Objects

Introduction to Objects

In JavaScript, an **object** is a collection of related data and functionality. These are represented as key-value pairs, where keys are property names, and values can be any data type, including other objects or functions.

Basic Syntax:

```
const objectName = {
  key1: value1,  // Property
  key2: value2,
  key3: function() { // Method
    // Function logic
  }
};
```

Example of an Object:

```
// REGULAR WAY OF CREATING OBJECTS

const person = {
  name: "John",
  age: 30,
  greet: function() {
    console.log(`Hi, my name is ${this.name}.`);
  }
};

const person2 = {
  name: "Andrea",
  age: 25,
  greet: function() {
```

```
    console.log(`Hi, my name is ${this.name}.`);
  }
};

const person3 = {
  name: "Jane",
  age: 15,
  greet: function() {
    console.log(`Hi, my name is ${this.name}.`);
  }
};

// CONSTRUCTOR

function Person(inputName, inputAge) {
    this.name = inputName;
    this.age = inputAge;
    this.greet = function() {
    console.log(`Hi, my name is ${this.name}.`);
  }
}

const person1 = new Person("John", 30);
const person2 = new Person("Andrea", 25);
const person3 = new Person("Jane", 15);


this.name // THIS WILL NOT WORK

// Accessing properties and methods
console.log(person.name); // Output: John
person.greet(); // Output: Hi, my name is John.
```

Constructors

A **constructor** is a special function used to create and initialize objects of a certain type. You can think of it as a blueprint for creating multiple objects with the same structure but different values.

Basic Syntax:

```
function ConstructorName(param1, param2) {
  this.key1 = param1;
  this.key2 = param2;
}
```

Example of a Constructor Function:

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
  this.displayInfo = function() {
    console.log(`This car is a ${this.year} ${this.make} ${th
is.model}.`);
  };
}

// Creating objects using the constructor
const car1 = new Car("Toyota", "Camry", 2020);
const car2 = new Car("Honda", "Civic", 2018);

// Accessing properties and methods
car1.displayInfo(); // Output: This car is a 2020 Toyota Camr
y.
car2.displayInfo(); // Output: This car is a 2018 Honda Civi
c.
```

Use Cases for Objects and Constructors

1. **Managing User Data:**
   Objects can represent users with properties like name, email, and role.

```
const user = {
  username: "jdoe",
  email: "jdoe@example.com",
  role: "admin"
};
```

2. **Creating Reusable Blueprints:**
   Constructors allow the creation of multiple instances of similar objects, e.g., cars, employees, or products.

```
function Employee(name, jobTitle, salary) {
  this.name = name;
  this.jobTitle = jobTitle;
  this.salary = salary;
}
```

3. **Encapsulating Logic:**
   Objects can include both data and behavior, such as methods to calculate or process information.

Cheat Sheet Summary

Concept	Syntax/Example
Object	`{ key: value, method: function() { ... } }`
Constructor	`function Name(param1, param2) { this.key1 = param1; this.key2 = param2; }`
Object Access	`objectName.key` or `objectName['key']`
Create Object	`new ConstructorName(param1, param2)`

Practice!

Exercise 1: Create a Simple Object

Prompt:

Create an object named `book` that represents a book. It should have the following properties:

1. `title` (e.g., "To Kill a Mockingbird")
2. `author` (e.g., "Harper Lee")
3. `pages` (e.g., 324)
4. A method `read()` that logs the message: `"Currently reading [title] by [author]"`.

Exercise 2: Build a Constructor Function

Prompt:

Create a constructor function named `Animal` that represents an animal. Each animal should have:

1. `species` (e.g., "Dog")
2. `name` (e.g., "Buddy")
3. `sound` (e.g., "Woof!")
4. A method `makeSound()` that logs the message: `"[name] says [sound]"`.

Create two animals and call their `makeSound()` method.

Exercise 3: Expand the Blueprint

Prompt:

Create a constructor function named `Student` that represents a student. Each student should have:

1. `name` (e.g., "Alice")
2. `grade` (e.g., 10)
3. `subject` (e.g., "Mathematics")
4. A method `introduce()` that logs the message: `"Hi, I am [name], studying [subject] in grade [grade]."`
5. A method `finishesSchoolYear()` that:
    - Adds 1 to the grade (e.g., from 10 to 11).
    - If the grade reaches 12, it prints: `"Congratulations, [name] has graduated!"` and does not increase the grade.

Create three students and have them introduce themselves and call `finishesSchoolYear()`.

Summary of Features

1. **Exercise 1:** Focuses on creating static objects and practicing basic property and method usage.
2. **Exercise 2:** Introduces constructor functions and dynamic creation of objects.
3. **Exercise 3:** Builds on constructors, adds multiple methods, and implements conditional logic to handle graduation.