Lesson:

Object Constructor

PW SKILLS

Topics to be covered-

- Introduction to Object Constructor
- Creating object constructor
- Understanding this keyword in the object constructor
- Properties and Methods
- Creating instances of objects
- Accessing object properties and method

Introduction to Object Constructor

An object constructor is a function that serves as a template for creating new objects of a particular type. It saves us from creating individual objects manually. Object constructor is a reusable way to create multiple objects with similar properties. These objects can be instantiated with unique data.

Created an Object Constructor

For creating an object constructor, we can define a function that will act as a constructor. This function will be the template by which we can create instances of it.
By convention, the constructor function's name is capitalized to differentiate it from the regular functions, it is not mandatory but just a good practice to follow. Within the constructor function, we can define properties and methods that will be shared by all instances created using this constructor.

Example for creating an object constructor

```
// Object constructor for creating Person objects
function Person(name, age) {
    this.name = name;
    this.age = age;

    this.sayHello = function() {
        console.log(`Hello, my name is ${this.name} and
I am ${this.age} years old.`);
    };
}
```

In the above example, Person is the object constructor, in which name, and age are the property, and sayHello is the method which is used to log the name and age in a properly formatted way.

Understanding this keyword in the object constructor

The ‘this’ keyword in JavaScript refers to the current instance of an object. In the context of an object constructor, this is used to assign values to the properties of the newly created object. It ensures that each instance gets its own set of properties.

Properties and Methods

Properties are the characteristics that define an object, while methods are functions that define an object's behavior. In the object constructor, you can define both properties and methods that will be present in every instance created using that constructor.
In the above-given example, name, and age were the property as it was defining the characteristics of the Person while the sayHello was a method which was a function used to log the name and age.

Creating instances of objects

For creating an instance of the object, we use the ‘new’ keyword of the javascript which was introduced in the ES6 version of the javascript followed by the function’s name and the required arguments.
This process allocates the memory for a new object and sets up ‘this’ keyword reference within the constructor function.

```
// Object constructor for creating Person objects
function Person(name, age) {
  this.name = name;
  this.age = age;
  this.sayHello = function () {
    console.log(
      `Hello, my name is ${this.name} and I am
${this.age} years old.`
    );
  };
}

// Creating instances using the constructor
const person1 = new Person("PW", 5);
const person2 = new Person("PW Skills", 2);
console.log(person1);
console.log(person2);
```

```
// Output
// Person { name: 'PW', age: 5, sayHello: [Function
(anonymous)] }
// Person { name: 'PW Skills', age: 2, sayHello:
[Function (anonymous)] }
```

In the above example, we created two instances Person1 and Person2 from the constructor and passed the required arguments name and age to it. Also, we can see in the output that these two instances have their own passed values in it as an object.

Accessing object properties and method

We can access the properties and method of an object instance by using the dot notation to access the properties and invoke methods. It is similar to what we were using to access the methods and properties of an object.

```
// Object constructor for creating Person objects
function Person(name, age) {
  this.name = name;
  this.age = age;

  this.sayHello = function () {
    console.log(
      `Hello, my name is ${this.name} and I am
${this.age} years old.`
    );
  };
}
// Creating instances using the constructor
const person1 = new Person("PW", 5);
const person2 = new Person("PW Skills", 2);

console.log(person1.name, person1.age);
console.log(person2.name, person2.age);
person1.sayHello();
person2.sayHello();

// Output
// PW 5
// PW Skills 2
// Hello, my name is PW and I am 5 years old.
// Hello, my name is PW Skills and I am 2 years old.
```

In the above example, we used the dot notation to access the properties and to invoke the methods of the instance.

Adding or Removing the properties and method from an instances

We can add or remove any properties or methods from the object constructor instance.

Adding a property and method to the instance

For adding any new method or property we can use the dot notation and directly assign the value to it.

```
// Object constructor for creating Person objects
function Person(name, age) {
  this.name = name;
  this.age = age;
  this.sayHello = function () {
    console.log(
      `Hello, my name is ${this.name} and I am
${this.age} years old.`
    );
  };
}

// Creating instances using the constructor
const person = new Person("PW", 5);

// adding the new property and method
person.role = "admin";
person.getRole = function () {
  console.log("I am a " + this.role);
};
console.log(person);

// Output
// Person {
//   name: 'PW',
//   age: 5,
//   sayHello: [Function (anonymous)],
//   role: 'admin',
//   getRole: [Function (anonymous)]
// }
```

Removing a property and method to the instance

For deleting a property or method from our instance, we can use the delete keyword as demonstrated in the below example.

```
// Object constructor for creating Person objects
function Person(name, age) {
  this.name = name;
  this.age = age;
  this.sayHello = function () {
    console.log(
      `Hello, my name is ${this.name} and I am
${this.age} years old.`
    );
  };
}

// Creating instances using the constructor
const person = new Person("PW", 5);
// adding the new property and method
person.role = "admin";
person.getRole = function () {
  console.log("I am a " + this.role);
};
console.log(person);

// removing the property and method
delete person.sayHello;
delete person.name;
console.log(person);

// Output
// Person {
//   name: 'PW',
//   age: 5,
//   sayHello: [Function (anonymous)],
//   role: 'admin',
//   getRole: [Function (anonymous)]
// }
// Person { age: 5, role: 'admin', getRole: [Function
(anonymous)] }
```

Note: We will learn more about adding methods to an object constructor after its declaration as per the requirement in the constructor, in the further chapter of prototype in javascript.