

5. [Error Handling](#error-handling)

6. [Hands-On Example](#hands-on)

7. [Common Mistakes](#mistakes)

---

## 1. JSON Deep Dive

### JSON Syntax Rules

```json

{

"name": "Alice",

"age": 30,

"isStudent": false,

"courses": ["JS", "HTML"],

"address": {

"city": "Cairo",

"street": "Nile"

}

}

```

- \*\*Keys must be double-quoted\*\*

- Values can be: strings, numbers, booleans, arrays, objects, `null`

- No functions or comments allowed

### JSON Methods

```javascript

// Convert JS → JSON

const jsonString = JSON.stringify({

name: "Alice",

score: 95

});

// Convert JSON → JS

const jsObj = JSON.parse('{"name":"Bob","age":25}');

// Formatting options

JSON.stringify(obj, null, 2); // Indent 2 spaces

```

### JSON vs JS Objects

| Feature | JSON | JS Objects |

|----------------|-------------------|--------------------|

| Keys | Always quoted | Unquoted allowed |

| Functions | Not allowed | Allowed |

| Dates | Converted to text | Native Date object |

| Comments | Not supported | Supported |

---

## 2. AJAX & APIs

### XMLHttpRequest (Traditional AJAX)

```javascript

const xhr = new XMLHttpRequest();

xhr.open('GET', 'https://jsonplaceholder.typicode.com/todos/1');

xhr.onload = function() {

if(xhr.status === 200) {

const data = JSON.parse(xhr.response);

console.log(data);

}

};

xhr.send();

```

### Fetch API (Modern Approach)

```javascript

// GET example

fetch('https://jsonplaceholder.typicode.com/posts/1')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

// POST example

fetch('https://jsonplaceholder.typicode.com/posts', {

method: 'POST',

body: JSON.stringify({

title: 'New Post',

body: 'Content here',

userId: 1

}),

headers: {

'Content-type': 'application/json'

}

})

.then(response => response.json())

.then(data => console.log('Created post:', data));

```

### API Concepts

- \*\*Endpoints\*\*: URL paths for resources

`GET /posts/1` vs `POST /posts`

- \*\*Status Codes\*\*:

- 200 OK

- 201 Created

- 404 Not Found

- \*\*Headers\*\*:

```javascript

fetch(url, {

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer token123'

}

});

```

---

## 3. Data Storage

### localStorage

```javascript

// Store data

localStorage.setItem('user', JSON.stringify(user));

// Retrieve data

const user = JSON.parse(localStorage.getItem('user'));

// Remove data

localStorage.removeItem('user');

// Clear all

localStorage.clear();

```

### sessionStorage

| Feature | localStorage | sessionStorage |

|-----------------|----------------------|----------------------|

| Persistence | Until manual clear | Tab/window close |

| Scope | Same origin | Per tab |

| Storage Limit | ~5MB | ~5MB |

---

## 4. Modern JavaScript Features

### The 'this' Keyword

\*\*Definition\*\*:

`this` refers to the execution context, varying based on how a function is called:

1. \*\*Global Context\*\*:

```javascript

console.log(this); // Window object (in browsers)

```

2. \*\*Object Method\*\*:

```javascript

const user = {

name: "Ali",

greet() {

console.log(this.name); // "Ali"

}

};

user.greet();

```

3. \*\*Event Handlers\*\*:

```javascript

button.addEventListener('click', function() {

console.log(this); // The button element

});

```

4. \*\*Arrow Functions\*\*:

```javascript

const obj = {

name: "Sarah",

greet: () => {

console.log(this.name); // undefined (inherits parent's 'this')

}

};

```

5. \*\*Explicit Binding\*\*:

```javascript

function showName() {

console.log(this.name);

}

const user = { name: "Ahmed" };

showName.call(user); // "Ahmed"

```

### Destructuring

\*\*Objects\*\*:

```javascript

const user = {

name: 'Ali',

age: 25,

address: { city: 'Cairo' }

};

const { name, address: { city } } = user;

console.log(name, city); // Ali Cairo

```

\*\*Arrays\*\*:

```javascript

const colors = ['red', 'green', 'blue'];

const [first, , third] = colors;

console.log(first, third); // red blue

```

### Spread Operator

\*\*Arrays\*\*:

```javascript

const arr1 = [1, 2];

const arr2 = [3, 4];

const combined = [...arr1, ...arr2]; // [1, 2, 3, 4]

```

\*\*Objects\*\*:

```javascript

const defaults = { theme: 'light', fontSize: 16 };

const userSettings = { fontSize: 14 };

const finalSettings = { ...defaults, ...userSettings };

// { theme: 'light', fontSize: 14 }

```

---

## 5. Error Handling

### Error Types

```javascript

try {

undefinedFunction(); // ReferenceError

JSON.parse('invalid'); // SyntaxError

null.property; // TypeError

} catch(error) {

console.error(`${error.name}: ${error.message}`);

}

```

### Try/Catch with Async

```javascript

async function loadData() {

try {

const response = await fetch('invalid-url');

const data = await response.json();

} catch(error) {

console.error('Failed to load:', error);

}

}

```

---

## 6. Hands-On Example

\*\*Blog Post Manager\*\*:

```html

<div id="posts"></div>

<button id="loadPosts">Load Posts</button>

<script>

const postsDiv = document.getElementById('posts');

const loadBtn = document.getElementById('loadPosts');

loadBtn.addEventListener('click', async function() {

try {

const response = await fetch('https://jsonplaceholder.typicode.com/posts');

const posts = await response.json();

// Using modern features

const [firstPost, ...otherPosts] = posts;

postsDiv.innerHTML = `

<h2>${firstPost.title}</h2>

<p>${firstPost.body}</p>

<p>Total posts: ${otherPosts.length + 1}</p>

`;

// Save to localStorage

localStorage.setItem('lastPost', JSON.stringify(firstPost));

} catch(error) {

console.error('Failed to load posts:', error);

}

});

</script>

```

---

## 7. Common Mistakes

1. \*\*JSON Security Issues\*\*

```javascript

// Dangerous!

const data = JSON.parse(untrustedString);

// Safe: Validate first

if(isValidJSON(untrustedString)) {

JSON.parse(untrustedString);

}

```

2. \*\*'this' Context\*\*

```javascript

// Wrong: Arrow function in event handler

button.addEventListener('click', () => {

console.log(this); // Window object

});

// Right: Regular function

button.addEventListener('click', function() {

console.log(this); // Button element

});

```

3. \*\*Ignoring Error Types\*\*

```javascript

// Bad: Generic catch

try { /\* code \*/ } catch { /\* \*/ }

// Good: Specific handling

try {

JSON.parse(invalidJson);

} catch(error) {

if(error instanceof SyntaxError) {

console.error("Invalid JSON");

}

}

```

---