JavaScript Projects for Kids

JavaScript Projects for Kids is an easy-to-follow, informative, and fun guide that takes a project-based approach to teaching programming in JavaScript. You will learn everything you need to get started with serious web application development.

This book will take your imagination to new heights by teaching you how to work with JavaScript from scratch. It will introduce you to HTML and CSS to enhance the appearance of your applications. You’ll then use your skills to build a cool Battleship game! From there, the book will introduce you to jQuery and show you how you can manipulate the DOM. You’ll get to play with some cool stuff using Canvas and will learn how to make use of Canvas to build a game along the lines of Pac-Man globally, only a whole lot cooler! Finally, this book will show you a few tricks with object-oriented programming (OOP) to make your code clean and will end with a few road maps on areas you can explore further.

Who this book is written for

If you’ve never written code before or you are completely new to the world of web programming, then this book is the right choice for you. This book is for kids aged 10 years and above and adults who are completely new to the world of programming and want to be introduced to programming.

What you will learn from this book

- Learn how to work with Google Developer tools to iterate, debug, and profile your code
- Develop a Battleship game using the basic concepts of HTML and CSS
- Get to know the fundamentals of JavaScript programming
- Create your own version of Pac-Man
- Discover the vital concepts of object-oriented programming
- Learn about several open source projects and play with their source code
- Build your own web applications following the guidelines provided


Syed Omar Faruk Towaha

Gear up for a roller-coaster ride into the world of JavaScript and programming with this easy-to-follow, fun, and entertaining project-based guide.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 2 'Solving Problems Using JavaScript'
- A synopsis of the book’s content
- More information on JavaScript Projects for Kids
Syed Omar Faruk Towaha has degrees in physics and computer engineering. He is a technologist, tech speaker, and physics lover from Shahjalal University of Science and Technology (SUST), Sylhet. He has a passion for programming, tech writing, and physics experiments.

His recent books include Easy Circuits for Kids, Fundamentals of Ruby, and How You Should Design Algorithms. He is an Oracle-certified professional developer currently involved with a number of projects that serve both physics and computer architecture.

He is the president of one of the largest astronomical organizations (Copernicus Astronomical Memorial of SUST (CAM-SUST)) in Bangladesh. He also volunteers for Mozilla as a representative.
As you can guess from the title of the book, this book is designed and set up for kids so that they can teach themselves JavaScript and create some projects using JavaScript.

By abstracting the core web programming in an unparalleled way, JavaScript changed websites and web apps forever. Boring static websites and non-interactive websites have now become quite awesome with the touch of JavaScript. Using JavaScript, you can develop web applications, even smartphone applications too, quickly without compromising quality. You can be very productive and deal with almost no configuration on your hardware and software if you start playing with JavaScript. Please remember that this is not a reference book, but you can learn every basic concepts of JavaScript from it. So, for the kids aged 10 and above, this will be a perfect book to discover the world of JavaScript.

What this book covers

Chapter 1, Exploring JavaScript in the Console, discusses JavaScript and the JavaScript Development Environment, including Google Developer Tools. We will install the necessary software and print a few simple lines of code in this chapter.

Chapter 2, Solving Problems Using JavaScript, covers JavaScript fundamentals from the main syntax to some easy commands in the console. We will learn how variables work and what can be achieved with arithmetic operators. We will also run some simple commands to solve problems inside the console.

Chapter 3, Introducing HTML and CSS, will make real use of JavaScript and will cover HTML, which empowers the readers to make use of JavaScript not only in the console but also in the browser's view. We will also explain the basics of CSS, such as CSS selectors, and CSS layouts.

Chapter 4, Diving a Bit Deeper, covers some of the more advanced features that JavaScript offers. We discuss for and while loops, if statements, and switches-case.
Preface

Chapter 5, Ahoy! Sailing into Battle, teaches us how to develop the famous game, Battleship. Building upon what we've learned in the previous chapters, the tiny tots will learn to put this information into use.

Chapter 6, Exploring the Benefits of jQuery, is all about jQuery, a famous JavaScript library, and the advantages of using it.

Chapter 7, Introducing the Canvas, discusses HTML canvas, and we will learn how we can use it on our projects.

Chapter 8, Building Rat-man!, teaches us to develop a famous game, Pac-Man, except there is a rat, some cats, and lots and lots of cheese balls to eat! ;)

Chapter 9, Tidying up Your Code Using OOP, teaches object-oriented programming (OOP) and discusses how JavaScript is an OOP language.

Chapter 10, Possibilities, shows the reader what is possible using the skills they have developed reading this book.
A block of code is set as follows:

document.write("Hello");
document.write("World");
document.write("!");

**New terms and important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

**Reader feedback**

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book’s title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

**Customer support**

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.
Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from https://www.packtpub.com/sites/default/files/downloads/JavaScript_Projects_for_Kids_ColorImages.pdf.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting http://www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to https://www.packtpub.com/books/content/support and enter the name of the book in the search field. The required information will appear under the Errata section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem.
Before we start talking about lines of codes, objects, variables, and so on, we need to know what JavaScript is. JavaScript is a programming language that is used to add interactivities to the web pages and build web applications. Static websites are not very popular these days, therefore, we use JavaScript to make our websites interactive.

Some people also call it a scripting language as it is an easy language and does not require compilers like other languages. JavaScript was not designed as a general purpose programming language, it was designed to manipulate web pages. You can write a desktop application using JavaScript. JavaScript can also access your machine's hardware. You can try making a desktop application with a software development kit (SDK) such as PhoneGap for mobile or the Microsoft app SDK for desktop. The JavaScript codes are interpreted on web pages and then run by a browser. Any modern Internet browser, for example Firefox, Safari, Google Chrome, UC Browser, Opera, and so on, supports JavaScript.

A compiler is a computer program that processes codes and turns them to machine language. Making a website interactive means adding features that are controlled by the users to the website. For example, online registration forms, online calculator, and so on. The Static website has fixed objects and contents and it displays the same information to all the visitors.

Basically, JavaScript is included on an HTML page or written on a separate file that has a .js extension. If you know nothing about HTML, don't worry as you will learn about it in Chapter 3, Introducing HTML and CSS. So, where can you use JavaScript?
The answer is simple, you can do the following:

- You can create an active user interface.
- You can control web browsers.
- You can validate user inputs (if they are typed wrong).
- You can create custom web pages that can pop up on the browser, holding information or images.
- You can create dynamic pages without Common Gateway Interface (CGI). CGI is used by the web servers to process a browser's information.

The thing that you should remember is JavaScript is not Java, the programming language developed by Sun Microsystems.

Throughout this book, we will use Google Chrome as the default browser and Atom as the text editor.

If you do not have these two software already installed on your computer, it is necessary to download and install them.

We will use the Atom text editor as it is a cross-platform editor, has a built-in package manager, does smart autocompletion, and has a lot of other advantages.

**Installing Google Chrome**

To install Google Chrome go to [http://www.google.com/chrome](http://www.google.com/chrome) and click Download now, as shown in the following screenshot:
Then press the **Accept and Install** button, as shown in the following screenshot:

The installation will be completed depending on your network speed and machine's hardware configurations.

Uncheck **Set Google Chrome as my default browser** if you don't want to use Google Chrome as your default browser.

### Installing Atom

To install the Atom text editor, follow the [https://atom.io/](https://atom.io/) link and press **Download Windows Installer**, as shown in the following screenshot:
You have learned how to print something using JavaScript on console in the previous chapter. Now, let's see the fundamentals behind JavaScript syntax, variables, arithmetic operators, and comments.

In the computer world, there is nothing but data. You can read, modify, and create new data; however, anything that isn't data simply does not exist. In JavaScript, we need to handle data to develop a website.

To understand the basic syntax of JavaScript, first of all you need to know that JavaScript is case sensitive. You cannot interchange lower case and upper case letters in JavaScript. Therefore, when dealing with the JavaScript syntax, you need to remember that writing the code is not the only important task, you must also watch the syntax whether it's written correctly.

Let me show you an example. In the previous chapter, you have successfully printed **Hello World** on your browser using the `document.write();` syntax.

What would happen if you wrote `Document.write("Hello World");`? Yes! It won't run successfully. You will get an error message. This kind of errors is known as **Uncaught SyntaxError**.

A JavaScript statement is typically written on one line. You may finish your statement with a semicolon or not. It is not mandatory to end a statement with a semicolon. However, it is a good practice to add a semicolon after each statement.

Let's consider the following example:

```javascript
document.write("Hello");
document.write("World");
document.write("!");
```
JavaScript keywords (such as for, while, if, switch, case, and so on) are always in lowercase. The build-in objects (such as Date, Math, Number, and so on) start with uppercase.

**Variables**

We already know that the computer world has nothing but data.

There are different types of data (we call them *data types*), as follows:

- Your name is a kind of data
- Your age is data
- Your grade is also data

Yet, they all are different. What is the difference between them? Your name only contains a group of *characters* or, as some people also call it, *string*. Your age is an *integer* type data. Your grade is a *float* type data. The wonderful thing in JavaScript is that you do not have to specify the data type before writing a variable’s name.

JavaScript allows working with three data types. Strings (for example, “This is an example of string”), numbers (for example, 2015, 3.1415, and so on), and Boolean (for example, true or false).

Did we discuss variables? Well, you already know the data types. You will need something to store your data. This something is called variable. In JavaScript, we use `var` before the variable names. Remember that `var` starts with small letter.
Let's consider the following example:

```javascript
var x;
var y;
var sum;
var name;
```

Let's say that we have 14 apples and 6 oranges. To store them in variables we will use the following:

```javascript
var apples = 14;
var oranges = 6;
```

The following example is not the same. Can you tell why?

```javascript
var Apples = 14;
var apples = 14;
var APPLES = 14;
var appleS = 14;
```

Yes, JavaScript is case sensitive. All the variables are different here, though the values of the variables are the same.

Now, let's do some coding. Previously, on console, you printed your name as homework. I hope you did it without any trouble. How about we now print your name differently using a variable? Assume that your name is Sherlock Holmes. What kind of data is it?

You are right, it is string type. Usually for string type data, we put the string between two quotes.

Let's consider the following example:

```javascript
var name = "Sherlock Holmes";
var occupation = "Detective"
```
To print them using console, you need to type each statement and press *Enter*. Take a look at the following image:

```javascript
var name = "Sherlock Holmes";
var occupation = "Detective";

document.write(name);
```

Do not copy and paste the codes on the console. You might get a syntax error.

You will see an extra line appearing after you hit *Enter*, stating *undefined*. Don't worry about this for now. It just returned a console log.

You stored the *Sherlock Holmes* string on the *name* variable and you stored *Detective* on *occupation*. Every time you access *name* or *occupation*, you can access the stated strings.

Consider that you want to print *Sherlock Holmes* on your screen. Just type the following:

```javascript
document.write(name);
```

After typing, hit *Enter*. You will see *Sherlock Holmes* is printed on the screen, as follows:
Type `document.write(occupation);` and hit `Enter`, as shown in the following screenshot:

![Console output example](image)

You may be wondering why there is no space between *Sherlock Holmes* and *Detective*. As, on the console, the history is not automatically removed from the web page on the left-hand side and after you hit `Enter` for your second output (occupation), the string places itself right after the previous string. This will always happen, unless you clear your console using the `Ctrl + L` keyboard shortcut and reload the web page pressing the key `F5`.

Your stored variables will also be erased from the memory when you reload the web page. Don't worry, you will be taught how to use your variables storing on a file in the next chapter.

If you want to join two (or multiple) variables, you add a plus sign (`+`) between the two variables, as follows:

```javascript
document.write(name+occupation);
document.write(occupation+name);
```

Can you tell me what will be output of these commands?

Yes, you are right. The output will be as follows:

*Sherlock Holmes* *Detective*

*Detective* *Sherlock Holmes*

Your output might be in one line on the web page. If you want to split the lines, add a `<br>` HTML tag. The simplest way to add this is to type `document.write("<br>");` and hit `Enter`. Your next output will be in a new line.
If you want to add any string (for example, a space) between the two strings other than any variables, just type the following:

```javascript
document.write(name+" "+occupation);
```

The output will be as follows:

**Sherlock Holmes Detective**

What will happen when you type the following code and hit *Enter*?

```javascript
document.write("My name is "+name+" and I am a "+occupation);
```

Yes! You are absolutely right. The output will be as shown in the following:

**My name is Sherlock Holmes and I am a Detective**

Now, add another variable on the console. Consider that *Sherlock Holmes* is 24 years old. Do you remember what kind of data age is?

Yes, it is an integer type of number. Therefore, type the following code and hit *Enter*:

```javascript
var age = 24;
```

You have the following three variables now:

- Name
- Occupation
- Age

Let's print the following output on the web page:

**My name is Sherlock Holmes, I'm 24 years old and I am a Detective**

What will our code be on the console?
The code is as follows:

```javascript
document.write("My name is "+name+", I\'m "+age+" years old and I am a "+occupation+);
```

The output can be seen as follows:

```
My name is Sherlock Holmes. I'm 24 years old. And am a Detective
```

Printing quotations/inverted commas

If you want to print Shakespeare said, "To be, or not to be: that is the question!" using the `document.write()` syntax, you will probably type the following code:

```javascript
document.write("Shakespeare said, "To be, or not to be: that is the question!"";
```

However, this will give you an error known as **SyntaxError**. To get rid of this error, you need to use a backward slash (\) before the two inverted commas. The correct code will be as follows:

```javascript
document.write("Shakespeare said, \"To be, or not to be: that is the question!\"";
```

The output will be as shown in the following:

**Shakespeare said, "To be, or not to be: that is the question!"**

The same rule applies for single inverted comma (').

Here is a quick exercise for you:

1. Suppose Tom has a cat (Lucy). The cat, Lucy, is 2.4 years old. Store the name, cat's name, and its age on three different variables and print the following output using console:

   **Tom's cat Lucy is 2.4 years old.**

2. Assume that you bought 4 pounds of apples. Each pound costs you $1.2. Store the price and quantity of apples on two different variables and print the following output using console:

   **I bought 4 pounds of apples. I had to pay $1.2 for each pound.**
Comments

Suppose you have done a lot of coding and some logical operations, and used a number of variables on JavaScript, and you want me to help you with the code if any errors occur. When you send me the code, I will not know what you have typed unless I have a clear knowledge of JavaScript or you have commented on the important lines.

A comment is basically a line of text or code that your browser ignores while running. You can compare comments to sticky notes or reminder.

Let’s consider the following example:

```javascript
var name = "Sherlock Holmes"; // This is a string
var occupation = "Detective"; // This variable stores information
var age = 14; // This is an integer type of data.
```

How do you make multiline comments? You mention the comment in the following manner:

```javascript
/*
This is a multiline comment.
The browser will ignore this.
You can type any important information on your comment.
*/
```

Your multiline comment should be between `/*` and `*/`, as shown in the following screenshot:
Arithmetic operators

In JavaScript, like other programming languages, we can do some arithmetic operations. In your school, you might have already learned how to add two numbers, subtract one number from another number, multiply two numbers, and divide a number with another. You can do all these things in JavaScript with the help of a few lines of code.

In JavaScript, we use the following arithmetic symbols for the operations:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>To add</td>
</tr>
<tr>
<td>-</td>
<td>To subtract</td>
</tr>
<tr>
<td>*</td>
<td>To multiply</td>
</tr>
<tr>
<td>/</td>
<td>To divide</td>
</tr>
<tr>
<td>%</td>
<td>To find the reminder (called modulus operator)</td>
</tr>
</tbody>
</table>

### Addition

Suppose you have two variables, \( x \) and \( y \), with the values 3 and 4, respectively. What should we do on the console to store the values on the variables?

Yes, we do the following:

```javascript
var x = 3; // 3 is stored on variable x
var y = 4; // 4 is stored on variable y
```

Then, press Enter.

Take another variable that will hold the summation of \( x \) and \( y \), as follows:

```javascript
var z = x+y; // This syntax stores the sum of x and y on z
```

Can you tell me what will happen when we print \( z \)?

```javascript
document.write(z);
```
Yes, you are correct, this will print 7, as shown in the following screenshot:

Subtraction

To subtract a number from another, you need to put a minus sign (-) between them.

Let's consider the following example:

```javascript
var x = 9; // 9 is assigned to the variable x.
var y = 3; // 3 is assigned to the variable y.
var z = x - y; // This syntax subtracts y from x and stores on z.
document.write(z); // Prints the value of z.
```

The output of this code is 6, as shown in the following screenshot:
Multiplication
To multiply two numbers or variables that have integer or float type of data stored on them, you just put an asterisk (*) between the variables or numbers.

Let's take a look at the following example:

```javascript
var x = 6; // 6 is assigned to the variable x.
var y = 2; // 2 is assigned to the variable y.
var z = x * y; // For two numbers you can type z = 6 * 2;
document.write(z); // Prints the value of z
```

The output of this code is 12, as shown in the following screenshot:

![Screenshot of JavaScript output]

Division
To divide a number with another, you need to put a forward slash (/) between the numbers.

Let's take a look at the following example:

```javascript
var x = 14; // assigns 14 on variable x.
var y = 2; // assigns 2 on variable y.
var z = x / y; // divides x with y and stores the value on z.
document.write(z); // prints the value of z.
```
The output of this code is 7, as shown in the following screenshot:

```
7
```

```
\>
var x = 14; // assigns 34 on variable x.
var y = 2; // assigns 2 on variable y.
var z = x / y; // divides x with y and stores the value on z.
document.write(z); // prints the value of z
\>
```

**Modulus**

If you want to find the modulus of a number with another, you need to put a percentage sign (%) between the numbers.

Let's consider the following example:

```
var x = 34; // assigns 34 on the variable x.
var y = 3; // assigns 3 on the variable y.
var z = x % y; // divides x with y and returns the reminder and stores on the variable z
document.write(z);
```

The output of this code is 1, as shown in the following screenshot:

```
1
```

```
\>
var x = 34; // assigns 34 on the variable x.
var y = 3; // assigns 3 on the variable y.
var z = x % y; // divides x with y and returns the reminder and stores on the variable z
document.write(z);
\>
```
What does modulus (%) operator do?

Well, from your math class, you have already learned how to divide one number with another. Say, you divide 10 by 2. The result will be 5, which is an integer type of number. However, what will happen if you divide 10 by 3? The answer will not be an integer. The value is 3.333333333333. You can also say that the answer is 3 and the remainder is 1. Consider the following:

\[ 10 = 9 + 1; \]

That is, \( (9+1)/3 \)

\[ = 9/3 + 1/3 \]

\[ = 3 + 1/3; \]

Therefore, the remainder is 1. What modulus does is that it finds out the remainder and returns it. Therefore, \( 10 \mod 3 = 1 \).

Now, let's summarize all the arithmetic operators that we learned so far in one single code.

Can you tell me the output of the following lines?

```javascript
var x = 5;
var y = 4;
var sum = x + y;
var sub = x - y;
var mul = x * y;
var div = x / y;
var mod = x % y;

document.write("The summation of x and y is " + sum + "<br>"elson.write("The subtraction of x and y is " + sub + "<br>");
document.write("The multiplication of x and y is " + mul + "<br>");
document.write("The division of x and y is " + div + "<br>");
document.write("The modulus of x and y is " + mod + "<br>");
```

You will get the following output:

The summation of x and y is 9
The subtraction of x and y is 1
The multiplication of x and y is 20
The division of x and y is 1.25
The modulus of x and y is 1
Solving Problems Using JavaScript

This output can be seen in the following screenshot:

I guess you nailed it. Now, let's explain them in the following:

- We assigned 5 and 4 to x and y, respectively
- We assigned the summation of x and y to the sum variable, the subtraction of x and y to the sub variable, the multiplication of x and y to the mul variable, the division of x and y to the div variable, and the modulus of x and y to the mod variable
- Then, we printed them using the `document.write();` syntax
- We used a `<br>` HTML tag to separate the output of each line

Consider the following example:

John has 56 pens. He wants to arrange them in seven rows. Each line will have an equal number of pens. Write a code that will print the number of pens in each row.

(Hint: take two variables for the number of pens and number of rows, divide the number of pens with the number of rows and store the value in a new variable.)

The sample output is as follows:

John will have to place XX pens on each line. // XX is the number of pens
More operators and operations

JavaScript has more operators other than those stated earlier. Let's go little bit deeper.

Increment or decrement operators

If you have an integer and you want to increment it by 1 or any number, you can type the following:

```javascript
var x = 4; // assigns 4 on the variable x.
x = x + 1;
/* since x = 4, and you are adding 1 with x, so the final value is
4 + 1 = 5, and 5 is stored on the same variable x. */
```

You can also increment your variable by 1, typing the following:

```javascript
var x = 4; // assigns 4 on the variable x.
x++; // This is similar to x = x + 1.
```

What will you do if you want to increment your variable by more than 1? Well, you can follow this:

```javascript
var x = 4; // assigns 4 on the variable x.
x = x + 3; // Say, you want to increment x by 3.
/* since x = 4, and you are adding 3 with x, so the final value is
4 + 3 = 7, and 7 is stored on the same variable x. */
```

You can increment your variable by typing the following as well:

```javascript
var x = 4; // assigns 4 on the variable x.
x += 3; // This is similar to x = x + 3.
```

Remember that you should not place a space between an operator (for example +, -, *, /, and so on) and equal sign (=).
The output will look similar to the following screenshot on the console:

```
> var x = 4; // assigns 9 on the variable x.
x = x - 1;
  /* since x = 9, and you are subtracting 1 from x, so the final value is 9 - 1 = 8, and 8 is stored on the same variable x. */
> var x = 4; // assigns 9 on the variable x.
x--; // This is similar to x = x - 1.
  /* since x = 9, and you are subtracting 1 from x, so the final value is 9 - 1 = 8, and 8 is stored on the same variable x. */
```

What about the decrement operator? Yes, you are absolutely right. Decrement operations are same as the increment operations. The only thing that changes is the sign. Your addition (+) operator will be replaced by the subtraction operator (-). Let's take a look at an example:

```
var x = 9; // assigns 9 on the variable x.
x = x - 1;
  /* since x = 9, and you are subtracting 1 from x, so the final value is 9 - 1 = 8, and 8 is stored on the same variable x. */
```

You can also decrement your variable by 1 typing the following:

```
var x = 9; // assigns 9 on the variable x.
x--; // This is similar to x = x - 1.
```

What will you do if you want to decrement your variable by more than 1? Well, you can follow this:

```
var x = 9; // assigns 9 on the variable x.
x = x - 4; // Say, you want to decrement x by 4.
  /* since x = 9, and you are subtracting 4 from x, so the final value is 9 - 4 = 5, and 5 is stored on the same variable x. */
```
You can also decrement your variable by typing the following:

```
var x = 9; // assigns 9 on the variable x.
x -= 4; // This is similar to x = x - 4.
```

The output of these codes can be seen in the following screenshot:

```
// var x = 9;
> x
```

These type of operations are very important for logical operations in JavaScript. You will learn about their uses in Chapter 4, Diving a Bit Deeper.

**Assignment operators**

An assignment operator assigns a value to an operator. I believe that you already know about assignment operators, don't you? Well, you use an equal sign (=) between a variable and its value. By doing this, you assigned the value to the variable.

Let's take a look at the following example:

```
var name = "Sherlock Holmes"
```
Solving Problems Using JavaScript

The Sherlock Holmes string is assigned to the name variable. You have already learned about increment and decrement operators. Can you tell me what will the output of the following codes be?

```javascript
var x = 3;
x *= 2;
document.write(x);
```

The output will be 6.

Do you remember why this has happened?

The `x *= 2;` equation is similar to `x = x * 2;` as `x` is equal to 3, and later it is multiplied by 2. The final number (3 * 2 = 6) is assigned to the same `x` variable. That's why we got the following output:

<table>
<thead>
<tr>
<th>6</th>
<th>Elements</th>
<th>Network</th>
<th>Sources</th>
<th>Timeline</th>
<th>Profiles</th>
<th>Resources</th>
<th>Console</th>
</tr>
</thead>
</table>
| > var x = 3;
x *= 2;
document.write(x);
| < undefined

Let's perform the following exercise:

What is the output of the following code?

```javascript
var w = 32;
var x = 12;
var y = 9;
var z = 5;
w++;
w--;
x*2;
y = x;
y--;
z%2;
document.write(" w = "+w+ ", x = "+x+ ", y = "+ y+", z = "+z +");
```

We will get the following output:

`w = 32, x = 12, y = 11, z = 5`
If you want to do something logical and compare two numbers or variables in JavaScript, you need to use a few logical operators. The following are a few examples of the comparison operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>Equal to</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal to</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Equal to or greater than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>
Solving Problems Using JavaScript

The following are a few examples that use these operators:

```javascript
> x = 3;
< 3
> x == 5;
< false
> x != 3;
< false
> x >> 4;
< true
> x > 3;
< false
> x => 3;
< function x => 3
> x <= 3;
< true
> x < 9 ;
< true
> |
```

You will learn more about the use of these operators in the following chapters.

Let's discuss a few bitwise logical operators and bitwise operators:

<table>
<thead>
<tr>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>This means the AND operator. To check whether two or more statements are true, we use this.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>~</td>
<td>This means the NOT operator.</td>
</tr>
<tr>
<td>^</td>
<td>This means the XOR operator.</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>This means the Right Shift operator.</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>This means the Left Shift operator.</td>
</tr>
</tbody>
</table>

They might be hard for you to learn right now. Don't worry, you don't have to use them now. We will use them in Chapter 4, Diving a Bit Deeper.
Summary

In this chapter, you learned about the JavaScript syntax. We discussed the JavaScript variables and how to assign a value to a variable. You learned how to comment on the code. You now know why commenting is important. You finally learned an important topic: operators and operations. JavaScript, without using operators and logical functions, will not be so rich nowadays. Therefore, learning about the logical operations is the key to gain good knowledge of JavaScript.

I would like to suggest you to practice all the code in this chapter at home. You just type them on the console, avoid copying and pasting the codes. This will hamper with your learning. As a programmer must have a good typing speed, copying and pasting the codes will not improve this skill. You may face problems in typing codes; however, you will learn.

You can solve any arithmetic problem using JavaScript. You can also check whether your logic is true or false on console. If you can do this, we can move on to the next chapter, Chapter 3, Introducing HTML and CSS, where you will learn about HTML, CSS, and so on.
Where to buy this book

You can buy JavaScript Projects for Kids from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.