Call-bind-apply

In JavaScript, `call`, `bind`, and `apply` are three methods available on functions that allow you to control how a function is executed and to manipulate the value of `this` within the function. Here's a brief explanation of each method along with their key points:

`call` Method:

- The `call` method is used to invoke a function immediately.
- It allows you to set the value of `this` explicitly when calling the function.
- You can pass arguments to the function as individual comma-separated values.
- Syntax: `functionName.call(thisArg, arg1, arg2, ...)`

Example:

```
var obj = {name: "Hayat"}

function sayHello(age){

  return "Hello " + this.name + ' is ' +age;
}

console.log(sayHello());//output: Hello  is undefined

console.log(sayHello.call(obj,24));// Hello Hayat is 24
```

`apply` Method:

- The `apply` method is similar to `call` but takes an array-like object of arguments.
- It is useful when you have an array of arguments that you want to pass to a function.
- Syntax: `functionName.apply(thisArg, [arg1, arg2, ...])`

```
var obj = {name: "Hayat"}

function sayHello(age,profession){

  return "Hello " + this.name + ' is ' +age + " and is an " + profession;
}

console.log(sayHello.apply(obj,[24, "Software Engineer"]));//output: Hello Hayat is 24 and is an Software Engineer
```

`bind` Method:

- The `bind` method is used to create a new function that is "bound" to a specific `this` value.
- It doesn't immediately invoke the function; instead, it returns a new function with the specified context.
- It's often used to create functions with a fixed context for event handlers or callbacks.

- Syntax: `const boundFunction = functionName.bind(thisArg, arg1, arg2, ...)`

```
var obj = {name: "Hayat"}

function sayHello(age,profession){

  return "Hello " + this.name + ' is ' +age + " and is an " + profession;
}

const bindFunc = sayHello.bind(obj)

console.log(bindFunc(24, "Software Engineer"))//Hello Hayat is 24 and is an Software Engineer
console.log(bindFunc(50, "Software Engineer"))//Hello Hayat is 50 and is an Software Engineer
```

Q: Call with function inside object

```
const age = 10;

var person = {
  name: "Hayat",
  age:24,
  getAge: function(){
    return this.age;//note: arrow function will point to window object
  }
}

var person2 = {age:24};

console.log(person.getAge.call(person2))// 24
console.log(person.getAge.apply(person2)) // 24
console.log(person.getAge.bind(person2))// function
console.log(person.getAge.bind(person2)())// 24
```

Q

```
const status = '10';

setTimeout(() => {

  const status = '20';
  const data = {
    status: '30',
    getStatus() {
      return this.status
    }
  }

  console.log(data.getStatus())//30
  console.log(data.getStatus.call(this))//output nothing
},0)
```

Q: Call printAnimals such that it prints all animals in object

```
const animals = [
  { species: "Lion", name: "King" },
  { species: "Whale", name: "Queen" },
```

```
]

function printAnimals(i) {
  this.print = function () {
    console.log("#" + i + " " + this.species + ": " + this.name)
  };
  this.print()
}

for (let i = 0; i < animals.length; i++) {
  printAnimals.call(animals[i], i)
  //#0 Lion: King
  //#1 Whale: Queen
}
```

Q: Append an array to another array

```
const array = ["a","b"];
const elements = [0,1,2];

array.push.apply(array,elements);
console.log(array)//[ "a", "b", 0, 1, 2 ]
```

Q: Find min/max number in an array

```
const numbers = [5,6,3,2,4];
console.log(Math.max.apply(null,numbers));//6
```

Q: Bound function

```
function f(){
  console.log(this);//output:window object
  }

  let user = {
  g: f.bind(null),
  };
  user.g()
```