Callback-less Asynchrony

ES6, Generators, and the next wave of JavaScript development.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

http://bit.ly/nodevember-2015

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

A Brief Introduction

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Nicholas Young

@nicholaswyoung.com

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Nicholas Young

@nicholaswyoung.com

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

ECMAScript

June 1997 – June 2015

ECMAScript 6 / ES6 / ES2015

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

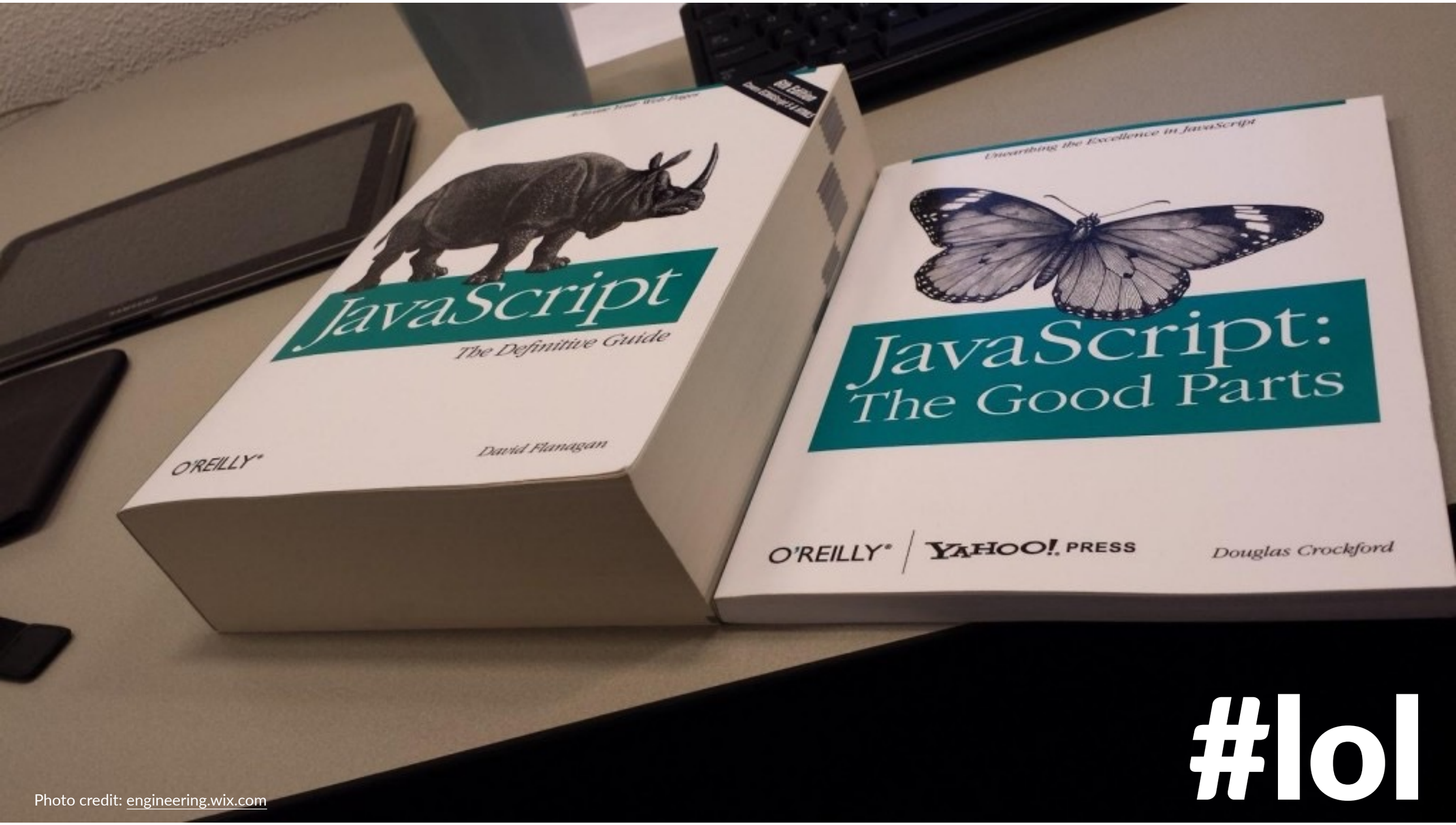


Photo credit: engineering.wix.com

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Flow Control

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri) {
  return JSON.parse(request.get(uri));
};
```

```
try {
  var data = getJSON('https://json.service');
} catch (e) {
  console.error(err);
}
```

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    done(null, JSON.parse(data));
  });
}
```

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    done(null, JSON.parse(data));
  });
}
```

Establishes a contract that is difficult to enforce
Doesn’t work with existing flow control constructs
Doesn’t handle errors as it should

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    try {
      done(null, JSON.parse(data));
    } catch(ex) {
      done(ex);
    }
  });
}
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    try {
      done(null, JSON.parse(data));
    } catch(ex) {
      done(ex);
    }
  });
}
```

Establishes a contract that is difficult to enforce
Doesn’t work with existing flow control constructs
Successfully handles synchronous errors in JSON.parse
The error callback could be called multiple times

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    try {
      data = JSON.parse(data);
    } catch(ex) {
      return done(ex);
    }
    done(null, data);
  });
};
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    try {
      data = JSON.parse(data);
    } catch(ex) {
      return done(ex);
    }
    done(null, data);
  });
};
```

Successfully handling all errors, to the best of our ability
Establishes a contract that is difficult to enforce
Doesn’t work with existing flow control constructs

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function parseJSON(json, done) {
  try {
    json = JSON.parse(json);
  } catch(ex) {
    done(ex);
  }
  done(null, data);
}
```

```
function getJSON(json, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    parseJSON(data, function(err, json) {
      if (err) return done(err);
      done(null, json);
    });
  });
}
```

async.js
step
futures
seq

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

What if async methods accepted simple arguments, returned values, and for the most part, still behaved like their synchronous relatives?

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Iteration Protocols

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Iteration Protocols

Are specifications that allow objects to customize their iteration behavior, such as what values are looped over in a for-of construct. Some built-in types use the standard iteration protocol, while others do not.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

The Iteration Protocol is a specification that allows iteration on any type that implements it's conventions:

1. Array
2. Map
3. String

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
var str = new String('Hello');

str[Symbol.iterator] = function( ) {
   return {
      next: function( ) {
         if (!this._first) return { done: true };
         this._first = false;
         return { value: 'goodbye', done: false };
      }
   };
};
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function makeIterable(values) {
  const index = 0;
  return {
    next: function() {
      return index < values.length ? {
        value: values[index++], done: false
      } : {
        done: true
      };
    }
  };
}

var iterable = makeIterable([1, 2, 3]);

console.log(iterable.next());
```

Generators

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Generators

Are a special kind of function that always returns a Generator Object. This generator object can be used to pass messages, process data, and enforce flow control. It adheres to the Iterable Protocol.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function* createId( ){
  var index = 0;
  while(index < 3) { yield index++; }
}
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function* createId(){
  var index = 0;
  while(index < 3) { yield index++; }
}
```

```
for (var value of createId()) {
  console.log(value)
}
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

What is Flow Control?

Governing execution state, with the ability to pause or resume at any point.

Determine which elements of a program are able to run sequentially, or in parallel.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function getJSON(uri) {
  return JSON.parse(request.get(uri));
};

try {
  var data = getJSON('https://json.service');
} catch (e) {
  console.error(err);
}
```

```
function getJSON(uri, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    done(null, JSON.parse(data));
  });
}
```

Promises

A Promise object represents the result of a single, asynchronous action. A Promise is immutable, and can only express a set number of states: pending, fulfilled, or rejected.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
new Promise(function(resolve, reject) {
 // Execute Async Operation

}).then(function(result) {
 // Use the returned value.

}).catch(function(err) {
 // Throw the error.

});
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
new Promise(function(resolve, reject) {
 // Execute Async Operation

}).then(function(result) {
 // Use the returned value.

}).catch(function(err) {
 // Throw the error.

}).finally(function() {});
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
.then(function(result) {
  // Use the returned value.
})
```

…but here's what's interesting about 'then': you can easily pass any object that returns a value or Promise object. This makes the API extremely fluent, and easily chainable.

Let’s return to where we began:

```
function parseJSON(json, done) {
  try {
    json = JSON.parse(json);
  } catch(ex) {
    done(ex);
  }
  done(null, data);
}
```

```
function getJSON(json, done) {
  request.get(uri, function(err, data) {
    if (err) return done(err);
    parseJSON(data, function(err, json) {
      if (err) return done(err);
      done(null, json);
    });
  });
}
```

```
getJSON('http://json.service')
.then(data => {
  return JSON.parse(data)
}).then(json => {
  console.log(json);
}).catch(err => {
  console.error(err);
});
```

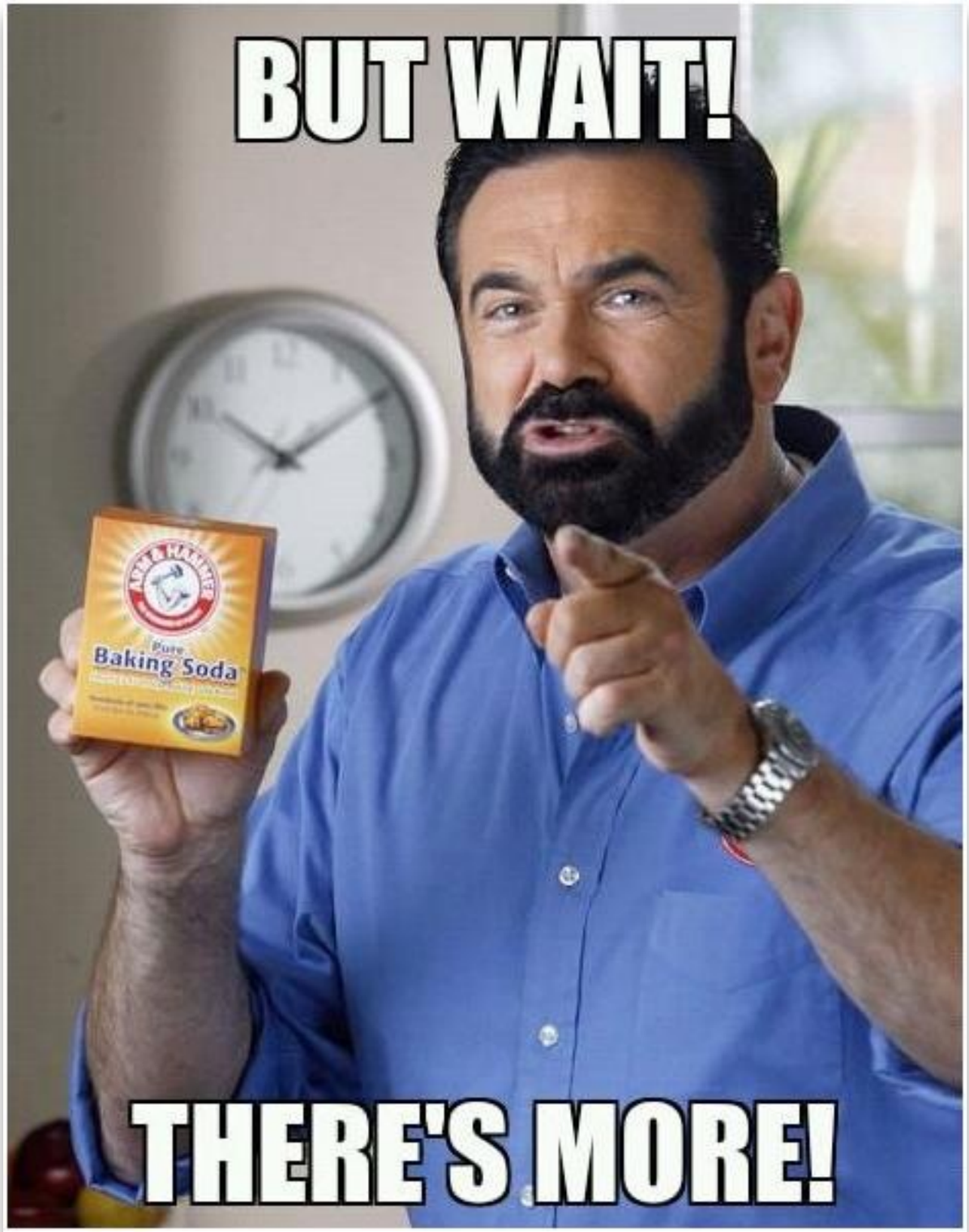
```
getJSON('http://json.service')
.then(JSON.parse)
.then(json => {
  console.log(json);
})
.catch(err => {
  console.error(err);
});
```

```
getJSON('http://json.service')
.then(JSON.parse)
//
```

Establishes a contract that can be easily enforced
Successfully handles errors in sync and async calls
Doesn’t work with existing flow control constructs

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember



Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

BABEL

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

BABEL

*A JavaScript transformer that allows you to write **code from the future**, but distribute code that runs in every browser now.*

babeljs.io

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

…how would I use this knowledge?

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

mixdown.co

API for Commercial Music

Media Playback Systems Algorithms

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

1. Select desired cities.
2. Retrieve top artists from local metro areas.
3. Output these statistics on a global admin dashboard.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
getHeatmap({
   city: 'nashville'
}, function (err, stats) {
   // cycle through stats with
   // async computation. how?
});
```

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

```
function* run() {
  var cities = [
    'chicago',
    'nashville',
    'los angeles',
    'austin'
  ];

  cities = cities.map(city => {
   query(city)
  });

  return yield Promise.all(cities);
}
```

1. Features are better when combined.
2. Documentation is still evolving.
3. Node's support for ES6 features is production ready today.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Thank you.

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Nicholas Young

Creative Generalist

@nicholaswyoung.com

nicholas@nicholaswyoung.com

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember

Q&A

Nicholas Young
Mixdown.co
#NextWave

November 2015
Nashville, Tennessee
#Nodevember