# Contents at a Glance

Introduction ........................................................................................................ 1

## Part I  Background

1 Why HTML5? ..................................................................................................... 7
2 Important Concepts for HTML5 ........................................................................ 15

## Part II  New HTML Elements

3 Getting Started with HTML5: Semantic Tags, Forms, and Drag and Drop ...... 21
4 Rich Media Tags: Video and Audio ............................................................... 69

## Part III  Canvas

5 2D Canvas ...................................................................................................... 103
6 Making Canvas Interactive and Stateful ....................................................... 205
7 Canvas Performance, Tips, and Peculiarities ................................................ 233
8 The Future of Canvas and 3D Canvas ........................................................... 263

## Part IV  HTML5’s JavaScript APIs

9 Geolocation API .......................................................................................... 277
10 HTML5 Storage Options ............................................................................. 301
11 Messaging and Web Workers ....................................................................... 333
12 Network Communication: WebSockets and XMLHttpRequest Level 2 ........ 353
13 Microdata, Other Small Things, and Beyond HTML5 .................................. 365

Index ................................................................................................................. 381
Table of Contents

Introduction .............................................................. 1
Who Should Read This Book? .............................................................. 2
HTML5 and Related Technologies ........................................................ 2
Software Requirements ........................................................................ 3
Code Examples ................................................................................... 3
How This Book Is Organized ................................................................. 3
Part I: Background ...................................................................... 4
Part II: New HTML Elements ................................................................ 4
Part III: Canvas ........................................................................... 4
Part IV: HTML5’s JavaScript APIs .................................................. 4
Links and Real-World Examples ........................................................... 5

1 Why HTML5? ........................................................................ 7
How Did We End Up Here? ................................................................. 7
The Web Takes Off ..................................................................... 9
The Rise of the Browser Plug-In .................................................... 9
Web 2.0 ................................................................................... 10
HTML5 ............................................................................................ 11
What Exactly Is HTML5? ........................................................... 11
The Importance of HTML5 ........................................................... 12
Are Plug-ins Dead? ............................................................................ 14
Summary ......................................................................................... 14

2 Important Concepts for HTML5 .................................................. 15
The Goals of HTML5 ......................................................................... 15
Improving the Native Web ................................................................ 15
More Done with Less Code .............................................................. 16
The Semantic Web ........................................................................ 16
Requisites for HTML5 Development ................................................... 16
Modern Browser Developer Tools ......................................................... 16
HTML5 Fallbacks: Shims, Shivs, and Polyfills ........................................ 17
Feature Support and Detection ......................................................... 18
Summary ......................................................................................... 19
3 Getting Started with HTML5: Semantic Tags, Forms, and Drag and Drop

Ensuring Backward Compatibility with the New HTML Tags ........................................... 21
  The HTML5 Shim ........................................................................................................ 22
  HTML5 Boilerplate ...................................................................................................... 23
Starting from the Top ........................................................................................................ 23
  The Doctype ................................................................................................................. 23
  Meta Character Encoding ............................................................................................. 24
  HTML5 Syntax and Validation ....................................................................................... 24
  How You Should Write Your HTML ........................................................................... 27
  Housekeeping .............................................................................................................. 29
HTML5 Semantic Elements and Other

Presentational Tags .......................................................................................................... 31
  HTML5 Semantic Tags ................................................................................................... 31
  Document Outlines in HTML5 ....................................................................................... 33
  Minor Semantic and Presentational HTML5 Tags ...................................................... 35
  Visual HTML5 Tags: <meter> and <progress> .......................................................... 37
HTML5 Forms .................................................................................................................. 40
  Form Input Types ......................................................................................................... 40
  New Form Input Attributes and Elements .................................................................... 47
Drag and Drop in HTML5 .................................................................................................. 53
  The Basics ...................................................................................................................... 53
  Drag Data and Effects .................................................................................................... 55
  Drag Events .................................................................................................................... 58
Summary ............................................................................................................................ 67

4 Rich Media Tags: Video and Audio .................................................................................. 69

The Video Element ........................................................................................................... 71
  Video Formats and Support ........................................................................................... 71
  Using the Video Element ............................................................................................... 75
The Audio Element .......................................................................................................... 77
  Audio Formats and Support ........................................................................................... 77
  Using the Audio Element ............................................................................................... 79
Encoding Your Media ......................................................................................................... 80
  Conversion Tools .......................................................................................................... 81
Supporting Older Browsers ............................................................................................... 81
Video and Audio Attributes .............................................................................................. 82
  Video-Only Attributes .................................................................................................... 82
  Attributes Shared Between Audio and Video ............................................................ 85
JavaScript API .................................................................................................................. 90
  The readyState and Starting at a Specified Time ....................................................... 92
  The playbackRate and Time Control ............................................................................ 93
### Loading Videos Sequentially ...................................................... 94
### Custom Controls ...................................................................... 95
### Advantages and Disadvantages of the HTML5 Media Tags Versus Flash ........................................... 95
    - Ease of Use and Extensibility ........................................ 95
    - Platform Support .......................................................... 96
    - Feature Support ............................................................ 96
    - Media Protection ......................................................... 97
### Future Developments ............................................................... 97
    - WebRTC ......................................................................... 97
    - WebVTT and the `<track>` Tag ...................................... 98
### The Full-Screen API ............................................................... 100
### The Web Audio API .............................................................. 100
### The Embed Element ............................................................. 101
### Summary .............................................................................. 101

## 5 2D Canvas

### Browser Support .................................................................... 104
    - Internet Explorer ......................................................... 105
    - Testing for Support ...................................................... 105
### What Canvas Can and Cannot Do ........................................... 106
    - A Comparison with SVG ................................................ 107
    - Where Canvas Shines .................................................... 109
### When to Not Use Canvas ...................................................... 109
    - Don’t Use Canvas for General UI .................................... 111
### Getting Started with Canvas .................................................. 111
    - An HTML Page with a Canvas ...................................... 113
### Canvas Attributes .................................................................. 115
    - `toDataURL` .................................................................... 116
### Context Methods and State .................................................... 117
    - A Quick Look at the Properties and State Available .......... 117
### Understanding Drawing, Starting with Rectangles ................. 118
    - `fillStyle` and `strokeStyle` .......................................... 119
### Paths .................................................................................... 120
### Understanding the Canvas Coordinate System ....................... 122
### Line Styles .......................................................................... 125
### Curves ................................................................................ 127
### Ellipses ............................................................................... 133
### `isPointInPath` ................................................................... 135
### Path Filling—The Winding Number Rule ................................. 135
Rectangles ............................................................................... 200
Paths ...................................................................................... 201
Image Drawing ........................................................................ 202
Transformation ................................................................. 202
Compositing ........................................................................... 202
Text ....................................................................................... 202
Image Data ............................................................................. 203
Summary ........................................................................................ 203

6 Making Canvas Interactive and Stateful 205

Canvas Coordinates—Mouse and Touch ............................................ 205
    getBoundingClientRect ........................................................... 206
Computing Element Offset ....................................................... 207
Canvas Animation ........................................................................... 210
    Letting the Browser Take Control with
        requestAnimationFrame .................................................. 210
Animation and Timing ............................................................ 212
Canvas Interactivity Example: Making and
Moving Shapes .............................................................................. 215
    Getting Started ........................................................................ 216
The Shapes We Draw ............................................................... 217
Keeping Track of Canvas State .................................................. 218
Mouse and Touch Events ......................................................... 219
Getting Input Coordinates........................................................ 222
Drawing .................................................................................. 223
Complete Canvas Interactivity Example .................................... 225
Summary ........................................................................................ 231

7 Canvas Performance, Tips, and Peculiarities 233

Canvas Peculiarities and Tips ............................................................ 234
    CSS Width and Height ............................................................. 234
Paths or Images Look Blurry ..................................................... 234
The Methods save and restore ................................................ 236
Clipping Regions Can Only Get Smaller and Cannot Be Reset 237
Security Exceptions, Cross-domain Images, and Image Data 238
Transformations Affect Drawing in Addition to Paths 238
A Performance Primer ...................................................................... 239
Tools of the Trade ........................................................................ 240
Before We Get to Canvas .......................................................... 245
The DOM and Canvas .............................................................. 248
Caching Context Properties ...................................................... 248
Stop Using save and restore .................................................... 250
Caching with Images and In-Memory Canvases .................................................. 251
Images ................................................................................................................. 252
Text ...................................................................................................................... 252
Shadows .............................................................................................................. 253
Gradients ............................................................................................................ 254
Paths ..................................................................................................................... 255
Multiple Canvases .............................................................................................. 256
Keeping Track of Objects .................................................................................... 257
Hit Testing ............................................................................................................ 257
Size Matters ....................................................................................................... 260
Summary .............................................................................................................. 261

8 The Future of Canvas and 3D Canvas ..................................................................... 263
The Future of 2D Canvas ...................................................................................... 263
New in the Specification ....................................................................................... 264
Hints from the Browsers ...................................................................................... 270
3D (WebGL) Canvas ............................................................................................. 271
WebGL Libraries .................................................................................................. 272
Summary .............................................................................................................. 275

9 Geolocation API .................................................................................................. 277
Understanding Latitude and Longitude .................................................................. 278
Types of Geolocation Data .................................................................................... 280
The Old Ways ......................................................................................................... 280
The HTML5 Way—New Methods for Geolocation .............................................. 281
HTML5 Geolocation API ...................................................................................... 282
Geolocation Support ............................................................................................. 283
Using Geolocation .................................................................................................. 283
The API .................................................................................................................. 283
Geolocation in Action ........................................................................................... 288
Where Am I? .......................................................................................................... 288
A Trailblazing App ................................................................................................ 293
Summary .............................................................................................................. 299

10 HTML5 Storage Options ..................................................................................... 301
Older Storage Methods .......................................................................................... 301
Browser Cookies .................................................................................................... 301
Flash Cookies ......................................................................................................... 302
userData .................................................................................................................. 302
Along the Way ....................................................................................................... 302
Web Storage—sessionStorage and localStorage ........................................ 303
  sessionStorage ................................................................. 303
  localStorage ..................................................................... 304
  API ..................................................................................... 304
WebSQL Database ........................................................................... 307
IndexedDB .................................................................................. 308
  Getting Started with IndexedDB ......................................... 308
  Looking Further ................................................................. 322
FileSystem API for Local Read/Write Access .................................. 323
  FileSystem API Example ..................................................... 324
Offline Pages and the Application Cache ....................................... 326
  Using the Application Cache ............................................... 327
  Important Notes About the Application Cache ....................... 330
Summary .................................................................................. 331

11 Messaging and Web Workers ....................................................... 333
  The Web Messaging API and Cross-Document Messaging .............. 333
  Sending and Receiving Messages .......................................... 334
  Channel Messaging ................................................................ 336
  Security with Web Messages ................................................ 338
Web Workers ................................................................................ 339
  Getting Started with Web Workers ........................................ 340
  A Simple Example ................................................................ 342
  Shared Web Workers ............................................................ 345
  Web Worker Considerations ................................................ 349
Summary .................................................................................. 351

12 Network Communication: WebSockets and XMLHttpRequest Level 2 353
  Real-Time Communication with WebSockets ................................ 354
  Before WebSockets ................................................................ 354
  Getting Started with WebSockets .......................................... 355
  A Complete WebSockets Example .......................................... 358
  Server-Side WebSockets ....................................................... 359
New AJAX Capabilities with XMLHttpRequest Level 2 .................... 359
  New Features in XHR2 ........................................................ 360
Summary .................................................................................. 364

13 Microdata, Other Small Things, and Beyond HTML5 ...................... 365
  Microdata ............................................................................ 365
  Getting Started with Microdata ............................................ 367
  A Microdata Recipe Example ................................................. 371
About the Author

**Simon Sarris** is a web developer focusing primarily on the HTML5 Canvas. Simon has earned a reputation as a go-to source for HTML5 answers. He contributes to the question-and-answer website StackOverflow and has provided the most answers for both the Canvas and HTML5 tags. Simon blogs about Canvas and JavaScript topics, and you can find him online at www.simonsarris.com.
Dedication

To my parents

Acknowledgments

If you look at the hours involved, writing at length is decidedly a solitary act, but it would have been impossible for me to finish this book without the support of several friends.

Book writing is not just time consuming, but life consuming, and I’d like to thank my girlfriend, Betsy Green, for enduring with patience and support over nearly a year of research and writing.

I’d like to express my deepest thanks to Aaron Friel, the greatest friend and colleague I have ever known, for his encouragement and advice for my entire conscious life and for his suggestions and reviews of draft material. I also owe huge thanks to Walter van Roggen, for being the most important mentor of my programming career and for reviewing large portions of this book.

I would like to sincerely thank the people at Sams, especially Neil Rowe and Betsy Gratner, for guiding me through the book-writing process. I owe extra thanks to Spike Xavier for his technical editing and his excellent, thoughtful suggestions.

Many thanks are due to the StackOverflow JavaScript chat room crowd for their encouragement, friendship, and expertise during my research and writing. Among many others I’d like to thank are Jason Brown, Robert Lemon, Abhishek Hingnikar, Amaan Cheval, and Florian Margaine.

My parents are not technology people, but I owe them the biggest thanks. They have supported me from cradle through college with love and resources, allowing me to freely learn and explore in this wonderful world.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn’t like about this book—as well as what we can do to make our books better.

*Please note that we cannot help you with technical problems related to the topic of this book.*

When you write, please be sure to include this book’s title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: consumer@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
Introduction

This is a book about the future of the Web.

For most of human history, it has proven difficult to speculate about the future. Only since the Industrial Revolution have we gotten a grasp of what it might mean to predict things years in advance. Aside from the promise of flying cars that occurred every decade in the 1900s, the future of technological change was often about predictable refinement.

We can imagine that few people considered ambitious futures in the Middle Ages, and only in the 1900s did people begin to see an optimistic nearness: The future was a time just 10 years away. New televisions that were more accurate, better waste treatment, maybe even a man on the moon.

By 1980, the future was clearly a computer-centric world, albeit one still a few years out. By 2010, one third of the world carried in their pockets what would have been billed as a supercomputer in 1980.

Today, the future is near instant. New gadgets and impressive technologies are released almost daily. One set of new technologies is called HTML5, a series of refinements to the Web that has seen rapid adoption since 2010.

This is a book about the future of the Web, and, fortunately for us, it’s already here.

HTML5 is an umbrella term for a series of new features, standards, and application programming interfaces (APIs) that collectively change the way web pages are created and used. With HTML5, applications that were once only possible on desktops or via browser plug-ins are now natively
possible in modern browsers. The adoption of HTML5 aims to take us to an age where the Web is more interoperable, consistent, and easier to author.

**Who Should Read This Book?**

Web developers and web designers exist in a Yin-Yang of roles, sometimes filled by the same person and sometimes by very large teams. This book is intended for both roles, and not only the ones that deal with pure HTML and JavaScript but also the developers and designers who have up until now exclusively worked in Flash and Silverlight. HTML5 offers several replacement opportunities for these rich media plug-ins. The goal of HTML5 is not to obsolete plug-ins, but the new functionality does intend to herald a web where plug-ins, especially ones that provide now-common functionality, are much less necessary.

HTML5 has been around in some agreed-upon form since 2006 and starting in 2009 has had the force of all major browser vendors behind its implementation. In recent years, it has graduated from being a novelty to a set of standards in use by some of the world’s largest websites. If you concern yourself with modern web development, then concerning yourself with HTML5 is essential, and you should read this book.

HTML5 is not strictly HTML; it also encompasses a good deal of new JavaScript APIs. Almost all the contents of HTML5 are relevant to both developers and designers, and even if you do not plan on using many of the parts of HTML5, it is a good idea to get a reading of what is now possible to achieve natively within the browser.

This book assumes basic prior knowledge of JavaScript and HTML. This book assumes no knowledge of JavaScript libraries, no matter how popular they may be, and this book’s code examples do not reference or introduce any libraries except where it is necessary for a component to reasonably function.

**HTML5 and Related Technologies**

HTML5 typically refers to two concepts:

- Technologies and changes contained within the new HTML specifications put forth by the World Wide Web Consortium (W3C) and Web Hypertext Application Technology Working Group (WHATWG).

- The new HTML specifications plus a larger set of new web technologies. This is sometimes called *HTML5 and friends*, or *HTML5 and related technologies*, but is often shortened to just *HTML5*.

There are several common misconceptions about what precisely is contained within HTML5. Mozilla used to host a page titled, "Technologies Often Called Part of HTML5 That Aren’t." They have since removed that page, and instead focus on covering HTML5 and related technologies like everybody else.

For those of us busy building the Web, any distinction does not matter. If a new technology is supported by enough browsers and suits your needs, then you should use it.
Therefore, like most of the HTML5 resources available today, this book encompasses \texttt{HTML5} and related technologies, and we casually call this \texttt{HTML5}.

**Software Requirements**

The code in this book is intended for use in development on modern browsers. When the term \texttt{modern browser} is referenced in this book, it refers to the versions of any popular desktop browser commonly available, except for Internet Explorer, where it refers to only Internet Explorer 9 and above. Although there are less-modern fallback options for many areas of HTML5, it is expected that you will be using a modern browser during development.

If there is a discrepancy in browser support, topics typically note which desktop and mobile browsers are supported. However, no mention of browser support in this book will be as up to date as online compatibility guides, and several websites provide compatibility tables for HTML5 features.

Many JavaScript-centric examples make use of the browser developer console to output data. This console is a common feature of any desktop browser and is accessible through the browser’s developer tools. Developer tools are different for every browser, but are typically enabled via a Tools menu, or with a hotkey such as Ctrl+Shift+I, or F12.

If you are a JavaScript developer or web designer and have never used the browser’s developer tools, I highly recommend seeking out a tutorial. There are several online guides on using the developer console, such as the one for Chrome at https://developers.google.com/chrome-developer-tools/docs/console.

**Code Examples**

The numbered source code listings in this book can be downloaded via the online repository at http://github.com/simonsarris/HTML5Unleashed or http://simonsarris.com/HTML5Unleashed.

Occasionally, when a line of code is too long to fit on one line in the printed book, a code-continuation arrow (\texttt{\textvisiblespace}) is used to mark the continuation.

**How This Book Is Organized**

This book is arranged into four parts. The first provides a briefing on the history and terminology of HTML5, and the other three represent the main areas of HTML5.

HTML5 contains a very broad set of features, and it’s unlikely that a developer would find all of them relevant for any given project. If you are totally new to HTML5 development, it would do you well to begin with Part I. After Part I, every chapter in this book is written to stand on its own, so that you may discover each topic as you please.
Part I: Background
Part I contains a short history and overview of HTML5, as well as explanations of common conventions used in many HTML5 resources, including this book.

▶ Chapter 1, “Why HTML5?”
▶ Chapter 2, “Important Concepts for HTML5”

Part II: New HTML Elements
Part II covers most of the new (and visual) HTML elements in HTML5. It begins with semantic tags, new HTML element attributes, and functionality. It then covers the new rich media tags, which enable native audio and video in the browser.

This part introduces two important concepts seen throughout HTML5: the semantic web (also visited in Chapter 13) and ways to achieve common functionality with less code and fewer plug-ins.

▶ Chapter 3, “Getting Started with HTML5: Semantic Tags, Forms, and Drag and Drop”
▶ Chapter 4, “Rich Media Tags: Video and Audio”

Part III: Canvas
Part III contains four chapters concerning HTML5 canvas. Those both new to and experienced with canvas will benefit from reading the first chapter, which gives a rundown of the API with many detailed notes about canvas context functionality. Canvas has a low-level API compared with Flash, and Chapter 6 covers basic interactivity and state management with the element. Chapter 7 covers canvas performance, but also contains a discussion on tips and peculiarities for canvas newcomers. Finally, Chapter 8 discusses the newer additions to the canvas API and briefly considers the 3D canvas (WebGL) API.

▶ Chapter 5, “2D Canvas”
▶ Chapter 6, “Making Canvas Interactive and Stateful”
▶ Chapter 7, “Canvas Performance, Tips, and Peculiarities”
▶ Chapter 8, “The Future of Canvas and 3D Canvas”

Part IV: HTML5’s JavaScript APIs
Part IV is composed of mostly JavaScript APIs, and is more relevant to developers than artists or designers. The topics in these chapters cover the new native solutions to needs that have arisen over the years as the Web has progressed. The book ends with the small-but-powerful API for adding truly semantic markup to HTML pages, and a brief look at the future.
Links and Real-World Examples

This book contains many links and real-world examples from existing websites. Links and project mentions do not constitute endorsement, and typically only the most popular projects and libraries are mentioned.

This book does not endorse any particular browser, but most examples try to use Chrome or Firefox because they are the most popular cross-platform browsers and widely support nearly every feature covered in the book.
This page intentionally left blank
Important Concepts for HTML5

This short chapter covers some important information to begin our path to HTML5 technologies. It explains some vocabulary used throughout this book that may be new to some readers, and also begins with a briefing on the recurrent goals you see throughout this book.

The Goals of HTML5

HTML5 was born out of visible needs in the browser ecosystem, and the aims of its specifications are all responses to these needs. This section details the three most prominent goals of HTML5, which can be thought of as themes that you see throughout the book.

Improving the Native Web

According to the World Wide Web Consortium (W3C) specification, HTML5 “introduces markup and APIs for emerging idioms, such as Web applications.” More specifically, HTML5 adds syntactic features to the Web that could previously only be accomplished with plug-ins. For instance, if serving video on the Web is a nearly ubiquitous expectation, web browsers ought to be able to accomplish it without additional help. The same goes for audio and other animated or dynamic content. Thus the <audio>, <video>, and <canvas> elements are some of HTML5’s most important additions to the Web.

HTML5 doesn’t just make plug-ins less necessary, it also increases the browser’s functionality to be more in line with native mobile applications. Browser vendors and standards committees have begun work on application programming interfaces (APIs) that expose functionality
of (mobile) devices within the browser. The most prominent example of this is the Geolocation API, which allows browsers to retrieve geographical location much like native phone apps do. There are several smaller niche APIs (such as one for device orientation) that also promise to afford more utility in the browser.

**More Done with Less Code**

One much more subtle feature of HTML5 is the ability to do more with less code. There are a lot of *de facto* standard web page features, such as placeholder text in forms, auto-focusing on a particular input element once the page loads, client-side validation of form input, date and time pickers, and so on.

All of these concepts are considered standard-issue stuff on a modern web page, but every one of them requires at least a little bit of JavaScript to work. Because of this, these concepts are implemented across websites in many different ways, and are at times buggy or inconsistent with each other.

HTML5 simplifies these common design patterns (and more) by creating standardized ways to accomplish them in HTML alone. This empowers designers and also reduces code maintenance and interoperability between platforms because the given feature’s functionality can be more contextually handled by the browser.

**The Semantic Web**

The semantic web is a long-held dream of the Web’s inventor, Tim Berners-Lee. He envisioned a web where content was not only readable by humans but also *understood* by machines. Just as we have to write carefully for humans to comprehend, it would also take a little footwork to make sure programs parsing web pages could pick up on meaningful content.

HTML5 represents the first big semantic push on the Web, and there are important semantic components discussed in Chapters 3 and 13 (“Getting Started with HTML5: Semantic Tags, Forms, and Drag and Drop” and “Microdata, Other Small Things, and Beyond HTML5,” respectively). Now web pages can be marked up to be better understood and categorized by screen readers, search engines, and other web-crawling software. Chapter 13 also contains a brief history of web semantics and their current utility.

**Requisites for HTML5 Development**

This section covers a few important considerations for developing HTML5 web apps. These represent nothing new to a seasoned web developer but are otherwise important for understanding the rest of this book.

**Modern Browser Developer Tools**

Browser developer tools have matured rapidly over the past six years. For both developers and designers, it is strongly recommended that you familiarize yourself with them, as they are referenced occasionally in this book.
Specifically, this book utilizes the JavaScript console in many of its examples, which is used to log messages. This increases the simplicity of the book’s code examples because we can create sample output without bothering with HTML page manipulation. We output to the console with the JavaScript method `console.log(someOutput)`.

Developer tools are typically launched via a Settings menu, or with the command Ctrl+Shift+I, or just F12 depending on the browser. The JavaScript console is found within most developer tools.

The developer console is very flexible, and can also be used to manipulate JavaScript on a page or merely for JavaScript experimentation. Writing directly into the console evaluates the statement and then provides its return value on the next line. Figure 2.1 shows the JavaScript console within the developer tools for Chrome and Firefox, with console-access buttons highlighted and a few commands entered.

![Figure 2.1](image)

This book also mentions newer features of developer tools that specifically aid in debugging some HTML5 features, like local storage and web workers. These are referenced and explained in their respective chapters.

The importance of learning browsers’ developer tools cannot be stressed enough. Familiarizing yourself with them is one of the most important job skills of web developers today. Chrome’s developer tools are top notch, and Firefox has very recently (March 2013) debuted a huge amount of useful new functionality to its toolset.

**HTML5 Fallbacks: Shims, Shivas, and Polyfills**

You’ll find the terms *shim*, *shiv*, and *polyfill* peppered throughout HTML5 resources. Where HTML5 is concerned, the three words represent roughly the same concept: a JavaScript...
library that provides HTML5-like functionality to older browsers, reproducing the native functionality as closely as possible.

In their most generous form, shims and polyfills are drop-in libraries that allow you to use HTML5 features without worrying about proper support for older browsers. The polyfill library detects these unsupporting browsers and attempts to re-create a particular HTML5 feature’s functionality through JavaScript or other means. At the least, these libraries ensure that new HTML content is styled correctly on older browsers.

For a few years, the lack of support in older browsers stalled implementation of HTML5 features. Today, barring impossible-to-reproduce functionality in some features, HTML5 features can safely be used without fear of leaving older browsers in the dust.

Online, you will be able to find a good deal of these polyfill libraries and very good lists of such libraries, such as the one in the Modernizr project: https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills (the project itself is mentioned later in this chapter). Chapter 3 also contains a section on some of the most popular HTML5 polyfill libraries.

**Feature Support and Detection**

Not every HTML5 feature can be reasonably supported with a polyfill. For some features, such as complex canvas applications, it is necessary to support a different kind of fallback. In the case of canvas, that usually entails displaying an image instead of a dynamic animation or a “sorry, please consider upgrading your browser” message instead of interactive content.

**How Do I Know What Features Are Supported?**

Before you use any particular HTML5 feature, it’s a good idea to look at a website of compatibility tables to see which browser versions currently support the feature. There are several of these websites, and the most popular ones are as follows:

- caniuse.com
- html5please.com
- mobilehtml5.org

Figure 2.2 shows a typical compatibility table from caniuse.com. You can see that all versions of Internet Explorer and many mobile browsers do not support WebGL, the 3D specification for HTML5 canvas (2D canvas is much more widely supported).

**Always Use Feature Detection**

Sometimes you’ll want to use a feature even if some browsers do not support it and there is no reasonable fallback. Instead of attempting to detect particular unsupporting browsers, it is always better to detect the existence of features.
For instance, the HTML5 canvas element is not supported on Internet Explorer 8 or below. To test for its support, you could create a canvas element and then check for one of the methods that you would expect to exist:

```javascript
var supportsCanvas = document.createElement('canvas').getContext != undefined;
```

The variable `supportsCanvas` will be true in Internet Explorer 9 and false in Internet Explorer 8. Using `document.createElement('canvas')` alone is not enough because it will successfully create an element of type `HTMLUnknownElement`. Instead, you check to see if the `getContext` method exists on the new element.

There are many other valid ways to test for canvas support (or most HTML5 features for that matter), and instead of bothering to find a working method for each, it is sometimes easier to use a library. The most popular feature detection library is Modernizr (modernizr.com), which can quickly detect all HTML5 and CSS3 (Cascading Style Sheets) features and enable you to respond by executing some appropriate JavaScript, or even conditionally loading different JavaScript and CSS files based on a feature’s support.

**Summary**

The resources in this chapter were popular when this book was written, but there may be better (or more popular) libraries out there today. When considering any kind of library, it is always a good idea to do a fresh search to see what’s most popular and why.

Now that you have the background and vocabulary needed for this book, it’s time to explore the many features of HTML5.
Index

Numbers

3D canvas. See WebGL canvas
45-degree rotations (canvas), 148
1990s rich media content, 69

canvas. See WebGL canvas

A

AAC audio format, 78
abort event, 360
abort() method, 330
add() method, 315
addColorStop() method, 200
aligning canvas text, 197-198
altitude attribute, 284
animations, 210
cancelAnimationFrame() method, 212
canvas interactivity app example
canvas state, tracking, 218-219
<canvas> tag, 216
complete code, 225-230
doDown, doMove, and doUp events, 221-222
finished example, 231
input coordinates, retrieving, 222-223
JavaScript functionality, adding, 217
mouse and touch events, 220-222
shape constructor with prototype methods, creating, 217-218
shapes, drawing, 223-224
frames per second, 212-213
requestAnimationFrame() method
browser support, 211-212
implementing, 210-211
syntax, 212
timing, 213-214

APIs (Application Programming Interfaces)
Application Cache, 326-327
cache files, swapping, 330
cache sections, 328
current state, checking, 329
download, stopping, 330
enabling, 327
events, handling, 330
example, 327-329
file updates, 328
manifest files, 327
specification website, 327
support, 326
troubleshooting, 330-331
updating, 330

WebGL canvas

Device Orientation events, 378
File, 373-375
file inputs, 374
loading files, 374
objects, 374
specification website, 374
support, 374
FileSystem, 323, 375
file system access, 324
specification website, 324
support, 324
writing to files, 325-326
Full Screen, 100, 377-378

Geolocation, 278
altitude, 284
coordinate information, 284
cords object, 284-285
current location on map, displaying, 288-293
direction of travel, 285
latitude and longitude, 278-280
methods, 283-284
position changes, 285-287
PositionError object, 285
reference, 287-288
request failure, 285
support, 283
syntax, 283
timestamp object, 284
trailblazing app, 293-298
user location data, gathering, 280-282
user speed, 285
asm.js project, 379
attributes
  altitude, 284
  autoIncrement, 315
  autoplay, 89
  binaryType, 356
  bufferedAmount, 356
  canvas, 115-116
  canvas context
    caching, 248-250
    compositing, 202
    image data, 203
    listing of, 118
    paths, 201-202
    shadows, 200
    styling, 200
    text, 202-203
case, 27
controls
  <audio> tag, 79, 86-88
  <video> tag, 76, 86-88
coords object, 284-285
currentTime, 92-93
data, 335
dataTransfer
  drag data, setting, 55-56
  drag effects, 56-57
datePublished, 370
empty values, 25
enableHighAccuracy, 285
extensions, 356
fillRule, 264
fillStyle, 119, 138, 200
font, 195, 202
form input, 47-51
  autocomplete, 51
  autofocus, 50
  autosave, 51
  formaction, 51
  formenctype, 51
  formmethod, 51
  formtarget, 51
  min/max/step, 48
  multiple, 50
  novalidate, 51
  pattern, 49
  placeholder, 49
  required, 48-49
  spellcheck, 51
fullScreenElement, 377
fullScreenEnabled, 377
globalAlpha, 184, 202
globalCompositeOperation, 184-186, 202
heading, 285
height
  canvas, 115-116
  <video> tag, 83
imageSmoothingEnabled, 171
indexNames, 315
itemprop, 369
itemscope, 368
JavaScript media API, 90
keyPath, 315
latitude/longitude, 284
lineCap
  defined, 139
  paths, styling, 125-126, 201
lineJoin
  defined, 139
  paths, styling, 125-126, 202
lineWidth, 139
  canvas, 125-126
  paths, 201
  strokeText() method, 194
location, 341
loop, 90
manifest, 327
maximumAge, 286
miterLimit, 125, 127, 139, 202
muted, 85
name, 315
origin, 335
playbackRate, 93
PositionError object, 285
poster, 84-85
preload, 89-90
protocol, 356
quotations, 25, 28
readyState, 92, 356
response, 362
responseType, 362
self, 341
sessionStorage, 303
shadowBlur, 182, 200
shadowColor, 182, 200
shadowOffsetX, 182, 200
shadowOffsetY, 182, 200
source, 335

How can we make this index more useful? Email us at indexes@samspublishing.com
attributes

speed, 285
src, 85-86
strokeStyle, 119, 138, 200
textAlign, 197-198, 202
textBaseline, 197, 202
timeout, 286, 364
type, 77
textBaseline, 197, 202
timeout, 286, 364
transaction, 315
type, 77
url, 356
video-only, 82-85
whitespace characters, 25
width
canvas, 115-116
<video> tag, 83
withCredentials, 362

audio
<audio> tag, 79-80
controls, 86-88
custom controls, creating, 95
encoding, 80-81
fallback options, 81-82
Flash comparison, 95
ease of use, 95-96
feature support, 96-97
flexibility, 96
media protection, 97
platform support, 96
formats
browser compatibility, 78
current support, 78
listing of, 77-78
testing, 78-79
future development
Web Audio API, 100-101
WebRTC, 98
JavaScript API, 90-91
attributes, 90
currentTime attribute, 92-93
events, 90
methods, 90
playbackRate attribute, 93
readyState attribute, 92
W3C demonstration page, 91
looping, 90
older browser support, 81-82
playback rates, 93
playing automatically, 89
preloading, 89-90
source file URL, specifying, 85-86
specified start times, 92-93
<audio> tag, 79-80
attributes
autoplay, 89
controls, 86-88
loop, 90
preload, 89-90
src, 85-86
canPlayType() method, 79
autocomplete attribute, 51
autofocus attribute, 50
autolncrement attribute, 51
autoplay attribute, 89
autosave attribute, 51

B
backward compatibility, 22
HTML5 Boilerplate templates, 23
Webshims library, 22-23
baselines (canvas text), 197
<bdi> tag, 37
beginPath() method, 138, 201
benchmarking, 240-244
Benchmark.js, 240-242
jsPerf.com, 242-244
Benchmark.js, 240-242
Berners-Lee, Tim, 7
semantic web vision, 366-367
W3C, 9
World Wide Web, creating, 7-8
Bespin app, 192
Bezier curves, 130-133
creating, 131-133
cubic, 130
defined, 130
ellipses, drawing, 133
quadratic, 130
bezierCurveTo() method, 130, 139, 201
Bi-directional Isolation tag, 37
binaryType attribute, 356
Bing microdata visualization tool, 373
<blink> tag, 30
Blob object, 374
blobs, 269
<body> tag, 25
Boilerplate templates, 23
<br/>tag, 25
browsers
  canvas
    hints, 270-271
    support, 104-105
compatibility
  application cache, 326
  audio formats, 78
  CSS3, 375
  Device Orientation events, 378
  drag and drop, 67
  feature detection, 18-19
  File API, 374
  FileSystem API, 324
  Geolocation API, 283
  IndexedDB, 309
  Media Capture APIs, 376
  Navigation Timing API, 378
  Page Visibility API, 377
  requestAnimationFrame() method, 211-212
  shared workers, 345
  tables, 18-19
  video formats, 73
  Web Audio API, 376
  Web Messaging API, 334
  Web Notifications, 377
  web workers, 339
  WebGL canvas, 272
  WebRTC, 376
  WebSockets, 354
  XHR2, 360
cookies, 301
development tools, 16-17
Google. See Google
Internet Explorer
  canvas support, 105
  drag and drop compatibility, 67
media
  control appearance, 88
  fallback options, 81-82
  support, 81-82
Mosaic, 8
new features
  CSS3, 375
  Device Orientation events, 378
  File API, 373-375
FileSystem API, 375
Full Screen API, 377-378
Navigation Timing, 378
page visibility, 377
Pointer Lock API, 378
Web Audio API, 376
web notifications, 377
WebRTC, 376
performance profilers, 244-245
plug-ins
  future, 14
  origins, 10
bufferedAmount attribute, 356

caching
  canvas context attributes, 248-250
  images, 251
text, 252
Cailliau, Robert, 8
cancelAnimationFrame() method, 212
canPlayType() method
  <audio> tag, 79
  <video> tag, 74
canvas
  advantages, 109
  animations, 210
cancelAnimationFrame() method, 212
  frames per second, 212-213
  requestAnimationFrame() method, 210-212
timing, 213-214
applications, 103
attributes, sizing, 115-116
browser support, 104
  Internet Explorer, 105
testing, 105
<canvas> tag
  content, 112
  syntax, 111
clearing, 151-152
clipping region, 189
  nonrectangular areas, clearing, 191-192
overview, 189
resetting, 269
rounded corner images, drawing, 189-190
setting, 189
troubleshooting, 237-238
compatibility, 19
compositing, 184
attributes, 202
globalAlpha attribute, 184
globalCompositeOperation attribute, 184-186
hollow paths, 187-188
opaque image portions, filling, 186
context. See context (canvas)
coordinate system, 122-124
creating, 114-115
curves, 127
arc() method, 127-128
arcTo() method, 128-130
Bezier, 130-133
circles, 128
pie slices, 128
rectangles, rounding, 129-130, 189-190
data URL, creating, 116-117
double-buffering, 162
ellipses
creating, 133-134
future, 266
examples, 113-114
fallback content, 112-113
fillRect() method, 115
full-screen API, 100
future, 263-264
blobs, 269
browser hints, 270-271
clipping region, resetting, 269
dashed lines, 266-267
ellipses, 266
fill rules, 264
hit testing, 267-268
image data, 269-270
methods, 270
path primitives, 264-265
text along paths, drawing, 267
text metrics, 269
transformation matrix, 268
gradients, 173
efficient way to fill multiple objects with same gradient, 178-179
linear, 173-175
performance, 254-255
poor way to fill multiple objects with same gradient, 177-178
radial, 175-177
gridlines, 124
history, 103
image data
blurring with nearest-neighbor interpolation, 169
cross-origin, troubleshooting, 238
future, 269-270
putting back onto canvas, 164-166
retrieving, 162-164
image security, 171-173
CORS, 172
drawImage() method with image from different domain, 171
information leakage, 171-172
local file access without restrictions, 172-173
origin-clean flag, 171
images, drawing, 160
implementations to avoid, 110-111
in-memory, 154-159
creating, 154
drawing app with proper transparency example, 157-159
patterns, creating, 180
performance, 155
simple drawing app with transparency problems example, 155-157
interactivity app example
canvas state, tracking, 218-219
<canvas> tag, 216
complete code, 225-230
doDown, doMove, and doUp events, 221-222
finished example, 231
input coordinates, retrieving, 222-223
JavaScript functionality, adding, 217
mouse and touch events, 220-222
shape constructor with prototype methods, creating, 217-218
shapes, drawing, 223-224
JavaScript app protection, 106
line styles, 125-127
corners, 126
ending points, 125-126
mitering ratio, 127
width, 125-126
mouse and touch inputs, 206
  getPos() method, 206-207
  getPos2() method, 207-209
  mouse versus touch events, 209-210
multiple, 256-257
nonzero winding number rule, 135-138
  three subpaths example, 136-137
  two subpaths example, 135
overhead, 106
overview, 103-104
page interaction, 106
paths, 120-122
  attributes, 201-202
  closing, 139
  filling, 122, 139
  lines, adding, 139
  methods, 201-202
  performance, 255-256
primitives, 264-265
starting, 138
stroking, 120-122
subpaths, creating, 139
text along, drawing, 267
tracing, 139
troubleshooting, 235
patterns, 179-182
  CanvasPattern object, 179
  creating, 180
  text, filling, 180
performance, 109, 239
  benchmarking, 240-244
  browser performance profilers, 244-245
  clearing and redrawing, 260-261
  context attributes, caching, 248-250
DOM, 248
double-buffering, 162
drawing on integers, 252
gradients, 254-255
hit testing, 257-260
image caching, 251
in-memory canvases, 155
loops, 245-246
math, 246-247
multiple canvases, 256-257
object tracking, 257
paths, 255-256
save()/restore() methods, 250-251
scaling images, 252
scope, 246
shadows, 253
size, 260
text, 252-253
pixel manipulation
  color examples, 167-169
  fillRect() versus putImageData() method, 166
  image smoothing, 171
  ImageData objects, creating, 164
rectangles
  clearing, 138
  drawing, 115, 118-119
  filling, 138
  methods, 200-201
  outlining, 138
  rect() method, 139
  rounding, 129-130
saving, 140-141
shadows, 182
  attributes, 182, 200
  blur, 254
  clipping region problems, 183
  faking, 254
  methods, 200
  performance, 253
  precomputing, 254
  transformation effects on offsets, 182-183
  zoom scale effects, 184
SVG, compared, 104, 107-109
text, 192
  attributes, 202-203
  accessibility problems, 192
  alignment, 197-198
  along paths, drawing, 267
  alphabetic baseline, 194
  methods, 202-203
  baselines, 197, 269
  bounding box, 269
  caching, 252
  drawing, 194-195
  fonts, 195-196
  future metrics, 269
  Greeking, 253
  kerning problems, 193-194
  measuring, 198-199, 253
  performance, 199, 252-253
  transformation problems, 193
  width, 194-195

How can we make this index more useful? Email us at indexes@samspublishing.com
transformations
- canvas, clearing, 151-152
- current matrix, transforming, 151
- current path disproportionate scaling, 146
- flipping, 145
- future, 268
- matrix written form, 150
- methods, 202
- resetting, 151
- rotations, 146-149
- save() and restore() methods placement, 143-144
- scaling, 143
- text problems, 193
- tracking, 152-154
- translations, 141-142
- troubleshooting, 239

troubleshooting
- blurred images, 235-236
- blurred paths, 235
- clipping regions, 237-238
- cross-origin image data, 238
- CSS width and height, 234-235
- restore() method, 236-237
- save() method, 236-237
- transformations, 239

WebGL, 271-272
- libraries, 272-274
- resources, 275
- support, 272
- website, 271

<canvas> tag
- content, 112
- fallback content, 112-113
- fillRect() method, 115
- syntax, 111
- CanvasGradient object, 173
- CanvasPattern object, 179
- case sensitivity (attributes/tags), 25, 27
- cell phone history, 278
- cellular network geolocation, 282, 291
- channel messaging, 336
  - example
    - first iFrame, 337
    - parent frame, 337-338
    - second iFrame, 338
    - ports, 336
- character encoding, 24
- Chrome
  - JavaScript console tutorial, 3
  - WebGL resource website, 275
- circles, drawing, 128
- circular hit testing, 258
- clear() method, 305, 315
- clearing canvas, 260-261
- clearRect() method, 119, 138, 201
- clearWatch() method, 284
- client/server communication
  - pre-WebSockets, 354-355
    - WebGL, 354
      - API, 354
      - attributes, 356
      - connections, creating, 355-356
      - errors, 357-358
      - events, 356-357
      - sample page, 358-359
      - sent data, limiting, 357
      - server-side, 359
      - specification, 354
      - support, 354
    - XHR2, 359-360
      - binary data, 362-363
      - cross-origin credentials, 362
      - cross-origin requests, 361
      - FormData objects, 363
      - progress events, 360-361
      - timeouts, 363-364
- clip() method, 201
- clipping regions, 189
- nonrectangular areas, clearing, 191-192
- overview, 189
- resetting, 269
- rounded corner images, drawing, 189-190
- setting, 189
- troubleshooting, 237-238
- close event, 356
- close() method
  - web workers, 341
  - WebGL, 357
- closePath() method, 139, 201
- closing databases, 310-312
- closures (tags), 25, 28-29
- code, reducing, 16
- CoffeeScript, 379
- color input (forms), 44-45
<command> tag, 35
commands. See methods
compatibility
application cache, 326
audio formats, 78
backward, 22
HTML5 Boilerplate templates, 23
Webshims library, 22-23
canvas, 104-105
CSS3, 375
Device Orientation events, 378
drag and drop, 67
feature detection, 18-19
File API, 374
FileSystem API, 324
Geolocation API, 283
IndexedDB, 309, 323
Media Capture APIs, 376
Navigation Timing API, 378
Page Visibility API, 377
requestAnimationFrame() method, 211-212
shared workers, 345
tables, 18-19
video formats, 73
Web Audio API, 376
Web Messaging API, 334
Web Notifications, 377
web workers, 339
WebGL, 272
WebRTC, 376
WebSockets, 354
XHR2, 360
compositing, 184
attributes, 202
globalAlpha, 184
globalCompositeOperation, 184-186
hollow paths, 187-188
opaque image portions, filling, 186
confusing elements, 29
connections
databases, 309-310
WebSocket, creating, 355-356
context (canvas)
attributes
caching, 248-250
compositing, 202
fillRule, 264
fillStyle, 138
font, 195
image data, 203
lineCap, 139
lineJoin, 139
lineWidth, 139, 194
listing of, 118
miterLimit, 139
paths, 201-202
shadows, 200
strokeStyle, 138
styling, 200
text, 202-203
textAlign, 197-198
textBaseline, 197
clipping region, 189
nonrectangular areas, clearing, 191-192
overview, 189
rounded corner images, drawing, 189-190
setting, 189
compositing, 184
globalAlpha attribute, 184
globalCompositeOperation attribute, 184-186
hollow paths, 187-188
opaque image portions, filling, 186
coordinate system, 122-124
curves, 127
arc() method, 127-128
arcTo() method, 128-130
Bezier, 130-133
circles, 128
pie slices, 128
rectangles, rounding, 129-130, 189-190
ellipses, 133-134
fillStyle attribute, 119
future
blobs, 269
browser hints, 270-271
clipping regions, resetting, 269
dashed lines, 266-267
ellipses, 266
fill rules, 264
hit testing, 267-268
image data, 269-270
methods, 270
path primitives, 264-265
text along paths, 267
text metrics, 269
transformation matrix, 268

How can we make this index more useful? Email us at indexes@samspublishing.com
gradients, 128-173
   efficient way to fill multiple objects with same gradient, 178-179
   linear, 173-175
   poor way to fill multiple objects with same gradient, 177-178
   radial, 175-177
gridlines, 124
isPointInPath() method, 135
line styles, 125-127
   corners, 126
   ending points, 125-126
   mitering ratio, 127
   width, 125-126
methods
   arc(), 139
   arcTo(), 139
   beginPath(), 138
   bezierCurveTo(), 139
   clearRect(), 119, 138
   closePath(), 139
   createImageData(), 164
   createLinearGradient(), 173
   createPattern(), 179
   createRadialGradient(), 175
   fill(), 139
   fillRect(), 138
   fillText(), 194
   getImageData(), 162-164
   image data, 203
   image drawing, 202
   isPointInPath(), 139
   lineTo(), 139
   measureText(), 198-199
   moveTo(), 139
   paths, 201-202
   putImageData(), 164-166
   quadraticCurveTo(), 139
   rect(), 139
   rectangles, 200-201
   rotate(), 148-149
   scale(), 143
   setTransform(), 151
   state(), 200
   stroke(), 139
   strokeRect, 138
   strokeText(), 194
   styling, 200
   text(), 202-203
   transform(), 151
   transformations, 202
   translate(), 141-142
   nonzero winding number rule, 135-138
   three subpaths example, 136-137
   two subpaths example, 135
   paths, 120-122
      filling, 122
      stroking, 120-122
   patterns, 179-182
   saving/restoring state, 140-141
   shadows, 182-184
   state, 117-118
   strokeStyle attribute, 119
   text, 192
      accessibility problems, 192
      alignment, 197-198
      alphabetic baseline, 194
      baselines, 197
      drawing, 194-195
      fonts, 195-196
      kerning problems, 193-194
      measuring, 198-199
      performance, 199
      transformation problems, 193
      width, 194-195
   transformations
      canvas, clearing, 151-152
      current matrix, transforming, 151
      current path disproportionate scaling, 146
      flipping, 145
      matrix written form, 150
      resetting, 151
      rotations, 146-149
      save() and restore() methods placement, 143-144
      scaling, 143
      text problems, 193
      tracking, 152-154
      translations, 141-142
   controls attribute
      <audio> tag, 79, 86-88
      <video> tag, 76, 86-88
   cookie recipe microdata example, 371-372
      marking with microdata, 372
      skeleton, 371
drawEllipse() method 391

cookies
  browser, 301
  Flash, 302
coordinate system (canvas), 122-124
cords object, 284-285
CORS (Cross-Origin Resource Sharing), 172, 361
createElement() method, 154
createImageData() method, 164, 203
createLinearGradient() method, 173, 200
createPattern() method, 179, 200
createRadialGradient() method, 175, 200
CreativeWork types, 370
cross-document messaging. See messaging
Cross-Origin Resource Sharing (CORS), 172, 361
CSS, 41-43
CSS3, 375
cubic Bezier curves, 130
current rich media content, 71
currentTime attribute, 92-93
cursors (IndexedDB), 319-320
curves, 127
  arcs
    arc() method, 127-128
    arcTo() method, 128-130
  Bezier, 130-133
    creating, 131-133
    cubic, 130
    defined, 130
    ellipses, drawing, 133
    quadratic, 130
circles, 128
  pie slices, 128
rectangles, rounding, 129-130, 189-190

data attribute, 335
data URLs, creating, 116-117
databases
  closing, 310-312
  connecting, 309-310
  deleting, 310-312
  exploring, 319
schema, upgrading, 312-314
transactions, 314-316
data: URLs, 350
<datalist> tag, 51-52
dataTransfer attribute
  drag data, setting, 55-56
  drag effects, 56-57
datePublished attribute, 370
dates and times
  form input, 44-46
  ISO 8601, 370
microdata, 370
datetime attribute, 37
db.js library, 323
debugging. See troubleshooting
declaring Doctypes, 23-24
delete() method, 315
deleting databases, 310-312
<details> tag, 35-36
developer tools, 16-17
development future. See Web, development future
Device Orientation events, 378
Doctypes, declaring, 23-24
document outlines, 33-35
DOM, canvas performance, 248
double-buffering (canvas), 162
drag and drop, 53
  browser compatibility, 67
dragging
  data, 55-56
  effects, 56-57
  events, 58
  requirements, 53-55
example, 59-60
  complete sample script, 63-66
  effects, adding, 62-63
  necessary events, 60-62
oddities, 58-59
dragend events, 58
dragenter events
  defined, 58
  example, 61
dragleave events, 58
dragover events
  defined, 58
  example, 61
dragstart event, 58, 60
drawEllipse() method, 133

How can we make this index more useful? Email us at indexes@samspublishing.com
drawImage() method, 159-160, 202
drawing. See canvas
drawPath() method, 157
drop events
defined, 58
element, 61

effects (drag)
adding, 62-63
overview, 56-57
eccentricity method, 134, 266
eccentricities

equator, 279
errors
event, 356, 360
web workers, 350
WebSockets, 357-358
ES6 (ECMAScript 6), 379
events
abort, 360
application cache, handling, 330
canvas mouse and touch, handling, 220-222
close, 356
Device Orientation, 378
drag
complete sample script, 63-66
data, setting, 55-56
effects, 56-57, 62-63
listing of, 58
necessary, 60-62
odities, 58-59
error, 356, 360
IDBRequest, handling, 310
JavaScript media API, 90
load, 360
loadend, 360
loadstart, 360
message, 356
message listeners, 335
mouse versus touch, 209-210
oncached, 330
onchecking, 330
ondownloading, 330
onerror, 330
onouptate, 330
onobsolet, 330
onprogress, 330
onupdateready, 330
onupgradeneeded, 313
open, 356
progress, 360-361
storage, 305-306
timeout, 360, 364
WebSockets, 356-357
Example.html, 113
exitFullScreen() method, 377
Extensible Hypertext Markup Language (XHTML), 9
Extensible Markup Language (XML), 9
extensions attribute, 356
f
failbacks. See polyfills
feature detection library, 19
feature support
browser compatibility tables, 18-19
detection, 18-19
ffmpeg command-line tool, 81
Fibonacci numbers web worker example, 342-345
HTML code, 343
worker code, 344-345
<figcaption> tag, 36
<figure> tag, 36
File API, 373-375
file inputs, 374
loading files, 374
objects, 374
specification website, 374
support, 374
File object, 374
FileList objects, 374
FileReader objects, 374
FileSystem API, 323, 375
  file system access, 324
  specification website, 324
  support, 324
  writing to files, 325-326
fill() method, 122, 139, 201
fill rules, 264
fillRect() method, 115, 138, 200
fillRule attribute, 264
fillStyle attribute, 119, 138, 200
fillText() method, 194, 203
Flash
  cookies, 302
  future, 14
  media elements, compared, 95
    ease of use, 95-96
    feature support, 96-97
    flexibility, 96
    media protection, 97
    platform support, 96
  origins, 10
flipping images (canvas), 145
font attribute, 195, 202
fonts, 195-196
<footer> tag, 32
<form> tags, 50
formaction attribute, 51
formats
  audio
    browser compatibility, 78
    current support, 78
    listing of, 77-78
    testing, 78-79
  video
    browser compatibility, 73
    current support, 74
    MP4, 72
    testing, 74-75
    Theora, 72
    WebM, 72
FormData object, 363
formenctype attribute, 51
formmethod attribute, 51
forms
  input attributes, 47-51
    autocomplete, 51
    autofocus, 50
    autosave, 51
    formaction, 51
    formenctype, 51
    formmethod, 51
    formtarget, 51
    min/max, 48
    multiple, 50
    nesting in <form> tags, 50
    novalidate, 51
    pattern, 49
    placeholder, 49
    required, 48-49
    spellcheck, 51
    step, 48
  input types, 41-47
    color, 44-45
    CSS styling to current state, 41-43
    dates and times, 44-46
    email, 46-47
    no presentational differences, 41
    number, 43-44
    presentational differences, 41
    range, 44
    search, 47
    telephone numbers, 47
  new features, 40
    <datalist> tag, 51-52
    <kegen> tag, 53
    <output> tag, 52
  formtarget attribute, 51
frames per second animations, 212-213
frames/framesets, 29
Full Screen API, 100, 377-378
fullScreenElement attribute, 377
fullScreenEnabled attribute, 377
functions. See methods
future
  canvas, 263-264
    blobs, 269
    browser hints, 270-271
    clipping region, resetting, 269
    dashed lines, 266-267
    ellipses, 266
fill rules, 264
hit testing, 267-268
image data, 269-270
methods, 270
path primitives, 264-265
text along paths, drawing, 267
text metrics, 269
transformation matrix, 268
WebGL, 271-275
IndexedDB, 323
web development, 379
asmjs project, 379
CoffeeScript, 379
Dart, 379

G

google.html, 289-291
Geolocation API, 278
  altitude, 284
cordinate information, 284
coods object, 284-285
current location on map, displaying,
  288-293
  browser support, testing, 289
cellular network, 291
GPS positions, 292
  source code listing, 289-291
  Wi-Fi data, 291
direction of travel, 285
latitude and longitude, 278-280
methods, 283-284
position changes, monitoring, 286-287
position options, 285-286
PositionError object, 285
reference, 287-288
request failure, 285
support, 283
syntax, 283
timestamp object, 284
trailblazing app, 293-298
  accuracy restrictions, turning off, 298
code listing, 294-297
coordinate data accuracy, 294
device sleeping/losing focus, 298
DOM content, 293
map updates, 294
state, 294
walking around action, displaying, 297
user location data, gathering
cellular networks, 282
GPS coordinates, 282
IP addresses, 281
user entry, 281
Wi-Fi, 282
user speed, 285
get() method, 315
getBoundingClientRect() method, 206-207
currentPosition() method, 283
getImageData() method, 162-164, 203
getItem() method, 304
getPos() method, 206
getPos2() method, 207-209
GLGE library, 274
globalAlpha attribute, 184, 202
globalCompositeOperation attribute,
  184-186, 202
goals (HTML5)
code reduction, 16
  native web, improving, 15-16
  semantic web, 16
Google
  Chrome
    JavaScript console tutorial, 3
    WebGL resource website, 275
Dart, 379
Gears browser extension, 302
Maps API
cellular network output, 291
GPS positions, 292
Wi-Fi data, 291
Maps API example, 288-293
browser support, testing, 289
source code listing, 289-291
microdata, visualizing, 373
GPS, 282, 292
generators, 173
filling multiple objects with same gradient
efficient example, 178-179
  poor example, 177-178
linear, 173-175
creating, 173
entire canvas example, 173-174
small shapes example, 175
performance, 254-255
radial, 175-177
Greeking text, 253
gridlines (canvas), 124
images 395

H

<head> tag, 25
<header> tag, 32
heading attribute, 285
height attribute
canvas, 115-116
<video> tag, 83
<hgroup> tag, 33
history
canvas, 103
cell phones, 278
client/server communication, 354-355
rich media content
1990s, 69
current, 71
storage
browser cookies, 301
Flash cookies, 302
Google Gears browser extension, 302
userData object, 302
Web Storage. See Web, storage
WebSQL, 307
Web
browser plug-ins, 10
creation, 7-8
HTML elements, 9
HTML5, introduction, 11-12
Web 2.0, 10
WHATWG, 11
XML/XHTML acceptance, 9
hit testing, 257-260
approximation, 258
circular, 258
future, 267-268
pixel-perfect, 259-260
rectangular, 258
hollow paths, creating, 187-188
<html> tag, 25
HTML4
HTML5, compared, 30
specifications removed, 29
HTML5
creation, 11
future, 14
goals
code reduction, 16
native web, improving, 15-16
semantic web, 16
Media Project, 82, 96
overview, 11-12
popularity, 12-14
specifications, 12
hypertext, 7
IDB. See IndexedDB
IDBObjectStore object, 314-315
IDBRequest events, handling, 310
IDBTransaction object, 314
<iframe> tag, 29
ImageData object
creating
blank, 164
current canvas bitmap, 162-164
putting data back on canvas, 164-166
images
caching, 251
data
blurring with nearest-neighbor interpolation, 169
cross-origin, troubleshooting, 238
future, 269-270
putting back onto canvas, 164-166
retrieving, 162-164
drawing, 160
future, 269-270
performance
drawing on integers, 252
scaling, 252

How can we make this index more useful? Email us at indexes@samspublishing.com
security, 171-173
CORS, 172
drawImage() method with image from
different domain, 171
information leakage, 171-172
local file access without restrictions,
172-173
origin-clean flag, 171
smoothing, 171
imageSmoothingEnabled attribute, 171
<img /> tag, 25
importScript() method, 349
IndexedDB, 308
  compatibility, 309
  complete example, 316-319
cursors, 319-320
databases
  closing, 310-312
  connecting, 309-310
deleting, 310-312
  exploring, 319
  schema, upgrading, 312-314
flow of operations, 321-322
future, 323
indexes, 320-321
libraries supported, 323
object stores, creating, 312-314
output, 318
overview, 308
polyfill, 308
read-only attributes, 315
request events, handling, 310
resources, 322
shortcomings, 322-323
size limitations, 322
specification website, 308
support, 308
transactions, 314-316
  example, 315-316
  IDBTransaction object, 314
  modes, 314
  object stores, accessing, 314-315
indexNames attribute, 315
in-memory canvases, 154-159
  creating, 154
drawing app example
  proper transparency, 157-159
  transparency problems, 155-157
patterns, creating, 180
performance, 155
input (forms)
  attributes, 47-51
    autocomplete, 51
    autofocus, 50
    autosave, 51
    formaction, 51
    formencetype, 51
    formmethod, 51
    formtarget, 51
    min/max, 48
    multiple, 50
    nesting in <form> tags, 50
    novalidate, 51
    pattern, 49
    placeholder, 49
    required, 48-49
    spellcheck, 51
    step, 48
types, 41-47
  color, 44-45
    CSS styling to current state, 41-43
    dates and times, 44-46
    email, 46-47
    no presentational differences, 41
    number, 43-44
    presentational differences, 41
    range, 44
    search, 47
    telephone numbers, 47
Internet Explorer
  canvas support, 105
    drag and drop compatibility, 67
invalid pseudoclass, 42
IP address geolocation, 281
ISO 8601, 370
isPointInPath() method, 135, 139, 202
itemprop attributes, 369
itemscope attribute, 369
J

J3D library, 274
JavaScript animations, 210
cancelAnimationFrame() method, 212
frames per second, 212-213
requestAnimationFrame() method, 210-212
timing, 213-214
application cache, 329
asm.js project, 379
benchmarking, 240-244
Benchmark.js, 240-242
jsPerf.com, 242-244
canvas attributes, 115-116
canvas interactivity app example
CanvasState constructor, 218-219
complete code, 225-230
input coordinates, retrieving, 222-223
mouse and touch events, 220-222
Shape constructor, 217-218
shapes, drawing, 223-224
canvas transformations, tracking, 152-154
channel messaging example
first iFrame, 337
parent frame, 337-338
second iFrame, 338
Chrome console tutorial, 3
console, 17
Geolocation API minimum, 283
HTML5 Media Project, 82
media API, 90-91
attributes, 90
currentTime attribute, 92-93
custom controls, creating, 95
events, 90
methods, 90
playbackRate attribute, 93
readyState attribute, 92
sequential playlists, creating, 94-95
W3C demonstration page, 91
MediaElement.js library, 82
performance optimization
loops, 245-246
math, 246-247
scope, 246

K

kappa, 134
<kegen> tag, 53
key() method, 305
keyPath attribute, 315
Khronos, 271

L

latitude and longitude, 278-280
latitude attribute, 284
libraries
Benchmark.js, 240-242
db.js, 323
feature detection, 19
HTML5 Media Project, 82, 96
IndexedDB supported, 323
jQuery-IndexedDB, 323
MediaElement.js, 82
polyfill, 18
PouchDB, 323
WebGL, 272-274
GLGE, 274
J3D, 274
PhiloGL, 274
SceneJS, 274
Three.js, 272-274
Webshims, 22-23
line styles (canvas), 125-127
corners, 126
ending points, 125-126
mitering ratio, 127
width, 125-126
linear gradients, 173-175
creating, 173
total canvas example, 173-174
small shapes example, 175

How can we make this index more useful? Email us at indexes@samspublishing.com
lineCap attribute, 125-126, 139, 201
lineJoin attribute, 125-126, 139, 202
lineTo() method, 139, 201
lineWidth attribute, 139
  canvas, 125-126
  paths, 201
  strokeText() method, 194
$link> tag, 25
linters, 27
listings
  animations
    frames per second, 213
    requestAnimationFrame() method
    polyfill, 211-212
    timing, 214
audio formats, testing, 79
canvas
  Bezier curves, 131
  bitmap manipulation examples, 167-169
  blurring image data with nearest-neighbor interpolation, 169
  drawImage and fillText, comparing, 241-242
  efficient way to fill multiple objects with same gradient, 178-179
  ellipses, drawing, 133
  examples, 113-114
  fallback content, 113
  filling text with patterns, 180
  flipping images, 145
  getPos() method, 206
  getPos2() method, 208-209
  gridlines, 124
  hollow paths, creating, 187-188
  image data, retrieving, 163
  ImageData.html, 165
  in-memory canvases drawing app, 157-159
  line widths, 125
  opaque portions of image, filling with compositing, 186
  patterns, creating, 180
  pie slice, 128
  poor way to fill multiple objects with same gradient, 177-178
  radial gradients, 176
  rotations around the center, 149
  rounded corner images, drawing, 189-190
  simple drawing app with transparency problems, 155-157
  three subpaths nonzero winding number rule, 136-137
  transformation examples, 146-147
  transformations, tracking, 152-154
  two subpaths nonzero winding number rule, 135
  canvas interactivity app example
    canvas events, 220-221
    complete code, 225-230
    doDown, doMove, and doUp events, 221-222
    input coordinates, retrieving, 223
    shape constructor with prototype methods, creating, 217-218
    shapes, drawing, 224
  channel messaging
    first iFrame, 337
    parent frame, 337-338
    second iFrame, 338
  closing databases, 311
  cookie recipe
    marking with microdata, 372
    skeleton, 371
  download website, 3
  drag and drop
    complete code example, 64-66
    dragenter events, 61
    dragging requirements, 54
    dragover events, 57, 61
    dragstart event, 57, 60
  Fibonacci numbers web worker example
    HTML code, 343
    worker code, 344-345
  FileSystem API, writing to files, 325-326
Geolocation
  mapping source code example, 289-291
  reference, 287-288
  IDBRequest events, handling, 310
IndexedDB
  complete example, 316-319
  cursors, 319-320
  indexes, 321
  object stores, creating, 313-314
  transaction, 315-316
  microdata person description, 368
  poorly-written-but-valid HTML5 page, 25
  responseType attribute, 362
sequential playlists, creating, 94
shared worker example
  JavaScript code, 348-349
  nested iFrame, 347-348
  parent page, 346-347
Three.js, 273-274
trailblazing app
  code listing, 294-297
  DOM content, 293
video format support, testing, 74
web worker JavaScript example, 341-342
Webshims library, loading, 22
WebSockets sample page, 358-359
XHR
  progress event, 361
  timeouts, 364
load event, 360
load() method, 90
loadend event, 360
loadstart event, 360
local Web storage, 304
location attribute, 341
longitude attribute, 284
long-polling, 355
loop attribute, 90
looping audio/video, 90
loops, 245-246

M

manifest attribute, 327
manual entry Geolocation, 281
<mark> tag, 36-37
max attribute, 48
maximumAge attribute, 286
measureText() method, 198-199, 203
measuring text (canvas), 198-199
Media Capture APIs, 376
media elements
  1990s, 69
  audio
    attributes, 85-90
    <audio> tag, 79-80
    formats, 77-79
    source file URL, specifying, 85-86
    controls, 86-88
    conversion tools, 81
current, 71
custom controls, creating, 95
<embed> tag, 101
encoding, 80-81
fallback options, 82
Flash comparison, 95
ease of use, 95-96
feature support, 96-97
flexibility, 96
media protection, 97
platform support, 96
future developments
  full-screen API, 100
  Web Audio API, 100-101
  WebRTC (Real-Time Communication), 98
  WebVTT, 98-100
JavaScript API, 90-91
  attributes, 90
  currentTime attribute, 92
  events, 90
  methods, 90
  playbackRate attribute, 93
  readyState attribute, 92
  sequential playlists, creating, 94-95
  W3C demonstration page, 91
looping, 90
older browser support, 81-82
playback rates, 93
playing automatically, 89
preloading, 89-90
specified start times, 92-93
video
  browser compatibility, 73
  controls, adding, 76
  current support, 74
download image, setting, 84-85
  formats, 71
  muting, 85
  sequential playlists, creating, 94-95
  source file URL, specifying, 85-86
  support, testing, 74-75
  <video> tag, 75-77
  video-only attributes, 82-85
MediaElement.js library, 82, 95
<menu> tag, 35
meridians, 279
message events, 335, 356
MessageChannel object, 336
MessagePort objects, 336

How can we make this index more useful? Email us at indexes@samspublishing.com
messaging

channel, 336
  first iFrame example, 337
parent frame, 337-338
ports, 336
second iFrame, 338
web, 334
  receiving messages, 335
  sending messages, 334-335

<meta> tag, 24
<meter> tag, 38-39
methods
 abort(), 330
  add(), 315
 addColorStop(), 200
append(), 363
arc(), 127-128, 139, 201
arcTo(), 128-130, 139, 201
  rectangles, rounding, 129-130, 189-190
 WHATWG specification, 129
beginPath(), 138, 201
bezierCurveTo(), 130, 139, 201
cancelAnimationFrame(), 212
canPlayType(), 74, 79
canvas context
  image data, 203
  image drawing, 202
  paths, 201-202
  rectangles, 200-201
  styling, 200
  text, 202-203
  transformations, 202
clear(), 305, 315
clearRect(), 119, 138, 201
clearWatch(), 284
clip(), 201
close()
  web workers, 341
  WebSockets, 357
closePath(), 139, 201
createElement(), 154
createImageData(), 164, 203
createLinearGradient(), 173, 200
createPattern(), 179, 200
createRadialGradient(), 175, 200
delete(), 315
drawEllipse(), 133
drawImage(), 159-160, 202
drawPath(), 157
drawImage(), 134, 266
drawRect(), 115, 138, 200
drawText(), 194, 203
Geolocation API, 283-284
get(), 315
getBoundingClientRect(), 206-207
getCurrentPosition(), 283
getImageData(), 162-164, 203
getItem(), 304
getPos(), 206
getPos2(), 207-209
importScript(), 349
isPointInPath(), 135, 139, 202
drawPath(), 139, 201
load(), 90
loading files, 374
measureText(), 198-199, 203
moveTo(), 139, 201
open(), 309
openCursor(), 315
postMessage() channel messaging, 336
  web messaging, 334
put(), 315
putImageData(), 164-166, 203
quadraticCurveTo(), 130, 139, 201
rect(), 139, 201
removeItem(), 5
requestAnimationFrame()
  browser support, 211-212
  frames per second, 212-213
  implementing, 210-211
  syntax, 212
  timing, 213-214
requestFullScreen(), 377
resetClip(), 191
resetTransform(), 151, 268
restore() (canvas), 236-237
  performance, 250-251
  state, 140-141, 200
rotate(), 148-149, 202
save() (canvas), 236-237
  performance, 250-251
  state, 140-141, 200
scale(), 143, 202
send(), 357  
setItem(), 304  
setTransform(), 151, 202, 268  
SharedWorker(), 345  
stroke(), 120-122, 139, 201  
strokeRect(), 138, 200  
strokeText(), 194  
swapCache(), 330  
terminate(), 341  
toblob(), 269  
toDataURL(), 116-117, 203  
transform(), 151, 202  
translate(), 141-142, 202  
update(), 330  
watchPosition(), 283, 286-287  
Worker(), 340  

microdata, 367  
  cookie recipe example, 371-372  
    marking with microdata, 372  
    skeleton, 371  
CreativeWork types, 370  
dates, 370  
nesting, 369  
nonvisible, 370  
person description example, 368-369  
reviews, 370  
schema hierarchy, 370  
types supported website, 370  
visualizing, 373  
vocabulary, 367  

microformats, 367  
min attribute, 48  
MinimalExample.html, 114  
Miro Video Converter, 81  
imiterLimit attribute, 125, 127, 139, 202  
Modernizr feature detection library, 19  
Mosaic web browser, 8  

mouse and touch inputs, 206  
  canvas interactivity app example  
    canvas state, tracking, 218-219  
    <canvas> tag, 216  
    complete code, 225-230  
    doDown, doMove and doUp events, 221-222  
    finished example, 231  
    input coordinates, retrieving, 222-223  
    JavaScript functionality, adding, 217  
    mouse and touch events, 220-222  

shape constructor with prototype methods, creating, 217-218  
shakes, drawing, 223-224  
getPos() method, 206-207  
getPos2() method, 207-209  
mouse versus touch events, 209-210  
moveTo() method, 139, 201  
Mozilla, 9  
  validator, 26  
  WebGL resource website, 275  
  MP3 audio format, 77  
  MP4 video format, 72  
  multiple attribute, 50  
  multiple canvases performance, 256-257  
  muted attribute, 85  
  muting video, 85  

name attribute, 315  
<nav> tag, 32  
Navigation Timing API, 378  
nesting  
    microdata, 369  
    web workers, 350  
new browser features  
  CSS3, 375  
    Device Orientation events, 378  
    File API, 373-375  
      file inputs, 374  
      loading files, 374  
      objects, 374  
      specification website, 374  
      support, 374  
    FileSystem API, 375  
    Full Screen API, 377-378  
    Media Capture APIs, 376  
    Navigation Timing, 378  
    page visibility, 377  
    Pointer Lock API, 378  
    Streams API, 376  
    Web Audio API, 376  
    web notifications, 377  
    WebRTC, 376  

How can we make this index more useful? Email us at indexes@samspublishing.com
new features
Doctypes, 23-24
document outlines, 33-35
drag and drop, 53
  browser compatibility, 67
drag data, 55-56
drag effects, 56-57
drag events, 58
dragging requirements, 53-55
  oddities, 58-59
forms, 40
  <datalist> tag, 51-52
  input attributes, 47-51
  input types, 41-47
  <keygen> tag, 53
  <output> tag, 52
frames/framesets, 29
HTML4
  HTML5, compared, 30
  specifications removed, 29
meta character encoding, 24
obsolete features, 29-30
semantic tags, 31-33
  <article>, 32
  <aside>, 33
  <bdi>, 37
  <command>, 35
  <details>, 35-36
  <figcaption>, 36
  <figure>, 36
  <header> and footer>, 32
  <hgroup>, 33
  <mark>, 36-37
  <menu>, 35
  <nav>, 32
  <ruby>/<rt>/<rp>, 37
  <section>, 32
  <summary>, 35-36
  <time>, 37
syntax, 25
validation, 25-27
visual tags, 37-40
  <meter>, 38-39
  <progress>, 39-40
nonvisible microdata, 370
nonzero winding number rule, 135-138
  three subpaths example, 136-137
  red directional arrows, blue crossing
  lines, 138
  separating, 137
  two subpaths example, 135
novalidate attribute, 51
numeric input (forms), 43-44
object stores, creating, 312-314
objects
  Blob, 374
  CanvasGradient, 173
  CanvasPattern, 179
  coords, 284-285
  File, 374
  FileList, 374
  FileReader, 374
  FormData, 363
  IDBObjectStore, 314-315
  IDBTransaction, 314
  ImageData
    creating blank, 164
    creating with current canvas bitmap, 162-164
    putting data back on canvas, 164-166
  MessageChannel, 336
  MessagePort, 336
  PositionError, 285
  TextMetrics, 199
  timestamp, 284
  tracking, 257
  userData, 302
obsolete features, 29-30
offline pages. See application cache
Ogg audio format, 77
oncached event, 330
onchecking event, 330
ondownloading event, 330
onerror events, 330
onerror events, 330
onobsolete event, 330
onprogress event, 330
onupdateready event, 330
onupgradeneeded event, 313
open event, 356
open() method, 309
openCursor() method, 315
optional pseudoclass, 41
O’Reilly, Tim, 10
origin attribute, 335
<output> tag, 52

P
Page Visibility API, 377
paths, 120-122
attributes, 201-202
blurred, troubleshooting, 235
closing, 139
filling, 122, 139
hollow, creating, 187-188
lines, adding, 139
methods, 201-202
performance, 255-256
primitives, 264-265
starting, 138
stroking, 120-122
subpaths, creating, 139
text along, drawing, 267
tracing, 139
pattern attribute, 49
patterns, 179-182
CanvasPattern object, 179
creating, 180
text, filling, 180
performance
canvas, 109, 239
benchmarking, 240-244
browser performance profilers, 244-245
clearing and redrawing, 260-261
context attributes, caching, 248-250
DOM, 248
double-buffering, 162
drawing on integers, 252
gradients, 254-255
hit testing, 257-260
image caching, 251
in-memory canvases, 155
loops, 245-246
math, 246-247
multiple canvases, 256-257
object tracking, 257
paths, 255-256
save() / restore() methods, 250-251
scaling images, 252
scope, 246
shadows, 253
size, 260
text, 199, 252-253
gradients
efficient way to fill multiple objects with
same gradient, 178-179
poor way to fill multiple objects with
same gradient, 177-178
performance profilers, 244-245
PhiloGL library, 274
pie slices, drawing, 128
pixel manipulations
blurring image data with nearest-neighbor
interpolation, 169
color examples, 167-169
fillRect() versus putImageData() method, 166
image data
putting back onto canvas, 164-166
retrieving, 162-164
image smoothing, 171
ImageData objects, creating, 164
placeholder attribute (forms), 49
playbackRate attribute, 93
Playr, 99-100
plug-ins. See browsers, plug-ins
Pointer Lock API, 378
polling servers, 354-355
polyfills
<canvas> tag content, 112-113
defined, 18
IndexedDBShim, 308
Modernizr project website, 18
requestAnimationFrame() method, 211-212
Webshims library, 22-23
poorly-written-but-valid HTML5 page listing, 25
ports (channel messaging), 336
PositionError object, 285
poster attribute, 84-85
postMessage() method
channel messaging, 336
web messaging, 334

How can we make this index more useful? Email us at indexes@samspublishing.com
PouchDB library, 323
preload attribute, 89-90
presentation tags
  <article>, 32
  <aside>, 33
  <bdi>, 37
  <command>, 35
  <details>, 35-36
  <document outlines, 33-35
  <figcaption>, 36
  <figure>, 36
  <header> and footer>, 32
  <meter>, 38-39
  <nav>, 32
  <progress>, 39-40
  <ruby>/<rt>/<rp>, 37
  <section>, 32
  <summary>, 35-36
  <time>, 37
prime meridian, 279
progress events, 360-361
<progress> tag, 39-40
properties. See attributes
protocol attribute, 356
put() method, 315
putImageData() method, 164-166, 203

Q

quadratic Bezier curves, 130
quadraticCurveTo() method, 130, 139, 201
quotations (syntax), 25, 28

R

radial gradients, 175-177
range input (forms), 44
read-only attributes, 315
readyState attribute, 92, 356
receiving messages, 335
rect() method, 139, 201
rectangles
  clearing, 138
  drawing, 115, 118-119
  filling, 138
  methods, 200-201
  outlining, 138
  rect() method, 139
  rounded corner, drawing, 129-130, 189-190
rectangular hit testing, 258
removeItem() method, 5
requestAnimationFrame() method
  browser support, 211-212
  frames per second, 212-213
  implementing, 210-211
  syntax, 212
  timing, 213-214
requestFullScreen() method, 377
required attribute, 48-49
requirements
  dragging, 53-55
  pseudoclass, 41
  software, 3
resetClip() method, 191
resetting canvas transformations, 151
resetTransform() method, 151, 268
resources
  IndexedDB, 322
  server-side WebSockets, 359
  WebGL canvas, 275
response attribute, 362
responseType attribute, 362
restore() method (canvas)
  performance, 250-251
  state, 140-141, 200
  troubleshooting, 236-237
reviews (microdata), 370
rotate() method, 148-149, 202
rotations (canvas), 146-149
  around the center, 149
  entire drawn scenes, 148
rounded rectangles, drawing, 129-130, 189-190
Royal Observatory at Greenwich (London), 279
<rp> tag, 37
<rt> tag, 37
Ruby annotations tags, 37
<ruby> tag, 37
S
save() method (canvas)
  performance, 250-251
  state, 140-141, 200
  troubleshooting, 236-237
Scalable Vector Graphics (SVG), 104
scale() method, 143, 202
scaling
  canvas, 143
  images, 252
SceneJS library, 274
scope (canvas performance), 246
<script> tag, 25
search input (forms), 47
<section> tag, 32
security
  canvas images, 171-173
  CORS, 172
drawImage() method with image from
different domain, 171
  information leakage, 171-172
  local file access without restrictions,
  172-173
  origin-clean flag, 171
web messaging, 338
self attribute, 341
self-closing tags syntax, 25
semantic tags, 31-33
  <article>, 32
  <aside>, 33
  <bdi>, 37
  <command>, 35
  <details>, 35-36
  <figcaption>, 36
  <figure>, 36
  <header> and footer>, 32
  <hgroup>, 33
  <mark>, 36-37
  <menu>, 35
  <nav>, 32
  <ruby>/<rt>/<rp>, 37
  <section>, 32
  <summary>, 35-36
  <time>, 37
semantic web, 16
  Berners-Lee vision, 366-367
  microdata, 367
    cookie recipe example, 371-372
    CreativeWork types, 370
dates, 370
  nesting, 369
  nonvisible, 370
person description example, 368-369
reviews, 370
schema hierarchy, 370
types supported website, 370
visualizing, 373
vocabulary, 367
search engine results, 366
send() method, 357
sending messages, 334-335
servers
  long-polling, 355
  polling, 354-355
server-side WebSockets, 359
session storage, 303
  sessionStorage attribute, 303
setItem() method, 304
setTransform() method, 151, 202, 268
shadowBlur attribute, 182, 200
shadowColor attribute, 182, 200
shadowOffsetX attribute, 182, 200
shadowOffsetY attribute, 182, 200
shadows
  attributes, 182, 200
  clipping region problems, 183
  performance, 253
    blur, 254
    faking, 254
    precomputing, 254
transformation effects on offsets, 182-183
zoom scale effects, 184
shared workers, 345
  constructor, 345
  example, 346-349
    JavaScript code, 348-349
  nested iFrame, 347-348
  output, 349
  parent page, 346-347
names, 345-346
support, 345

How can we make this index more useful? Email us at indexes@samspublishing.com
SharedWorker() method, 345
shims/shivs. See polyfills
size (canvas), 260
Skywriter, 192
smartphones, 278
software requirements, 3
source attribute, 335
specifications, 12
speed attribute, 285
spellcheck attribute, 51
src attribute, 85-86
starting web workers, 340-341
state (canvas), saving/restoring, 117-118, 140-141
step attribute, 48
stopping web workers, 341
storage
application cache, 326-327
cache files, swapping, 330
cache sections, 328
current state, checking, 329
download, stopping, 330
enabling, 327
events, handling, 330
example, 327-329
file updates, 328
manifest files, 327
specification website, 327
support, 326
troubleshooting, 330-331
updating, 330
event, 305-306
FileSystem API, 323
file system access, 324
specification website, 324
support, 324
writing to files, 325-326
history
browser cookies, 301
Flash cookies, 302
Google Gears browser extension, 302
userData object, 302
IndexedDB, 308
closing databases, 310-312
compatibility, 309
complete example, 316-319
connecting to databases, 309-310
cursors, 319-320
database schema, upgrading, 312-314
deleting schema, 310-312
exploring databases, 319
flow of operations, 321-322
future, 323
indexes, 320-321
libraries supported, 323
object stores, creating, 312-314
output, 318
overview, 308
polyfill, 308
read-only attributes, 315
request events, handling, 310
resources, 322
shortcomings, 322-323
size limitations, 322
specification website, 308
support, 308
transactions, 314-316
Web, 303
clear() method, 305
debugging, 306
getItem() method, 304
key() method, 305
local, 304
removeItem() method, 305
session, 303
setItem() method, 304
storage event, 305-306
website, 303
WebSQL, 307
Streams API, 376
stroke() method, 120-122, 139, 201
strokeRect() method, 138, 200
strokeStyle attribute, 119, 138, 200
strokeText() method, 194
styling methods/attributes (canvas context), 200
stylistic syntax
case, 27
quotations, 28
tag closures, 28-29
subpaths, creating, 139
<summary> tag, 35-36
SVG (Scalable Vector Graphics), 104, 107-109
swapCache() method, 330
syntax
  new features, 25
  stylistic
    case, 27
    quotations, 28
    tag closures, 28-29

T

tags
  <audio>, 79-80
    attributes, 85-90
    canPlayType() method, 79
  backward compatibility, 22
    HTML5 Boilerplate templates, 23
    Webshims library, 22-23
  <blink>, 30
  <body>, 25
  <br />, 25
  <canvas>
    content, 112
    fallback content, 112-113
    fillRect() method, 115
    syntax, 111
  case, 25, 27
  closing tags syntax, 25
  closures, 28-29
  <datalist>, 51-52
  document outlines, 33-35
  <embed>, 101
  <form>, 50
  <head>, 25
  <html>, 25
  <iframe>, 29
  <img />, 25
  <keygen>, 53
  <link>, 25
  <meta>, 24
  <output>, 52
  quotations, 25
  <script>, 25
  semantic, 31-33
    <article>, 32
    <aside>, 33
    <bdi>, 37
    <command>, 35
    <details>, 35-36
  document outlines, 33-35
  <figcaption>, 36
  <figure>, 36
  <header> and footer>, 32
  <hgroup>, 33
  <mark>, 36-37
  <menu>, 35
  <nav>, 32
  <ruby>/<rt>/<rp>, 37
  <section>, 32
  <summary>, 35-36
  <time>, 37
  <video>, 75-77
    audio shared attributes, 85-90
    canPlayType() method, 74
    controls attribute, 76
    source element type attribute,
      specifying, 77
    syntax, 75
    video-only attributes, 82-85
    visual, 37-40
    <meter>, 38-39
    <progress>, 39-40
  telephone numbers (forms), 47
  templates, 23
  terminate() method, 341
  testing
    audio formats, 78-79
    canvas support, 105
      hit:
        approximation, 258
        circular, 258
        pixel-perfect, 259-260
        rectangular, 258
    video formats, 74-75
  text, 192
    accessibility problems, 192
    alignment, 197-198
    along paths, drawing, 267
    alphabetic baseline, 194
    attributes, 202-203
    baselines, 197, 269
    bounding box, 269
    caching, 252
    drawing, 194-195
    fonts, 195-196
    future metrics, 269
    Greeking, 253
    kerning problems, 193-194
measuring, 198-199, 253
methods, 202-203
patterns, filling, 180
performance, 199, 252-253
transformation problems, 193
width, 194-195
textAlign attribute, 197-198, 202
textBaseline attribute, 197, 202
TextMetrics objects, 199
Theora video format, 72
Three.js, 272-274
<time> tag, 37
timeout attribute, 286, 364
timeout event, 360, 364
timeouts (XHR2), 363-364
times. See dates and times
timestamp object, 284
timing animations, 213-214
toblob() method, 269
toDataURL() method, 116-117, 203
tools
   benchmarking
      Benchmark.js, 240-242
      jsPerf.com, 242-244
   browser
      developer, 16-17
      performance profilers, 244-245
ffmpeg command-line, 81
linters, 27
media conversion, 81
microdata, visualizing, 373
validators, 26
touch input coordinates. See mouse and
touch inputs
tracking canvas
   objects, 257
   transformations, 152-154
trailblazing app, 293-298
   accuracy restrictions, turning off, 298
   code listing, 294-297
   coordinate data accuracy, 294
   device sleeping/losing focus, 298
DOM content, 293
   map updates, 294
   state, 294
   walking around action, displaying, 297
transaction attribute, 315
transactions (IndexedDB), 314-316
   IDBTransaction object, 314
   modes, 314
   object stores, accessing, 314-315
   sample, 315-316
transform() method, 151, 202
transformations
   canvas, clearing, 151-152
   current matrix, transforming, 151
   current path disproportionate scaling, 146
   flipping, 145
   future, 268
   matrix written form, 150
   methods, 202
   resetting, 151
   restore() method placement, 143-144
   rotations, 146-149
      around the center, 149
      entire drawn scenes, 148
   save() method placement, 143-144
   scaling, 143
   shadow offsets, 182-183
   text problems, 193
   tracking, 152-154
   translating, 141-142
   troubleshooting, 239
Transform.js, 152-154
translate() method, 141-142, 202
translations (canvas), 141-142
troubleshooting
   application cache, 330-331
   canvas
      blurred images, 235-236
      blurred paths, 235
      clipping regions, 237-238
      cross-origin image data, 238
      CSS width and height, 234-235
      restore() method, 236-237
      save() method, 236-237
      transformations, 239
Web Storage, 306
web workers, 350
type attribute, 77
U

update() method, 330
updating application cache, 330
upgrading database schema, 312-314
url attribute, 356
user location data, gathering
  GPS coordinates, 282
  new methods
    cellular networks, 282
    Wi-Fi, 282
  old ways
    IP addresses, 281
    user entry, 281
userData object, 302

V

valid pseudoclass, 42
validation
  new features, 25-27
  validators, 26
versions (HTML), 9
video
  attributes, 82-85
  controls, 76, 86-88
  conversion tools, 81
  custom controls, creating, 95
download image, setting, 84-85
  encoding, 80-81
  fallback options, 81-82
Flash comparison, 95
  ease of use, 95-96
  feature support, 96-97
  flexibility, 96
  media protection, 97
  platform support, 96
formats, 71
  browser compatibility, 73
  current support, 74
MP4, 72
testing, 74-75
Theora, 72
WebM, 72

future developments
  full-screen API, 100
  WebRTC, 98
  WebVTT, 98-100
JavaScript API, 90-91
  attributes, 90
  currentTime attribute, 92-93
  events, 90
  methods, 90
  playbackRate attribute, 93
  readyState attribute, 92
  W3C demonstration page, 91
looping, 90
muting, 85
older browser support, 81-82
playback rates, 93
playing automatically, 89
preloading, 89-90
sequential playlists, creating, 94-95
sources, specifying
  URLs, 85-86
  types, 77
  specified start times, 92-93
  <video> tag, 75-77
<video> tag, 75-77
attributes, 82-90
  autoplay, 89
  controls, 86-88
  height, 83
  loop, 90
  muted, 85
  poster, 84-85
  preload, 89-90
  src, 85-86
  width, 83
canPlayType() method, 74
controls attribute, 76
source element type attribute, specifying, 77
syntax, 75
Video.js, 95
visual tags, 37-40
  <meter>, 38-39
  <progress>, 39-40
visualizing microdata, 373
VLC media player, 81
W

W3C (World Wide Web Consortium)
  Doctypes, 24
  HTML versions, 9
  HTML5
    specifications, 12
    stable/complete declaration, 14
  JavaScript media API demonstration
    page, 91
  origins, 9
  validator, 26
  WHATWG, 11
  XML/XHTML acceptance, 9
  watchPosition() method, 283, 286-287
  WAV audio format, 78
  Web
    Audio API, 100-101, 376
    development future, 379
    asm.js project, 379
    CoffeeScript, 379
    Dart, 379
    improving, 15-16
    messaging, 334
      receiving messages, 335
      security, 338
      sending messages, 334-335
      specification website, 334
      support, 334
  notifications, 377
  origins
    browser plug-ins, 10
    creation, 7-8
    HTML elements, history, 9
    HTML5, introduction, 11-12
    Web 2.0, 10
    WHATWG, 11
    XML/XHTML acceptance, 9
  semantic. See semantic web
  storage, 303
    debugging, 306
    local, 304
    methods
      clear(), 305
      getItems(), 304
      key(), 305
      removeItem(), 5
      setItem(), 304
  session, 303
  storage event, 305-306
  website, 303
  Web 2.0, 10
  Web Hypertext Application Technology Working
  Group. See WHATWG
  web workers, 339-340
    compatibility table, 339
    creating, 340
    debugging, 350
    embedding, 350
    errors, 350
    Fibonacci numbers example, 342-345
      HTML code, 343
      worker code, 344-345
    global scope, 341
    Google Chrome file access error, 340
    importing scripts, 349-350
    JavaScript example, 341-342
    nesting, 350
    shared, 345
      constructor, 345
      example, 346-349
      names, 345-346
      support, 345
    source file location, 341
    specification website, 340
    starting, 340-341
    stopping, 341
    worker information, receiving, 340
  WebGL canvas, 271-272
    libraries, 272-274
      GLGE, 274
      J3D, 274
      PhiloGL, 274
      SceneJS, 274
      Three.js, 272-274
    resources, 275
    support, 272
    website, 271
  WebRTC (Real-Time Communication), 98, 376
  Webshims library, 22-23
  websites
    application cache specification, 327
    Benchmark.js, 240
    Berners-Lee semantic web vision, 367
    Bing microdata visualization tool, 373
    browser compatibility tables, 18-19
Chrome JavaScript console tutorial, 3  
cross-origin resource sharing, 361  
CSS3, 375  
db.js, 323  
Device Orientation events specification, 378  
File API specification, 374  
FileSystem API specification, 324, 375  
Full Screen API specification, 378  
GLGE, 274  
Google structured data testing tool, 373  
HTML4 versus HTML5, 30  
HTML5  
   Boilerplate templates, 23  
   Media Project library, 82, 96  
IndexedDB  
   resources, 322  
   specification, 308  
IndexedDBShim, 308  
ISO 8601, 370  
J3D, 274  
jQuery-IndexedDB, 323  
linter, 27  
Media Capture APIs specifications, 376  
MediaElement.js library, 82, 95  
microdata  
   CreativeWork types, 370  
   schema hierarchy, 370  
   types supported, 370  
microformats, 367  
Miro Video Converter, 81  
Mozilla validator, 26  
Navigation Timing specification, 378  
obsolete features, 30  
Page Visibility specification, 377  
PhiloGL library, 274  
Playr, 99  
Pointer Lock specifications, 378  
polyfill libraries, 18  
PouchDB, 323  
SceneJS, 274  
server-side WebSockets, 359  
shared worker compatibility, 345  
source code listings downloads, 3  
Streams API specification, 376  
Three.js repository, 274  
Video.js, 95  
VLC media player, 81  
W3C (World Wide Web Consortium)  
   JavaScript media API demonstration, 91  
   validator, 26  
Web  
   Audio API, 101, 376  
   Messaging API specification, 334  
   Notifications specification, 377  
   Storage API, 303  
web worker specification, 340  
WebGL canvas, 271  
   resources, 275  
   support, 272  
WebRTC, 98, 376  
Webshims library, 23  
WebSockets, 354  
WebSQL specification, 307  
WebSockets, 354  
   attributes, 356  
   connections, creating, 355-356  
   errors, 357-358  
   events, 356-357  
   sample page, 358-359  
   sent data, limiting, 357  
   server-side, 359  
   specification, 354  
   support, 354  
WebSQL, 307  
WebVTT, 98-100  
WHATWG (Web Hypertext Application Technology Working Group), HTML5, 11  
   creating, 11  
   specifications, 12  
   whitespace characters (attributes), 25  
width  
   canvas text, 194-195  
   line styles, 125-126  
width attribute  
   canvas, 115-116  
   <video> tag, 83  
Wi-Fi geolocation, 282, 291  
winding number rule. See nonzero winding number rule  
withCredentials attribute, 362  
Worker() method, 340  
World Wide Web Consortium. See W3C  
WorldWideWeb project, 8  

How can we make this index more useful? Email us at indexes@samspublishing.com
X–Z

XHR2 (XMLHttpRequest Level 2), 359-360
  binary data, 362-363
  cross-origin
    credentials, 362
    requests, 361
  FormData objects, 363
  progress events, 360-361
  support, 360
  timeouts, 363-364
XHTML (Extensible Hypertext Markup
  Language), 9
XML (Extensible Markup Language), 9