# Ajax and Data Tips and Tricks

pluralsight
hardcore developer training

- How to Use Ajax Promises
- Consolidating jQuery Ajax Calls
- Resolving Multiple Ajax Calls with $.when()
- Adding Headers to an Ajax Request
- Custom Ajax Converters
- Storing Data with the data() Function
- Working with HTML5 data-* Attributes

## How to use Ajax Promises

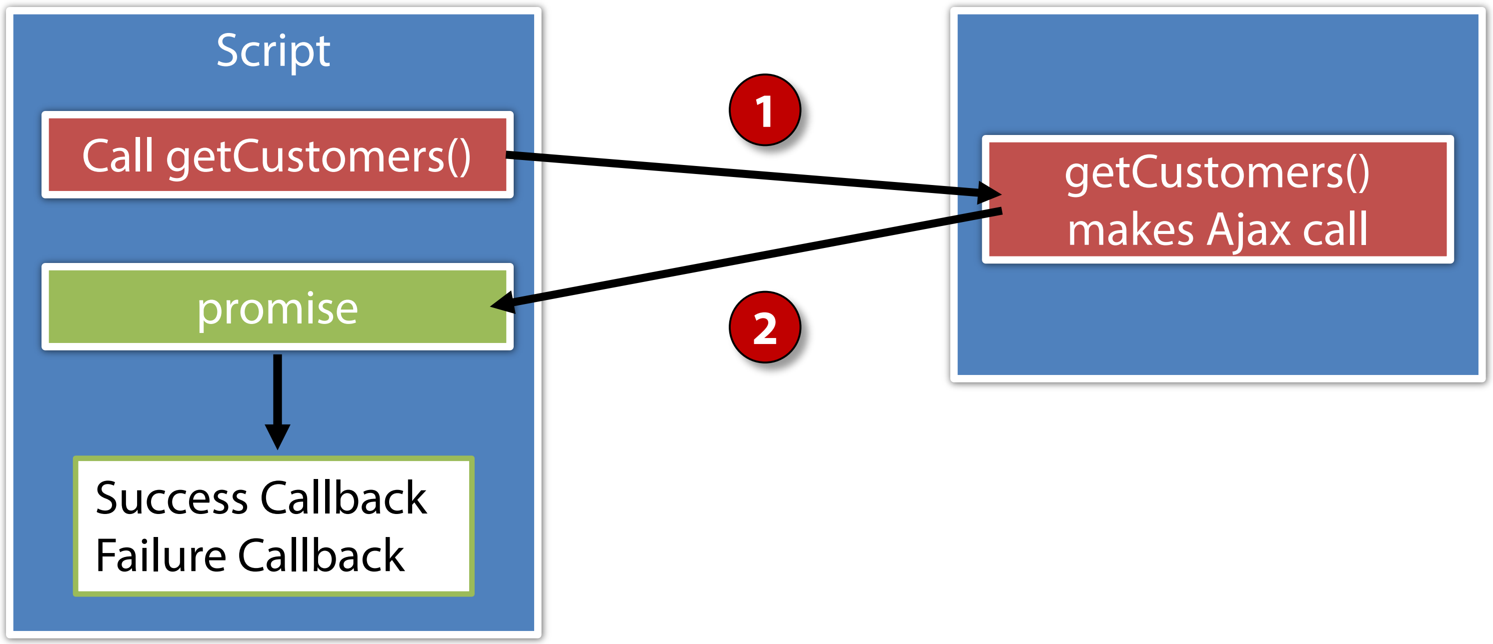
## What is a Promise?

- A promise is an object that helps deal with deferred results

- Typically used with asynchronous operations

- Example scenario that uses promises:
  1. Client calls an asynchronous function named getAccount()
  2. getAccount() immediately returns a promise object
  3. Client uses promise object to wire up success/error callbacks
  4. getAccount() continues to execute in the background
  5. Client callback(s) runs once getAccount() returns

# Promises in Action



## Using Ajax Promises with jQuery #1

```
function getCustomers() {
    return $.getJSON("http://localhost/api/customers");
}

//Consuming the promise and wiring up callbacks
getCustomers()
    .done(function (custs) {
        //Process data
    })
    .fail(function (data) {
        alert(data.statusText);
    });
```

Success callback

Failure callback

## Using Ajax Promises with jQuery #2

```
function getCustomers() {
    return $.getJSON("http://localhost/api/customers");
}

//Consuming the promise and wiring up callbacks
getCustomers()
    .then(function (custs) {
        //Process data
    },
    function (data) {
        alert(data.statusText);
    });
```

Success callback

Failure callback

## Consolidating jQuery Ajax Calls

pluralsight
hardcore developer training

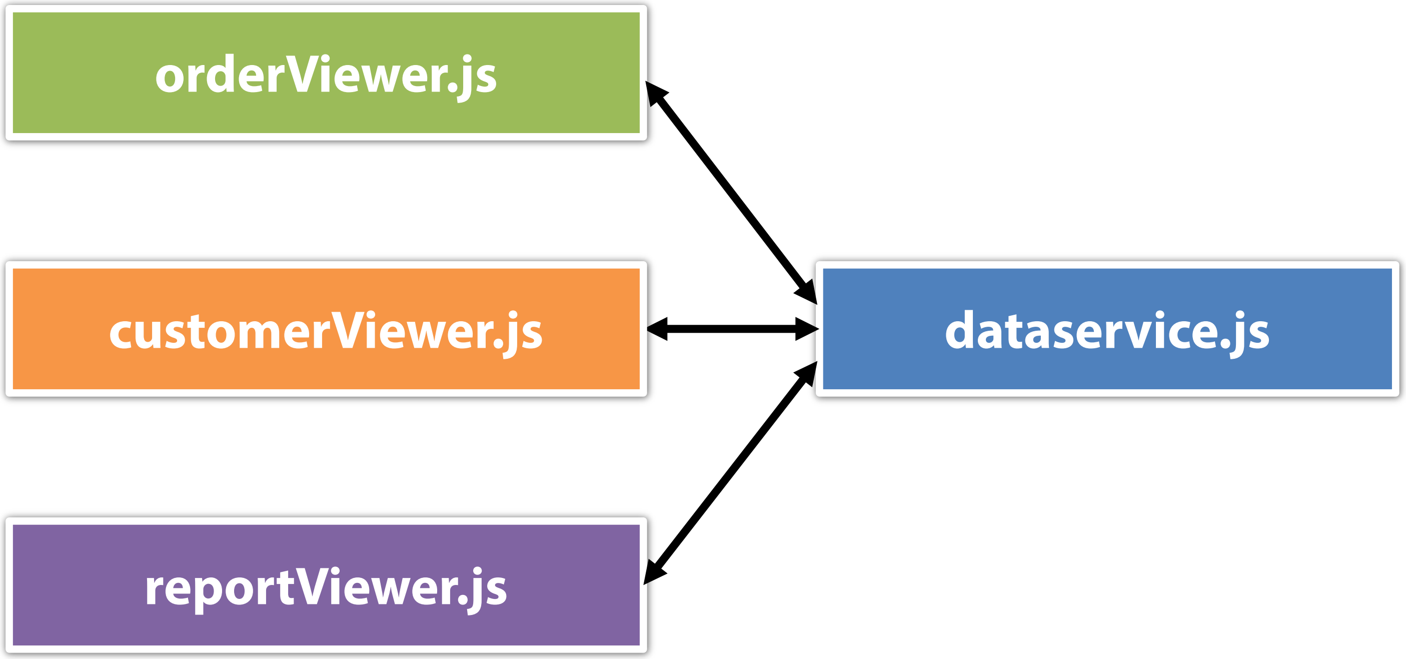
## Where do you Put your Ajax Code?

- Many apps scatter Ajax calls throughout the code:

```
$("#CustomerButton").click(function () {
    $.getJSON("api/Customers",
        function (data) {
            var cust = data[0];
            $("#ID").text(cust.ID);
            $("#FirstName").val(cust.FirstName);
            $("#LastName").val(cust.LastName);
        });
});
```

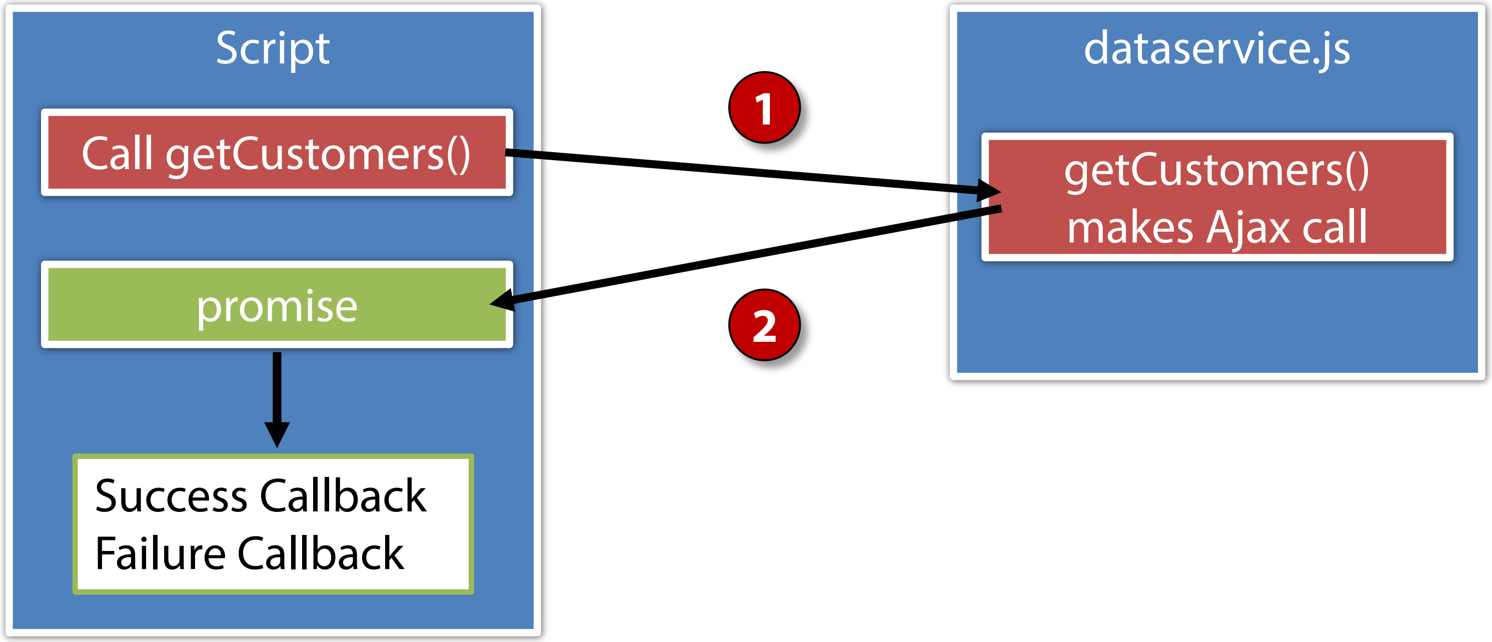
- What if Ajax calls were consolidated instead of scattered?

# Consolidating Ajax Calls



## Creating a dataService.js Script

- JSON data retrieved from the server
- dataservice.js responsible for making Ajax calls
- dataService functions accept parameters and returns promise objects



# dataservice.js Example

```
var dataService = new function () {
    var serviceBase = '/api/dataService/',
    getCustomers = function() {
        return $.getJSON(serviceBase + 'customers');
    };

    return {
        getCustomers : getCustomers
    };
}();
```

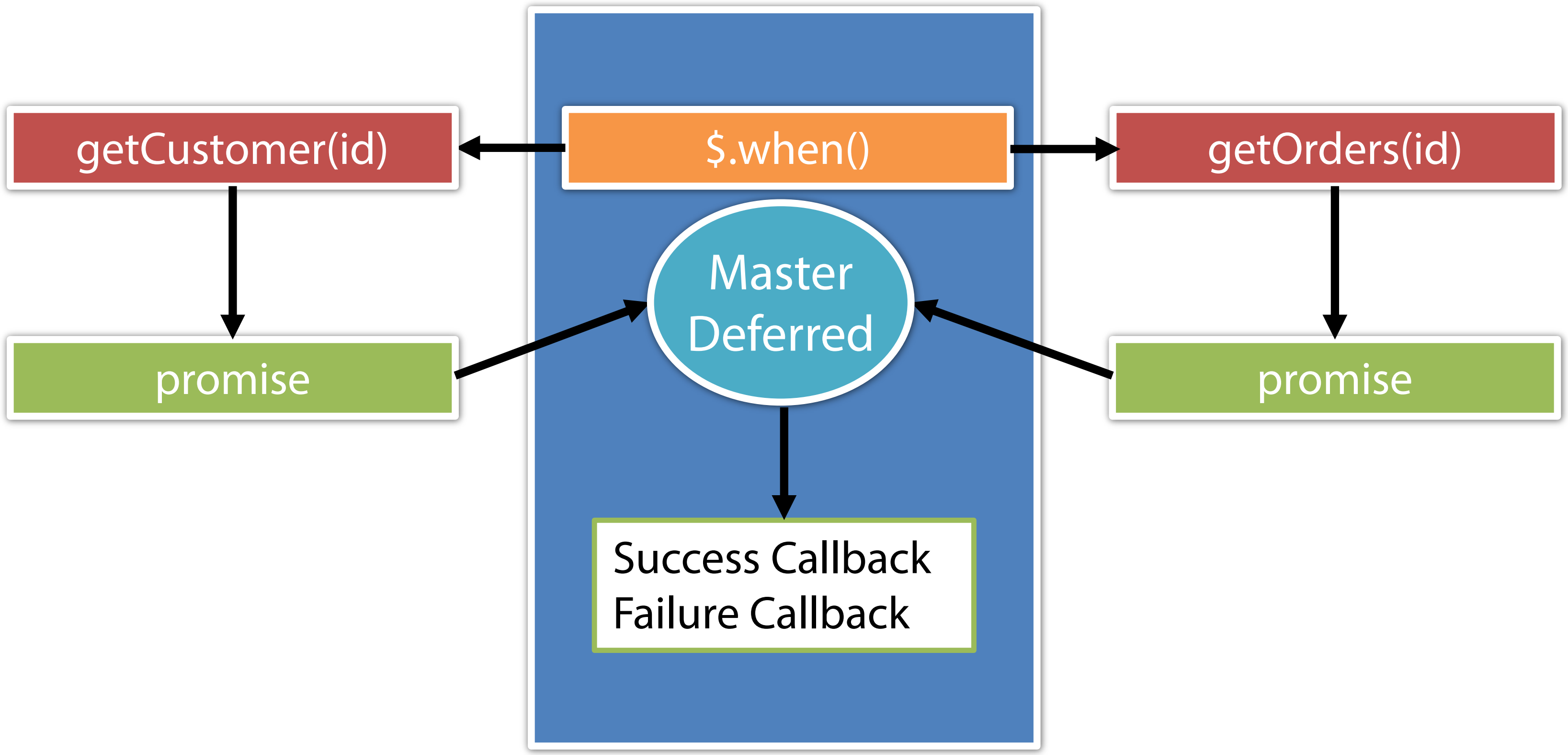
Return promise

## Resolving Multiple Ajax Calls with $.when

pluralsight
hardcore developer training

## Using $.when()

- $.when() allows multiple deferred calls to be resolved:



## Resolving Multiple Ajax Calls with $.when()

```
$.when(ajaxCall1(), ajaxCall2(), ajaxCall3())
    .done(function (c1Data, c2Data, c3Data) {
         //Process data
    })
    .fail(function (jqXHR, statusText, err) {
         //Handle error
    });
```

## Adding Headers to an Ajax Request

## Adding Headers to a Request

- $.ajax() function supports adding custom headers to requests
- beforeSend() function provides hook:

```
$.ajax({
    url: "/api/authentication",
    type: "POST",
    beforeSend: function (request) {
        request.setRequestHeader("AuthToken", authToken);
    }
});
```

Add Header

## Retrieving Header Values

- Header values can be retrieved once an Ajax response returns:

```
authenticate(authToken)
    .success(function (data, statusText, jqXHR) {
        $("#authToken").html(
          jqXHR.getResponseHeader("AuthToken"));
    })
    .fail(function(jqXHR, statusText, err) {
        alert("Error authenticating: " + err);
    });
```

Get Header

## Custom Ajax Converters

## Using Custom Ajax Converters

- **What if you need to "tweak" data returned from an Ajax call before the callback receives it?**

- **jQuery allows custom Ajax converters to be injected into the data processing pipeline**

## Creating an Ajax Converter

- Custom converters can be added “globally” in an application using $.ajaxSetup()

```
$.ajaxSetup({
    converters: {
        "json jsond": function (data) {
            return data && data.hasOwnProperty("d") ?
              data.d : data;
        }
    }
});
```

## Storing Data with the data() function

## Using the data() Function

- Data can be stored on a given element using the jQuery data() function:

```
$("selector").data("YourKey", "Data to store");
```

- Accessing stored data:

```
var value = $("selector").data("YourKey");
```

- jQuery removes data when a DOM element is removed

## Storing Multiple Values

- The data() function can be called and assigned to a variable to simply storing multiple values:

```
var custData = $("#custInfo").data();
custData.orders = orders;
custData.shippingInfo = shippingInfo;
```

## Working with HTML5 data-* Attributes

## Introduction to HTML5 data-* Attributes

- HTML5 data-* attributes provide a way to store custom data on elements:

```
<div id="pet"
    data-type="dog"
    data-name="Baron"
    data-age="2"
    data-trained="true"
    data-object='{"type": "dog", "name": "Baron",
                 "age": 2, "trained": true}'>
    Baron the Dog
</div>
```

## Access data-* Attributes using attr() and data()

- attr() function can be used to get/set data-* values:

```
var $pet = $("#pet");
var attrName = $pet.attr("data-name");
var attrName = $pet.attr("data-name","Fido");
```

Get/set data-name attribute value

- data() function can be used to get data-* values:

```
var $pet = $("#pet");
var dataName = $pet.data("name");
var dataName = $pet.data("name","Fido");
```

Get current data-name attribute Value

Will NOT set attribute value

## Don’t Mix attr() and data() with data-* Attributes

- **Use attr() when:**
  - You want to get or set HTML5 data-* attribute values
  - Attribute values need to be accessed as strings
- **Use data() when:**
  - You’re OK with data-* attributes being pulled in after the first call to the data() function (but not after that)
  - You need to cache a simple or complex value
  - You want data-* values automatically converted to a JavaScript value (bool, number, object, etc.)
- **Bottom-line: attr() and data() do not “sync” data-* attribute values**

## Summary

- **Key tips/tricks to consider when working with Ajax calls and data:**
  - Consolidate your jQuery Ajax calls using promises
  - Use $.when() to resolve multiple promises
  - jQuery supports custom headers
  - Use Ajax Converters to convert between data types
  - Cache data with the data() function where appropriate
  - Use attr() when directly manipulating HTML5 data-* attribute values