More JavaScript Concepts

Objective:

By the end of this lesson, students will:

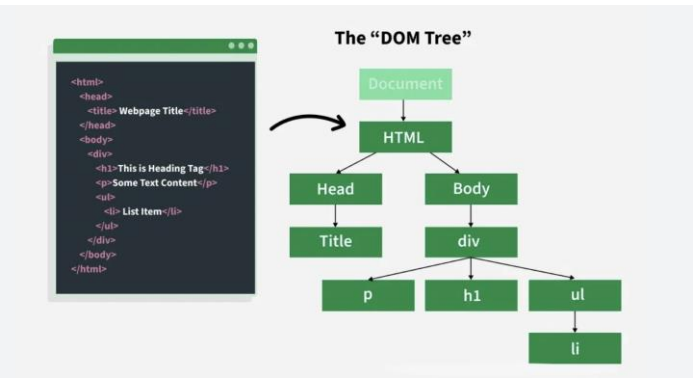
1. Understand the basics of the **DOM (Document Object Model)** and how to manipulate it using JavaScript.
2. Learn what **JSON** is and how it is used to structure data.
3. Be able to use the **Fetch API** to make **GET requests** to external APIs.
4. Gain practical experience by building a simple web application that fetches data from an API and displays it on a webpage.

---

1. Introduction to the DOM (Document Object Model)

What is the DOM?

- The **DOM** is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of objects, where each object corresponds to a part of the document (like elements, attributes, and text).
- JavaScript can be used to manipulate the DOM, allowing you to dynamically change the content, structure, and style of a webpage.



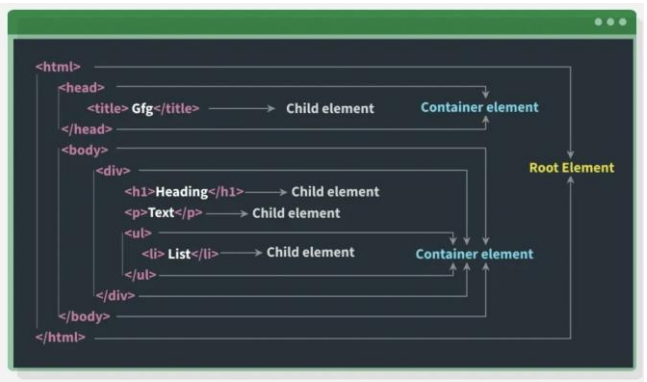


Image from geeksforgeeks

Key Concepts:

- **Selecting Elements**: You can select elements using methods like `document.getElementById()`, `document.querySelector()`, etc.
- **Manipulating Elements**: Once selected, you can change the content (`innerHTML`), attributes (`setAttribute()`), or styles (`style.property`) of elements.
- **Event Handling**: You can add event listeners to elements to respond to user actions like clicks, mouse movements, etc.

Example: Changing Text Content

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Example</title>
</head>
<body>
    <h1 id="header">Hello, World!</h1>
```

```
    <button id="changeTextBtn">Change Text</button>

    <script>
        // Select the button and header elements
        const button = document.getElementById('changeTextBtn');
        const header = document.getElementById('header');

        // Add an event listener to the button
        button.addEventListener('click', function() {
            header.textContent = 'Text Changed!';
        });
    </script>
</body>
</html>
```

Exercise 1

- Create a webpage with a button and a paragraph. When the button is clicked, change the text of the paragraph to “You clicked the button!”

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git

2. Understanding JSON (JavaScript Object Notation)

What is JSON?

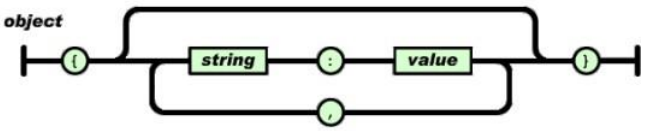
- **JSON** is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.
- It is commonly used to send data between a server and a client (e.g., in API responses).

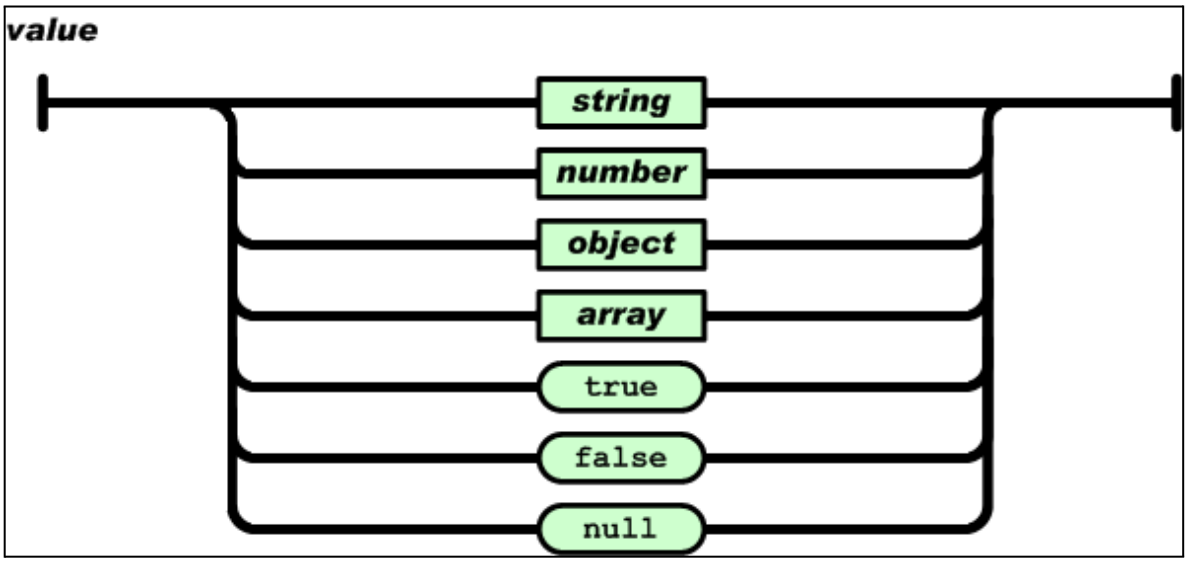
Key Concepts:

- **Structure**: JSON data is represented as key-value pairs, similar to JavaScript objects.
- **Parsing JSON**: You can convert JSON strings into JavaScript objects using `JSON.parse()`.
- **Stringifying Objects**: You can convert JavaScript objects into JSON strings using `JSON.stringify()`.

- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value
- Unordered sets of name/value pairs
- Begins with { (left brace)
- Ends with } (right brace)
- Each name is followed by : (colon)
- Name/value pairs are separated by , (comma)

```
{
"employee_id": 1234567,
"name": "Jeff Fox",
"hire_date": "1/1/2013",
"location": "Norwalk, CT",
"consultant": false
}
```





Example: Working with JSON

```
// JSON string
const jsonString = '{"name": "John", "age": 30, "city": "New York"}';

// Parse JSON string into a JavaScript object
const jsonObject = JSON.parse(jsonString);

console.log(jsonObject.name);   // Output: John
console.log(jsonObject.age);    // Output: 30

// Convert JavaScript object back to JSON string
const newJsonString = JSON.stringify(jsonObject);
console.log(newJsonString);   // Output: {"name":"John","age":30,"city":"New York"}
```

Exercise 2

- Create a JSON object representing a person with properties like `name`, `age`, and `hobbies`. Use `JSON.stringify()` to convert it to a JSON string and log it to the console.

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git

3. Using the Fetch API to Make GET Requests

What is Fetch?

- The **Fetch API** provides a modern way to make HTTP requests in JavaScript. It is used to retrieve data from a server or send data to a server.
- The most common type of request is a **GET request**, which retrieves data from a server.

Key Concepts:

- **Promise-based**: Fetch returns a promise, which resolves to the response of the request.
- **Response Handling**: You can use `.json()` to parse the response as JSON.

Example: Fetching Data from an API

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fetch API Example</title>
</head>
<body>
    <h1>Random User Data</h1>
    <div id="userData"></div>

    <script>
        // Fetch data from a public API
        fetch('https://randomuser.me/api/')
            .then(response => response.json())  // Parse the response as JSON
            .then(data => {
                // Extract user data from the response
                const user = data.results[0];
                const userDataDiv = document.getElementById('userData');

                // Display user data on the page
                userDataDiv.innerHTML = `
                    <p>Name: ${user.name.first} ${user.name.last}</p>
                    <p>Email: ${user.email}</p>
                    <img src="${user.picture.medium}" alt="User Image">
                `;
```

```
            })
            .catch(error => console.error('Error fetching data:', error));
    </script>
</body>
</html>
```

Exercise 3

- Modify the above example to display additional information about the user, such as their location (city and country) and phone number.

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git

4. Making an API GET Request

What is an API?

- An **API (Application Programming Interface)** is a set of rules and protocols that allows one software application to interact with another.
- APIs often return data in **JSON** format, which can be easily consumed by JavaScript.

Steps to Make a GET Request:

1. Use the `fetch()` function to send a GET request to the API endpoint.
2. Handle the response using `.then()` or `async/await`.
3. Parse the response as JSON using `.json()`.

Example: Fetching Data from a Weather API

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>WeatherAPI Example</title>
</head>
<body>
  <h1>WeatherAPI Example</h1>

  <button onclick="callAPI()">Fetch Weather Data</button>

  <div>
    <h2>Weather Details</h2>
    <p><strong>City:</strong> <span id="weather_city">--</span></p>
    <p><strong>Temperature:</strong> <span id="weather_temperature">--</span></p>
    <p><strong>Condition:</strong> <span id="weather_condition">--</span></p>
  </div>

    <script>
        // Replace with your WeatherAPI key
        const apiKey = 'cbb0b276f1724b6fb1420151241811';
```

```
        // Change this to the city you want to search
        const city = 'Davao';

        function callAPI() {
            const url =
`https://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${city}&aqi=no`;

            fetch(url)
            .then(response => response.json())
            .then(data => {
                console.log(data);

                document.querySelector("#weather_city").textContent =
data.location.name || 'N/A';
                document.querySelector("#weather_temperature").textContent =
`${data.current.temp_c}°C` || 'N/A';
                document.querySelector("#weather_condition").textContent =
data.current.condition.text || 'N/A';
            });
        }
    </script>
</body>
</html>
```

Exercise 4

- Sign up for a free API key from OpenWeatherMap and replace `'YOUR_API_KEY'` in the code above with your actual API key. Run the code and see the weather data for London displayed on the page.

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git

5. Putting It All Together: Build a Simple Web App

Project: Random Quote Generator

- **Objective**: Build a simple web app that fetches random quotes from an API and displays them on the page.
- **Steps**:
    1. Use the Fetch API to make a GET request to a quote API (e.g., Quotable API).
    2. Parse the JSON response and display the quote and author on the page.
    3. Add a button that allows users to fetch a new quote when clicked.

Code Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Random Quote Generator</title>
```

```
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 50px;
            background-color: #f4f4f9;
        }
        #quoteDisplay {
            margin: 20px 0;
            font-size: 1.5em;
            color: #333;
        }
        #author {
            font-style: italic;
            color: #555;
        }
        button {
            padding: 10px 20px;
            font-size: 1em;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1>Random Quote Generator</h1>
    <div id="quoteDisplay">
        <p id="quote">Click the button to get a random quote!</p>
        <p id="author"></p>
    </div>
    <button id="newQuoteBtn">Get New Quote</button>
    <script>
        const quoteElement = document.getElementById('quote');
        const authorElement = document.getElementById('author');
        const button = document.getElementById('newQuoteBtn');

        // Function to fetch a random quote
        function fetchQuote() {
            // Show loading message while fetching
            quoteElement.textContent = 'Loading quote...';
            authorElement.textContent = '';

            // Fetch all quotes from the DummyJSON API
            fetch('https://dummyjson.com/quotes')
                .then(response => {
                    if (!response.ok) {
                        throw new Error(`HTTP error! status: ${response.status}`);
```

```
                    }
                    return response.json();
                })
                .then(data => {
                    // Get a random quote from the list of quotes
                    const randomIndex = Math.floor(Math.random() *
data.quotes.length);
                    const randomQuote = data.quotes[randomIndex];

                    // Display the quote and author
                    quoteElement.textContent = `"${randomQuote.quote}"`;
                    authorElement.textContent = `- ${randomQuote.author}`;
                })
                .catch(error => {
                    // Handle any errors that occur during the fetch
                    console.error('Error fetching quote:', error);
                    quoteElement.textContent = 'Failed to load quote. Please try
again.';
                    authorElement.textContent = '';
                });
        }

        // Fetch a quote when the button is clicked
        button.addEventListener('click', fetchQuote);

        // Fetch a quote when the page loads
        fetchQuote();
    </script>
</body>
</html>
```

Exercise 5:

- Customize the app by adding styles (CSS) to make it look more appealing. You can also add functionality to share the quote on social media or copy it to the clipboard.

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git

Conclusion:

In this lesson, we covered:

1. **DOM Manipulation**: How to select and modify elements on a webpage.
2. **JSON**: How to work with JSON data in JavaScript.
3. **Fetch API**: How to make GET requests to APIs and handle responses.
4. **Building a Simple Web App**: A practical project that combines all the concepts learned.

Additional Resources:

- MDN Web Docs - DOM
- MDN Web Docs - Fetch API
- JSON.org
- Free Public APIs

Hands-on Activity

Build Your Own Interactive Web Page Using an API

In this hands-on activity, you'll create an interactive web page by choosing a public API (aside from the examples provided) that interests you, such as weather, movies, jokes, or news APIs, and fetching data from it using JavaScript *fetch()* method. Start by exploring APIs from the internet and if required, sign up for an API key.

Plan your web page by sketching a design that includes user interaction elements like buttons or search bars, and decide how to display the fetched data creatively. Write your code in an HTML file with embedded CSS for styling and JavaScript to handle the API call, parse the JSON response, and update the DOM dynamically. Ensure your page is visually appealing, responsive, and includes error handling for failed API requests.

Test your project thoroughly, then share it with the class by presenting your API choice, explaining how your page works, and showcasing its functionality live in the browser. For a bonus challenge, add extra interactivity, such as search features, animations, or local storage integration.

Paste your GitHub URL here	

https://github.com/kzndrick/W8handsonactivity.git