Introduction

Welcome Back!

Today's lesson is about jQuery.

jQuery is a JavaScript library, which means it's a whole lot of JavaScript code that has already been written and tested. That's good for you because the folks who wrote jQuery have already done all the "heavy lifting" of creating JavaScript for special effects and other things that require advanced programming knowledge, complex math skills, or both. They've also taken care of the many compatibility issues that arise with older Web browsers.

While we won't delve too deeply into jQuery, it's important to cover some basics. jQuery *is* JavaScript after all. As many of you know by now, software languages like CSS, HTML, and JavaScript really aren't geared towards rote memorization that allows you to write fluently for the rest of your life. There is simply too much to memorize, and due to the inventiveness of programmers you'll find that there are many different ways to do things.

The bottom line is that you only need to learn enough to be able to understand the code you are looking at. From there you can educate yourself and look things up online, as needed. However, once you get to a point where you start researching things online for JavaScript, you'll find just as much information about jQuery as you will about JavaScript. That's because most developers treat them as one and the same.

Let's start by talking about where to get jQuery and how to start using it in your own work.

Download

Below is the link to the ZIP file, which contains the following resources you'll need for this lesson:

- **jQuery Template:** This is a simple HTML page you'll use to create a *jQuery Template*.
- **jQuery Buttons:** This page demonstrates how jQuery can be used to apply JavaScript to multiple elements of the same type.
- **jQuery Demo:** This page already has some of the basic jQuery you need to experiment with some jQuery special effects.

Download Lesson-11_ClassProject.zip

1.2 KB (Gigabytes) ZIP

You're welcome to type these pages from scratch, or simply s*ave them* in your *Intro JavaScript* folder.

Chapter 1: Getting Started with jQuery

What is jQuery?

jQuery is a JavaScript Library

You've already learned about creating and using external JavaScript files, which allow you to share code across multiple pages in your site. But you aren't limited to creating your own external JavaScript files. You can also use any prewritten JavaScript code, written by someone other than yourself, that's delivered in a file. Such code is commonly referred to as a *JavaScript library*. And there are quite a few JavaScript libraries on the Internet. The most successful and widely used is jQuery.

Simply stated, jQuery is a free external file of JavaScript code that you can use in your own site for free, no strings attached. jQuery's motto is *Write less, do more* because that's exactly what's it's all about. You can do "big" JavaScript things by writing small bits of code because you can call on JavaScript code that's in the jQuery library to do the "heavy lifting," as I mentioned before, accomplishing with one line of code something that might require dozens of lines of code if written from scratch.

jQuery Versions

Like most software, jQuery evolves over time, so there are always multiple versions available. All of the versions are numbered.

- Versions with higher numbers are newer than versions with smaller numbers. (meaning version 2.0 is older than version 3.0)
- Updates within versions get larger over time as well, so version 3.3 is older than 3.5.
- Patches to versions also get larger over time, so version 3.5.0 is older than 3.5.1.

When choosing which jQuery version to use, the answer is generally simple: from the page where you're obtaining the code, use whichever version has the largest number.

How to Use jQuery

There are two basic ways to use jQuery. One is to download it and add it to your website so that people who view your pages also get that file. That's an old and clumsy way to do things though. The more modern way is to use a *CDN (Content Delivery Network)* to provide the library for you, and your users. The downloads are much, much faster that way. That method also frees you from the drudgery of managing the file yourself within your site.

jQuery File Types

You'll also find different types of files you can download or link to:

- **Uncompressed**: Contains all the original source code including comments and indents for formatting. Use this if you want to examine or tinker with the code inside the jQuery library.
- **Minified**: Same as uncompressed with all comments and extra spaces removed so the file is smaller and downloads more quickly.
- **Slim**: Same as uncompressed but without AJAX, which is a language that allows JavaScript to interact with the web server. You might want this if you're using a separate AJAX library of your own and don't want to use the AJAX capabilities of jQuery.
- **Slim minified**: Same as slim but with comment and extra spaces removed to minimize the size of the file to speed downloading.

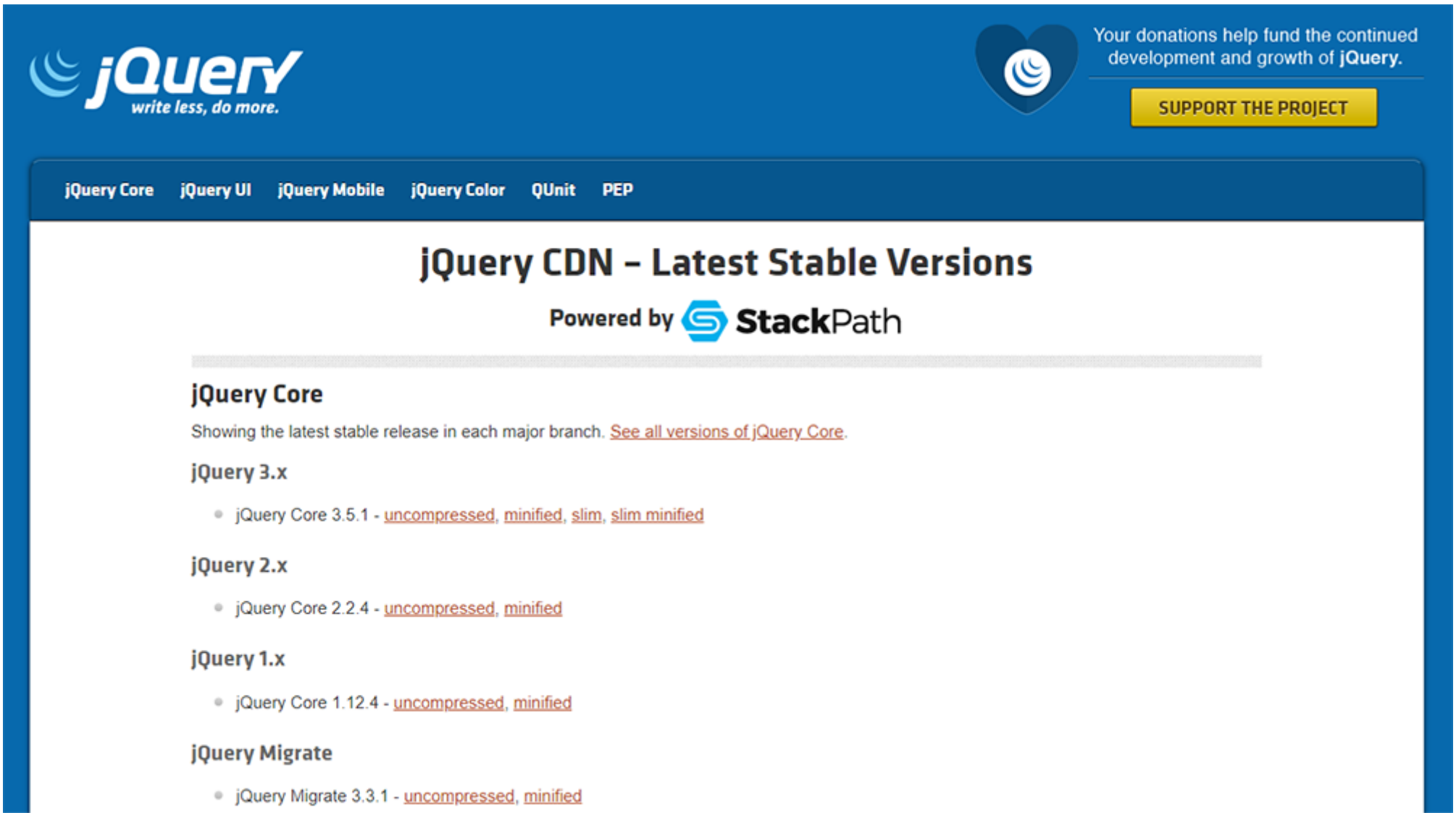
Unless you have some compelling need to view source code or eliminate jQuery AJAX from your site, minified is generally considered the best one to use.

Creating a CDN

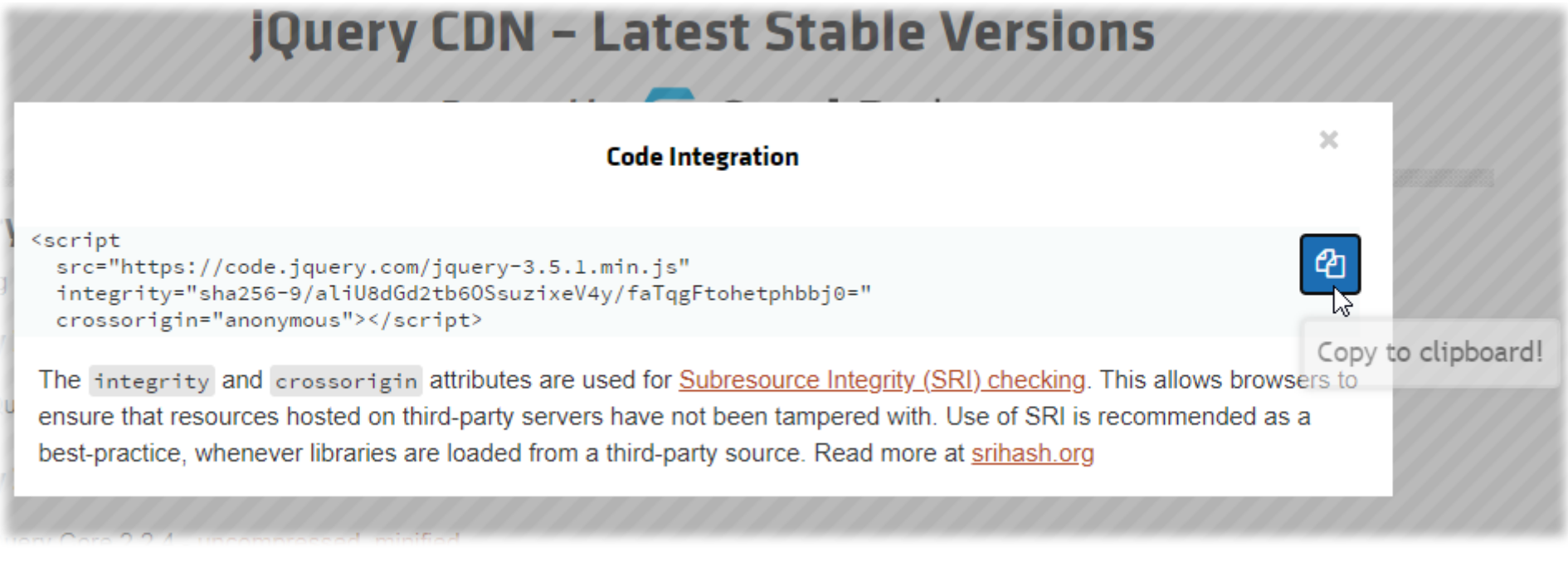
By far, using the minified version via CDN is probably the most common approach for your typical website. Using it in your code is easy. Let's give it a try!

Here are the Steps

1. Open your *jQueryTemplate.html* in your editor.
2. In your browser, go to https://code.jquery.com.



3. Click the **minified** link for the most up-to-date jQuery version on the page. A popup box will appear with the code you need for the jQuery CDN link.



4. Click the **Copy to Clipboard** button.

5. Below the <title>...</title> tags, type the following comment as a future reminder to yourself:

```
<!-- jQuery CDN from code.jquery.com -->
```

6. Just below that line, paste in the code you copied from the CDN.

7. Save your changes.

Sneak Peek

The final *jQueryTemplate.html* page should look something like this:

```
<!DOCTYPE html>

<html>

<head>

<title>jQueryTemplate</title>

<!-- jQuery CDN from code.jquery.com -->

<script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-
9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0=" crossorigin="anonymous">

</script>

</head>

<body>

</body>

</html>
```

Note

Any JavaScript code can be stored in an external file and linked to the current page using a <script src="..."> tag.

Now that you have a web page with a link to jQuery in it, you can write jQuery code in that page.

Writing jQuery

Even though jQuery is JavaScript, the syntax for using jQuery is a bit different. To refer to code in the jQuery library, you use the $ symbol. Think of the dollar sign as meaning one or all of the following:

- *using jQuery*library
- *get from the jQuery* library
- *retrieve from jQuery* library

Whatever word works for you is fine, so long as you understand that the $ is basically telling the browser to look in the jQuery library for the JavaScript code to execute.

Execute jQuery after Page Load

As with regular JavaScript, *before* jQuery attempts to act on an element it's important that they meet the following criteria:

- They actually exist in the page.
- They are fully rendered on the screen.

If jQuery code attempts to act on any elements in the page that either nonexistent or not fully rendered it causes an error. So, like regular JavaScript, jQuery has a way of saying *Let the entire page load before trying to execute any jQuery code*. The syntax looks like this:

```
<script>
$(document).ready(function() {
 // jQuery code to execute after page load
});
</script>
```

The <script> . . . </script> tags are necessary because it's JavaScript code. But what is the jQuery code doing?

Let's break it down.

Text equivalent start.

Topic	Information

Topic	Information
`$`	This is the character that tells the browser to look in the jQuery library for the code to execute on the rest of the line.
`(document).ready`	This is jQuery syntax that says: *when the page is fully loaded and ready*.
`(function(){…})`	This part sets up an *anonymous function*—one or more lines of code that are executed in sequence, like any JavaScript function. It's called an *anonymous* function because it doesn't have a specific name. It's just any amount of code that becomes available as soon as the *document.ready* event loads, telling the browser that it's safe to execute any jQuery code.

Read the topic in the first column. Then read the second column for the information.

Text equivalent stop.

The parentheses and curly braces are a little daunting in jQuery. In fact, for beginners, they're probably the scariest and hardest thing to get used to. To help clarify, the closing curly brace and parentheses at the end are really just dangling there. They are simply closing the open parentheses and curly brace.

Text equivalent start.

```
$(document) .ready (function() {
        //jQuery code to execute after page load
    });
```

Text equivalent stop.

All the jQuery code for any given page is usually placed inside the curly braces and parentheses of that block to ensure that none of the code tries to execute before the page is ready for code execution.

Note

Virtually all jQuery code that you write yourself should be placed inside a *$(document).ready(function () { . . . });* block. The *document.ready* code is always the same, so it's unlikely that you'd ever have to type it from scratch. If you keep a copy someplace where it's easy to find (or just look it up online as needed), then you can just copy and paste it into any page that uses jQuery.

Shorthand Syntax

There is a shorthand alternative syntax that you can use. That syntax is:

```
<script>
$(function() {
  // jQuery code to execute after page load
});
</script>
```

Some people prefer it because it's shorter, however, it's also less descriptive. The longer *document.ready()* syntax at gives you a better idea as to what the code means.

So in this course, I'll show the longer syntax. Out in the real world, you may see the shorter syntax used quite often, and of course you're welcome to use it yourself, if you prefer.

Placing jQuery on the Page

As mentioned, all of the jQuery code on a page is typically placed inside the curly braces and parentheses of the *document.ready* block (or its shorthand equivalent). Let's add the document-ready code to our sample working page.

Here are the Steps

1. Go back to *jQueryTemplate.html* in your editor.

2. Put the cursor just after the closing </script> tag, and press **ENTER** to add a blank line.

3. Type or copy/ paste the following code so it's below that closing </script> tag but still above the </head> tag in the page.

```
<script>
$(document).ready(function() {
 // jQuery code to execute after page load
});
</script>
```

4. Save the page.

Sneak Peek

Now your *jQueryTemplate.html page* should look something like this:

```
<!DOCTYPE html>

<html>

<head>

    <title>jQueryTemplate</title>

    <!-- jQuery CDN from code.jquery.com -->

    <script src="https://code.jquery.com/jquery-3.5.0.min.js"

        integrity="sha256-xNzN2a4ltkB44Mc/Jz3pT4iU1cmeR0FkXs4pru/JxaQ="
crossorigin="anonymous">

        </script>

    <script>

        $(document).ready(function () {

            // jQuery code to execute after page load

        });

    </script>

</head>

<body>
</body>

</html>
```

Note

You can keep this file as a kind of a template for any new pages that you create in the future that will use jQuery code. That way you don't have to retype all of that complicated jQuery code from scratch to get started on the new page.

Now, let's look at some cool things you can do with jQuery.

Chapter 2: Writing jQuery Code

Similar But Different from JavaScript

Now, let's take a closer look at how you write jQuery code. Even though jQuery is JavaScript, the syntax is different from what you learned in previous lessons because you rely on the *$* to call prewritten code from the jQuery library. The basic syntax of a chunk of JavaScript code follows the same pattern as the *document.ready* code in which it will be contained:

```
$("selector").event(function(){
  code to execute;
 });
```

- Replace *selector* with the element, or type of element, that will call the code.
- Replace *event* with the name of the action that, when performed on the element or element type, will call the code.

In that generic $(*selector*).*event* syntax, you can put just about any CSS selector in place of *selector* to apply the code to any number of elements on a page, just as you can use CSS selectors to apply CSS styles to any number of elements on a page.

Example

In the following code we used the *button* selector and the *click* event:

```
$("button").click(function(){
 code to execute;
});
```

This means that whenever the user *click*s a *button* on the page, the *code to execute* will run.

Important

In the example above, it means *any button*, not just a specific button. Having the option to easily write code that applies to multiple similar elements on the page is one of jQuery's best features. However, it also means that you need to make sure the *code to execute* is something you want for all of said elements.

Let's explore the selectors in a bit more depth.

Understanding jQuery Selectors

The jQuery language uses a selector naming scheme similar to that of CSS to apply code to elements on a page. One of the beauties of jQuery is that you get the best of both worlds with selectors.

- You can write a jQuery function that applies to a single element on a page.

  OR

- You can write a jQuery function to apply to many elements on a page.

The selector syntax that you use to target the function is the same as the selector syntax that you use for targeting CSS style rules.

Common jQuery Selectors

Here's a reference to some of the most commonly used CSS-style selectors that can be used in jQuery:

CSS-Style Selector	jQuery Selector	What it Targets
*	$("*")	All elements on the page
element type	$("li")	All text in . . . tags
#name	$("#OKbutton")	Element that contains *id="OKbutton"*
.classname	$(".picture")	All elements that contain *class = "picture"*
:first	$("h1:first")	Text in first . . . tags
:last	$("div:last")	Text in last . . . tags

CSS-Style Selector	jQuery Selector	What it Targets
:even	$("tr:even")	All even-numbered table rows
:odd	$("tr:odd")	All odd-numbered table rows
:first-child	$("li:first-child"	All list items that are the first child of their parent element
:last-child	$("li:last-child")	All list items that are the last child of their parent element
:nth-child(*x*)	$("tr:nth-child(2)")	Second row in all tables
parent descendant	$("#test li")	All list items contained in the list identified by *id="test"*
[attribute]	$("[href]")	All tags that contain an href= attribute
[attribute = value]	$("[type=button]")	All tags that contain type="button"

Commonly Used Selectors

Applying jQuery to All Similar Elements

If the selector matches an element type, then the code applies to all elements of that type. If you use *"p"* as the selector, the jQuery code apply to all paragraphs (text in . . . tags) on the page. Using *"h1"* as the selector makes the code apply to all text in <h1> . . . </h1> tags on the page. Using *"button"* as the selector makes the code apply to all <button> . . . </button> tags on the page.

Let's look at an example to illustrate.

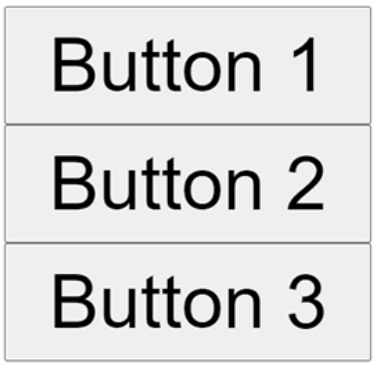
Here are the Steps

1. Open the *jQueryButtons.html* file you downloaded earlier in your editor.

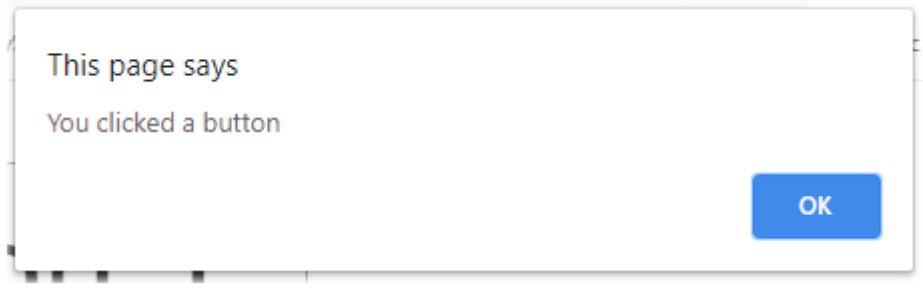
2. Notice that it uses *$("button").click* as a jQuery selector.

3. Also notice, that in the body of the page, there are three buttons, each defined with a pair of <button> . . . </button> tags.

4. Now, open jQueryButtons.html in your browser. You should see three buttons displayed on the page.



5. Click any one of the buttons. This executes the JavaScript code and displays an alert box that says *You clicked a button* alert box.



In real life, we often want a specific element, not all elements of a particular type, to trigger some event. Of course, the folks who created jQuery knew this and made it easy to do things that way, too.

Applying jQuery to a Single Element

As you may recall, in HTML you can uniquely identify any element on a page by giving it an ID name. In CSS, you can use # in a selector to apply the style to the element that has that ID name. The same concept applies to jQuery. You can use #*name* as a selector to target one element on a page.

Let's change our *jQueryButtons.html* page to see how it works.

Here are the Steps

1. Go back to *jQueryButtons.html* in your editor.

2. Notice that the buttons already have ID names: *button1*, *button2*, and *button3*.

3. To tie the jQuery to just one button on the page, replace the generic *button* selector with the more specific selector *#button2*.

This targets the specific button you want.

In this case, we've changed the code so that the alert only shows if the user clicks *button2*.

4. Save your changes.

5. Now, open or refresh *jQueryButtons.html* in your browser.

6. Click *Button 1* or *Button 3*. Nothing happens right?

7. Now, click *Button 2*. This time the alert box should appear.

As you can see, while the page still shows three buttons clicking Button 1 or Button 3 no longer calls the alert box. This is because now the code uses the *#button2* selector in the jQuery code, which links the code directly to the one and only button that contains *id="button2"*.

Understanding jQuery Events

While the selector describes the element or elements that trigger an event or are acted upon by jQuery, the event describes an action that must be performed to fire (execute) the jQuery code. So, let's look refer back to our original *jQueryButtons.html* example:

```
$("button").click(function(){
 alert("You clicked a button");
});
```

Here *button* is the selector and it applies to all buttons on the page, whereas *click* is the event that must happen on a button in order for the code to execute.

Common jQuery Events

Just like regular JavaScript, jQuery offers handlers for many different kinds of events. Below are some other commonly used jQuery events.

Event Handler	jQuery Event	What Action Needs to Happen
click	$("img").click	The user clicks or taps an image on the page.
dblclick	$("img").dblclick	The user double-clicks an image on the page.
mouseenter	$("img").mouseenter	The mouse pointer hovers over an image.

Event Handler	jQuery Event	What Action Needs to Happen
mouseleave	$("img").mouseleave	The mouse pointer moves away from hovering over an image.
mousedown	$("img").mousedown	The user puts the pointer on the image and presses any mouse button.
mouseup	$("img").mouseup	The mouse pointer is on an image, and the user has pressed and released a mouse button.
focus	$("input").focus	The cursor or keyboard focus is on an input element.
blur	$("input").blur	The cursor or keyboard focus has left an input element.
keydown	$("input").keydown	The cursor is in an input element, and the user has pressed a key.
keyup	$("input").keyup	The cursor is in an input element, and the user has pressed and released a key.

Commonly Used JQuery Events

In all your jQuery code, you'll combine a selector with an event to define which element (or elements) on the page trigger some code. That code can be regular JavaScript or it can be more jQuery code. One of the most common things is to have the event call jQuery code that forms some special effect on the page. Up next, we'll show you some neat things you can do with jQuery effects.

Chapter 3: Adding Special Effects with jQuery

Understanding jQuery Effects

The ability to apply JavaScript similarly to how you can apply CSS is one of jQuery's best features and part of the reason for its widespread success. However, in addition to being able to execute any JavaScript you want within jQuery, you can also use some of its built-in *effects*. These are similar to CSS3 transition effects, but have one advantage main advantage: unlike CSS3 transitions, jQuery effects are written to work with older browsers, not just the newest browsers that have CSS3 support.

Common jQuery Effects

Here is a quick overview of some of the most popular effects:

Effect	What It Does
.fadeIn	Fades from hidden to opaque
.fadeOut	Fades from opaque to invisible
.fadeTo	Fades to a specified opacity
.fadeToggle	Switches between opaque and invisible by fading
.hide()	Hides the selected elements
.show()	Shows the selected elements
.slideDown()	Reveals selected element by sliding down
.slideToggle()	Shows or hides the selected element by sliding up or down
.slideUp()	Hides the selected element by sliding up
.toggle()	Shows the element if hidden, hides it if visible

Common JQuery Effects

jQuery Effect Syntax

The syntax for implementing an effect is similar to the syntax for everything else in jQuery. Here is the basic syntax:

```
$("selector").effect(value);
```

- The *selector*, as always, identifies the element or elements to which the effect is to be applied.
- The *effect* is any valid jQuery effect name.
- The *value* part is optional and can be omitted. If included, it has to be a valid jQuery keyword, like *slow* or *fast*, or a number expressing the number of milliseconds for the effect to play out, such as 1000 for one second.

The effect code is usually contained within some event-handling code that indicates when the effect should be played out. So from a larger perspective, the syntax would look like this:

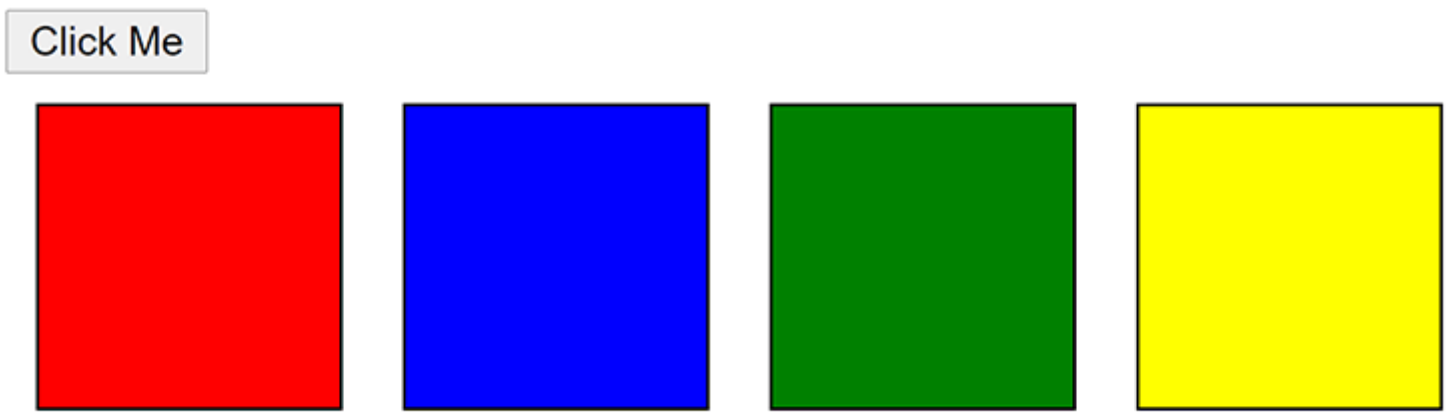
```
<script>

$(document).ready(function () {

// jQuery code to execute after page load

$("selector").event(function () {

$("selector").effect(value);

});

});

</script>
```

- The *document.ready* ensures that no jQuery code is executed until the page is loaded and ready for action.
- The first *selector* and the *event* indicate the element and action that will trigger the effect.
- The second *selector* and the *effect* indicates the element (or elements) to which an effect will be applied when the specified *event* happens.

That's a lot to think about, but it's actually simpler than it sounds. A little hands-on always helps, so let's give it a try.

How jQuery Effects Work

If you open the *jQueryDemo1.html* page you downloaded earlier in a browser, you should see a button and four colored squares:



- The HTML code for the button and four squares is in the <body>...</body> tags.
- The style of each box in terms of size, border, and appearing side by side are CSS defined by the div{} style rule in the internal style sheet.
- If you click the **Click Me** button, the boxes should fade out of view.
- If you click the **Click Me** button a second time, the boxes will fade back into view.

You can click the button any number of times to fade the boxes into and out of view. That part is handled by jQuery code:

```
// jQuery code to execute on any button click
  $("button").click(function () {
   $("div").fadeToggle();
  });
```

Let's discuss how that part works.

Text equivalent start.

Topic	Information
`$("button").click(function () {`	The event is *click*, and the selector is *button*. This line of jQuery applies to any button on the page. Since there's only one button on this sample page, it applies to that one button. When the button is clicked, the next line of jQuery code executes.

Topic	Information
`$("div")`	This part means "every div on this page" (each div being defined by a pair of . . . tags).
`.fadeToggle`	This effect makes all the divs fade out of view if they're showing or back into view if they're hidden.
`();`	No *value* was specified inside the parentheses of the fadeToggle() effect. So the default duration of 400ms (a little less than one second) is used.

Read the topic in the first column. Then read the second column for the information.

Text equivalent stop.

All jQuery code that you write in the future will be based on the basic principles you learned today. Of course, there are limitless possibilities. We've only shown you enough to illustrate the basic code syntax for jQuery, but it's enough to get you started.

In the assignment for this lesson, we'll play around with some other effects and durations. For now, let's wrap up with a review of what you've learned today.

Review

Lesson 11 Review

jQuery is so ubiquitous in Web development, it's considered to be virtually synonymous with JavaScript. In fact, when researching JavaScript on your own online, you'll often find answers to your questions written in jQuery syntax rather than standard JavaScript. For that reason you want to learn jQuery as part of learning about JavaScript.

- **Getting Started with jQuery:** Today's lesson introduced you to a whole new language of JavaScript named jQuery. jQuery actually is a widely used library of prewritten JavaScript code that allows you to do more with JavaScript using relative short, simple code. You can link to this library using a <script> tag with the src= attribute pointing to a copy of the jQuery library on your own hard disk or on a Content Delivery Network (CDN) such as code.jquery.com.

- **Writing jQuery Code:** You discovered that the $ symbol is key to writing jQuery code. To write a jQuery event handler, use the syntax *$("selector").event(function () { code to execute });* inside the document read block. The *selector* is a CSS-style selector that indicates the element (or elements) on the page that can trigger the code. The *event* is the jQuery name of the event that triggers the code, such as *click* for a mouse click. The *code to execute* is the JavaScript or jQuery you want to run in response to the event.

- **Adding Special Effects with jQuery:** jQuery provides numerous special effects that make it easy to add some motion and animation to all browsers, even pre-CSS3 browsers that don't support CSS3 transitions. To apply a jQuery effect, use the syntax $("*selector*").*effect*(*value*); inside the event-handling block. Replace *selector* with the element (or elements) to which the effect will be applied. Replace *effect* with the effect to apply. And use the optional value argument to define how long the effect should last.

In the next lesson, you'll have a chance to learn more about jQuery and try out some more fun effects that can add real pizzazz to your Web pages!

Lesson 11 Assignment

Modifying jQuery

At the end of this lesson, you created a page named *jQueryDemo1.html*. It contains a button that, when clicked, causes four colored boxes to fade into or out of view. Your challenge for this assignment is to change the code on that page so that the following happens:

- When you click the button, the boxes *slide up* out of view or *slide down* into view.
- The duration of the slide should be two seconds (2000 milliseconds).

Try it on your own first, if you need some help, click the sentence in the **Here are the Steps** box below for step-by-step instructions.

Tip

You can use the *slideToggle* effect. You do not need the *fadeToggle* effect if you use *slideToggle* instead.

Here are the Steps

If you did everything right, the boxes should appear or disappear with each button click. But rather than fading, they slide into and out of view by growing or shrinking in height.

Lesson 11 Resources for Further Learning

jQuery (https://jquery.com/)

https://jquery.com/

As the name and URL imply, this is home to the entire jQuery effort. You can download the jQuery library from here, keep up on news and events, and explore their tutorials and references. And it's all free!

SelectTutorial (http://css.maxdesign.com.au/selectutorial/)

http://css.maxdesign.com.au/selectutorial/

Selectors play a huge role in jQuery, just as they do in CSS. This is one of the best tutorials around for learning CSS selectors. You can use the same selectors in jQuery to target your code to elements on the page.

---

jQuery Tutorial for Beginners: Nothing but the Goods (https://www.impressivewebs.com/jquery-tutorial-for-beginners/)

https://www.impressivewebs.com/jquery-tutorial-for-beginners/

Here's another good tutorial for first learning jQuery, though again it gets into some more-advanced stuff rather quickly.

---

jQuery Tutorial (https://www.w3schools.com/jquery/default.asp)

https://www.w3schools.com/jquery/default.asp

W3Schools is always a good resource for all things related to Web development. Click this link to get to the home page for the jQuery tutorials and references.

---

jQuery Cheat Sheet (https://cdn.woorkup.com/wp-content/uploads/2016/01/jQuery-1.5-Visual-Cheat-Sheet.pdf)

https://cdn.woorkup.com/wp-content/uploads/2016/01/jQuery-1.5-Visual-Cheat-Sheet.pdf

This is a multipage cheat sheet and so doesn't really jibe with the usual one-page notion of a cheat sheet. But it's worth a look. See the next link for more.