Promises, Promises: Unlocking the Power of jQuery's Deferreds

Brian Klaas
Johns Hopkins Bloomberg School of Public Health
bklaas@jhsph.edu
@brian_klaas

Code for today's session:

github.com/brianklaas/
ncdevcon2013-promises

The Problem

Is it done?

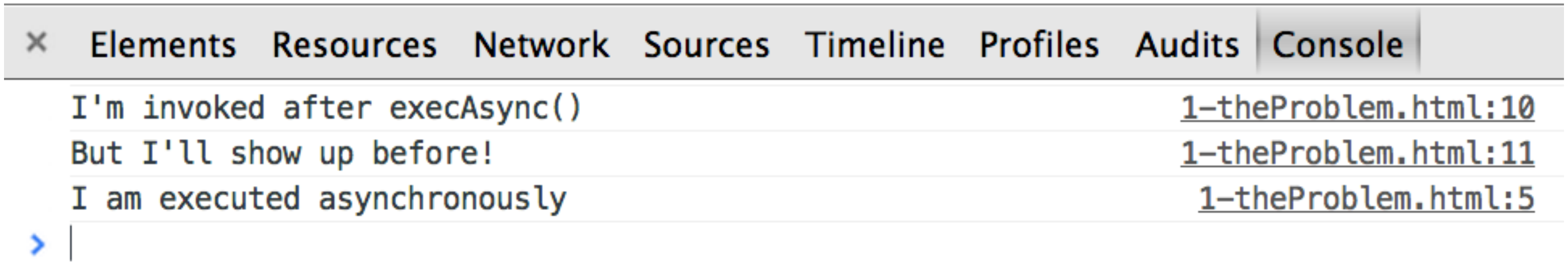
showTheProblem()

```
function execAsync() {
  console.log("I am executed asynchronously");
}

function showTheProblem() {
  setTimeout(execAsync, 0);
  console.log("I'm invoked after execAsync()");
  console.log("But I'll show up before!");
  return;
}

showTheProblem();
```

showTheProblem()


× Elements Resources Network Sources Timeline Profiles Audits Console

```
I'm invoked after execAsync()                1-theProblem.html:10
But I'll show up before!                     1-theProblem.html:11
I am executed asynchronously                  1-theProblem.html:5
>
```

wat?

Dealing with as yet unknown values from unfinished work is not fun.

Try setTimeout()

```
function processForm(data) {
  makeAjaxCall(data);
  setTimeout(showSuccessMsg, 3000);
}

function showSuccessMsg() {
  ...
}
```

Resolve parallel processes with counters

```
var stepsNeededToBeDone = 0;

function makeDinner(data) {
  $.ajax( "makePizza.cfm" ).done(function() {
    stepsNeededToBeDone++; }).fail(function() { alert("error"); });
  $.ajax( "makeSalad.cfm" ).done(function() {
    stepsNeededToBeDone++; }).fail(function() { alert("error"); });
  $.ajax( "openBeer.cfm" ).done(function() {
    stepsNeededToBeDone++; }).fail(function() { alert("cry!"); });

    setTimeout(checkForCompletion, 2000);
}

function checkForCompletion() {
  if (stepsNeededToBeDone == 3) {
    dinnerIsReady();
  } else {
    setTimeout(checkForCompletion, 2000);
  }
}
```

Get into callback hell

```
function makeDinner(data) {
  $.ajax( "makePizza.cfm" )
  .done(function() {
    $.ajax( "makeSalad.cfm" )
    .done(function() {
      $.ajax( "openBeer.cfm" )
      .done(function() { dinnerIsReady(); })
      .fail(function() { alert("Nooooooo"); });
   }).fail(function() { alert("No spring mix"); });
  }).fail(function() { alert("Pizza burned"); });
}
```

Wouldn’t this be a whole lot nicer?

```
when(
  $.ajax("/makePizza.cfm"),
  $.ajax("/makeSalad.cfm"),
  $.ajax("/openBeer.cfm"),
).then(dinnerIsReady);
```

The solution for asynchronous work:

Promises

What is a promise?

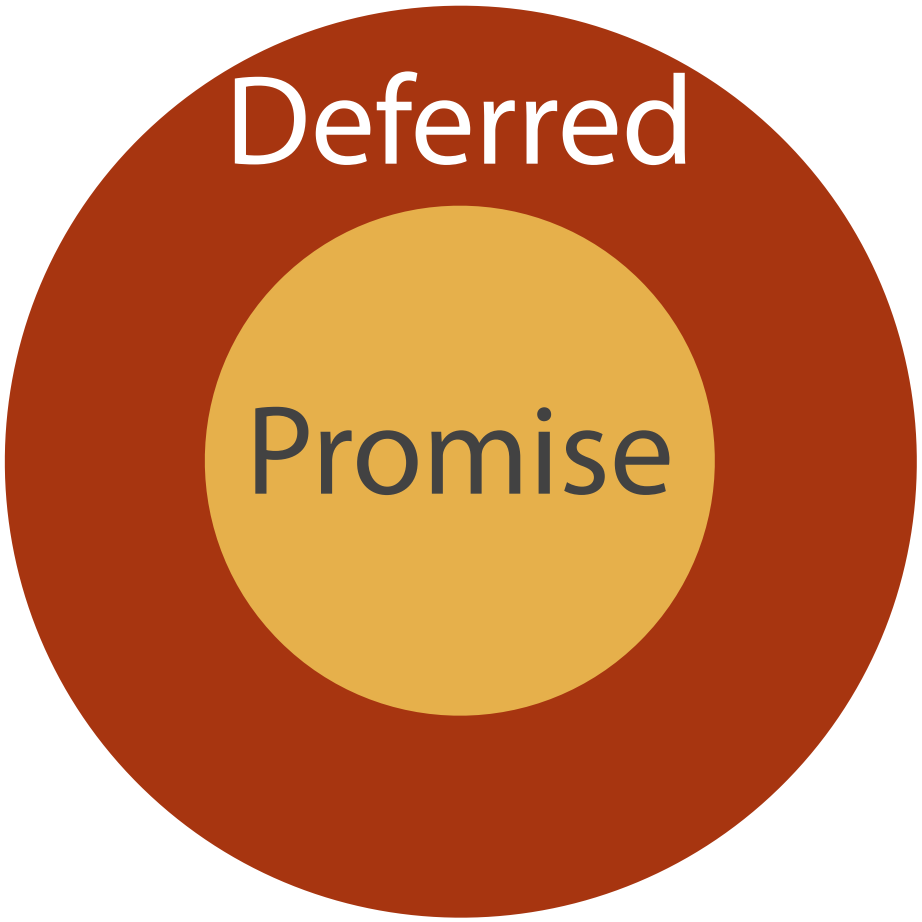
Or, more to the point: what is a deferred?

Promises enable developers to think and express asynchronous code in a more synchronous way.

A promise is an object that represents a one-time event, typically the outcome of an async task like an AJAX call.

Rather than waiting for the AJAX call to complete, we "promise" that the value will eventually be satisfied.

You can make multiple async calls and defer them to a single promise.


Deferred
Promise

promise = value that is not yet known

deferred = work that is not yet finished

A deferred has methods that allow its owner to resolve or reject it.

A promise returns a deferred. When the deferred is resolved (pass or fail), the promise is notified.

At first, a promise is in a pending state.

Eventually, it's either resolved (meaning the task is done) or rejected (if the task failed).

You can attach callbacks to the promise, which will fire when the promise is resolved or rejected.

Promises make it easy to say: "When all of these things have happened, do this other thing."

```
when(
 $.ajax("/makePizza.cfm"),
 $.ajax("/makeSalad.cfm"),
 $.ajax("/openBeer.cfm"),
).then(dinnerIsReady);
```

~~Promises/A+~~

http://promises-aplus.github.io/promises-spec/

jQuery Promise Basics

An Example Deferred

```
console.log("Using a base Deferred");
var deferred = new $.Deferred();

console.log(deferred.state());  // "pending"
deferred.resolve();
console.log(deferred.state());  // "resolved"
deferred.reject(); // no effect, the Promise was already resolved
console.log(deferred.state());  // "resolved"
```

An Example Promise

```
console.log("Now using Promises");
var deferred = new $.Deferred();
var promise = deferred.promise();

console.log(promise.state());  // "pending"
deferred.reject();
console.log(promise.state());  // "rejected"
```

An Example Promise: wait()

```
function wait(ms) {
  var deferred = $.Deferred();
  setTimeout(deferred.resolve, ms);
  return deferred.promise();
}

wait(1500).then(function () {
  console.log("We waited 1500ms");
});
```

`$.done()` = promise is resolved

`$.fail()` = promise is rejected

`$.when()`

For waiting on multiple deferreds to resolve

`$.when()` returns a new promise that obeys these rules:

- When *all* of the given promises are resolved, the new promise is resolved.
- If *any* of the given promises is rejected, the new promise is rejected.

$.when() Example

```
$.when(
  promiseOne,
  promiseTwo
)
  .done(function () {
    console.log('promiseOne and promiseTwo are done');
})
  .fail(function () {
    console.log('One of our promises failed');
});
```

`$.then()`

Returns a new promise that can filter the status and values of a deferred through a function.

$.then() Example

```
promiseOne.then(function () {
   console.log('promiseOne done');
   promiseTwo.then(function () {
      console.log('promiseTwo done');
      console.log('All done');
   });
});
```

`$.then()`

```
promise.then(doneCallback, failCallback, alwaysCallback)
```

You need to return a promise from `then()`
if you want to chain it.

$.then() Example

```
promiseOne.then(function () {
   console.log('promiseOne done');
   promiseTwo.then(function () {
      console.log('promiseTwo done');
      console.log('All done');
   });
});
```

A Cleaner $.then()

```
promiseOne.then(function () {
  console.log('promiseOne done');
}).then(function () {
  console.log('calling promiseTwo');
  return promiseTwo;
}).then(function () {
  console.log('All done');
});
```

Even better `$.then()`: Chain using named functions

Example: Animating with Promises

Syncing animations is not fun.

Is it done?

jQuery’s `animate()` toolbox uses promises

Parallel Animation

```
$.when(
  fadeOut('#divToFadeOut'),
  fadeIn('#divToFadeIn')
).done(function () {
  console.log('Parallel animation finished');
  $('p').css('color', 'red');
});
```

Chained Animation

```
fadeOut('#divToFadeOut')
  .then(function (el) {
    console.log("Fading in");
    fadeIn(el);
  }).then(function (el) {
    console.log("Chained animation finished");
  });
```

Solution:
Chain using named functions.

Promises and AJAX

In which you discover that you are already using promises.

jQuery returns promises
from all of its AJAX methods.

```
var jqxhr = $.ajax( "example.cfm" )
    .done(function() { alert("success"); })
    .fail(function() { alert("error"); })
    .always(function() { alert("complete"); });
```

jqXHR.success(), jqXHR.error()
are deprecated!

Use jqXHR.done(), jqXHR.fail(), and
jqXHR.always() instead.

http://api.jquery.com/jQuery.ajax/

```
var jqxhr = $.ajax( "example.cfm" )
    .then(function() { alert("Finished!"); });
```

```
jqXHR.then(
  function(data, textStatus, jqXHR) {},
  function(jqXHR, textStatus, errorThrown) {}
);
```

```
$.when($.ajax("/page1.cfm"), $.ajax("/page2.cfm"))
  .then(successFunc, failureFunc);
```

Form Submission Example

```
var onSubmitOrderForm = function() {
  $.post( "/processOrder.cfm", $("#orderForm").serialize())
    .done(onDoneFunc)
    .fail(onErrorFunc)
    .always(ajaxLogger);
  return false;
}

$("#orderForm").submit(onSubmitOrderForm);
```

Displaying content only after it loads

```
$.when(
  getLatestNews(),
  getLatestTweets(),
  loadContentAsync()
).then(function(){
  console.log( "UI fully loaded." );
}).fail(function(){
  console.log( "Something went wrong loading content!" );
});
```

Promises and UI/App Notifications

`.progress()`

Allows you to attach callbacks that are executed when `notify()` is called on the deferred.

Progress Meter Example

```
var fileProgress = $.Deferred();

// Called by the notify() method of a deferred object
fileProgress.progress(function(valToShow) {
  $("#progBar").val(valToShow);
});
```

Polling Your Server

`setInterval()` polling happens whether it's successful or not

A promise would handle both success or failure scenarios

Polling Your Server

```
var check = function () {
  var settings = {};
  settings.url = urlToCheck;
  settings.cache = false;
  var request = $.ajax(settings);
  request.then(success, failure);
}

setTimeout(check, 1000);
```

Persistence Adapters

by Christophe Coenraets

Decouple client code from the specific data access strategy (in-memory, AJAX, LocalStorage)

Use deferreds to handle the loading of the data

Persistence Adapters

by Christophe Coenraets

In the client, you have a function which requires backend access:

```
findByName(name)
.done(function(employees) {
  // loop through employees
}
```

Persistence Adapters

by Christophe Coenraets

Your adapters all implement `findByName(name)` and return deferreds.

Makes it easy to switch from inline JS mocking to real AJAX calls.

Unit Testing Promises

Basic strategy:

1. Mock/spy the call to the promise
2. Immediately resolve the promise with the data you expect.

Jasmine Example

```
spyOn($, 'ajax').andCallFake(function (req)
{
  var d = $.Deferred();
  d.resolve(data_you_expect);
  // or, d.reject(fail_result);
  return d.promise();
});
```

Beyond jQuery: Backbone, Node, and More

Backbone.js returns a jQuery deferred object when invoking any of the persistence methods (`save`,`fetch`, etc.).

Chaining on Backbone's default persistence methods

```
this.model.save().then(null, function(obj)
{
  console.log(obj.responseText);
});
```

```
$.when(collectionA.fetch(),collectionB.fetch())
.done(function(){
    //success code here.
});
```

Promise–based APIs

```
Parse.User.logIn("user", "pass").then(function(user) {
  return query.find();
}).then(function(results) {
  return results[0].save({ key: value });
}).then(function(result) {
  // the object was saved.
});
```

ahead

aligator

api–chain

assure

augur

avow

bond

branches

cancellation

clues

concurrent

covenant

deferred–queue

faithful

faithful–exec

futures

holdup

jasync

kew

Q

Q

Can exchange promises with jQuery.

Implements `then()`, `fail()`, `fin()` and more.

Available as an AMD and Common.js module, and in Bower.

Q

Calls a promise via `Q.fcall(function(){})`

Processing Credit Card Payments with Q

```
var deferred = Q.defer();
httpRequest.post({
    url: "https://api.balancedpayments.com"+MARKETPLACE_URI+url,
    auth: { user, pass, sendImmediately: true },
    json: params
}, function(error,response,body){
    if(body.status_code>=400){
        deferred.reject(body.description);
        return;
    }
    // Successful Requests
    deferred.resolve(body);
});
return deferred.promise;
```

Other Cool Stuff in Q: `spread()`

```
$("#getCustomer").click(function(cust) {
  var id = $("#cust-id").val();
  Q.spread([
    getCustomer(id),
    getContacts(id),
    getOrders(id),
    getAccountsRecv(id)
  ],
  function(cust, contacts, orders, ar){
     cust.contacts = contacts;
     cust.orders = orders;
     cust.ar = ar;
     // Do something with the
     // fully-populated customer object
  });
});
```

Promises and MongoDB

Magnolia: A promises–based library for MongoDB

```
magnolia('user')
  .filter({_id: ObjectID('4e4e1638c85e808431000003')})
  .one()
  .then(function (user) {
    console.log('hello', user.name);
  });
```

Best Practices

Avoid nested promise hell.

Use named promises whenever possible.

```
var step2 = step1.then()
```

Separate functionality into reusable bits.

Separate handler functions from the promise logic by calling a named function from `.then()`.

```
promiseTwo = promiseOne.then(someFunction);
```

Promises are a powerful tool for handling precise units of ***asynchronous*** action (login, data request, etc.).

The Future of Futures

Go Do

Session Evaluation

ncdevcon.com/sessions/

Thank you!

Brian Klaas
Johns Hopkins Bloomberg School of Public Health
bklaas@jhsph.edu
@brian_klaas
www.iterateme.com
github.com/brianklaas

Resources Used in Building this Presentation

- http://net.tutsplus.com/tutorials/javascript-ajax/wrangle-async-tasks-with-jquery-promises/
- http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt1-theory-and-semantics
- http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt2-practical-use
- http://addyosmani.com/blog/jquery-1-7s-callbacks-feature-demystified/
- http://msdn.microsoft.com/en-us/magazine/gg723713.aspx
- https://gist.github.com/ThomasBurleson/1910025
- http://eng.wealthfront.com/2012/12/jquerydeferred-is-most-important-client.html

Resources Used in Building this Presentation

- http://blog.parse.com/2013/01/29/whats-so-great-about-javascript-promises/
- http://www.html5rocks.com/en/tutorials/async/deferred/
- http://coenraets.org/blog/2013/04/building-pluggable-and-mock-data-adapters-for-web-and-phonegap-applications/
- http://stackoverflow.com/questions/13148356/how-to-properly-unit-test-jquerys-ajax-promises-using-jasmine-and-or-sinon
- http://www.jonnyreeves.co.uk/2012/unit-testing-async-javascript-with-promises-and-stubs/
- http://tech.pro/blog/1402/five-patterns-to-help-you-tame-asynchronous-javascript
- https://hacks.mozilla.org/2013/07/so-you-wanna-build-a-crowdfunding-site/