Async Functions in ES7 (ES2016)

Brian McKelvey
Dollar Shave Club

Github: theturtle32

JavaScript? ECMAScript? Huh?

JavaScript™

...trademark owned by Oracle

JavaScript? ECMAScript? Huh?

ECMAScript

The non-trademarked name used for the standard

Naming Confusion: ES6/7/2015/2016...???

ES6 = ECMAScript 6

ES7 = ECMAScript 7

Naming Confusion: ES6/7/2015/2016...???

ES6 = ES2015

ES7 = ES2016(?)

Small Demo App

1. Lookup user by email address.

2. Load user's friends.

3. In parallel, load user's company and interests.

4. console.log() the data

Road to Async Functions

1. Promises
2. Iterators
3. Iterables
4. Generators
5. Async Functions

Promises

```
var promise = User.findByEmail(
                        'theturtle32@gmail.com');

promise.then (user  => console.log(user.name));
promise.catch(error => console.log(error));
```

Iterators.
Iterables.
Generators.

Iterators

```
var array = [1,2,3];
var i = array.values();

> i.next();
   { value: 1, done: false }
> i.next();
   { value: 2, done: false }
> i.next();
   { value: 3, done: false }
> i.next();
   { value: undefined, done: true }
```

Iterables

```
var myObj = {};

myObj[Symbol.iterator] = function() {
  var state = 0;
  function next() {
    switch (state++) {
      case 0:  return { value: 1, done: false };
      case 1:  return { value: 2, done: false };
      case 2:  return { value: 3, done: false };
      default: return { value: undefined, done: true };
    }
  }
  return { next };
}

for (let i of myObj) console.log(i);
```

Symbols!

A symbol is a unique and immutable data type and may be used as an identifier for object properties

Iterables

```
var myObj = {};

myObj[Symbol.iterator] = function() {
  var state = 0;
  function next() {
    switch (state++) {
      case 0:  return { value: 1, done: false };
      case 1:  return { value: 2, done: false };
      case 2:  return { value: 3, done: false };
      default: return { value: undefined, done: true };
    }
  }
  return { next };
}

for (let i of myObj) console.log(i);
```

Generators

```
var myObj = {};

myObj[Symbol.iterator] = function*() {
   yield 1;
   yield 2;
   yield 3;
}

for (let i of myObj) console.log(i);
```

Generators

```
function *myGenerator() {
  yield 1;
  console.log("Hello, %s!", yield 2);
  yield 3;
}

> var i = myGenerator();
> i.next().value;
  1
> i.next().value;
  2
> i.next("Brian").value;
  Hello, Brian!
  3
```

Async via Generators & Promises

```
function slowHello(name) {
  return new Promise( (resolve, reject) => {
    setTimeout(() => resolve(`Hello slowly, ${name}`), 2000);
  });
}
```

Async via Generators & Promises

```
function slowHello(name) {
  return new Promise( (resolve, reject) => {
    setTimeout(() => resolve(`Hello slowly, ${name}`), 2000);
  });
}

run(function *() {
  console.log("Ok, lets get this going");
  console.log(yield slowHello("Brian"));
  console.log("Ok, we're done.");
});
```

Async via Generators & Promises

*Error handling skipped in this example for brevity

```
function run(generator) {
  return new Promise((resolve, reject) => {
    var iterator = generator();
    step();

    function step(resolution) {
      var yielded = iterator.next(resolution);
      if (yielded.done) return resolve(yielded.value);
      if (isPromise(yielded.value)) return yielded.value.then(step);
      step(yielded.value);
    }
  });
}

function isPromise(p) { return p && typeof p.then === 'function'; }
```

Async via Generators & Promises

```
function slowHello(name) {
  return new Promise( (resolve, reject) => {
    setTimeout(() => resolve(`Hello slowly, ${name}`), 2000);
  });
}

run(function *() {
  console.log("Ok, lets get this going");
  console.log(yield slowHello("Brian"));
  console.log("Ok, we're done.");
});
```

Async Functions

```
function slowHello(name) {
  return new Promise( (resolve, reject) => {
    setTimeout(() => resolve(`Hello slowly, ${name}`), 2000);
  });
}

(async function() {
  console.log("Ok, lets get this going");
  console.log(await slowHello("Brian"));
  console.log("Ok, we're done.");
})();
```

Async Functions in ES7 (ES2016)

Brian McKelvey
Dollar Shave Club

Github: theturtle32