**jQuery**

See It to Believe It

So far, we've built web pages using HTML and styled them using CSS. Our pages look great, but they're not interactive —we can't drag elements around the page, open and close sliding panels, animate HTML elements, or add new elements to our HTML pages simply by clicking a button.

All that's about to change, though. In this track, you're going to learn jQuery, which will allow you to do all these things and more.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Magic!</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div id="red"></div>

<div id="blue"></div>

<div id="yellow"></div>

<div id="green"></div>

</body>

</html>

**CSS**

div {

height:100px;

width:100px;

display: inline-block;

}

#red {

background-color:#FF0000;

}

#blue {

background-color:#0000FF;

}

#yellow {

background-color:#E2BE22;

}

#green {

background-color:#008800;

}

**jQuery Script**

$(document).ready(function() {

$('div').mouseenter(function() {

$(this).animate({

height: '+=10px'

});

});

$('div').mouseleave(function() {

$(this).animate({

height: '-=10px'

});

});

$('div').click(function() {

$(this).toggle(1000);

});

});

The Document Object Model

To get the most out of jQuery, we should review how an HTML page is put together.

An HTML document is structured according to the Document Object Model, or DOM. It's by interacting with the DOM that jQuery is able to access and modify HTML.

The DOM consists of every element on the page, laid out in a hierarchical way that reflects the way the HTML document is ordered. Remember how we could [think of the HTML document as a tree](http://www.codecademy.com/courses/web-beginner-en-WF0CF/1#!/exercises/0)? You can think of the DOM the same way. Just as with an HTML document, elements in the DOM can have parents, children, and siblings.

Check out the code in **script.js**. We'll explain the code more soon, but for now the main idea is that when the HTML document loads, the code will select an HTML element and make it disappear.

We'll want to target the div element here, so on [line 2](javascript:void(0)) inside the $( ), add "div".

Inside the empty $(), you'll want to enter the name of the HTML element you want to fade away when clicked. For instance, if it were a paragraph, we'd type

$('p').fadeOut(1000);

In this case, though, we want to make our div disappear.

Don't forget the quotes!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height:100px;

width:100px;

background-color:#FF0000;

border-radius:5px;

}

**jQuery Script**

$(document).ready(function() {

$("div").fadeOut(1000);

});

Changing Targets

Don't be intimidated by the amount of code you're seeing—we'll go through it piece by piece to make sure you understand it thoroughly.

Just like the CSS div refers to the HTML element <div>, the jQuery 'div' refers to the same HTML element <div>. You can think of the element name passed to jQuery as identical to the CSS element, only wrapped in quotes. So, for instance, you could target anything of class button with

$('.button').someAction

As you'll remember, .button in your CSS file is how you'd target anything of class="button" in your HTML file.

Put a selector (remember [CSS selectors](http://www.codecademy.com/courses/web-beginner-en-WF0CF)?) inside $() so instead it points to the ID #green.

Remember to use the pound sign (#) for IDs!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div></div>

<div id="green"></div>

</body>

</html>

**CSS**

div {

height:100px;

width:100px;

background-color:#FF0000;

border-radius:5px;

margin-bottom:5px;

}

#green {

background-color:#008800;

}

**jQuery Script**

$(document).ready(function() {

$('#green').fadeOut(1000);

});

What is jQuery?

jQuery is a library, or set of helpful add-ons, to the JavaScript programming language. It may seem counterintuitive to learn how to use a library before learning the actual language, but there are a few good reasons for this.

1. It takes a while to become comfortable with JavaScript, and it's trickier to manipulate HTML elements directly with JavaScript than with jQuery. In order to help you build awesome websites faster, we're starting you off with jQuery.
2. jQuery provides a simple interface for the underlying JavaScript. It's easier for many users to learn jQuery first, then dive into the nitty-gritty JavaScript details later.
3. jQuery is much better at giving you immediate, visual results than regular JavaScript. By the end of this lesson, you'll have built your own interactive button!

On the script.js tab, add the missing pieces on [line 2](javascript:void(0)) so that your jQuery code calls fadeOut on the div with the ID #notready.

The $() should contain the element you want to affect; the fadeOut keyword should be between $(). and (1000);.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>What Say You?</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div id="ready">I'm ready!</div>

<div id="notready">You'll never take me alive, jQuery!</div>

</body>

</html>

**CSS**

div {

height:100px;

width:100px;

border-radius:5px;

display: inline-block;

text-align: center;

vertical-align: middle;

font-family: Verdana, Arial, Sans-Serif;

margin-right:5px;

}

#ready {

background-color:#008800;

color:#FFFFFF;

}

#notready {

background-color:#FF0000;

color:#FFFFFF;

}

**jQuery Script**

$(document).ready(function() {

$('#notready').fadeOut(1000);

});

Linking Your HTML and JavaScript Files

Great! Now we need to link our HTML page to our jQuery script so our jQuery magic will affect our HTML.

Just like we need a <link> tag to connect our HTML and CSS, we need a <script> tag to connect our HTML and jQuery. The tag looks like this:

<script type="text/javascript" src="script.js"></script>

Note that the <script> tag is not self-closing; it requires a closing </script> tag.

Add a <script> tag to index.html. Its type should be "text/javascript" and its src should be "script.js".

You don't strictly need the type attribute in HTML5, but we're including it here so you're not surprised if you see it in other people's code.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<!--Add your script tag here.-->

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div></div>

</body>

</html>

Getting Started

Next, we'll need to start up our jQuery magic using the $(document).ready(); syntax you've seen. It works like this:

* $() says, "hey, jQuery things are about to happen!"
* Putting document between the parentheses tells us that we're about to work our magic on the HTML document itself.
* .ready(); is a function, or basic action, in jQuery. It says "hey, I'm going to do stuff as soon as the HTML document is ready!"
* Whatever goes in .ready()'s parentheses is the jQuery event that occurs as soon as the HTML document is ready.

So,

$(document).ready(something);

says: "when the HTML document is ready, do something!" (We'll show you how to replace something with an action in the next exercise.)

Note that .ready(); ends with a semicolon. This tells jQuery that you're done giving it a command.

Add a $(document).ready(); call in script.js. Remember, the document is special: it's not an HTML element like <p> or <div>, so it doesn't go in quotes.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<!--Add your script tag here.-->

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**jQuery Script**

$(document).ready();

The Functional Approach

Perfect! Now we need to put something inside our ready() function.

Remember, when we say "function," you can think "action." Functions are the basic unit of doing work in jQuery.

For this reason, jQuery includes a function keyword. The syntax looks like this:

function(){

jQuery magic;

}

If we add our function inside our .ready(), jQuery will run the code in our function as soon as the HTML document loads. The syntax would then look like this:

$(document).ready(function() {

jQuery magic;

});

Remember, we end our jQuery statements with a semicolon.

Place a function(){} inside your .ready(). You don't need to put anything inside your function's curly braces ({}) yet—we'll get to that in the next exercise.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<!--Add your script tag here.-->

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

});

Electr(on)ic Slide

Almost there! Now we just need to include an action in the body of our function. In this case, we'll make a header div for our HTML page slide into view.

Between the {}s of our function(), we'll want to turn our div into a jQuery object so jQuery can manipulate it. Much like we use .ready() on $(document), we'll use the .slideDown() action on our div jQuery object.

Inside .slideDown's parentheses, we'll want to tell it how quickly to slide down. Let's enter 'slow' (include the quotes).

On [line 2](javascript:void(0)) in script.js, make 'div' into a jQuery object and call the .slideDown() action on it. Inside slideDown()'s parentheses, type 'slow' so jQuery knows how quickly to slide your div down.

Your jQuery should look something like this:

$(document).ready(function() {

$('div').action(howfast);

});

Where action is the slideDown action and howfast should be 'slow'.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Ta Daaa!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div><br/><strong>I'm the Header!</strong></div>

</body>

</html>

**CSS**

div {

position: absolute;

margin: -10px 0 0 -8px;

height: 60px;

width: 100%;

background-color: #008800;

color: #FFFFFF;

font-family: Verdana, Arial, Sans-Serif;

display:none;

}

**jQuery Script**

$(document).ready(function() {

$('div').slideDown('slow');

});

Mid-Lesson Breather

Great work! We've covered a lot so far. Let's take a second to review what we've learned:

* What jQuery is and how it can manipulate HTML elements
* jQuery syntax

At the Ready!

All right! Let's get started.

1. In your index.html tab, add a <script> tag to connect your HTML to script.js.
2. In your script.js tab, put document into a jQuery object and call .ready().

Click "Stuck? Get a hint!" for examples.

Remember, your <script> tag should be of the form

<script type='text/javascript' src='script.js'></script>

You can turn document into a jQuery object by putting it inside $(). To call .ready you can add .ready(); to the end of your document jQuery object, like this:

$(document).ready();

**HTML**

<html>

<head>

<title>Button Magic</title>

<script type="text/javascript" src="script.js"></script>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

</head>

<body>

<div><br/><strong>Click Me!</strong></div>

</body>

</html>

**CSS**

div {

height: 60px;

width: 100px;

border-radius: 5px;

background-color: #69D2E7;

text-align: center;

color: #FFFFFF;

font-family: Verdana, Arial, Sans-Serif;

opacity: 0.5;

}

**jQuery Script**

$(document).ready();

Get Yourself In...

Great! Next, let's include our function keyword and two new actions together, mouseenter() and fadeTo().

mouseenter() does what you might expect: it produces a change when your mouse enters a given HTML element. For example,

$(document).ready(function() {

$('div').mouseenter(function() {

$('div').hide();

});

});

would hide every <div> on the page as soon as you mouse over one. (We'll find out how to affect just one <div> among many in the next lesson.) For now, we only have one <div>, so this setup is okay.

Instead of hide(), however, we'll place fadeTo() *inside* mouseenter(). fadeTo() takes two arguments, or inputs, between its parentheses, separated by a comma: the speed at which to fade, and the opacity (or transparency) to fade to. For example,

fadeTo('fast', 0.25);

would quickly fade the target element to 25% of its original opacity, making it very light-colored.

Using the examples above, make it so your 'div' fades to 1 (100%) opacity when your mouse enters the 'div'. Make the animation speed 'fast'. (Make sure to give fadeTo() its inputs in order—speed, then opacity.)

Click Save & Submit Code, then slowly mouse over your 'div' to see the effect!

Your code should look very much like the code in the instructions, only instead of

$('div').hide();

you should use .fadeTo('fast', 1);.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Button Magic</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div><br/><strong>Click Me!</strong></div>

</body>

</html>

**CSS**

div {

height: 60px;

width: 100px;

border-radius: 5px;

background-color: #69D2E7;

text-align: center;

color: #FFFFFF;

font-family: Verdana, Arial, Sans-Serif;

opacity: 0.5;

}

**jQuery Script**

$(document).ready(function() {

$('div').mouseenter(function() {

$('div').fadeTo('fast', 1);

});

});

...and Get Yourself Out!

Excellent! Your button looks great now—it stands out when the user mouses over it.

However, you'll notice that when you take your cursor off the button, it stays dark. What we really want is for our button to become light again when our mouse leaves.

You might have guessed that jQuery includes a mouseleave() action. If so, you're right! (If you're curious, you can learn more about these actions in the [jQuery documentation](http://api.jquery.com/mouseleave/).)

Between the }); that ends your mouseenter() and the }); that ends your $(document).ready(), add a mouseleave() that uses fadeTo to return your div's opacity to 0.5. Like the previous fadeTo, it should have 'fast' as its first input.

Click Save & Submit Code, then slowly mouse over your 'div' to see the effect!

This new code should look almost identical to your previous code, except it uses mouseleave() instead of mouseenter() and sets the opacity to 0.5 instead of 1.

Note that you need to mouse over your 'div' to continue the exercise!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Button Magic</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div><br/><strong>Click Me!</strong></div>

</body>

</html>

**CSS**

div {

height: 60px;

width: 100px;

border-radius: 5px;

background-color: #69D2E7;

text-align: center;

color: #FFFFFF;

font-family: Verdana, Arial, Sans-Serif;

opacity: 0.5;

}

**jQuery Script**

$(document).ready(function() {

$('div').mouseenter(function() {

$('div').fadeTo('fast',1);

});

$('div').mouseleave(function() {

$('div').fadeTo('fast', 0.5);

});

});

Eureka!

Great job! You've gone from static HTML and CSS to dynamic jQuery goodness in no time flat.

Now you can see some of the power behind jQuery, and as you learn more, you'll be able to work all kinds of amazing interactive magic.

Feel free to play around with your new button (what happens when you change 'fast' to 'slow? What happens when you change the opacity values?).

Functions, Part I: $(document).ready

Functions are the basic unit of action in jQuery. The main entry point of most jQuery applications is a block of code that looks like this:

$(document).ready(function() {

Do something

});

Let's go through it bit by bit.

* $(document) is a jQuery object. The $() is actually a function in disguise; it turns the document into a jQuery object.
* .ready() is a type of function; you can think of it as sort of a helper that runs the code inside its parentheses as soon as the HTML document is ready.
* function(){} is the action that .ready() will perform as soon as the HTML document is loaded. (In the above example, the Do something placeholder is where those actions would go.)

Practice makes perfect! Set up your jQuery code in the script.js tab with $(document).ready() and pass it a function that will immediately hide the orange div on the Result tab. (You can use the jQuery .hide() effect to do this.)

You should call .hide() on your jQuery object, like so:

$('selector').hide();

Where 'selector' is the HTML element you want to select.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

background-color: #FA6900;

border-radius: 5px;

}

**jQuery Script**

$(document).ready( function() {

$('div').hide();

});

Functions, Part II: Functions Explained

A function is made up of three parts: the function keyword, any inputs that function takes (they go between the ()s and are separated by commas if there are more than one), and whatever actions the function should perform (these go between the {}s). The general form is:

function(input1, input2, etc) {

Do a thing

Do another thing

Do yet another thing!

}

One of the nice things about jQuery is that you can give a function just about anything as an input—even another function! That's why .ready() can take function between its parentheses; it's taking a function as input.

(If you're interested, you can learn more about functions in the [JavaScript track](http://codecademy.com/tracks/javascript).)

Check out the code in script.js. It includes a new jQuery action, .click(), that we'll learn more about soon!

See how we pass functions to both .ready() and .click() as inputs? When you're ready, click Save & Submit Code, then click on the orange div in the Result tab to see the fadeOut() effect.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

background-color: #FA6900;

border-radius: 5px;

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$('div').fadeOut('slow');

});

});

Variables

Variables are a place for us to store information for use at a later time. Variables can hold any type of information you work with, so just think of them as containers.

The single = sign is used to assign values. For instance, in jQuery, you can write

var lucky = 7;

var name = "Codecademy";

var $p = $('p');

Our first variable contains a number, 7, while the second variable contains some text, "Codecademy". Our 3rd variable stores the result of a jQuery selector $('p') in the variable $p . As you can see, this is just a handy way to save this information for later.

Why would we do this? Well, up until now we haven't had to modify anything more than once. If we wanted to change the webpage based on new information, we would need to store that information in variables. Maybe you want to create a loading page and have it vanish as you receive that information. It'd be a good idea to use variables.

Check out the preview window. All looks well and good, except for #4—that's nonsense!

Create a variable called $target in script.js and use the = sign to assign it to the jQuery selector that represents #4 in the ordered list. When you run your code, it should fade away!

You're going to need your CSS knowledge for this one and there is more then one way to achieve this. Check the Hint if you need a reminder!

Remember, you can modify things directly using an id or class.

You may also modify things indirectly by using :nth-child() or :last-child

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Simplify, Simplify</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div> Remember!

<ul>

<li>

<ol>

<li>Start with the function keyword</li>

<li>Inputs go between ()</li>

<li>Actions go between {}</li>

<li>jQuery is for chumps!</li>

</ol>

</li>

<li>Inputs are separated by commas.</li>

<li>Inputs can include other functions!</li>

</ul>

</div>

</body>

</html>

**jQuery Script**

// Write your jQuery code on line 3!

$(document).ready(function() {

var $target = $('ol li:nth-child(4)');

$target.fadeOut('fast');

});

$p vs $('p')

You probably noticed that we used both $ and $() in the last exercise:

var $p = $('p');

There's a subtle difference between the two.

$p is just a variable name. There is *no* magic associated with the $ in $p; it's just a convention for saying, "hey, this variable contains a jQuery object." We could call $p anything we want: $paragraph, paragraph, space\_cows, whatever! It's just helpful for people reading our code to see $p, since this is a concise way of saying "hey, there's a 'p' jQuery object in here."

$(), on the other hand, *is* magic. This is the function in disguise that creates jQuery objects. If you're making a jQuery object, you gotta use it!

Remember when we told you to ignore the paragraph = in the last lesson? Well, now you know what it does! (You also know that it should have been $p.)

Create a variable $div, and assign it to a div jQuery object using =. No need to use $(document).ready() or any additional code!

Remember, you can set a variable like this:

variableName = jQueryObject;

And all jQuery objects are just selector names (like 'p' or 'div') inside a $().

Don't forget: jQuery commands end with semicolons!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**jQuery Script**

// Write your jQuery code below!

var $div = $('div');

Using Functions to Select HTML Elements

Now that you know more about how functions work, you understand that when we have something like

$(document).ready(function() {

$('div').hide();

});

we're passing .ready() a function (which itself takes no inputs; that's why its () are empty) and that function's job is to .hide() the div jQuery object.

Give .ready() a function(){} as an input. Inside your function's {}s, call fadeIn('slow') on the (currently invisible) div to bring it into view. That's right, there's a fade in, as well!

Remember the structure:

$(document).ready(function() {

$(thingToClick).event(function() {

$(thingToAffect).effect();

});

});

We're not clicking anything, so we'll just need

$(document).ready(function() {

$(thingToAffect).effect();

});

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Fade In!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div>Test</div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

background-color: #F38630;

display: none;

border-radius: 5px;

}

**jQuery Script**

$(document).ready(function() {

var $div = $(div);

$(div).fadeIn('slow');

});

Selecting by Class

We don't have to limit ourselves to selecting HTML elements like <p> and <div>; essentially, we can put any CSS selector in quotes and pass it into $(). This means we can select classes, too!

Recall that we can select classes in CSS by using a dot (.). If we have class="red" in our HTML, we can target it in CSS with .red. In jQuery, all we need to do is put '.red' in quotes, and we can pass it to $() to make a jQuery object.

Let's get some practice with the .click() action.

Finish the jQuery code so that it will make all four divs of class '.vanish' fadeOut() 'slow'ly when the button is .click()ed.

On [line 2](javascript:void(0)), you'll want to put the button element inside $() to make it a jQuery object.

On [line 3](javascript:void(0)), you'll want to select '.vanish' and tell it to .fadeOut() with the input 'slow'. Remember, by making '.vanish' the jQuery object, all .vanishs are affected. We'll learn how to affect just one element soon!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Vanishing Act</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div class="vanish"></div>

<div class="vanish"></div>

<div class="vanish"></div>

<div class="vanish"></div>

<br/><button>Click Me!</button>

</body>

</html>

**CSS**

.vanish {

height: 100px;

width: 100px;

display: inline-block;

background-color: #F38630;

border-radius: 5px;

}

**jQuery Script**

$(document).ready(function() {

$('button').click(function() {

$('.vanish').fadeOut('slow');

});

});

Selecting by ID

If we can select by class, it follows that we can also select by ID. We do this by putting the ID name (in quotes) inside $(). Just as we need the . for classes, we need the # for IDs. We could target id="header" like so:

$('#header');

The semicolon at the end is important—it's how jQuery knows we're done giving it a command. For now, a good rule of thumb is that you should put semicolons at the end of any line that does not end with an open {. (The editor will try to help you out with your semicolon placement, so pay attention to its warnings.) Examples of correct and incorrect semicolon use are in the Hint.

Update your jQuery code so that it only .fadeOut()s the div with the ID #blue.

Example of correct semicolon use:

$(document).ready(function() {

$('div').click(function() {

$('div').fadeOut('fast');

});

});

Example of **incorrect** semicolon use:

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

$('div').click(function(); {

$('div').fadeOut('fast')

})

})

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Vanishing Act</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div id="blue"></div>

<div></div>

<div></div>

<div></div>

<br/><button>Click Me!</button>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

display: inline-block;

background-color: #F38630;

border-radius: 5px;

}

#blue {

background-color: #A7DBD8;

}

**jQuery Script**

$(document).ready(function() {

$('button').click(function() {

$('#blue').fadeOut('slow');

});

});

Flexible Selections

Anything you can target with CSS, you can modify with jQuery. For example, we can apply a fadeTo() to a p selector like this:

$('p').fadeTo('slow', 0);

We can apply a fadeTo() to an li selector like this:

$('li').fadeTo('slow', 0);

We can apply a fadeTo() to both the p and li selectors like this:

$('p, li').fadeTo('slow', 0);

This is called a compound selector.

Let's use a compound selector to apply a fadeTo() to both the .pink and .red selectors.

Let's use a compound selector to apply a fadeTo() to both the .pink and .red selectors.

Remember we can apply a fadeTo() to two selectors like this:

$('selector1, selector2').fadeTo('slow', 0);

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div class='red'></div>

<div></div>

<div class='pink'></div>

<div></div>

</body>

</html>

**CSS**

div {

height: 50px;

width: 50px;

border-radius: 100%;

background-color: #FF9C5B;

display: inline-block;

}

.red {

background-color: #E84A5F;

}

.pink {

background-color: #FF847C;

}

**jQuery Script**

$(document).ready(function(){  
$(".pink, .red").fadeTo("slow", 0);  
});

'this' is Important!

In the last lesson, we had some code that looked like this:

$(document).ready(function() {

$('div').mouseenter(function() {

$('div').hide();

});

});

The second line is good: this tells us that when we mouse into a div, we should take a certain action. However, $('div').hide(); won't just hide the div you mouse into; it will hide *all* the divs on the page. How can we tell jQuery we only want to affect *this* particular div?

With this, of course!

The this keyword refers to the jQuery object you're currently doing something with. Its complete rules are a little tricky, but the important thing to understand is if you use an event handler on an element—that's the fancy name for actions like .click() and .mouseenter(), since they handle jQuery events—you can call the actual event that occurs (such as fadeOut()) on $(this), and the event will *only* affect the element you're currently doing something with (for example, clicking on or mousing over).

Check out the code in script.js. Instead of using fadeOut() on all 'div's, we're just using it on this div, where this refers to the div you're clicking on. Click Save & Submit Code, then click on your divs one by one to see it work!

Now you know how we worked our disappearing div magic in that very first exercise. You've been initiated into the mysteries of jQuery!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Check 'this' Out!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

<div></div>

<div></div>

<div></div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

background-color: #4ECDC4;

border-radius: 5px;

display: inline-block;

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).fadeOut('slow');

});

});

Ready?

All right! Time to use our new jQuery knowledge to add another interactive component to our website: a sliding panel that moves up and down when clicked.

We've set up the necessary HTML and CSS for you. Your assignment: animate it with jQuery.

You know the drill! Turn document into a jQuery object and call the .ready() action. (No need to put anything in .ready()'s parentheses just yet.)

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Slide Panel</title>

<script type="text/javascript" src="script.js"></script>

<link rel="stylesheet" type="text/css" href="stylesheet.css"></link>

</head>

<body>

<div class="panel">

<br />

<br />

<p>Now you see me!</p>

</div>

<p class="slide"><a href="#" class="pull-me">Slide Up/Down</a></p>

</body>

</html>

**CSS**

body {

margin:0 auto;

padding:0;

width:200px;

text-align:center;

}

a:hover{

-webkit-box-shadow: 0 0 8px #FFD700;

-moz-box-shadow: 0 0 8px #FFD700;

box-shadow: 0 0 8px #FFD700;

cursor:pointer;

}

.panel {

background: #ffffbd;

background-size:90% 90%;

height:300px;

display:none;

font-family:garamond,times-new-roman,serif;

}

.panel p{

text-align:center;

}

.slide {

margin:0;

padding:0;

border-top:solid 2px #cc0000;

}

.pull-me {

display:block;

position:relative;

right:-25px;

width:150px;

height:20px;

font-family:arial,sans-serif;

font-size:14px;

color:#ffffff;

background:#cc0000;

text-decoration:none;

-moz-border-bottom-left-radius:5px;

-moz-border-bottom-right-radius:5px;

border-bottom-left-radius:5px;

border-bottom-right-radius:5px;

}

.pull-me p {

text-align:center;

}

**jQuery Script**

$(document).ready();

Click and Pull

Good! Now we want to trigger an event when the "Slide Up/Down" tab is clicked (that tab's class is .pull-me).

1. Inside .ready()'s parentheses, add your function(){}.
2. Your function won't take any inputs between ()s, but between its {}s, it should call the .click() event handler on the .pull-me jQuery object.

Your code should look something like this:

$(document).ready(function() {

$('item').eventHandler();

});

Where 'item' is the element, class, or ID you want to select, and .eventHandler(); is the event handler you want to use (in this case, .click()).

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Slide Panel</title>

<script type="text/javascript" src="script.js"></script>

<link rel="stylesheet" type="text/css" href="stylesheet.css"></link>

</head>

<body>

<div class="panel">

<br />

<br />

<p>Now you see me!</p>

</div>

<p class="slide"><a href="#" class="pull-me">Slide Up/Down</a></p>

</body>

</html>

**CSS**

body {

margin:0 auto;

padding:0;

width:200px;

text-align:center;

}

a:hover{

-webkit-box-shadow: 0 0 8px #FFD700;

-moz-box-shadow: 0 0 8px #FFD700;

box-shadow: 0 0 8px #FFD700;

cursor:pointer;

}

.panel {

background: #ffffbd;

background-size:90% 90%;

height:300px;

display:none;

font-family:garamond,times-new-roman,serif;

}

.panel p{

text-align:center;

}

.slide {

margin:0;

padding:0;

border-top:solid 2px #cc0000;

}

.pull-me {

display:block;

position:relative;

right:-25px;

width:150px;

height:20px;

font-family:arial,sans-serif;

font-size:14px;

color:#ffffff;

background:#cc0000;

text-decoration:none;

-moz-border-bottom-left-radius:5px;

-moz-border-bottom-right-radius:5px;

border-bottom-left-radius:5px;

border-bottom-right-radius:5px;

}

.pull-me p {

text-align:center;

}

**jQuery Script**

$(document).ready(function() {

$('.pull-me').click();

});

Toggling Our Panel

Perfect! Just one more step: we need to tell .click() what to do. In this case, when our pull tab is clicked, we want our sliding panel (with the class .panel) to open or close.

The jQuery event we need to toggle our sliding panel is (you guessed it): .slideToggle()! We'll pass it one input, which is the speed of our slide animation.

Inside .click()'s parentheses, create a .panel jQuery object and call the .slideToggle() effect. Give it the input 'slow' for the animation speed.

Your code should look something like this:

$(document).ready(function() {

$('.pull-me').click(function() {

$('elem').effect('speed');

});

});

Where 'elem' is the element, class, or ID you want to select, effect is .slideToggle();, and 'speed' is 'slow'.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Slide Panel</title>

<script type="text/javascript" src="script.js"></script>

<link rel="stylesheet" type="text/css" href="stylesheet.css"></link>

</head>

<body>

<div class="panel">

<br />

<br />

<p>Now you see me!</p>

</div>

<p class="slide"><a href="#" class="pull-me">Slide Up/Down</a></p>

</body>

</html>

**CSS**

body {

margin:0 auto;

padding:0;

width:200px;

text-align:center;

}

a:hover{

-webkit-box-shadow: 0 0 8px #FFD700;

-moz-box-shadow: 0 0 8px #FFD700;

box-shadow: 0 0 8px #FFD700;

cursor:pointer;

}

.panel {

background: #ffffbd;

background-size:90% 90%;

height:300px;

display:none;

font-family:garamond,times-new-roman,serif;

}

.panel p{

text-align:center;

}

.slide {

margin:0;

padding:0;

border-top:solid 2px #cc0000;

}

.pull-me {

display:block;

position:relative;

right:-25px;

width:150px;

height:20px;

font-family:arial,sans-serif;

font-size:14px;

color:#ffffff;

background:#cc0000;

text-decoration:none;

-moz-border-bottom-left-radius:5px;

-moz-border-bottom-right-radius:5px;

border-bottom-left-radius:5px;

border-bottom-right-radius:5px;

}

.pull-me p {

text-align:center;

}

**jQuery Script**

$(document).ready(function() {

$('.pull-me').click(function() {

$('.panel').slideToggle('slow');

});

});

Well Done!

Great work! See how easy this is? With a little jQuery magic, you can make your websites do all kinds of amazing things.

Websites like Twitter, Facebook, and Gmail all make use of jQuery to help make their websites interactive. By learning jQuery, you're well on your way to Internet greatness!

Any time you have questions about what jQuery can do or how it works, you can always check out the [jQuery documentation](http://docs.jquery.com/), which covers every aspect of the library.

Creating HTML Elements

Dynamically adding elements to our HTML page is a powerful tool—it lets us modify not only the formatting, but the actual *structure* of our websites in response to a user's actions. For example, when you get a Gchat, each message is actually a new <div> being dynamically added to the page. Cool, right?

If you think about it, we've sort of done this already: all we're doing is setting a variable equal to a jQuery object. In this case, however, instead of just having something like:

$p = $('p');

We'll want to pass in an entire HTML element in quotes:

$p = $("<p>I'm a new paragraph!</p>");

When we put text in quotes like this, we call it a string (as in a "string of characters"). From now on, when we say "string," you can think "text" or "phrase." Strings are always in single or double quotes.

In script.js, create a variable, $h1, and set it equal to a jQuery object containing an <h1> tag with the text "Hello". The tag should be a string.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body></body>

</html>

**jQuery Script**

$h1 = $("<h1>Hello</h1>");

Inserting Elements

We can insert our newly created elements using a few jQuery actions.

.append() inserts the specified element as the last child of the target element. .prepend() inserts the specified element as the *first* child of the target element. If we have a div of class .info,

$(".info").append("<p>Stuff!</p>");

$(".info").prepend("<p>Stuff!</p>");

will add a paragraph containing the text "Stuff!" inside all divs of class .info. .append() will make the paragraph the last child of each div; .prepend() will make the paragraph the first child of each div. (Note: see the Hint if you're using single quotes.)

.appendTo() does the same as .append(), but it just reverses the order of "what to add" and "where to add it." The code

$('<p>Stuff!</p>').appendTo('.info');

has the same effect as the .append() code above. .prependTo() has a similar relationship to .prepend().

In the script.js tab, go ahead and .append() a paragraph to the body of your HTML document. Your paragraph should contain the text: "I'm a paragraph!"

You'll want to call .append() on the HTML document's body, like so:

$(document).ready(function () {

$("body").append(element);

});

where element should be replaced by the paragraph containing the text "I'm a paragraph!"

If you use single quotes for your string, you'll need to **escape** the apostrophe in "I'm a paragraph!" by putting a \ before it (like so: 'I\'m a paragraph!'). This lets jQuery know not to end the string when it hits the apostrophe.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body></body>

</html>

**jQuery Script**

$(document).ready(function() {

$('body').append("<p>I'm a paragraph!</p>");

});

Before and After

We can specify where in the DOM we insert an element with the .before() and .after() functions. The syntax looks like this:

$('target').after('<tag>To add</tag>');

Where 'target' is the element after which you want to add something and the bit between <tag>s is the HTML element you want to add. You can add <h1>s, <div>s, or any other valid HTML you like.

Go ahead and add a <p> tag .after() the <div> with the ID #one. You can place any text you like in your <p> tag.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div class="container">

<h2>Greetings</h2>

<div id="one">Div #1</div>

<div id="two">Div #2</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$('#one').after('<p>New paragraph</p>');

});

Moving Elements Around

Moving elements around in the DOM is a snap—all we need to do is use the jQuery functions we just learned on existing elements instead of creating new ones.

var $paragraph = $("p"); // existing element

$("div").after($paragraph); // Move it!

// Same as:

$("div").after($("p"));

1. We can select an element using $("p") and assign it to a variable
2. We can move the position in the DOM by using the variable in our after() statement

Note: This does not copy the element from one location to another, it moves the original element effectively saving you from having to delete the original

1. In script.js, add a new line below your code to move your <p> tag after the <div> with id #one
2. create a new jQuery selector that targets the <div> with the id #two
3. Using after again, select the <p> tag you created and move it after the <div> with id #two

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div class="container">

<h2>Greetings</h2>

<div id="one">Div #1</div>

<div id="two">Div #2</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$('#two').after('<p>New paragraph</p>');

});

Removing Elements

Adding elements to our HTML documents is great, but without the ability to remove them, our pages can quickly become cluttered. Thankfully, we have two jQuery functions, .empty() and .remove(), that help us delete content from our pages.

.empty() deletes an element's content and *all its descendants*. For instance, if you .empty() an 'ol', you'll also remove all its 'li's and their text.

.remove(), not only deletes an element's content, but deletes the element itself.

Go ahead and .remove() your <p> tag from the HTML document.

Your syntax should look like this:

$('element to remove').remove();

(You should be removing a 'p', not a '<p>'.)

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div class="container">

<h2>Greetings</h2>

<div id="one">Div #1</div>

<div id="two">Div #2</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$('#two').after('<p>New paragraph</p>');

$('p').remove();

});

Adding and Removing Classes

We don't have to limit ourselves to adding or removing entire elements, though—we can fine-tune our jQuery superpowers to alter classes, CSS, and even the contents of our HTML elements.

Let's start with classes. jQuery includes two functions, .addClass() and .removeClass(), that can be used to add or remove a class from an element. This is great if, for example, you have a highlighted class that you want to apply to an element when clicked.

The syntax looks like this:

$('selector').addClass('className');

$('selector').removeClass('className');

where 'selector' is the HTML element you want and 'className' is the class name you want to add or remove.

Remember: You aren't selecting anything, you are modifying your element. This means that you do not need # or . before your class.

Now that we think of it, that highlighted class seems like a great idea. We've supplied the HTML and CSS; in the script.js tab, add the jQuery code necessary to make your #text div highlighted when clicked!

MAKE SURE to click "Submit," and THEN click "Highlight me, too!" to see the results.

Your code should look something like this:

$(document).ready(function() {

$('SEL').click(function() {

$(this).FUNC('CLASS');

});

});

where SEL is the HTML element you should target (in this case, the #text), FUNC is the jQuery function you should use to add a class, and CLASS is the class name you should add (in this case, 'highlighted').

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Highlights</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div id="title" class="highlighted">I'm highlighted!</div>

<div id="text">Highlight me, too!</div>

</body>

</html>

**CSS**

#title {

background-color: #C02942;

border-radius: 5px;

text-align: center;

font-family: Verdana, Arial, Sans-Serif;

color: #FFFFFF;

width: 200px;

height: 25px;

}

#text {

background-color: #0B486B;

border-radius: 5px;

text-align: center;

font-family: Vivaldi, Cursive;

color: #FFFFFF;

width: 200px;

height: 25px;

}

.highlighted {

-webkit-box-shadow: 0 0 8px #FFD700;

-moz-box-shadow: 0 0 8px #FFD700;

box-shadow: 0 0 8px #FFD700;

cursor:pointer;

}

**jQuery Script**

$(document).ready(function() {

$('#text').click(function() {

$(this).addClass('highlighted');

});

});

Toggling Classes

What if we want to toggle a class back and forth, though? That is, what if we want jQuery to automatically check to see whether our #text is .highlighted, so that when we click on it, it adds the class if it isn't there and removes it if it is?

As you probably guessed, jQuery includes a .toggleClass() function that does exactly this. If the element it's called on has the class it receives as an input, .toggleClass() removes that class; if the target element doesn't have that class, .toggleClass() adds it.

Change your .addClass() to a .toggleClass(). Click on your #text a bunch of times. See how it turns the highlighting on and off?

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Highlights</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div id="title" class="highlighted">I'm highlighted!</div>

<div id="text">Highlight me, too!</div>

</body>

</html>

**CSS**

#title {

background-color: #C02942;

border-radius: 5px;

text-align: center;

font-family: Verdana, Arial, Sans-Serif;

color: #FFFFFF;

width: 200px;

height: 25px;

}

#text {

background-color: #0B486B;

border-radius: 5px;

text-align: center;

font-family: Vivaldi, Cursive;

color: #FFFFFF;

width: 200px;

height: 25px;

}

.highlighted {

-webkit-box-shadow: 0 0 8px #FFD700;

-moz-box-shadow: 0 0 8px #FFD700;

box-shadow: 0 0 8px #FFD700;

cursor:pointer;

}

**jQuery Script**

$(document).ready(function() {

$('#text').click(function() {

$(this).toggleClass('highlighted');

});

});

Changing Your Style

What if we want to fine-tune individual CSS property values, though? Remember style="height:300px; width:300px;"? jQuery makes it a snap!

Because resizing elements is so common, jQuery has specific .height() and .width() functions that can be used to change the heights and widths of HTML elements. For instance:

$("div").height("100px");

$("div").width("50px");

would give all <div>s on the page a height of 100 pixels and a width of 50 pixels.

jQuery also includes a general-purpose .css() function that takes two inputs: the first is the CSS element to alter, and the second is the value to set it to. For example:

$("div").css("background-color","#008800");

would give all <div>s on the page a green background color. You can modify any element's CSS attributes this way.

Let's get some practice in. Change the <div> in our HTML document so it has a height of 200 pixels, a width of 200 pixels, and a border-radius of 10 pixels. Let's make this happen immediately, without the need for a click.

Remember, if you want an event to occur as soon as the document loads, you can just leave out the event handler:

$(document).ready(function() {

$("element").effect1();

$("element").effect2();

$("element").effect3();

});

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 50px;

width: 300px;

border-radius:0;

background-color: #2C7CC3;

}

**jQuery Script**

$(document).ready(function() {

$('div').height('200px');

$('div').width('200px');

$('div').css('border-radius','10px');

});

Modifying Content

Finally, we can update the contents of our HTML elements—that is, the bit between the opening and closing tags—using the .html() and .val() functions.

.html() can be used to set the contents of the first element match it finds. For instance,

$('div').html();

will get the HTML contents of the *first* div it finds, and

$('div').html("I love jQuery!");

will set the contents of the first div it finds to "I love jQuery!"

.val() is used to get the value of form elements. For example,

$('input:checkbox:checked').val();

would get the value of the first checked checkbox that jQuery finds.

We've added a paragraph to our HTML document. Use the .html() function to set its text to "jQuery magic in action!" (Don't forget the exclamation point!)

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<p>I'm about to change!</p>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$('p').html("jQuery magic in action!");

});

Set Up

All right! It's time to apply our newfound knowledge. In this section, we'll build an interactive "to do" list that will add items to a checklist and remove them as they're checked off.

We've provided the necessary HTML and CSS; you'll provide the jQuery.

You know the drill. Get your document .ready()!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>To Do</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<h2>To Do</h2>

<form name="checkListForm">

<input type="text" name="checkListItem"/>

</form>

<div id="button">Add!</div>

<br/>

<div class="list"></div>

</body>

</html>

**CSS**

h2 {

font-family:arial;

}

form {

display: inline-block;

}

#button{

display: inline-block;

height:20px;

width:70px;

background-color:#cc0000;

font-family:arial;

font-weight:bold;

color:#ffffff;

border-radius: 5px;

text-align:center;

margin-top:2px;

}

.list {

font-family:garamond;

color:#cc0000;

}

**jQuery Script**

$(document).ready();

Click Da Button! Do It Naoughw!

You'll notice we've set up an HTML form for grabbing the user's input. We'll need to store the user's input in a variable, which will allow us to append that input to the body of the HTML document later on.

You can set a variable equal to the contents of the input field using .val(), like so:

//Get the value from an input

var input = $('input[name=checkListItem]').val();

1. Our selector finds our specific input using a css selector on our checkListItem input
2. We call val() to get the value of the field
3. Let's create a .click() event handler for our #button element
4. Make sure you pass it a function(){}
5. Inside your function(){} create the variable toAdd and set it equal to the contents of the input field.

Your code should look something like this:

$('#button').click(function() {

var toAdd = $('input[name=checkListItem]').val();

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>To Do</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<h2>To Do</h2>

<form name="checkListForm">

<input type="text" name="checkListItem"/>

</form>

<div id="button">Add!</div>

<br/>

<div class="list"></div>

</body>

</html>

**CSS**

h2 {

font-family:arial;

}

form {

display: inline-block;

}

#button{

display: inline-block;

height:20px;

width:70px;

background-color:#cc0000;

font-family:arial;

font-weight:bold;

color:#ffffff;

border-radius: 5px;

text-align:center;

margin-top:2px;

}

.list {

font-family:garamond;

color:#cc0000;

}

**jQuery Script**

$(document).ready(function() {

$('#button').click(function() {

var toAdd = $('input[name=checkListItem]').val();

});

});

Append to Body

Perfect! Now we want to add our HTML element to the document. We can do this using our handy .append() function.

Let's go ahead and append to our div with the .list class. We'll append a <div> with class="item", since we'll want a way to target our appended <div>s later when we remove them. (A "to do" list is no good if we can't check things off it.)

We'll want the contents of our div to be the contents of our input field, which we saved in the variable toAdd. That means when we append, we'll want to append

'<div class="item">' + toAdd + '</div>'

Go ahead and .append() a <div> with class="item" to the .list div of your HTML document, then MAKE SURE to click your button to add an item—the exercise will wait for you to do so!

You'll need to alternate quotation marks here, since the HTML string you're appending needs to be in quotes, but so does your class name. The example in the instructional text above is one way to do this.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>To Do</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<h2>To Do</h2>

<form name="checkListForm">

<input type="text" name="checkListItem"/>

</form>

<div id="button">Add!</div>

<br/>

<div class="list"></div>

</body>

</html>

**CSS**

h2 {

font-family:arial;

}

form {

display: inline-block;

}

#button{

display: inline-block;

height:20px;

width:70px;

background-color:#cc0000;

font-family:arial;

font-weight:bold;

color:#ffffff;

border-radius: 5px;

text-align:center;

margin-top:2px;

}

.list {

font-family:garamond;

color:#cc0000;

}

**jQuery Script**

$(document).ready(function() {

$('#button').click(function() {

var toAdd = $('input[name=checkListItem]').val();

$('.list').append('<div class="item">' + toAdd + '</div>');

});

});

Remove What's Been Clicked

Great job! Finally, we want to be able to check items off our list.

You might think we could do this:

$('.item').click(function() {

$(this).remove();

});

and that's not a bad idea. The problem is that it won't work—jQuery looks for all the .items when the DOM is loaded, so by the time your document is ready, it's already decided there are no .items to .remove(), and your code won't work.

For this, we'll need a new event handler: .on(). You can think of .on() as a general handler that takes the event, its selector, and an action as inputs. The syntax looks like this:

$(document).on('event', 'selector', function() {

Do something!

});

In this case, 'event' will be 'click', 'selector' will be '.item', and the thing we'll want to do is call .remove() on this.

Complete your program by adding the .on() event handler as described above. It should go inside your $(document).ready(), but after and outside your $('#button').click().

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>To Do</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css"/>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<h2>To Do</h2>

<form name="checkListForm">

<input type="text" name="checkListItem"/>

</form>

<div id="button">Add!</div>

<br/>

<div class="list"></div>

</body>

</html>

**CSS**

h2 {

font-family:arial;

}

form {

display: inline-block;

}

#button{

display: inline-block;

height:20px;

width:70px;

background-color:#cc0000;

font-family:arial;

font-weight:bold;

color:#ffffff;

border-radius: 5px;

text-align:center;

margin-top:2px;

}

.list {

font-family:garamond;

color:#cc0000;

}

**jQuery Script**

$(document).ready(function() {

$('#button').click(function() {

var toAdd = $('input[name=checkListItem]').val();

$('.list').append('<div class="item">' + toAdd + '</div>');

});

$(document).on('click','.item',function() {

$(this).remove();

});

});

You Did It!

Great work! You now know how to dynamically update the content of your HTML page, including adding and removing elements.

Now that you can handle manipulating the DOM on the fly, the hard part is over. In the next two lessons, we'll cover a wider range of jQuery event handlers and effects, which will allow you to apply your core programming skills to a variety of challenges.

Review of jQuery Events

You know a lot about jQuery events already, but it never hurts to review the basics.

The setup almost always looks like this:

$(document).ready(function() {

$('thingToTouch').event(function() {

$('thingToAffect').effect();

});

});

where "thing to touch" is the HTML element you'll click on, hover over, or otherwise interact with, and "thing to affect" is the HTML element that fades away, changes size, or undergoes some other transformation.

Sometimes these elements are one and the same—you might hover over a <div> to change its opacity. Other times, you might interact with a separate element; for example, you might click on a button to resize a <div>.

Sometimes if you want an effect to occur right away, without an event like .click() or .hover(), you'll skip the second line in the above:

$(document).ready(function() {

$('thingToAffect').effect();

});

Take a look at the code in script.js. Make sure you understand what it'll do! Click Save & Submit Code when you're ready to test your hypothesis.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Result</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<form>

MESSAGE: <input type="text" name="message" value="Type your text here!">

</form>

<button>Add!</button><br/>

<div id="messages"></div>

</body>

</html>

**CSS**

form {

font-size: 12px;

font-family: Verdana, Arial, Sans-Serif;

display: inline-block;

}

#messages {

font-size: 14px;

font-family: Garamond, Times, Serif;

}

**jQuery Script**

$(document).ready(function() {

$('button').click(function() {

var toAdd = $("input[name=message]").val();

$('#messages').append("<p>"+toAdd+"</p>");

});

});

Cutting to the Chase

Let's quickly review how to trigger an effect without a special event handler like .click()—we just want our effect to happen as soon as our document is .ready().

Remember: practice makes perfect!

Your div represents the planet [Krypton](http://bit.ly/jSAGQL). Make it .fadeOut() 'fast'! (Go ahead and do this immediately, without the need for a click.)

We would have made our Krypton div blow up, if only there were a jQuery .blowUp() effect.

Your syntax should look like this:

$(document).ready(function() {

$('element').fadeOut('speed');

});

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Kapow!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

.red {

background-color: #CC0000;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#330000), to(#CC0000));

background-image: -webkit-linear-gradient(left, #330000, #CC0000);

background-image: -moz-linear-gradient(left, #330000, #CC0000);

background-image: -ms-linear-gradient(left, #330000, #CC0000);

background-image: -o-linear-gradient(left, #330000, #CC0000);

}

**jQuery Script**

$(document).ready(function() {

$('div').fadeOut('fast');

});

Adding an Event Handler

Great! Now let's review how to make something happen *with* an event handler. In this case, we'll use .click().

Rewrite your jQuery code so that Krypton only fades out when you .click() on it. It should still fade out 'fast'! MAKE SURE to click Save & Submit Code, then click on Krypton.

Your code should now look like this:

$(document).ready(function() {

$('element').event(function() {

$(this).fadeOut('fast');

});

});

Where 'element' is the element you want to click on (in this case, your 'div') and event is .click.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Kapow!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

.red {

background-color: #CC0000;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#330000), to(#CC0000));

background-image: -webkit-linear-gradient(left, #330000, #CC0000);

background-image: -moz-linear-gradient(left, #330000, #CC0000);

background-image: -ms-linear-gradient(left, #330000, #CC0000);

background-image: -o-linear-gradient(left, #330000, #CC0000);

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).fadeOut('fast');

});

});

Combining .click() and .hover()

Well done! Let's add one more jQuery event to our "destruction of Krypton" simulation. Krypton didn't just vanish, it exploded! Let's make it turn red.

$('div').hover(function(){

$('div').addClass('green');

});

1. Following the pattern we have been learning, we target Krypton, our $('div')
2. We then apply our hover event to our target.
3. Finally, we execute the code inside the function(){} which adds a class of green to our target.
4. Below your .click() event code from the previous lesson, create a new selector $('div')
5. Apply a hover event to this selector and have it add the class red to Krypton, our div

Check out the example above!

Remember, we can use .addClass() to add our class, like so:

$(this).addClass('red');

// OR

$('div').addClass('red');

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Kapow!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

.red {

background-color: #CC0000;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#330000), to(#CC0000));

background-image: -webkit-linear-gradient(left, #330000, #CC0000);

background-image: -moz-linear-gradient(left, #330000, #CC0000);

background-image: -ms-linear-gradient(left, #330000, #CC0000);

background-image: -o-linear-gradient(left, #330000, #CC0000);

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).fadeOut('fast');

});

$('div').hover(function() {

$(this).addClass('red');

});

});

The .dblclick() Event

Now that we've reviewed our jQuery event handlers, let's learn a new one.

We might want to cause a jQuery effect when a user *double* clicks on an element, rather than just single-clicking. We can do this with the .dblclick() event handler.

Use the .dblclick() handler to make our 'div' .fadeOut()s 'fast'. Make sure to double-click on your div to see the effect!

Your code should look something like this:

$(document).ready(function() {

$('elem').event(function() {

$(this).effect('speed');

});

});

Where 'elem' is the HTML element you want to click, event is your .dblclick() event, effect is your fadeOut, and your 'speed' is 'fast'!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Dubba Click</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

border-radius: 5px;

background-color: #ABCDEF;

}

**jQuery Script**

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

$('div').dblclick(function() {

$(this).fadeOut('fast');

});

});

Hover

What if you wanted to create an effect when your mouse is on top of an object, then have that effect vanish when your mouse moved away? You might notice this effect in use on many site's navigation bars!

$('div').hover(

function(){

$(this).addClass('highlight');

},

function(){

$(this).removeClass('highlight');

}

);

1. We first select the element we want to modify $('div')
2. Secondly notice that our hover effect is able to take two functions(){} separated by a comma. The comma is very important!
3. The first function(){} we pass will be run when we first mouse over our target. Here we apply a class of highlight
4. The second function(){} will be called when our mouse leaves the object. This is where we remove the class highlight

Your second function(){} doesn't have to be the opposite of the first function(){}, but it would be very common!

We're ready to apply a hover effect to our site's navigation bar

1. In script.js use your selector to target our menu items
2. When we hover over the menu item, let's apply a class of active
3. When our mouse leaves the menu item, let's remove the active class

Your code should look very close to the example, except we're going to do

$(this).addClass('active');

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Ghostly Divs</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div>Home</div>

<div>About Us</div>

<div>Contact</div>

</body>

</html>

**CSS**

div {

border-radius: 5px;

background-color: #ABCDEF;

transition: background-color 0.5s ease;

display:inline;

font-size:25px;

padding:20px;

border:1px solid #ccc;

margin-top:10px;

}

.active {

background-color:#556677;

}

**jQuery Script**

$(document).ready(function(){

$('div').hover(

function(){

$(this).addClass('active');

},

function(){

$(this).removeClass('active');

}

);

});

Let's .focus()!

Another event we can make use of is .focus(). We say an element has **focus** when we click on it or tab over to it. If you've ever filled out a form on a web page and seen how each text box lights up when you tab to it or click on it, you've seen focus in action!

The .focus() event handler only works on elements that can receive focus—the list of these elements is a bit vague, but HTML elements like <textarea>s and <input>s are the usual suspects.

Check out the form we've set up in the Result tab. If you click on the input field, you'll see it automatically highlights in a delightful baby blue. Too bad baby blue is for babies! We want our highlighting to be red.

We can do this with two tools: .focus() and our .css() function from the last section. We want to write a bit of jQuery code that will change our 'input''s 'outline-color' to 'red' when it gains focus.

In script.js, get your document .ready(). Do two things:

1. Turn 'input' into a jQuery element and call the .focus() event handler on it.
2. Inside your event handler function, call .css() on your 'input' element and pass it two inputs: the CSS element to change ('outline-color') and the color to change it to ('#FF0000').
3. Not all browsers have equal CSS implementation. So for this lesson, if you are using Firefox, you will need to first change the outline-style to solid.

Your syntax should look like this:

$(document).ready(function() {

$('elem').event(function() {

$(this).func('css','color');

});

});

Where 'elem' is the 'input' element you want to target, event is the .focus event, func is the .css function, 'css' is the 'outline-color' CSS selector, and 'color' is the hex color '#FF0000' (red).

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Time to Focus!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<form>

Name: <input type='text' name='name'></input>

</form>

</body>

</html>

**CSS**

input {

font-family: Verdana, Arial, Sans-Serif;

}

**jQuery Script**

$(document).ready(function() {

$('input').focus(function() {

$(this).css('outline-color','#FF0000');

});

});

The .keydown() Event

You're not limited to mouse events in jQuery—you can trigger events using the keyboard, as well!

The .keydown() event is triggered whenever a key on the keyboard is pressed. It only works on whatever page element has focus, so you'll need to click on the window containing your div before pressing a key in order for you to see its effects.

Let's go ahead and combine our new event with a new effect: .animate()! We'll use this to move an object on the screen whenever we press a key.

The .animate() effect takes two inputs: the animation to perform, and the time in which to perform the animation. Here's an example:

$(document).ready(function() {

$('div').animate({left:'+=10px'},500);

});

This will take the first div it finds and move it ten pixels to the right. Remember, increasing the distance from the left margin moves something to the right; the += bit is just a shorthand for "take the existing number and add ten to it." In this case, it add ten pixels to the current distance from the left margin.

After setting up with $(document).ready(), call the .keydown() event on $(document). (We want the whole document object to respond whenever a key is pressed.)

Inside your .keydown() handler, .animate() your 'div' with the same inputs as above: {left:'+=10px'} and 500.

Your code should look something like this:

$(document).ready(function() {

$(document).event(function() {

$('div').effect(anim, duration);

});

});

Where event is your .keydown(), effect is .animate(), and the two inputs are the animation to be performed and the time it should take.

Now your div should move to the right when [any key](http://en.wikipedia.org/wiki/Any_key) is pressed!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Div Racer</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 50px;

width: 50px;

border-radius: 10px;

background-color: #FF0000;

position: relative;

}

**jQuery Script**

$(document).ready(function() {

$(document).keydown(function() {

$('div').animate({left:'+=10px'},500);

});

});

Getting Ready

Now that you've got a solid foundation in jQuery events and have covered important functions like .keydown() and .animate(), we'll put our newfound knowledge to work. In this section, we'll animate a **sprite**, or two-dimensional image on the screen.

For this momentous event, we've chosen a familiar face. Check him out in the Result tab!

Let's get started. First, get your document .ready().

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Super Mario!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<img src="http://community.digitalmediaacademy.org/wp-content/uploads/2010/10/Mario-8-bit.gif"/>

</body>

</html>

**CSS**

img {

position: relative;

}

**jQuery Script**

$(document).ready();

Using .keydown()

Good! Now we want to add a line of jQuery that will handle the keydown event.

Inside your $(document).ready() setup, call the .keydown() event on $(document). Pass .keydown() a function(){} as an input. Here's the twist: your function should take a single input, key, which will help jQuery figure out which key was pressed.

Your code should look something like this:

$(document).ready(function() {

$(document).event(function(key){});

Where event is .keydown().

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Super Mario!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<img src="http://community.digitalmediaacademy.org/wp-content/uploads/2010/10/Mario-8-bit.gif"/>

</body>

</html>

**CSS**

img {

position: relative;

}

**jQuery Script**

$(document).ready(function() {

$(document).keydown(function(key) {

});

});

Filling Out the Cases

Great work! Now it's time to animate our character based on the user's input from the keyboard.

Every key press on a keyboard is translated into a number for the computer to use. Don't worry about memorizing them, for now we've given you the basics in script.js

// Left arrow key pressed

case 37:

('img').animate({left: "-=10px"}, 'fast');

1. The left arrow key on our keyboards translates to number 37 to the computer. When that key is pressed, we animate our image to the left by subtracting 10px
2. To move up we subtract 10px from the top
3. To move right we add 10px to the left
4. Finally, to move down we add 10px to the top

Can you fill in Up, Down, and Right? What happens if you add pixels +=10px instead of subtracting?

Let's make our image move!

1. Complete the code for Up, Right, and Down
2. Have fun!

Your three lines of code should look very similar to what's on [line 5](javascript:void(0)).

Remember,

left: "+=10px"

will move your character 10 pixels to the right, and is the same as

right: "-=10px"

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Super Mario!</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<img src="http://community.digitalmediaacademy.org/wp-content/uploads/2010/10/Mario-8-bit.gif"/>

</body>

</html>

**CSS**

img {

position: relative;

left: 0;

top: 0;

}

**jQuery Script**

$(document).ready(function() {

$(document).keydown(function(key) {

switch(parseInt(key.which,10)) {

case 65:

$('img').animate({left: "-=10px"}, 'fast');

break;

case 83:

$('img').animate({top: "+=10px"}, 'fast');

break;

case 87:

$('img').animate({top: "-=10px"}, 'fast');

break;

case 68:

$('img').animate({right: "+=10px"}, 'fast');

break;

default:

break;

}

});

});

You Did It!

Great work! Your sprite now moves to the left, right, up, and down when you hit the **a**, **s**, **w**, and **d** keys. What other effects would you want to add in order to make a basic game? How might you add a background, give your character the ability to jump, or include text? Feel free to check out the [jQuery documentation](http://docs.jquery.com/) to see what other jQuery magic is available to you.

You Know This!

You already know a lot about jQuery effects, so let's try a training wheels-free exercise. You can do this! We believe in you.

Write the necessary jQuery code to .hide() your 'div' as soon as the HTML document is ready.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Vanishing Act</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #ABCDEF;

}

**jQuery Script**

$(document).ready(function() {

$('div').hide();

});

More Practice with .animate()

Let's get in a little more practice with the .animate() effect. That's the one with the slightly more complicated input—if we wanted to move a div 10 pixels down, we'd type something like

$('div').animate({top:'+=10px'},500);

Where the bit between curly braces says "hey, jQuery! Add 10 pixels to the current top margin," and the second input says "do it in 500 milliseconds!" (1,000 milliseconds = one second.)

Going down?   
1. Let's begin begin by targeting our image inside $()  
2. Then let's animate our image to go down by 100px  
3. Let's make it happen in 1 second or 1000 milliseconds

Your code should look something like this:

$(document).ready(function() {

$('img').effect(anim, speed);

});

Where 'effect' is the .animate() effect, anim is the input that tells jQuery how far to move the image and in what direction, and speed is 1000 (1,000 milliseconds = 1 second).

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Going Down</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

</head>

<body>

<img src="http://bit.ly/W6WbsH"/>

</body>

</html>

**CSS**

img {

position: relative;

}

**jQuery Script**

$(document).ready(function() {

$('img').animate({top:'100px'},1000);

});

Introducing: jQuery UI

All right! Time to blaze new jQuery trails with a new jQuery library: **jQuery UI**.

jQuery UI includes a number of ultra-fancy animations you can use to make your websites do incredible things.

For instance, remember when we lamented that jQuery didn't include a .blowUp() effect for our planet Krypton? Well, that's still true. But jQuery UI has an .effect() effect, and we are totally going to give it the input 'explode'.

Note that we've included an extra <script> tag in our HTML documents; this is used to include jQuery UI in our webpages. We don't have to do this with regular jQuery, since Codecademy automatically includes it for us.

In the script.js tab, get your document .ready() and pass in a .click() event handler on your 'div'. Inside that, call .effect() on your div and give it the input 'explode'. Head over to the Result tab and click on Krypton to watch it explode!

Your code should look something like this:

$(document).ready(function() {

$('elem').event(function() {

$(this).effect('explode');

});

});

Where 'elem' is your 'div' and event is the .click event handler.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Krypton Redux</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).effect('explode');

});

});

.bounce()

Cool, right? But we can do much more than just blow things up.

Another possible effect is 'bounce'. We give this as an input to .effect() just like 'explode', but we add an extra input to tell it how many times to bounce. This code will make our target 'div' bounce twice in 200 milliseconds:

$('div').effect('bounce', {times:2}, 200);

Change your 'explode' to a bounce, and make it so Krypton bounces three times in 500 milliseconds when you click on it.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Krypton Redux</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).effect('bounce', {times:3}, 500);

});

});

.slide()

We can also make Krypton .slide() into view. Not surprisingly, we do this by calling the .effect() effect and passing in 'slide' as an input.

If you want to see everything jQuery UI can do, you can check out the documentation [here](http://jqueryui.com/)!

Go ahead and change your 'bounce' to a 'slide', and remove the other two inputs.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Krypton Redux</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div></div>

</body>

</html>

**CSS**

body {

background-image: url('http://bit.ly/UpQgJ6');

repeat: no-repeat;

}

div {

height: 100px;

width: 100px;

border-radius: 100%;

background-color: #008800;

background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#003500), to(#008800));

background-image: -webkit-linear-gradient(left, #003500, #008800);

background-image: -moz-linear-gradient(left, #003500, #008800);

background-image: -ms-linear-gradient(left, #003500, #008800);

background-image: -o-linear-gradient(left, #003500, #008800);

}

**jQuery Script**

$(document).ready(function() {

$('div').click(function() {

$(this).effect('slide');

});

});

Special Effects

The .effect() effect has all kinds of magical goodness in it, but it's not the most amazing thing jQuery UI can do. The library has a number of built-in effects that can make your website look sleek and professional with surprisingly little code.

You can learn more in the [jQuery UI documentation](http://jqueryui.com/)!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Behold!</title>

<link rel='stylesheet' type='text/css' href='http://code.jquery.com/ui/1.9.1/themes/base/jquery-ui.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div id="menu">

<h3>jQuery</h3>

<div>

<p>jQuery is a JavaScript library that makes your websites look absolutely stunning.</p>

</div>

<h3>jQuery UI</h3>

<div>

<p>jQuery UI includes even more jQuery goodness!</p>

</div>

<h3>JavaScript</h3>

<div>

<p>JavaScript is a programming language used in web browsers, and it's what powers jQuery and jQuery UI. You can learn about JavaScript in the <a href="http://www.codecademy.com/tracks/javascript" target="blank" style="text-decoration:none; color:#F39814">JavaScript track</a> here on Codecademy.</a></p>

</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$("#menu").accordion({collapsible: true, active: false});

});

Drag Racing

jQuery UI includes a .draggable() function that can make any HTML element draggable—you can click on it and move it anywhere on the page!

We thought you might be tired of <div> blocks, so we made you a CSS car. We worked super hard on it.

Call the .draggable() function on '#car'. Click Save & Submit Code, then go to the Result tab and drag your car around the page!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div id="car">

<div id="top"></div>

<div id="bottom"></div>

<div id="fwheel"></div>

<div id="bwheel"></div>

</div>

</body>

</html>

**CSS**

#top {

position: relative;

height: 50px;

width: 50px;

border-radius: 5px;

background-color: #cc0000;

left: 25px;

}

#bottom {

position: relative;

height:25px;

width: 100px;

background-color: #cc0000;

border-top-left-radius: 5px;

border-top-right-radius: 5px;

top: -25px;

}

#fwheel {

position: relative;

height:20px;

width:20px;

border-radius: 100%;

background-color: black;

top: -35px;

left: 5px;

}

#bwheel {

position: relative;

height:20px;

width:20px;

border-radius: 100%;

background-color: black;

top: -55px;

left: 75px;

}

**jQuery Script**

$(document).ready(function() {

$('#car').draggable();

});

One Resize Fits All

You didn't love our mega sweet car? We slaved over that for months! Back to regular <div>s for you!

(Just kidding. That little guy took us *decades* of hard work.)

Call the .resizable() function on your 'div'. Click Save & Submit Code, then go to the Result tab and resize your <div> to your heart's content!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div>Resize Me!</div>

</body>

</html>

**CSS**

div {

height: 100px;

background-color: #ABCDEF;

font-family: Verdana, Arial, Sans-Serif;

font-size: 1em;

text-align: center;

}

**jQuery Script**

$(document).ready(function() {

$('div').resizable();

});

A Greater Selection

Images and <div>s aren't the only elements we can target with jQuery UI—we can also enhance our ordered and unordered lists.

Check out the HTML in index.html. We've set up a pretty nifty ordered list, but when it shows up in the Result tab, it's a little bland. With jQuery UI, we can fix that!

In the script.js tab, get your document .ready() and call the .selectable() function on your 'ol'. Click Save & Submit Code, then click on the list items in your ordered list in the Result tab to see how much sleeker it looks.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Select Ye Favorite</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<ol>

<li>Super Mario Bros.</li>

<li>Tetris</li>

<li>Legend of Zelda: Link's Awakening</li>

<li>Kirby's Dream World</li>

<li>Burger Time</li>

<li>Pokémon Red</li>

<li>Pokémon Blue</li>

</ol>

</body>

</html>

**CSS**

ol {

list-style-type: none;

position: relative;

left: -20px;

}

ol li {

background: #eeeeee;

border-radius: 5px;

border: 1px solid black;

margin: 3px;

padding: 0.4em;

font-size: 1em;

height: 16px;

font-family: Verdana, Arial, Sans-Serif;

}

ol .ui-selected {

background: #F39814; color: white;

}

**jQuery Script**

$(document).ready(function() {

$('ol').selectable();

});

Let's Sort Things Out

Lists are great, and jQuery UI makes them greater. While selecting is cool, it doesn't necessarily make our list as dynamic as we'd like—what if we want to reorder the elements in our list?

Enter the .sortable() function.

In the script.js tab, replace your .selectable() with a .sortable(). Click Save & Submit Code, then rearrange your list items as you see fit!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title>Select Ye Favorite</title>

<link rel='stylesheet' type='text/css' href='stylesheet.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<ol>

<li>Super Mario Bros.</li>

<li>Tetris</li>

<li>Legend of Zelda: Link's Awakening</li>

<li>Kirby's Dream World</li>

<li>Burger Time</li>

<li>Pokémon Red</li>

<li>Pokémon Blue</li>

</ol>

</body>

</html>

**CSS**

ol {

list-style-type: none;

position: relative;

left: -20px;

}

ol li {

background: #eeeeee;

border-radius: 5px;

border: 1px solid black;

margin: 3px;

padding: 0.4em;

font-size: 1em;

height: 16px;

font-family: Verdana, Arial, Sans-Serif;

}

ol .ui-selected {

background: #F39814; color: white;

}

**jQuery Script**

$(document).ready(function() {

$('ol').sortable();

});

jQuery UI, is There Anything You Can't Do?

All right! Time to use jQuery UI to create that awesome-looking accordion-style dropdown menu we saw in the last section. For this, we'll be using jQuery UI's built-in .accordion() function. (Handy, no?)

First, however, we'll need some additional HTML elements.

We got you started in the index.html tab, but a menu's no good with only one section! Add two more like the first one. They should go inside the '#menu' div but after the "Section 1" div.

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel='stylesheet' type='text/css' href='http://code.jquery.com/ui/1.9.1/themes/base/jquery-ui.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div id="menu">

<h3>Section 1</h3>

<div>

<p>I'm the first section!</p>

</div>

<!--Add two more sections below!-->

<h3>Section 2</h3>

<div>

<p>I'm the second section!</p>

</div>

<h3>Section 3</h3>

<div>

<p>I'm the third section!</p>

</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

});

At the Ready

Perfect! Now we'll add the necessary jQuery magic to script.js.

You know this part by heart: get your document .ready()!

**jQuery Script**

$(document).ready(function() {

});

Break Out Your .accordion()!

Perfect! Now all we need to do is call .accordion() on our '#menu'.

Go ahead and call .accordion() on your '#menu'. Click Save & Submit Code to see your menu come to life!

**HTML**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel='stylesheet' type='text/css' href='http://code.jquery.com/ui/1.9.1/themes/base/jquery-ui.css'/>

<script type='text/javascript' src='script.js'></script>

<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

</head>

<body>

<div id="menu">

<h3>Section 1</h3>

<div>

<p>I'm the first section!</p>

</div>

<!--Add two more sections below!-->

<h3>Section 2</h3>

<div>

<p>I'm the second section!</p>

</div>

<h3>Section 3</h3>

<div>

<p>I'm the third section!</p>

</div>

</div>

</body>

</html>

**jQuery Script**

$(document).ready(function() {

$('#menu').accordion();

});

Victory!

You did it! Doesn't that look great?

With the jQuery projects you've mastered over the last several courses, you're now prepared to make a truly impressive interactive website. And when you make it, we'd love to see it!