CoGrammar

Welcome to this session:

Fetching Data and Displaying with DOM - Tutorial

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.***
- Find all the lecture **content** in your **Lecture Backpack** on GitHub.
- If you are hearing impaired, kindly use your computer's function through Google chrome to enable captions.

Which method is used to select an element by its ID in the DOM?

A. getElementById()
B. querySelector()
C. getElementByClass()
D. selectById()

CoGrammar

Which method is used to parse the response body as JSON?

A. response.text()
B. response.json()
C. response.parse()
D. response.data()

CoGrammar

Learning Outcomes

- Implement the DOM to create dynamic HTML elements
- Understand how dynamic elements can connect to external data sources using the Fetch API
- Implement the Web Storage API to cache API responses

Lecture Overview

- ➔ Create static page content using HTML
- ➔ Create dynamic content using the DOM
- ➔ Use fetch to get data from an API
- ➔ Using the Web Storage API to store API responses locally.

Problem Statement

Create an online catalog for displaying products from the Fake Store API.

The following should be included on the website:

- Show the products on the home page
- When a user click on a product, they should be presented with a product details page which provides information on the product
- Product information should be cached to reduce the number of calls that are made to the API as the user navigates between the catalog and products page.

CoGrammar

Development Process

1. Explore the API
2. Identify and implement the response models
3. Design and implement the user interface
4. Link to the API

CoGrammar

Exploring the API

- Before we can call an API in our application, we need to know how the API works.
- We can test the API using
  - Postman
  - Console
  - cURL
- The goal is to:
  - Understand how API requests are structured
  - Identify any requirements for making specific queries and how those requirements can be met *(eg, a specific API call might require an ID, where would this ID come from)*
  - Understand how API responses are structured and identify relevant information.

CoGrammar

Identify the Model

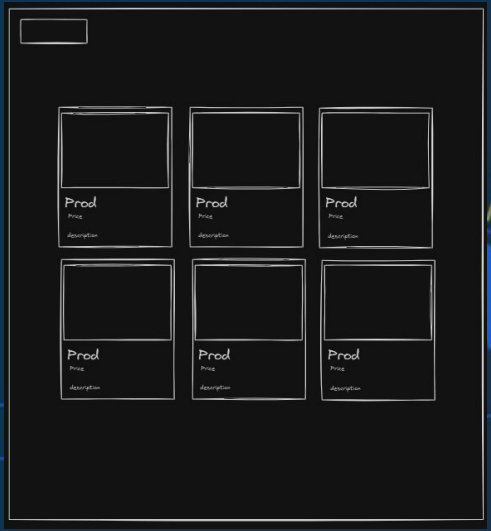
- APIs usually return more information than we need
- We can identify the parts of the response that are important to us and use them in our application.
- We can use **Object Constructors** to describe our response models.

```
function Product(id, title, price, description, image) {
    this.id = id;
    this.title = title;
    this.price = price;
    this.description = description;
    this.image = image;
}
```

CoGrammar

User Interface

- It's important to plan out the user interface before writing any code
- A UI design can be used to identify **static** and **dynamic** components on our page.
- We can use a simple wireframe to show where everything will be on the screen.



CoGrammar

Adding HTML Structure

- Not all of the content will be dynamic
- We can create the structure of the website normal HTML and CSS
- We can include containers where our dynamic content will be displayed.

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

    <header>
        <h1>Fake Store</h1>
    </header>

    <main>
        <h2>Products</h2>
        <ul id="products">
        </ul>
    </main>

    <script src="models.js"></script>
    <script src="app.js"></script>
</body>
</html>
```

CoGrammar

Create Dynamic Content

- We can use the **DOM** to generate the dynamic elements on the page.
- We are able to display our content by connecting to an existing element on the HTML page.

```
const productListContainer = document.getElementById('products');

function createProductElement(product) {
    const productItem = document.createElement('li');
    productItem.innerHTML = `<h3>${product.title}</h3>
                    <p>${product.description}</p>
                    <img src="${product.image}" alt="${product.title}">
                    <p>Price: ${product.price}</p>`;

    productListContainer.appendChild(productItem);
}
```

CoGrammar

Getting Data From the API

- We can use the **fetch** method to make calls to our API
- We can extract the information we need from the response and create an object to store the data.
- We will need to guard against any errors when making a call to the API

```
async function getProducts() {
    try {
        const response = await fetch('https://fakestoreapi.com/products');
        const data = await response.json();
        const products = data.map(product => {
            return new Product(product.id, product.title, product.price, product.description, product.image);
        });

        return products;
    } catch (error) {
        console.error(error);
        return null;
    }
}
```

CoGrammar

Questions and Answers

CoGrammar

Displaying the Products Dynamically

- We can create a function that will trigger all of the operations that need to take place when the page loads.

```
async function main() {
    const products = await getProducts();
    products.forEach(product => createProductElement(product));
}
```

CoGrammar

Web Storage API

- Everytime we load the page, we're making a new call to the API
- Making repeated calls can get expensive
  - If your using a public API, it might be billed per request or you have a fixed number of calls per price tier
  - If you have a private API, the more traffic there is, the more server resources you need to prevent slow performance.
- We can use **sessionStorage** or **localStorage** to cache responses
- This technique is good for storing API calls that are made frequently and don't change their results often.

CoGrammar

Web Storage API

- Before making our API call, we need to check if we have any stored data
- If we make the API call, we need to store the results before returning the data.

```javascript
async function getProducts() {
    // Check if we have products in session storage
    if (sessionStorage.getItem('products')) {
        const data = JSON.parse(sessionStorage.getItem('products'));
        return data.map(product => {
            return new Product(product.id, product.title, product.price, product.description, product.image);
        });
    }

    try {
        const response = await fetch('https://fakestoreapi.com/products');
        const data = await response.json();
        const products = data.map(product => {
            return new Product(product.id, product.title, product.price, product.description, product.image);
        });

        // Save products to session storage
        sessionStorage.setItem('products', JSON.stringify(products));

        return products;
    } catch (error) {
        console.error(error);
        return null;
    }
}
```

CoGrammar

Which two types of storage are provided by the Web Storage API?

A. Cache Storage and Cookie Storage
B. IndexedDB and Local Storage
C. Cookie Storage and IndexedDB
D. Session Storage and Local Storage

CoGrammar

Which two types of storage are provided by the Web Storage API?

A. localStorage can only store strings, whereas sessionStorage can store objects.
B. sessionStorage is faster than localStorage.
C. localStorage persists even after the browser is closed, while sessionStorage is cleared when the session ends.
D. localStorage requires more permissions than sessionStorage.

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education