CALL APPLY BIND

JS

INTERVIEW QUESTIONS-46

Follow on in

@Duvvuru Kishore

FIRST WE WILL HAVE A LOOK AT THIS KEYWORD

In JavaScript,

- this keyword refers to an object.
- Every function gets this property automatically
- And will always refer to a single object

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
  printName: function() {
    console.log(this.firstName + ' ' + this.lastName);
  }
};

person.printName(); // This will print "John Doe" to the console
```

WHY WE NEED CALL,BIND AND APPLY

- call, apply, and bind are methods in JavaScript that provide more flexibility and control when working with functions.
- especially in scenarios where you need to specify the context (this value) or pass arguments explicitly.

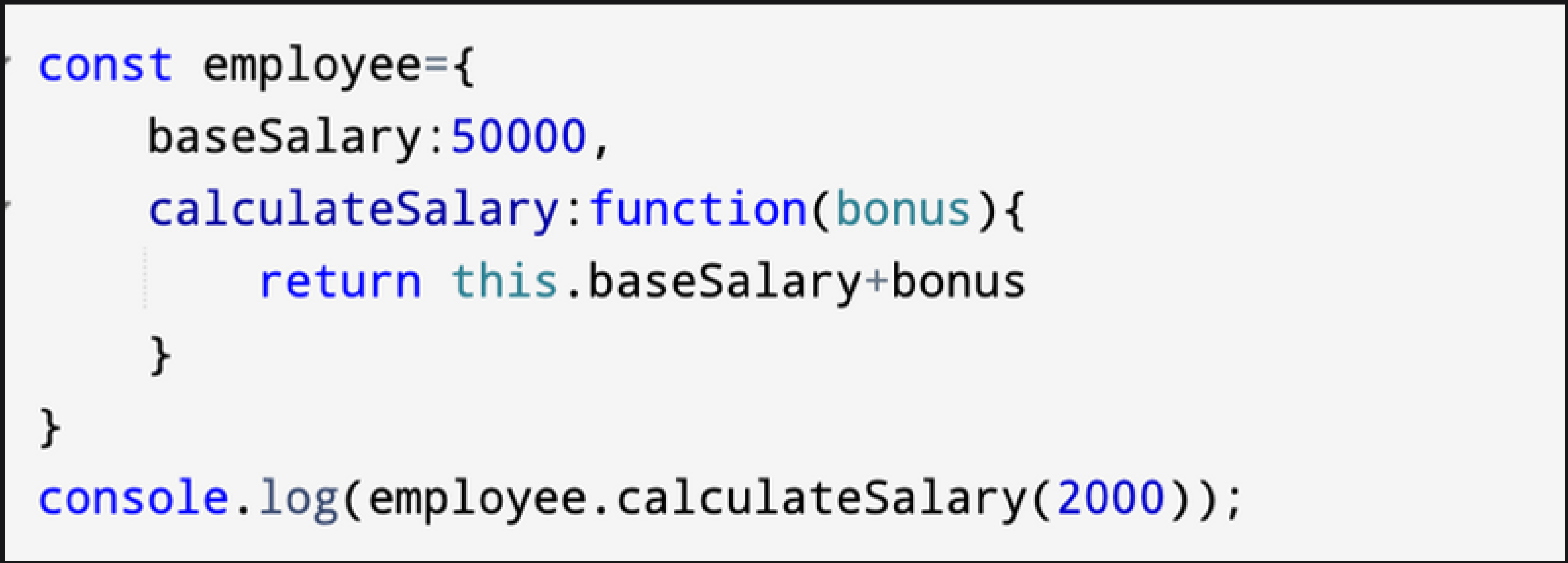
CALL()

WHEN TO USE CALL ?

Let’s have a example where programmatically it tells when to use Call()

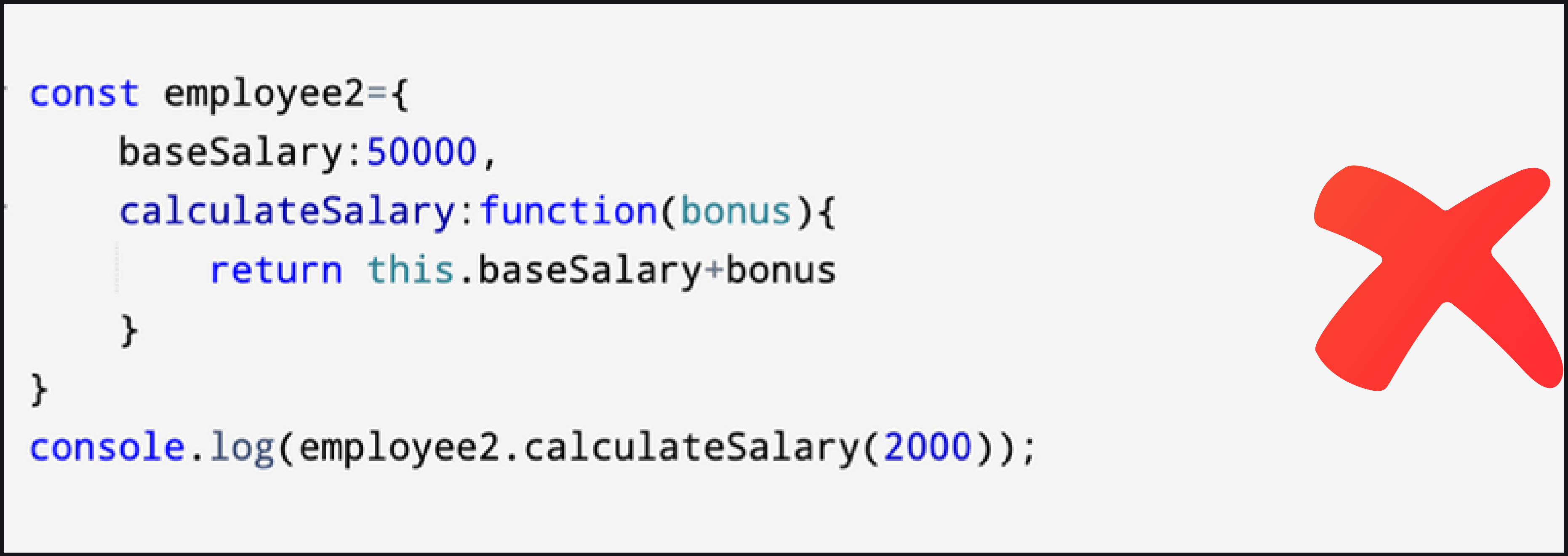
lets have a look at how to calculate salary and bonus of different employess.

CALCULATING EMPLOYEE 1 SALARY



```
const employee={
    baseSalary:50000,
    calculateSalary:function(bonus){
        return this.baseSalary+bonus
    }
}
console.log(employee.calculateSalary(2000));
```

CALCULATING EMPLOYEE 2 SALARY

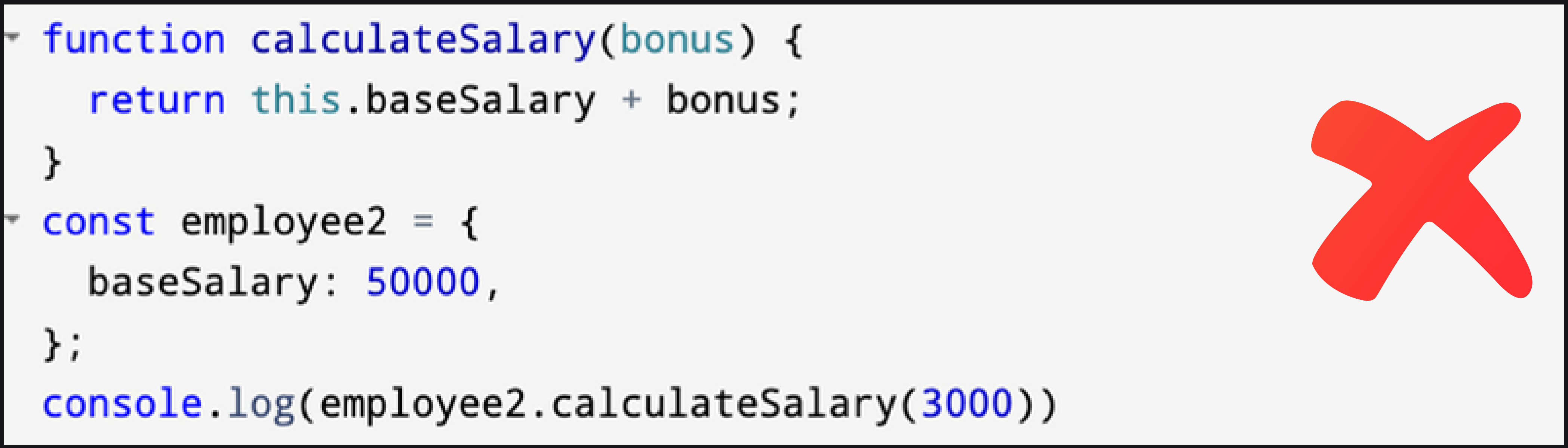


```
const employee2={
    baseSalary:50000,
    calculateSalary:function(bonus){
        return this.baseSalary+bonus
    }
}
console.log(employee2.calculateSalary(2000));
```

NOT A GOOD PRACTICE BEACUSE?

- We are using same functions into different objects, which is creating duplicte functions.
- so need to remove duplicate functions and need to make it work with single function for good programming practice .

MAY BE YOU CAN THINK YOU CAN WRITE IN THIS WAY

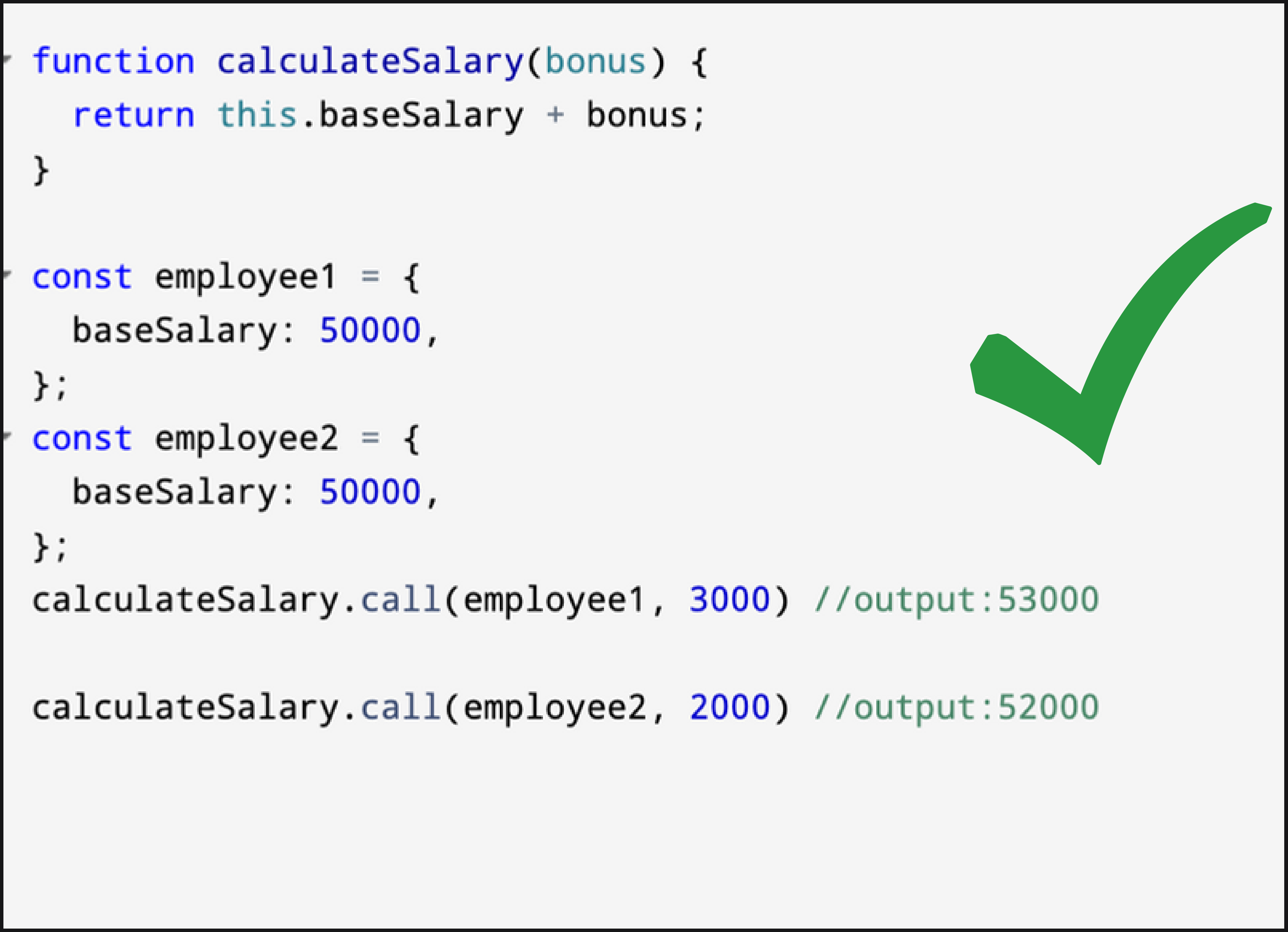


```
function calculateSalary(bonus) {
  return this.baseSalary + bonus;
}
const employee2 = {
  baseSalary: 50000,
};
console.log(employee2.calculateSalary(3000))
```

calculateSalary function is not a method of the employee2 object, so it doesn't have access to this.baseSalary.

THIS IS WHERE CALL COMES IN RESCUE

SAME EXAMPLE WITH CALL()



```javascript
function calculateSalary(bonus) {
  return this.baseSalary + bonus;
}

const employee1 = {
  baseSalary: 50000,
};
const employee2 = {
  baseSalary: 50000,
};
calculateSalary.call(employee1, 3000) //output:53000

calculateSalary.call(employee2, 2000) //output:52000
```

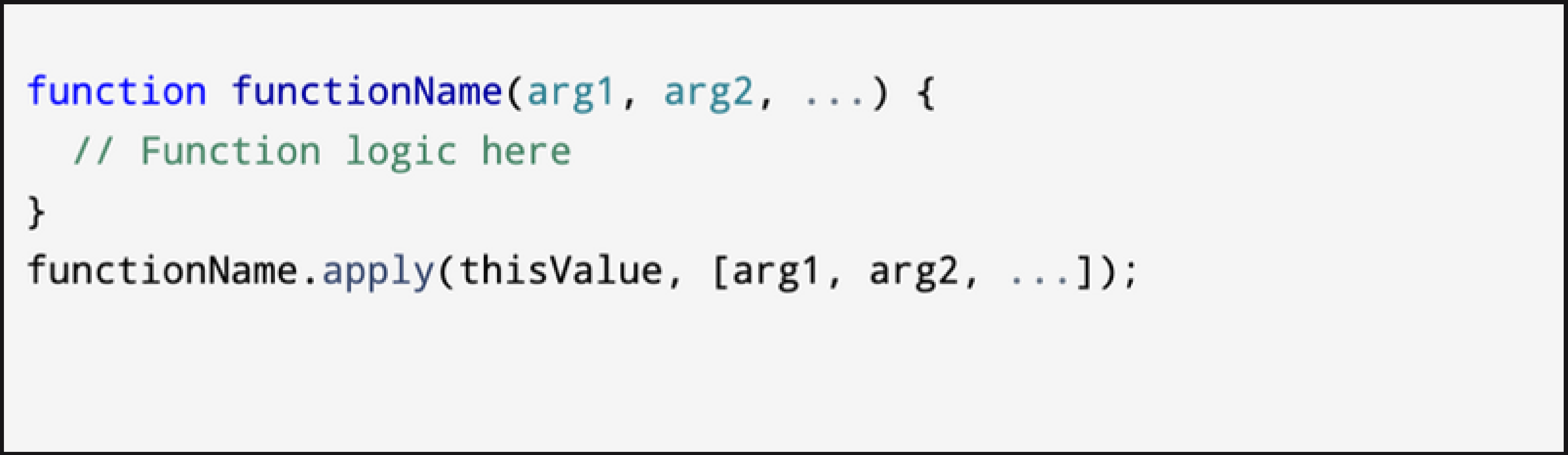
- **calculateSalary added with prototype call() to pass objects.**
- **In call along with objects , we can pass parameters (3000,2000)**

CALL DEFINITION COME EXPLANATION

- The call() method is used to invoke a function with a specified this value and arguments provided individually.
- It allows you to call a function as if it were a method of an object, even if it's not originally defined as a method of that object.

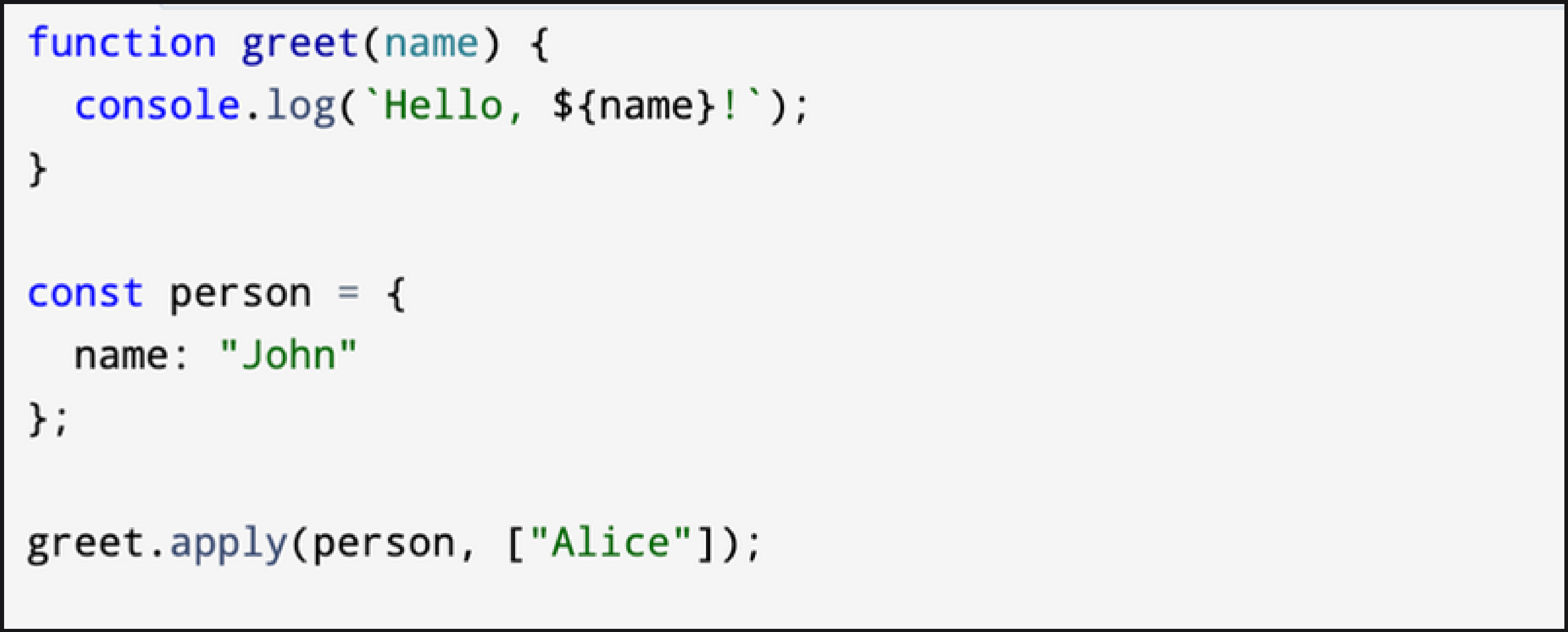
APPLY()

- apply() method in JavaScript is similar to the call() method, but it accepts arguments as an array rather than individually.
- It allows you to call a function with a specified this value and an array or array-like object of arguments



```
function functionName(arg1, arg2, ...) {
  // Function logic here
}
functionName.apply(thisValue, [arg1, arg2, ...]);
```

EXAMPLE



```javascript
function greet(name) {
  console.log(`Hello, ${name}!`);
}

const person = {
  name: "John"
};

greet.apply(person, ["Alice"]);
```

BIND 👉

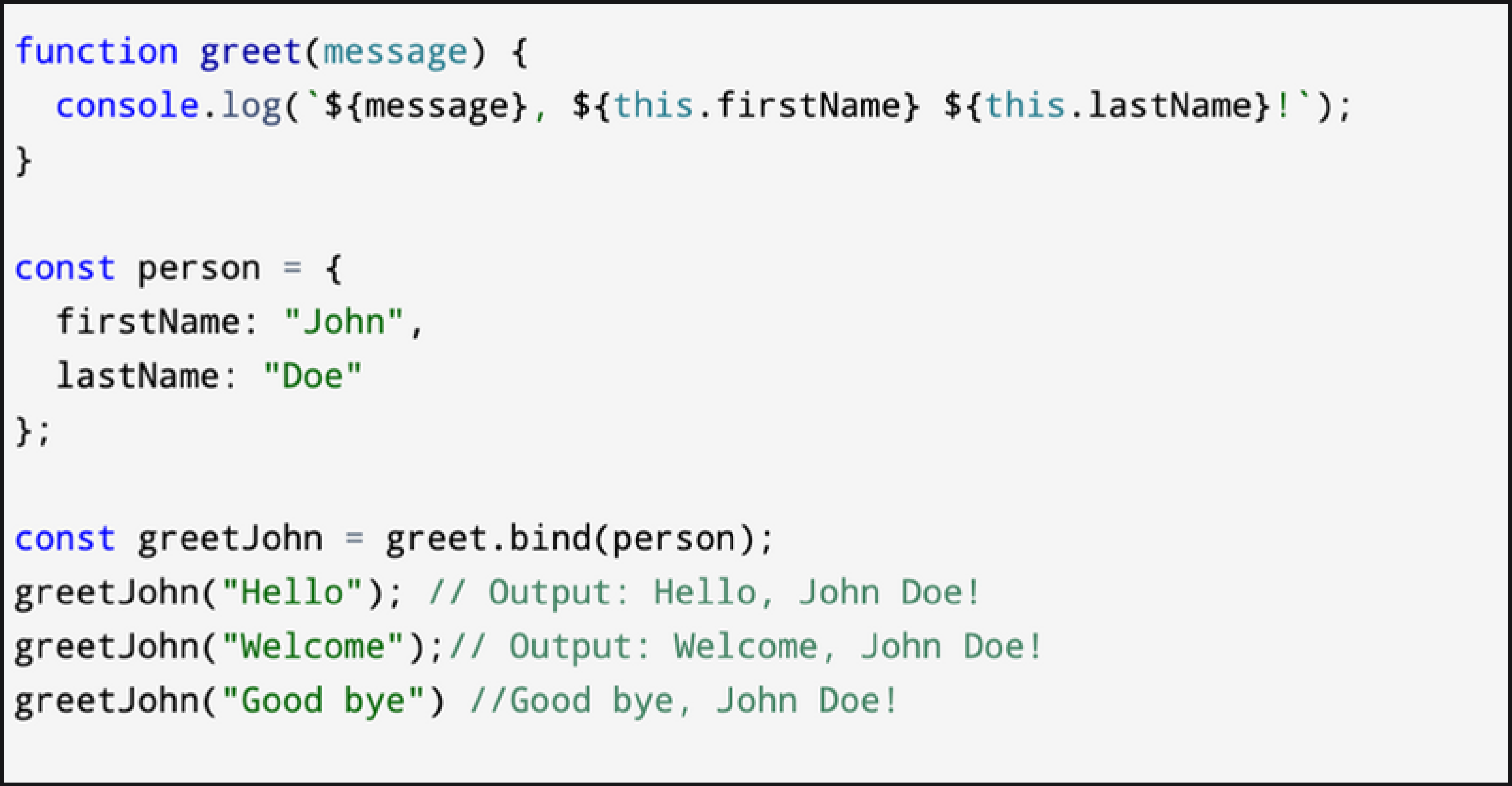
BIND()

bind() is a method that creates a new function with a specific this context and optionally prepends arguments to the argument list.

NOTE:

call() is a method that allows you to invoke a function immediately. where as bind() is a method that creates a new function .

EXAMPLE



```
function greet(message) {
  console.log(`${message}, ${this.firstName} ${this.lastName}!`);
}

const person = {
  firstName: "John",
  lastName: "Doe"
};

const greetJohn = greet.bind(person);
greetJohn("Hello"); // Output: Hello, John Doe!
greetJohn("Welcome");// Output: Welcome, John Doe!
greetJohn("Good bye") //Good bye, John Doe!
```

As bind creates new function with specific this context , we see greetJhon() called with different values like hello, welcome etc...which is giving the respective output with this context.and it’s not possible with call() as it’s immediately invoked.

Follow

ON!

Follow on in

@Duvvuru Kishore