Javascript: Object Oriented Programming

Build sophisticated web applications by mastering the art of Object-Oriented Javascript

A course in three modules

BIRMINGHAM - MUMBAI

Javascript: Object Oriented Programming

Published on: August 2016

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78712-359-5

www.packtpub.com

Preface

It may seem that everything that needs to be written about JavaScript has already been written. However, JavaScript is changing rapidly. ECMAScript 6 has the potential to transform the language and how we code in it. Node.js has changed the way we write servers in JavaScript. Newer ideas such as React and Flux will drive the next iteration of the language. If you are already an experienced JavaScript developer, you will realize that modern JavaScript is vastly different from the language that most people have known. Tools are more powerful and slowly becoming an integral part of the development workflow.

Object-oriented programming, also known as OOP, is a required skill in absolutely any modern software developer job. It makes a lot of sense because object-oriented programming allows you to maximize code reuse and minimize the maintenance costs. However, learning object-oriented programming is challenging because it includes too many abstract concepts that require real-life examples to make it easy to understand.

JavaScript has moved from being mostly used in browsers for client-side technologies to being used even on server side.

This course will help you change some common coding practices and empower you by giving you the tools you need for more efficient development. We'll look at implementing these principles to build sophisticated web applications.

We hope that you enjoy this course as much as we enjoyed developing it.

What this learning path covers

Module 1, Mastering Javascript, provides you with a detailed overview of the language's fundamentals and some of the modern tools and libraries, such as jQuery, Underscore.js, and Jasmine.

Module 2 Learning Object-Oriented Programming, helps you to learn how to capture objects from real-world elements and create object-oriented code that represents them.

Module 3, Object-Oriented JavaScript - Second Edition, This module doesn't assume any prior knowledge of JavaScript and works from the ground up to give you a thorough understanding of the language What you need for this learning path. Exercises at the end of the chapters help you assess your understanding.

What you need for this learning path

- A computer with Windows 7 or higher, Linux, or Mac OS X installed.
- The latest version of the Google Chrome or Mozilla Firefox browser.
- A text editor of your choice. Sublime Text, vi, Atom, or Notepad++ would be ideal. The choice is entirely yours
- A computer with at least an Intel Core i3 CPU or equivalent with 4 GB RAM, running on Windows 7 or a higher version, Mac OS X Mountain Lion or a higher version, or any Linux version that is capable of running Python 3.4, and a browser with JavaScript support.
- You will need Python 3.4.3 installed on your computer. You can work with your favorite editor or use any Python IDE that is compatible with the mentioned Python version.
- In order to work with the C# examples, you will need Visual Studio 2015 or 2013.
- You can use the free Express editions to run all the examples. If you aren't working on Windows, you can use Xamarin Studio 5.5 or higher.
- In order to work with the JavaScript examples, you will need web browsers such as Chrome 40.x or higher, Firefox 37.x or higher, Safari 8.x or higher, Internet Explorer 10 or higher that provides a JavaScript console
- You need a modern browser—Google Chrome or Firefox are recommended—and an optional Node.js setup. The latest version of Firefox comes with web developer tools, but Firebug is highly recommended. To edit JavaScript you can use any text editor of your choice.

Who this learning path is for

JavaScript developers looking to enhance their web developments skills by learning object-oriented programming.

Reader feedback

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

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

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

Customer support

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

Downloading the example code

You can download the example code files for this course from your account at `http://www.packtpub.com`. If you purchased this course elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the course in the **Search** box.
5. Select the course for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this course from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the course's webpage at the Packt Publishing website. This page can be accessed by entering the course's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the course is also hosted on GitHub at `https://github.com/PacktPublishing/Object-oriented-programming-for-JavaScript-developers`. We also have other code bundles from our rich catalog of books, videos, and courses available at `https://github.com/PacktPublishing/`. Check them out!

Errata

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

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

Contents

Course Module 1: Mastering JavaScript

Course Module 2: Learning Object-Oriented Programming

Course Module 3: Object-Oriented JavaScript - Second Edition

Module 1

Mastering JavaScript

Explore and master modern JavaScript techniques in order
to build large-scale web applications

1
JavaScript Primer

It is always difficult to pen the first few words, especially on a subject like JavaScript. This difficulty arises primarily because so many things have been said about this language. JavaScript has been the *Language of the Web*—lingua franca, if you will, since the earliest days of the Netscape Navigator. JavaScript went from a tool of the amateur to the weapon of the connoisseur in a shockingly short period of time.

JavaScript is the most popular language on the web and open source ecosystem. `http://githut.info/` charts the number of active repositories and overall popularity of the language on GitHub for the last few years. JavaScript's popularity and importance can be attributed to its association with the browser. Google's V8 and Mozilla's SpiderMonkey are extremely optimized JavaScript engines that power Google Chrome and Mozilla Firefox browsers, respectively.

Although web browsers are the most widely used platforms for JavaScript, modern databases such as MongoDB and CouchDB use JavaScript as their scripting and query language. JavaScript has become an important platform outside browsers as well. Projects such as **Node.js** and **io.js** provide powerful platforms to develop scalable server environments using JavaScript. Several interesting projects are pushing the language capabilities to its limits, for example, **Emscripten** (`http://kripken.github.io/emscripten-site/`) is a **Low-Level Virtual Machine** (**LLVM**)-based project that compiles C and C++ into highly optimizable JavaScript in an **asm.js** format. This allows you to run C and C++ on the web at near native speed.

JavaScript is built around solid foundations regarding, for example, functions, dynamic objects, loose typing, prototypal inheritance, and a powerful object literal notation.

While JavaScript is built on sound design principles, unfortunately, the language had to evolve along with the browser. Web browsers are notorious in the way they support various features and standards. JavaScript tried to accommodate all the whims of the browsers and ended up making some very bad design decisions. These bad parts (the term made famous by Douglas Crockford) overshadowed the good parts of the language for most people. Programmers wrote bad code, other programmers had nightmares trying to debug that bad code, and the language eventually got a bad reputation. Unfortunately, JavaScript is one of the most misunderstood programming languages (`http://javascript.crockford.com/javascript.html`).

Another criticism leveled at JavaScript is that it lets you get things done without you being an expert in the language. I have seen programmers write exceptionally bad JavaScript code just because they wanted to get the things done quickly and JavaScript allowed them to do just this. I have spent hours debugging very bad quality JavaScript written by someone who clearly was not a programmer. However, the language is a tool and cannot be blamed for sloppy programming. Like all crafts, programming demands extreme dedication and discipline.

A little bit of history

In 1993, the **Mosaic** browser of **National Center for Supercomputing Applications** (**NCSA**) was one of the first popular web browsers. A year later, Netscape Communications created the proprietary web browser, **Netscape Navigator**. Several original Mosaic authors worked on Navigator.

In 1995, Netscape Communications hired Brendan Eich with the promise of letting him implement **Scheme** (a Lisp dialect) in the browser. Before this happened, Netscape got in touch with Sun Microsystems (now Oracle) to include Java in the Navigator browser.

Due to the popularity and easy programming of Java, Netscape decided that a scripting language had to have a syntax similar to that of Java. This ruled out adopting existing languages such as Python, **Tool Command Language** (**TCL**), or Scheme. Eich wrote the initial prototype in just 10 days (`http://www.computer.org/csdl/mags/co/2012/02/mco2012020007.pdf`), in May 1995. JavaScript's first code name was **Mocha**, coined by Marc Andreessen. Netscape later changed it to **LiveScript**, for trademark reasons. In early December 1995, Sun licensed the trademark Java to Netscape. The language was renamed to its final name, JavaScript.

How to use this book

This book is not going to help if you are looking to get things done quickly. This book is going to focus on the correct ways to code in JavaScript. We are going to spend a lot of time understanding how to avoid the bad parts of the language and build reliable and readable code in JavaScript. We will skirt away from sloppy features of the language just to make sure that you are not getting used to them—if you have already learned to code using these habits, this book will try to nudge you away from this. There will be a lot of focus on the correct style and tools to make your code better.

Most of the concepts in this book are going to be examples and patterns from real-world problems. I will insist that you code each of the snippets to make sure that your understanding of the concept is getting programmed into your muscle memory. Trust me on this, there is no better way to learn programming than writing a lot of code.

Typically, you will need to create an HTML page to run an embedded JavaScript code as follows:

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript" src="script.js"></script>
  <script type="text/javascript">
    var x = "Hello World";
    console.log(x);
  </script>
</head>
<body>
</body>
</html>
```

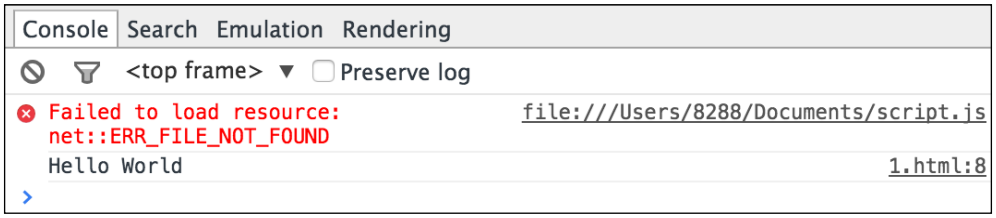
This sample code shows two ways in which JavaScript is embedded into the HTML page. First, the `<script>` tag in `<head>` imports JavaScript, while the second `<script>` tag is used to embed inline JavaScript.

Downloading the example code

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

You can save this HTML page locally and open it in a browser. On Firefox, you can open the **Developer** console (Firefox menu | **Developer** | **Web Console**) and you can see the **"Hello World"** text on the **Console** tab. Based on your OS and browser version, the screen may look different:

You can run the page and inspect it using Chrome's **Developer Tool**:



A very interesting thing to notice here is that there is an error displayed on the console regarding the missing `.js` file that we are trying to import using the following line of code:

```
<script type="text/javascript" src="script.js"></script>
```

Using browser developer consoles or an extension such as **Firebug** can be very useful in debugging error conditions in the code. We will discuss in detail the debugging techniques in later chapters.

Creating such HTML scaffolds can be tedious for every exercise in this book. Instead, we want to use a **Read-Eval-Print-Loop** (**REPL**) for JavaScript. Unlike Python, JavaScript does not come packaged with an REPL. We can use Node.js as an REPL. If you have Node.js installed on your machine, you can just type `node` on the command line and start experimenting with it. You will observe that Node REPL errors are not very elegantly displayed.

Let's see the following example:

```
EN-VedA:~$ node
>function greeter(){
  x="World"l
SyntaxError: Unexpected identifier
  at Object.exports.createScript (vm.js:44:10)
  at REPLServer.defaultEval (repl.js:117:23)
  at bound (domain.js:254:14)
  …
```

After this error, you will have to restart. Still, it can help you try out small fragments of code a lot faster.

Another tool that I personally use a lot is **JS Bin** (`http://jsbin.com/`). JS Bin provides you with a great set of tools to test JavaScript, such as syntax highlighting and runtime error detection. The following is a screenshot of JS Bin:

Based on your preference, you can pick the tool that makes it easier to try out the code samples. Regardless of which tool you use, make sure that you type out every exercise in this book.

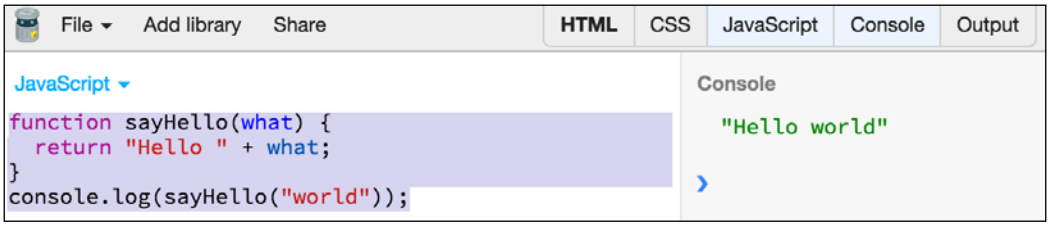
Hello World

No programming language should be published without a customary Hello World program—why should this book be any different?

Type (don't copy and paste) the following code in JS Bin:

```
function sayHello(what) {
  return "Hello " + what;
}
console.log(sayHello("world"));
```

Your screen should look something as follows:



An overview of JavaScript

In a nutshell, JavaScript is a prototype-based scripting language with dynamic typing and first-class function support. JavaScript borrows most of its syntax from Java, but is also influenced by Awk, Perl, and Python. JavaScript is case-sensitive and white space-agnostic.

Comments

JavaScript allows single line or multiple line comments. The syntax is similar to C or Java:

```
// a one line comment

/* this is a longer,
   multi-line comment
 */

/* You can't /* nest comments */ SyntaxError */
```