

Call - apply - bind

ال `call` و ال `apply` هما `methods` بقدر من خلالهم ان انا لو عندي `object` حالي اقدر ابدله ب `object` جديد
ال `call` و ال `apply` مفيش فرق بينهم غير ان ال `call` بيطلع الاوبجكت زي ماهو مبيقبلش مصفوفة اللي هي `array`
ال `apply` هو اللي بنستخدمه لو هنضيف `array` لل `object` الجديد
ال `bind` زيهم ولكن هو بيطلع ال `functions` باللي فيها مش ال `objects` نفسها
بنستخدمهم عشان اشاور علي اي `parameters` و يطلعني اللي جواها سواء `properties` او `variables`

مثال عال `call`

```
1
2
3 var obj = {num:2};
4
5
6 var addToThis = function(a){
7
8     return this.num + a;
9 };
10
11 console.log(addToThis.call(obj, 3));
12 // |functionname.call(obj, functionargumentes);
```

Sources Timeline Profiles Resources Audits Console

▼ ☐ Preserve log

```

1
2
3 var obj = {num:2};
4
5
6 var addToThis = function(a, b, c){
7
8     return this.num + a + b + c;
9 };
10
11 //console.log(addToThis.call(obj, 1, 2, 3));
12 //functionname.call(obj, functionarguments);
13
14 var arr = [1,2,3];
15 console.log(addToThis.apply(obj, arr));
16
17

```

Call is a function that helps you change the context of the invoking function. In layperson's terms, it helps you replace the value of `this` inside a function with whatever value you want.

Apply is very similar to the `call` function. The only difference is that in `apply` you can pass an array as an argument list.

Bind is a function that helps you create another function that you can execute later with the new context of `this` that is provided.

Example on bind :

```
// object definition
const student1 = {
  name: "Jack",
  grade: "5",
  introduction: function () {
    console.log(this.name + "studies in grade" + this.grade + ".");
  },
};

// object definition
const student2 = {
  name: "Jimmy ",
  grade: " 6",
};

// the object student2 is borrowing introduction method from student1
let result= student1.introduction.bind(student2);

// invoking introduction() function
result();

// Output:
// Jimmy studies in grade 6.
```