Explore Core Features of Functional Programming

Adhithi Ravichandran

Software Consultant | Author | Speaker

@AdhithiRavi | www.adhithiravichandran.com

Eric Elliott

FP is the process of building software by composing pure functions, avoiding shared state, mutable data, and side-effects.

Overview

- Setup
- First Class Functions
- Pure Functions
- Side Effects
- Closures
- Real-world Example

Setup

Setup

Install Node (npm is included with Node.js installation)

Install npm as package manager

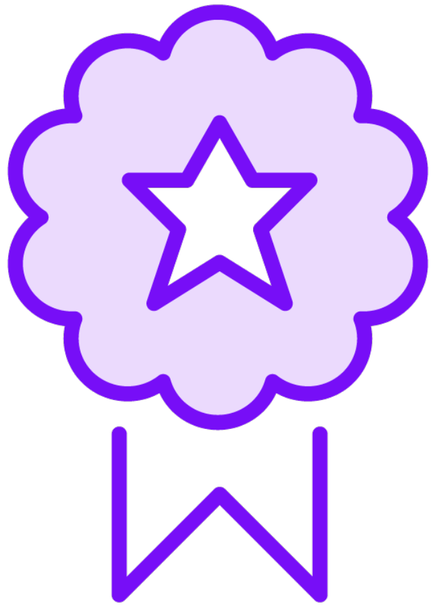
Download code from exercise folder or clone GitHub repo:
https://github.com/adhithiravi/Functional-JS

First Class Functions

Functions in FP are treated as a data type and can be used like any other value!

First Class Functions

A function is treated like a data type!



JavaScript supports first class functions by default!

First class functions in JS:

- Can be assigned to regular variables
- Passed as arguments to functions
- Returned as results of functions
- Included in any data structures

First class functions in JavaScript

Functions are treated like variables – First class citizens!

First Class Functions

Reusable

Flexible

Predictable

Testable

Readable

Maintainable

Pure Functions

Pure Function

1. Given the same input, always returns the same output.
2. Produces no side-effects.

Pure Function

```
const topic = (t) => `You are learning ${t}`;
```

Side-effect is simply an interaction with the outside world, outside of the function.

Side Effects

Side Effects

Changing or mutating the input

Querying or updating the DOM object

Making API calls via network (fetch/xhr)

More Side Effects

Logging

Reading/writing to a file

Reading from a global state

A pure function should be side-effect free!

Eric Elliott

FP is the process of building software by composing pure functions, avoiding shared state, mutable data, and side-effects.

Closure

Closure

A closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures

Demo

Closure in JavaScript

A Real-world Example of Functional Programming

Recap

- Closures
- First Class Functions
- Pure Functions
- Side Effects

Redux

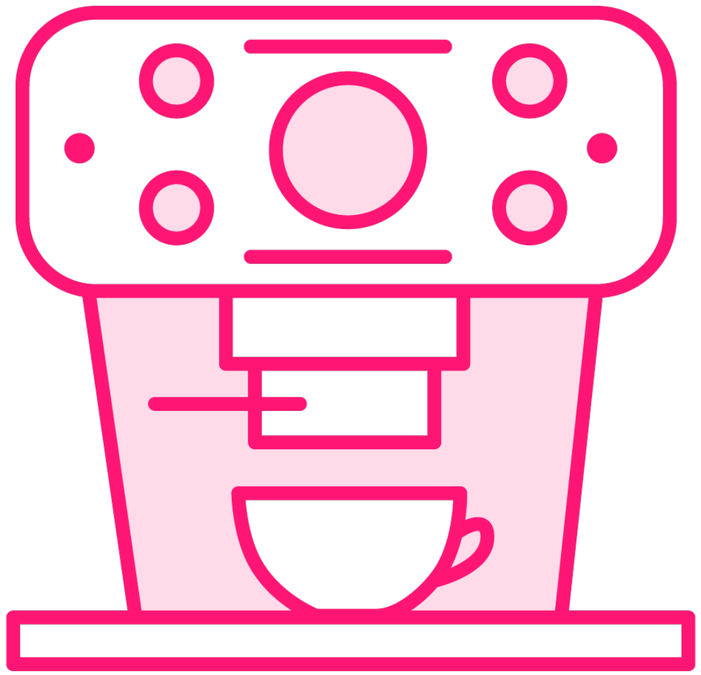
Redux is a real-world example, that uses functional programming paradigms and pure functions.

```
function(state, action) => newState
```

Reducers in Redux

It takes in an old state and action and brews a new state!

Reducers in Redux Are Pure Functions



Reducers are like coffee makers

Coffee powder + water = Fresh cup of coffee

Functions that take in current state (coffee powder) and actions (water) and brew a new state (fresh cup of coffee)!

```
function courseDetails(state =
initialState, action) {
   switch (action.type) {
      case RATE_COURSE:
         return {
           ...state,
           rating: action.rate,
         };
      default:
         return state;
   }
}
```

- Redux follow functional programming paradigms.
- A reducer in Redux should be a pure function.
- No side-effects.
- The reducer here, will never mutate state inside it.
- It uses the Spread (...) operator to return a new copy of the state instead!
- *courseDetails()* will take the current state and action as the parameters and return a new state.

Up Next:

Functional Composition