HTML5 Canvas

What is Canvas?

The HTML5 canvas element can be used to draw graphics on the webpage via JavaScript. The canvas was originally introduced by Apple for the Mac OS dashboard widgets and to power graphics in the Safari web browser. Later it was adopted by the Firefox, Google Chrome and Opera. Now the canvas is a part of the new HTML5 specification for next generation web technologies.

By default the `<canvas>` element has 300px of width and 150px of height without any border and content. However, custom width and height can be defined using the CSS `height` and `width` property whereas the border can be applied using the CSS `border` property.

Understanding Canvas Coordinates

The canvas is a two-dimensional rectangular area. The coordinates of the top-left corner of the canvas are (0, 0) which is known as origin, and the coordinates of the bottom-right corner are (*canvas width*, *canvas height*). Here's a simple demonstration of canvas default coordinate system.

(0,0)

Tip: Place your mouse pointer within the canvas area demonstrated above and you will get its current coordinates relative to the canvas. The `<canvas>` element is supported in all major web browsers such as Chrome, Firefox, Safari, Opera, IE 9 and above.

Drawing Path and Shapes on Canvas

In this section we're going to take a closer look at how to draw basic paths and shapes using the newly introduced HTML5 canvas element and JavaScript.

Here is the base template for drawing paths and shapes onto the 2D HTML5 canvas.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing on Canvas</title>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        // draw stuff here
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

All the lines except those from 7 to 11 are pretty straight forward. The anonymous function attached to the `window.onload` event will execute when the page load. Once the page is loaded, we can access the canvas element with `document.getElementById()` method. Later we have defined a 2D canvas context by passing 2d into the `getContext()` method of the canvas object.

Drawing a Line

The most basic path you can draw on canvas is a straight line. The most essential methods used for this purpose are `moveTo()` , `lineTo()` and the `stroke()` .

The `moveTo()` method defines the position of drawing cursor onto the canvas, whereas the `lineTo()` method used to define the coordinates of the line's end point, and finally the `stroke()` method is used to make the line visible. Let's try out an example:

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
```

Drawing a Arc

You can create arcs using the `arc()` method. The syntax of this method is as follow:

```
context.arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise);
```

The JavaScript code in the following example will draw an arc on the canvas.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
```

Drawing a Rectangle

You can create rectangle and square shapes using the `rect()` method. This method requires four parameters x, y position of the rectangle and its width and height.

The basic syntax of the `rect()` method can be given with:

```
context.rect(x, y, width, height);
```

The following JavaScript code will draw a rectangle shape centered on the canvas.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.stroke();
    };
</script>
```

Drawing a Circle

There is no specific method for creating circle like rectangle's `rect()` method. However, you can create a fully enclosed arc such as circle using the `arc()` method.

The syntax for drawing a complete circle using the `arc()` method can be given with:

```
context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
```

The following example will draw a complete circle centered on the canvas.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        context.stroke();
    };
</script>
```

Applying Styles and Colors on Stroke

The default color of the stroke is black and its thickness is one pixel. But, you can set the color and width of the stoke using the `strokeStyle` and `lineWidth` property respectivley.

The following example will draw an orange color line having 5 pixels width.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 5;
        context.strokeStyle = "orange";
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
```

You can also set the cap style for the lines using the `lineCap` property. There are three styles available for the line caps — butt, round, and square. Here's an example:

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 10;
        context.strokeStyle = "orange";
        context.lineCap = "round";
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
```

Filling Colors inside Canvas Shapes

You can also fill color inside the canvas shapes using the `fillStyle()` method.

The following example will show you how to fill a solid color inside a rectangle shape.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.fillStyle = "#FB8B89";
        context.fill();
        context.lineWidth = 5;
        context.strokeStyle = "black";
        context.stroke();
    };
</script>
```

Tip: While styling the shapes on canvas, it is recommended to use the `fill()` method before the `stroke()` method in order to render the stroke correctly.

Similarly, you can use the `fillStyle()` method to fill solid color inside a circle too.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        context.fillStyle = "#FB8B89";
        context.fill();
        context.lineWidth = 5;
        context.strokeStyle = "black";
        context.stroke();
    };
</script>
```

Filling Gradient Colors inside Canvas Shapes

You can also fill gradient color inside the canvas shapes. A gradient is just a smooth visual transition from one color to another. There are two types of gradient available — *linear* and *radial*.

The basic syntax for creating a linear gradient can be given with:

```
var grd = context.createLinearGradient(startX, startY, endX, endY);
```

The following example uses the `createLinearGradient()` method to fill a linear gradient color inside a rectangle. Let's try it out to understand how it basically works:

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        var grd = context.createLinearGradient(0, 0, canvas.width, canvas.height);
        grd.addColorStop(0, '#8ED6FF');
        grd.addColorStop(1, '#004CB3');
        context.fillStyle = grd;
        context.fill();
        context.stroke();
    };
</script>
```

Similarly, you can fill canvas shapes with radial gradient using the `createRadialGradient()` method. The basic syntax for creating a radial gradient can be given with:

```
var grd = context.createRadialGradient(startX, startY, startRadius, endX, endY,
endRadius);
```

The following example uses the `createRadialGradient()` method to fill a radial gradient color inside a circle. Let's try it out to understand how it actually works:

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        var grd = context.createRadialGradient(150, 100, 10, 160, 110, 100);
        grd.addColorStop(0, '#8ED6FF');
        grd.addColorStop(1, '#004CB3');
        context.fillStyle = grd;
        context.fill();
        context.stroke();
    };
</script>
```

Drawing Text on Canvas

You can also draw text onto canvas. These texts can contain any Unicode characters. The following example will draw a simple greeting message "Hello World!" onto a canvas.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font = "bold 32px Arial";
        context.fillText("Hello World!", 50, 100);
    };
</script>
```

You can additionally set the color and alignment of the text on the canvas, like this:

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font = "bold 32px Arial";
        context.textAlign = "center";
        context.textBaseline = "middle";
        context.fillStyle = "orange";
        context.fillText("Hello World!", 150, 100);
    };
</script>
```

You can also apply stroke on text using the `strokeText()` method. This method will color the perimeter of the text instead of filling it. However if you want to set both the fill and stroke on canvas text you can use both the `fillText()` and the `strokeText()` methods together.

Example

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font = "bold 32px Arial";
        context.textAlign = "center";
        context.textBaseline = "middle";
        context.strokeStyle = "orange";
        context.strokeText("Hello World!", 150, 100);
    };
</script>
```

Tip: While styling the text on canvas, it is recommended to use the `fillText()` method before the `strokeText()` method in order to render the stroke correctly.