Promises

Wrangling Callbacks

Recap

- jQuery Selectors

Use jQuery CSS 3 selectors to more easily grab and manipulate objects on the page.

- More working with Ajax - developed chat room using resources loaded from the server.

Today’s Outline

- Simplify callback code with promises.
- Write mini-shopping cart.

Promises

As we’ve seen with AJAX calls to the server, asynchronous code does not necessarily run in the order we expect.

```
function findItem( itemName ) {
	$.getJSON(‘/items’, function (items) {
		return items.filter({ ‘name’ : ‘pants’ });
	});
}

var item = findItem(‘pants’);

console.log(item);  // what’s wrong with this?
```

Promises

“A promise represents the eventual result of an asynchronous operation.”

- This is an inversion and abstraction for callbacks, allowing flow of control to be more easily managed.

- A literal “promise” to return a value at some point in the future.

Promises - Difference

Without promises, we rely on callbacks after async code to guide flow. In one off scenarios this is acceptable, but quickly gets out of hand and can be hard to read.

- Get a specific room

```
$.getJSON('/rooms', function (rooms) {

    $.getJSON('/room/' + room[0].id, function (room) {
        // draw room on screen.
    });

});
```

Promises - Difference

```
function getRooms() {
	return $.getJSON('/rooms'); // This returns a promise already
}

function getSpecificRoom(rooms) {
	return $.getJSON('/room/' + room[0].id, function (room) {
}

getRooms()
	.then(getSpecificRoom)
	.then(drawRoomToScreen)
	.catch(handleErrors)
```

Promises - How

Promises need to be chainable, this is achieved by making promises “thenable”.

A promise is “thenable” if it provides a “then” function to access it’s current or eventual state.

You can identify a promise by it’s “then” function.

Promises - How

The “then” function takes two parameters, a success handler and a rejected handler.

“then” functions must always return another promise, this is what allows chaining.

```
promise
	.then(successFunction, errorFunction)		// returns a new promise
```

Promises - States

Promises can be in one of three states

- fulfilled - code has finished executing and the promise now has a value that cannot change.
- rejected - there was a problem in the promise, the promise will return a reason for the error (an error object).
- pending - the promise has not completed executing and will not have a value.

Promises

Some promise libraries will attach special methods that allow you to specify actions to happen at completion in either success or error case.

```
startLoadingAnimation()
        .then(findUsers)
        .then(doSomethingElse)
        .catch(handleErrors)
        .done(killLoadingAnimation);
```

Promises - Creation

Until ES 6 is formally adopted, you need to use a third-party library to create promises.

Under normal circumstances, I typically use the “q” library for promises. Angular adopted a sub-set of “q” and it’s muscle memory at this point.

jQuery has the Deferred object to help us along the way if you’re already using the library.

Promises - Creation

```
function userCanDrink( age ) {
    var deferred = $.Deferred();
    $.getJSON('/agelimit/us', function ( limit ) {
        if ( age > limit ) {
            deferred.resolve(true);
        } else {
            deferred.resolve(false);
        }
    })
    .fail(function () {
        deferred.reject('Problem communicating with the server.');
    });

    return deferred.promise();
}
```

Promises - All the Things

When loading complex pages with AJAX, it’s typical to have more than one resource. Loading the resources in parallel will speed up load time for the user.

Instead of using “then”, if you need to perform an operation only after multiple promises have been resolved, you can use Promise.all, Promise.when, depending on your library.

Promises - All The Things

```
function loadTweets() {
	return promise;
}
function loadCategories() {
	return promise;
}

Promise.all([ loadTweets(), loadCategories() ])
	.then(function (results) {
		// results is an array with both results
	});
```

Popular Promise Libraries

- q - https://github.com/kriskowal/q
- when - https://github.com/cujojs/when
- rsvp - https://github.com/tildeio/rsvp.js
- jQuery - (kinda) via the Deferred object https://api.jquery.com/category/deferred-object/

Citations & Further Reading

https://github.com/promises-aplus/promises-spec

https://github.com/domenic/promises-unwrapping/blob/master/docs/states-and-fates.md

Workshop