Making Everything Easier!™

# Coding with JavaScript

FOR DUMMIES®

A Wiley Brand

## Learn to:

- Go from no coding experience to being handy with JavaScript
- Add interactive elements to a web page or site
- Develop simple apps built on JavaScript

**Chris Minnick**
**Eva Holland**


by Chris Minnick and Eva Holland

FOR
DUMMIES
A Wiley Brand

**Coding with JavaScript For Dummies®**

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com



Published simultaneously in Canada



Library of Congress Control Number: 2015938674

ISBN: 978-1-119-05607-2

ISBN 978-1-119-05607-2 (pbk); ISBN 978-1-119-05605-8 (ePDF); ISBN 978-1-119-05606-5 (ePub)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance


*Introduction* ..... 1

*Part I: Getting Started with JavaScript* ..... 5

Chapter 1: The World's Most Misunderstood Programming Language ..... 7
Chapter 2: Writing Your First JavaScript Program ..... 19
Chapter 3: Working with Variables ..... 39
Chapter 4: Understanding Arrays ..... 55
Chapter 5: Working with Operators, Expressions, and Statements ..... 67
Chapter 6: Getting into the Flow with Loops and Branches ..... 81

*Part II: Organizing Your JavaScript* ..... 95

Chapter 7: Getting Functional ..... 97
Chapter 8: Making and Using Objects ..... 117

*Part III: JavaScript on the Web* ..... 131

Chapter 9: Controlling the Browser with the Window Object ..... 133
Chapter 10: Manipulating Documents with the DOM ..... 147
Chapter 11: Using Events in JavaScript ..... 169
Chapter 12: Integrating Input and Output ..... 181
Chapter 13: Working with CSS and Graphics ..... 195

*Part IV: Beyond the Basics* ..... 211

Chapter 14: Searching with Regular Expressions ..... 213
Chapter 15: Understanding Callbacks and Closures ..... 225
Chapter 16: Embracing AJAX and JSON ..... 237

*Part V: JavaScript and HTML5* ..... 253

Chapter 17: HTML5 APIs ..... 255
Chapter 18: jQuery ..... 271

*Part VI: The Part of Tens* ..... 289

Chapter 19: Ten JavaScript Frameworks and Libraries to Learn Next ..... 291
Chapter 20: Ten Common JavaScript Bugs and How to Avoid Them ..... 303
Chapter 21: Ten Online Tools to Help You Write Better JavaScript ..... 313

*Index* ..... 325

# Table of Contents


*Introduction* ..... 1

About This Book ..... 1
Foolish Assumptions ..... 2
Icons Used In This Book ..... 3
Beyond the Book ..... 4
Where to Go from Here ..... 4

*Part 1: Getting Started with JavaScript* ..... 5

**Chapter 1: The World's Most Misunderstood Programming Language** ..... 7

What Is JavaScript? ..... 8
The Eich-man cometh ..... 8
Mocha-licious ..... 9
We need more effects! ..... 9
JavaScript grows up ..... 9
Dynamic scripting language ..... 10
What Does JavaScript Do? ..... 12
Why JavaScript? ..... 13
JavaScript is easy to learn ..... 13
Where is JavaScript? JavaScript is everywhere! ..... 14
JavaScript is powerful! ..... 18
JavaScript is in demand ..... 18

**Chapter 2: Writing Your First JavaScript Program** ..... 19

Setting Up Your Development Environment ..... 19
Downloading and installing Chrome ..... 20
Downloading and installing a code editor ..... 21
Reading JavaScript Code ..... 29
Running JavaScript in the Browser Window ..... 29
Using JavaScript in an HTML event attribute ..... 30
Using JavaScript in a script element ..... 31
Including external JavaScript files ..... 33
Using the JavaScript Developer Console ..... 36
Commenting your code ..... 37

**Chapter 3: Working with Variables .....39**

Understanding Variables .....39
Declaring Variables .....41
Understanding Global and Local Scope .....42
Naming Variables .....44
Creating Constants Using the const Keyword .....46
Working with Data Types .....46
Number data type .....47
String data type .....49
Boolean data type .....52
NaN data type .....53
undefined data type .....53

**Chapter 4: Understanding Arrays .....55**

Making a List .....55
Array Fundamentals .....57
Arrays are zero indexed .....57
Arrays can store any type of data .....58
Creating Arrays .....59
Using the new keyword method .....59
Array literal .....59
Populating Arrays .....60
Understanding Multidimensional Arrays .....60
Accessing Array Elements .....62
Looping through arrays .....63
Array properties .....63
Array methods .....64
Using array methods .....64

**Chapter 5: Working with Operators, Expressions, and Statements .....67**

Express Yourself .....68
Hello, Operator .....68
Operator precedence .....68
Types of Operators .....72
Assignment operators .....72
Comparison operators .....73
Arithmetic operators .....73
String operator .....75
Bitwise operators .....75
Logical operators .....77
Special operators .....78
Combining operators .....80

**Chapter 6: Getting into the Flow with Loops and Branches .....81**

Branching Out ..... 81
if . . . else ..... 82
Switch ..... 84
Here We Go: Loop De Loop ..... 85
for ..... 86
for . . . in ..... 88
while loops ..... 90
do . . . while ..... 91
break and continue ..... 92

*Part II: Organizing Your JavaScript ..... 95*

**Chapter 7: Getting Functional .....97**

Understanding the Function of Functions ..... 97
Using Function Terminology ..... 99
Define a function ..... 99
Function head ..... 99
Function body ..... 99
Call a function ..... 100
Defining parameters and passing arguments ..... 100
Return a value ..... 100
The Benefits of Using Functions ..... 101
Writing Functions ..... 104
Returning Values ..... 105
Passing and Using Arguments ..... 106
Passing arguments by value ..... 107
Passing arguments by reference ..... 109
Calling a function without all of the arguments ..... 109
Setting default parameter values ..... 109
Calling a function with more argument than parameters ..... 110
Getting into arguments with the arguments object ..... 110
Function Scope ..... 111
Anonymous Function ..... 111
Knowing the differences between anonymous and named functions ..... 112
Self-executing anonymous functions ..... 112
Do it Again with Recursion ..... 113
Functions within Functions ..... 114

**Chapter 8: Making and Using Objects .....117**

Object of My Desire ..... 117
Creating Objects ..... 119
Defining objects with object literals ..... 119
Defining objects with an Object constructor ..... 120
Retrieving and Setting Object Properties ..... 120
Dot notation ..... 120
Square bracket notation ..... 121
Deleting Properties ..... 123
Working with Methods ..... 123
Using this ..... 124
An Object-Oriented Way to Become Wealthy: Inheritance ..... 125
Constructing Objects with constructor functions ..... 127
Modifying an object type ..... 129
Creating Objects with Object.create ..... 129

*Part III: JavaScript on the Web ..... 131*

**Chapter 9: Controlling the Browser with the Window Object .....133**

Understanding the Browser Environment ..... 133
The user interface ..... 134
Loader ..... 134
HTML parsing ..... 136
CSS parsing ..... 136
JavaScript parsing ..... 136
Layout and rendering ..... 137
Igniting the BOM ..... 137
The Navigator object ..... 137
The Window object ..... 140
Using the `Window` object's methods ..... 145

**Chapter 10: Manipulating Documents with the DOM .....147**

Understanding the DOM ..... 147
Node Relationships ..... 149
Using the Document Object's Properties and Methods ..... 153
Using the Element Object's Properties and Methods ..... 155
Working with the Contents of Elements ..... 159
innerHTML ..... 160
Setting attributes ..... 161
Getting Elements by ID, Tag Name, or Class ..... 161
getElementById ..... 161
getElementsByTagName ..... 162
getElementsByClassName ..... 163

Using the Attribute Object's Properties ..... 165
Creating and appending elements ..... 165
Removing elements ..... 166

**Chapter 11: Using Events in JavaScript ..... 169**

Knowing Your Events ..... 169
Handling Events ..... 171
Using inline event handlers ..... 172
Event handling using element properties ..... 173
Event handling using addEventListener ..... 174
Stopping propagation ..... 179

**Chapter 12: Integrating Input and Output ..... 181**

Understanding HTML Forms ..... 181
The form element ..... 181
The label element ..... 183
The input element ..... 184
The select element ..... 185
The textarea element ..... 186
The button element ..... 186
Working with the Form Object ..... 187
Using Form properties ..... 187
Using the Form object's methods ..... 188
Accessing form elements ..... 190
Getting and setting form element values ..... 191
Validating user input ..... 192

**Chapter 13: Working with CSS and Graphics ..... 195**

Using the Style Object ..... 195
Getting the current style of an element ..... 196
Setting style properties ..... 199
Animating Elements with the Style Object ..... 200
Working with Images ..... 203
Using the Image object ..... 203
Creating rollover buttons ..... 203
Grow images on mouseover ..... 205
Creating an image slideshow ..... 206
Using the Style Object's Animation Properties ..... 207

*Part IV: Beyond the Basics ..... 211*

**Chapter 14: Searching with Regular Expressions ..... 213**

Finding It Out with Regular Expressions ..... 213
Writing Regular Expressions ..... 215
Using the RegExp object ..... 216

Regular expression literals ..... 217
Testing regular expressions ..... 219
Special character in regular expressions ..... 219
Using Modifiers ..... 220
Coding with Regular Expressions ..... 221

**Chapter 15: Understanding Callbacks and Closures ..... 225**

What Are Callbacks? ..... 225
Passing functions as arguments ..... 226
Writing functions with callbacks ..... 226
Using named callback functions ..... 227
Understanding Closures ..... 230
Using Closures ..... 233

**Chapter 16: Embracing AJAX and JSON ..... 237**

Working Behind the Scenes with AJAX ..... 237
AJAX examples ..... 238
Viewing AJAX in action ..... 240
Using the `XMLHttpRequest` object ..... 243
Working with the same-origin policy ..... 245
Using CORS, the silver bullet for AJAX requests ..... 247
Putting Objects in Motion with JSON ..... 248

***Part V: JavaScript and HTML5 ..... 253***

**Chapter 17: HTML5 APIs ..... 255**

Understanding How APIs Work ..... 255
Checking HTML5 API browser support ..... 256
Getting to know HTML5's APIs ..... 257
Using Geolocation ..... 259
What does geolocation do? ..... 259
How does geolocation work? ..... 260
How do you use geolocation ..... 261
Combining geolocation with Google maps ..... 263
Accessing Audio and Video ..... 266

**Chapter 18: jQuery ..... 271**

Writing More and Doing Less ..... 271
Getting Started with jQuery ..... 272
The jQuery Object ..... 273
Is Your Document Ready? ..... 274
Using jQuery Selectors ..... 274
Changing Things with jQuery ..... 275

Getting and setting attributes ... 276
Changing CSS ... 276
Manipulating elements in the DOM ... 277
Events ... 278
Using on( ) to attach events ... 279
Detaching with off( ) ... 280
Binding to events that don't exist yet ... 281
Other event methods ... 281
Effects ... 282
Basic effects ... 282
Fading effects ... 282
Sliding effects ... 283
Setting arguments for animation methods ... 283
Custom effects with animate( ) ... 283
Playing with jQuery animations ... 284
AJAX ... 285
Using the ajax( ) method ... 285
Shorthand AJAX methods ... 287

*Part VI: The Part of Tens ... 289*

**Chapter 19: Ten JavaScript Frameworks and Libraries to Learn Next ... 291**

Angular JS ... 291
Backbone.js ... 293
Ember.js ... 294
Famo.us ... 295
Knockout ... 296
QUnit ... 297
underscore.js ... 297
Modernizr ... 298
Handlebars.js ... 299
jQuery ... 300

**Chapter 20: Ten Common JavaScript Bugs and How to Avoid Them ... 303**

Equality Confusion ... 304
Avoiding misuse of assignment ... 304
Dodging the equals pitfalls ... 304
Mismatched Brackets ... 305
Mismatched Quotes ... 306
Missing Parentheses ... 306
Missing Semicolon ... 307
Capitalization Errors ... 307

Referencing Code Before It's Loaded ..... 307
Bad Variable Names ..... 310
Scope Errors ..... 310
Missing Parameters in Function Calls ..... 310
Counting Errors: Forgetting That JavaScript Counts from 0 ..... 311

**Chapter 21: Ten Online Tools to Help You Write Better JavaScript ..... 313**

JSLint ..... 313
JSFiddle.net ..... 314
JSBin ..... 315
javascriptcompressor.com ..... 316
jsbeautifier.org ..... 317
JavaScript RegEx generator ..... 318
JSONformatter ..... 319
jshint.com ..... 320
Mozilla Developer Network ..... 321
Douglas Crockford ..... 322

*Index ..... 325*

# Introduction

JavaScript is hot! What started as a quick-and-dirty language created for one of the first web browsers has turned into the world's most popular programming language. Demand for JavaScript programmers is at an all-time high and only continues to grow.

This book is your key to becoming proficient in the core concepts of JavaScript. Whether your goal is to land a high-paying job as a programmer or to make your own personal website more interactive, you can be confident that the content and techniques presented in this book are fully up to date with the most current JavaScript standards and best practices.

Coupled with engaging and interactive online exercises, each chapter contains complete examples of real code that you can try and test in your own web browser at home.

Just as the only way to Carnegie Hall is to practice, practice, practice, the only way to become a better programmer is to code, code, code!

## About This Book

This book is a friendly and approachable guide to getting started with writing JavaScript code. As programming languages go, JavaScript is fairly easy to pick up and start using. Because it's so accessible, many people who started as web page authors have found themselves in the position of being responsible for maintaining, modifying, and writing JavaScript code. If that describes you, this book will quickly and easily bring you up to speed.

Whether you know a little JavaScript or you've never seen it, this book shows you how to write JavaScript the right way.

Topics covered in this book include the following:

- Understanding the basic structures of JavaScript programs
- Integrating JavaScript with HTML5 and CSS3
- Structuring your programs with functions
- Working with JavaScript Objects

- Using advanced JavaScript techniques, such as AJAX, callbacks, and closures
- Getting started with jQuery

Learning JavaScript isn't only about learning the syntax of the language. It's also about accessing the tools and community that has been built around the language. Professional JavaScript programmers have greatly refined the tools and techniques used to write JavaScript over the language's long and exciting history. Throughout the book, we mention important best practices and tools for testing, documenting, and writing better code faster!

To make this book easier to read, keep in mind the following:

- As a convention for this book, all JavaScript code and all HTML and CSS markup appears in monospaced type like this:

```
document.write("Hi!");
```

- The margins on a book page don't have the same room as your monitor likely does. Therefore, long lines of HTML, CSS, and JavaScript may break across multiple lines. Remember that your computer sees such lines as single lines of HTML, CSS, or JavaScript. We indicate that everything should be on one line by breaking it at a punctuation character or space and then indenting any overage, like so:

```
document.getElementById("anElementInTheDocument").
          addEventListener("click",doSomething,false);
```

- HTML and CSS don't care very much about whether you use uppercase or lowercase letters or a combination of the two, but JavaScript cares a lot! In order to make sure that you get the correct results from the code examples in the book, always stick to the same capitalizations that we use.

# Foolish Assumptions

We have a policy at our company, WatzThis?, to never assume (but, frankly, Eva is better at following the policy than Chris is). If you were ever 12 years old, you've probably heard the saying about what happens when you assume. If you don't know, email us.

You don't need to be a programming ninja or a hacker to understand programming. You don't need to understand how the guts of your computer work. You don't even need to know how to count in binary.

However, we do need to make a couple of assumptions about you. We assume that you can turn your computer on, that you know how to use a mouse and a keyboard, and that you have a working Internet connection and web browser. If you already know something about how to make web pages (it doesn't take much!), you have a jump start on the material.

The other things you need to know to write and run JavaScript code are details we cover in this book. And the one thing you'll find to be true is that programming requires attention to details.

# Icons Used In This Book

Here's a list of the icons we use in this book to flag text and information that's especially noteworthy:


This icon highlights helpful tips that show you easy ways or shortcuts that will save you time or effort.


Whenever you see this icon, pay special attention. You won't want to forget the information you're about to read.


Be careful — very careful. This icon warns you of pitfalls to avoid.


This icon highlights the great exercises you can find on the website. If you're interested in trying your hand at JavaScript, go online and visit `www.dummies.com/go/codingwithjavascript`.


This icon highlights technical details that you may or may not find interesting. Feel free to skip this information, but if you're the techie type, you might enjoy reading it.

# Beyond the Book

Here's where you can find the online content for this book:

- **Exercises:** You can find all the exercises online by going to `www.dummies.com/go/codingwithjavascript` to access the exercises at Codeacademy.
- **Examples:** You can find all the examples in the chapters at `www.dummies.com/go/codingwithjavascript`. Here you will find a directory labeled by chapter. Within the chapter, you will find each example labeled by its listing number
- **Cheat Sheet:** You can find lists of useful information at `www.dummies.com/cheatsheet/codingwithjavascript`.
- **Extras:** You can even find additional articles related to each part of the book. You can access this extra content at `www.dummies.com/extras/codingwithjavascript`.
- **Updates:** From time to time, we will need to make updates to a book. Code and specifications are constantly changing, so the commands and syntax that work today may not work tomorrow. You can find this information at `www.dummies.com/extras/codingwithjavascript`.

# Where to Go from Here

Coding with JavaScript is fun, and once you get a little knowledge under your belt, the world of interactive web applications is your oyster! So buckle up! We hope you enjoy the book and our occasional pearls of wisdom.

# Part I

# Getting Started with JavaScript



Visit http://www.dummies.com for great Dummies content online.

## *In this part . . .*

- Find out how to write your first JavaScript program.
- Get the inside scoop on how to work with variables and arrays.
- Discover how to work with operators, expressions, and statements.
- Use loops and branches in your JavaScript coding.
- Visit `http://www.dummies.com` for great Dummies content online.

# Chapter 1

# The World's Most Misunderstood Programming Language

### *In This Chapter*

- Getting to know JavaScript
- Figuring out what JavaScript does
- Understanding why you need JavaScript

*"People understand me so poorly that they don't even understand my complaint about them not understanding me."*

— Søren Kierkegaard

JavaScript hasn't always been as highly regarded as it is today. Some people have called it the best and worst programming language in the world. Over the last few years, there have been a great number of improvements made to the way programmers write JavaScript and to JavaScript interpreters. These improvements have made JavaScript a much better language today than it's been in the past.

In this chapter, you discover what JavaScript is and a little bit of the history of the language. You also find out what JavaScript does and why you need to know it.


Don't forget to visit the website to check out the online exercises relevant to this chapter!

# What Is JavaScript?

Back in the very early days of the web, browsers were simple readers for web pages (see Figure 1-1). They had virtually no capabilities themselves, except for the ability to display text in various sized fonts. As soon as Microsoft released its Internet Explorer browser, the browser wars were on, and the features started flying! One browser introduced the ability to display images, then another introduced the capability to have different fonts, and then blinking text, moving text, and all sorts of other wacky capabilities were introduced!

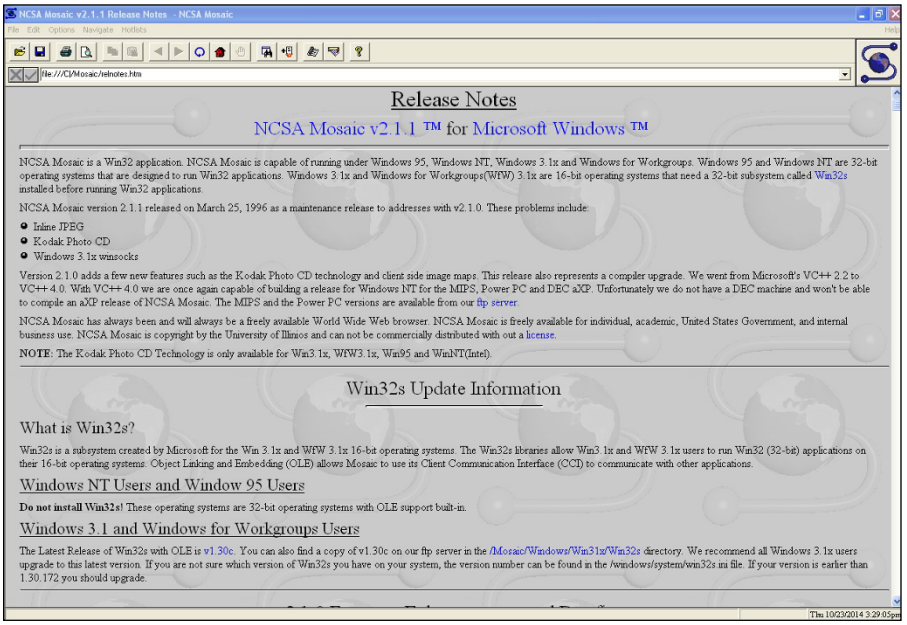


**Figure 1-1:** The first web browsers weren't much to look at.

It wasn't long before someone got the idea that browsers could actually do useful things themselves, rather than just acting as fancy document display programs.

## The Eich-man cometh

JavaScript got its start back in 1995 at Netscape. The creator of JavaScript, Brandon Eich, wrote JavaScript in record time (some say in as few as ten days!) by borrowing many of the best features from various other programming languages. The rush to market also created some interesting quirks (or, less politely described, mistakes) in the design of the language. The result is a sort of Esperanto-like language that looks deceptively familiar to people who are experienced with other programming languages.

## Mocha-licious

The original name of JavaScript was Mocha. It was renamed LiveScript with the first beta deployment of Netscape Navigator and was then changed to JavaScript when it was built into the Netscape 2 browser in 1995. Microsoft very quickly reverse-engineered JavaScript and introduced an exact clone of it in Internet Explorer, calling it Jscript in order to get around trademark issues.

Netscape submitted JavaScript to the standards organization known as Ecma International, and it was adopted and standardized as EMCAScript in 1997.


Brandon Eich, the creator of JavaScript, famously commented about the name of the standardized language; stating that ECMAScript was an "unwanted trade name that sounds like a skin disease."


Not only is ECMAScript an unappealing name for a programming language, the name given to the language by Netscape and which most people refer to it as, is rather unfortunate as well. If you already know how to program in Java or if you learn how to at some point, it's a very good idea to keep in mind that the two languages may have some similarities, but they are, in fact, quite different animals.

## We need more effects!

When JavaScript debuted, it quickly became very popular as a way to make web pages more dynamic. So-called Dynamic HTML (DHTML) was an early result of JavaScript being built into web browsers, and it enabled all sorts of fun effects, like the falling snowflake effect (see Figure 1-2), pop-up windows, and curling web page corners, but also more useful things like drop-down menus and form validation.

## JavaScript grows up

Now entering its third decade, JavaScript has become the world's most widely used programming language and virtually every personal computer in the world has at least one browser on it that can run JavaScript code.

JavaScript is flexible enough that it can be used and learned by nonprogrammers, but powerful enough that it can (and is) used by professional programmers to enable functionality on nearly every website on the Internet today, ranging from single-page sites to gigantic sites like Google, Amazon, Facebook, and many, many others!

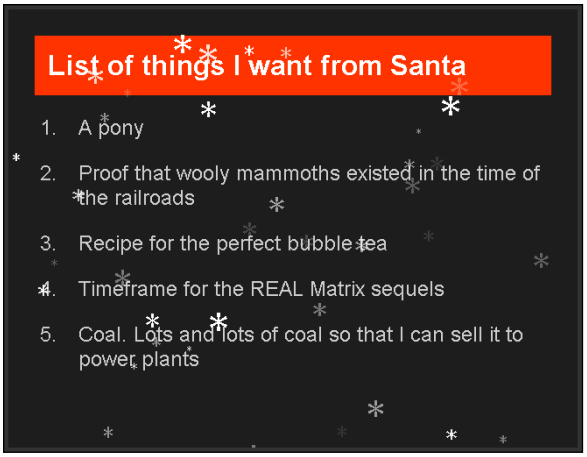


**Figure 1-2:** JavaScript made it possible to have snowflakes falling on your web page.

## Dynamic scripting language

JavaScript is often described as a *dynamic scripting language*. In order to understand what this means, we need to first define a couple of terms and provide some context.

### Common misconceptions about JavaScript

Over the years, JavaScript has had some pretty nasty things said about it. While sometimes rumors are interesting, they aren't always true. The following list explains some common misconceptions about JavaScript:

- **Myth:** JavaScript is not a real programming language. **Reality:** JavaScript is often used for trivial tasks in web browsers, but that doesn't make it any less of a programming language. In fact, JavaScript has many advanced features that have raised the bar for programming languages and are now being imitated in languages such as PHP, C++, and even Java.
- **Myth:** JavaScript is related to Java. **Reality:** Nope. The name JavaScript was invented purely as a marketing strategy because Java was incredibly popular at the time JavaScript came out.
- **Myth:** JavaScript is new. **Reality:** JavaScript has been around for over 20 years! Some of the professional JavaScript programmers we know weren't even born when JavaScript was created.
- **Myth:** JavaScript is buggy and runs differently in different browsers. **Reality:** While this used to be true in some cases, browser makers decided to support the standardized version of JavaScript long ago. Every browser will run JavaScript the same today.

*Computer programs* are sets of instructions that cause computers to do things. Every computer programming language has a set of instructions and a certain way that humans must write those instructions. The computer can't understand these instructions directly. In order for a computer to understand a programming language, it needs to go through a conversion process that translates human-readable (and writable) instructions into machine language. Depending on when this translation takes place, programming languages can be roughly divided into two types: compiled and interpreted (see Figure 1-3).

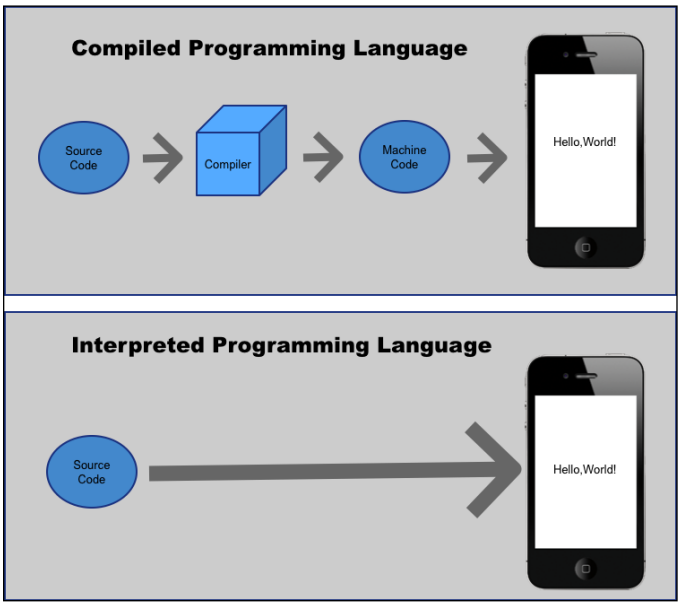


**Figure 1-3:** Programming languages are classified according to when the compilation takes place.

## Compiled programming languages

*Compiled programming languages* are languages in which a programmer must write the code and then run it through a special program called a *compiler* that interprets the given code and then converts it into machine language. The computer can then execute the compiled program.

Examples of compiled languages include C, C++. Fortran, Java, Objective-C, and COBOL.

## Interpreted programming languages

*Interpreted languages* are technically still compiled by the computer into machine language, but the compiling takes place by the user's web browser

right as the program is being run. Programmers who write interpreted languages don't need to go through the step of compiling their code prior to handing it off to the computer to run.

The benefit of programming in an interpreted language is that it's easy to make changes to the program at any time. The downside, however, is that compiling code as it's being run creates another step in the process and can slow down the performance of programs.

Partially because of this performance factor, interpreted languages have gotten a reputation for being less than serious programming languages. However, because of better just-in-time compilers and faster computer processors, this perception is rapidly changing. JavaScript is having a big impact in this regard.

Examples of interpreted programming languages include PHP, Perl, Haskell, Ruby and of course, JavaScript

# What Does JavaScript Do?

If you use the web, you're making use of JavaScript all the time. The list of things that can be enabled with JavaScript is extensive and ranges from simple notices you get when you forget to fill out a required field on a form to complex applications, such as Google Docs or Facebook. Here's a short list of the most common uses for JavaScript on the web:

- Nifty effects
- Input validation
- Rollover effects
- Drop-down/fly-out menus
- Drag and drop features
- Infinitely scrolling web pages
- Autocomplete
- Progress bars
- Tabs within web pages
- Sortable lists
- Magic Zoom (see Figure 1-4)

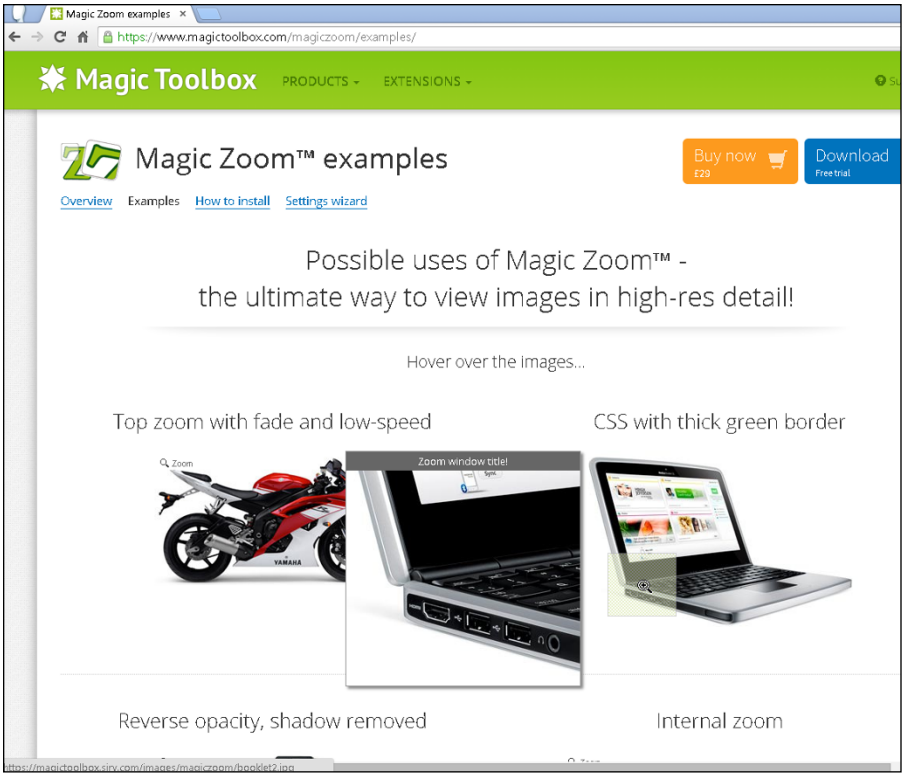


**Figure 1-4:** So-called Magic Zoom effects are enabled using JavaScript.

# Why JavaScript?

JavaScript has become the standard for creating dynamic user interfaces for the web. Pretty much any time you visit a web page with animation, live data, a button that changes when you hover over it, or a drop-down menu, JavaScript is at work. Because of its power and ability to run in any web browser, JavaScript coding is the most popular and necessary skill for a modern web developer to have.

## JavaScript is easy to learn

Keep in mind that programming languages were created in order to give people a simple way to talk to computers and tell them what to do. Compared with machine language, the language that the computer's CPU speaks, every programming language is easy and understandable. To give

you a sample of what sort of instructions your computer is actually obeying, here is a machine language program to write out `"Hello World"`.

```
b8    21 0a 00 00
a3    0c 10 00 06
b8    6f 72 6c 64
a3    08 10 00 06
b8    6f 2c 20 57
a3    04 10 00 06
b8    48 65 6c 6c
a3    00 10 00 06
b9    00 10 00 06
ba    10 00 00 00
bb    01 00 00 00
b8    04 00 00 00
cd    80
b8    01 00 00 00
cd    80
```

Now look at one way you can accomplish this simple task with JavaScript:

```
alert("Hello World");
```

Much easier, yes?


Once you learn the basic rules of the road (called the *syntax*), such as when to use parentheses and when to use curly brackets ({}), JavaScript actually resembles plain old English.


The first step in learning any language, including programming languages, is to get over your fear of getting started. JavaScript makes this easy. There are thousands of sample bits of JavaScript code on the web that anyone can just pick up and start messing around with. You already have all the tools you need (see Chapter 2), and it's easy to start small with JavaScript and gradually build up to making great and wonderful things.

## Where is JavaScript? JavaScript is everywhere!

Although JavaScript was originally designed to be used in web browsers, it has found a home in many other places. Today, JavaScript runs on smartphones and tablets, on web servers, in desktop applications, and in all sorts of portable devices.

### JavaScript in the web browser

The most common place to find JavaScript, and what it was originally designed to do, is running in web browsers. When JavaScript runs in this way, it's called *client-side JavaScript*.

Client-side JavaScript adds interactivity to web pages. It accomplishes this in several ways:

- By controlling the browser itself or making use of functionality of the browser
- By manipulating the structure and content of web pages
- By manipulating the styles (such as fonts and layout) of web pages
- By accessing data from other sources

In order to understand how JavaScript is able to manipulate the structure and style of web pages, you need to know a little bit about HTML5 and CSS3.

#### HTML5

Hypertext Markup Language (HTML) is the language used to structure web pages. It works by marking up content (text and images) to give web browsers information about the content, such as what is a heading, what is a paragraph, where an image goes, and so on. Listing 1-1 shows a simple HTML document. Figure 1-5 shows how a web browser displays this document.

**Listing 1-1: A Simple HTML Document**

```
<!DOCTYPE html>
<html>
<head>
 <title>Hello, HTML!</title>
</head>
<body>
 <h1>This is HTML</h1>
 <p id="introduction">This simple document was written
           with Hypertext Markup Language.</p>
</body>
</html>
```

Here is everything you need to know about HTML right now in order to move forward with learning JavaScript:

- In HTML, the characters surrounded by angle brackets are called *tags*.
- The *ending tag* (which comes after the content being marked up) has a slash after the first angle bracket. For example `</p>` is an ending tag.

- A group of two tags (beginning and ending), plus the content in between them, is called an *element*.
- Elements are generally organized in a hierarchal way (with elements nested within elements).
- Elements may contain name/value pairs, called *attributes*. If an element has attributes, they go in the beginning tag. *Name/value pairs* assign values, in quotes, to names (which aren't in quotes) by putting an equals sign between them. For example, in the following tag, `width` and `height` are both attributes of the `div` element:

```
<div width="100" height="100"></div>
```

- Some elements don't have content and therefore don't need an ending tag. For example, the `img` tag, which simply inserts an image into a web page, looks like this:

```
<img src="myimage.jpg" width="320" height="200"
        alt="Here is a picture of my dog.">
```

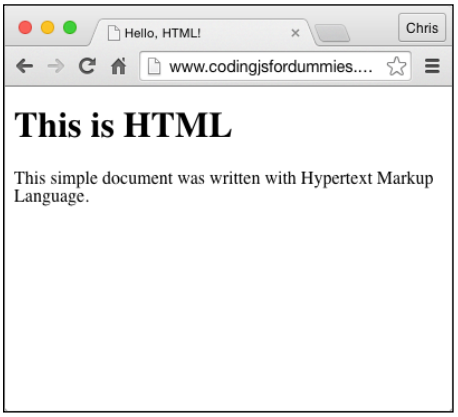


**Figure 1-5:** Web browsers use HTML to render web pages.

All the data necessary to show the image is included in the beginning tag using attributes, so the `img` tag doesn't require an ending tag.

When you write a web page with HTML, you can include JavaScript code directly in that document, or you can reference JavaScript code file (which end in `.js`) from the HTML document. Either way, your viewer's web browser will download the JavaScript code and run it when a user accesses a web page containing that JavaScript.


Client-side JavaScript runs inside of your users' web browsers.

### CSS3

*Cascading Style Sheets* (CSS) is the language used to add formatting and different layouts to web pages. The word *style,* when used in CSS, refers to many aspects of how the HTML document is presented to the user, including

- Typefaces (or font faces)
- Type size
- Colors
- Arrangement of elements in the browser window
- Sizes of elements
- Borders
- Backgrounds
- Creation of rounded corners on element borders

Like JavaScript, CSS can be either placed directly into an HTML document, or it can be linked to from the HTML document. Once it's downloaded, it immediately does its thing and formats the document according to your specifications.

Style sheets in CSS are made up of CSS rules, which contain properties and values that should be applied to an element or a group of elements. Here's an example of a CSS rule:

```
p{font-size: 14px; font-color: black; font-family: Arial,
          sans-serif}
```

This rule, reading from left to right, specifies that all `p` elements (which indicate paragraphs in HTML) should be displayed in text that is 14px large, black, and using the Arial font. If Arial isn't available on the user's computer, it should be displayed in some sans serif typeface.

The part of the CSS rule that's outside of the curly brackets is called the *selector.* It selects the elements that the properties within the curly brackets apply to.


Throughout this book, you find out how to use JavaScript with HTML and CSS. We provide just enough information here to be able to show you how HTML and CSS work. If you need to learn more, you can find some excellent books about them. One that we highly recommend is *Beginning HTML5 and CSS3 For Dummies* by Ed Tittel and Chris Minnick (Wiley).

## JavaScript is powerful!

JavaScript running in a web browser used to be slow, and JavaScript got a bad reputation early on among programmers. Today, JavaScript code runs 80 percent as fast as compiled code. And, it keeps getting faster all the time. What this means is that today's JavaScript is much more powerful than the JavaScript of just a few years ago. And, it's many times more powerful than the JavaScript that was first introduced in 1995.

## JavaScript is in demand

JavaScript is not only the most widely known programming language, it's also one of the most in-demand skills in the job market. It's projected that the job market for JavaScript programmers will increase by 22 percent between 2010 and 2020. Exciting things are happening with JavaScript, and there has never been a better time than right now to learn it.

# Chapter 2

# Writing Your First JavaScript Program

### In This Chapter

- Arranging your development environment
- Getting to know JavaScript code
- Understanding a simple JavaScript program
- Understanding the value of commenting your code

*"The secret of getting ahead is getting started."*

— Mark Twain

Simple JavaScript programming isn't difficult to understand. In this chapter, you go through the process of setting up your computer for writing JavaScript. You also write your first JavaScript program and get to know the basic syntax behind everything you'll do with JavaScript in your future as a programmer.


Don't forget to visit the website to check out the online exercises relevant to this chapter!

## Setting Up Your Development Environment

It's important to have all of your tools set up and in place before beginning to write your first JavaScript program. We walk you through the process of downloading and installing our favorite JavaScript development tools, which

are, coincidentally, the ones we use in this book. If you have similar tools that you prefer, please feel free to use those. However, we recommend that you still read this section of the book in order to learn why we've chosen these tools and to make your own decisions about whether to use them.

After you install each of the tools, we share some tips and tricks with you for how to get the most out of each of them.

## Downloading and installing Chrome

The web browser that we prefer to use when working with JavaScript is Google Chrome. If you prefer to use a different web browser day to day, that's fine, of course. All browsers will run JavaScript very fast and correctly. However, some of the instructions in this book will be specific to Google Chrome, so we recommend that you at least go through the process of installing it on your computer in this chapter. We chose to use Google Chrome in this book because it offers excellent tools for making JavaScript programmers' jobs easier and because it's currently the most widely used web browser on the Internet. (Yes, it's even more popular than Internet Explorer.)

If you don't have Chrome installed, follow these steps to install it:

1. **Go to `www.google.com/chrome`.**

   Figure 2-1 shows you what Google Chrome looks like.

2. **Hover over the Download tab and choose the appropriate version for your computer.**
3. **Open the downloaded file and follow the instructions to install Chrome.**

### Now you have a supercharged JavaScript engine!

Google Chrome uses Google's V8 JavaScript engine to parse, compile, and run JavaScript code. Depending on whose benchmarking test you believe, Chrome is either the fastest way to run JavaScript in a browser, or it's one of the fastest. The major browser makers are constantly competing to outdo each other. It doesn't matter too much who is actually the fastest at any one time; the competition has increased the speed of every browser's JavaScript engine by leaps and bounds in recent years.

If you want to see actual comparisons of how different browsers do in JavaScript performance tests, you can do so at `http://arewefastyet.com` (see figure). This site, which is maintained by Mozilla, creator of the Firefox browser, automatically checks and graphs JavaScript performance of the most popular browsers and is updated multiple times every day.

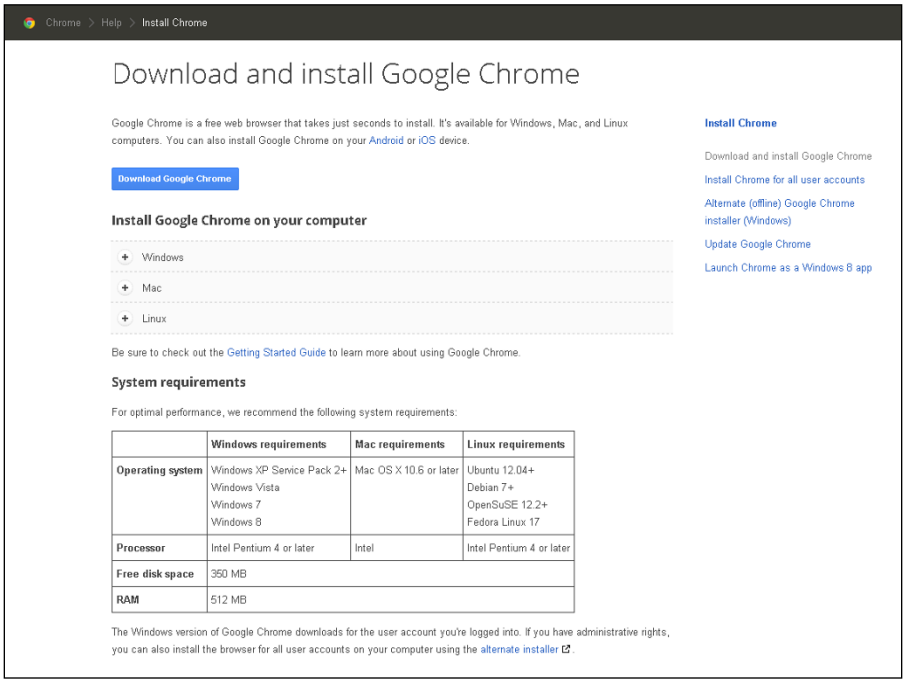


**Figure 2-1:** Installing Chrome is easy on either Mac or Windows.

# Downloading and installing a code editor

A *source code editor,* commonly referred to as code editor, is a text editor with added functionality that helps you write and edit programming code. The one we use is Sublime Text.


There are many code editors to choose from, so if you already have a favorite that you like to use and that you're comfortable with, please use it! A programmer's code editor is a very personal choice, and many people will find that they just feel more comfortable with a specific one. If you find that Sublime Text just doesn't fit your style, Table 2-1 lists some other options.

**Table 2-1 Examples of Other Code Editors**

| *Name* | *Location* | *Compatible with . . .* |
|---|---|---|
| Coda | `http://panic.com/coda` | Mac only |
| Aptana | `www.aptana.com` | Mac or Windows |
| Komodo Edit | `www.activestate.com/komodo-edit/downloads` | Mac or Windows |

*(continued)*

**Table 2-1** *(continued)*

| *Name* | *Location* | *Compatible with . . .* |
|---|---|---|
| Dreamweaver | `http://adobe.com/products/dreamweaver.html` | Mac or Windows |
| Eclipse | `www.eclipse.org` | Mac or Windows |
| Notepad++ | `http://notepad-plus-plus.org` | Windows only |
| TextMate | `http://macromates.com` | Mac only |
| BBEdit | `www.barebones.com/products/bbedit` | Mac only |
| EMacs | `www.gnu.org/software/emacs` | Mac or Windows |
| TextPad | `www.textpad.com` | Windows only |
| vim | `www.vim.org` | Mac or Windows |
| Netbeans | `https://netbeans.org` | Mac or Windows |

We use Sublime Text (see Figure 2-2) for this book because it's popular among JavaScript programmers, and it provides a simple user interface along with a large number of plugins for handling more advanced programming tasks as you gain more programming experience.

To install Sublime Text, follow these steps:

1. **Go to `http://sublimetext.com` and choose the appropriate version for your operating system.**
2. **Open the downloaded file and follow the instructions for installing Sublime Text.**

### *Getting started with Sublime Text*

When you first open Sublime Text, you see a simple blank page with a cursor on it (see Figure 2-3).

If you've used Sublime Text, you may see a sidebar on the left, as shown in Figure 2-4. This sidebar shows your open files and the files in your project, if you've created one. The sidebar is useful, and we recommend that you have it open.


To open the sidebar, click View➪Sidebar➪Show Sidebar.