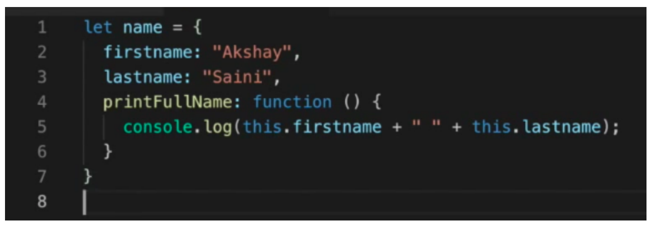
call, apply and bind methods

Thursday, 24 August 2023 12:33 AM

→ Every f^n in JS has access to this keyword.

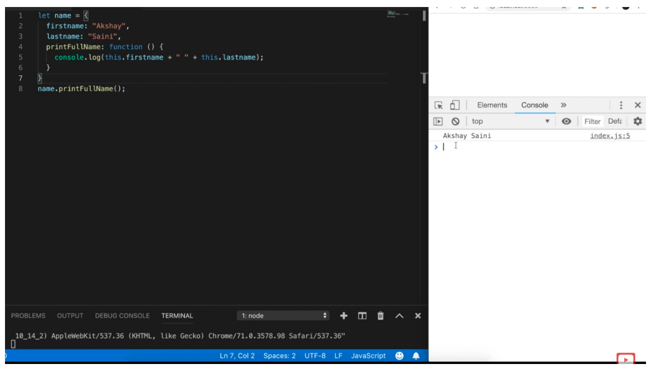


```javascript
let name = {
  firstname: "Akshay",
  lastname: "Saini",
  printFullName: function () {
    console.log(this.firstname + " " + this.lastname);
  }
}
```

→ So this f^n has access to this keyword which is here pointing to 'name' object.

→ So o/p of above will be Akshay Saini.

→ How to invoke the above f^n?

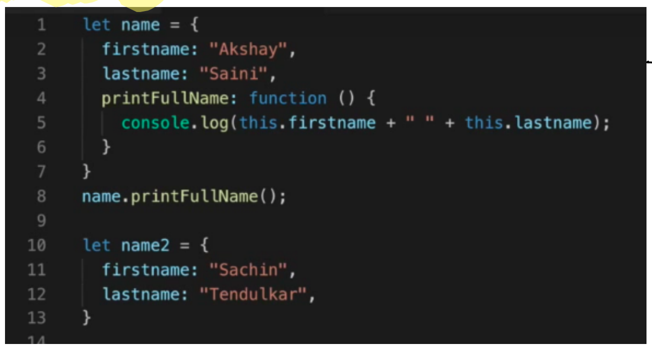


```javascript
let name = {
  firstname: "Akshay",
  lastname: "Saini",
  printFullName: function () {
    console.log(this.firstname + " " + this.lastname);
  }
}
name.printFullName();
```

```
Akshay Saini                                   index.js:5
>
```

```
10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36"
```

→ Call():

→ 

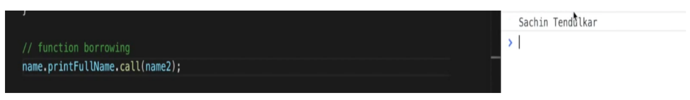
```javascript
let name = {
  firstname: "Akshay",
  lastname: "Saini",
  printFullName: function () {
    console.log(this.firstname + " " + this.lastname);
  }
}
name.printFullName();

let name2 = {
  firstname: "Sachin",
  lastname: "Tendulkar",
}
```

→ Now if we have another object and want to print firstName & lastName again, what we can do is copy & paste printFullName() f^n.

→ But instead we do function borrowing. So, we can borrow f^ns from other objects.

→ We do function borrowing using call() method.

→ Each & every f^n has access to call() method.



```javascript
// function borrowing
name.printFullName.call(name2);
```

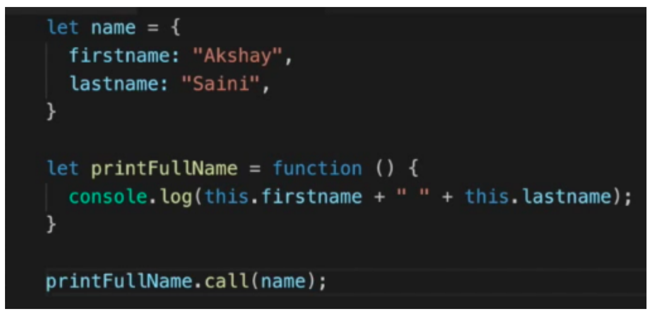
```
Sachin Tendulkar
>
```

→ Syntax for using call() method.

→ Here the 1st argument passed into call is the object which 'name' object's this will be pointing to.

→ So here 'this' will point to 'name2''s key: value pairs.

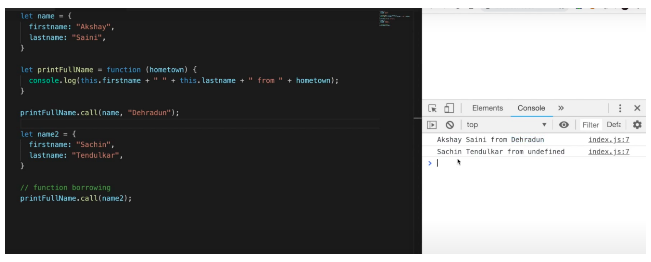
Note: Normally, we don't write f^ns inside an object & instead declare them outside.



```javascript
let name = {
  firstname: "Akshay",
  lastname: "Saini",
}

let printFullName = function () {
  console.log(this.firstname + " " + this.lastname);
}

printFullName.call(name);
```

→ So, for this case, we will directly call the call() method on the f^n & pass the object reference as argument, so that 'this' has access to the attributes of that object.

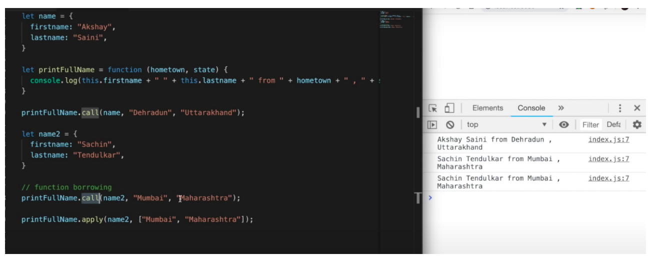
Note: If we need to pass more arguments to the f^n expression, then we can do so by passing these arguments as 2nd or 3rd or so on arguments to the call() method as first argument will always be the object reference.



```
let name = {
  firstname: "Akshay",
  lastname: "Saini",
}

let printFullName = function (hometown) {
  console.log(this.firstname + " " + this.lastname + " from " + hometown);
}

printFullName.call(name, "Dehradun");

let name2 = {
  firstname: "Sachin",
  lastname: "Tendulkar",
}

// function borrowing
printFullName.call(name2);
```

```
Akshay Saini from Dehradun          index.js:7
Sachin Tendulkar from undefined     index.js:7
>
```

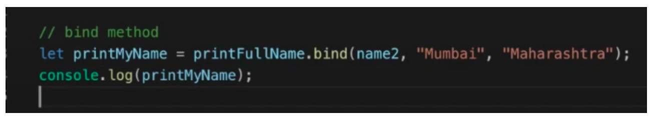
→ apply () : The only difference between call() & apply () methods are the way we pass arguments. Instead of individually passing the arguments, we pass them as a list in apply (). Rest everything is same.



```
let name = {
  firstname: "Akshay",
  lastname: "Saini",
}

let printFullName = function (hometown, state) {
  console.log(this.firstname + " " + this.lastname + " from " + hometown + " , " + s
}

printFullName.call(name, "Dehradun", "Uttarakhand");

let name2 = {
  firstname: "Sachin",
  lastname: "Tendulkar",
}

// function borrowing
printFullName.call(name2, "Mumbai", "Maharashtra");

printFullName.apply(name2, ["Mumbai", "Maharashtra"]);
```

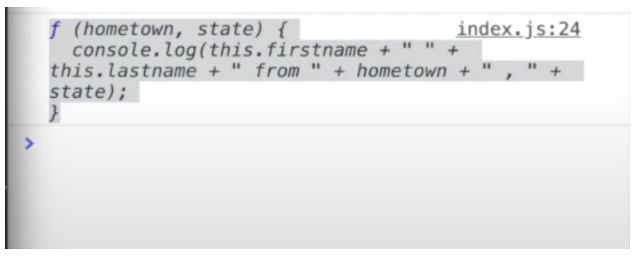
```
Akshay Saini from Dehradun ,         index.js:7
Uttarakhand
Sachin Tendulkar from Mumbai ,       index.js:7
Maharashtra
Sachin Tendulkar from Mumbai ,       index.js:7
Maharashtra
>
```

→ bind () : It looks just like the call method, but instead of directly calling the method, it binds the method to the object & returns a copy of the f^n to be invoked later directly.

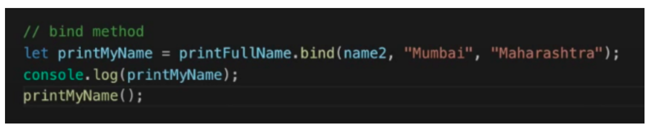


```
// bind method
let printMyName = printFullName.bind(name2, "Mumbai", "Maharashtra");
console.log(printMyName);
```

o/p :

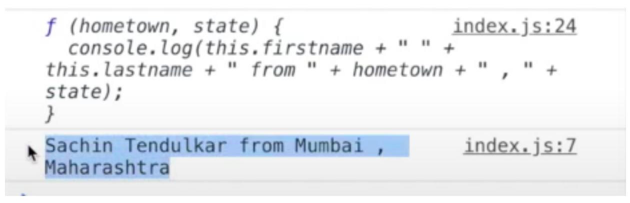


```
ƒ (hometown, state) {                        index.js:24
  console.log(this.firstname + " " +
this.lastname + " from " + hometown + " , " +
state);
}
>
```



```
// bind method
let printMyName = printFullName.bind(name2, "Mumbai", "Maharashtra");
console.log(printMyName);
printMyName();
```

o/p :



```
ƒ (hometown, state) {                        index.js:24
  console.log(this.firstname + " " +
this.lastname + " from " + hometown + " , " +
state);
}
Sachin Tendulkar from Mumbai ,               index.js:7
Maharashtra
```