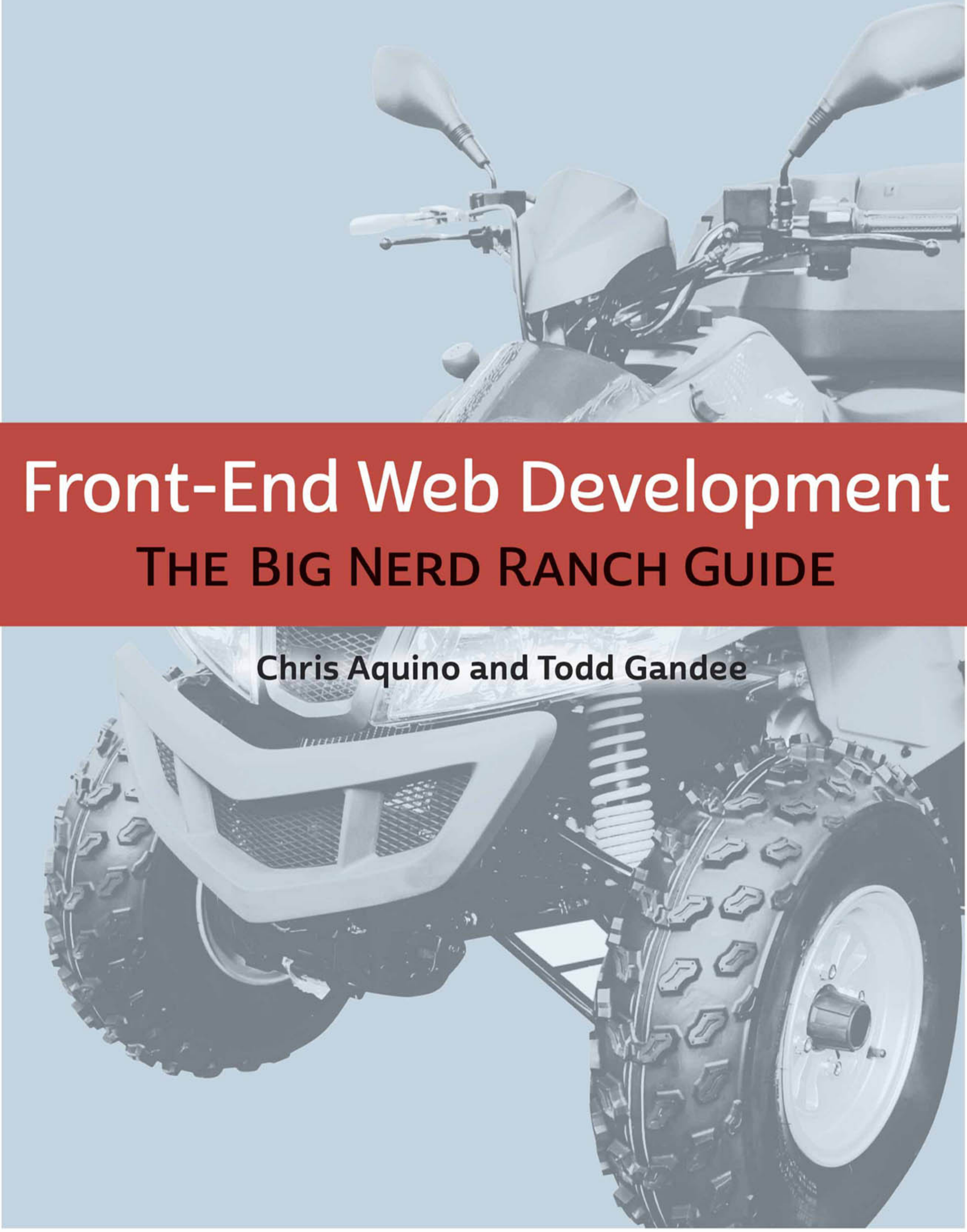
Big Nerd Ranch

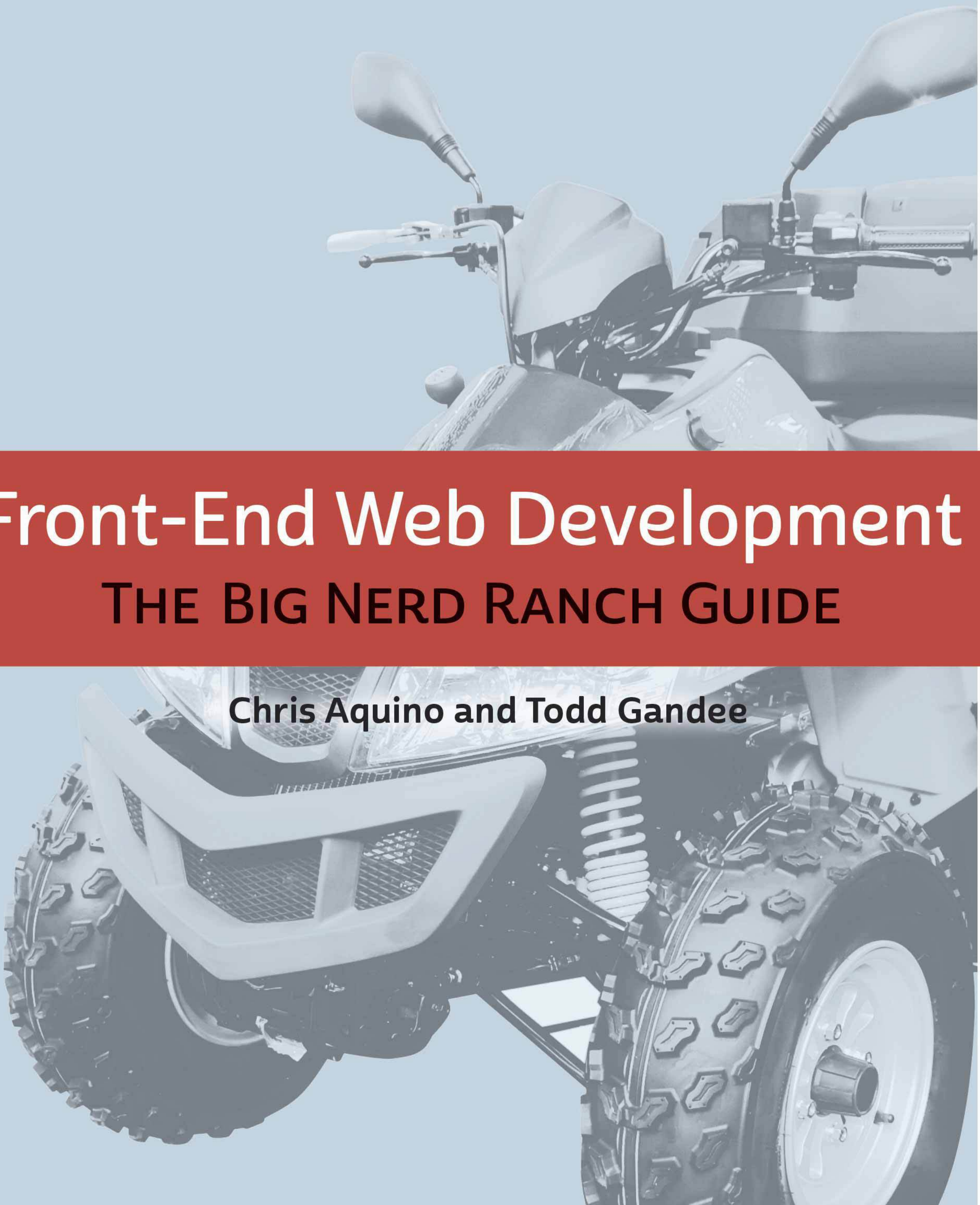


Front-End Web Development

The Big Nerd Ranch Guide

Chris Aquino and Todd Gandee

Big Nerd Ranch



Front-End Web Development

THE BIG NERD RANCH GUIDE

Chris Aquino and Todd Gandee

Front-End Web Development: The Big Nerd Ranch Guide

by Chris Aquino and Todd Gandee

Big Nerd Ranch, LLC
200 Arizona Ave NE
Atlanta, GA 30307
(770) 817-6373
http://www.bignerdranch.com/
book-comments@bignerdranch.com

Exclusive worldwide distribution of the English edition of this book by

Pearson Technology Group
800 East 96th Street
Indianapolis, IN 46240 USA
http://www.informit.com

ISBN-10 0134432576
ISBN-13 978-0134432571

First edition, first printing, July 2016
Release D.1.1.1

Dedication

To Mom and Dad, for buying us that computer. To Dave and Glenn, for letting your little brother completely hog it. And to Angela, for giving me a life away from the screen.

— C.A.

To my mom and dad, thank you for giving me room to find my own way. To my wife, thank you for loving a nerd.

— T.G.

Acknowledgments

As authors, we can take full credit for typing the words and creating the diagrams. (Yay, us!) But the whole truth is that we would still be staring at a blank page if not for the efforts of an army of contributors, collaborators, and mentors.

- Aaron Hillegass, for believing that the two of us could produce a work worthy of the Big Nerd Ranch name. Thank you for your immeasurable faith and support.
- Matt Mathias, for guiding us through the development of this book, especially during the crucial last mile. You made sure that time that would have been spent watching cat videos or Downton Abbey reruns was instead dedicated to writing.
- Brandy Porter, for the care and (literal) feeding of the authors on numerous occasions. You worked your magic behind the curtain, orchestrating a sequence of events that made finishing this work possible. Thank you.
- Jonathan Martin, our coinstructor and language maven. Thank you for enthusiastically teaching the in-progress course materials on which this book is based and offering thoughtful criticism throughout the many revisions.
- Our proofreaders, technical reviewers, and guinea pigs: Mike Zornek, Jeremy Sherman, Josh Justice, Jason Reece, Garry Smith, Andrew Jones, Stephen Christopher, and Bill Phillips. Thank you for volunteering as tribute.
- Elizabeth Holaday, our infinitely patient and reassuring editor. Thank you for breaking us out of the echo chamber, being the voice of reason, and reminding us always of our readers.
- Ellie Volckhausen, who designed our cover.
- Simone Payment, our proofreader, who kept things consistent.
- Chris Loper at IntelligentEnglish.com, who designed and produced the print and ebook versions of the book. His DocBook toolchain made life much easier, too.

Lastly, thank you to the countless students who have taken the week-long training. Without your curiosity and your questions, none of this matters. This work is a reflection of the insight and inspiration you have given us over the span of those many weeks. We hope the otters made the training a little lighter.

Table of Contents

Introduction

Learning Front-End Web Development

Doing front-end web development may require a shift in perspective, as it is a very different animal from development for other platforms. Here are a few things to keep in mind as you are learning.

The browser is a platform.

Perhaps you have done native development for iOS or Android; written server-side code in Ruby or PHP; or built desktop applications for OS X or Windows. As a front-end developer, your code will target the browser – a platform available on nearly every phone, tablet, and personal computer in the world.

Front-end development runs along a spectrum.

At one end of the spectrum is the look and feel of a web page: rounded corners, shadows, colors, fonts, whitespace, and so on. At the other end of the spectrum is the logic that governs the intricate behaviors of that web page: swapping images in an interactive photo gallery, validating data entered into a form, sending messages across a chat network, etc. You will need to become proficient with the core technologies all along this spectrum, and you will often need to use multiple technologies in synergy to create a good web application.

Web technologies are open.

There is no one company that controls how browsers should work. That means that front-end developers do not get a yearly SDK release that contains all the changes they will need to deal with for the next twelve months. Native platforms are a frozen pond on which you can comfortably skate. The web is a river; it curves, moves quickly, and is rocky in some places – but that is part of its appeal. The web is the most rapidly evolving platform available. Adapting to change is a way of life for a front-end developer.

This book's purpose is to teach you how to develop for the browser. As you follow this guide, you will be taken through the process of building a series of projects. Each project will call for a different mixture of technologies along the front-end spectrum. Because of the sheer number of front-end tools, libraries, and frameworks available, this book will focus on the most essential and portable patterns and techniques.

Prerequisites

This book is not an introduction to programming. It assumes you have experience with the fundamentals of writing code. You are expected to be familiar with basic types, functions, and objects.

That said, it also does not assume you already know JavaScript. It introduces you to JavaScript concepts in context, as you need them.

How This Book Is Organized

This book walks you through writing four different web applications. Each application has its own section of the book. Each chapter in a section adds new features to the application you are building.

Doing the work of building these four applications takes you from one extreme of the front-end spectrum to the other.

Ottergram	In your first project, you will create a web-based photo gallery. Building Ottergram will teach you the fundamentals of programming for the browser using HTML, CSS, and JavaScript. You will build the user interface manually, learning how the browser loads and renders content.
CoffeeRun	Part coffee order form, part checklist, CoffeeRun takes you through a number of JavaScript techniques including writing modular code, taking advantage of closures, and communicating with a remote server using Ajax. Your focus will shift from manually creating the UI to creating and manipulating it programmatically.
Chattrbox	Chattrbox has the shortest section and is the most distinct of the apps. You will use JavaScript to build a chat system, writing a chat server with Node.js as well as a browser-based chat client.
Tracker	Your final project uses Ember.js, one of the most powerful frameworks for front-end development. You will create an application that catalogs sightings of rare, exotic, and mythical creatures. Along the way, you will learn your way around the rich ecosystem that powers the Ember.js framework.

As you work through these applications, you will be introduced to a number of tools, including:

- the Atom text editor and some useful plug-ins for working with code
- documentation resources like the Mozilla Developer Network
- the command line, using the OS X Terminal app or the Windows command prompt
- browser-sync
- Google Chrome's Developer Tools
- normalize.css
- Bootstrap
- jQuery and libraries like crypto-js and moment
- Node.js, the Node package manager (npm), and nodemon
- WebSockets and the wscat module
- Babel, Babelify, Browserify, and Watchify
- Ember.js and addons like Ember CLI, Ember Inspector, Ember CLI Mirage, and Handlebars
- Bower
- Homebrew
- Watchman

How to Use This Book

This book is not a reference book. Its goal is to get you over the initial hump to where you can get the most out of the reference and recipe books available. It is based on our five-day class at Big Nerd Ranch, and, as such, it is meant to be worked through from the beginning. Chapters build on each other, and skipping around would be unproductive.

In our classes, students work through these materials, but they also benefit from the right environment – a dedicated classroom, good food and comfortable board, a group of motivated peers, and an instructor to answer questions.

As a reader, you want your environment to be similar. That means getting a good night's rest and finding a quiet place to work. These things can help, too:

- Start a reading group with your friends or coworkers.
- Arrange to have blocks of focused time to work on chapters.
- Participate in the forum for this book at `forums.bignerdranch.com`, where you can discuss the book and find errata and solutions.
- Find someone who knows front-end web development to help you out.

Challenges

Most chapters in this book end with at least one challenge. Challenges are opportunities to review what you have learned and take your work in the chapter one step further. We recommend that you tackle as many of them as you can to cement your knowledge and move from *learning* JavaScript development from us to *doing* JavaScript development on your own.

Challenges come in three levels of difficulty:

- Bronze challenges typically ask you to do something very similar to what you did in the chapter. These challenges reinforce what you learned in the chapter and force you to type in similar code without having it laid out in front of you. Practice makes perfect.
- Silver challenges require you to do more digging and more thinking. Sometimes you will need to use functions, events, markup, and styles that you have not seen before, but the tasks are still similar to what you did in the chapter.
- Gold challenges are difficult and can take hours to complete. They require you to understand the concepts from the chapter and then do some quality thinking and problem solving on your own. Tackling these challenges will prepare you for the real-world work of JavaScript development.

You should make a copy of your code before you work on the challenges for any chapter. Otherwise, the changes that you make may not be compatible with subsequent exercises.

If you get lost, you can always visit `forums.bignerdranch.com` for some assistance.

For the More Curious

Many chapters also have “For the More Curious” sections. These sections offer deeper explanations or additional information about topics presented in the chapter. The information in these sections is not absolutely essential, but we hope you will find it interesting and useful.

Part I

Core Browser Programming

1

Setting Up Your Development Environment

There are countless tools and resources for front-end development, with more being built all the time. Choosing the best ones is challenging for developers of all skill levels. Throughout the projects in this book, we will guide you in the use of some of our favorites.

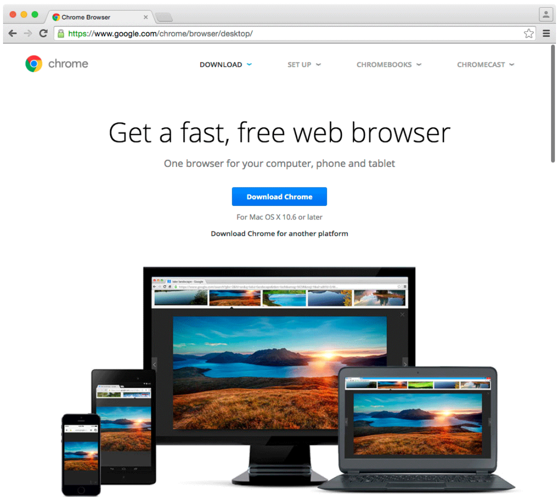
To get started, you will need three basic tools: a browser, a text editor, and good reference documentation for the many technologies used in front-end development. Also, there are several extras that – while not essential – will make your development experience smoother and more enjoyable.

For the purposes of this book we recommend that you use the same software we use to get the most benefit from our directions and screenshots. This chapter walks you through installing and configuring the Google Chrome browser, the Atom text editor, Node.js, and a number of plug-ins and extras. You will also find out about good documentation options and get a crash course in using the *command line* on Mac and Windows. In the next chapter, you will put all these resources to use as you begin your first project.

Installing Google Chrome

Your computer should already have a browser installed by default, but the best one to use for front-end development is Google Chrome. If you do not already have the latest version of Chrome, you can get it from `www.google.com/chrome/browser/desktop` (Figure 1.1).

Figure 1.1 Downloading Google Chrome

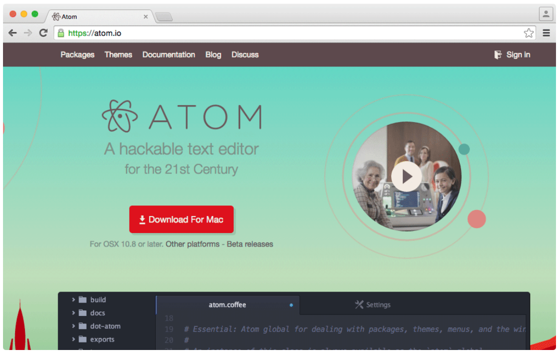

Chrome Browser
https://www.google.com/chrome/browser/desktop/
chrome
DOWNLOAD
SET UP
CHROMEBOOKS
CHROMECAST
Get a fast, free web browser
One browser for your computer, phone and tablet
Download Chrome
For Mac OS X 10.6 or later
Download Chrome for another platform

Installing and Configuring Atom

Of the many text editor programs out there, one of the best for front-end development is the Atom editor by GitHub. It is a good choice because it is highly configurable, has many plug-ins to help with writing code, and is free to download and use.

You can download Atom for Mac or Windows from `atom.io` (Figure 1.2).

Figure 1.2 Downloading Atom



Follow the installation instructions for your platform. After Atom is installed, there are several plug-ins you will want to install as well.

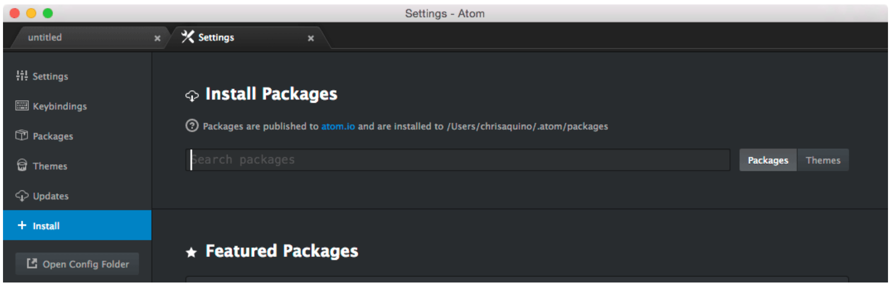
Atom plug-ins

The primary things you want out of your text editor are documentation lookup, autocompletion, code formatting, and code linting (more on that in a bit). Atom gives you some of these features by default, but installing a few plug-ins will make it even better.

Open Atom and reveal its Settings screen. On a Mac, this is done by choosing Atom → Preferences... or using the keyboard shortcut Command-, (that is, the Command key plus the comma). On Windows, you can access it via File → Settings or using the keyboard shortcut Ctrl-,.

On the lefthand side of the Settings screen, click + Install (Figure 1.3).

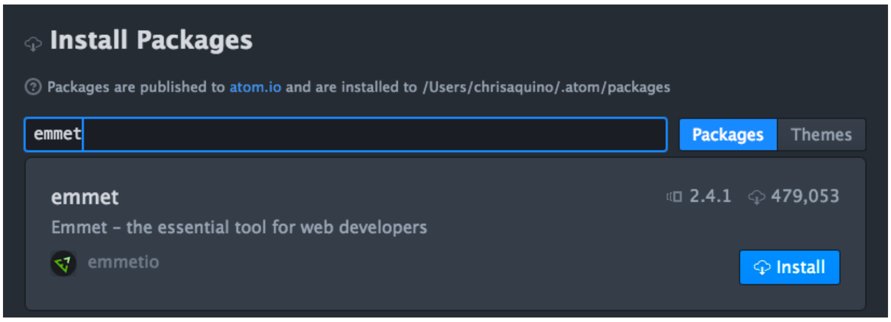
Figure 1.3 Atom's Install Packages screen



Here, you can search for plug-in packages by name. Begin by searching for “emmet.”

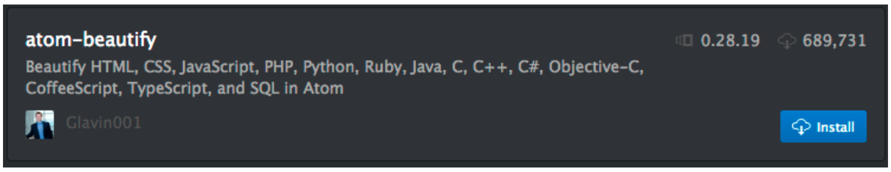
Writing a lot of HTML can be very tedious and is error-prone. The emmet plug-in (Figure 1.4) lets you write well-formatted HTML using a convenient shorthand. Click the Install button to get emmet.

Figure 1.4 Installing emmet


Install Packages
Packages are published to atom.io and are installed to /Users/chrisaquino/.atom/packages
emmet
Packages
Themes
emmet
2.4.1
479,053
Emmet - the essential tool for web developers
emmetio
Install

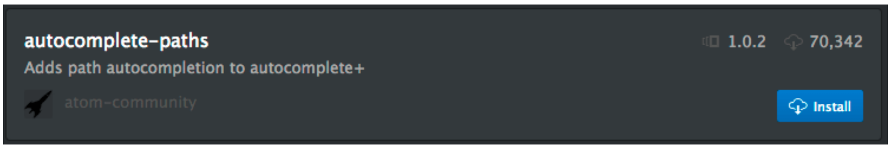
Next, search for “atom-beautify.” The atom-beautify plug-in (Figure 1.5) helps with the indentation of your code, which helps with readability. Again, click Install to get this plug-in.

Figure 1.5 Installing atom-beautify


atom-beautify
0.28.19
689,731
Beautify HTML, CSS, JavaScript, PHP, Python, Ruby, Java, C, C++, C#, Objective-C, CoffeeScript, TypeScript, and SQL in Atom
Glavin001
Install

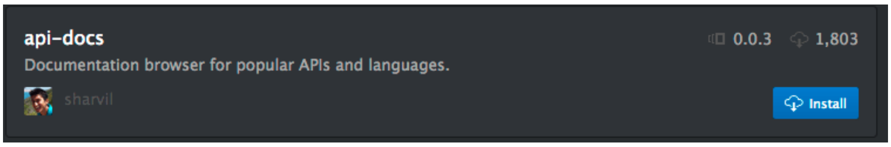
Search for and install the autocomplete-paths plug-in (Figure 1.6). Very often, your code will need to refer to other files and folders in your project. This plug-in helps by offering filenames in an autocomplete menu as you type.

Figure 1.6 Installing autocomplete-paths



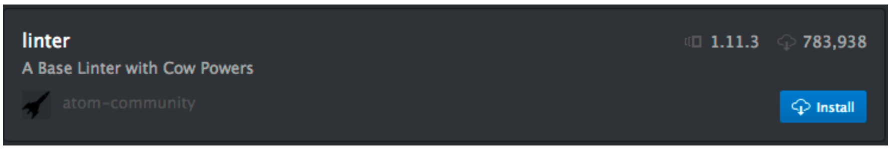
Your next plug-in to install is the api-docs package (Figure 1.7), which lets you look up documentation based on keyword. It displays the documentation in a separate tab in the editor.

Figure 1.7 Installing api-docs



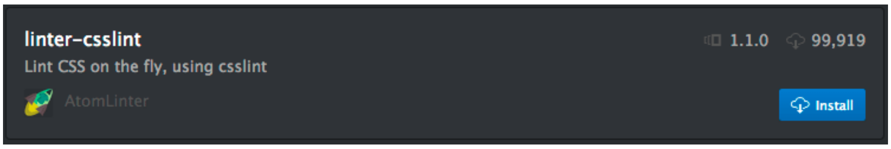
Next, search for and install the linter package (Figure 1.8). A *linter* is a program that checks the syntax and style of your code. Make sure you find and install the package that is just named “linter.” This is a base linter that works with language-specific plug-ins. You will need it in order to use the other linter plug-ins below.

Figure 1.8 Installing linter



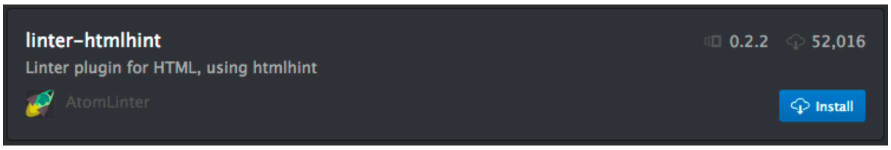
There are three companions to linter that you will want to install to check your CSS, HTML, and JavaScript code. Start with linter-csslint (Figure 1.9), which ensures that your CSS is syntactically correct and also offers suggestions about writing performant CSS.

Figure 1.9 Installing linter-csslint



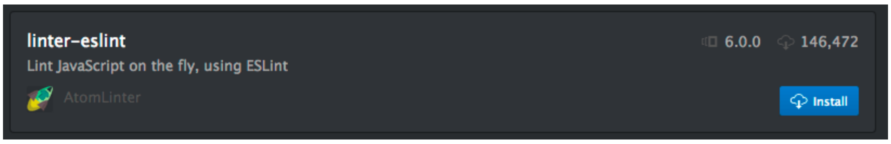
The next linter companion plug-in to install is linter-htmlhint (Figure 1.10), which confirms that your HTML is well formed. It will warn you about mismatched HTML tags.

Figure 1.10 Installing linter-htmlhint



The last linter companion plug-in to install is linter-eslint (Figure 1.11). This plug-in checks the syntax of your JavaScript and can be configured to check the style and formatting of your code (for example, how many spaces lines are indented or how many blank lines come before and after comments).

Figure 1.11 Installing linter-eslint



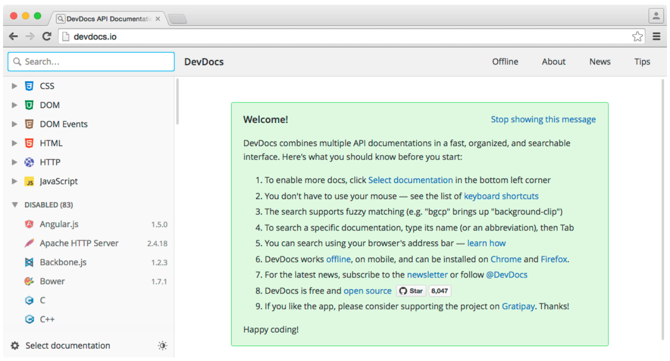
Chrome and Atom are now ready for front-end development. There are just a few more steps to completing your coding environment: accessing documentation, learning command-line basics, and downloading two final tools.

Documentation and Reference Sources

Front-end development is different from programming for platforms like iOS and Android. Aside from the obvious differences, front-end technologies have no official developer documentation other than the technical specifications. This means that you will need to look elsewhere for guidance. We recommend that you familiarize yourself with the resources below and consult them regularly as you work through the book and continue on with front-end development.

The Mozilla Developer Network (MDN) is the best reference for anything to do with HTML, CSS, and JavaScript. One way to access it is `devdocs.io`, an excellent documentation interface (Figure 1.12). It pulls documentation from MDN for core front-end technologies – and it can work offline, so you can check it even when you do not have an internet connection.

Figure 1.12 Accessing documentation via `devdocs.io`



Welcome!

DevDocs combines multiple API documentations in a fast, organized, and searchable interface. Here's what you should know before you start:

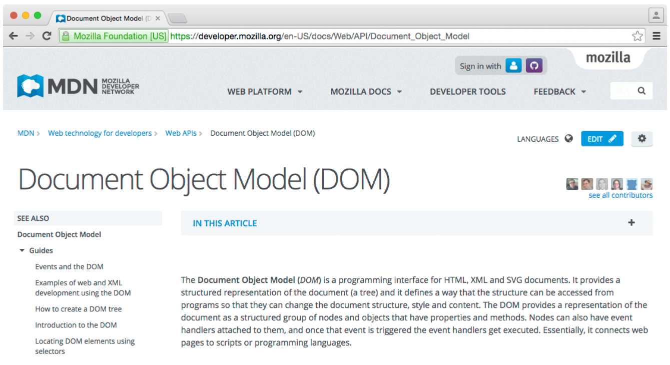
1. To enable more docs, click Select documentation in the bottom left corner
2. You don't have to use your mouse — see the list of keyboard shortcuts
3. The search supports fuzzy matching (e.g. "bgcp" brings up "background-clip")
4. To search a specific documentation, type its name (or an abbreviation), then Tab
5. You can search using your browser's address bar — learn how
6. DevDocs works offline, on mobile, and can be installed on Chrome and Firefox.
7. For the latest news, subscribe to the newsletter or follow @DevDocs
8. DevDocs is free and open source
9. If you like the app, please consider supporting the project on Gratipay. Thanks!

Happy coding!

Note that Safari currently does not support the offline caching mechanism used by `devdocs.io`. You will need to use a different browser, such as Chrome, to access it.

You can also use MDN's website, `developer.mozilla.org/en-US` (Figure 1.13), or simply add "MDN" as a search engine keyword to find the information you need.

Figure 1.13 The Mozilla Developer Network website

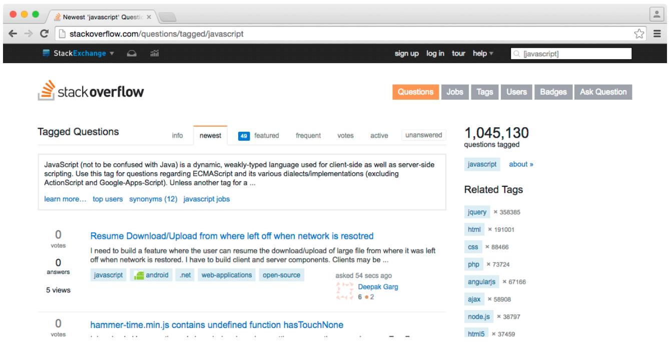

Document Object Model (D ×
Mozilla Foundation [US] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
Sign in with
mozilla
MDN MOZILLA DEVELOPER NETWORK
WEB PLATFORM
MOZILLA DOCS
DEVELOPER TOOLS
FEEDBACK
MDN > Web technology for developers > Web APIs > Document Object Model (DOM)
LANGUAGES
EDIT
see all contributors
SEE ALSO
Document Object Model
Guides
Events and the DOM
Examples of web and XML development using the DOM
How to create a DOM tree
Introduction to the DOM
Locating DOM elements using selectors
IN THIS ARTICLE

Document Object Model (DOM)

The **Document Object Model** (*DOM*) is a programming interface for HTML, XML and SVG documents. It provides a structured representation of the document (a tree) and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods. Nodes can also have event handlers attached to them, and once that event is triggered the event handlers get executed. Essentially, it connects web pages to scripts or programming languages.

Another site to know about is `stackoverflow.com` (Figure 1.14). Officially, this is not a source of documentation. It is a place where developers can ask each other about code. The answers vary in quality, but are often very thorough and quite helpful. So it is a useful resource – as long as you bear in mind that the answers are not definitive, due to its crowdsourced nature.

Figure 1.14 The Stack Overflow website



Tagged Questions

JavaScript (not to be confused with Java) is a dynamic, weakly-typed language used for client-side as well as server-side scripting. Use this tag for questions regarding ECMAScript and its various dialects/implementations (excluding ActionScript and Google-Apps-Script). Unless another tag for a ...

0 votes 0 answers 5 views

Resume Download/Upload from where left off when network is resotred

I need to build a feature where the user can resume the download/upload of large file from where it was left off when network is restored. I have to build client and server components. Clients may be ...

javascript android .net web-applications open-source

asked 54 secs ago Deepak Garg 6 ● 2

0 votes

hammer-time.min.js contains undefined function hasTouchNone

1,045,130 questions tagged

Related Tags

jquery	× 358385
html	× 191001
css	× 88466
php	× 73724
angularjs	× 67166
ajax	× 58908
node.js	× 38797
html5	× 37459

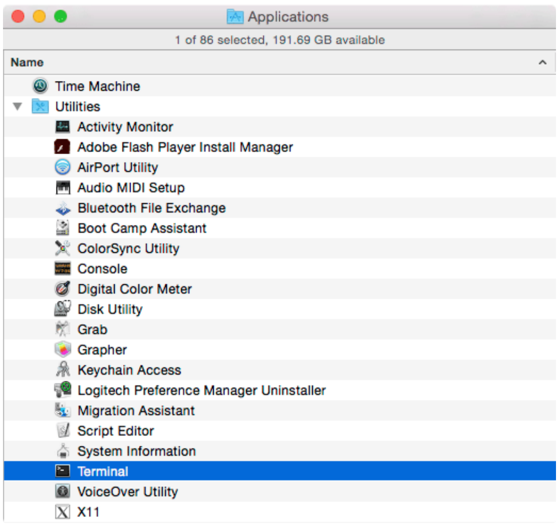
Web technologies are always changing. Support for features and APIs will vary from browser to browser and over time. Two websites that can help you determine which browsers (and which versions of individual browsers) support what features are `html5please.com` and `caniuse.com`. When you need information about feature support, we suggest starting with `html5please.com` to know whether a feature is recommended for use. For more detailed information about which browser versions support a specific feature, go to `caniuse.com`.

Crash Course in the Command Line

Throughout this book, you will be instructed to use the *command line* or *terminal*. Many of the tools you will be using run exclusively as command-line programs.

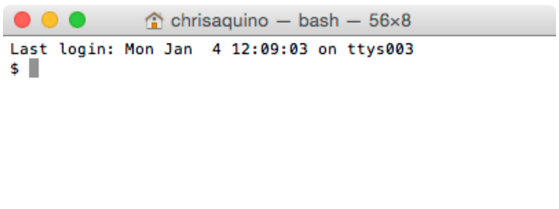
To access the command line on a Mac, open Finder and go to the Applications folder, then the Utilities folder. Find and open the program named Terminal (Figure 1.15).

Figure 1.15 Finding the Terminal app on a Mac



You should see a window that looks like Figure 1.16.

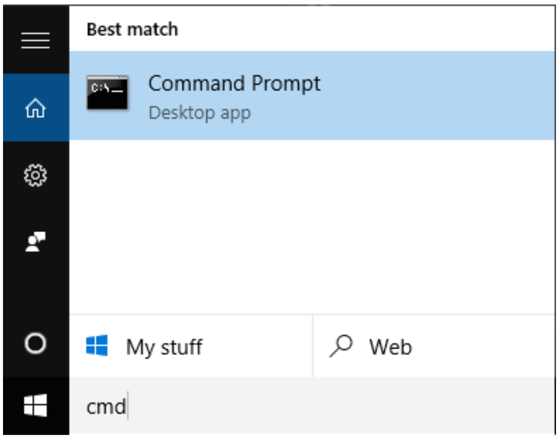
Figure 1.16 Mac command line



```
Last login: Mon Jan  4 12:09:03 on ttys003
$ 
```

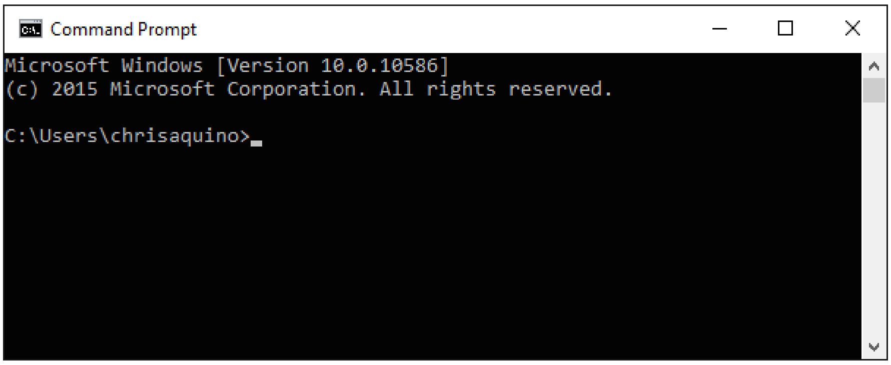
To access the command line on Windows, go to the Start menu and search for “cmd.” Find and open the program named Command Prompt (Figure 1.17).

Figure 1.17 Finding the Command Prompt program on Windows



Click it to run the standard Windows command-line interface, which looks like Figure 1.18.

Figure 1.18 Windows command line



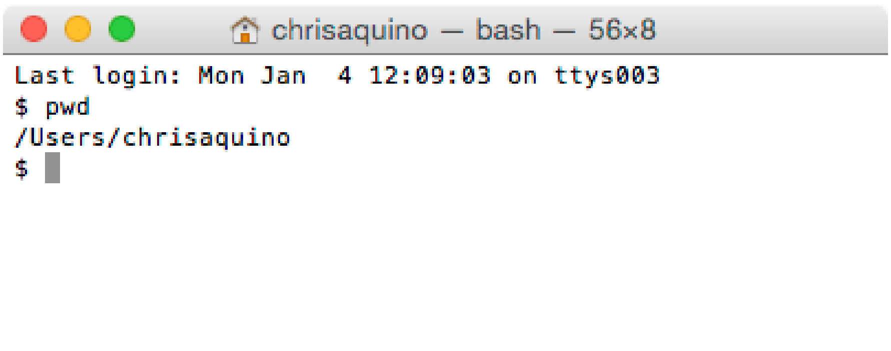
```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\chrisaquino>_
```

From now on, we will refer to "the terminal" or "the command line" to mean both the Mac Terminal and the Windows Command Prompt. If you are unfamiliar with using the command line, here is a short walkthrough of some common tasks. All commands are entered by typing at the prompt and pressing the Return key.

Finding out what directory you are in

The command line is location based. That means that at any given time it is "in" a particular directory within the file structure, and any commands you enter will be applied within that directory. The command-line prompt shows an abbreviated version of the directory it is in. To see the whole path on a Mac, enter the command `pwd` (which stands for "print working directory"), as in Figure 1.19.

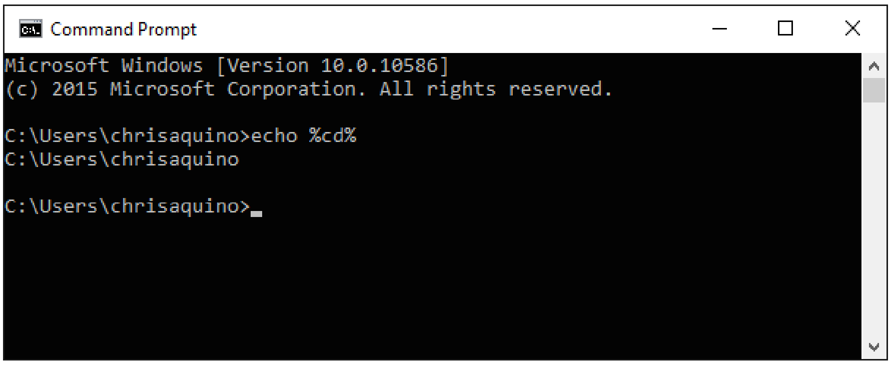
Figure 1.19 Showing the current path using pwd on a Mac



```
Last login: Mon Jan  4 12:09:03 on ttys003
$ pwd
/Users/chrisaquino
$ 
```

On Windows, use the command `echo %cd%` to see the path, as in Figure 1.20.

Figure 1.20 Showing the current path using `echo %cd%` on Windows



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\chrisaquino>echo %cd%
C:\Users\chrisaquino

C:\Users\chrisaquino>_
```

Creating a directory

The directory structure of front-end projects is important. Your projects can grow quickly, and it is best to keep them organized from the beginning. You will create new directories regularly during your development. This is done using the `mkdir` or “make directory” command followed by the name of the new directory.

To see this command in action, set up a directory for the projects you will build as you work through this book. Enter this command:

```
mkdir front-end-dev-book
```

Next, create a new directory for your first project, Ottergram, which you will begin in the next chapter. You want this new directory to be a subdirectory of the `front-end-dev-book` directory you just created. You can do this from your home directory by prefacing the new directory name with the name of the projects directory and, on a Mac, a slash:

```
mkdir front-end-dev-book/ottergram
```

On Windows, you use the backslash instead:

```
mkdir front-end-dev-book\ottergram
```

Changing directories

To move around the file structure, you use the command `cd`, or “change directory,” followed by the path of the directory you want to move into.

You do not always need to use the complete directory path in your `cd` command. For example, to move down into any subdirectory of the directory you are in, you simply use the name of the subdirectory. So when you are in the `front-end-dev-book` directory, the path of the `ottergram` folder is just `ottergram`.

Move into your new project directory:

```
cd front-end-dev-book
```

Now, you can move into the `ottergram` directory:

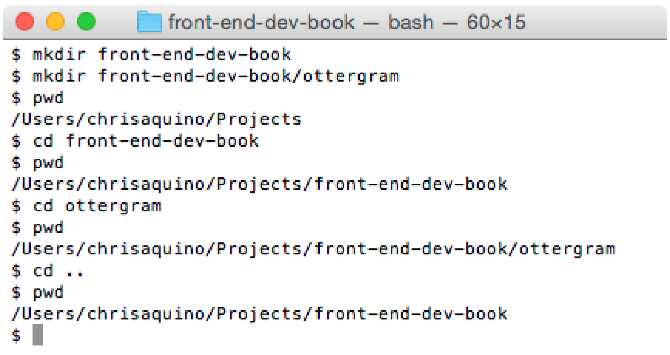
```
cd ottergram
```

To move up to the parent directory, use the command `cd ..` (that is, `cd` followed by a space and two periods). The pair of periods represents the path of the parent directory.

```
cd ..
```

Remember that you can check your current directory by using the `pwd` command (or `echo %cd%` on Windows). Figure 1.21 shows the author creating directories, moving between them, and checking the current directory.

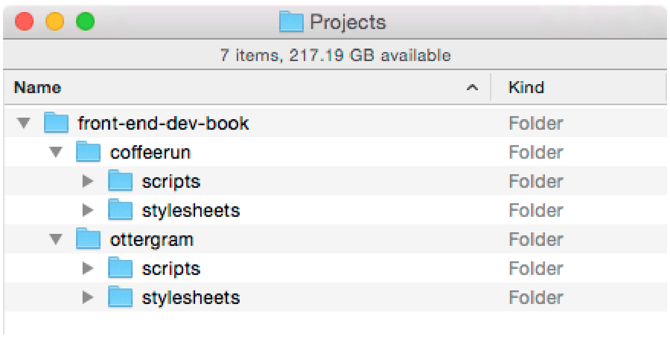
Figure 1.21 Changing and checking directories



```
$ mkdir front-end-dev-book
$ mkdir front-end-dev-book/ottergram
$ pwd
/Users/chrisaquino/Projects
$ cd front-end-dev-book
$ pwd
/Users/chrisaquino/Projects/front-end-dev-book
$ cd ottergram
$ pwd
/Users/chrisaquino/Projects/front-end-dev-book/ottergram
$ cd ..
$ pwd
/Users/chrisaquino/Projects/front-end-dev-book
$ 
```

You are not limited to moving up or down one directory at a time. Let's say that you had a more complex directory structure, like the one shown in Figure 1.22.

Figure 1.22 An example file structure



Name	Kind
▼ front-end-dev-book	Folder
▼ coffeerun	Folder
▶ scripts	Folder
▶ stylesheets	Folder
▼ ottergram	Folder
▶ scripts	Folder
▶ stylesheets	Folder

Suppose you are in the `ottergram` directory and you want to go directly to the `stylesheets` directory inside of `coffeerun`. You would do this with `cd` followed by a path that means "the `stylesheets` directory inside the `coffeerun` directory inside the parent directory of where I am now":

```
cd ../coffeerun/stylesheets
```

On Windows, you would use the same command but with backslashes:

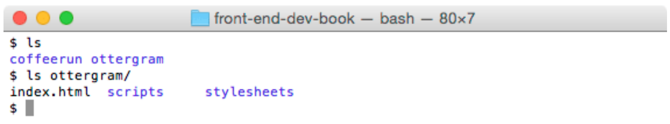
```
cd ..\coffeerun\stylesheets
```

Listing files in a directory

You may need to see a list of files in your current directory. On a Mac, you use the `ls` command for that (Figure 1.23). If you want to list the files in another directory, you can supply a path:

```
ls
ls ottergram
```

Figure 1.23 Using `ls` to list files in a directory



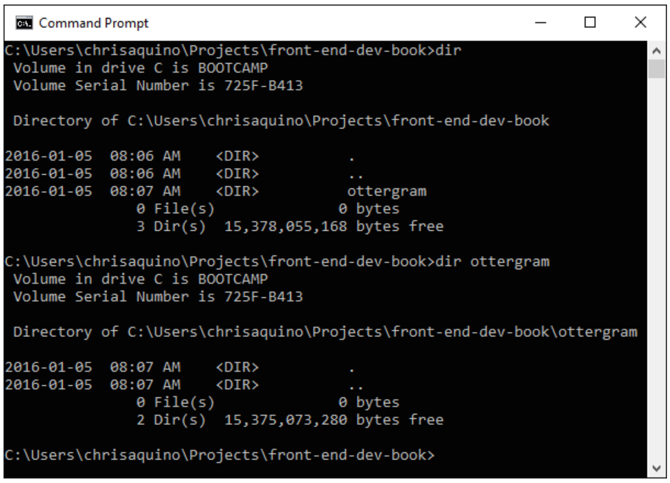
```
$ ls
coffeerun ottergram
$ ls ottergram/
index.html  scripts     stylesheets
$ 
```

By default, `ls` will not print anything if a directory is empty.

On Windows, the command is `dir` (Figure 1.24), which you can optionally give a path:

```
dir
dir ottergram
```

Figure 1.24 Using `dir` to list files in a directory



```
C:\Users\chrisaquino\Projects\front-end-dev-book>dir
 Volume in drive C is BOOTCAMP
 Volume Serial Number is 725F-B413

 Directory of C:\Users\chrisaquino\Projects\front-end-dev-book

2016-01-05  08:06 AM    <DIR>          .
2016-01-05  08:06 AM    <DIR>          ..
2016-01-05  08:07 AM    <DIR>          ottergram
               0 File(s)              0 bytes
               3 Dir(s)  15,378,055,168 bytes free

C:\Users\chrisaquino\Projects\front-end-dev-book>dir ottergram
 Volume in drive C is BOOTCAMP
 Volume Serial Number is 725F-B413

 Directory of C:\Users\chrisaquino\Projects\front-end-dev-book\ottergram

2016-01-05  08:07 AM    <DIR>          .
2016-01-05  08:07 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  15,375,073,280 bytes free

C:\Users\chrisaquino\Projects\front-end-dev-book>
```

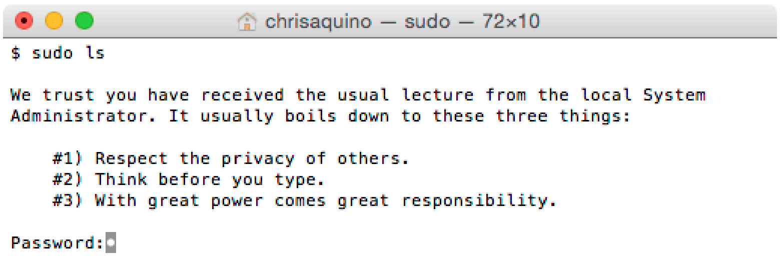
By default, the `dir` command will print information about dates, times, and file sizes.

Getting administrator privileges

On some versions of OS X and Windows, you will need *superuser* or administrator privileges in order to run some commands, such as commands that install software or make changes to protected files.

On a Mac, you can give yourself privileges by prefixing a command with `sudo`. The first time you use `sudo` on a Mac, it will give you a stern warning, shown in Figure 1.25.

Figure 1.25 sudo warning

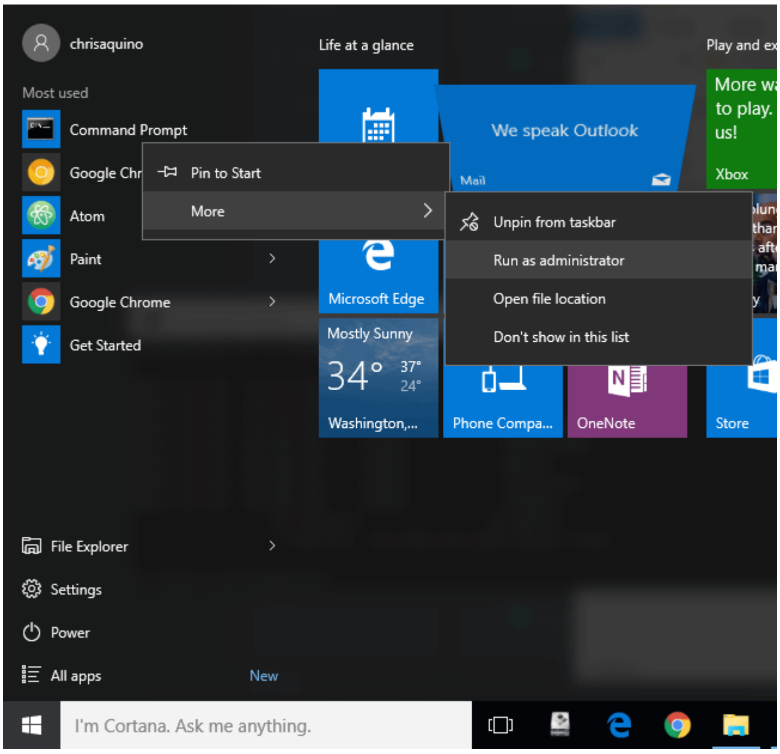


```
$ sudo ls

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

Password:
```

`sudo` will prompt you for your password before it runs the command as the superuser. As you type, your keystrokes will not be echoed back, so type carefully.

On Windows, if you need to give yourself privileges you do so in the process of opening the command-line interface. Find the command prompt in the Windows Start Menu, right-click it, and choose Run as Administrator (Figure 1.26). Any commands you run in this command prompt will be run as the superuser, so be careful.

Figure 1.26 Opening the command prompt as an administrator



Quitting a program

As you proceed through the book, you will run many apps from the command line. Some of them will do their job and quit automatically, but others will run until you stop them. To quit a command-line program, press Control-C.

Installing Node.js and browser-sync

There is one final set-up step before you begin your first project.

Node.js (or simply “Node”) lets you use programs written in JavaScript from the command line. Most front-end development tools are written for use with Node.js. You will learn lots more about Node.js in Chapter 15, but you will begin using one tool that depends on it, `browser-sync`, right away.

Install Node by downloading the installer from `nodejs.org` (Figure 1.27). The version of Node.js used in this book is 5.11.1, and you will likely see a different version available for download.

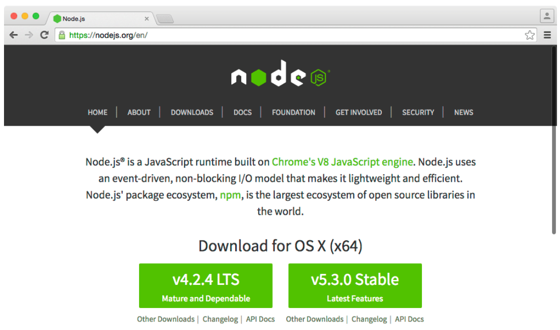


Figure 1.27 Downloading Node.js

Node.js
https://nodejs.org/en/
HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS
v4.2.4 LTS
Mature and Dependable
v5.3.0 Stable
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Download for OS X (x64)

Double-click the installer and follow the prompts.

When you install Node, it provides two command-line programs: `node` and `npm`. The `node` program does the work of running programs written in JavaScript. You will not need to use it until Chapter 15. The other program is the Node package manager, `npm`, which is needed for installing open-source development tools from the internet.

`browser-sync` is one such tool, and it will be invaluable to you throughout the book. It makes your example code easier to run in the browser and automatically reloads the browser when you save changes to your code.

Install `browser-sync` using this command at the command line:

```
npm install -g browser-sync
```

(The `-g` in the command stands for “global.” Installing the package globally means that you will be able to run `browser-sync` from any directory.)

It does not matter what directory you are in when you run this command, but you will probably need superuser privileges. If that is the case, run the command using `sudo` on a Mac:

```
sudo npm install -g browser-sync
```

If you are on Windows, first open a command prompt as the administrator, as shown above.

When you start `browser-sync`, as you will in the next chapter, it will run until you press Control-C. It is a good idea to quit `browser-sync` when you are done working on a project for a while. That means that you will need to start `browser-sync` each time you begin work on the first two projects in this book (Ottergram and CoffeeRun).

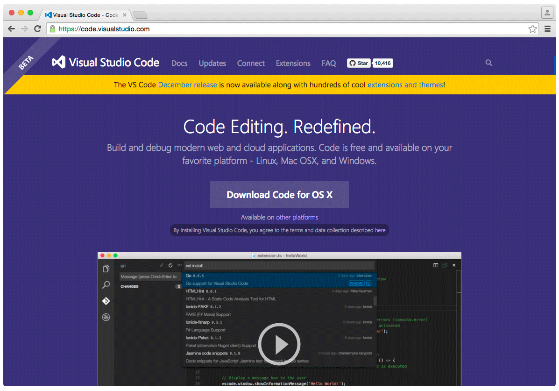
With that, you have the tools you need to get started on your Ottergram project!

For the More Curious: Alternatives to Atom

There are many, many text editors to choose from. If you are not that keen on Atom, when you are done working through the projects in this book you may want to try out one of the following two options. Both are available for free for Mac and Windows, and both have a large number of plug-ins to customize your development experience. Also, like Atom, both are built using HTML, CSS, and JavaScript, but run as desktop applications.

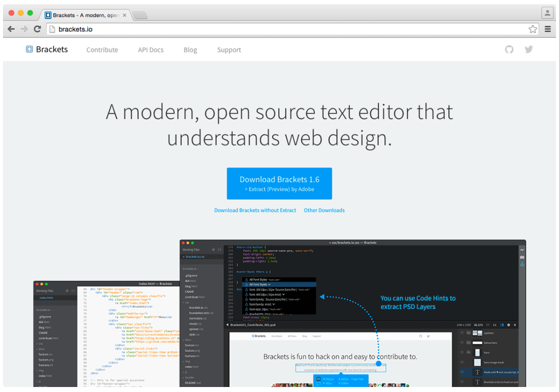
Visual Studio Code is Microsoft's open source text editor, made specifically for developing web applications. It can be downloaded from `code.visualstudio.com` (Figure 1.28).

Figure 1.28 The Visual Studio Code website


Visual Studio Code
Docs
Updates
Connect
Extensions
FAQ
The VS Code December release is now available along with hundreds of cool extensions and themes!
Code Editing. Redefined.
Build and debug modern web and cloud applications. Code is free and available on your favorite platform - Linux, Mac OSX, and Windows.
Download Code for OS X
Available on other platforms
By installing Visual Studio Code, you agree to the terms and data collection described here

Adobe's Brackets text editor is particularly good for building user interfaces with HTML and CSS. In fact, it provides an extension for helping you work with Adobe's layered PSD image files. Brackets is available from `brackets.io` (Figure 1.29).

Figure 1.29 The Adobe Brackets website


Brackets
Contribute
API Docs
Blog
Support
A modern, open source text editor that understands web design.
Download Brackets 1.6
+ Extract (Preview) by Adobe
Download Brackets without Extract
Other Downloads
You can use Code Hints to extract PSD Layers
Brackets is fun to hack on and easy to contribute to.

2

Setting Up Your First Project

When you visit a website, your browser has a conversation with a server, another computer on the internet.

Browser: “Hey there! Can I please have the contents of the file named `cat-videos.html`?”

Server: “Certainly. Let me take a look around … here it is!”

Browser: “Ah, it’s telling me that I need another file named `styles.css`.”

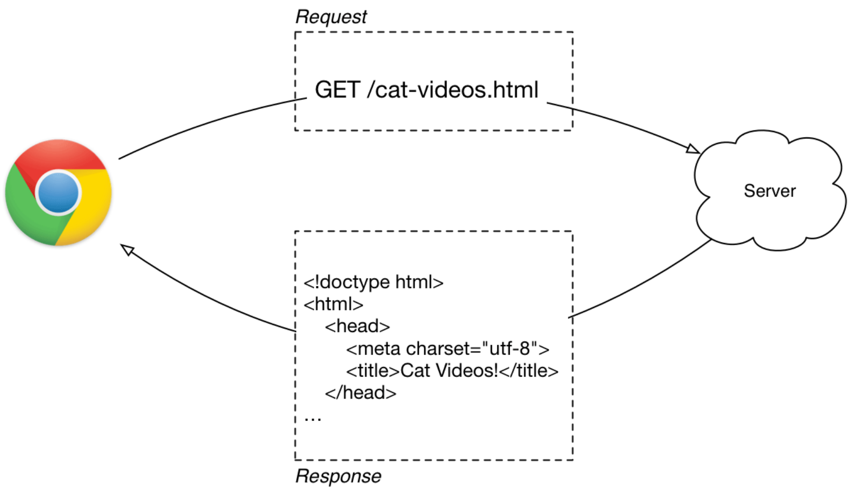
Server: “Sure thing. Let me take a look around … here it is!”

Browser: “OK, that file says that I need another file named `animated-background.gif`.”

Server: “No problem. Let me take a look around … here it is!”

That conversation goes on for some time, sometimes lasting thousands of milliseconds (Figure 2.1).

Figure 2.1 The browser sends a request, the server responds



```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Cat Videos!</title>
  </head>
…
```

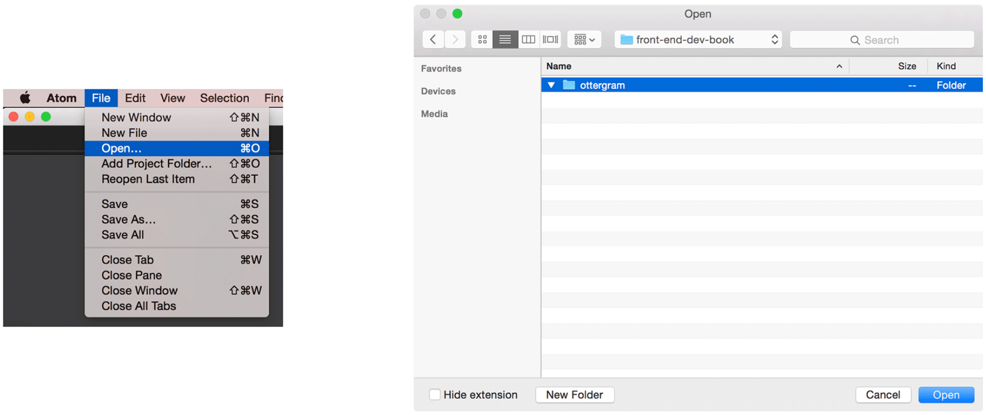
It is the browser's job to send requests to the server; interpret the HTML, CSS, and JavaScript it receives in the response from the server; and present the result to the user. Each of these three technologies plays a part in the user's experience of a website. If your app were a living creature, the HTML would be its skeleton and organs (the mechanics), the CSS would be its skin (the visible layer), and the JavaScript would be its personality (how it behaves).

In this chapter, you are going to set up the basic HTML for your first project, Ottergram. In the next chapter, you will set up your CSS, which you will refine in Chapter 4. In Chapter 6, you will begin adding JavaScript.

Setting Up Ottergram

In Chapter 1, you created a folder for the projects in this book as well as a folder for Ottergram. Start your Atom text editor and open the `ottergram` folder by clicking File → Open (or File → Open Folder... on Windows). In the dialog box, navigate to the `front-end-dev-book` folder and choose the `ottergram` folder. Click Open to tell Atom to use this folder (Figure 2.2).

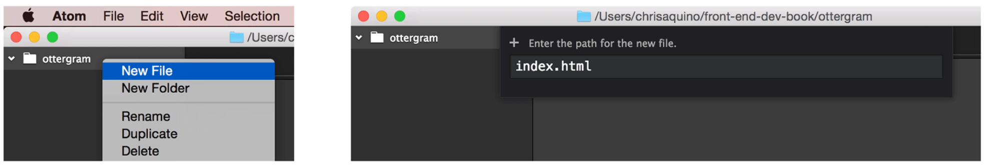
Figure 2.2 Opening your project folder in Atom



You will see the `ottergram` folder in the lefthand panel of Atom. This panel is for navigating among the files and folders in your project.

You are going to create some files and folders within the `ottergram` project folder using Atom. Control-click (right-click) `ottergram` in the lefthand panel and click New File in the pop-up menu. You will be prompted for a name for the new file. Enter `index.html` and press the Return key (Figure 2.3).

Figure 2.3 Creating a new file in Atom

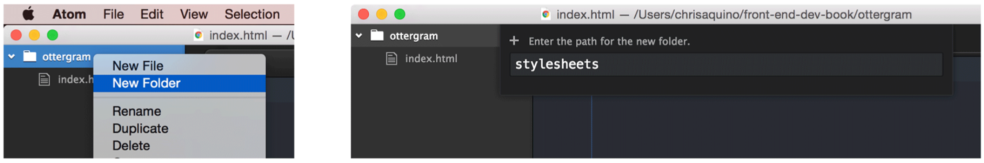


Enter the path for the new file.

```
index.html
```

You can use the same process to create folders using Atom. Control-click (right-click) `ottergram` in the lefthand panel again, but this time click New Folder in the pop-up. Enter the name `stylesheets` in the prompt that appears (Figure 2.4).

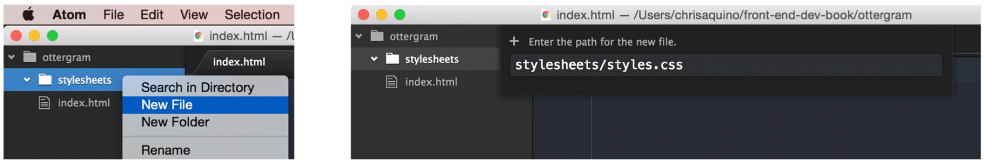
Figure 2.4 Creating a new folder in Atom


Atom File Edit View Selection
index.html — /U
ottergram
index.h
New File
New Folder
Rename
Duplicate
Delete
index.html — /Users/chrisaquino/front-end-dev-book/ottergram
ottergram
index.html
Enter the path for the new folder.

stylesheets

Finally, create a file named `styles.css` in the `stylesheets` folder: Control-click (right-click) `stylesheets` in the lefthand panel and choose New File. The prompt will pre-fill the text "stylesheets/". After this, enter `styles.css` and press the Return key (Figure 2.5).

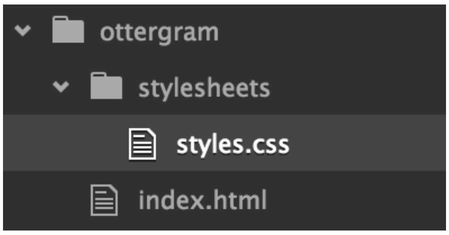
Figure 2.5 Creating a new CSS file in Atom



stylesheets/styles.css

When you are finished, your project folder should look like Figure 2.6.

Figure 2.6 Initial files and folders for Ottergram



There are no rules about how to structure your files and folders or what to name them. However, Ottergram (like the other projects in this book) follows conventions used by many front-end developers. Your `index.html` file will hold your HTML code. Naming the main HTML file `index.html` dates back to the early days of the web, and the convention continues today.

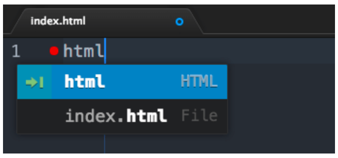
The `stylesheets` folder, as the name suggests, will hold one or more files with styling information for Ottergram. These will be CSS, or "cascading style sheets," files. Sometimes developers give their CSS files names that describe what part of the page or site they pertain to, such as `header.css` or `blog.css`. Ottergram is a simple project and only needs one CSS file, so you have named it `styles.css` to reflect its global role.

Initial HTML

Time to get coding. Open `index.html` in Atom and add some basic HTML to get started.

Start by typing `html`. Atom will offer you an autocomplete option, as shown in Figure 2.7. (If it does not, make sure you installed the emmet plug-in as directed in Chapter 1.)

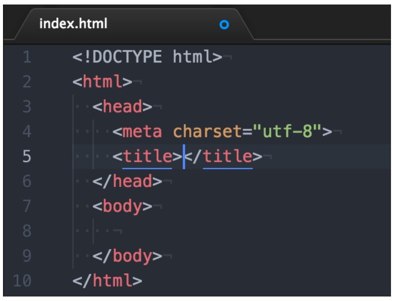
Figure 2.7 Atom's autocomplete menu



```
html
```

Press the Return key, and Atom will provide bare-bones HTML elements to get you started (Figure 2.8).

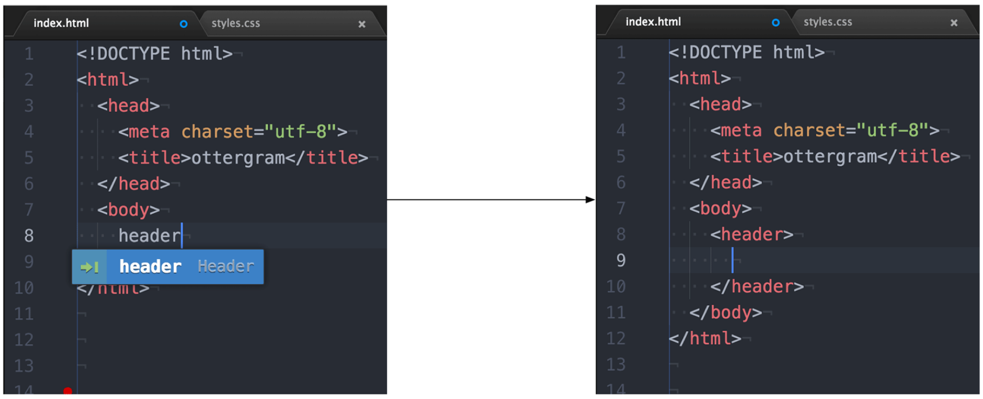
Figure 2.8 HTML created using autocomplete



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

Your cursor is between `<title>` and `</title>` – the opening and closing `title` tags. Type "ottergram" to give the project a name. Now, click to put your cursor in the blank line between the opening and closing `body` tags. There, type "header" and press the Return key. Atom will convert the text "header" into opening and closing `header` tags with a blank line between them (Figure 2.9).

Figure 2.9 Header tag created with autocomplete



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ottergram</title>
  </head>
  <body>
    header
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ottergram</title>
  </head>
  <body>
    <header>

    </header>
  </body>
</html>
```

Next, type “h1” and press Return. Again, your text is converted into tags, this time without a blank line. Enter the text “ottergram” again. This will be the heading that will appear on your web page.

Your file should look like this:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ottergram</title>
  </head>
  <body>
    <header>
      <h1>ottergram</h1>
    </header>
  </body>
</html>
```

Atom and emmet have together saved you some typing and helped you build well-formed initial HTML.

Let’s examine your code. The first line, `<!doctype html>`, defines the *doctype* – it tells the browser which version of HTML the document is written in. The browser may render, or draw, the page a little differently based the doctype. Here, the doctype specifies HTML5.

Earlier versions of HTML often had long, convoluted, and hard to remember doctypes, such as:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Often, folks had to look up the doctype each time they created a new document.

With HTML5, the doctype is short and sweet. It is the one that will be used throughout all of the projects in this book, and you should use it for your apps.

After the doctype is some basic HTML markup consisting of a `head` and a `body`.

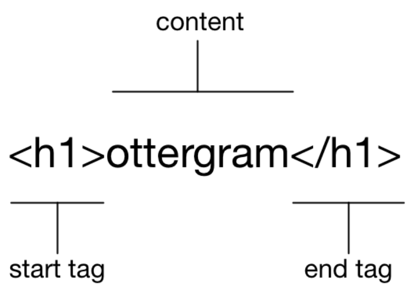
The `head` will hold information about the document and how the browser should handle the document. For example, the title of the document, what CSS or JavaScript files the page uses, and when the document was last modified are all included in the `head`.

Here, the `head` contains a `<meta>` tag. `<meta>` tags provide the browser with information about the document itself, such as the name of the document’s author or keywords for search engines. The `<meta>` tag in Ottergram, `<meta charset="utf-8">`, specifies that the document is encoded using the UTF-8 character set, which encompasses all Unicode characters. Use this tag in your documents so that the widest range of browsers can interpret them correctly, especially if you expect international traffic.

The `body` will hold all of the HTML code that represents the content of your page: all the images, links, text, buttons, and videos that will appear on the page.

Most tags enclose some other content. Take a look at the h1 heading you included; its anatomy is shown in Figure 2.10.

Figure 2.10 Anatomy of a simple HTML tag



HTML stands for "hypertext markup language." Tags are used to "mark up" your content, designating their purpose – such as headings, list items, and links.

The content enclosed by a set of tags can also include other HTML. Notice, for example, that the <header> tags enclose the <h1> tag shown above (and the <body> tags enclose the <header>!).

There are a lot of tags to choose from – more than 140. To see a list of them, visit MDN's HTML element reference, located at developer.mozilla.org/en-US/docs/Web/HTML/Element. This reference includes a brief description of each element and groups elements by usage (e.g., text content, content sectioning, or multimedia).

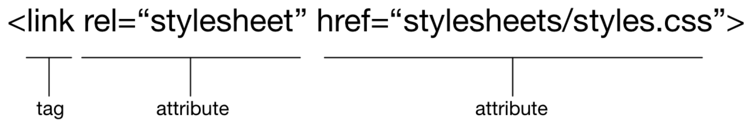
Linking a stylesheet

In Chapter 3, you will write styling rules in your stylesheet, styles.css. But remember the conversation between the browser and the server at the beginning of this chapter? The browser only knows to ask for a file from the server if it has been told that the file exists. You have to *link* to your stylesheet so that the browser knows to ask for it. Update the head of index.html with a link to your styles.css file.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ottergram</title>
    <link rel="stylesheet" href="stylesheets/styles.css">
  </head>
  <body>
...
```

The `<link>` tag is how you attach an external stylesheet to an HTML document. It has two *attributes*, which give the browser more information about the tag's purpose (Figure 2.11). (The order of HTML attributes does not matter.)

Figure 2.11 Anatomy of a tag with attributes



You set the `rel` (or "relationship") attribute to `"stylesheet"`, which lets the browser know that the linked document provides styling information. The `href` attribute tells the browser to send a request to the server for the `styles.css` file located in the `stylesheets` folder. Note that this file path is *relative* to the current document.

Save `index.html` before you move on.

Adding content

A web page without content is like a day without coffee. Add a list after your `header` to give your project a reason for living.

You are going to add an *unordered list* (that is, a bulleted list) using the `<ul>` tag. In the list, you will include five list items using `<li>` tags, and in each list item you will include some text surrounded by `<span>` tags.

The updated `index.html` is shown below. Note that throughout this book we show new code that you are adding in bold type. Code that you are to delete is shown struck through. Existing code is shown in plain text to help you position your changes within the file.

We encourage you to make use of Atom's autocompletion and autoformatting features. With your cursor in position, type "ul" and press Return. Next, type "li" and press Return twice, then type "span" and press Return once. Enter the name of an otter, then create four more list items and spans in the same way.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ottergram</title>
    <link rel="stylesheet" href="stylesheets/styles.css">
  </head>
  <body>
    <header>
      <h1>ottergram</h1>
    </header>
    <ul>
      <li>
        <span>Barry</span>
      </li>
      <li>
        <span>Robin</span>
      </li>
      <li>
        <span>Maurice</span>
      </li>
      <li>
        <span>Lesley</span>
      </li>
      <li>
        <span>Barbara</span>
      </li>
    </ul>
  </body>
</html>
```

The `<span>` tags nested inside each `<li>` tag do not have any special meaning. They are generic containers for other content. You will be using them in Ottergram for styling purposes, and you will see other examples of container elements as you continue through this book.

Next, you will add images of otters to go with the names you have entered.

Adding images

The resource files for all the projects in this book are at `www.bignerdranch.com/downloads/front-end-dev-resources.zip`. They include five Creative Commons-licensed otter images taken by Michael L. Baird, Joe Robertson, and Agunther that were found on `commons.wikimedia.org`.

Download and unzip the resources. Inside the `ottergram-resources` folder, locate the `img` folder. Copy the `img` folder to your `ottergram/` project directory. (The .zip contains other resources, but for now you will only need the `img` folder.)

You want your list to include clickable thumbnail images in addition to the titles. You will achieve this by adding anchor and image tags to each item in your `ul`. We will explain these changes in more detail after you enter them. (If you use autocompletion, note that you will need to move the `</a>` tags so that they follow the `span`s.)

```
...
    <ul>
      <li>
        <a href="#">
          <img src="img/otter1.jpg" alt="Barry the Otter">
          <span>Barry</span>
        </a>
      </li>
      <li>
        <a href="#">
          <img src="img/otter2.jpg" alt="Robin the Otter">
          <span>Robin</span>
        </a>
      </li>
      <li>
        <a href="#">
          <img src="img/otter3.jpg" alt="Maurice the Otter">
          <span>Maurice</span>
        </a>
      </li>
      <li>
        <a href="#">
          <img src="img/otter4.jpg" alt="Lesley the Otter">
          <span>Lesley</span>
        </a>
      </li>
      <li>
        <a href="#">
          <img src="img/otter5.jpg" alt="Barbara the Otter">
          <span>Barbara</span>
        </a>
      </li>
    </ul>
...
```

If your lines are not nicely indented, you can take advantage of the atom-beautify plug-in that you installed. Click Packages → Atom Beautify → Beautify and your code will be aligned and indented for you.

Let's look at what you have added.

The <a> tag is the *anchor* tag. Anchor tags make elements on the page clickable, so that they take the user to another page. They are commonly referred to as "links," but beware: They are not like the `<link>` tag you used earlier.

Anchor tags have an `href` attribute, which indicates the resource the anchor points to. Usually the value is a web address. Sometimes, though, you do not want a link to go anywhere. That is the case for now, so you assigned the "dummy" value # to the `href` attributes. This will make the browser scroll to the top of the page when the image is clicked. Later you will use the anchor tags to open a larger copy of an image when the thumbnail is clicked.

Inside the anchor tags you added `<img>`, or *image*, tags with `src` attributes indicating filenames in the `img` directory you added earlier. You also added a descriptive `alt` attribute to your image tags. `alt` attributes contain text that replaces the image if it is unable to load. `alt` text is also what screen readers use to describe an image to a user with a visual impairment.

Image tags are different from most other elements in that they do not wrap other elements, but instead refer to a resource. When the browser encounters an `<img>` tag, it draws the image to the page. This is known as a *replaced element*. Other replaced elements include embedded documents and applets.

Because they do not wrap content or other elements, `<img>` tags do not have a corresponding closing tag. This makes them *self-closing* tags (also known as *void* tags). You will sometimes see self-closing tags written with a slash before the right angle-bracket, like `<img src="otter.jpg"/>`. Whether to include the slash is a matter of preference and does not make a difference to the browser. In this book, self-closing tags are written without the slash.

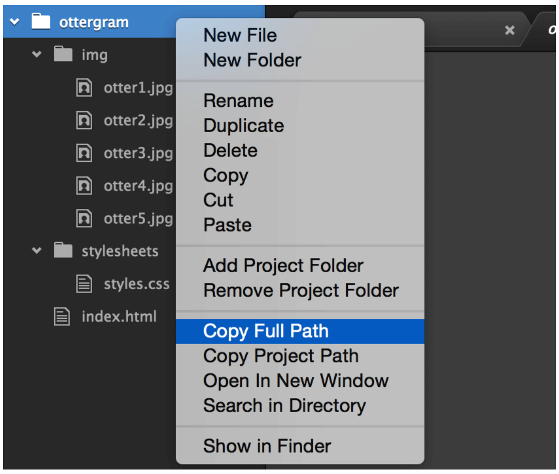
Save `index.html`. In a moment, you will see the results of your coding.

Viewing the Web Page in the Browser

To view your web page, you need to be running the `browser-sync` tool that you installed in Chapter 1.

Open the terminal and change directory to your `ottergram` folder. Recall from Chapter 1 that you change directory using the `cd` command followed by the path of the folder you are moving into. One easy way to get the `ottergram` path is to Control-click (right-click) the `ottergram` folder in Atom's lefthand panel and choose Copy Full Path (Figure 2.12). Then, at the command line, type `cd`, paste the path, and press Return.

Figure 2.12 Copying the `ottergram` folder path from Atom



The path you enter might look something like this:

```
cd /Users/chrisaquino/Projects/front-end-dev-book/ottergram
```

Once you are in the ottergram directory, run the following command to open Ottergram in Chrome. (We have broken the command across two lines so that it fits on the page. You should enter it on a single line.)

```
browser-sync start --server --browser "Google Chrome"
                   --files "stylesheets/*.css, *.html"
```

If Chrome is your default browser, you can leave out the `--browser "Google Chrome"` portion of the command:

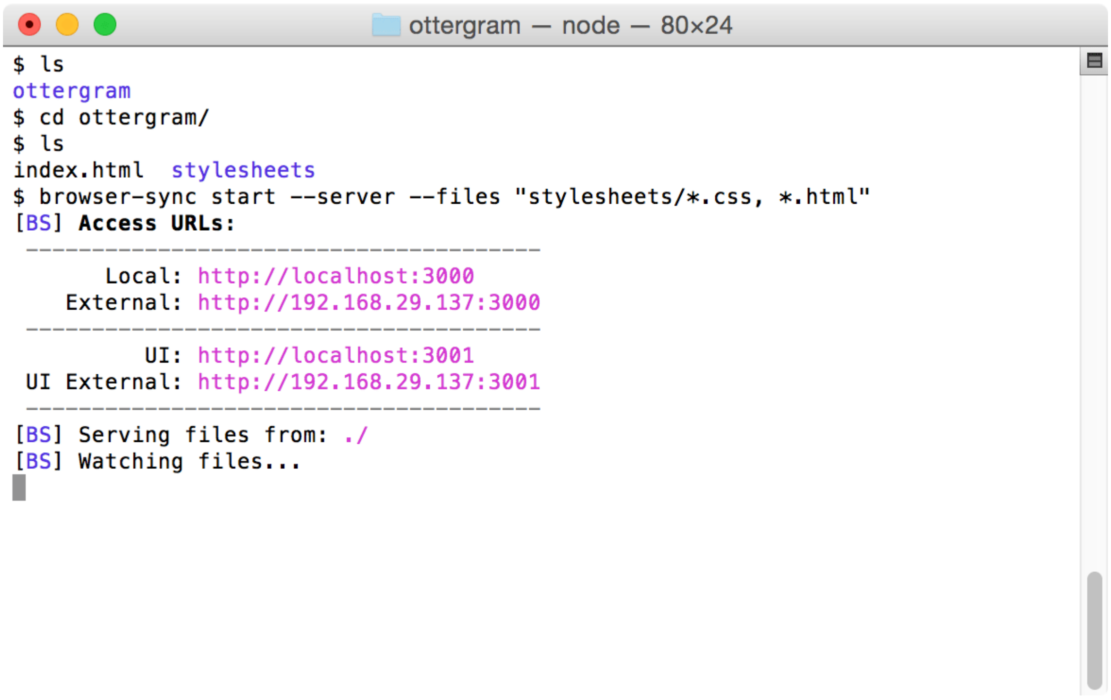
```
browser-sync start --server --files "stylesheets/*.css, *.html"
```

This command starts browser-sync in server mode, allowing it to send responses when a browser sends a request to get a file, such as the index.html file you created.

The command you entered also tells browser-sync to automatically reload the browser if any HTML or CSS files are changed. This makes the development experience much nicer. Before tools like browser-sync, you had to manually reload the page after every change.

Figure 2.13 shows the result of entering this command on a Mac.

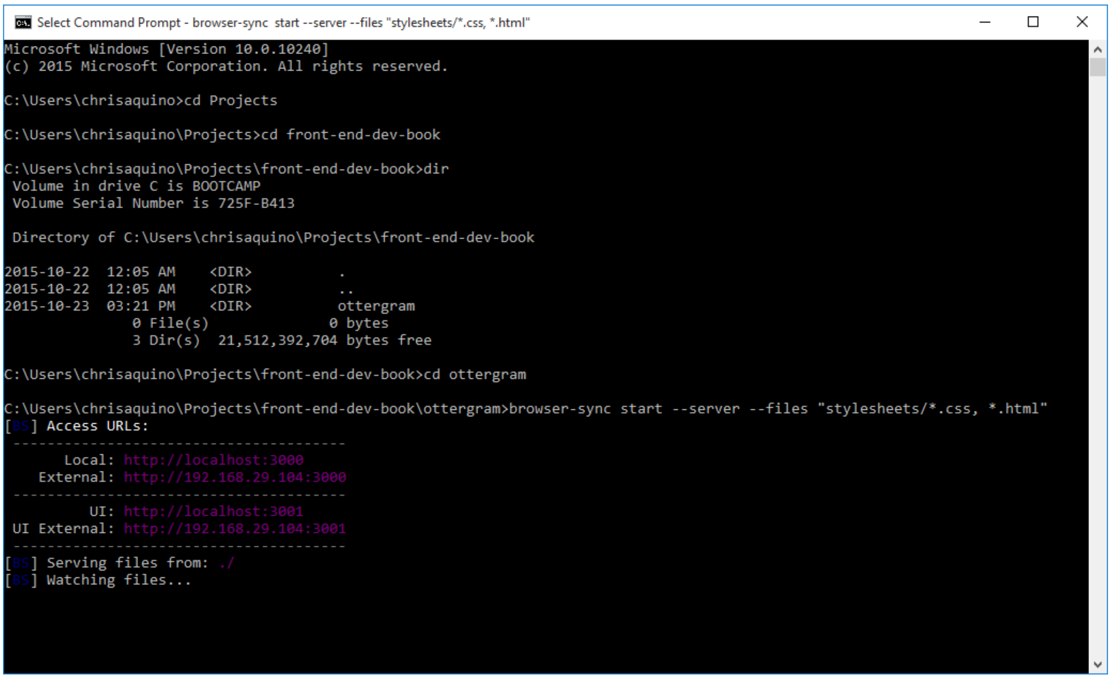
Figure 2.13 Starting browser-sync in the OS X Terminal



```
$ ls
ottergram
$ cd ottergram/
$ ls
index.html  stylesheets
$ browser-sync start --server --files "stylesheets/*.css, *.html"
[BS] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://192.168.29.137:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://192.168.29.137:3001
 --------------------------------------
[BS] Serving files from: ./
[BS] Watching files...
```

You should see the same output on Windows (Figure 2.14).

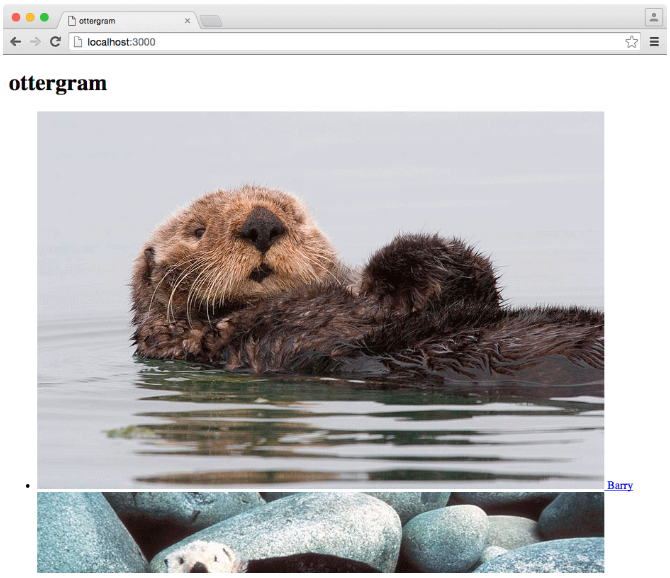
Figure 2.14 Starting browser-sync in the Windows Command Prompt



```
Select Command Prompt - browser-sync  start --server --files "stylesheets/*.css, *.html"
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\chrisaquino>cd Projects

C:\Users\chrisaquino\Projects>cd front-end-dev-book

C:\Users\chrisaquino\Projects\front-end-dev-book>dir
 Volume in drive C is BOOTCAMP
 Volume Serial Number is 725F-B413

 Directory of C:\Users\chrisaquino\Projects\front-end-dev-book

2015-10-22  12:05 AM    <DIR>          .
2015-10-22  12:05 AM    <DIR>          ..
2015-10-23  03:21 PM    <DIR>          ottergram
               0 File(s)              0 bytes
               3 Dir(s)  21,512,392,704 bytes free

C:\Users\chrisaquino\Projects\front-end-dev-book>cd ottergram

C:\Users\chrisaquino\Projects\front-end-dev-book\ottergram>browser-sync start --server --files "stylesheets/*.css, *.html"
[BS] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://192.168.29.104:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://192.168.29.104:3001
 --------------------------------------
[BS] Serving files from: ./
[BS] Watching files...
```

Once the Ottergram page has loaded in Chrome, you should see your page with the “ottergram” heading, “ottergram” as the tab label, and a series of otter photos and names (Figure 2.15).

Figure 2.15 Viewing Ottergram in the browser



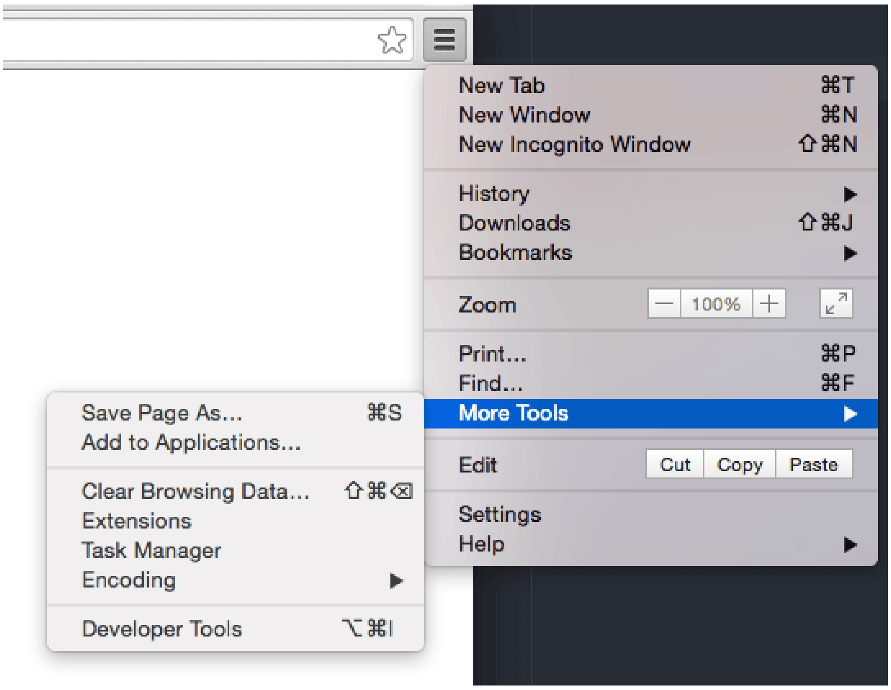
The Chrome Developer Tools

Chrome has built-in Developer Tools (commonly known as “DevTools”) that are among the best available for testing styles, layouts, and more on the fly. Using the DevTools is much more efficient than trying things out in code. The DevTools are very powerful and will be your constant companion as you do front-end development.

You will start using the DevTools in the next chapter. For now, open the window and familiarize yourself with its major areas.

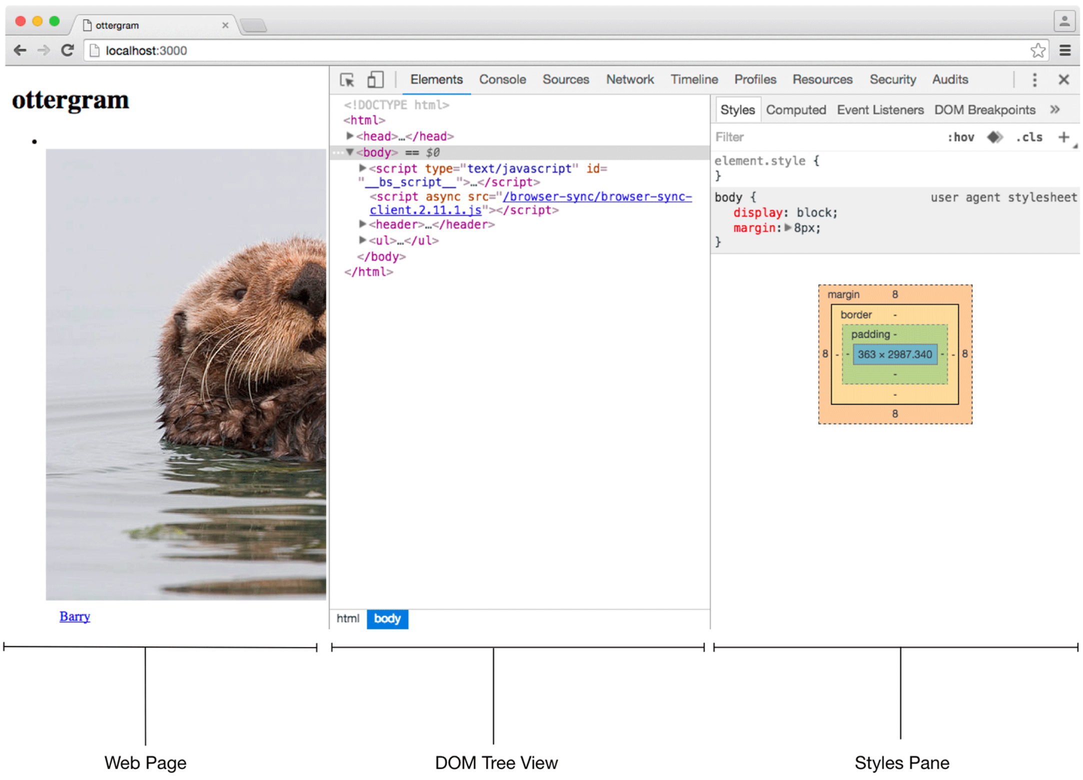
To open the DevTools, click the ≡ icon to the right of the address bar in Chrome. Next, click More Tools → Developer Tools (Figure 2.16).

Figure 2.16 Opening the Developer Tools



Chrome displays the DevTools to the right by default. Your screen will look something like Figure 2.17.

Figure 2.17 The DevTools showing the elements panel



```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body> == $0
   ▶<script type="text/javascript" id=
   "__bs_script__">…</script>
    <script async src="/browser-sync/browser-sync-
    client.2.11.1.js"></script>
   ▶<header>…</header>
   ▶<ul>…</ul>
  </body>
</html>
```

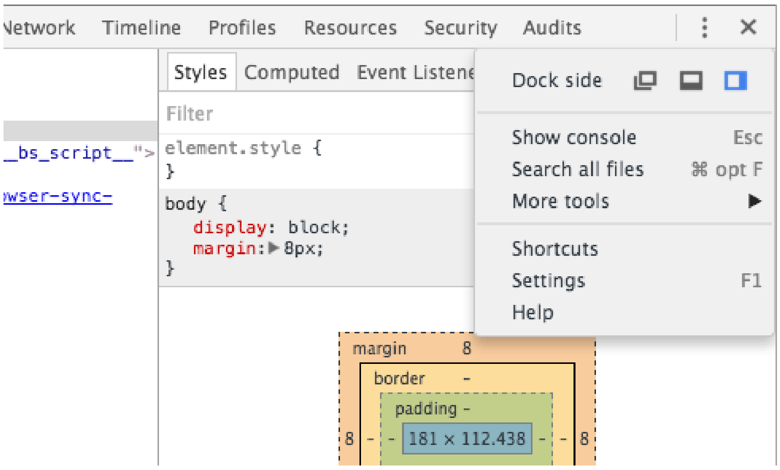
```
element.style {
}
body {                                   user agent stylesheet
   display: block;
   margin:▶8px;
}
```

The DevTools show the relationship between the code and the resulting page elements. They let you inspect individual elements' attributes and styles and see immediately how the browser is interpreting your code. Seeing this relationship is critical for both development and debugging.

In Figure 2.17, you can see the DevTools next to the web page, displaying the elements panel. The elements panel is divided into two sections. On the left is the *DOM tree view*. This is a representation of the HTML, interpreted as DOM elements. (You will learn much more about DOM, which stands for "document object model," in upcoming chapters.) On the righthand side of the elements panel is the styles pane. This shows any visual styles applied to individual elements.

Having the DevTools docked on the right side of the screen while you are working is usually convenient. If you want to change the location of the DevTools, you can click the ⋮ button near the upper-right corner. This will show you a menu of options, including buttons for the Dock side, which will change the anchor location of the DevTools (Figure 2.18).

Figure 2.18 Changing the dock side of the DevTools



```
element.style {
}
body {
   display: block;
   margin:▶8px;
}
```

With your otters and markup in place and the DevTools open, you are ready to begin styling your project in the next chapter.

For the More Curious: CSS Versions

The version history of CSS includes standard versions 1, 2, and 2.1. After 2.1, it was decided that the standard needed to be broken up because it was getting too big.

There is no version 3. Instead, CSS3 is a blanket term for a number of modules, each with its own version number.

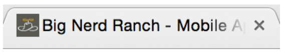
Table 2.1 CSS versions, real and imagined

Version Number	Release Year	Notable Features
1	1996	Basic font properties (`font-family`, `font-style`), foreground and background colors, text alignment, margin, border, and padding.
2	1998	Absolute, relative, and fixed positioning; new font properties.
2.1	2011	Removed features that were poorly supported by browsers.
“3”	Various	A collection of different specifications, such as media queries, new selectors, semi-transparent colors, `@font-face`.

For the More Curious: The favicon.ico

Have you ever noticed the little icon that appears at the left end of your browser’s address bar when you visit most websites? Sometimes they also appear in your browser tab, as in Figure 2.19.

Figure 2.19 The `bignerdranch.com` `favicon.ico`



Big Nerd Ranch - Mobile A