Learn how Script Tags and Document Ready Work

Now we're ready to learn jQuery, the most popular JavaScript tool of all time. Don't worry about JavaScript itself - we will cover it soon.

Before we can start using jQuery, we need to add some things to our HTML.

First, add a script element at the top of your page. Be sure to close it on the following line.

Your browser will run any JavaScript inside a script element, including jQuery.

Inside your script element, add this code: $(document).ready(function() {to your script. Then close it on the following line (still inside your script element) with: });

We'll learn more about functionslater. The important thing to know is that code you put inside this function will run as soon as your browser has loaded your page.

This is important because without your document ready function, your code may run before your HTML is rendered, which would cause bugs.

Target HTML Elements with Selectors Using jQuery

Now we have a document ready function.

Now let's write our first jQuery statement. All jQuery functions start with a $, usually referred to as a dollar sign operator, or as bling.

jQuery often selects an HTML element with a selector, then does something to that element.

For example, let's make all of your button elements bounce. Just add this code inside your document ready function:

$("button").addClass("animated bounce");

Note that we've already included both the jQuery library and the Animate.css library in the background so that you can use them in the editor. So you are using jQuery to apply the Animate.css bounce class to your button elements.

Additionally make sure to use $("button").addClass("animated bounce"); instead of $('button').addClass("animated bounce"); since single-quote selectors will not pass our tests.

Target Elements by Class Using jQuery

You see how we made all of your button elements bounce? We selected them with $("button"), then we added some CSS classes to them with .addClass("animated bounce");.

You just used jQuery's .addClass()function, which allows you to add classes to elements.

First, let's target your div elements with the class well by using the $(".well") selector.

Note that, just like with CSS declarations, you type a . before the class's name.

Then use jQuery's .addClass()function to add the classes animatedand shake.

For example, you could make all the elements with the class text-primary shake by adding the following to your document ready function:

$(".text-primary").addClass("animated shake");

Target Elements by ID Using jQuery

You can also target elements by their id attributes.

First target your button element with the id target3 by using the $("#target3") selector.

Note that, just like with CSS declarations, you type a # before the id's name.

Then use jQuery's .addClass()function to add the classes animatedand fadeOut.

Here's how you'd make the buttonelement with the id target6 fade out:

$("#target6").addClass("animated fadeOut").

Target the same element with multiple jQuery Selectors

Now you know three ways of targeting elements: by type: $("button"), by class: $(".btn"), and by id $("#target1").

Although it is possible to add multiple classes in a single .addClass() call, let's add them to the same element in *three separate ways*.

Using .addClass(), add only one class at a time to the same element, three different ways:

Add the animated class to all elements with type button.

Add the shake class to all the buttons with class .btn.

Add the btn-primary class to the button with id #target1.

**Note**  
You should only be targeting one element and adding only one class at a time. Altogether, your three individual selectors will end up adding the three classes shake, animated, and btn-primary to #target1.

Remove Classes from an element with jQuery

In the same way you can add classes to an element with jQuery's addClass() function, you can remove them with jQuery's removeClass() function.

Here's how you would do this for a specific button:

$("#target2").removeClass("btn-default");

Let's remove the btn-default class from all of our button elements.

Change the CSS of an Element Using jQuery

We can also change the CSS of an HTML element directly with jQuery.

jQuery has a function called .css()that allows you to change the CSS of an element.

Here's how we would change its color to blue:

$("#target1").css("color", "blue");

This is slightly different from a normal CSS declaration, because the CSS property and its value are in quotes, and separated with a comma instead of a colon.

Delete your jQuery selectors, leaving an empty document ready function.

Select target1 and change its color to red.

Disable an Element Using jQuery

You can also change the non-CSS properties of HTML elements with jQuery. For example, you can disable buttons.

When you disable a button, it will become grayed-out and can no longer be clicked.

jQuery has a function called .prop()that allows you to adjust the properties of elements.

Here's how you would disable all buttons:

$("button").prop("disabled", true);

Disable only the target1 button.

Change Text Inside an Element Using jQuery

Using jQuery, you can change the text between the start and end tags of an element. You can even change HTML markup.

jQuery has a function called .html()that lets you add HTML tags and text within an element. Any content previously within the element will be completely replaced with the content you provide using this function.

Here's how you would rewrite and emphasize the text of our heading:

$("h3").html("<em>jQuery Playground</em>");

jQuery also has a similar function called .text() that only alters text without adding tags. In other words, this function will not evaluate any HTML tags passed to it, but will instead treat it as the text you want to replace the existing content with.

Change the button with id target4by emphasizing its text.

Remove an Element Using jQuery

Now let's remove an HTML element from your page using jQuery.

jQuery has a function called .remove() that will remove an HTML element entirely

Remove element target4 from the page by using the .remove()function.

Use appendTo to Move Elements with jQuery

Now let's try moving elements from one div to another.

jQuery has a function called appendTo() that allows you to select HTML elements and append them to another element.

For example, if we wanted to move target4 from our right well to our left well, we would use:

$("#target4").appendTo("#left-well");

Move your target2 element from your left-well to your right-well.

one an Element Using jQuery

In addition to moving elements, you can also copy them from one place to another.

jQuery has a function called clone()that makes a copy of an element.

For example, if we wanted to copy target2 from our left-well to our right-well, we would use:

$("#target2").clone().appendTo("#right-well");

Did you notice this involves sticking two jQuery functions together? This is called function chaining and it's a convenient way to get things done with jQuery.

Clone your target5 element and append it to your left-well.

Target the Parent of an Element Using jQuery

Every HTML element has a parentelement from which it inheritsproperties.

For example, your jQuery Playground h3 element has the parent element of <div class="container-fluid">, which itself has the parent body.

jQuery has a function called parent() that allows you to access the parent of whichever element you've selected.

Here's an example of how you would use the parent() function if you wanted to give the parent element of the left-well element a background color of blue:

$("#left-well").parent().css("background-color", "blue")

Give the parent of the #target1element a background-color of red.

Target the Children of an Element Using jQuery

Many HTML elements have children which inherit their properties from their parent HTML elements.

For example, every HTML element is a child of your body element, and your "jQuery Playground" h3element is a child of your <div class="container-fluid"> element.

jQuery has a function called children() that allows you to access the children of whichever element you've selected.

Here's an example of how you would use the children() function to give the children of your left-wellelement the color of blue:

$("#left-well").children().css("color", "blue")

Give all the children of your #right-well element a color of orange.

Target a Specific Child of an Element Using jQuery

You've seen why id attributes are so convenient for targeting with jQuery selectors. But you won't always have such neat ids to work with.

Fortunately, jQuery has some other tricks for targeting the right elements.

jQuery uses CSS Selectors to target elements. target:nth-child(n) css selector allows you to select all the nth elements with the target class or element type.

Here's how you would give the third element in each well the bounce class:

$(".target:nth-child(3)").addClass("animated bounce");

Make the second child in each of your well elements bounce. You must target the children of element with the target class.

Target Even Numbered Elements Using jQuery

You can also target all the even-numbered elements.

Here's how you would target all the odd-numbered elements with class target and give them classes:

$(".target:odd").addClass("animated shake");

Note that jQuery is zero-indexed, meaning that, counter-intuitively, :odd selects the second element, fourth element, and so on.

Try selecting all the even-numbered elements and giving them the classes of animated and shake.

Use jQuery to Modify the Entire Page

We're done playing with our jQuery playground. Let's tear it down!

jQuery can target the body element as well.

Here's how we would make the entire body fade out: $("body").addClass("animated fadeOut");

But let's do something more dramatic. Add the classes animated and hinge to your body element.