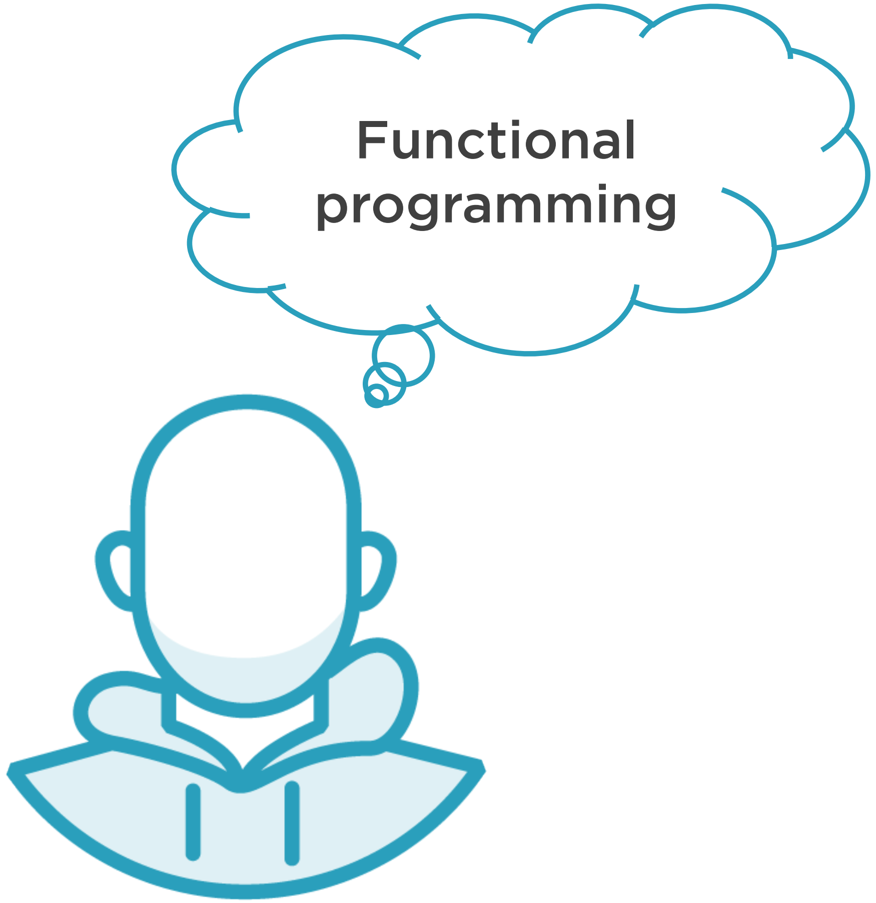
Declare What You Mean

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com


Functional programming

Sunware PitStop

```
if(isHousehold){
    …
    let birthdate = members[0].birthdate;
    …
}
```

```
for (var i = 0; i < members.length; i++){
    if (members[i].id == currentId) {
        birthdate = members[i].birthdate;
    }
}
```

All I wanted was the birthdate for the user I was working with.

Can that function be written in declarative style?

Array Functions

```
for(let i = 0; i < 10; i++){
    <some code>
}
```

How

```
for(let i = 0; i < users.length; i++){
    let u = users[i];  1
    if(u.isActive){  2
        activeUsers.push[u];  3
    }
}
```

What

Finding a Specific Item

```
for (let i = 0;
    i < users.length;
    i++){
   if(users[i].id === id){
      user = users[i];
   }
}
```

```
users.find((u) => {
    return u.id === id;
});
```

```
for (let i = 0;
    i < users.length;
    i++){
   if(users[i].id === id){
      user = users[i];
   }
}
```

```
users.find((u) => {
    return u.id === id;
});
```

Checking Elements in an Array

```
let approved = true;

for(let i = 0;

    i < p.length;

    i++){


   if(!p[i].approved){

      approved = false;

    }

}
```

```
products.every((p) => {
    return p.approved;
});
```

```
let approved = false;

for(let i = 0;

    i < p.length;

    i++){


    if(p [i].approved){

        approved = true;

    }

}
```

```
products.some((p) => {
     return p.approved;
});
```

Creating a New List

```
let onSale = [];
for(let i = 0;
    i < p.length;
    i++){

    if(p[i].onSale){
        onSale.push(p[i]);
    }
}
```

```
products.filter((p) =>{
      return p.onSale;
});
```

Updating Items in a List

```
for(let i = 0;

    i < users.length;

    i++){

    users[i].upgraded = true;

}
```

```
users.map((u) => {
    u.upgraded = true;
    return u;
});
```

Reducing an Array

```
for(let i = 0;
    i < nums.length;
    i++){
    sum += numbers[i];
}
```

```
nums((accumulator, n) => {
      return accumulator + n
}, 0);
```

```
for(let i = 0;
    i < nums.length;
    i++){
    sum += numbers[i];
}
```

```
nums((accumulator, n) => {
      return accumulator + n
}, 0);
```

```
[1,2,3].reduce((acc, n) => {return acc + n;}, 0)
```

Iteration	Accumulator	Array Item	Final Value
1	0	1	1

```
[1,2,3].reduce((acc, n) => {return acc + n;}, 0)
```

Iteration	Accumulator	Array Item	Final Value
1	0	1	1
2	1	2	3

```
[1,2,3].reduce((acc, n) => {return acc + n;}, 0)
```

Iteration	Accumulator	Array Item	Final Value
1	0	1	1
2	1	2	3
3	3	3	6

Function Chaining

Reduce Value by 1

Sum All Numbers Divisible By 3

```
let sum = 0;
let numbers = [2, 4, 6, 10, 16]
for (let i = 0; i < numbers.length; i++){
  numbers[i] = numbers[i] - 1;

  if (numbers[i] % 3 === 0) {
      sum += numbers[i];
  }
}
// sum = 27
```

```
let reduced = numbers.map((n) => { return n - 1; });
let divisible = reduced.filter((n) => {
    return n % 3 === 0
});
let sum = divisible.reduce((acc, n) => {
    return acc + n;
}, 0);
// sum = 27
```

```
let sum = numbers.map((n) => { return n - 1; })
  .filter((n) => { return n % 3 === 0 })
  .reduce((acc, n) => { return acc + n; }, 0);
```

```
let sum = numbers.map(subtractOne) ①
            .filter(isDivisibleBy3) ②
            .reduce(add, 0); ③
```

Libraries

RamdaJS

lodash/fp

```
R.drop(2,[‘a’,‘b’,‘c’,‘d’]);


R.dropLast(2,
    [‘a’,‘b’,‘c’,‘d’]
);
```

◀ [‘c’, ‘d’]

◀ [‘a’,’b’]

R.

_.

Flattening an Array

```
orders = [[1.25], [5.00, 10.23], [2.00]];
R.flatten(orders);


// [1.25, 5.00, 10.23, 2.00]
```

Array Logic

```
!products.some((p) => {return p.approved;});
```

```
R.none((p) => {return p.approved;}, products);
```

Selecting Values From Array

```
for(let i = 0;
    i < users.length;
    i++){
    ids.push({
        id: users[i].id
    });
}
```

```
R.pluck(‘id’, users);
```

Eliminating Values

```
R.reject((x) => {
    return x % 2 === 0;
}, [1, 2, 3, 4])
```

```
[1,3]
```

Objects

Eliminate Properties

```
const newUser = {
  firstName = user.firstName,
  lastName = user.lastName,
  id = user.id
}
```

Good for small objects

```
Object.keys(user).forEach((k) => {
  if (k !== 'password') {
      newUser[k] = user[k];
  }
});
```

Good if not eliminating too many

```
R.omit([‘password’], user);
```

```
R.pick([‘id’,
    ‘firstname’,
    ‘lastname’], user);
```

Dealing with **`undefined`**

Guarding Against Undefined

```
if(users && users.currentUser &&
   users.currentUser.address){

      const address = users.currentUser.address;

      const street = `${address.line1} ${address.line2}`;

      ...

}
```

```
const addr = R.path([‘currentUser’,‘address’], users);
```

Returns value or undefined

```
const address = R.pathOr({}, [‘currentUser’, ‘address’],
users);
```

Return value or empty object {}

Selecting Data

```
select u.firstName, u.lastName
from users u;
```

SQL: The ultimate declarative language

Project

```
let pets = [{ id: 1, name: 'Sassy', type: 'Cat' },
            { id: 2, name: 'Elmo', type: 'Cat' },
            {id: 3, name: 'Chocolate Chip', type: 'Dog'}];

R.project(['name', 'type'], pets);
```

```
[{"name": "Sassy", "type": "Cat"}, {"name": "Elmo", "type":
"Cat"}, {"name": "Chocolate Chip", "type": "Dog"}]
```

```
R.map((p) => {
    return {
        name: p.name,
        type: p.type
    };
}, pets);
```

Can also be accomplished with map

Summary

Can you make it declarative?

```
let member = members.find((m) => {
    return m.id === id;
});

return member.birthdate;
```

<u>What</u> Not How