Static Methods

This lesson explains what static methods are and how they are implemented using an example.

WE'LL COVER THE FOLLOWING

- What are Static Methods?
- Defining Static Methods
  - Example
    - Explanation

What are Static Methods?

Methods that we define inside the class get assigned to the prototype object of the class and belong to all the objects instances that get created from that class.

Let's consider the example of the class `Student`:

```
class Student {
  constructor(name,age,sex,marks) {
    this.name = name
    this.age = age
    this.sex = sex
    this.marks = marks
  }
  //method defined in class gets assigned to the prototype of the class Students
  displayName(){
    console.log("Name is:", this.name)
  }
}
var student1 = new Student('Kate',15,'F',20)
student1.displayName()
console.log("Age:",student1.age)
console.log("Sex:",student1.sex)
console.log("Marks:",student1.marks)
```

The class `Student` has the method `displayName` defined. Any object instance created will inherit this method from `Student.prototype` just as `student1` inherits it.

Now, let's suppose we have multiple `Student` objects, and we need a *method* to return their count, i.e., the total number of students. Such a method, when defined inside the `Student`, should only belong to the *class* instead of a particular object instance as it gives the total count of students.

Methods that are used to implement functions belonging only to a class and not to any particular object of it are known as **static methods**.

> **Note:** Static functions can only be accessed by the class itself since they do not belong to its objects.

Defining Static Methods

As discussed above, a static method belongs to the class only and not to the objects created from it. So how is it defined?

The ES6 version of JavaScript introduces the *keyword* `static` which assigns a method directly to the class instead of the class `prototype` object, from which the objects would be able to inherit it.

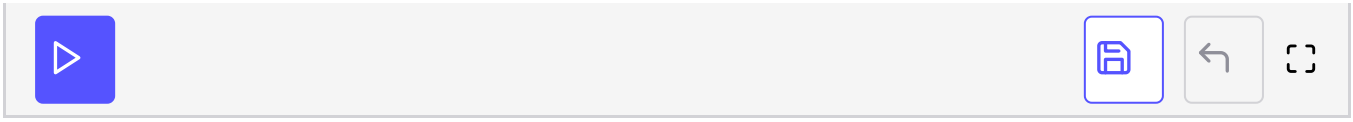
> **Note:** Since `static` methods are not defined on the prototype of class, they cannot be accessed; hence, they cannot be overridden by the child object inheriting it.

Example

Let's take a look at an example defining a static method:

```
class Student {
  constructor(name,age,sex,marks) {
    this.name = name
    this.age = age
    this.sex = sex
    this.marks = marks
  }
  //defining a static method called "compareMarks"
  //it compares marks of two students
```

```
  static compareMarks(student1, student2){
    if (student1.marks > student2.marks){
      temp1 = student1.marks - student2.marks
      console.log(`${student1.name} scored ${temp1} marks more than ${student2.name}`)
    }
    else if(student1.marks < student2.marks){
      temp1 = student2.marks - student1.marks
      console.log(`${student2.name} scored ${temp1} marks more than ${student1.name}`)
    }
    else{
      console.log(`Both students scored ${student1.marks}`)
    }
  }
}
//change values of marks of both students to execute different if conditions in the function
var student1 = new Student('Kate',15,'F',25)
console.log("Kate's marks are",student1.marks)
var student2 = new Student('Joe',15,'M',20)
console.log("Joe's marks are",student2.marks)
//calling the static function
Student.compareMarks(student1,student2)
```



Explanation

In the example above:

- a static method, `compareMarks`, is declared in **line 10**.
- It takes *two* objects, `student1` and `student2` as its parameters.
- The function compares the `marks` of the two students.
- For the student who scored more, the function displays the *difference* by which that student scored more than the other.
- If both the students score equal marks the function simply displays the score instead of the difference.

A static method is always called from the class itself, i.e., the *class name* followed by the *dot operator* then the *static method's* name just as shown in **line 30**.

Now that all the concepts regarding classes in the ES6 version of JavaScript have been discussed, it's time to test those concepts with a quiz!