JS Call() → Apply() → Bind()

Call() → Used to invoke a function with specified this value & arguments provided individually.

```
let ql = doc.querySelector("p");

function greet(message){
    console.log(this.name);
    return `${message}, ${this.name}`;
}

const person = {
    name: 'John'
}

ql.innerHTML = greet.call(person, "Hi");
```

o/p
Hi, John
console o/p
{name: John}

Syntax → function.call(thisArg, 'arg1', arg2, ...)

Specify the value of this within the function when it is invoked

```
<Script>
    let one = doc.getEle..ById("one");
    let two = doc.getEle.ById("two");

    const person = {
        fullname: function(location, rollno){
            return `${this.firstname} ${this.lastname}
                from ${location}) Rollno ${rollno}`;
        }
    };

    const person1 = {firstname: "Arjun", lastname: "Navnoor"}
    const person2 = {firstname: "Tejun", lastname: "Noor"}

    one.innerHTML = person.fullname.call(person1, "Mumbai", 45);
    two.innerHTML = person.fullname.call(person2, "Mumbai", 46);
</Script>
```

o/p
Arjun Navnoor from Mumbai, Roll No 45
Tejun Noor from Mumbai Rollno 46

Apply()

→ Similar to call() but it accepts arguments as an array.

Eg: In call() → Modify last line of 1st example.

```
g1.innerHTML = greet.apply(person, ["Hi"]);
```

o/p: Hi, John

↳ It takes Array like object of arguments instead of individual arguments.

Syntax: function.apply(thisArg, [arg1, arg2...])

Eg: In call() → Modify last line of 2nd example

```
one.innerHTML = person.fullName.apply(person1, ["Mumbai", 32]);
two.innerHTML = person.fullName.apply(person2, ["Mumbai", 34]);
```