Learn by doing: less theory, more results

HTML5 Game Development by Example

Second Edition

Make the most of HTML5 techniques to create exciting games from scratch

Beginner's Guide

Makzan

[PACKT] PUBLISHING open source community experience distilled

HTML5 Game Development by Example Beginner's Guide

Second Edition

Make the most of HTML5 techniques to create exciting games from scratch

Makzan

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

HTML5 Game Development by Example Beginner's Guide
Second Edition

First published: August 2011

Second edition: June 2015

Production reference: 2250615

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

ISBN 978-1-78528-777-0

www.packtpub.com

Credits

Author

Makzan

Reviewers

Lauri Hosio

Dan Nagle

Matt Palmerlee

Leonardo Risuleo

Commissioning Editor

Dipika Gaonkar

Acquisition Editors

Vivek Anantharaman

Sam Wood

Content Development Editor

Arwa Manasawala

Technical Editor

Menza Mathew

Copy Editors

Ameesha Green

Jasmine Nadar

Project Coordinator

Shweta H Birwatkar

Proofreader

Safis Editing

Indexer

Tejal Daruwale Soni

Production Coordinator

Manu Joseph

Cover Work

Manu Joseph

About the Author

Makzan focuses on the fields of web development and game design. He has over 14 years of experience in building digital products. He has worked on real-time multiplayer interaction games, iOS applications, and rich interactive websites.

He has written three books, on building a Flash virtual world, and creating games with HTML5 and the latest web standards and developed a video course as well. He currently teaches courses in Hong Kong and Macao SAR. He writes tutorials and shares his know-how on `makzan.net`.

I wish to thank my wife, Candy, for her patience and understanding.
I would also like to thank the entire team at Packt Publishing. The book would not have been possible without their help. I thank all the reviewers for providing useful comments from which I have learned a lot.

About the Reviewers

Lauri Hosio has been making games since he discovered QBASIC with his friends in elementary school. Professionally, he's worked with web and mobile games for over 7 years.

Previously, he worked at Rovio as the acting lead game programmer on Angry Birds Friends. At other companies, he has also worked on web-based MMO games and general web development.

Before HTML5, he made independent web games with Flash that were played by millions of players. Some of his games have been published by AddictingGames and ArmorGames. Lauri currently works in Kansas City, Missouri, on mobile games and full-stack web development.

Dan Nagle has, since graduating as a valedictorian in computer engineering from the Mississippi State University in 2003, written and published apps for Android, Windows, Mac, Linux, iOS, numerous web apps, network servers, and pure embedded C. He is the author of the book, *HTML5 Game Engines: App Development and Distribution*, which was published by CRC Press. He has also written articles and spoken at conferences about developing websites and HTML5-based games.

For about 4 years, he owned and operated a web company that focused on website hosting and custom game development. Before that, he was an electrical engineer who developed embedded systems.

Currently, he is a senior software engineer who writes control software, web interfaces, and mobile apps for network devices that distribute HD video. He can be reached through his website at `http://DanNagle.com/`.

Matt Palmerlee has been developing software professionally since 2001 and has a passion for JavaScript and HTML5 game development. He built Astriarch, an open source multiplayer space strategy game, using HTML5 and Node.js. Matt has developed other HTML5 games, which were published by Mastered Software, his software development consulting company.

Matt occasionally blogs about interesting aspects of software development and HTML5 games on his personal website at `http://mattpalmerlee.com/`.

Leonardo Risuleo is the owner and creative director of Small Screen Design. He is a designer and developer with several years of experience in mobile, new media, and user experience. Leonardo is a highly dedicated professional, and he's passionate about what he does. He started his career in 2003, and in the last few years, he has worked on a variety of different mobile and embedded platforms for a number of well-known brands and studios. Leonardo designs, prototypes, and develops mobile applications, games, widgets, and websites.

From 2008 to 2010, he had the honor of being the Nokia Developer Champion, a recognition and reward program for top mobile developers worldwide. In 2008, Leonardo formally founded Small Screen Design (`https://www.smallscreendesign.com`), a design and development studio focused on mobile design and user experience. In 2015, he became Digital Champion—an ambassador for the Digital Agenda—for Squillace, to help every European become digital.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface **xi**

Chapter 1: Introducing HTML5 Games **1**

Discovering new features in HTML5 **2**

Canvas 2

Audio 2

Touch Events 2

GeoLocation 2

WebGL 3

WebSocket 3

Local storage 3

Offline applications 4

Discovering new features in CSS3 **4**

CSS3 transition 5

CSS3 transform 6

CSS3 animation 6

The benefit of creating HTML5 games **7**

Free and open standards 7

Support for multiple platforms 7

Native app-rendering performance in particular scenarios 8

Breaking the boundary of usual browser games 8

Building HTML5 games 9

What others are playing with HTML5 **9**

Coca-Cola's Ahh campaign 9

Asteroid-styled bookmarklet 10

X-Type 10

Cursors.io 11

What we are going to create in this book **12**
Preparing the development environment **13**
Summary **13**
Chapter 2: Getting Started with DOM-based Game Development **15**
Preparing the HTML documents for a DOM-based game **16**
Time for action – installing the jQuery library **16**
New HTML5 doctype 18
Header and footer 18
The best practice to place the JavaScript code 19
Choosing the jQuery file 19
Running jQuery inside a scope 20
Running our code after the page is ready 20
Downloading the image assets 21
Setting up the Ping Pong game elements **22**
Time for action – placing Ping Pong game elements in the DOM **22**
Using jQuery 25
Understanding basic jQuery selectors 26
Understanding the jQuery CSS function 27
Manipulating game elements in DOM with jQuery 27
Understanding the behavior of absolute position 28
Declaring global variables in a better way 29
Getting mouse input **29**
Time for action – moving DOM objects by mouse input **30**
Getting the mouse event 32
RequestAnimationFrame 32
Checking the console window 32
Moving a DOM object with JavaScript Interval 33
Time for action – Moving the ball with JavaScript Interval **34**
Creating a JavaScript timer with the setInterval function 37
Understanding the game loop 38
Separating the data and the view logic 38
Beginning collision detection **39**
Time for action – hitting the ball with the paddles **39**
Controlling the left paddle movement **41**
Time for action – auto moving the left paddle **42**
What just happened? 42
Showing text dynamically in HTML **42**
Time for action – Showing the score of both players **43**
What just happened? 44
Summary **45**

Chapter 3: Building a Card-matching Game in CSS3 47
Moving game objects with CSS3 transition 48
Time for action – moving a playing card around 48
2D transform functions 51
3D transform functions 51
Tweening the styles using CSS3 transition 52
Creating a card-flipping effect 54
Time for action – flipping a card with CSS3 54
Toggling a class with jQuery's toggleClass function 56
Introducing CSS' perspective property 57
Introducing backface-visibility 58
Creating a card-matching memory game 59
Downloading the sprite sheet of playing cards 59
Setting up the game environment 60
Time for action – preparing the card-matching game 60
Cloning DOM elements with jQuery 66
Selecting the first child of an element in jQuery using child filters 66
Vertically aligning a DOM element 66
Using CSS sprite with a background position 67
Adding game logic to the matching game 68
Time for action – adding game logic to the matching game 69
Executing code after the CSS transition has ended 72
Delaying code execution on flipping cards 72
Randomizing an array in JavaScript 72
Storing internal custom data with an HTML5 custom data attribute 74
Accessing custom data attribute with jQuery 75
Making other playing card games 76
Embedding web fonts into our game 77
Time for action – embedding a font from the Google Fonts directory 77
Choosing different font delivery services 80
Summary 80

Chapter 4: Building the Untangle Game with Canvas and the Drawing API 81
Introducing the HTML5 canvas element 82
Drawing a circle in the Canvas 83
Time for action – drawing color circles in the Canvas 83
Putting in fallback content when the web browser does not support the Canvas 85
The Canvas context 85
Drawing circles and shapes with the Canvas arc function 86
Converting degrees to radians 86

Executing the path drawing in the Canvas 86
Beginning a path for each style 87
Closing a path 88
Wrapping the circle drawing in a function 88
Time for action – putting the circle drawing code into a function 88
Dividing code into files 90
Generating random numbers in JavaScript 91
Saving the circle position 92
Time for action – saving the circle position 92
Defining a basic class definition in JavaScript 93
Drawing lines in the Canvas 94
Time for action – drawing straight lines between each circle 94
Introducing the line drawing API 96
Using mouse events to interact with objects drawn in the Canvas 97
Time for action – dragging the circles in the Canvas 97
Detecting mouse events in circles in the Canvas 100
Game loop 101
Clearing the Canvas 101
Detecting line intersection in the Canvas 103
Time for action – distinguishing the intersected lines 103
Determining whether two line segments intersect 106
Adding touch support for tablets 108
Time for action – adding the touch input support 108
Handling touches 109
Mouse move and Touch move 110
Summary 110

Chapter 5: Building a Canvas Game's Masterclass 111

Making the Untangle puzzle game 112
Time for action – making the Untangle puzzle game in Canvas 112
Defining the leveling data 118
Determining level-up 118
Displaying the current level and completeness progress 119
Drawing text in the Canvas 119
Time for action – displaying the progress level text inside the canvas element 119
Using embedded web font inside the Canvas 122
Time for action – embedding a Google web font into the canvas element 122
Drawing images in the Canvas 124
Time for action – adding graphics to the game 124
Using the drawImage function 127

Decorating the Canvas-based game 128
Time for action – adding CSS styles and image decoration to the game 129
Animating a sprite sheet in Canvas 131
Time for action – making a game guide animation 131
Creating a multilayer Canvas game 136
Time for action – dividing the game into four layers 136
Mixing a CSS technique with Canvas drawing 142
Summary 143

Chapter 6: Adding Sound Effects to Your Games 145

Adding a sound effect to the Play button 146
Time for action – adding sound effects to the Play button 146
Defining an audio element 149
Playing a sound 151
jQuery's selector versus browser selector 152
Pausing a sound 152
Adjusting the sound volume 152
Using the jQuery hover event 153
File format for WebAudio 153
Building a mini piano musical game 154
Time for action – creating a basic background for the music game 154
Creating scenes in games 156
Creating a slide-in effect in CSS3 157
Visualizing the music playback 158
Time for action – creating the playback visualization in the music game 159
Choosing the right song for the music game 164
Playing audio on mobile devices 165
Storing and extracting the song-level data 165
Getting the elapsed time of the game 166
Creating music dots 167
Moving the music dots 167
Creating a keyboard-driven mini piano musical game 169
Time for action – creating a mini piano musical game 169
Hitting the three music lines by key down 171
Determining music dot hits on key down 172
Removing an element in an array with the given index 172
Adding additional features to the mini piano game 173
Adjusting the music volume according to the player 174
Time for action – removing missed melody notes 174
Removing dots from the game 176

Storing the success count in the last five results 176
Recording music notes as level data 176
Time for action – adding functionalities to record the music level data 177
Adding touch support 179
Time for action – indicating a game over event in the console 179
Handling the audio event in playback complete events 180
Time for action – indicating a game over event in the console 180
Handling audio events 181
Summary 182

Chapter 7: Saving the Game's Progress 183

Storing data using HTML5 local storage 184
Creating a game over dialog 185
Time for action – creating a game over dialog with the elapsed played time 185
Saving scores in the browser 188
Time for action – saving the game score 188
Storing and loading data with local storage 190
The local storage saves the string value 191
Treating the local storage object as an associative array 192
Saving objects in the local storage 192
Time for action – saving the time alongside the score 192
Getting the current date and time in JavaScript 195
Using the native JSON to encode an object into a string 195
Loading a stored object from a JSON string 196
Inspecting the local storage in a console window 197
Notifying players when they break a new record with a nice ribbon effect 198
Time for action – creating a ribbon in CSS3 199
Saving the entire game progress 202
Time for action – saving all essential game data in the local storage 202
Removing a record from the local storage 205
Cloning an array in JavaScript 205
Resuming the game progress 206
Time for action – resuming a game from the local storage 206
Caching the game for offline access 209
Time for action – adding the AppCache Manifest 209
The AppCache file 210
Summary 211

Chapter 8: Building a Multiplayer Draw-and-Guess Game with WebSockets **213**

Installing a WebSocket server **214**
Installing the Node.js WebSocket server 214
Time for action – installing Node.js **215**
Creating a WebSocket server to send connection count 216
Time for action – running a WebSocket server **216**
Initializing the WebSocket server 217
Listening to the connection event on the server side 217
Creating a client that connects to a WebSocket server and getting the total connections count 217
Time for action – showing the connection count in a WebSocket application **218**
Establishing a WebSocket connection 220
WebSocket client events 220
Sending a message to all connected browsers 220
Time for action – sending total count to all users **220**
Defining class and instant instance methods 223
Handling a newly connected user 223
Exporting modules 223
Sending messages to the client 223
Building a chatting application with WebSockets **223**
Sending a message to the server 224
Time for action – sending a message to the server through WebSockets **224**
Sending a message from the client to the server 226
Receiving a message on the server side 226
Sending every received message on the server side to create a chat room **226**
Time for action – sending messages to all connected browsers **226**
Comparing WebSockets with polling approaches 228
Making a shared drawing whiteboard with Canvas and WebSockets **230**
Building a local drawing sketchpad 230
Time for action – making a local drawing whiteboard with the Canvas **230**
Sending the drawing to all the connected browsers 233
Time for action – sending the drawing through WebSockets **233**
Defining a data object to communicate between the client and the server 237
Packing the drawing lines data into JSON for sending 237
Recreating the drawing lines after receiving them from other clients 238
Building a multiplayer draw-and-guess game **238**
Time for action – building the draw-and-guess game **238**
Inheriting the Room class 245

Controlling the game flow of a multiplayer game 246
Room and Game Room 247
Improving the game 247
Improving the styles 247
Storing drawn lines on each game 247
Improving the answer checking mechanism 248
Summary 248

Chapter 9: Building a Physics Car Game with Box2D and Canvas 249
Installing the Box2D JavaScript library 250
Time for action – installing the Box2D physics library 251
Using b2World to create a new world 253
Setting the gravity of the world 254
Setting Box2D to ignore the sleeping object 254
Creating a static ground body in the physics world 254
Time for action – creating a ground in the world 254
Pixel per meter 255
Creating a shape with a fixture 256
Creating a body 256
Setting the bouncing effect with the restitution property 256
Drawing the physics world in the Canvas 257
Time for action – drawing the physics world into the canvas 257
Creating a dynamic box in the physics world 259
Time for action – putting a dynamic box in the world 259
Advancing the world time 260
Time for action – setting up the world step loop 260
Adding wheels to the game 261
Time for action – putting two circles in the world 262
Creating a physical car 263
Time for action – connecting the box and two circles with a revolute joint 263
Using a revolute joint to create an anchor point between two bodies 265
Adding force to the car with a keyboard input 266
Time for action – adding force to the car 266
Applying force to a body 267
Clearing Force 267
Understanding the difference between ApplyForce and ApplyImpulse 267
Adding ramps to our game environment 268
Time for action – creating the world with ramps 268
Checking collisions in the Box2D world 269

Time for action – checking a collision between the car and the destination body **270**
Getting the collision contact list 271
Restarting the game **272**
Time for action – restarting the game while pressing the R key **272**
Adding a level support to our car game **274**
Time for action – loading the game with levels data **274**
Replacing the Box2D outline drawing with graphics **278**
Time for action – adding a flag graphic and a car graphic to the game **278**
Using userData in shape and body 281
Drawing graphics in every frame according to the state of its physics body 282
Rotating and translating an image in the canvas 282
Adding a final touch to make the game fun to play **283**
Time for action – decorating the game and adding a fuel limitation **284**
Adding fuel to add a constraint when applying force 289
Presenting the remaining fuel in a CSS3 progress bar 289
Adding touch support for tablets **290**
Time for action – adding touch support **290**
Summary **294**

Chapter 10: Deploying HTML5 Games **295**

Preparing the deploying materials **296**
Putting the game on the Web **296**
Hosting the node.js server **296**
Deploying as a mobile web app in the home screen **297**
Time for action – adding a meta tag for a mobile web app **297**
Building an HTML5 game into a Mac OS X app **298**
Time for action—putting the HTML5 games into a Mac app **298**
Building an HTML5 game into a mobile app with the Web View **306**
Building with the PhoneGap build **307**
App store's reviewing process **308**
Summary **308**

Appendix: Pop Quiz Answers **311**

Index **315**

Preface

HTML5 promises to be the hot new platform for online games. HTML5 games work on computers, smartphones, tablets, iPhones, and iPads. Be one of the first developers to build HTML5 games today and be ready for tomorrow!

This book will show you how to use the latest HTML5 and CSS3 web standards to build card games, drawing games, physics games, and even multiplayer games over the network. With this book, you will build six example games with clear systematic tutorials.

HTML5, CSS3, and the related JavaScript API are the latest hot topic in the Web. These standards bring us the new game market of HTML5 games. With the new power from them, we can design games with HTML5 elements, CSS3 properties, and JavaScript to play in browsers.

The book is divided into 10 chapters with each one focusing on one topic. While building the six games in the book, you will learn how to draw game objects, animate them, add audio, connect players, and build physics game with the Box2D physics engine.

What this book covers

Chapter 1, *Introducing HTML5 Games*, introduces the new features of HTML5, CSS3, and the related JavaScript API. It demonstrates what games we can make with these features and their benefits.

Chapter 2, *Getting Started with DOM-based Game Development*, kickstarts the game development journey by creating a traditional Ping Pong game in DOM and jQuery.

Chapter 3, *Building a Card-matching Game in CSS3*, walks you through the new features of CSS3 and discusses how we can create a memory card-matching game in DOM and CSS3.

Chapter 4, *Building the Untangle Game with Canvas and the Drawing API*, introduces a new way to draw things and interact with them in a web page with the new canvas element. This also demonstrates how to handle dragging on touch devices.

Chapter 5, *Building a Canvas Game's Masterclass*, extends the Untangle game to show how we can draw gradients and images in the Canvas. It also discusses sprite sheet animations and multilayer management.

Chapter 6, *Adding Sound Effects to Your Games*, adds sound effects and background music to the game by using the Audio element. It discusses the audio format capability among web browsers and creates a keyboard-driven music game by the end of the chapter.

Chapter 7, *Saving the Game's Progress*, extends the CSS3 memory-matching game to demonstrate how we can use the Local Storage API to store and resume game progress and records the best scores.

Chapter 8, *Building a Multiplayer Draw-and-Guess Game with WebSockets*, discusses the WebSockets API that allows browsers to establish persistent connection with the socket server. This allows multiple players to play the game together in real time. A draw-and-guess game is created at the end of the chapter.

Chapter 9, *Building a Physics Car Game with Box2D and Canvas*, teaches you how to integrate a famous physics engine, Box2D, into our canvas games. It discusses how to create physics bodies, apply force, connect them together, associate graphics with the physics, and finally create a platform car game.

Chapter 10, *Deploying HTML5 Games*, shares the different ways in which we can publish our games. It discusses wrapping the web into a native app for Apple's App Store.

Appendix, *Pop Quiz Answers*, gives the answers to the pop quiz questions in each of the chapters.

What you need for this book

You need the latest modern web browsers, a good text editor, and a basic knowledge of HTML, CSS, and JavaScript. In *Chapter 8*, *Building a Multiplayer Draw-and-Guess Game with WebSockets*, we need the Node.js server, which we will help you to install in that chapter.

Who this book is for

This book is for web designers who have a basic knowledge of HTML, CSS, and JavaScript and want to create Canvas or DOM-based games that run on browsers.

Sections

In this book, you will find several headings that appear frequently (Time for action, What just happened?, Pop quiz, and Have a go hero).

To give clear instructions on how to complete a procedure or task, we use these sections as follows:

Time for action – heading

1. Action 1
2. Action 2
3. Action 3

Instructions often need some extra explanation to ensure they make sense, so they are followed with these sections:

What just happened?

This section explains the working of the tasks or instructions that you have just completed.

You will also find some other learning aids in the book, for example:

Pop quiz – heading

These are short multiple-choice questions intended to help you test your own understanding.

Have a go hero – heading

These are practical challenges that give you ideas to experiment with what you have learned.

Conventions

You will also find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Open the `index.html` file in the code editor."

A block of code is set as follows:

```
var matchingGame = {};
matchingGame.deck = [
  'cardAK', 'cardAK',
  'cardAQ', 'cardAQ',
  'cardAJ', 'cardAJ',
  'cardBK', 'cardBK',
  'cardBQ', 'cardBQ',
  'cardBJ', 'cardBJ',
];
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
$(function(){
  matchingGame.deck.sort(shuffle);

  for(var i=0;i<11;i++){
    $(".card:first-child").clone().appendTo("#cards");
  }
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "In MAC, click on the **Get the code** tab and you will see the following screenshot; this shows a guide on how to embed this font into our web page."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

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

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

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

Customer support

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

Downloading the example code

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

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/7770OS_ColoredImages.pdf`.

Errata

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

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

Piracy

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

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

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

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to

1
Introducing HTML5 Games

Hypertext Markup Language, HTML, has been shaping the Internet in the last few decades. It defines how content is structured in the Web and the linkage between related pages. HTML has kept evolving from version 2 to HTML 4, and later to XHTML 1.1. Thanks to the web applications and social networking applications, it the era of HTML5 now.

***Cascading Style Sheet (CSS)** defines how web pages are presented visually. It styles all HTML elements and the styles of their states, such as hover and active.*

JavaScript is the logic controller of a web page. It makes the web page dynamic and provides client-side interaction between the page and users. It accesses the HTML through **Document Object Model** *(***DOM***). It controls the new HTML features via their APIs.*

There are modern web browsers in most desktop and mobile devices. These latest web techniques bring us the new game market—the HTML5 games. With the new power from these techniques, we can design games with HTML5 elements, CSS3 properties, and JavaScript to play in most browsers and mobile devices.

In this chapter, we will cover the following topics:

- Discovering new features in HTML5
- Discussing what makes us so excited around HTML5 and CSS3
- Previewing what games we are going to build in later chapters
- Preparing the development environment

So, let's get started.

Discovering new features in HTML5

There are many new things introduced in HTML5 and CSS3. Before getting our hands dirty by creating the games, let's take an overview of the new features and see how we can use them to create games.

Canvas

Canvas is an HTML5 element that provides drawing shapes and bitmap manipulation functions in low levels. We can imagine the Canvas element as a dynamic image tag. The traditional `<img>` tag shows a static image. This image is usually static after it's loaded. We can change the `<img>` tag to another image source or apply styles to the image, but we cannot modify the image's bitmap context itself.

On the other hand, Canvas is like a client-side dynamic `<img>` tag. We can load images inside it, draw shapes there, and interact with it using JavaScript.

Canvas plays an important role in HTML5 game development. It is one of our main focus areas in this book.

Audio

Background music and sound effects are essential elements in game design. HTML5 comes with native audio support from the `audio` tag. Thanks to this feature, we do not require the proprietary Flash Player to play sound effects in our HTML5 games. However, there have been some restrictions on using Web Audio on the Web. We will discuss the usage of the `audio` tag in *Chapter 6*, *Adding Sound Effects to Your Games*.

Touch Events

Besides the traditional keyboard and mouse events, there are touch events that we can use to handle single and multi-touch events. We can design a game that works on mobile devices with touches. We can also handle gestures by observing the touch patterns.

GeoLocation

GeoLocation lets the web page retrieve the latitude and longitude of the user's computer. For example, Google's Ingress game makes use of GeoLocation to let players play the game in their real city. This feature may not have been so useful years ago when everyone was using the Internet with their desktop. There are not many things for which we need the accurate location of the road of the user. We can get the rough location by analyzing the IP address.

These days, more and more users are going on the Internet with their powerful smartphones. Webkit and other modern mobile browsers are in everyone's pocket. GeoLocation lets us design mobile applications and games to play with the inputs of a location.

WebGL

WebGL extends the Canvas element by providing a set of 3D graphics APIs in the web browser. The APIs follow the standard of OpenGL ES 2.0. WebGL provides a powerful GPG-accelerated, 3D rendering API for HTML5 games. Some 3D game engines support the export of WebGL, including the popular Unity engine. We can expect to see more HTML5 3D games waiting to be released using WebGL.

The techniques used to create games with WebGL are quite different than using Canvas. Creating games in WebGL requires handing the 3D models and using an API similar to OpenGL. Therefore, we will not discuss WebGL game development in this book.

WebGL has a better performance than 2D Canvas because of the GPU-rendering support. Some libraries allow a game to use Canvas 2D drawing API, and the tools render the canvas by drawing on WebGL to gain performance. Pixi.js (`http://www.pixijs.com`), EaselJS (`http://blog.createjs.com/webgl-support-easeljs/`) and WebGL-2D (`https://github.com/corbanbrook/webgl-2d`) are several such tools among them.

WebSocket

WebSocket is part of the HTML5 spec to connect the web page to a socket server. It provides us with a persistent connection between the browser and server. This means that the client does not need to poll the server for new data within short periods. The server will push updates to the browsers whenever there is any data to update. One benefit of this feature is that game players can interact with each other in almost real time. When one player does something and sends data to the server, we can send the individual player the update to create one-on-one real-time page play, or we can iterate all the connections in the server to send an event to every other connected browser to acknowledge what the player just did. This creates the possibility of building multiplayer HTML5 games.

Local storage

HTML5 provides a persistent data storage solution to web browsers.

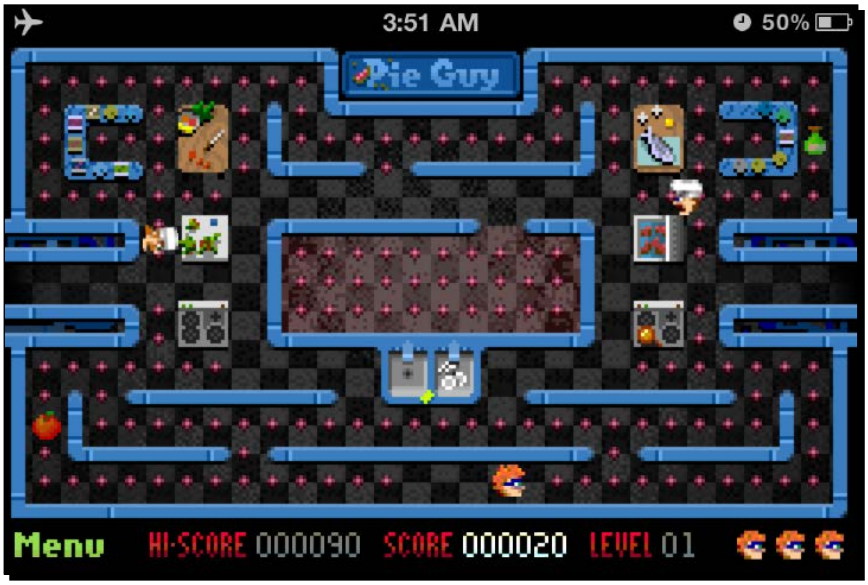
Local Storage stores key-value pair data persistently. The data is still there after the browser terminates. Moreover, the data is not limited to be accessible only to the browsers that created it. It is available to all browser instances with the same domain. Thanks to Local Storage, we can easily save a game's status, such as progress and earned achievements, locally in web browsers.

Another database on web browser is IndexedDB. It's key-value pair too, but it allows storing objects and querying data with condition.

Offline applications

Normally, we need an Internet connection to browse web pages. Sometimes, we can browse cached offline web pages. These cached offline web pages usually expire quickly. With the next offline application introduced by HTML5, we can declare our cache manifest. This is a list of files that will be stored for later access when there is no Internet connection.

With the cache manifest, we can store all the game graphics, game control JavaScript files, CSS stylesheets, and the HTML files locally. We can also pack our HTML5 games as offline games on the desktop or mobile devices. Players can play the games even in the airplane mode. The following screenshot from the Pie Guy game (`http://mrgan.com/pieguy`) shows an HTML5 game being played on an iPhone without an Internet connection; note the little airplane symbol indicating the offline status:



Discovering new features in CSS3

CSS is the presentation layer and HTML is the content layer. It defines how the HTML looks. We cannot miss CSS when we create games with HTML5, especially DOM-based games. We may purely use JavaScript to create and style the games with a Canvas element. However, we need CSS when we create DOM-based HTML5 games. Therefore, let's take a look at what is new in CSS3 and how we can use the new properties to create games.

Instead of directly drawing and interacting on Canvas' drawing board, new CSS3 properties let us animate the DOM in different ways. This makes it possible to make more complicated DOM-based browser games.

CSS3 transition

Traditionally, the style changes immediately when we apply a new style to an element. CSS3 transition renders in-between styles during the style changes of the target elements over duration. For example, here, we have a blue box and want to change it to dark blue when we do a mouseover. We can do this by using the following code snippets:

HTML:

```
<a href="#" class="box"></a>
```

CSS:

```
a.box {
  display: block;
  width: 100px;
  height: 100px;
  background: blue;
}
a.box:hover {
  background: darkblue;
}
```

The box changes to dark blue immediately when we do a mouseover. With CSS3 transition applied, we can tween the styles for a specific duration and easing value:

```
a.box {
  transition: all 0.5s ease-out;
}
```

Downloading the example code

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

In the past, we needed JavaScript to calculate and render the in-between styles; this is much slower than using CSS3 transition because the browser natively makes the effects happen.

Since some CSS3 specifications are still in the draft stage and not yet fixed, implementation from different browser vendors may have some minor differences to the W3C spec. Therefore, browser vendors tend to implement their CSS3 properties with a vendor prefix to prevent conflict.

Safari uses the `-webkit-` prefix. Opera uses the `-o-` prefix. Firefox uses the `-moz-` prefix and IE uses the `-ms-` prefix. Chrome used to use `-webkit-`, but now it doesn't use any prefix after switching its engine to Blink. It is a little complex now to declare a CSS3 property, such as flex, with several lines of the same rule for several browsers. We can expect the prefix to be dropped after the property spec is fixed.

In order to make the code cleaner in this book, I will use non-vendor prefix for all the properties in this book. I recommend you to use JavaScript-based libraries to automatically add the required vendor prefix for different web browsers. The prefix-free library (`http://leaverou.github.io/prefixfree/`) is one of them.

Alternatively, if you are using preprocessors, the compilation process may also add the necessary vendor prefix for you.

CSS3 transform

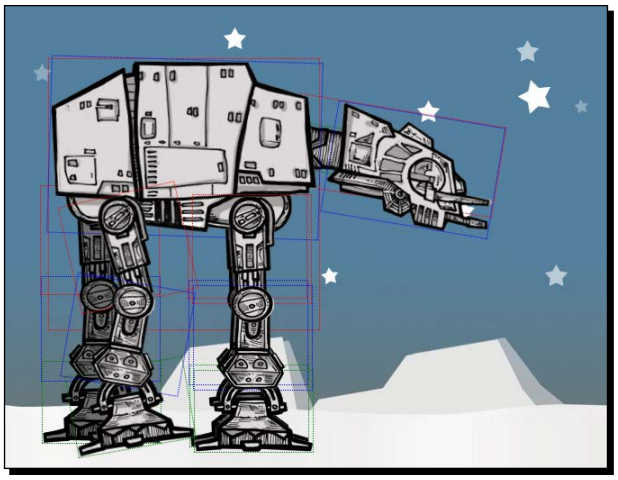
CSS3 transform lets us scale the elements, rotate them, and translate their position. CSS3 transform is divided into 2D and 3D. By combining the transform origin and 3D rotation and translation, we can animate 2D graphics in a 3D space.

CSS3 animation

CSS3 transition is one type of animation. It declares the tweening animation between two styles of the elements.

CSS3 animation is one step further in animation. We can define key frames of an animation. Each key frame contains a set of properties that should change at any particular moment. It is like a set of CSS3 transitions that are applied in sequence to the target element.

The AT-AT Walker (`http://anthonycalzadilla.com/css3-ATAT/index-bones.html`) shows a nice demo on creating a skeleton bone animation with CSS3 animation key frames, transform, and transition. This is shown in the following diagram:



The benefit of creating HTML5 games

We have explored several new key features from HTML5 and CSS3. With these features, we can create HTML5 games on browsers. But why do we need to do that? What is the benefit of creating HTML5 games?

Free and open standards

The web standards are open and free for use. In contrast, third-party tools are usually proprietary and they cost money. With proprietary technologies, the support from them may drop because of changes to the company's focus. The standardization and openness of HTML5 ensures that we will have browsers that support it.

Support for multiple platforms

With the built-in support of all the HTML5 features in modern browsers, we do not require the users to preinstall any third-party plugin in order to play any file. These plugins are not standard. They usually require an extra plugin installation that you may not be able to install. For instance, millions of Apple iOS devices around the world do not support third-party plugins, such as Flash Player, in their mobile Safari. Despite whatever the reason might be, Apple does not allow Flash Player to run on their Mobile Safaris, instead, HTML5 and the related web standard are what they get in their browsers. We can reach this user base by creating HTML5 games that are optimized for mobiles.

Native app-rendering performance in particular scenarios

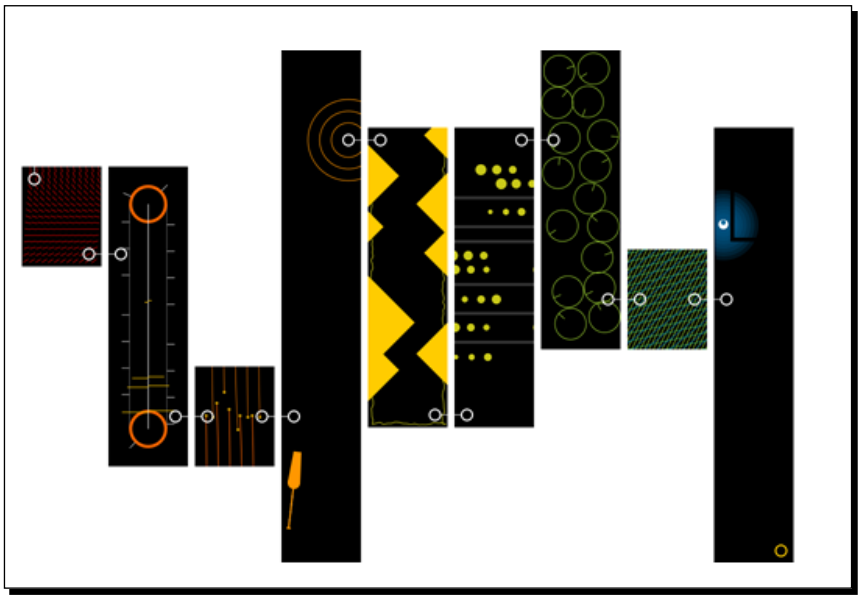
When we code the game in a Canvas, there are some rendering engines that can translate our Canvas drawing code into OpenGL, thus rendering in native mobile device. This means that while we are still coding the game for a web browser, our game can gain benefits in mobile devices by the native app OpenGL rendering. **Ejecta** (`http://impactjs.com/ejecta`) and **CocoonJS** (`http://ludei.com/cocoonjs`) are two such engines.

Breaking the boundary of usual browser games

In traditional game designing, we build games within a boundary box. We play video games on a television. We play Flash games in web browsers with a rectangle boundary.

Using creativity, we are not bound in a rectangle game stage any more. We can have fun with all the page elements.

Twitch (`http://reas.com/twitch/`) is a game from Chrome Experiments. It is a collection of mini games where the player has to carry the ball from the starting point to the end point. The fun part is that each mini game is a small browser window. When the ball reaches the destination point of that mini game, it is transferred into the newly created mini game browser to continue the journey. The following screenshot shows the whole map of Twitch with the individual web browsers:



Building HTML5 games

Thanks to the new features of HTML5 and CSS3, we can now create an entire game in the browser. We can control every element in the DOM. We can animate each document object with CSS3. We have Canvas to dynamically draw things and interact with them. We have an audio element to handle the background music and sound effects. We also have Local Storage to save game data, and WebSocket to create a real-time multiplayer game. Most modern browsers are already supporting these features. It is now time to build HTML5 games.

What others are playing with HTML5

This is a good opportunity to study how different HTML5 games perform by watching other HTML5 games that are made with different techniques.

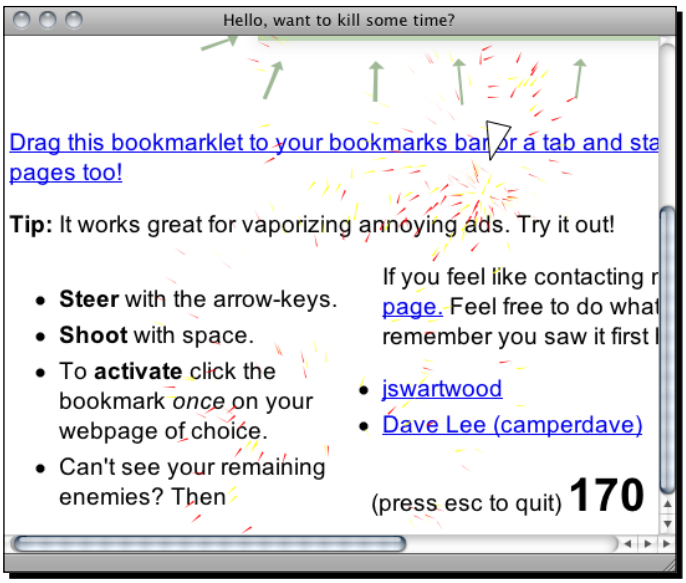
Coca-Cola's Ahh campaign

Coca-Cola had run a campaign known as **Ahh** (`http://ahh.com`) with lots of interactive mini games. The interactions combined several techniques that included canvas and device rotation. Most of them work well in both desktop and mobile devices.

Asteroid-styled bookmarklet

Erik, a web designer from Sweden, created an interesting bookmarklet. This is an asteroid-styled game for any web page. Yes, any web page! It shows an abnormal way to interact with any web page. It creates a plane on the website you are reading from. You can then fly the plane using arrow keys and fire bullets using the space bar. The fun part is that the bullets will destroy the HTML elements on the page. Your goal is to destroy all the things on the web page that you choose. This bookmarklet is another example of breaking the boundary of usual browser games. It tells us that we can think outside the box while designing HTML5 games.

The following screenshot shows the plane destroying the contents on the web page:



The bookmarklet is available for installation at `http://kickassapp.com`. You can even design the space ship that you control.

X-Type

The creator of a Canvas-based game engine named Impact, created this X-Type (`http://phoboslab.org/xtype/`) shooting game for different platforms, including web browsers, iOS, and Wii U. The following screenshot shows the game running smoothly in iPhone.

Cursors.io

Cursors.io (`http://cursors.io`) demonstrates a nicely designed real-time multiplayer game. Every user controls an anonymous mouse cursor and takes a journey through the levels of the game by moving the cursor to the green exit. The fun part of the game is that players must help the others to advance to the level. There are toggles that some cursors click on them to unlock the doors. The anonymous players must take up the role to help the others. Someone will take your role so that you can advance to the next level. The more players that help you, the higher your chance is to succeed in the game. In case only a few players are playing and you can't experience the game, I have recorded my playing screen in 12 x speed (at `http://vimeo.com/109414542`) to let you have a glimpse of how this multiplayer game works. This has been captured in the following screenshot: