In this chapter, we’ll explore what you can do with the Canvas API—a cool API that enables you to dynamically generate and render graphics, charts, images, and animation. We’ll walk you through using the basics of the rendering API to create a drawing that can scale and adjust to the browser environment. We’ll show you how to create dynamic pictures based on user input in a heatmap display. Of course, we’ll also alert you to the pitfalls of using Canvas and share tricks to overcome them.

This chapter presumes only a minimal amount of graphics expertise, so don’t be afraid to jump in and try out one of the most powerful features of HTML5.

Overview of HTML5 Canvas

An entire book could be written about the use of the Canvas API (and it wouldn’t be a small book). Because we have only a chapter, we’re going to cover (what we think is) the most commonly used functionality in this very extensive API.

History

The canvas concept was originally introduced by Apple to be used in Mac OS X WebKit to create dashboard widgets. Before the arrival of canvas, you could only use drawing APIs in a browser through plug-ins such as Adobe plug-ins for Flash and Scalable Vector Graphics (SVG), Vector Markup Language (VML) only in Internet Explorer, or other clever JavaScript hacks.

Try, for example, to draw a simple diagonal line without a canvas element—it sounds easy, but it is a fairly complex task if you do not have a simple two-dimensional drawing API at your disposal. Canvas provides just that, and because it is an extremely useful thing to have in the browser, it was added to the HTML5 specification.

Early on, Apple hinted at possibly reserving the intellectual property rights in the WHATWG draft of the canvas specification, which caused concern at the time among some followers of web standardization. In the end, however, Apple disclosed the patents under the W3C’s royalty-free patent licensing terms.

SVG versus Canvas

Peter says: “Canvas is essentially a bitmap canvas, and as such images that are drawn on a canvas are final and cannot be resized in the way that Scalable Vector Graphic (SVG) images can. Furthermore, objects drawn on a canvas are not part of the page’s DOM or part of any namespace—something that is considered a weakness if you need hit detection or object-based updates. SVG images, on the other hand
can be scaled seamlessly at different resolutions and allow for hit detection (knowing precisely where an image is clicked).

Why then, would the WHATWG HTML5 specification not use SVG exclusively? Despite its obvious shortcomings, the HTML5 Canvas API has two things going for it: it performs well because it does not have to store objects for every primitive it draws, and it is relatively easy to implement the Canvas API based on many of the popular two-dimensional drawing APIs found in other programming languages. Ultimately, it is better to have one bird in the hand than two in the bush.”

What Is a Canvas?

When you use a canvas element in your web page, it creates a rectangular area on the page. By default, this rectangular area is 300 pixels wide and 150 pixels high, but you can specify the exact size and set other attributes for your canvas element. Listing 2-1 shows the most basic canvas element that can be added to an HTML page.

Listing 2-1. A Basic Canvas Element

```html
<canvas></canvas>
```

Once you have added a canvas element to your page, you can use JavaScript to manipulate it any way you want. You can add graphics, lines, and text to it; you can draw on it; and you can even add advanced animations to it.

The Canvas API supports the same two-dimensional drawing operations that most modern operating systems and frameworks support. If you have ever programmed two-dimensional graphics in recent years, you will probably feel right at home with the Canvas API because it is designed to be similar to existing systems. If you haven’t, you’re about to discover how much more powerful a rendering system can be than the previous images and CSS tricks developers have used for years to create web graphics.

To programmatically use a canvas, you have to first get its context. You can then perform actions on the context and finally apply those actions to the context. You can think of making canvas modifications as similar to database transactions: you start a transaction, perform certain actions, and then commit the transaction.

Canvas Coordinates

As shown in Figure 2-1, coordinates in a canvas start at x=0, y=0 in the upper-left corner—which we will refer to as the origin—and increase (in pixels) horizontally over the x-axis and vertically over the y-axis.