JavaScript Polymorphism

The polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects. As JavaScript is not a type-safe language, we can pass any type of data members with the methods.

JavaScript Polymorphism Example 1

Let's see an example where a child class object invokes the parent class method.

```
<script>
class A
 {
    display()
  {
    document.writeln("A is invoked");
  }
 }
class B extends A
 {
 }
var b=new B();
b.display();
</script>
```

Test it Now

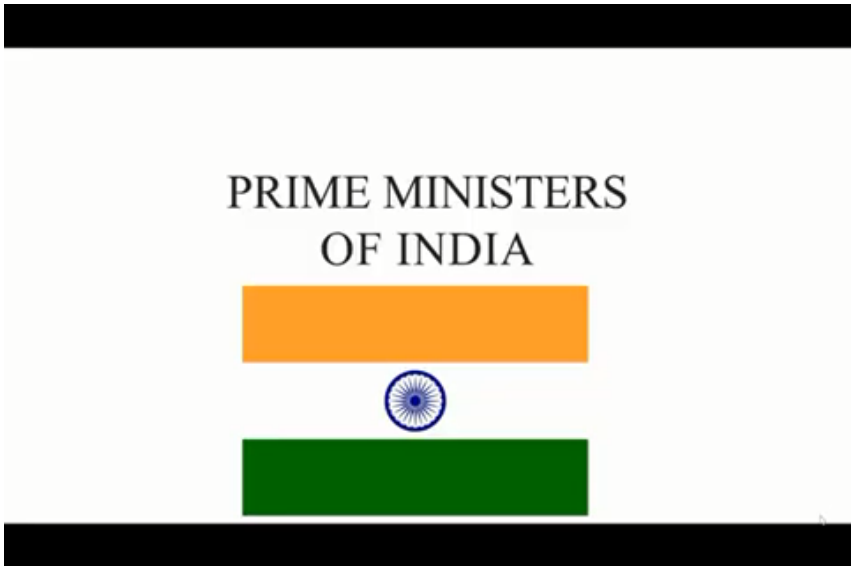
Output:

```
A is invoked
```

Example 2

Let's see an example where a child and parent class contains the same method. Here, the object of child class invokes both classes method.

⇧ SCROLL TO TOP



[]

```
<script>
class A
 {
    display()
  {
    document.writeln("A is invoked<br>");
  }
 }
class B extends A
 {
  display()
  {
    document.writeln("B is invoked");
  }
 }

var a=[new A(), new B()]
a.forEach(function(msg)
{
msg.display();
});
</script>
```

Test it Now

Output:

⇧ SCROLL TO TOP

```
A is invoked
B is invoked
```

Example 3

Let's see the same example with prototype-based approach.

```
<script>
function A()
{
}
A.prototype.display=function()
{
  return "A is invoked";
}
function B()
{

}

B.prototype=Object.create(A.prototype);

var a=[new A(), new B()]

a.forEach(function(msg)
{
  document.writeln(msg.display()+"<br>");
});
<script>
```

Test it Now

Output:

```
A is invoked
B is invoked
```

⇧ SCROLL TO TOP