Short Exercises

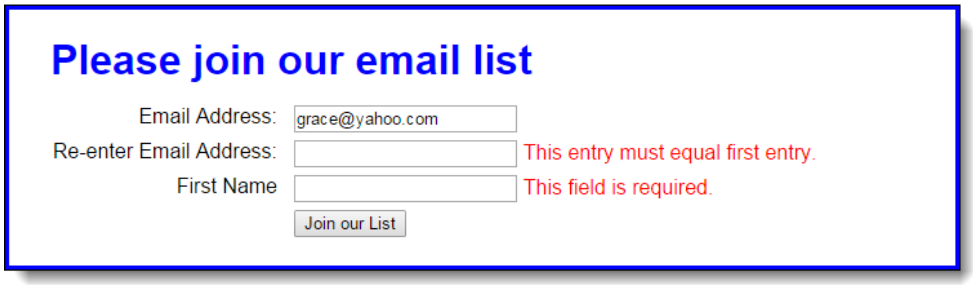
Each of these exercises is designed to test just one or two JavaScript or jQuery skills, and each is designed so it can be done in from 5 to 30 minutes. At the start of each exercise, you'll see an estimated time for the exercise. A list of the short exercises follows.

Guidelines for doing the short exercises

- For all the short exercises, you will start with the HTML, CSS, and JavaScript or jQuery for the application. Then, you will modify the JavaScript or jQuery as directed by the exercise.
- Unless an exercise specifies that you need to modify the HTML or CSS, you won't have to do that.
- Do the exercise steps in sequence. That way, you will work from the most important tasks to the least important.
- If you are doing an exercise in class with a time limit set by your instructor, do as much as you can in the time limit.
- Feel free to copy and paste code from the book applications or exercises that you've already done.
- Use your book as a guide to coding.

Short 1-1 Test an application and find an error

In this exercise, you'll run the Email List application and discover that it stops running due to a coding error. Then, you'll use Chrome to identify the statement that caused the error. Estimated time: 5 to 10 minutes.


Please join our email list

Email Address:	grace@yahoo.com	
Re-enter Email Address:		This entry must equal first entry.
First Name		This field is required.

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch01\email_list\`

2. Start the application, enter an email address in the first text box, and click the Join our List button. Note the error messages that are displayed to the right of the other two text boxes.

3. Enter a different email address in the second text box, and enter your name in the third text box. Then, click the Join our List button to see what error messages are displayed.

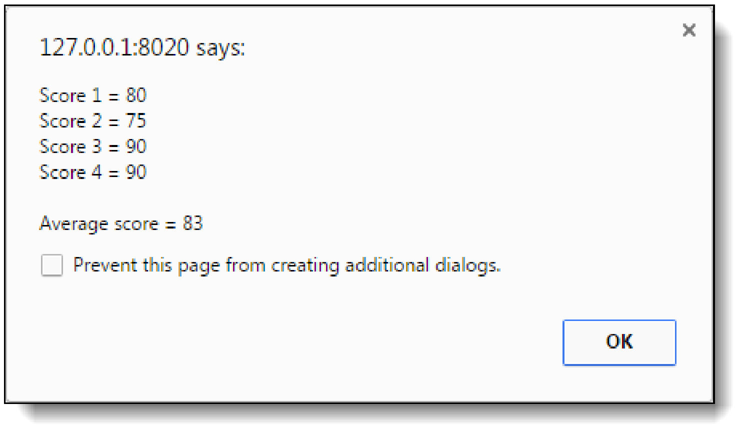
4. Enter valid data in all three text boxes and click the Join our List button. Then, note that nothing happens.

5. Use Chrome's developer tools to locate the statement that caused the error.

6. Use your editor or IDE to fix the error (change *submitt* to *submit*). Then, save your files, and test the application again with valid data. This time, a new page should be displayed when you click the Join our List button.

Short 2-1 Modify the Test Scores application

In this exercise, you'll modify the Test Scores application so it provides for four test scores and displays the results in a dialog box like the one that follows. Estimated time: 10 to 20 minutes.



127.0.0.1:8020 says:

```
Score 1 = 80
Score 2 = 75
Score 3 = 90
Score 4 = 90

Average score = 83
```

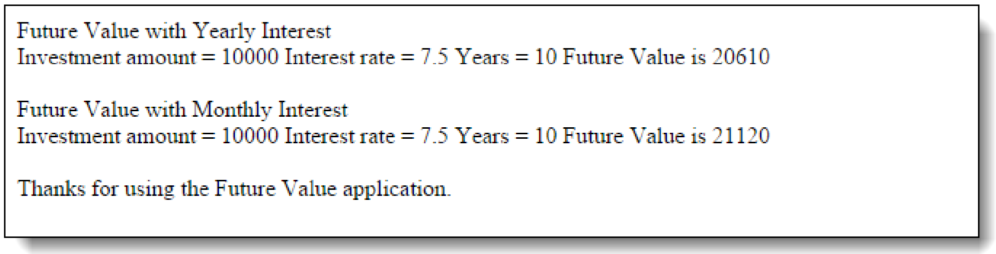
1. Open this file:

   `exercises_short\ch02\scores.html`

2. Run the application, and note that it works like the one in the book and that it writes the results in the browser page. Then, review the JavaScript code, and note that it's slightly different than the code in the book, although it gets the same results.
3. Modify the application so it provides for a fourth test score.
4. Modify the application so it displays the results in a dialog box like the one above, as well as in the browser page after the dialog box is closed.

Short 3-1 Enhance the Future Value application

In this exercise, you'll make a quick enhancement that shows not only the future value when interest is compounded yearly, but also when it's compounded monthly. As a result, the display in the browser should look like this:



```
Future Value with Yearly Interest
Investment amount = 10000 Interest rate = 7.5 Years = 10 Future Value is 20610

Future Value with Monthly Interest
Investment amount = 10000 Interest rate = 7.5 Years = 10 Future Value is 21120

Thanks for using the Future Value application.
```

Estimated time: 15 to 20 minutes.

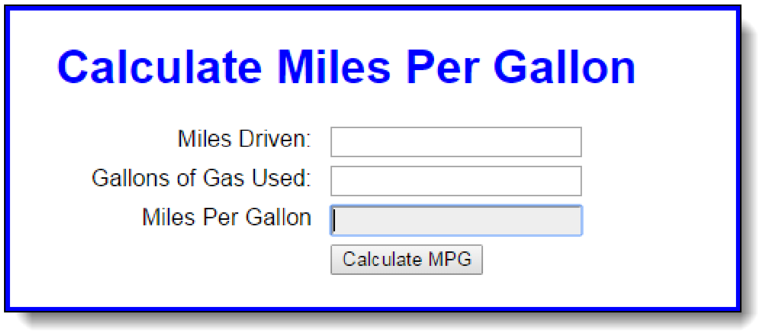
1. Open this file:

   `exercises_short\ch03\future_value.html`

2. Run the application, review its code, note that it is just like the one in the book, and note that the interest is calculated each year.
3. Add the code for calculating the future value when interest is calculated each month. Then, add the code for the displaying the results, as shown above.

Short 4-1 Enhance the MPG application

In this exercise, you'll make a couple of quick enhancement to the Miles Per Gallon application, like clearing the two entries if the user double-clicks in the Miles Per Gallon text box. Estimated time: 10 to 15 minutes.



1. Open the mpg.html file in this folder:

   `exercises_short\ch04\mpg\`

2. Run the application to see that it works just like the one in the book. Then, in the JavaScript in the HTML file, note that there's a clearEntries() function that isn't used.

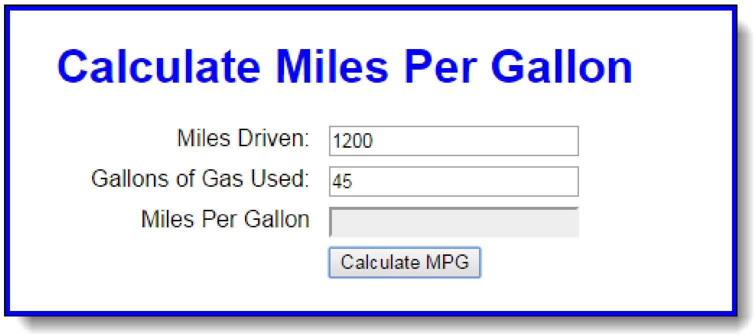
3. Enhance the application so the entries are cleared when the user double-clicks in the Miles Per Gallon text box. (Incidentally, this won't work if the text box is disabled.)

4. Enhance the application so the Miles Driven text box is cleared when it receives the focus. Then, do the same for the Gallons of Gas Used text box.

5. Enhance the application so the calculation is done when the focus leaves the Gallons of Gas Used text box. To do the calculation, you just need to run the processEntries() function when that event occurs.

Short 5-1 Debug the MPG application

This exercise gives you a chance to use your debugging skills to find the cause of an error. Estimated time: 5 to 15 minutes.



Calculate Miles Per Gallon

Miles Driven:	1200
Gallons of Gas Used:	45
Miles Per Gallon	

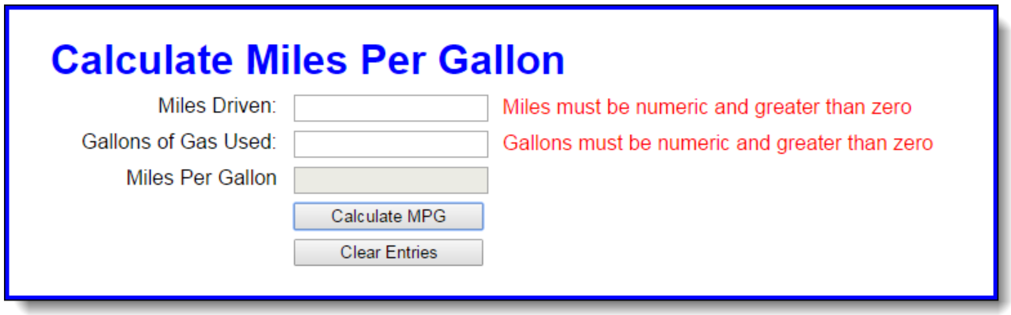
1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch05\mpg\`

2. Start the application, enter the values shown above, and click on the Calculate MPG button. Nothing happens.
3. Use Chrome's developer tools to find the cause or causes of the problem. Then, use your IDE or text editor to fix the code.
4. Start the application again, enter the values shown above, and click on the Calculate MPG button. This time, 26 will be displayed in the Miles Per Gallon text box.
5. Fix the application so the miles per gallon will be displayed with one decimal place.

Short 6-1 Upgrade the MPG application

In this exercise, you'll upgrade a version of the MPG application so the error messages are displayed in span elements to the right of the text boxes. Estimated time: 10 to 15 minutes.



Miles must be numeric and greater than zero

Gallons must be numeric and greater than zero

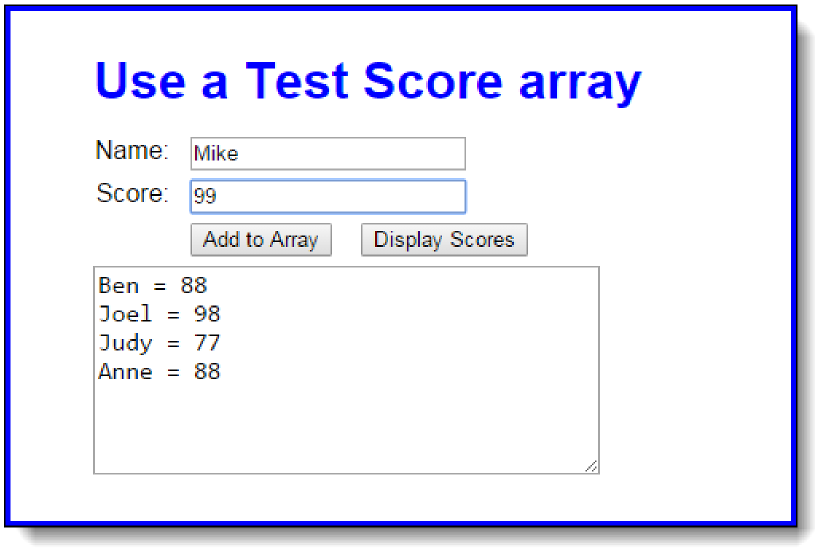
1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch06\mpg\`

   Then, run the application and click the Calculate MPG button to see that an error message is displayed in an alert dialog box for each of the two input fields.
2. In the HTML file, add a span element after the input element for the first two text boxes.
3. Enhance the data validation so it displays the error messages in the span elements instead of in alert dialog boxes.
4. If the user entries are valid, clear the contents of the span elements.
5. If the user clicks the Clear Entries button, set the contents of each span element to an asterisk.

Short 6-2 Display Test Score arrays

In this exercise, you start with the declarations for two arrays, a names array and a scores array. Then, you'll add an event handler that displays the names and scores in the text area of the interface. Estimated time: 20 to 30 minutes.



```
Ben = 88
Joel = 98
Judy = 77
Anne = 88
```

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch06\test_scores\`

   If you run the application, you can enter a name and score and then click the Add to Array button to add names and scores to the arrays, but you can't use the other button to display the scores.

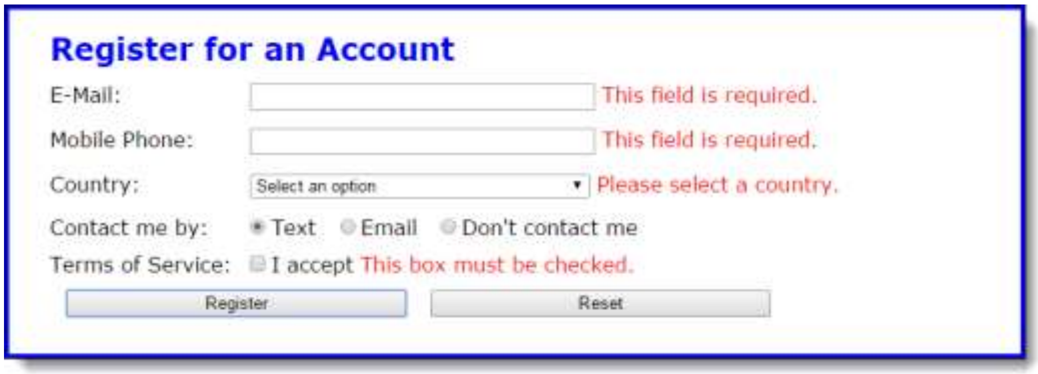
2. Look at the JavaScript code to see that it starts with the declarations for two arrays that contain four elements each. The first is an array of names. The second is an array of scores. Note too that the JavaScript code includes the $ function, an onload event handler, and the addScore() function for adding a name and score to the arrays.

3. Add an event handler for the Display Scores button that displays the names and scores in the arrays, as shown above. You will also need to use the onload event handler to attach this function to the click event of the Display Scores button. After you test this with the starting array values that are shown above, add a name and score to the arrays and test the display again.

4. Modify the event handler for the Add to Array button so the data in the text area is cleared after a name and score are added to the arrays.

Short 7-1 Use the event object to prevent form submission

In this exercise, you'll change the way the Register application submits the form. Instead of using a regular button and calling the submit() method in code if the validation passes, you'll use a submit button and cancel its default action if the validation fails. Estimated time: 5 to 10 minutes.



Register for an Account

E-Mail:		This field is required.
Mobile Phone:		This field is required.
Country:	Select an option	Please select a country.
Contact me by:	[✓ Text] Email Don't contact me	
Terms of Service:	I accept	This box must be checked.

1. Open the HTML and JavaScript files in this folder:

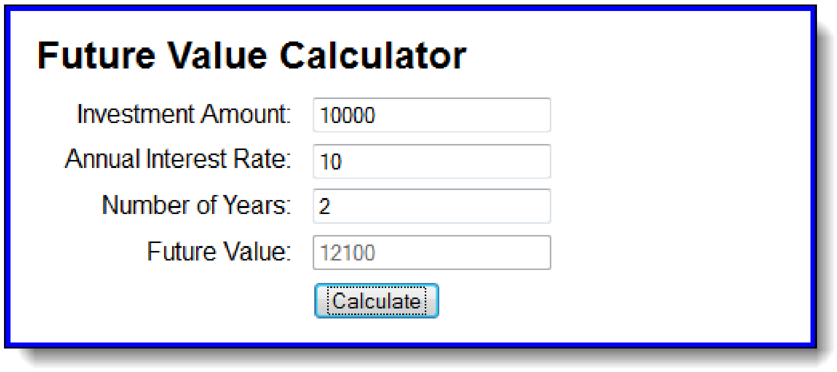
   `exercises_short\ch07\register\`

2. In the index.html file, find the Register input element and change its type attribute from "button" to "submit". This means the form will be submitted when the button is clicked. Then, run the application and click the Register button to see that you're taken to the confirmation page even though the data is invalid.

3. In the main JavaScript file (register.js), change the processEntries() function so it accepts the event object as a parameter. Then, change the function so it prevents the default action of the submit button if the validation fails. For this, you can assume you're using the Chrome browser so you don't have to provide for cross-browser compatibility.

Short 8-1 Redo a Future Value application with jQuery

In this exercise, you'll rewrite the code for a Future Value application that uses JavaScript with jQuery. That will show you how jQuery can simplify an application, but also that JavaScript is still needed with jQuery applications. Estimated time: 10 to 15 minutes.



Future Value Calculator

Investment Amount:	10000
Annual Interest Rate:	10
Number of Years:	2
Future Value:	12100

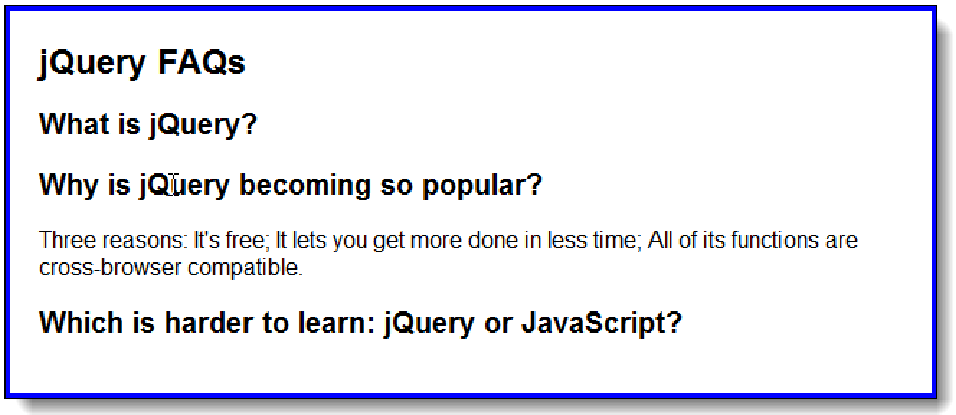
1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch08\future_value\`

   Then, run the application to refresh your memory about how it works.
2. Add a script element to the HTML file that gives you access to the jQuery library.
3. Rewrite the code in the JavaScript file so it uses as much jQuery as possible. That includes the code for the ready() and click() event methods.
4. When you have the application working right, add a JavaScript statement that moves the focus to the Investment Amount text box each time the Calculate button is clicked.

Short 8-2 Create a FAQs Rollover application

In this exercise, you'll write the jQuery code for an application that displays the answer to a question in a list of questions when the user hovers the mouse over that question. Estimated time: 10 to 15 minutes.



jQuery FAQs

What is jQuery?

Why is jQuery becoming so popular?

Three reasons: It's free; It lets you get more done in less time; All of its functions are cross-browser compatible.

Which is harder to learn: jQuery or JavaScript?

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch08\faqs_rollover\`

2. Review the HTML to see that it includes an unordered list with list items that consist of an h2 element and a <p> element. The h2 element contains a question, and the <p> element, which is hidden, contains the answer to the question.

3. Add jQuery code that runs when the user moves the mouse pointer into or out of a question in the list. When the mouse pointer moves into the question, the answer should be displayed. When the mouse pointer moves out of the question, the answer should be hidden.

Short 9-1 Add effects to the Image Gallery application

In this exercise, you'll use effects to change the way the images are displayed and hidden. Estimated time: 5 to 10 minutes.

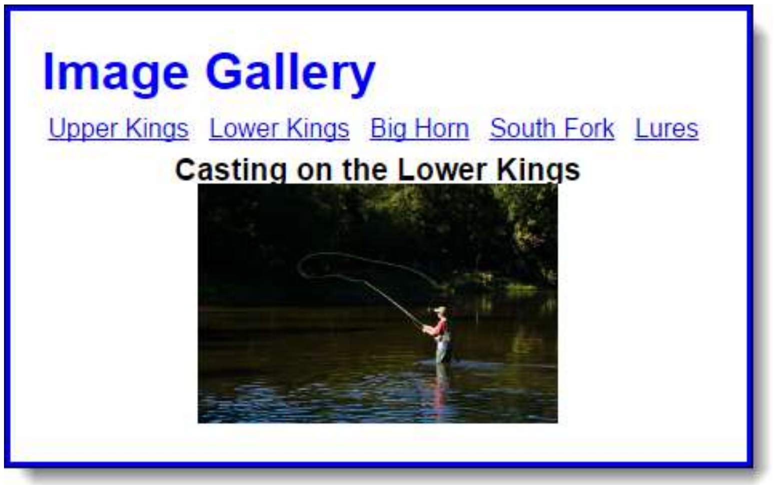


Image Gallery
Upper Kings Lower Kings Big Horn South Fork Lures
Casting on the Lower Kings

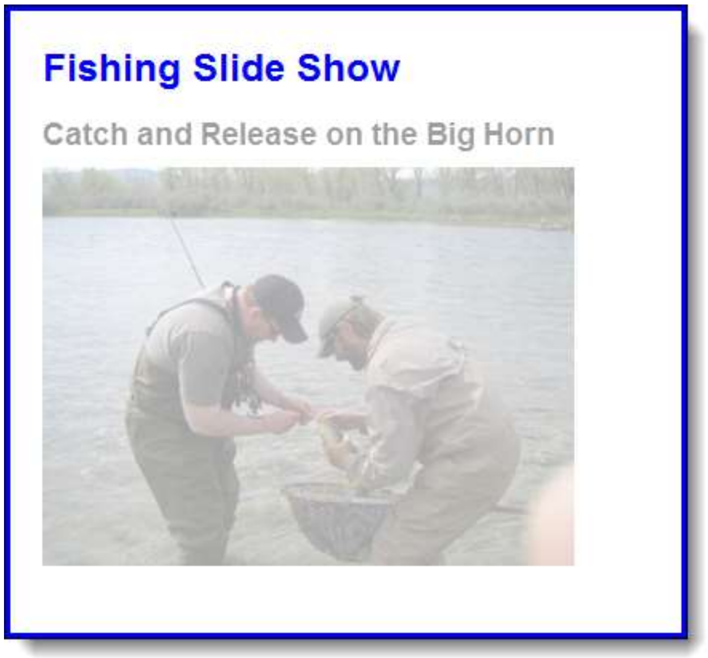
1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch09\image_gallery`

   Then, run the application to refresh your memory about how it works.
2. Modify the jQuery so that when a link is clicked, the current image is hidden using a sliding motion that takes two seconds.
3. Modify the jQuery so that after the caption and URL are set for the new image, that image is displayed using a sliding motion that takes two seconds.

Short 9-2 Debug a Slide Show application

This Slide Show application is like the one in the book, but it has a bug in it. Instead of displaying the slides in sequence, it displays every other slide. Your problem is to debug this application. Estimated time: 5 to 10 minutes if you can find the problem.



1. Open the HTML and JavaScript files in this folder:

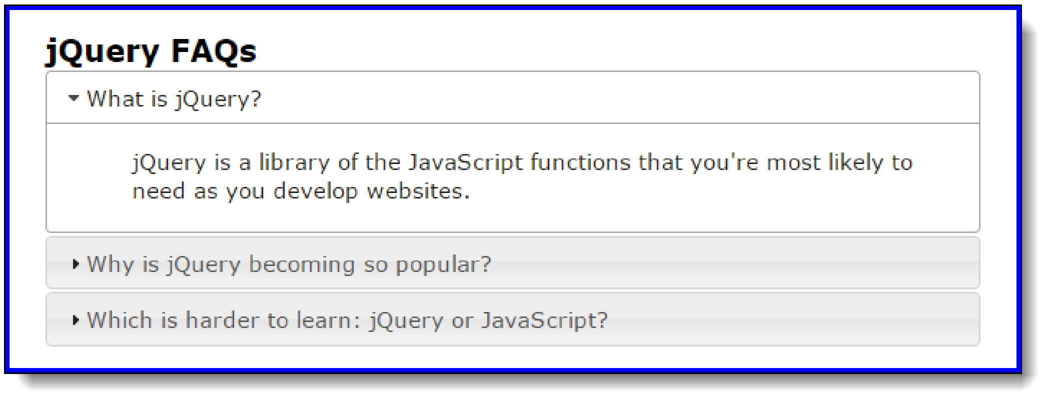
   `exercises_short\ch09\slide_show\`

2. Review the HTML to see how the images for the slides are structured. Then, run the application and note that it displays every other slide. Because there are only five slides in the HTML, this means that it displays slides 1, 3, 5, 2, 4, and so on.

3. Debug this application.

Short 11-1 Convert the FAQs app to an Accordion widget

This exercise has you change the FAQs application from JavaScript code to a jQuery UI Accordion widget. Estimated time: 15 to 20 minutes.


jQuery FAQs

▾ What is jQuery?

jQuery is a library of the JavaScript functions that you're most likely to need as you develop websites.

▸ Why is jQuery becoming so popular?

▸ Which is harder to learn: jQuery or JavaScript?

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch11\faqs\`

2. In the HTML file, note that the link and script tags that you need for jQuery UI have been coded for you. Then, rewrite the HTML so it's consistent with the HTML that the Accordion widget needs. Here's what the jQuery UI website says about the Accordion widget:

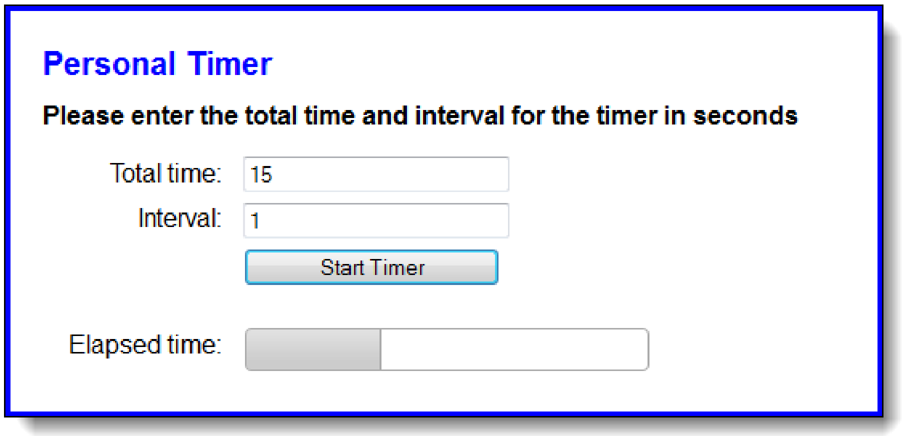
   "The underlying HTML markup is a series of headers (h3 tags) and content divs so the content is usable without JavaScript."

3. In the JavaScript file, comment out everything within the ready() function. Then, write the code for using the Accordion widget so all the panels can close and the panel height is based on the content height.

4. In the CSS file, comment out everything that's no longer needed for this application.

Short 11-2 Create a Progressbar widget that uses a timer

In this exercise, you'll modify an application that uses a timer to display the elapsed minutes and seconds in a text box so it displays the elapsed time in a Progressbar widget. Estimated time: 10 to 15 minutes to research the widget and another 10 to 15 minutes to implement the changes.


Personal Timer

Please enter the total time and interval for the timer in seconds

Total time: 15

Interval: 1

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch11\personal_timer\`

   Now, run this application to see that when you enter a total time and interval and click the Start Timer button, the elapsed minutes and seconds are displayed in the last text box and are updated each time the interval passes.

2. Go to the jQuery UI website at http://jqueryui.com and display the documentation for the Progressbar widget. Review the code for one or more of the examples, then review the API documentation.

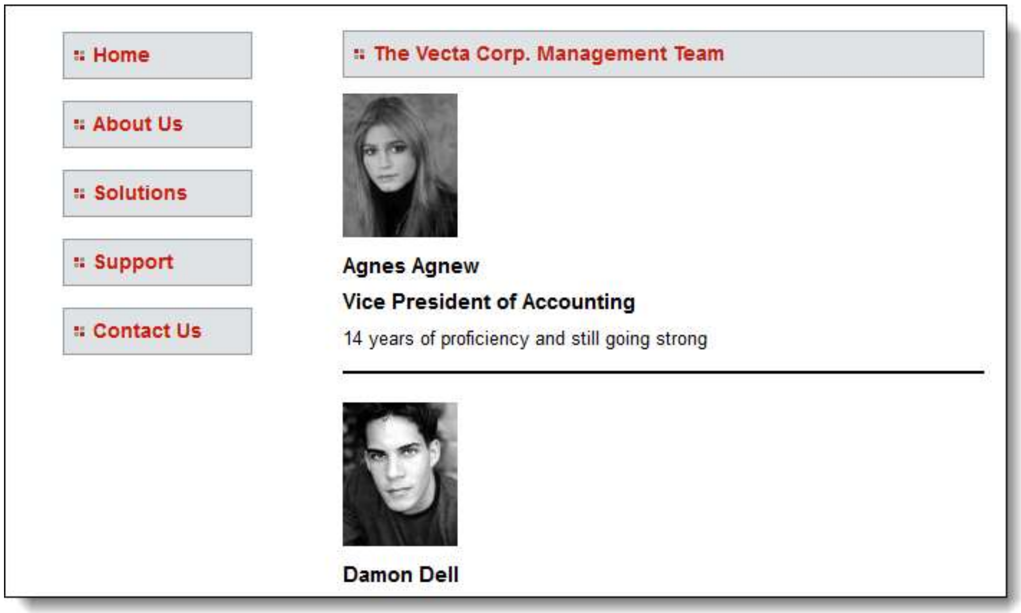
3. Modify the HTML for the application so it uses a Progressbar widget to display the elapsed time instead of a text box. Be sure to give the widget an id of "progressbar" so the CSS for the application works correctly.

4. Add a statement to the JavaScript file that activates the widget with a starting value of 0.

5. Modify the code that's executed if the two entries are valid so it uses the widget. When you do that, you can delete the code that calculates and displays the elapsed minutes and seconds since it's no longer needed. Then, you can add code that calculates and changes the value of the Progressbar widget.

Short 12-1 Convert a $.getJSON() method to $.ajax()

In this exercise, you will modify a JSON application that uses the $.getJSON() method so it uses the $.ajax() method instead. You'll also add a loading message that is displayed while the data is loading. Estimated time: 10 to 15 minutes.



Agnes Agnew
Vice President of Accounting
14 years of proficiency and still going strong
Damon Dell

1. Open the HTML file in this folder:

   `exercises_short\ch12\get_json\`

   Now, run this application to see how it works. Then, review the JavaScript in the last script element of the HTML file to see that it uses a $.getJSON() method to get the JSON data.

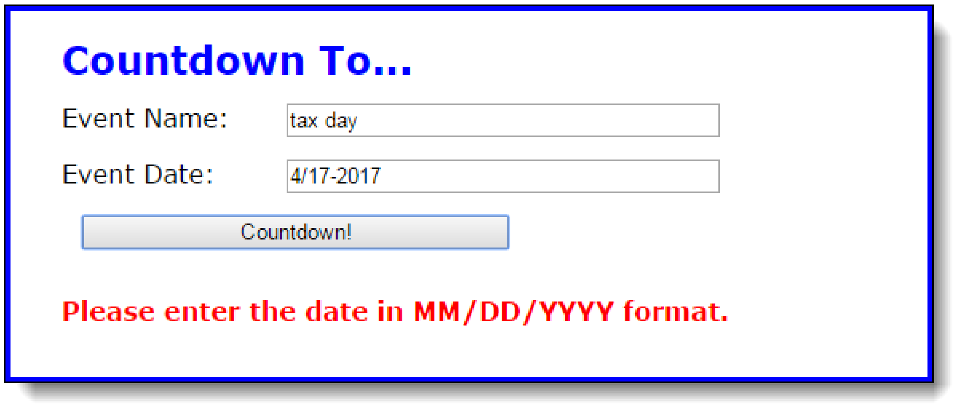
2. Comment out the $.getJSON() method. Then, code a $.ajax() method that gets the same results.

3. Add an option to the $.ajax() method that displays an alert message if the request fails. This message should include the error code and error message that's returned by the method. To test this, you can change the name of the file that's used so the file can't be found and then change it back after the test.

4. Add an option to the $.ajax() method that displays "Getting data..." in the div element that will receive the JSON data. Once the data is received, this message should be replaced by the JSON data. Because this happens so fast, you may not be able to see the message before it is replaced. In that case, you can test this change by changing the name of the file that's used by the $.ajax() method so the alert error message is displayed. Then, you'll see the "Getting data..." message behind the alert message box.

Short 13-1 Improve the validation of the Countdown application

This exercise has you make a modification to the way the Countdown application in chapter 13 validates the date entered by the user. When you're done, this application should only allow dates with two slashes. Estimated time: 10 to 15 minutes.


Countdown To...

Event Name: tax day

Event Date: 4/17-2017

Please enter the date in MM/DD/YYYY format.

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch13\countdown\`

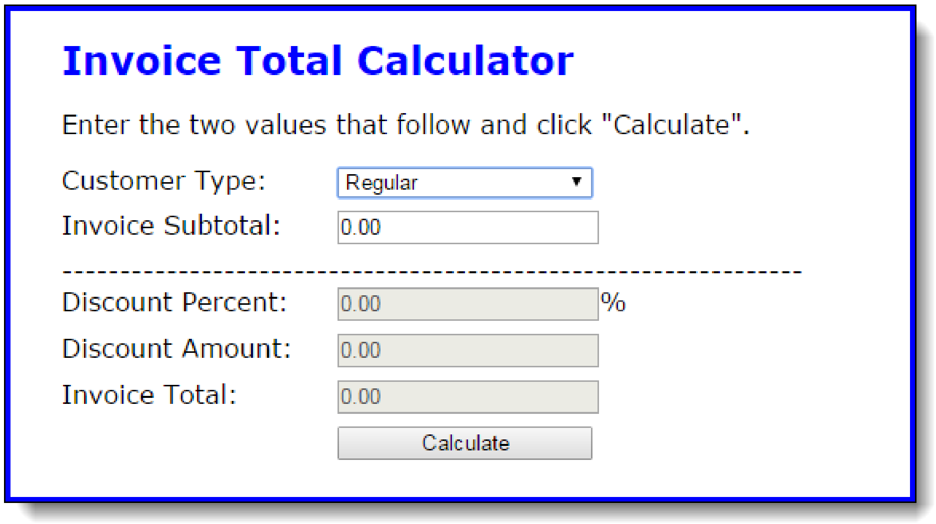
2. Start the application, enter a date that has only one slash (as above), and note that the application accepts the date.

3. In the JavaScript file, find the if statement that checks whether the date string has slashes. Note that it's only checking whether there is one or more slash. Then, modify the code so it checks to make sure that the date entry has two slashes. If not, display the same error message as shown above.

   To do this, you need to store the result of the indexOf method and use it to first check whether a first slash was found and then use it as the starting point for trying to find a second slash.

Short 14-1 Add a default subtotal value to the Invoice application

In this exercise, you'll modify the Invoice application so it provides a default value for the Invoice Subtotal field. Estimated time: 5 to 10 minutes.


Invoice Total Calculator

Enter the two values that follow and click "Calculate".

Customer Type:	Regular
Invoice Subtotal:	0.00
Discount Percent:	0.00 %
Discount Amount:	0.00
Invoice Total:	0.00

1. Open the HTML and JavaScript files in this folder:

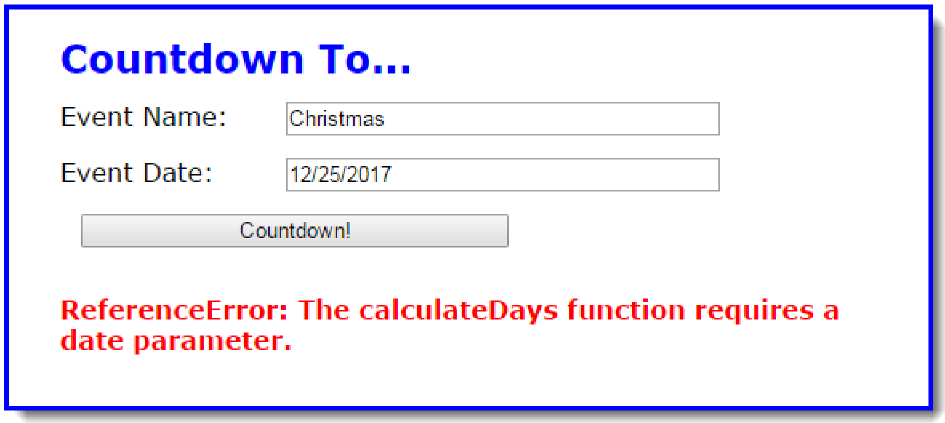
   `exercises_short\ch14\invoice\`

2. Start the application and click the Calculate button without entering anything in the Invoice Subtotal field. Note that it displays several NaN results.

3. In the JavaScript file, use the OR operator to provide a default value of 0 for the subtotal value. If you've done this right, the application should display zeros when you click the Calculate button without entering anything in the Invoice Subtotal field, as shown above.

Short 14-2 Add exception handling to the Countdown application

In this exercise, you'll add exception handling to the Countdown application to make sure an argument is passed to a function. When you're done, the Countdown application will display a custom error message if the argument isn't sent (see below). Estimated time: 20 to 30 minutes.



Event Name:	Christmas
Event Date:	12/25/2017

ReferenceError: The calculateDays function requires a date parameter.

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch14\countdown\`

2. In the main JavaScript file, comment out the call to the calculateDays() function. Then, code a new call that passes no arguments to the function. Now, run the application and see that nothing happens.

3. Open the Console panel and view the error message there - "Cannot read property 'getTime' of undefined". Note that this error message gives you information about the problem, but you'll still need to do some work to track down what's wrong.

4. In the JavaScript file, modify the calculateDays() function so it makes sure the value of the "date" parameter isn't undefined. If it isn't, the function should be done. Otherwise, this function should create and throw an error object with the custom message shown above.

5. Run the application again and then view the error message in the Console panel. This time, the error message explains exactly what is wrong.

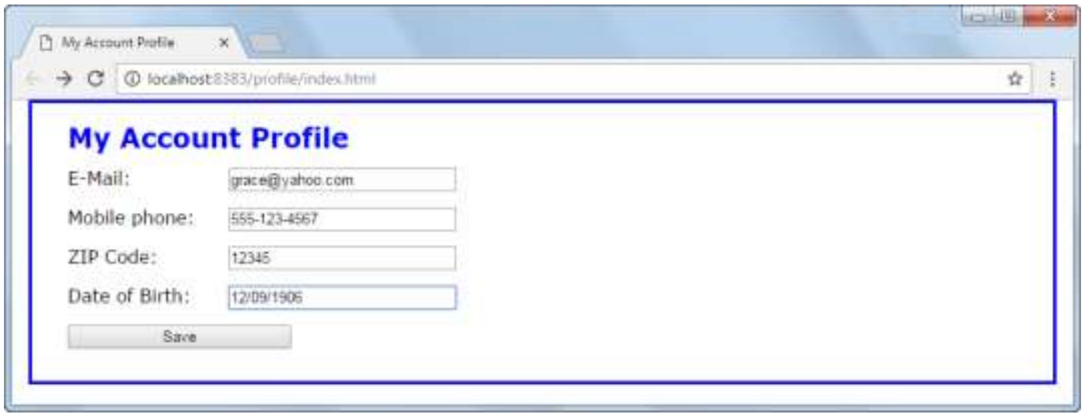
6. Put the call to the calculateDays() function and the if statements that follow it inside a try-catch statement. In the catch block of the statement, display the custom error message in the span tag whose id is "message". Run the application and view the error message in the page.

7. Fix the application so the call to the calculateDays() function passes the date as an argument like it originally did.

Short 15-1 View the query string of a URL

In this exercise, you'll adjust the Account Profile application so that it displays the data that is in the query string of the URL. Estimated time: 5 to 10 minutes.

Here is the enhanced application, with account profile information about to be saved:

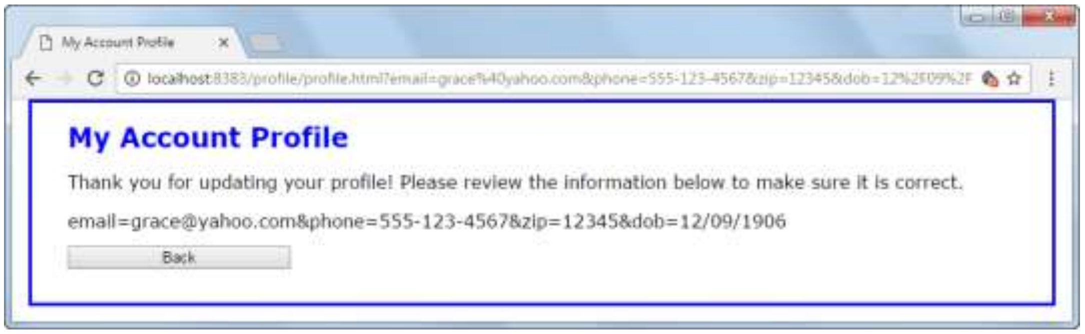

My Account Profile

localhost:8383/profile/index.html

My Account Profile

E-Mail:	grace@yahoo.com
Mobile phone:	555-123-4567
ZIP Code:	12345
Date of Birth:	12/09/1906

And here is the second page of the application after the profile is saved. Notice that the profile data is in the URL, and special characters like "@" and "/" are encoded:


My Account Profile

localhost:8383/profile/profile.html?email=grace%40yahoo.com&phone=555-123-4567&zip=12345&dob=12%2F09%2F

My Account Profile

Thank you for updating your profile! Please review the information below to make sure it is correct.

email=grace@yahoo.com&phone=555-123-4567&zip=12345&dob=12/09/1906

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch15\profile\`

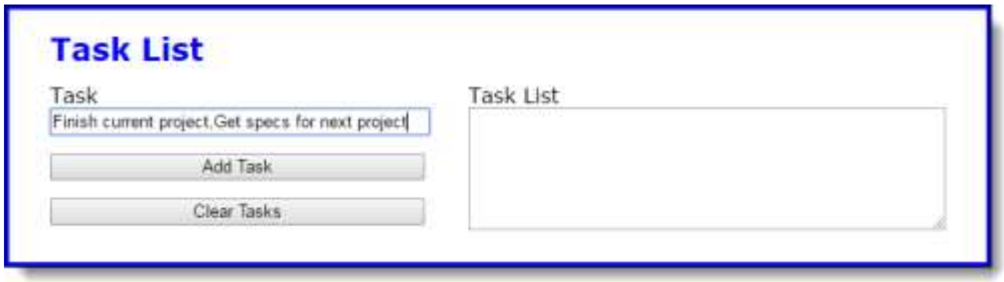
2. Review the index.html file, and notice that the elements on the page are coded within a form element with its method attribute set to "get" and its action attribute set to "profile.html". Then, review the JavaScript file, and notice that when the validation checks pass, the form is submitted. This is how the form data gets in the query string.

3. In the embedded JavaScript for the profile.html file, use the location object to retrieve the query string value. Remove the question mark from that value and then display the value in the span element whose id is "profile". Use the decodeURIComponent() function so the special characters display correctly.

Short 16-1 Allow multiple task entries in the Task List application

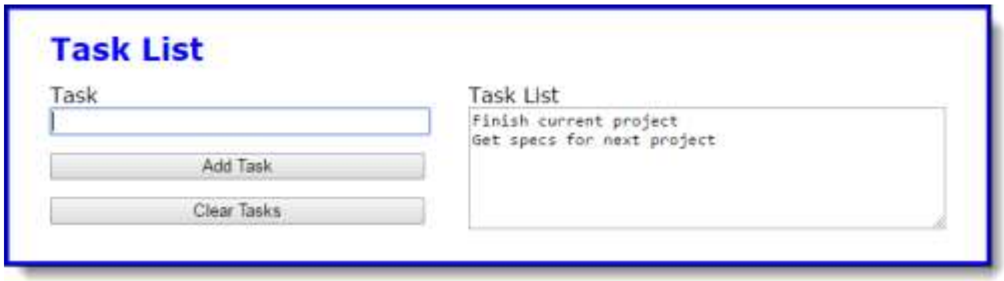
In this application, you'll make an enhancement that allows you to enter multiple tasks separated by commas in a single entry. Estimated time: 10 to 20 minutes.

Here is the enhanced application, with multiple tasks about to be entered:



Finish current project,Get specs for next project

And here is the application after the multiple tasks have been entered:



```
Finish current project
Get specs for next project
```

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch16\task_list\`

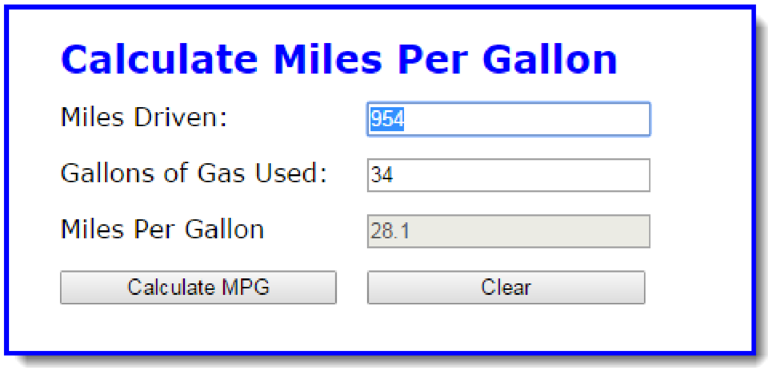
2. Run the application and add two tasks, separated by a comma. Note that the tasks are stored as one task, exactly as you entered it.

3. In the JavaScript file, find the handler for the click event of the Add Task button. Then, find the code that adds the task entered by the user to the tasks array. Comment out that code, and replace it with code that works for one or more tasks in an entry.

   To do that, you can use the split() method of the String object to convert the user's entry into an array. Then, you can use the concat() method of the tasks array to add the new tasks to the tasks array.

Short 17-1 Use the arguments property instead of named parameters

In this exercise, you'll update the calculate() method of the mpg object to use values in the arguments property rather than named parameter values. When you're done, the application should work the same as it did before. Estimated time: 5 to 10 minutes.



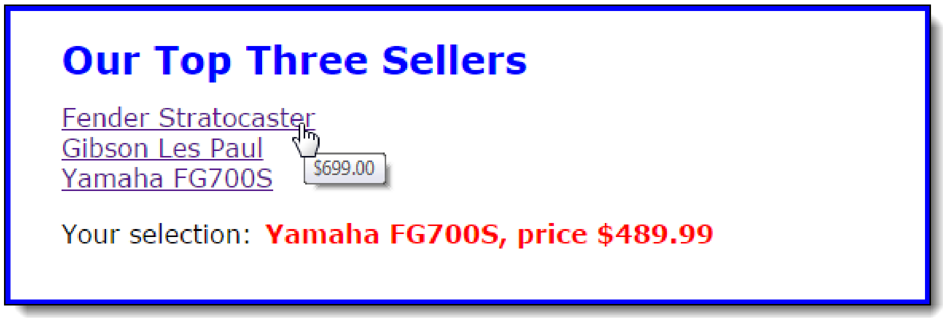
1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch17\mpg\`

   Run and test the application.
2. Review the library file (library.js). Note that this version of the mpg object contains a single calculate() method. This method accepts "miles" and "gallons" parameters, and it uses the values in those parameters to calculate and return the miles per gallon.
3. In the calculate() method, delete the named parameters in the parameters list of the function definition. Then, change the code so the method gets the values it needs from the arguments property of the method.
4. Run and test the application again to make sure it still works as expected.

Short 18-1 Fix the closure loop problem

In this exercise, you'll modify the Top Products application to fix a problem where the links all display the same data when clicked. Estimated time: 10 to 15 minutes.



Our Top Three Sellers

Fender Stratocaster
Gibson Les Paul
Yamaha FG700S

$699.00

Your selection: Yamaha FG700S, price $489.99

1. Open the HTML and JavaScript files in this folder:

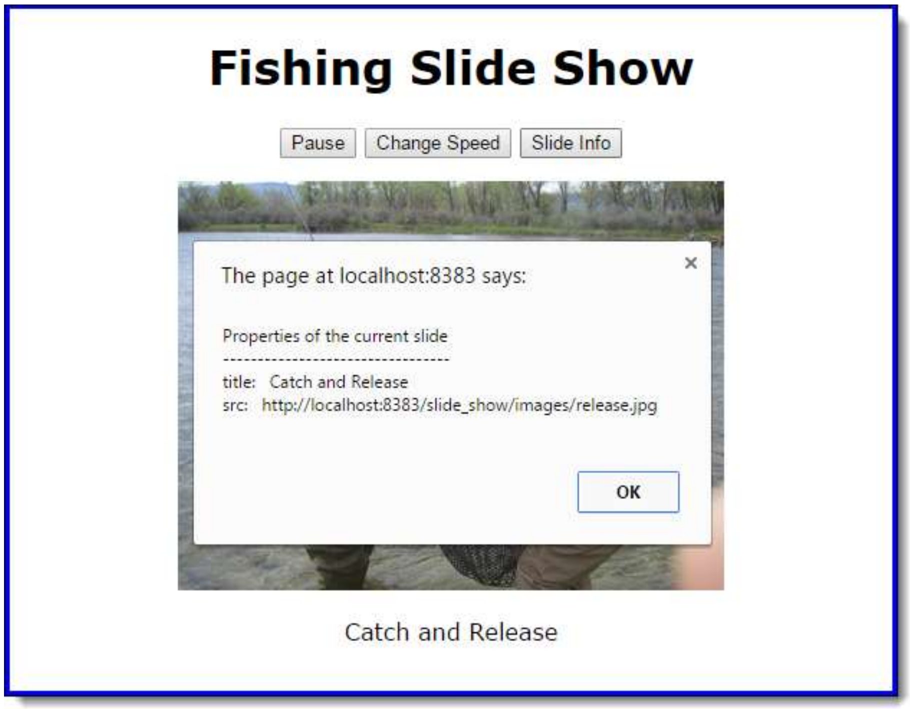
   `exercises_short\ch18\top_products\`

2. Run the application and note that the name for each product displays correctly. Then, hover over each link and note that the price for each product displays correctly, too. Finally, click on each link and note that no matter which link you click, the name and price of the last product is displayed.

3. In the main JavaScript file, find the code that attaches the event handler for the onclick events of the links. Then, change this code so it works correctly.

Short 18-2 Add properties to the Slide Show application

In this exercise, you'll change the Slide Show application so it displays information about the current slide when the Slide Info button is clicked. Estimated time: 15 to 20 minutes.



The page at localhost:8383 says:

```
Properties of the current slide
--------------------------------
title:  Catch and Release
src:   http://localhost:8383/slide_show/images/release.jpg
```

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch18\slide_show\`

2. Run the application and click on the Slide Info button. Note that a dialog box appears but it doesn't contain any information about the current slide.

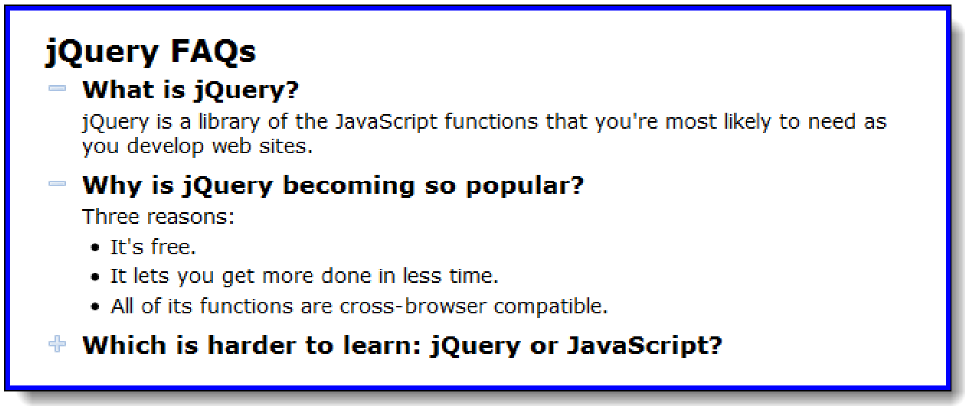
3. In the property descriptors in the slideshow library file (library_slide_show.js), add a read-only accessor property named currentTitle that returns the title of the current slide. Then, add a read-only accessor property named currentSrc that returns the src of the current slide.

   To add these properties, you'll work with the img object in the slideshow object's private state. This object has a cache property that's an array containing an Image object for each slide in the slide show, and a counter property that holds the index of the current slide.

4. In the main JavaScript file (slide_show.js), adjust the message code for the button whose id is "info" so it displays the title and src of the current slide. Then, test to make sure these changes work.

Short 18-3 Create a FAQs plugin

In this exercise, you'll create a plugin that hides or displays the answer to a frequently asked question in a list of questions when the user clicks on a question. Estimated time: 10 to 15 minutes.



jQuery FAQs

What is jQuery?

jQuery is a library of the JavaScript functions that you're most likely to need as you develop web sites.

Why is jQuery becoming so popular?

Three reasons:

- It's free.
- It lets you get more done in less time.
- All of its functions are cross-browser compatible.

Which is harder to learn: jQuery or JavaScript?

1. Open the HTML and JavaScript files in this folder:

   `exercises_short\ch18\faqs`

   Now, run this application to see how it works.

2. Review the code in the jquery.faqs.js file to see that it contains code for the standard plugin structure with the method name set to expandCollapse().

3. Copy the code in the ready event handler of the faqs.js file into the each() method of the plugin. Then, modify the selector for the each() method in this copied code so its function is executed for each h2 element that's a descendant of the element that the expandCollapse() method is executed on.

4. Return to the faqs.js file, and replace the code in the ready event handler with a statement that calls the expandCollapse() method for the "faqs" section.

5. Add a script element for the jquery.faqs.js file to the HTML before the script element for the faqs.js file. The application should now run just like it did before.