Advanced Constructs: Advanced Function Concepts-1

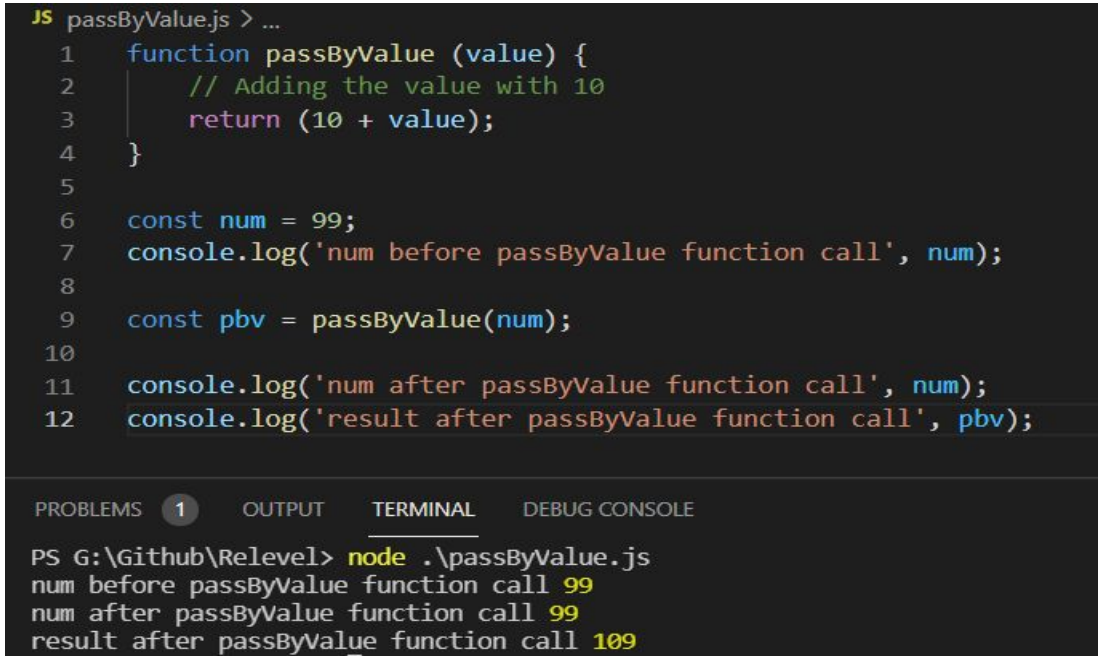
Relevel
by Unacademy

Topics Covered

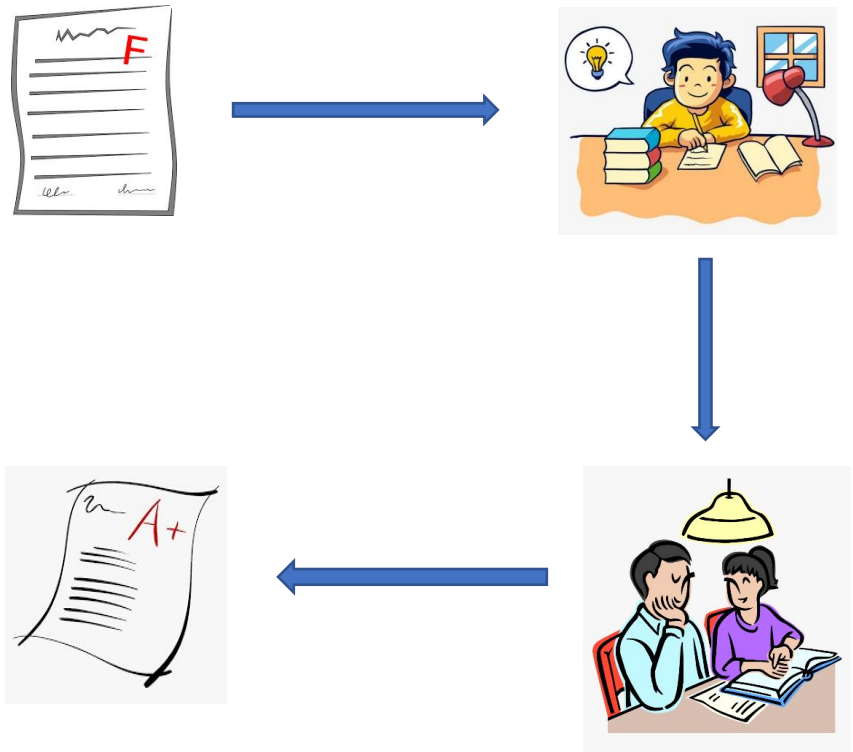
- Pass by Value
- Pass by Reference
- Pure / Impure function and Side Effects
- Closures

Pass by Value

- A function is called by directly passing the value of the variable and any change in the variable inside the function does not affect the outside value of variable.
- As a beginner the above statement is bit confusing, let me explain this with real time example.
- In this example passByValue function will add value in the variable with 10 => (value + 10) , so adding 10 with the variable does not change the value in the num variable.
- Because we are making the copy and send the value as an argument, here primitive type play an role in this example.



Pass by Value



Let’s consider you have scored low mark in the examination and your teacher asked you to get a signature in the copy of your mark sheet from your parent.

But getting signature for a low mark sheet is not easy, so you are changing the marks in the mark sheet copy and get a sign from your parent.

In this scenario, changing the marks in the copied mark sheet won’t affect in the real mark sheet and at the end you are happy and the teacher also happy for completing the work.

If you get caught then that is out of scope in this example.

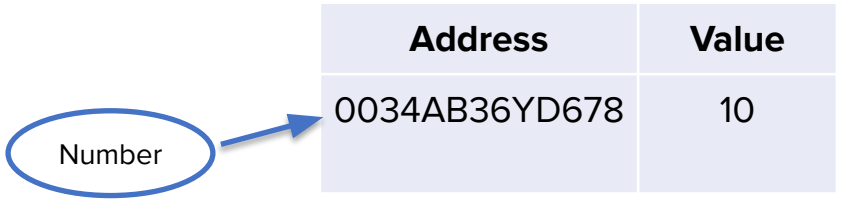
Pass by Value

- Below are the data types which support pass by value,
- Number
- String
- Boolean

- Let me explain how pass by value works in terms of memory allocation in javascript.

- Consider a variable with name **number** and assigning value as **10**

let number = 10



Pass by Value

- In computers the variables are stored in memory with address and value pairs and the variable number is pointing to the address.
- Let us assign the variable number to variable newNumber

Let newNumber = number

Pass by Value

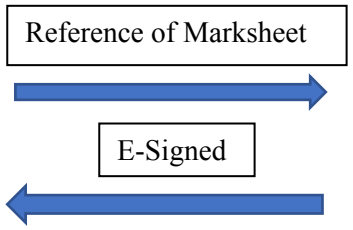
- Now both number and newNumber are pointing to same memory address.
- Let me change the add value of newNumber with 12. In this case both are pointing to same address and that is holding value 10, so changing the value in newNumber will change the value in number, but here the primitive data type comes into picture because they are immutable so the changes should not happen.

```
newNumber = newNumber + 12
```

Address	Value
0034AB36YD678	10
00HJ98FH54RT9	22

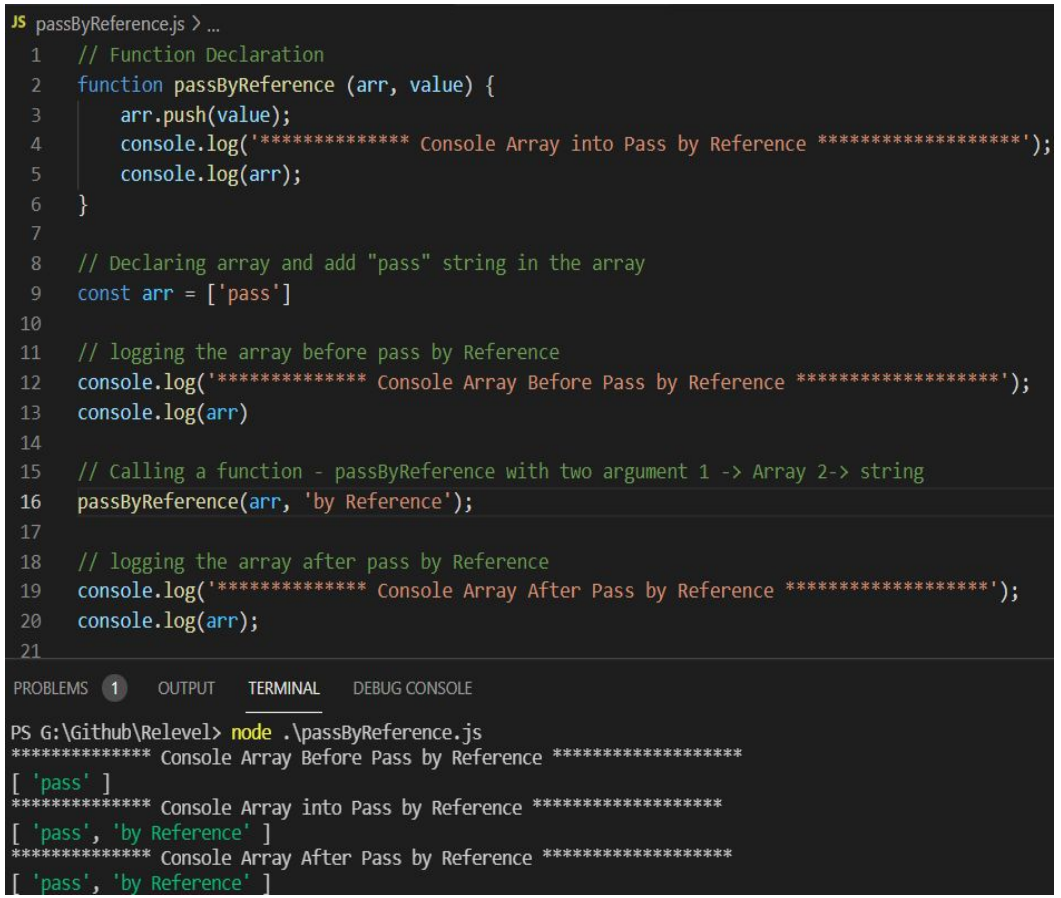
Pass by Reference

- A function is called by passing the reference of the variable and any change in the variable inside the function will make a change in the value of the outside variable.
- I hope this is also bit confusing, let me come up with an similar example of pass by value.
- Let's consider you have attended an online examination and scored low mark and your teacher asked you to get a e-signature in the original mark sheet from your parent which was uploaded in the school website.
- But getting signature for a low mark sheet is not easy, but you have convinced your parent and got a e-sign from your parent. In this scenario, your parents have signed the mark sheet in their home but that changes are reflected in the original marksheet.
- Marksheet is in the school Database but parents are opening the same marksheet in their home and e-signing by referring the marksheet.



Programming Example

- In this example we have declared an array with string ‘pass’ and declared one function called **passByReference**, it should ask two argument one is array and second one is string.
- Here we are passing the array having one index and adding the string ‘by Reference’ in the array inside the function.
- But changing the variable inside the function make a change in the outside function because we are passing the reference of an array inside the function.
- Here the Non primitive data type comes in the picture because the value inside an array is mutable. Once it is initialized, we can change the value, adding the value, deleting the value inside an array.



Programming Example

- Below are the data types which support pass by reference,
  - ➢ Object
  - ➢ Array

- Let me explain how pass by reference works in terms of memory allocation in javascript.
- Consider a object variable with name **obj** and having property name as javascript and score as 100

  let obj = {name: 'javascript', score: 100}

Address	Value
0034AB36YD678	{name: 'javascript', score: 100}

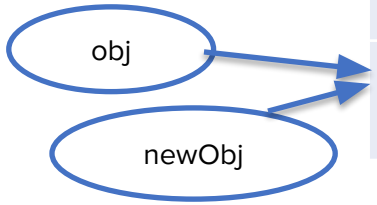
In computers the variables are stored in memory with address and value pairs and the variable obj is pointing to the address.

Let us assign the variable obj to variable newObj

Let newObj = obj

Programming Example

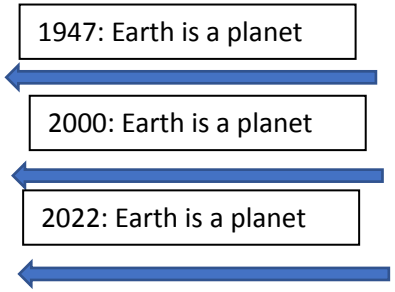
- Now both obj and newObj are pointing to same memory address.
- Let me change the newObj property score as 90. In this case both are pointing to same address and that is holding value 10, so changing the property in newObj will change the property in obj since these are non primitive data type.



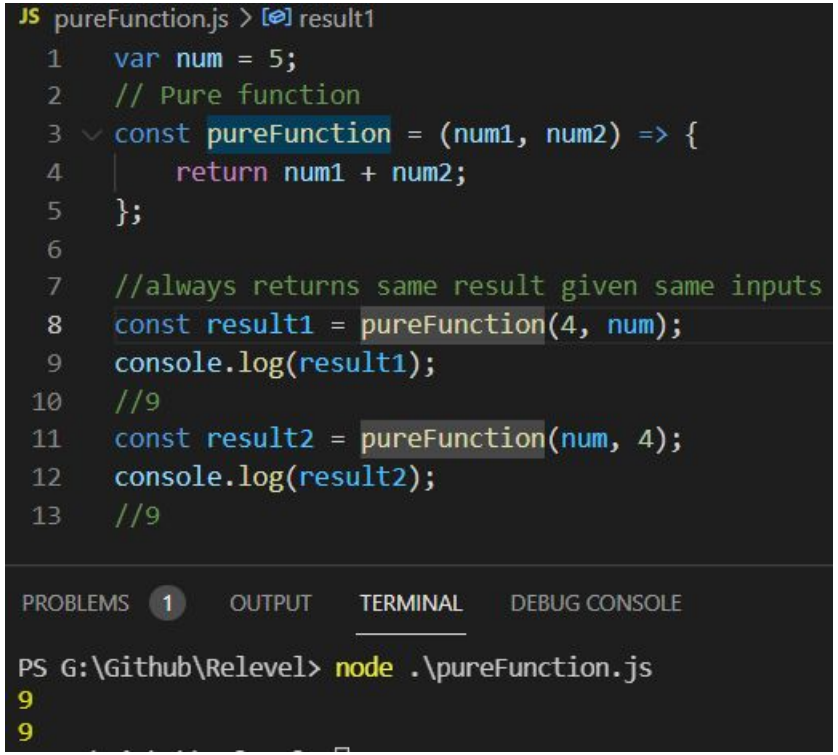
Address	Value
0034AB36YD678	{name: 'javascript', score: 100}

Pure Function

- Pure function does not change the state of variables out of its scope and it will always return same output if we pass the same input multiple number of times.
- The above definition is little bit hard to understand, so let me explain this with example.
- This real time example will show whenever you are studying about our planet earth every time the statement ‘Earth is a planet’ is same.



Programming Example



Lets quickly jump into the coding example, here the variable num is declared outside of the function scope ie. num is a global variable and the function is not mutating the variable.

So every time we call the function getting the expected output and it has no side effects.

Impure Function

- Impure function will change the state of variables out of its scope and it will always return different output if we pass the same input multiple number of times.
- The above definition is little bit hard to understand, so let me explain this with example.
- This above real time example will show whenever you are studying about pluto in school days we heard that pluto is the 9th planet in our solar system but now pluto is not a planet. It shows that every time it is changing and it has side effects.

1930: pluto is a 9th planet

2022: pluto is not a planet

Programming Example

impureFunction.js > ...

```
1   //Impure function
2   let mutateNum = 0;
3   const impureFunction = (num) => {
4     return (mutateNum += num);
5   };
6
7   //returns different result given same inputs
8   const result1 = impureFunction(5);
9   console.log(result1);
10  //5
11  const result2 = impureFunction(5);
12  console.log(result2);
13  //10
14  console.log('mutateNum', mutateNum);
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```
PS G:\Github\Relevel> node .\impureFunction.js
5
10
mutateNum 10
```

Lets quickly jump into the coding example, here the variable mutateNum is declared outside of the function scope

ie. mutateNum is a global variable and the function is mutating the variable every time when we are calling the function getting the different output and it has side effects.

Comparison

Pure Function	Impure Function
It has no side-effects	It may have side-effects
It will return same output if same arguments are passed how many times it executes	it will return different output if same argument passed on multiple times
It will always returns something	It may take effect without returning anything
It is useful in some use cases	It is useful in some use cases
It has no side-effects	It may have side-effects

Closure

- Closure is one of the important concept in Javascript. It is widely discussed concept in javascript world and still confusing concept. Let's understand closure in very simple ways.
- Closure is a function having access to the parent scope, even after the parent function has closed.
- Lets quickly check this definition by splitting the definition
    - Closure is a function having access to the parent scope => which means a function(Parent function) is returning a function(child function)
    - even after the parent function has closed => the variable which are present in the parent functions are accessible from the child function after called the parent function.

Programming Example

```
JS closure.js > counter
1   function counter () {
2       let count = 0;
3
4       return function (value) {
5           count += value;
6           console.log(count);
7       }
8   }
9
10  const counterCall = counter();
11  counterCall(1);
12  counterCall(2);
13  counterCall(3);
```

```
PROBLEMS 1   OUTPUT   TERMINAL   DEBUG CONSOLE
PS G:\Github\Relevel> node .\closure.js
1
3
6
```

- In this example function counter is returning the function which is called as anonymous function and that function is referring the variable which is available in the parent function.
- In the 10th line we are calling the function counter and storing the anonymous function in the variable, then calling the anonymous function will make the variable in the parent function available for child function.
- This concept is used to achieve the private function in javascript.

Programming Example

```
for (var i = 0; i < 4; i++) {
    setTimeout(function() {
        console.log(i);
    }, i * 1000);
}

// Answer: 4 4 4 4

for (let i = 0; i < 4; i++) {
    setTimeout(function() {
        console.log(i);
    }, i * 1000);
}

// 0 1 2 3
```

- Let me explain you one more example with setTimeout function,
- In 1st for loop the output should be 4 for every execution, why because javascript won't wait for the setTimeout to be executed so the loop will execute and move the settimeout function into execution and loop stopped at the end 'i' vaue is 4 and in the var keyword every 'i' in the log is pointing to the same memory location so the value is logging as 4
- In 2nd for loop the let keyword will allow to make a copy of value and pass into the settimeout function, so every execution in for loop has a copy of 'i' so it is logging the expected output 0, 1, 2, 3

Try this Question

Implement the mutation using object in javascript.

```
Given object const obj = {
            Javascript: 'hard',
            Java: 'easy',
            Python: 'medium'
              }
```

Change property of Javascript from ‘hard’ to ‘easy’

Solution: https://jsfiddle.net/saravananslb/fhr4Lq5o/1/

Explanation: Object property can be access using ‘.’ and directly assign the value.

Try this Question

Write a program array containing string and sort the string array in descending order. (Input array: [‘a’, ‘b’, ‘c’, ‘z’, ‘h’])

Solution: https://jsfiddle.net/saravananslb/sv46jLdf/1/

Explanation: sort function will accept comparator passing localeCompare and comparing the string.

Try this Question

Write a program to find the particular element in an array using find function. (Input array: [‘a’, ‘b’, ‘c’, ‘z’, ‘h’]) find ‘c’

Solution: https://jsfiddle.net/saravananslb/1mjnqawh/1/

Explanation: find function will return the **value** if the array satisfies the given condition else **undefined**.

Try this Question

What is the output of the below code?

```
// Ordinary function for multiplication using 3 arguments
const multiply = (a, b, c) => {
    return (a * b * c)
}

// Currying function for multiplication
const currymultiply = (multi) => {
    return (a) => {
        return (b) => {
            return (c) => {
                return multi(a, b, c);
            }
        }
    }
}

const multiplication = currymultiply(multiply);
console.log(multiplication(1)(2)(3));
```

Solution: output is 6

Explanation: currymultiply function is called with multiply function as argument and invoking the series of function with 1, 2 and 3, at last it is returning the argument of 1st function value as function and referring the series of value passed from the function as an argument of multi function

Try this Question

What is the output of the below code?

```
const obj = {
    Javascript: 'hard',
    Java: 'easy',
    Python: 'medium'
}

const newObj = obj;

newObj.Javascript = 'easy';
obj.Javascript = 'very easy';
console.log(newObj.Javascript);
```

Solution: very easy

Explanation: Reference of obj is assigned to newObj, so any change in obj or newObj will make a change in both variable because both variables are referring a same memory location.

Try this Question

Write a function which will assign object to another variable change of object property will not affect the object property of another variable.

```
const obj = {
    Javascript: 'hard',
    Java: 'easy',
    Python: 'medium'
}

// Destructuring the object
const newObj = {...obj};

newObj.Javascript = 'easy';
obj.Javascript = 'very easy';
console.log(newObj);
console.log(obj);
```

Explanation: De-structuring the object will make an copy of the object, so changing the property of one object won't affect the property of another object.

Try this Question

What is the output of the below code?

```
const arrayOfOddNumbers = [1, 3, 5];
arrayOfOddNumbers[100] = 199;
console.log(arrayOfOddNumbers.length);
```

Solution: 101

Explanation: The reason for this solution is as follows: JavaScript places empty as a value for indices 3 - 99. Thus, when you set the value of the 100th index

Assignment

1) Write a program to multiply the value in the given array and return a result (use array functions)

Array = [1, 2, 3, 4, 5, 6, 7]

Assignment

1) Write a JavaScript program to sort by id an array of JavaScript objects.

```
Object =[ {
        Id: 45,
        Name: 'ram'
    },{
        Id: 4,
        Name: 'raju'
    },{
        Id: 90,
        Name: 'kumar'
    }]
```

Assignment

1) Write a program to sort an integer array with custom number as reference.

Sample Input:

2356481790

0 1 2 3 9 5 6 2 8 1 9

Sample Output:

2 2 3 5 6 8 1 1 9 9 0

Explanation:

- Our current integer is not 0123456789 it is 2356481790 as per the given input format.
- So the integer should be sorted in the given order and the output is 2 2 3 5 6 8 1 1 9 9 0

Solution link - https://jsfiddle.net/saravananslb/oLpwmy4v/

Thank You