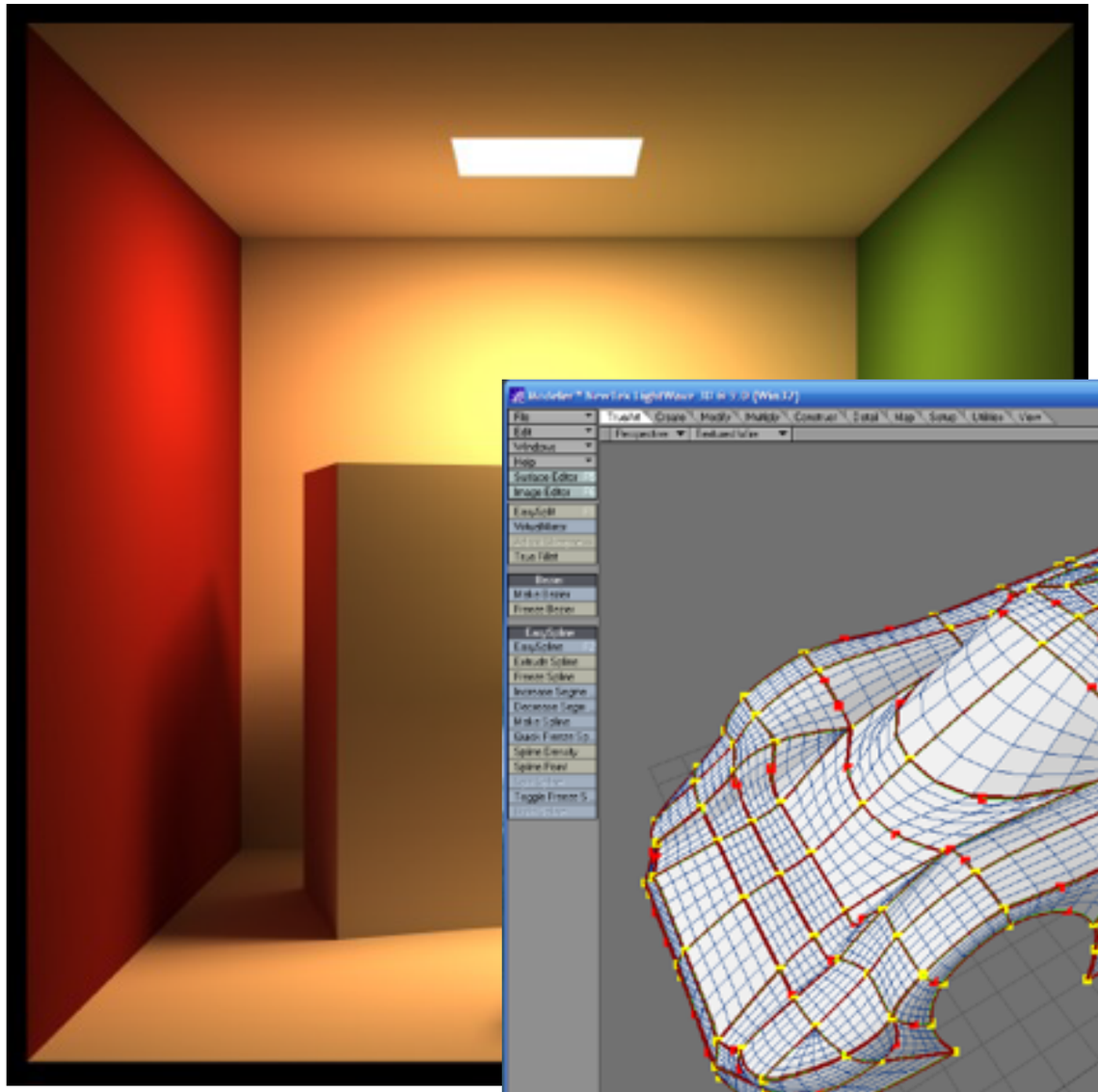
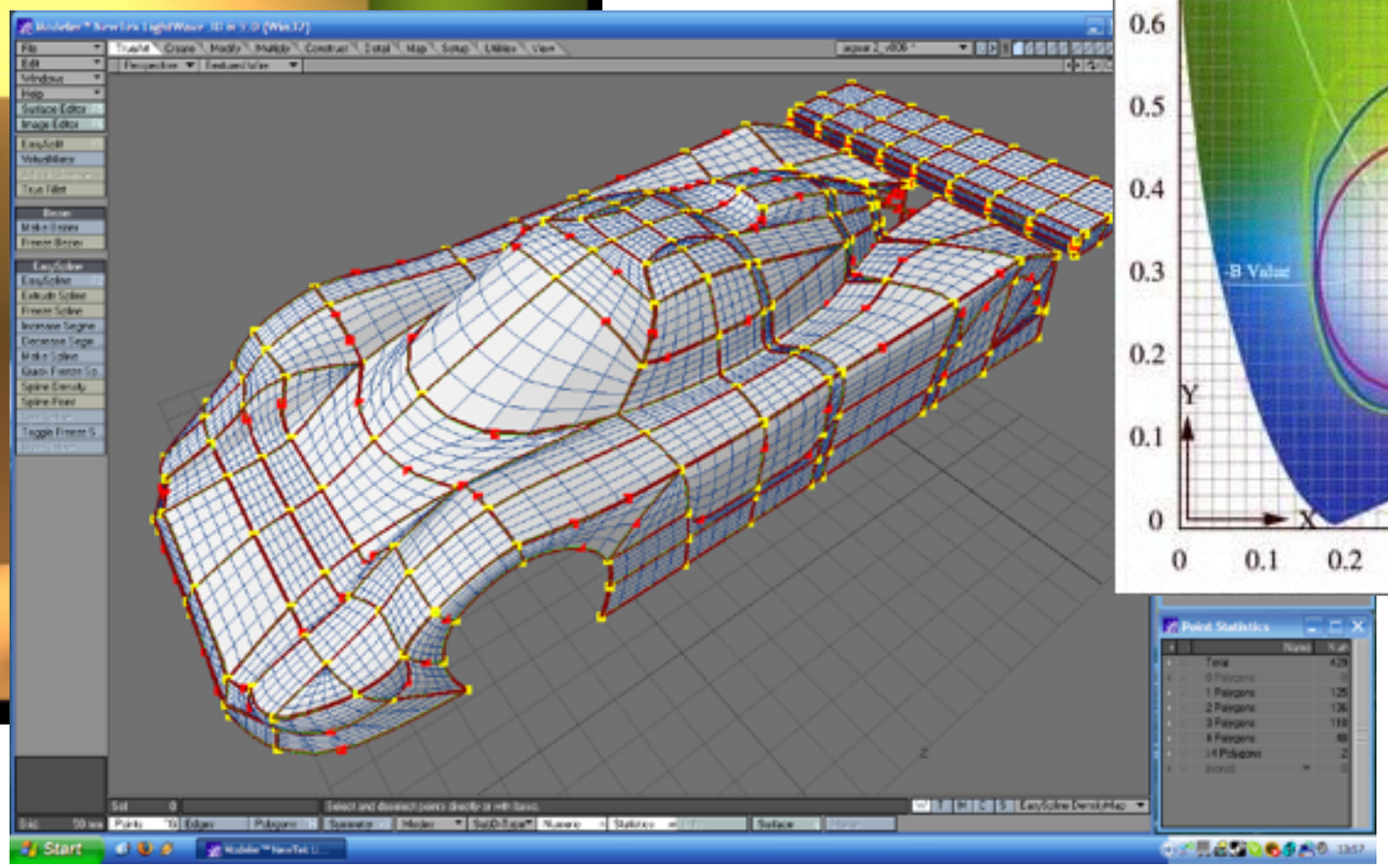
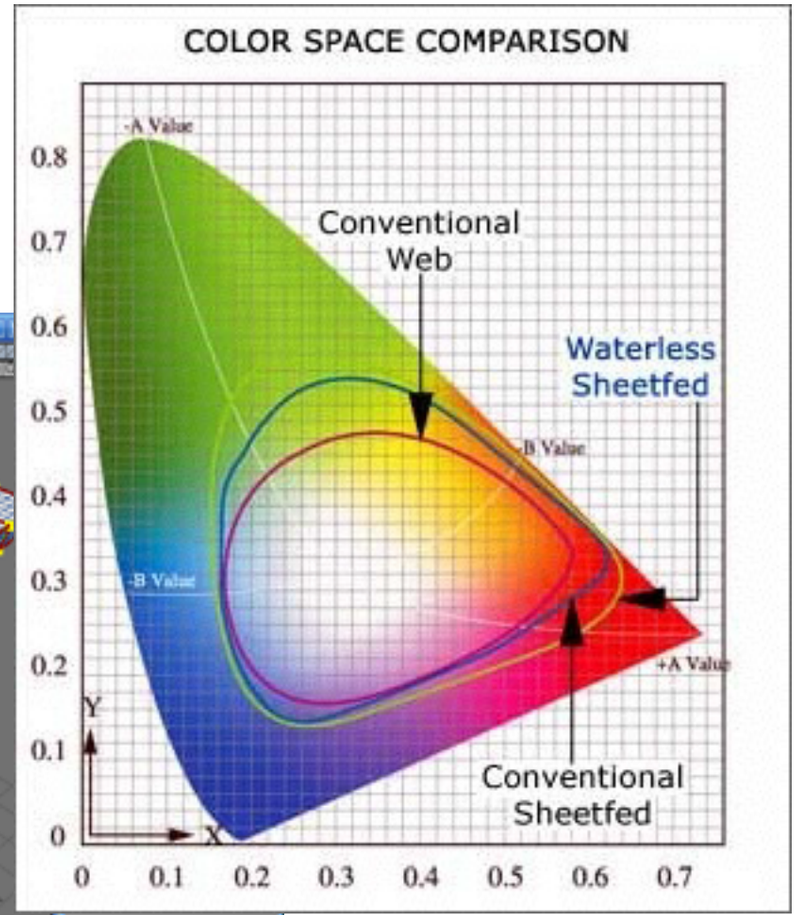






Lecture 2 : Where do I draw?
(Intro to 2D Canvas, drawing and interface elements)

Tuesday September 12th 2023

Pre-lecture logistics and announcements

- Hopefully you have signed up for Piazza (and maybe used it, if you had questions!)
- Be prepared to have your first programming assignment released to you later this week (by Thursday at the latest). You will have at least one full week before it is due.
- The subject of the assignment will *very* closely track the ideas/tools discussed today (and Thursday).
- Announcements will be made on Piazza when the assignment is released. It will also show up on Canvas.

Today's lecture

- We will jump right in and start drawing (somehow!)
  - Instead of dwelling on the theory of how to do this, let's first try by example! (we will still review the steps that brought us here, a bit later)
  - The objective is to get a "feel" of the drawing API, and also start getting an exposure to the complications of what it takes to draw something practical
  - Ask questions! Remember, that this will be the basis of your upcoming programming assignment!

A simple drawing example (via JSbin)

jsbin.com/bovuhok

Week2/Demo0

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

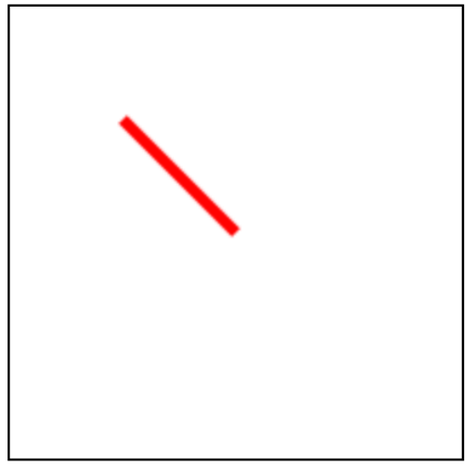
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS ☑



A simple drawing example (via JSbin)

jsbin.com/bovuhok

[Logistics] When I include a link colored like this, you can go to this URL for a live demonstration

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

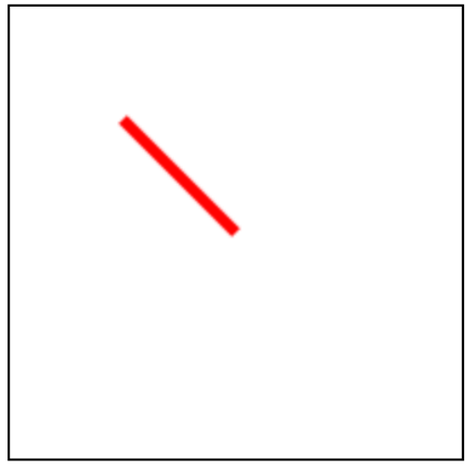
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



A simple drawing example (via JSbin)

jsbin.com/bovuhok

Week2/Demo0

[Logistics] Other times, a link in blue will point you to a GitHub repository directory for a demo

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

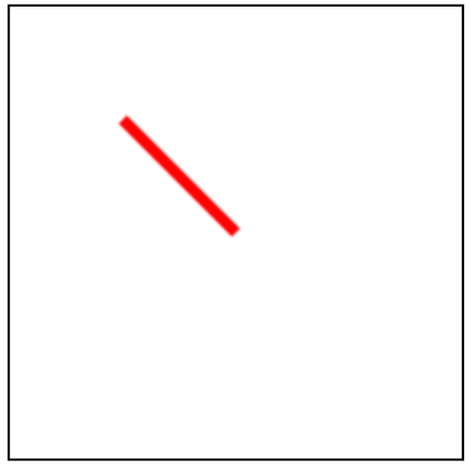
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output Run with JS Auto-run JS



A simple drawing

This is what you see when you visit this page on a browser
(yes, we will do our drawing in a browser!)
You might have to click "Edit in JSBin" to get to editable code

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

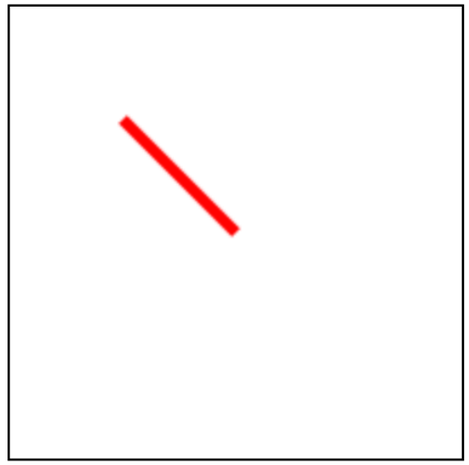
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS ☑



A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

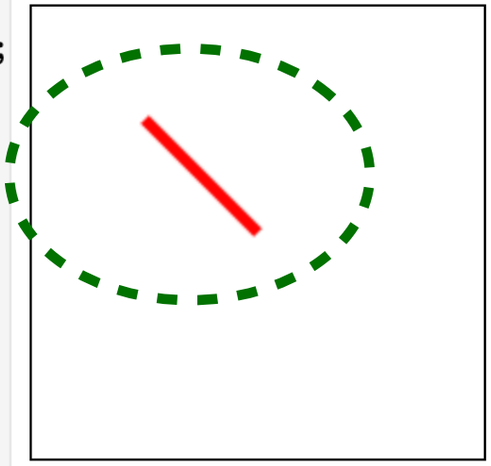
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



This demo draws a line

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

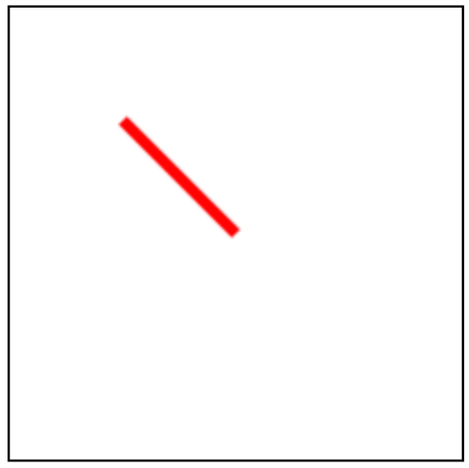
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS ☑



This demo draws a red line with width approximately 5 pixel units

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

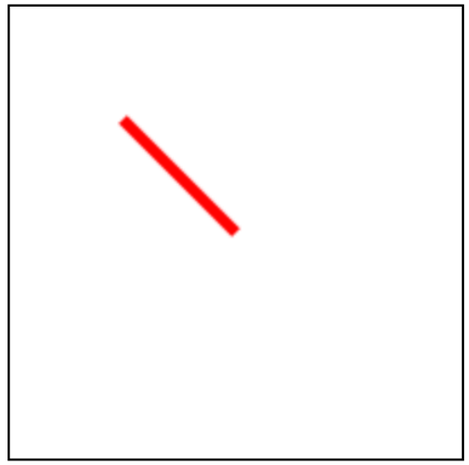
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



The line consists of line segments; in fact we have just a single line segment that starts at coordinates (50,50) and ends at coordinates (100,100)

Did you catch the ambiguity in this statement …?

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

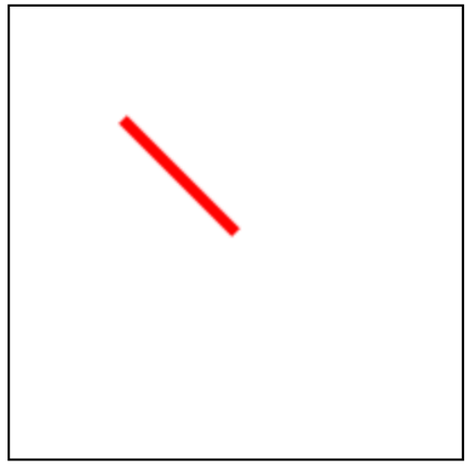
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



What exactly do we mean by "at coordinates (50,50)" ?
Or "at coordinates (100,100)" ?

(Formal definition later - in your readings - but let's work from your math experience and intuition …)

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

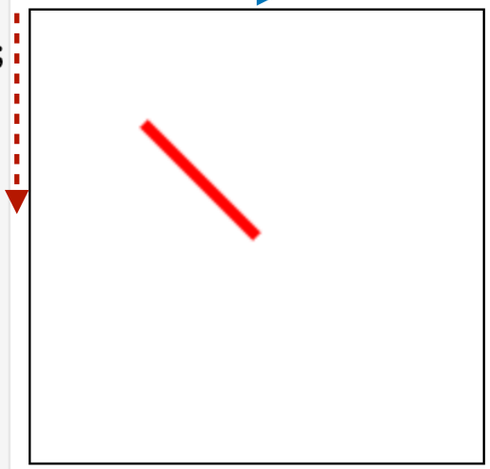
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



The two numbers in the coordinate pair designate offsets ("right", "down") from the top-leftmost corner of the drawing window. In this context, the numbers count "pixel units"

Queue up question … what if I'm trying to draw an image of a scene in the real world. How do I know what "pixel location" things are at?

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

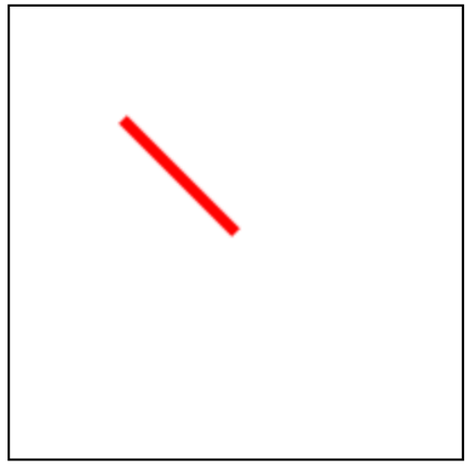
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS ☑



This entire drawing demo is a webpage!
The "output" window is what this webpage displays as ...

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

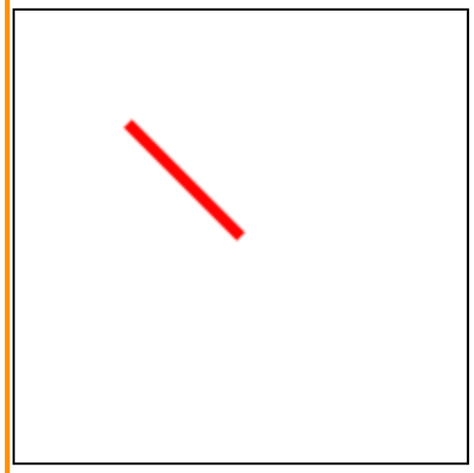
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



… while the code windows on the left/middle are the source code of that page

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share HTML

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

JavaScript ▾

```
var canvas
var context
// a thick
context.lin
context.str

// the actu
context.beg
context.mov
context.lin
context.str
```

In fact, you can get the page source as a single .html file via File>Download …

Page source as an html file:

```
<!DOCTYPE html>
<!--
Created using JS Bin
http://jsbin.com

Copyright (c) 2020 by sifakis (http://jsbin.com/bovuhok/3/edit)

Released under the MIT license: http://jsbin.mit-license.org
-->
<meta name="robots" content="noindex">
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
<script id="jsbin-javascript">
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
</script>
</body>
</html>
```

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share HTML

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

JavaScript ▾

```
var canvas
var context
// a thick
context.lin
context.str

// the actu
context.beg
context.mov
context.lin
context.str
```

Note that the page source includes HTML code (in black) and JavaScript code (in purple)

(we'll introduce elements of both soon, don't worry!)

Page source as an html file:

```
<!DOCTYPE html>
<!--
Created using JS Bin
http://jsbin.com

Copyright (c) 2020 by sifakis (http://jsbin.com/bovuhok/3/edit)

Released under the MIT license: http://jsbin.mit-license.org
-->
<meta name="robots" content="noindex">
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
<script id="jsbin-javascript">
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
</script>
</body>
</html>
```

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share HTML

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
```

JavaScript ▾

```
var canvas
var context
// a thick
context.lin
context.str

// the actu
context.beg
t.mov
t.lin
t.str
```

You can actually "run" this .html file directly, by opening it with a browser (this is actually the preferred, and only reasonable way to code up a page …)

But for in-class demonstrations I'm using JSBin (jsbin.com) as a live-pastebin

Page source as an html file:

```
<!DOCTYPE html>
<!--
Created using JS Bin
http://jsbin.com

Copyright (c) 2020 by sifakis (http://jsbin.com/bovuhok/3/edit)

Released under the MIT license: http://jsbin.mit-license.org
-->
<meta name="robots" content="noindex">
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
<script id="jsbin-javascript">
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
</script>
</body>
</html>
```

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

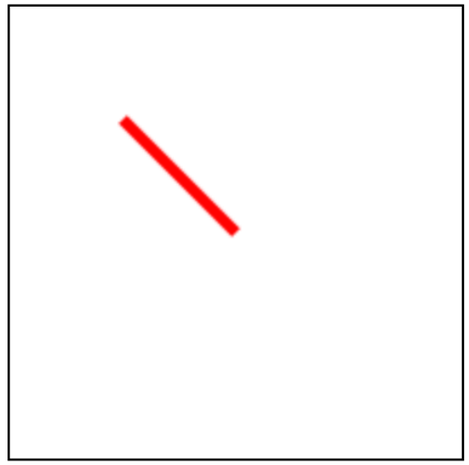
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



What does JSBin do? (remember, this is for my convenience of demonstration … you should resist the urge to develop in this environment; you'll see why)

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

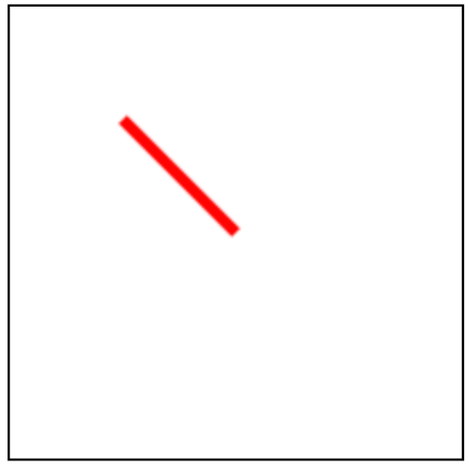
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



What does JSbin do? (remember, this is for my convenience of demonstration … you should resist the urge to develop in this environment; you'll see why)

JSbin separates out (it's a trivial exercise …) the JavaScript code from the containing HTML code, and presents them in two editable windows for web-editing

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

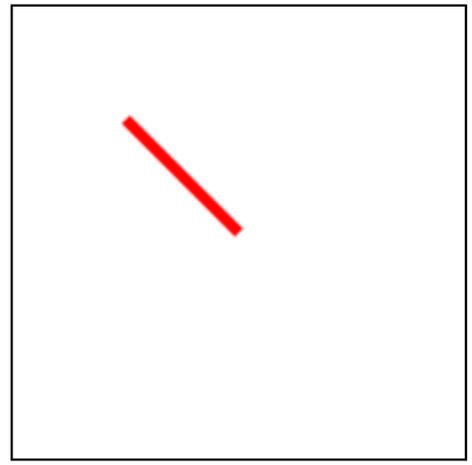
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS ☑



What does JSbin do? (remember, this is for my convenience of demonstration … you should resist the urge to develop in this environment; you'll see why)

It also juxtaposes the code with a "live run" of the webpage, which is either re-run automatically as you edit (if you check the box), or manually when you click the button.

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

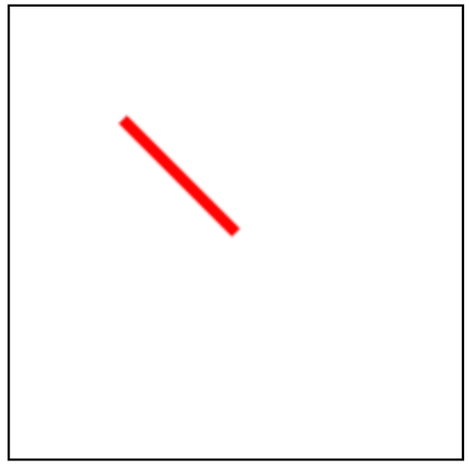
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



What does JSbin do? (remember, this is for my convenience of demonstration … you should resist the urge to develop in this environment; you'll see why)

When you "save" a newly created, or re-edited page, you will get a unique link that hashes your changes (repeat warning: do not develop in JSbin!!)

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

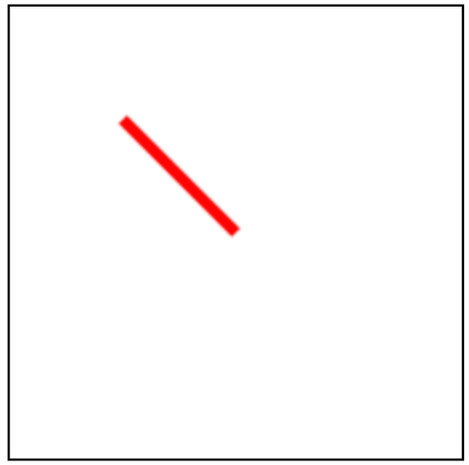
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



A walk-through of the HTML component of this webpage (again, by example …)

You have the typical <html> element, for a page, the <head> element (here just listing a page title), and the <body> element with the main page content.

(There is also a <script> element in the .html file, which JSbin automatically inserts, while embedding the JavaScript code within it)

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

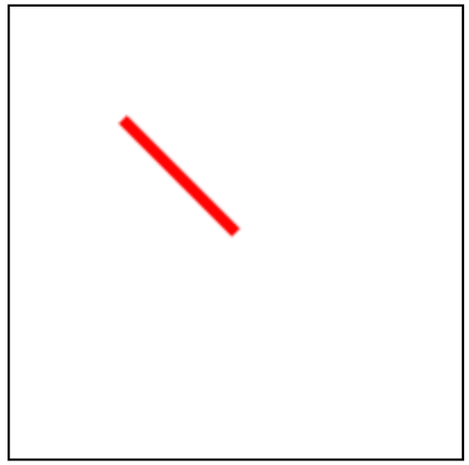
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



A walk-through of the HTML component of this webpage (again, by example …)

The <canvas> element is the sole visual constituent of this page. Essentially, a "drawable" image (contrast with an <img> element …)

The canvas element has an ID (we could have had multiple in a page), and properties that specify dimensions (in pixels), border style and color.

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

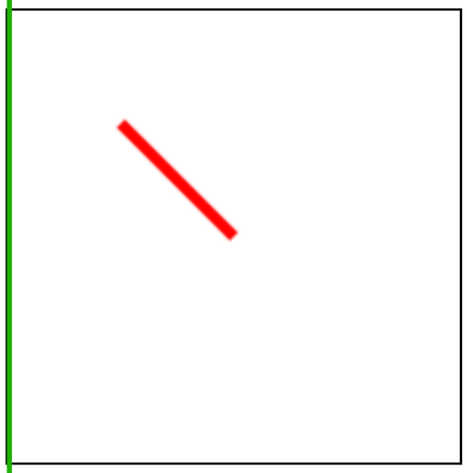
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



Time for some JavaScript …

JavaScript is an object oriented, members of objects (like those of "document") are accessed via the dot "." operator

The type of objects or members is automatically inferred (see how the "var" keyword is used)

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

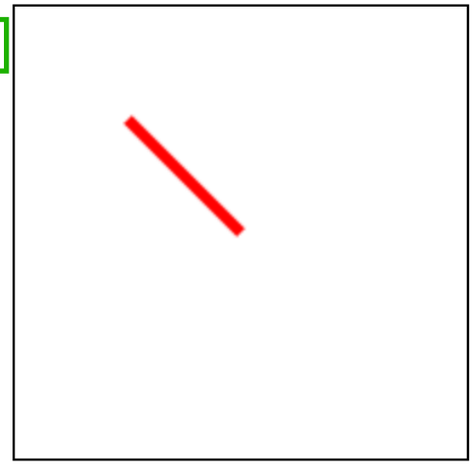
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



Time for some JavaScript …

Here, the "canvas" variable is an object that encapsulates the instance of our drawing space that has been assigned the ID "myCanvas" (in the HTML code)

Remember, we could have had several canvases … distinguished by IDs

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

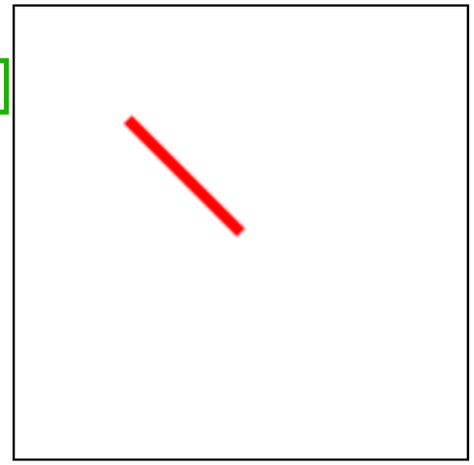
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



Time for some JavaScript …

The context is an API (for drawing) with an associated state.
There are different types of drawing APIs that can be used ('2d', 'webgl', etc)

Most of the drawing operations we will be using are methods/functions of the 2D drawing API ….

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

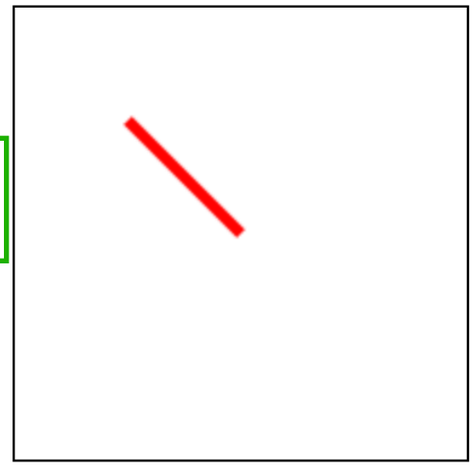
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



Time for some JavaScript …

Member variables of the context can be used to control the color and thickness of the drawing pen

A simple drawing example (via JSbin)

jsbin.com/bovuhok

https://jsbin.com/bovuhok/edit

Apps

File ▾ Add library Share | HTML | CSS | JavaScript | Console | Output | Account Blog Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

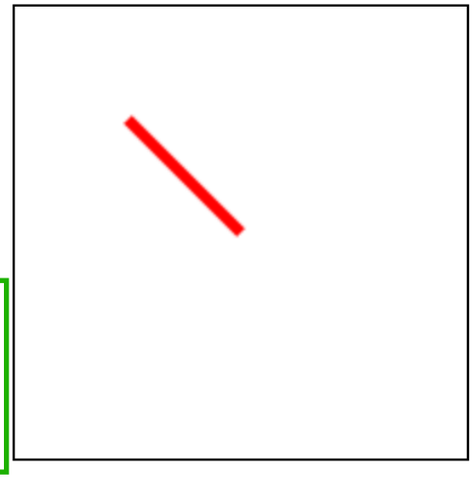
JavaScript ▾

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// a thick red line
context.lineWidth = 5;
context.strokeStyle = "red";

// the actual line
context.beginPath();
context.moveTo(50,50);
context.lineTo(100,100);
context.stroke();
```

Output | Run with JS | Auto-run JS



Time for some JavaScript …

And paths (with line-segment, or even curved components can be prescribed by point-to-point strokes. The stroke() method implies a drawn line path, as opposed to a "filled" shape (that would be the fill() method).

Why develop by editing .html files (as opposed to JSBin)

- Most important: There are excellent debuggers built-in to browsers (and you can use them with .html pages)
  - If you run them over JSbin, you are debugging jsbin.com, NOT your program
- JSbin is severely limited in what it can support within it (that's why we will only use it to get started, and primarily for demonstrations)
- We will describe better development and code maintenance practices (version control) later on, and none of these are viable with JSbin

Additional examples (flash preview)

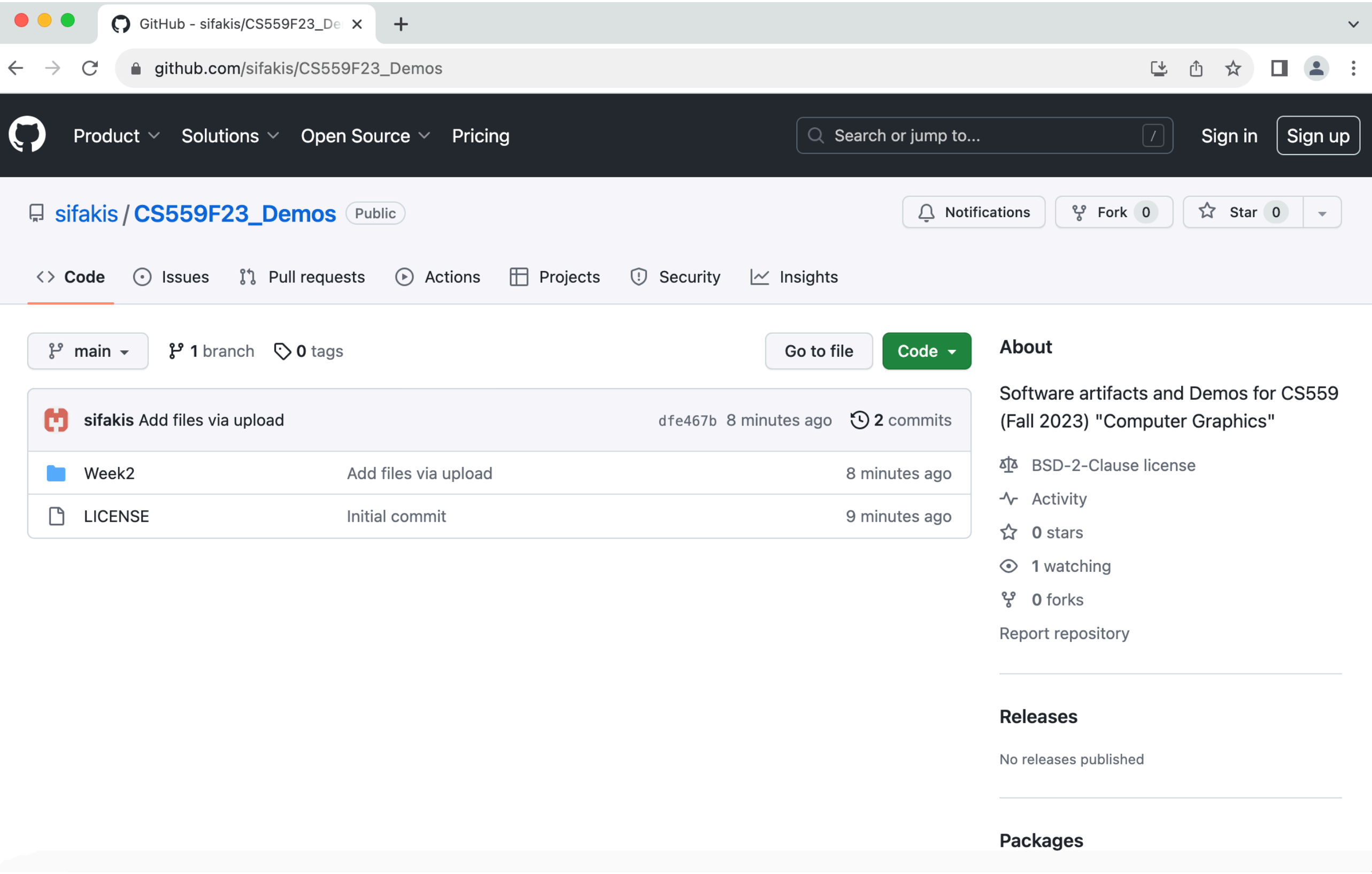
jsbin.com/suhujar

Week2/Demo1

- An "L-shaped" polygon with X and Y axes. Features demonstrated:
  - Function calls
  - Closed paths (and drawing filled polygons)
  - Passing parameters to functions

Quick practical notes (more on Thursday)

- All the examples we show in class will be "duplicated" (or more accurately, "properly" implemented) in a public GitHub repository that you can clone or download.
- Repeat reminder: DON'T develop in jsbin! Instead, download the proper source code, and write/debug using that copy.
- We'll see a flash preview of what debugging means in a little bit (way more on this in Thursday's lecture)

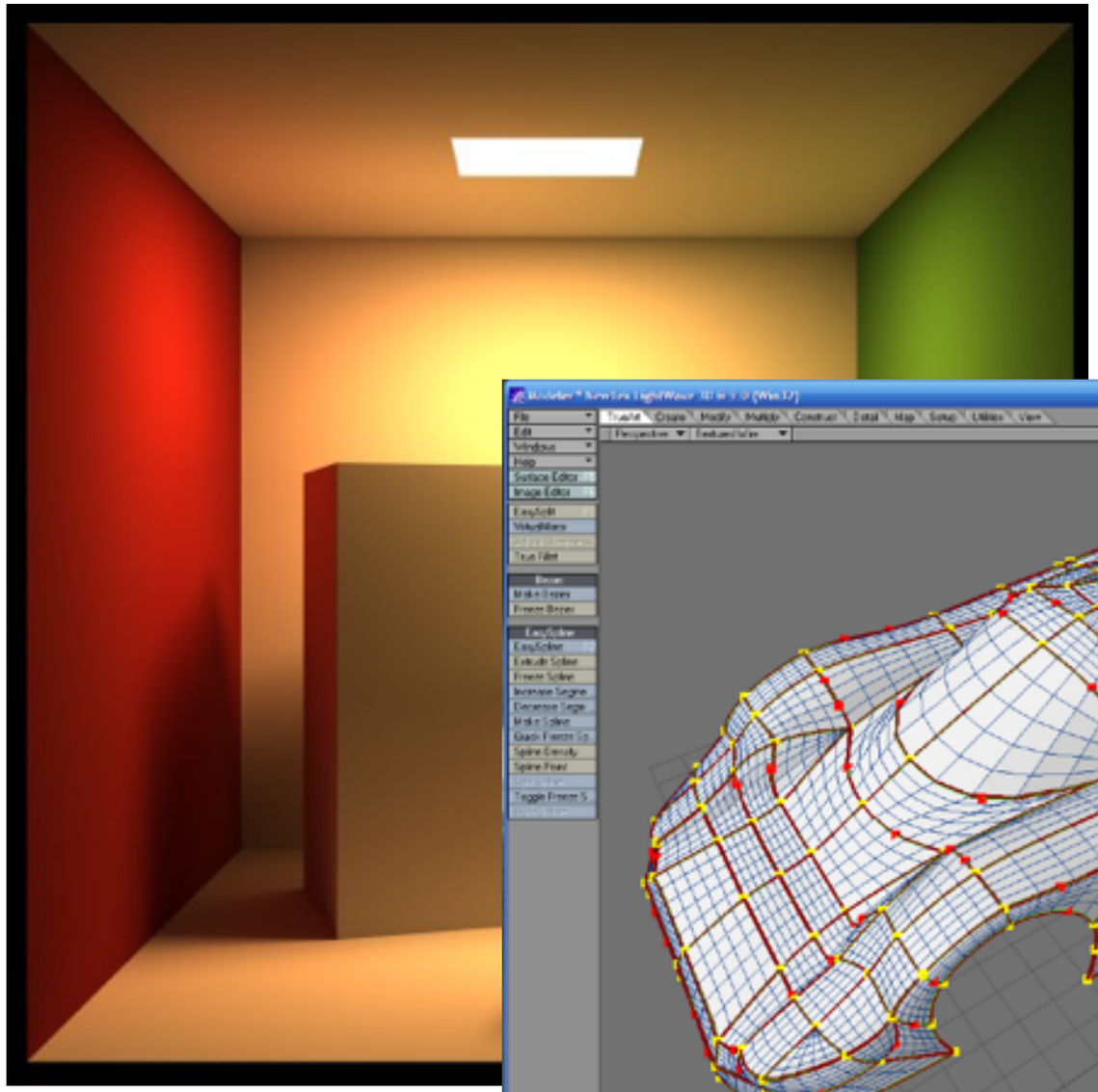

GitHub - sifakis/CS559F23_De
github.com/sifakis/CS559F23_Demos
Product
Solutions
Open Source
Pricing
Search or jump to...
Sign in
Sign up
sifakis / CS559F23_Demos
Public
Notifications
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
main
1 branch
0 tags
Go to file
Code
sifakis Add files via upload
dfe467b 8 minutes ago
2 commits
Week2
Add files via upload
8 minutes ago
LICENSE
Initial commit
9 minutes ago
About
Software artifacts and Demos for CS559 (Fall 2023) "Computer Graphics"
BSD-2-Clause license
Activity
0 stars
1 watching
0 forks
Report repository
Releases
No releases published
Packages

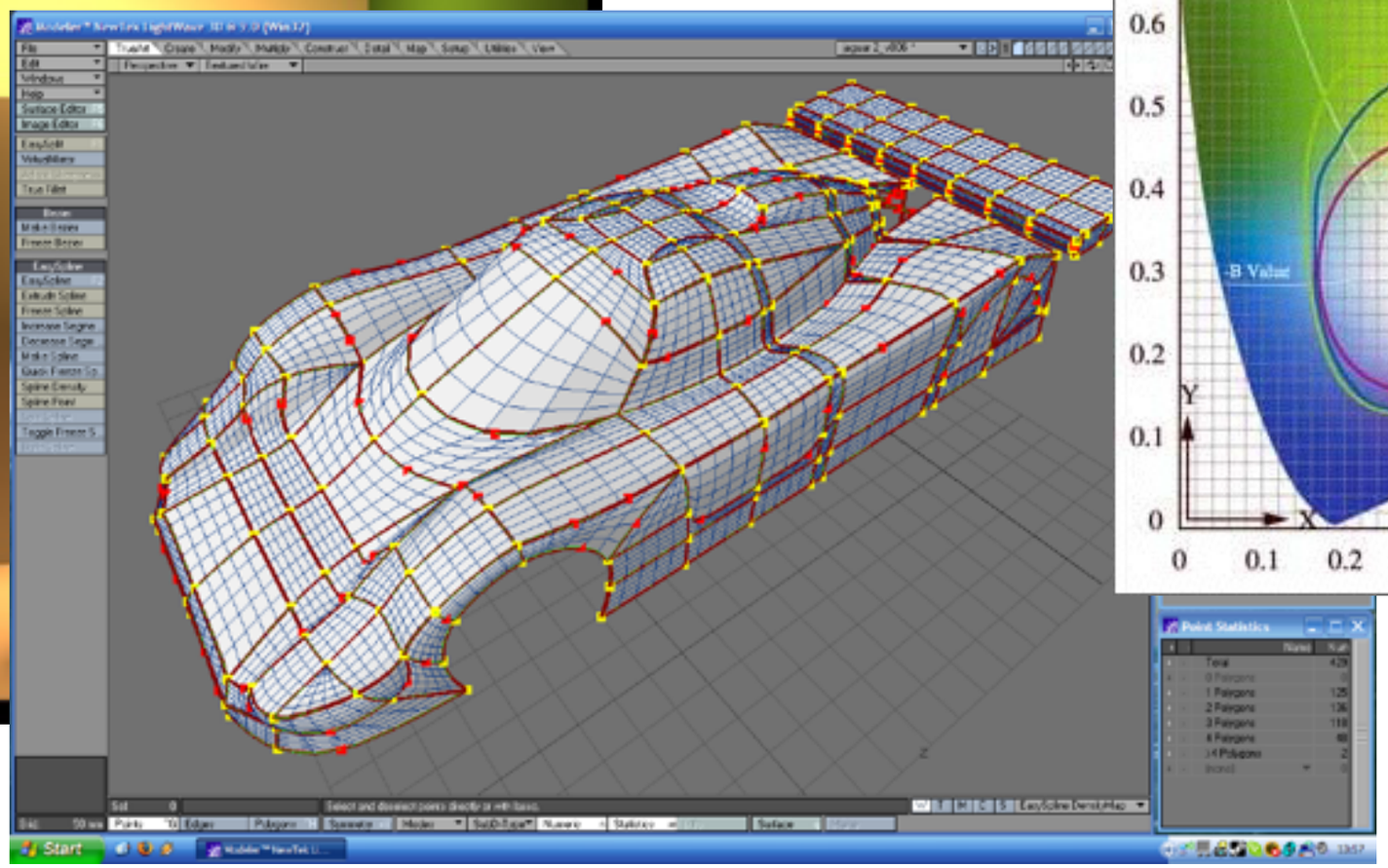
Additional examples (flash preview)

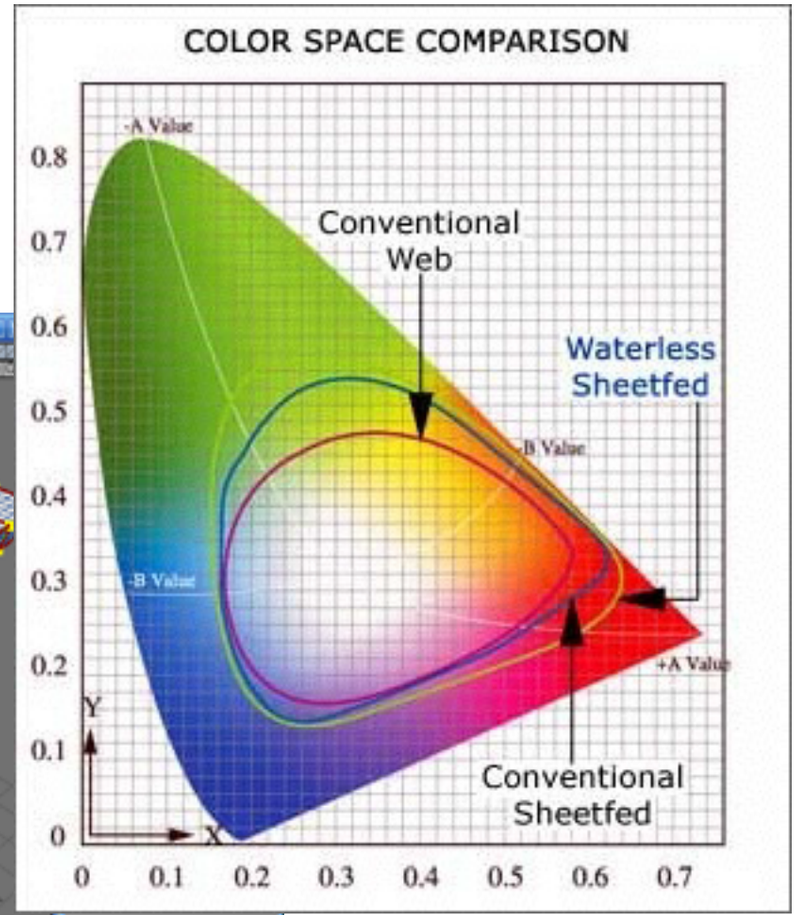
jsbin.com/fesukexori

Week2/Demo2

- An "L-shaped" polygon with axes and transforms
  Features demonstrated:
  - window.onload callback mechanism
  - Clearing the screen
  - Interface elements (sliders) and retrieving their values
  - EventListeners
  - Introduction (by example) to transforms.







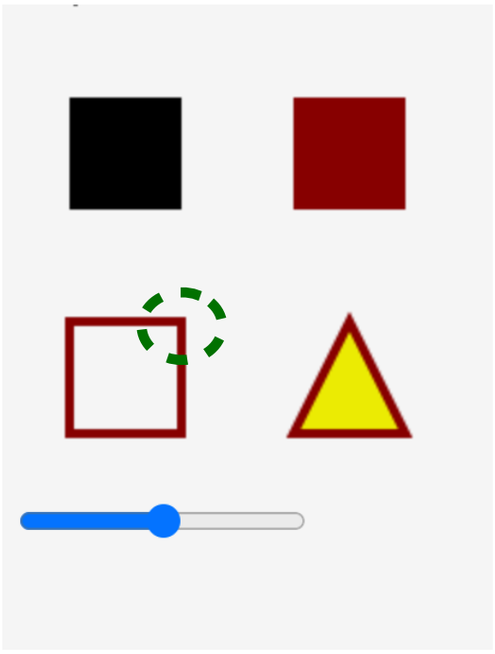
Lecture 2 : Where do I draw?
(Intro to 2D Canvas, drawing and interface elements)

Tuesday September 12th 2023

(Overflow slides)

Your first programming assignment

- Where do I find it?
  - Look at the Assignment Description page on Canvas which is also where you will turn in your work.
  - Will be posted no later than Thursday
  - TL;DR: Create a simple 2D drawing, as an html page (not in JSbin) and include both non-filled strokes and filled shapes. Use slider(s) to change something in the drawing. If submitting more than a single .html, put in a .zip.



Maybe moving the slider changes the location of this vertex …

Clone the repository, or download a .zip archive

GitHub - sifakis/CS559F23_De

github.com/sifakis/CS559F23_Demos

Product Solutions Open Source Pricing

Search or jump to... / Sign in Sign up

sifakis / **CS559F23_Demos** Public

Notifications Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights

main 1 branch 0 tags

Go to file Code

sifakis Add files via upload dfe467b 8 minutes ago 2 commits

Week2	Add files via upload	8 minutes ago
LICENSE	Initial commit	9 minutes ago

About

Software artifacts and Demos for CS559 (Fall 2023) "Computer Graphics"

BSD-2-Clause license

Activity

0 stars

1 watching

0 forks

Report repository

Releases

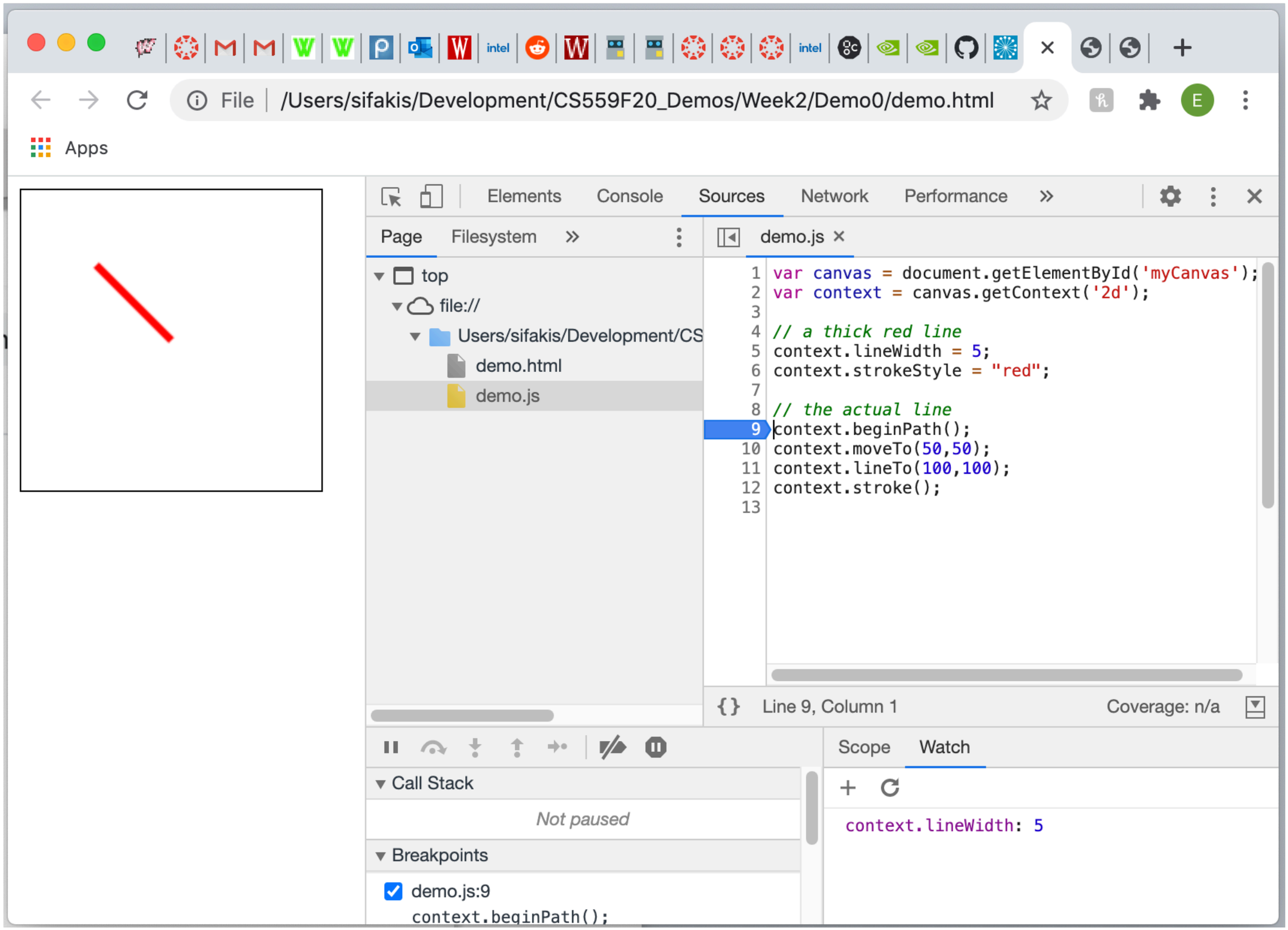
No releases published

Packages

Then, simply "open" the .html file with your browser …

Try right-click and then …
… "View Page Source" to get to the HTML source (browser dependent)
… click on "demo.js" to get the JavaScript source (here it's separate)
… or "Inspect" to launch the debugger (in Chrome)

The debugger within Chrome (similar in other browsers)



Is this the only way?

- Running your code in your browser, from a local copy, is infinitely better than via JSbin (for reasons discussed)
- Even better: Host the page on a web-server
    - This allows much broader functionality, as browsers are peculiar with how they allow manipulation of files if not run over a web server
    - Instead of hosting the page on the web (your homepage?) you can run it via a local server
    - We may talk about this option in greater detail, later (if the nature of assignments strongly suggests it)

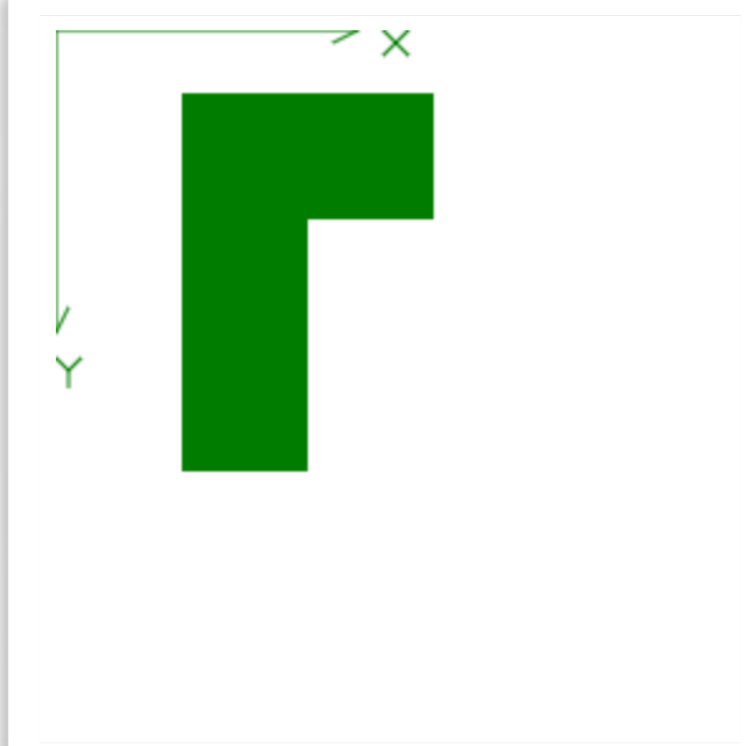
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
   }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}
draw();
```

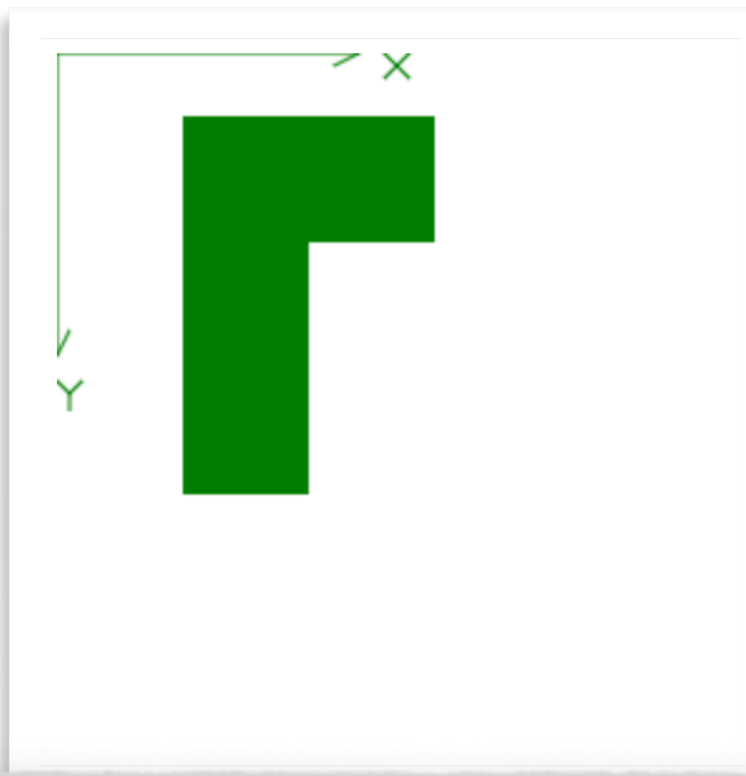
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Functions, nested functions, and calls (with parameters)

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
   }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}
draw();
```

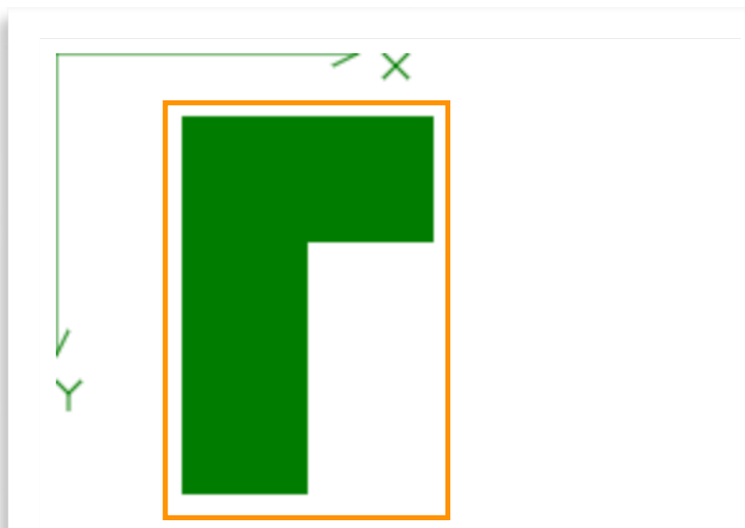
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Drawing closed, filled polygons, with specified fillStyle (vs strokeStyle)

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
   }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}
draw();
```

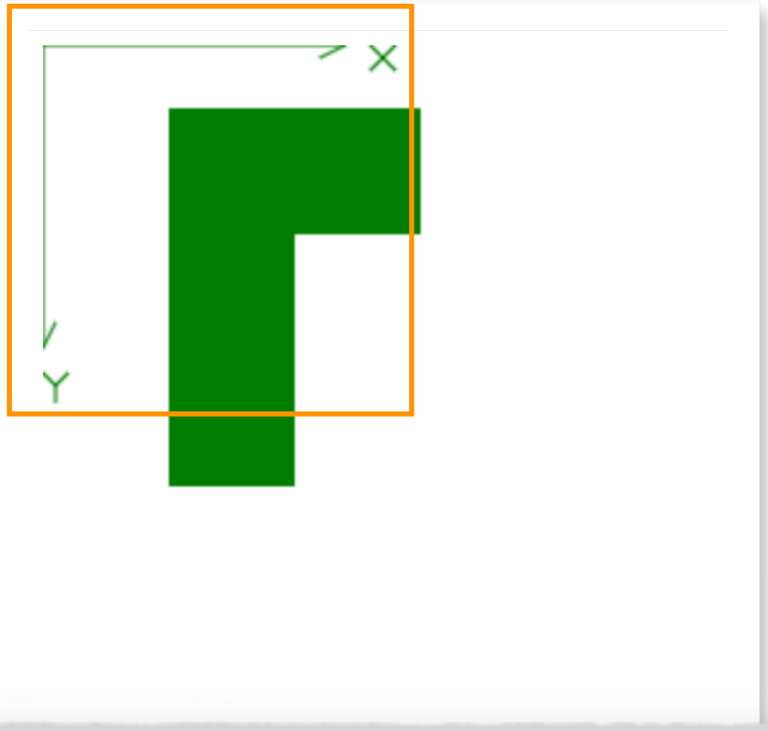
Additional features (JavaScript/Canvas)

jsbin.com/suhujar

Week2/Demo1

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coordinate axes and a filled shape</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Drawing several (disconnected) chains of line segments (notice moveTo/lineTo succession)

demo.js

```
var canvas = document.getElementById('myCanvas');

function draw() {
  var context = canvas.getContext('2d');

  function DrawLshape(color) {
    context.beginPath();
    context.fillStyle = color;
    context.moveTo(50,25);
    context.lineTo(150,25);
    context.lineTo(150,75);
    context.lineTo(100,75);
    context.lineTo(100,175);
    context.lineTo(50,175);
    context.closePath();
    context.fill();
  }

  function DrawAxes(color) {
    context.strokeStyle=color;
    context.beginPath();
    // Axes
    context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
    // Arrowheads
    context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
    context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
    // X-label
    context.moveTo(130,0);context.lineTo(140,10);
    context.moveTo(130,10);context.lineTo(140,0);
    // Y-label
    context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
    context.moveTo(5,135);context.lineTo(5,142);

    context.stroke();
  }

  // make sure you understand these
  DrawAxes("green");
  DrawLshape("green");

}
draw();
```

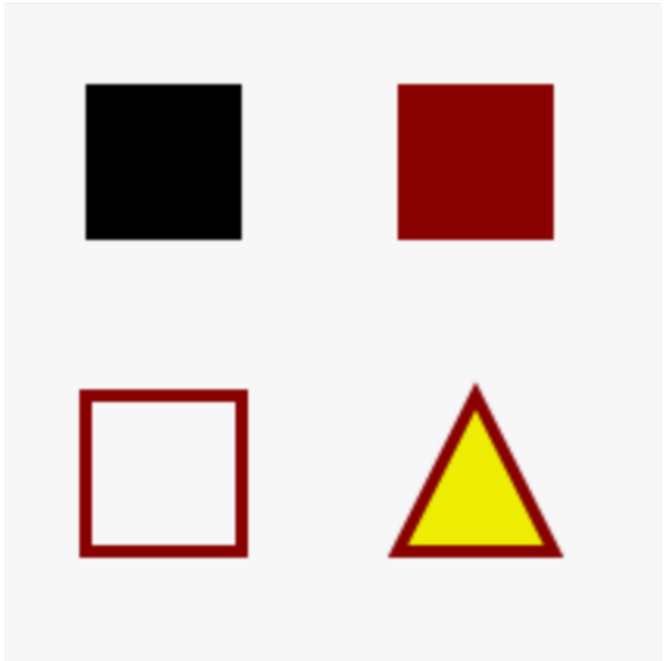
Additional features (JavaScript/Canvas)

jsbin.com/kisuhox

Week2/Demo3

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
// function draw() {
window.onload = function() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.rect(25,25,50,50);
    context.fill();

    context.beginPath();
    context.rect(125,25,50,50);
    context.fillStyle = "#800";
    context.fill();

    context.beginPath();
    context.rect(25,125,50,50);
    context.strokeStyle = "#800";
    context.lineWidth = 4;
    context.stroke();

    context.beginPath();
    context.moveTo(150,125);
    context.lineTo(125,175);
    context.lineTo(175,175);
    context.closePath();
    context.fillStyle="#EE0";
    context.fill();
    context.stroke();
};
// window.onload = draw;
```

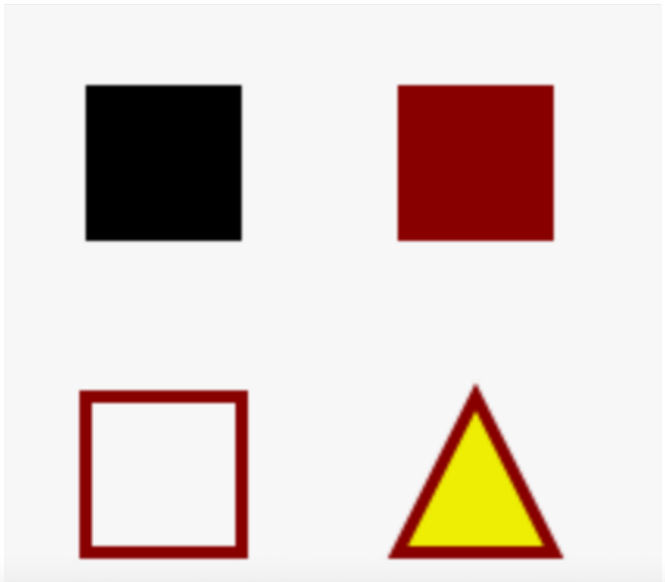
Additional features (JavaScript/Canvas)

jsbin.com/kisuhox

Week2/Demo3

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Specifying which function is run upon "loading" of the page (check tutorial)
Also, contrast 2 forms of syntax …

demo.js

```
// function draw() {
window.onload = function() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.rect(25,25,50,50);
    context.fill();

    context.beginPath();
    context.rect(125,25,50,50);
    context.fillStyle = "#800";
    context.fill();

    context.beginPath();
    context.rect(25,125,50,50);
    context.strokeStyle = "#800";
    context.lineWidth = 4;
    context.stroke();

    context.beginPath();
    context.moveTo(150,125);
    context.lineTo(125,175);
    context.lineTo(175,175);
    context.closePath();
    context.fillStyle="#EE0";
    context.fill();
    context.stroke();
};
// window.onload = draw;
```

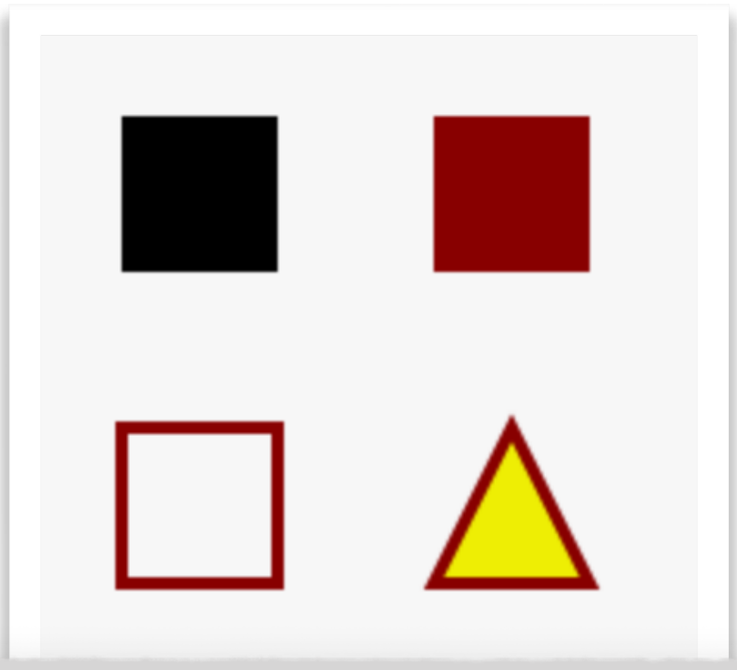
Additional features (JavaScript/Canvas)

jsbin.com/kisuhox

Week2/Demo3

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Drawing rectangles, and drawing polygons with different fill color and different outline color

demo.js

```
// function draw() {
window.onload = function() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.rect(25,25,50,50);
    context.fill();

    context.beginPath();
    context.rect(125,25,50,50);
    context.fillStyle = "#800";
    context.fill();

    context.beginPath();
    context.rect(25,125,50,50);
    context.strokeStyle = "#800";
    context.lineWidth = 4;
    context.stroke();

    context.beginPath();
    context.moveTo(150,125);
    context.lineTo(125,175);
    context.lineTo(175,175);
    context.closePath();
    context.fillStyle="#EE0";
    context.fill();
    context.stroke();
};
// window.onload = draw;
```

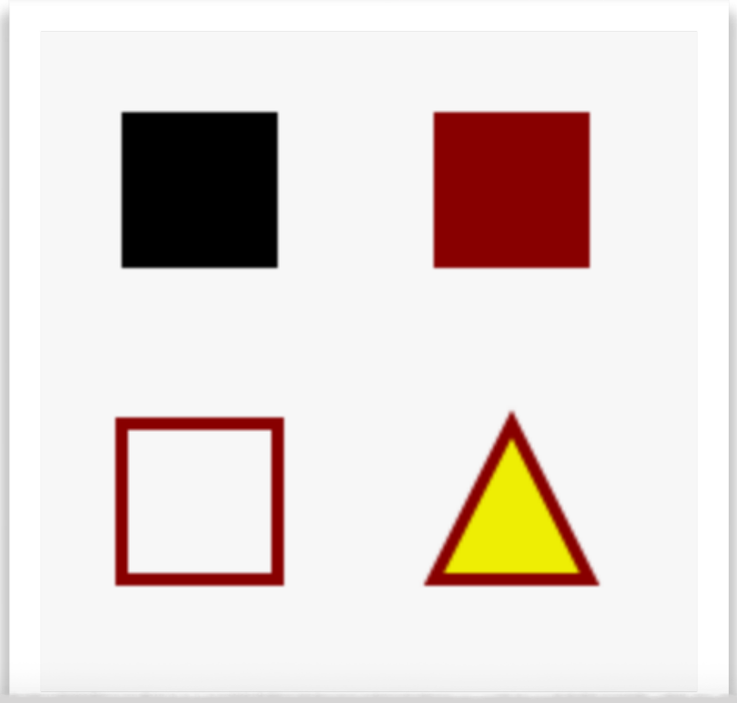
Additional features (JavaScript/Canvas)

jsbin.com/kisuhox

Week2/Demo3

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>A few static shapes</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="200" height="200">
    </canvas>
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Note alternate specification for color (see this tutorial as well)

demo.js

```
// function draw() {
window.onload = function() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.rect(25,25,50,50);
    context.fill();

    context.beginPath();
    context.rect(125,25,50,50);
    context.fillStyle = "#800";
    context.fill();

    context.beginPath();
    context.rect(25,125,50,50);
    context.strokeStyle = "#800";
    context.lineWidth = 4;
    context.stroke();

    context.beginPath();
    context.moveTo(150,125);
    context.lineTo(125,175);
    context.lineTo(175,175);
    context.closePath();
    context.fillStyle="#EE0";
    context.fill();
    context.stroke();
};
// window.onload = draw;
```

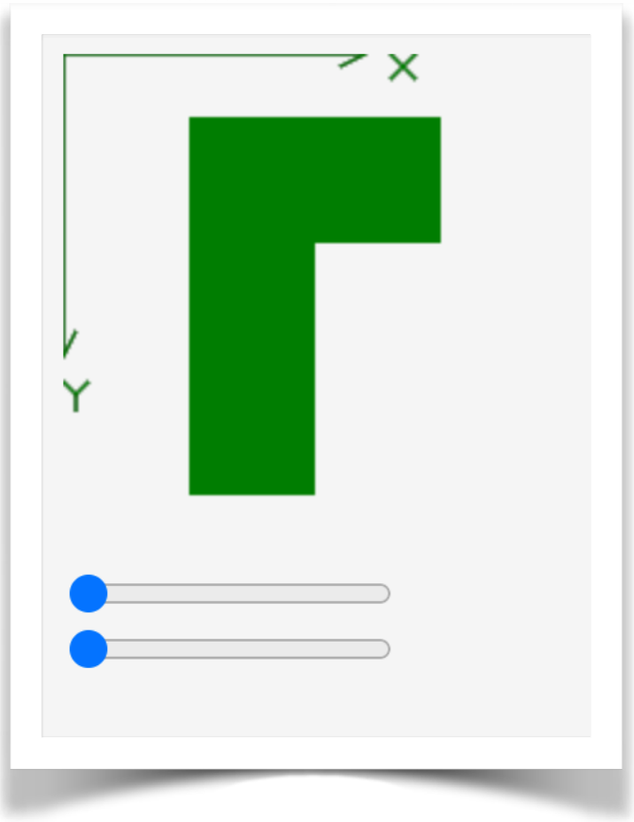
Additional features (JavaScrip

jsbin.com/fesukexori

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



demo.js

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
     }

    // make sure you understand these

    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

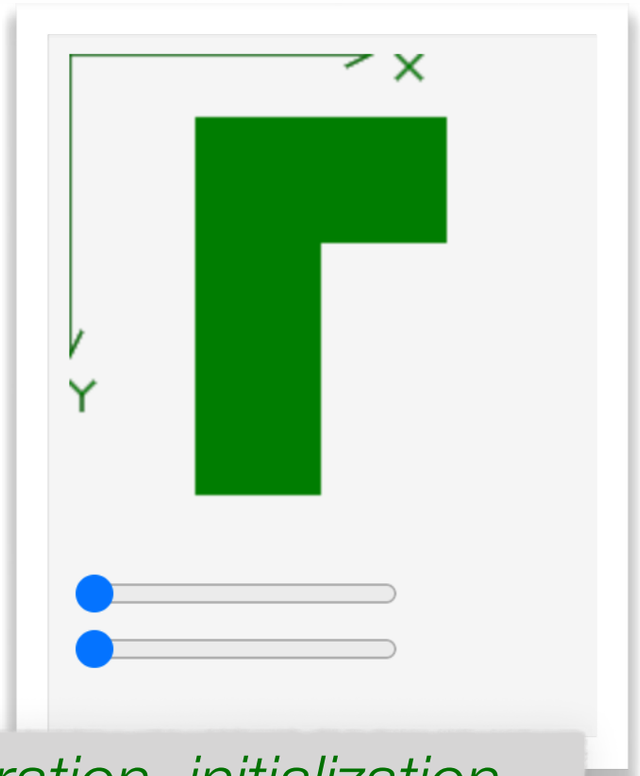
Additional features (JavaScrip

jsbin.com/fesukexori

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Sliders (declaration, initialization, retrieval of values)

demo.js

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
     }

    // make sure you understand these

    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```

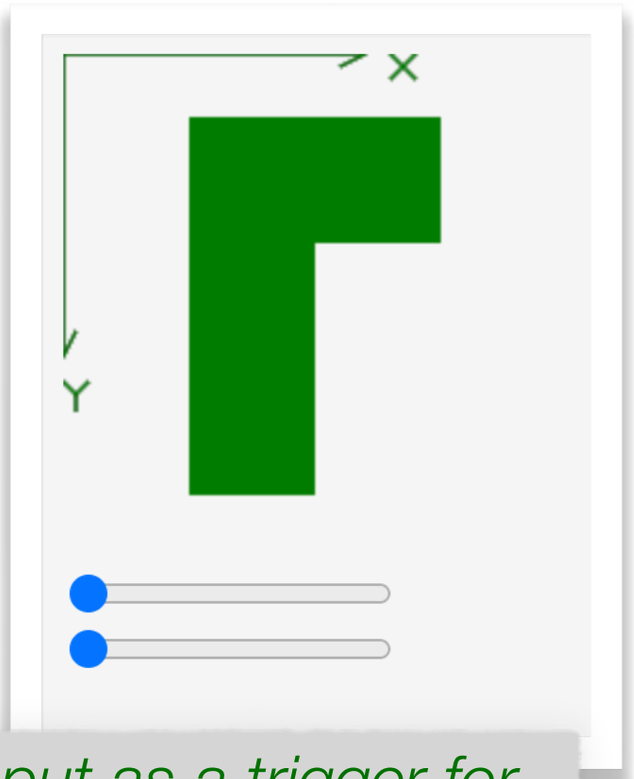
Additional features (JavaScrip

jsbin.com/fesukexori

Week2/Demo2

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple demonstration of slider interface</title>
  </head>
  <body>
    <canvas id="myCanvas"
        width="400" height="400">
    </canvas>
    <br/>
    <input id="slider1" type="range" min="0" max="100" />
    <br/>
    <input id="slider2" type="range" min="0" max="100" />
    <script src="demo.js" id="module"></script>
  </body>
</html>
```



Using slider input as a trigger for re-drawing (again, tutorial)

demo.js

```
function setup() {
  var canvas = document.getElementById('myCanvas');
  var slider1 = document.getElementById('slider1');
  slider1.value = 0;
  var slider2 = document.getElementById('slider2');
  slider2.value = 0;
  function draw() {
    var context = canvas.getContext('2d');
    canvas.width = canvas.width;
    // use the sliders to get various parameters
    var dx = slider1.value;
    var dy = slider2.value;

    function DrawLshape(color) {
      context.beginPath();
      context.fillStyle = color;
      context.moveTo(50,25);context.lineTo(150,25);context.lineTo(150,75);
      context.lineTo(100,75);context.lineTo(100,175);context.lineTo(50,175);
      context.closePath();
      context.fill();
    }

    function DrawAxes(color) {
      context.strokeStyle=color;
      context.beginPath();
      // Axes
      context.moveTo(120,0);context.lineTo(0,0);context.lineTo(0,120);
      // Arrowheads
      context.moveTo(110,5);context.lineTo(120,0);context.lineTo(110,-5);
      context.moveTo(5,110);context.lineTo(0,120);context.lineTo(-5,110);
      // X-label
      context.moveTo(130,0);context.lineTo(140,10);
      context.moveTo(130,10);context.lineTo(140,0);
      // Y-label
      context.moveTo(0,130);context.lineTo(5,135);context.lineTo(10,130);
      context.moveTo(5,135);context.lineTo(5,142);

      context.stroke();
    }

    // make sure you understand these

    DrawAxes("black");
    context.save();
    context.translate(dx,dy);
    DrawAxes("green");
    DrawLshape("green");
    context.restore();
  }
  slider1.addEventListener("input",draw);
  slider2.addEventListener("input",draw);
  draw();
}
window.onload = setup;
```