Unit-3

HTML5 - Canvas

HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

Here is a simple <canvas> element which has only two specific attributes **width** and **height** plus all the core HTML5 attributes like id, name and class, etc.

```
<canvas id = "mycanvas" width = "100" height = "100"></canvas>
```

You can easily find that <canvas> element in the DOM using *getElementById()* method as follows −

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

Let us see a simple example on using <canvas> element in HTML5 document.

```
<!DOCTYPE HTML>

<html>
   <body>
     <canvas id = "mycanvas" width = "100" height = "100"></canvas>
  </body>
</html>
```

The Rendering Context

The <canvas> is initially blank, and to display something, a script first needs to access the rendering context and draw on it.

The canvas element has a DOM method called **getContext**, used to obtain the rendering context and its drawing functions. This function takes one parameter, the type of context**2d**.

Browser Support

The latest versions of Firefox, Safari, Chrome and Opera all support for HTML5 Canvas but IE8 does not support canvas natively.

You can use ExplorerCanvas to have canvas support through Internet Explorer.

HTML5 Canvas Examples

1. HTML5 Canvas - Drawing Rectangles

- **fillRect(x,y,width,height)**

This method draws a filled rectangle.

- **strokeRect(x,y,width,height)**

This method draws a rectangular outline.

- **clearRect(x,y,width,height)**

This method clears the specified area and makes it fully transparent

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The rect() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.rect(20, 20, 150, 100);
ctx.stroke();
</script>
</body>
</html>
```

2. HTML5 Canvas - Drawing Paths

- **beginPath()**

  This method resets the current path.

- **moveTo(x, y)**

This method creates a new subpath with the given point.

- **closePath()**

This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.

- **fill()**

This method fills the subpaths with the current fill style.

- **stroke()**

This method strokes the subpaths with the current stroke style.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The beginPath() Method</h2>
```

```
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.beginPath();
ctx.lineWidth = "5";
ctx.strokeStyle = "green";
ctx.moveTo(0, 75);
ctx.lineTo(250, 75);
ctx.stroke();
</script>
</body>
</html>
```

3. HTML5 Canvas - Drawing Lines

Same as Canvas path

4. HTML5 Canvas - Drawing Bezier Curves

- **bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)**

  This method adds the given point to the current path, connected to the previous one by a cubic Bezier curve with the given control points.

  The x and y parameters in bezierCurveTo() method are the coordinates of the end point. cp1x and cp1y are the coordinates of the first control point, and cp2x and cp2y are the coordinates of the second control point.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The bezierCurveTo() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.beginPath();
ctx.moveTo(20, 20);
ctx.bezierCurveTo(20, 100, 200, 100, 200, 20);
ctx.stroke();
</script>
```

```
</body>
</html>
```

5. HTML5 Canvas - Drawing Quadratic Curves

- **quadraticCurveTo(cpx, cpy, x, y)**

  This method adds the given point to the current path, connected to the previous one by a quadratic Bezier curve with the given control point.

  The x and y parameters in quadraticCurveTo() method are the coordinates of the end point. cpx and cpy are the coordinates of the control point.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The quadraticCurveTo() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.beginPath();
ctx.moveTo(20, 20);
ctx.quadraticCurveTo(20, 100, 200, 20);
ctx.stroke();
</script>
</body>
</html>
```

6. HTML5 Canvas - Save and Restore States

- **save()**

  This method pushes the current state onto the stack..

- **restore()**

This method pops the top state on the stack, restoring the context to that state.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The save() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
```

```
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.fillStyle = "green";
ctx.fillRect(10, 10, 50, 50);
ctx.save();
ctx.restore();
ctx.fillRect(200, 10, 50, 50);
</script>
</body>
</html>
```

7. HTML5 Canvas - Translation

HTML5 canvas provides **translate(x, y)** method which is used to move the canvas and its origin to a different point in the grid.

Here argument x is the amount the canvas is moved to the left or right, and y is the amount it's moved up or down.

Syntax

***context*.translate(*x*, *y*)**

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The translate() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px
solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.fillRect(10, 10, 100, 50);
ctx.translate(70, 70);
ctx.fillRect(10, 10, 100, 50);
</script>
</body>
</html>
```

8. HTML5 Canvas – Rotation

HTML5 canvas provides **rotate(angle)** method which is used to rotate the canvas around the current origin.

This method only takes one parameter and that's the angle the canvas is rotated by. This is a clockwise rotation measured in radians.

Syntax

context.rotate(*angle*)

angle	The rotation angle in radians Radians = degrees*Math.PI/180. To rotate 5 degrees: 5*Math.PI/180

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The rotate() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.rotate(20 * Math.PI / 180);
ctx.fillRect(50, 20, 100, 50);
</script>
</body>
</html>
```

9. HTML5 Canvas – Scaling

- HTML5 canvas provides **scale(x, y)** method which is used to increase or decrease the units in our canvas grid. This can be used to draw scaled down or enlarged shapes and bitmaps.
- This method takes two parameters where x is the scale factor in the horizontal direction and y is the scale factor in the vertical direction. Both parameters must be positive numbers.
- Values smaller than 1.0 reduce the unit size and values larger than 1.0 increase the unit size. Setting the scaling factor to precisely 1.0 doesn't affect the unit size.

Syntax

context.scale(*scalewidth, scaleheight*)

scalewidth	Scales the width (1=100%, 0.5=50%, 2=200%)
scaleheight	Scales the height (1=100%, 0.5=50%, 2=200%)

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The scale() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.strokeRect(5, 5, 25, 15);
ctx.scale(2, 2);
ctx.strokeRect(5, 5, 25, 15);
</script>
</body>
</html>
```

10.**HTML5 Canvas – Transforms**

- The transform() method scales, rotates, moves, and skews the context.
- Each object on the canvas has a transformation matrix.
- The transform() method replaces the transformation matrix, and multiplies the it with a matrix described by:

a c e
b d f
0 0 1

Syntax

***context*.transform(*a, b, c, d, e, f*)**

a	Scales the drawing horizontally
b	Skew the the drawing horizontally
c	Skew the the drawing vertically
d	Scales the drawing vertically
e	Moves the the drawing horizontally
f	Moves the the drawing vertically

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The transform() Method</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px
solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.fillStyle = "yellow";
ctx.fillRect(0, 0, 250, 100)
ctx.transform(1, 0.5, -0.5, 1, 30, 10);
ctx.fillStyle = "red";
ctx.fillRect(0, 0, 250, 100);
ctx.transform(1, 0.5, -0.5, 1, 30, 10);
ctx.fillStyle = "blue";
ctx.fillRect(0, 0, 250, 100);
</script>
</body>
</html>
```

11.**HTML5 Canvas – Composition**

The globalCompositeOperation property sets or returns how a source are drawn over a destination.

Source = drawings you are about to draw on the canvas.

Destination = drawings that are already drawn on the canvas.

Syntax

context.globalCompositeOperation = "value"

source-over	Default Displays the source over the destination
source-atop	Displays the source on top of the destination. The part of the source image that is outside the destination is not shown
source-in	Displays the source in the destination. Only the part of the source that is INSIDE the destination is shown, and the destination is transparent
source-out	Displays the source out of the destination. Only the part of the source that is OUTSIDE the destination is shown, and the destination is transparent
destination-over	Displays the destination over the source

destination-atop	Displays the destination on top of the source. The part of the destination that is outside the source is not shown
destination-in	Displays the destination in the source. Only the part of the destination that is INSIDE the source is shown, and the source is transparent
destination-out	Displays the destination out of the source. Only the part of the destination that is OUTSIDE the source is shown, and the source is transparent

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The globalCompositeOperation Property</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid
grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 75, 50);
ctx.fillStyle = "blue";
ctx.globalCompositeOperation = "source-over";
ctx.fillRect(50, 50, 75, 50);
ctx.fillStyle = "red";
ctx.fillRect(150, 20, 75, 50);
ctx.fillStyle = "blue";
ctx.globalCompositeOperation = "destination-over";
ctx.fillRect(180, 50, 75, 50);
</script>
</body>
</html>
```

12.**HTML5 Canvas - Text and Fonts**

- The font property sets or returns the font properties for canvas text.
- The font property uses the same syntax as the CSS front property.
- The default value is 10px sans-serif.

Syntax

```
context.font = "italic small-caps bold 12px arial"
```

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The font Property</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px
solid grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
</body>
</html>
```

13.**HTML5 Canvas - Pattern and Shadow**

- The shadowBlur property sets or returns the blur level for shadows.
- The default value is 0.

Syntax

```
context.shadowBlur = number
```

```
<html>
<body>
<h1>HTML5 Canvas</h1>
<h2>The shadowBlur Property</h2>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid
grey"></canvas>
<script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.shadowBlur = 20;
ctx.shadowColor = "black";
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 100, 80);
</script>
</body>
</html>
```