Rahul Aher

Page No. 1

Date

JavaScript - Day 10

● MOTIVATION :- Keep data separate from functions.

* FUNCTIONAL PROGRAMMING :-

● PURE ~~FUNCTION~~ FUNCTIONS :-

→ The function should always return the same output for a given output.

→ The function should not modify anything outside of itself. (No side effect).

1. No side effects.

Example of a function with side effect.

```
const obj= {
        name: 'Rahul',
        language: 'JavaScript'
    }
function modifyName (objInput) {
        ObjectInput.name = 'John';
  }
modifyName (obj)
console.log (obj)

 > {name: '~~Rahul~~ John", language: 'JavaScript' }
```

→ Since this function is modifying something outside of itself (obj) it produces side effects and hence is not a pure function.

→ Imagine multiple functions modifying this obj, it can get really hard to keep track of current value of obj and who modified it.

Raheel Aher

Page No. ②

→ Let's change the same example to have no side effects.

```
function modifyName (objInput) {
   let objTemp = object.assign ({}, objInput);
      objTemp.name = 'John';
      return objTemp;
}
```

→ We now created a local copy inside modifyName and modified that instead. So obj outside does not change.

2. Return same output for same input.

```
function allNum(num1, num2) {
   return num1 + num2;
}
allNum(8,2);
```

→ If you run this function a 100 times, it's always going to return 8+2 ⇒ 10. That means given the same input it always return the same output.

* WHY IS THAT IMPORTANT ?

→ Well, it makes our code more predictable right?

→ So, are side effects bad? I mean at some point a function will have to interact with the browser, manipulate the DOM etc.

Scanned by CamScanner

Rahul Aher

→ To write some meaningful code we will end up having side effects, but can minimize them and organise our code well enough so that it's predicatable and we know what is happening where.

Scanned by CamScanner