



Welcome to this session: 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

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.



What is the main reason for creating functions in JavaScript?

- A. **Code Reusability:** To avoid duplicating code and make it reusable across different parts of the program.
 - B. **Improved Organization:** To structure code more clearly, making it easier to read and maintain.
 - C. **Encapsulation:** To group related operations and data together, promoting modular design.
 - D. **Parameterization:** To create flexible code that can operate on different data inputs.
 - E. All of the above.
- 



What do you believe is the primary purpose of the `.length` property on a string in JavaScript?

- A. **To Determine String Length:** To find out how many characters are in a string.
- B. **To Check for Empty Strings:** To easily check if a string is empty by comparing `.length` to zero.
- C. **To Facilitate String Manipulation:** To assist in string operations like slicing or substring extraction.
- D. **To Loop Through Characters:** To provide a way to iterate over each character in the string using its length.
- E. All of the above.

Learning Outcomes

- Grasp the fundamentals of Object-Oriented Programming in JavaScript.
- Create, access, and manipulate objects in JavaScript to maintain and organize data efficiently.
- Explain the use of the *this* keyword and *prototype inheritance* in JavaScript.

Lecture Overview

- Introduction to Object-Oriented Programming (OOP)
- JavaScript Objects
- Working with Objects in javascript (Accessing and manipulation of objects)

INTRODUCTION TO OOP

- ❖ What is object-oriented programming?
 - Object-Oriented Programming (OOP) mirrors how people perceive the world—through objects.
 - OOP organises code around real-world entities, simplifying complex problems and making systems intuitive.
 - Concepts like “windows” or “folders” in software mirror their physical counterparts, reducing cognitive load for users and developers.

INTRODUCTION TO OOP

- ❖ Procedural programming versus object-oriented programming
 - Procedural programming - instructions are executed sequentially, one after another. Functions are independent but can access global variables. The data is stored separately from the code that processes it. The key features are sequential execution and the separation of data from the functions that operate on it.

The four pillars of OOP

- Encapsulation.
- Abstraction.
- Inheritance.
- Polymorphism.

Is JavaScript considered as a pure OOP language?

- Prototype-Based OOP.
- Objects and Functions.
- Flexible Typing.
- Mixins and Composition.
- No Enforced Structure.

INTRODUCTION TO OOP

- OOP - a system is designed in terms of objects that communicate with each other to accomplish a given task. Instead of separating data and code that manipulates the data, these two are encapsulated into a single module. Data is passed from one module to the next using methods.

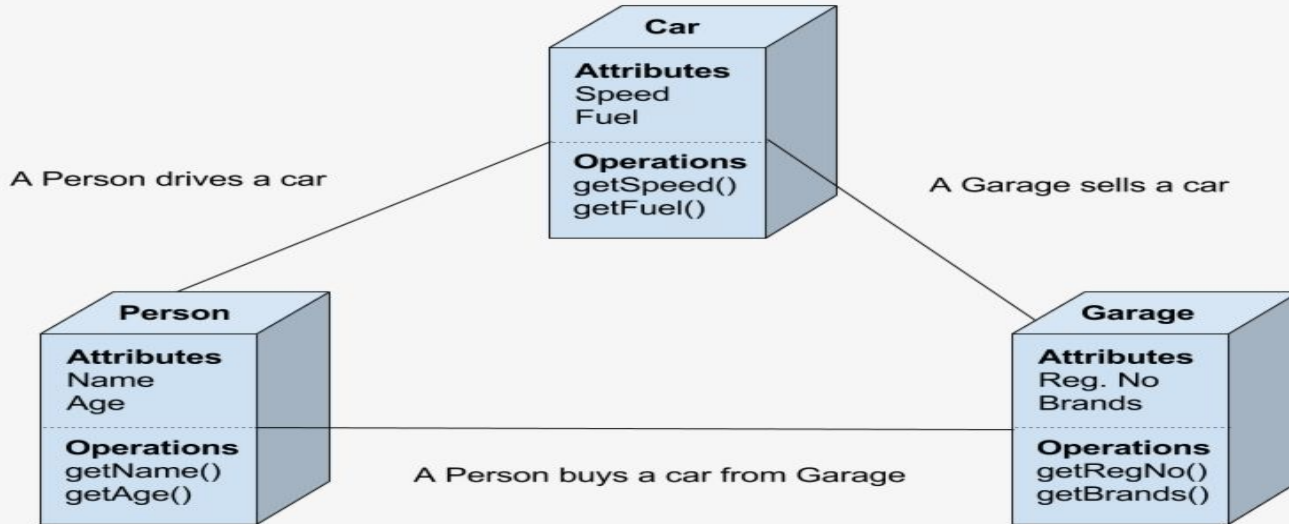
INTRODUCTION TO OOP

- ❖ How do we design object-oriented programming systems?
 - In most OOP languages, an object is created using a class. A class is a blueprint from which objects are made, consisting of both data and the code that manipulates the data.
- ❖ The image below illustrates the relationships between different objects in OOP.

INTRODUCTION TO OOP

Object Oriented Programming

Objects interact with each other by calling operations on other objects



INTRODUCTION TO OOP

- ❖ The image above shows that a **Person** interacts with a **Car** by driving it, and a **Person** buys a **Car** from a **Garage**, which sells cars. Each of these entities – **Person**, **Car**, and **Garage** – has specific characteristics (like a person's name or a car's speed) and actions (like driving or selling). The image demonstrates how these objects are connected and how they work together by exchanging actions and information.

JAVASCRIPT OBJECTS

- ❖ There is more than one way of creating objects.
 - Method 1: Using object literals
 - Method 2: Using object constructors

JAVASCRIPT OBJECTS

❖ Method 1: Using object literals

- Objects in JavaScript store a collection of key-value pairs called properties.
- When a function is declared within an object, it is known as a method. Methods are a series of actions that can be carried out on an object. An object can have properties that store different types of data, such as numbers, strings, functions, or even other objects. Therefore, an object method is simply a property that has a function as its value.

JAVASCRIPT OBJECTS

Object Literals

- Let's consider the following object declaration:

```
let car = {  
  // Object properties  
  brand: "Tesla",  
  model: "Model S",  
  year: 2023,  
  color: "Gray",  
  
  // Object methods  
  howOld() {  
    return `The car was made in the year ${this.year}`;  
  },  
};
```

JAVASCRIPT OBJECTS

❖ Method 2: Using object constructors

- This method provides an effective way of creating many objects using an object constructor.
- A constructor is a special type of function that is used to make or construct several different objects.
- Let's consider the example below:

JAVASCRIPT OBJECTS

Object Constructors

```
function carDescription(brand, model, year, color) {  
  // Object properties  
  this.brand = brand;  
  this.model = model;  
  this.year = year;  
  this.color = color;  
}  
  
// Creating three objects of the carDescription class  
let car1 = new carDescription("Tesla", "Model S", 2023, "Gray");  
let car2 = new carDescription("Audi", "e-tron", 2022, "Red");  
let car3 = new carDescription("Porsche", "Taycan", 2021, "Blue");
```

JAVASCRIPT OBJECTS

- ❖ Accessing object properties:
 - Dot notation - When using dot notation, we need to specify the name of the object followed by a dot and the key of the property you would like to access.
 - Example of dot notation:

JAVASCRIPT OBJECTS

```
function carDescription(brand, model, year, colour) {  
  // Object properties  
  this.brand = brand;  
  this.model = model;  
  this.year = year;  
  this.color = colour;  
}  
  
// Creating three objects of the carDescription class  
let car1 = new carDescription("Tesla", "Model S", 2023, "Gray");  
let car2 = new carDescription("Audi", "e-tron", 2022, "Red");  
let car3 = new carDescription("Porsche", "Taycan", 2021, "Blue");  
  
// Accessing the object properties using dot notation  
console.log(car1.brand); // Output: Tesla  
console.log(car2.model); // Output: e-tron  
console.log(car3.colour); // Output: Blue
```

JAVASCRIPT OBJECTS

- ❖ Accessing object properties:
 - Bracket notation - When using bracket notation, we have to make sure that the property name is passed in as a string within the brackets.
 - Example of bracket notation:

JAVASCRIPT OBJECTS

```
function carDescription(brand, model, year, color) {  
  // Object properties  
  this.brand = brand;  
  this.model = model;  
  this.year = year;  
  this.color = color;  
}  
  
// Creating three objects of the carDescription class  
let car1 = new carDescription("Tesla", "Model S", 2023, "Gray");  
let car2 = new carDescription("Audi", "e-tron", 2022, "Red");  
let car3 = new carDescription("Porsche", "Taycan", 2021, "Blue");  
  
// Accessing the object properties using square brackets notation.  
// The property name is passed as a string inside the square brackets.  
console.log(car1["brand"]); // Output: Tesla  
console.log(car2["model"]); // Output: e-tron  
console.log(car3["year"]); // Output: 2021
```

WORKING WITH OBJECTS IN JAVASCRIPT

- ❖ Javascript getters and setters - getter and setter methods act as interceptors because they offer a way to intercept property access and assignment.
 - Getters -These are methods that get and return the properties of an object.
 - Setters - These are methods that change the values of existing properties within an object.



Which statement is true about OOP?

- A. Organises code around real-world entities, simplifying complex problems and making systems intuitive.
- B. In OOP, instructions are executed sequentially, one after another.
- C. The key features of OOP are sequential execution and the separation of data from the functions that operate on it.
- D. All of the above.



Which statement is true about procedural programming?

- A. The data is stored separately from the code that processes it.
- B. The key features are sequential execution and the separation of data from the functions that operate on it.
- C. Functions are independent but can access global variables.
- D. All of the above.



**Objects in JavaScript store a collection of key-value pairs
called properties.
(True or False)**

- A. True
- B. False

Let's take a
break



Questions and Answers



Thank you for attending



CoGrammar




Department
for Education