02/4/24

Call, apply and bind

Call, apply and bind are used to call particular function by changing the context of the originating function.

Eg: If we want to call A object's method in B object then we can call that fn by changing context from A to B by this.

Call

Eg:
```
let obj1 = {
    name: 'Abhinav',
    age: 24
    printName: function() {
        console.log(this.name + ' ' + this.age)
    }
}

let obj2 = {
    name: 'Ashish',
    age: 43
}
```

→ want to print Name for obj2.

```
obj1.printName.call(obj2)
```

(this) context

Apply

The only difference between call and apply is the way we pass arguments

Eg:
```
let place1 = {
    street : 'Vir Sawarkar'
}

let place2 = {
    street: 'Vir Sawarkar'
}

function printAddress(homeTown, state) {
    console.log( "this. street + "is from " + hometown + " of
                 " + state)
}

printAddress.call( place1, "Mumbai", "Maharashtra") ;
printAddress.apply(place2,["PostBlair", "Andam&Nic"]);
```

Bind

Here we will not directly call the function rather Stores the copy of function and calls it later

Eg:
```
let printAddressOfVirS = printAddress.bind(place1, 'Mum', 'Mah')

printAddressofVirs ( );

(or)

let printAddressofVirs = printAddress.bind(place2)
printAddressofVirs ('postBlair', 'Andaman');
```

* Bind should have this context as argument, but remaining parameter cans be given while invoking.

The arguments can be given while storing (or) invoking (or) mix.

Bind polyfill

```
function printAddress (homeTown, state) {
    console.log ('this.street' + "is from " + homeTown + " of "
                                              + state)
}
```

Step 1

Use function.prototype to attach any fn to any object,

```
Function.prototype.myBind = function ....
```

Step 2

It is called afterwards, hence it returns a function

```
Function.prototype.myBind = function () {
    return function () {
    }
}
```

Step 3

It should be attached to printAddress.myBind, that means it has reference to this printAddress as this inside obj

```
Fn.ptype.myB = fn() {
    let obj = this;
    return fn() {
        obj.apply( ) ...
    }
```

Step 4

Check / maintain arguments

```
Fn.ptype.myB = fn(...args) {
    args[0] // this (street)
    return fn(...args) {
        obj.apply(...args)
    }
```

```
let street1 = {
    street: 'Veer savarkar'
};

let printAddressOfStreet1 = printAddress.myBind(street1, 'Andaman')
    printAddressOfStreet1('Andaman')
```

Polyfill

```
Function.prototype.myBind = function(...args1) {
    let obj = this;
    let params = args1.slice(1);
    return function(...args2) {
        obj.apply(args1[0], [...params, ...args2])
    }
};
```