Web Development Technologies: HTML Canvas

Paul Fodor

CSE316: Fundamentals of Software Development

Stony Brook University

http://www.cs.stonybrook.edu/~cse316

HTML Canvas

- The <canvas> tag is part of HTML5
  - Allow drawing arbitrary text and images
  - Allows custom sizing of canvas
  - Allows placement of text and images
  - Allows styling of text
  - Allows arbitrary graphnics

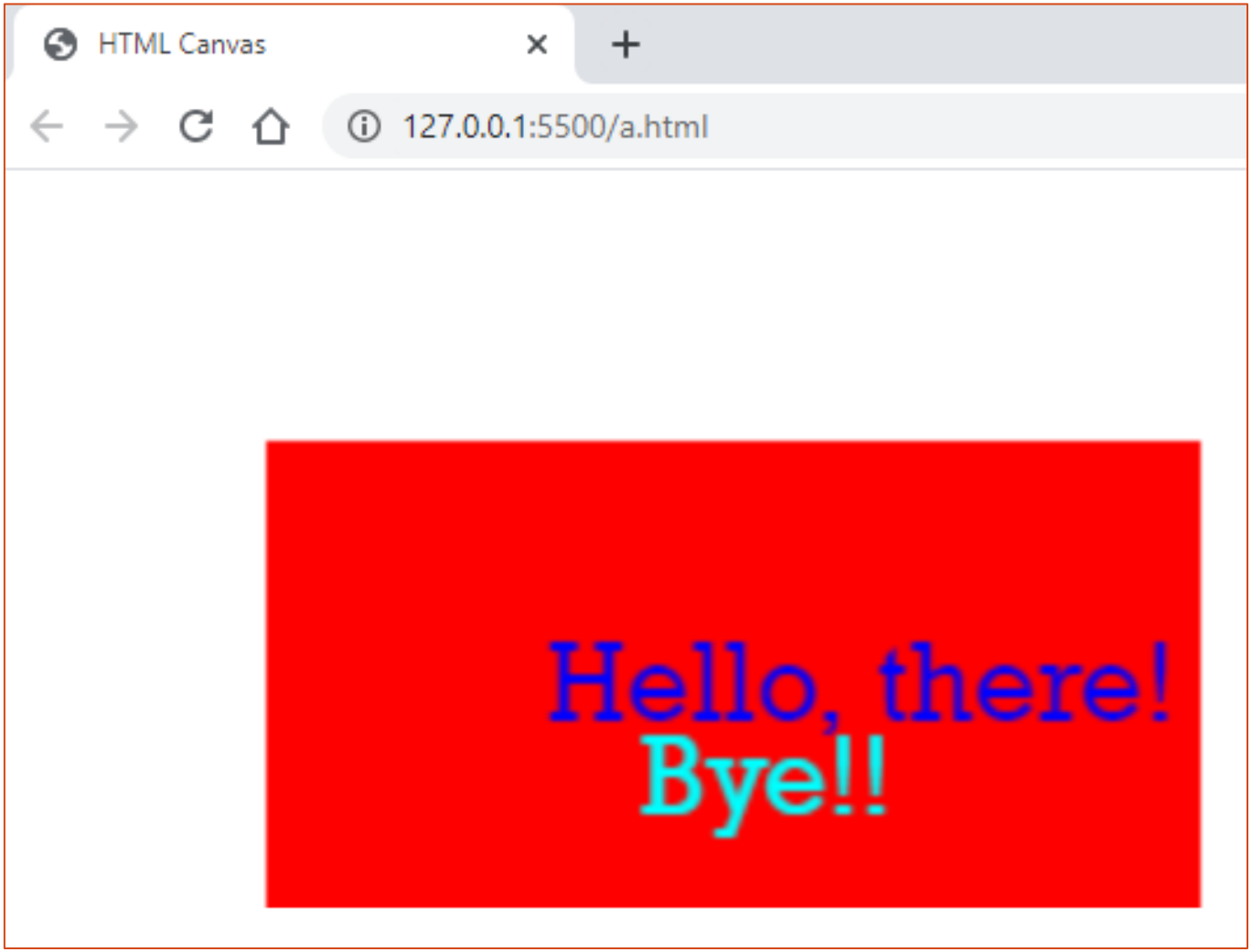
HTML Canvas History

- Canvas was initially introduced by Apple for use in their own Mac OS X WebKit component in 2004
  - In 2005 it was adopted by Gecko browsers
  - In 2006 it was adopted by Opera
  - Later, it was standardized by the Web Hypertext Application Technology Working Group (WHATWG)

HTML Canvas Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HTML Canvas</title>
</head>
<body>
    <canvas id="myCanvas"></canvas>
    <script>
        var canvas = document.getElementById("myCanvas");
        var ctx = canvas.getContext("2d");
        ctx.fillStyle = "#FF0000";
        ctx.fillRect(50,50, 200,200);
        ctx.font = "18pt Rockwell";
        ctx.fillStyle = "#0000FF";
        ctx.fillText("Hello, there!", 110, 110);
        ctx.fillStyle = "#00FFFF";
        ctx.fillText("Bye!!", 130, 130);
    </script>
</body>
</html>
```

HTML Canvas Example



Canvas and Drawing context

- Canvas is set up with a tag pair in HTML:
  - <canvas id="someId"></canvas>
- You can retrieve a 2d context
  - This context is used for all drawing
    - Images
    - Text
    - Shapes (rectangle, arcs, circles)
  - Shapes can be filled
    - Solid
    - gradients
  - You can change fonts, fill styles and other features

Creating a Canvas with HTML

- Canvas can have **width** and **height** properties.
- The text between start and end tags is displayed if the browser has no canvas support!
  - very rare, but possible for old browsers (HTML 5 was released in 2008)

```
<canvas id="myCanvas" width="400" height="200"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.
</canvas>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HTML Canvas</title>
</head>
<body>
    <canvas id="myCanvas" width="400" height="200"
       style="border:1px solid #d3d3d3;">
            Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var canvas = document.getElementById("myCanvas");
        var ctx = canvas.getContext("2d");
        ctx.fillStyle = "#FF0000";
        ctx.fillRect(50,50, 200,200);
        ctx.font = "18pt Rockwell";
        ctx.fillStyle = "#0000FF";
        ctx.fillText("Hello, there!", 110, 110);
        ctx.fillStyle = "#00FFFF";
        ctx.fillText("Bye!!", 130, 130);
    </script>
</body>
</html>
```

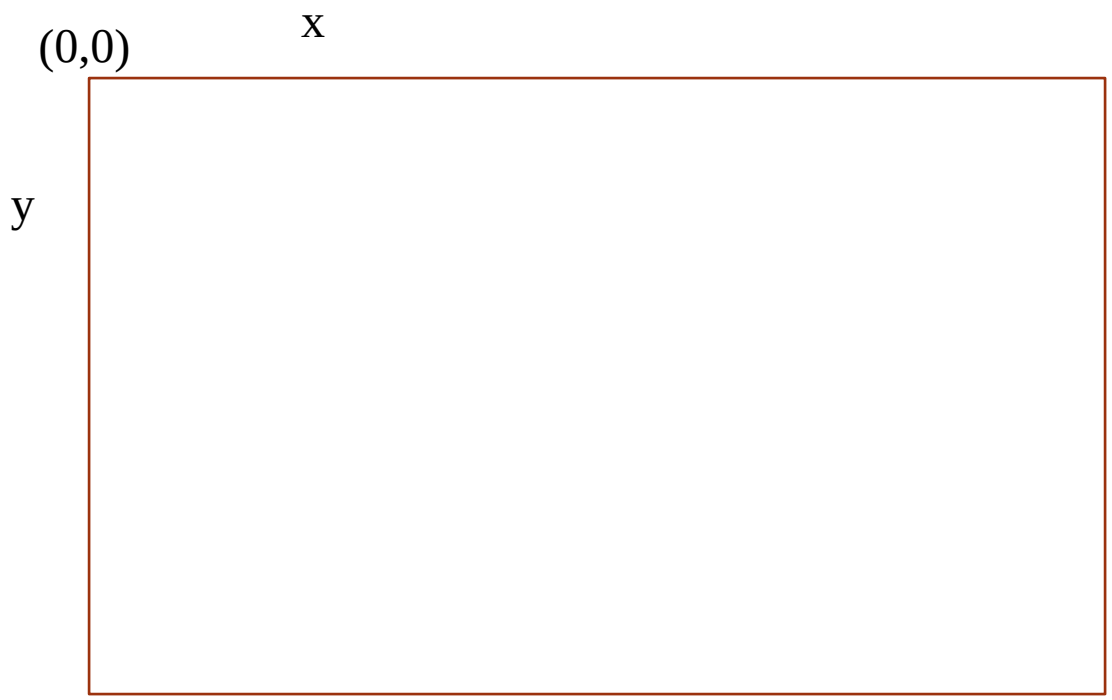
Getting the canvas and context

- Retrieve canvas as usual with getElementById()
- Retrieve context from canvas with getContext()

```
var myCanvas = document.getElementById("myCanvas");
var ctx = myCanvas.getContext("2d");
```

Coordinates

- Coordinates in canvas are from upper left (0,0) to lower right (size specified when creating canvas)

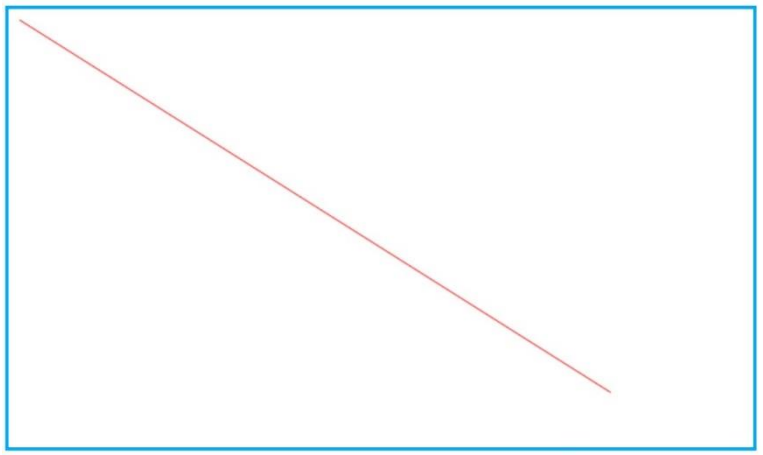


Drawing Lines

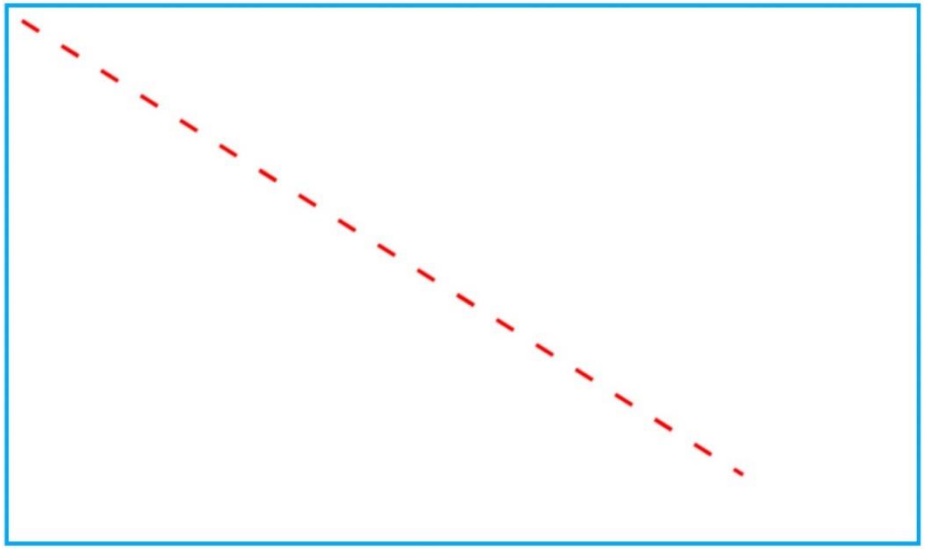
- Several methods to know:
  - **moveTo(x,y);** // moves the 'pen' to the x, y coordinates given
  - **drawTo(x,y);** // draws from the current location to the new x,y coordinates given
  - **stroke();** // render the lines and objects described so far
  - **beginPath();** // start a new line or set of lines with given features
  - **setLineDash();** // sets a dash pattern for the line(s)
- Some attributes to know:
  - **strokeStyle** – Sets the color of the line
  - **lineWidth** - Sets the width of the line drawn

Examples: Lines

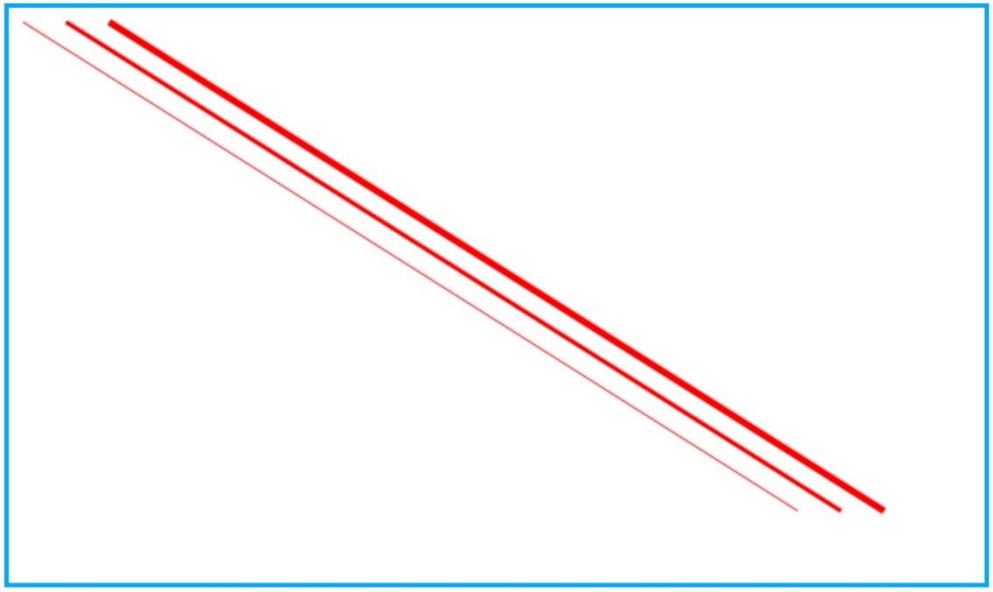
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.strokeStyle = "#FF0000";
  ctx.lineWidth=1;
  ctx.moveTo(10,10);
  ctx.lineTo(550, 350);
  ctx.stroke();
</script>

</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="680" height="400"
 style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.strokeStyle = "#FF0000";
  ctx.lineWidth=3;
  ctx.setLineDash([15, 20, 15, 20]);
  ctx.moveTo(10,10);
  ctx.lineTo(550, 350);
  ctx.stroke();
</script>

</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.strokeStyle = "#FF0000";
 ctx.beginPath();
 ctx.lineWidth=1;
 ctx.moveTo(10,10);
 ctx.lineTo(550, 350);
 ctx.stroke();
 ctx.beginPath();
 ctx.lineWidth=3;
 ctx.moveTo(40,10);
 ctx.lineTo(580, 350);
 ctx.stroke();
 ctx.beginPath();
 ctx.lineWidth=5;
 ctx.moveTo(70,10);
 ctx.lineTo(610, 350);
 ctx.stroke();
</script>
</body>
</html>
```

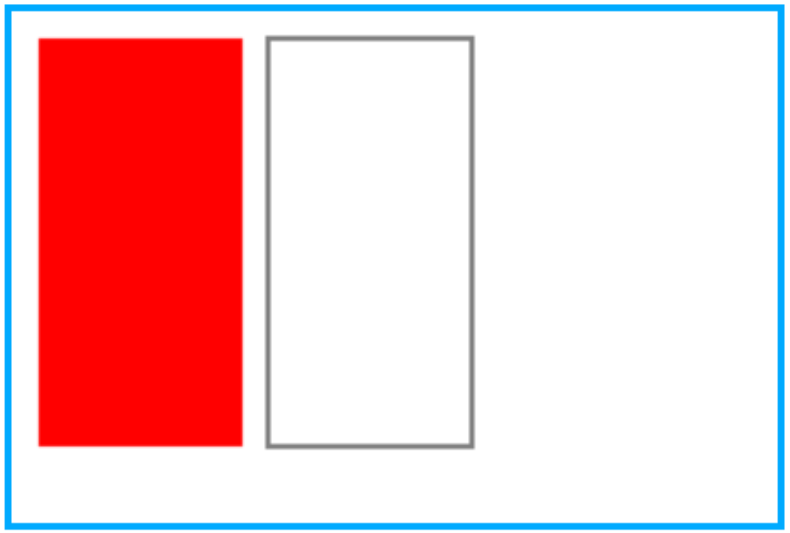


Filling Rectanges and Shapes

- Basic shape calls:
  - **fillRect(x,y,width,height);** // Draw a rectangle filled with the color in the current fillStyle at the location specified with the given width and height
  - **strokeRect(x,y,width,height);** // Draw a wireframe rectangle (no fill) at the location specified with the given width and height
  - **arc(x,y,r,startAngle,endAngle);** // Draw an arc. X,y is center, r is radius, start and end angle (in Radians) define the begin and endpoint.
  - **fill();** // Fill any objects (rectangles, arcs, etc) drawn since beginPath()

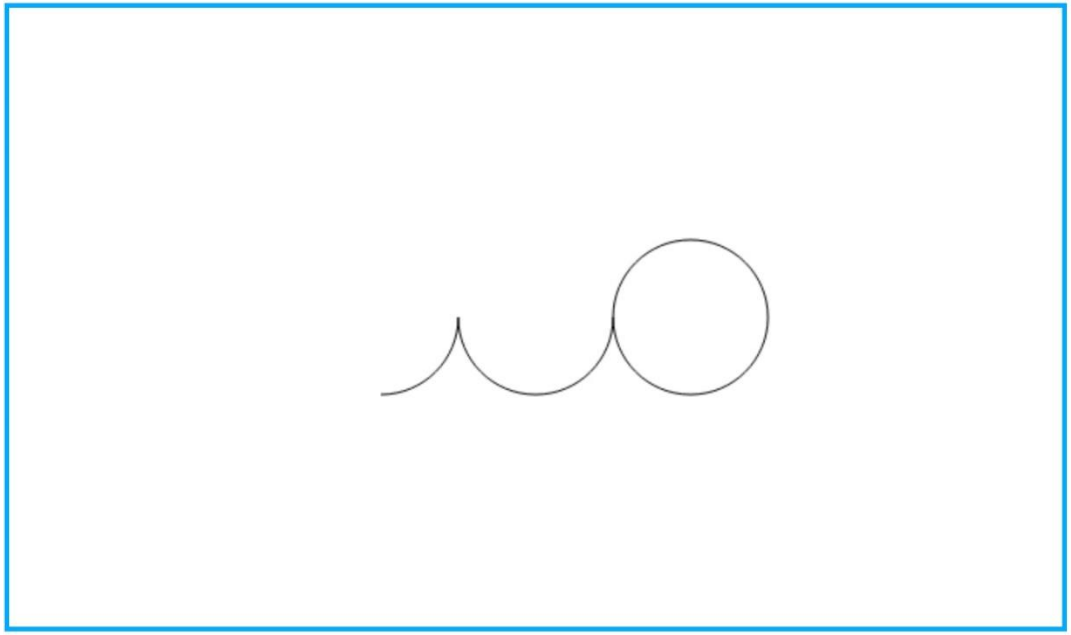
Examples: Rectangles

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="300" height="200" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.fillStyle="#FF0000";
  ctx.fillRect(10,10,80,160);
  ctx.strokeRect(100,10,80,160);
</script>

</body>
</html>
```



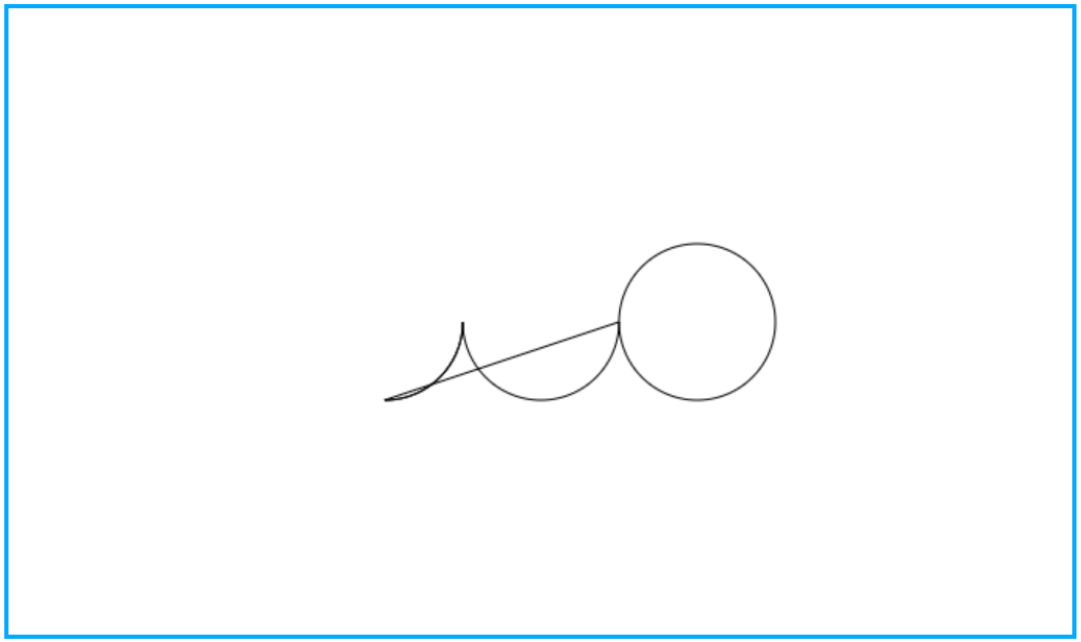
Examples: Arcs and Circles

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.fillStyle="#00aaaa";
 ctx.beginPath();
 ctx.arc(240,200,50,0,Math.PI/2);
 ctx.stroke();
 ctx.beginPath();
 ctx.arc(340,200,50,0,Math.PI);
 ctx.stroke();
 ctx.beginPath();
 ctx.arc(440,200,50,0,2*Math.PI);
 ctx.stroke();
</script>

</body>
</html>
```



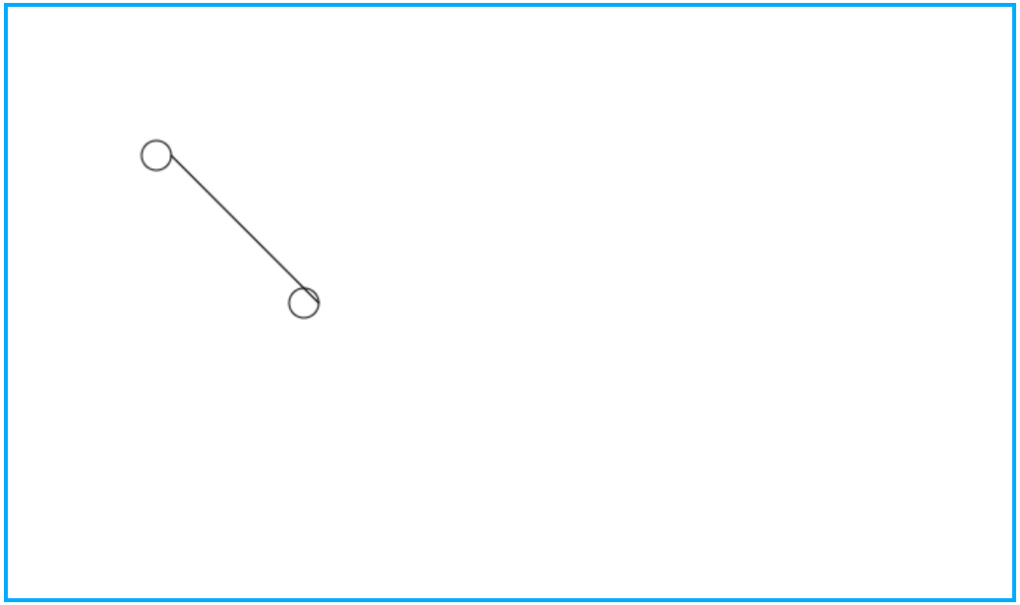
Examples: Arcs and Circles

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.fillStyle="#00aaaa";
 ctx.beginPath();
 ctx.arc(240,200,50,0,Math.PI/2);


 ctx.arc(340,200,50,0,Math.PI);
 ctx.stroke();
 ctx.beginPath();
 ctx.arc(440,200,50,0,2*Math.PI);
 ctx.stroke();
</script>

</body>
</html>
```

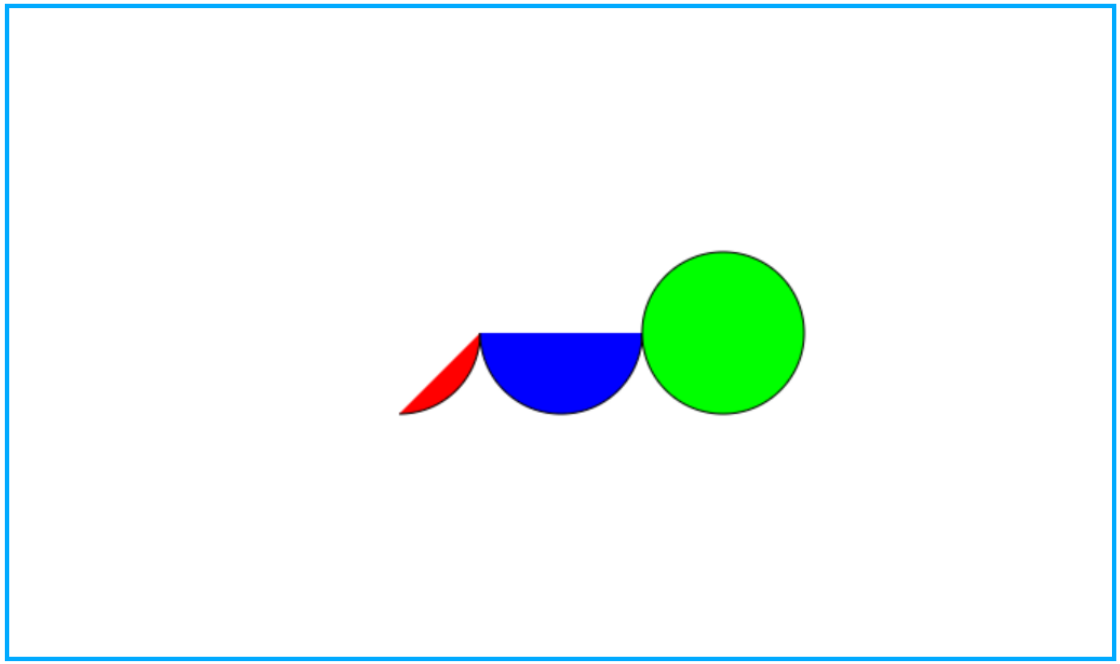


Examples: Arcs and Circles

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.fillStyle="#00aaaa";
 ctx.beginPath();
 ctx.arc(100,100,10,0,2*Math.PI);
 ctx.arc(200,200,10,0,2*Math.PI);
 ctx.stroke();
</script>

</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.fillStyle="#FF0000";
 ctx.beginPath();
 ctx.arc(240,200,50,0,Math.PI/2);
 ctx.fill();
 ctx.stroke();
 ctx.fillStyle="#0000FF";
 ctx.beginPath();
 ctx.arc(340,200,50,0,Math.PI);
 ctx.fill();
 ctx.stroke();
 ctx.fillStyle="#00FF00";
 ctx.beginPath();
 ctx.arc(440,200,50,0,2*Math.PI);
 ctx.fill();
 ctx.stroke();
</script>
</body>
</html>
```



Gradients

- A number of methods provide a way to generate gradient
  - **createLinearGradient(x, y, x1, y1);**

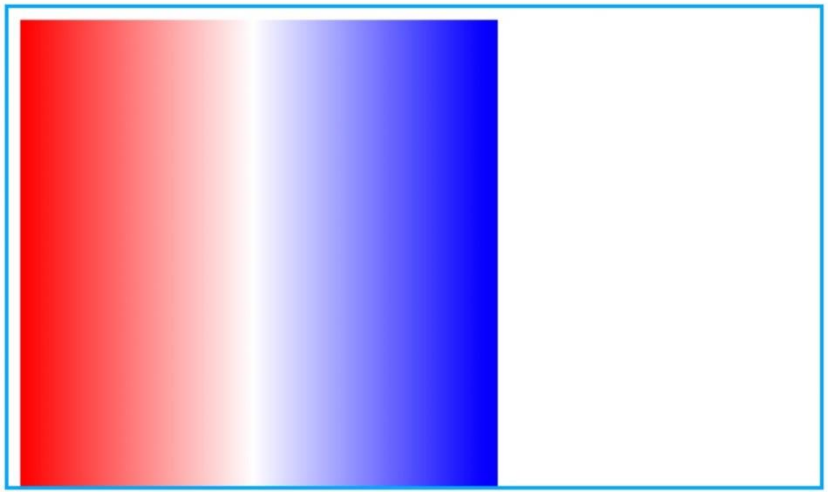
  // Generates a gradient along a linear path
  - **createRadialGradient(x, y, r, x1, y1, r1);**

  // Create a radial gradiant
  - **addColorStop(position[0-1], color);**

  // Adds a 'color' at a specific percentage through the gradient.

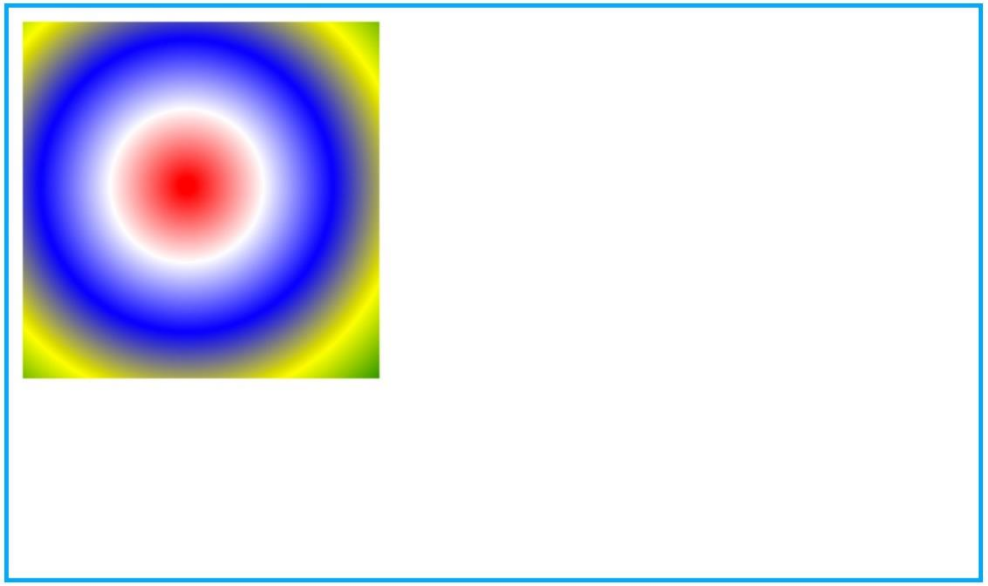
Examples: Gradients (Linear)

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 var grad = ctx.createLinearGradient(10,10,400,10);
 grad.addColorStop(0, "red");
 grad.addColorStop(0.5, "white");
 grad.addColorStop(1, "blue");
 ctx.fillStyle = grad;
 ctx.fillRect(10,10, 400, 400);
</script>

</body>
</html>
```



Examples: Gradients (Radial)

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
//  var grad = ctx.createRadialGradient(75,50,5, 90,60,100);
//  var grad = ctx.createRadialGradient(75,50,5, 120,90,200);
 var grad = ctx.createRadialGradient(125,125,5, 125,125,200);
 grad.addColorStop(0, "red");
 grad.addColorStop(0.25, "white");
 grad.addColorStop(.5, "blue");
 grad.addColorStop(.75, "yellow");
 grad.addColorStop(1, "green");

 ctx.fillStyle = grad;
//  ctx.fillRect(10,10, 350, 350);
 ctx.fillRect(10,10, 250, 250);
</script>

</body>
</html>
```



Text

- Important methods to know:
  - fillText(); // Adds text with given fill style (solid or gradient)
  - strokeText(); // Adds text with stroke lines (no fill)
- Important properties:
  - font // provides font characteristics including…
    - Size
    - Font family
    - Qualities (bold, italic)

Examples: Text

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.font = "italic bold 50px Courier";
 ctx.fillStyle = "blue";
 ctx.fillText("Hello, ", 50, 50);
 ctx.font = "bold 40px Rockwell";
 ctx.fillStyle = "red";
 ctx.fillText(" world!", 75, 75);
 ctx.strokeText("Hello, World!!!", 150, 150);
</script>
</body>
</html>
```

Images

- Canvases allow the placement of images as wel
  - **drawImage(img, x, y, width, height);**

  // Draw the image in the image object (img) at location x, y with the width and height provided

Example: Images

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="680" height="400" style="border:3px solid #00aaff;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 var img = new Image();
 img.src = "http://www3.cs.stonybrook.edu/~cse316/H1st_OOD.jpg";
 img.onload = function(){
    ctx.drawImage(img,100,100, 200, 200); // Or at whatever offset you like
 };
</script>
</body>
</html>
```

Multiple objects

```
<!DOCTYPE html>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
<canvas id="myCanvas" width="400" height="400" style="border:3px solid red;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
 function disp (ctx, x, y, style, data, fInfo) {
    console.log("Text: " + data);
    ctx.font=fInfo;
    ctx.fillStyle=style;
    ctx.fillText(data, x, y);
 }
 var coords = [50, 50, 100, 100, 150, 150];
 var strings = ["Hello", "there", "world!"];
 var fontInfo = "40pt Rockwell";
 var styles = ["#FF0000", "#0000FF", "#00FF00"];
 var canvas = document.getElementById("myCanvas");
 var ctx = canvas.getContext("2d");
 ctx.fillStyle = "#000000";
 var i;
 for (i = 0; i<3; i++) {
    disp(ctx, coords[i*2], coords[i*2+1], styles[i], strings[i], fontInfo);
 }
</script>
</body>
</html>
```

HTML Canvas Game

Learn how to make games, using nothing but HTML and JavaScript:

To make a game, start by creating a gaming area, and make it ready for drawing:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
canvas {
    border: 1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<body onload="startGame()">
<script>
function startGame() {
    myGameArea.start();
}
var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    }
}
</script>
<p>We have created a game area! (or at least an empty canvas)</p>
</body>
</html>
```

HTML Canvas Game

Add a red square onto the game area:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<body onload="startGame()">
<script>
var myGamePiece;
function startGame() {
    myGameArea.start();
    myGamePiece = new component(30, 30, "red", 10, 120);
}
var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    }
}
function component(width, height, color, x, y) {
    this.width = width;
    this.height = height;
    this.x = x;
    this.y = y;
    ctx = myGameArea.context;
    ctx.fillStyle = color;
    ctx.fillRect(this.x, this.y, this.width, this.height);
}
</script>
</body>
</html>
```

HTML Canvas Game

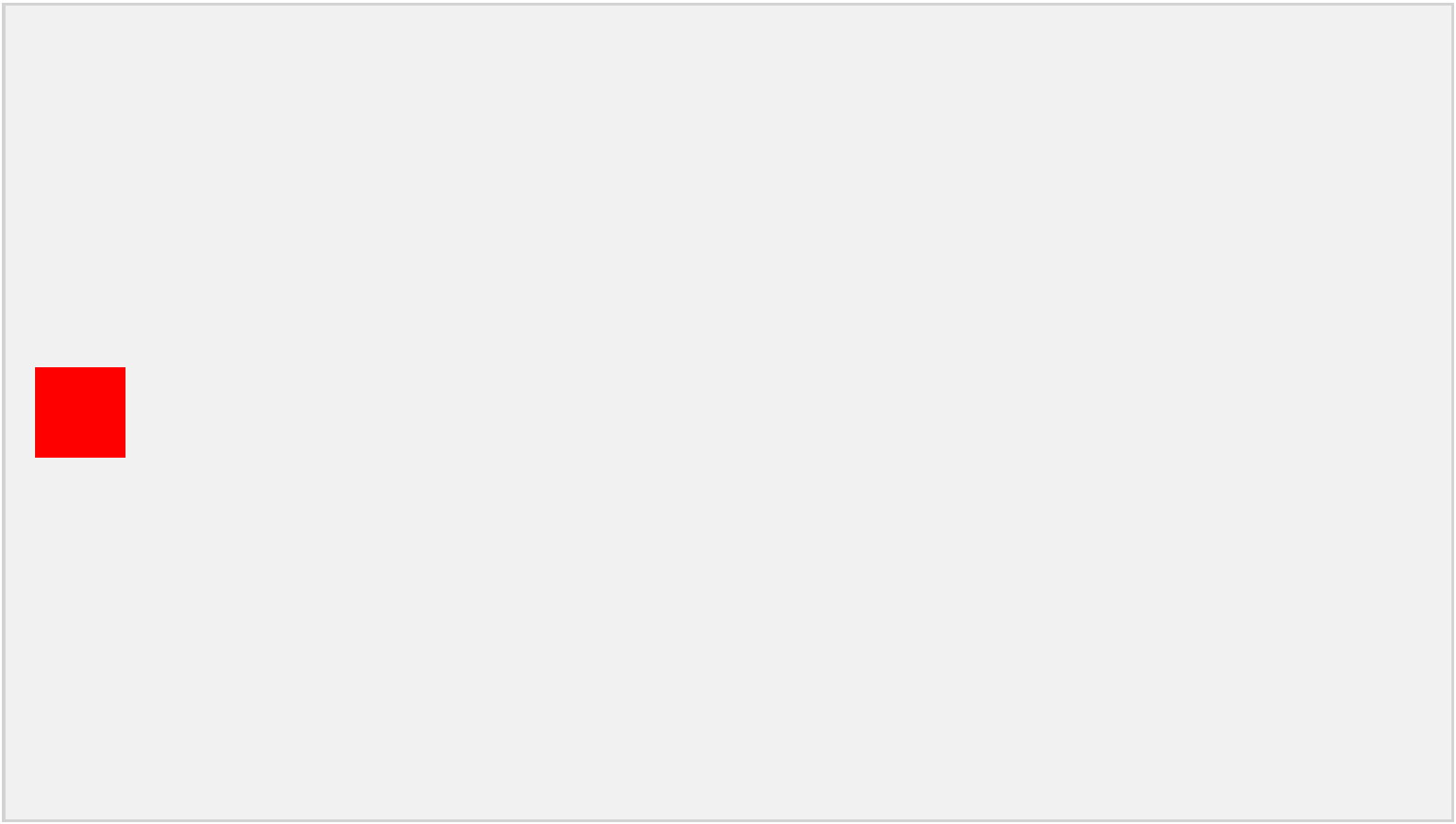
To make the game ready for action, we will update the display 50 times per second, which is much like frames in a movie.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<body onload="startGame()">
<script>

var myGamePiece;

function startGame() {
    myGamePiece = new component(30, 30, "red", 10, 120);
    myGameArea.start();
}

var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas,
document.body.childNodes[0]);
        this.interval = setInterval(updateGameArea, 20);
    },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width,
this.canvas.height);
    }
}

function component(width, height, color, x, y) {
    this.width = width;
    this.height = height;
    this.x = x;
    this.y = y;
    this.update = function(){
        ctx = myGameArea.context;
        ctx.fillStyle = color;
        ctx.fillRect(this.x, this.y, this.width, this.height);
    }
}

function updateGameArea() {
    myGameArea.clear();
    myGamePiece.update();
}
</script>
<p>The red square is actually being drawn 50 times per second.</p>
</body>
</html>
```

HTML Game

Game Controllers.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<body onload="startGame()">
<script>

var myGamePiece;

function startGame() {
    myGamePiece = new component(30, 30, "red", 10, 120);
    myGameArea.start();
}

var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
        this.interval = setInterval(updateGameArea, 20);
    },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    }
}

function component(width, height, color, x, y) {
    this.width = width;
    this.height = height;
    this.speedX = 0;
    this.speedY = 0;
    this.x = x;
    this.y = y;
    this.update = function() {
        ctx = myGameArea.context;
        ctx.fillStyle = color;
        ctx.fillRect(this.x, this.y, this.width, this.height);
    }
    this.newPos = function() {
        this.x += this.speedX;
        this.y += this.speedY;
    }
}

function updateGameArea() {
    myGameArea.clear();
    myGamePiece.newPos();
    myGamePiece.update();
}

function moveup() {
    myGamePiece.speedY -= 1;
}

function movedown() {
    myGamePiece.speedY += 1;
}

function moveleft() {
    myGamePiece.speedX -= 1;
}

function moveright() {
    myGamePiece.speedX += 1;
}
</script>
<div style="text-align:center;width:480px;">
  <button onclick="moveup()">UP</button><br><br>
  <button onclick="moveleft()">LEFT</button>
  <button onclick="moveright()">RIGHT</button><br><br>
  <button onclick="movedown()">DOWN</button>
</div>

<p>If you click a button the red square will start moving. Click the
same button many times, and it will move faster and faster.</p>
</body>
</html>
```



UP

LEFT RIGHT

DOWN

More info on canvases!

- W3schools tutorial:
  https://www.w3schools.com/graphics/canvas_intro.asp
- W3schools reference:
  https://www.w3schools.com/graphics/canvas_reference.asp

SVG earlier standard

- Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based standard for drawing shapes in browsers with support for interactivity and animation
  - The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.
    - All major modern web browsers—including Mozilla Firefox, Internet Explorer, Google Chrome, Opera, Safari, and Microsoft Edge—have SVG rendering support.
  - Unlike canvas, which is raster-based, SVG is vector-based, so that each drawn shape is remembered as an object in a scene graph or Document Object Model, which is subsequently rendered to a bitmap.
    - This means that if attributes of an SVG object are changed, the browser can automatically re-render the scene.

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>SVG</title>
</head>

<body>

    <svg width="391" height="391" viewBox="-70.5 -70.5 391 391" xmlns="http://www.w3.org/2000/svg">
        <rect fill="#fff" stroke="#000" x="-70" y="-70" width="390" height="390" />
        <g opacity="0.8">
            <rect x="25" y="25" width="200" height="200" fill="green" stroke-width="4" stroke="pink" />
            <circle cx="125" cy="125" r="75" fill="orange" />
            <polyline points="50,150 50,200 200,200 200,100" stroke="red" stroke-width="4" fill="none" />
            <line x1="50" y1="50" x2="200" y2="200" stroke="blue" stroke-width="4" />
        </g>
    </svg>

</body>

</html>
```