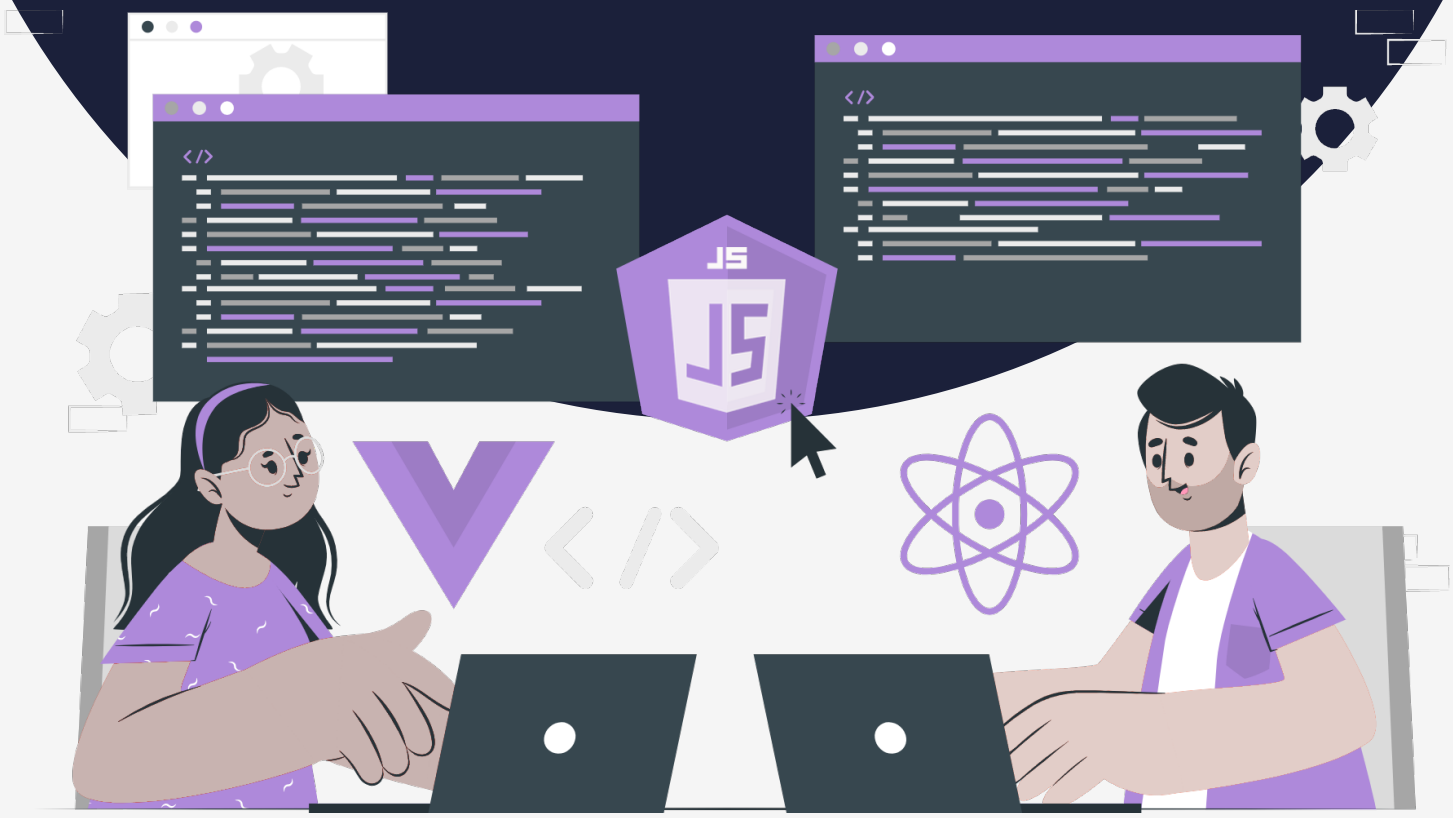
Lesson:

Getters and Setters



List of Content:

Getters

In JavaScript, a getter is a special type of function that is used to get/retrieve the value of an object property. It allows you to define a method to be executed when a specified property is accessed.

To define a getter, you can use the get keyword followed by the property name, and then add a function that returns the desired value. Here's an example:

```
const person = {
    firstName: 'Anuj',
    lastName: 'Kumar',
    get fullName() {
      return `${this.firstName} ${this.lastName}`;
   }
  };
    console.log(person.fullName);
```

In this example, the fullName property is defined as a getter. When person.fullName is accessed, the getter function is executed and the result is returned.

Getters can be useful for calculating or formatting a property value on the fly, without needing to store it as a separate property. They can also be used to provide read-only access to an object property since you cannot directly set the value of a property that only has a getter.

It's important to note that getters should not have any side effects, since they may be called multiple times and should not modify the object they are defined on.

Setters

In JavaScript, a setter is a special type of function that is used to set the value of an object property. It allows you to define a method to be executed when a specified property is assigned a new value.

To define a setter, you can use the set keyword followed by the property name, and then add a function that takes a single parameter representing the new value. Here's an example:

```
const person = {
    firstName: 'Anuj',
    lastName: 'Kumar',
    set fullName(name) {
      const parts = name.split(' ');
      this.firstName = parts[0];
      this.lastName = parts[1];
    }
  };
    person.fullName = 'Rohan Sharma';
    console.log(person.firstName);
    console.log(person.lastName);
```

In this example, the fullName property is defined as a setter. When person.fullName is assigned a new value, the setter function is executed with the new value as its parameter, and it updates the firstName and lastName properties accordingly.

Setters can be useful for enforcing constraints or validating input before setting a property value. They can also be used to provide write-only access to an object property, since you cannot directly read the value of a property that only has a setter.

It's important to note that setters should not have any effects beyond setting the value of the corresponding property. They should not return any values or perform any other operations beyond updating the object state.