Call Apply Bind

2nd Jun, 2020 12:00 PM

Albert Sebastian

Week # 7 | Day # 2

this

We are going to discuss about `this` keyword again.

We dont know what `this` references until the execution of that function. `When a function is invoked, the object thats to the left, and that is what the`this` keyword will be referencing. This is an implicit way of binding this

Lets look at an example of a function inside an object

```
var person = {
    name:'haren',
    age:25,
    type:'person',
    display:function(){
        console.log(this.type,' is ', this.name)
    },
    company:{
        name:'masai',
        type:'organisation',
        display:function(){
            console.log(this.type,' is ', this.name)
        }
    }
}

person.display()
person.company.display()

// lets try to reuse that display function

function displayWrapper(obj){
    obj.display = function(){
        console.log(this.type,' is ', this.name)
    }
}

var person2 = {
    name:'manu',
    age:22,
    type:'person'
}

displayWrapper(person)
displayWrapper(person2)
displayWrapper(person.company)

// person is haren
// organisation is masai
// person is manu
```

call, apply, bind

Every function has a call, apply and a bind method onto it. We can `explicitly bind` an object to a function.

- With the call() method, you can write a method that can be used on different objects.
- The call() method calls a function with a given this value and arguments provided individually.

call MDN

```
function display(){
    console.log(this.type,' is ', this.name)
}

var person = {
    name:'haren',
    age:25,
    type:'person',
    company:{
        name:'masai',
        type:'organisation'
    }
}
// invokes the function with context of the object passed in the argument
display.call(person)
diplay.call(person.company)


// call takes in a list of arguments
function display(arg1, arg2){
    console.log(this.type," is"  this.name ,arg1, arg2)
}

display.call(person, 'argument1', 'argument2')
```

apply

The apply() method calls a function with a given this value, and arguments provided as an array (or an array-like object).

While the syntax of this function is almost identical to that of apply(), the fundamental difference is that call() accepts an argument list, while apply() accepts a single array of arguments.

```
// invokes the function with context of the object passed in the argument
along with array of arguments
display.apply(person, [arg1,arg2])
display.apply(person.company,[arg1,arg2])
```

apply MDN

bind

The bind() method creates a new function that, when called, has its this keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is called.

```
// bind returns a new function with context of the object passed in the
argument that can be invoked later
d1 = display.bind(person)
d2 = display.bind(person.company)

// d1 will invoke diplay with person in context
d1()
// d2 will invoke display with person.company in context
d2()
```

bind MDN

“call”, "apply", "bind" is a method on every function that allows you to invoke the function specifying in what context the function will be invoked.

What happens if you invoke a function without any kind of object as reference.

In this case it will be binded to the window object

```
window.type = 'window'
window.name = 'invoked'

// global variables take window scope.

function display(){
    console.log(this.type,' is ', this.name)
}

display()
// window is invoked

// we can use a 'use strict';
// to ensure that this will throw an error

function display(){
    'use strict';
    console.log(this.type,' is ', this.name)
}
```

use strict

`"use strict";` Defines that JavaScript code should be executed in "strict mode".

Strict mode is declared by adding `"use strict";` to the beginning of a script or a function.

Declared at the beginning of a script, it has global scope (all code in the script will execute in strict mode):

```
"use strict";
myFunction();

function myFunction() {
  y = 3.14;   // This will also cause an error because y is not declared
}


 x = 3.14;       // This will not cause an error.
myFunction();

function myFunction() {
  "use strict";
  y = 3.14;   // This will cause an error
}
```

This article gives also provides a good explanation:

2nd Jun, 2020 12:59 PM