

Short Exercises

Each of these exercises is designed to test just one or two JavaScript skills, and each is designed so it can be done in from 5 to 45 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	2
Short 1-1 Test an application and find an error	3
Short 2-1 Modify the Test Scores application	4
Short 3-1 Enhance the Future Value application	5
Short 4-1 Enhance the MPG application	6
Short 5-1 Upgrade the Email List application	7
Short 5-2 Display Test Score arrays	8
Short 5-3 Modify the FAQs application	9
Short 6-1 Debug the Email List application	10
Short 7-1 Improve the validation of the Countdown app	11
Short 8-1 Add a default subtotal value to the Invoice app	12
Short 9-1 Allow multiple task entries in the Task Manager app	13
Short 10-1 Use the arguments property in the Task Manager	14
Short 11-1 Add a method to the native String object	15
Short 12-1 Add exception handling to the Task Manager app	16
Short 13-1 Use the event object to prevent form submission	17
Short 14-1 Fix the closure loop problem	18
Short 15-1 Add properties to the Slide Show application	19
Short 16-1 Display a JSON string	20
Short 17-1 Convert the FAQs app to an Accordion widget	21

Guidelines for doing the short exercises

- For all the short exercises, you will start with the HTML, CSS, and JavaScript for the application. Then, you will modify the JavaScript 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: 5-10 minutes.

Please join our email list

Email Address:

Re-enter Email Address: This entry must equal first entry.

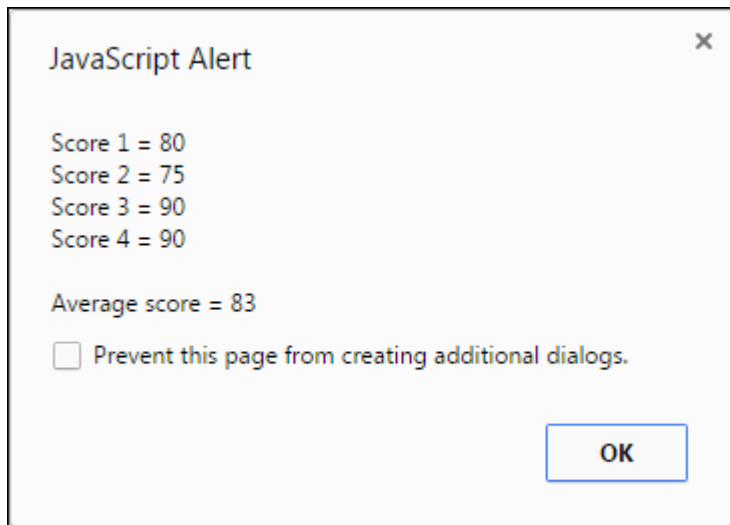
First Name This field is required.

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: 10-20 minutes.



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 application, 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

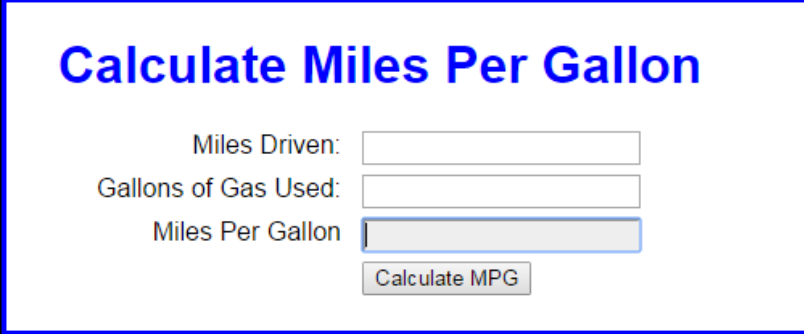
Thanks for using the Future Value application.
```

Estimated time: 15-20 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 application, 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: 10-15 minutes.



The screenshot shows a web application titled "Calculate Miles Per Gallon" in blue text. Below the title, there are three input fields: "Miles Driven:", "Gallons of Gas Used:", and "Miles Per Gallon:". Each field is followed by a text input box. The "Miles Per Gallon" input box is currently selected, indicated by a blue border. Below the input fields is a button labeled "Calculate MPG".

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 Drive 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 Upgrade the Email List application

In this exercise, you'll upgrade the code in the Email List application of chapter 4 so the span elements that display the error messages don't require id attributes.

Estimated time: 10 to 15 minutes.

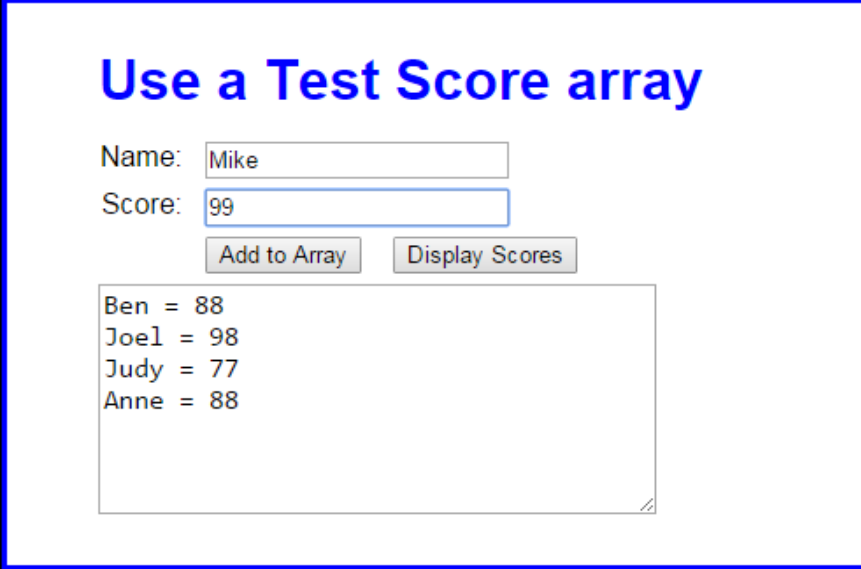


The screenshot shows a web form titled "Please join our email list" in blue text. Below the title are three input fields: "Email Address:", "Re-enter Email Address:", and "First Name:". Each field is followed by a red error message: "This field is required." Below the input fields is a button labeled "Join our List".

1. Open the HTML and JavaScript files in this folder:
`exercises_extra\ch05\email_list\`
Then, run the application to see that the validation works as shown above.
2. In the HTML file, delete the id attributes for the span elements.
3. In the JavaScript file, use the properties of the Node interface to display the error messages without using the id attributes.

Short 5-2 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: 20-30 minutes.



Use a Test Score array

Name:

Score:

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

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

`exercises_short\ch05\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 5-3 Modify the FAQs application

This exercise has you make a minor modification to the FAQs application in chapter 6. When you're done, this application should work the same as before, except that only one answer can be displayed at a time. In other words, when the user clicks on a heading to display the answer, the other answers must be hidden.

This takes just a few lines of code, but you have to think it through. Estimated time: 10 minutes to think it through and 5 minutes to add the code.

JavaScript FAQs

+ **What is JavaScript?**

– **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?**

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

`exercises_short\ch05\faqs\`

Then, run the application to refresh your memory about how it works.

2. Enhance the code in the JavaScript file so only one answer can be displayed at a time.

Short 6-1 Debug the Email List 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.

Please join our email list

Email Address:

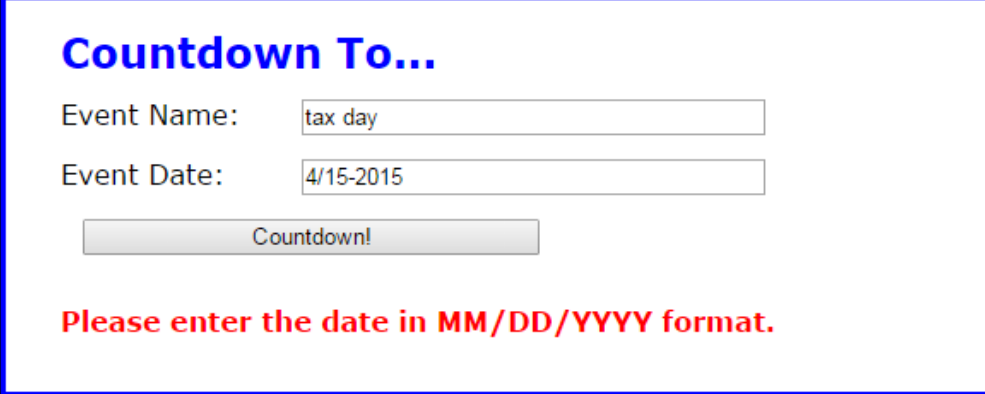
Re-enter Email Address:

First Name

1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch06\email_list\`
2. Start the application and click on the Join our List button with making any entries. Nothing happens.
3. Enter a valid email address in the first text box, and click on the Join our List button again. Nothing happens.
4. Use Chrome's developer tools to find the cause or causes of the problem. This might include the use of the Elements panel to see whether any changes have been made to the DOM.

Short 7-1 Improve the validation of the Countdown app

This exercise has you make a modification to the way the Countdown application in chapter 7 validates the date entered by the user. When you're done, this application should only allow dates with two slashes. Estimated time: 10-15 minutes.

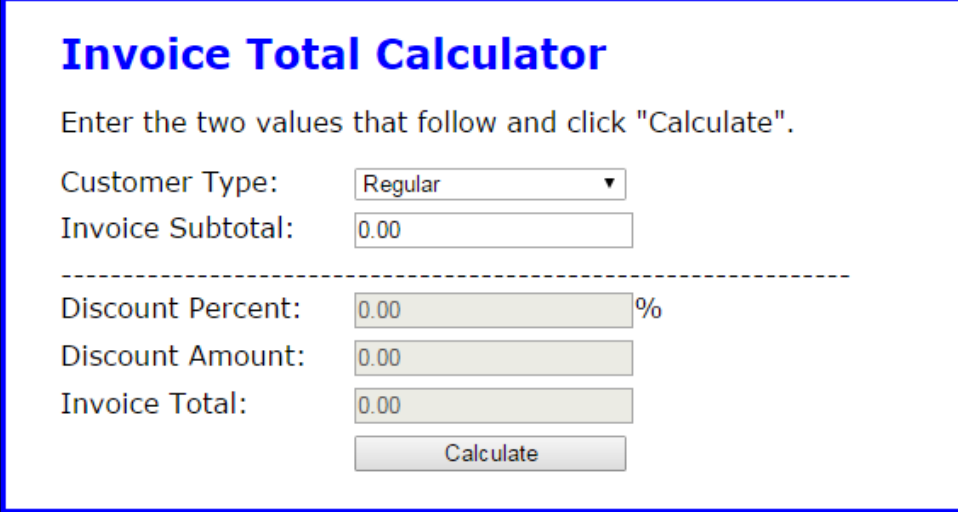


The screenshot shows a web form titled "Countdown To...". It has two input fields: "Event Name:" with the value "tax day" and "Event Date:" with the value "4/15-2015". Below the fields is a button labeled "Countdown!". At the bottom of the form, a red error message reads: "Please enter the date in MM/DD/YYYY format."

1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch07\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 8-1 Add a default subtotal value to the Invoice app

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



Invoice Total Calculator

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

Customer Type:

Invoice Subtotal:

Discount Percent: %

Discount Amount:

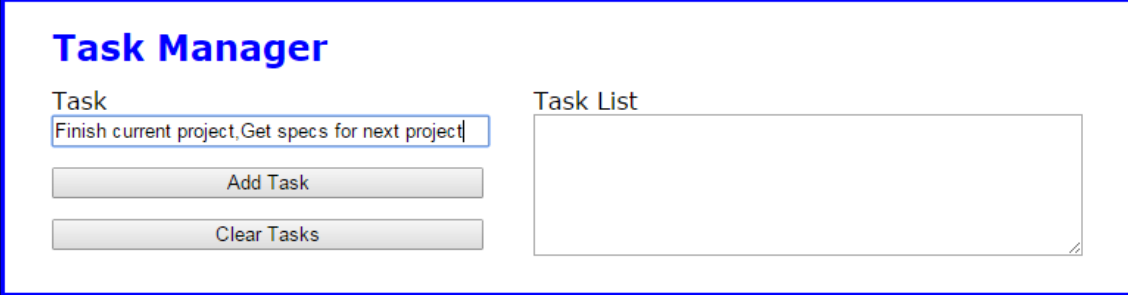
Invoice Total:

1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch08\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 0.0 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 9-1 Allow multiple task entries in the Task Manager app

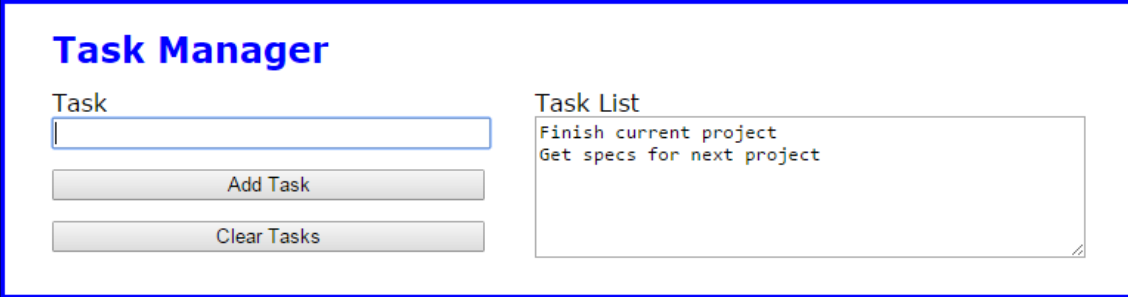
In this application, you'll make an enhancement that allows you to enter multiple tasks separated by commas in a single entry. Estimated time: 10-20 minutes.

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



The screenshot shows a web application titled "Task Manager". On the left, there is a "Task" input field containing the text "Finish current project, Get specs for next project". Below the input field are two buttons: "Add Task" and "Clear Tasks". On the right, there is a "Task List" area, which is currently empty.

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



The screenshot shows the same "Task Manager" application. The "Task" input field is now empty. The "Task List" area on the right now contains two lines of text: "Finish current project" and "Get specs for next project", each on a new line.

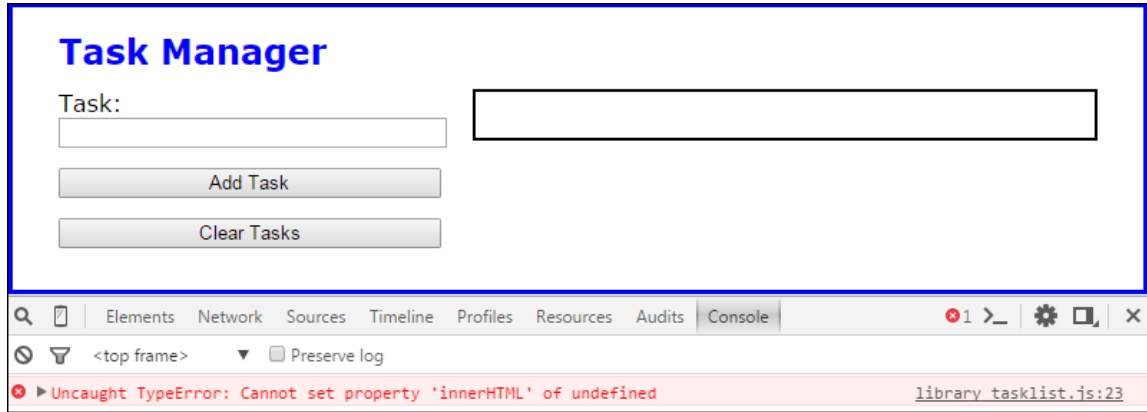
1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch09\task_manager\`
2. Run the application and add two tasks, separated by a comma. Note that the tasks are stored as one task, exactly as you entered it.
3. In the JavaScript file, find the code in the `addToTaskList` function that adds the task entered by the user to the tasks array. Comment out that code, and replace it with code that works for one task in an entry and also for more than one task 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, if there is more than one task in the entry, you can use the `concat` method of the tasks array to add the new tasks to the tasks array.

Short 10-1 Use the arguments property in the Task Manager

In this exercise, you'll use the arguments property of one of the functions in a JavaScript library to make sure it receives the correct number of arguments.

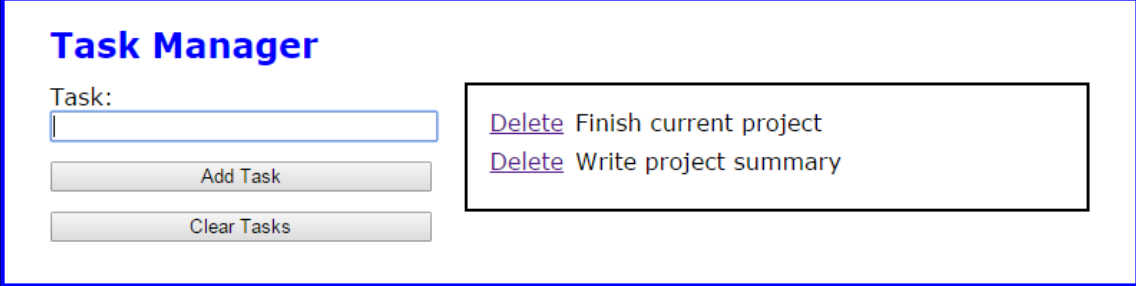
Estimated time: 10-15 minutes.



1. Open the HTML and JavaScript files in this folder:
exercises_short\ch10\task_manager
Run the application and, if necessary, add a task or two.
2. In the main JavaScript file (task_list.js), find the code in the displayTaskList function that calls the displaySortedTaskList function and passes it three arguments. Comment out this code and replace it with a call to the displaySortedTaskList function that passes only one argument, the tasks array.
3. Test the application and note that the tasks no longer display. Then, open the Console panel and view the error message (shown above). Click on the link to see what statement caused the error.
4. In the task list library file (library_tasklist.js), add code to the displaySortedTaskList function that checks to make sure three arguments have been passed to it before it runs the rest of the code. If three arguments aren't passed, display an alert dialog box with this error message:
The displaySortedTaskList function requires 3 parameters.

Short 11-1 Add a method to the native String object

In this exercise, you'll add a new method to the native String object that checks to see if its value is an empty string. Then, you'll use the new method in the Task Manager application. When you're done, the application should work the same as it did before. Estimated time: 5-10 minutes.



The screenshot shows a web application titled "Task Manager" in blue text. Below the title, there is a label "Task:" followed by a text input field. Under the input field are two buttons: "Add Task" and "Clear Tasks". To the right of the input field, there is a rectangular box containing two lines of text, each starting with a purple "Delete" link followed by a task description: "Delete Finish current project" and "Delete Write project summary".

1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch11\task_manager\`
2. In the library for native objects (`library_native_object.js`), code a new method called `isBlank` on the String object's prototype. The new method should check whether the string's value is an empty string and return `true` if it is, `false` if it isn't.
3. In the storage and task library files (`library_storage.js` and `library_task.js`), find every place that the code checks whether a string's value is an empty string and rewrite it to use the new `isBlank` method.

Short 12-1 Add exception handling to the Task Manager app

In this exercise, you'll add exception handling to the Task Manager app. It will use the arguments property to make sure the correct number of arguments are passed to a function. When you're done, the Task Manager application will display a custom error message if the correct number of arguments isn't sent (see below). Estimated time: 20-30 minutes.



The screenshot shows a web application titled "Task Manager" in blue text. Below the title, there is a label "Task:" followed by a text input field. Underneath the input field are two buttons: "Add Task" and "Clear Tasks". To the right of the input field, a red-bordered box contains the message: "The displaySortedTaskList function of the tasklist library requires three arguments." in red text.

1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch12\task_manager\`
In the HTML file, note that there is now a span tag with an id of "message" inside the div tag with an id of "tasks". In the CSS file, note that there is now a rule set for the id "message" that sets its color to red.
2. In the tasklist library file (library_tasklist.js), modify the displaySortedTaskList function so it checks the number of arguments that are passed to it. If three arguments are passed, the function should be done. Otherwise, this function should create and throw an error object with the custom message shown above.
3. In the main JavaScript file, comment out the call to the displaySortedTaskList function. Then, code a new call that only passes the tasks array to the function. Now, run the application and view the custom error message in the Console panel.
4. Put the call to the displaySortedTaskList function 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".
5. Fix the application by undoing what you did in step 3, but keep the corrected call to the displaySortedTaskList function within the try block.

Short 13-1 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: 5-10 minutes.

Register for an Account

E-Mail: This field is required.

Mobile Phone: This field is required.

Country: 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\ch13\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 14-1 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: 10-15 minutes.



1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch14\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 15-1 Add properties to the Slide Show application

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



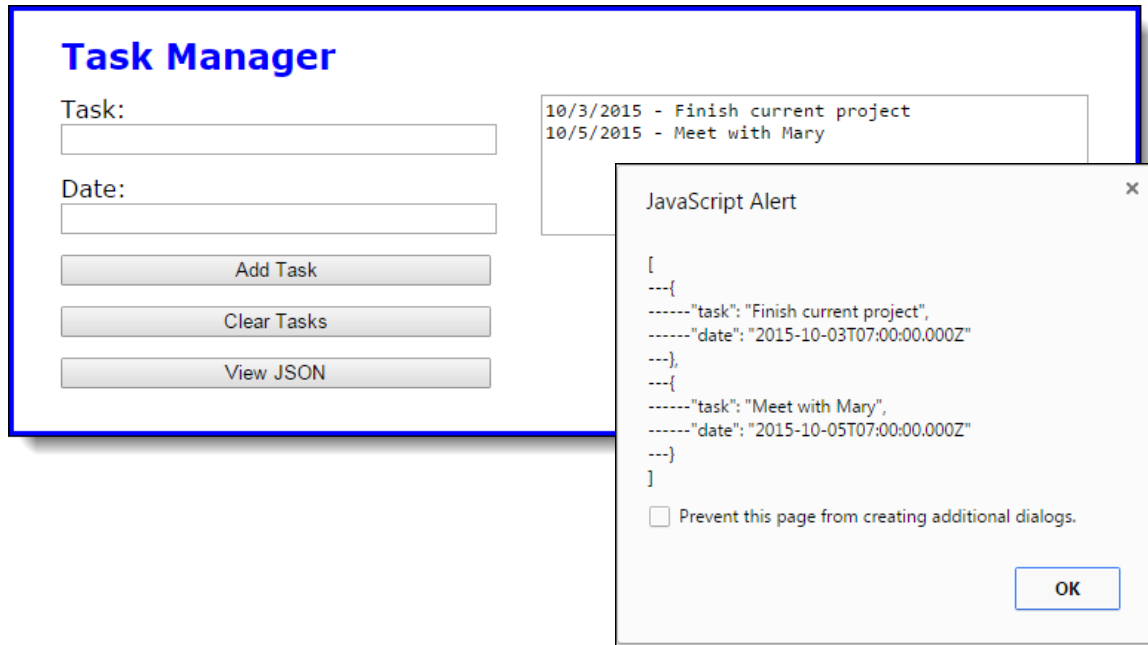
1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch15\slide_show\`
2. Run the application and click on the Slide Info button. Note that a dialog box appears but it doesn't contain any information about the current slide.
3. In the property descriptors in the slideshow library file (`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 16-1 Display a JSON string

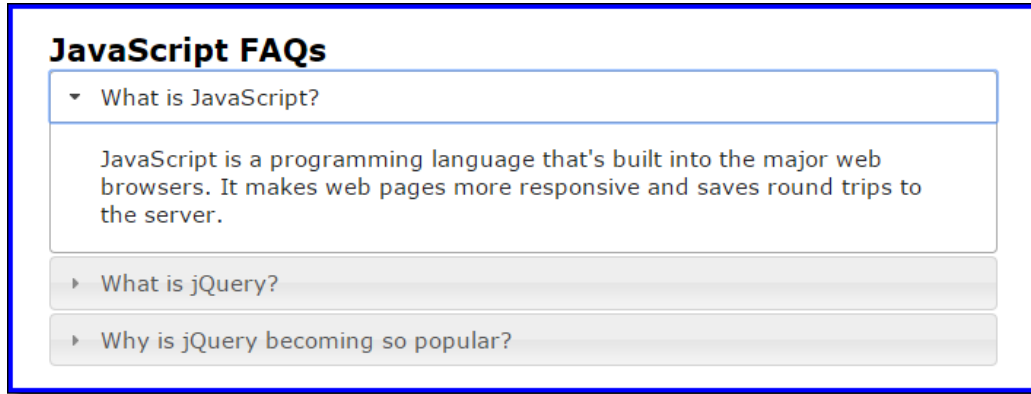
In this exercise, you'll use the `stringify` method of a JSON object to display a JSON string in an alert dialog box when the View JSON button is clicked. Estimated time: 10-15 minutes.



1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch16\task_manager\`
2. In the main JavaScript file, note that there's the start of an event handler named `viewJSON` that is attached to the View JSON button.
3. Within that event handler, use the `stringify` method and an `alert` statement to display the tasks array as a JSON string. Then, run the application, add a few tasks, and test the View JSON button to make sure this works.
4. Add the number 5 as a spacer argument to the `stringify` method and test the application to see what the JSON string looks like. Remember that `spacer` is the third parameter, after the object to be stringified and the optional `replacer` parameter. If you don't use a `replacer` function, you can use `null` for the second parameter.
5. Change the `spacer` argument from a number to the string `"---"`, and test the application to make sure the display of the JSON string is similar to the one shown above.

Short 17-1 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: 15-20 minutes.



1. Open the HTML and JavaScript files in this folder:
`exercises_short\ch17\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 but the "use strict" directive. 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. Make sure the DOM is ready before attaching the widget.