# Functional Programming

Motivation : keep data separate from functions ♥

## Pure functions

* The function should always return the same output for a given output
* The function should not modify anything outside of itself. (No side effects)

@CodeWithSimran

1. No side effects.

Example of a function with side effects

```
const obj = { name : 'Simran', language : 'JavaScript' }

function modifyName( objInput )
 {
    objInput.name = 'John';
 }

modifyName(obj)
console.log(obj)
▶ {name : 'John', language : 'JavaScript'}
```

@codeWithSimran

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.

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)
```

@codeWithSimran

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 returns 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

@codeWithSimran

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 predictable and we know what is happening where.