Bizzare Javascript Paradigm(s)

•••

@ncthis . nadeesha.me

… or, why my mother warned me about javascript and I didn’t listen.

Why was javascript bad?

I’m glad you asked.

```
''          ==    '0'           // false

0           ==    ''            // true

0           ==    '0'           // true

false       ==    'false'       // false

false       ==    '0'           // true

false       ==    undefined     // false

false       ==    null          // false

null        ==    undefined     // true

" \t\r\n"   ==    0             // true
```

So, what’s the case for Javascript?

Language with a few rules,
that you can break whenever you want.

The power of javascript:

Being a Lego block for things that are more awesome

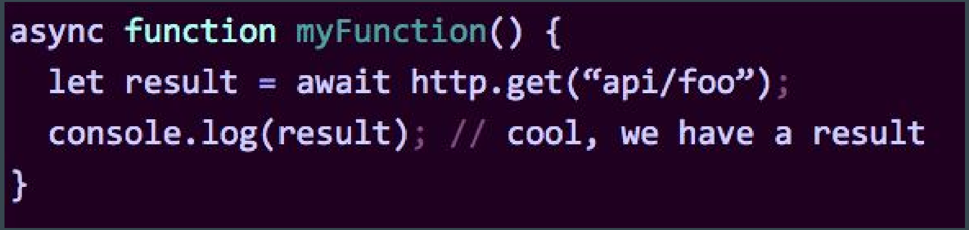
BYOP
(Bring your own paradigm)

The maturing JS ecosystem:

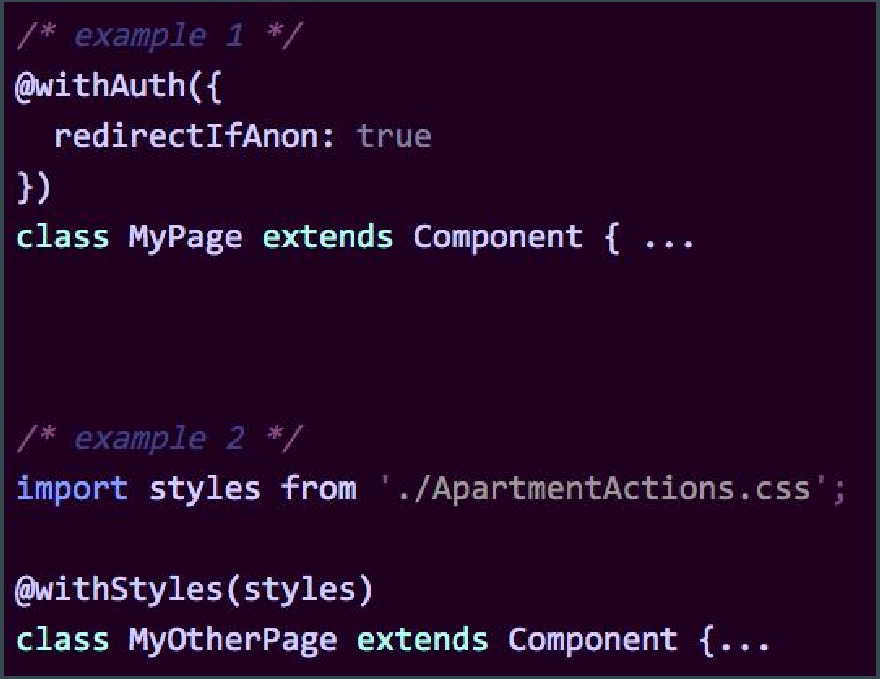
BABEL

Transpilation

… because waiting for ES* is too hard.



```
async function myFunction() {
  let result = await http.get(“api/foo”);
  console.log(result); // cool, we have a result
}
```



```
/* example 1 */
@withAuth({
  redirectIfAnon: true
})
class MyPage extends Component { ...



/* example 2 */
import styles from './ApartmentActions.css';

@withStyles(styles)
class MyOtherPage extends Component {...
```

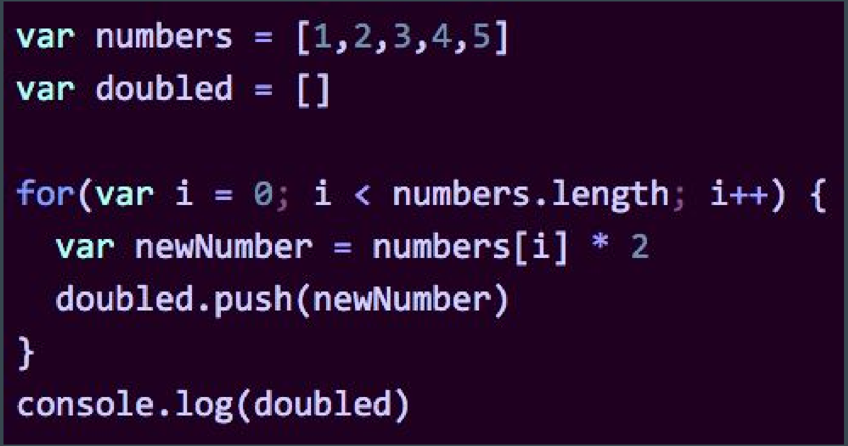
UIs as functions

```
var Comment = React.createClass({
  render: function() {
    return (
      <div className="comment">
        <h2 className="commentAuthor">
          {this.props.author}
        </h2>
        {this.props.children}
      </div>
    );
  }
});

var CommentList = React.createClass({
  render: function() {
    return (
      <div className="commentList">
        <Comment author="Pete Hunt">This is one comment</Comment>
        <Comment author="Jordan Walke">This is *another* comment</Comment>
      </div>
    );
  }
});

/* https://facebook.github.io/react/docs/tutorial.html */
```

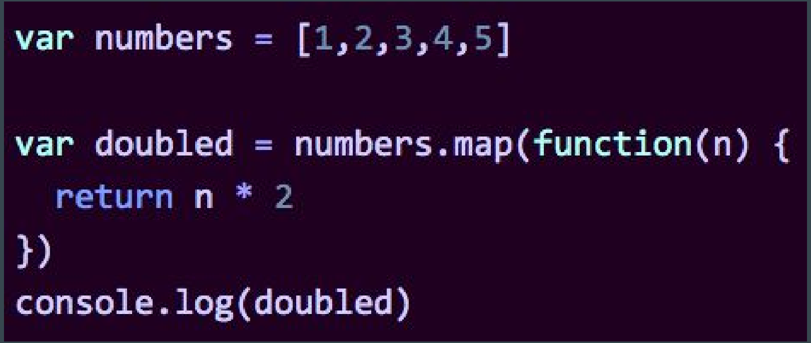
Better separation of UI concerns
Composable UI components
Better performance
Testability

Functional (reactive) programming

...what it means to have functions as first class citizens



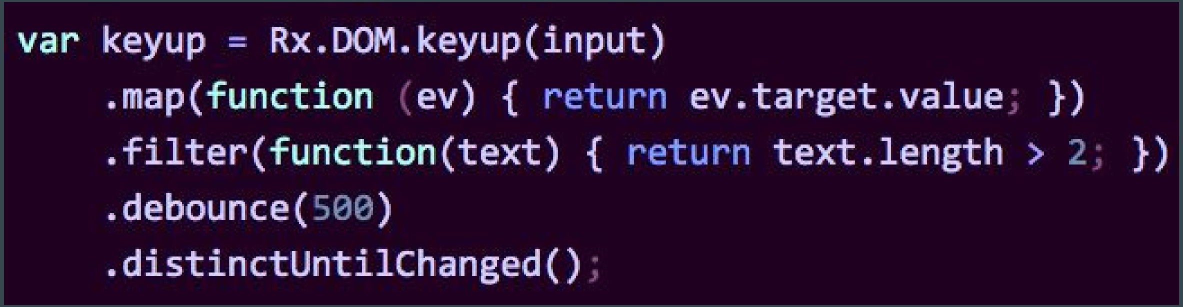
```
var numbers = [1,2,3,4,5]
var doubled = []

for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled)
```



```
var numbers = [1,2,3,4,5]

var doubled = numbers.map(function(n) {
  return n * 2
})
console.log(doubled)
```

Lodash: functional programming
RxJS: reactive extensions
Immutable.js: immutable data structures

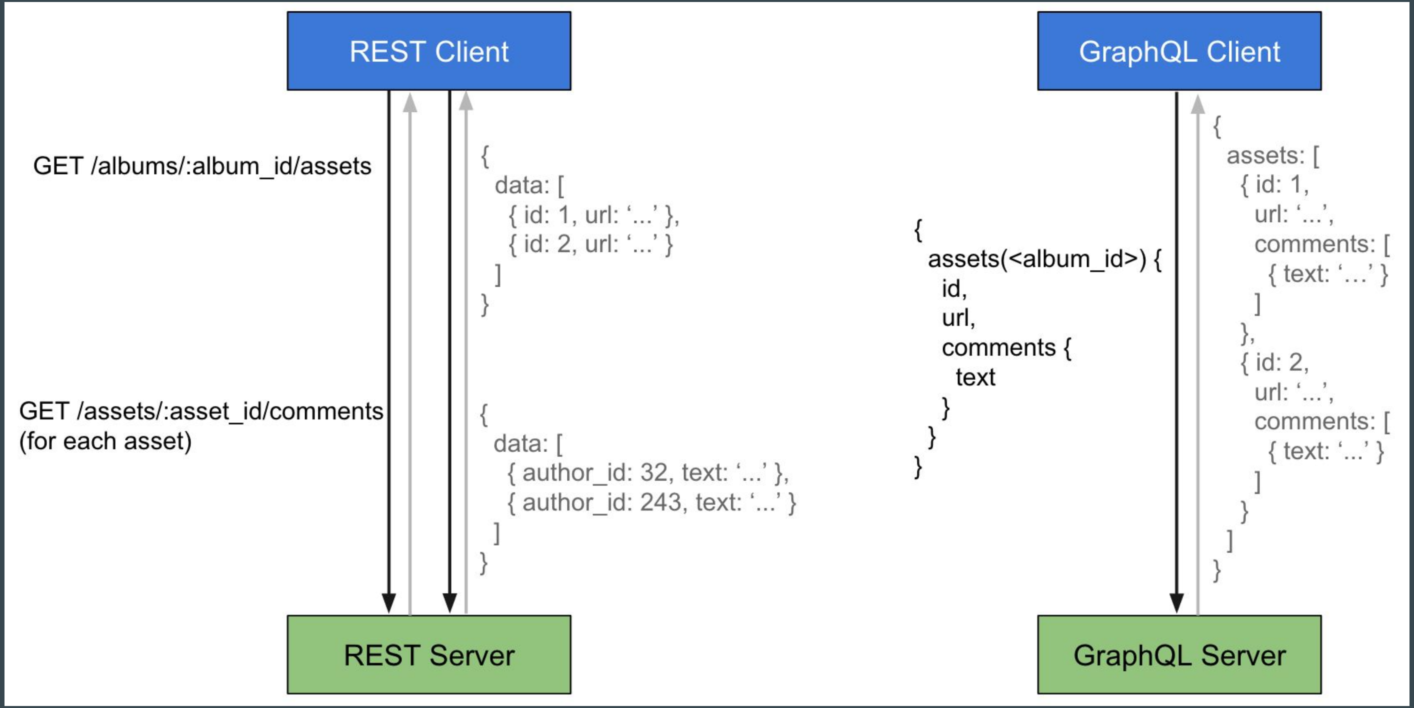
Declarative querying for an autocomplete



```
var keyup = Rx.DOM.keyup(input)
    .map(function (ev) { return ev.target.value; })
    .filter(function(text) { return text.length > 2; })
    .debounce(500)
    .distinctUntilChanged();
```

GraphQL/Falcor

… changing the way we REST



GET /albums/:album_id/assets

```
{
  data: [
    { id: 1, url: '...' },
    { id: 2, url: '...' }
  ]
}
```

GET /assets/:asset_id/comments
(for each asset)

```
{
  data: [
    { author_id: 32, text: '...' },
    { author_id: 243, text: '...' }
  ]
}
```

```
{
  assets(<album_id>) {
    id,
    url,
    comments {
      text
    }
  }
}
```

```
{
  assets: [
    { id: 1,
      url: '...',
      comments: [
        { text: '...' }
      ]
    },
    { id: 2,
      url: '...',
      comments: [
        { text: '...' }
      ]
    }
  ]
}
```

source: https://medium.com/chute-engineering/graphql-in-the-age-of-rest-apis-b10f2bf09bba

Declarative CRUD Ops
Query co-location

Slow death of framework adoption.
And rise of small, well-defined libraries.

It’s an exciting time to be a JS developer.

@ncthis . nadeesha.me