Function advance Stuff

CODEMIND
Your Success Partner

By CODEMIND Technology

Contact us 966 5044 698
966 5044 698

JavaScript Function advance stuff

- Function is a First class citizen
- Higher order function
- call( )
- apply( )
- bind( )
- Function Currying

Functions are known as first class citizens in JavaScript

It is because of three reasons

1. Function can be stored in a variable
2. Function can be passed as an argument to another function
3. A function can return another function

```
// 1. Function can be stored in a variable
function show() {
    console.log("I am learning JS");
}
let result = show;
result();
```

```
// 2. Function can be passed as an argument to another function
function show() {
    console.log("I am learning JS");
}
function say_hello(show_fun) {
    console.log("Hello");
    show_fun();
}
say_hello(show);
```

```
// 3. A Function can return another function
function message() {
    console.log("Hello dude");
    return function() {
        console.log("I am learning JS");
    }
}
message()();
```

Higher order Functions

A function that accepts functions as a parameters and or returns function

```
function do_homework(callbacks) {
    console.log("Doing home work.. Solving tricky problem");
    console.log("Finally, solved ");
    callbacks();
    let complete_homework = function () {
        console.log("Completed Home work,, Thank you, Friend");
    }
    return complete_homework;
}

function copy_homework() {
    console.log("Copy homework from friend's notes");
}
let homework_done = do_homework(copy_homework);
homework_done();
```

call( ) method

The call method invokes a function with a given this value and arguments provided one by one. Syntax: functionName.call(object, arg1, arg2, arg3, argN);

```
1   const person = {
2       fullName: "Bill gates",
3       company: "Microsoft"
4   }
5   function greetings(message){
6       console.log(`Hi ${this.fullName} ${message}`);
7   }
8   greetings.call(person, "Good Morning");
```

PROBLEMS 13 DEBUG CONSOLE OUTPUT TERMINAL Filter

```
/opt/homebrew/bin/node ./test.js
Hi Bill gates Good Morning
```

apply( ) method

The apply() method invokes a function with a given this value and allows us to pass an arguments in array. Syntax: functionName.apply(object, [arg1, arg2, arg3, argN]);

```
1   const person = {
2       fullName: "Bill gates",
3       company: "Microsoft"
4   }
5   function greetings(message, role){
6       console.log(`Hi the ${role} Mr. ${this.fullName} ${message}`);
7   }
8   greetings.apply(person, ["Good Morning", "CEO"]);
```

PROBLEMS 13 DEBUG CONSOLE OUTPUT TERMINAL Filter (e.g. text, !ex...

```
/opt/homebrew/bin/node ./test.js
Hi the CEO Mr. Bill gates Good Morning
```

bind( ) method

The bind( ) method is little bit different than call( ) and apply( ) methods.
It returns a new function, and allows us to pass any number of arguments.

Syntax: `const newFunction = functionName.bind(object);`
`newFunction(arg1, arg2, argN )`

```
1   const person = {
2       fullName: "Bill gates",
3       company: "Microsoft"
4   }
5   function greetings(greet, role, word){
6       console.log(`Hi the ${role} Mr. ${this.fullName} ${greet} ${word}`);
7   }
8   const newFun = greetings.bind(person);
9   newFun("Good Morning", "CEO", "you are great");
```

PROBLEMS 13 | DEBUG CONSOLE | OUTPUT | TERMINAL | Filter (e.g. text, !exclude)

```
/opt/homebrew/bin/node ./test.js
Hi the CEO Mr. Bill gates Good Morning you are great
```

Note

Call and apply are pretty interchangeable. Both execute the current function immediately. we need to decide whether it’s easier to send in an array or a comma separated list of arguments.

We can remember by treating Call is for comma (separated list) and Apply is for Array.

Whereas bind creates a new function that will have this set to the first parameter passed to bind().

Function currying

Function currying is the process of taking a function with multiple arguments and turning it into a sequence of functions each with only a single argument. Currying is named after a mathematician Haskell Curry. By applying currying, a n-ary function turns it into a unary function.

```
const multiArgFunction = (a, b, c) => a + b + c;
console.log(multiArgFunction(1,2,3));// 6
const curryUnaryFunction = a => b => c => a + b + c;
curryUnaryFunction (1); // returns a function: b => c => 1 + b + c
curryUnaryFunction (1) (2); // returns a function: c => 3 + c
curryUnaryFunction (1) (2) (3); // returns the number 6
```

Self invoking Function or IIFE → Immediately Invoked Function Expression

IIFE is a function defined as an expression and executed immediately after creation. The primary reason to use an IIFE is to obtain data privacy because any variables declared within the IIFE cannot be accessed by the outside world. i.e, If you try to access variables with IIFE then it throws an error as below,

```
(function(){
    ...
    ...
    ...
})();
```

```
(function () {
    console.log("Hello I am inside IEFE");
})();
```

Thank you