Functional Programming in JavaScript

by N C Patro

Saturday 21st October 2017

What is Functional Programming

In computer science, functional programming is a programming paradigm—a style of building the structure and elements of computer programs—that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. (taken from Wikipedia)

Different programming paradigm or pattern

- Procedural Programming
- Object Oriented Programming
- Meta Programming

Major procedural programming languages are COBOL, BASIC, C, ADA and GO

Functional programming pattern will be more close to OOP.

Metaprogramming have the ability to treat programs as their data

Declarative pattern

Declarative pattern focuses on what the program should accomplish without specifying how the program should achieve the result.

Functional programming follows declarative pattern.

Imperative pattern focuses on describing how a program operates.

```
2    for (var i = 0; i < meetups.length; i++) {
3        meetups[i].lastUpdate = new Date();
4    }
5
6    meetups.map((u)=>{
7        u.lastUpdate = new Date();
8        return u;
9    });
```

Mathematical Function or Pure Function

In mathematics, a function is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.

In functional programming, these kind of functions called pure function which only depends on the passed input data to the function and does not alter the data except the returned data.

Math.random() is not pure function because it always return new value on each call

Math.min(1,2) is example of pure function which always returns same value with same set of inputs

Why Functional Programming

Its pure function, provides confidence of not changing stuff outside of its scope.

Its reduces the complexity, need not to worry about how it is doing it, focus will be only on what it is doing.

Ease of testing, because it does not depend on state of the application and result verification also will be easy.

It makes the code more readable.

Functional programming makes code easier to understand.

Array Functions

Find

Filter

Map

Reduce

Every

Some

Array Function Examples

```
2   let meetups = [{name:'JS',isActive:true,members:450},
3   {name:'Angular',isActive:true,members:900},
4   {name:'Node',isActive:false,members:900}];
5   let activeMeetups = [];
6
7   for (let i=0; i<meetups.length; i++ ) {
8     let m = meetups[i];
9     if (m.isActive) {
10      activeMeetups.push(m);
11    }
12  }
13  console.log(activeMeetups);
```

Array Function (Filter method)

```
14    let activeMeetupsFP = [];
15    activeMeetupsFP = (meetups.filter((m)=>{
16      return m.isActive;
17    }));
18    console.log(activeMeetupsFP);
```

Array Function (find method)

```
let meetup;

for (let i = 0; i < meetups.length; i++) {
  if (meetups[i].name == 'JS' ) {
    meetup = meetups[i];
  }
}
console.log(meetup);


let meetupFP;
meetupFP = meetups.find((m) => {
  return m.name === 'JS';
});
console.log(meetupFP);
```

Function Chaining

```
1
2   let sumFPChain = meetups.filter((m) => {
3           return m.isActive;
4       })
5       .map((m) => {
6           return m.members - (0.1 * m.members);
7       })
8       .reduce((acc, m) => {
9           return acc + m;
10      }, 0);
11  console.log(sumFPChain);
12
```

Libraries to support FP

There are libraries which provide utility functions to make code more declarative.

RamdaJS

UnderscoreJS

lodash

First Class Functions

A programming language is said to have First-class functions when functions in that language are treated like any other variable. For example, in such a language, a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.

Callbacks and promises are good example passing function to another function.

```
2    var isActive = (meetup) => {
3        return meetup.isActive;
4    }
5    meetups.filter(isActive);
```

Currying

Function can return function as well.

currying is the technique of translating the evaluation of a function that takes multiple arguments into evaluating a sequence of functions, each with a single argument.

Ramda JS also provides curry function.

```
let byName = (name) =>{
    return (meetup) =>{
        return meetup.name == name;
    }
}
meetups.find(byName('JS'))
```

Function Composition

Combining functions, one function's output feed to another function.

Ramda pipe is a good example of function composition

```
let byName = (name) =>{
    return (meetup) =>{
        return meetup.name == name;
    }
}
let isValid = isFinite(meetups.find(byName('JS')).members);
```

Side Effects

Function or expression is said to have a side effect if it modifies some state outside its scope or has an observable interaction with its calling functions or the outside world besides returning a value.

```
const scheduleMeetup = (date, place) => {
    meetup.date = date;
    meetup.place = place;
    if (meetup.members < 5)
        meetup.isActive = false;
}

const publishMeetup = () => {
    if (meetup.isActive)
        meetup.publish = true;
}
```

Immutability

Immutability is important to make sure one function does not change the original data rather than should return new copy of the data after manipulation.

Like if Array and Objects are passing around multiple function and we are not maintaining immutability then functions might not get the original copy of the array and object.

It is really hard to debug if something goes wrong in case of mutable object and array.

Mutability is not bad but it should be when to have it.

Immutable Libraries

JavaScript does not provide anything to make the object and array Immutable.

There are libraries which can help us to achieve it.

Seamless-immutable

Immutable JS

Thank You