CoGrammar

Welcome to this session:

Task Walkthrough

Skills Bootcamp - Task 12-14

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

- Create and manipulate HTML elements using JavaScript, updating the DOM in response to user input.
- Define JavaScript functions to add, modify, and delete items within an interactive list.
- Use event listeners to make a webpage interactive, responding to user actions such as clicks and keyboard input.
- Apply CSS dynamically to indicate changes in task status visually.

What does fetch() in JavaScript do?

A. Retrieves data from an API.
B. Modifies the DOM directly.
C. Stores data locally.
D. Updates an array in memory.

Which of the following methods processes a JSON response from fetch?

A. .json()
B. .text()
C. .process()
D. .data()

Lecture Overview

- ➔ Presentation of the Task
- ➔ Introduction
- ➔ Task Walkthrough

Task: Build an Interactive To-Do List

- **Objective**: Apply your knowledge of fetch() and DOM manipulation to create a functional to-do list application.
- **Requirements**:
  - **Fetch Data**: Retrieve a list of to-do items from https://jsonplaceholder.typicode.com/todos?_limit=5.
  - **Display Tasks**: Dynamically render fetched tasks in the DOM.
  - **Add New Tasks**: Include an input box and a button to add new tasks.
  - **Mark Tasks as Completed**: Enable users to toggle tasks as completed.
- **Bonus**:
  - Style the application using CSS.
  - Allow users to delete tasks.

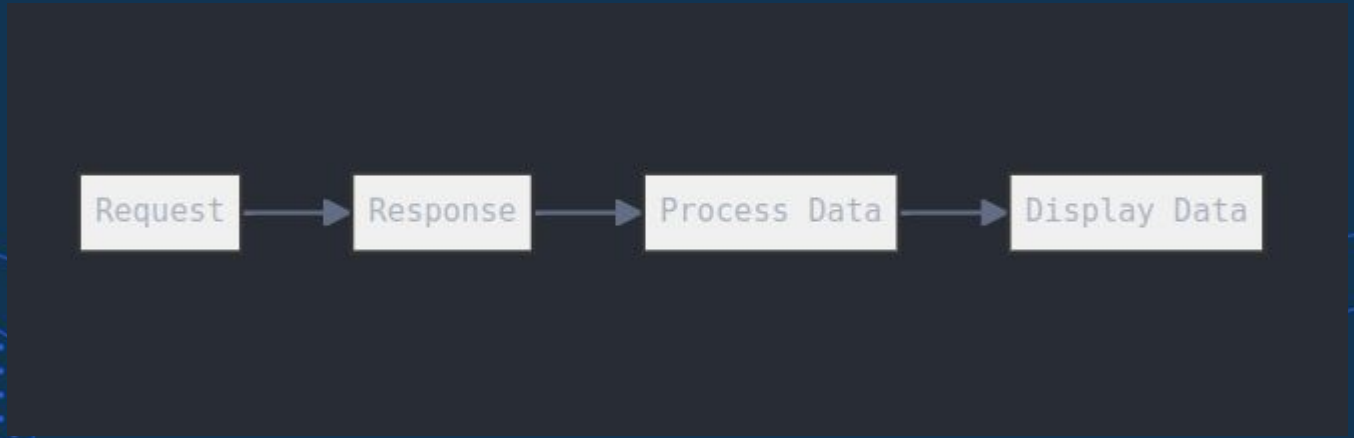
What is `fetch()`?

- **Definition:**
  - The `fetch()` API is a modern interface for retrieving resources (e.g., JSON data) from servers.
- **Key Features:**
  - Returns a Promise.
  - Handles HTTP requests and responses.

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

Steps in Fetching Data

- Use `fetch(url)` to make a request.
- Handle the response:
  - Convert the response to JSON using `.json()`.
- Process and display the data.
- Handle errors with `.catch()`.



Setting Up the Project

- **Folder Structure:**
  - project/
    - index.html
    - script.js
    - style.css

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>To-Do List</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Interactive To-Do List</h1>
  <ul id="task-list"></ul>
  <input type="text" id="task-input" placeholder="Add a new task">
  <button id="add-task">Add Task</button>
  <script src="script.js"></script>
</body>
</html>
```

Fetching Data

- **Key Points:**
  - The `_limit` query limits results to 5 tasks.
  - Always use `.catch()` to handle potential errors.

```
1  // Fetching tasks from a public API
2  fetch('https://jsonplaceholder.typicode.com/todos?_limit=5')
3    .then(response => response.json())
4    .then(data => {
5      console.log(data); // Log fetched tasks
6      displayTasks(data); // Call function to display tasks
7    })
8    .catch(error => console.error('Error:', error));
```

Displaying Data in the DOM

- **Key Concepts:**
  - Loop through the array of tasks.
  - Create a `<li>` element for each task.
  - Append it to the `<ul>` in the DOM.

```
function displayTasks(tasks) {
  const taskList = document.getElementById('task-list');
  tasks.forEach(task => {
    const li = document.createElement('li');
    li.textContent = task.title;
    taskList.appendChild(li);
  });
}

```

Adding New Tasks

- **Key Points:**
  - Listen for button clicks.
  - Retrieve the input value and create a new task.
  - Clear the input field after adding the task.

```
const addTaskButton = document.getElementById('add-task');
addTaskButton.addEventListener('click', () => {
  const taskInput = document.getElementById('task-input');
  const taskText = taskInput.value;

  if (taskText) {
    const li = document.createElement('li');
    li.textContent = taskText;
    document.getElementById('task-list').appendChild(li);
    taskInput.value = '';
  }
});
```

Enhancing the To-Do List

- Mark Tasks as Completed

```
1   const taskList = document.getElementById('task-list');
2   taskList.addEventListener('click', event => {
3     if (event.target.tagName === 'LI') {
4       event.target.classList.toggle('completed');
5     }
6   });
```

Delete Tasks

```
function deleteTask(taskElement) {
  taskElement.remove();
}
```

CSS for Completed Tasks

```
.completed {
  text-decoration: line-through;
  color: gray;
}
```

CoGrammar

Summary

- **Today We Covered:**
  - Fetching data from APIs using `fetch()`.
  - Dynamically rendering data in the DOM.
  - Adding interactivity (adding, marking, and deleting tasks).

What is the purpose of .then() in the fetch() API?

A. To handle the promise returned by fetch().
B. To fetch additional data.
C. To render elements in the DOM.
D. To log data to the console.

How do you dynamically create a new DOM element?

A. document.createElement()
B. document.appendChild()
C. document.getElementById()
D. document.removeChild()

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education