Objects

HyperionDev

Muhammad Zahir Junejo

Lecture - Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
  - ❑ Please review Code of Conduct (in Student Undertaking Agreement) if unsure
- ❑ No question is daft or silly - **ask them!**
- ❑ Q&A session at the end of the lesson, should you wish to ask any follow-up questions.
- ❑ Should you have any questions after the lecture, please schedule a mentor session.
- ❑ For all non-academic questions, please submit a query: www.hyperiondev.com/support

Lecture Objectives

1. Pivotal role of objects in structuring data and modeling real-world entities.
2. Creating objects using object literals, constructor functions, and ES6 classes.
3. Defining instance variables and methods to encapsulate data and functionality.
4. Understanding JSON as a data interchange format and its serialization/deserialization.

Role of Objects

- ❑ Objects are fundamental data structures in JavaScript.
- ❑ They allow us to represent and organize data in a structured manner.
- ❑ Objects can model real-world entities, making them a crucial concept in programming.
- ❑ Example:

```javascript
// Object representing a car
const car = {
  make: "Toyota",
  model: "Camry",
  year: 2022
};
```

Creating Objects

- **Object Literals:**
  - Create objects using curly braces {}.
  - Define properties and their values within the braces.

```
        const person = {
  firstName: "John",
  lastName: "Doe",
  age: 30
};
```

Hyperiondev

Creating Objects

- **Constructor Functions:**
  - Create custom object types using constructor functions.
  - Define properties and methods using the this keyword.

```
function Person(firstName, lastName, age) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
}

const person = new Person("Jane", "Smith", 25);
```

Creating Objects

- **ES6 Classes:**
  - Create classes using the class keyword for cleaner syntax.
  - Define properties and methods within the class.

```
class Animal {
  constructor(name, species) {
    this.name = name;
    this.species = species;
  }
  greet() {
    console.log(`Hello, I'm ${this.name} the ${this.species}.`);
  }
}
const dog = new Animal("Buddy", "Dog");
dog.greet();
```

Hyperiondev

Instance Variables and Methods

- Instance Variables:
  - Properties that hold data unique to each object instance.
  - Set and access instance variables using dot notation.
  - Example:

```
console.log(person.firstName); // Output: John
```

Instance Variables and Methods

- ❑ Instance Methods:
  - ❑ Functions that operate on instance-specific data.
  - ❑ Access instance properties using the this keyword.

```
Animal.prototype.bark = function() {
  console.log(`${this.name} is barking.`);
};
dog.bark(); // Output: Buddy is barking.
```

Working with JSON

- ❑ JSON is a lightweight data interchange format.
- ❑ Example:

```
{
  "name": "Alice",
  "age": 28,
  "city": "New York"
}
```

Working with JSON

- ❑ Convert JavaScript objects to JSON string using JSON.stringify().
- ❑ Convert JSON strings to JavaScript objects using JSON.parse().
- ❑ Example:

```
const person = { name: "Bob", age: 35 };
const jsonString = JSON.stringify(person);
const parsedPerson = JSON.parse(jsonString);
```

Manipulating Objects

- Use dot notation or bracket notation to add properties.

```
person.gender = "Male"; // Adding using dot notation
person["occupation"] = "Engineer"; // Adding using bracket notation
```

- Access properties and update their values.

```
person.age = 31; // Modifying the age property
```

- Remove properties using the delete keyword.

```
delete person.city; // Deleting the city property
```

References

- ❑ https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_objects
- ❑ https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

Questions and Answers

Thank You!