Functional and Reactive Workshop

HTTPS://GITHUB.COM/MATTPODWYSOCKI/JSCONF.CO-2015-WORKSHOP

What is Functional Programming?

In functional programming, programs are executed by evaluating *expressions*, in contrast with imperative programming where programs are composed of *statements* which change global *state* when executed. Functional programming **typically avoids using mutable state**.

Functional programming requires that functions are *first-class*, which means that they are treated like any other values and can be passed as arguments to other functions or be returned as a result of a function. Being first-class also means that it is possible to define and manipulate functions from within other functions

Functional Programming in JS?

```
// Accept a function
function exec (cmd) {
  return cmd();
}

// Return a function
function addPartial(x) {
  return y => x + y;
}
```

Going from Imperative…

```
const values = ['1','foo','3,'4','bar'];
let result = 0;
for (var i = 0; i < values.length; i++) {
  let val = parseInt(values[i], 10);
  if (!Number.isNaN(val) {
    result += val;
  }
}
console.log(result);
```

To Functional…

```
const values = ['1','foo','3,'4','bar'];
let result = values
   .map(x => parseInt(x, 10))
   .filter(x => !Number.isNaN(x))
   .reduce((sum, x) => sum + x);

console.log(result);
```

What is Reactive Programming?

Merriam-Webster defines reactive as *"readily responsive to a stimulus"*, i.e. its components are "active" and always ready to receive events. This definition captures the essence of reactive applications, focusing on systems that:

react to events

the event-driven nature enables the following qualities

react to load

focus on scalability by avoiding contention on shared resources

react to failure

build resilient systems with the ability to recover at all levels

react to users

honor response time guarantees regardless of load

From Functional…

```
const values = ['1','foo','3,'4','bar'];
let result = values
   .map(x => parseInt(x, 10))
   .filter(x => !Number.isNaN(x))
   .reduce((sum, x) => sum + x);

console.log(result);
```

To Reactive…

```
const values = ['1','foo','3,'4','bar'];
let result = Rx.Observable.from(values)
  .delay(1000)
  .map(x => parseInt(x, 10))
  .filter(x => !Number.isNaN(x))
  .reduce((sum, x) => sum + x);
let subscription = result.subscribe(
  x => console.log(x)
);
```

Reactive Demos

REACTIVE APPLICATIONS IN PRACTICE

Go Build Something Amazing!

HTTPS://GITHUB.COM/MATTPODWYSOCKI/JSCONF.CO-2015-WORKSHOP