

An Introduction to `<canvas>`

Rob Larsen

2013.11.6

Download (<https://github.com/roblarsen/rob-larsen-presentations>)

What we're going to talk about

- About me
- <canvas>
- Canvas Libraries

Me



Work

- I've been making web sites since 1997
- I've been primarily focused on HTML, CSS & JavaScript that whole time
- Formerly an agency guy/consultant. Lots of big brand stuff (Adidas, Motorola, Samsung, etc.)
- Nowadays I'm the client & I come with 100% more suits

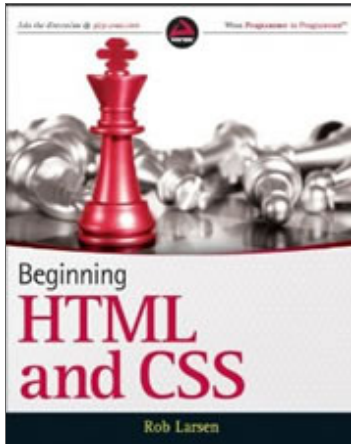
Seriously, **suits**



Institutional investment management is serious business

Work

- I'm [roblarsen](#) on Github
- I'm [@roblarsenwww](#) (web stuff) [@robreact](#) (everything else) on Twitter
- I have a blog @ [htmlcssjavascript.com](#)
- I wrote this book. I'm working on another one. Best question today gets a free copy of my book!



Art



DrunkenFist.com and the freshly launched **Java+++**

What's Canvas?

The `<canvas>` element and associated API started life as an Apple extension to HTML. From there it blossomed into one of the early stars of the HTML5 era.

The `<canvas>` element provides a scriptable interface for drawing two-dimensional images in the browser. Think... dynamic PNGs. Even without full browser support on the desktop, developers have embraced canvas fully.

What's Canvas?

It's been used for everything from high traffic visualizations to game engines, a system for delivering custom fonts, and a port of the Processing programming language into JavaScript.

Control

It's an extremely low level API. This means you often have to do a bit of work but you have complete, pixel level control over the image.

That's Both Good and Bad

Full control and a low level API sometimes you have to do things that are just silly.

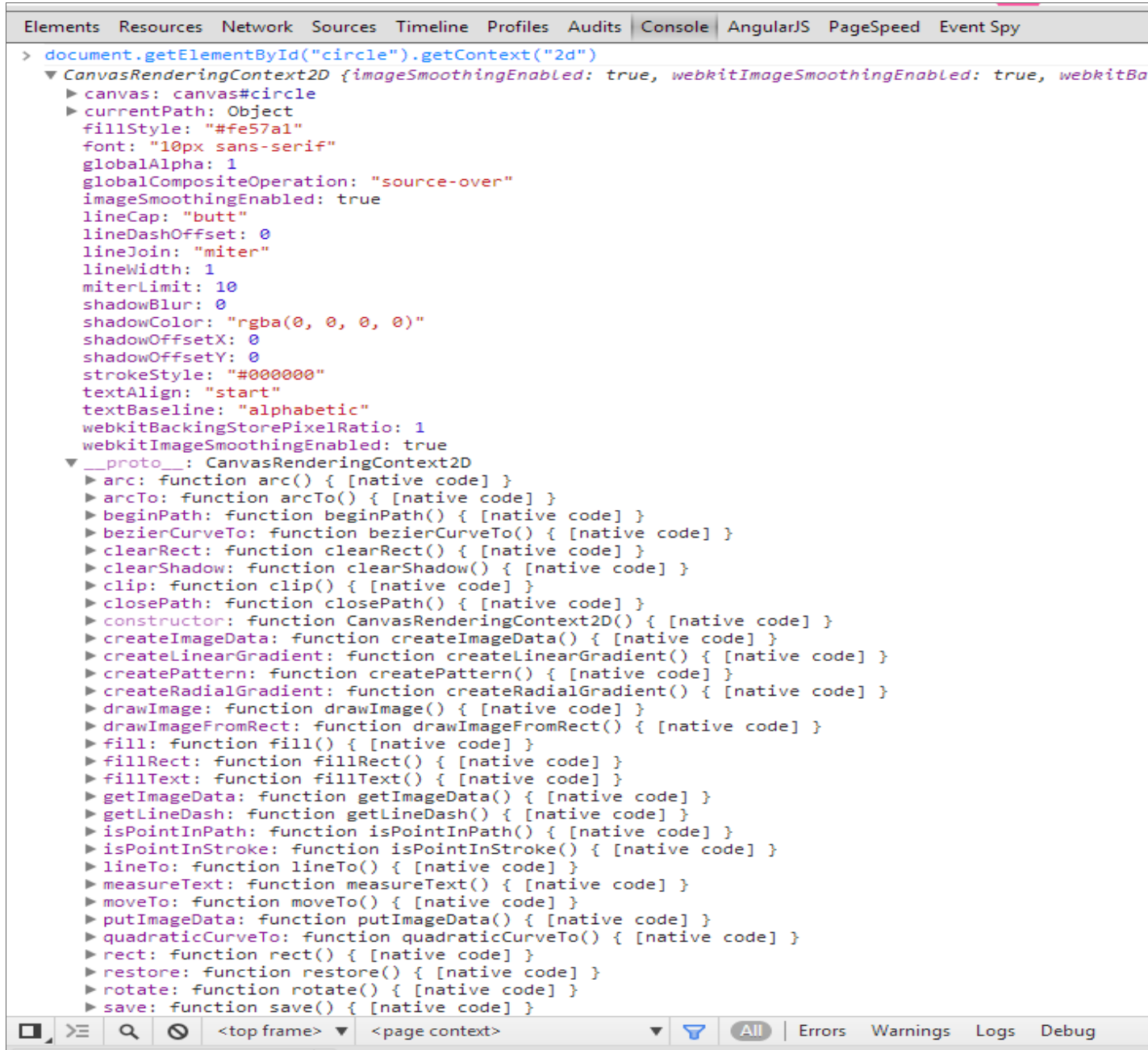
In Action

The simplest part of working with canvas is the `<canvas>` element itself. A `<canvas>` element operates much like any other replaced element like `<video>` or ``. The big difference is that it lacks a `src` attribute. The “src” of the canvas image is provided by JavaScript.

In Action

The JavaScript is just series of commands that will manipulate the canvas in different ways. It begins with a `context` which is a reference to canvas. the context provides methods to interact with the canvas and properties about the canvas.

The Context

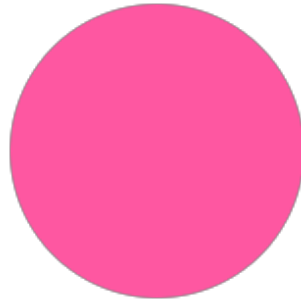


The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The breadcrumb at the top reads 'Elements > Resources > Network > Sources > Timeline > Profiles > Audits > Console > AngularJS > PageSpeed > Event Spy'. The console log shows the command `document.getElementById("circle").getContext("2d")` and its return value, a `CanvasRenderingContext2D` object. The object's prototype is `CanvasRenderingContext2D`. The console output is as follows:

```
> document.getElementById("circle").getContext("2d")
▼ CanvasRenderingContext2D {imageSmoothingEnabled: true, webkitImageSmoothingEnabled: true, webkitBa
  ► canvas: canvas#circle
  ► currentPath: Object
  fillStyle: "#fe57a1"
  font: "10px sans-serif"
  globalAlpha: 1
  globalCompositeOperation: "source-over"
  imageSmoothingEnabled: true
  lineCap: "butt"
  lineDashOffset: 0
  lineJoin: "miter"
  lineWidth: 1
  miterLimit: 10
  shadowBlur: 0
  shadowColor: "rgba(0, 0, 0, 0)"
  shadowOffsetX: 0
  shadowOffsetY: 0
  strokeStyle: "#000000"
  textAlign: "start"
  textBaseline: "alphabetic"
  webkitBackingStorePixelRatio: 1
  webkitImageSmoothingEnabled: true
  __proto__: CanvasRenderingContext2D
  ► arc: function arc() { [native code] }
  ► arcTo: function arcTo() { [native code] }
  ► beginPath: function beginPath() { [native code] }
  ► bezierCurveTo: function bezierCurveTo() { [native code] }
  ► clearRect: function clearRect() { [native code] }
  ► clearShadow: function clearShadow() { [native code] }
  ► clip: function clip() { [native code] }
  ► closePath: function closePath() { [native code] }
  ► constructor: function CanvasRenderingContext2D() { [native code] }
  ► createImageData: function createImageData() { [native code] }
  ► createLinearGradient: function createLinearGradient() { [native code] }
  ► createPattern: function createPattern() { [native code] }
  ► createRadialGradient: function createRadialGradient() { [native code] }
  ► drawImage: function drawImage() { [native code] }
  ► drawImageFromRect: function drawImageFromRect() { [native code] }
  ► fill: function fill() { [native code] }
  ► fillRect: function fillRect() { [native code] }
  ► fillText: function fillText() { [native code] }
  ► getImageData: function getImageData() { [native code] }
  ► getLineDash: function getLineDash() { [native code] }
  ► isPointInPath: function isPointInPath() { [native code] }
  ► isPointInStroke: function isPointInStroke() { [native code] }
  ► lineTo: function lineTo() { [native code] }
  ► measureText: function measureText() { [native code] }
  ► moveTo: function moveTo() { [native code] }
  ► putImageData: function putImageData() { [native code] }
  ► quadraticCurveTo: function quadraticCurveTo() { [native code] }
  ► rect: function rect() { [native code] }
  ► restore: function restore() { [native code] }
  ► rotate: function rotate() { [native code] }
  ► save: function save() { [native code] }
```

The bottom of the console shows the status bar with icons for expand, search, and mute, followed by the text '< top frame>' and '< page context>'. On the right, there are filters for 'All', 'Errors', 'Warnings', 'Logs', and 'Debug'.

The element looks like this



Viewing Source

```
<canvas id="circle" width="800" height="300"></canvas>
```

```
var ctx = document.getElementById( "circle" ).getContext( "2d" );
ctx.beginPath();
ctx.arc( 400, 150, 75, 0, Math.PI*2, true );
ctx.fillStyle = "#fe57a1";
ctx.closePath();
ctx.fill();
```

Did I say low level API?

I sure did. Everyone knows `arc` is all you need to draw a circle when you have `Math.PI*2`

```
ctx.arc( 400, 150, 75, 0, Math.PI*2, true );
```

It's not the friendliest API. The arguments map to "center x", "center y", "radius", "start angle, in radians", "end angle, in radians" and "clockwise, as a Boolean"

Support

5.5 6 7 8 9 10 11

2 3 3.5 3.6 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27

4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32

3.1 3.2 4 5 5.1 6 6.1 7

9 9.5-9.6 10.0-10.1 10.5 10.6 11 11.1 11.5 11.6 12 12.1 12.5 15 16 17 18

3.2 4.0-4.1 4.2-4.3 5.0-5.1 6.0-6.1 7

5.0-7.0

2.1 2.2 2.3 3 4 4.1 4.2 4.3 4.4

7 10

0

0

Canvas Strengths

Canvas is great for applications and visualization that require intense animation, real-time interaction and/or many "elements." Other common animated elements like SVG elements or DIVs are DOM elements so working with them is kind of scary. Thousands of "elements" in the canvas is still just the one element being manipulated pixel by pixel.

Canvas Strengths

- Games
- Real time visualizations
- Image manipulation

In Practice

Let's make some stuff with canvas.

Core Canvas

“This specification defines the 2D Context, Level 2 for the HTML canvas element. The 2D Context provides objects, methods, and properties to draw and manipulate graphics on a canvas drawing surface.”

View Source

```
function draw( data ) {  
    ctx.fillRect( 0, 0, 900, 300 );  
    var len = data.length,  
        width = 900/len,  
        gradient = ctx.createLinearGradient( 0, 0, 0, 300 );  
    gradient.addColorStop( ".5", "#003366" );  
    gradient.addColorStop( "1.0", "#ccff00" );  
    ctx.fillStyle = gradient;  
    for ( var i=0; i<len; i++ ) {  
        ctx.fillRect( i, 300, width, -data[i] );  
    }  
}
```

Core Canvas +-

- Pluses:
 - Not a library
 - POWERFUL
- Minuses:
 - Lower level API

Canvas: Cee.js

“Cee.js is a small helper library for the Canvas 2d API. The goal is to extend and enhance the basic API while still remaining familiar”

I'm partially to blame (also, yay **@bobholt** and **@marcneuwirth**)

The image shows the text 'cee.js' in a highly stylized, 3D font. The letters are orange with a thick blue outline. Each letter has a blue shadow cast beneath it, giving it a three-dimensional appearance. The font is playful and rounded, with the 'c's being particularly large and the 'j' having a small dot. The text is centered within a white rectangular frame.

<http://roblarsen.github.io/CeeJS/>

View Source

```
function draw(data) {
  ctx.reset();
  var len = data.length,
      width = 900/len,
      fill;
  for ( var i=0; i<len; i = i + 10 ) {
    var dataLen = data[i];
    for ( var j = 0; j < dataLen; j = j+10 ){
      fill = "rgb("+j+", "+(255-j)+" ,255)";
      ctx.fillCircle({
        x : i,
        y : 300 - j,
        radius : 4,
        fillStyle : fill
      });
    }
  }
}
```

Canvas: Cee.js +-

- Pluses:
 - Familiar API, just enhanced
 - Tiny (4k gzipped)
 - It's got a really nice personality
 - Chaining
- Minuses:
 - Full of youthful exuberance
 - Remains low level
 - Not a fancy game developer tool

Canvas: JavaScript InfoVis Toolkit

*“The JavaScript InfoVis Toolkit provides tools for creating
Interactive Data Visualizations for the Web.”*

View Source

```
icicle = new $jit.Icicle({
  injectInto : 'infovis',
  animate : animate,
  offset : 1,
  cushion : false,
  constrained : true,
  levelsToShow : 4,
  Tips : {
    enable : true,
    type : 'Native',
    offsetX : 20,
    offsetY : 20,
    onShow : function(tip, node){
      var count = 0;
      node.eachSubnode(function(){
        count++;
      });
      tip.innerHTML = "<div class=\"tip-title\"><b>Name:</b> "
        + node.name + "</div><div class=\"tip-text\">" + count
        + " children</div>";
    }
  },
  Events: {
    enable : true,
    onClick : function( node ){
      if ( node ) {
        icicle.tips.hide();
        icicle.enter( node );
      }
    }
  }
});
```



Canvas:JIT +-

- Pluses:
 - Strong foundation
 - Straightforward API
- Minuses:
 - Out of the box visualizations are skeletal

Canvas: Other Libraries

- **sigma.js** "an open-source lightweight JavaScript library to draw graphs, using the HTML canvas"
- **paper.js** "Scriptographer ported to JavaScript and browser, using HTML5 Canvas"
- **Peity** "progressive <canvas> piecharts"
- **arbor.js** "a graph visualization library using web workers and jQuery"
- **envision** "a library for creating fast, dynamic and interactive HTML5 visualizations."
- **CanvasQuery** "use HTML5 Canvas with jQuery syntax"
- **Processing.js** "the sister project of the popular Processing visual programming language"

Canvas Reference

MSDN

MDN

Mark Pilgrim's tutorial

Thanks!

[roblarsen](#) on Github | [@robreact](#) on Twitter
blog @ htmlcssjavascript.com