JavaScript Inheritance

The JavaScript inheritance is a mechanism that allows us to create new classes on the basis of already existing classes. It provides flexibility to the child class to reuse the methods and variables of a parent class.

The JavaScript **extends** keyword is used to create a child class on the basis of a parent class. It facilitates child class to acquire all the properties and behavior of its parent class.

Points to remember

- It maintains an IS-A relationship.
- The extends keyword is used in class expressions or class declarations.
- Using extends keyword, we can acquire all the properties and behavior of the inbuilt object as well as custom classes.
- We can also use a prototype-based approach to achieve inheritance.

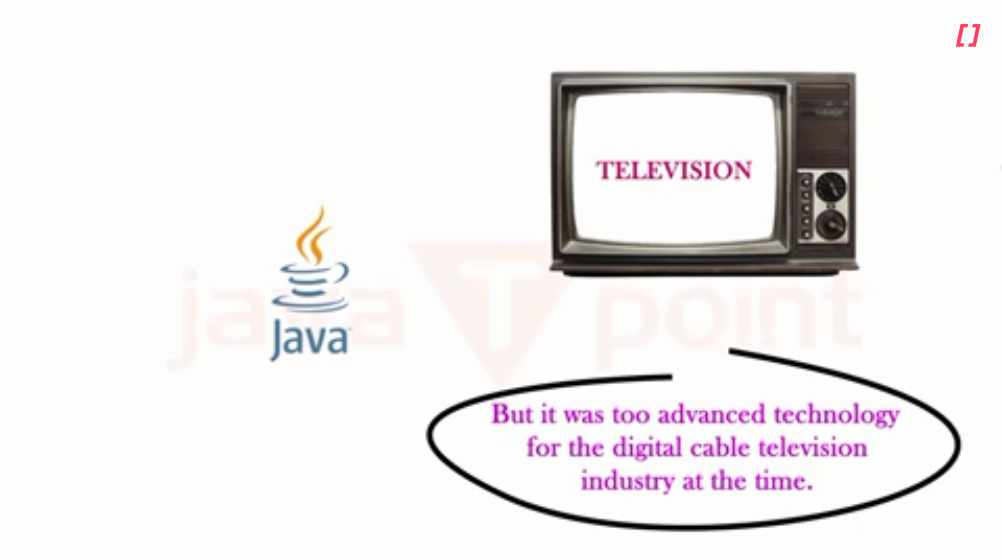
JavaScript extends Example: inbuilt object

In this example, we extends **Date** object to display today's date.

```
<script>
class Moment extends Date {
  constructor() {
    super();
  }}
var m=new Moment();
document.writeln("Current date:")
document.writeln(m.getDate()+"-"+(m.getMonth()+1)+"-"+m.getFullYear());
</script>
```

Test it Now

Output:




```
Current date: 31-8-2018
```

Let's see one more example to display the year value from the given date.

```
<script>
class Moment extends Date {
  constructor(year) {
    super(year);
  }}
var m=new Moment("August 15, 1947 20:22:10");
document.writeln("Year value:")
document.writeln(m.getFullYear());
</script>
```

Test it Now

Output:

```
Year value: 1947
```

JavaScript extends Example: Custom class

In this example, we declare sub-class that extends the properties of its parent class.

```
<script>
class Bike
```

⇧ SCROLL TO TOP

```
  {
    this.company="Honda";
  }
}
class Vehicle extends Bike {
  constructor(name,price) {
   super();
    this.name=name;
    this.price=price;
  }
}
var v = new Vehicle("Shine","70000");
document.writeln(v.company+" "+v.name+" "+v.price);
</script>
```

Test it Now

Output:

```
Honda Shine 70000
```

JavaScript extends Example: a Prototype-based approach

Here, we perform prototype-based inheritance. In this approach, there is no need to use class and extends keywords.

```
<script>
//Constructor function
function Bike(company)
{
    this.company=company;
}

Bike.prototype.getCompany=function()
{
  return this.company;
}
//Another constructor function
```

⇧ SCROLL TO TOP me,price) {

```
 this.name=name;
 this.price=price;
 }
var bike = new Bike("Honda");
Vehicle.prototype=bike; //Now Bike treats as a parent of Vehicle.
var vehicle=new Vehicle("Shine",70000);
document.writeln(vehicle.getCompany()+" "+vehicle.name+" "+vehicle.price);
</script>
```

Test it Now

Output:

```
Honda Shine 70000
```

← Prev Next →