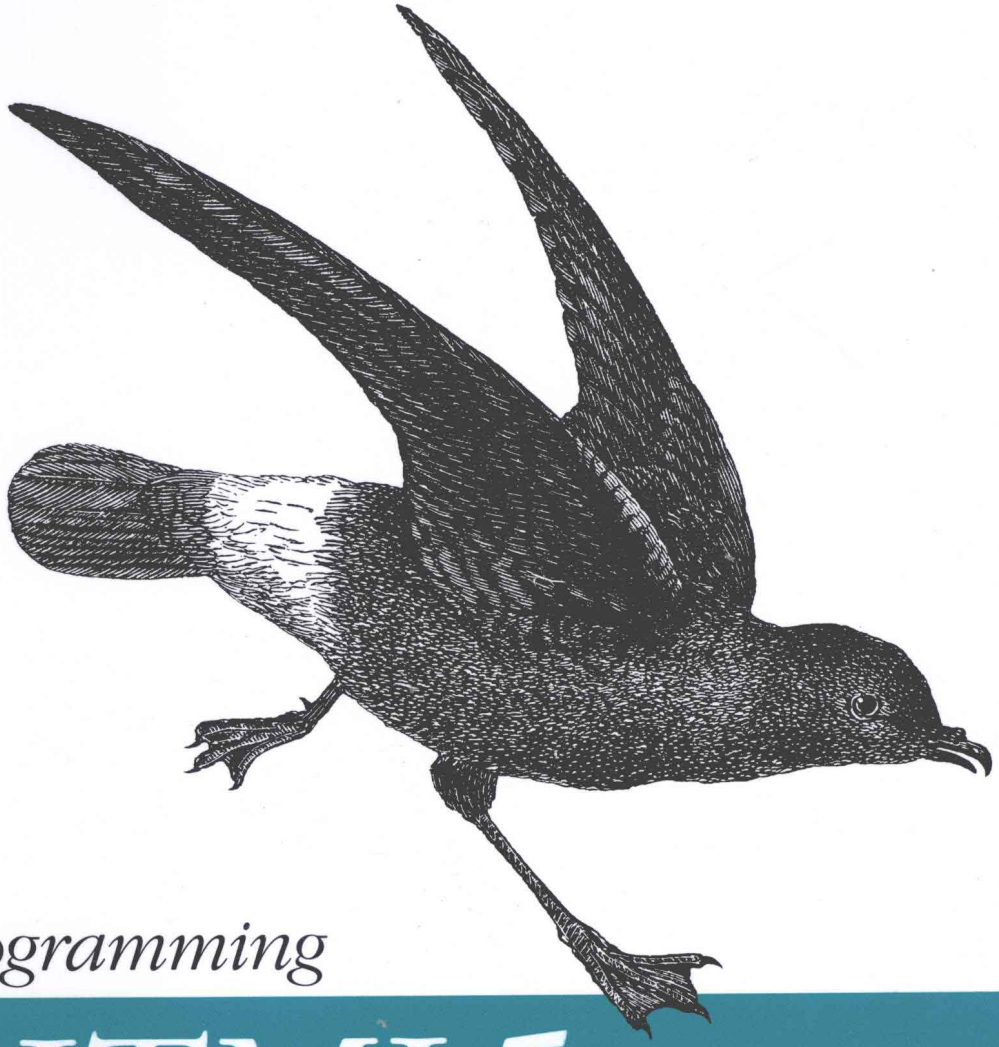
HTML5应用编程（影印版）



Programming

HTML5 Applications

REILLY®
大學 出版社

Zachary Kessin 著

HTML5应用编程（影印版）

Programming HTML5 Applications

Zachary Kessin 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

HTML5应用编程：英文/（美）凯西恩（Kessin, Z.）著. —影印本. —南京：东南大学出版社，2012.6

书名原文：Programming HTML5 Applications

ISBN 978-7-5641-3413-6

I. ①H… II. ①凯… III. ①超文本标记语言－程序设计－英文 IV. ①TP312

中国版本图书馆CIP数据核字（2012）第065807号

江苏省版权局著作权合同登记

图字：10-2011-421号

HTML5应用编程（影印版）

出版发行：东南大学出版社

地　　址：南京四牌楼2号　　邮编：210096

出 版 人：江建中

网　　址：http://www.seupress.com

电子邮件：press@seupress.com

印　　刷：扬中市印刷有限公司

开　　本：787毫米×980毫米　16开本

印　　张：8.75

字　　数：171千字

版　　次：2012年6月第1版

印　　次：2012年6月第1次印刷

书　　号：ISBN 978-7-5641-3413-6

定　　价：32.00元（册）

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

Preface

This book reflects the evolution of the Web. Less and less can programming be treated as a distinct activity shoehorned into web pages through scripts. Instead, HTML and JavaScript are now intertwined in producing an enchanting user experience. With this book, you can master the latest in this evolution.

How This Book Is Organized

The elements of this book are as follows:

Chapter 1, *The Web As Application Platform*
: Introduces the reasons for programming on the new HTML5 platforms and what they offer to the JavaScript programmer

Chapter 2, *The Power of JavaScript*
: Explains some powerful features of JavaScript you may not already know, and why you need to use them to exploit the HTML5 features and associated libraries covered in this book

Chapter 3, *Testing JavaScript Applications*
: Shows how to create and use tests in the unique environment provided by JavaScript and browsers

Chapter 4, *Local Storage*
: Describes the `localStorage` and `sessionStorage` objects that permit simple data caching in the browser

Chapter 5, *IndexedDB*
: Shows the more powerful NoSQL database that supports local storage

Chapter 6, *Files*
: Describes how to read and upload files from the user's system

Chapter 7, *Taking It Offline*
: Describes the steps you must go through to permit a user to use your application when the device is disconnected from the Internet

Chapter 8, *Splitting Up Work Through Web Workers*
: Shows the multithreading capabilities of HTML5 and JavaScript

Chapter 9, *Web Sockets*
: Shows how to transfer data between the browser and server more efficiently by using web sockets

Chapter 10, *New Tags*
: Summarizes tags introduced in HTML5 that are of particular interest to the web programmer

Appendix, *JavaScript Tools You Should Know*
: Describes tools used in the book, and others that can make coding easier and more accurate

Conventions Used in This Book

The following typographical conventions are used in this book:

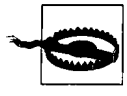
Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords

`Constant width bold`
: Shows commands or other text that should be typed literally by the user

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming HTML5 Applications* by Zachary Kessin (O'Reilly). Copyright 2012 Zachary Kessin, 978-1-449-39908-5."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search more than 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://shop.oreilly.com/product/0636920015116.do

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

A book is a team effort, and I could not have written this book without a great team behind me. First of all, I must thank Simon St. Laurent for giving me the chance to write this book and supporting me through the process of putting it together. I must also thank Andy Oram for his editorial prowess and ability to make the book better. Also, thank you to my technical reviewers, Shelley Powers and Dionysios Synodinos, for great feedback.

I must also thank the Israeli developer community for existing: my former coworkers at Mytopia, who supported me in this project for more than a year, and the gang at Sayeret Lambda, which has become the place in Tel Aviv to talk about programming.

Finally, I would like to thank my wife, Devora, for all her support in this project. I could not have done it without you.

Table of Contents

CHAPTER 1

The Web As Application Platform

HTML5 makes the Web a first-class environment for creating real applications. It reinforces JavaScript's existing tool set with key extensions to the browser APIs that make it easier to create applications that feel (and can be) complete in themselves, not just views on some distant server process.

The Web began as a way to share files, stored on a web server, that changed only occasionally. Developers quickly figured out how to generate those files on the fly, taking the first big step toward building applications. The next big step was adding interactivity in the browser client. JavaScript and the Document Object Model (DOM) let developers create Dynamic HTML, as the "browser wars" raged and then suddenly stopped. After a few years, Ajax brought these techniques back into style, adding some tools to let pages communicate with the server in smaller chunks.

HTML5 builds on these 20 years of development, and fills in some critical gaps. On the surface, many of HTML5's changes add support for features (especially multimedia and graphics) that had previously required plug-ins, but underneath, it gives JavaScript programmers the tools they need to create standalone (or at least more loosely tethered) applications using HTML for structure, CSS for presentation, and JavaScript for logic and behavior.

Adding Power to Web Applications

HTML5 raises the bar for web applications. While it still has to work under security constraints, it finally provides tools that desktop developers have expected for years:

Local data storage
: It can store up to 5 MB of data, referenced with a key-value system.

Databases
: Originally a SQLite-based API, the tide seems to have shifted to IndexedDB, a NoSQL system that is natively JavaScript.

Files
: While applications still can't freely access the filesystem (for obvious security reasons), they can now work with files the user specifies and are starting to be able to create files as well.

Taking it offline
: When a laptop or phone is in airplane mode, web applications are not able to communicate with the server. Manifest files help developers work around that by caching files for later use.

Web Workers
: Threads and forks have always been problematic, but JavaScript simply didn't offer them. Web Workers provide a way to put application processes into separate spaces where they can work without blocking other code.

Web sockets
: Hypertext Transfer Protocol (HTTP) has been the foundation of the Web, despite a few updates over time. Web sockets transform the request-response approach to create much more flexible communication systems.

There's much more, of course—from geolocation to audio and video to Canvas graphics to a wide variety of minor new tags—but these provide the foundations for building industrial-strength applications in HTML5.

Developing Web Applications

In the old days, a complex web application might be a catalog, which would be static pages derived from a database, or a JavaScript loan calculator. But no one would have dreamed of doing complex applications in JavaScript. Those required Java or maybe a dedicated client/server application written in C or C++. Indeed, in the days before the DOM and Ajax, developing complex applications in JavaScript would have been pretty much impossible. However, Ajax introduced the ability to interact with the server without reloading the page, and the DOM allowed the programmer to change HTML on the fly.

In 2007, Google introduced Gears, a browser extension that gave the developer a lot more power than had been there before. Gears allowed the browser to work offline, to enable users to store more data in the browser and have a worker pool to offload long-running tasks. Gears has since been discontinued, as most of its features have migrated into HTML5 in modified forms.

The modern Web features a full range of sites, from things that are still effectively old-style collections of documents, like Wikipedia, to sites that offer interactions with other people, such as Facebook, YouTube, and eBay, to things that can serve as replacements for desktop applications, such as Gmail and Google Docs. Many formerly standalone applications, such as mail clients, have become part and parcel of the web experience.

In the modern Web, the line between applications and pages has blurred. The difference at this point is only in the intent of the site.

Running an application in the browser has some major advantages for both the user and the developer. For the user, there is no commitment to the application: you try it out, and if you don't like it, you can move on to the next page with nothing left behind to clutter up your disk. Trying new applications is also reasonably safe, in that they run in a sandboxed environment. New versions of the application are automatically downloaded to the browser when the developer updates the code. Web applications rarely have version numbers, at least public ones.

For the developer, the case is even stronger. First of all, the things that are an advantage to the users are also good for the developers. There is no installation program to write, and new versions can automatically be sent to the users, making small, incremental updates not only possible but practical. However, there are other bonuses as well.

The Web is cross-platform. It is possible to write a web page that will work on Windows XP, Windows Vista, Windows 7, Mac OS X, Linux, the iPhone/iPad, and Android. Doing that with a conventional development tool would be a monumental task. But with the Web and some forethought it almost comes for free. A web application built on standards with a library like jQuery will be able to run on major browsers on all those platforms and a few others. While at one point Sun hoped that its Java applets would define the Web as a platform, JavaScript has turned out to become the default web platform.

You can even run web applications on mobile devices, at least the ones that today are called smartphones. With a wrapper like PhoneGap, you can create an HTML5 app and package it for sale in the App Store, the Android Market, and more. You might create an application that interacts heavily with a web server, or you might create a completely self-contained application. Both options are available.

The real place that the Web, prior to HTML5, traditionally falls short is that a web application, running on a computer with gigabytes of memory and disk space, acts almost like it is running on an old VT320 terminal. All data storage must be done on a server, all files must be loaded from the server, and every interaction pretty much requires a round-trip to the server. This can cause the user experience to feel slow, especially if the server is far away from the user. If every time the user wishes to look up something there is a minimum response time of 400 milliseconds before any actions can be taken, the application will feel slow. From my office in Tel Aviv to a server in California, the round-trip time for an ICMP ping is about 250 ms. Any action on the server would be extra and slow that down even more. Mobile device communications can, of course, be even slower.

JavaScript's Triumph

Though JavaScript has been a key component of web development since it first appeared in 1995, it spent a decade or so with a bad reputation. It offered weak performance, was saddled with a quirky syntax that led to mysterious bugs, and suffered from its dependence on the DOM. Browsers kept it locked in a "sandbox," easing users' security concerns but making it very difficult for developers to provide features that seemed trivial in more traditional desktop application development.

Scripting culture created its own problems. Although providing a very low barrier to entry is a good thing, it does come with costs. One of those costs is that such a language often allows inexperienced programmers to do some very ill-advised things. Beginning programmers could easily find JavaScript examples on the Web, cut and paste them, change a few things, and have something that mostly worked. Unfortunately, maintaining such code becomes more and more difficult over time.

With the Ajax revival, developers took a new look at JavaScript. Some have worked on improving the engines interpreting and running JavaScript code, leading to substantial speed improvements. Others focused on the language itself, realizing that it had some very nice features, and consequently developing best practices like those outlined in *JavaScript: The Good Parts* by Douglas Crockford (O'Reilly, 2008).

Beyond the core language, developers built tools that made debugging JavaScript much easier. Although Venkman, an early debugger, had appeared in 1998, the 2006 release of Firebug became the gold standard of JavaScript debuggers. It allows the developer to track Ajax calls, view the state of the DOM and CSS, single-step through code, and much more. Browsers built on WebKit, notably Apple's Safari and Google Chrome, offer similar functionality built in, and Opera Dragonfly provides support for Opera. Even developers working in the confined spaces of mobile devices can now get Firebug-like debugging with weinre (WEb INspector REmote).

The final key component in this massive recent investment in JavaScript was libraries. Developers still might not understand all the code they were using, but organizing that code into readily upgradeable and sometimes even interchangeable libraries simplified code management.

jQuery

If anything can be described as the gold standard of JavaScript libraries, it would have to be John Resig's jQuery library, which forms a wrapper around the DOM and other JavaScript objects such as the `XMLHttpRequest` object, and makes doing all sorts of things in JavaScript a lot easier and a lot more fun. In many ways, jQuery is the essential JavaScript library that every JavaScript programmer should know.

To learn jQuery, see the jQuery website (*http://jquery.org*) or a number of good books on the subject, such as *Head First jQuery* by Ryan Benedetti and Ronan Cranley or *jQuery Cookbook* by Cody Lindley, both published by O'Reilly. Many examples in this book are written using jQuery.

ExtJS

Whereas jQuery forms a wrapper around the DOM, Sencha's (*http://sencha.com*) ExtJS tries to abstract it away as much as possible. ExtJS features a rich widget set that can live in a web page and provide many of the widgets, such as trees, grids, forms, buttons, and so on, that desktop developers are familar with. The entire system is very well thought out, fits together well, and makes developing many kinds of applications a joy. Although the ExtJS library takes up a lot of space, the expenditure is worthwhile for some kinds of application development.

One nice feature of ExtJS is that many of its objects know how to save their state. So if a user takes a grid and reorganizes the columns, the state can be saved so that the same order appears the next time the user views that grid. "Using localStorage in ExtJS" on page 53 will show how to use the HTML5 `localStorage` facility with this feature.

Google Web Toolkit, etc.

Tools such as GWT allow the programmer to write Java code, which is then compiled down to JavaScript and can be run on the browser.

CHAPTER 2

The Power of JavaScript

Although JavaScript is not a difficult language to program, it can be challenging to rise to the level of a true expert. There are several key factors to becoming a skilled JavaScript programmer. The techniques in this chapter will appear repeatedly in the libraries and programming practices taught in the rest of this book, so you should familiarize yourself with these techniques before continuing with those chapters.

There are a number of excellent tools for JavaScript programming, some of them listed in the Appendix. These tools can provide you with a lot of assistance. Specifically, JSLint will catch a large number of errors that a programmer might miss. Sites such as StackOverflow (*http://stackoverflow.com/*) and O'Reilly Answers (*http://answers.oreilly.com*) will be a good source of other tools.

This chapter is not a full introduction to the power of JavaScript. O'Reilly publishes a number of excellent books on Javscript, including:

- *JavaScript, The Good Parts* by Douglas Crockford
- *JavaScript: The Definitive Guide* by David Flanagan
- *High Performance JavaScript* by Nicholas C. Zakas
- *JavaScript Patterns* by Stoyan Stefanov

Nonblocking I/O and Callbacks

The first key to JavaScript, after learning the language itself, is to understand event-driven programming. In the environment where JavaScript runs, operations tend to be asynchronous, which is to say that they are set up in one place and will execute later after some external event happens.

This can represent a major change from the way I/O happens in traditional languages. Take Example 2-1 as a typical case of I/O in a traditional language, in this case PHP. The line `$db->getAll($query);` requires the database to access the disk, and therefore will take orders of magnitude more time to run than the rest of the function. While the program is waiting for the server to execute, the query statement is blocked and the