**DIFFERENCE BETWEEN BIND,CALL AND APPLY IN JAVASCRIPT**

In Object Oriented JavaScript, everything is an object and therefore we can set and access additional properties to functions and methods via the prototype

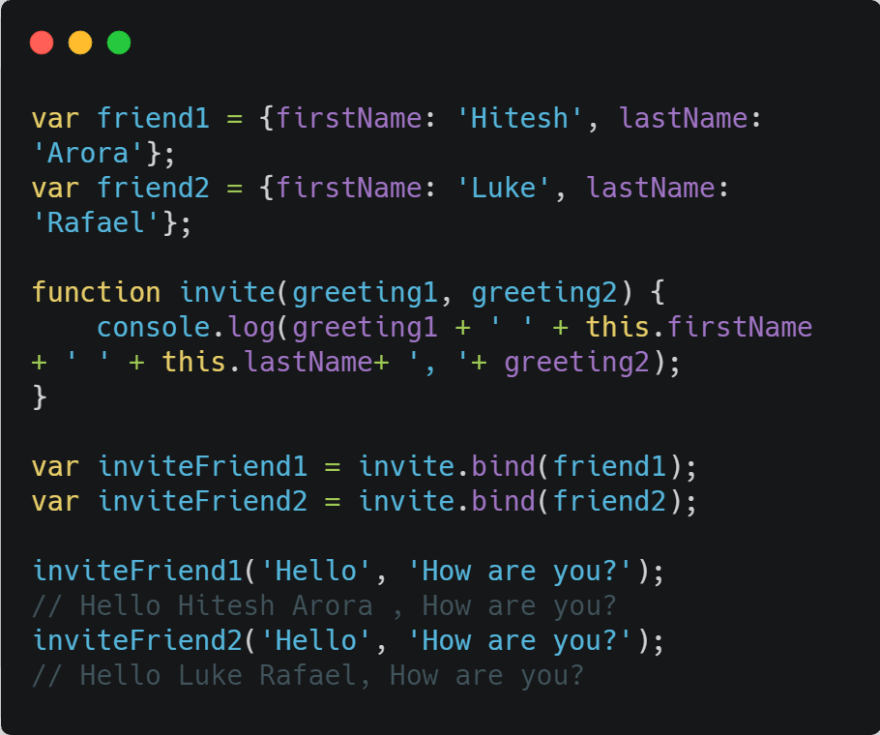
Call( ): The call() method invokes a function with a given 'this' value and arguments provided one by one. This means that we can call any function, and explicitly specify what 'this' should reference within the calling function.



Apply( ): Invokes the function and allows you to pass in arguments as an array.



Bind(): returns a new function, allowing you to pass in an array and any number of arguments.



When we use the bind() method:

1.The JS engine is creating a new *invite* instance and binding *friend1* and *friend2* as its 'this' variable. So basically it copies the *invite* function.

2.After creating a copy of the *invite* function it is able to call *inviteFriend1( )* and *inviteFriend2( )*, although it wasn’t on the *friend1* and *friend2* object initially. It will now recognizes its properties and its methods.

*Call* and *apply* are pretty interchangeable. Both execute the current function immediately. You need to decide whether it’s easier to send in an array or a comma separated list of arguments. Whereas Bind creates a new function that will have this set to the first parameter passed to bind().

**call(), bind(), and apply()** control the invocation of functions just like you call a function in the normal way just like this function:

**doSomething**(param1, param2, etc.);

can be written in another way using **call()** like this:

**doSomething**.**call**(this, param1, param2, etc.);

You can use call()/apply() to invoke the function immediately. bind() returns a bound function that, when executed later, will have the correct context (**“this”**) for calling the original function. So bind() can be used when the function needs to be called later in certain events when it’s useful.

## **call() or Function.prototype.call()**

//Demo with javascript .call()

**var** obj = {name:"John"};

**var** greeting = **function**(a,b,c){

**return** "welcome "+**this**.name+" to "+a+" "+b+" in "+c;

};

**console**.log(greeting.call(obj,"Newtown","KOLKATA","WB"));

// returns output as welcome John to Newtown KOLKATA in WB

So **call()** calls the function with (**this or custom object**) as the first parameter then function parameters if exist are passed one by one and returns the value of execution.

## **apply() or Function.prototype.apply()**

apply() is the same as call() except that it takes the parameters as any array:

//Demo with javascript .apply()

**var** obj = {name:"John"};

**var** greeting = **function**(a,b,c){

**return** "welcome "+**this**.name+" to "+a+" "+b+" in "+c;

};

// array of arguments to the actual function

**var** args = ["Newtown","KOLKATA","WB"];

**console**.log("Output using .apply() below ")

**console**.log(greeting.apply(obj,args));

/\* Will output

welcome John to Newtown KOLKATA in WB \*/

## **bind() or Function.prototype.bind()**

The last method is **bind()** and it’s different than **call()** and **apply()** in that it creates a new function with a given this value, and returns that function without executing it.

//Use .bind() javascript

**var** obj = {name:"John"};

**var** greeting = **function**(a,b,c){

**return** "welcome "+**this**.name+" to "+a+" "+b+" in "+c;

};

//creates a bound function that has same body and parameters

**var** bound = greeting.bind(obj); // this will return a function bound()

// calling bound()

**console**.log("Output using .bind() below ");

**console**.log(bound("Newtown","KOLKATA","WB")); //call the bound function

/\* will output

welcome John to Newtown KOLKATA in WB \*/

Consider also this example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Bind - Call - Apply</title>

<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

</head>

<body>

<button>Get Random Person</button>​

​ <input type="text">​

​

​ <script type="text/javascript">

var user = {

data: [

{name:"T. Woods", age:37},

{name:"P. Mickelson", age:43}

],

clickHandler:**function** (event) {

var randomNum = ((Math.random () \* 2 | 0) + 1) - 1;

$ ("input").val(**this**.data[randomNum].name + " " + **this**.data[randomNum].age);

}

}

$("button").click(user.clickHandler.bind(user));

</script>

</body>

</html>

As you see in the previous example in this code block **$(“button”).click(user.clickHandler.bind(user))** will bind the user object to the **clickHandler** function so you can access all properties in the user object from the **clickHandler** function.