jQuery Promises and Deferred Objects

Dr. Michael Whitney

Callback Review

Functions are *first-class citizens*

- Can be passed as arguments or stored in variables or fields

Callbacks

- Functions that are passed as an argument to another function

```
function randomFunction(callbackFunction) {
    callbackFunction();             // call the function that was passed in
}
randomFunction(function() {         // call the random function and pass it this simple function
    console.log('I will be passed to randomFunction!');
});
```

What is the purpose of Callbacks?

Allow calling code to deliver some logic that will be used in the called function.

- e.g., onclick

Perform delivered logic after an asynchronous operation

- e.g. success: or error:

Typical Ajax

```
$.ajax({type: "GET",
        url: "csciqm.json",
        dataType: "json",
        success: function(returnedData) {  // anonymous callback function
                                            // executed after async query finished
            console.log(returnedData);
        }
    });
```

Callback Hell

Some of the AJAX code we have done until now can easily become spaghetti like

- Only one function can be used for success, error, and complete
- Reuse of the $.ajax() is difficult

Messy Ajax: Only 1 success function

```
$.ajax({type: "GET",
        url: "csciqm.php",
        dataType: "POST",
        success: function(returnedData) {  // anonymous callback function
            $('#messages').animate({
                opacity: 0.25,
                height: "toggle"
            }, 5000, function() {
                setTimeout(function() {
                    $(this).val('Finished!');
            }, 1000); });
        }
    });
```

*http://brandonmartinez.github.io/

Promises to the Rescue!

- Promises have actually been around since 1976

- CommonJS formalized promises for javascript
  - jQuery implements something like Promises/A
    - http://wiki.commonjs.org/wiki/Promises/A
    - ^^^^ might be good to read to help understand

Promises/A from CommonJS

A promise represents the eventual value returned from the single completion of an operation. A promise may be in one of the three states, unfulfilled, fulfilled, and failed. The promise may only move from unfulfilled to fulfilled, or unfulfilled to failed. Once a promise is fulfilled or failed, the promise's value MUST not be changed, just as a values in JavaScript, primitives and object identities, can not change (although objects themselves may always be mutable even if their identity isn't). The immutable characteristic of promises are important for avoiding side-effects from listeners that can create unanticipated changes in behavior and allows promises to be passed to other functions without affecting the caller, in same way that primitives can be passed to functions without any concern that the caller's variable will be modified by the callee.

Simple Promise

Abstraction over asynchronous functions and tasks in JavaScript

A javaScript object with a *then* function which must return a promise

Then function either returns a fulfilled value or throws an exception

Fulfilled values allow one to chain multiple promises together

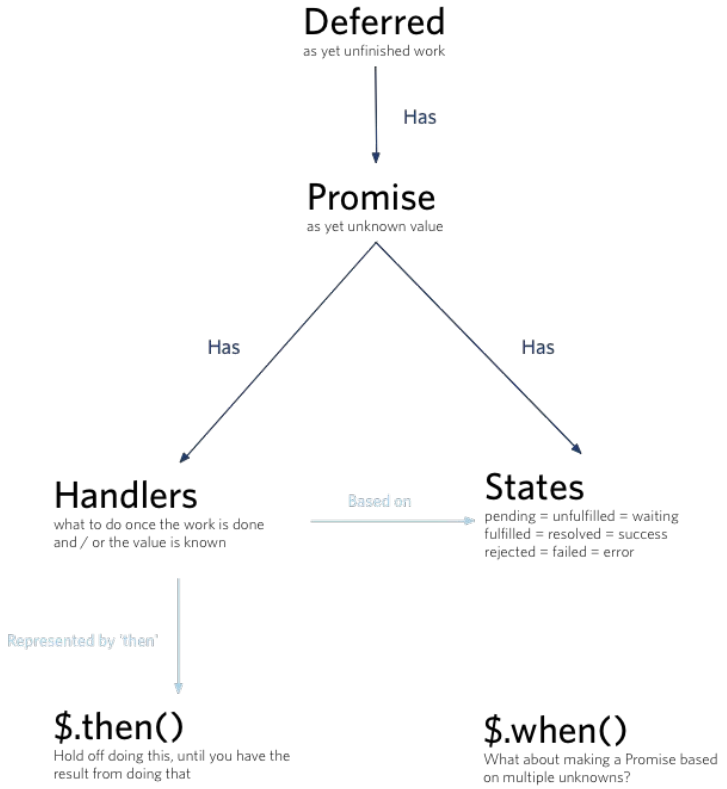
*http://brandonmartinez.github.io/

Deferred Object

- Before we can look at promises we must look at the deferred object $.Deferred()

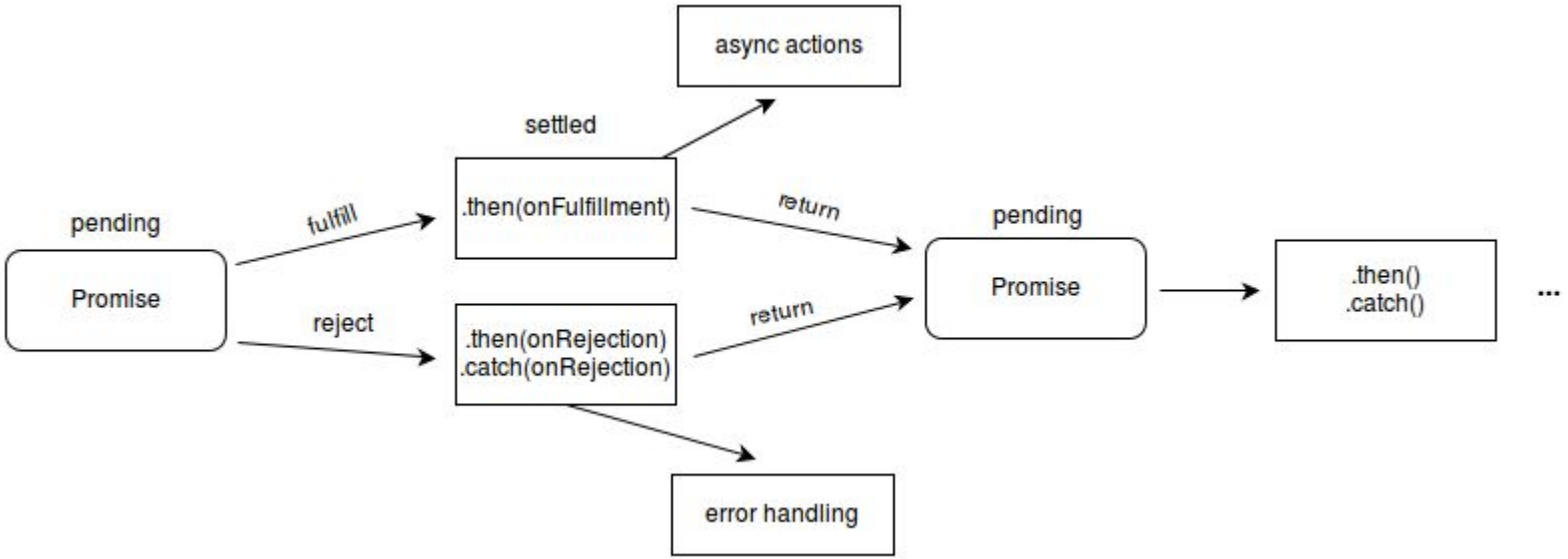
- Calling $.Deferred() will create a new deferred object for you to use
  - Starts as an unfulfilled or pending state
  - Your asynchronous code can either resolve() or reject() the deferred object
  - One way only - no going back

What does this look like?



*http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt1-theory-and-semantics

Another way to look at a Promise?



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

Deferred Object

- When the deferred object is pending you can notify() others of your progress()
  - notify() calls the progress() callbacks
- When the deferred object is resolved you are done() and do something else
  - You can have as many done()'s as you want (or none!)
  - Pre or post registered (if post and resolved it is executed immediately)

Deferred Object

- When the deferred object is rejected your code has fail()ed
  - You can have as many fail()'s as you want (or none!)
- You can also have an always()
  - always() happens anytime the deferred object is in the resolved or rejected state, never in the pending

AJAX and Deferred Objects

- Recent versions of jQuery (any you would conceivably use) have $.ajax() return a deferred object!
  - No need to attach success, error, and complete to the ajax() setup object
    - WARNING: These functions still exist on the returned deferred object, DO NOT use them!

Back to Ajax

```
$.ajax({
        url: "/ServerResource.txt",
        success: successFunction,
        error: errorFunction
});
```

Ajax Promise

Ajax calls' return object (jQuery XMLHttpRequest (jqXHR)) implements the CommonJS Promises/A interface

```
var promise = $.ajax({
      url: "/ServerResource.txt"
});

promise.done(successFunction);
promise.fail(errorFunction);
promise.always(alwaysFunction);
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Ajax Promise

Ajax calls' return object (jQuery XMLHttpRequest (jqXHR)) implements the CommonJS Promises/A interface

```
var promise = $.ajax({
    url: "/ServerResource.txt"
});

promise.done(successFunction);
promise.fail(errorFunction);
promise.always(alwaysFunction);
```

```
$.ajax( "example.php" )
    .done(function() { alert("success"); })
    .fail(function() { alert("error"); })
    .always(function() { alert("complete");
});
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Ajax then()

Another way to combine the handlers is to use the then() method of the Promise interface.

```
$.ajax({url: "/ServerResource.txt"}).then([successFunction1, successFunction2, successFunction3],
                                          [errorFunction1, errorFunction2]);

                //same as

                var jqxhr = $.ajax({
                  url: "/ServerResource.txt"
                });

                jqxhr.done(successFunction1);
                jqxhr.done(successFunction2);
                jqxhr.done(successFunction3);
                jqxhr.fail(errorFunction1);
                jqxhr.fail(errorFunction2);
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Ajax then()

then() method returns a new promise that can filter the status and values of a deferred through a function. Callbacks are guaranteed to run in the order they were passed in.

```
var promise = $.ajax({
        url: "/ServerResource.txt"
});

promise.then(successFunction, errorFunction);


var promise = $.ajax({
        url: "/ServerResource.txt"
});

promise.then(successFunction); //no handler for the fail() event
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Chaining then() functions

Then functions can be chained together to call multiple functions in succession:

```
var promise = $.ajax("/myServerScript1");

function getStuff() {
	return $.ajax("/myServerScript2");
}

promise.then(getStuff).then(function(myServerScript2Data){
	// Do something with myServerScript2Data
});
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Combining Promises

The Promise $.when() method is the equivalent of the logical AND operator. Given a list of Promises,

when() returns a new Promise whereby:

- When all of the given Promises are resolved, the new Promise is resolved.
- When any of the given Promises is rejected, the new Promise is rejected.

The following code uses the when() method to make two simultaneous Ajax calls and execute a function when both have successfully finished:

```
var jqxhr1 = $.ajax("/ServerResource1.txt");
var jqxhr2 = $.ajax("/ServerResource2.txt");

$.when(jqxhr1, jqxhr2).done(function(jqxhr1, jqxhr2) {
        // Handle both XHR objects
  alert("all complete");
});
```

** http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html

Promises Conclusion

Much easier to handle asynchronous processes