Asynchronous Behavior with Promises and $q

Before Promises - Callbacks

```
asyncFunction(function () {
  // do when asyncFunction is done
});
```

- No easy & straightforward way to pass the results of asyncFunction back to its caller
  - Especially, if the real recipient of the result is a few layers away

Before Promises - Callbacks

```
asyncFunction1(function () {
  asyncFunction2(function () {
    asyncFunction3(function () {
      // do when asyncFunction is done
    });
  });
});
```

- Hard to read the code
- What if we wanted all these to execute in parallel?
  - Not trivial to accomplish

Promise

Object which can be passed around or returned that holds references to the outcome of asynchronous behavior

- In Angular, promises are created through the the $q service

AngularJS $q Service Promise

```
function asyncFunction () {
  var deferred = $q.defer();

  if (…) { deferred.resolve(result); }

  else { deferred.reject(error); }

  return deferred.promise;
}
```

Creates async environment with all the hooks into it, including the promise object

Marks successful completion, wraps data for the promise

Marks *unsuccessful* completion, wraps data for the promise

Returns promise to caller
(a hook back to this entire process)

AngularJS $q Service Promise

```
function asyncFunction () {
  var deferred = $q.defer();

  if (…) { deferred.resolve(result); }

  else { deferred.reject(error); }

  return deferred.promise;
}
```

Can be done asynchronously

AngularJS $q Service Promise

```
var promise = asyncFunction();

promise.then(function (result) {
  // do something with result
},

function (error) {
  // do something with error
});
```

AngularJS $q Service Promise

```
var promise = asyncFunction();

promise.then(function (result) {
  // do something with result
},

function (error) {
  // do something with error
});
```

AngularJS $q Service Promise

```
var promise = asyncFunction();

promise.then(function (result) {
  // do something with result
},

function (error) {
  // do something with error
}).then(…);
```

AngularJS $q Service Promise

```
$q.all([promise1, promise2])

.then(function (result) {
  // do something with result
})

.catch(function (error) {
  // handle error
});
```

Summary

- Promises give us a lot of flexibility when dealing asynchronous behavior
- The $q service is the Angular implementation of Promise API
- Promises either get resolved or rejected
- The ‘then’ method takes 2 arguments (both function values)
  - 1st – function to handle success or ‘resolve’ outcome
  - 2nd – function to handle error or ‘reject’ outcome
  - ‘then’ itself returns a Promise, so it chainable
- $q.all method allows us to execute multiple promises in parallel, handling success/failure in one central place