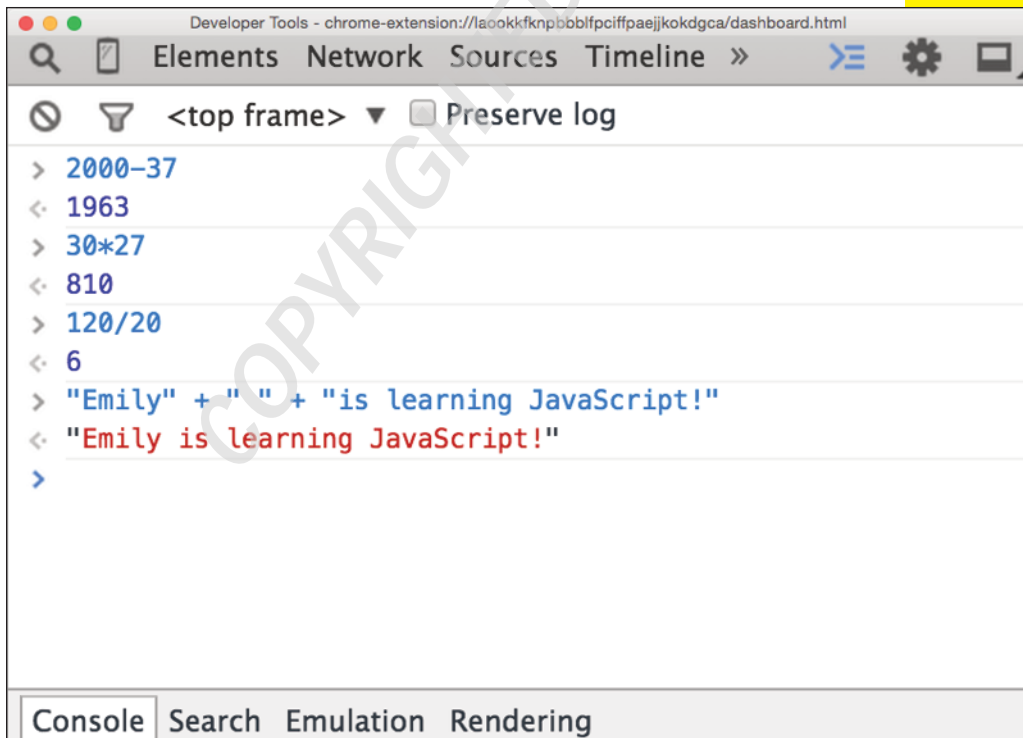


Programming the Web

JavaScript is a powerful language that's easy to learn! In this chapter, we explain the basics of programming, tell you what JavaScript is, and get you started with writing your first JavaScript commands.

One of the most important parts of starting any new project is to make sure you have your workshop stocked with all the correct tools. In this chapter, you install and configure all the programs you need and start experimenting with some real JavaScript programs!



The screenshot shows the Chrome Developer Tools interface. The top bar includes the title "Developer Tools - chrome-extension://laookkfknpbbblfpciffpaejjkokdgc/dashboard.html" and navigation tabs for "Elements", "Network", "Sources", and "Timeline". Below the tabs, there are icons for search, mobile emulation, and settings. The main area displays the console for the "<top frame>" with a "Preserve log" checkbox. The console shows a series of log entries: a range "2000-37", a value "1963", a range "30*27", a value "810", a range "120/20", a value "6", a string concatenation operation `"Emily" + " " + "is learning JavaScript!"`, and the resulting string `"Emily is learning JavaScript!"`. The bottom of the console has tabs for "Console", "Search", "Emulation", and "Rendering".

```
Developer Tools - chrome-extension://laookkfknpbbblfpciffpaejjkokdgc/dashboard.html
Elements Network Sources Timeline »
<top frame> Preserve log
> 2000-37
< 1963
> 30*27
< 810
> 120/20
< 6
> "Emily" + " " + "is learning JavaScript!"
< "Emily is learning JavaScript!"
>
```

What Is Programming?

A *computer program* is a series of instructions that can be understood and followed by a computer. *Computer programming*, also known as *coding*, is what we call it when we write these instructions. Computers can't do things on their own. They need a computer program to tell them what to do. Computer programmers write code to make computers do all sort of things.



Another name for a computer program is *software*.

The women who invented programming

Electronic computers as we know them were first invented in the 1930s. But it was the middle of the 1800s when the first computer program — a set of instructions designed to be carried out by a machine — was written.

The author of the first computer program — and, therefore, the world's first computer programmer — was a woman named Ada Lovelace. A mathematician in England, she was the first person to envision computers that could do much more than just crunch numbers. She foresaw computers being able to do all the things we use computers for today: including working with words, displaying pictures, and playing music. Her unique insights earned her the nickname "The Enchantress of Numbers."

Compilers are programs for converting programming languages into machine language. The first compiler was created by Grace Murray Hopper in 1944. This invention led to computer programs that could run on different types of computers, and eventually to JavaScript. Hopper is also credited with being the inventor of the term *debugging* for fixing problems in computer programs. The term was inspired by the removal of an actual moth from an early computer. Hopper became known as "The Queen of Software" or "Amazing Grace" for her contributions to modern computing.

Computer programs help people to do many thousands of things, including the following:

- ✓ Playing music and videos
- ✓ Performing scientific experiments
- ✓ Designing cars
- ✓ Inventing medicines
- ✓ Playing games
- ✓ Controlling robots
- ✓ Guiding satellites and spaceships
- ✓ Creating magazines
- ✓ Teaching people new skills

Can you think of more examples of things that computers can do?

Talking to Computers

At the heart of every computer is a central processing unit (CPU). This CPU is made up of millions of tiny, very fast switches (called *transistors*) that can be either on or off. The position of each of these switches at any time determines what the computer will do.

Software written by programmers tells these switches when to turn on or off and in what combination by using *binary codes*. Binary codes use zeros and ones to form letters, numbers, and symbols that can be put together in order to perform tasks.

Every single thing that a computer does is the result of a different combination of many zeros and ones. For example, to represent a lowercase letter *a*, computers use the following binary code:

0110 0001

Each zero or one in a binary number is called a *bit*, and a combination of eight bits is called a *byte*. When you hear the words *kilobyte*, *megabyte*, and *gigabyte* used to tell how big a file is, what it's talking about is the number of eight-bit binary codes it takes to store the file.

Table 1-1 lists the most commonly used storage sizes.

Name	Number of Bytes	What It Can Store
Kilobyte (KB)	1,024	Two to three paragraphs of text
Megabyte (MB)	1,048,576	800 pages of text
Gigabyte (GB)	1,073,741,824	250 songs (as MP3s)
Terabyte (TB)	1,099,511,627,776	350,000 digital pictures
Petabyte (PB)	1,125,899,906,842,624	41,943 Blu-ray discs

A typical small computer program might contain anywhere from a couple kilobytes to a couple megabytes of instructions, images, and other data. Because it's unlikely that you have enough time in your busy day to type out thousands, or even millions, of ones and zeros, if you want to tell a computer what to do, you need a translator who speaks both human languages and computer (or *machine*) language. Computer programming languages are this translator.

Every computer program is written using a computer programming language. Programming languages allow you to write complex series of instructions that can be translated (also known as *compiled*) into machine language. Through compilation, these instructions are eventually turned into binary codes that a computer can understand.

Choosing a Language

People have created hundreds of different computer programming languages. You might ask yourself why there are so many programming languages, if they all essentially do the same thing: translate human language into machine language. That's an excellent question!

There are a few main reasons why there are so many different programming languages. New programming languages are written to allow programmers to

- ✔ Write programs in new and better ways than were previously available.
- ✔ Write programs for new or specialized types of computers.
- ✔ Create new kinds of software.

Examples of computer programming languages include the following:

- ✔ C
- ✔ Java
- ✔ JavaScript
- ✔ Logo
- ✔ Objective C
- ✔ Perl
- ✔ Python
- ✔ Ruby
- ✔ Scratch
- ✔ Swift
- ✔ Visual Basic



Our short list of programming languages only scratches the surface. For a more complete list of programming languages, visit http://en.wikipedia.org/wiki/List_of_programming_languages.

With so many programming languages to choose from, how do you know which one to use? In many cases, the answer is determined by what you want to do with the languages. For example, if you want to program apps for the iPhone, you have three choices: Objective C, JavaScript, or Swift. If you want to program games to run on Mac or Windows, you have more choices, including C, Java, or JavaScript. If you want to make an interactive website, you need to use JavaScript.

Are you seeing a pattern here? JavaScript is everywhere.

What Is JavaScript?

In the early days of the web, every web page consisted of nothing but plain text in different sizes with links between pages. There were no web forms, there certainly wasn't any animation, and there weren't even different styles of text or pictures!

We're not complaining! When the web was new, it was exciting to click from page to page and discover new things. Even more exciting was how easy the web made it for anyone to be able to publish anything at all and have the potential for anyone else on the Internet to read it.

But when people got a taste of what the web could do, they wanted more features! Graphics, text colors, forms, and many other features were introduced very quickly.

Of all the things that were invented in the earliest days of the web, the thing that has had the biggest impact over the longest time was JavaScript.

JavaScript was created in order to make it possible for web browsers to be interactive. Interactive web pages can range from simple

forms that provide feedback when you make a mistake, to 3D games that run in your web browser. Whenever you visit a website and see something moving, or you see data appearing and changing on the page, or you see interactive maps or browser-based games, chances are, it's JavaScript at work.

To see some examples of websites that are made possible by JavaScript, open up your web browser and visit the following sites:

- ✔ **ShinyText** (<http://cabbi.bo/ShinyText/>): ShinyText is an experimental website that uses JavaScript to display a word. You can adjust different properties of the word, such as Reflection Power and Repulsion Power to see what effect these changes have on how the letters in the word react when you move them around with your mouse. Figure 1-1 shows ShinyText in action.



Even if you don't understand how it works (we sure don't!), ShinyText is fun to play with, and it's a great example of what's possible with JavaScript.

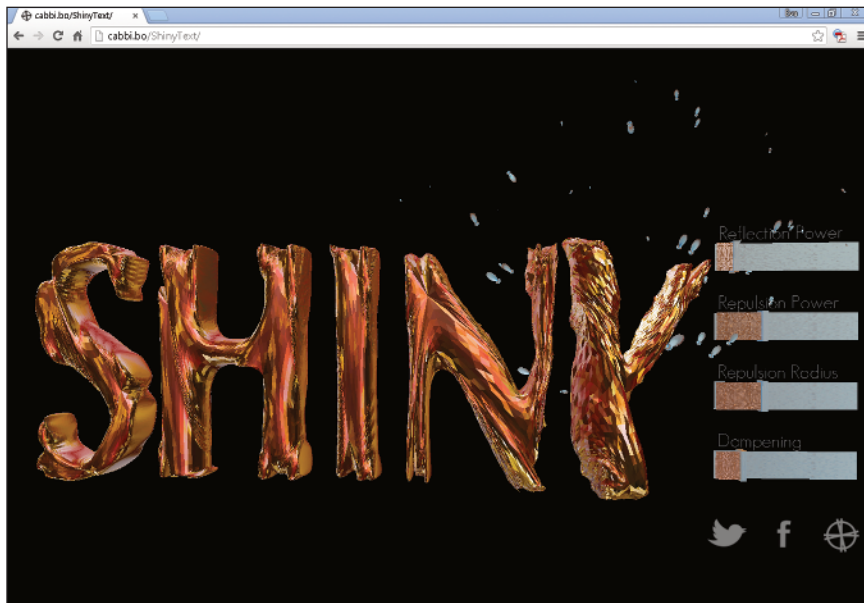


Figure 1-1: ShinyText uses JavaScript to produce a 3D physics simulation.

- ✔ **Interactive Sock Puppet (www.mediosyproyectos.com/puppetic/):** Interactive Sock Puppet is another 3D animation. This time, you can control the movements and facial expressions of a JavaScript puppet. Figure 1-2 shows the Interactive Sock Puppet looking quite happy.

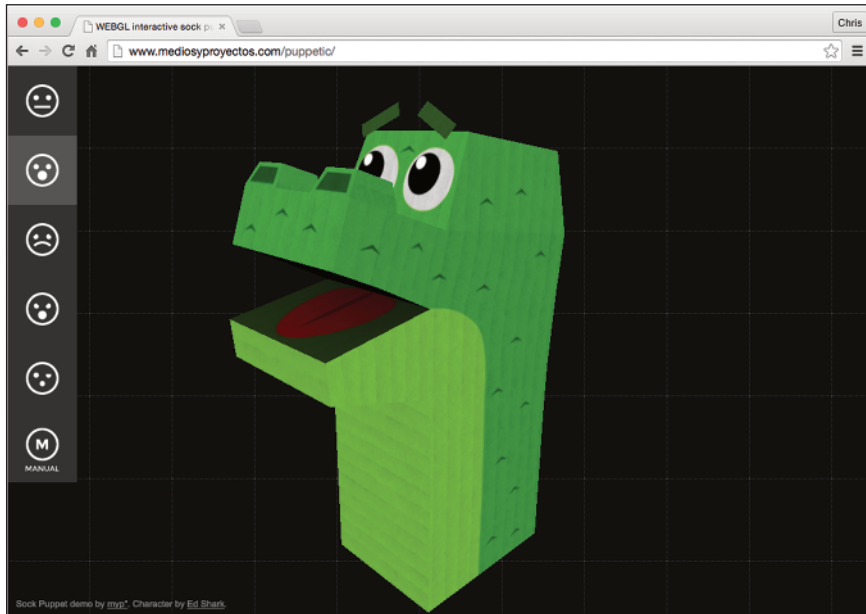


Figure 1-2: Interactive Sock Puppet lets you control a JavaScript dinosaur sock puppet.

- ✔ **Facebook (www.facebook.com):** Facebook uses a lot of JavaScript (see Figure 1-3). When you see a smooth animation or video playback, or when a list of posts updates by itself, that's JavaScript at work!



TIP Some of these examples use some very advanced features of web browsers. We recommend that you use the latest version of Google Chrome to view these. The examples may not work in older web browsers.

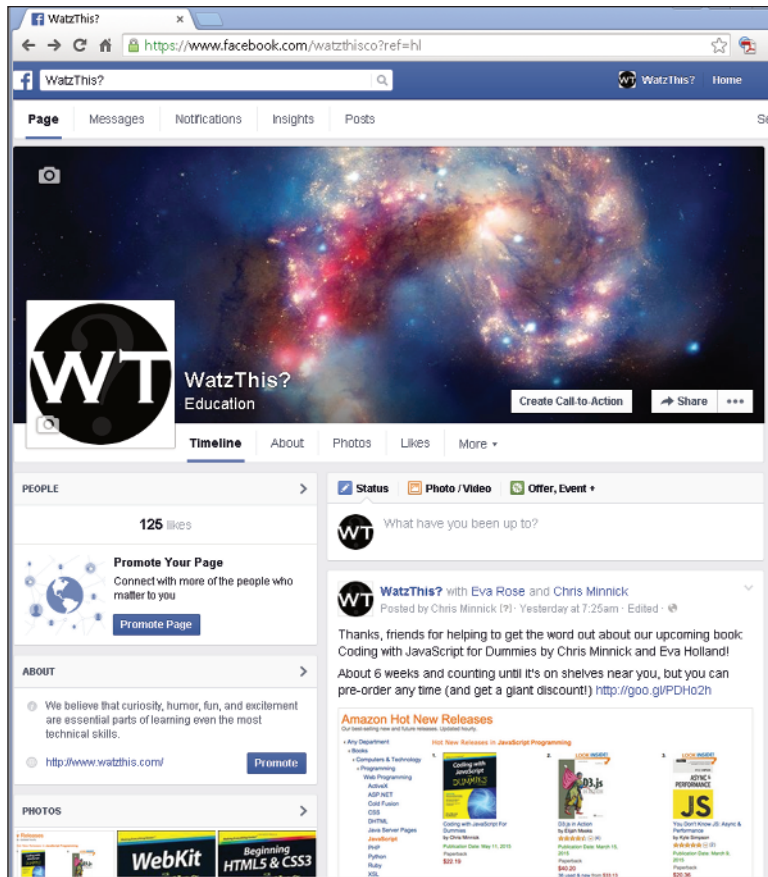


Figure 1-3: Facebook uses JavaScript to do everything.

Get Your Browser Ready

The one essential tool that you need for working with JavaScript is a web browser. You have many different web browsers to choose from, and nearly all of them will do a great job running JavaScript. Odds are, you already have a web browser on your computer.

The most widely used web browsers today are Firefox, Safari, Chrome, Internet Explorer, and Opera. For this book, we'll be using Chrome. Google Chrome is currently the most popular web browser. It has a number of great tools for working with JavaScript.

If you don't already have Chrome installed, you'll need to download and install it. You can install Chrome by opening any web browser and going to www.google.com/chrome/browser/desktop. Follow the instructions found on that page to install Chrome on your computer. When you have Chrome installed, start it up.

In the next section, we show you the Chrome Developer Tools, which help website designers and JavaScript programmers to see exactly what's going on inside the browser so they can write better web pages and programs.

Opening the Web Developer Tools

After you have Chrome installed and launched, look at the top of the browser window. In the upper-right corner, you see three lines. This is the icon for the Chrome menu. If you expand the Chrome menu, you see a list of options similar to those shown in Figure 1-4.

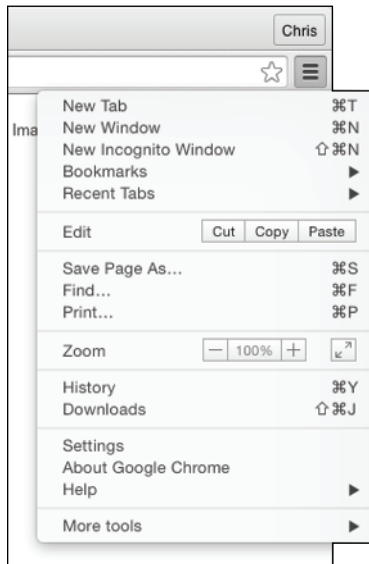


Figure 1-4: The Chrome menu.

If you scroll down to the bottom of this menu and select More Tools, a new menu of options appears, as shown in Figure 1-5. These secret tools are the JavaScript coder's best friends.

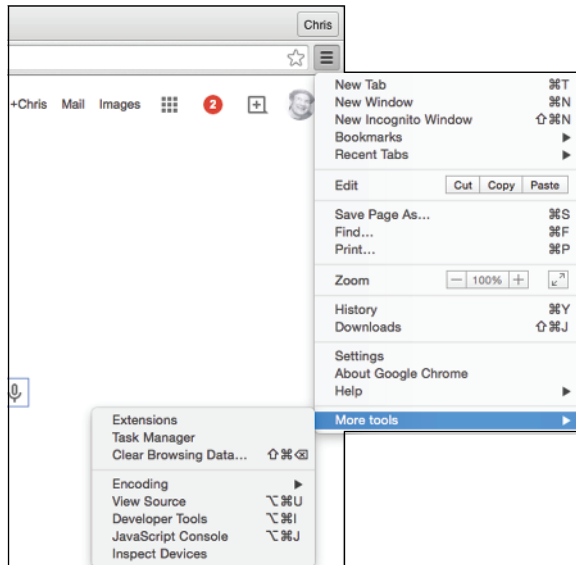


Figure 1-5: The MoreTools menu.

Select Developer Tools from the More Tools menu. A new panel opens at the bottom of your browser window that looks like Figure 1-6.

The Developer Tools give you all the information you need for finding out how any web page works, for testing and improving your own web pages and JavaScript programs, and much more.

Notice that there's a menu at the top of the Developer Tools with different options, including Elements, Network, Sources, Timeline, Profiles, Resources, Audits, and Console. If you click each of these, you'll see a different set of options and data in the Developer Tools panel.

We describe the different components of the Developer Tools as they become necessary throughout this book, but for now, the most important part of the Developer Tools is the one labeled Console. Click the Console tab now.

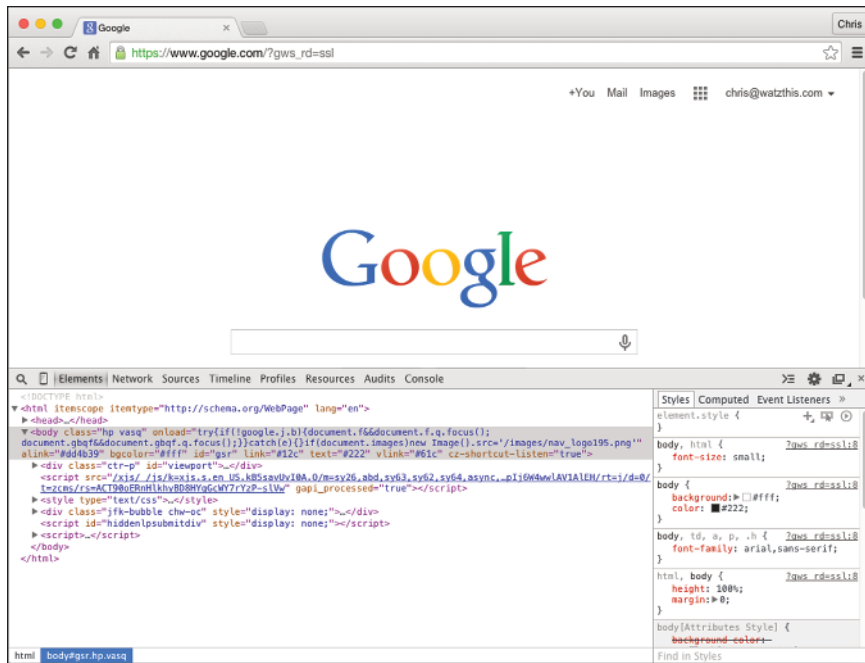


Figure 1-6: The Developer Tools.

Introducing the JavaScript Console

The Developer Tools Console, also known as the JavaScript Console, shown in Figure 1-7, gives you information about the JavaScript that's currently running in the browser window.

If there are errors in the JavaScript code of a web page, you see information about the errors in the console. This is a very helpful tool and one of the main features of the JavaScript Console.

Another very cool capability of the console is that you can type JavaScript into the console panel and it will run. In the next section, you learn why this is useful and how to do it.



The JavaScript Console is a useful tool for JavaScript programmers, but it also has the potential to be misused. If someone you don't know or trust asks you to paste code into the JavaScript Console, make sure you understand what that code does first.

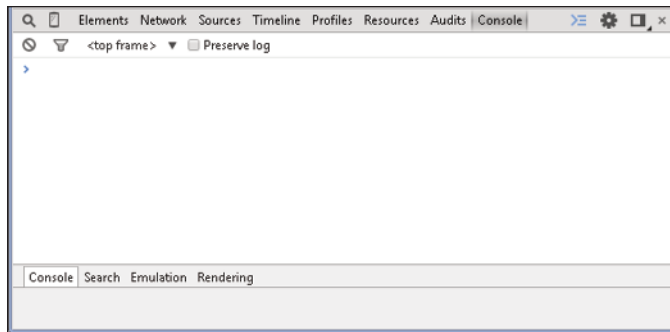


Figure 1-7: The JavaScript Console.

Running Your First JavaScript Commands

Now it's time to start experimenting with some real JavaScript code! If you don't already have it open, open the JavaScript Console by selecting it from the Other Tools menu under the Chrome menu, or by clicking the Console tab in the Developer Tools.

Follow these steps to run your first JavaScript commands:

1. Click inside the JavaScript console, near the `>`, to start inserting code.
2. Type `1 + 1` and then press Return (Mac) or Enter (Windows).

The browser gives you the answer on the next line.

Notice that when the answer is returned to you, it has an arrow on the left side of it that points to the left. This arrow indicates that the value came from JavaScript rather than from your input. Any value that comes from JavaScript is called a *return value*. Every command that you run in JavaScript produces some sort of return value.

Simple math is one thing, but JavaScript can do much, much more. Let's try out some other commands and see just how quickly we can get some answers around here.

Before we get started, let's clean up the console and remove any previous commands, errors, and return values in there. To clear the console, look at the upper-left corner and click the circle with the line through it. Everything inside the console will be erased, and now you've got a clean slate.

Click your mouse next to the `>` and try out the following JavaScript commands. Make sure to press Return (Mac) or Enter (Windows) after each one to see the results.

JavaScript Command	Description
<code>2000 - 37</code>	This is a simple math problem, but this time we're using the minus sign to subtract the number on the right from the number on the left.
<code>30 * 27</code>	The asterisk (*) is how you tell JavaScript to multiply numbers.
<code>120 / 20</code>	The forward slash (/) tells JavaScript to divide the number on the left by the number on the right.
<code>"Your name" + " " + "is learning JavaScript!"</code>	Yes, you can add words together with JavaScript! When you run a command that adds words together, it's called <i>concatenation</i> . The result will be that the words are combined into a single word. Notice that the words in the above JavaScript command are inside quotes. These quotes are very important. We tell you exactly why they're important in Chapter 2.
<code>Your name + + is learning JavaScript!</code>	When you don't use quotes, JavaScript doesn't like that one bit. It returns an error message containing the keyword <code>SyntaxError</code> . A syntax error means that you've written something that isn't valid JavaScript. Any time you see a syntax error, it means that you've goofed. Take a close look at your code and look for typos, missing punctuation, or missing quotes.

Having Fun with Math

Now it's your turn to try out some math problems on your own! Clear out your commands and the return values and errors from the previous section and experiment with the console.

Here are some ideas to get you started:

- ✔ Multiply together two decimal numbers.
- ✔ Run multiple commands in one line (for example, `1 + 1 * 4 / 8`).
- ✔ Type a number without any symbols at all and then run it.
- ✔ Add a word (remember to use quotes!) to a number (without quotes).
- ✔ Add a number (without quotes) to a word (with quotes).
- ✔ Combine your first name with the last name of your celebrity crush. Remember to add a space between the first and last name! For example, `"Eva" + " " + "Harry Styles"`.
- ✔ Try to produce extremely large return values.
- ✔ Try to produce extremely small return values.
- ✔ Try to do an impossible math problem, such as dividing a number by zero.
- ✔ Try multiplying a number by a word (in quotes). For example, `343 * "hi!"`. The result of this will be NaN, which stands for “not a number.”