Getting Started with HTML5 Canvas

graphics.cs.wisc.edu/WP/tutorials/getting-started-with-html5-canvas/

Why do I need to write my own Canvas tutorial for CS559?

There are some good tutorials out there (I have a few on alist, but that's just a start). I recommend that you use them AFTER this one. Before you read this one, you might want to read my post on what Canvas is and why we use it.

The problem with those other tutorials is that they all assume that you know a certain set of stuff from prior tutorials. You should be able to figure this out yourself, but in case you need to be walked through it…

You should start with the explanation of what I mean by Canvas

Then we can start…

To draw, you need to answer three questions:

1. Where do I draw?
2. When do I draw?
3. What do I draw?

But since you can't do any without the other, we'll look at them all together.

The simplest possible program

JS Bin on jsbin.com

Notice that this is an HTML file – you can load it into your web browser. I am using a spiffy thing to embed it on this web page so you can see the code and the output side by side. You can edit the code and see what the changes do. You might want to move the divider between the code and the output so the lines don't wrap.

You should recognize some of the basic HTML stuff. This page is really minimal – it has no "<HEAD>" section (which is where I would set the title and other useful stuff). This is bad, since some of the stuff in the HEAD is important (like declaring the character set). In practice, you might not want to make your pages so minimal. I wanted to keep it simple here (so it fits on the screen). Later, I'll show you a more realistic example.

In the body, you'll see there are two tags.

The first is the canvas tag. This creates a blank rectangular space on the web page. The size is set by the width and height. I have given it a name (or "id") of "myCanvas" – this is important, since I will need to refer to it later.

The second thing in the body is a "script" tag – inside this script tag, I put my javascript code. Technically, I probably should have had the script tag say javascript – but the web browser will figure it out.

The script is simple. It finds the canvas (that I made above) by looking up its name. It then gets the drawing context from this canvas. The drawing context is the object that we use to actually do the drawing – it stores all the information about drawing state.

Aside… I could have done this differently. Rather than naming the canvas and searching for it, I could have searched for the first (or last) canvas on the web page – but this is bad practice, since it makes assumptions about the rest of the web page. In fact, I could have just assumed there wasn't a canvas and added one to the end (or beginning) of the web page right in the JavaScript code. This is less desirable because it makes it hard to mix the canvas with other stuff on the page.

Aside… This code works and is simple, but in practice is way too simple. For example, there is no error checking. When writing a JavaScript program you need to worry about errors. Also sticking the program right in the html becomes unwieldy when the program is more than a few lines long.

Then, finally, I can actually do some drawing. Notice that I need to start a path before I can do anything. Then I add a rectangle to the path. Then I fill inside the path.

Besides the fact that the picture I drew is really boring (a black square), there are some over-simplifications in this example that you should understand before trying to draw more.

Where do I draw?

There are two parts to the where question.

The first part of the question is the space we'll actually be drawing on. This is a canvas element on a web page. To be able to create this, you'll need to know enough about web pages to make one that you can put a space on, and then actually put the canvas tag someplace.

Web pages usually give a lot more information about themselves, so that the browser doesn't have to guess. It's probably a good idea to do some of that.

You can put other web-page stuff around the canvas element – this is useful for making titles, or putting controls (like sliders), or even documentation.

Once you start drawing, you'll need to understand the coordinate system of the canvas (so when you position things, you'll know where things will go). For canvas, this is simple: by default, things are measured in pixels, the origin in the upper left corner, and the positive Y axis goes down.

When do I draw?

Or… where do I put my code?

If you notice, I placed my program code right in the web page within a script tag. This is OK for a tiny program like this, but you can imagine that quite quickly you'll want to put stuff elsewhere to keep your code organized as it grows bigger.

There is another issue as to when this code actually gets executed. It would be nice to assume that everything works all linearly. The script tag gets processed after the canvas tag is done being processed, so the code inside the script tag only gets run once the canvas has been created and is ready to go. In practice, web browsers are more complicated than that. Especially since something can modify the canvas later on (for example, to relocate it to a different place on the page, or put a border around it). So, in a real program, you want to do something smarter about having your code run only when things are really ready.

I've written a separate tutorial about this. I recommend that you read it. Either right now, or after you finish this tutorial.

The real answer can be complicated, because it gets to questions of how to best organize your program.

But here's a more realistic version of that same minimal program

HTML Test on jsbin.com

Here, I've turned off the output pane (you can turn it back on to see that the program draws the same back rectangle).

If you read the "When do I Draw" tutorial, this should make sense to you (since it's one of the examples). If not, what's happening is I am putting my code inside of a function that gets called when the window is done loading. But the important thing is that the drawing commands (begin path, …) happen at an appropriate time. This is a more realistic program for you to start with. I suggest that you take this and start tinkering with the stuff that actually does the drawing…

What do I Draw?

Now the fun part… we can draw stuff. Preferably something more fun than a black rectangle.

OK, here's something that's just a little bit better.

HTML Test on jsbin.com

Notice that drawing has a few parts (that I do over and over)…

1. I start a path.
2. I add shapes to the path, either as a whole shape (the rectangle) or by moving a "pen"

around (the MoveTo and LineTo that draws the triangle). If the shape is closed, I make sure to close it.
3. I decide how I want that shape to appear. You can see I set the fill color, the stroke color, and the width of the stroke. I could set other things (like making the lines dashed, or patterns to fill the shapes).
4. Then I draw it (using the currently set style parameters). I can either fill it, or stroke it.

You'll notice that I refer to colors in a funny way – as a string beginning with a # (hash mark) and a hexidecimal number following it. This is explained in another tutorial on color.

There are a bunch of different basic shapes (primitives) you can draw. Look at another tutorial to get a more complete list. There are a bunch of different style things you can do. Again, look at some other tutorial. But at this point, you can probably have enough of the basics so that you can start drawing stuff.

If you haven't done it already, now might be a good time to go back to the "when do I draw" tutorial, so you can get some ideas on JavaScript code organization idioms.

There is a resource list for more Canvas info as part of the "Canvas: What and Why" tutorial.