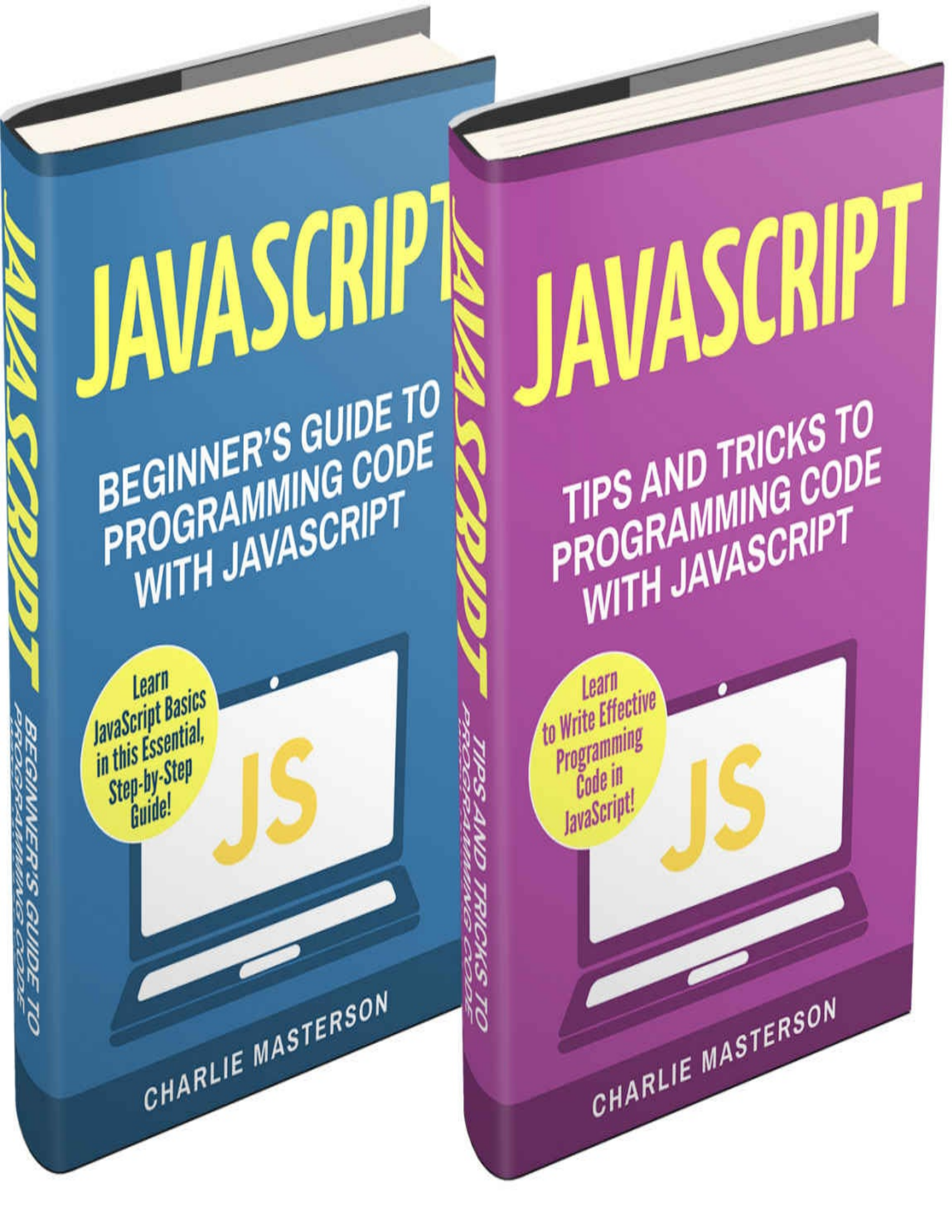

JAVASCRIPT
BEGINNER'S GUIDE TO PROGRAMMING CODE WITH JAVASCRIPT
Learn JavaScript Basics in this Essential, Step-by-Step Guide!
JS
CHARLIE MASTERSON
JAVASCRIPT
TIPS AND TRICKS TO PROGRAMMING CODE WITH JAVASCRIPT
Learn to Write Effective Programming Code in JavaScript!
JS
CHARLIE MASTERSON

JavaScript: 2 Books in 1

Beginner's Guide to Programming Code with JavaScript

Tips and Tricks to Programming Code with JavaScript

Charlie Masterson

About this Bundle:

Congratulations on downloading *JavaScript: 2 Books in 1 - Beginner's Guide and Tips and Tricks to Programming Code with JavaScript* and thanks for doing so.

What you are about to read is a collection of two separate books on how to learn JavaScript computer programming.

Each book will discuss different levels towards learning JavaScript, arranged in proper order that takes you from beginner to intermediate level.

Book 1:

JavaScript: Beginner's Guide to Programming Code with JavaScript

Here you will learn the basic essentials of learning JavaScript – the necessary topics you require in gaining beginner level knowledge.

Book 2:

JavaScript: Tips and Tricks to Programming Code with JavaScript

Here you will progress and learn intermediate level useful tips and tricks in learning about the JavaScript programming language – to help you progress in your path towards JavaScript mastery.

Thanks again for owning this book!

Let us begin with the first book in the JavaScript bundle:

JavaScript:

Beginner's Guide to Programming Code with JavaScript

Charlie Masterson

Table of Contents

Introduction

Congratulations on downloading *JavaScript: Beginner's Guide to Programming Code with JavaScript* and thanks for doing so.

The following chapters will discuss JavaScript and everything that you need to know to understand this popular scripting language.

JavaScript is a complex language that is going to take a lot of patience to learn. However, when you do learn it, you are going to be able to do a lot of different things for the web pages you develop as your browser executes the JavaScript code you have inputted on your web pages. With JavaScript, your web pages can become more interactive and more user-friendly with the functionality it provides.

This book is targeted to readers who know some HTML and CSS – since JavaScript is a language that works with and handles web pages. If you aren't familiar with both technologies, there are definitely several titles out there that you can check out. But even if you are not a programmer per se, this is fine. You can learn JavaScript even without a computer programming background.

There are plenty of JavaScript books on this subject on the market, so thanks again for choosing this one!

Every effort was made to ensure it is full of as much useful information as possible, please enjoy!

Chapter 1: JavaScript…What is it and How will it Benefit me?

JavaScript is a client-side, scripting language. What this means is that the language or code is sent to a user's computer and it executes the code there. This is in contrast with server-side technologies such as ASP.NET or PHP where the code is being run on the web server first before the results are delivered to the user's computer.

With JavaScript you are going to be interacting with different web pages so that you are able to create a web application. Netscape Communications first implemented JavaScript in 1995.

In building web pages, you usually come across JavaScript as part of the three main languages being used, with the other two being HTML and CSS.

So while HTML does the content and structure of web pages and CSS works on the presentation of the page like specifying its background color, it is JavaScript that functions as the programming language for the interactivity and behavior on the page.

For example, what happens when you type the wrong value in a form field on a Contact Us form page? The ability to make a web page interactive and perform added functionality that HTML and CSS cannot do makes you discover the usefulness of JavaScript for a website.

Let me point out though that JavaScript wasn't created to be used as a general, do-it-all language; it has it limits. For instance, it cannot have access to your computer's file system nor communicate with a database. But while this is the case, what it does in terms of making web pages interactive, it does really well.

JavaScript is easily confused with Java because the two of them have very similar names; however, they are not the same thing. The good thing though is that it can interact together on the client side like your web browser for instance, because they can be embedded directly into the web source.

Whenever you are looking at where JavaScript is supported, you are going to be surprised to find that it is supported almost everywhere. In browsers such as Firefox, Google Chrome, and Safari, if you are unsure if your browser is a vendor that has been licensed by JavaScript, you can go to your settings and see if you can enable or disable JavaScript. If you can, then your browser was licensed by JavaScript.

Benefits

Knowing JavaScript is going to benefit you in being able to either advance you in your career or find a different career. If you want to go into I.T. or even web design, JavaScript is a must.

Should you already have a job in one of these areas, you are going to be able to advance in your career because you have furthered your knowledge in something that is going to be beneficial to you.

JavaScript is not only good if you are in I.T. or web design. JavaScript can be used in any job, if you are unsure if it can be used in your job, ask your boss. Showing an initiative to learn beyond what your job calls for is going to impress your boss and earn you favor.

Chapter 2:
Writing your first JavaScript code

As mentioned in the previous chapter, the JavaScript that helps run interactive websites does not work alone and functions alongside HTML and CSS, forming the three core languages that make up a web page.

So let us begin writing JavaScript. Since JavaScript will run in a web page we need to first have a web page where the JavaScript language would be written.

Below is a simple web page example to use as our reference:

```
<!DOCTYPE html>
<html>
<head>
<title>Title here</title>
</head>

<body>
<p>This is a web page</p>
</body>
</html>
```

Code is composed of basic HTML and no CSS, very basic stuff. So now if you copy the code and paste it on a Notepad application, then save it as an HTML file, for example “first.html” and then double-click that file, (if the browser does not open the file when you double-click it, open the browser manually and drag the file to its window) then the web browser installed in your computer should be able to render the code successfully and produce the result of

This is a web page.

And just as we use opening and closing p tags to surround a paragraph, for JavaScript we enclose it with script tags. <script> </script>

For the first JavaScript code to add to this HTML file example, we are going to use a JavaScript popup box called alert that produces a popup message box upon running the web page and output the text ‘Hello World’.

```
<script>
alert ("Hello World!");
</script>
```

Note that in order for the popup box to appear, make sure your browser’s Privacy Settings would allow this.

Now putting everything together, we add the JavaScript code to the rest of the HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>Title here</title>
</head>
<body>
<p>This is a web page</p>

<script>
alert ("Hello World!");
</script>
```

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

Save the file and double-click on it to open it up in the web browser. If this does not work, open the browser and just drag the file to the browser window; it will open the webpage you are creating. You should then see the alert message box pop up with the text 'Hello World!'.

Result: Hello World!

Congratulations – you have written your first JavaScript code. In the next chapters, we will discuss more about the language and how you can use it to produce better web pages.

Chapter 3:
Data and JavaScript variables

Data is normally stored so that it can be used at a later date. This is a very important aspect of writing code. JavaScript is a wonderful programming language that is going to store the data so that you can use it later and do not have to worry about it getting lost.

When you are writing code, you are going to be able to have it interact with the users by asking for information that may be able to be used at a later time.

Code

You would like for the user to enter their last name so that you can personalize a greeting for them.

```
var surname = prompt("Greetings user, may I ask what your surname is?");
```

After the code has been run, a box is going to appear asking the user what their last name is. All the user is going to need to do is type in their last name before they hit enter or click the okay button for the information to be moved and stored in the proper place.

Congratulations, you have now created a variable that will be the surname of the user.

Variables

Variables are going to be easier to understand if you think of them as shelves. Each shelf has a name that is created once the variable has been created. So, for the previous code, your shelf's name is going to be "surname" and the object that is setting on that shelf is the user's surname.

Anything you want to make as a variable can be made. But, it is recommended that you name your variables first so that you know which variable is which instead of naming them things such as numbers or letters.

At the point in time that you tell the browser that you are using to locate a variable, it is not only going to be located, but the data on that "shelf" is going to be given back to you. The object on the shelf is also known as a value.

The data that is stored in the value is called a string. So the letters that make up someone's last name are going to be the surname variables value and string. Strings do not just have to be letters; they can be numbers as well. Basically, string is an array of any possible character to be used.

Essentially, every variable is going to have a value as well as a name and this is how data is going to be stored.

Variables are created through two parts an initialization and a declaration.

After you create it, you will then assign it to the proper value.

Declaration

The declaration process will declare that the variable exists. Back to thinking of the variable like a shelf, the declaration is like when you are first picking out the shelf in the store.

To set a variable aside, the word var is used before the name of the variable is written in. This tells the code what the variable is named and how it should be run.

Code:

```
var birthday;
var first_name;
```

Note: Every line in JavaScript is going to end in a semicolon so that the code is run as JavaScript and not as regular text.

Initialization

To give a variable the value that it needs, you will use the initialization process. These values can be changed at a later date, but it is only going to be initialized one time.

The initialization of a variable will use the equals sign so that the statement reads that the value from the variable that is listed on the left side of the equals sign will be with the data that is on the left.

Code:

```
var name = “Sue”;
```

The string is going to be the name Sue and that string is going to be surrounded by a set of quotes either double or single.

Code:

```
var age = 22;
```

Since the value for the variable is a number, it must not be in quotes to be recognized as a number. This type of variable is called an integer.

Assigning

Setting the value to the variable is known as assigning. The value can be assigned to the variable an unlimited number of times. It is going to be much like the initialization process that was mentioned earlier.

There is going to be an equals sign used but you are not going to need to use the var keyword since the code already knows that the value is a variable.

Code:

```
Name = “Sue”;
Age = 22;
```

But the variable has to be declared first before it can be assigned!

Chapter 4:
Math with JavaScript

The variables that you declare in JavaScript can use the strings that are stored along with any numbers that are stored. But, when you are doing math, JavaScript is going to focus mainly on numbers.

In your script, you are going to create two different variables with two different numbers.

Code:

```
var apples = 5, oranges = 10;
```

Now you have created your two variables. The reason that var was only used once is because you did an initialization and gave the variables a value. When you write the keyword once and separate them with a comma, you save yourself extra work.

With the help of JavaScript, you are going to be able to figure out how many pieces of fruit you have.

Code:

```
var piecesOfFruit = apples + oranges;
```

The code is going to tell the browser that is needs to figure out the sum from the number on the right before it reassigns that result to how many piecesOfFruit there are. But, the names from the variables are not going to be added together, only the values are going to be added. Your browser is going to know what you are doing and will only give you the sum of the values.

Your sum is not going to be pieces of fruit but instead it will be piecesOfFruit since your variables cannot contain any spaces. It is easy to notice that the O and F in Of and Fruit are capitalized. This is known as camel casing and it helps to make the variable easier to read.

Keep in mind that there are various ways for code to be read. That is why it is so important for you to be sure that your code is easy to read and understand.

Splitting

There is more than just addition when you are doing math. For code, there is division.

Code:

```
var piecesForEachPerson = piecesOfFruit / 3 ;
```

JavaScript is able to do this math without you having to get out a calculator to make sure that the math is done properly.

Division is done with a forward slash to ensure that you are dividing the left side from the right side.

Operators

To get the sum of a set of numbers, you are going to use symbols for every operation.

- Multiplication (*)
- Addition (+)

- Division (/)
- Subtraction (-)

Each symbol is called an operator since they are going to operate some sort of mathematical data. Just like any math problem, there are going to be a particular order of how things are going to be done, it is called the operator precedence.

Seeing a set of parentheses means that they need to be done first before you do the multiplication, division, addition, or subtraction process. An easy way to remember this order of operations is PEMDAS (Parentheses, Exponents, Multiplication and Division, and Addition and Subtraction).

Code:

```
(10+2) / 2 + 4 * 2
12 / 2 + 4 * 2
6 + 4 * 2
6 + 8
```

Chapter 5:
JavaScript Logic

When you are programming, it is important to have the ability to compare different values so that decisions can be made when it comes to the code and how it is going to be carried out. The comparison will either make your outcome false or true. Then, there is a special data type that is known as a Boolean data type.

Using JavaScript to do math will bring out a set of operators that are going to work with Booleans. These comparisons are going to be made when you are working on a console. Whichever result you get from every line is going to be shown as true or false.

Equality

To determine if two different values are equal you will use three equals signs. (===)

Code:

```
15.234 === 15.234
```

True

To check if two variables are not equal, you will need to use two equals signs and also use exclamation sign before them. (!==)

```
15.234 !== 18.4545
```

True

Strings that contains numbers, are equal to that number in value, but not in type, so the === comparison is going to return false

“10” === 10

False

However, if you compare these two using two equals marks (==) it is going to return true, because two equals marks only compares the value of variables, not the type! So this will return true:

"10" == 10

True

Greater than/ less than

Two numbers can be compared as one being larger or smaller than the other. When a number is greater than the other you are going to use a greater than symbol of >.

Code:

10 > 5

True

The less than symbol will be the opposite of the greater than symbol (<).

Code:

```
20.4 < 20.2
```

False

Combined comparison

Numbers can be compared when they are of equality and the size is going to be able to be compared with a greater than or equal to (>=) or the less than or equal to (<=) operators.

Code:

```
10 >= 10
```

True

```
10 <= 5
```

False

Chapter 6:
JavaScript Conditions

Using JavaScript logic will make the decision for the code based on what sort of comparison has been made. Before the comparison can be made, there has to be a condition set. There are several conditions that are used in JavaScript, but the most common one is the If statement.

If statement

When the condition is found to be true, the code set is going to be run the way that it is supposed to. Thanks to the if statement, you are going to be able to add in extra code that has to be run should the condition turn out to not be true. When the condition is returned as false, then there will have to be extra conditions and blocks of code that are going to be run optionally. This option is going to be the if-else statement.

Code:

```
if(10 > 5) {
        // Run the code here

}
```

Any code that has been placed between the curly brackets is going to be considered a block and will be connected to the If statement. However, it is only going to be run if the condition that falls between the parentheses turns out to be true.

Note: the double slashes are going to indicate a comment that has been placed in the code and it is not going to be run as code. The comments are placed so that the code can be easily understood by others.

If-else

The if-else statement is a different form of the If statement that will use a piece of code and run it instead of the original code if the condition comes back as false. The code that falls in the If code block is going to be ignored so that the else block can be run.

Code:

```
if ( 43 < 2) {
        // Run the code here
} else {
        //A different code is run
}
```

Chapter 7:
Looping with JavaScript

Just like a circle, a loop is going to continue to repeat the same code over multiple times. Loops are usually used when an code needs to be carried out for an object in an array.

Two loops that you are going to use most often are going to be *while loops* and *for loops*. The conditions are going to be combined together and then the block of code is going to be run until the condition is forced to no longer be true.

While loops

While loops will repeat the code while the condition turns out to be true. While loops are very similar to if statements and the condition that is placed inside the parentheses is true.

Code:

```
var i = l;
while ( i < 10 ) {
        alert (i);
i = i + 1 ;
}
// i is now going to be 10
```

Once the loop has been completed, the code will keep running even after the closing bracket. This code will produce popup alert boxes displaying numbers from 1 to 9, and when it completes the loop, value for i will be equal to 10.

For loops

For loops are also similar to if statements, the main difference being that there will be three semicolons used to separate bits of information that is placed between a set of parentheses. A for loop is going to initialize the condition to get a final expression.

While initializing, you will be creating the variable so that you can track how far into the loop you are going to go just like you did with the while statement. Your condition is going to be where the loop's logic will go. The final expression is going to be run at the end of every loop that has been created.

Code:

```
for(var i = 1 ; i < 10 ; i++ ) {
        alert (i) ;
}
```

In the code, an alert box is going to pop up that is going to give the numbers 1 to 10 in their proper order. The i++ will be the same as i = i + 1. (i++) is called "incrementing" the value of i. As you can see, both for and while loops works the same in this case. You probably wonder when to use for and when to use while loop. This is a short explanation.

While loops are generally used when you don't know the number of how many times the loop will circle around. For example, user is playing a game, where he needs to guess the hidden number. You can never know how lucky the user is, so you don't know when he will pick the right number. That is why you set the while loop and just wait for the right number to come up.

Code:

```
var number = 0;
// 7 is the hidden number!
prompt("Try to guess the number!");
while (number !== 7)
{
        number = parseInt(prompt("Wrong number, try again!"));
}
```

```
alert("Congratulations, you guessed it correctly!");
```

When the user enters number 7, it will display the congratulations message, but if user enters any other number, it will continue to display an error message until the user enters the number 7.

On the other hand, in using for loops, we use them when we know exactly how many times we need to execute the loop; it is mostly used when working with arrays, which we will learn in a later chapter.

Chapter 8:
Functions Used in JavaScript

Functions are going to reuse the blocks of code so a specific task can be carried out. In order to execute the code inside a function that you call, the function will be allowed to pass any argument that is used so that a value is returned to what originally called it.

Functions can be saved for the value of your variable. The function is going to use the variable and the parentheses to invoke the function.

Code:

```
doSomething() ;
findAnInterestingThing() ;
```

Creating a function means you are going to use the function keyword. After that has been used, the arguments are going to be listed inside a set of parentheses so that the block has the functions code.

Code:

```
var add = function(a, b) {
        return a + b ;
} ;
```

The a and b are going to be the parameter for the functions while the value is going to be returned by the keyword.

The keyword that is used for the return is going to stop the code that has been placed in the function; therefore nothing is going to run after it.

Code :

```
var result = add( 1, 2); // the result is going to be 3.
```

The call is going to add the argument of one and two, which have been saved for the a and b variables.

Chapter 9:
Objects Located in JavaScript

When an object has been found in JavaScript, it is going to be much like a real life object that contains a set of properties and abilities.

The object will have a collection of properties that are named, and methods that are going to determine how it is going to function.

The object is going to be saved into a variable while everything else can be obtained through a phrase using dot notation.

As an analogy, if you look at a person, they are going to have a name, they have an age, and they are able to move, learn, and talk. Should that human be placed into JavaScript, their name and age are going to be considered properties which are nothing more than pieces of data. The ability to move, talk, and learn are going to be the functions of the human being because they are complex behaviors. An object that has theses abilities is known as a method.

Variables can hold objects while they create the object through the use of a special syntax that is going to be enclosed in a set of brackets.

Code:

```
var jedi = {
        Name: “Luke”,
        Age: 899,
        Talk: function() {
           alert(“another... Sky... walk …”);
        }
};
```

The name and age of the Jedi are going to be the properties or variables that

are sitting inside the object. Objects are able to store any data that a variable can. The talking that the Jedi can do is going to be the property where the function is being held, also known as the method. The data can come back from the object through the use of the phrases using dot notation.

Code:

```
jedi.Name;
Luke

jedi.Age ;
899

jedi.Talk() ;
//an alert box is going to be produced.
```

The properties can also be reassigned to the object.

Code:

```
jedi.Name = “Mace Windu”;
```

Or you can add in new variable objects that were not added in earlier.

```
jedi.lightsaber = “purple” ;
```

The properties do not have to be any specific data; they can be anything as well as objects and arrays. When one object is listed as the property for a different object, you have successfully created a nested objected.

Code:

```
var person = {
        age: 122
};

person.name = {
        First: “Jeanne” ,
        Last: “Calment”
};
```

Now if you want to know what the person's first name is, you will use this.

Code:

```
alert(person.name.First);
```

To create the item that will be empty you will need to add the properties and methods.

Code:

```
var dog = {};
dog.bark = function() {
        alert(“Woof!” );
};
```

Chapter 10:
JavaScript Arrays

An array is a list of any sort of data and it is not limited to including other arrays. Every item that is listed in an array is going to have an index that will be used in retrieving an element from the array.

The index is going to start at zero always and all the elements that fall in line are going to increase by one so that the last element will be one number less that the length of the total array.

Creating an array in JavaScript means that you use an array-literal syntax.

Code:

```
var emptyArray = [] ;
var shoppingList = [ 'Rice', 'Soda', 'Buns'];
```

You are going to be able to get a specific element with the use of a square bracket syntax.

Code:

```
shoppingList[0];
```

Rice

You can also set the value to the particular index by using the square bracket syntax

Code:

```
shoppingList[1] = 'Chips';
// your shoppingList is now going to be ['Soda', 'Chips', 'Rice']
```

The quantity of items present in the array is going to use the length property to tell you how many elements there are.

Code:

```
shoppingList.length;
```

3

The push and pop methods can be used so that elements can be added and removed from the finish of the array.

Code:

```
shoppingList.push ('A new car');
// shoppingList [ 'Soda', 'Buns', 'Rice', 'A new car']

shoppingList.pop();
// shoppingList is now ['Soda', 'Buns', 'Rice']
```

The code below is going to create a push, pop, and then iterate it over the array so that it can be passed to the name of the function that is labeled as helloFrom. Then helloFrom is going to be returned to the string for the greeting of: “Hello from” and then the user’s name. After you have finished with the push and pop method, the list of user’s name is going to be “Burt”, “Luke”, “Ron”, and “Tim”.

Code:

```
var helloFrom = function(personName) {
        return “Hello from” + personName;
}
var people = ['Burt', 'Luke', 'Ron'];

people.push('Tim');
people.push('Dr Evil');

people.pop();

for(var I = 0 ; I < people.length; I ++) {
        var greeting = helloFrom(people[I]);
        alert(greeting) ;
}
```

Chapter 11:
DOM

DOM stands for Document Object Model and it will manipulate how an HTML page is structured and styled. It is going to be represented by the internal pages that the browser is going to see so that the developer is able to alter the JavaScript code.

In order to look at the DOM for a page, you are going to open the developer tool inside of the browser that you are using and locate the "elements" pane. You can do this by clicking right click somewhere on the page in your browser and clicking "Inspect Element" from the menu list that appears. This will give you a good insight as to how the browser thinks. In most browsers, you are going to have the ability to modify and remove elements directly without asking for permission.

Trees and Branches

HTML is similar to the structure of an XML in that the elements are going to contain parent nodes from the structure that it creates along with children nodes. It is much like tree branches. HTML is going to have a single root element that contains branches much like a head and body and the limbs that are going to be attached to it. Because of this, DOM is also known as a DOM tree.

When altering a DOM, you are going to pick an element and change whatever you want about it. This process is normally done with JavaScript. In order to access DOM from JavaScript, you are going to need the document for the object that is being used. It will be provided by the browser so that the code is allowed to interact with the code that is on the page's content.

Getting Elements

When you are getting an element, you are going to have several ways that you can do it depending on the browser that you are using.

ID

document. getElementById will be the method that you use when you are trying to retrieve an item with its Id.

Code:

```
var pageHeader = document.getElementById('pageheader');
```

Using this code, you can select element with ID of pageheader and store it in the pageHeader variable. Later on, you can modify it, get its content, or even hide and display it. Here is one example.

Code:

```
alert(pageHeader.innerHTML);
```

This will popup the contents of element with ID of pageHeader. For example, if pageheader looks like this:

Code:

```
<p id="pageheader">This is the page header</p>
```

It will popup "This is the page header" text.

You are going to notice that getElementById is going to be the method that is used with an object of a document. Some of the other methods are going to have access to the page that has been located on the file for the element.

Tag name

document. getElementsByTagName is going to work the same way that the getElementById does but it is going to need a tag name such as a, li, or ul instead of having an ID. It will then be returned to the NodeList which is basically an list or array of elements from DOM.

If you have for example only two <p> elements like this:

Code:

```
<p>This is one paragraph</p>
<p>This is another paragraph</p>
```

Then you can store these two elements in paragraphs variable using:

Code:

```
var paragraphs = document.getElementsByTagName("p");
alert(paragraphs[0].innerHTML);
alert(paragraphs[1].innerHTML);
```

Alert functions will display the content of the first paragraph, and then the content of the second paragraph.

Class name

Document. getElementsByClassName will return the same list that the tag name method does. The biggest difference is that you are going to have to pass a class name that will match instead of a tag name.

CSS

With modern browsers, there are a few new methods available for you to use that makes selecting elements easier through the use of a CSS selector. The syntax for CSS is: document. querySelector and document. QuerySelectorAll.

Code:

```
var pageHeader = document.querySelector(“#example”);
var buttons = document.querySelectorAll(“.btn”);
```

The querySelector is going to be like when you search for an element by the ID, you are going to have only one element that is returned. But, querySelectorAll is going to return the NodeList. So, if multiple elements match your search criteria, then you are going to need to use the querySelector so that you are only getting the first one returned instead of all of them.

Chapter 12:
Callbacks and Events with JavaScript

The code that is located inside of a browser is going to be event driven and when you are writing applications that are interactive, you are going to have to wait for as well as react to the events that are going to ultimately alter the behavior from the browser. An event is going to occur when the page has been loaded, or the user has interacted with something on the page. There are also other times that are going to be triggered, but these have to be done manually.

When you need to react to an event, you are going to have to listen for it so that you can supply a function. The function is going to be called by the browser whenever the event has occured and it is known as a callback.

Code:

```
var handleClick = function(event) {
        // choose a different action !
};

var button = document.querySelector('#big-button');
button.addEventListener('click',handleClick);
```

addEventListener is a method that is going to be found on all of the DOM elements. When it is called upon, the element is going to be saved in the variable of button. The argument that is listed first in the string is going to be the name of the event that you should be listening for. The event in this code is the tapping of a mouse or someone's finger. The call back is going to be the handleClick.

If you are using Internet Explorer, addEventListener is not going to be supported if you are running a version that is earlier than version 9. In that case, you should use attachEvent instead.

Code:

```
button.attachEvent('onclick', handleClick);
```

If you look, you will notice that the onclick is required over a click. This is going to happen for a majority of the events that you come across. It is objects such as this that have created the libraries such as jQuery to aid you so that even if the way of listening is different between two browsers, it is not going to matter thanks to the event that was passed as a callback.

Data for any particular event that has been passed is a callback. When you look at the handleClick that was declared above, you are going to realize that the argument or the event is going to be where the object's properties are described.

Code:

```
{
offsetX: 74,
offsetY: 10,
pageX: 154,
pageY: 698,
target: h2,
timestamp: 215897564812,
type: "click",
x: 154,
y: 395
}
```

Chapter 13:
AJAX with JavaScript

Before moving to this section, it is very important to notice that a lot of browsers automatically blocks AJAX, if it is invoked directly from a user's machine. In order for AJAX to work properly, you will either need a webhost or localhost installed.

Whenever the web first came around, things were simpler than they are now. The page now has text and styles while possibly having links that are going to take you to a different web page. To put new content on a page, you have to move it from one page to another. The developers of web pages have been more ambitious trying to get web pages to be interactive. But, it is obvious that new content needs a new way to be loaded onto the page without fully reloading the page each time.

In retrieving new content, there is software that will help known as XHR or XML HTTP Request that is applied. Applications for the internet use AJAX applications. The letters of AJAX mean Asynchronous JavaScript and XML.

Most web pages are going to be able to pull the new content without having to reload the page with this technique. Microsoft is actually the one who decided that XML Http Request should be used when they were creating Outlook.

XML HTTP Request

You know what XML HTTP Request is, but how is it displayed?

Code:

```
var req = new XMLHttpRequest () ;
req.onload = function ( event)  { … };
req.open('get' , 'some-file.txt' , true);
```

```
req.send();
```

When creating a XMLHttpRequest, you are going to need a new keyword that is going to call upon XMLHttpRequest similar to how a function is called upon. The call back will happen at the point in time that the information has been made available and then it will skip the input from that event into the fist argument.

The data is going to be extracted by using req . open. The disput will be the HTTP technique. Next is going to be the URL fetch that will be identical to the aspect of a link by href.

The next one will be a Boolean that is going to specify if the request of asynchronous can be found as true. therefore, the XMLHttpRequest will be fired off and then the code that is executed will continue to be executed until there is an answer from the system that cause the callback to be executed.

The parameters for an asynchronous is going to be defaulted to false which is going to cause the execution of the code to be paused until the appropriate data has been requested which is called synchronous.

Synchronous XMLHttpRequest are very rarely used which is an offer to the system that can sometimes take a long time to load. The last line is going to allow the web page to deny the request for any information.

In using XMLHttpRequest, you are going to be able to load JSON, XML, HTML, and a series of text over the HTTP and HTTPS servers while continuing to support rules such as FTP. Ultimately they are useful when it comes to a range of tasks that are going to be involved in the development of JavaScript applications.

AJAX and Libraries

Techniques used with AJAX are being created so that there are now applications that are single pages with a demand that brings up the JavaScript script so that it is allowed to appear asynchronously any other scripting that needs to be loaded from the server. There are entire libraries and frameworks that are built in order to help this happen.

Chapter 14:
JSON

JSON stands for JavaScript Object Notation. This is a set of text formatting rules that are used for storing and moving data around in a machine so that a human can read it. It is going to look similar to the object literal syntax that is used with JavaScript but this is where JSON is going to orginate.

JSON may not be part of JavaScript's official code, it is a big part of JavaScript so while it is a different language, it is still important for you to understand how it works when it works with JavaScript.

Code:

```
{ "name" : "Luke" , age: 894, "lightsaber" : { "color" : "green" } }
```

Just like with normal JavaScript code, the brackets are used to notate specific parts of the code.

JSON is going to transfer the infromation that needs to be moved between the browser to its server or it will be saved in a text file that will be retrieved later since it is nothing more than text. Because of this, you are not going to be allowed to store complex data such as functions. However, arrays and any other objects that contains simple data is going to be used.

JSON will take over XML when the data transfer format from the web and the new APIs are written which are going to serve JSON exclusively. This means that you are going to be able to use AJAX technology in order to grab JSON.

Using JSON

Data has the ability to be convereted from or to JSON by using the stringify method along with the parse method. JSON is when the object that is available is available through most all modern browsers, but you can add it to your browser if it does not already exist there.

Code:

```
var jsonString = JSON.stringify( {
        Make: "McLaren",
        Model: "MP4-12C",
        Miles: 5023
} ) ;
```

JSON. Stringify will convert an object into a JSON string. Like in the code the jsonString is going to be { “make” : “McLaren”, “model” : “MP4-12C”, “miles”: 5023 }

Code:

```
var car = JSON.parse(jsonString);
```

The string is going to be allowed to be converted into a JavaScript object by using JSON.Parse .car where it is now a JavaScript object that can be used where you can set the properties.

Code:

```
car.model = “P1” ;
```

Chapter 15:
Scopes with JavaScript

A scope is going to be the term that will determine what the variable visibility is. The variables scope is going to be the part of the code that is going to access as well as modify the variable. There is a function scope that is going to be used with JavaScript.

Scope going to be dictated by the block of code where the variable has been declared. The block will be anything that is contained inside of two curly brackets or is indented.

Code:

```
var a = 10 ;

if(a > 5) {
        var b = 5;
}

var c = a + b; //this is bad programming practice!
```

Global variables are going to be read and modified wherever you put them in your application, but they are not going to be the best to use because they are going to expose the security issues in the code which is ultimately going to make the code harder to maintain.

It is important to remember that your code needs to read more than it is written. Whenever code is read by a human but is unable to be determined as to where the variable has come from or what potential values there are, you are going to have a problem.

You are going to get your best results when you limit the scope of the variable as much as you possibly can which will make it visible to as few parts of the code as possible.

Function scope

Since JavaScript does things differently, it is going to use a function scope. A function scope is going to mean that the variables will not be visible when they are outside of the function where they were declared. If they have not been declared in a function, then they are going to be visible globally.

Code:

```
var doSomething = function() {
        var a = 10 ;
} ;

doSomething () ;

console.log(a) ; // this shows that a has been undefined.
```

Child scopes

Variables are also going to come in child scopes that will have their own scopes. For the code, if the doSomethingElse is a child of the doSomething function then the variable "a" will be visible when it is placed inside of doSomethingElse.

Code:

```
var doSomething = function() {
        var a = 10;

        var doSomethingElse = function() {
        console.log(a); //a will be 10
        } ;

        doSomethingElse ();
```

```
} ;

doSomething() ;
```

Functional scopes are going to be very powerful tools when it comes to creating elegant codes, but it takes a while to actually get the hang of it.

Chapter 16:
jQuery in JavaScript

There is going to be a set of tools that is going to be used when modifying and controlling the DOM. This is a little bit messy though because there are different DOMs across the browser types. So that you can make the coding experience simpler, there are a large amount of resources that have been set up so the large mistakes are hidden and it makes a more uniform way for when it interacts with the DOM. Most of the time it is also going to provide the AJAX functionality which will then take the stress of how complex coding is away.

jQuery is a well-known DOM library that is going to be used across a variety of pages. But, the process you use to communicate with the DOM will be called an Application Programming Interface or API.

jQuery is going to have a syntax that is very distinctive because it is going to be based on the dollar symbol.

Code:

```
$('.btn').click (
        function() {
                //choose a different action
        }
) ;
```

The code for this is going to attach to the click handler of all the elements that are listed in the btn class. The selector syntax will be the core of jQuery.

jQuery can be placed on the website with the file like a scripting element. You can pull the file from jquery.com or you can add from a CDN.

Code:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
```

You should not always use a library, but you should choose if it is going to have a structure that is established on the product of the developers choosing. For example, if jQuery has an extensive document that needs to be introduced, then the downloading is going to be slow for the page, especially when it comes to mobile browsers that might have weak connections.

Chapter 17:
jQuery DOMs and APIs

When elements are selected and then interacting with the DOM, jQuery is most likely going to be used. Thankfully, it is easy for you to be able to use since the syntax is simple when it comes to choosing an element or even a set of elements. This is almost exactly like the CSS.

jQuery allows the performance of actions that are done on elements to be done at the same time which saves some time in your coding process.

The $('.note') is going to be selecting all of the elements that has "note" class.

Code:

```
$('.note').css('background', 'red').height(100);
```

jQuery has a syntax that is chainable which will work when you are setting any kind of method. jQuery will return the same element as the selector function. The dollar sign is going to be a function that helps to return a jQuery wrapped that is around an element. The .css method is going to change the background color to red and .height sets the height of the element that you have selected. There is also one that can be used for the width.

Getters and Setters

As shown above, the .css and .height are going to be there to set the value for the element, but these methods are also known as getters. When you call on the .height without having any value to be set, then the current height of the element will call the .css with a simple CSS property to retrieve the value of the property.

Code:

```
var currentHeight = $('.note').height(),
        currentColor = $ ('.note').css('color') ;
```

When there is more than one element that has been selected, then the getter is going to retrieve the value for the first one of them.

Context

There are going to be times that the DOM needs to be limited from where the element can be selected. The area for the DOM being used is going to be known as a different context.

You will need to tell the jQuery the area that you want selected form the second argument that will be a DOM element or a string selector or even a jQuery object. The jQuery is only going to search for the context with your selector and nothing else.

Code:

```
var $header = $('header'),
        $headerBoxes = $('.note' , $header);
```

Chapter 18:
jQuery and AJAX

There are some AJAX helper methods when it comes to jQuery that are going to allow you to save time and make your code painless to look at. They are going to be methods of the $ variable: $.get, $.post, and $.ajax.

The $. AJAX strategies will be the primary technique that is going to be the main method you will use in order to construct the AJAX request. The other methods are going to be used as shortcuts for configurations such as getting data or posting it.

Code:

```
$.ajax(  {
        url: '/data.json',
        method: 'GET',
        success: function (data) {
        console.log(data);
        }
} ) ;
```

The configuration of an object is going to be used whenever jQuery needs to retrieve data. The basics are going to be supplied such as the URL. This method is going to default to the get method and the function is going to be called whenever the data has been retrieved as it is going to be named success callback.

$.get

The code for this method is going to get a series of information and being that this is the basis of the function; the jQuery is going to provide a helper.

Code:

```
$.get ('/data.json', function ( data ) {
        Console.log(data);
} ) ;
```

A callback error is also going to be provide to let you know if that is not right and thus making it to where the server cannot be found.

Code:

```
$.get('/data.json', function ( data ) {
        Console . log ( data ) ;
} ) . fail ( function () {
        // Oh no, that is not right
} ) ;
```

$. post

When you are directing input to the system, you will use the $. post method. The argument that is second will cause the data to be sent to anywhere but the function. jQuery is going to figure out a way for the data to be sent.

Code:

```
$.post('/save', { username: 'burt' } , function (data) {
        Console . log (data) ;
} ) . fail (function () {
        // wrong action
} ) ;
```

$.ajax

To have a margin of authority of how information is directed, you will use the $. ajax method

Code:

```
$.ajax( {
        url: '/save' ,
        method: 'POST' ,
        data: { username: 'burt' } ,
        success: function (data) {
        console . log (data) ;
} ,
Error: function () {
        //wrong action
}
} ) ;
```

Chapter 19:
jQuery and Other Tricks

The jQuery is going to aid you in doing other tasks that are common, especially when things are inconsistent across a series of browsers.

DOM Content Loaded

There are going to be times that JavaScript is going to run whenever the DOM has been loaded, for code, when you are trying to maneuver the objects to various sections on the site, or design a new site. This is going to purely be JavaScript. But, it is not going to work with every browser.

Code:

```
var doSomething = function(event) {...} ;
window.addEventListener('DOMContentLoaded', doSomething);
```

However, it is easier to do this when you are using jQuery and it is going to be carried across multiple browsers.

Code:

```
$(doSomething);
```

Load

When it comes to certain situations, you are going to need to wait for the page to load fully before you try and do anything with it. When you are using jQuery, you are going to listen for the event to load into its window.

Code:

```
window.addEventListener ('load', doSomething);
```

With jQuery, it is going to be simpler

Code:

```
$(window).load(doSomething) ;
```

Type Checking

To find out the data that has been stored in a variable, JavaScript is going to be awkward, but jQuery is going to be able to show you how to do it without the awkwardness.

Code:

```
$.isArray([1, 2, 3]);
```

True

```
$.isFunction(function() { });
```

True

```
$.isNumeric(10);
```

True

```
$.isPlainObject({name: 'Burt'});
```

True.

Chapter 20:
JavaScript Control Flow Statements

In earlier chapters we have discussed if, else and loops, but this section is going to explain these in more detail.

Source files are going to have narratives that will be executed in the exact order that they have been written out. A control flow statement will deal with the way that the operators will flow based on the looping, decision making, branching, and how programs are enabled to operate due to specific chunks of code that have conditions attached to them.

If-then

An if then statement is going to be the simplest control statement that you are going to use. This statement will execute your code only if the expression in the brackets is true.

Driver's Ed class will only enable the breaks to slow down the car's acceleration only if the car is already moving.

Code:

```
function applyBrakes() {
        // the "if" clause: car is already in motion
        if(isMoving) {
        // the "then" clause: lower car's acceleration
        acceleration -- ;
        } }
```

If the condition is false then the control statement is going to move on to the next statement that is in the block of code. The curly brackets are going to be strictly optional; however, this is only going to be the case if the statement that is between them is a single statement.

Code:

```
function applyBrakes() {
// same as above code, only the curly brackets are not going to be here.
        if(isMoving) CurrentSpeed -- ;
}
```

Whenever curly brackets have been omitted, you are making your code breakable. If another expression is placed in the code, then a requirement will be added which is going to mean that you have to use curly brackets, but sometimes they are forgotten by the coder. The JavaScript complier is not going to catch this mistake; it is only going to give you improper results.

If- then- else

If then else clauses are going to create a secondary path for the code to follow if the clause ends up failing.

Code:

```
function applyBrakes() {
        if(isMoving) {
        currentSpeed -- ;
        } else {
        alert("The car is not moving!");
        }
}
```

Switches

A switch statement offers different paths for code to be executed. The switch

statements will work with int, short, char, and byte data along with enumerated data types.

Code:

```
var month = 8;
var monthString ;
switch(month) {
case 1: monthString = "January";
break;
case 2: monthString = "February";
break;
case 3: monthString = "March";
break;
case 4: monthString = "April";
break;
case 5: monthString = "May";
break;
case 6: monthString = "June";
break;
case 7: monthString = "July";
break;
case 8: monthString = "August";
break ;
case 9: monthString = "September";
break ;
case 10: monthString = "October";
break ;
case 11: monthString = "November";
break ;
case 12: monthString = "December";
```

```
break ;
default: monthString = "Invalid month";
break ;
}
alert(monthString);
```

As part of the switch blog, the main statement is going to allow for comments to fall in the switch block so that multiple cases or default labels are carried out. All of the switch statements are going to be evaluated before they are carried out by JavaScript. When a statement has identical tags as another statement, it is going to be successful.

Code:

```
var month = 8
if(month === 1) {
        alert("January");
} else if (month === 2) {
        alert("February");
}
```

… so on and so forth.

When an if then else statement has been used, the clarity of the expression is going to have to be examined. If then else statements are going to test any expression to see if the conditions and any values that have to be used so that it can be tested based on the enumerated values, string values, and integers that have been used. Switch statements are going to end any break statements.

The control flow is going to continue to execute with any comment that has been placed after the switch's block. break statements are only going to be used whenever the switch block has fallen through but all other statements are

going to need to have identical labels so that it can be executed in the order that it has been placed. It does not matter where the case labels are or until the break statement has been reached.

Code:

```
var futureMonths = [ ];
var month = 8;
switch (month) {
case 1: futureMonths.push ("January") ;
case 2: futureMonths.push ("February") ;
case 3: futureMonths.push ("March") ;
case 4: futureMonths.push ("April") ;
case 5: futureMonths.push ("May") ;
case 6: futureMonths.push ("June") ;
case 7: futureMonths.push ("July") ;
case 8: futureMonths.push ("August") ;
case 9: futureMonths.push ("September") ;
case 10: futureMonths.push ("October") ;
case 11: futureMonths.push ("November") ;
case 12: futureMonths.push ("December") ;
break ;
break ;
Default: break ;
}
if(futureMonths.length === 0) {
        alert ("Invalid month number") ;
} else {
        for(var i=0; i<futureMonths.length; i++) {
        alert(monthName);
        }
}
```

The output for the code will popup the months from August through December.

When the break ends, you are not going to need to replace any of the code because the code will have vanished from the switch statement. This break is going to be highly recommended since it makes modifying the code easier therefore you will reduce any errors that you may receive. The section of the

code that falls under default is going to deal with the values that have not been placed in the case section. The code is going to be able to have multiple cases.

Code:

```
var month = 2;
var year = 2016;
var numDays = 0;
switch (month) {
case 1: case 3: case 5:
case 7: case 8: case 10:
case 12:
numDays = 28;
break ;
case 4: case 6:
case 9: case 11:
numDays = 31;
break;
case 2:
if ((year % 4 === 0) &&
!(year % 100 === 0)
|| (year % 400 === 0))
numDays = 29;
else
numDays= 28;
break;
default: alert ("Invalid month.") ;
break ;
}
alert("Number of Days = " + numDays) ;
```

The result will be 29 being that there are 29 days found in February.

Switch statement strings

The string objects are going to be placed into switch statements. At some point in time a switch is going to be placed with the expression being compared to other expressions based on their case labels. These strings can be accepted into a switch only if the case is lowercase and the case labels are lowercased

as well.

Code:

```
var month = "January";
var monthNumber = 0;
switch (month.toLowerCase()) {
case "january":
monthNumber = 1
break ;
}
```

The statement is going to continue to be executed until the point in time that all twelve months have been written out in lower case along with the number that is associated with that month.

While and do while statements

While statements are going to work on the statement blocks continuously only if a condition is met to be true.

Syntax
While (expression) {

Statement(s)

}

A while statement is going to evaluate any and all expressions before returning a Boolean statement. When the condition is found to be true, it is going to

execute any statements that are found in the while block. But, when the statement is tested, it will be executed up until the statement has been proven as false.

Code:

```
var count = 5 ;
while (count < 15) {
alert("Count is: "+ count) ;
count++;
}
```

The infinite loops are going to be implemented in a while statement with this syntax:

```
while (true){
// your script belongs in this section
}
```

A “do while” statement can be part of the coding for JavaScript as well.

```
do {
Statement(s)
} while (expression) ;
```

The main dissimilarity between a do while statement and an infinite loop is going to be that the expressions found as condition for while loop to execute will be evaluated first as JavaScript works its way to the top. At some point, every expression is going to be evaluated at least once in the loop, but some

are going to be evaluated twice.

Code:

```
var count = 5
do {
alert("Count is: "+ count) ;
count++;
} while (count < 10) ;
```

For statement

For statements are going to provide a way for the values to be iterated that are found inside of a range. The for loop is going to continue until the condition that has been set for it is satisfied as it should be.

Syntax:

```
for (initialization; termination;
Increment) {
Statement(s)
}
```

When you are dealing with for statements, you have to remember that termination is going to happen whenever the condition is found as false. Increment expressions are going to called on after the iteration of a loop, but it will be alright if this expression goes up or down in value. Finally, the initialization of an expression will initialize the loop and execute it once the

loop has been started.

Code:

```
for (var i=1 ; i<12; i++){
alert ("Count is: " + i);
}
```

The output will be the numbers one through ten.

When a code is stated, the variable is going to be initialized in the expression. variable scopes will then extend from where the variable has been declared to the end of the code block. For statements are going to be used when an expression has to be terminated.

Should the variable be declared in a for statement, it cannot be used outside of the loop being that it was declared during the initialization process. variables such as j, k, and I are used when it comes to loop controls. Once the variable has been declared, then the life span of that variable is going to be cut down so that there are less of a chance for errors to occur. In a for loop there are going to be three expressions that will be used.

Code:

```
for (var i=1 ; i<91; i++){
alert ("Count is: " + i);
}
```

The for statement can contain a different variation that is used during the iteration process. This variation is going to be known for the enhancement of the statement and will make the loop smaller and easier for people to read.

Code:

```
var numbers = [11, 12, 13, 14, 15, 16, 17, 18, 19];
for (var i=0; i<numbers.length; i++) {
alert ("Count is : "+ numbers[i]) ;
}
```

Any variable that holds value such as the code that is shown above will give you an output of one through ten yet again. The use of this variation is going to write out the statements in the general that that a for statement is normally written out.

break statement

There are two break statements that are used in JavaScript, the labeled and the unlabeled statement. An unlabeled statement is going to be used when a switch statement is used. These breaks are also going to terminate most loops.

Code:

```
var arrayOfInts = [72, 57, 5, 579, 14, 1179, 2120, 9, 655, 145];
var searchfor = 14;
var I;
var foundIt = false;
for (I = 0; I < arrayOfInts.length; I++) {
if (arrayOfInts [I] === searchfor) {
foundIt = true;
break;
}
```

```
}
if (foundIt) {
alert("found " + searchfor + " at index " + I);
} else {
alert ("not in the array");
}
```

break statements have to be placed in bold print wherever the loop has been terminated before the value that you want to extract is reached. Most control flow statements can be transferred into loops.

In the code, above, the output will be the number twelve and that it was found at the fourth index.

breaks are going to be unlabeled and therefore will terminate any inside parts of the switch, while, do while, and for statements. When a break is labeled, it will stop any statement that is found on the outside.

Code:

```
var arrayOfInts = [ [ 32, 87, 3, 589 ], [ 12, 1076, 2000, 8 ], [ 622, 127, 77, 955] ];
var searchfor = 12;
var I;
var j = 0;
var foundIt = false;
search:
for(I =0; I < arrayOfInts.length; I++) {
for (j = 0; j < arrayOfInts[I].length;
```

```
j++) {
if (arrayOfInts[I][j] === searchfor) {
foundIt = true;
break search;
}
}
}
if (foundIt) {
alert("Found " + searchfor + " at " + I + ", " +j);
} else {
alert(searchfor + " not in the array");
}
```

The output will be that twelve was found in the first index.

Labeled statements are going to stop at break statements and then the flow is not going to be transferred. When a control flow statement is transferred, the label is going to stop the statement.

Continue statements

Continue statements are going to skip over any iteration that you place inside of a for, do while, or while loop. The forms that are unlabeled will be skipped so that the code on the inside is evaluated before the Boolean expression that is going to control your loop.

Code:

```
var searchMe = "Peter Piper picked a " + "peck of pickled peppers";
```

```
var max = searchMe.length;
var numPs=0;
for (var I = 0; I < max; I++) {
// interested only in p's
if (searchMe[I] !== 'p')
continue;
// process p's
numPs++;
}
alert("found " + numPs + " p-s in the string. ");
```

There are 7 p's that were found in the code above.

Whenever the continue statement is removed, there are really going to be thirty-five p's found. The statement is going to be skipped it is found to have a label for the current iteration on the outer loop.

In this code you are going to see that there are two loops that have been nested together so that the substring can be searched. At the point in time that two loops are nested, it will be required that one is iterated over the substring while the other string is searched.

Code:

```
var searchMe = "Look for a substring in me" ;
var substring = "sub";
var foundIt = false;
var max = searchMe.length - substring.length;
test:
```

```
for (var I = 0; I <= max; I++) {
var n = substring.length;
var j = I;
var k =0;
while (n-- !== 0) {
if (searchMe[j++] !== substring[k++]) {
continue test;
}
}
foundIt = true;
break test;
}
alert(foundIt ? "Found it" : "Didn't find it");
```

Return statement

A return statement can be found at the end of a branching statement. Return statements are used as exit methods for current control flow statements so that they can return to the point that the method was first invoked. Just like the other expression, there are going to be two different expressions that can be used for a statement that has been returned.

When a value is found in one statement, it is going to be returned while the value for the other will not be. To get a value returned, you are going to need to have the value listed after the keyword that you want returned. The syntax will be return ++ count ; .

The data type that has been returned is going to need to match the type that the method declared. Once the method has been declared then the value will be bad and it is not going to return a value.

Conclusion

Thank for making it through to the end of *JavaScript: Beginner's Guide to Programming Code with JavaScript*, let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever it may be.

The next step is to use the coding skills that you have learned and create your own code. It is not going to be easy at first, but it is going to be worth it. There will be times that you are going to mess up, but you should be patient because just like when you learn anything new, you are going to make mistakes.

Do not give up because your code does not work the first time. Go over the lessons in this book again and reread them. Learning how to write code is going to be confusing until you get the hang of it because of all the different conditions and exceptions that you are going to learn.

Once you have gotten the hang of code, you are going to have the ability to create your own web pages, mobile applications, and so much more. JavaScript is a versatile code that is going to help you in more ways than one.

Finally, if you found this book useful in anyway, a review on Amazon is always appreciated!

JavaScript:

Tips and Tricks
to Programming Code
with JavaScript

Charlie Masterson

Table of Contents

holder.

Introduction

JavaScript, a widely used programming language is an integral part of a web programmer's life and programming skills. It is the foundation of what makes web pages interactive and dynamic; it's used to create forms, polls, and quizzes and is often used in lightweight programming.

It's the most common computer language utilized in all major browsers from the humble Internet Explorer to the advanced Google Chrome and Safari. Most computers already come with JavaScript installed into their system, so it's guaranteed that visitors from all around the world will be able to view what you have on the internet, software and computer programs.

Everyone and anyone can use JavaScript without purchasing licenses.

Also, in case you are wondering, JavaScript and Java are completely two different computer languages used for various programming needs.

The thing about JavaScript is that there are plenty of JavaScript codes pre-written that you can copy and plug it directly into your web page for a similar effect, instead of writing all the codes again on your own. All you every need to know is find out what the code does and then where to copy and paste these codes in its particular places to get the desired effect, like baking a cake.

If you have opened this book, we are not going to go into the specifics of JavaScript and teach you the basics or the steps of coding and programming or pasting these codes into your web page.

What this book will do however, is give you loads of tips and tricks to help you navigate your way into becoming a better programmer and use JavaScript more efficiently.

A Brief History of JavaScript

Back in the 1990s when the world wide web was still in its developing stages, all web pages were static- there was no way of interaction between user and the website. What you saw was exactly what was set up.

Two of the most popular browsers at that time were Internet Explorer and Netscape Navigator. While the Internet Explorer was a lot more famous, Netscape was the first to bring about a programming language that enabled a webpage to be interactive with the user. This language was called LiveScript and it was integrated into Netscape's browsers. Anyone using Netscape could now interact with the pages thanks to this language. LiveScript did not need a plugin.

On another note, there was another programming language that required a separate plugin to run and that was called Java. At this point, Java became very well known and Netscape decided to cash in on this by renaming LiveScript to JavaScript.

As mentioned earlier, there is a wide difference between Java and JavaScript, although their codes appear similar.

And then Internet Explorer, with the need to innovate, is updated and supported by two integrated languages which was called vbscript based on BASIC programming and another called Jscript which was similar to JavaScript.

By the time Internet Explore became the most used browser, JavaScript also became the most accepted and widely used programming language to be used in up and coming web browsers.

Fast forward to 1996, scripting language became so important that it was risky to leave it to competing browser developers. In that same year, JavaScript was given to the ECMA, the international standards body who was responsible for the development of programming language. This language was officially renamed to ECMAScript but programmers and everyone else still know it as JavaScript.

It's a lot cooler sounding too.

The Importance of JavaScript

To understand how JavaScript is used is to understand the basic roles of other programming that goes into the development of a webpage.

Simply put:

1- HTML is the language that is used to markup content. HTML defines what the content is, and not how it looks on a webpage.

2- CSS is used to define the appearance of a webpage. This entails specifying which media corresponds to which command

Just by using these two languages, you can already create a static webpage that can be easily accessed regardless of what device you view the webpage on and what browser is used to access it.

However, the biggest disadvantage of web pages like these is that the only way your visitor can interact with your static site is by filling up a form and waiting for a new page to load.

That's where JavaScript comes in.

JavaScript converts static pages into one that is interactive so that site visitors won't need to wait for a new page to load every time they make a request. JavaScript enables your web pages to have a behavior - to make them capable of responding to a variety of actions in real-time.

With JavaScript, you can:

- Validate each field in a form as a user enters the information and not fill up the whole form and only be told the user made a typo after submitting the form.
- Enables your page to be interactive in more ways than involving forms

- Add animations to a page, attracting attention to a specific part or making the website easier to use
- Load new and heavier images and scripts on the webpage
- Enables you to improve your visitor’s experiences

Chapter 1:
Simple Tricks to Help You Learn JavaScript Faster

Oftentimes when anyone is on the path of learning JavaScript or any coding skill, they often experience some of these issues:

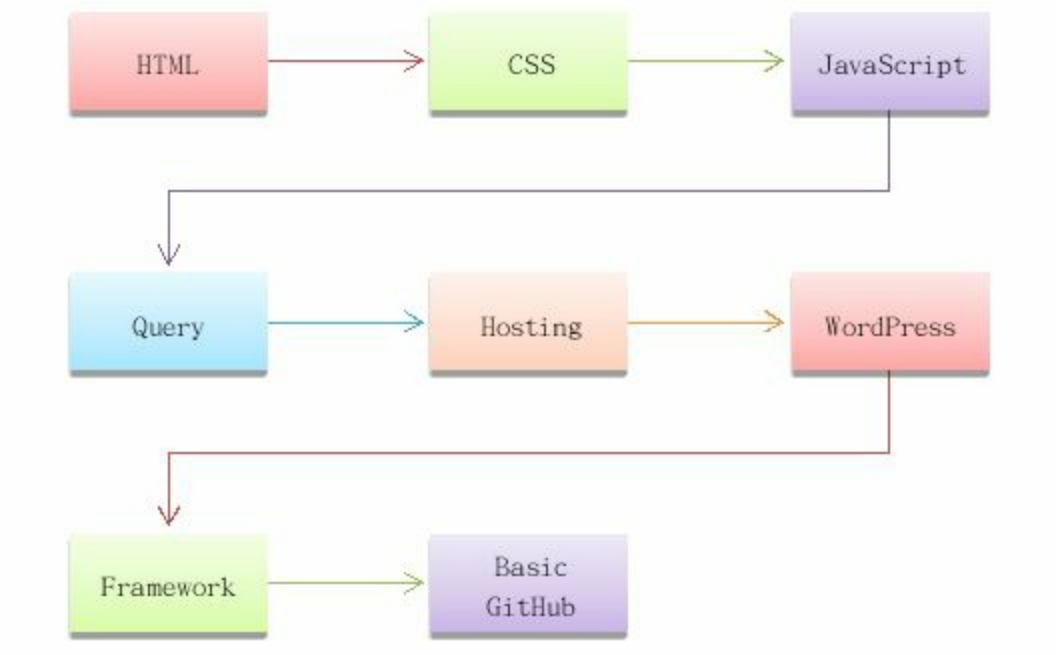
- Confusing concepts
- Time and motivation to learn
- Easy to forget codes
- Tooling landscape keeps changing

In this chapter, we are going to look at conquering these challenges with some mind tricks that can help you learn the JavaScript landscape faster, making you a more productive coder.

TIP 1: CREATE A ROADMAP OF WHAT YOU WANT TO LEARN

Let's face it- there are plenty of frameworks to use and new coders are always at the crossroads of which to start off with first. Instead of researching which frameworks to start on, one of the best ways to overcome this would be to create your roadmap of learning what you need to know.

For example, if you want to become a front-end developer, then your roadmap could look like this:



HTML and CSS are the most basic coding knowledge you need to know before you proceed to learn other things. Then you can move on to JavaScript and jQuery. Once you have created your roadmap, it'll make it easier for you to focus on what is essential first so you don't waste time and effort.

TIP 2: TAKE YOUR TIME

Wanting to understand coding quickly will be one of the harmful things that can halt your progress of learning coding and JavaScript.

Attempting to skim something and then moving on to the next issue at hand will only make you forget things even more. You'll forget what you read the first time, go back to it and read it to refresh your memory and then move on. By this time, you'd have forgotten something else entirely. This cycle will keep repeating, and then you'll find yourself discouraged. To prevent this from happening, you can do this:

- Limit the amount of coding to a particular task. Learn one thing at a time
- Practice what you've learned

By learning and practicing your coding skills, you'll find that it is a lot easier to remember things and absorb them as you go on learning new things. This

process may seem longer, but practice does make it perfect.

TIP 3: PRACTICE WITH THE RIGHT MINDSET

Remember this: if you try to take shortcuts in your JavaScript practice, you will only end up taking a longer time to learn it. Yes practicing is boring and repetitive but here are some ways to make it exciting:

Try treating each new concept in JavaScript like a new toy, car or iPhone or even a new dress. You are excited to try it, especially when it's new and you use it several times. Apply something cool with the concept that you've just learned; surprise yourself and show your friends your new skill.

With a slightly more playful approach to learning new things, you can have fun while learning. You'll also find that you remember these concepts longer and you learn a lot faster.

TIP 4: COMPARTMENTALIZING YOUR TIME

Finding time to practice coding can be difficult. However, the same people who code are also heavy social media users, often spending time on YouTube and Facebook, Reddit or even Wikipedia. Whether or not you are like this, there are valuable lessons to be learned from using social media.

The fact that we all spend so much time on Facebook even if all we just wanted to do was just to look at one notification says a lot. We get sucked into Facebook because one thing leads to another and because you constantly have people updating their status and walls and your news feed always have something new to see.

Using this same psychology, you can use this to your advantage in learning code. Instead of committing several hours of learning codes, you can just tell yourself that you will commit to 15 whole minutes without any disturbance.

Then look at Facebook, take a little break.

Then come back again and do another 15 minutes.

But, this technique can only work if stay focused to your goal, and not allow Facebook or anything other suck you into, and make you totally forget what you were doing.

TIP 5: GIVE YOURSELF TIME

There's a saying that goes if you take your time to think slowly, you'll learn a lot faster. Coding needs time. You need time to understand it. Otherwise, you'll only get more confused. Take your time to figure out what it is that you are trying to learn. Explain it to someone if you have to. Take your time to go through each step, and each line of code before you figure it all in one go.

TIP 6: WRITE IN PLAIN LANGUAGE

Coding can be incredibly complex. If what you are reading becomes too complicated or unfamiliar, then try writing it out in simple terms- or in any way that you can understand it first. That way, you can figure out what it is that you want to code before actually writing it.

This will enable you to code much easily the next time around, and you'll also be faster at writing your codes because you wouldn't need to stop and think with each line of code. You'll also be able to spot bugs even before you code them in.

BOTTOM LINE

Many of these tips can make you learn JavaScript easily and at a faster rate. The truth is, you would be able to apply these tips to learn any sort of lesson you need to.

Chapter 2:
JavaScript Cheat Sheet – Must Haves for Every Programmer

By now you know how important learning JavaScript is if you are serious about becoming a programmer. JavaScript is no doubt the most practical programming language in our digital world. It is used in app development, web apps, web pages and programming in general.

In this chapter, we introduce to you some of the must-have cheat sheets you can get your hands on to make yourself better at coding and writing the programming language. These cheat sheets are handy to have as they help you master the basics while you are learning.

CHEATOGRAPHY JAVASCRIPT CHEAT SHEET

With this site, you can find various listing methods and functions that include guides to common expressions as well as more complicated items such as the XMLHttpRequest object as well as many references to JavaScript.

QUICKLYCODE- PROGRAMMING CHEAT SHEET AND MANY OTHERS

This site links you to even more cheat sheets and other resources for your programming needs from JavaScript to SQL, Java, WordPress, Ruby on Rails and PHP.

[FIRSTSITEGUIDE JAVASCRIPT CHEAT SHEET]

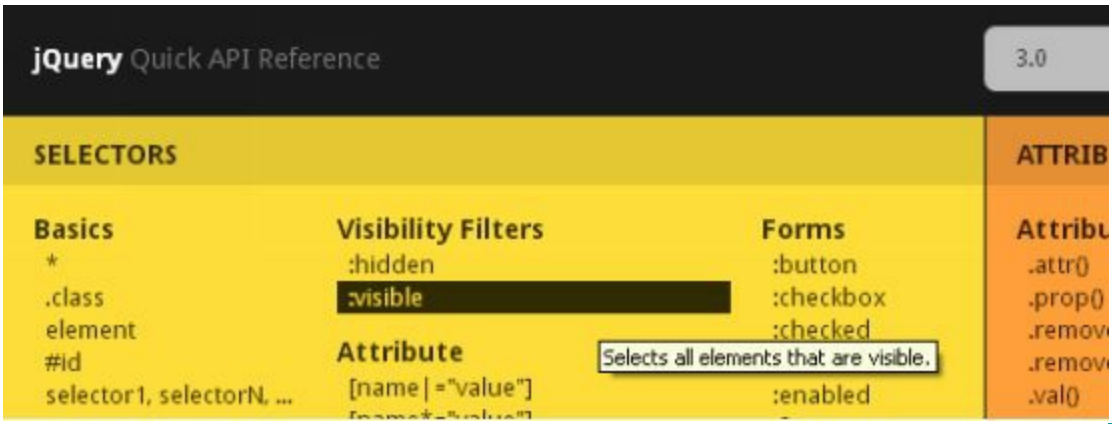
FirstSiteGuide comprises several self-confessed web geeks who offer advice and tutorials on everything from SEO to marketing, web hosting, design and development. They also link you up to free resources and toolkits.

[MOOTOOLS CHEAT SHEET]

This site gives you all kinds of JavaScript utilities that are excellent for JavaScript developers in the intermediate or advanced levels. You can write powerful and flexible coding with the many well documented and elegant APIs.

[jQUERY CHEATSHEET]

Managed by Oscar Otero, the jQUERY Cheat Sheet enables developers to find functions and properties related to jQuery 1.3 library.



[STEAMFEED HTML & JAVASCRIPT SHORTCUTS]

The good people at SteamFeed came up with a great cheat sheet that puts HTML codes and JavaScript information in a clean and easily readable infographic that you can print and place at reading distance so you have easy access when practicing your coding.

[WEB DESIGN CHEAT SHEET]

If you need a site that you can go to for all sorts of cheat sheets related to web design, then Sellfy has some good resources for CSS, responsive web design cheat sheets, HTML tags and many more.

[THE ULTIMATE JAVASCRIPT CHEAT SHEET]

This cheat list from the website codementor, offers a quick overview of the JavaScript language. You have the choice to either read it from the start or to jump to a topic that you like.

The Ultimate JavaScript Cheat Sheet

Chapter 3:
Tips and Tricks to Design and Build Your Own JavaScript Library

JavaScript Library you say?

No, it doesn't contain books. It does, however, include packaged codes that web designers, developers, and programmers can use in their projects that will save time and effort. In other words, you do not need to reinvent the wheel when you want to apply the same package of codes to deliver the same kind of effect on different websites and programs.

So what makes a JavaScript Library?

Here are a few distinct features of a JavaScript Library:

- It has one file or several files in a single folder
- Its coding must be maintained separately and remain as-is when you use it to implement a project
- It should allow the user to set project-specific behavior or configuration

In this chapter, we will talk about why you should build a JavaScript library and how they are built- in the simplest ways possible.

Why build your own JavaScript Library?

Firstly, it makes things convenient for programmers and web developers alike, especially if you code and develop plenty of programs, websites, and web applications. Having a library with existing codes makes your job easier and saves you time since you do not need to keep digging up old projects from some obscure folder in your hard drive. All you just need to is pull it out of your library. You have a convenient place to store all your codes; you have a place to fragment all your applications and keep them all in an easier to maintain location.

Essentially, any code that makes your work easier and which can be reused is a great component to be added to the library. One such good example is jQuery. jQuery's API is a more simplified DOM API. Back when cross-browser DOM manipulation was difficult, jQuery was essential.

And if an open-source venture becomes increasingly popular, more developers use it, and it will eventually encourage more developers to join in and assist with this venture by contributing to the code and submitting issues that will help make it easier to manipulate. This contribution will benefit the library and all other experiments that depend on it.

Popular open-source projects can also lead to fantastic opportunities for developers alike. A tech firm or any organization may be impressed by the quality of a developer's work, and they will be offered a job. It could also mean that a company will provide a developer to collaborate on a project or get them to integrate their project into an application of their own.

But for most developers or coders, having their own library is a hobby; one that enables them to write multiple codes and help others and contribute to the process.

Your Goals and Scope of Work

Before you set out writing your codes, you need to be clear on what your goals are for this library. With this, it will help you keep focused on what problems you encounter and how to solve them with your library.

Your primary goal is also to make your library easy to use. Essentially, the simpler your API, the easier it will be for anyone to use your library.

So ask yourself- what problems does your library solve and how can you solve it? Will you utilize codes from someone else's library or will you write everything yourself?

In chapter 1, we discussed creating a roadmap. This roadmap for your library will be one of the techniques you use in any other code development you set out to do.

A Roadmap for your library:

- List out the features you want in your library
- Make these features minimum and practical
- Create milestones for each feature

What you are doing with the roadmap is setting goals for each function,

essentially breaking your project into manageable sizes so it is easier to accomplish and the process of creating it is a lot more enjoyable. It'll also keep you sane and focused.

Your Library API Design

When creating your library, think of yourself as the end-user. By putting yourself as the user, you will work towards a user-centric design and development. This will make your library more convenient to use by anyone who has access to it.

Testing Your API

The best way to test the quality of your API is to use it for your projects. What you can do is to substitute the application code with your library and see if the result it gives covers all the features you want. Keep to the minimum bare essentials as possible and also provide room for flexibility to change and revise anything that may seem slightly off.

Here is one example of what the implementation coding or User-Agent string library outline would look like:
Code:

```
// Start with empty UserAgent string
var userAgent = new UserAgent;

// Create and add first product: EvilCorpBrowser/1.2 (X11; Linux; en-us)
var application = new UserAgent.Product('EvilCorpBrowser', '1.2');
application.setComment('X11', 'Linux', 'en-us');
userAgent.addProduct(application);

// Create and add second product: Blink/20420101
var engine = new UserAgent.Product('Blink', '20420101');
userAgent.addProduct(engine);

// EvilCorpBrowser/1.2 (X11; Linux; en-us) Blink/20420101
userAgent.toString();
```

```
// Make some more changes to engine product
engine.setComment('Hello World');

// EvilCorpBrowser/1.2 (X11; Linux; en-us) Blink/20420101 (Hello World)
userAgent.toString();
```

Depending on how complex your library is, you also need to give some thought to structuring. One way to do this is to utilize design patterns to structure your library as well as overcome any technical issues you may experience along the way.

Structuring also reduces the risk of re-aligning huge parts when you are adding new features to your library.

Flexibility and Customization

A cool thing about most JavaScript libraries out there is its flexibility. However, many developers face problems when deciding what they can and cannot customize. One of this is the chart.js and D3.js.

Both elements are great libraries to visualize data. If you need more control over your graphics, then the D3.js is what you want. If you want something that is easy to create yet give you different types of built-in charts, then Chart.js is the one to use.

There are three ways that you can give your user control over your library:

1- Configuration

2- Exposing public methods

3- Callbacks & events

Configuring a library is often done at the project initialization stage, but some of these libraries also allow you to set during run-time. These configurations are commonly limited to simple issues as changing these settings will only update these values for use later on.

Code:

```
// Configure at initialization
var userAgent = new UserAgent({
  commentSeparator: ';'
});

// Run-time configuration using a public method
userAgent.setOption('commentSeparator', '-');

// Run-time configuration using a public property
userAgent.commentSeparator = '-';
```

Certain methods are exposed intentionally to interact with an instance (getters) such as to retrieve data from the instance or to put data in the instance (setters) as well as perform certain actions.

Code:

```
var userAgent = new UserAgent;

// A getter to retrieve comments from all products
userAgent.getComments();

// An action to shuffle the order of all products
userAgent.shuffleProducts();
```

Callbacks are also sometimes accepted and passed using public methods and it often used to run user-codes after an asynchronous task.

```
var userAgent = new UserAgent;

userAgent.doAsyncThing(function asyncThingDone() {
  // Run code after async thing is done
});
```

Events have a huge potential in JavaScript libraries. Slightly similar to callbacks, they have a slight difference where adding event handlers do not

cause trigger actions. Events are also used to indicate certain events and can provide additional information to the user as well as return a specific value for the library to correspond with.

Code:

```
var userAgent = new UserAgent;

// Validate a product on addition
userAgent.on('product.add', function onProductAdd(e, product) {
  var shouldAddProduct = product.toString().length < 5;

  // Tell the library to add the product or not
  return shouldAddProduct;
});
```

You may also enable your users to extend your own library. In order to do this, you need to make available your public method or property so that users can populate it, such as the Angular modules.

Code:

`(angular.module('myModule'))` and jQuery's `fn(jQuery.fn.myPlugin)`

You can also just do nothing of the above and allow users to access your library through its namespace.

Code:

```
// AngryUserAgent module
// Has access to UserAgent namespace
(function AngryUserAgent(UserAgent) {
  // Create new method .toAngryString()
  UserAgent.prototype.toAngryString = function() {
    return this.toString().toUpperCase();
  };

})(UserAgent);
```

```
// Application code
var userAgent = new UserAgent;
// ...

// EVILCORPBROWSER/1.2 (X11; LINUX; EN-US) BLINK/20420101
userAgent.toAngryString();
```

Similarly, this allows you to overwrite methods as well.

```
// AngryUserAgent module
(function AngryUserAgent(UserAgent) {

  // Store old .toString() method for later use
  var _toString = UserAgent.prototype.toString;

  // Overwrite .toString()
  UserAgent.prototype.toString = function() {
    return _toString.call(this).toUpperCase();
  };

})(UserAgent);

var userAgent = new UserAgent;
// ...

// EVILCORPBROWSER/1.2 (X11; LINUX; EN-US) BLINK/20420101
userAgent.toString();
```

In this case, giving users access to your library's namespace also gives you less control over how plugins and extensions are defined.

To allow for extensions to follow some certain convention, you need to write documentation.

Testing

For test-driven development, an outline is essential. In other words, this is the time when you write down the features and criteria in the form of tests, even before you write down the actual library.

If your test show you that a feature behaves like it was supposed to, and you write it down before writing it in your library, then this strategy is called behavior-driven development.

If your tests cover every aspect of your library then and if your code surpasses all these tests, then you can safely say that your library is working well.

Other testing frameworks that you can use are:

Unit Test Your JavaScript Using Mocha and Chai by Jani Hartikainen, where you can write unit tests in Mocha.

Testing JavaScript with Jasmine, Travis, and Karma by Tim Evko shows you how to set up a testing pipeline coherently with another framework called Jasmine.

Below mentioned is another outline that you can try by Tim Severien. He has used a Jasmine test for his library and it looks something like this.

Code:

```
describe('Basic usage', function () {
  it('should generate a single product', function () {
    // Create a single product
    var product = new UserAgent.Product('EvilCorpBrowser', '1.2');
    product.setComment('X11', 'Linux', 'en-us');

    expect(product.toString())
      .toBe('EvilCorpBrowser/1.2 (X11; Linux; en-us)');
  });

  it('should combine several products', function () {
    var userAgent = new UserAgent;

    // Create and add first product
    var application = new UserAgent.Product('EvilCorpBrowser', '1.2');
    application.setComment('X11', 'Linux', 'en-us');
    userAgent.addProduct(application);

    // Create and add second product
```

```
    var engine = new UserAgent.Product('Blink', '20420101');
    userAgent.addProduct(engine);

    expect(userAgent.toString())
      .toBe('EvilCorpBrowser/1.2 (X11; Linux; en-us) Blink/20420101');
  });

  it('should update products correctly', function () {
    var userAgent = new UserAgent;

    // Create and add first product
    var application = new UserAgent.Product('EvilCorpBrowser', '1.2');
    application.setComment('X11', 'Linux', 'en-us');
    userAgent.addProduct(application);

    // Update first product
    application.setComment('X11', 'Linux', 'nl-nl');

    expect(userAgent.toString())
      .toBe('EvilCorpBrowser/1.2 (X11; Linux; nl-nl)');
  });
});
```

Once you have done your testing and you are completely satisfied with the API design for your first library version, then you can start thinking about the site's architecture and how exactly will your library be used.

Module Loader Compatibility

Not many developers use module loaders, and this is entirely up to you too. However, you need to be aware that some developers that use your library might choose to use a module loader, so you need to make your library compatible with this feature.

You can choose some of the popular module loaders such as AMD, RequireJS, and CommonJS. If choosing between these functions is hard for you, you can also try Universal Module Definition (UMD) which is aimed at supporting several types of module loaders.

To make your library UMD compatible, you can also find some variations such

as the UMD GitHub repository. Begin to create your library by using one of these templates. Another tip is to use Babel to compile ES5 combined with Babel's UMD plugin if you choose to use ES2015 import/export syntax function. This way, you can use ES2015 in your projects as well as produce a library that can be used by all kinds of developers using various modules.

Documentation

You need to start your documentation with basic information such as your project name as well as a short description of what it is about. This will help others understand what your library is meant to do and whether using your library is a good choice for them. You can also include a little bit more information on things such as scope and goals so users are better informed. Your roadmap can also be included so users know what they can expect in the future in your library development and how they can also contribute to it.

API, Tutorials and Examples

API documentation is also an essential part of documentation as it will tell users how exactly to use your library. Other good additions would be examples as well as tutorials. This can take up some time so you can do this as you library development is on-going.

Meta-tasks

Users will also want to make changes to your library and this is usually in the form of a contribution but some may want to create a custom build only for private use. For these users, you can include documentation for meta-tasks that include a list of commands to build the library, run testing, generate and convert or download data.

Contribution

Contributions are great especially when you open-source your library. Another documentation you can add in is to guide contributors to explain the steps in making a contribution to the library and what kind of criteria should they fill

before making a contribution. This will make your review process easier to accept contributions that can actively upgrade and enhance your library.

License
Licenses are important too. Include licenses so everyone knows what material is copyrighted. ChooseALicense.com is recommended as an excellent resource for acquiring a license without any legal advisor. Save your chosen license as text in a LICENSE.txt file format in your project's root.

Adding Versions/Releases to Your Git Repository

Versioning is necessary for a good library. The current standard for version naming is via the SemVer, short for Semantic Versioning. This version comprises three numbers to indicate different changes for major, minor and patch. If you do have a git repository, you can also add in version numbers. Consider the snapshots of your repository or Tags. To create a tag, open its terminal and do the following:

```
# git tag -a [version] -m [version message]
git tag -a v1.2.0 -m "Awesome Library v1.2.0"
```

GitHub for example provides a comprehensive view of all your versions as well as download links for each one of them.

Publishing to Common Repositories

npm

Most programming languages are equipped with package managers or they at least have a third party package manager that enables users to pull in specific libraries for that specific programming language. For example, PHP's manager is Composer and Ruby on Rails is RubyGems. For a standalone JavaScript engine such as Node.js you have npm. By default, your npm package will be available publicly. But you also have the option of publishing the following:

- private packages

- set up a private registry
- completely avoid publishing

However, should you decide to publish your package, then you need the package.json file. You can do this manually or use an interactive wizard for this. To begin this wizard, start by typing:

```
npm init
```

The `version` property should match your git tag. Include the `README.md` file as well. Just like GitHub, npm uses that for the page presenting your package.

Once done, publish your package by typing:

```
npm publish
```

That's it! You have successfully published your npm package.

Bower

Another useful package manager called Bower surfaced a few years ago. This is a package manager designed for the web. Here, you can find all major front-end asserts. The only thing is, your library will benefit from using Bower to publish your package is if your library set up to be browser-compatible.

npm packages are primarily for JavaScript and many front-end packages still use npm but Bower is still fundamentally popular so go ahead and publish using Bower too.

To generate a bower.json file, type:

```
bower init
```

Just like `npm init`, the instructions are self-explanatory. Finally, to publish your package:

```
bower register awesomelib https://github.com/you/awesomelib
```

Now you have your library available for everyone to use in their Node projects or on the web!

Conclusion

Remember that your code development is the library. Make sure your library solves a problem, it is convenient and easy to use and above all stable. This will ensure that you and many other developers who use it are happy.

The tasks explained above are easily automated such as running tests and creating tags, updating your versions and publishing your packages.

Chapter 4:
JavaScript Tips & Tricks to Keep and Remember

In this chapter, we will look into some short and useful tips related to improving your JavaScript skills and enhancing your code writing.

USING === Instead of ==	The == (or !=) operator performs an automatic type conversion if needed. The === (or !==) operator will not perform any conversion.
	
	What it will do is compare the value and the type, which could be considered faster (jsPref) than ==.
	
	[10] == 10 // is true
	[10] === 10 // is false
	
	'10' == 10 // is true
	'10' === 10 // is false
	
	[] == 0 // is true
	[] === 0 // is false
	
	" == false // is true but true == "a" is false
	" === false // is false

charAt() Shorthand	To do this, you can utilize the eval() function however this bracket notation shorthand technique is much simpler than an evaluation Longhand: "myString".charAt(0); Shorthand: "myString"[0]; // Returns 'm'

Strings convert to Numbers	The fastest and easiest (jsPerf) way to do this would be using the + (plus) operator. var one = '1'; var numberOne = +one; // Number 1

	You can also use the - (minus) operator which type-converts the value into number but also negates it. var one = '1'; var negativeNumberOne = -one; // Number -1
Truthy/falsy values converted to boolean	You can convert a Truthy or Falsy value to true boolean with '!!' !!"" // false !!0 // false !!null // false !!undefined // false !!NaN // false !!"hello" // true !!1 // true !!{} // true !![] // true

Using JSON.Stringify	Let's say there is an object with properties "prop1", "prop2", "prop3". A**dditional params can be passed** to **JSON.stringify** to selectively write properties of the object to string like:
	
	var obj = {
	'prop1': 'value1',
	'prop2': 'value2',
	'prop3': 'value3'
	};
	
	var selectedProperties = ['prop1', 'prop2'];
	
	var str = JSON.stringify(obj, selectedProperties);
	
	// str
	// {"prop1":"value1","prop2":"value2 "}
	
	The **"str"** will contain only info on selected properties only. Instead of array we can pass a function also.

	function selectedProperties(key, val) {
	  // the first val will be the entire object, key is empty string
	  if (!key) {
	  return val;
	  }
	
	  if (key === 'prop1' || key === 'prop2') {
	  return val;
	  }
	
	  return;
	}
	The last optional paramit takes is to modify the way it writes the object to string.
	var str = JSON.stringify(obj, selectedProperties, '\t\t');
	
	/* str output with double tabs in every line.
	{
	  "prop1": "value1",
	  "prop2": "value2"
	}
	*/

Single Method for Arrays and Writing Single Element	Instead of writing separate methods to hold an array and a single element parameter, you can write functions so that they can handle both.
	You just have to concat everything into an array first. Array.concat will accept an array or a single element.
	
	function printUpperCase(words) {
	 var elements = [].concat(words || []);
	 for (var i = 0; i < elements.length; i++) {
	  console.log(elements[i].toUpperCase());
	 }
	}
	
	Print UpperCase is now ready to accept a single node or an array of nodes as its parameter. It also avoids the potentialTypeError that would be thrown if no parameter was passed.

	printUpperCase("cactus");
	// => CACTUS
	printUpperCase(["cactus", "bear", "potato"]);
	// => CACTUS
	// BEAR
	// POTATO

Measure performance for a JavaScript Block	For quickly measuring performance of a JavaScript block, we can use the console functions
	like console.time(label) andconsole.timeEnd(label)
	console.time("Array initialize");
	var arr = new Array(100),
	  len = arr.length,
	  i;
	for (i = 0; i < len; i++) {
	  arr[i] = new Object();
	};
	console.timeEnd("Array initialize"); // Outputs: Array initialize: 0.711ms

Hoisting	Hoisting will help you manage your function scope. Variable declarations and functions are hoisted to the top where as variable definitions are not.
	
	function doTheThing() {
	// ReferenceError: notDeclared is not defined
	console.log(notDeclared);
	
	// Outputs: undefined
	console.log(definedLater);
	var definedLater;
	
	definedLater = 'I am defined!'
	// Outputs: 'I am defined!'
	console.log(definedLater)
	
	// Outputs: undefined
	console.log(definedSimulateneously);
	var definedSimulateneously = 'I am defined!'
	// Outputs: 'I am defined!'
	console.log(definedSimulateneously)
	
	// Outputs: 'I did it!'
	doSomethingElse();
	
	function doSomethingElse(){
	console.log('I did it!');
	}
	
	// TypeError: undefined is not a function
	functionVar();
	
	var functionVar = function(){
	console.log('I did it!');
	}

Chapter 5:
JavaScript Skills to Know Moving Forward

Whether you like it or not, as a programmer or web or app developer, you need to learn JavaScript. It is considered the computer language of the future simply because it is an integral part of web ad app development.

As common as it is, JavaScript is pretty complicated especially when trends keep changing and with that, comes the ensuing chaos. Despite all that, things are improving in the JavaScript landscape, and with technology evolving, old problems with JavaScript are fixed, no doubt with new ones coming in.

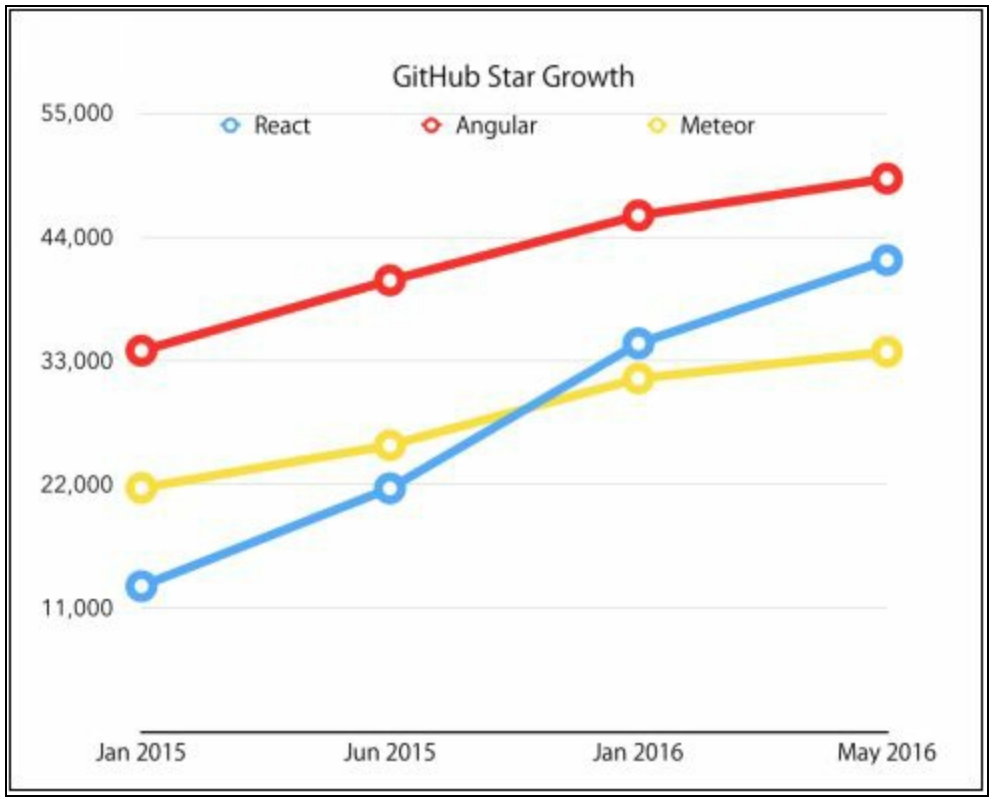
If you are a new developer in 2016, your willingness and ability to learn JavaScript will definitely contribute to the success of your career and your skills as a programmer/coder/hacker or developer.

So what are the trends of JavaScript development should programmers work on going forward?

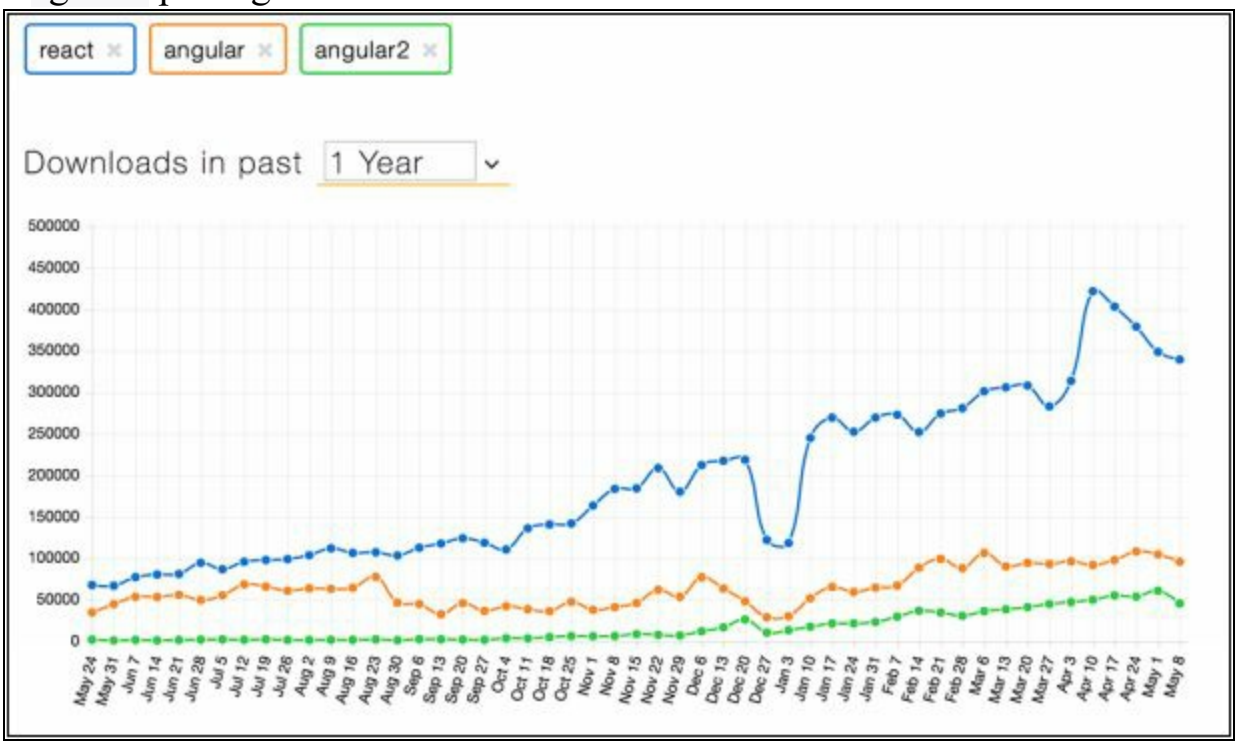
In this chapter, we will look at the various things that will go big in the programming language world.

React & Redux

In 2015, React gained traction especially when more and more programmers started complaining about the AngularJS framework as 2014 came to an end. The fact that news broke out that Angular 2 will be unsuitable to run with Angular 1 made it even worse for the Angular community. React came in and closed this gap in 2016.



Estimates based on screenshots from the internet archive. Perhaps a more telling pointer would be the npm download count for react, angular, and Angular2 packages:



For the past year, React has been getting more traction from the JavaScript community than AngularJS has. So if you look at the graph above, you are probably wondering what happened after August? It shows that npm downloads for React have surged significantly and it also shows exactly when GitHub stars for React being to catch up with AngularJS's stars.

The change is due to Redux, created by Dan Abramov during the ReactEurope2015 conference. Redux had a simple flux implementation that convinced many programmers that it was time to adopt the one and only React.

Before the materialization of Redux, React was turbulent as people were trying to figure out how to best execute the 'Flux' architecture. Since then, many other implementations begin surfacing, and while they aren't bad, they weren't exactly solving any issues that made most JavaScript developers wary.

In spite of the universal rendering and the gold stars for React and its virtual DOM, programmers were still cautious about using it in their projects. Despite this, more and more people started using Redux and realized it makes testing and debugging apps running on JavaScript a lot easier. From then on, Redux rose from the Flux are the real winner.

React, and Redux definitely is the hottest trend for 2016, and it doesn't look like it'll fade away come 2017, regarding front-end 'framework' JavaScript. Netflix, Yahoo!, and Dropbox have all adopted React into their tech arsenal, giving it a confidence boost. Even so, Facebook's Relay looks like a prospective contender in 2017.

Other Frameworks of Interest

Other frameworks may come into the foray and while they aren't comparable but here's a general overview of what other frameworks are out there. Again, things may change with anything regarding technology. What may not work so well this year might end up becoming top of the league the next year.

Without further ado, here are some other frameworks to look into.

Angular2

Plenty of non-tech companies will be looking into using Angular 2 especially businesses that use Microsoft's.NET framework. Angular 2, developed through a partnership between Google and Microsoft was designed to make JavaScript

a more manageable language.

Microsoft themselves were pushing for the.NET aggressively last year as it open-sourced quite some elements and strove to make Microsoft tools more accessible to developers. After the rewrite, the teams hope to fix any sort of escalating issues with the Angular 1 apps, and performance so far has been stellar. Angular2 has been developed to support web components, and according to Google, it is the future for web development. The AngularJS community though has differing opinions about the direction that Angular is taking- all this despite the efforts the Angular team has been investing in helping developers transition properly from Angular 1 to Angular 2.

This rewrite has also allowed React to develop and mature as with its community growing larger each day. Because of this, programmers are skeptical that Angular 2 will be as successful as Angular 1. However, as mentioned above, things can change in the tech world so it could be Angular 2 will have room to grow in 2017. Angular2 is still preferred as an alternative to the React app.

Meteor

Another upcoming trend to look out for is Meteor, which has been enjoying steady downloads and GitHub stars. It integrates well with React and Angular and has plenty of nifty features. As a lightweight, full-stack JavaScript framework, developers are finding the Meteor framework a pleasure to work with. Meteor, always known as a solid beginner framework is great for prototyping as well.

The thing that has prevented Meteor from becoming a mainstream success is due to the hidden complexities that keep cropping up over time particularly when production is concerned. Meteor, despite having raised loads of money, isn't backed by a leading tech enterprise with a broad pool of engineers, like what React & AngularJS are. Notwithstanding the fact that AngularJS has plenty of issues and yet is backed up by Google, Meteor, on the other hand, is considered an investment risk as it's a framework for both the backend and frontend development. To get the most out of the deployment ease, Meteor would also need to be hosted.

Another thing that isn't working for Meteor at the moment is that many developers aren't big fans of the MongoDB and well, Mongo is Meteor's default database. Meteor will remain a nice though for 2017 since more developers will likely wait for more reliable evidence that Meteor can be used for large and more complex application development. Needless to say, Meteor will likely remain niche going forward in 2016 or even 2017. Most professional developers will wait for any evidence that it can be used for developing large, complex applications.

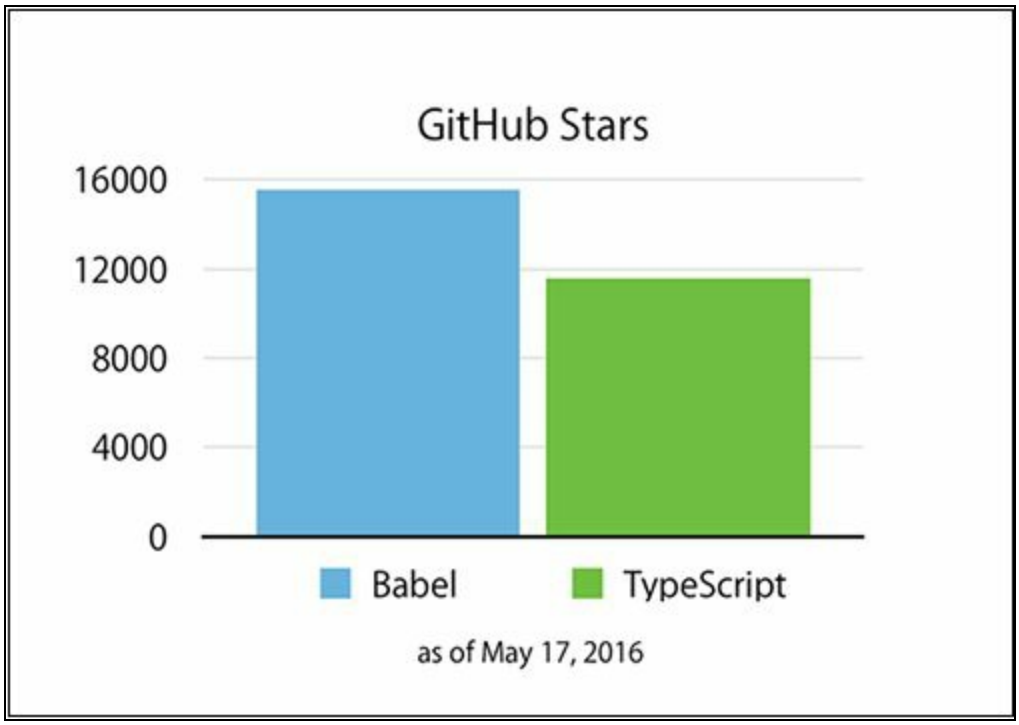
ES6 At Last

One of the things that JavaScript developers do in 2016 moving forward 2017 is to ensure that all their apps are ES2015 compliant. But like many other complications with tech apps and programming language, we have the Babel vs. TypeScript issue.

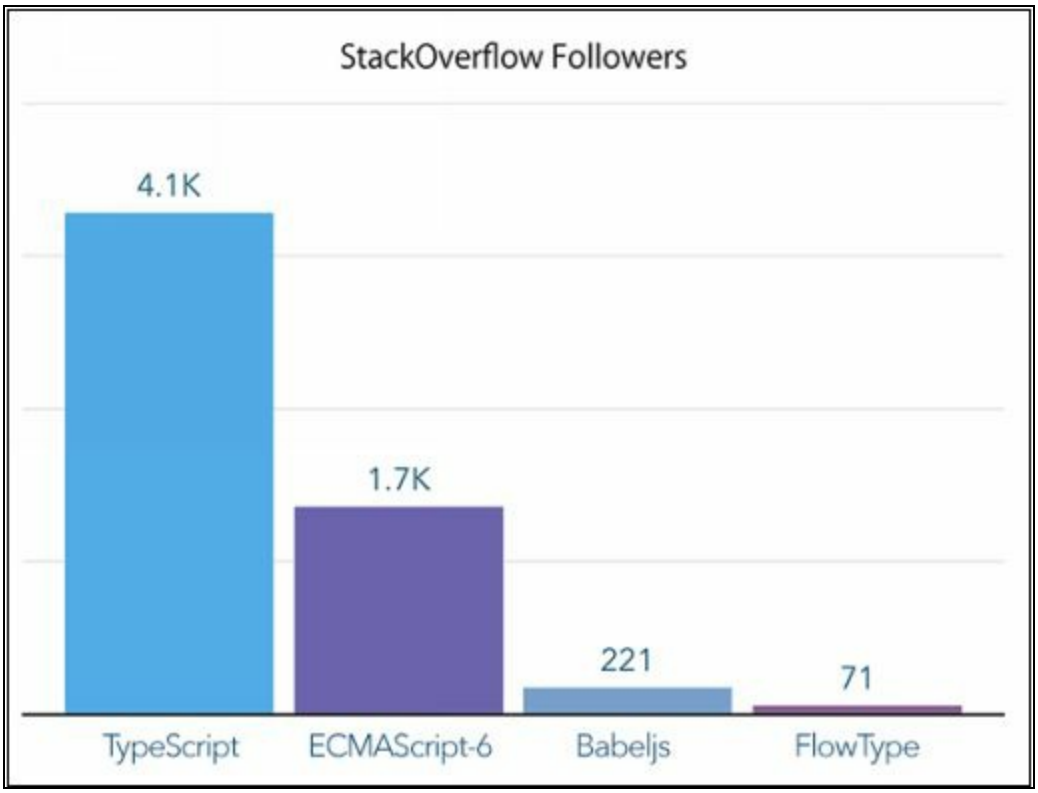
Babel mostly translates the ES6 code to ES5 whereas TypeScript is really a superset of JavaScript that adds necessary but optional static typing to JavaScript and then compiles all of this to plain JavaScript as well. Babel and TypeScript have been created with totally different purposes and therefore not comparable.

Babel, developed by Facebook, supports Flow which also adds in the static type-checking for JavaScript. Using Babel makes a little bit more sense since you can use both consecutively. Babel was also the first to helped bridge the difference between ES5 and ES6 applications.

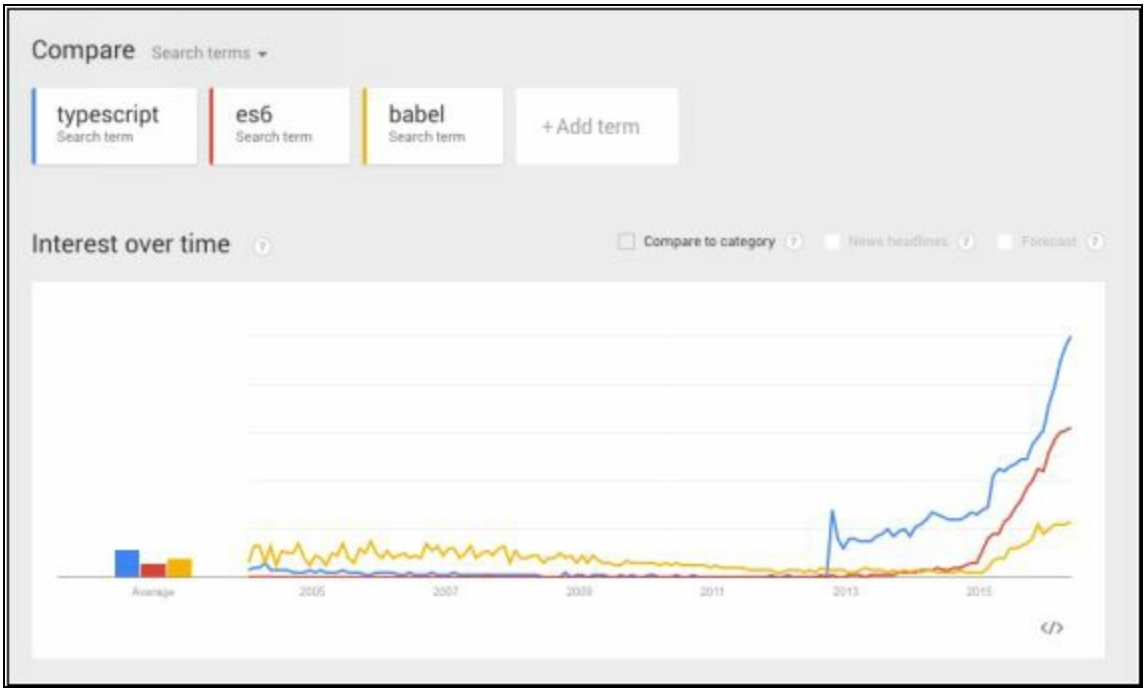
TypeScript on the other hand only supports ES6, thus playing second fiddle to Babel when it comes to progressiveness. Checking in with the GitHub stars, it looks like Babel is the most preferred transpiler. React developers also prefer to use the Babel + Webpack combo.



Despite the stars, TypeScript seems to be a lot more viable and sometimes and even preferred choice especially if you are looking for a solution to make your JavaScript codebase much more manageable. JavaScript has always been known to be a difficult to read language making it even more difficult to debug, due to its weak type-checking system. So in terms of support, TypeScript has enjoyed a larger following than Flow.



The graph below shows the Google Trends Graph showing the growth of interest for TypeScript.



Another reason that TypeScript could be getting popular is because of

Angular2, but just because JavaScript apps get larger focus, that isn't the contributing reason of why TypeScript becomes popular.

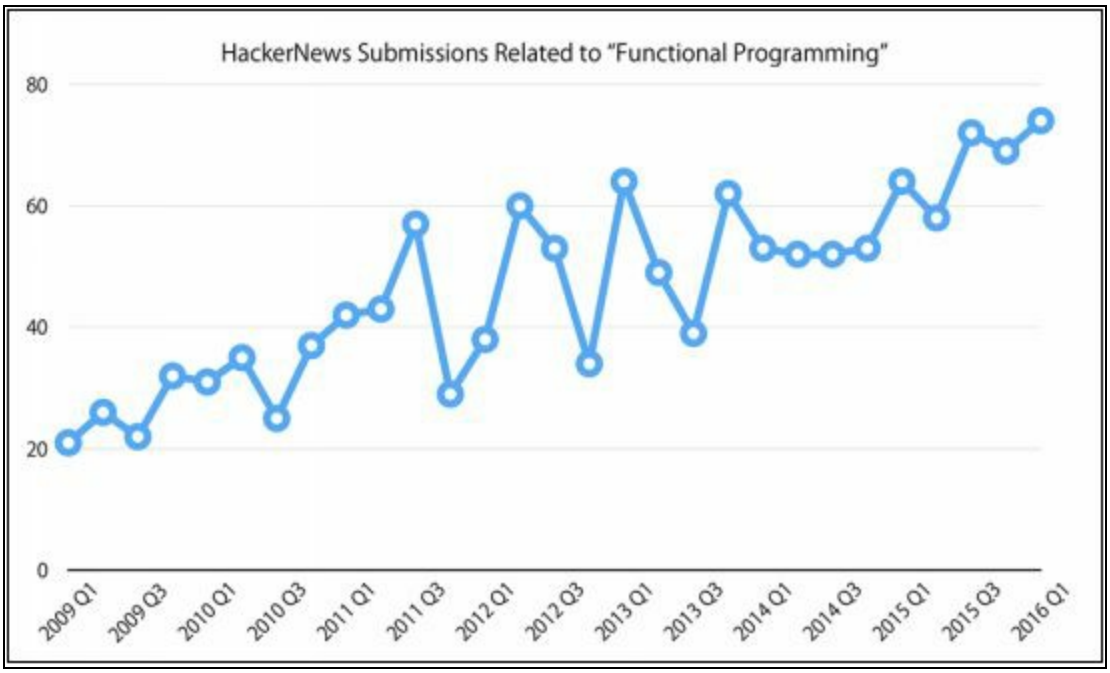
While Redux has made testing much easier, TypeScript makes codebase a lot easier to maintain. Apart from this, Google, Facebook, and Microsoft are collaborating to add on static typing to JavaScript and thus paving the way to EMCAScript.

At the same time, Google dropped AtScript and AngularJS in favor for TypeScript by Microsoft. Flow by Facebook, on the other hand, doesn't have the same community popularity as TypeScript.

This goes to show that statically typed JavaScript is an active and growing trend that even JavaScript non-believers will approve of where is TypeScript is the most solid solution.

Functional Programming is Becoming Mainstream

Plenty of functional programming platforms have been making their way into mainstream programming but very recently, the 'functional programming model' has gained plenty of attention especially with the rise of complex web apps.



Scala is coming back as a favored language among backend developers who want to embrace functional programming. However, Facebook's React, on the other hand, has encouraged front-end JavaScript developers to take on UI development using a practical approach.

More and more positive feedback is coming in for functional programming, and it looks like it will become mainstream in 2017. At this point of time, front-end web development is taking the forefront in reactive and functional programming.

So the React+Redux agenda is really the most sought after beginner-friendly solution for those that are comfortable with programming that is object-oriented. In other words, React only needs developers to take a functional approach to UI whereas Redux is mostly a system that encourages a practical approach to handling data. All other things are done in the OOP-style.

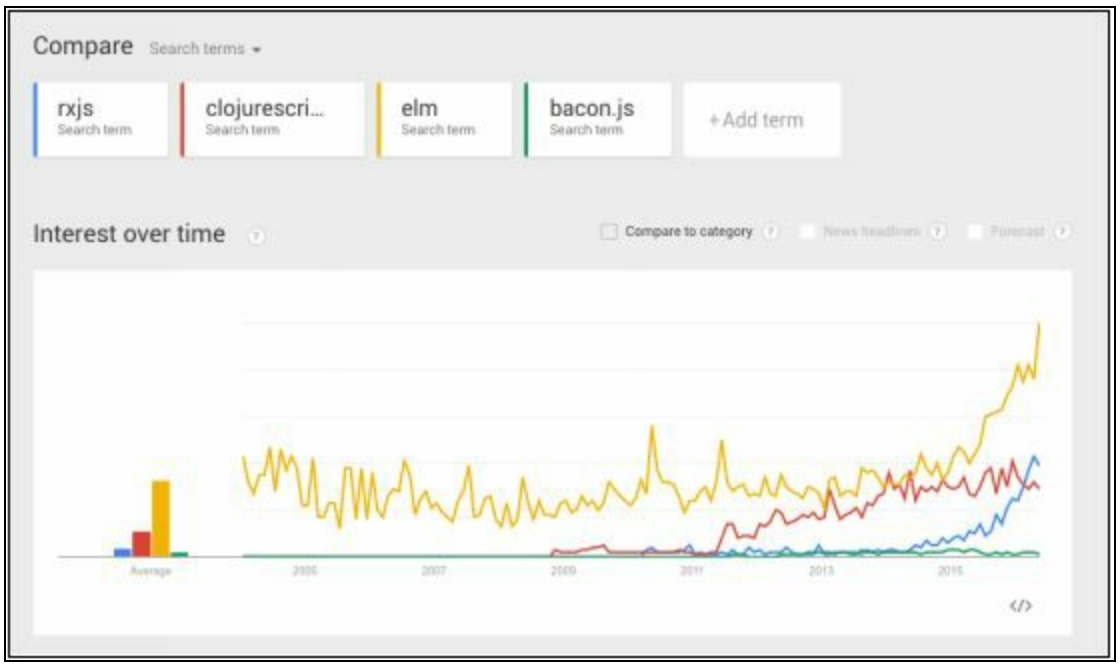
RxJS

But for those who still want to go on track with the full-on functional reactive programming, which is considered the 'true' FRP and still want to use JavaScript, then RxJS is a necessary skill to learn. RxJS is an extension of JavaScript that is reactive that can be used to replace the Flux architecture. It may sound like overkill for simple and small apps but in truth, it builds a powerful foundation for web applications that will go through a complicated

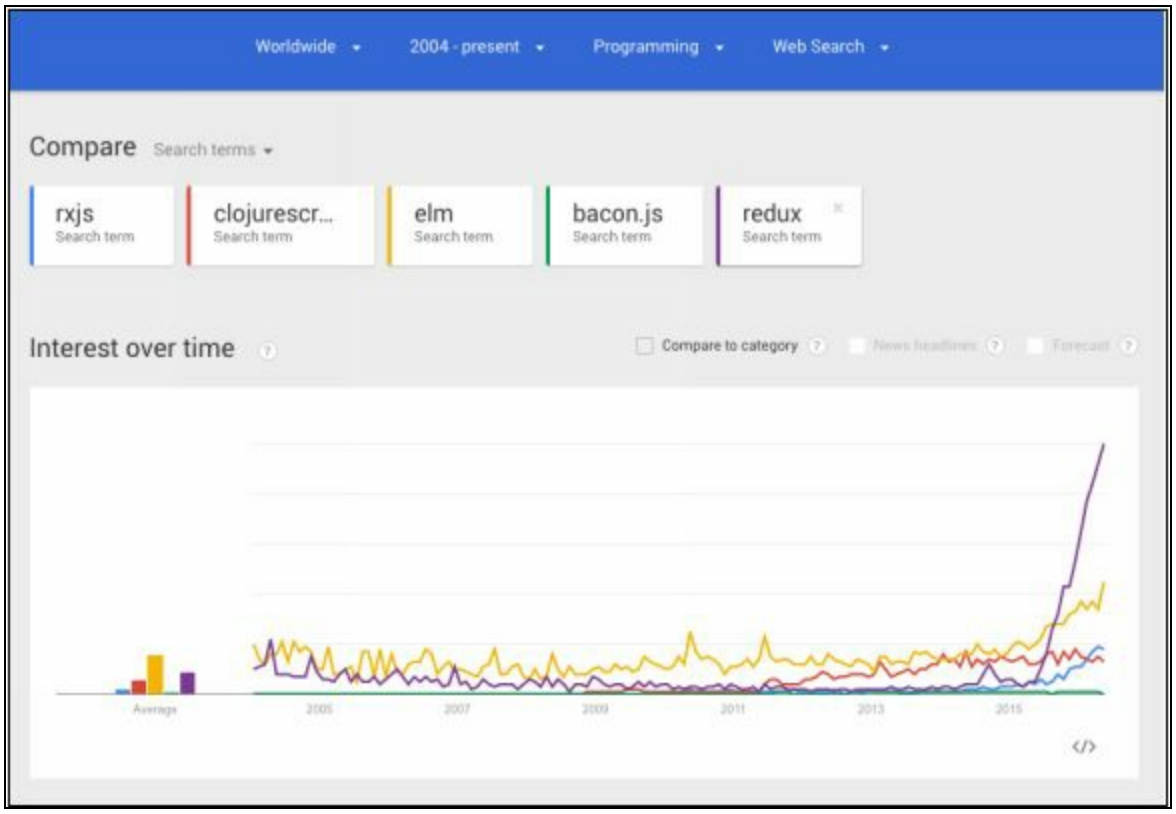
stream of processing.

Netflix uses RxJS while Angular2 also relies on RxJS. Developed by Microsoft, RxJS works incredibly well with TypeScript thus developers can expect that RxJS will continue to be improved.

Learning RxJS though is an entirely different arena. It has a steep learning curve that make most developers give up on doing any FRP in JavaScript and pick up other functional programming languages that are compatible with JavaScript.



Some of the compatible programming that developers choose to use instead of RxJS is ClojureScript and Elm. However all these program's growth and popularity pales in comparison with Redux+React.



The thing is, looking at how functional programming fits perfectly with addressing modern issues in web development, one has to agree that functional programming paradigm and its equivalent concepts will stick around as a necessary skill that JavaScript developers will need to harness and learn.

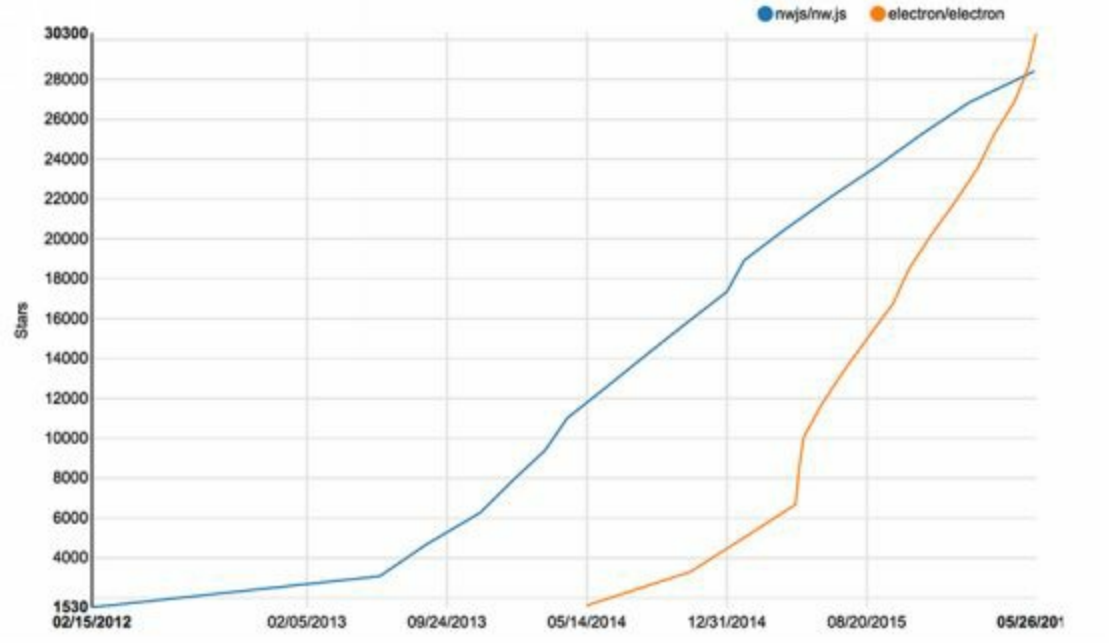
Desktop Framework Showdown: Nw.js vs. Electron

Plenty of data nowadays requires synchronizing data from various platforms such a desktop and mobile. However, many services mostly start out as web apps before they are converted as desktop apps for enhanced user-experience. It is done this way because web app developments are a lot faster and more easily updated. Above all, users can also try out web apps more instantly without having to install anything else.

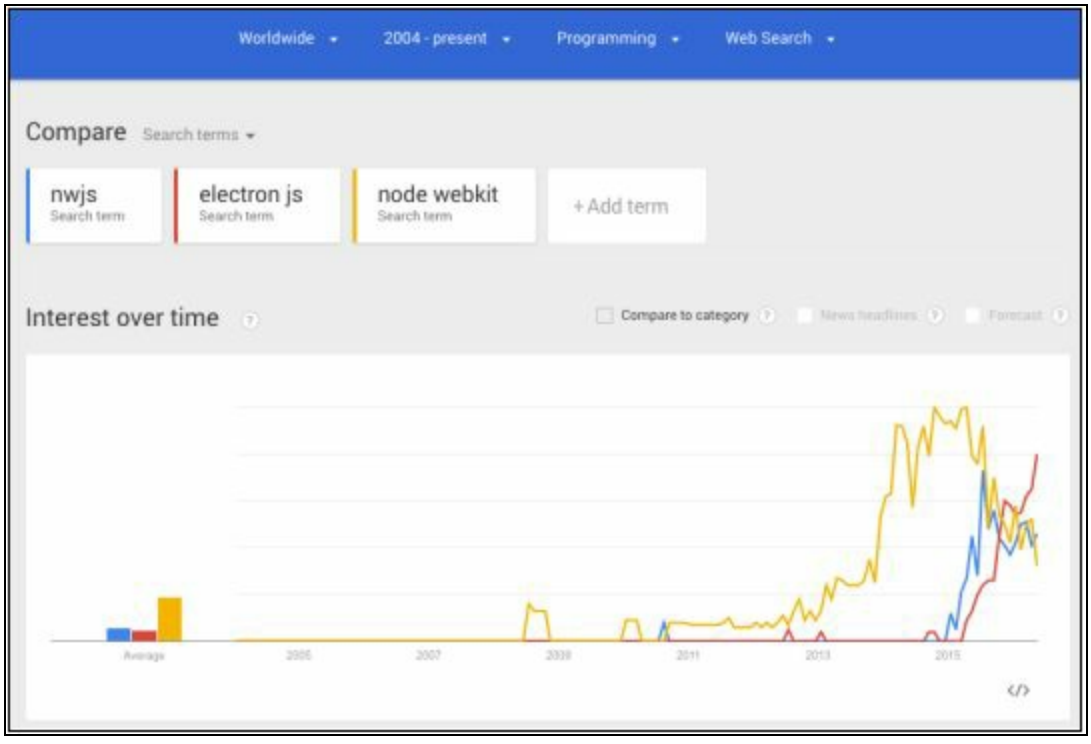
Developers in the past were confined to using CEF if they wanted to use web technologies to create front-end UI for desktop apps. Not an easy process and the apps themselves weren't truly cross-platform compatible. Though that, Node.js has made cross-platform apps easier for desktop frameworks. The shift of using web technologies for desktop app development began to rise

starting 2014.

Thus, let's take a look at the GitHub Star history for both projects:



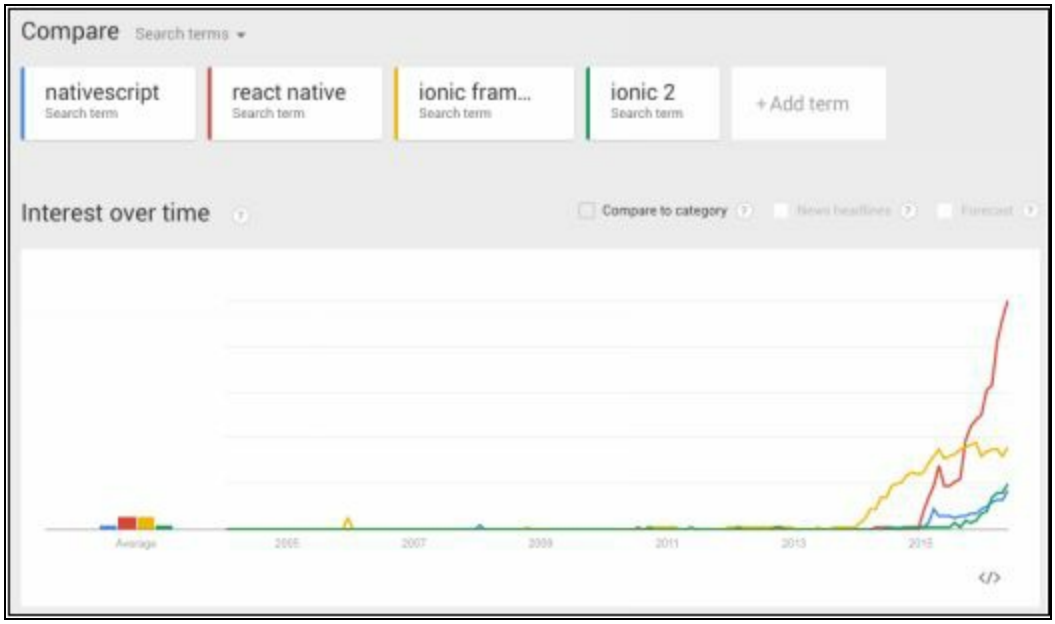
Although Nw.js broke into the scene much earlier and is considered older, it looks like Electron is growing at an explosive rate, while nw.js is growing at a monotonous linear rate.



While Electron is pretty new, it is used by several established organizations like Slack, Microsoft (Visual Studio Code), WordPress, and Sencha. But with the ease of use that Electron's has, the network and community grew very fast. This wave of popularity is very likely to go on into 2017, thus enabling Electron to become the framework of choice for desktop and web app development.

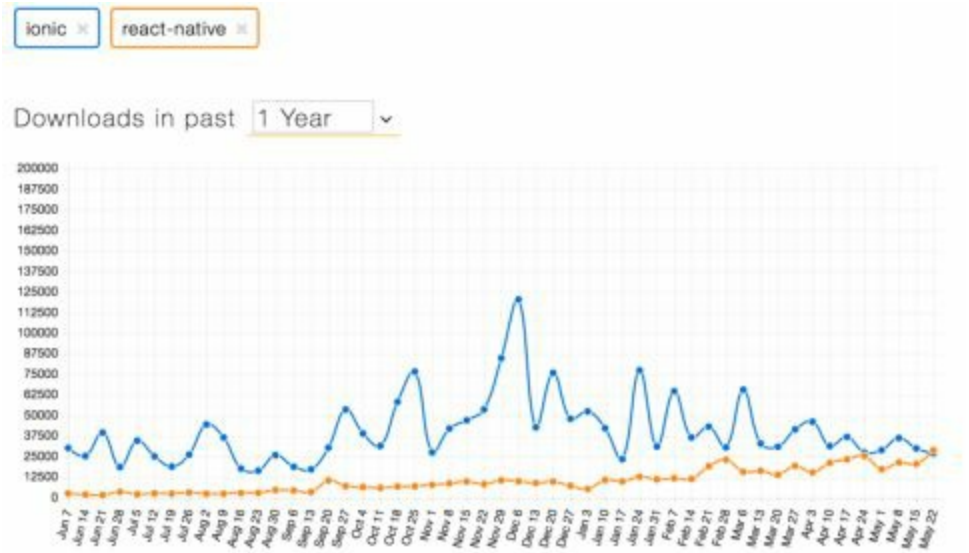
Mobile App Framework Showdown: React Native vs. Ionic

When Reactive Native came out, it was predicted that it would take over and charts the course of mobile development with web technologies since it can be used to develop cross-platform apps natively. So is 2017 the year of React Native? There might be some expectations to that through this graph:



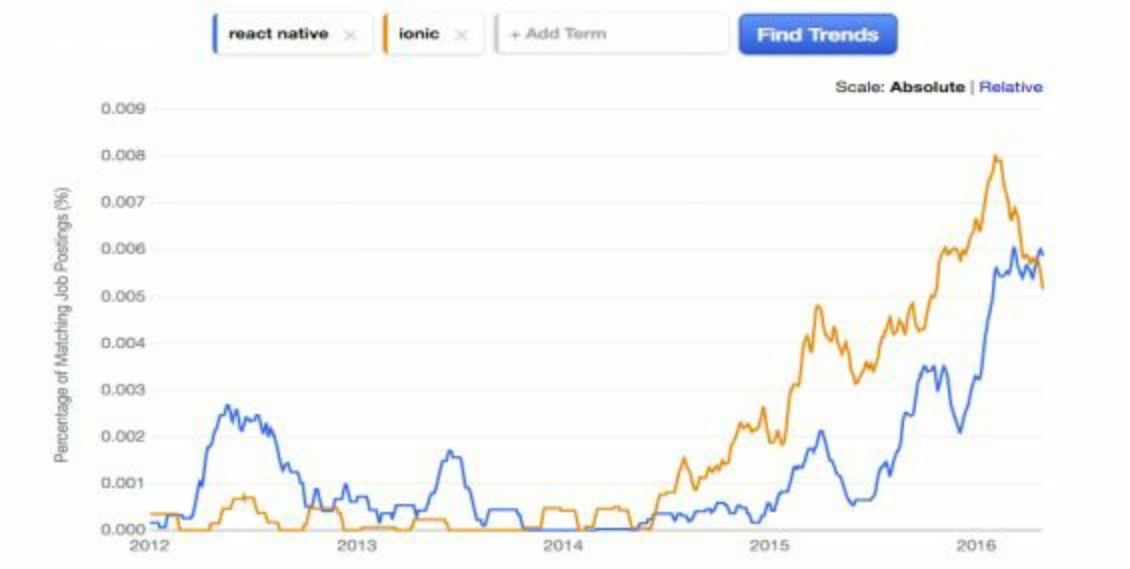
At this moment, React Native has not yet released any updated software as it is still at version 0.26. Whether or not it will become mainstream in 2017 depends largely on developer's patience to deal with app- development changes that needs refactoring the whole codebase together with other issues.

Therefore, let's take a look at the approximate npm downloads of React Native vs Ionic:



From the graph, it seems that React Native is on track to overtake ionic as the framework of choice for cross-platform mobile development using web technologies.

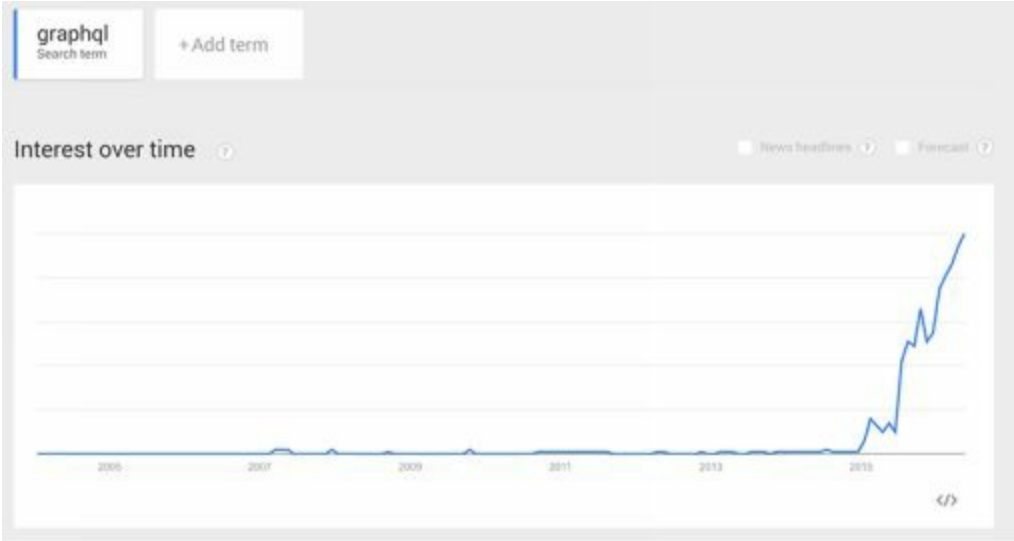
In terms of the job market, React Native is also becoming more in-demand than Ionic:



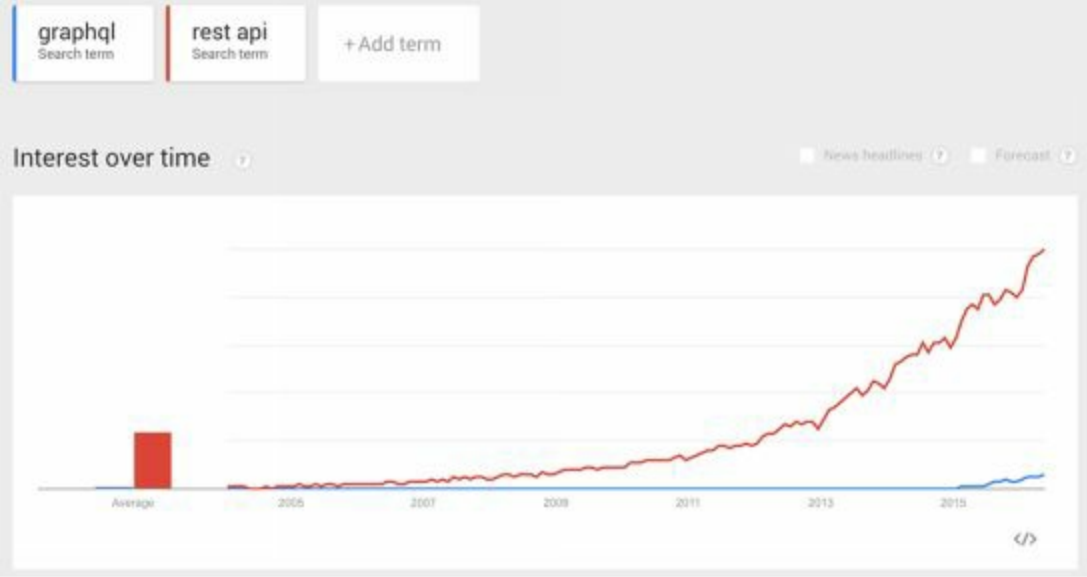
A quick search on job postings, show that at least 75 job postings are looking for developers with ionic skills whereas 65 job postings are looking for developers with React skills. This shows that there is an increasing surge of the familiarity with React Native that will most likely be a huge push to any up and coming developers going forward 2017.

The Future of Web API: GraphQL vs REST

Soon after Facebook mentioned that they were open-sourcing GraphQL, plenty of JavaScript developers jumped onto the hype train earlier on in 2016, especially since Facebook turned into the Apple of JavaScript when it came to open source projects.



GraphQL was aiming to sort of replace REST APIs, but considering how ubiquitous REST APIs are, this is an unlikely scenario:



GraphQL would do well with REST APIs, but it will definitely not replace REST as some developers thought it would have. Even more to do with this is

that GraphQL is relatively new and linked with Facebook Relay, so there aren't any learning resources or best practices for programmers and developers to pick the skill up elsewhere. With that said, 2017 is still too early for GraphQL to be going mainstream despite the strong hype it has now.

JavaScript Trends Conclusion

In conclusion, the community of JavaScript moves and changes at a fast pace, keeping up with any upcoming trends and in line with mainstream technology which means more support, relevance, and resources. Based on the trends discussed in this chapter, JavaScript developers should equip themselves with Redux & React and also with other familiar functional programming such as TypeScript. Not only that, having experience with React Native and Electron will definitely give upcoming programmers a lot more push and credibility in a job search.

Conclusion

Thank for making it through to the end of *JavaScript: Tips and Tricks to Programming Code with JavaScript*. Let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever it may be.

The next step is to use the coding skills that you have learned and apply it to your own projects. Learning takes time, but you should be patient because just like when you learn anything new, you are going to make mistakes.

Go over the lessons in this book again and reread them. Learning how to write code is going to be confusing until you get the hang of it because of all the different conditions and exceptions that you are going to learn.

Finally, if you found this book useful in anyway, a review on Amazon is always appreciated!

About the Author

Charlie Masterson is a computer programmer and instructor who have developed several applications and computer programs.

As a computer science student, he got interested in programming early but got frustrated learning the highly complex subject matter.

Charlie wanted a teaching method that he could easily learn from and develop his programming skills. He soon discovered a teaching series that made him learn faster and better.

Applying the same approach, Charlie successfully learned different programming languages and is now teaching the subject matter through writing books.
With the books that he writes on computer programming, he hopes to provide great value and help readers interested to learn computer-related topics.