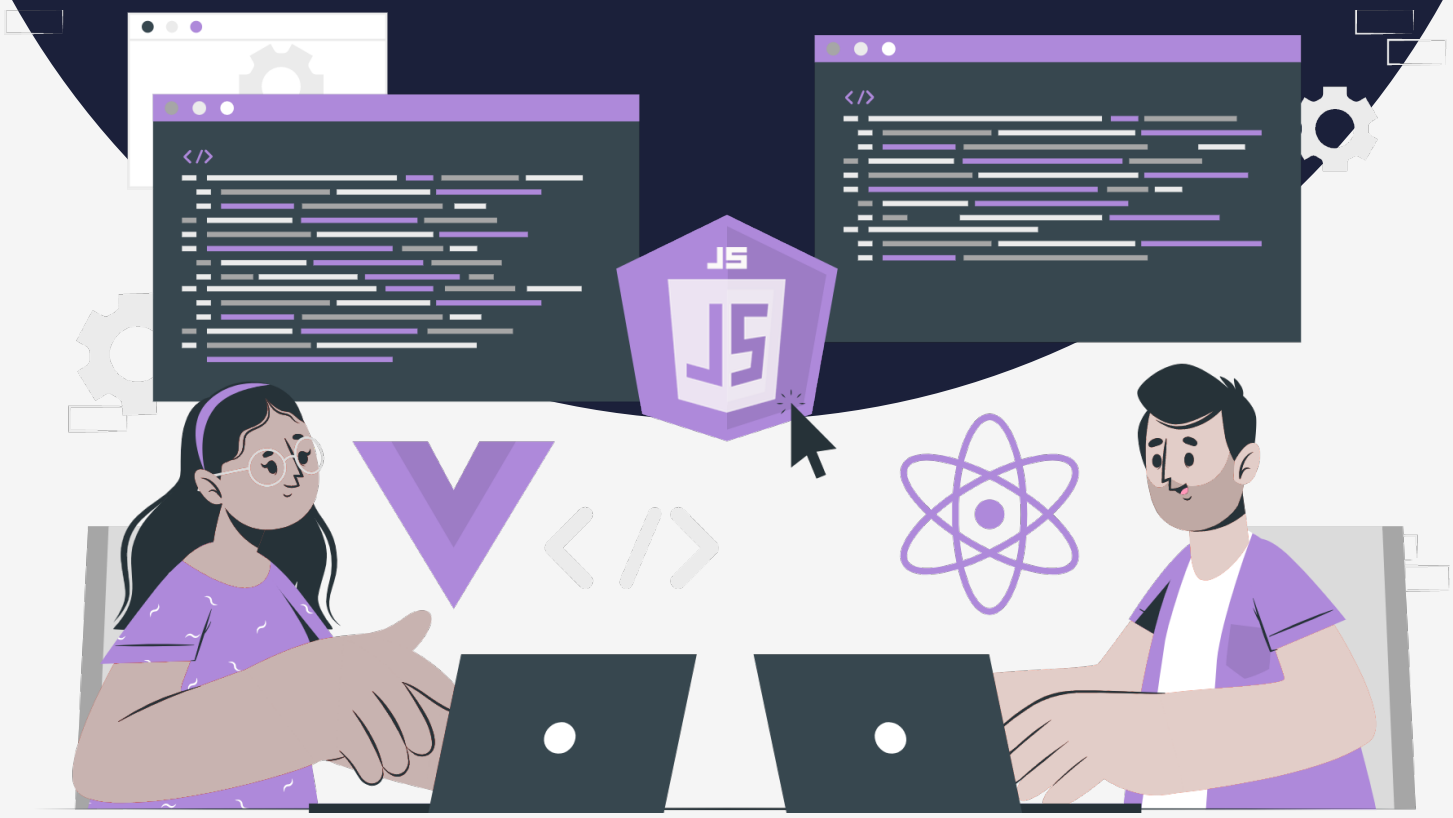
Lesson:

Explaining Prototypes



PW SKILLS

List of Content:

What is Prototype?

A prototype is an object that contains properties and methods that are shared among all instances of a particular object. Every JavaScript object has a prototype property.

How to add a method and property to the prototype?

Let's first see the constructor function: A constructor function is a simple javascript function used to create objects using the **"new"** keyword. So instead of calling the function directly, we have to invoke it with the **"new"** keyword.

Here's what happens when we invoke a function with the new keyword:

1. A new object is created.
2. "**this**" keyword inside the function refers to the newly created object.
3. The new object is linked to a prototype object.
4. The function body is executed, which can modify the newly created object by adding properties or methods to it.
5. The newly created object is automatically returned from the function, unless the function returns an object explicitly.

For example, let's say we have a constructor function Person that takes two arguments **name** and **age**, and we use the **new** keyword to create a new instance of the **Person** object:

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

const john = new Person('John', 30);
```

Here, **new Person('John', 30)** creates a new object that is an instance of the **Person** constructor function. The "**this**" keyword inside the constructor function refers to the new object being created, and the **name** and **age** properties are added to the object.

The **new** keyword also returns the newly created object john, which can be stored in a variable or used directly.

Now let's create a new constructor function and add property and method to the prototype.

```
function Person(name) {
  this.name = name;
}

// adding a method to the Person prototype
Person.prototype.sayHello = function() {
  console.log(`Hello, my name is ${this.name}.`);
}
console.log(studentTwo.age) // 15
// adding a property to the Person prototype
Person.prototype.age = 0;

var person1 = new Person("John");
person1.sayHello(); // "Hello, my name is John."
console.log(person1.age); // 0
```

Here we define a constructor function called **Person** that takes a **name** parameter and assigns it to the **name** property of the new object created by the constructor.

Then, two properties are added to the prototype object of the **Person** function. The first one is a method called **sayHello**, which logs a message to the console with the person's name. The second one is a property called **age**, which is initialized to zero.

When a new instance of **Person** is created using the **new** keyword and assigned to the **person1** variable, it will have access to both the **sayHello** method and the **age** property through the prototype object of the **Person** function.

Finally, the **sayHello** method is called on the **person1** instance, which logs a greeting message to the console with the person's name. And the **age** property is accessed and logged to the console, which outputs 0 since it was initialized to that value on the prototype.

we can also add methods and properties to an existing object's prototype, including the built-in Array and String objects. Here's an example of adding a method to the Array and string prototype:

```
String.prototype.strLength = function(){
return this.length;
Array.prototype.arrlength = function(){
  return this.length
}

console.log("Jhon".strLength()) // 4
console.log([1,2,3,4].arrlength()) // 4
```

When to use the prototype in JavaScript?

Here are some common use cases for prototypes in JavaScript:

Adding properties and methods: You can use prototypes to add new properties and methods to existing objects. This is often done by modifying the prototype of the object.

Creating reusable objects: Prototypes are a great way to create objects that can be reused throughout your code. By defining common properties and methods on a prototype object, you can easily create new instances of the object without having to repeat the same code over and over again.

Inheritance: Prototypes are used to implement inheritance in JavaScript. When you define a new object, you can set its prototype to another object, and the new object will inherit all the properties and methods of the prototype.

Performance optimization: Prototypes can be used to optimize the performance of JavaScript code. For example, by using prototypes, you can avoid creating multiple copies of the same function or object.

Key Takeaways

- A prototype is an object which associates with every functions and objects. Additionally, it is invisible, but all the properties inside the prototype are accessible.
- When a programmer needs to add new properties like variables and methods at a later point in time, and these properties need sharing across all the instances, then the prototype will be very handy.
- The prototype can add both variables and methods to an existing object dynamically.