

HTML5 Canvas

Notes for Professionals

Chapter 1: Getting started with HTML5 Canvas

Section 1.1: Detecting mouse position on the canvas

This example will show how to get the mouse position relative to the canvas, such that (0,0) is the top-left corner of the HTML5 Canvas. The `onmousemove` and `onclick` will get the mouse position on the document, to change this to be based on the top of the canvas we subtract the left and top canvas from the client X and Y.

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "10px Arial";

canvas.addEventListener("mousemove", function(e) {
    var clientX = e.clientX;
    var clientY = e.clientY;
    // Get the canvas position
    var canvasRect = canvas.getBoundingClientRect();
    // Subtract the left and top canvas from the client X and Y
    ctx.moveTo(clientX - canvasRect.left, clientY - canvasRect.top);
    ctx.fillText("X", clientX - canvasRect.left, clientY - canvasRect.top);
});
```

Runnable Example

The use of `Math.round` is due to ensure the X,Y positions are integers, as the browser does not have integer positions.

Section 1.2: Canvas size and resolution

The size of a canvas is the area it occupies on the page and is defined by the CSS `width` and `height` properties.

```
canvas {
    width: 1000px;
    height: 1000px;
}
```

The canvas resolution defines the number of pixels it contains. The resolution width and height properties. If not specified the canvas defaults to 300 by 150.

The following canvas will use the above CSS style but as the width and height are 300 by 150.

```
<canvas id="myCanvas" width="300" height="150"></canvas>
```

This will result in each pixel being stretched unevenly. The pixel aspect ratio is broader will use bilinear filtering. This has an effect of blurring out pixels.

For the best results when using the canvas ensure that the canvas resolution is the same pixel count as the style defines.

```
<canvas id="myCanvas" width="1000" height="1000"></canvas>
```

Section 1.3: Rotate

The `rotate(r)` method of the 2D context rotates the canvas by the specified amount `r` of radians around the origin.

HTML

```
<canvas id="canvas" width=240 height=240 style="background-color:#000000;"></canvas>
<button type="button" onclick="rotate_ctx()">Rotate context</button>
```

JavaScript

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var ox = canvas.width / 2;
var oy = canvas.height / 2;
ctx.font = "10px serif";
ctx.textAlign = "center";
ctx.fillStyle = "white";
ctx.fillText("Hello World", ox, oy);

rotate_ctx = function() {
    // translate so that the origin is now (ox, oy) the center of the canvas
    ctx.translate(ox, oy);
    // convert degrees to radians with radians = (Math.PI/180)*degrees
    ctx.rotate(Math.PI / 180 * 45);
    ctx.fillText("Hello World", 0, 0);
    // translate back
    ctx.translate(-ox, -oy);
}
```

Live Demo on JSFiddle

Section 1.4: Save canvas to image file

You can save a canvas to an image file by using the method `canvas.toDataURL()`, that returns the data URI for the canvas image data.

The method can take two optional parameters `canvas.toDataURL([type], [encoderOptions])` type is the image format (if omitted the default is image/png); encoderOptions is a number between 0 and 1 indicating image quality (default is 0.92).

Here we draw a canvas and attach the canvas data URI to the "Download to myImage.jpg" link:

HTML

```
<canvas id="canvas" width=240 height=240 style="background-color:#000000;"></canvas>
Download to myImage.jpg</img>
```

JavaScript

```
var canvas = document.getElementById("canvas");
```

HTML5 Canvas Notes for Professionals

Chapter 9: Media types and the canvas

Section 9.1: Basic loading and playing a video on the canvas

The canvas can be used to display video from a variety of sources. This example shows how to load a video as a file resource, display it and add a simple click on screen play/pause toggle.

This stackoverflow self answered question [How do I display a video using HTML5 canvas tag](#) shows the following example code in action.

Just an image

A video is just an image as far as the canvas is concerned. You can draw it like any image. The difference being the video can play and has sound.

Get canvas and basic setup

It is assumed you know how to add a canvas and correctly size it.

```
var canvas = document.getElementById("myCanvas"); // get the canvas from the page
var videoContainer = // object to hold video and associated info
```

Creating and loading the video

```
var video = document.createElement("video"); // create a video element
video.src = "urlToVideo.mp4"; // the video will now begin to load
// As some additional info is needed we will place the video in a videoContainer object for convenience
video.loop = false; // ensure that the video does not auto play
videoContainer = { // we will add properties as needed
    video: video,
    ready: false;
};
```

Unlike images, videos don't have to be fully loaded to be displayed on the canvas. Videos also provide a host of extra events that can be used to monitor status of the video.

In this case we wish to know when the video is ready to play. `oncanplay` means that enough of the video has loaded to play some of it, but there may not be enough to play to the end.

```
video.oncanplay = readyToPlayVideo; // let the event to the play function that can be found below
```

Alternatively you can use `onplay` which will fire when enough of the video has loaded so that it can be played to the end.

```
video.onplay = readyToPlayVideo; // let the event to the play function that can be found below
```

Only use one of the `oncanplay` or `onplay` events not both.

The can play event (equivalent to image onload)

```
function readyToPlayVideo(event) { // this is a reference to the video
    // the video may not match the canvas size so find a scale to fit
    videoContainer.scale = Math.min(
        canvas.width / this.videoWidth,
```

HTML5 Canvas Notes for Professionals

109

100+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with HTML5 Canvas	2
Section 1.1: Detecting mouse position on the canvas	2
Section 1.2: Canvas size and resolution	2
Section 1.3: Rotate	3
Section 1.4: Save canvas to image file	3
Section 1.5: How to add the Html5 Canvas Element to a webpage	4
Section 1.6: An index to Html5 Canvas Capabilities & Uses	5
Section 1.7: Off screen canvas	6
Section 1.8: Hello World	6
Chapter 2: Text	8
Section 2.1: Justified text	8
Section 2.2: Justified paragraphs	13
Section 2.3: Rendering text along an arc	17
Section 2.4: Text on curve, cubic and quadratic beziers	22
Section 2.5: Drawing Text	25
Section 2.6: Formatting Text	26
Section 2.7: Wrapping text into paragraphs	27
Section 2.8: Draw text paragraphs into irregular shapes	28
Section 2.9: Fill text with an image	30
Chapter 3: Polygons	31
Section 3.1: Render a rounded polygon	31
Section 3.2: Stars	32
Section 3.3: Regular Polygon	33
Chapter 4: Images	35
Section 4.1: Is "context.drawImage" not displaying the image on the Canvas?	35
Section 4.2: The Tained canvas	35
Section 4.3: Image cropping using canvas	36
Section 4.4: Scaling image to fit or fill	36
Chapter 5: Path (Syntax only)	39
Section 5.1: createPattern (creates a path styling object)	39
Section 5.2: stroke (a path command)	41
Section 5.3: fill (a path command)	45
Section 5.4: clip (a path command)	45
Section 5.5: Overview of the basic path drawing commands: lines and curves	47
Section 5.6: lineTo (a path command)	49
Section 5.7: arc (a path command)	50
Section 5.8: quadraticCurveTo (a path command)	52
Section 5.9: bezierCurveTo (a path command)	53
Section 5.10: arcTo (a path command)	54
Section 5.11: rect (a path command)	55
Section 5.12: closePath (a path command)	57
Section 5.13: beginPath (a path command)	58
Section 5.14: lineCap (a path styling attribute)	61
Section 5.15: lineJoin (a path styling attribute)	62
Section 5.16: strokeStyle (a path styling attribute)	63
Section 5.17: fillStyle (a path styling attribute)	65

Section 5.18: lineWidth (A path styling attribute)	67
Section 5.19: shadowColor, shadowBlur, shadowOffsetX, shadowOffsetY (path styling attributes)	68
Section 5.20: createLinearGradient (creates a path styling object)	70
Section 5.21: createRadialGradient (creates a path styling object)	73
Chapter 6: Paths	77
Section 6.1: Ellipse	77
Section 6.2: Line without blurriness	78
Chapter 7: Navigating along a Path	80
Section 7.1: Find point on curve	80
Section 7.2: Finding extent of Quadratic Curve	81
Section 7.3: Finding points along a cubic Bezier curve	82
Section 7.4: Finding points along a quadratic curve	83
Section 7.5: Finding points along a line	84
Section 7.6: Finding points along an entire Path containing curves and lines	84
Section 7.7: Split bezier curves at position	91
Section 7.8: Trim bezier curve	94
Section 7.9: Length of a Cubic Bezier Curve (a close approximation)	96
Section 7.10: Length of a Quadratic Curve	97
Chapter 8: Dragging Path Shapes & Images on Canvas	98
Section 8.1: How shapes & images REALLY(!) "move" on the Canvas	98
Section 8.2: Dragging circles & rectangles around the Canvas	99
Section 8.3: Dragging irregular shapes around the Canvas	103
Section 8.4: Dragging images around the Canvas	106
Chapter 9: Media types and the canvas	109
Section 9.1: Basic loading and playing a video on the canvas	109
Section 9.2: Capture canvas and Save as webM video	111
Section 9.3: Drawing an svg image	116
Section 9.4: Loading and displaying an Image	117
Chapter 10: Animation	119
Section 10.1: Use requestAnimationFrame() NOT setInterval() for animation loops	119
Section 10.2: Animate an image across the Canvas	120
Section 10.3: Set frame rate using requestAnimationFrame	121
Section 10.4: Easing using Robert Penners equations	121
Section 10.5: Animate at a specified interval (add a new rectangle every 1 second)	125
Section 10.6: Animate at a specified time (an animated clock)	126
Section 10.7: Don't draw animations in your event handlers (a simple sketch app)	127
Section 10.8: Simple animation with 2D context and requestAnimationFrame	129
Section 10.9: Animate from [x0,y0] to [x1,y1]	129
Chapter 11: Collisions and Intersections	131
Section 11.1: Are 2 circles colliding?	131
Section 11.2: Are 2 rectangles colliding?	131
Section 11.3: Are a circle and rectangle colliding?	131
Section 11.4: Are 2 line segments intercepting?	131
Section 11.5: Are a line segment and circle colliding?	133
Section 11.6: Are line segment and rectangle colliding?	133
Section 11.7: Are 2 convex polygons colliding?	134
Section 11.8: Are 2 polygons colliding? (both concave and convex polys are allowed)	135
Section 11.9: Is an X,Y point inside an arc?	136
Section 11.10: Is an X,Y point inside a wedge?	137
Section 11.11: Is an X,Y point inside a circle?	138

Section 11.12: Is an X,Y point inside a rectangle?	138
Chapter 12: Clearing the screen	139
Section 12.1: Rectangles	139
Section 12.2: Clear canvas with gradient	139
Section 12.3: Clear canvas using composite operation	139
Section 12.4: Raw image data	140
Section 12.5: Complex shapes	140
Chapter 13: Responsive Design	141
Section 13.1: Creating a responsive full page canvas	141
Section 13.2: Mouse coordinates after resizing (or scrolling)	141
Section 13.3: Responsive canvas animations without resize events	142
Chapter 14: Shadows	144
Section 14.1: Sticker effect using shadows	144
Section 14.2: How to stop further shadowing	145
Section 14.3: Shadowing is computationally expensive -- Cache that shadow!	145
Section 14.4: Add visual depth with shadows	146
Section 14.5: Inner shadows	146
Chapter 15: Charts & Diagrams	151
Section 15.1: Pie Chart with Demo	151
Section 15.2: Line with arrowheads	152
Section 15.3: Cubic & Quadratic Bezier curve with arrowheads	153
Section 15.4: Wedge	154
Section 15.5: Arc with both fill and stroke	155
Chapter 16: Transformations	157
Section 16.1: Rotate an Image or Path around it's centerpoint	157
Section 16.2: Drawing many translated, scaled, and rotated images quickly	158
Section 16.3: Introduction to Transformations	159
Section 16.4: A Transformation Matrix to track translated, rotated & scaled shape(s)	160
Chapter 17: Compositing	167
Section 17.1: Draw behind existing shapes with "destination-over"	167
Section 17.2: Erase existing shapes with "destination-out"	167
Section 17.3: Default compositing: New shapes are drawn over Existing shapes	168
Section 17.4: Clip images inside shapes with "destination-in"	168
Section 17.5: Clip images inside shapes with "source-in"	168
Section 17.6: Inner shadows with "source-atop"	169
Section 17.7: Change opacity with "globalAlpha"	169
Section 17.8: Invert or Negate image with "difference"	170
Section 17.9: Black & White with "color"	170
Section 17.10: Increase the color contrast with "saturation"	171
Section 17.11: Sepia FX with "luminosity"	171
Chapter 18: Pixel Manipulation with "getImageData" and "putImageData"	173
Section 18.1: Introduction to "context.getImageData"	173
Credits	175
You may also like	176

About

Please feel free to share this PDF with anyone for free,
latest version of this book can be downloaded from:

<https://goalkicker.com/HTML5CanvasBook>

This *HTML5 Canvas Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official HTML5 Canvas group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to web@petercv.com

[Click here to download full PDF material](#)