ES7 Async Functions

Brian McKelvey

Road to Async Functions

- Promises
- Iterators
- Iterables
- Generators
- Async Functions

Promises

```
var promise = User.findByEmail('theturtle32@gmail.com');
promise.then(user => console.log(user.name));
```

Promises

```
var promise = User.findByEmail('theturtle32@gmail.com');
promise.then(user => console.log(user.name));
promise.catch(error => console.log(error));
```

Promises

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

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);
```

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

```
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

*Error handling skipped in this example for brevity

```
function run(generator) {
  var p = Promise.defer();
  var i = generator();
  var result = i.next();

  function step(resolution) {
    var yielded = i.next(resolution);
    if (yielded.done) return p.resolve(yielded.value);
    resolveable = yielded.value;
    if (resolveable && typeof resolveable['then'] === 'function') {
      return resolveable.next(step);
    }
    step();
  }

  var resolveable = result.value;
  if (resolveable) { resolveable.then(step); }
  return p;
}
```

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.");
})();
```

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