Preface

In the summer of 2001, after 15 years of developing graphical user interfaces and graphics-intensive applications, I read a best-selling book about implementing web applications by someone I did not know—Jason Hunter—but whom, unbeknownst to me, would soon become a good friend on the No Fluff Just Stuff (NFJS) tour.

When I finished Jason's Servlets book,¹ I put it in my lap and stared out the window. After years of Smalltalk, C++, and Java, and after writing a passionate 1622 pages for Graphic Java Swing,² I thought to myself, *am I really going to implement user interfaces with print statements that generate HTML?* Unfortunately, I was.

From then on, I soldiered on through what I consider the Dark Ages of software development. I was the second Apache Struts committer and I invented the Struts Template Library, which ultimately became the popular Tiles project. I spent more than six years on the JavaServer Faces (JSF) Expert Group, spoke about server-side Java at more than 120 NFJS symposiums and many other conferences, and coauthored a book on JSF.³ I got excited about Google Web Toolkit and Ruby on Rails for a while, but in the end the Dark Ages was mostly concerned with the dull business of presenting forms to users on the client and processing them on the server, and I was never again able to capture that passion that I had for graphics and graphical user interfaces.

In the summer of 2010, with HTML5 beginning its inexorable rise in popularity, I came across an article about Canvas, and I knew salvation was nigh. I immediately dropped everything in my professional life and devoted myself fulltime to write the best Canvas book that I could. From then on, until the book was finalized in March 2012, I was entirely immersed in Canvas and in this book. It's by far the most fun I've ever had writing a book.

Canvas gives you all the graphics horsepower you need to implement everything from word processors to video games. And, although performance varies on specific platforms, in general, Canvas is fast, most notably on iOS5, which hardware accelerates Canvas in Mobile Safari. Browser vendors have also

^{1.} *Java Servlet Programming*, 2001, by Jason Hunter with William Crawford, published by O'Reilly.

^{2.} *Graphic Java* 2, *Volume* 2, *Swing*, 1999 by David Geary, published by Prentice Hall.

^{3.} Core JavaServer Faces, 2004, by David Geary and Cay Horstmann, published by Prentice Hall.

done a great job adhering to the specification so that well-written Canvas applications run unmodified in any HTML5-compliant browser with only minor incompatibilities.

HTML5 is the Renaissance that comes after the Dark Ages of software development, and Canvas is arguably the most exciting aspect of HTML5. In this book I dive deeply into Canvas and related aspects of HTML5, such as the Animation Timing specification, to implement real-world applications that run across desktop browsers and mobile devices.

Reading This Book

I wrote this book so that in the Zen tradition you can read it without reading.

I write each chapter over the course of months, constantly iterating over material without ever writing a word. During that time I work on outlines, code listings, screenshots, tables, diagrams, itemized lists, notes, tips, and cautions. Those things, which I refer to as scaffolding, are the most important aspects of this book. The words, which I write only at the last possible moment after the scaffolding is complete, are meant to provide context and illustrate highlights of the surrounding scaffolding. Then I iterate over the words, eliminating as many of them as I can.

By focusing on scaffolding and being frugal with words, this book is easy to read without reading. You can skim the material, concentrating on the screenshots, code listings, diagrams, tables, and other scaffolding to learn a great deal of what you need to know on any given topic. Feel free to consider the words as second-class citizens, and, if you wish, consult them only as necessary.

An Overview of This Book

This book has two parts. The first part, which spans the first four chapters of the book and is nearly one half of the book, covers the Canvas API, showing you how to draw shapes and text into a canvas, and draw and manipulate images. The last seven chapters of the book show you how to use that API to implement animations and animated sprites, create physics simulations, detect collisions, and develop video games. The book ends with a chapter on implementing custom controls, such as progress bars, sliders, and image panners, and a chapter that shows you how to create Canvas-based mobile applications.

The first chapter—*Essentials*—introduces the canvas element and shows you how to use it in web applications. The chapter contains a short section on getting

started with HTML5 development in general, briefly covering browsers, consoles, debuggers, profilers, and timelines. The chapter then shows you how to implement Canvas essentials: drawing into a canvas, saving and restoring Canvas parameters and the drawing surface itself, printing a canvas, and an introduction to offscreen canvases. The chapter concludes with a brief math primer covering basic algebra, trigonometry, vector mathematics, and deriving equations from units of measure.

The second chapter—*Drawing*—which is the longest chapter in the book, provides an in-depth examination of drawing with the Canvas API, showing you how to draw lines, arcs, curves, circles, rectangles, and arbitrary polygons in a canvas, and how to fill them with solid colors, gradients, and patterns. The chapter goes beyond the mere mechanics of drawing, however, by showing you how to implement useful, real-world examples of drawing with the Canvas API, such as drawing temporary rubber bands to dynamically create shapes, dragging shapes within a canvas, implementing a simple retained-mode graphics subsystem that keeps track of polygons in a canvas so users users can edit them, and using the clipping region to erase shapes without disturbing the Canvas background underneath.

The third chapter—*Text*—shows you how to draw and manipulate text in a canvas. You will see how to stroke and fill text, set font properties, and position text within a canvas. The chapter also shows you how to implement your own text controls in a canvas, complete with blinking text cursors and editable paragraphs.

The fourth chapter—*Images and Video*—focuses on images, image manipulation, and video processing. You'll see how to draw and scale images in a canvas, and you'll learn how to manipulate images by accessing the color components of each pixel. You will also see more uses for the clipping region and how to animate images. The chapter then addresses security and performance considerations, before ending with a section on video processing.

The fifth chapter—Animation—shows you how to implement smooth animations with a method named requestAnimationFrame() that's defined in a W3C specification titled *Timing control for script-based animations*. You will see how to calculate an animation's frame rate and how to schedule other activities, such as updating an animation's user interface at alternate frame rates. The chapter shows you how to restore the background during an animation with three different strategies and discusses the performance implications of each. The chapter also illustrates how to implement time-based motion, scroll an animation's background, use parallax to create the illusion of 3D, and detect and react to user gestures during an animation. The chapter concludes with a look at timed animations and the implementation of a simple animation timer, followed by a discussion of animation best practices.

The sixth chapter—*Sprites*—shows you how to implement sprites (animated objects) in JavaScript. Sprites have a visual representation, often an image, and you can move them around in a canvas and cycle through a set of images to animate them. Sprites are the fundamental building block upon which games are built.

The seventh chapter—*Physics*—shows you how to simulate physics in your animations, from modeling falling objects and projectile trajectories to swinging pendulums. The chapter also shows you how to warp both time and motion in your animations to simulate real-world movement, such as the acceleration experienced by a sprinter out of the blocks (ease-in effect) or the deceleration of a braking automobile (ease-out).

Another essential aspect of most games is collision detection, so the eighth chapter in the book—*Collision Detection*—is devoted to the science of detecting collisions between sprites. The chapter begins with simple collision detection using bounding boxes and circles, which is easy to implement but not very reliable. Because simple collision detection is not reliable under many circumstances, much of this chapter is devoted to the Separating Axis Theorem, which is one of the best ways to detect collisions between arbitrary polygons in both 2D and 3D; however, the theorem is not for the mathematically faint of heart, so this chapter goes to great lengths to present the theorem in layman terms.

The ninth chapter—*Game Development*—begins with the implementation of a simple but effective game engine that provides support for everything from drawing sprites and maintaining high scores to time-based motion and multitrack sound. The chapter then discusses two games. The first game is a simple Hello World type of game that illustrates how to use the game engine and provides a convenient starting point for a game. It also shows you how to implement common aspects of most games such as asset management, heads-up displays, and a user interface for high scores. The second game is an industrial-strength pinball game that draws on much of the previous material in the book and illustrates complex collision detection in a real-world game.

Many Canvas-based applications require custom controls, so the 10th chapter—*Custom Controls*—teaches you how to implement them. The chapter discusses implementing custom controls in general and then illustrates those techniques with four custom controls: a rounded rectangle, a progress bar, a slider, and an image panner.

The final chapter of this book—*Mobile*—focuses on implementing Canvas-based mobile applications. You'll see how to control the size of your application's viewport so that your application displays properly on mobile devices, and how to account for different screen sizes and orientations with CSS3 media queries.

You'll also see how to make your Canvas-based applications indistinguishable from native applications on iOS5 by making them run fullscreen and fitting them with desktop icons and startup screens. The chapter concludes with the implementation of a keyboard for iOS5 applications that do not receive text through a text field.

Prerequisites

To make effective use of this book you must have more than a passing familiarity with JavaScript, HTML, and CSS. I assume, for example, that you already know how to implement objects with JavaScript's prototypal inheritance, and that you are well versed in web application development in general.

This book also utilizes some mathematics that you may have learned a long time ago and forgotten, such as basic algebra and trigonometry, vector math, and deriving equations from units of measure. At the end of the first chapter you will find a short primer that covers all those topics.

The Book's Code

All the code in this book is copyrighted by the author and is available for use under the license distributed with the code. That license is a modified MIT license that lets you do anything you want with the code, including using it in software that you sell; however, you may not use the code to create educational material, such as books, instructional videos, or presentations. See the license that comes with the code for more details.

When implementing the examples, I made a conscious decision to keep comments in code listings to a bare minimum. Instead, I made the code itself as readable as possible; methods average about 5 lines of code so they are easy to understand.

I also adhered closely to Douglas Crawford's recommendations in his excellent book *JavaScript*, *The Good Parts*.⁴ For example, all function-scoped variables are always declared at the top of the function, variables are declared on a line of their own, and I always use === and its ilk for equality testing.

Finally, all the code listings in this book are color coded. Function calls are displayed in blue, so they stand out from the rest of the listing. As you scan listings, pay particular attention to the blue function calls; after all, function calls are the verbs of JavaScript, and those verbs alone reveal most of what you need to know about the inner workings of any particular example.

^{4.} JavaScript, The Good Parts, 2008, by Douglas Crawford, published by O'Reilly.

The Future of Canvas and This Book

The HTML5 APIs are constantly evolving, and much of that evolution consists of new features. The Canvas specification is no exception; in fact, this book was just days from going to the printer when the WHATWG Canvas specification was updated to include several new features:

- An ellipse() method that creates elliptical paths
- Two methods, getLineDash() and setLineDash(), and an attribute lineDashOffset used for drawing dashed lines
- An expanded TextMetrics object that lets you determine the exact bounding box for text
- A Path object
- A CanvasDrawingStyles object
- Extensive support for hit regions

At that time, no browsers supported the new features, so it was not yet possible to write code to test them.

Prior the March 26, 2012 update to the specification, you could draw arcs and circles with Canvas, but there was no explicit provision for drawing ellipses. Now, in addition to arcs and circles, you can draw ellipses with the new ellipse() method of the Canvas 2d context. Likewise, the context now explicitly supports drawing dashed lines.

The TextMetrics object initially only reported one metric: the width of a string. However, with the March 26, 2012 update to the specification, you can now determine both the width and height of the rectangle taken up by a string in a canvas. That augmentation of the TextMetrics object will make it much easier, and more efficient, to implement Canvas-based text controls.

In addition to ellipses and an improved TextMetrics object, the updated specification has also added Path and CanvasDrawingStyles methods. Prior to the updated specification, there was no explicit mechanism for storing paths or drawing styles. Now, not only are there objects that represent those abstractions, but many of the Canvas 2d context methods have been duplicated to also take a Path object. For example, you stroke a context's path by invoking context.stroke(), which strokes the *current* path; however, the context now has a method stroke(Path) and that method strokes the path you send to the method instead of the context's current path. When you modify a path with Path methods such as addText(), you can specify a CanvasDrawingStyle object, which is used by the path, in this case to add text to the path.

The updated specification contains extensive support for hit regions. A hit region is defined by a path, and you can associate an optional mouse cursor and accessibility parameters, such as an Accessible Rich Internet Application (ARIA) role and a label, with a hit region. A single canvas can have multiple hit regions. Among other things, hit regions will make it easier and more efficient to implement collision detection and improve accessiblity.

Finally, both the WHATWG and W3C specifications have included two Canvas context methods for accessibility, so that applications can draw focus rings around the current path, letting users navigate with the keyboard in a Canvas. That functionality was not part of the March 26, 2012 update to the specification, and in fact, has been in the specification for some time; however, while the book was being written, no browser vendors supported the feature, so it is not covered in this book.

As the Canvas specification evolves and browser vendors implement new features, this book will be updated on a regular basis. In the meantime, you can read about new Canvas features and preview the coverage of those features in the next edition of this book, at corehtml5canvas.com.

The Companion Website

This book's companion website is http://corehtml5canvas.com, where you can download the book's code, run featured examples from the book, and find other HTML5 and Canvas resources.