jQuery Basics

This is the final module in the Coding and Web Fundamentals course and we're going out in style with a look at one of the fastest and most widely used tools on the web – jQuery.

We'll see how to use jQuery to work with parts of a page and provide the user with a truly interactive experience.

There's lots of hands-on exercises in this module, so get those fingers ready on the keyboard!

Review of Previous Module

In the previous module we continued looking at JavaScript, and in particular some basic event handling.

We also covered the switch conditional structure and the while loop.

Finally we looked at the use of arrays for storing multiple related items in a structured way.

Topics for this module

In this module going to take a look at jQuery. Topics include the following:

- Introduction to jQuery
- Installing jQuery
- Using jQuery to select elements
- Using jQuery to update elements
- Using event handlers with jQuery
- Hiding and showing elements in a page
- DOM interaction – adding elements

Introduction to jQuery

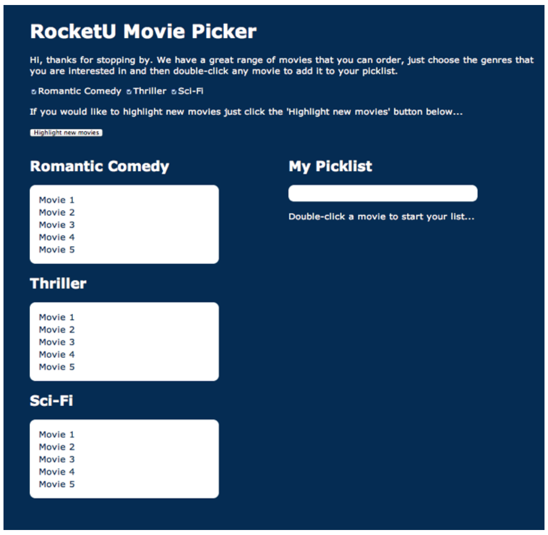
In today's module we are going to use some HTML and CSS files as a starting point and then use jQuery to add some functionality.

First of all create a new folder for this module. Name the folder 'movies' and then create a sub folder named 'assets' and then two sub folders under that named 'css' and 'js'

```
    movies
       |
    assets
   |       |
 css      js
```

Put the 'index.html' file in the movies folder,and then the 'application.css' file in the 'css' folder and the 'application.js' and 'jquery-1.10.2.min.js' files in the 'js' folder. Open the index page and check that it looks similar to the following slide...

Introduction to jQuery



RocketU Movie Picker

Hi, thanks for stopping by. We have a great range of movies that you can order, just choose the genres that you are interested in and then double-click any movie to add it to your picklist.

☑ Romantic Comedy ☑ Thriller ☑ Sci-Fi

If you would like to highlight new movies just click the 'Highlight new movies' button below...

Romantic Comedy

Movie 1
Movie 2
Movie 3
Movie 4
Movie 5

Thriller

Movie 1
Movie 2
Movie 3
Movie 4
Movie 5

Sci-Fi

Movie 1
Movie 2
Movie 3
Movie 4
Movie 5

My Picklist

Double-click a movie to start your list...

Installing jQuery

Before we can use jQuery on our pages and sites we need to make it available.

In previous modules we looked at how to import an external script that we had written.

Now we're going to do the same thing but instead of writing a new script to import we're going to download the jQuery library.

The official jQuery site is at:

http://jquery.com

Let's go to the site now...

Installing jQuery

On the jQuery site click on the 'Download' option and, on the next page, click the following link:

Download the compressed, production jQuery 1.10.2

This will display a page of minimized code. Copy the code and then paste it into the empty 'jquery-1.10.2.min.js' file.

In your index.html file add the following just before the closing body tag:

```
<script src="assets/js/jquery.1.10.2.min.js"></script>
<script src="assets/js/application.js"></script>
```

Reload the page and check that it still looks the same.

Your instructor will talk through this with you...

Using jQuery to select elements

In a previous module we looked at a way that we could select an element in the DOM with a statement similar to the following:

```
document.getElementById('field1').value;
```

This is quite a complex expression.

In jQuery we can use an easier way to select elements – in fact the same way that we do in CSS – by the use of selectors.

Suppose we want to use CSS to target all unordered lists in a page.

We would use the following selector:

```
ul { /* CSS rules here... */ }
```

Using jQuery to select elements

In CSS we would use the following selector to find all ul elements:

```
ul { /* CSS rules here... */ }
```

And if we wanted to target all li elements for an ordered list with a class of ‘people’ we would use:

```
ul.people li {...}
```

If we wanted to find similar elements with jQuery we would use:

```
$(“ul”)

$(“ul.people li”)
```

Using jQuery to select elements

jQuery uses the following syntax for finding/selecting elements:

- First of all a dollar symbol
- Then, in parenthesis, a quoted string that represents a CSS selector

```
$(“ul, ol”)
```

The result will be to find/select all elements that match the selector, which in the example above would be all unordered and ordered lists.

What would we use to select all <p> elements inside a <div> with an id of ‘sidebar‘ ?

Using jQuery to select elements

Once we have used the initial syntax to find an element, or group of elements, we can add functions to carry something out.

Some examples:

// Hide all <ul> elements

```
$(“ul”).hide();
```

// show all <li> elements for a <ul> with a class of ‘people’

```
$(“ul.people li”).show();
```

Exercise

Edit the ‘application.js’ page and add the following statement:

```
$("#picklist ul").hide();
```

Reload the page – what is different?

The document.ready event

In the previous exercise you added a statement that will hide the empty picklist.

It worked, but there can be situations where a piece of jQuery executes before the DOM is fully created, and this may cause issues.

To be safe we'll make sure that the document is ready before trying to do anything..

Exercise

Edit the code in your application.js file as follow:

```
$(document).ready(function() {

  $("#picklist ul").hide();

});
```

Reload the page – it should still work.

The document.ready event

What we have done now is to enclose the jQuery statement inside a document ready event handler.

This is a special jQuery function that will only be called when the DOM is completely created and ready for use.

```
$(document).ready(function() {

  // Any code in here will only be executed when the
  // document is read

});
```

Adding event handlers to other

In the previous module we added event handler code to HTML elements by defining an attribute for the element.

For example:

```
<button onclick="greetUser();">Click me</button>
```

This works, but it’s a bit like adding inline CSS styles – it can make the code difficult to maintain – and we are mixing HTML code and JavaScript in the same file, which we should try to avoid if we can.

With jQuery we can use the selector syntax that we saw before to identify the element or elements that we want to add an event handler to and then specify the code that should be executed.

Let’s start with the button that highlights the new releases...

Exercise

Edit the code in your application.js file again and add the following code in red inside the document ready handler after the existing statement:

```
$(document).ready(function() {

   $("#picklist ul").hide();

    $("#btn-newmovies").click(function() {

      window.alert('New movies button clicked');

   });

});
```

Reload the page and click the button – what happens?

Adding event handlers to other

In the previous exercise we added an event handle code to to the button.

First we used the selector syntax to identify the button:

```
$("#btn-newmovies")
```

Then we added the name of the event that we wanted to listen for – in this case the ‘click’ event – and then a definition of a function to be called when the event occurs:

```
 $("#btn-newmovies").click(function() {

    // Code to be executed goes here

 });
```

Exercise

Edit the code in your application.js file again and change the code for the button event handler as follows. Remove the alert as well.

```
$("#btn-newmovies").click(function() {

    var btnText = $(this).text();

    if (btnText == 'Highlight new movies') {
        $("li.new").css('background-color', 'lightgreen');
        $(this).text('Remove highlights');
    } else {
        $("li.new").css('background-color', 'transparent');
        $(this).text('Highlight new movies');
    }
});
```

Reload the page and click the button – what happens now?

Adding event handlers

In this exercise we used some additional methods to implement the functionality:

```
var btnText = $(this).text();
```

This statement uses a special keyword – **this**

The **this** keyword is used to identify whichever element has triggered an event, in this case the button.

The **text()** function then returns the button's text.

Finally, the text is assigned to a variable:

```
var btnText = $(this).text();
```

Getting and Setting Attributes

Once we have the button's text stored in a variable we can test it with a conditional expression.

If the text is 'Highlight new movies' then we do the following:

- Change the background color of any list items that have a class of 'new'
- Change the button's text to 'Remove highlights'

```
if (btnText == 'Highlight new movies') {

  $("li.new").css('background-color', 'lightgreen');
  $(this).text('Remove highlights');

}
```

Getting and Setting Attributes

jQuery's **css()** function is used to **apply CSS styling to an element.**

Inside the parenthesis we list the name of the CSS property and the value to be applied.

```
$("li.new").css('background-color', 'lightgreen');
```

Next we used the text() function again, but this time to change the button's text:

```
$(this).text('Remove highlights');
```

If the **text()** function is used **without anything in the parenthesis** it will **return the current text value** for an element. If a **string is included** in the parenthesis then the effect will be to **change the element's text** to the new string.

Getting and Setting Attributes

Next, we added an **else** section to a conditional structure to deal with the situation when the button's text was no longer 'Highlight new movies – in other words if the new movies were already highlighted and the button text had been changed.

```
 else {

        $("li.new").css('background-color', 'transparent');
        $(this).text('Highlight new movies');

    }
```

In this case we remove the highlighting and change the text back to the original.

Hiding and showing elements

Now we're going to add the code to hide and show the individual lists when the checkboxes are used.

In this case we want to add an event handler to each checkbox.

We can use selector syntax to identify all the checkboxes in one go:

```
$("input[type='checkbox']")
```

To detect when a checkbox is checked or unchecked we use a 'change' event:

```
$("input[type='checkbox']").change(function() {
   //code here
});
```

Exercise

Edit the code in your application.js file again and add the following code inside the document ready handler, after the button event handler:

```
$("input[type='checkbox']").change(function() {

   var isChecked = $(this).is(':checked');
   var movieGenre = $(this).attr('name');

   if (isChecked) {
       $("ul." + movieGenre).show();
   } else {
       $("ul." + movieGenre).hide();
   }

});
```

Reload the page and click the button – what happens?

Hiding and showing elements

The lists now get hidden and redisplayed when a checkbox is clicked, but the headings are still showing.

We need a way of selecting a <h2> element that comes before a list. This actually can't be done easily with a CSS selector but luckily jQuery has a solution:

The prev() function will select the element before another element, so:

```
$("ul." + movieGenre).prev().show();
```

Will select the element that comes immediately before an unordered list with a particular class list – in other words the heading.

Exercise

Edit the code in your application.js file again and add the following code inside the if structure:

```
if (isChecked) {
   $("ul." + movieGenre).show();
     $("ul." + movieGenre).prev().show();
} else {
   $("ul." + movieGenre).hide();
     $("ul." + movieGenre).prev().show();
}
```

Reload the page and try again.

Adding elements to the DOM

Great, our button and checkboxes are working – all that's left is to add the functionality to add movies to our picklist.

To add an element inside another element – such as adding a <li> element inside a <ul> – we can use jQuery's append() function:

`$("#picklist ul").append(`*html goes here...*`);`

We just need to create the appropriate HTML and include it in the parenthesis.

Let's do that now...

Exercise

Edit the code again and add the following code inside the document ready handler after the checkbox event handler.

This time we are adding a dblclick event handler – to listen for double-clicks.

```
$(".movies li").dblclick(function() {

    var movieName = $(this).text();

    $("#picklist ul").append("<li>" + movieName + "</li>");
    $("#picklist ul").show();
    $("#picklist ul + p").hide();

});
```

Reload the page and double-click a movie name.

The final page

That's it – our page is now using jQuery to provide the user with a range of interactive options.

This is really just a flavor of what can be done with jQuery – there are lots more effects and features that can be added to sites and applications.

Some useful resources are listed below:

http://jquery.com
http://www.w3schools.com/jquery/
http://plugins.jquery.com

Summary and Q&A

In this module we took a first look at jQuery.

We said that jQuery is actually JavaScript that has been used to create a library of functions that can be used to assist with cross browser compatibility and to provide a simpler way of working with elements in a document.

We looked how we can add event handlers to individual elements or groups of elements in one go and then use those to apply different effects such as hiding and showing, chaging text, applying CSS and adding new items to lists.

jQuery's motto is:

Write less, do more!

Questions

Do you have any questions before we finish the course

?

Don't worry, if you think of something later then just ask!

And we’re done

That’s the end of the Coding and Web Fundamentals course.

We hope that you’ve enjoyed it and wish all the best with your future in web development.

If you are attending the Bootcamps we look forward to continuing the journey with you there!