How to use Brackets to develop JavaScript applications

This document is a free download from Murach books. It is especially designed for people who are using *Murach's JavaScript and jQuery*, because that book shows how to use Aptana as the IDE for developing JavaScript applications. Unfortunately, Aptana is no longer supported and has become difficult to install.

Our solution is to install and use Brackets instead of Aptana. Brackets was developed by Adobe, is easy to install and use, has some excellent features, and runs on Windows, Mac OS, and Linux systems. Also, if you're using the latest edition of our HTML and CSS book, you should already know how to use Brackets. So in this document, you can just focus on the features that apply to JavaScript development.

If you aren't already familiar with Murach books, this document will show you how quickly and thoroughly you can learn from our books. For more information about our books, please go to www.murach.com.

Mike Murach & Associates, Inc.
1-800-221-5528 • (559) 440-9071 • Fax: (559) 440-0963
www.murach.com • murachbooks@murach.com

How to use Brackets to develop JavaScript applications

Brackets is a text editor that was created by Adobe. We like it because it's free, easy to use, has some excellent features, and runs on Windows, Mac OS, and Linux systems. In this document, you'll learn how to use Brackets to develop web pages with the emphasis on JavaScript applications. If you're already using Brackets for HTML and CSS development, you can read just the topics that focus on JavaScript development.

How to install Brackets

Figure 1 shows how to download and install Brackets. This is a simple procedure that takes just a few minutes.

On a Windows system

The first part of this figure shows how to download and install Brackets on a Windows system.

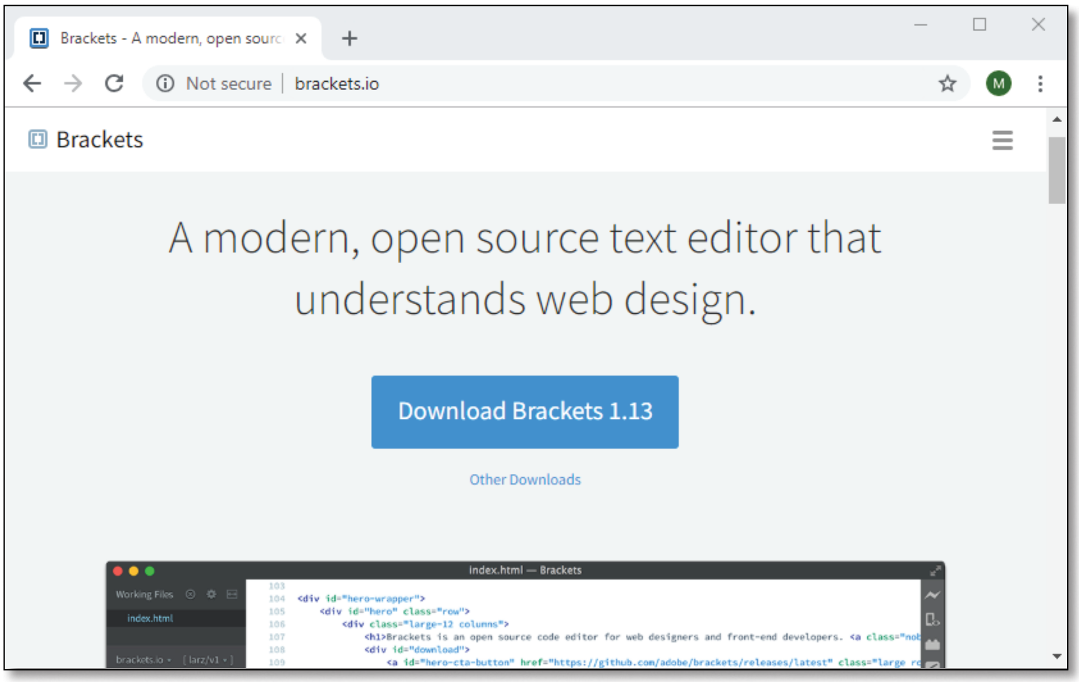
On a Mac OS X system

The second part of this figure shows how to download and install Brackets on a Mac OS X system.

The website address for downloading Brackets

http://brackets.io

The home page for Brackets



A modern, open source text editor that understands web design.

```
<div id="hero-wrapper">
    <div id="hero" class="row">
        <div class="large-12 columns">
            <h1>Brackets is an open source code editor for web designers and front-end developers. <a class="nob
            <div id="download">
                <a id="hero-cta-button" href="https://github.com/adobe/brackets/releases/latest" class="large rc
```

How to install Brackets on a Windows system

1. Go to the website address above.
2. Click on the Download Brackets button to download an installer file that will have a name like Brackets.Release.1.13.msi.
3. When the download finishes, run it. One way to do that is to use Windows Explorer or File Explorer to find the msi file and double-click on it.
4. As you step through the wizard in the usual way, you can accept all of the default settings.

How to install Brackets on a Mac OS X system

1. Go to the website address above.
2. Click on the Download Brackets button to download an installer file that will have a name like Brackets.Release.1.13.dmg.
3. When the download finishes, run it. One way to do that is to use Finder to find the dmg file, and double-click on it.
4. As you step through the wizard in the usual way, you can accept all of the default settings.

Description

- Brackets is a free text editor that runs on Windows, Mac, and Linux systems.

Figure 1 How to install Brackets

How to work with folders and files

The topics that follow show how to open and close folders and files and how to start new HTML, CSS, and JavaScript files.

How to open and close the folder for a website

To work with a web application in Brackets, you start by opening the folder that contains all of the subfolders and files for the application. To do that, you can use the procedure in figure 2. After you open a folder for a website, Brackets considers it to be the current *project*, and its subfolders are displayed in a *file tree* in the left pane beneath the name of the project folder. In this example, the current project is book_apps.

Normally, a project consists of the folders and files for a single website. To make it easier to work with the applications for our JavaScript and jQuery book, though, we recommend that you use Brackets to open the folder that contains all of them. This is illustrated by the dialog box in this figure, which is going to open the book_apps folder.

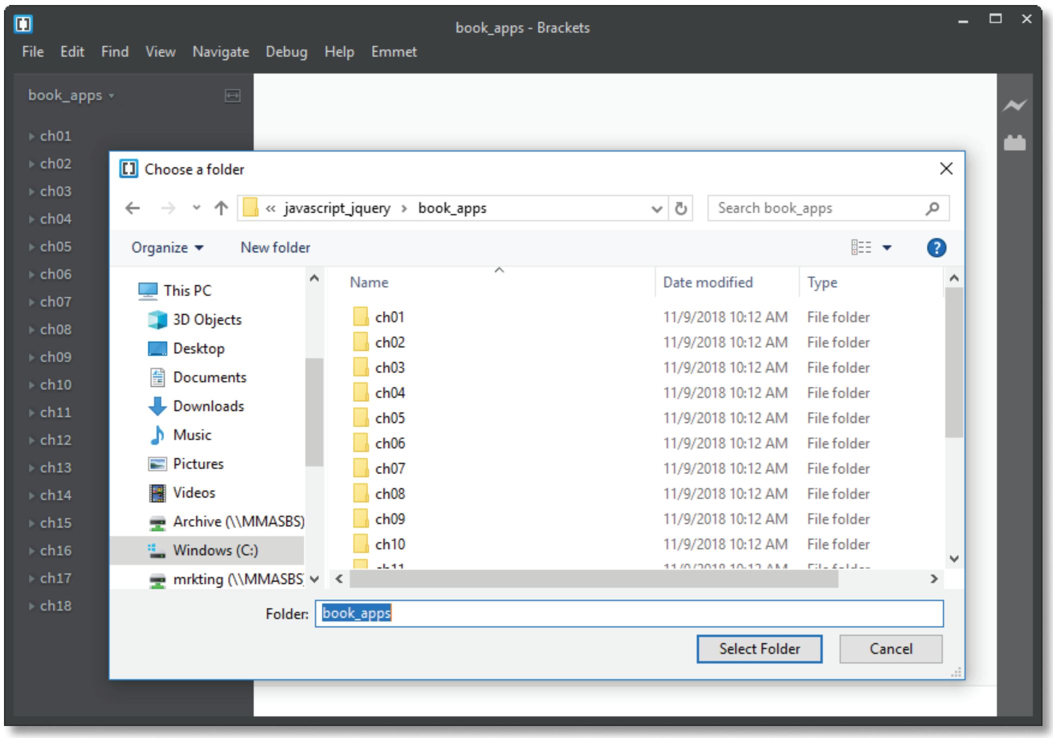
After you open that folder, all of the chapter folders are listed in the file tree at the left side of the window, as shown in this figure. Then, you'll be able to expand these folders to view and work with the subfolders and files that they contain.

When a file tree is displayed, you can click on the name of the current project at the top of the tree to display a list of recently opened folders. Then, you can select the next project that you want to work with from that list. Or, you can click on the Open Folder command at the top of the list to open another project. Either way, when another project is opened, it becomes the current project and the previous project is closed.

Please note, however, that when you start Brackets for the first time, the current project is called Getting Started. It introduces some of the features of Brackets. Beneath the project name, you will see a folder named screenshots, a file named index.html, and a file named main.css. After you learn how to preview an HTML file, you may want to preview the index.html page. You may also want to review the code in the HTML and CSS files. You can always open this folder by clicking on whatever the current project is in the file tree and selecting Getting Started.

Incidentally, notation like File→Open Folder means to drop-down the File menu and select the Open Folder. Similarly, View→Vertical Split means to select the Vertical Split command from the View menu. You will see this notation used throughout this document.

The Choose a Folder dialog box



Name	Date modified	Type
ch01	11/9/2018 10:12 AM	File folder
ch02	11/9/2018 10:12 AM	File folder
ch03	11/9/2018 10:12 AM	File folder
ch04	11/9/2018 10:12 AM	File folder
ch05	11/9/2018 10:12 AM	File folder
ch06	11/9/2018 10:12 AM	File folder
ch07	11/9/2018 10:12 AM	File folder
ch08	11/9/2018 10:12 AM	File folder
ch09	11/9/2018 10:12 AM	File folder
ch10	11/9/2018 10:12 AM	File folder

Folder: book_apps

How to open a project folder

- Use the File→Open Folder command. Or, click on the name of the current project folder in the file tree and then click on the Open Folder command.
- In the Choose a Folder dialog box that's displayed, choose the folder that you want to open and click the Select Folder button.

How to re-open a project folder after it has been closed

- Click the name of the current project folder to display a list of recently opened folders. Then, select a folder from that list.

Description

- When you open a folder, it is displayed in a *file tree* at the left side of the window. Brackets considers this folder to be the current *project*.
- The first time you start Brackets, it displays a Getting Started project. You can preview its index.html file to learn about some of the features of Brackets.
- When you open another project folder, the previous project is closed.
- In general, each Brackets project should contain the folders and files for one web application. For the purposes of this book, however, you can treat all of the book applications as a single project and all of the exercises as another project.

Figure 2 How to open and close the folder for a website

How to display, open, and close files

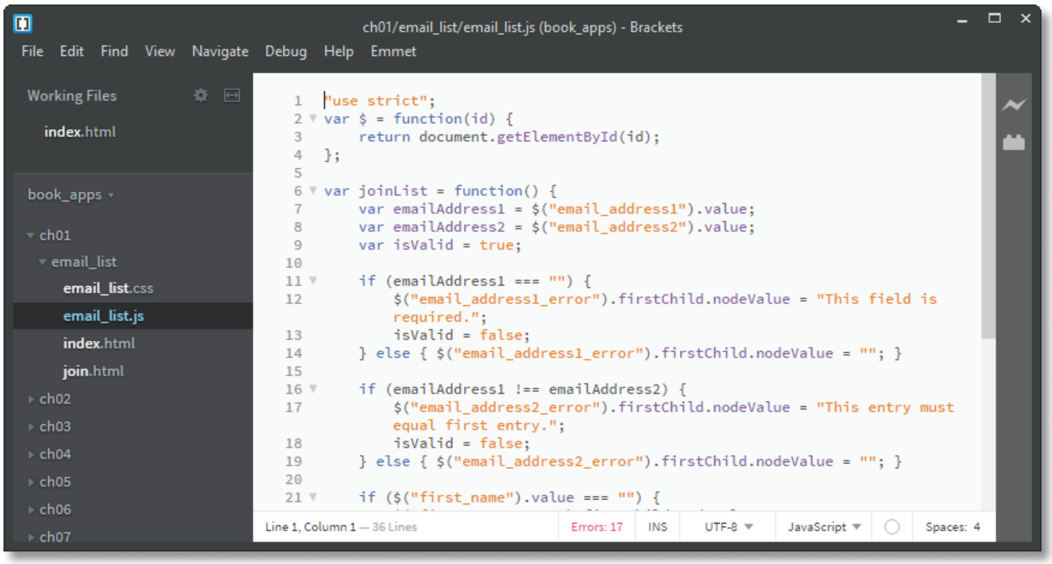
Figure 3 presents several techniques that you can use to open, close, and display files. Once you open the folder that contains the file, you can drill down to the file that you want to open by clicking on the ► symbols for the folders. In this example, the ch01 folder has been expanded so you can see the four files in the email_list folder, which is the application for this chapter.

To display one of the files in the tree, just click on it. And to open a file, double-click on it. You can also open a file that isn't part of the current project. To do that, you use the File→Open File command to locate and select the file.

When you open a file in Brackets, it appears in the Working Files list that's above the file tree at the left side of the window. Then, you can switch between open files by clicking on the file you want to display. You can also click on a file in the file tree to display a file without opening it. Then, if you make a change to the file, it's opened and added to the Working Files list.

To close a file in the Working Files list, you point to it and click the "X" to the left of the file name. Then, if you've made changes to the file and haven't saved them, Brackets will ask you if you want to save them.

Brackets with one open file and another file displayed



```
"use strict";
var $ = function(id) {
    return document.getElementById(id);
};

var joinList = function() {
    var emailAddress1 = $("email_address1").value;
    var emailAddress2 = $("email_address2").value;
    var isValid = true;

    if (emailAddress1 === "") {
        $("email_address1_error").firstChild.nodeValue = "This field is
        required.";
        isValid = false;
    } else { $("email_address1_error").firstChild.nodeValue = ""; }

    if (emailAddress1 !== emailAddress2) {
        $("email_address2_error").firstChild.nodeValue = "This entry must
        equal first entry.";
        isValid = false;
    } else { $("email_address2_error").firstChild.nodeValue = ""; }

    if ($("first_name").value === "") {
```

How to display a file

- Locate the file in the file tree or in the Working Files list and click on it.

How to open a file

- To open a file in the file tree, double-click on it.
- To open a file that's already displayed, start editing it.
- To open a file that isn't in the open project, use the File→Open command to locate and select the file.

How to close a file

- Select the file in the Working Files list and click the "X" to the left of the file name.

Description

- When you open a file with Brackets, it is put into the Working Files list.

Figure 3 How to display, open, and close files

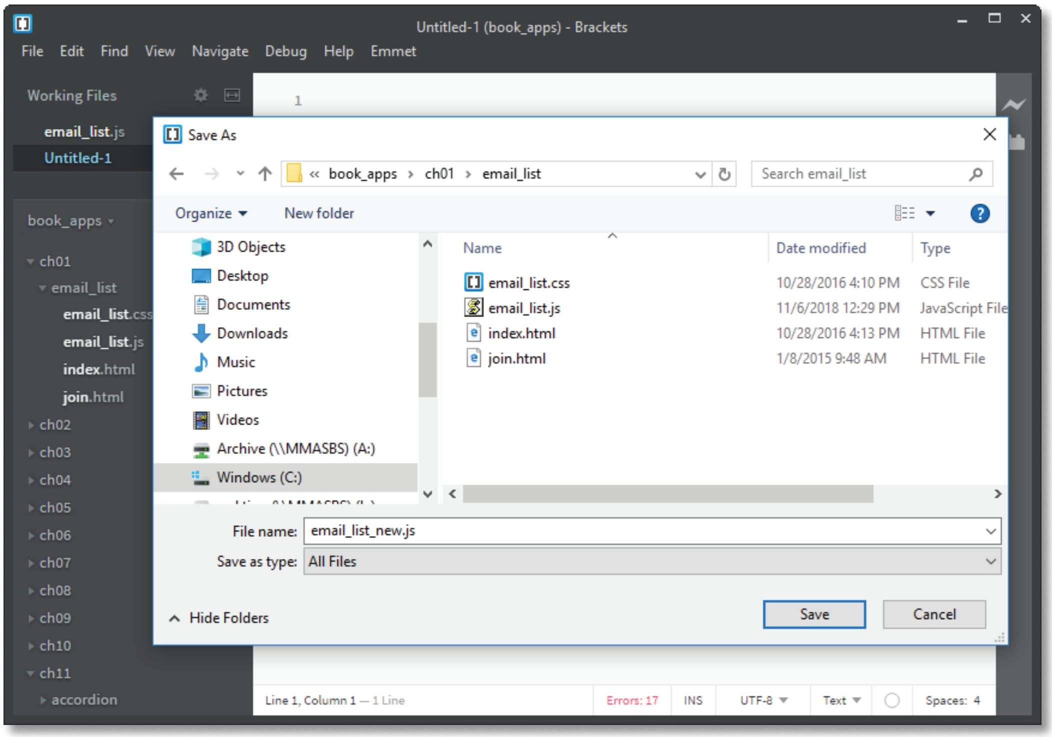
How to start a new HTML, CSS, or JavaScript file

To start a new HTML, CSS, or JavaScript file from scratch, you can use the first procedure in figure 4. After you start the new file by using the File→New command, you save it with the right extension for the type of file you're creating. The result is a new, but empty, file.

Often, though, you will want to start a new file from an old file that has some of the code that you're going to need for the new file. To do that, you can use the second procedure in this figure. After you create the new file, you can delete the code in the old file that you don't want, keep the code that you do want, and add the rest of the code that's needed.

As you get more experience, you may want to create HTML, CSS, and JavaScript files that contain the starting code that's required for each type of file. Then, you can use those files as *templates* for creating other files. When you start a new file from a template, though, you must remember to use the Save As command to save the template with a new name before you modify the template. That way, the original template file remains unchanged.

The Save As dialog box for a JavaScript file



Name	Date modified	Type
email_list.css	10/28/2016 4:10 PM	CSS File
email_list.js	11/6/2018 12:29 PM	JavaScript File
index.html	10/28/2016 4:13 PM	HTML File
join.html	1/8/2015 9:48 AM	HTML File

File name: email_list_new.js
Save as type: All Files

How to start a new file from scratch

1. Select the File→New command to create a new, empty file.
2. Use the File→Save command to select the folder that the new file should be saved in, and enter a name for the new file including its extension.
3. Use .html as the extension for HTML files, .css for CSS files, and .js for JavaScript files.

How to start a new file from another file

1. Open the file that you want to base the new file on. Then, use the File→Save As command to save the file with a new name.
2. Delete the code that you don't want in the new file, keep the code that you do want, and edit the file for the new application.

Description

- When you start a new file from scratch, be sure to save it with the right extension for the type of file that you're going to create.
- If you're going to create a new file that's similar to an existing file, you can open the existing file, save it with a new name, and edit it to suit your purposes.
- If you create a file that has all the starting code that you need for an HTML, CSS, or JavaScript file, you can use that file as a *template* for new files of that type.

Figure 4 How to start a new HTML, CSS, or JavaScript file

How to edit files

In the topics that follow, you'll learn how to use Brackets to edit HTML, CSS, and JavaScript files as well as the CSS or JavaScript that's embedded within an HTML file.

How to edit HTML code

Figure 5 shows how to edit an HTML file. When you open a file with an htm or html extension, Brackets knows what type of file you're working with so it can use color to highlight the syntax components. That makes it easier to spot errors in your code.

As you enter new code, the *code hints* feature presents lists of words that you can enter into your code. If, for example, you enter the left bracket (<) for a new element, a list of all of the elements is displayed. Then, you can select an element and press the Enter key to insert it into your code. You can also enter one or more letters after the starting bracket to filter the list so it only displays elements that start with those letters. In the example in this figure, I entered the letter *h* so only the elements that start with that letter are displayed.

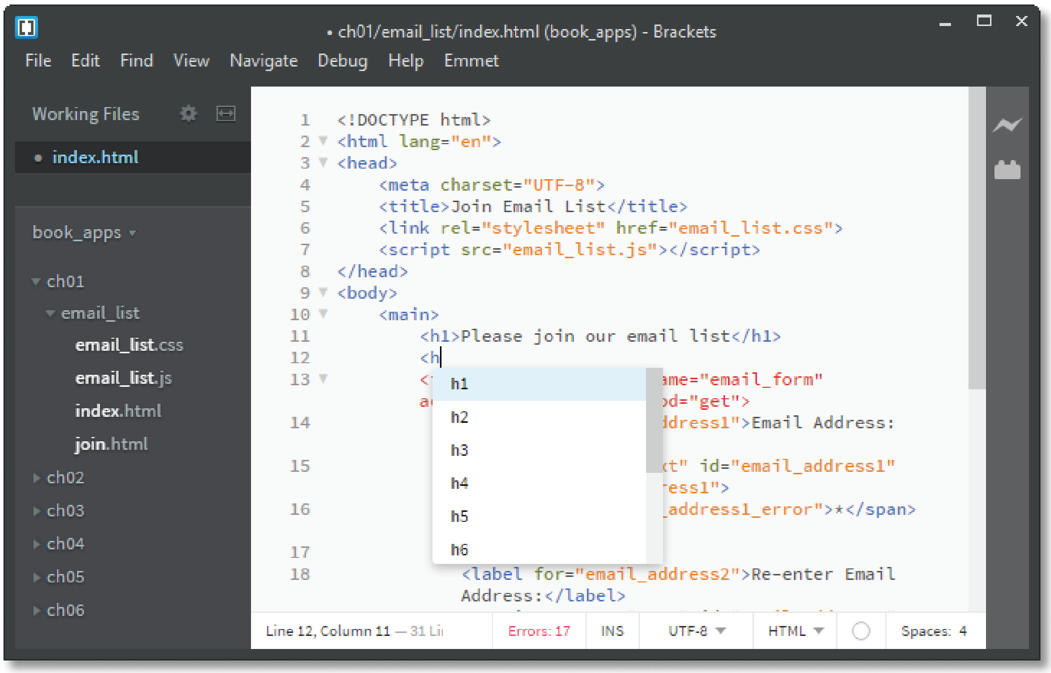
You can use a similar technique to enter attributes within an opening tag. To do that, enter a space and one or more letters after the attribute name. That displays a list of all the attributes that start with those letters. Then, you can select the attribute you want to add and press the Enter key. When you do that, Brackets inserts the attribute along with an equal sign and double quotes, and you can enter the value of the attribute between the quotes.

After you've entered all of the attributes, you type the closing bracket (>) for the tag. Then, Brackets adds the closing tag so all you have to do is enter the content for the tag.

This figure also lists some common coding errors. Often, the color coding will help you spot the first three types of errors. If, for example, you misspell an element name or if you forget to code a closing quotation mark, the color coding will indicate that the code isn't correct.

On the other hand, Brackets has no way of knowing what the correct code should be for file references in link, img, or <a> elements. As a result, you must discover those errors when you test the web page. If, for example, the file reference for a style sheet in a link element is incorrect, the CSS won't be applied. If the file reference in an img element is incorrect, the image won't be displayed and the value of the alt attribute will be displayed. And if the file reference for an <a> element is incorrect, the browser won't access the correct page when the link is clicked.

Brackets with code hints for an HTML element



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Join Email List</title>
    <link rel="stylesheet" href="email_list.css">
    <script src="email_list.js"></script>
</head>
<body>
    <main>
        <h1>Please join our email list</h1>
        <h
        <                     ame="email_form"
        a                     od="get">
                              ddress1">Email Address:

                              xt" id="email_address1"
                              ress1">
                              _address1_error">*</span>

            <label for="email_address2">Re-enter Email
            Address:</label>
```

Common HTML coding errors

- An opening tag without a closing tag
- Misspelled element or attribute names
- Quotation marks that aren't paired
- Incorrect file references in link, img, or <a> elements

How to edit an HTML file

- Brackets displays the different parts of an HTML file in different colors so they're easy to recognize. This helps you spot some of the common errors. For this to work, the file must have the htm or html extension.
- The *code hints* feature displays a list of elements that start with what you've typed. To insert an element, click on it or use the arrow keys to highlight it and press the Enter key. You can also use this feature to insert attributes.
- You can use the commands in the Edit and Find menus to do common editing tasks like commenting or uncommenting lines of code or finding and replacing code selections.
- You can also use standard editing practices like cutting, copying, and pasting.

Figure 5 How to edit HTML code

How to edit CSS code

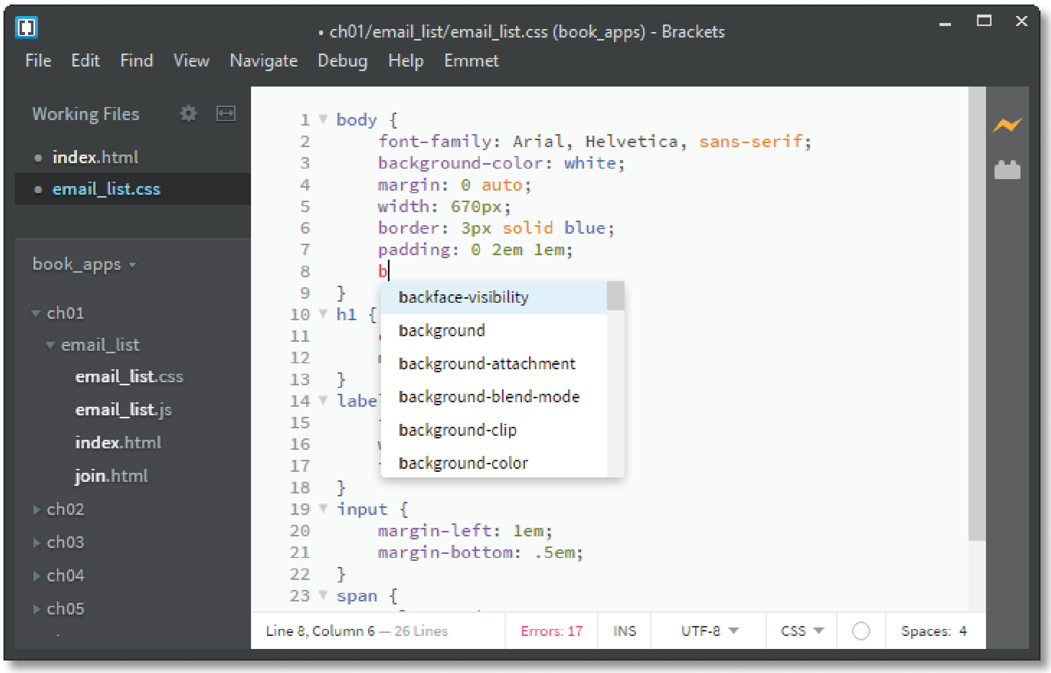
Figure 6 shows how to edit a CSS file or the CSS within a Script element in an HTML file. In either case, Brackets knows that it's working with CSS so it can use color to highlight the syntax components. That makes it easier to spot errors in your code.

Here again, the code hints feature helps you enter the code correctly. When you type the first letter of a property, for example, the code hints feature displays a list of properties that start with that letter. Then, you can select the property you want and press the Enter key to insert the name of the property and the colon that follows. In some cases, Brackets will display additional code hints that you can use to select the value of the property.

Besides that, Brackets automatically adds a right brace after you enter a left brace. Then, you can enter the declarations within the braces.

This figure also lists four common coding errors. Although color coding, code hints, and matching braces can help you avoid some of these errors, it can't help you avoid all of them. For example, Brackets has no way of knowing if you code an id or class name incorrectly. Instead, you'll find that out during testing when you notice that the style rules that you've specified for the elements haven't been applied.

Brackets with code hints for a CSS property



```
body {
    font-family: Arial, Helvetica, sans-serif;
    background-color: white;
    margin: 0 auto;
    width: 670px;
    border: 3px solid blue;
    padding: 0 2em 1em;
    b
}
h1 {


}
label


}
input {
    margin-left: 1em;
    margin-bottom: .5em;
}
span {
```

```
Line 8, Column 6 — 26 Lines    Errors: 17    INS    UTF-8    CSS    Spaces: 4
```

Common CSS coding errors

- Braces that aren't paired correctly
- Misspelled property names
- Missing semicolons
- Id or class names that don't match the names used in the HTML

How to edit the CSS in a CSS file or a Style element of an HTML file

- To edit CSS code, you can use the same techniques that you use to edit an HTML file. However, a CSS file must have the .css extension.
- By default, when you type the left brace after a selector, Brackets automatically adds the right brace. Then, you can enter the declarations between the braces.
- The code hints feature displays a list of properties that start with what you've typed. After you select one, Brackets automatically adds the colon after the property name. Then, as you enter the property name, the code hints feature lists possible values.

Figure 6 How to edit CSS code

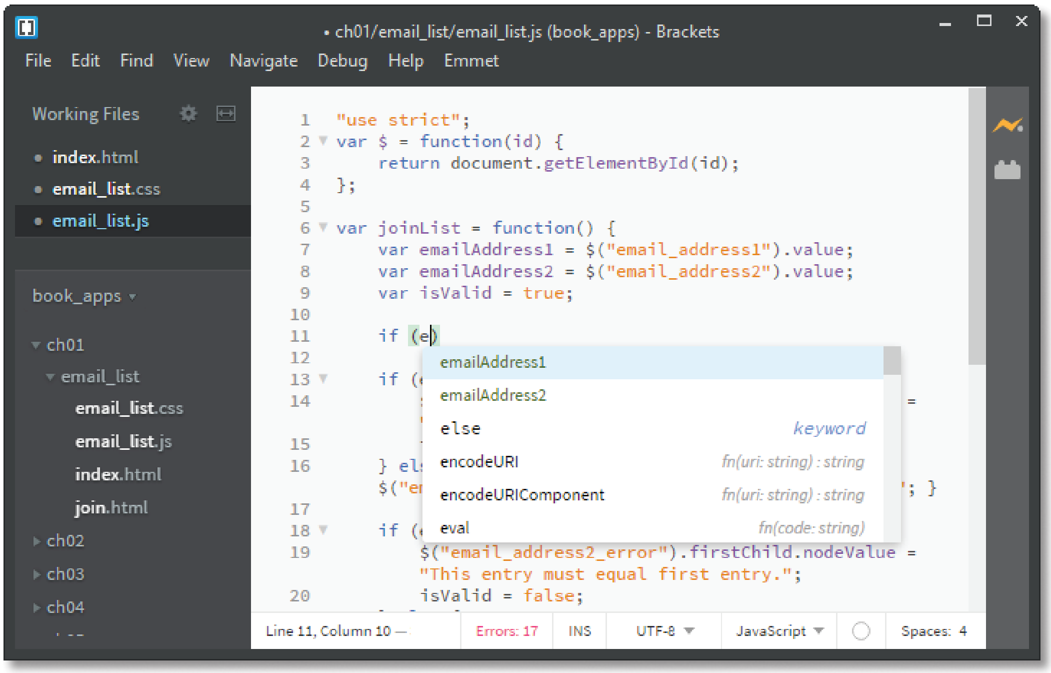
How to edit JavaScript code

Figure 7 shows how to edit a JavaScript file or the JavaScript that's in a Script element in an HTML file. In either case, Brackets knows that it's working with JavaScript so it can use color to highlight the syntax components. That makes it easier to spot errors in your code.

Here again, the code hints feature helps you enter your JavaScript statements correctly. When you type the first letter of a variable name, for example, the code hints feature displays a list of the variable names that have been declared that start with that letter. Then, you can select the name that you want and press the Enter key to insert the name into your code. Besides that, Brackets will add the right brace, parenthesis, or quotation mark whenever you enter the left one.

This figure also lists four common coding errors. Although color coding, code hints, and pairing braces, parentheses, and quotation marks can help you avoid some of these errors, it can't help you avoid all of them. For example, Brackets doesn't check to make sure the syntax of each statement is correct. Instead, you'll find that out during testing when your JavaScript application stops running or doesn't run at all.

Brackets with code hints for a JavaScript statement



```
"use strict";
var $ = function(id) {
    return document.getElementById(id);
};

var joinList = function() {
    var emailAddress1 = $("email_address1").value;
    var emailAddress2 = $("email_address2").value;
    var isValid = true;

    if (e)

    if ((                                                        =
        
        
    } el
    $("er                                                        "; }

    if ((
        $("email_address2_error").firstChild.nodeValue =
        "This entry must equal first entry.";
        isValid = false;
```

```
emailAddress1
emailAddress2
else                                    keyword
encodeURI                               fn(uri: string) : string
encodeURIComponent                      fn(uri: string) : string
eval                                    fn(code: string)
```

Line 11, Column 10 — Errors: 17 INS UTF-8 JavaScript Spaces: 4

Common JavaScript coding errors

- Braces, parentheses, or quotation marks that aren't paired correctly
- Misspelling keywords, like getElementByID instead of getElementById
- Omitting the semicolon at the end of a statement
- Misspelling or incorrectly capitalizing an identifier, like defining a variable named salesTax and referring to it later as salestax

How to edit the JavaScript in a JavaScript file or a Script element in an HTML file

- To edit JavaScript code, you can use the same techniques that you use to edit HTML or CSS. However, a JavaScript file must have the .js extension.
- By default, when you type the left brace, left parenthesis, or left quotation mark in a JavaScript statement, Brackets adds the right brace, parenthesis, or quotation mark.
- Depending on what you're entering, the code hints feature displays a list of the variable names, properties, or methods that are appropriate.

Figure 7 How to edit JavaScript code

How to use split view and the Quick Edit feature

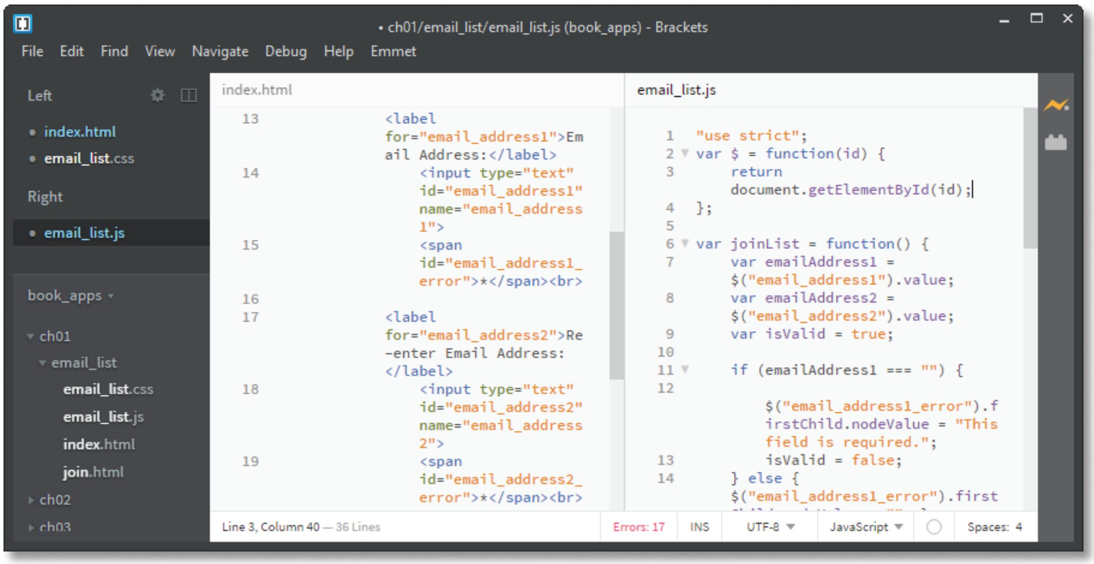
As you write the code for web pages, you often want to look at the HTML for a page and the related CSS at the same time. Or, you might want to look at the code for an HTML page and the related JavaScript at the same time. Brackets provides two ways to do that, and they are summarized in figure 8.

The first way is to split the screen horizontally or vertically, and then display the HTML in one part of the split screen and the CSS or JavaScript in another. This is illustrated by the example in this figure, which is split vertically. Here, the HTML is in the left pane and the related JavaScript is in the right pane.

The other way to do this is to use the Quick Edit feature. For instance, you can put the cursor in an HTML element name, id attribute, or class attribute and then press Ctrl+E (or Cmd+E for Mac OS X). When you do that, a Quick Edit window is opened that displays all of the style rules for that element, id, or class. You can also use this feature to display the code for a JavaScript function when you put the cursor in the name for a function call.

Once the code is displayed in the Quick Edit window, you can edit it just as if you were working in the file that it's in. That is especially useful when you're working with long files or applications with several files because the Quick Edit feature finds the related code for you. You don't have to scroll through a long file to find it or figure out which file the code is in.

Brackets with a vertical split screen



```
                <label
                for="email_address1">Em
                ail Address:</label>
                    <input type="text"
                    id="email_address1"
                    name="email_address
                    1">
                    <span
                    id="email_address1_
                    error">*</span><br>

                <label
                for="email_address2">Re
                -enter Email Address:
                </label>
                    <input type="text"
                    id="email_address2"
                    name="email_address
                    2">
                    <span
                    id="email_address2_
                    error">*</span><br>
```

```
"use strict";
var $ = function(id) {
    return
    document.getElementById(id);
};

var joinList = function() {
    var emailAddress1 =
    $("email_address1").value;
    var emailAddress2 =
    $("email_address2").value;
    var isValid = true;

    if (emailAddress1 === "") {

        $("email_address1_error").f
        irstChild.nodeValue = "This
        field is required.";
        isValid = false;
    } else {
    $("email_address1_error").first
```

How to display two files with a split screen

- Select View→Vertical Split or View→Horizontal Split to split the editor into two panes. Then, click in a pane and click on a file in the file tree to open it in that pane.
- To close split view, select View→No Split.

How to use the Quick Edit feature to display the CSS for an HTML element

- Place the cursor in an element name or id or class attribute and press Ctrl+E for Windows or Cmd+E for Mac OS X. That will display the style rules for that element, id, or class.
- You can then edit the style rules or create new rules.

How to use the Quick Edit feature to display the code for a JavaScript function call

- Place the cursor in the name of a function that's being called. Then, press Ctrl+E for Windows or Cmd+E for Mac OS X. That will display the code for the function, even if it's in a separate file.
- You can then edit the function, even if it's in a separate file.

Description

- The split screen feature lets you review and edit related portions of code in HTML, CSS, and JavaScript files.
- The Quick Edit feature provides another way to review and edit related portions of code. This is valuable in applications with multiple files and many lines of code.

Figure 8 How to use split view and the Quick Edit feature

How to work with ESLint and JSLint

Besides JavaScript code hints, Brackets provides *ESLint* and *JSLint*. These are *code linters* that are designed to find patterns of code that don't adhere to some specific style guidelines. Brackets displays the messages for any problems that ESLint or JSLint find in a panel below the JavaScript code. As a result, these linters can help you discover problems with your JavaScript code before you execute it.

The trouble with these linters is that they often issue error and warning messages for statements that are acceptable JavaScript. As a result, you won't need to make any changes for many of the messages, and reviewing all of them can be time-consuming. Remember too that you will find and correct all of the JavaScript errors when you test and debug your applications, as shown in the next two figures, so using the linters isn't essential.

To illustrate what ESLint and JSLint do, figure 9 shows the messages for a 35-line JavaScript file. Here, JSLint has issued 15 warning messages, although that accordion panel is hidden, and ESLint has issued one error message. In this case, only the one ESLint message identifies a problem that has to be fixed.

So, one way to deal with ESLint and JSLint is to turn them both off by unchecking the Lint Files on Save box in the View menu. That's the simplest approach, and one that often makes sense for beginning JavaScript programmers.

But as you get more experience with JavaScript, a better way to deal with ESLint and JSLint is to hide the panel for the JSLint messages, but review the ESLint messages. That's because the ESLint messages are more likely to identify problems that need to be fixed rather than code that doesn't adhere to certain style guidelines.

If you take this approach, you might also want to add this comment to the top of your JavaScript file:

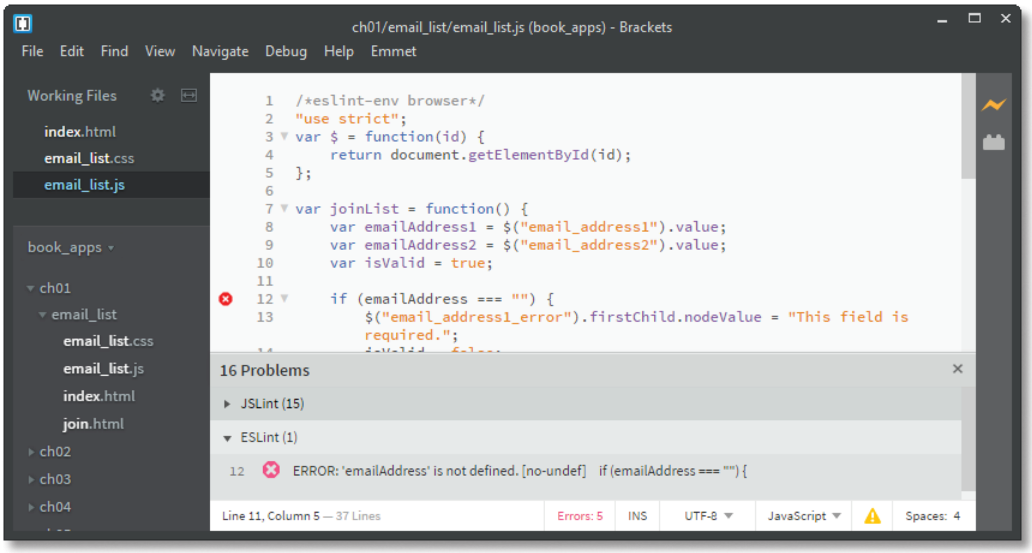
```
/*eslint-env browser*/
```

That will stop error messages that treat the document and window objects as "not defined." This comment tells ESLint that the environment for this JavaScript code is going to be the web browser so those objects are valid.

In this example, you can see that line 12 of the JavaScript code has a red error marker before it. You can also see that the message in the ESLint panel below says that the variable name "emailAddress" is "not defined". That's because the variable name should be "emailAddress1". In this case, the message points out a real JavaScript problem that can be fixed before the application is executed.

If you're also interested in the JSLint messages, you can open the accordion panel for those messages. There, you'll see that JSLint generates warning messages for unnecessary spaces and other style issues like that. Although JSLint does generate some messages for JavaScript errors that do have to be fixed before the application will run, you usually have to find them within the many warning messages for coding that doesn't have to be fixed.

Brackets with an ESLint error marker and message in a JavaScript file



```
/*eslint-env browser*/
"use strict";
var $ = function(id) {
    return document.getElementById(id);
};

var joinList = function() {
    var emailAddress1 = $("email_address1").value;
    var emailAddress2 = $("email_address2").value;
    var isValid = true;

    if (emailAddress === "") {
        $("email_address1_error").firstChild.nodeValue = "This field is
        required.";
```

16 Problems
JSLint (15)
ESLint (1)
12 ERROR: 'emailAddress' is not defined. [no-undef] if (emailAddress === "") {

Line 11, Column 5 — 37 Lines | Errors: 5 | INS | UTF-8 | JavaScript | Spaces: 4

How to turn off ESLint and JSLint

- In the View menu, uncheck Lint Files on Save.

How to hide and display ESLint and JSLint messages

- To hide the panel that displays the ESLint and JSLint messages, you can close the panel. To open that panel, you can click the Warning icon at the bottom of the editing pane.
- To display or hide the ESLint or JSLint messages, open or close its accordion panel.

How to stop the ESLint errors for the document and window objects

- ESLint treats the document and window objects as “not defined” because they are web objects. To fix that, add this comment at the start of your JavaScript code:

```
/*eslint-env browser*/
```

Description

- *ESLint* and *JSLint* are *code linters* that check for JavaScript code that doesn’t adhere to specific syntax and style guidelines. These linters then provide either error or warning messages.
- Although ESLint often identifies JavaScript errors that will cause errors when the application is run, JSLint focuses more on coding style.
- By default, Brackets provides ESLint and JSLint checking whenever a JavaScript file is saved and displays their messages in a panel below the JavaScript code.
- Because ESLint and JSLint generate many messages that don’t identify JavaScript errors that have to be fixed, you may prefer to turn both of these linters off.

Figure 9 How to work with ESLint and JSLint

How to preview and debug a JavaScript application

To test and debug a JavaScript application, you can use the normal techniques for starting the application in a browser. Then, you can use the developer tools of the browser to find any JavaScript errors. When you use Brackets, though, you can use its Live Preview feature to improve upon these standard procedures.

How to preview an application

Figure 10 shows how use the Live Preview feature of Brackets to run a JavaScript application. To do that, you display the HTML file or JavaScript file for the application in Brackets and click the Live Preview icon. Then, the page is displayed in a separate copy of Chrome.

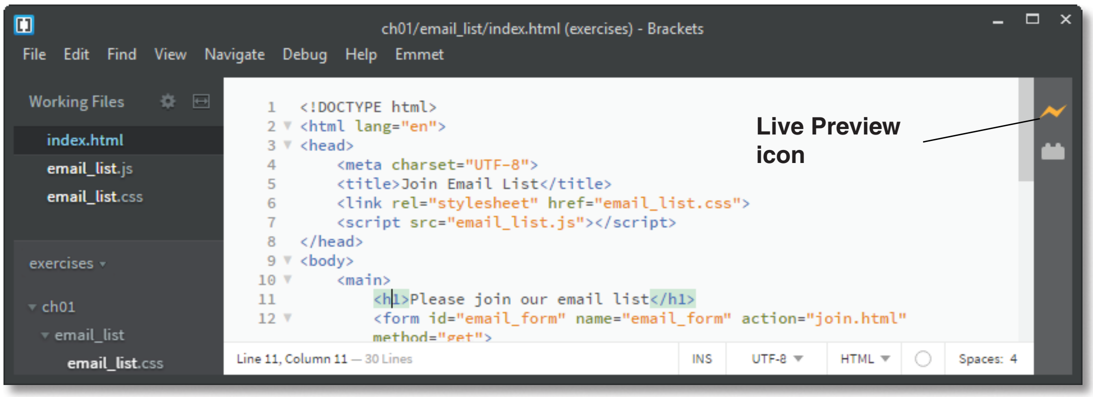
While a preview is displayed, you can make changes to the HTML and CSS code for the page and it will be reflected automatically in the preview. That makes it easy to test different content and formatting without actually saving the HTML or CSS files. You can also make changes to the JavaScript code while a preview is displayed. In that case, though, you have to save the file and reload the page in the browser before the changes take effect.

You can also highlight an HTML element in the browser preview by placing the cursor in the opening tag of the element in the HTML file. And you can highlight all of the elements that a style rule is applied to in the browser preview by placing the cursor in that style rule in the CSS file.

In the Chrome browser, you can test the JavaScript application by making the required entries and doing the user actions. If the JavaScript application stops working, though, you need to use Chrome's developer tools to find the statement that caused the problem, as shown in the next figure. Then, you need to return to Brackets, fix the error, and save the file. At that point, you can return to Chrome and press the Reload button to restart the application with the saved JavaScript file.

While Live Preview is active, the Live Preview icon in Brackets is displayed in orange. Then, to disconnect from the browser window, you can click this icon. Note, however, that this doesn't close the browser window, so you'll want to do that when you're done previewing the page. You can also close the browser window while Live Preview is still connected. In that case, Brackets will display a message indicating that Live Preview was cancelled for an unknown reason.

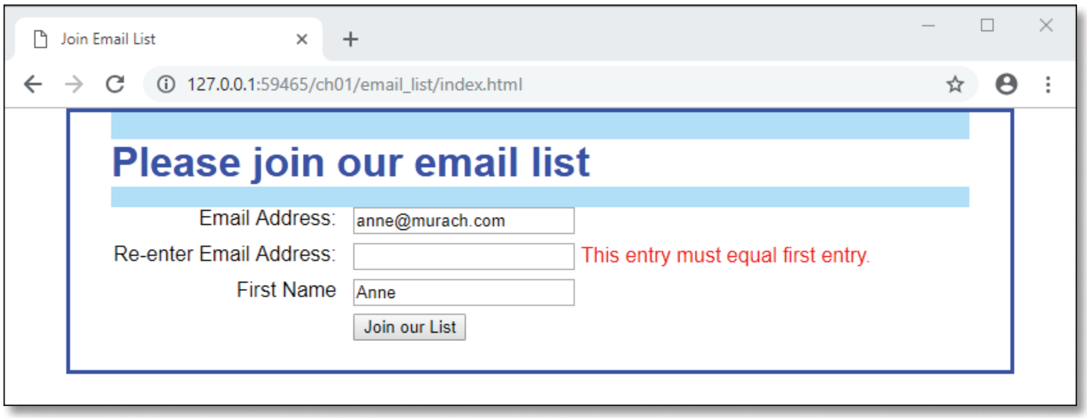
The Live Preview icon



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Join Email List</title>
    <link rel="stylesheet" href="email_list.css">
    <script src="email_list.js"></script>
</head>
<body>
    <main>
        <h1>Please join our email list</h1>
        <form id="email_form" name="email_form" action="join.html"
        method="get">
```

Line 11, Column 11 — 30 Lines INS UTF-8 HTML Spaces: 4

The Chrome browser with the selected h1 element highlighted



Please join our email list

Email Address:	anne@murach.com	
Re-enter Email Address:		This entry must equal first entry.
First Name	Anne	
	Join our List	

Description

- To preview an application, display its HTML or JavaScript file and click the Live Preview icon at the right side of the window. Then, the preview is displayed in a copy of Chrome that's separate from any other copies you may have open.
- If you place the cursor in an element in an HTML file, that element is highlighted in the browser preview. If you place the cursor in a style rule in a CSS file or Style element, all elements that the style rule is applied to are highlighted.
- If you make any changes to the HTML or CSS code in Brackets, the changes will be displayed in the browser preview immediately…you don't have to save the files.
- If you make any changes to the JavaScript code, the file has to be saved before the changes will be implemented in the browser preview. You will also have to reload the page in the browser.

Figure 10 How to preview an application

How to use Chrome's developer tools to find runtime errors

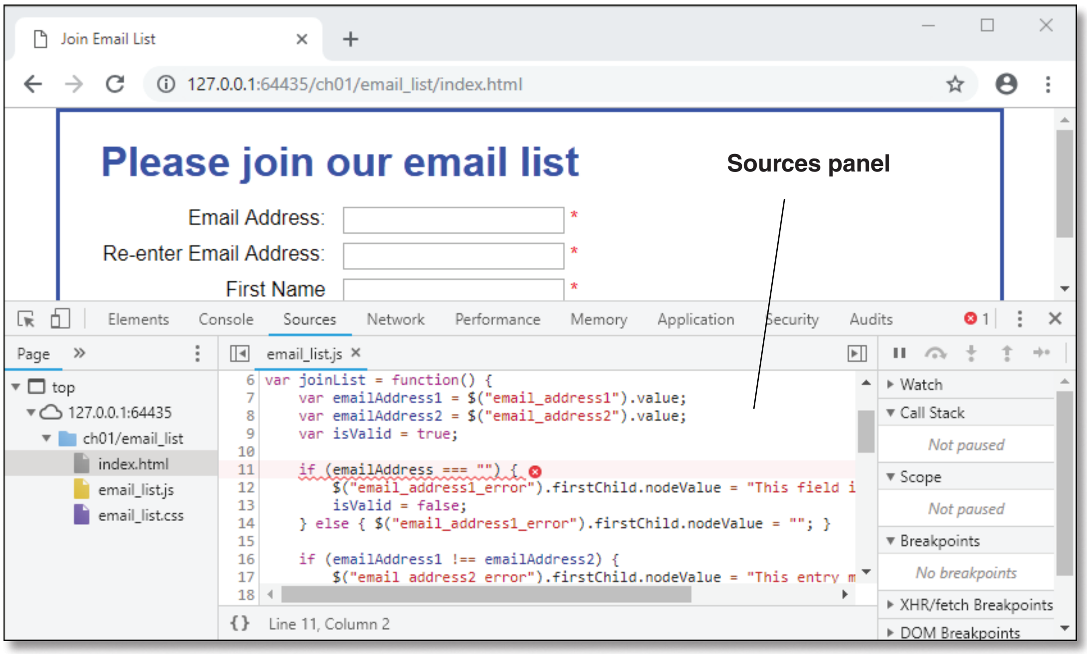
As you test even the simplest of JavaScript applications, you're likely to have errors in your code. Then, when the JavaScript engine can't run one of the JavaScript statements, the application will stop running. This is known as a *runtime error*. In the example in figure 11, the application stopped when the user clicked the Join our List button below the text boxes for the user entries.

To find the cause of a runtime error, you can use Chrome's *developer tools*, as shown in this figure. Although there are several ways to open these tools, the quickest way is to press the F12 key. That's why the developer tools for Chrome and other browsers are often referred to as the *F12 tools*.

Once the tools are open, you can open the Console panel to see if there's an error message. If there is, you can click on the link in the message, which will take you to the Sources panel where the statement that caused the error will be highlighted. This panel is shown in this figure with the statement in line 11 highlighted, so that's the statement that caused the runtime error. In this case, "emailAddress" should be "emailAddress1".

Incidentally, you can also go directly to the Sources panel when you open the developer tools. In that case, though, you need to scroll through the JavaScript statements to find the highlighted error statement. The benefit of using the link for the error in the Console panel is that it takes you right to the error statement in the Sources panel.

Chrome with an open Sources panel that shows an error



```
var joinList = function() {
    var emailAddress1 = $("email_address1").value;
    var emailAddress2 = $("email_address2").value;
    var isValid = true;

    if (emailAddress === "") {
        $("email_address1_error").firstChild.nodeValue = "This field i
        isValid = false;
    } else { $("email_address1_error").firstChild.nodeValue = ""; }

    if (emailAddress1 !== emailAddress2) {
        $("email address2 error").firstChild.nodeValue = "This entry m
```

When and how to open Chrome's developer tools

- When the JavaScript application stops working, which indicates that a runtime error has occurred, it's time to open the developer tools.
- To open or close these tools, press F12.

How to find the JavaScript statement that caused the runtime error

- In the Console panel of the tools, you should see an error message along with a link to the statement that caused the error. You can then click on that link to go to the Sources panel where you'll see the error statement highlighted within the rest of the JavaScript code.

Description

- When a JavaScript application stops running, that's usually because the JavaScript engine couldn't execute one of the JavaScript statements. This is known as a *runtime error*.
- You can then use Chrome's *developer tools* to find the statement that caused the error. Because you start these tools by pressing the F12 key, these tools are often referred to as the *F12 tools*.
- Once you know which statement caused the error, you should be able to fix the error with Brackets and save the changed file. Then, you can return to Chrome and click the Reload button to re-run the application from the start.
- This works whether or not you're using the Live Preview feature of Brackets.

Figure 11 How to use Chrome's developer tools to find runtime errors

Perspective

Now that you've completed this document, you should be able to install Brackets and use it to create and edit HTML, CSS, and JavaScript files. You should also be able to test and debug those files by previewing them with the Live Preview feature of Brackets and debugging the JavaScript by using Chrome's developer tools.

Terms

Brackets project
file tree
template
code hint
code linter
ESLint
JSLint
runtime error
developer tools
F12 tools

Summary

- Brackets is a free text editor that can be used to edit HTML, CSS, and JavaScript code.
- To help you read the code, Brackets displays the syntax components with different colors. Brackets also provides *code hints*, and it matches a left brace, parenthesis, or quotation mark entry with its right equivalent.
- Brackets also provides *ESLint* and *JSLint*, two *code linters* that check JavaScript code for syntax and style errors. Of the two, ESLint is more likely to identify JavaScript errors that need to be fixed.
- To test an application, you can use the Live Preview feature of Brackets. This makes it easy to see the effects of changes to the HTML or CSS code without saving the related files.
- To debug a JavaScript application when a *runtime error* occurs, you can use Chrome's *developer tools*. However, you do need to save the JavaScript files whenever you make changes to them.

Before you do the exercise…

Before you do the exercise that follows, you should install the Chrome browser as well as the applications and exercises for *Murach's JavaScript and jQuery (3rd Edition)*, as shown in the appendix for that book. You should also download and install Brackets, as shown in figure 1 of this document.

Exercise 1 Get acquainted with Brackets

This exercise will help you get acquainted with Brackets. It does that by stepping you through the skills that are presented in this document.

Set up the projects for *Murach's JavaScript and jQuery (3rd edition)*

1. Start Brackets. Then, use the File→Open Folder command to open a project folder for the applications that are presented in *Murach's JavaScript and jQuery*:

   ```
   C:\murach\javascript_jquery\book_apps
   ```

   This should display a file tree with one folder for each chapter in the book.
2. Use the same command to open a project folder for the book exercises:

   ```
   C:\javascript_jquery\exercises
   ```

   This should close the book_apps folder and replace it with a new file tree.
3. You now have easy access to all of the book applications and exercises for this book. To illustrate, click on the name of the exercises folder at the top of the file tree and select book_apps from the drop-down list to reopen the book_apps folder. Then, switch back to the exercises folder.

Edit the HTML and CSS files and use Live Preview to preview them

4. In the file tree for the exercises, click on the ► symbol before ch01 to display the files in that folder. Next, display the files in the email_list folder. Then, double-click on the file named index.html to open and display that file. Note that it is now listed under Working Files.
5. Start a new h2 element right after the existing h1 element by typing the left bracket and *h*, and note how Brackets provides code hints. Select h2 from the hints, type the closing bracket (>) for the opening tag, and note how Brackets adds the closing tag.
6. Undo the changes that you made by pressing Ctrl+Z until the file is restored to its starting code. Or just delete the element that you started.
7. Click on the Live Preview icon shown in figure 10 to preview the file in Chrome. If possible, size and position Brackets and Chrome so you can see them both at the same time. Then, click on an element in the HTML file to see that it is now highlighted in the Chrome preview.
8. Double-click on the file named email_list.css to open that file. Then, change the color property for the h1 element to red, and look at the browser preview to see that the heading is now displayed in red, even though you didn't save the changed file.

9. In the Chrome browser, enter valid data into the first two text boxes and click the Join our List button. An error message should be displayed. Then, enter valid data into the third text box, and click the button again. A thank-you message should be displayed on a new web page.

Edit a JavaScript file

10. Drop down the View menu and uncheck the Lint Files on Save box. That will turn off the two code linters: ESLint and JSLint.
11. In the file tree for the exercises, click on the email_list.js file to display it. Then, with the screenshot in figure 7 as a guide to the placement of the code, start an if statement like this:

    ```
    if (
    ```

    Note that Brackets adds the closing parenthesis, and note that the file is opened and placed in Working Files as soon as you start to edit it.
12. Within the parentheses of the if statement you started, type the letter *e* to display a list of the possible entries that start with that letter. In that list, emailAddress1 and emailAddress2 are the names of the variables that were created by the var statements in lines 7 and 8. Now, press Enter to insert the first variable name within the parentheses.
13. That shows how Brackets helps you enter and edit JavaScript code. Now, undo the changes that you made by pressing Ctrl+Z until the file is restored to its starting code. Or just delete the if statement that you started.

Test and debug a JavaScript application

14. In the first if statement in the JavaScript file, change the variable name from emailAddress1 to emailAddress. That introduces an error in the JavaScript file because a variable named emailAddress hasn't been defined.
15. Before you can test the change you've made, you need to save the JavaScript file. To do that, you can right-click on the file name in Working Files and select the Save command. Or, with the file displayed, you can use the File→Save command.
16. To test the change you've made, switch to the Chrome browser that should still be open in Live Preview mode. Then, click the Back button to return to the starting page for the application, and click the Reload button to run the application with this change.
17. Without entering any data, click on the Join our List button. That should display two messages, but nothing happens. That indicates that a runtime error has occurred.
18. With figure 11 as a guide, use Chrome's developer tools to find the statement that caused the error. It is of course the statement that you changed in step 14.
19. Switch to Brackets, restore the variable name to emailAddress1, and save the file. Then, switch to Chrome, click the Reload button, and test the application again. This time it should work.

Start new files

20. Use the File→New command to start a new file, and use the File→Save As command to save it with the name testpage.html in the ch01 folder. That creates an empty HTML file.
21. Display the email_list.js file, and use the File→Save As command to save the file as template.js in the ch01 folder. Then, delete the joinList function (lines 5-29), delete the two statements within the onload function (lines 32-33), and save the file. This file now has the starting code that's required by most JavaScript files, and you can use this file as a template.

Use the split screen feature

22. Use the View→Vertical Split command to split the screen. Then, display the index.html file in the left pane and the email_list.js file in the right pane. To do that, you can click on a pane to put the focus on it and then click on the file in the file tree that you want to display in it. You can also drag a file from the Left group in Working Files to the Right group, or vice versa.
23. With the index.html and email_list.js files in the left and right panes, scroll the code so the HTML attributes and the JavaScript that refers to them are side by side as in figure 8.
24. Use the View→No Split command to end the split screen feature.

Use the QuickEdit feature

25. Click on the index.html file in Working Files to display that file. Then, put the cursor in the h1 tag, and press Ctrl+E (or Cmd+E) to display the CSS for that element. Now, change the color back to blue, and note the change in Chrome.
26. Display the email_list.js file. Then, in the window.onload function at the end of the file, put the cursor in *joinList* and press Ctrl+E (or Cmd+E). That will display the entire joinList function, and you can modify it if necessary. In a short program like this, you don't really need this feature, but this comes in handy in large programs.
27. Click the X in the upper left of the Quick Edit feature to close it. Then, switch back to the index.html file and close that Quick Edit feature too.

Use ESLint and JSLint

28. Display the email_list.js file. Then, drop down the View menu and check the Lint Files on Save box. That will turn on the two code linters: ESLint and JSLint.
29. If the ESLint and JSLint messages aren't displayed, save the file. Then, with figure 9 as a guide, review the two errors that are in the ESLint panel. Note, however, that those aren't really errors because "document" and "window" are valid objects in web applications.
30. To get rid of these errors, enter this JavaScript comment at the top of the JavaScript file, as shown in figure 9:

    ```
    /*eslint-env browser*/
    ```

    This tells ESLint that the environment for the application is the browser. Then, save the file, and note that the ESLint messages have been removed.

31. In the first var statement in the joinList function in the JavaScript file, change the variable name from emailAddress1 to emailAddress. Then, save the file to see that this generates three ESLint messages and puts error markers in front of three lines of code. This shows how ESLint can help you find JavaScript errors before you test an application. Now, fix this error, and save the file again.
32. Review the JSLint error messages, and note that these are warning messages that identify style issues rather than JavaScript errors that need to be fixed. You know that because the application worked correctly until you introduced the error in step 31. Occasionally, though, one of the JSLint messages will identify a JavaScript error that does need to be fixed.

Experiment and clean up

33. If you want to experiment more, by all means do that. When you're through, continue with the next steps.
34. Use the File→Close All command to close all of the open files. Or, move the cursor to a file in the Working Files list and click the X on its left to close one file at a time. Either way, if a file hasn't been saved, a dialog box will be displayed so you can save it.
35. When all of the files have been closed, close the instance of Chrome that has been previewing the files. Then, close Brackets.