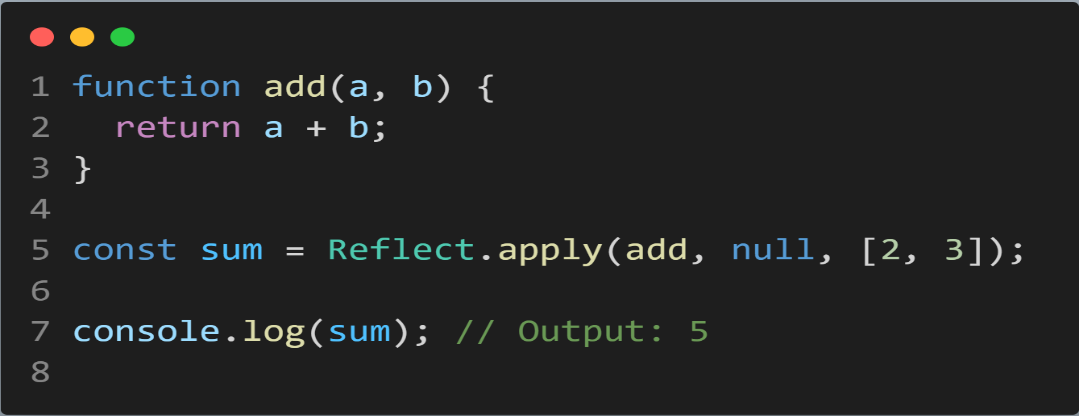
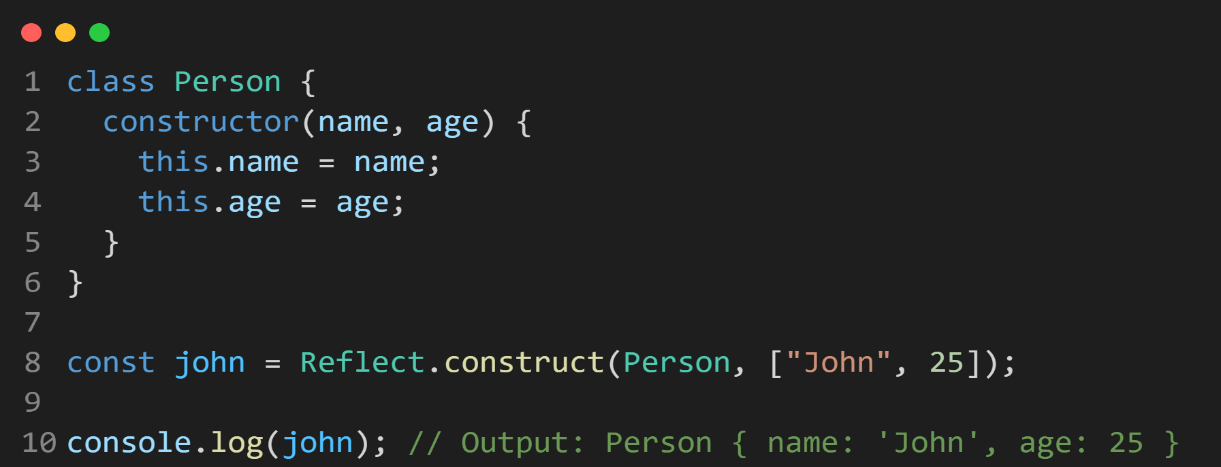
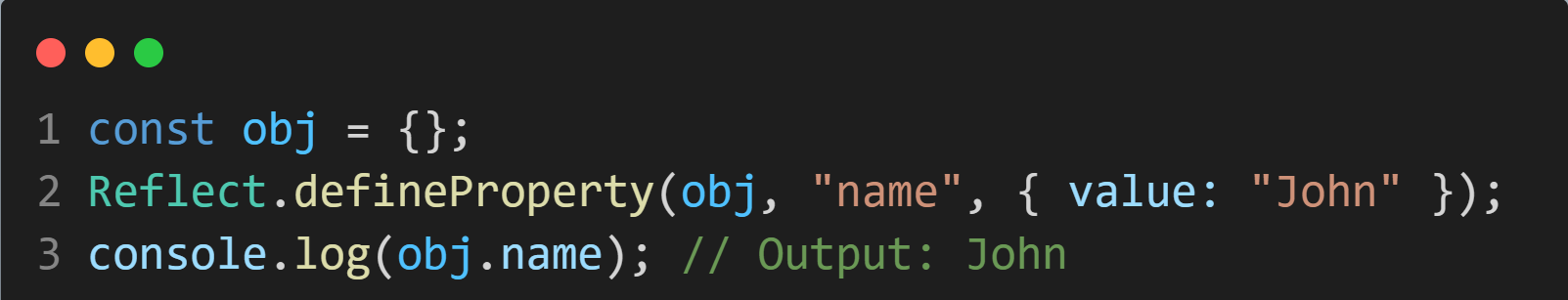
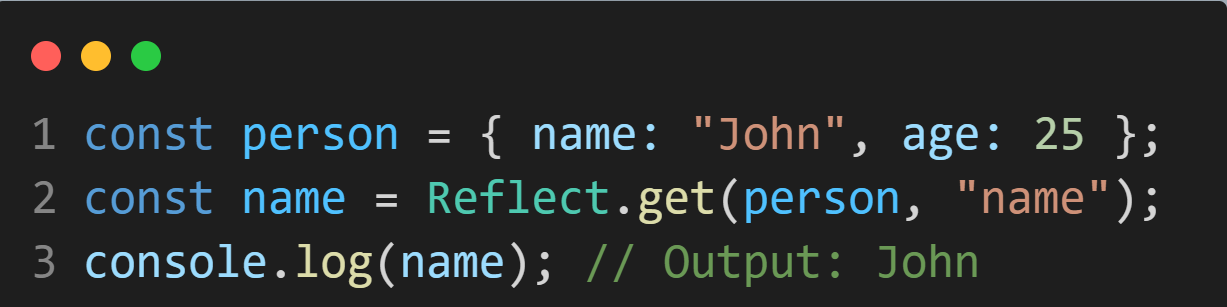
* **Introduction**:
* The **Reflect API** is a built-in object in JavaScript that **provides methods for intercepting JavaScript operations**. It is a **part of ECMAScript 6** (ES6) and can be **used to manipulate object**-oriented programming constructs such as classes, properties, and methods.
* **Some of the key benefits of using the Reflect API include**:
* **It provides** a consistent and reliable way to perform common object operations.
* **It enables** you to access and manipulate properties and methods of objects.
* **It makes** it easier to implement proxy objects, which can be used to provide additional functionality or security.
* **In this report**, we will explore **some of the most useful methods** of the **Reflect API** with examples.
* **Reflect.apply()**
* The **Reflect.apply()** method calls a function with a given this value and arguments. The syntax of the method is as follows:
* **Reflect.apply(target, thisArg, argumentsList)**
* Here, **target** is the function to be called, **thisArg** is the value of this to be used by the function, and **argumentsList** is an array-like object containing the arguments to be passed to the function.
* **Example**:
* **In the above example**, we have defined a simple function that takes two arguments and returns their sum. We then use the **Reflect.apply()** method to call this function with the arguments [2, 3]. Since we have passed **null** as the value of **thisArg**, the function will use the global **this** value.
* **Reflect.construct()**
* The **Reflect.construct()** method creates a new instance of a class with a given set of arguments. The syntax of the method is as follows:
* **Reflect.construct(target, argumentsList[, newTarget])**
* Here, **target** is the constructor function of the class to be instantiated, **argumentsList** is an array-like object containing the arguments to be passed to the constructor, and **newTarget** is an optional argument that specifies the constructor function to be used for creating the new instance.
* **Example**:
* **In the above example**, we have defined a class called Person with a constructor that takes two arguments (name and age). We then use the **Reflect.construct()** method to create a new instance of the class with the arguments ['John', 25].
* **Reflect.defineProperty()**
* The Reflect.defineProperty() method defines a new property on an object or modifies an existing one. The syntax of the method is as follows:
* **Reflect.defineProperty(target, propertyKey, attributes)**
* Here, **target** is the object on which the property is to be defined or modified, **propertyKey** is the name of the property, and attributes is an object that contains the attributes of the property.
* **Example**:
* **In the above example**, we have defined an empty object called **obj**. We then use the **Reflect.defineProperty()** method to define a new property called name with the value 'John'.
* **Reflect.get()**
* The **Reflect.get()** method retrieves the value of a property from an object. The syntax of the method is as follows:
* **Reflect.get(target, propertyKey[, receiver])**
* Here, **target** is the object from which the property is to be retrieved, **propertyKey** is the name of the property, and **receiver** is an optional argument that specifies the value of this within the getter function.
* **Example**:
* **In the above example**, we have defined an object called person with two properties (name and age). We then use the **Reflect.get()** method to retrieve the value of the name property.
* **Overall**, the Reflect API **provides a set of methods that can be used to manipulate JavaScript objects and classes** in a flexible and reliable way. **The methods we have discussed in this report are just a few examples** of the many methods available in the Reflect API.