CoGrammar

Welcome to this session:

DOM Manipulation and Event Handling (Revision)

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Cloud Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query:

  www.hyperiondev.com/support

- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***

- We would love your feedback on lectures: Feedback on Lectures

- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

By the end of this lesson, learners should be able to:

- Fetch and display real-time data using AJAX and JSON APIs, such as personal information.
- Create and manipulate JavaScript objects and JSON data for storing structured information.
- Use session storage to save and retrieve user data in a web application.
- Develop simple applications that combine data storage, dynamic updates, and asynchronous fetching for engaging user experiences.

What is the purpose of AJAX in JavaScript?

A. To make HTTP requests and retrieve data asynchronously.
B. To improve the performance of JavaScript code execution.
C. To handle user input validation on the client side.
D. To manipulate DOM elements dynamically.

CoGrammar

What is the purpose of JSON?

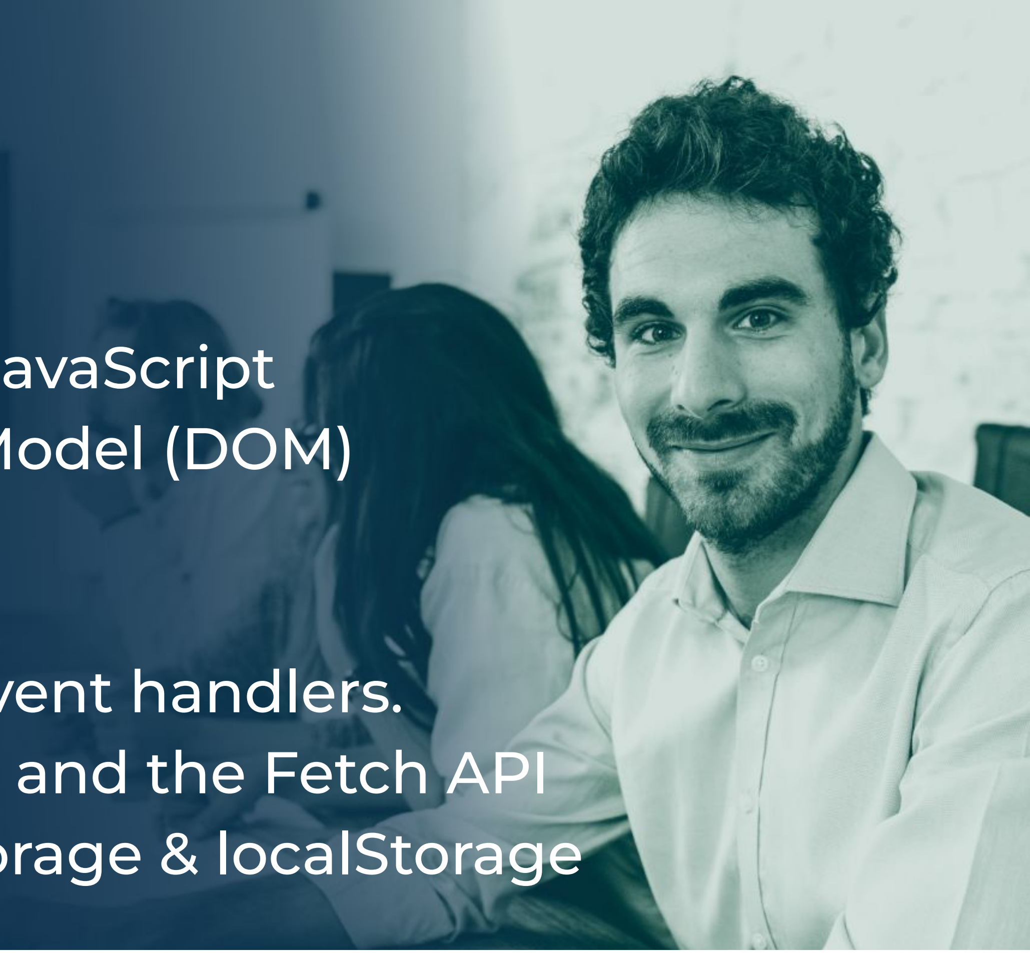
A. To compile JavaScript code for faster execution.
B. To create interactive user interfaces.
C. To store and exchange data in a lightweight, text-based format.
D. To define the structure of a database.

What is meant by DOM Manipulation?

A. Rewriting the JavaScript code to make the page more efficient.
B. Modifying the structure, style, or content of an HTML document through JavaScript.
C. Creating and handling events like clicks and form submissions.
D. Parsing JSON data to display it on the webpage.

CoGrammar

Lecture Overview



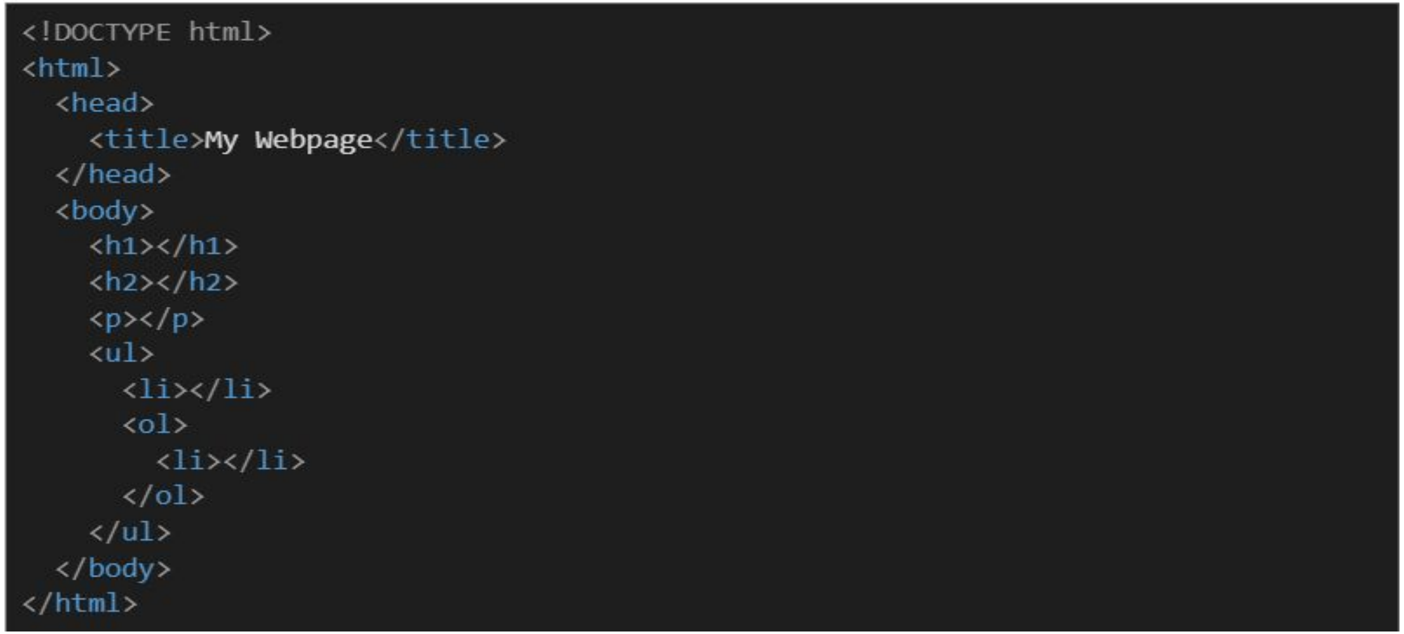
- → Integrating HTML with JavaScript
- → The Document Object Model (DOM)
- → Query Selectors
- → DOM manipulation
- → JavaScript events and event handlers.
- → Asynchronous Requests and the Fetch API
- → Web Storage: sessionStorage & localStorage

The Document Object Model (DOM)

- ❖ What is the Document Object Model (DOM)?
  - ➢ **The Document Object Model (DOM)** is a programming interface for web documents.
  - ➢ It represents the **structure** of HTML documents as a hierarchical **tree** of **objects**.
  - ➢ Each **node** in the tree corresponds to a **part** of the document, such as elements, attributes, or text content.
  - ➢ The DOM **provides** a structured representation of the document, allowing **scripts** to **dynamically** access, modify, and manipulate its content and structure.

CoGrammar

The Document Object Model Example

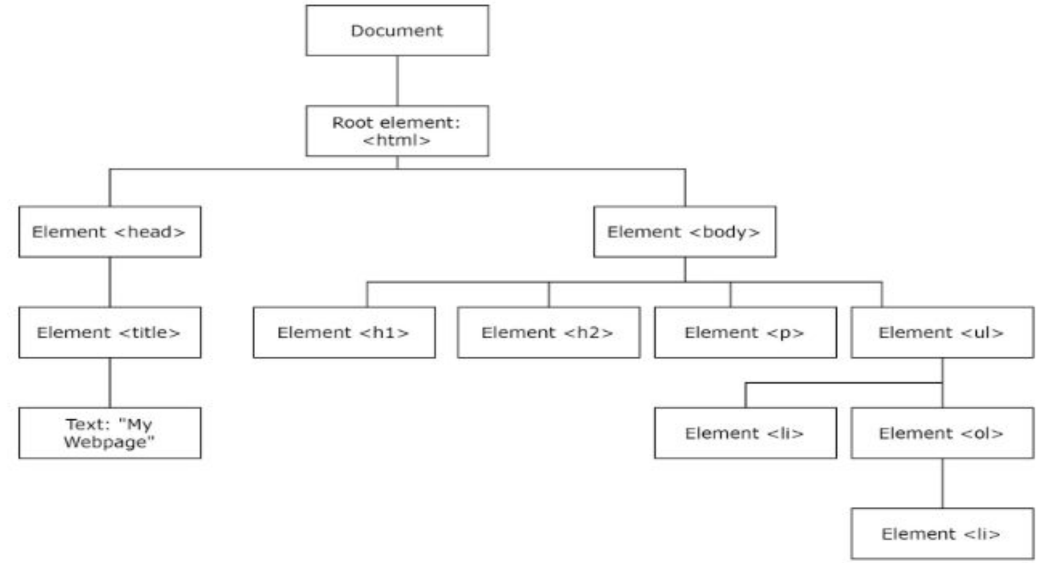


```
<!DOCTYPE html>
<html>
  <head>
    <title>My Webpage</title>
  </head>
  <body>
    <h1></h1>
    <h2></h2>
    <p></p>
    <ul>
      <li></li>
      <ol>
        <li></li>
      </ol>
    </ul>
  </body>
</html>
```

CoGrammar

The Document Object Model Example

❖ This would be graphically represented as a tree like this:



CoGrammar

DOM Manipulation

- What is **DOM** Manipulation?
    - DOM manipulation refers to the process of **dynamically** altering the structure, content, or style of web documents using **scripting languages** like JavaScript.
    - It allows developers to create **interactive** and **dynamic** web pages by **accessing** and **modifying** elements in the Document Object Model (DOM).

CoGrammar

DOM Manipulation

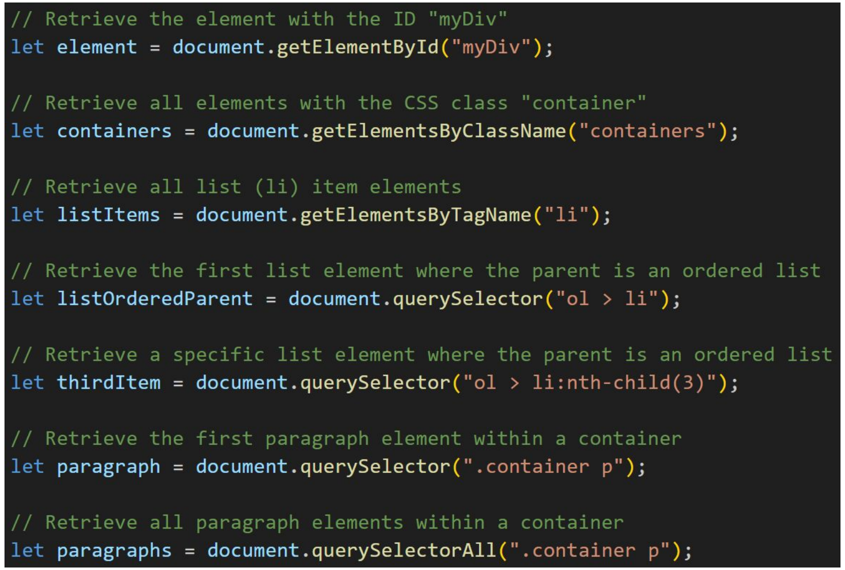
- ❖ Why is **DOM** Manipulation significant?
  - ➢ It enables developers to respond to user actions, update content dynamically, and create engaging user experiences without page reloads.
  - ➢ It forms the foundation for modern web applications, facilitating tasks such as form validation, content updates, and animation effects.
  - ➢ It empowers developers to create responsive and interactive interfaces, enhancing usability and user engagement.

Document Object

- What is the **document** object?
  - The **document** object represents the entire HTML document as a tree structure.
  - It serves as the entry point to the web page's content, allowing **manipulation** and **interaction** with its elements.
  - The **document** object serves as the root node of the Document Object Model (DOM) tree.
  - It offers various **properties** and **methods** for interacting with the document's structure and content.
  - It serves as a fundamental component for **dynamic** web development, enabling developers to create **responsive** and **interactive** user interfaces.

DOM Traversal

❖ DOM **traversal** involves navigating through the DOM tree to access or manipulate elements.



```
// Retrieve the element with the ID "myDiv"
let element = document.getElementById("myDiv");

// Retrieve all elements with the CSS class "container"
let containers = document.getElementsByClassName("containers");

// Retrieve all list (li) item elements
let listItems = document.getElementsByTagName("li");

// Retrieve the first list element where the parent is an ordered list
let listOrderedParent = document.querySelector("ol > li");

// Retrieve a specific list element where the parent is an ordered list
let thirdItem = document.querySelector("ol > li:nth-child(3)");

// Retrieve the first paragraph element within a container
let paragraph = document.querySelector(".container p");

// Retrieve all paragraph elements within a container
let paragraphs = document.querySelectorAll(".container p");
```

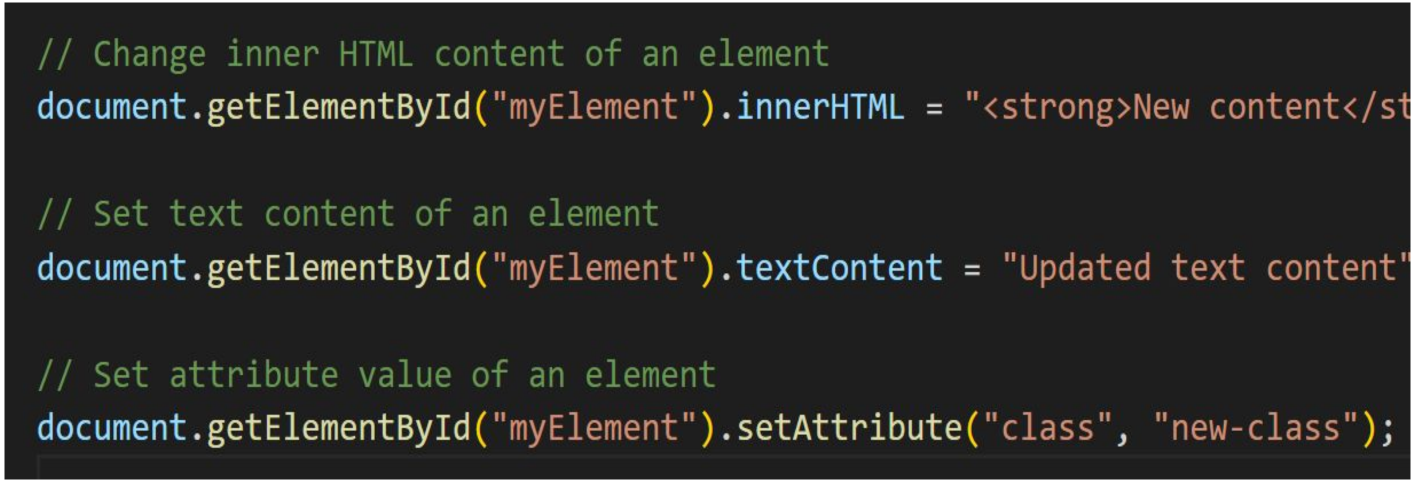
CoGrammar

DOM Manipulation

- Creating new elements:
  - It is possible to create entirely new elements using the createElement method in JavaScript
  - *document.createElement("p");*
- Appending to elements:
  - We can add both text and new elements to existing elements in our HTML file.
  - *document.body.appendChild(myItem);*

DOM Manipulation

❖ Modifying elements:

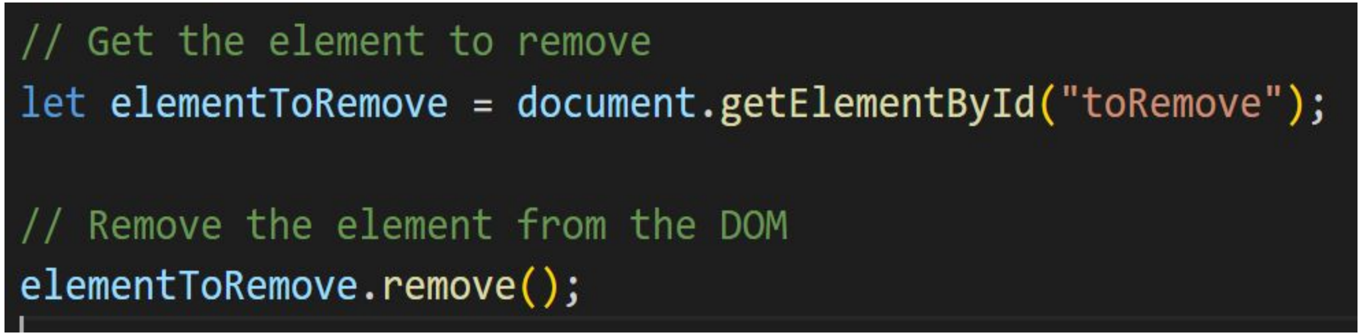


```
// Change inner HTML content of an element
document.getElementById("myElement").innerHTML = "<strong>New content</st

// Set text content of an element
document.getElementById("myElement").textContent = "Updated text content"

// Set attribute value of an element
document.getElementById("myElement").setAttribute("class", "new-class");
```

CoGrammar

DOM Manipulation

❖ Removing elements:



```
// Get the element to remove
let elementToRemove = document.getElementById("toRemove");

// Remove the element from the DOM
elementToRemove.remove();
```

CoGrammar

Events

- ❖ JavaScript is often used to handle events that occur on a website.
- ❖ An **event** is an **action** that occurs that your program responds to.
- ❖ DOM **events** are either things that a user does, such as clicking a button or entering text into a text box, or actions caused by the browser, such as when a web page has been completely loaded.
- ❖ **Event handling** is the process of **capturing**, **processing**, and **responding** to events.

CoGrammar

Inline Event Handlers

❖ The most common events you'll deal with when getting started are:

Event	Description
`onchange`	Triggered when the value of an HTML element (like an input box or dropdown menu) changes.
`onclick`	Happens when you click on an HTML element, such as a button or link.
`onmouseover`	Occurs when you move the mouse pointer over an HTML element.
`onmouseout`	Happens when you move the mouse pointer away from an HTML element.
`onkeydown`	Triggered when you press a key on the keyboard.
`onload`	Occurs when the entire HTML page and all its resources (like images) have finished loading.

CoGrammar

Inline Event Handlers

- HTML attribute equivalents for event handlers exist, but using them is discouraged due to bad practice.
- Inline event handler attributes may seem convenient for quick tasks but lead to inefficiency and manageability issues.
- Mixing HTML and JavaScript makes code harder to read; separating them promotes better organization and reuse across multiple documents.
  - In files with multiple elements (e.g., 100 buttons), using inline handlers results in excessive attributes, complicating maintenance.
- Many server configurations restrict inline JavaScript for security reasons.
- Overall, HTML event handler attributes are outdated and should be avoided.

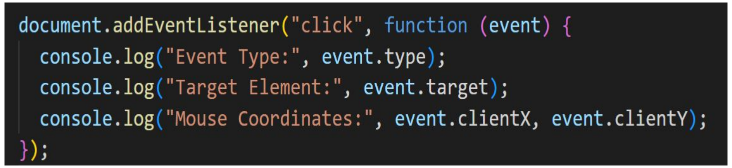
CoGrammar

Handling UI Events

- One common approach to handle UI events is by using the **addEventListener()** method. This method allows developers to attach event listeners to DOM elements and specify callback functions to be executed when the events occur.
- The method has the following syntax:
  - *element.**addEventListener**(**event**, **handler**, **options**);*
  - **event**: A string representing the type of event to listen for (e.g., "click", "keydown").
  - **handler**: A function that is called when the event occurs.
  - **options** (optional): An object that can specify properties like *capture*, *once*, and *passive*.

The event object

- In JavaScript, when an event occurs, an **event** object is automatically created by the browser.
- The event object contains information about the event that occurred, such as its type, target element, and additional event-specific data.
- In codebases, you may see the event object argument named as one of the following:
  - event
  - evt
  - e (most commonly found)

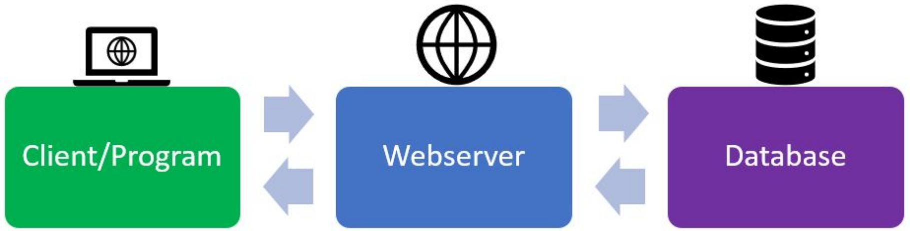


```
document.addEventListener("click", function (event) {
  console.log("Event Type:", event.type);
  console.log("Target Element:", event.target);
  console.log("Mouse Coordinates:", event.clientX, event.clientY);
});
```

CoGrammar

What are APIs?

- An API (**A**pplication **P**rogramming **I**nterface) allows different software systems to communicate with each other.
- It defines the methods and data formats that applications can use to request and exchange information.
- APIs are fundamental for web development, enabling interaction with remote servers and services.



CoGrammar

Sending Objects between a Web Server and Clients

- ❖ **HTTP** allows information to be transferred across the internet. You need to keep in mind these 2 important facts about HTTP:
  - ➢ HTTP is a **stateless protocol** — it doesn’t see any link between two requests being successively carried out on the same connection. **Cookies** and the **Web Storage API** are used to store necessary state information.
  - ➢ HTTP **transfers text** (not objects or other complex data structure). To send objects or other structured data, we need to turn them into text so they can be transferred using HTTP. Later, we can convert that text back into the original data structure once it's received.

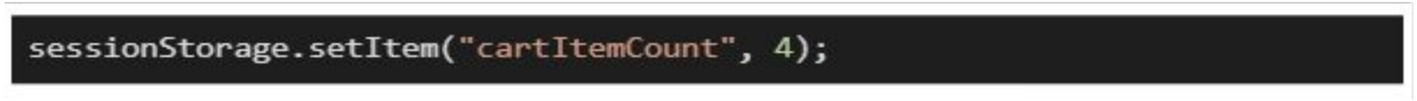
CoGrammar

Web Storage API
Session & Local Storage

- The Web Storage API allows us to save data in the browser using key-value pairs.
  - sessionStorage
    - Stores data only for the duration of the current session. In other words, the data remains accessible only while the browser is open. Once the browser is closed, any data stored in sessionStorage is deleted.
  - localStorage
    - Retains data across multiple sessions. Unlike sessionStorage, the data remains accessible even after the browser is closed and reopened, staying available until it is manually deleted.

Web Storage API
Session Storage syntax

❖ Adding a value to sessionStorage:



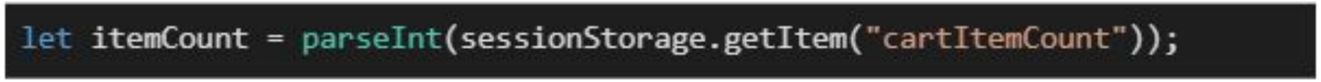
```
sessionStorage.setItem("cartItemCount", 4);
```

- .setItem() is a method used to add or update data in sessionStorage.
- The first argument, "cartItemCount", is the key.
- The second argument, 4, is the value.

CoGrammar

Web Storage API
Session Storage syntax

❖ Retrieving a value from sessionStorage:



```javascript
let itemCount = parseInt(sessionStorage.getItem("cartItemCount"));
```

- .getItem() is a method used to get the value of a specific item stored in sessionStorage.
- The argument "cartItemCount" is the key used to identify the data you want to retrieve.
    - If there's no value stored with that key, itemCount will be null.

CoGrammar

What is JSON?

- ❖ JSON (**J**avaScript **O**bject **N**otation) is a lightweight, language-independent data format used to exchange data between a web server and a client.
- ❖ Key features of JSON:
  - ➢ Lightweight.
  - ➢ Language independent.
  - ➢ Easy to parse.

CoGrammar

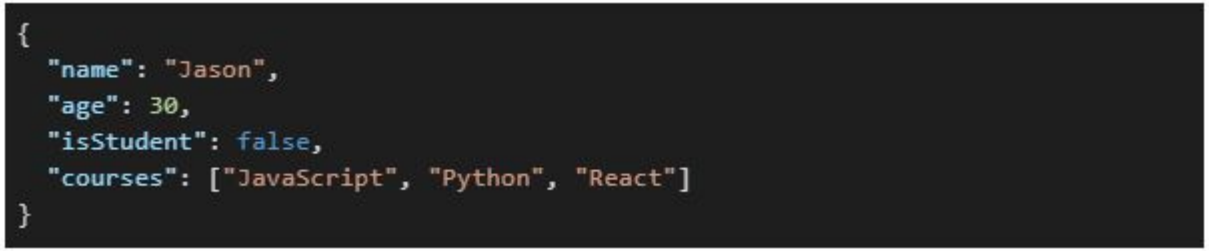
JSON’s syntax

❖ JSON’s syntax is similar to JavaScript object notation, but it has its own rules and conventions:

➢ Data in key-value pairs.

➢ Property names and values.

➢ Objects and arrays.



```
{
  "name": "Jason",
  "age": 30,
  "isStudent": false,
  "courses": ["JavaScript", "Python", "React"]
}
```

JSON's Methods: JSON.parse()

- **Reminder**: HTTP transfers **text** (not objects or other complex data structures).
- JSON.parse()
  - The JSON.parse() method in JavaScript is used to convert a JSON string into a corresponding JavaScript object.
  - Key Points:
    - **Input**: A string in valid JSON format.
    - **Output**: A JavaScript object or array based on the JSON data.
    - Throws an error if the string is not valid JSON.

CoGrammar

JSON's Methods: JSON.stringify()

- **Reminder**: HTTP transfers **text** (not objects or other complex data structures).
- JSON.stringify()
  - The JSON.stringify() method in JavaScript is used to convert a JavaScript object or value into a JSON string.
  - Key Points:
    - **Input**: A JavaScript object or value (e.g., array, boolean, number).
    - **Output**: A JSON-formatted string.

CoGrammar

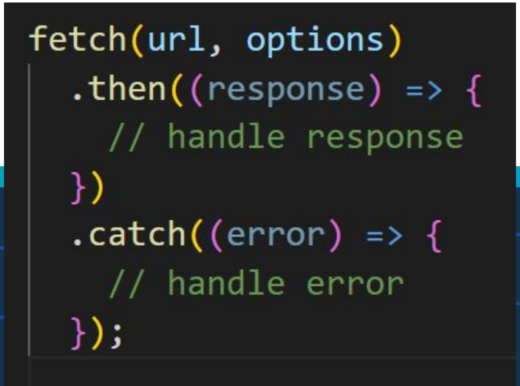
Overview of the Fetch API

- ❖ The Fetch API is a modern JavaScript interface for making **asynchronous** HTTP requests.
- ❖ Fetch follows a **promise-based** approach, which simplifies the handling of asynchronous operations, making code more readable and maintainable.
- ❖ Fetch returns a **Promise** that resolves to a **Response** object, allowing for seamless chaining of asynchronous operations using .then() and .catch() methods.

CoGrammar

fetch() Method

- ❖ The fetch() function is used to initiate a request to the specified url.
- ❖ It returns a promise that resolves to the Response object representing the response to the request.
- ❖ The options parameter is an optional object containing settings for the request such as method, headers, body, etc.



```
fetch(url, options)
  .then((response) => {
    // handle response
  })
  .catch((error) => {
    // handle error
  });
```

CoGrammar

fetch() Method Parameters

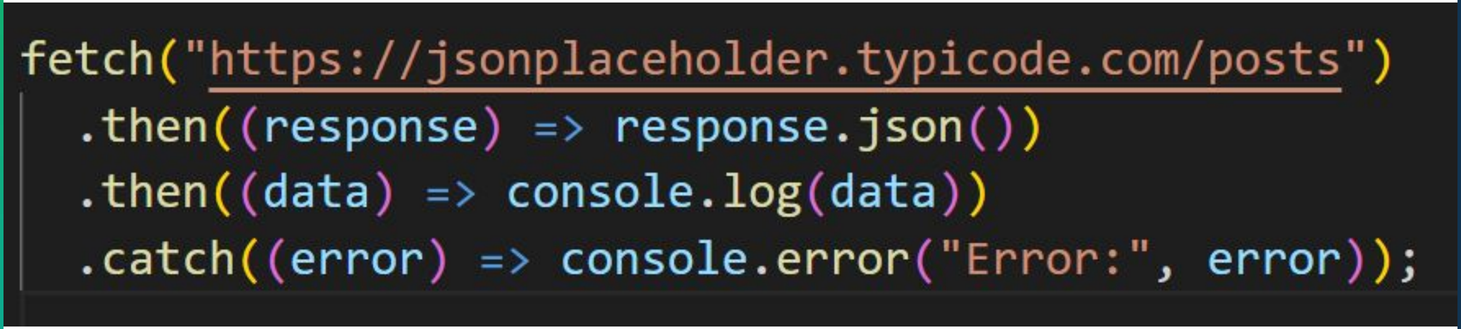
- URL (required):
  - Specifies the URL to which the request is made.
- Options (optional): An object containing various settings for the request such as:
  - method: HTTP method (GET, POST, PUT, DELETE, etc.)
  - headers: Headers to include in the request.
  - body: Data to send with the request (e.g., JSON, FormData)

CoGrammar

Response Handling

- Response Object:
  - Represents the response to the request made by fetch().
  - Contains properties and methods to access response data and metadata.
- .json():
  - Parses the response body as JSON.

Making a GET Request



```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```

CoGrammar

What is the purpose of sessionStorage?

A. To store data that persists across multiple browser sessions.
B. To store data temporarily for the duration of the page session.
C. To store data on the server for later retrieval.
D. To track user activity on different websites.

CoGrammar

Can JSON objects contain methods?

A. Yes, JSON objects can contain methods.
B. No, JSON objects can only contain data (key-value pairs).
C. JSON objects can contain methods but only if they are serialized.
D. JSON objects can contain methods but only in certain browsers.

What does the addEventListener() method do in JavaScript?

A. It creates HTML elements.
B. It stores data for events.
C. It stops events from happening.
D. It adds an event to an HTML element.

CoGrammar

Let's take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you
for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education