# Drawing with HTML5

In this chapter, we will learn about two things:

* Lesson 1: Drawing by using the <canvas> element
* Lesson 2: Using scalable vector graphics

## Lesson 1: Drawing by using the <canvas> element

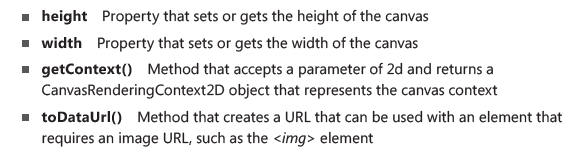
Below is an example of using the <canvas> element. Note that the content within the <canvas> tag is only shown if the browser does not support the <canvas> element.

e.g.  


**JsFiddle**: https://jsfiddle.net/qevzyoLL/

### The <canvas> element reference

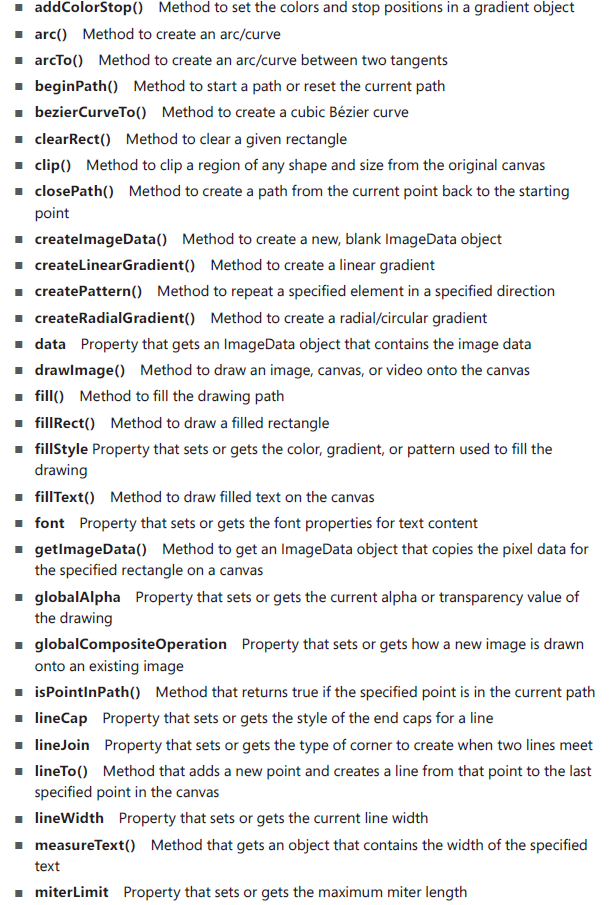
The <canvas> element has the following attributes:

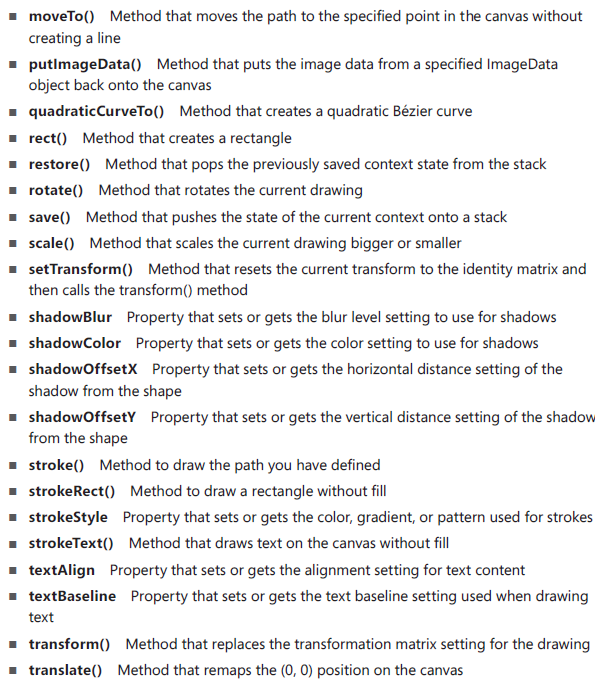


### CanvasRenderingContext2D context object reference

The <canvas> element is simply a graphics container; the **context object** that is returned from the *getContext* method is **what we** **actually use to draw on the canvas**.

The following is a list of the context object's members:

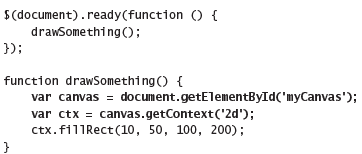




### Implementing the canvas

To work with the canvas object, call the getContext method and use the canvas context returned.

The getContext method **takes one parameter, which is '2d' in all browsers**, and **returns a CanvasRenderingContext2D** object for making **2-dimensional drawings**. (However in Firefox and Chrome, the getContext method accepts an ' experimental-webgl' parameter, which supports three-dimensional drawings)

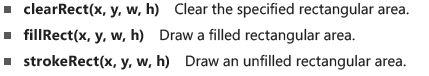
e.g.   


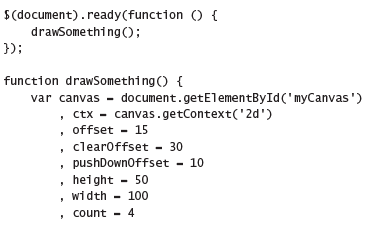
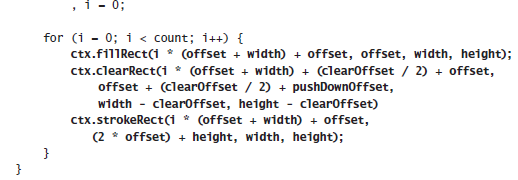
**JsFiddle**: https://jsfiddle.net/qevzyoLL/1/

### Drawing rectangles

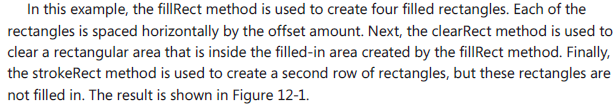
The method for creating rectangles accepts four parameters

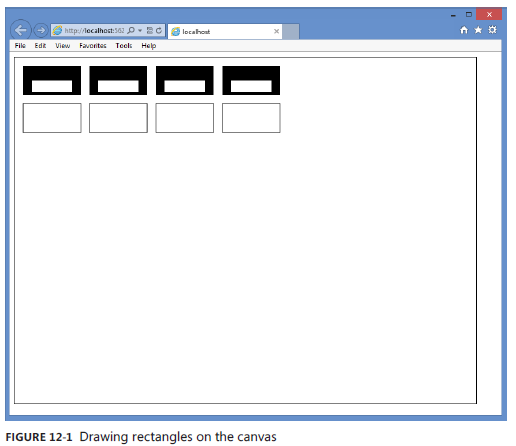
* First two parameters are x and y locations of the upper-left corner of the rectangle
* The last two parameters are the width and height of the rectangle

We can create rectangles using any of the following methods:  


e.g.   
  


**Code explanation:**



**Code Result:  
**

**JsFiddle:** https://jsfiddle.net/qevzyoLL/2/

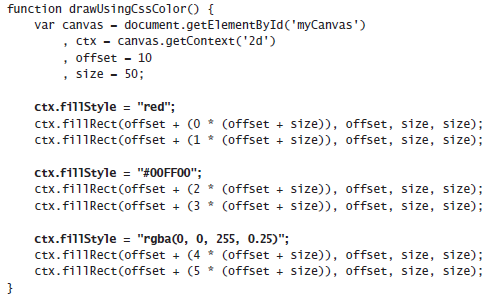
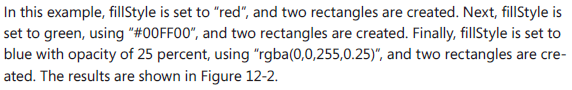
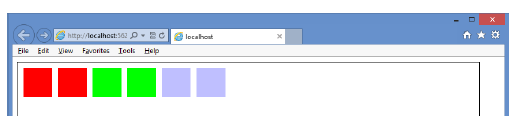
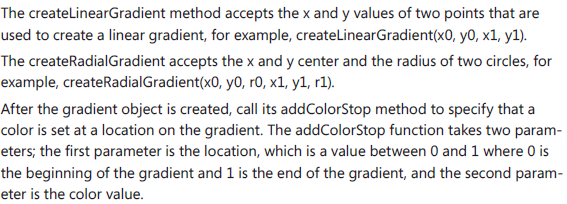
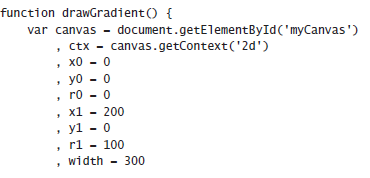
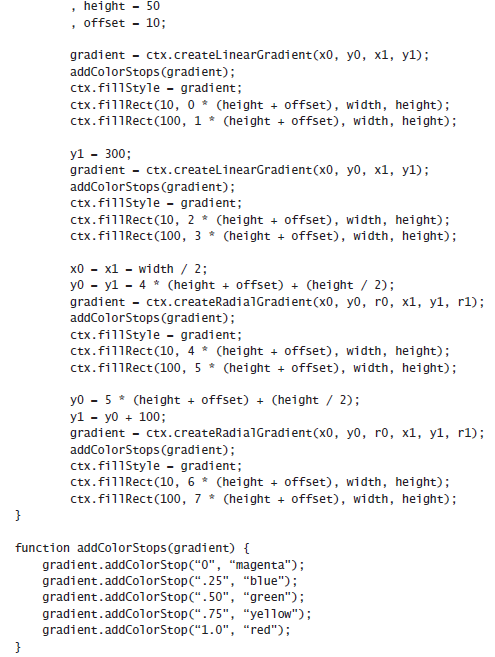
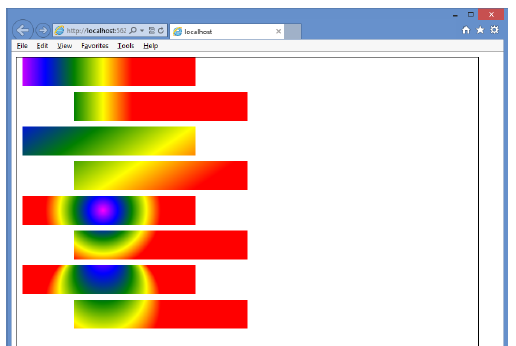
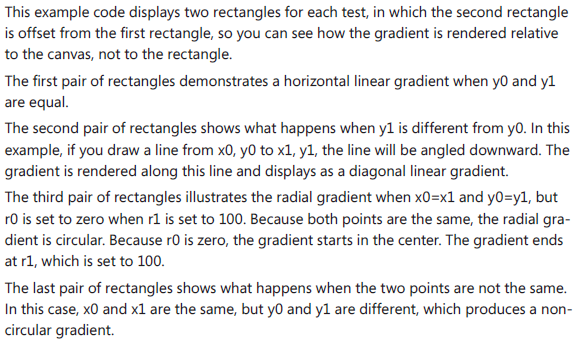
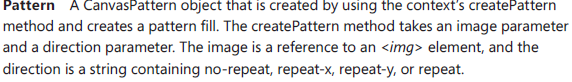
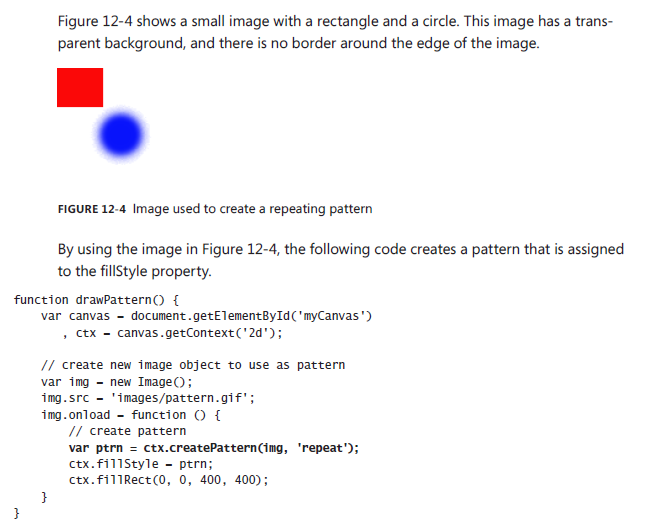
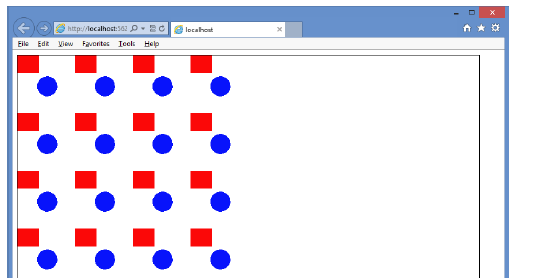
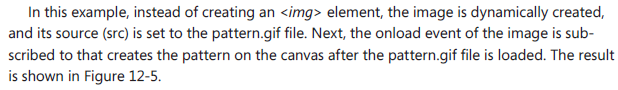
### Configuring the drawing state

The canvas context has properties that you can set before you call any of the drawing methods. After you change a property, the new value is used for subsequent drawing statements.

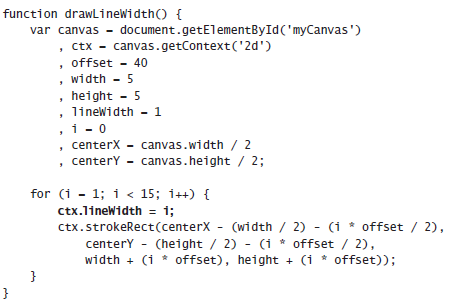
#### Setting fillStyle

Use the fillStyle property to specify how you want to fill shapes.

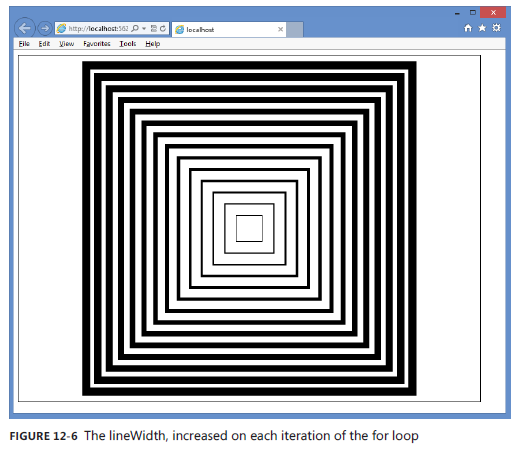
We can set the fillStyle to any of the following values:

* fillStyle **CSS color** property**.**   
  e.g.  
    
  **Code explanation:** **Code Result:**  
   **JsFiddle:** https://jsfiddle.net/qevzyoLL/3/
* fillStyle **Gradient** property**.**A gradient is a CanvasGradient object that is created by the context’s createLinearGradient or createRadialGradient method to create a gradient fill.  
  ****  
  e.g.  
  ****  
  ****  
    
  **Code Result**:   
  **  
    
  Code Explanation:  
  **
* fillStyle **Pattern** property  
  ****  
  e.g.  
  ****  
  **Code Result:  
    
    
  Code explanation:  
  **

### Setting lineWidth

The lineWidth property specifies the thickness of any line you draw.  
e.g.  
****

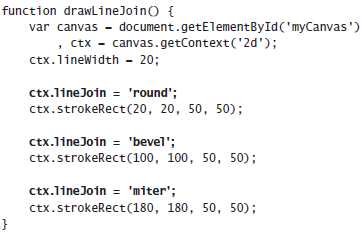
**Code Explanation:  
  
  
Code Result:**

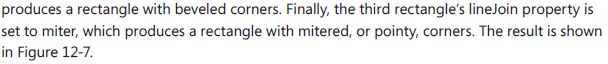
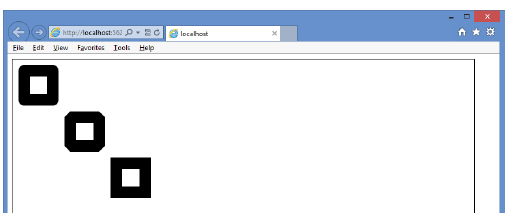
**  
JsFiddle:** https://jsfiddle.net/qevzyoLL/4/

### Setting lineJoin

The lineJoin property specifies the way lines that join each other are drawn.

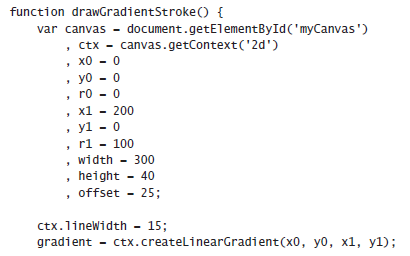
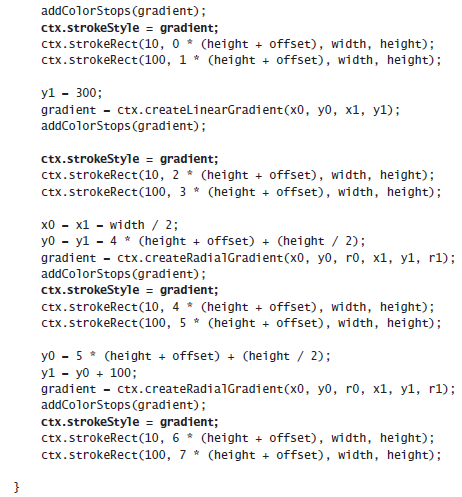
You can set the lineJoin property to round, bevel, or miter. The default value is miter

e.g.  


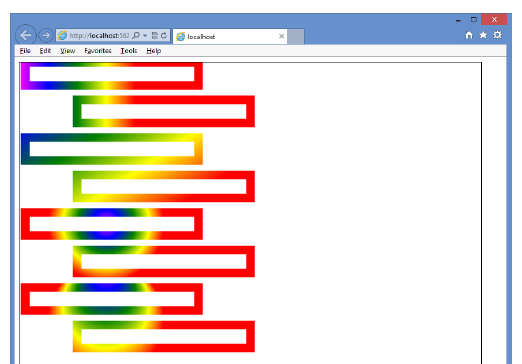
**Code explanation:  
  
  
  
Code result:  
**

### Setting strokeStyle

The strokeStyle property specifie the way you want to draw lines.

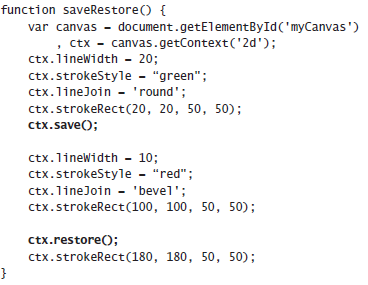
e.g.  
  


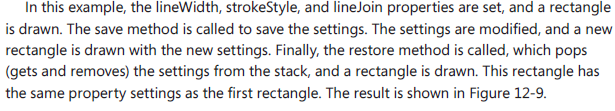
**Code Explanation**: Reference the fillStyle drawGradient function for an explanation (exactly the same concept)

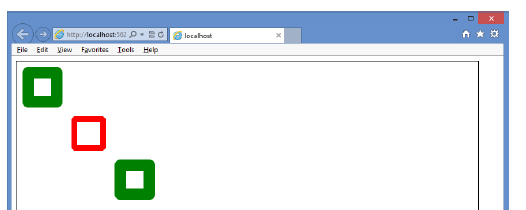
**Code Result:**  


### Saving and restoring the drawing state

It is possible to save all the context object properties to a stack, which is a last-in, first-out (LIFO) collection. This does not save the actual canvas; just the settings are saved. The save method saves the current settings, and the restore method restores the settings.

e.g.  


**Code Explanation:**  


**Code Result:**

**JsFiddle:** https://jsfiddle.net/h33fu3u6/

### Drawing by using paths

A path is a set of lines that are used to draw shapes.

Steps for creating a shape using paths:

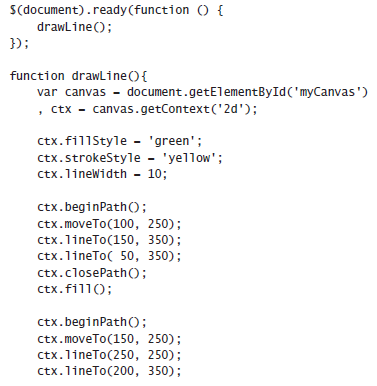
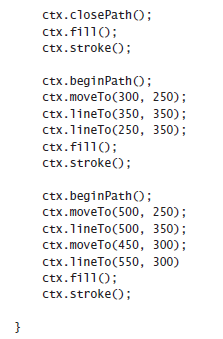
* Start a path by calling the beginPath method.
* Go to the starting position by calling the moveTo method.
* Draw sub-paths calling methods such as lineTo and rect.
* End the path by optionally calling the closePath method.
* Render the filled shape or outlined shape by calling fill or stroke methods, respectively.

Using the HTML & CSS code below, we will be drawing a bunch of shapes

HTML:  


CSS:  


#### Drawing Lines

Javascript:  
  


**Code Result:**   


#### Drawing rectangles