JQUERY

Ajax

Ajax

All web pages use Ajax forms of communication nowadays so its hard to see why this is a big deal

Traditional web pages used to reload the entire page for every server request!

reload the entire page

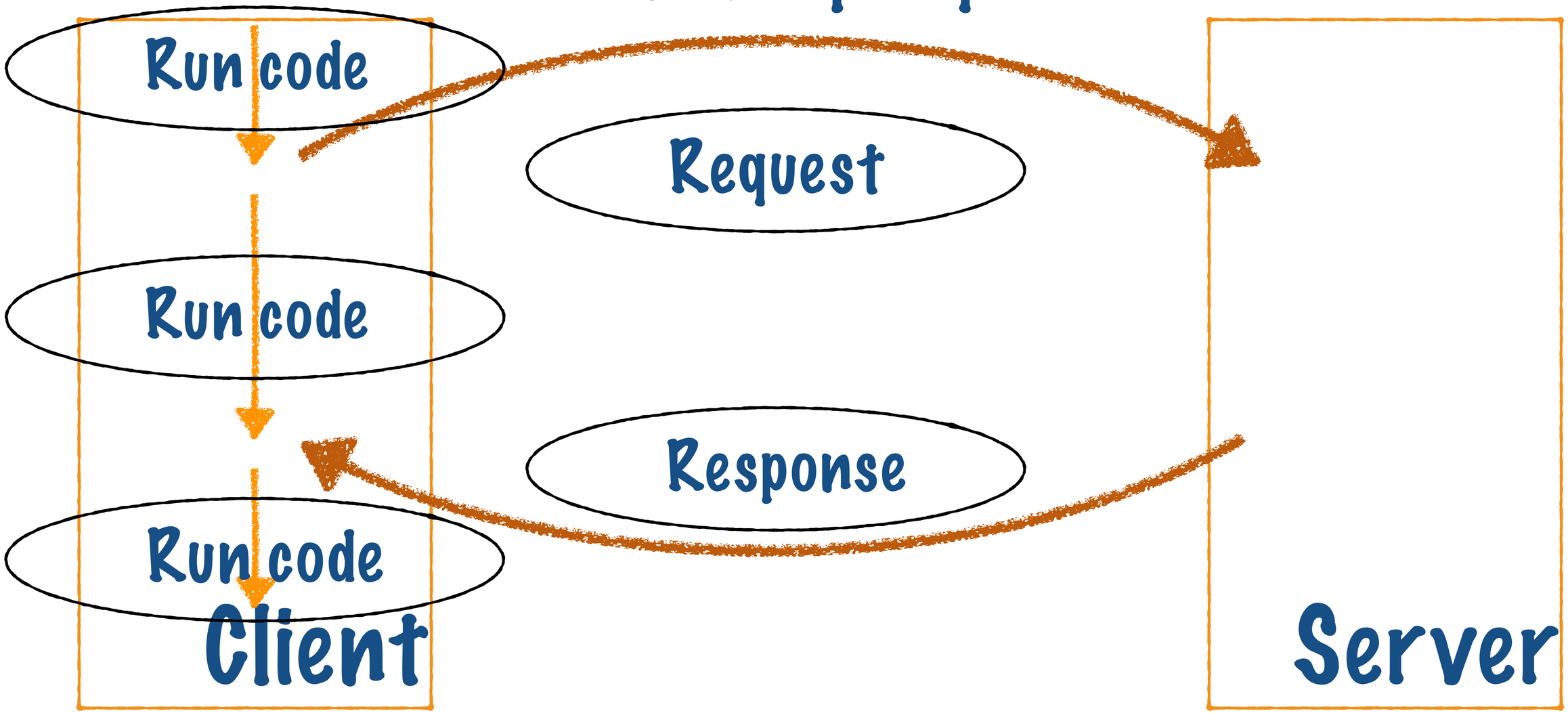
Imagine reloading your entire inbox
just because you had one new message

The solution to this was the
XMLHttpRequest

XMLHttpRequest

This allows you to request data from the server asynchronously

XMLHttpRequest



XMLHttpRequest

Javascript triggers an XMLHttpRequest to the server

Ajax

XMLHttpRequest

Javascript triggers an XMLHttpRequest to the server

You can specify a callback to run once a response has been received

XMLHttpRequest

Javascript triggers an XMLHttpRequest to the server

You can specify a callback to run once a response has been received

The rest of the code continues to run, the web page is interactive even as it awaits a response

XMLHttpRequest

Javascript triggers an
XMLHttpRequest to the server

You can specify a callback to run
once a response has been received

The rest of the code continues to run,
the web page is interactive even as it
awaits a response

Ajax

XMLHttpRequest

Javascript triggers an XMLHttpRequest to the server

Callback function is executed once a response has been received

The rest of the code continues to run, the web page is interactive even as it awaits a response

As with all browser standards, different browsers implement this request a little differently

Ajax

XMLHttpRequest

Javascript triggers an
XMLHttpRequest to the server

You can specify a callback to run
once a response has been received

The rest of the code continues to run,
the web page is interactive even as it
awaits a response

jQuery provides Ajax support that abstracts away browser differences

Ajax

XMLHttpRequest

Javascript triggers an XMLHttpRequest to the server

The callback function runs once a response has been received

The rest of the code continues to run, the web page is interactive even as it awaits a response

Just a note that Ajax does not work for cross-domain requests unless a server enables it using CORS

Ajax

Just a note that Ajax does not work for cross-domain requests unless a server enables it using CORS

Cross-Origin Resource Sharing is beyond the scope of this class - we'll only talk of Ajax requests to our own server

XMLHttpRequest

Javascript triggers an
XMLHttpRequest to the server

You can specify a callback to run
once a response has been received

The rest of the code continues to run,
the web page is interactive even as it
awaits a response

EXAMPLE 40

Ajax.html

Ajax.html

Example40Demo

Ajax.html

```
<body>
<div class="getdatasuccess link">
    Get data - success
</div>
<div class="getdatafail link">
    Get data - fail
</div>
<br>
<br>
<div id="container">
    <table>
        <tr>
            <th>Name</th>
            <th>Class</th>
        </tr>
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
…
       <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
    </table>

</div>
<br>
</body>
```

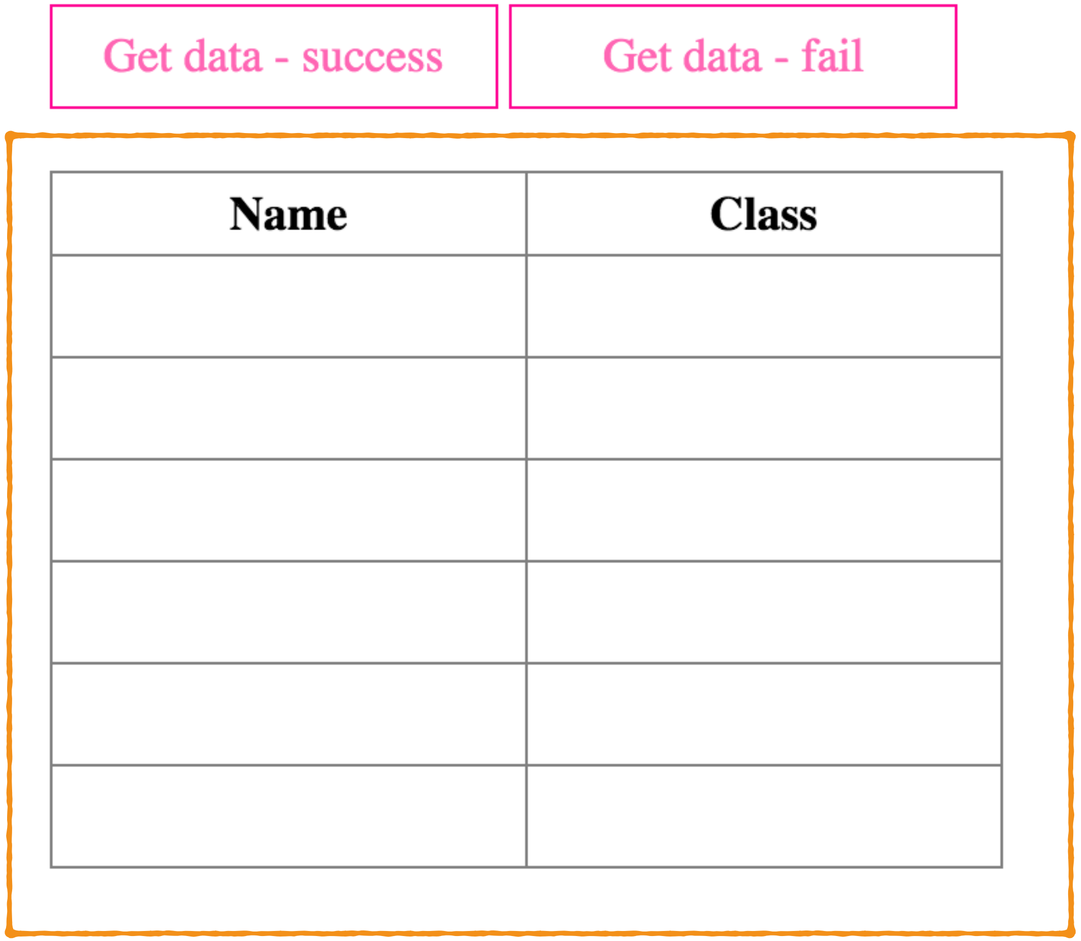
Get data - success | Get data - fail

Name	Class

Ajax.html

```
<body>
<div class="getdatasuccess link">
    Get data - success
</div>
<div class="getdatafail link">
    Get data - fail
</div>
<br>
<br>
<div id="container">
    <table>
        <tr>
            <th>Name</th>
            <th>Class</th>
        </tr>
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
…
       <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
    </table>

</div>
<br>
</body>
```



Ajax.html

```
$(document).ready(function () {
    $('.getdatasuccess').click(function() {
        $.ajax({
            url: 'data.txt',
            type: 'GET',
            dataType: 'json'
        })
        .done(function(response) {
            var index = 0;
            $('.row').each(function() {
                var obj = response[index++];
                $(this).children().eq(0).html(obj['name']);
                $(this).children().eq(1).html(obj['subject']);
            });
        })
        .always(function(xhr, status) {
            console.log('This is always called - status: ' + status);
        });
    });
    $('.getdatafail').click(function() {
        $.ajax({
            url: 'doesnotexist.txt',
            data: {start: 1, number: 10},
            type: 'GET'
        })
        .done(function(response) {
            console.log('This will not be called');
        })
        .fail(function(xhr, status, error) {
            console.log('Error: ' + error);
            console.log('Status: ' + status );
            console.dir(xhr);
        })
        .always(function(xhr, status) {
            console.log('This is always called - status: ' + status);
        });

    });
});
```

2 requests are made on this page - one which succeeds and one which fails

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

This successfully gets a JSON response from the server to fill up the table contents

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

The $.ajax() method makes an XMLHttpRequest to the server

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

It takes a configuration object to specify options in the request

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

The URL to which the request is made

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

The type of request - here it is a simple GET

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

We only want to retrieve data and not make changes on the server - a GET makes sense

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

The data type of the response - here it is JSON

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

It can be text, html, a script etc

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

The callback to be called if the request was a success

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

The callback receives the response as an argument

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

```
[
  {
    "name" : "Bathsheba Babbling",
    "subject" : "Ancient Runes"
  },
  {
    "name" : "Cuthbert Binns",
    "subject" : "History Of Magic"
  },
  {
    "name" : "Charity Babbage",
    "subject" : "Muggle Studies"
  },
  {
    "name" : "Filius Flitwick",
    "subject" : "Charms"
  },
  {
    "name" : "Firenze",
    "subject" : "Divination"
  },
  {
    "name" : "Aurora Sinistra",
    "subject" : "Astronomy"
  }
]
```

Iterate through every object in the response

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

```
[
  {
    "name" : "Bathsheba Babbling",
    "subject" : "Ancient Runes"
  },
  {
    "name" : "Cuthbert Binns",
    "subject" : "History Of Magic"
  },
  {
    "name" : "Charity Babbage",
    "subject" : "Muggle Studies"
  },
  {
    "name" : "Filius Flitwick",
    "subject" : "Charms"
  },
  {
    "name" : "Firenze",
    "subject" : "Divination"
  },
  {
    "name" : "Aurora Sinistra",
    "subject" : "Astronomy"
  }
]
```

The .each() method iterates through every selected jQuery element

Ajax.html

```
$('.getdatasuccess').click(function() {
    $.ajax({
        url: 'data.txt',
        type: 'GET',
        dataType: 'json'
    })
    .done(function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

```
[
  {
    "name" : "Bathsheba Babbling",
    "subject" : "Ancient Runes"
  },
  {
    "name" : "Cuthbert Binns",
    "subject" : "History Of Magic"
  },
  {
    "name" : "Charity Babbage",
    "subject" : "Muggle Studies"
  },
  {
    "name" : "Filius Flitwick",
    "subject" : "Charms"
  },
  {
    "name" : "Firenze",
    "subject" : "Divination"
  },
  {
    "name" : "Aurora Sinistra",
    "subject" : "Astronomy"
  }
]
```

Set the values of the name and subject in the right tables cells

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

```
This is always called - status: success
```

This is a callback which is always called, whether the request was a success or failure

Ajax.html

```
$('.getdatasuccess').click(function() {
     $.ajax({
         url: 'data.txt',
         type: 'GET',
         dataType: 'json'
     })
     .done(function(response) {
         var index = 0;
         $('.row').each(function() {
             var obj = response[index++];
             $(this).children().eq(0).html(obj['name']);
             $(this).children().eq(1).html(obj['subject']);
         });
     })
     .always(function(xhr, status) {
         console.log('This is always called - status: ' + status);
     });
 });
```

```
This is always called - status: success
```

This time the request succeeds and the status is success

Ajax.html

```
$(document).ready(function () {
    $('.getdatasuccess').click(function() {
        $.ajax({
            url: 'data.txt',
            type: 'GET',
            dataType: 'json'
        })
        .done(function(response) {
            var index = 0;
            $('.row').each(function() {
                var obj = response[index++];
                $(this).children().eq(0).html(obj['name']);
                $(this).children().eq(1).html(obj['subject']);
            });
        })
        .always(function(xhr, status) {
            console.log('This is always called - status: ' + status);
        });
    });
    $('.getdatafail').click(function() {
        $.ajax({
            url: 'doesnotexist.txt',
            data: {start: 1, number: 10},
            type: 'GET'
        })
        .done(function(response) {
            console.log('This will not be called');
        })
        .fail(function(xhr, status, error) {
            console.log('Error: ' + error);
            console.log('Status: ' + status );
            console.dir(xhr);
        })
        .always(function(xhr, status) {
            console.log('This is always called - status: ' + status);
        });

    });
});
```

Let's look at a request which fails

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

An Ajax request made for a file which does not exist

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

An Ajax request made for a file which does not exist

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

The data property is to specify the parameters we want to send along with the request

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

In a GET request they will be visible in the URL

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

The callback for when the response is a success

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

The callback for when the response is a failure

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

The original xhr sent, the status of the request and the error which occurred

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

This is called only when the request fails

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

```
▶ GET http://localhost:8080/examples/Example40/doesnotexist.txt?start=1&number=10 404 (Not Found)
▶ XHR finished loading: GET "http://localhost:8080/examples/Example40/doesnotexist.txt?start=1&number=10".
Error: Not Found
Status: error
▶ Object
This is always called - status: error
```

Ajax.html

```
$('.getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });
});
```

```
▶ GET http://localhost:8080/examples/Example40/doesnotexist.txt?start=1&number=10 404 (Not Found)
▶ XHR finished loading: GET "http://localhost:8080/examples/Example40/doesnotexist.txt?start=1&number=10".
Error: Not Found
Status: error
▶ Object
This is always called - status: error
```

Ajax.html

```
$('#getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

This configuration object has a bunch of other options you can specify as well

Ajax.html

```
$('#getdatafail').click(function() {
    $.ajax({
        url: 'doesnotexist.txt',
        data: {start: 1, number: 10},
        type: 'GET'
    })
    .done(function(response) {
        console.log('This will not be called');
    })
    .fail(function(xhr, status, error) {
        console.log('Error: ' + error);
        console.log('Status: ' + status );
        console.dir(xhr);
    })
    .always(function(xhr, status) {
        console.log('This is always called - status: ' + status);
    });

});
```

Whether the request should be async, whether to use a cached response if available etc.

EXAMPLE 41

GetGetScriptGetJson.html

jQuery provides a number of wrapper functions around common requests

GetGetScriptGetJson.html

jQuery provides a number of
wrapper functions around common
requests

These call the $.ajax() function under the hood and preset some of the configuration options

syntactic sugar

GetGetScriptGetJson.html

Example41Demo

GetGetScriptGetJson.html

```
<body>
<div class="getdata link">
    Get
</div>
<div class="getscript link">
    Get Script
</div>
<div class="getjson link">
    Get Json
</div>
<br>
<br>
<div id="htmldata">
</div>
<div id="scriptdata">
</div>
<br>
<div id="jsondata">
    <table>
        <tr>
            <th>Name</th>
            <th>Class</th>
        </tr>
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
…
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
    </table>
</div>
<br>
</body>
```

Get | Get Script | Get Json

Name	Class

GetGetScriptGetJson.html

```
<body>
<div class="getdata link">
    Get
</div>
<div class="getscript link">
    Get Script
</div>
<div class="getjson link">
    Get Json
</div>
<br>
<br>
<div id="htmldata">
</div>
<div id="scriptdata">
</div>
<br>
<div id="jsondata">
    <table>
        <tr>
            <th>Name</th>
            <th>Class</th>
        </tr>
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
…
        <tr class="row">
            <td class="name"></td>
            <td class="subject"></td>
        </tr>
    </table>
</div>
<br>
</body>
```

Get | Get Script | Get Json

Name	Class

GetGetScriptGetJson.html

```
$('.getdata').click(function() {
    $.get('data.html', function(response) {
        $('#htmldata').html(response);
    })
});
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
$('.getjson').click(function() {
    $.getJSON('data.txt', function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
});
```

GetGetScriptGetJson.html

```
$('.getdata').click(function() {
    $.get('data.html', function(response) {
        $('#htmldata').html(response);
    })
});
```

The $.get() makes a GET request to the specified URL

GetGetScriptGetJson.html

```
$('.getdata').click(function() {
    $.get('data.html', function(response) {
        $('#htmldata').html(response);
    })
});
```

Notice how the request becomes really concise

GetGetScriptGetJson.html

```
$('.getdata').click(function() {
    $.get('data.html', function(response) {
        $('#htmldata').html(response);
    })
});
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
$('.getjson').click(function() {
    $.getJSON('data.txt', function(response) {
        var index = 0;
        $('.roy').children().each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['object']);
        });
    })
});
```

```
<div style="font-size: 33px"><b>I am HTML!</b></div>
```

The response is in the HTML format

GetGetScriptGetJson.html

```
$('.getdata').click(function() {
    $.get('data.html', function(response) {
        $('#htmldata').html(response);
    })
});
```

```
<div style="font-size: 33px"><b>I am HTML!</b></div>
```

Get | Get Script | Get Json

I am HTML!

I am a script!

Name	Class

We simply display it

GetGetScriptGetJson.html

```
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
```

The $.getScript() function retrieves a script from the server

GetGetScriptGetJson.html

```
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
```

It is a GET request which expects a script in the response

GetGetScriptGetJson.html

```
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
```

Log the response

GetGetScriptGetJson.html

```
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
```

```
document.getElementById('scriptdata').innerHTML = 'I am a script!';
```

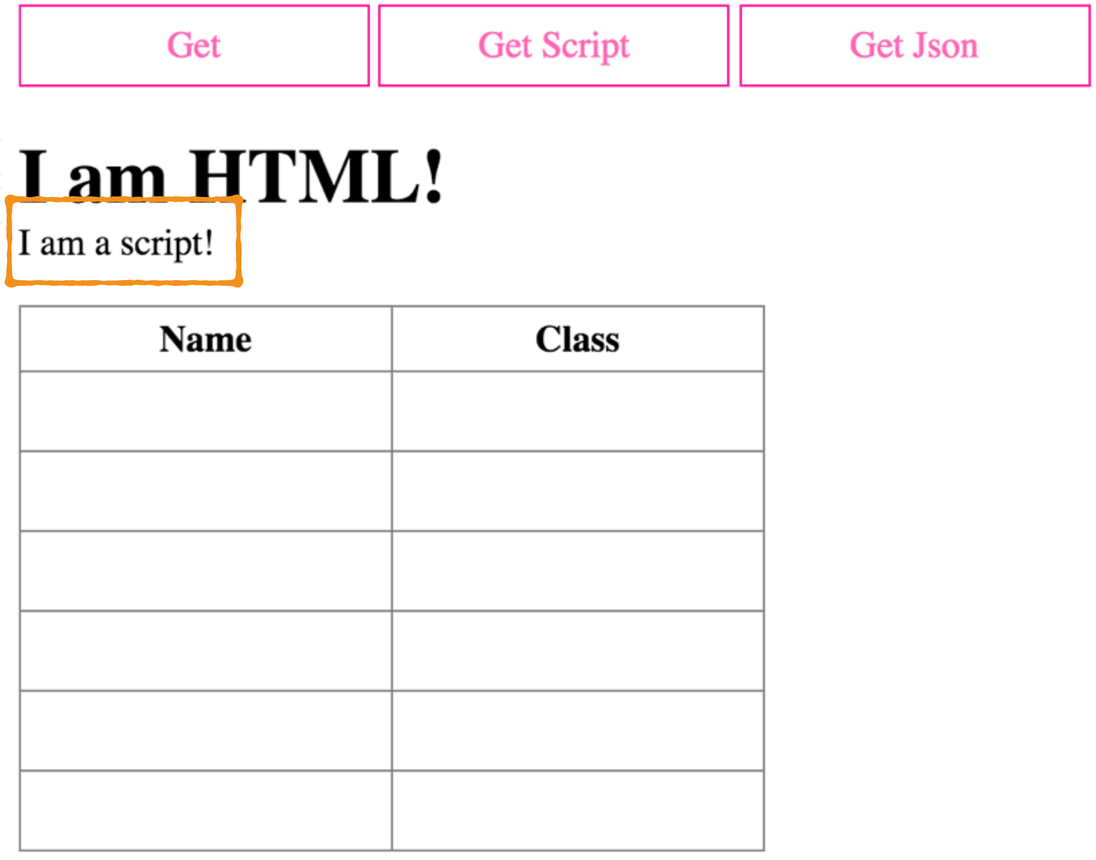
The script is executed when it is received on the client

GetGetScriptGetJson.html

```
$('.getscript').click(function() {
    $.getScript('data.js', {}, function(response) {
        console.log(response);
    });
});
```

```
document.getElementById('scriptdata').innerHTML = 'I am a script!';
```

This renders a message on the screen



GetGetScriptGetJson.html

```
$('.getjson').click(function() {
    $.getJSON('data.txt', function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
});
```

The $.getJSON() is the exact equivalent of the $.ajax() function that we saw in the previous example

GetGetScriptGetJson.html

```
$('.getjson').click(function() {
    $.getJSON('data.txt', function(response) {
        var index = 0;
        $('.row').each(function() {
            var obj = response[index++];
            $(this).children().eq(0).html(obj['name']);
            $(this).children().eq(1).html(obj['subject']);
        });
    })
});
```

It populates the data in the table as before

EXAMPLE 42

Load.html

Load.html

Example42Demo

Load.html

```
$(document).ready(function () {
    $('.load').click(function() {
        $('#images').load('images.html');
    });
});
```

The .load() method is the easiest to use of the lot when you have HTML refactored into different files

Load.html

```
$(document).ready(function () {
    $('.load').click(function() {
        $('#images').load('images.html');
    });
});
```

It is the only function which runs on a selection i.e. is associated with a DOM even before it is executed

Load.html

```
$(document).ready(function () {
    $('.load').click(function() {
        $('#images').load('images.html');
    });
});
```

Get the HTML from this URL

Load.html

```
$(document).ready(function () {
    $('.load').click(function() {
        $('#images').load('images.html');
    });
});
```

```
<div>
<img src="Moje.jpg" width="400px" height="400px"/>
<img src="Oba.jpg" width="400px" height="400px" />
</div>
```

Load.html

```
$(document).ready(function () {
    $('.load').click(function() {
        $('#images').load('images.html');
    });
});
```

```
<div>
<img src="Moje.jpg" width="400px" height="400px"/>
<img src="Oba.jpg" width="400px" height="400px" />
</div>
```

Populate the selected elements with the retrieved HTML

EXAMPLE 43

FormSerializeAndSerializeArray.html

FormSerializeAndSerializeArray.html

jQuery provides some easy ways to serialize forms input to send to the server

You can also perform client side validations before the form is submitted

Example43Demo

FormSerializeAndSerializeArray.html

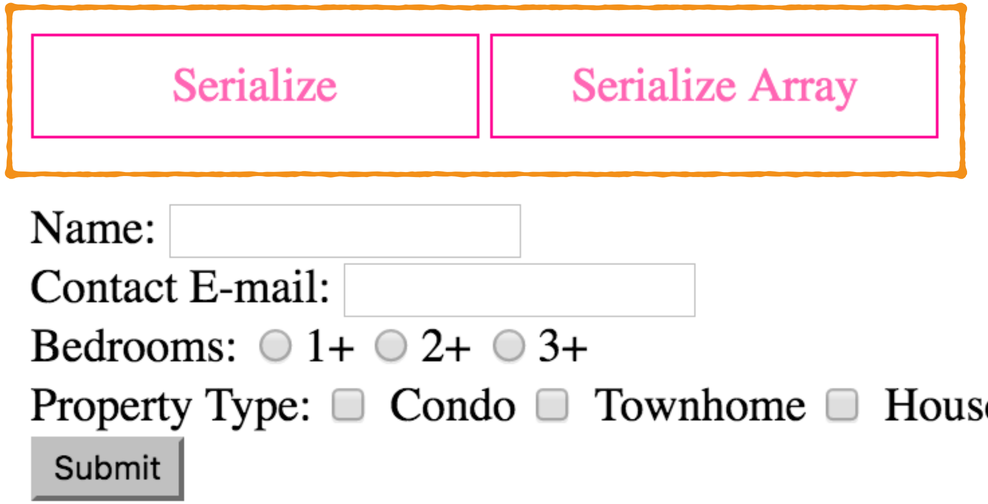
```
<body>
<div class="serialize link">
    Serialize
</div>
<div class="serializearray link">
    Serialize Array
</div>
<form>
    Name:
    <input name="clientName" class="required"/>
    <span class="error" style="color: red"></span>
    <br>

    Contact E-mail:
    <input name="clientEmail" type="email"/>
    <br>

    Bedrooms:
    <input type="radio" name="onebr" value="1" />1+
    <input type="radio" name="twobr" value="2" />2+
    <input type="radio" name="threebr" value="3" />3+ <br>

    Property Type:
    <input type="checkbox" name="condo"/> Condo
    <input type="checkbox" name="townhome" /> Townhome
    <input type="checkbox" name="house" /> House <br>

    <input type="submit" />
</form>
<div id="serializeid">
</div>
</body>
```



FormSerializeAndSerializeArray.html

```
<body>
<div class="serialize link">
    Serialize
</div>
<div class="serializearray link">
    Serialize Array
</div>
<form>
    Name:
    <input name="clientName" class="required"/>
    <span class="error" style="color: red"></span>
    <br>

    Contact E-mail:
    <input name="clientEmail" type="email"/>
    <br>

    Bedrooms:
    <input type="radio" name="onebr" value="1" />1+
    <input type="radio" name="twobr" value="2" />2+
    <input type="radio" name="threebr" value="3" />3+ <br>

    Property Type:
    <input type="checkbox" name="condo"/> Condo
    <input type="checkbox" name="townhome" /> Townhome
    <input type="checkbox" name="house" /> House <br>

    <input type="submit" />
</form>
<div id="serializeid">
</div>
</body>
```

Serialize | Serialize Array

Name:

Contact E-mail:

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

FormSerializeAndSerializeArray.html

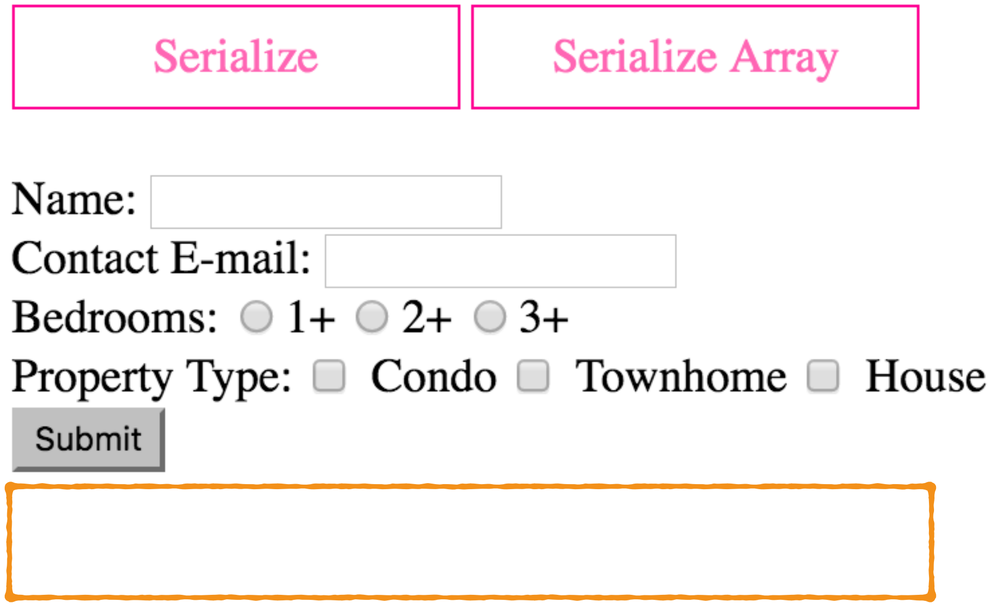
```
<body>
<div class="serialize link">
    Serialize
</div>
<div class="serializearray link">
    Serialize Array
</div>
<form>
    Name:
    <input name="clientName" class="required"/>
    <span class="error" style="color: red"></span>
    <br>

    Contact E-mail:
    <input name="clientEmail" type="email"/>
    <br>

    Bedrooms:
    <input type="radio" name="onebr" value="1" />1+
    <input type="radio" name="twobr" value="2" />2+
    <input type="radio" name="threebr" value="3" />3+ <br>

    Property Type:
    <input type="checkbox" name="condo"/> Condo
    <input type="checkbox" name="townhome" /> Townhome
    <input type="checkbox" name="house" /> House <br>

    <input type="submit" />
</form>
<div id="serializeid">
</div>
</body>
```



FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

The form data can be serialized into a query string

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Select the form and call serialize on it

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

```
clientName=Janani%20Ravi&clientEmail=janani%40gmail.com&threebr=3
```

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

clientName=Janani%20Ravi&clientEmail=janani%40gmail.com&threebr=3

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Or if you want an array of objects i.e. JSON you use the .serializeArray() function on the form

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

[{"name":"clientName","value":"Janani Ravi"},
{"name":"clientEmail","value":"janani@gmail.com"},
{"name":"threebr","value":"3"},
{"name":"townhome","value":"on"}]

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

[{"name":"clientName","value":"Janani Ravi"},
{"name":"clientEmail","value":"janani@gmail.com"},
{"name":"threebr","value":"3"},
{"name":"townhome","value":"on"}]

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

[{"name":"clientName","value":"Janani Ravi"},
{"name":"clientEmail","value":"janani@gmail.com"},
{"name":"threebr","value":"3"},
{"name":"townhome","value":"on"}]

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

[{"name":"clientName","value":"Janani Ravi"},
{"name":"clientEmail","value":"janani@gmail.com"},
{"name":"threebr","value":"3"},
{"name":"townhome","value":"on"}]

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Name: Janani Ravi

Contact E-mail: janani@gmail.com

Bedrooms: 1+ 2+ 3+

Property Type: Condo Townhome House

Submit

[{"name":"clientName","value":"Janani Ravi"},
{"name":"clientEmail","value":"janani@gmail.com"},
{"name":"threebr","value":"3"},
{"name":"townhome","value":"on"}]

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Some client side validation for required fields

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

This callback is run before the form is submitted

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

You can check for required fields, properly formatted fields, a whole bunch of stuff

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Check whether all required fields have some value

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

If not, find the first sibling used to display errors and display a warning

FormSerializeAndSerializeArray.html

```
$('.serialize').click(function() {
    $('#serializeid').html($('form').serialize());
});
$('.serializearray').click(function() {
    $('#serializeid').html(JSON.stringify($('form').serializeArray()));
});

$('form').submit(function(e) {
    var requiredJqEl = $('.required');
    if (requiredJqEl.val().length == 0) {
        var el = requiredJqEl.siblings('.error').first();
        el.html('*required');

        e.preventDefault();
    }
});
```

Do not submit the form with errors!

EXAMPLE 44

AjaxEvents.html

Ajax events are used to indicate state changes when we make XMLHttpRequests to the server

We can hook into events to run callbacks on successful requests as well as failed requests

jQuery allows us to listen to two kinds of Ajax events

Local

Global

AjaxEvents.html

Local

These are callbacks you can listen to on a specific Ajax request

```
$.ajax({
    beforeSend: function(xhr) {
    },
    success: function() {
    },
    error: function() {
    },
    complete: function(xhr) {
    }
});
```

```
$.ajax({
})
.done(function(response) {
})
.always(function(xhr, status) {
});
```

AjaxEvents.html

Local

```
$.ajax({
    beforeSend: function(xhr) {
    },
    success: function() {
    },
    error: function() {
    },
    complete: function(xhr) {
    }
});
```

```
$.ajax({
})
.done(function(response) {
})
.always(function(xhr, status) {
});
```

There are a number of ways to hook into these events

AjaxEvents.html

jQuery allows us to listen to two kinds of Ajax events

Local

Global

global

These are request related callbacks triggered on the document

All requests made on the web app trigger these events

global

These are request related callbacks triggered on the document

ajaxSend, ajaxComplete, ajaxError are some of these global events

AjaxEvents.html

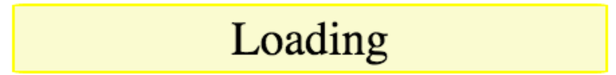
jQuery allows us to listen to two kinds of Ajax events

Local

Global

AjaxEvents.html

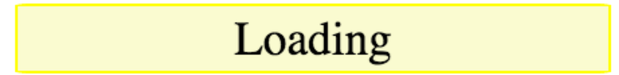
```
<body>
<div class="butter" id="loading">
    Loading
</div>
<div class="ajax link">
    Make request
</div>
</body>
```



This button makes an Ajax request to the server

AjaxEvents.html

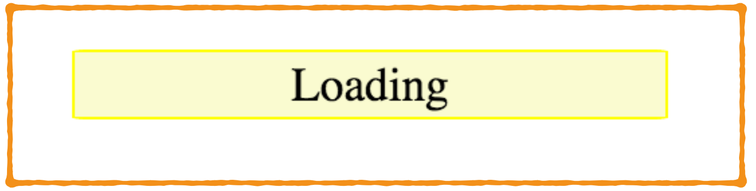
```
<body>
<div class="butter" id="loading">
    Loading
</div>
<div class="ajax link">
    Make request
</div>
</body>
```



We will listen to both global and local events of this request

AjaxEvents.html

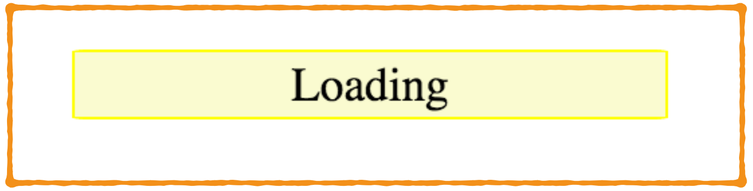
```
<body>
<div class="butter" id="loading">
    Loading
</div>
<div class="ajax link">
    Make request
</div>
</body>
```



This is a loading message which shows whenever we're waiting on the response of any Ajax request

AjaxEvents.html

```
<body>
<div class="butter" id="loading">
    Loading
</div>
<div class="ajax link">
    Make request
</div>
</body>
```



Triggered by a global event!

AjaxEvents.html

```
$(document).ready(function () {
    $('.ajax').click(function() {
        $.ajax({
            url: 'data.txt',
            type: 'GET',
            dataType: 'json',
            beforeSend: function(xhr) {
                console.log('beforeSend');
                xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
                xhr.setRequestHeader('Accept-Language', 'en-US');
            },
            success: function() {
                console.log('success');
            },
            error: function() {
                console.log('error');
            },
            complete: function(xhr) {
                console.log('complete');
                console.log(xhr.getAllResponseHeaders());
            }
        });

    });

    $('#loading').hide();
    $(document).bind("ajaxSend", function(){
        $("#loading").show();
    }).ajaxComplete(function() {
        setTimeout(function() {
            $("#loading").hide();
        }, 8000);
    });
});
```

Let's look at the local request specific events

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

The URL, request type and response format

AjaxEvents.html

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

An event triggered before the request is sent to the server

AjaxEvents.html

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

You can modify the request object here

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

Amongst other stuff you can override the mime type and set new headers

AjaxEvents.html

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

Called on successful request

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

Called when the request fails

AjaxEvents.html

```
$.ajax({
    url: 'data.txt',
    type: 'GET',
    dataType: 'json',
    beforeSend: function(xhr) {
        console.log('beforeSend');
        xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
        xhr.setRequestHeader('Accept-Language', 'en-US');
    },
    success: function() {
        console.log('success');
    },
    error: function() {
        console.log('error');
    },
    complete: function(xhr) {
        console.log('complete');
        console.log(xhr.getAllResponseHeaders());
    }
});
```

Called on both request success and failure

AjaxEvents.html

```
$(document).ready(function () {
    $('.ajax').click(function() {
        $.ajax({
            url: 'data.txt',
            type: 'GET',
            dataType: 'json',
            beforeSend: function(xhr) {
                console.log('beforeSend');
                xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
                xhr.setRequestHeader('Accept-Language', 'en-US');
            },
            success: function() {
                console.log('success');
            },
            error: function() {
                console.log('error');
            },
            complete: function(xhr) {
                console.log('complete');
                console.log(xhr.getAllResponseHeaders());
            }
        });

    });

    $('#loading').hide();
    $(document).bind("ajaxSend", function(){
        $("#loading").show();
    }).ajaxComplete(function() {
        setTimeout(function() {
            $("#loading").hide();
        }, 8000);
    });
});
```

Now for global Ajax events

AjaxEvents.html

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

The ajaxSend event is triggered before any request is run

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

Show the loading indicator just as the request is sent

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

This ajaxComplete event will be triggered every time an Ajax event completes

AjaxEvents.html

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

Hide the indicator here

AjaxEvents.html

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

We hide on a timeout so that our example runs long enough for you to see the indicator

AjaxEvents.html

```
$('#loading').hide();
$(document).bind("ajaxSend", function(){
    $("#loading").show();
}).ajaxComplete(function() {
    setTimeout(function() {
        $("#loading").hide();
    }, 8000);
});
```

This timeout is purely for clarity it should not exist in real code

AjaxEvents.html

Example44Demo