Introduction

Welcome Back!

In Lesson 11, you learned about jQuery, which is a "write less, do more" JavaScript library. You got to see how you can incorporate the jQuery library into your own websites. You learned a bit about its selectors and event handlers, and the rather odd-looking syntax it uses. But you didn't really get a chance to create anything useful with it yet.

This lesson aims to rectify that. Today you'll use jQuery to create *collapsible panels* and *accordions*. You might not recognize those names. But once you see them in action, you'll probably recognize them as things you've used on other peoples' websites. We'll use jQuery to create these things, mainly because they're exactly the kinds of things the jQuery is so good at allowing you to create with a minimum of code.

So, let's get right to it!

Download

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

- **Collapsible Panel:** We'll use this to create a collapsible panel. This is a sample page with just the HTML and CSS you need.
- **Accordion:** We'll use this to create an accordion. This is a sample page with just the HTML and CSS you need.

Download Lesson-12_ClassProject.zip

1.6 KB (Gigabytes) ZIP

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

Chapter 1: Creating a Collapsible Panel

What is a Collapsible Panel?

A *collapsible panel* is a fairly common control in modern Web pages and mobile apps. It is basically a bar that shows some brief text, then when clicked it reveals expanded information.

Text equivalent start.

Topic	Information
This is a collapsible panel. A user who's interested in more information about the topic (or the answer to the question) can click or tap the bar.	Doing so causes a panel to slide down and display the detailed information or answer. When you're done viewing that, you can click or tap the top bar again, and the panel will collapse up into the top bar, disappearing from view.

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

Text equivalent stop.

Because of its built-in effects, jQuery is the perfect language for creating this kind of collapsible panel. Rather than have the panel appear instantly on hover (as we'd get with CSS), the user can click or tap the top bar, and we can have the panel slide up or down slowly and gracefully.

Styling Your cPanel

The *cPanel.html* page currently starts with the hidden panel already showing, so you can see everything and make any stylistic adjustments.

Note

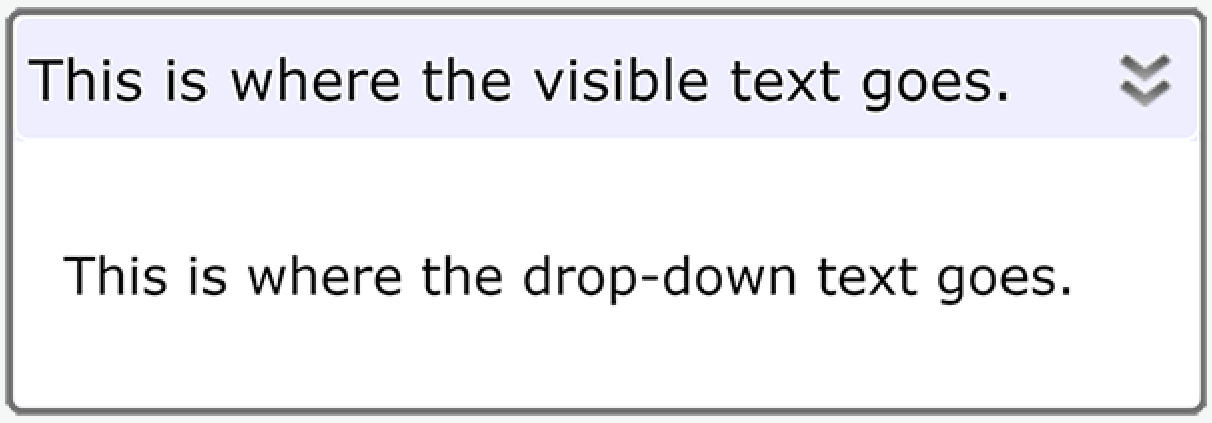
The page uses one of the sample icons from the *icons* folder that you downloaded for this course, that icon will only show if that *icons* folder is in the same folder as the page.

Here are the Steps

1. Open *cPanel.html* in your browser.

Sneak Peek

You should see a collapsible panel with the hidden panel already expanded out.


This is where the visible text goes.

This is where the drop-down text goes.

2. Click the top bar. This will have no effect, because we haven't written any code for that yet.

Now, let's take a look at how this page is set up. We'll start with the HTML:

```
<div class="cPanel">
        <!-- Text for the top bar goes in this first div -->
        <div class="top expand">
            This is where the visible text goes.
        </div><!-- End top -->
        <!-- The slide-down text goes in this second div -->
        <div class="slidepanel">
            <p>This is where the drop-down text goes.</p>
        </div><!-- End slidepanel -->
    </div><!-- End cpanel -->
```

When using the code in real life...

Make sure you put both the top bar text and the slide-down text in the div that has the ID of *cPanel*. You can use paragraphs (<p> . . . </p> tags), images (<img> tags), and any other HTML elements inside that div. It's just a standard HTML page division and doesn't need any special treatment.

As per the comments in the code, brief text for the top bar must be placed inside the div to which the class names *top* and *expand* are applied.

```
<!-- Text for the top bar goes in this first div -->
<div class="top expand">
    This is where the visible text goes.
</div><!-- End top -->
```

Content for the slide-down panel should be placed inside the div to which the class name *slidepanel* is applied.

```
        <!-- The slide-down text goes in this second div -->
        <div class="slidepanel">
            <p>This is where the drop-down text goes.</p>
        </div><!-- End slidepanel -->
    </div><!-- End cpanel -->
```

Most of the styling is handled by CSS, of course. Here are the CSS style rules in the page that we're using as our working example:

```
    <style>
    /* Style for the entire collapsible panel */
    .cPanel {
        background-color: #eef;
        width: 80%;
        margin: 1em auto;
        border: solid 2px #777;
        border-radius: 5px;
        font-family: Verdana, Tahoma, Sans-Serif;
    }
    /* Style for the top bar */
    .top {
        padding: 8px 48px 8px 4px;
        border: solid 1px white;
        border-radius: 5px;
        font-size: 1.1em;
    }
    /* Shows collapse button */
    .collapse {
        background: url(icons/collapse.png) no-repeat 98% center;
    }
    /* Shows expand button */
    .expand {
        background: url(icons/expand.png) no-repeat 98% center;
    }
    /* Style for panel that drops down.*/
    .cPanel .slidepanel {
        margin: 0;
        padding: 1em;
        background-color: white;
        border-bottom-left-radius: 5px;
        border-bottom-right-radius: 5px;
    }
</style>
```

When using the code in real life...

We put the CSS right in the page, just to keep the code together for easy future reference. However, you're welcome to put the CSS code in an external style sheet to share across pages. Feel free to change fonts, colors, sizes, margins, and padding to your liking.

The *.collapse* and *.expand* style rules simply display a background image from the *icons* folder. The *.collapse* style class shows the up-pointing chevron, a common symbol for "collapse." The *.expand* style class shows the down-pointing chevron image, a common symbol for "expand." We're using CSS style classes to display one image or the other because jQuery makes it easy to toggle between two different style classes as needed in a page.

```
/* Shows collapse button */
.collapse {
    background: url(icons/collapse.png) no-repeat 98% center;
}
/* Shows expand button */
.expand {
    background: url(icons/expand.png) no-repeat 98% center;
}
```

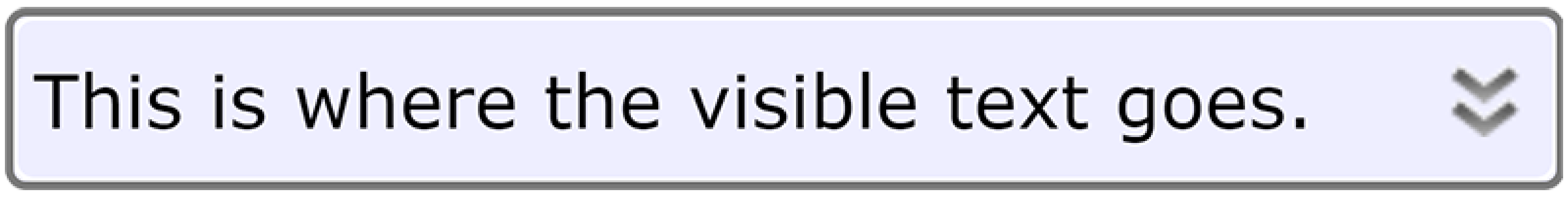
Feel free to make any stylistic adjustments you'd like to the page. You can customize the text displayed, add images, change the background colors, remove or thicken borders, and so on.

Hiding the Slide Panel

For the moment, the code is allowing the slide panel to show on the page, which makes it easier to play around with the styling. However, for our collapsible panel we need that to start out hidden. Let's go ahead and make that happen now.

Here are the Steps

1. Open *cPanel.html* in your editor.
2. Locate the style rule that has *.cPanel .slidepanel* as its selector.
3. Add *display:none;* to the bottom of that style rule. This will make it so that the slide-down panel is hidden by default.
4. Save the page.
5. Now open or reload or refresh the page in the browser, the slide-down panel will disappear.


This is where the visible text goes.

Next, we need to write some jQuery code to make it expand and collapse in response to a mouse click or tap.

jQuery for Collapsible Panels

To control the action on our collapsible panel, we're going to use some jQuery code. The page already contains the *document.ready* code and link to the jQuery library that you learned about earlier. So we just need to work inside the *document.ready* block:

```
 <script>
//jQuery code.
$(document).ready(function () {
});
</script>
```

First, we need an event handler. In other words, we need a way of saying to jQuery, "When the user clicks that top bar, execute the following code."

Remember

Event handlers in jQuery are different from event handlers in JavaScript. In jQuery, we use this syntax inside the *document.ready* block:

```
$('elements(s)').event(function () {
});
```

Identifying the Trigger

For this sample page, the element that we want to trigger the panels is the top bar. So, it will be the div that has *class="top"*in its tag. To play it safe, we can be more specific and say *the div that has class="top" that's inside a div that has class="cPanel"*.

The specific event that we want to capture is a mouse click (or a tap on a touchscreen), which is the *.click* event in jQuery. So that means the element and event need to be written as *$('.cPanel .top').click*.

Here are the Steps

1. Open *cPanel.html* in your editor.

2. Put the cursor just below the *$(document).ready(function () {* line.

3. Type or copy and paste the following comment and code:

```
//User clicked top of a collapsible panel
$('.cPanel .top').click(function () {
});
```

Important

If you type it yourself, be careful with your uppercase and lowercase letters, and watch those parentheses and curly braces. Everything has to be exact for this to work. All those parentheses and curly braces (not to mention the case-sensitivity) take some getting used to in jQuery.

4. Save changes.

Expanding and Collapsing Panels

So now we have a jQuery event handler that says when the user clicks a top div inside a cPanel div, we want something to happen. We actually want two things to happen:

1. Show the collapse button if the expand button is showing (or vice versa).
2. Expand the slide-down panel if it's hidden (or vice versa).

jQuery's various *.toggle* methods are ideal for switching between two possible states. Two that will really come in handy here are:

- **.toggleClass:** toggles a CSS class name on or off or between CSS class names (like *collapse* and *expand*, each of which shows a different background image)
- **.slideToggle:** slides an element down into view, or up out of view, in a smooth motion

Tip

In computer languages, the word *toggle* is often used for any value that can switch between two states. A light switch is a good example. It's always in one of two possible states, either *on* or *off*.

The tricky part, as always, is getting the syntax right in the code.

.toggleClass

When you specifically want to toggle back and forth between two CSS class names, you can use this syntax:

```
$(elements).toggleClass('classname1 classname2');
```

- Replace *elements* with a selector that identifies the element or elements to which the CSS class should be applied.
- Replace *classname1* and *classname2* with the class names of the style classes to switch between. Note that they must be separated by exactly one space. And, as always, case-sensitivity reigns supreme.

For our purposes, the element upon which we want to toggle the class names is whichever *top* div the user clicked or tapped. But there might be multiple collapsible panels on a page. So we don't want to identify all the elements with *class="top"*— just the specific one that the user clicked or tapped.

In jQuery (and JavaScript too), the keyword *this* identifies the specific element that triggered an event. So we can write our *toggleClass* line of code like this:

```
$(this).toggleClass('collapse expand');
```

That line says, "On whatever top bar the user clicked or tapped, apply the *expand* style class if the *collapse* class is currently applied, or apply the *collapse* class if the *expand* class is currently applied." That's all we need to have to switch between the up-pointing and down-pointing chevron images.

.slideToggle

We also need the hidden panel to slide down if it's currently hidden. Or, if it's already showing, we need to collapse it back up and out of view. For that we can use jQuery's *.slideToggle*, which requires this syntax:

```
$(elements).slideToggle(optionalSpeed);
```

- The *elements* should identify the element (or elements) that should slide in or out of view.
- The *optionalSpeed* can be nothing (empty parentheses), in which case the slide takes 400 milliseconds (ms) to complete. Optionally, you can put in your own number, in milliseconds, or the keyword *'fast'* or *'slow'*. If you use a keyword, you must enclose it in single or double quotation marks.

For our example, the syntax is a little tricky because we don't want the item that was clicked to slide into and out of view. We want the div that has the class name *slidepanel* that's directly *below* the top div we clicked to slide into and out of view.

Fortunately, jQuery offers a *next* method that gets the next element below any specified element. So we can refer to the *slidepanel* div that's just below the clocked item as *$(this).next('.slidepanel')*. That phrase basically says, "the next slidepanel item just below this thing that was just clicked or tapped."

However, that part just identifies the element we want to apply an effect to. We also have to specify which effect we want to apply. We'll use *.slideToggle* because we want it to slide down into view (if it's hidden) or up out of view (if it's showing).

The speed is entirely up to you. You can set the speed of the slider to *'slow'* to *'fast'* or with a number in milliseconds. We'll use *slow* for our working example. The exact syntax would be:

```
$(this).next('.slidepanel').slideToggle('slow');
```

Note

When using the words fast or slow to define the speed, the words must be in quotation marks:

```
$(this).next('.slidepanel').slideToggle('slow');
```

When using numbers to define the speed in milliseconds, no quotations are necessary:

```
$(this).next('.slidepanel').slideToggle(3000);
```

Let's put in the code and try it out!

Here are the Steps

1. Open *cPanel.html* in your editor.
2. Put the cursor just below the *$('.cPanel .top').click(function () {* event handler, but above the *});* that marks the end of that block.
3. Type or copy and paste the following comments and code.

```
//Switch between the two background images
$(this).toggleClass('collapse expand');
//Slide the panel up or down
$(this).next('.slidepanel').slideToggle('slow');
```

4. Save the page.
5. Open or refresh the page in the browser.
6. Click the top bar. The sliding panel should slide down into view, and the chevron icon should switch to pointing up.
7. Then click the top bar again. The panel should slide up out of view, and again the chevron icon flips over.

You can put as many collapsible panels as you wish onto a page. However, if you're going to have a stack of them close to each other, you may be better off using an *accordion*. You may have seen some of those in webpages or mobile apps. If you're not sure, don't worry you'll create your own accordion next and see it action.

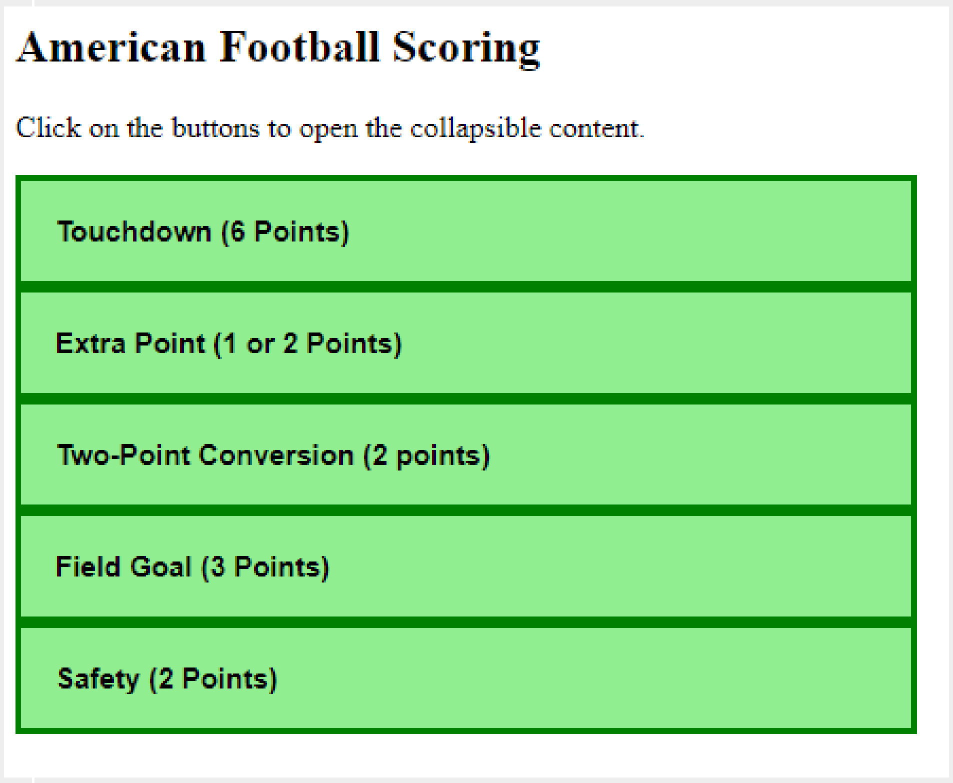
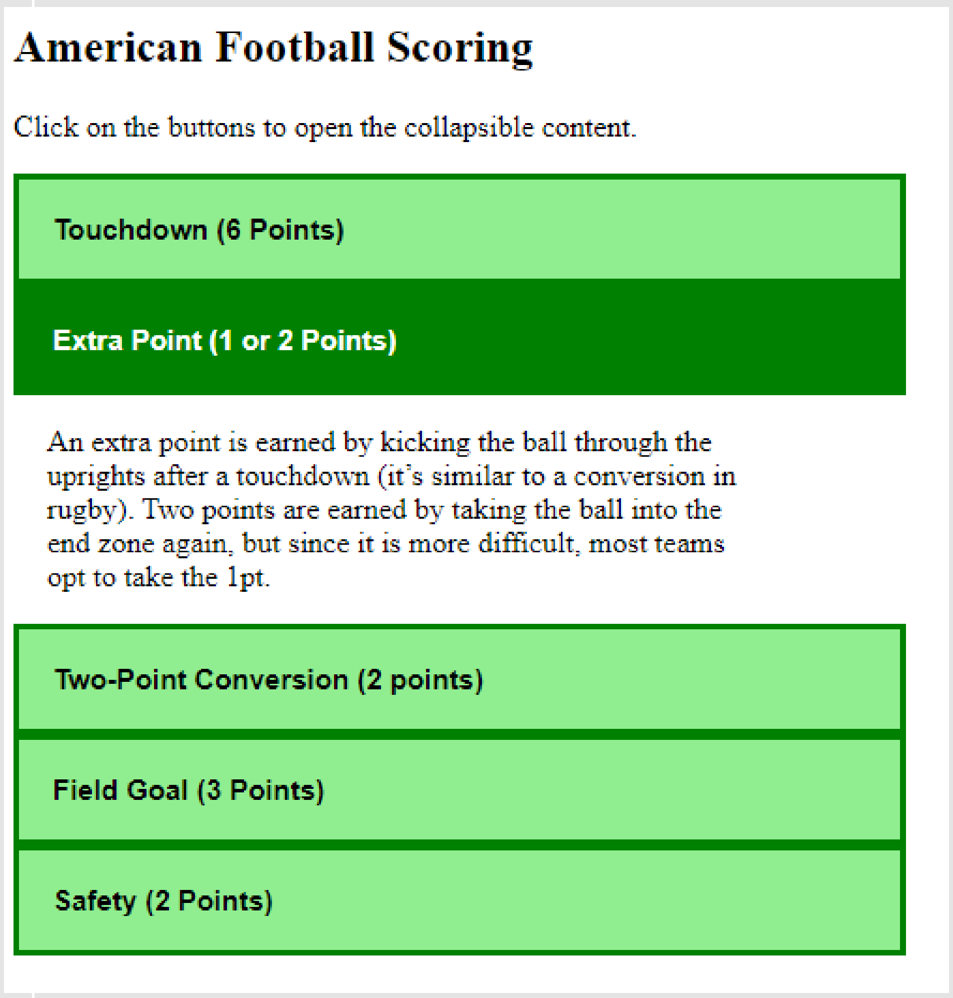
Chapter 2: Creating an Accordion Control

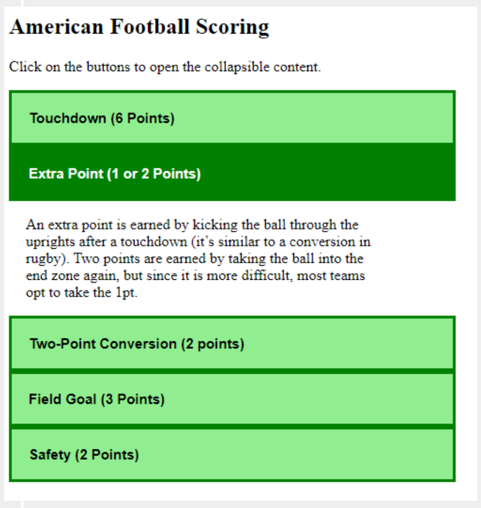
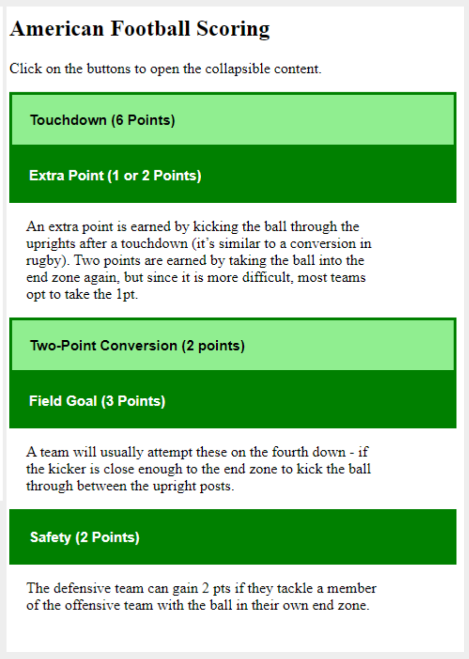
What is an Accordion?

Accordion controls are fairly common in mobile apps, and are also sometimes used in websites. When we say accordion, we don't mean the musical instrument. We're talking about a feature similar to the collapsible panel, that enables you to have multiple items expand and collapse.

Text equivalent start.

Topic	Information

Topic	Information
Accordion Control Collapsed	The accordion control is ideal for definitions or details about other short items of text because it allows the user to see all the options at a glance:  American Football Scoring Click on the buttons to open the collapsible content. Touchdown (6 Points) Extra Point (1 or 2 Points) Two-Point Conversion (2 points) Field Goal (3 Points) Safety (2 Points)
Accordion Control Expanded	To see a definition, the user just clicks (or taps) the term. The definition slides down below the term.  American Football Scoring Click on the buttons to open the collapsible content. Touchdown (6 Points) Extra Point (1 or 2 Points) An extra point is earned by kicking the ball through the uprights after a touchdown (it's similar to a conversion in rugby). Two points are earned by taking the ball into the end zone again, but since it is more difficult, most teams opt to take the 1pt. Two-Point Conversion (2 points) Field Goal (3 Points) Safety (2 Points)

Topic	Information
Accordion Control Options depending on how you set up your accordion, it can either:	 **American Football Scoring** Click on the buttons to open the collapsible content. Touchdown (6 Points) Extra Point (1 or 2 Points) An extra point is earned by kicking the ball through the uprights after a touchdown (it's similar to a conversion in rugby). Two points are earned by taking the ball into the end zone again, but since it is more difficult, most teams opt to take the 1pt. Two-Point Conversion (2 points) Field Goal (3 Points) Safety (2 Points)  **American Football Scoring** Click on the buttons to open the collapsible content. Touchdown (6 Points) Extra Point (1 or 2 Points) An extra point is earned by kicking the ball through the uprights after a touchdown (it's similar to a conversion in rugby). Two points are earned by taking the ball into the end zone again, but since it is more difficult, most teams opt to take the 1pt. Two-Point Conversion (2 points) Field Goal (3 Points) A team will usually attempt these on the fourth down - if the kicker is close enough to the end zone to kick the ball through between the upright posts. Safety (2 Points) The defensive team can gain 2 pts if they tackle a member of the offensive team with the ball in their own end zone. • Close the one that's already open. In this case, there's always only one definition visible at a time. • Keep the one that's already open. In this case, all definitions can be visible at the same time and you need to click the term again to close it. Either way, the whole control grows and shrinks like an accordion as you look at different definitions.

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

Text equivalent stop.

We can use the jQuery *slideToggle* once again for the panel that slides in and out of view. But we'll need some additional code to ensure that only one definition panel is visible at a time. As a rule, accordion controls don't have expand and collapse icons. Usually, there's no icon (as in the example above) or just a simple angle bracket as a visual code.

Styling Your Accordion

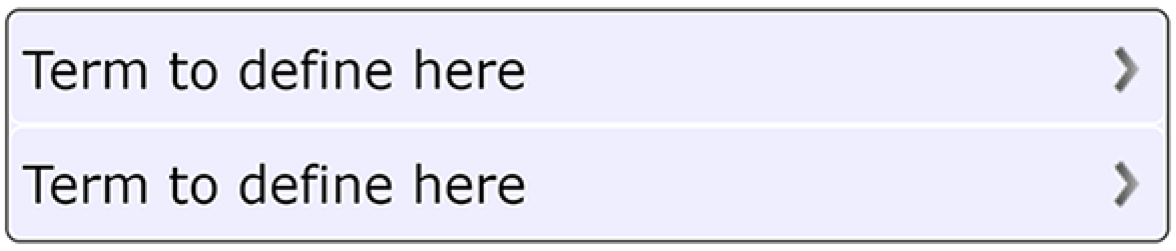
The HTML is fairly straightforward. We have a single div with a class name of *accordion* assigned to it. Within that div, we have a div with a class name of *term,* which will contain a word or short phrase to be defined or expounded upon. Immediately below that, a pair a <p> . . . </p> tags will contain the longer definition or explanatory text. You can have as many terms and paragraphs as you want within any single accordion. In our working example, we've started with two:

```
<!-- Start accordion -->
    <div class="accordion">
        <!-- The short text goes in the term div -->
        <div class="term">
            Term to define here
        </div><!-- End term -->
        <!-- The definition or longer text goes in the next paragraph -->
        <p>Definition here.</p>
        <!-- The short text goes in the term div -->
        <div class="term">
            Term to define here
        </div><!-- End term -->
        <!-- The definition or longer text goes in the next paragraph -->
        <p>Definition here.</p>
    </div><!-- End accordion -->
```

The CSS is nothing too fancy—mostly sizes, fonts, margins, and padding that you can style to taste. The *tinyArrow.png* image from the *icons* folder is being used as a background image. You can use any image you like, of course.

```
<style>
/* Container for all terms and definitions */
.accordion {
       background-color: #eef;
       width: 400px;
       margin: 1em auto;
       border: solid 1px #555;
       border-radius: 5px;
       font-family: Verdana, Tahoma, Sans-Serif;
}
/* term box contains the term to be defined */
.term {
       padding: 8px 48px 8px 4px;
       background: url(icons/tinyArrow.png) no-repeat 98% center;
       border: solid 1px white;
       border-radius: 5px;
       font-size:1.1em;
}
/* Paragraph contains the definition. */
.accordion p {
       margin: 0;
       padding: 0.5em;
       background-color: white;
       border-radius: 5px;
       display: none;
}
</style>
```

In the browser, you'll see the *Term to define here* dummy text.


Term to define here
Term to define here

But clicking won't have any effect, because we haven't written the JavaScript code for that. We'll do that next.

jQuery for the Accordion Control

In *accordion.htm*, we've already put in the standard code for getting the jQuery library and doing the *document.ready* business. So now we just need an event handler and some action that slides down the definition of whatever term the user clicked and hides all the others.

Identifying the Trigger

The element that triggers the action is the *term* div inside an *accordion* div. In jQuery, we'd express that as *$('.accordion .term)* with exactly one space before *.term* to get a proper descendant selector. The action to trigger the event will be a mouse click or tap, which is *.click* in jQuery.

So let's put in that part of the code first.

Here are the Steps

1. Open *accordion.htm* in your editor.
2. Place the cursor just below the *$(document).ready(function () {* line but above its closing *});*.
3. Type or copy/paste the following jQuery code:

```
          $('.accordion .term').click(function () {
               });
```

4. Save your progress

Expanding and Collapsing Accordions

So now we have a block of code that says, "When the user clicks inside a *term* div in the accordion div, *do something*." We want it to do two things:

- Slide up and hide any panel that's already showing in the accordion.
- Slide down the panel under the term the user clicked.

Let's look at each action to see how it's done.

Collapse Visible Accordion Panels

All of the longer explanatory texts in the accordion are in paragraphs (<p> . . . </p> tags). To refer to all of them in single selector, use *'.accordion p'*, which translates to "all text that's in <p> . . . </p> tags within the element that has *class="accordion"* in its opening tag."

The action we want to perform on those paragraphs is to slide them up out of view. In jQuery, that syntax is *.slideUp(optionalSpeed)*. It's the same syntax as *slideToggle* but with the word *Up* in place of *Toggle,* so it only slides up.

The *optionalSpeed* can be nothing at all, the word *'fast'* or *'slow'*, or a number in milliseconds, just as in the *.slideToggle* method. For our working example, let's say we want to slide them all up quickly. That line of code will be:

```
$('.accordion p').slideUp('fast');
```

So, that means, "Slide up all the paragraphs on the accordion div, using your fast speed."

Expand Clicked Accordion Panel

Next, we want to slideToggle the paragraph that's directly below the term div the user clicked.

Remember

The keyword *this* refers to the specific element they clicked. You can use *.next* to refer to the first element below that.

So the syntax to refer to the next paragraph directly below the item clicked is *$(this).next('p')* (the next paragraph under the thing they clicked). We'll *slideToggle* that one, again using *'fast'* as our working example. So that line of code looks like this:

```
$(this).next('p').slideToggle('fast');
```

That's all we need. We can add them to the page now, with a couple of comments for future reference.

Here are the Steps

1. Open *accordion.htm* in your editor.

2. Put the cursor just below the *$('.accordion .term').click(function () {* line but above the *});* for that block.

3. Type or copy/paste the following comments and code:

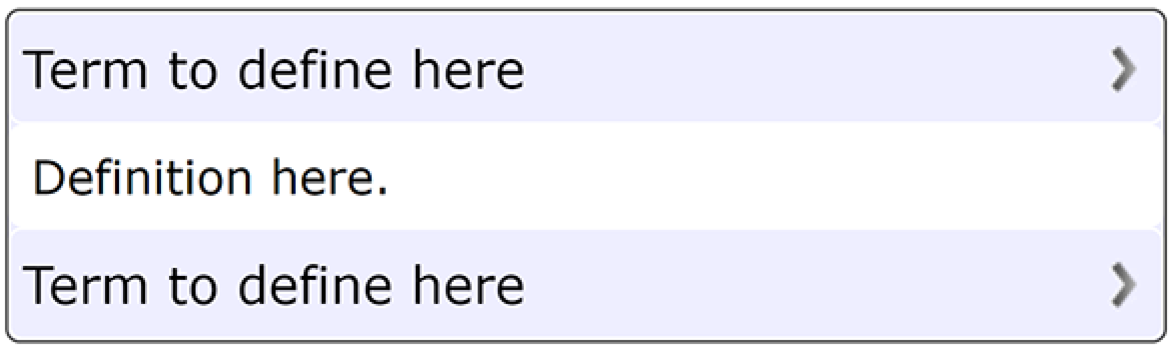
```
//Hide all dropdown panels
   $('.accordion p').slideUp('fast');

   //Toggle the paragraph below the term that was clicked
   $(this).next('p').slideToggle('fast');
```

4. Save the page.

5. Open or refresh *accordion.html* in a browser. You should still see the *Term to define here* text in an accordion control.

6. If you click one of the terms, the paragraph panel will slide down.

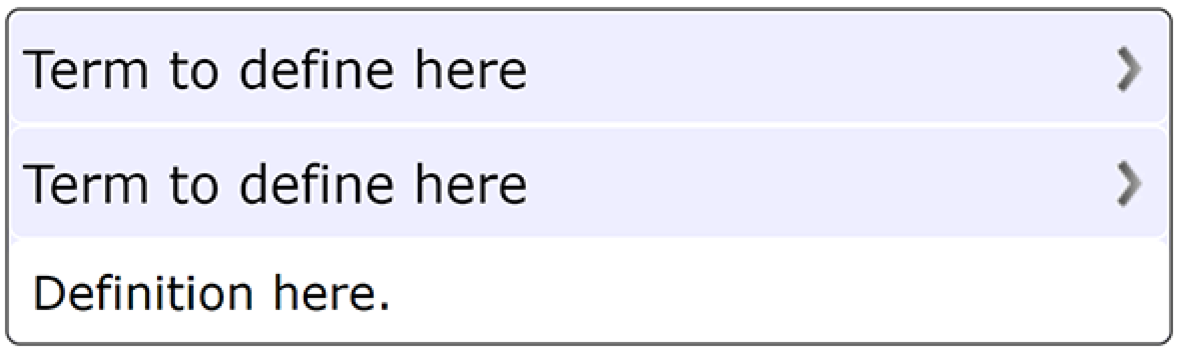


Term to define here

Definition here.

Term to define here

7. Click the other one, and the first panel will close, and the new term will open. So at any one time, only one item is expanded, which is what we wanted in our accordion control.



Term to define here

Term to define here

Definition here.

If you play around with it for a minute, you might notice one small problem. If you click the same term twice, the definition will slide away, then come right back. The accordion is doing exactly what the code says it should do: when the user clicks something, it hides *all* panels and toggles the one you just tapped. Unfortunately, because the one you clicked was visible, when you click it a second time it gets hidden by the *hide all* action, and then the toggle makes it slide back into view. This is not a great experience for the user.

To solve this problem, we're going to need our jQuery code to make a decision. We'll show you how to do that next.

Review

In today's lesson, you learned about a couple of cool custom controls that you can create in jQuery: collapsible panels and accordions. We call them *controls* because they allow users to control what they see on their screen, in much the same way the controls in a car (accelerator, steering wheel, brake) allow you to drive the car. jQuery is the perfect language for both controls because it has some useful tools built right in for controlling the action.

- **Creating a Collapsible Panel:** The collapsible panel control lets you show a question or text on the page. If users are interested in learning more, they can click that text, and a longer panel will slide down to reveal the answer to detailed text. Clicking a second time allows the lengthy answer or detailed text back up out of view. You learned about jQuery's *.toggleClass*, which allows it to switch between CSS style classes. You used that to switch between two different background images in the collapsible panel, an image that points up and an image that points down.

- **Creating an Accordion Control:** You also created an accordion control, which is kind of like a stack of collapsible panels, except that only one larger panel is visible at a time. You found that the *.slideToggle* allows you to control the speed at which an element on the page slides down into view or slides back up out of view.

- **Making Decisions in jQuery:** We also talked about decision-making in jQuery. Since jQuery is JavaScript, the syntax for *if . . . then* decisions is exactly the same as in JavaScript. You used an *if* decision in jQuery to make a decision as to whether or not to hide all panels in an accordion div based on whether or not the item the user just clicked was already showing. That decision helped you put the finishing touch on an accordion panel to make it behave in an intuitive way.

You've made an epic journey into and through the world of JavaScript. If you made it this far, you need to give yourself a big pat on the back. Together HTML, CSS, and JavaScript are the most important tools for digital creativity in the 21st century—not just for websites, but for many forms of development. The demand for people skilled in all three languages will surely continue to grow for years, perhaps decades, to come. So congratulations on a job well done, we hope you enjoyed taking this course taking this course!

Instructor Farewell

We started off with a description of what JavaScript is and where it came from. Simply stated, it is a programming language that all Web browsers can execute and is in fact used in the vast majority of websites today, just like HTML and CSS.

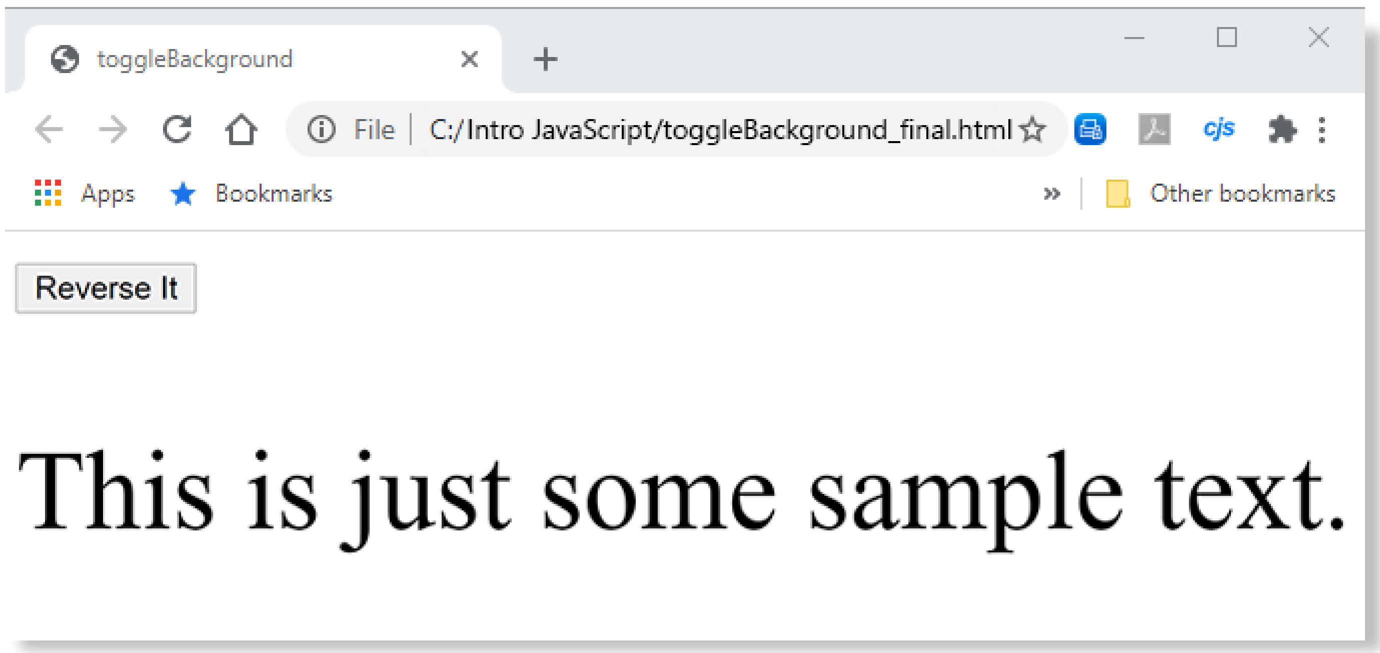
- HTML plays the role of markup language, defining what each element on a page *is*.
- CSS is the style sheet language, defining how each element *looks*.
- JavaScript is the scripting language that defines how things *act*.

At first you may have been surprised to see jQuery make an appearance in the last three lessons. After all, this is a course in JavaScript. However, as you now know, jQuery *is* JavaScript. More specifically, it's a JavaScript library designed to allow you to *write less and do more* and has become so wildly successful, it's almost as synonymous with Web development as HTML, CSS, and JavaScript are. Because jQuery is so widely used, it would have been remiss *not* to cover it in this course!

Lesson 12 Assignment

Toggle the CSS Class

For today's challenge, you're going to start with a relatively simple webpage that contains a button, some text, and one CSS style class.



This is just some sample text.

Download

Below is the link to the ZIP which contains following files you'll need:

- **toggleBackground:** This file contains *toggleBackground.html* and *toggleBackground.txt* which contains the HTML and CSS, and most of the JavaScript code.

Download L12-Assignment.zip

1.8 KB (Gigabytes) ZIP

You can retype or copy and paste if you'd like to run through the code for the page, or you can simply download and save the files to your *Intro to JavaScript* folder.

Right now, when you click the button, nothing happens. That's because there's no code in the page to handle the click.

Your challenge is to add jQuery code that does the following:

- Toggles the CSS class named *reverse* (which you can see in the page CSS code) whenever the user clicks the button.

Tips

Here are some tips

- **You don't have to type all of the code from scratch.** You can copy and paste in any relevant code that you already have.
- **There's only one button on the page.** Since there is only one button, your code doesn't have to be written to handle any specific button.
- **Use the body element.** You want to change the CSS class for all the content in the body, so you can use the body element for your toggle.
- **Use one CSS class.** You can use just one CSS class name with *.toggleClass* to toggle that one class on and off.

Good luck, and don't worry if it seems pretty daunting. When you need help, you can open the steps to get the complete step-by-step instructions.

Here are the Steps

Don't be discouraged if that assignment proved to be too challenging. It's easy to forget the many extra rules and odd syntax that goes along with jQuery.

Lesson 12 Resources for Further Learning

.toggleClass() (https://api.jquery.com/toggleClass/)

https://api.jquery.com/toggleClass/

This is the official jQuery documentation for toggleClass, right from the jQuery website.

---

jQuery toggleClass() Method (https://www.w3schools.com/jquery/html_toggleclass.asp)

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

Here is more documentation for the toggleClass method.

.slideToggle() (https://api.jquery.com/slideToggle/)

https://api.jquery.com/slideToggle/

Click this link for the official jQuery documentation on the slideToggle method.

---

jQuery slideToggle() Method (https://www.w3schools.com/jquery/eff_slidetoggle.asp)

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

Here you'll find the W3Schools reference page for slideToggle.

---

jQuery—Effects (https://www.tutorialspoint.com/jquery/jquery-effects.htm)

https://www.tutorialspoint.com/jquery/jquery-effects.htm

This page offers a nice overview of jQuery effects for quick review. Links in the left column provide access to more jQuery information.

How to Tell if an Element Is Visible With jQuery (https://electrictoolbox.com/jquery-element-is-visible/)

https://electrictoolbox.com/jquery-element-is-visible/

Click this link for a good discussion of using jQuery to test an element's visibility using the :visible selector.

---

jQuery if / else statements (http://css-plus.com/2011/07/jquery-if-else-statements/)

http://css-plus.com/2011/07/jquery-if-else-statements/

Here's where you'll find some good info on jQuery *if . . . else* decisions.