

GYMNASIUM

---

# THE JAVASCRIPT & JQUERY SURVIVAL GUIDE

---

*Lesson 1 Handout*

*Getting Started*

# ABOUT THIS HANDOUT

This handout includes the following:

- A list of the core concepts covered in this lesson
- The assignment(s) for this lesson
- A list of readings and resources for this lesson including books, articles, and websites mentioned in the videos by the instructor, plus bonus readings and resources hand-picked by the instructor
- A transcript of the lecture videos for this lesson

## CORE CONCEPTS

1. Working with the command line to clone the course lesson files from a remote repository on GitHub and installing Node.js and task runner packages like Bower and Gulp to setup a local web server environment for working locally
2. Understanding the ins and outs of the JavaScript console, also known as Developer (Dev) Tools, within various popular web browsers such as Google Chrome, Apple Safari, Microsoft Internet Explorer, and Mozilla Firefox.
3. Reviewing the basics of JavaScript expressions, specifically arithmetic operators, to better understand the output shown in the JavaScript console

# ASSIGNMENTS

- Using the command line, Terminal on Mac and Command Prompt on Windows, install the course lesson files and setup a local web server for working locally. *See pages 5–7 for detailed step-by-step instructions.*

# INTRODUCTION

*(Note: This is an edited transcript of the The JavaScript & jQuery Survival Guide lecture videos. Some students work better with written material than by watching videos alone, so we're offering this to you as an optional, helpful resource. Some elements of the instruction, like live coding, can't be recreated in a document like this one.)*

This is JavaScript and jQuery's survival guide—a short online course developed by Aquent. My name is Kevin Chisholm. I've been working in the technology field since 1996. For the last several years, I've worked exclusively as a front-end web developer, in most cases providing JavaScript consulting services to various clients. I also enjoy teaching and writing about front-end technology.

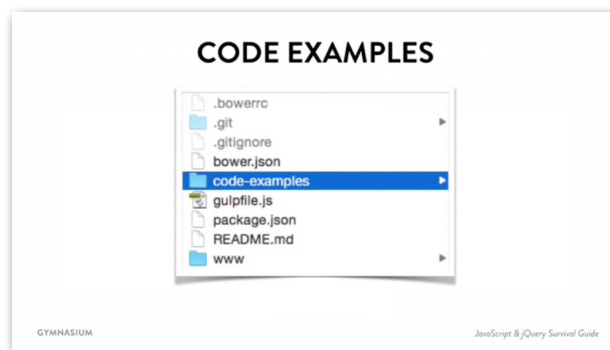
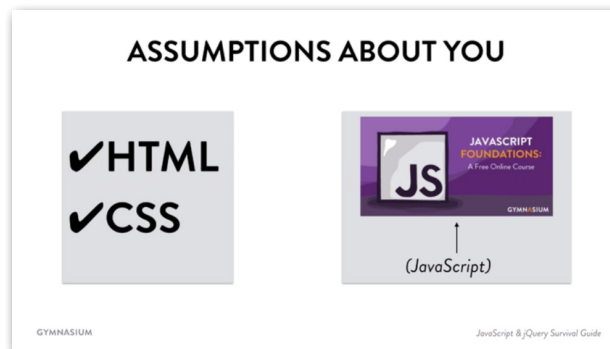
There are some assumptions that I'm making about you. My main assumption is that you have a working knowledge of HTML and CSS and that you understand the roles that they play in a web page. This video builds upon Aquent's JavaScript foundations course. I'm assuming you've completed it and are comfortable with the topics you've learned.

If you're not familiar with that course, I recommend that you stop here and take the time to review it. This way, the topics that I cover will make more sense. There will be some overlap between the JavaScript foundations course and this video, but this is intentional, as I wanted to be sure to emphasize key areas.

In this course, I'll cover executing JavaScript in the console, managing events with jQuery, and making changes to a web page with jQuery. I chose these topics because when I started out as a front-end web developer, I found myself repeatedly faced with similar challenges, and I felt that it might save you some time if you are familiar with common ways to solve these problems.

If you're not familiar with jQuery, it's an open source JavaScript library that makes selecting and modifying web page elements significantly easier. There are code examples included with this course so that you can follow along with the video and see the results in your own browser. You can find these code examples in the root of the project repository. I'll demonstrate how to access these code examples very shortly.

There are several technologies that we'll leverage in order to set up the example web page. Node.js is an open source runtime environment which allows JavaScript code to be executed on a hard drive, much like other programs.



NPM is a package manager that simplifies the process of using third party modules with Node. Git is a popular version control system which simplifies the process of collaborating on software projects. Gulp is a node module that helps you automate tasks such as running a local build. And Bower provides package management for client-side assets such as JavaScript files.

At this point, you may be wondering, well, why not just download a zip file? Well, I hope you'll feel comfortable taking a leap of faith here. The use of these kinds of tools is considered a best practice in front-end web development, and I encourage you to learn more about them in your spare time. In this video, we'll only interact with these tools in the very beginning, so there's no need to be too concerned with the details. But again, keep them on your radar. They help make front-end web development much more efficient.

If you're using an Apple computer, I'm assuming that you have Node and Git installed. If you do not, please pause this video and use the links that you see to install these programs on your computer. As a side note, if you are interested in learning more about either Node or Git, there are two videos that cover these topics in further depth here on Aquent Gymnasium. Please note that watching these videos is not required for this course, however.

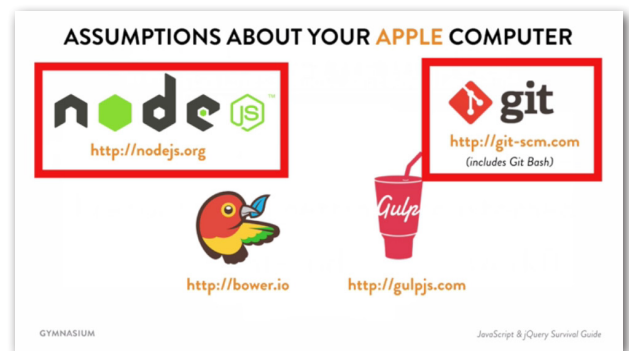
Now, although Bower and Gulp are node modules, they are also requirements and need to be installed globally. This is a one-time setup task, and since they are installed using your terminal application, I'll demonstrate these steps very shortly. If you're using a Windows computer, Node, Git, Bower, and Gulp are required as well. It's important to note that when you install Git on your Windows computer, an application called Git Bash is installed as well. On Windows, you'll need to use Git Bash as your terminal application.

In the next chapter, I'll show you how to install both Bower and Gulp on your own system. See you soon.

## INSTALLING LESSON FILES WITH BOWER AND GULP

In this chapter, I'll walk through the process of installing Bower and Gulp for both Apple and Windows. Although I'll be demonstrating everything on a Mac, I'll also give you the Windows instructions at the same time. Open up your terminal application on Windows. That will be Git Bash. In your terminal application, type `sudo npm install -g bower`. If you're using Windows, you can omit the `sudo` part of the command.

If you did use `sudo`, you'll be prompted for your password. Enter the password that you used to log on to your computer. After you enter your password, the Bower installation should start. It may take a few seconds. And then once it completes, we can verify that this installation was successful by typing `bower --v`, which should indicate a version number. That indicates that the installation was successful.



Next, we need to install Gulp. In your terminal application, type `sudo npm install -g gulp`. If you're using a Windows computer, you can omit the `sudo` part of the command. The Gulp installation may take a few seconds. And if you see any warnings, you can very likely just simply ignore them.

Once the Gulp installation has completed, we can verify it was successful by using the hyphen-hyphen V flag again. So in your terminal application, type `Gulp --v` And that should give us a version number, which indicates the installation was successful.

Setting up our code examples will be simple. Using a terminal application or Git Bash on Windows, you'll first moved to your desktop directory and then download the code from GitHub, an online service that hosts Git repositories. This is accomplished by typing `git clone`, followed by the repository URL.

Next, we'll check out a branch that's been created specifically for this video. If you're not familiar with Git Branching, just know that it's a version of the repository that contains the files that we want.

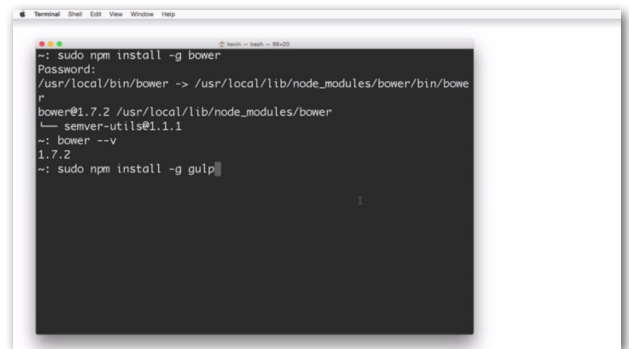
Next, we'll use NPM Install, which downloads other files that our sample web page depends on. After that, the command `bower install` will download all of our client side assets for us. And finally, the command `gulp serve` will start our local web server. Once we see that our local web server is running, we can type `localhost:5000` in any browser in order to see the sample web page.

I'll demonstrate these steps right now. Open up your terminal application. If you're using Windows, you'll need to use Git Bash. In your terminal application, type `cd ~/Desktop`.

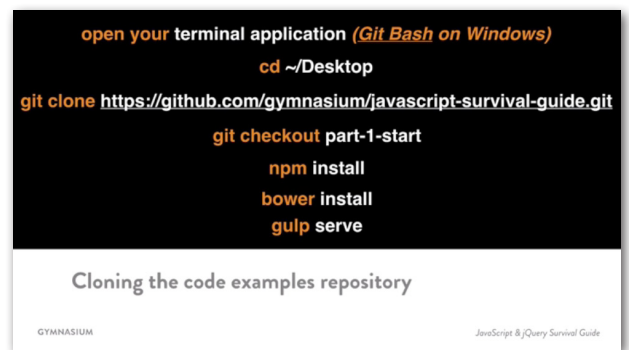
Now I'll note that in most cases, your terminal prompt will probably look different than mine. I've customized my terminal prompt for simplicity. You can ignore the difference between our prompts and we can just focus on the commands that we execute.

Next, we need to clone the project repository. In your terminal application, type `git clone`, and the URL of the project repository. When you do that, it will create a new folder on your desktop that contains the project repository.

Next, we need to move into that folder. Now depending on your operating system, you can very likely use a little shortcut that will save some time. Normally, you would type `cd`, space, and the path to that folder, which will differ on your computer than it will on mine.



```
Terminal Shell Edit View Window Help
~: sudo npm install -g bower
Password:
/usr/local/bin/bower -> /usr/local/lib/node_modules/bower/bin/bower
bower@1.7.2 /usr/local/lib/node_modules/bower
└─ semver-utils@1.1.1
~: bower --v
1.7.2
~: sudo npm install -g gulp
```



open your terminal application (*Git Bash on Windows*)

`cd ~/Desktop`

`git clone https://github.com/gymnasium/javascript-survival-guide.git`

`git checkout part-1-start`

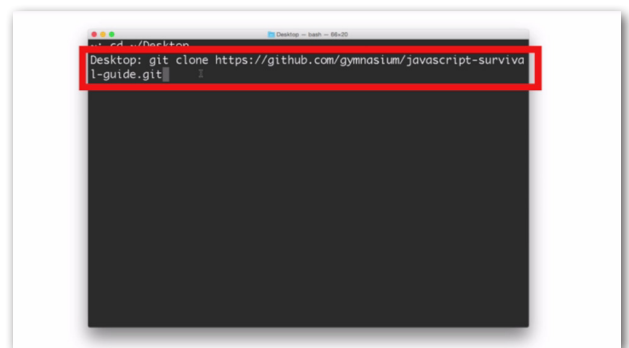
`npm install`

`bower install`

`gulp serve`

Cloning the code examples repository

GYMNASIUM JavaScript & jQuery Survival Guide



```
Desktop: git clone https://github.com/gymnasium/javascript-survival-guide.git
```

You should be able to do this by simply typing `cd`, space, and then dragging that folder from your desktop to your terminal application, which moves us into that folder.

Next, we need to check out the branch that contains the code samples that we want. So in your terminal application, type `git checkout part-1-start`. That moved to the branch that we want.

Next, we need to do an NPM installation. In your terminal application, type `npm install`. If you receive an error after typing this command, this means that you have not installed Node yet. And as I mentioned earlier, that is a requirement.

If you need to install Node, simply go to [nodejs.org](https://nodejs.org). You may see a few warnings during the NPM installation. You can very likely ignore those warnings.

Next, we need to install the Bower components. In your terminal application, type `bower install`. After the bower installation is complete, we need to start our local web server. In your terminal application, type `gulp serve`. Once we see that our local web server is running, we can take a look at the example web page in our browser.

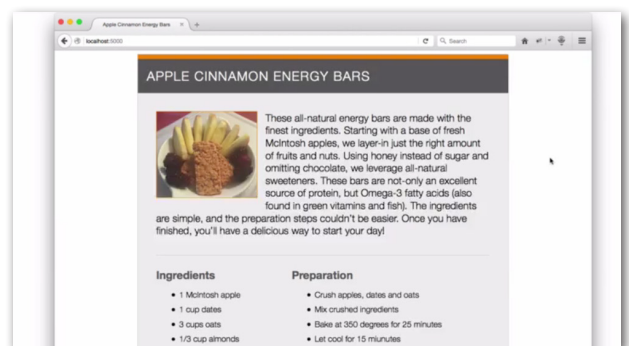
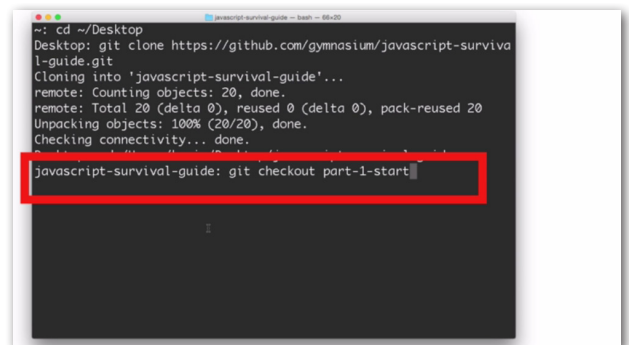
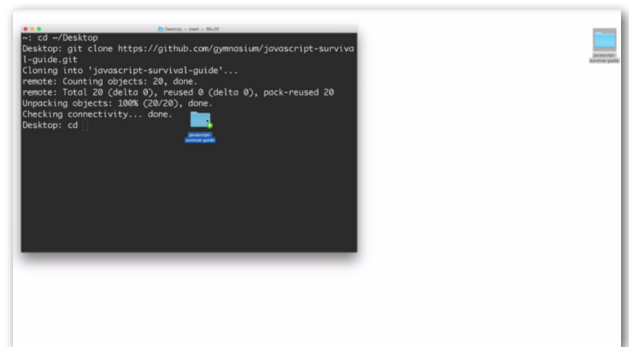
Open up your browser and in the address bar type `localhost:5000`. Then you'll see the example web page that it will be using to demonstrate the code examples for the rest of this course. Now that you have everything you need, let's begin coding.

In the next chapter, you'll begin with a review of how to effectively use the JavaScript console. See you soon.

## REVIEWING THE JAVASCRIPT CONSOLE

In this chapter, I'll provide an introduction to the JavaScript console. If you're not familiar with this concept, the JavaScript console is an extension of the browser. It allows you to execute JavaScript in real time without the need to add your code to a file, save it, and then refresh the web page.

The reason why I chose to include this topic is because of the tremendous power that the JavaScript console offers. As you continue to learn about front end web development, particularly JavaScript, I think you'll find that this is an invaluable asset. I'll demonstrate how to access the JavaScript console in all of the major browsers. And then for the rest of this course, I'll use it to demonstrate most of the code examples.



First, I'll cover Google Chrome, Apple Safari, and Microsoft Internet Explorer, all of which have a built-in JavaScript console. When I discuss Mozilla Firefox, I'll demonstrate installation of Firebug, a popular add-on which features a number of developer tools, including a JavaScript console. Now, while Firefox does have a built-in suite of developer tools, most front end web developers prefer Firebug, so I'll focus on that add-on.



I'll be using some of the code examples in this chapter. So let's take a quick look at where to find them. In the last chapter, we cloned the project repository that contains the code examples we need. There should be a folder on your desktop named JavaScript Survival Guide.

Open that folder. Open the subfolder named Code Examples. This is where you'll find all of the code examples for this course. Using your favorite text editor, open the file named JavaScript-console. These are the code examples we'll use in this chapter.

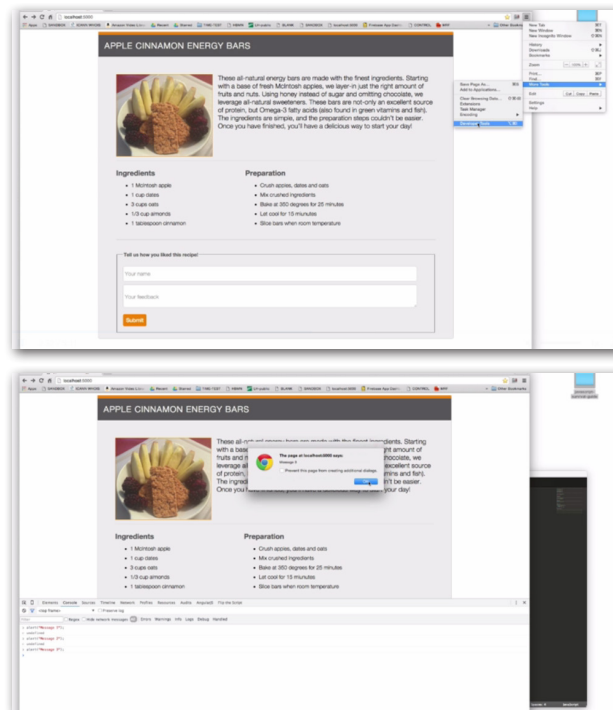
Google Chrome's JavaScript console is part of its developer tools, also known as DevTools. Chrome DevTools is a feature that makes front end web development work much easier. We'll focus on the JavaScript console. But if you're not familiar with Chrome DevTools, I recommend that you take a deeper dive when time permits. Chrome DevTools makes a significant difference in your troubleshooting efforts when working on a web page.

Open Google Chrome, and then click this icon you see in the upper right hand corner of the browser. This is sometimes referred to as the hamburger menu. Click More Tools and then click Developer Tools.

By default, Chrome's JavaScript console operates in single-line mode. This means that each time you input a line of JavaScript, and then press the Enter or Return key, that JavaScript is executed. Let me demonstrate this. Open up the code examples in your text editor and copy the first alert message that you see. Paste that alert message into the input area of Chrome DevTools and then press the Enter or Return key, and you'll see that code executed.

Go back and copy the second alert message and paste it into the input area and then press the Enter key. And you'll see that code executed. And then follow the same steps for the third alert message.

In each case, when you press the Enter or Return key, the code is executed. That's called single-line mode. Now you can operate in something called multi-line mode, and what that means is you can enter multiple lines of JavaScript before any of them are executed. Go back to your code examples and copy the first alert message. I'm just going to clear the console here.



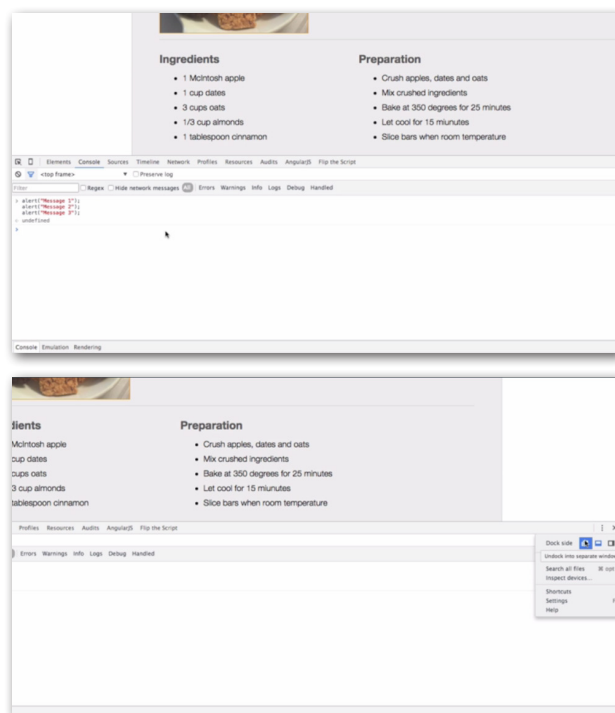


OK, paste that alert message in, but don't press the Enter key yet. Hold down the Shift key and then press Enter. Now you notice my JavaScript was not executed. My cursor went down to the next line. Go back to your code examples and copy the second alert message and paste it into the input area, hold down the Shift key, and then press the Enter key, and then go back and copy the third alert message.

Paste it into the input area and now simply press the Return key. Our code now executes. In this case, it's three alert messages. So we're seeing one alert after the other. So that's the difference between multi-line and single-line mode. In either mode, you can use the up and down arrow keys to access previously entered code.

The JavaScript console can be repositioned in Chrome DevTools. Click the three dots you see in the right side of DevTools. And in the submenu, you have options to reposition the JavaScript console to the right side or in a detached window. I'm going to move it to the right side. And now I'll use the same steps to move it to a detached window. And you can always return it to its default position, which is the bottom.

That's a very basic introduction to the Chrome development tools. As I mentioned earlier, in the upcoming chapters we'll explore how you can access these tools in other browsers—specifically, Safari, Internet Explorer, and then Firefox. If Chrome is your favorite browser, feel free to skip those videos and jump right to Chapter 7 where I'll be walking through a quick exercise using JavaScript expressions.



## APPLE SAFARI DEVELOPER TOOLS

Apple Safari's JavaScript console is part of its Web Inspector, which, much like Chrome DevTools, provides a set of powerful debugging features. In fact, it's similar to Chrome DevTools, so if you use both Chrome and Safari, you're likely to find that the experience is consistent in a number of areas.

In order to access Safari's Web Inspector, you'll first need to make sure that the Develop menu is visible. With Safari open, click Safari, Preferences. In the Preferences dialog, click the gear icon where you see the word Advanced. In the Advanced tab at the bottom, click the checkbox that says "show develop menu in menu bar" and close the Preferences dialog. Click the Develop menu and then click Show Error Console.

The Safari Web Inspector operates in single line mode by default. Open up your code examples in your Text Editor, and once again, copy and paste the alert messages. In each case, paste the message and then press the Enter key and you will see the results of the code that you had entered.



I'm going to press Enter and I see the result of the code, and then one more time with message number three.

Now, similar to Chrome DevTools, the Safari Web Inspector allows you to operate in multi-line mode. So let's follow the same steps again for multi-line mode. Copy the alert messages, and in each case when you paste it in, hold down the Shift key before you press the Enter key and you'll see that it allows you to enter multiple lines of code before executing them. Now I'm pasting in the second alert message. Hold down the Shift key and press Enter, and then here's the third alert message.

And in this case, I'm going to press the Enter key without holding down Shift and it executes all the code in one shot. So I see all three alert messages. Use the up and down arrow keys to access previously entered lines of code.

You can reposition the JavaScript console by clicking this Multi Window icon on the left side. By default, that click always opens it in a detached window. Then you have the option of returning the JavaScript console to the bottom or positioning to the right. I'll position it to the right now. And then, following the exact same steps, I'll position it at the bottom, which is the default location.

That's it for Safari's development tools for now. In the next chapter, you'll take a quick look at Internet Explorer.

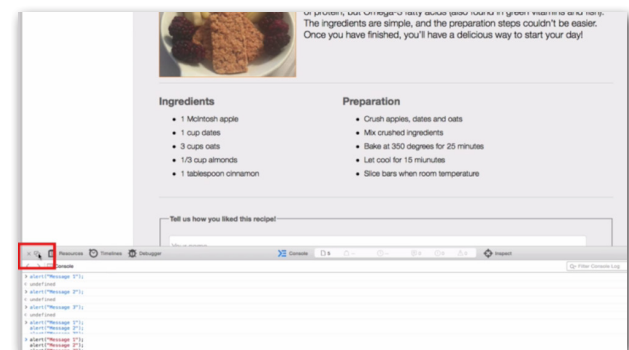
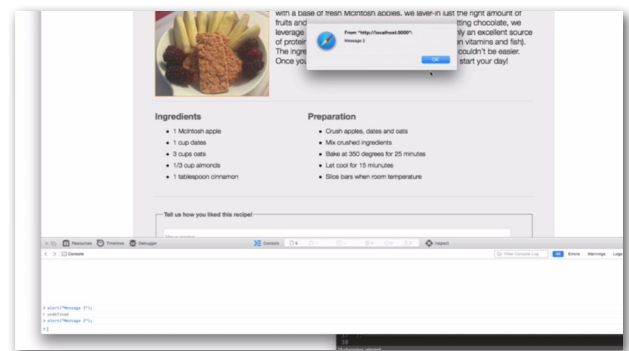
## MICROSOFT IE DEVELOPER TOOLS

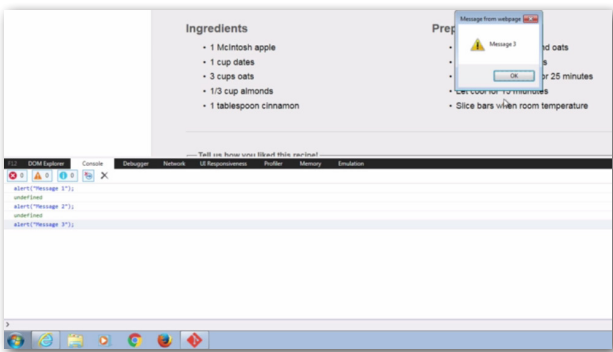
Microsoft Internet Explorer has a feature named Developer Tools, which contains a JavaScript console. Conceptually, Internet Explorer's Developer Tools has the same purpose as Chrome Dev Tools and Safari's Web Inspector, but the look and feel differs.

In order to access the Developer Tools JavaScript console, click the gear? Icon you see in the upper right hand corner of the browser, and then click F12 Developer Tools. The Developer Tool JavaScript console defaults to single line mode. Copy and paste the three alerts from your code examples, just as we did previously.

In each case, when you press the Enter or Return key, the code that you have inputted will be executed.

You can also operate in Multi-line mode by clicking the double up arrow in the lower right hand corner. Now, when I paste in multiple lines of code, I can press the Enter key without the code being executed. So I'm going to paste in one more alert.





And you can see, it's very easy to edit the code here. So now that I've entered multiple lines of code, in the lower right hand corner, you can press the right arrow or Play button, and that will execute all of the code you've entered. And there's our three alert messages.

Now, when I go back to Single Line mode, you can use the up and down arrows to access previously entered lines of JavaScript.

You can also reposition the JavaScript console by clicking the Multiple Window icon you see on the right. There are two options—Detached mode, so now the JavaScript console's in a detached window. Or you can return it to its default position, which is the bottom.



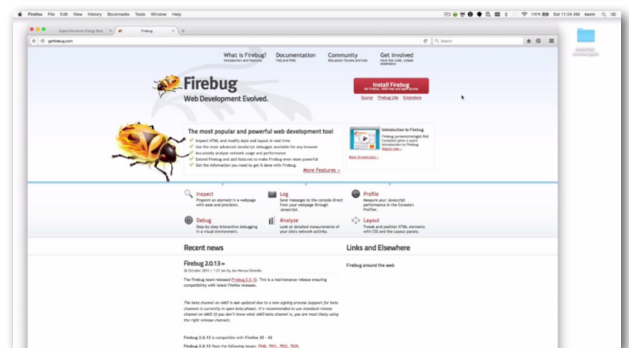
That's it for IE's development tools for now. In the next chapter, you'll look at Firefox's Firebug Developer toolbar.

## FIREFOX FIREBUG DEVELOPER TOOLS

While of the most recent versions of Mozilla Firefox have a JavaScript console built in, the majority of front-end web developers prefer Firebug. You're welcome to use the built-in Firefox JavaScript console. But for this course, I'll cover how to install Firebug, and then I'll use that add-on to demonstrate the code examples.

When using Firefox, Firebug can be installed by visiting [getfirebug.com](http://getfirebug.com). Open Firefox. Open a new tab. And then in that tab, go to [getfirebug.com](http://getfirebug.com). Click the Install Firebug button.

You'll be taken to a new page. Where you see current stable, click the Download link. You'll be taken to a new page again. Click the Add To Firefox button. When prompted, click the Install button, and you'll see a message indicating that the Firebug installation was successful. You can close this tab.



In the upper right-hand corner, click the Firebug icon. Firebug defaults to single-line mode. Use the exact same steps as you did previously to copy and paste the code examples in single-line mode. In each case, you'll see that when you press the Enter key, the code that you inputted is executed immediately.

In single-line mode, you can use the up and down arrow keys to access previously entered code. Firebug features a multi-line mode. Click the double-up arrow in the lower right-hand corner, and you'll see the multi-line input area.

Go back to your code examples and copy all three alert messages. Paste those alert messages in the multi-line input area, and then click the Run button. You'll see that all the code is executed at once.

There's also a History link which is very helpful. For example, I'll enter a few more commands, and then I'll click the History link, which allows us to access previously entered code. When you're ready to return to single-line mode, click the right arrow.

Firebug can be repositioned by clicking the Firebug icon and then click Firebug Location or Firebug UI Location. You can move it to a detached window. And you can also move the Firebug location to the top, to the left, to the right. And then you can return it to the bottom, which is the default location.

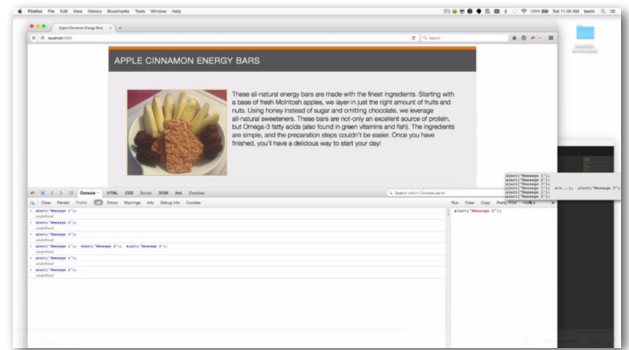
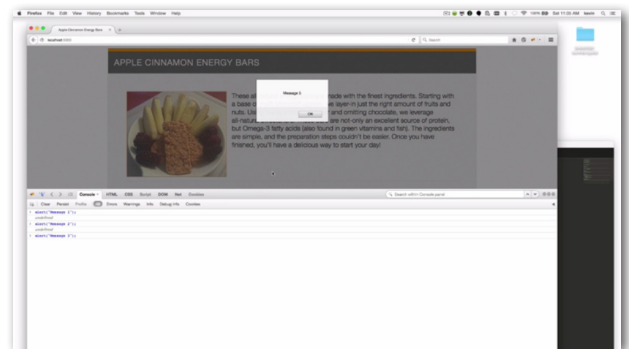
That's our quick look at Firefox Firebug. A very important note I'd like to mention is that at the time of this recording, this plugin was actively being developed. However, recently, the developers have decided to integrate it directly into the browser, and so Firebug at some point will cease to exist.

If Firefox is your browser of choice, be sure to look in the forum for this course, and we'll post any updates to this tutorial as needed. In the next chapter, we're going to do a short but important review of how to interpret JavaScript expressions within the console.

## JAVASCRIPT EXPRESSIONS REVIEW

Before we move on to the next chapter, there are a few things I'd like to point out about the JavaScript console. Sometimes, developers are thrown off by some of the output they see. If you're just getting started with the JavaScript console, this could be confusing. In order to clear this up, let's review some basics about JavaScript expressions.

An expression in JavaScript is simply a line of code that produces a result. For example, if I type 1 plus 1 in my console input area and click Run, I'll see the number 2 in



the console output area. That's because the result of this expression is 2. Notice that I didn't type 1 plus 1 equals 2, I typed 1 plus 1. And when I executed that code, I saw the result of that expression, which is 2. I could have easily typed 1 plus 2, in which case the result of that expression would be the number 3.

Open up your code examples and copy the function Get Number. Paste that function into your input area and execute the code by clicking the Run button. Now you should see the number 123 in your console. You see that number because we have an expression, and that expression is a function. That function returns a number—that number is 123. So the result of our expression is the number 123, and the console has outputted the result of that expression.

In your code examples, copy and paste the function Get Hello. When you execute this code, you'll see the word Hello in your console, and that's because we have an expression—this expression is a function that returns the word Hello. So the result of that expression is outputted in the console. It's the word Hello.

In your code examples, copy the function Do Nothing. When you execute this code, something a little bit different happens. You'll notice that the word Undefined appears in the console. The reason for this is, in JavaScript, when a function explicitly returns no value or when it does not return an explicit value, it returns the value Undefined. So in this case, we have an expression—that expression is a function—the function does not explicitly return a value, which means it returns the value of Undefined.

In your code examples, copy the function Return Nothing. In this case, we have a function that does some things—the things are not particularly valuable to us, but it does things. But like the previous function, the Return Nothing function does not explicitly return a value, which means it returns Undefined. When you execute this code, you'll see the value Undefined in your console.

So, just keep in mind that the JavaScript console will always indicate the result of an expression. That expression could be a simple line of code—like 1 plus 1, which equals 2—or it could be a function that may do things but not explicitly return a value. And when a function does not return a value, the result is Undefined.

In the following chapters, we'll be spending a lot of time in the JavaScript console. Hopefully, you now have a clear understanding of the output that you'll see when we execute each code example. That's it for this lesson. In the next lesson, you'll continue to rely heavily on the JavaScript console as we begin to explore basic ways to use JQuery to add interactivity to your projects.

