DAY 96 Dev 40

Mycall
MyBind
Myapply
IIFE
Function as Object
Closure

Date → 24 Dec, 21
Day → Friday

let obj = { fun1 : function (frnd1, frnd2) { ... } }

→ console.log ("This person is called " + this.fullName + "His/her age is " + this.age + ".");

→ console.log (this.fullName + "says hello to " + frnd1 + ".");

→ console.log (this.fullName + "says hello to " + frnd2 + ".");

→ console.log(arguments);

}

fullName: "Suneet Malik"
age: 34

};

// object 2 ke.

let o2 = {
fullName: "Neha"
age = 33
}

SIMPLE CALL

↳ Simple way of calling a function will be used.

→ obj.fun1("Vikas", "Navdeep", "Jitu", "Rajneesh", "Kapil");

Here, fun1 is called where the first & second argument ie "Vikas" & "Navdeep" is passed as frnd1 & frnd2.
Jitu, Rajneesh, Kapil are also passed as arguments.

Output

this fullName here is obj. fullName

called

This person is called Sumeet Malik. His/her age is 34.
Sumeet Malik says hello to Vikas
Sumeet Malik says hello to Navdeep.
[Arguments] { '0': 'Vikas',
'1': 'Navdeep',
'2': 'Jitu',
'3': 'Rajneesh',
'4': 'Kapil'
}

this, age here is obj. age.

CALL() Function

Vikas & Navdeep is passed as frnd1 & frnd2 in fun 1

```
obj.fun1.call(O2, "Vikas", "Navdeep", "Jitu",
                  "Rajneesh", "Kapil");
```

object2 (O2) is passed here to override the default this

this.fullName = "Sumeet Malik"
this.age = 34
~~this~~ NOTE this → O2.fullName = "Neha"
O2.age = 33

Vikas, Navdeep, Jitu, Rajneesh, Kapil, all 5 are passed as arguments

Use ES11

Output

This person is called Neha. His/her age is 33.
Neha says hello to Vikas
Neha Says hello to Navdeep.
[Arguments] { '0': 'Vikas',
'1': 'Navdeep',
'2': 'Jitu',
'3': 'Rajneesh',
'4': 'Kapil'
}

Apply() और call() में बस यही difference है कि Apply() में argument को array में डालके pass करना पड़ता है

APPLY () FUNCTION

obj1.fun1.apply (O2, ["Vikas", "Navdeep", "Jitu", "Rajneesh", "Kapil"]);

O2 will be overtake the this

Vikas, Navdeep, Jitu, Rajneesh, Kapil are passed as an array. This array is considered as arguments array.

we will use O2.fullName & O2.age instead of this.fullName & this.age.

Output

- This person is called Neha. His/her age is 33.
- Neha says hello to Vikas.
- Neha says hello to Navdeep.
- [Arguments] {
  '0' : 'Vikas',
  '1' : 'Navdeep'
  '2' : 'Jitu'
  '3' : 'Rajneesh'
  '4' : 'Kapil'
  }

BIND () FUNCTION

Simple call, call() function, apply() function से fun1() function call हो जाता है

Bind से सीधा function call नहीं होता। Bind में सारे arguments (comma separated) वैसे ही pass करते हैं, जैसे call() में, Bind() function एक function return करता है

वो function है BoundFunction

जब हम BoundFunction को call करेंगे तो हमें अलग से argument pass करने की जरूरत नहीं पड़ेगी क्योंकि हमने Bind() function में args pass किए थे।

bind() function से हमें bound Function मिलता है in return

Bind में हम आगे arguments को pass किया है जिसे call में

```
let boundFunction = obj.fun1.bind(O2,
    "Vikas", "Navdeep",
    "Jitu", "Rajneesh", "Kapil");

boundFunction();

boundFunction("Jasbir", "Pankaj");
```

Jasbir & Pankaj will also be passed as an argument but they will be the last argument

```
This person is called Neha. His/her age is 33.
Neha says hello to Vikas
Neha says hello to Navdeep
[Arguments] { '0': 'Vikas',
    '1': 'Navdeep',
    '2': 'Jitu',
    '3': 'Rajneesh',
    '4': 'Kapil',
}
```

```
This person is called Neha. His/her age is 33.
Neha says hello to Vikas.
Neha says hello to Navdeep
[Arguments] { '0': 'Vikas',
    '1': 'Navdeep',
    '2': 'Jitu',
    '3': 'Rajneesh',
    '4': 'Kapil',
    '5': 'Jasbir',
    '6': 'Pankaj',
}
```

MYBIND (CUSTOM BIND FUNCTION)

```
Function.prototype.myBind = function()
{
    let orgFun = this;
    let argsArray = Array.from(arguments);
    let newThis = argsArray[0];
    let newParams = argsArray.slice(1);

    let myFun = Function()
    {
        let moreParams = Array.from(arguments);
        let TotalParams = newParams.concat(moreParams);
        orgFun.apply(newThis, totalParams);
    }
    return myFun;
}
```

Notes:

- this से original function होता है क्योंकि उसी पर myBind function call होगा
- myBind function में जो भी arguments दिए उन arguments को array में store करेंगे। Remember that first parameter passed is object तो first parameter के बाद वाले सारे parameter होंगे arguments array में store
- newThis → first parameter passed is an object or function
- moreParams → myFun function के arguments होंगे more Params
- TotalParams → तो Total Parameters होंगे myBind के argument + myFun के argument
- orgFun.apply → original function पर इसने (NewThis; Totalparameters) pass करादिए
- return myFun → myFun function को हमने return कर देगे।

→ O_2/object 2 pass नहीं किया तो null या undefined तो pass करना ही पड़ेगा

→ this change/override करने के लिए ही bind use करते हैं

→ let boundFunction = obj.fun1.myBind(
02,
"Vikas",
"Navdeep",
"Jitu",
"Rajneesh",
"Kapil"
);

→ boundFunction("Jasbir", "Pankaj");

→ Output

→ This person is called Neha.
His/her age is 33.
→ Neha says hello to Vikas.
→ Neha says hello to Navdeep.
→ [Arguments] { '0': Vikas
'1': Navdeep
'2': Jitu
'3': Rajneesh
'4': Kapil
'5': Jasbir
'6': Pankaj }

MY CALL (CUSTOM CALL FUNCTION)

↳ call() function को actually call करना होता है।
call function कोई function() return तही करती जैसे bind करता है।

```
let obj = {

    fun1: function (frnd1, frnd2)
    {console.log( " This person is called ".
                + this.fullName +
                " His/her age is " +
                  this.age
                );
    console.log( this.fullName +
                 " Says Hello to " +
                 frnd1 +
                 "!."
                );

    console.log (this.fullName +
                 "Says Hello to" +
                 frnd2 +
                 "!"
                );
    console.log(Arguments);

};

fullName : "Sumeet Malik",
age: 34
};
```

obj में है

```
let o2 = {
    fullName: "Neha"
    age: 33
}
```

this is the object 2

Now we are making the myCall function

```
Function.prototype.myCall = function() {
    let orgFun = this;
    let args = Array.from(arguments);
    let thisForCall = args[0];
    let params = args.slice(1);
    orgFun.apply(thisForCall, params);
}
```

myCall function पर call होगा तो this में होगा orgFun में store करेंगे

we are collecting the parameters in an array & that array is stored in args

args array में सभी arguments होंगे (o2 & all the funds)

object 2 will override the default this for the function

this.fullName और this.age को override करेंगे o2.fullName & o2.age

params में सभी friends को (o2 को छोड़कर सभी args को) store करा दिया

this (o2)

apply func is called

params एक array है

array को pass करने के लिए we need to use apply function

MY CALL WITHOUT USING APPLY

```
Function.prototype.myCall = function ()
{
  let orgFun = this;
  let args = Array.from(arguments);
  let thisForCall = args[0];
  let params = args.slice(1);
```

↳ array में लेगा as return

```
  thisForCall.fun = orgFun;
```

↓

thisForCall में हम O2 (object 2) में basically हम O2 में एक fun नाम की property add कर दें जिसमें हमने originalFunction (this) i.e obj को डाल दिया।

```
O2 : { fullName : "Neha"
       age : 33
       fun : orgFun
     }
```

↳ Here orgFun is वो function जिसे myCall call हुआ i.e obj में जो fun था वो है this

तो thisForCall.fun का मतलब है O2.fun

→ this.ForCall.fun(...params);

params एक array होगा
उस array के सभी elements,
अलग अलग items बन
जाते हैं, इसके (...) dot लगाने से

this.ForCall(O2.fun)

तो अब जब हम fun को call करेंगे

fun("Vikas", "Navdeep", "Jitu", "Rajneesh", "Kapil")

fun में ये सब items pass होजाएंगे

→ delete this.ForCall.fun

(O2 से fun property को delete करदो)

}

Now, we can use our myCall() function

Obj.fun1. myCall(O2, "vikas", "Navdeep", "Jitu", "Rajneesh", "Kapil");

→ This will give the same output as the Call() function gives

MY APPLY (CUSTOM Apply function):

```
Function.prototype.myApply = function () {

    let orgFun = this;
    let args = Array.from(arguments);
    let thisForCall = args[0];
    let params = args[1];

    thisForCall.fun = orgFun;

    thisForCall.fun(...params);

    delete thisForCall.fun;
};
```

this ForCall will be first parameter ओ2

arguments का Array के second parameter

ओ2 (object 2) में एक new property बनाई जिसका नाम fun है

ओ2 के fun functions में सारी items pass होती है params के

Array के elements की individual items बन जाती है

ओ2 के fun property को delete करते है

This gives the same result as Apply function gives

Page No.
Date

```
obj.fun1.myApply(02, ["Vikas",
"Navdeep", "Jatu", "Rajnish", "Kapil"]);
```

arr = [10, 20, 30, 40, 50]

...arr = 10, 20, 30, 40, 50.

list को individual array

Simple array के elements को अलग अलग items बनाकर हम using (...)