**1. Introduction to APIs (Application Programming Interfaces)**

**What is an API?**

An **API** is a way for two systems (like a website and a server) to talk to each other.

**Example**:  
API is like a **waiter**. You (user) tell what you want, waiter (API) gives it to the kitchen (server), then brings food (data) back to you.

**Types of APIs**

| **Type** | **Description** |
| --- | --- |
| RESTful API | Uses HTTP methods like GET, POST, PUT, DELETE. |
| GraphQL | More flexible, but advanced. Not covered here. |

**API Endpoints**

An endpoint is a specific URL where the server provides some data.

**Example Endpoint**:  
https://jsonplaceholder.typicode.com/users

**HTTP Methods**:

| **Method** | **What it does** |
| --- | --- |
| GET | Get data |
| POST | Send new data |
| PUT | Update data |
| DELETE | Delete data |

**2. Using the Fetch API**

**What is fetch()?**

fetch() is a tool in JavaScript to get or send data from APIs.

**Basic GET Request**

// Get user data from API

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json()) // convert response to JSON

.then(data => console.log(data)) // show data in console

.catch(err => console.log('Error:', err)); // if error, show error

**Simple POST Request**

// Send new post to API

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

method: 'POST', // type of request

headers: {

'Content-Type': 'application/json' // sending JSON data

},

body: JSON.stringify({ // convert JS object to JSON

title: 'Hello',

body: 'This is a post',

userId: 1

})

})

.then(res => res.json()) // get response as JSON

.then(data => console.log(data)); // show returned data

**Error Handling with Fetch**

**✅ Status Codes:**

| **Code** | **Meaning** |
| --- | --- |
| 200 | OK (Success) |
| 404 | Not Found |
| 500 | Server Error |

**Check for Errors**

// Try to get data from wrong URL

fetch('https://jsonplaceholder.typicode.com/wrong-url')

.then(res => {

if (!res.ok) { // if not successful

throw new Error('Error: ' + res.status); // throw error

}

return res.json(); // convert to JSON

})

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

.catch(err => console.log(err.message)); // show error message

**Practical Activity: Build a simple web application that fetches data from a public API and displays it dynamically**

<!DOCTYPE html>

<html>

<head>

<title>User List</title>

</head>

<body>

<h2>User List</h2>

<ul id="userList"></ul> <!-- Users will be shown here -->

<script>

// Get UL element

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

// Fetch data from API

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json()) // Convert to JSON

.then(users => {

// Loop through users

users.forEach(user => {

const li = document.createElement('li'); // create list item

li.textContent = `${user.name} - ${user.email}`; // set text

userList.appendChild(li); // add to list

});

})

.catch(err => {

// If error, show message

userList.innerHTML = `<li>Error: ${err.message}</li>`;

});

</script>

</body>

</html>

**Working with JSON Data and Dynamic Rendering**

**1. What is JSON?**

**JSON** stands for **JavaScript Object Notation**.  
It is a way to **store** and **exchange** data between a client (browser) and a server.

Example (real life):  
Imagine JSON as a **box of data** with labels.  
Like:

{

"name": "Ali",

"age": 25

}

It looks like a JavaScript object — but in **string** format.

**2. JSON Syntax and Structure**

**Key-Value Pair:**

{

"name": "Zara",

"city": "Lahore"

}

**Array of Objects:**

[

{ "name": "Ali", "age": 25 },

{ "name": "Sara", "age": 30 }

]

**3. Parsing JSON Data**

JSON from an API is in **string format**.  
To use it in JavaScript, convert it into an object using:

JSON.parse(jsonString);

**Example:**

let jsonStr = '{"name":"Zara","city":"Lahore"}'; // JSON string

let obj = JSON.parse(jsonStr); // convert to JS object

console.log(obj.name); // Output: Zara

**4. Stringifying JavaScript Object**

To send data to an API, we must convert the object to a string using:

JSON.stringify(object);

**Example:**

let student = { name: "Ali", age: 20 };

let jsonText = JSON.stringify(student); // convert to JSON string

console.log(jsonText); // Output: {"name":"Ali","age":20}

**5. Fetching JSON Data from API**

We use fetch() to get JSON data from a server.

**Example:**

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json()) // convert to JS object

.then(data => console.log(data)); // show data

**6. Dynamically Rendering JSON Data in DOM**

Let's create a small project:

* We'll fetch JSON user data.
* Display name and email in the browser.

**Complete Example (All in One File)**

<!DOCTYPE html>

<html>

<head>

<title>JSON and DOM Example</title>

</head>

<body>

<h2>User List</h2>

<ul id="userList"></ul>

<script>

// Get the UL where users will appear

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

// Fetch data from API

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json()) // Parse JSON

.then(users => {

// Loop and display each user

users.forEach(user => {

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

li.textContent = user.name + " - " + user.email;

userList.appendChild(li);

});

})

.catch(err => {

userList.innerHTML = `<li>Error: ${err.message}</li>`;

});

</script>

</body>

</html>

**Handling API Data and Conditional Rendering**

**1. Handling Complex API Data**

**What is Nested JSON?**

Sometimes API data is **not flat**, it’s **nested** (inside another object or array).

**Example of Nested JSON:**

{

"name": "Ali",

"address": {

"city": "Lahore",

"zipcode": "54000"

}

}

**How to access nested data?**

Use **dot notation**:

let user = {

name: "Ali",

address: {

city: "Lahore",

zipcode: "54000"

}

};

console.log(user.address.city); // Output: Lahore

**Real API Nested Example (GET from API)**

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json())

.then(data => {

console.log(data[0].name); // User name

console.log(data[0].address.city); // Nested city

});

**2. Conditional Rendering in JavaScript**

**What is Conditional Rendering?**

It means:  
**"Show something only if a condition is true."**

**Example:**

let isLoggedIn = true;

if (isLoggedIn) {

console.log("Welcome back!");

} else {

console.log("Please login.");

}

You can also use this to show or hide HTML content:

let isAdmin = false;

if (isAdmin) {

document.body.innerHTML = "<h1>Admin Panel</h1>";

} else {

document.body.innerHTML = "<h1>Access Denied</h1>";

}

**3. Handling Empty States and Errors**

**What is an Empty State?**

When API gives **no data** or result is **empty**, we should show a friendly message.

**Example:**

let users = [];

if (users.length === 0) {

console.log("No users found.");

}