

Lesson:

**Class methods and
Properties with initial
value**

List of Content:

1. Class Method
2. Static Methods
3. Properties with initial values

In JavaScript, class methods are functions that belong to a class and can be called on instances of that class. They are defined within a class using the "method" syntax.

Methods are defined on the prototype of each class instance and are shared by all instances. We will learn about prototypes in detail as we proceed with the lectures.

We always add a constructor() method before adding any number of methods.

Syntax:

```
class ClassName {  
  constructor() { ... }  
  method1() { ... }  
  method2() { ... }  
  .  
  .  
  method() {.....}  
}
```

Let us look at this example to understand the syntax and use.

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
  sayHello() {  
    console.log(`Hello, my name is ${this.name}.`);  
  }  
}
```

```
const person = new Person("Rohan");  
person.sayHello();
```

Class methods can access properties of the class instances through the 'this' keyword, and can also access other methods defined on the class using 'this'.

Static methods

Static class methods are defined on the class itself. They can be called without creating an instance of the class and are typically used to implement utility functions that operate on class-level data.

You cannot call a static method on an object but only on an object class.

Static methods are often utility functions, such as functions to create or clone objects.

Example :

```
class MyClass {
  static myStaticMethod() {
    console.log('This is a static method');
  }
}

MyClass.myStaticMethod();
```

Static methods are declared using the static keyword and are called using the class name followed by the method name.

Properties with initial values

For some properties, we might have an initial value when we build a class.

Some properties may have an initial value when we build a class for them. For instance, if you are learning a programming language, your level would be of a beginner which also means level 0. This means that we might start off with one skill increase our level in it and pick up more after some time.

```
class Person {
  constructor(fName, lName, age, city) {
    this.fName = fName
    this.lName = lName
    this.age = age
    this.city = city
    this.level = 0
    this.skills = []
  }
  getFullName() {
    const fullName = this.fName + ' ' + this.lName
    return fullName
  }
}

const person1 = new Person('Anuj', 'Kumar', 31 , 'Delhi')
const person2 = new Person('Rohan', 'Sharma', 28 , 'Jaipur')

console.log(person1.level)
console.log(person2.level)
console.log(person1.skills)
console.log(person2.skills)
```

1. Person Class:

The Person class is defined with a constructor function that takes four parameters: fName, lName, age, and city. The constructor function sets the properties of the new object created by the class instance to the values of these parameters. Additionally, two properties are also set with default values of 0 and an empty array respectively. The getFullName() method is also defined on the Person class, which concatenates the fName and lName properties to return the full name of the person.

2. Creating instances of the Person class:

Two instances of the Person class are created using the new keyword with different values for the fName, lName, age, and city parameters.

3. Accessing properties of the instances:

The console.log() statements output the value of the level and skills properties of person1 and person2 instances respectively. As the level property has a default value of 0 and skills is an empty array, both console.log() statements output 0 and [] respectively.