ES6 Classes & Inheritance

→ What are the ES6 Classes?

→ ES6 classes is the common JavaScript pattern of getting the class-like inheritance hierarchies using functions and prototypes. They are effectively simple compared to prototype-based OO, offering a convenient declarative form for class patterns that encourage interoperability.

Ex:

```
CLASS SIMPLEDATE {
    CONSTRUCTOR(YEAR, MONTH, DAY) {
        THIS.YEAR = YEAR;
        THIS.MONTH = MONTH;
        THIS.DAY = DAY;
    }
    ADDMONTH(NMONTHS) {
    }
    GETMONTH() {
        RETURN THIS.MONTH;
    }
}
```

→ Constructors:

→ The constructor method's purpose is to initialize an instance to a valid state, and it will be called automatically, so

We cannot forget to initialize our objects. The constructor function is an object blueprint.

→ Defining Methods:

→ The common practise is to assign methods directly to the prototype instead of in the initialization. But, with classes, this syntax is simplified, and we can add the method to the class, Using the method definition introduced in ES6 JavaScript, defining a method is an even easy and concise process.

```
CLASS CAR {
    CONSTRUCTOR(NAME, YEAR) {
        THIS.NAME = NAME;
        THIS.YEAR = YEAR;
    }
    GREET() {
        RETURN `${THIS.NAME} SAYS HELLO`;
}}
```

→ We will create a new instance of Car using the new keyword and assign some values.

```
CONST CAR1 = NEW CAR('AUDI', 2018);
```

→ Inheritance:

Page No.
Date

→ Inheritance is one of the most important concepts of object-oriented programming. The significant advantage of using ES6 classes over pre-ES6 functions is that class definitions are cleaner and easy to inherit. Without writing prototype code for an ~~instan~~ inheritance, class inheritance in JavaScript is easier and similar to other object-oriented language like Java. Following is the way of applying the inheritance concept in JavaScript:

→ Extending a Class:

→ The most useful feature of constructor functions and classes is that they can be ~~extendended~~ extended into new object blueprints based on the parent. Before ES6, the new constructor functions can be created from the parent using the call() method. However, with the help of ES6 classes, the super keyword is used in place of the call to access the parent functions. We will use the extends keyword to refer to the parent class.

```
// new class from parent car.
CLASS BIKE EXTENDS CAR {
    CONSTRUCTOR(NAME, YEAR, SPEED) {
        // constructor with super
```

```
    super(name, year);
    // Adding new property
    this.speed = speed;
  }
}
```

→ Now we create a new Bike instance in the same manner.

```
const bike1 = new Bike('TREK', 2019, 200);
```