CoGrammar

Welcome to this session:

Task Walkthrough - Object-Oriented Programming

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 JavaScript object constructors** to define custom objects with specific properties.
- **Implement functions to interact with objects** in an array by searching, sorting, and updating values.
- **Present data clearly in the console,** using console.table() and template literals to structure outputs.
- **Develop problem-solving skills for organising and managing dynamic data** in real-world applications.

What can an object in JavaScript be used for?

A. To perform mathematical calculations.
B. To create functions.
C. To loop through arrays.
D. To store multiple values as key-value pairs.

Which of the following is a valid way to create an object in JavaScript?

A. let myObject = {};
B. function() { return this; }
C. myObject = []
D. const newArray = new Array()

Lecture Overview

- ➔ Presentation of the Task
- ➔ Introduction to WD
- ➔ Introduction to HTML
- ➔ Task Walkthrough

OOP Task

Imagine you're managing a book collection for an online bookstore 📚! For each book, you'll keep track of its title, author, ISBN, quantity, and cost per copy. Using JavaScript, you'll create Book objects and store them in an array to simulate a catalogue. Then, you'll build functions to search for books, find the most and least valuable books, update book information, and sort books by value.

- Create a Book constructor function.
- Write the necessary functions for the required functionality.
- Use console.table() and template literals to format and display each function's output in an easy-to-read layout.
- Test the system by creating an array of books.

What are Objects in JavaScript?

- **Objects** in JavaScript are fundamental data structures consisting of **key-value** pairs.
- **Keys** are strings (or symbols), and **values** can be any data type, including other objects.
- Objects provide a powerful way to represent complex **data structures** and **entities** in JavaScript.

```
// Creating an object
let person = {
    name: "John Doe",
    age: 30,
    city: "New York"
};
```

CoGrammar

Classes in JavaScript

- **ES6** introduced **class** syntax to JavaScript, offering a more familiar and structured way to create objects and manage inheritance.
- Classes provide syntactic sugar over prototype-based inheritance, making object-oriented programming in JavaScript more intuitive.

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    greet() {
        return `Hello, my name is ${this.name} and I'm ${this.age} years old.`;
    }
}

// Creating an instance of the class
let person1 = new Person("Alice", 25);
console.log(person1.greet());
```

Constructor Functions

- **Constructor functions** are traditional JavaScript functions used for **creating** objects before the introduction of classes.
- They are invoked using the **new** keyword and initialize object properties using **this**.

```
// Constructor function
function Car(make, model) {
    this.make = make;
    this.model = model;
}

// Creating an instance using the constructor function
let car1 = new Car("Toyota", "Camry");
console.log(car1.make);
```

Working with Object Methods

- Objects in JavaScript can have **methods**, which are functions associated with the **object**.
- These methods can **access** and manipulate the object's properties.

```
// Object methods
let person = {
    name: "John Doe",
    greet: function() {
        return `Hello, my name is ${this.name}.`;
    }
};

console.log(person.greet()); // Output: "Hello, my name is John Doe."
```

Understanding ‘this’ in JavaScript

- In JavaScript, the **this** keyword refers to the current execution **context**.
- Its **value** is determined by how a function is called.

```
let person = {
    name: "John Doe",
    greet: function() {
        return `Hello, my name is ${this.name}.`;
    }
};

let anotherGreet = person.greet;
console.log(anotherGreet()); // Output: "Hello, my name is undefined."
```

CoGrammar

Accessor Properties in JavaScript Objects

- Accessor properties in JavaScript objects are defined using **getters** and **setters**.
- They allow for **controlled access** and **manipulation** of object **properties**.

```
let obj = {
    _name: "John",
    get name() {
        return this._name.toUpperCase();
    },
    set name(value) {
        this._name = value;
    }
};

obj.name = "Alice";
console.log(obj.name); // Output: "ALICE"
```

Iterating Over Object Properties

- JavaScript provides **various** methods for **iterating** over object **properties**, including **for...in loop** and **Object.keys()**, **Object.values()**, and **Object.entries()** methods.

```
let obj = {
    name: "John",
    age: 30,
    city: "New York"
};

// Using for...in loop
for (let key in obj) {
    console.log(`${key}: ${obj[key]}`);
}

// Using Object.keys()
let keys = Object.keys(obj);
console.log(keys); // Output: ["name", "a
```

OOP Task

Imagine you're managing a book collection for an online bookstore 📚! For each book, you'll keep track of its title, author, ISBN, quantity, and cost per copy. Using JavaScript, you'll create Book objects and store them in an array to simulate a catalogue. Then, you'll build functions to search for books, find the most and least valuable books, update book information, and sort books by value.

- Create a Book constructor function.
- Write the necessary functions for the required functionality.
- Use console.table() and template literals to format and display each function's output in an easy-to-read layout.
- Test the system by creating an array of books.

What does console.table() do in JavaScript?

A. Creates a table in HTML.
B. Outputs array data as a formatted table in the console.
C. Loops through an array and prints each element.
D. Summarises data in an object.

Which method can be used to sort an array of objects in ascending order by a specific property?

A. array.reverse()
B. array.push()
C. array.sort()
D. array.filter()

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