Intro to drawing

The <canvas> tag

The area enclosed by the <canvas> tags can be used for drawing and animation

<canvas> </canvas>

Attributes

- <canvas id="myCanvas" width="600" height="400"> </canvas>

- Before the canvas can be used for drawing and animation, it must have an ID, width, and height assigned to it.

- These may appear in the HTML, or they may be created with JavaScript/jQuery.

Default content

In Web browsers that do not support HTML5 , the canvas will not appear.

```
<canvas id="myCanvas" width="600"
height="400">
<p>Some default content can appear
here.</p> </canvas>
```

This is all you'll see in the HTML document.
Everything else will need JavaScript

The <*canvas*> element reference

The <*canvas*> element exposes an abundance of functionality through its canvas context, which is accessible using JavaScript. This element provides the following members.

- **height** Property that sets or gets the height of the canvas
- **width** Property that sets or gets the width of the canvas
- **getContext()** Method that accepts a parameter of 2d and returns a CanvasRenderingContext2D object that represents the canvas context
- **toDataUrl()** Method that creates a URL that can be used with an element that requires an image URL, such as the <*img*> element

- **addColorStop()** Method to set the colors and stop positions in a gradient object
- **arc()** Method to create an arc/curve
- **arcTo()** Method to create an arc/curve between two tangents
- **beginPath()** Method to start a path or reset the current path
- **bezierCurveTo()** Method to create a cubic Bézier curve
- **clearRect()** Method to clear a given rectangle
- **clip()** Method to clip a region of any shape and size from the original canvas
- **closePath()** Method to create a path from the current point back to the starting point
- **createImageData()** Method to create a new, blank ImageData object
- **createLinearGradient()** Method to create a linear gradient
- **createPattern()** Method to repeat a specified element in a specified direction
- **createRadialGradient()** Method to create a radial/circular gradient
- **data** Property that gets an ImageData object that contains the image data
- **drawImage()** Method to draw an image, canvas, or video onto the canvas
- **fill()** Method to fill the drawing path
- **fillRect()** Method to draw a filled rectangle
- **fillStyle** Property that sets or gets the color, gradient, or pattern used to fill the drawing
- **fillText()** Method to draw filled text on the canvas
- **font** Property that sets or gets the font properties for text content
- **getImageData()** Method to get an ImageData object that copies the pixel data for the specified rectangle on a canvas
- **globalAlpha** Property that sets or gets the current alpha or transparency value of the drawing
- **globalCompositeOperation** Property that sets or gets how a new image is drawn onto an existing image
- **isPointInPath()** Method that returns true if the specified point is in the current path
- **lineCap** Property that sets or gets the style of the end caps for a line
- **lineJoin** Property that sets or gets the type of corner to create when two lines meet
- **lineTo()** Method that adds a new point and creates a line from that point to the last specified point in the canvas
- **lineWidth** Property that sets or gets the current line width
- **measureText()** Method that gets an object that contains the width of the specified text
- **miterLimit** Property that sets or gets the maximum miter length

- **moveTo()** Method that moves the path to the specified point in the canvas without creating a line
- **putImageData()** Method that puts the image data from a specified ImageData object back onto the canvas
- **quadraticCurveTo()** Method that creates a quadratic Bézier curve
- **rect()** Method that creates a rectangle
- **restore()** Method that pops the previously saved context state from the stack
- **rotate()** Method that rotates the current drawing
- **save()** Method that pushes the state of the current context onto a stack
- **scale()** Method that scales the current drawing bigger or smaller
- **setTransform()** Method that resets the current transform to the identity matrix and then calls the transform() method
- **shadowBlur** Property that sets or gets the blur level setting to use for shadows
- **shadowColor** Property that sets or gets the color setting to use for shadows
- **shadowOffsetX** Property that sets or gets the horizontal distance setting of the shadow from the shape
- **shadowOffsetY** Property that sets or gets the vertical distance setting of the shadow from the shape
- **stroke()** Method to draw the path you have defined
- **strokeRect()** Method to draw a rectangle without fill
- **strokeStyle** Property that sets or gets the color, gradient, or pattern used for strokes
- **strokeText()** Method that draws text on the canvas without fill
- **textAlign** Property that sets or gets the alignment setting for text content
- **textBaseline** Property that sets or gets the text baseline setting used when drawing text
- **transform()** Method that replaces the transformation matrix setting for the drawing
- **translate()** Method that remaps the (0, 0) position on the canvas

Implementing the canvas

```
$(document).ready(function () {
    drawSomething();
});

function drawSomething() {
    var canvas = document.getElementById('myCanvas');
    var ctx = canvas.getContext('2d');
    ctx.fillRect(10, 50, 100, 200);
}
```

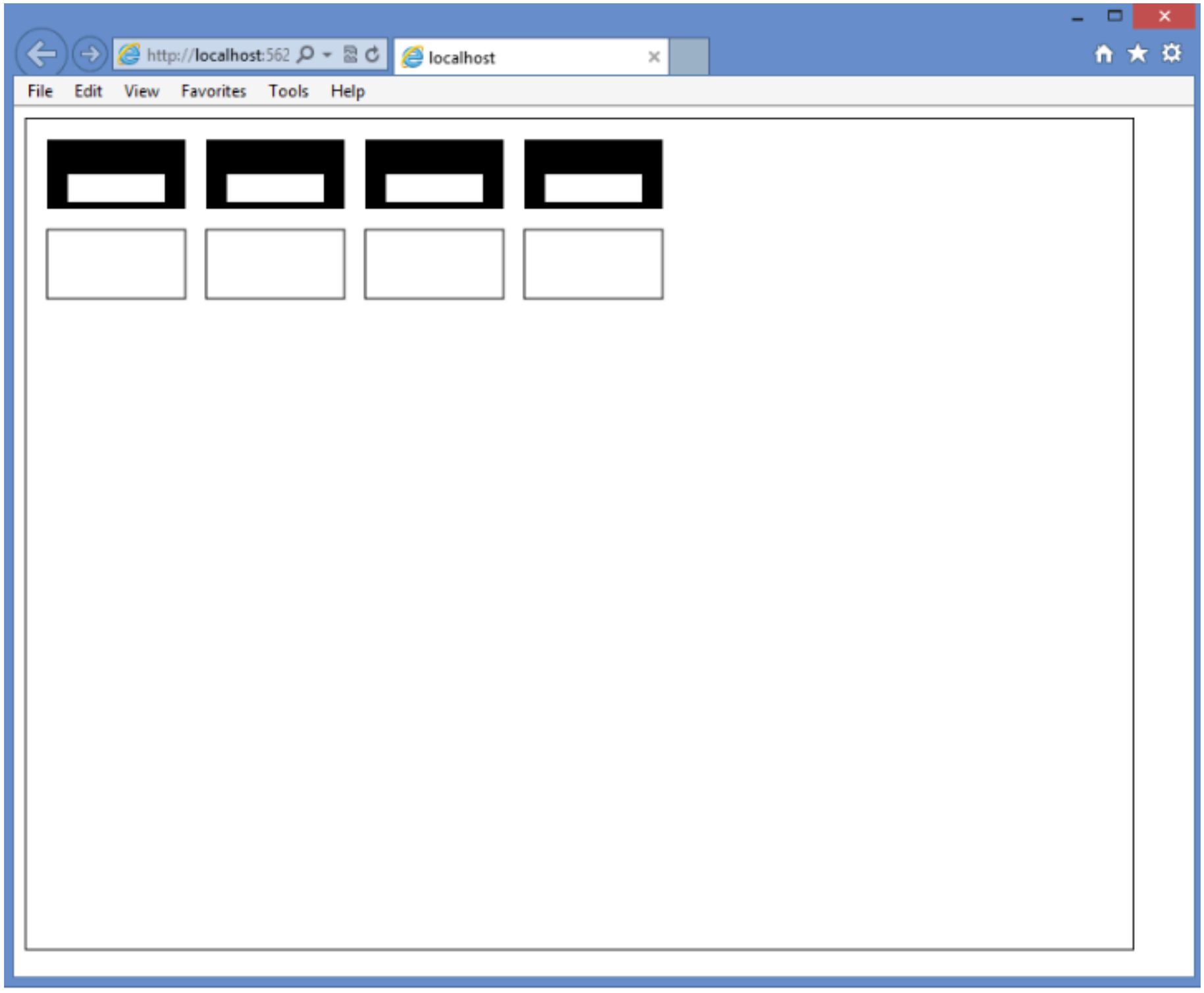
Drawing rectangles

The methods for creating rectangles accept four parameters. The first two parameters are the x and y locations of the upper-left corner of the rectangle. The last two parameters represent the width and height of the rectangle. You can create rectangles by using one of the following methods.

- **clearRect(x, y, w, h)** Clear the specified rectangular area.
- **fillRect(x, y, w, h)** Draw a filled rectangular area.
- **strokeRect(x, y, w, h)** Draw an unfilled rectangular area.

```
$(document).ready(function () {
 drawSomething();
 });
  function drawSomething() {
  var canvas = document.getElementById(‘myCanvas')
  , ctx = canvas.getContext(‘2d')
  , offset = 15
  , clearOffset = 30
  , pushDownOffset = 10
  , height = 50
  , width = 100
  , count = 4
  , i = 0;

for (i = 0; i < count; i++) {
    ctx.fillRect(i * (offset + width) + offset, offset, width,
       height); ctx.clearRect(i * (offset + width) +
       (clearOffset / 2) + offset, offset + (clearOffset / 2) +
       pushDownOffset, width - clearOffset, height -
       clearOffset) ctx.strokeRect(i * (offset + width) +
       offset, (2 * offset) + height, width, height);
       }
}
```


http://localhost:562
localhost
File
Edit
View
Favorites
Tools
Help

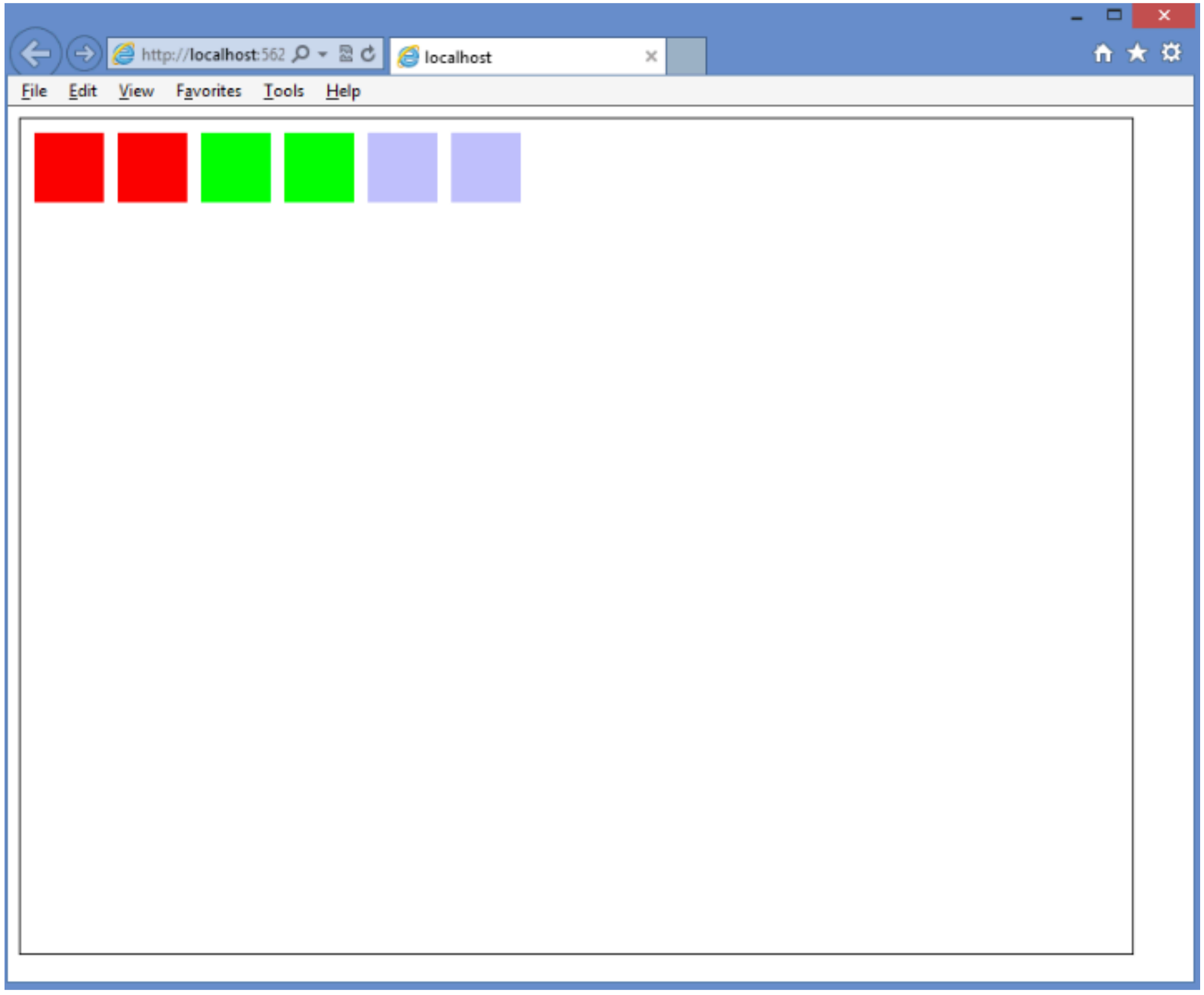
- **CSS color** Creates a solid color fill based on a valid CSS color value such as black, red, or #00FF00. The following is an example of setting fillStyle by using a *CSS color* value.

```
function drawUsingCssColor() {
    var canvas = document.getElementById('myCanvas')
        , ctx = canvas.getContext('2d')
        , offset = 10
        , size = 50;

    ctx.fillStyle = "red";
    ctx.fillRect(offset + (0 * (offset + size)), offset, size, size);
    ctx.fillRect(offset + (1 * (offset + size)), offset, size, size);

    ctx.fillStyle = "#00FF00";
    ctx.fillRect(offset + (2 * (offset + size)), offset, size, size);
    ctx.fillRect(offset + (3 * (offset + size)), offset, size, size);

    ctx.fillStyle = "rgba(0, 0, 255, 0.25)";
    ctx.fillRect(offset + (4 * (offset + size)), offset, size, size);
    ctx.fillRect(offset + (5 * (offset + size)), offset, size, size);
}
```

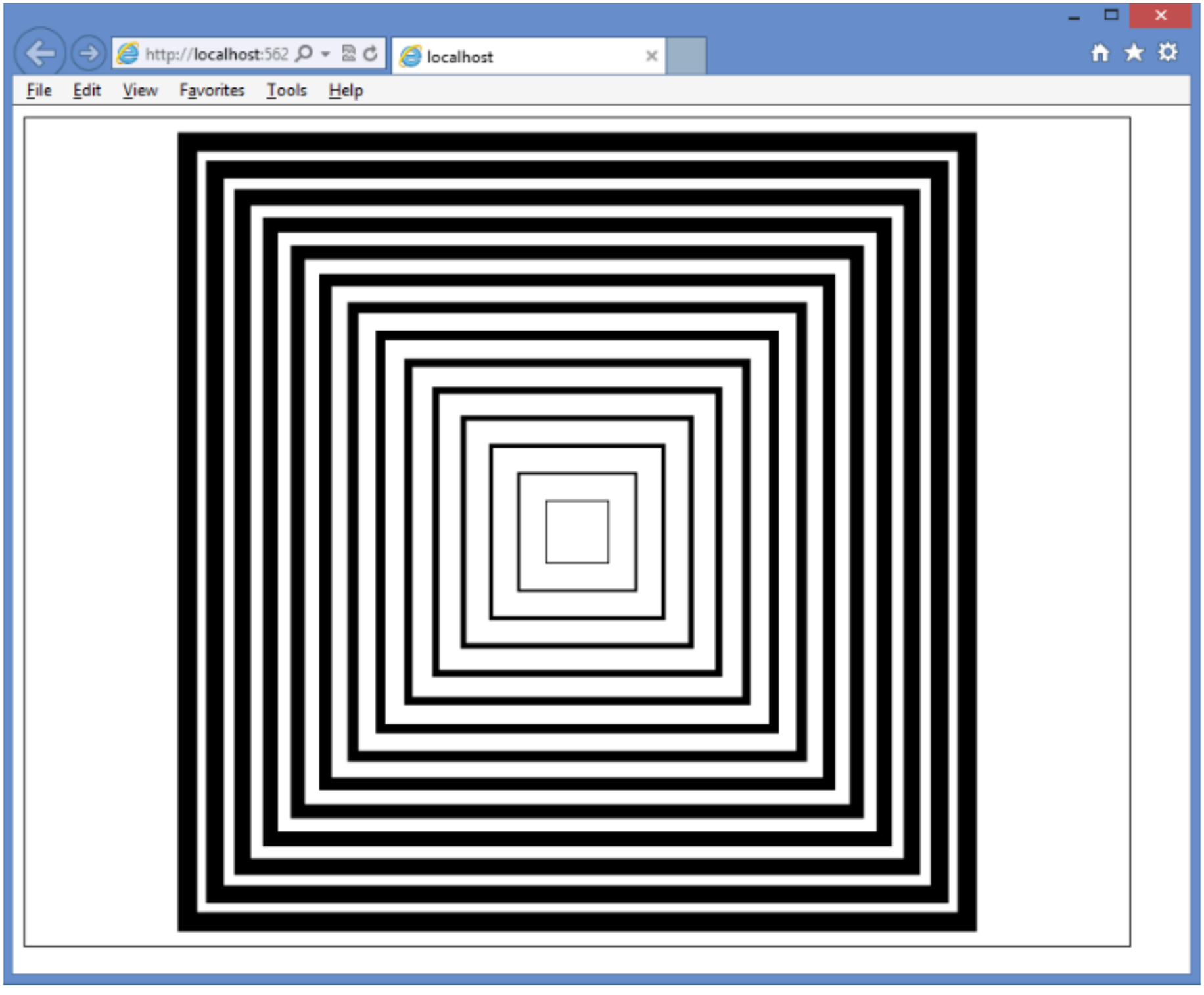

http://localhost:562
localhost
File Edit View Favorites Tools Help

Setting lineWidth

The lineWidth property specifies the thickness of any line you draw. The following code example draws rectangles by using different lineWidth settings.

```
function drawLineWidth() {
    var canvas = document.getElementById('myCanvas')
        , ctx = canvas.getContext('2d')
        , offset = 40
        , width = 5
        , height = 5
        , lineWidth = 1
        , i = 0
        , centerX = canvas.width / 2
        , centerY = canvas.height / 2;

    for (i = 1; i < 15; i++) {
        ctx.lineWidth = i;
        ctx.strokeRect(centerX - (width / 2) - (i * offset / 2),
            centerY - (height / 2) - (i * offset / 2),
            width + (i * offset), height + (i * offset));
    }
}
```


http://localhost:562
localhost
File Edit View Favorites Tools Help