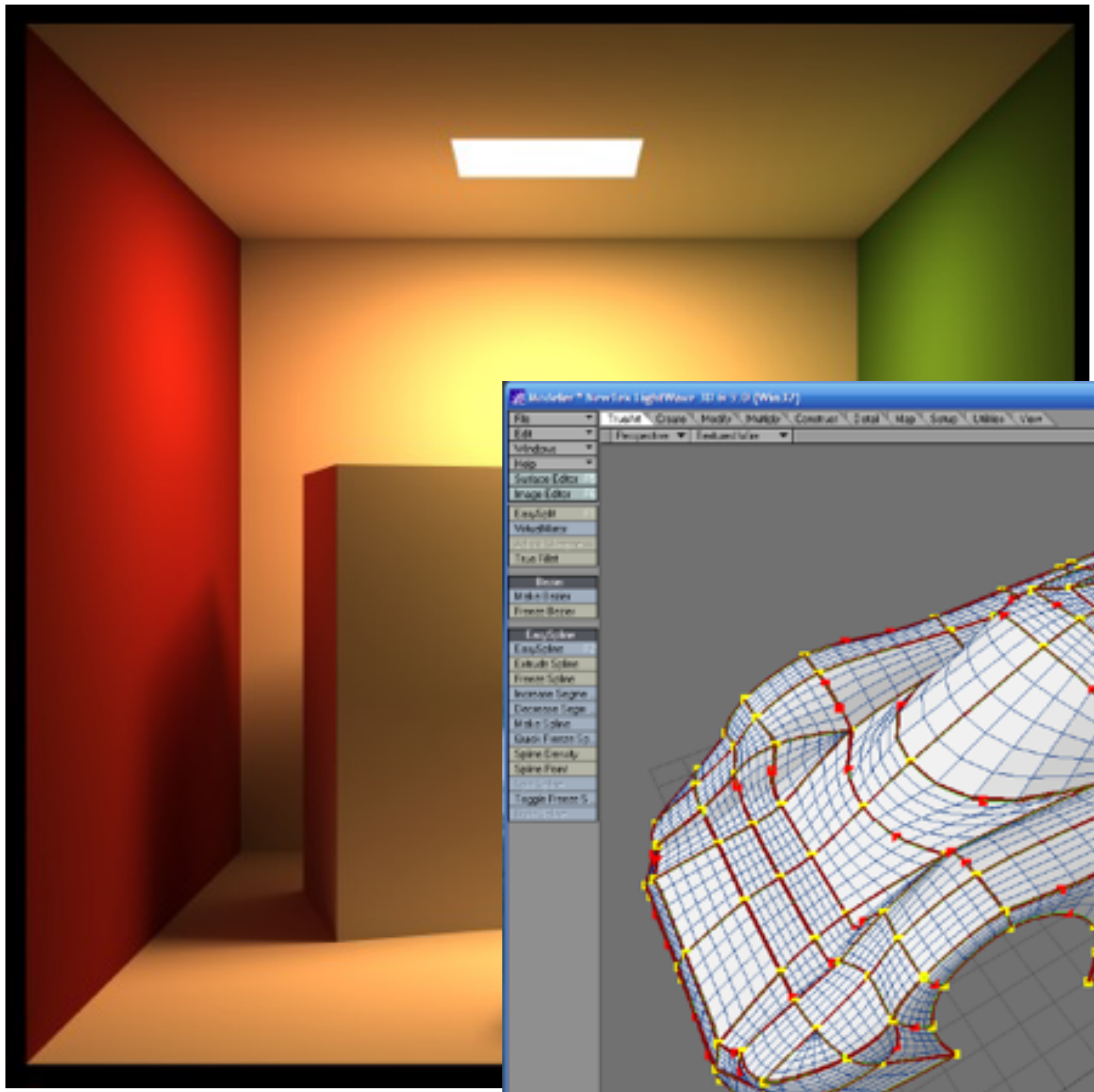
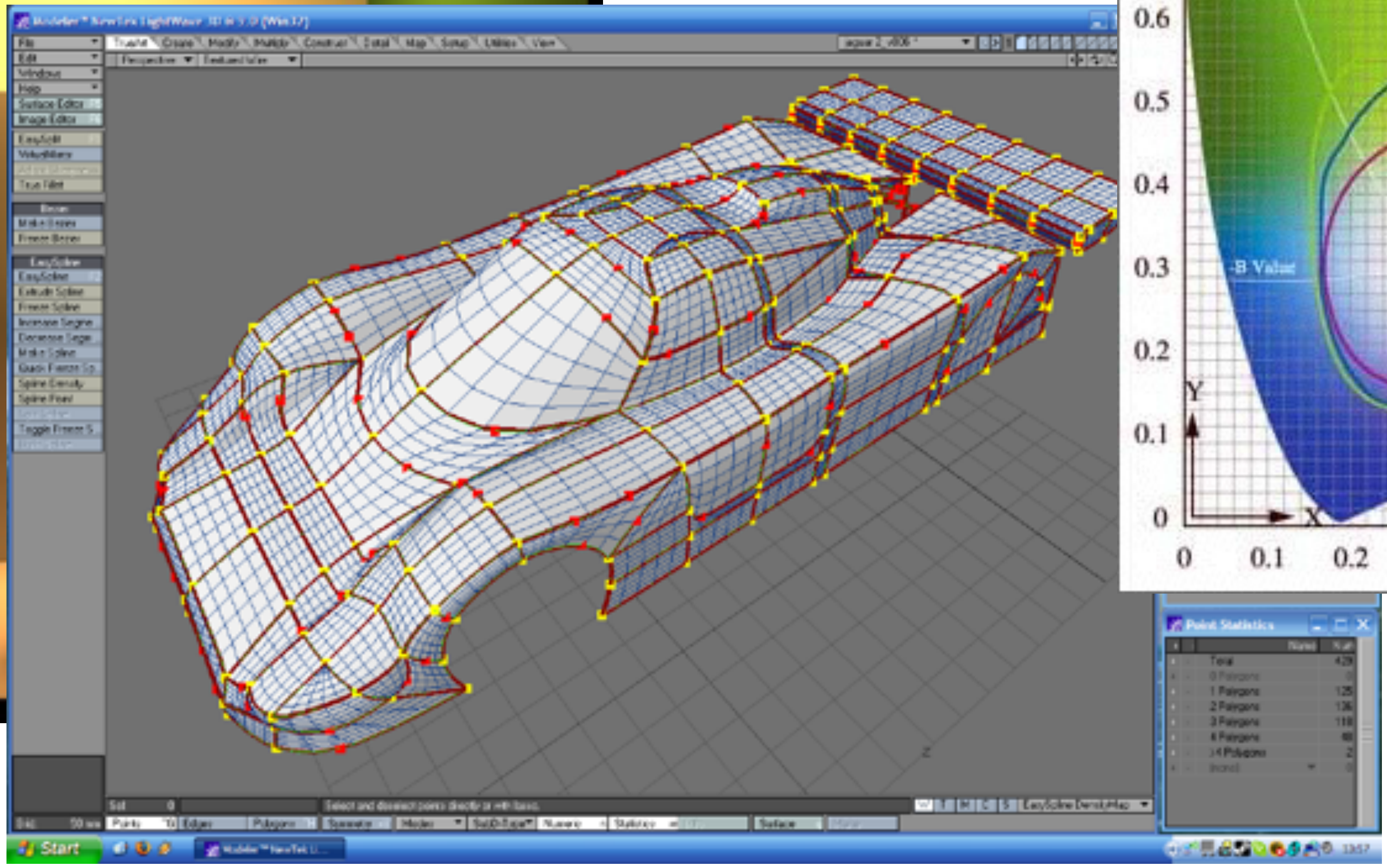
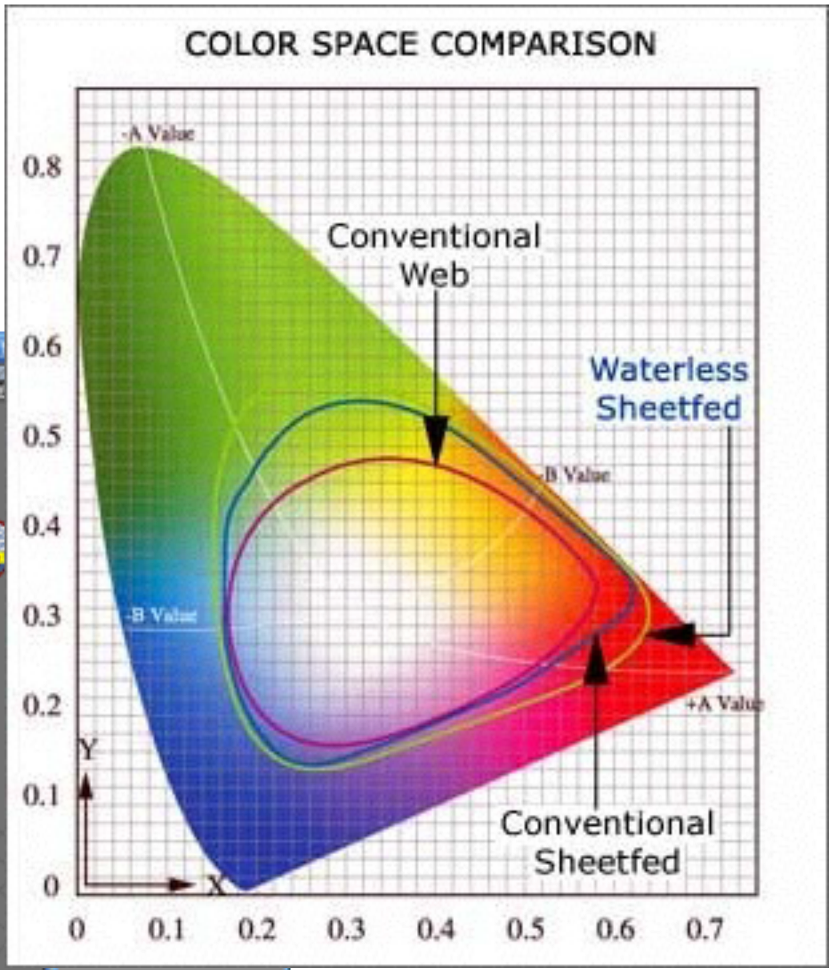






Lecture 2 : Where do I draw?
(Intro to 2D Canvas, drawing and interface elements)

Tuesday September 14th 2021

Pre-lecture logistics and announcements

- We (hopefully) have accommodated everyone on the waitlist; ping us on Piazza if you still want to get on (no guarantees - but we'll try in earnest!)
- Bear with us while we work with issues (WordPress...) with respect to the class website. This *will* have a resolution by next lecture (even if we have to post everything on Canvas, instead)
- Hopefully you have signed up for Piazza (and maybe used it, if you had questions!)

Pre-lecture logistics and announcements

- 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.

Pre-lecture logistics and announcements

- A question by some of the students (piazza, email, elsewhere...) "Is it ok to attend some/any/all of the lectures online?"
- "Quick" answer(s)
  - I wish it was possible, but we don't have the resources.
  - Lectures slides are available, lecture recordings are not.
  - Any accommodations will be on a truly exceptional basis (e.g. extra office hours or pointers to extra materials)
  - Office hours shouldn't be used to bridge the gap between slides and the in-class lecture experience.

Pre-lecture logistics and announcements

- A question by some of the students (piazza, email, elsewhere...) "Is it ok to attend some/any/all of the lectures online?"
- Slightly more nuanced answers:
  - Occasional absence (<10%) shouldn't be difficult to compensate for (readings, friends' notes, office hours).
  - We won't be taking attendance, we trust you'll do your best to participate in our in-class lectures, and not use office hours to make up for persistent absences.
  - If your circumstances suggest you will predictably miss many in-person lectures, maybe reconsider taking 559.

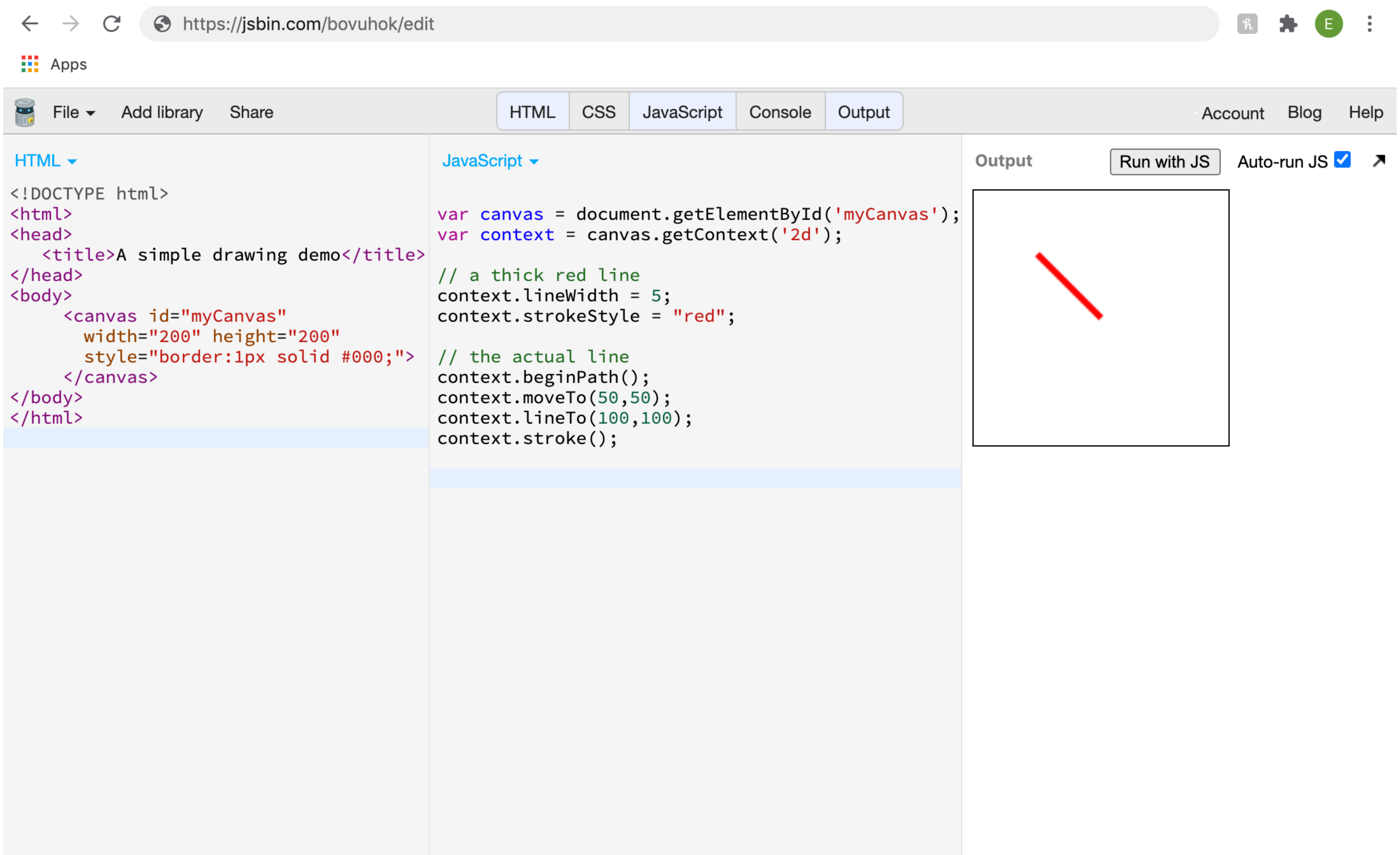
Today's lecture

- 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



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

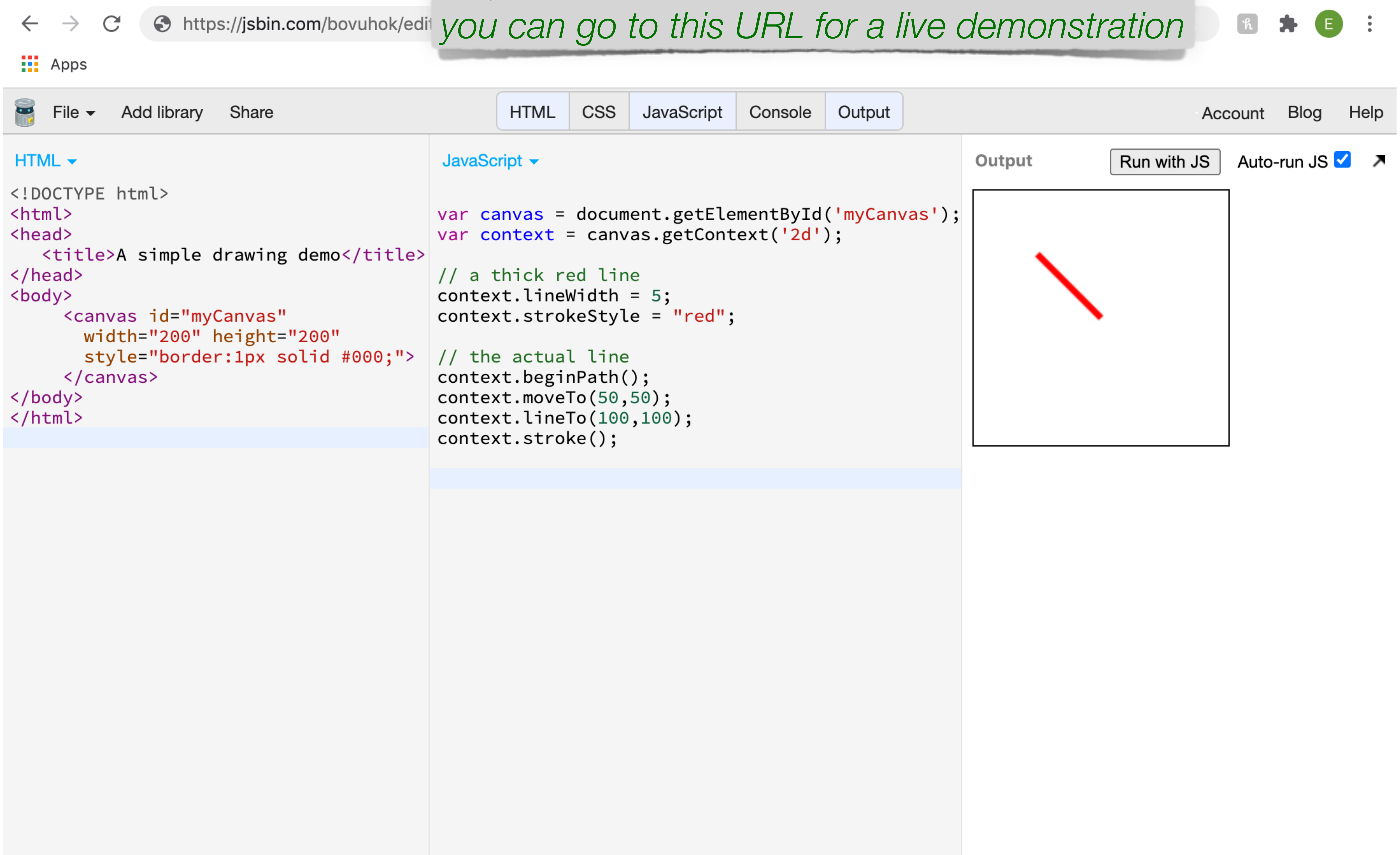
```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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

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



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

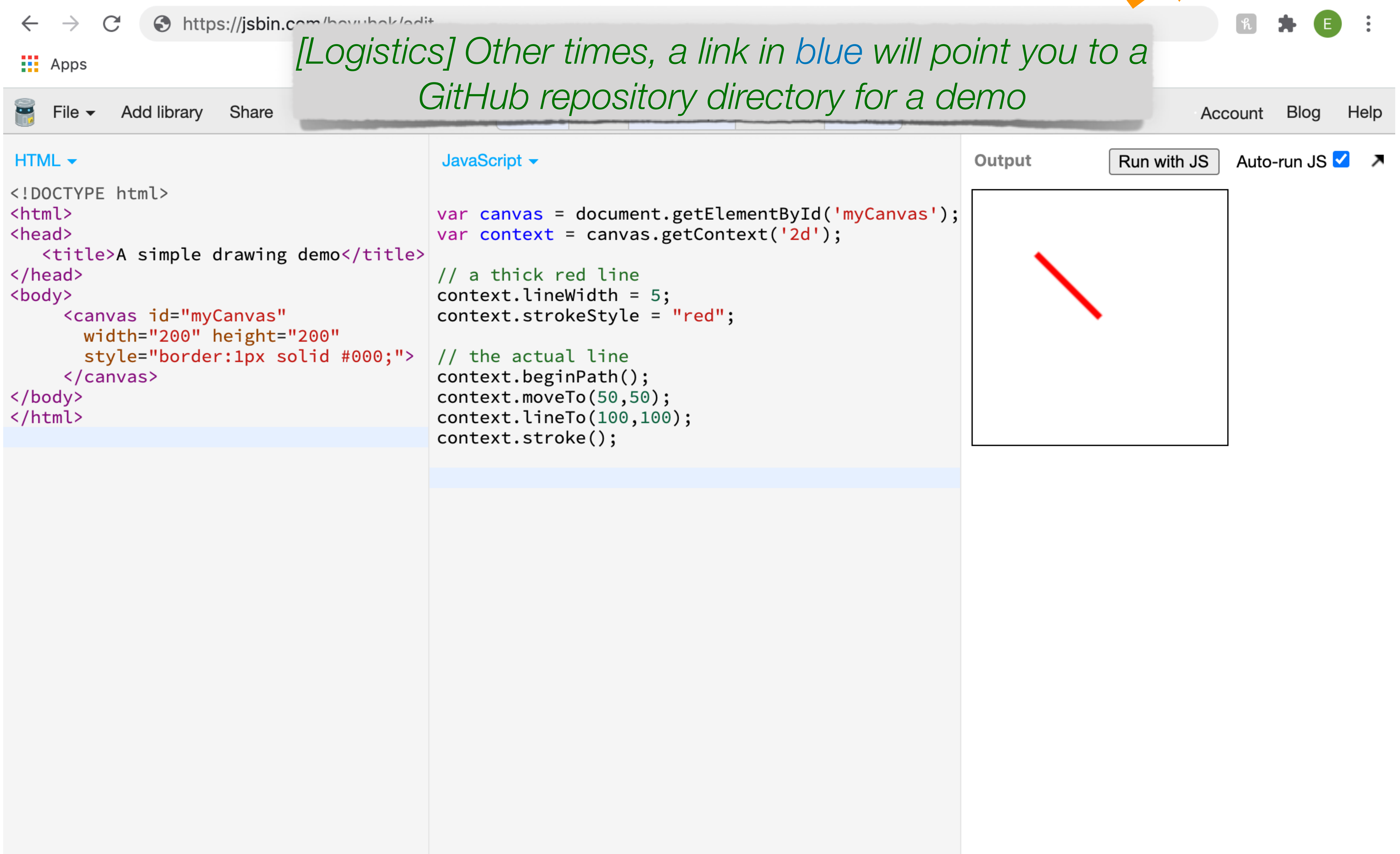
```javascript
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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

A simple drawing example (via JSbin)

jsbin.com/bovuhok

Week2/Demo0



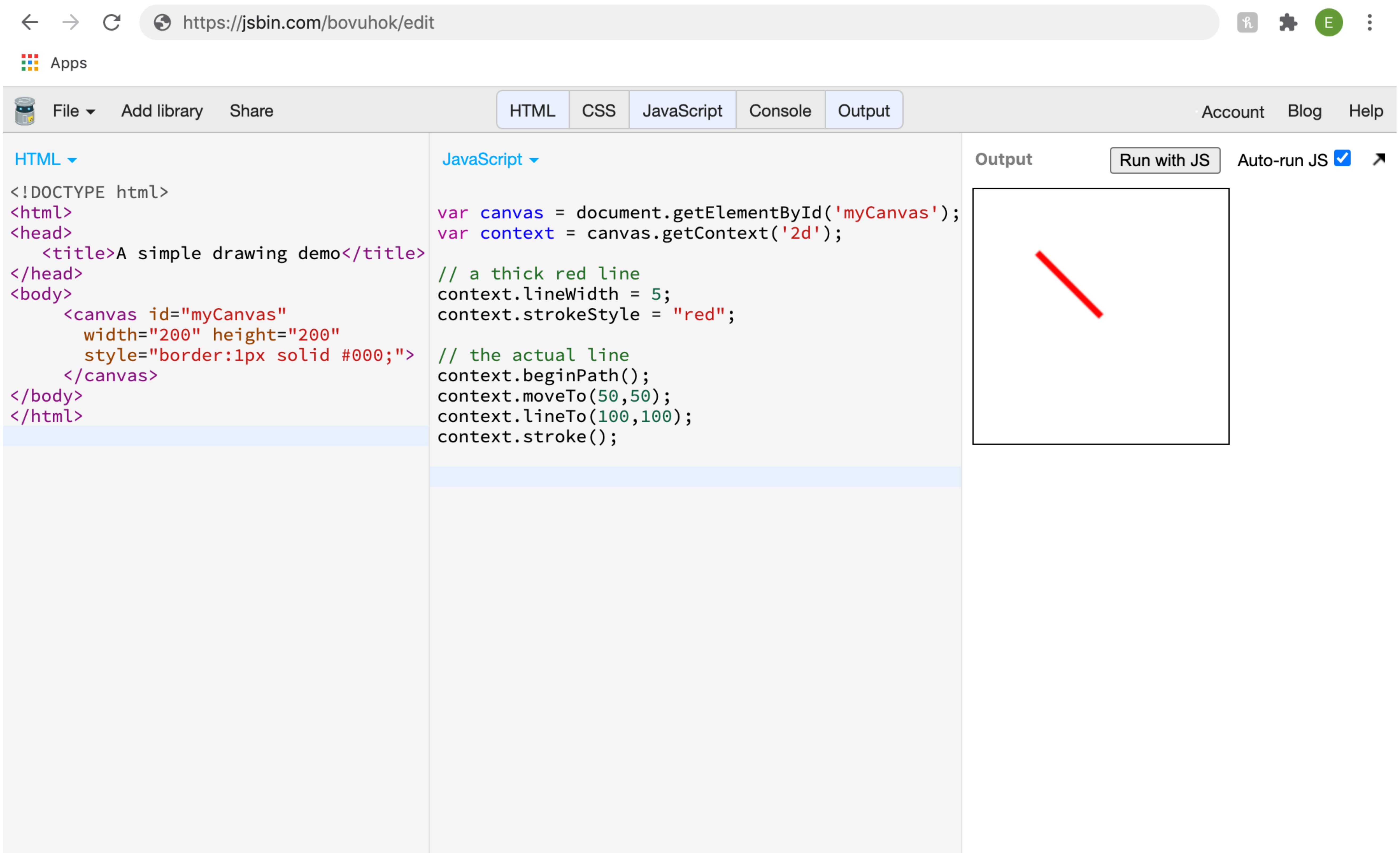
[Logistics] Other times, a link in blue will point you to a GitHub repository directory for a demo

```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();
```

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



HTML

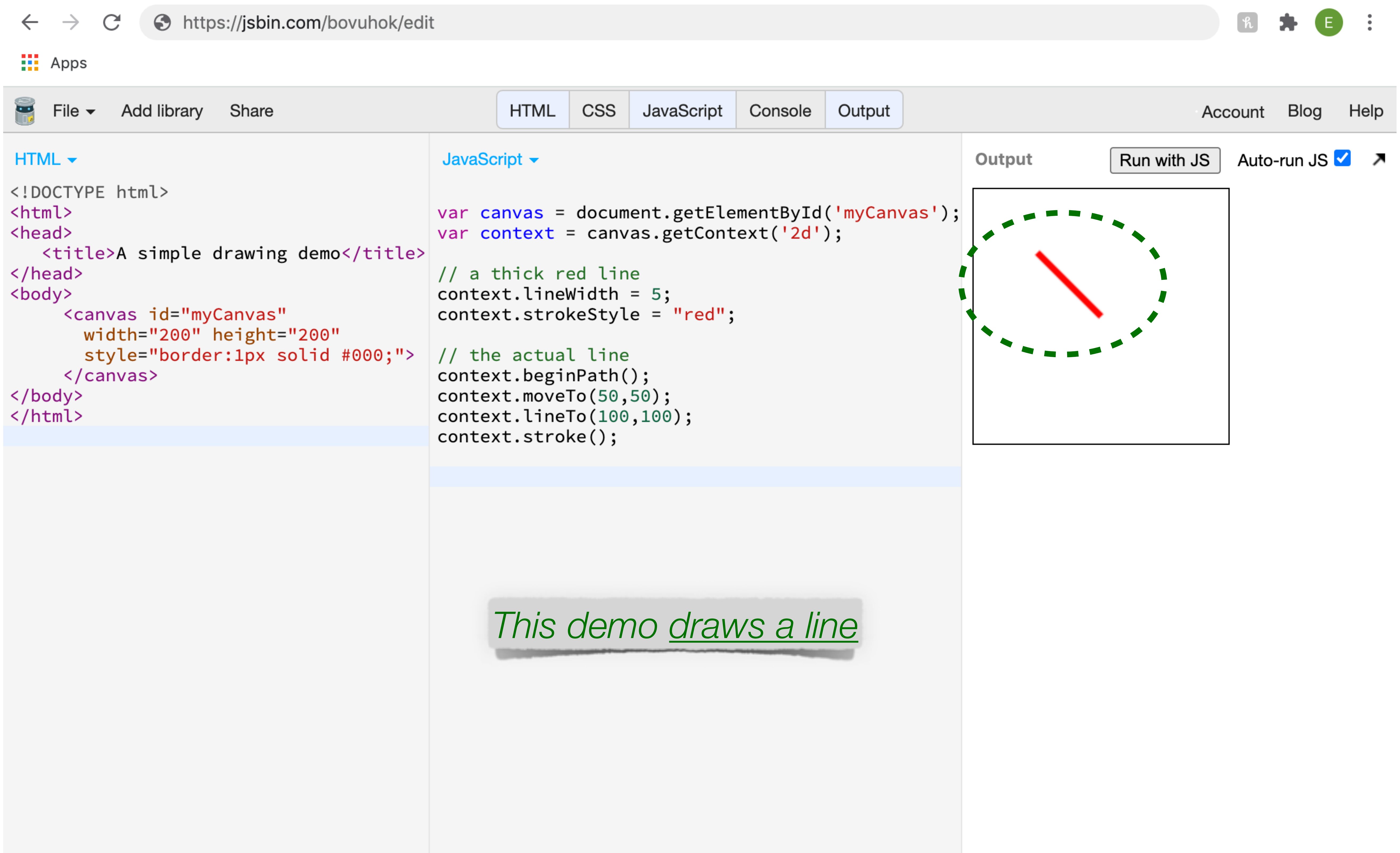
```
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

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();
```

A simple drawing example (via JSbin)

jsbin.com/bovuhok



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>
```

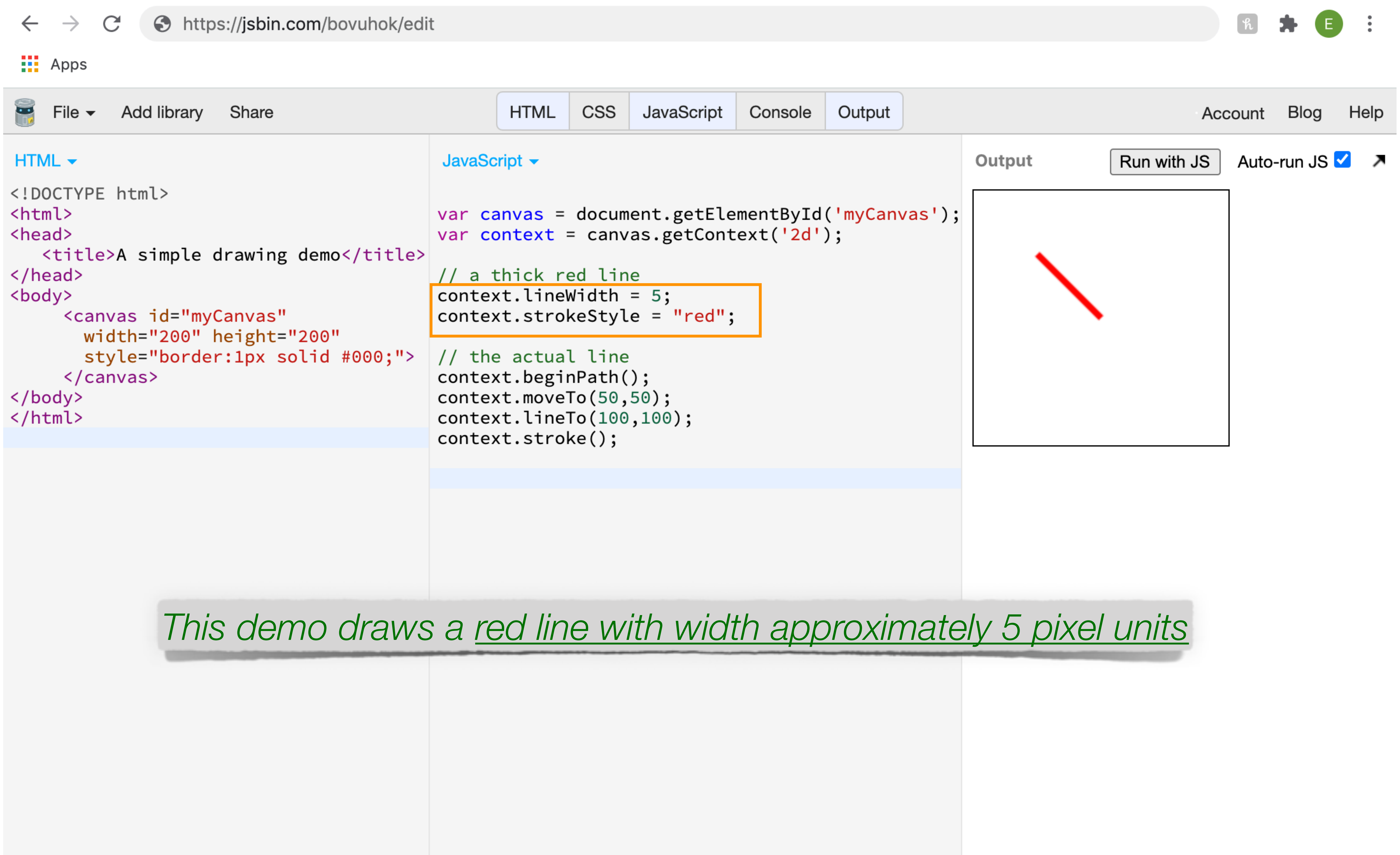
JavaScript

```
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();
```

This demo draws a line

A simple drawing example (via JSbin)

jsbin.com/bovuhok



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>
```

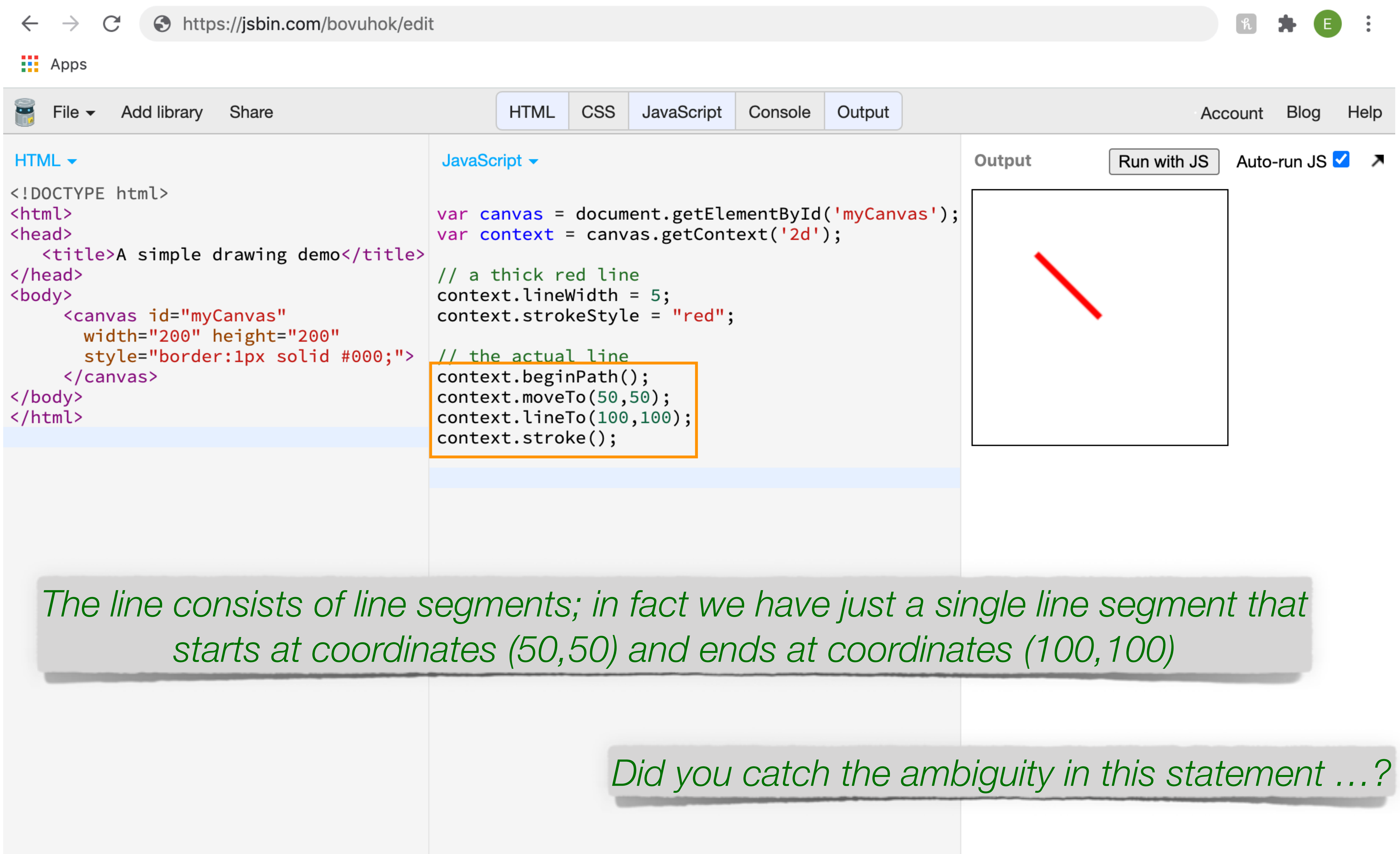
JavaScript

```
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();
```

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

A simple drawing example (via JSbin)

jsbin.com/bovuhok



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();
```

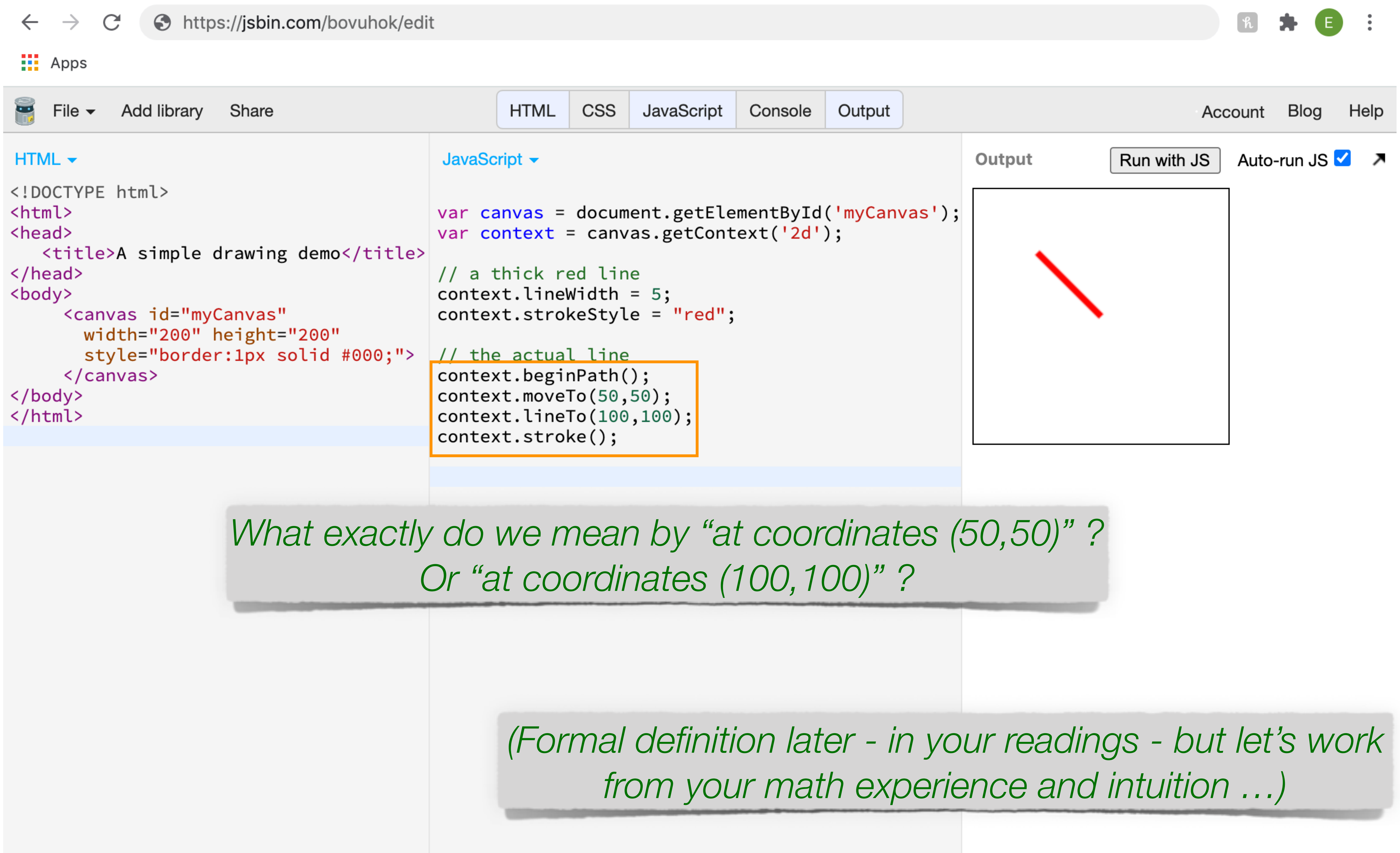
Output

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



```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>
```

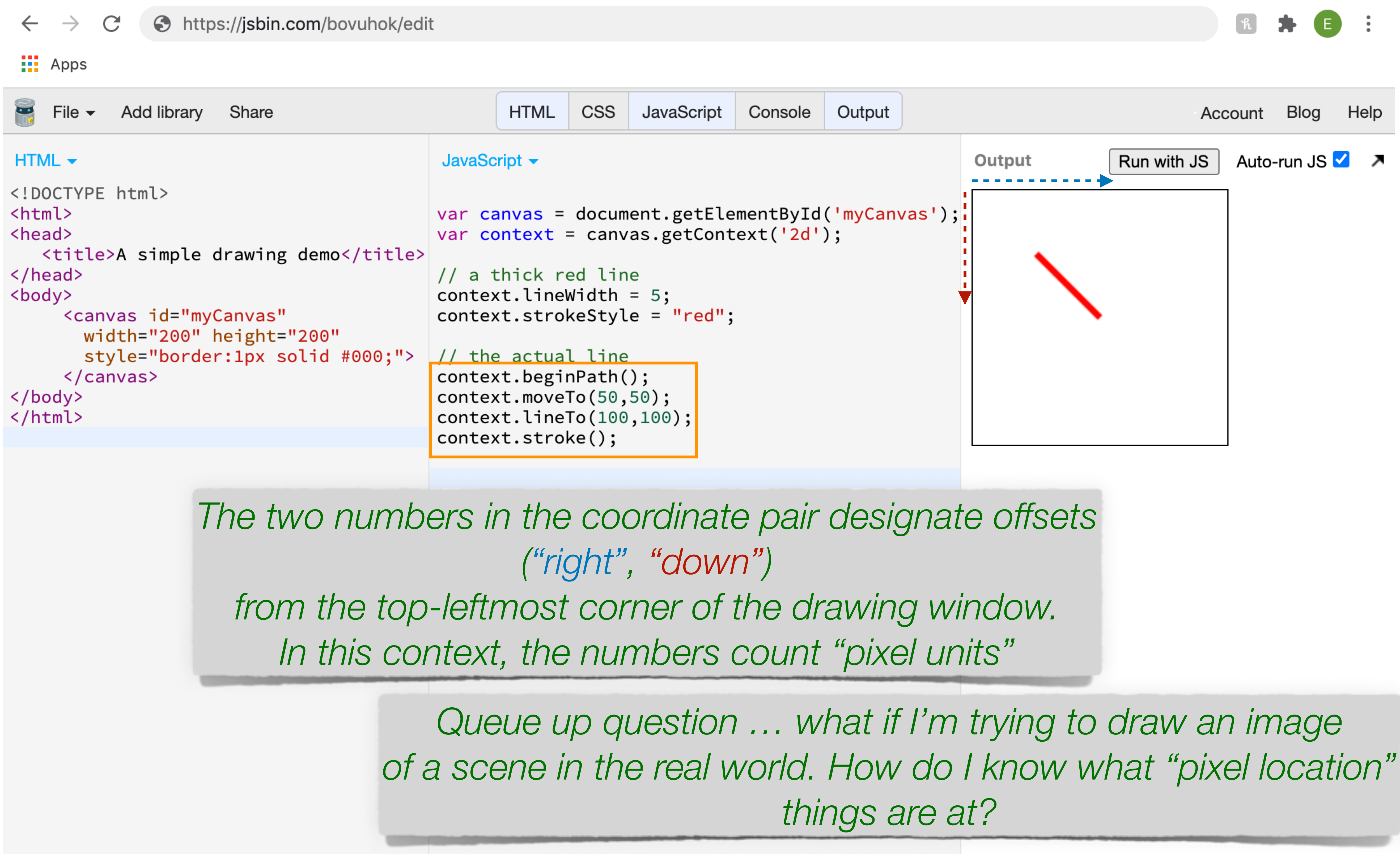
```javascript
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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();
```

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



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

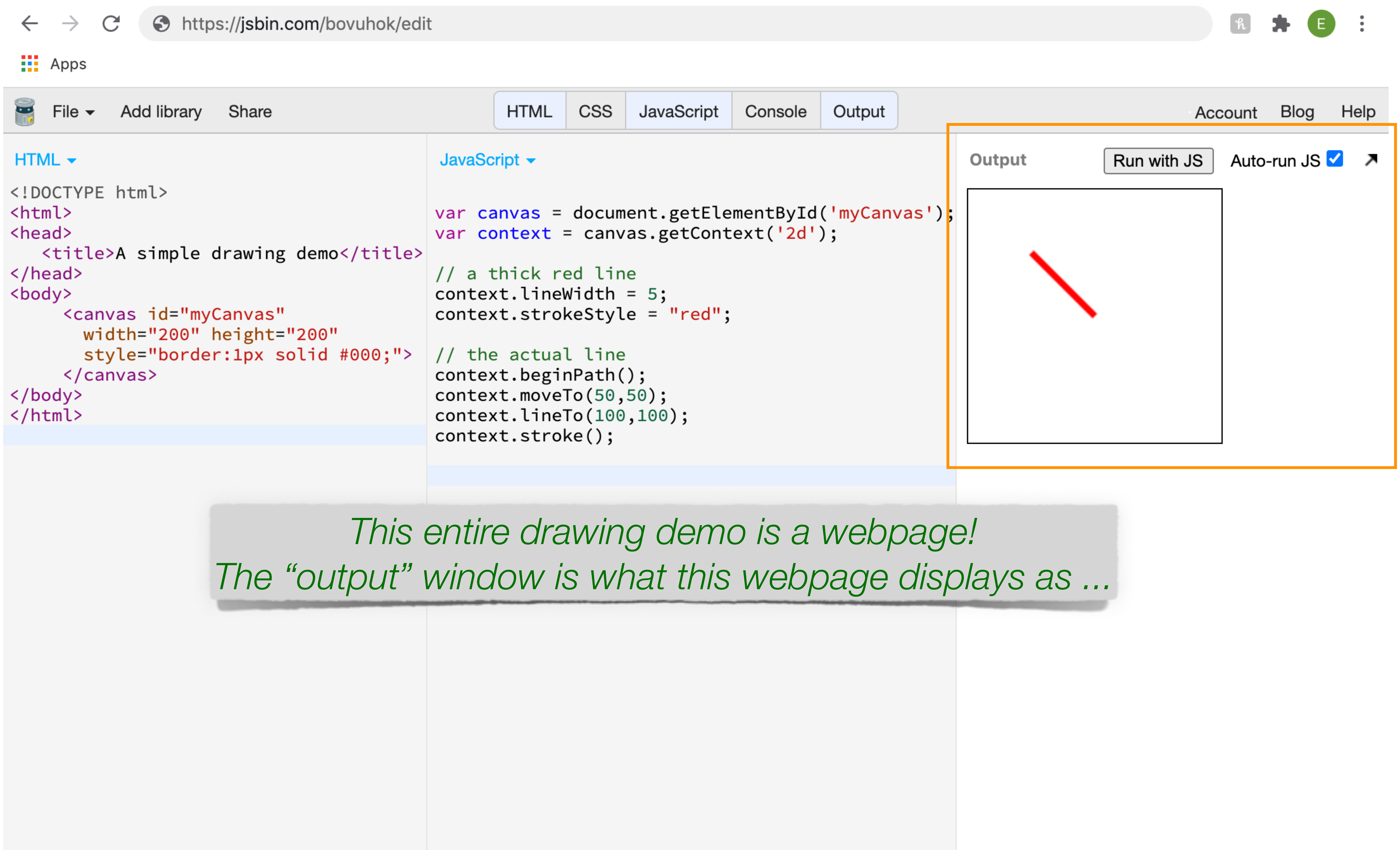
```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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();
```

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



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>
```

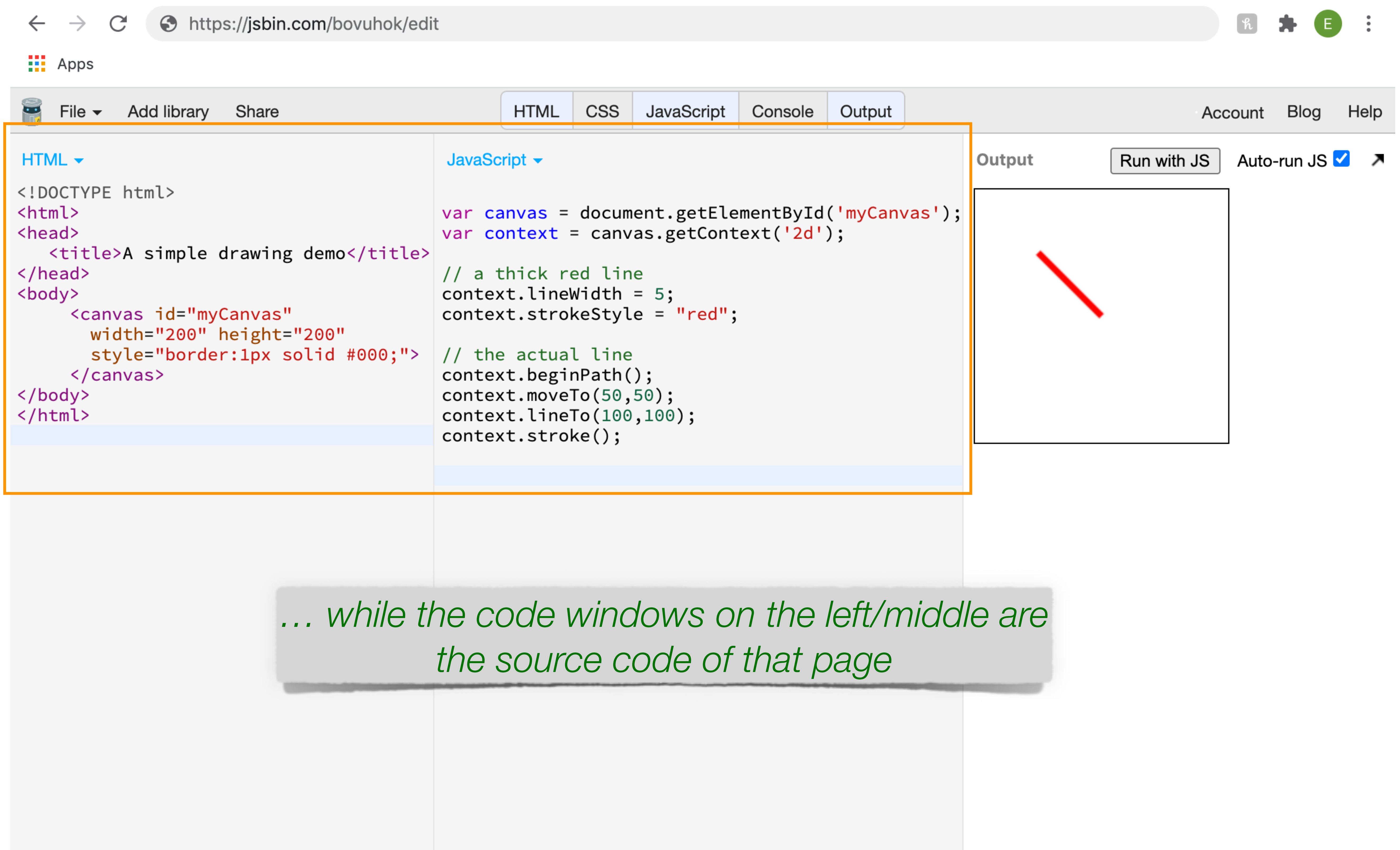
JavaScript

```
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();
```

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



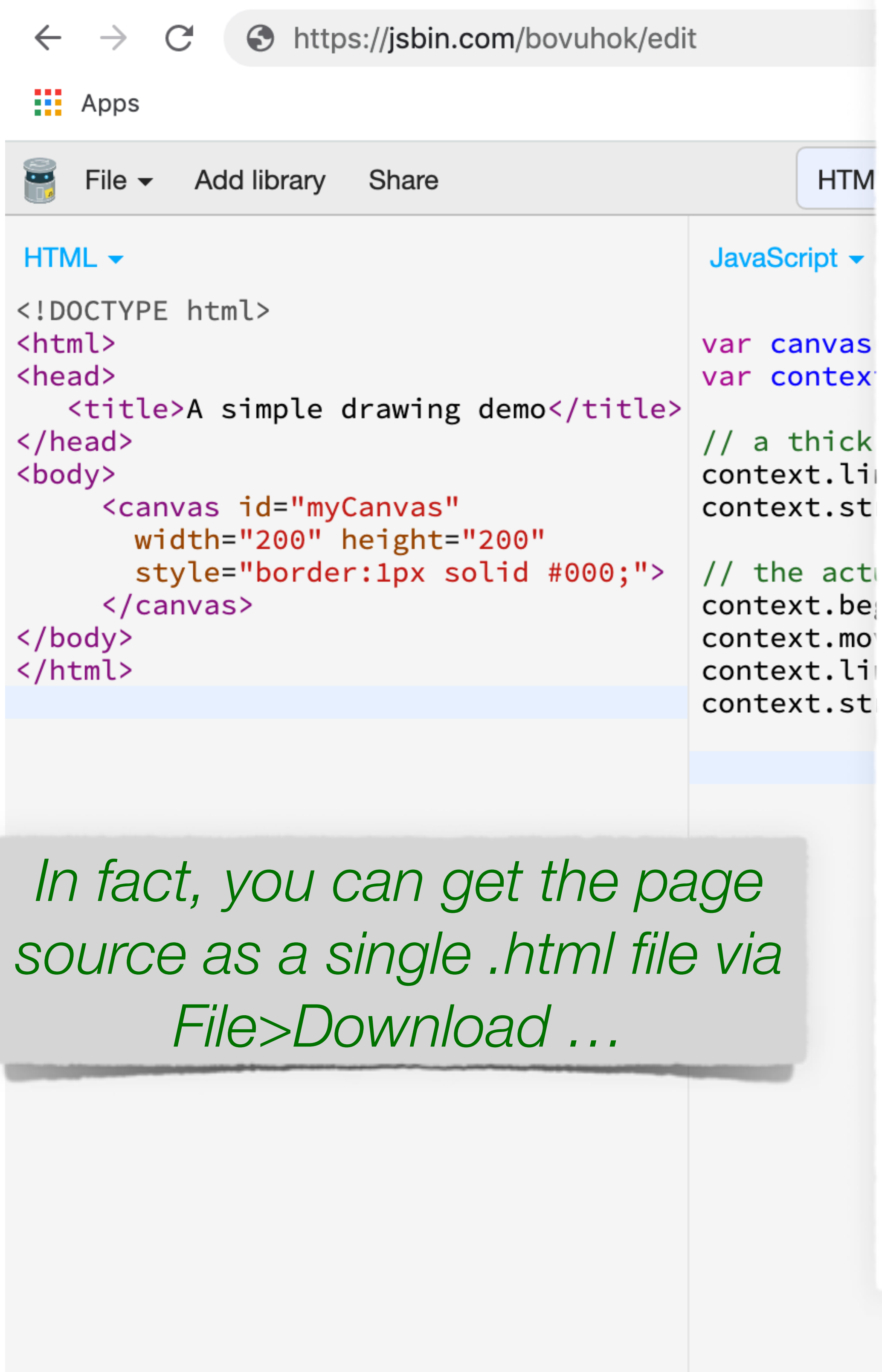
```html
<!DOCTYPE html>
<html>
<head>
   <title>A simple drawing demo</title>
</head>
<body>
     <canvas id="myCanvas"
       width="200" height="200"
       style="border:1px solid #000;">
     </canvas>
</body>
</html>
```

```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();
```

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

A simple drawing example (via JSbin)

jsbin.com/bovuhok



```
<!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>
```

```
var canvas
var contex

// a thick
context.li
context.st

// the actu
context.be
context.mo
context.li
context.st
```

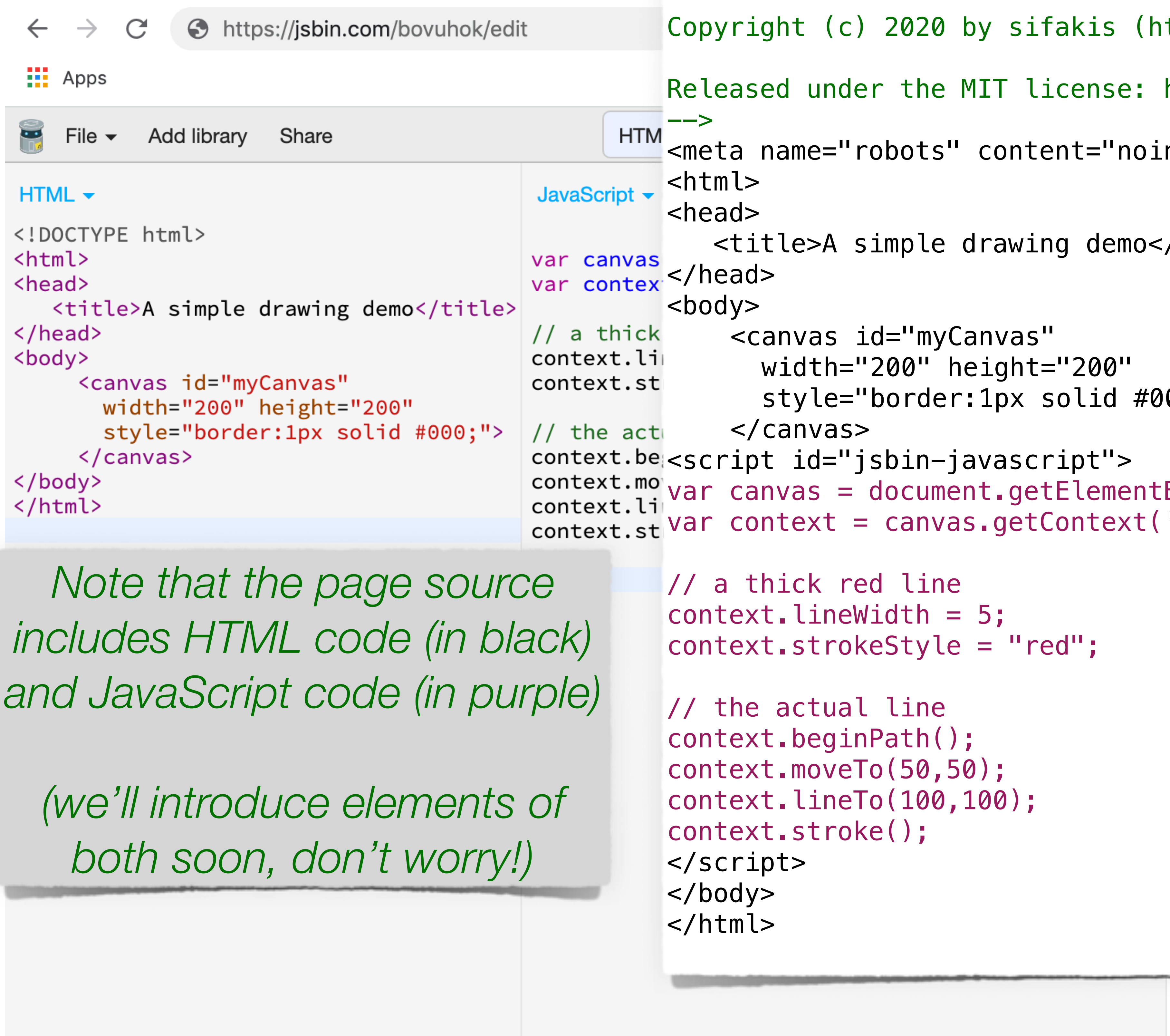
In fact, you can get the page source as a single .html file via File>Download …

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 exam

jsbin.com/bovuhok



```
<!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>
```

```
var canvas
var contex

// a thick
context.li
context.st

// the act
context.be
context.mo
context.li
context.st
```

Page source as an html file:

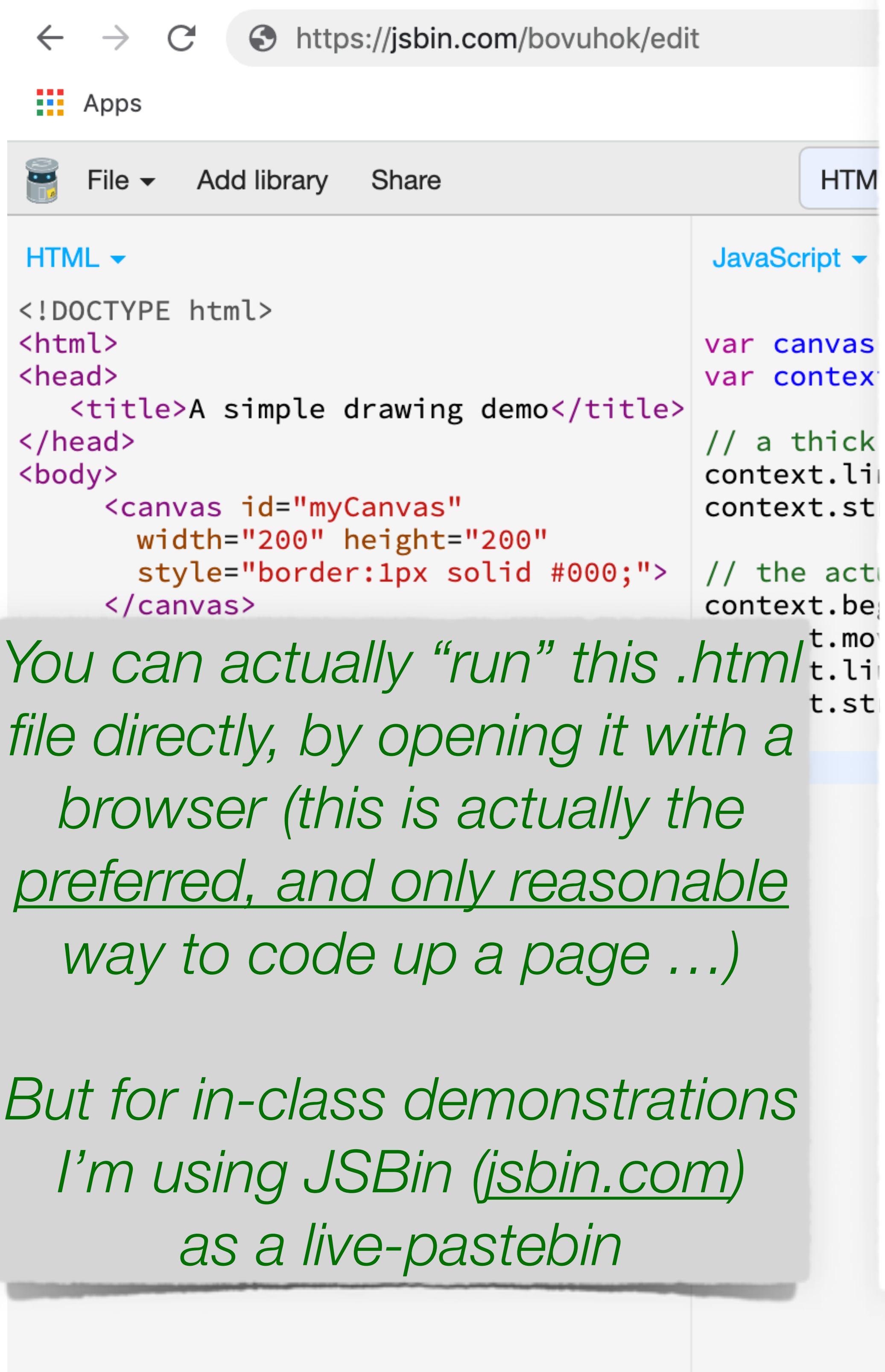
```
<!DOCTYPE html>
<!--
Created using JS Bin
http://jsbin.com

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>
```

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!)

A simple drawing exam

jsbin.com/bovuhok



```
<!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>
```

```
var canvas
var contex

// a thick
context.li
context.st

// the act
context.be
t.mo
t.li
t.st
```

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

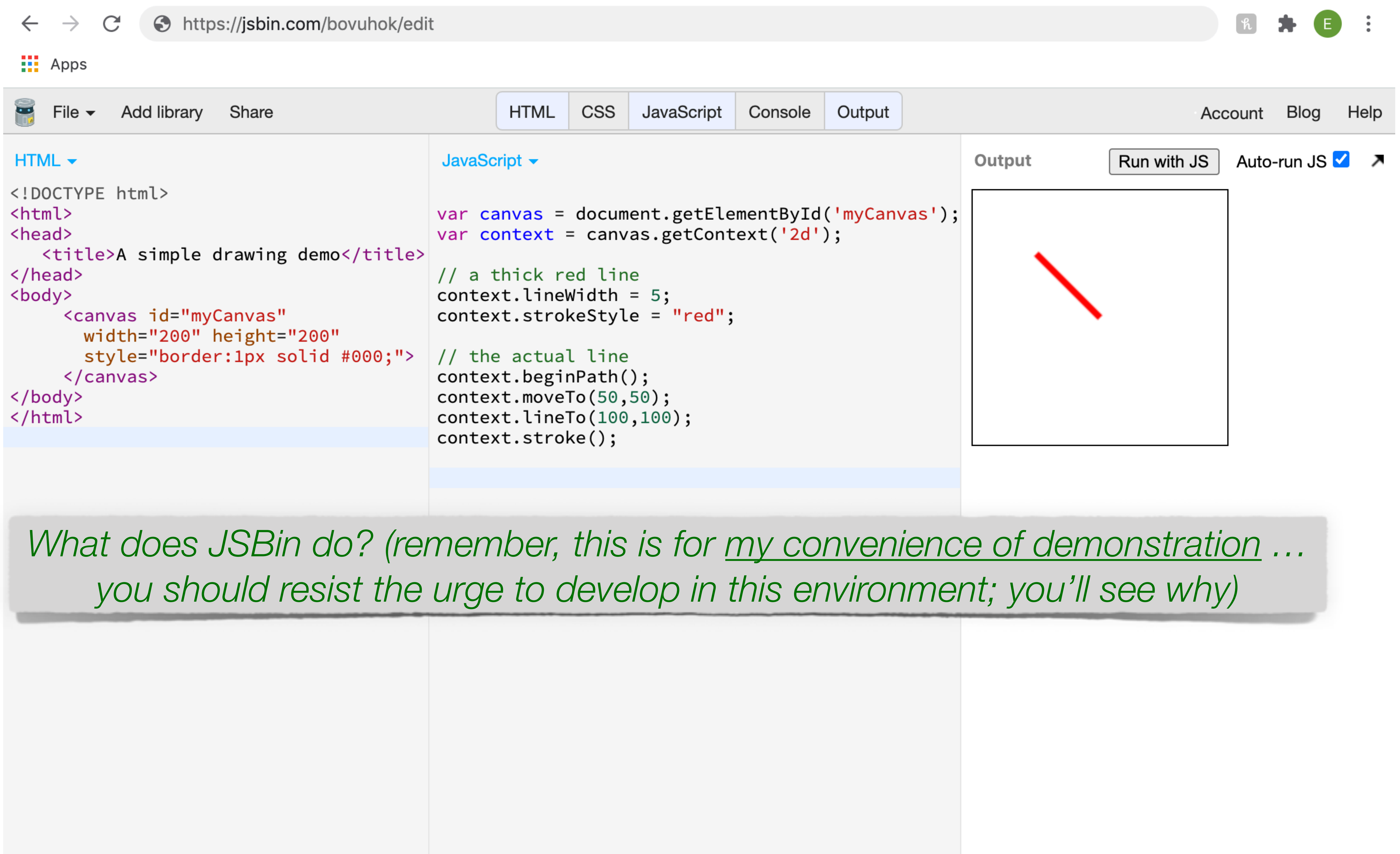
But for in-class demonstrations I'm using JSBin (<u>jsbin.com</u>) as a live-pastebin

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



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>
```

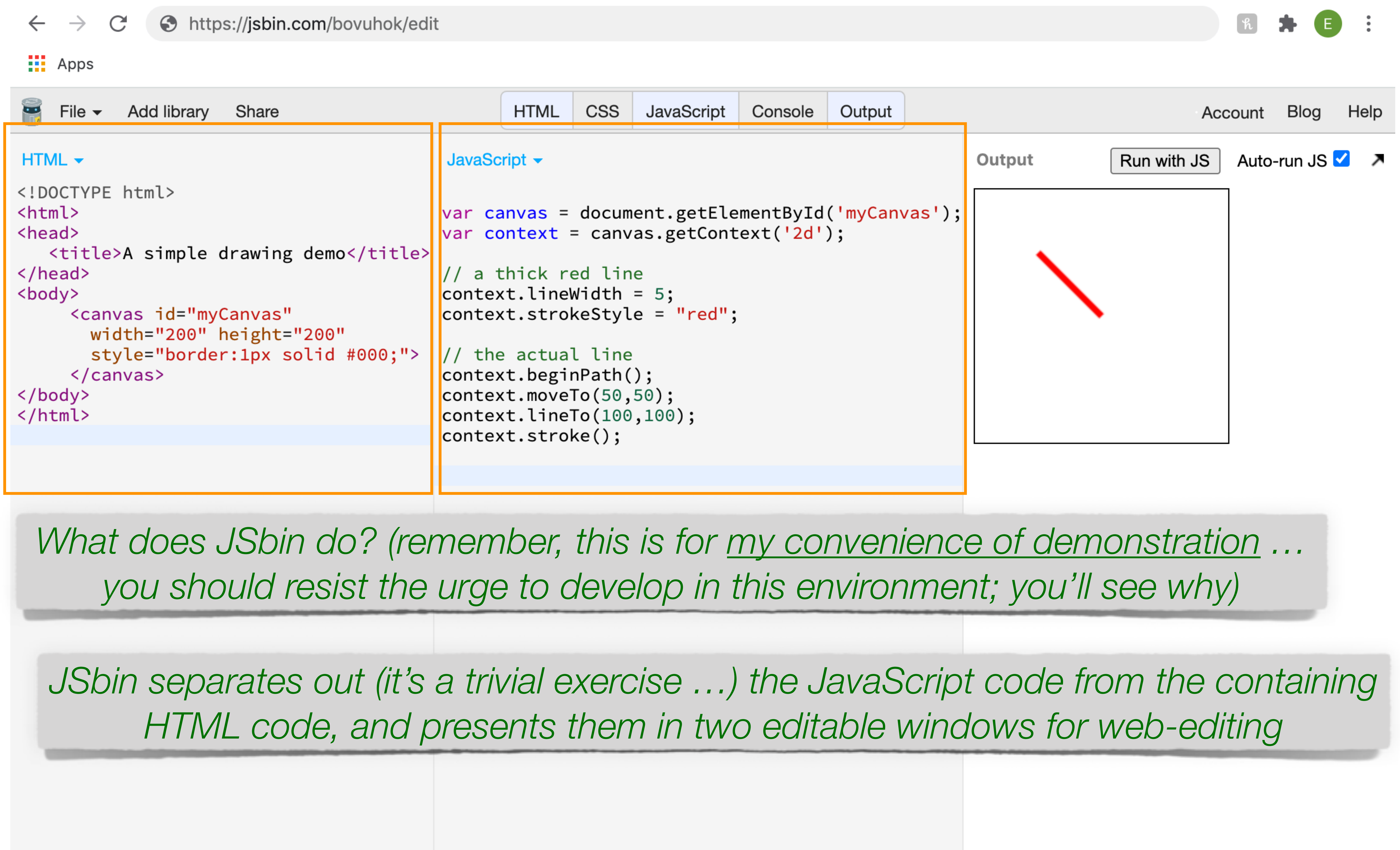
JavaScript

```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();
```

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



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

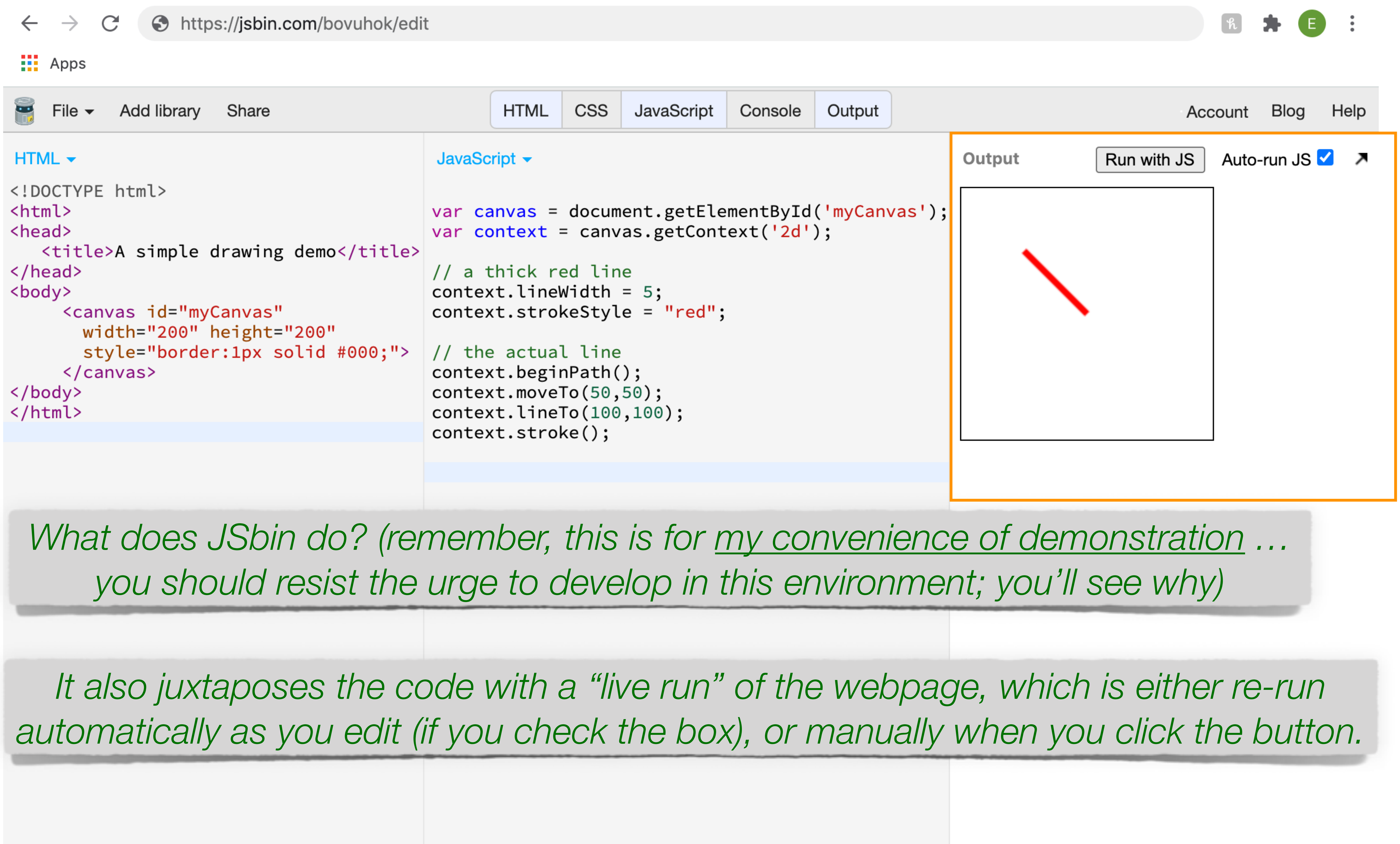
```javascript

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();
```

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



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

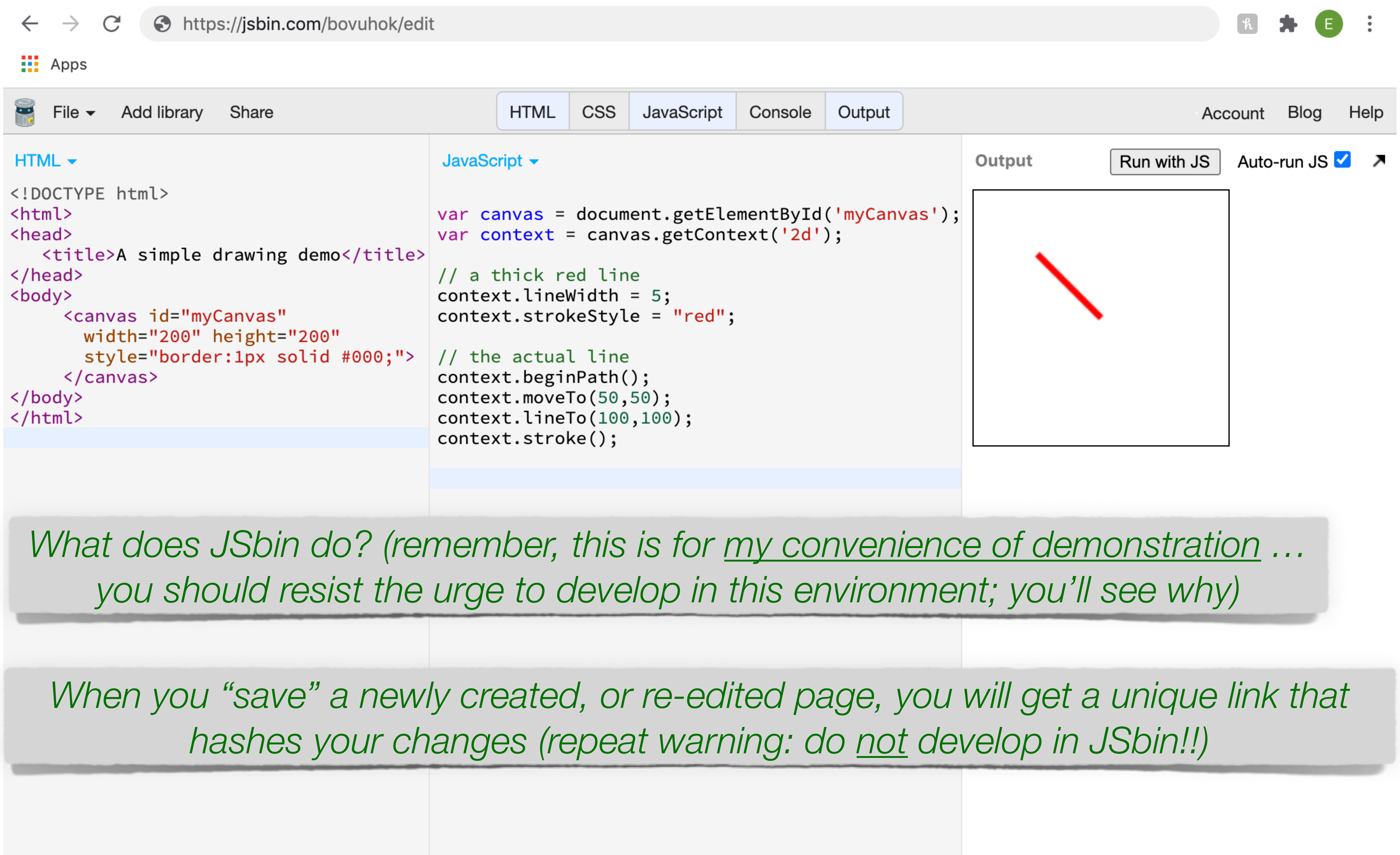
```javascript
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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();
```

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



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

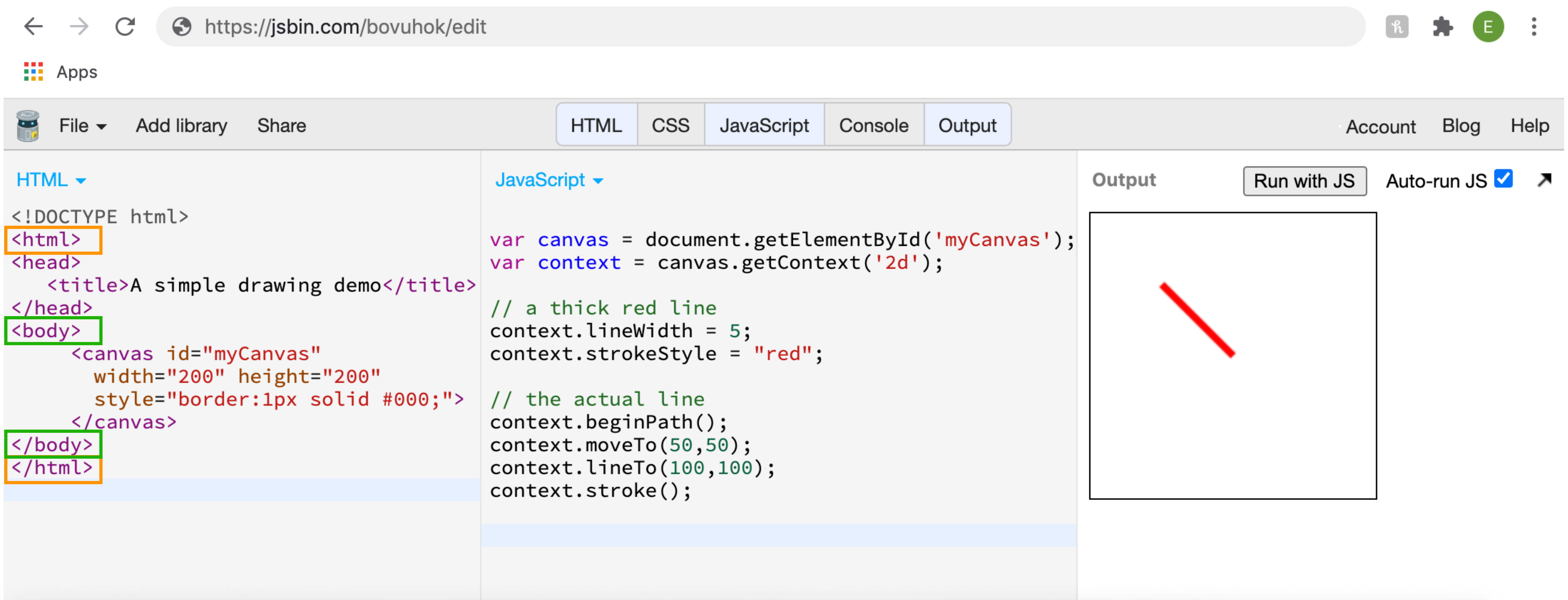
```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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();
```

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



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();
```

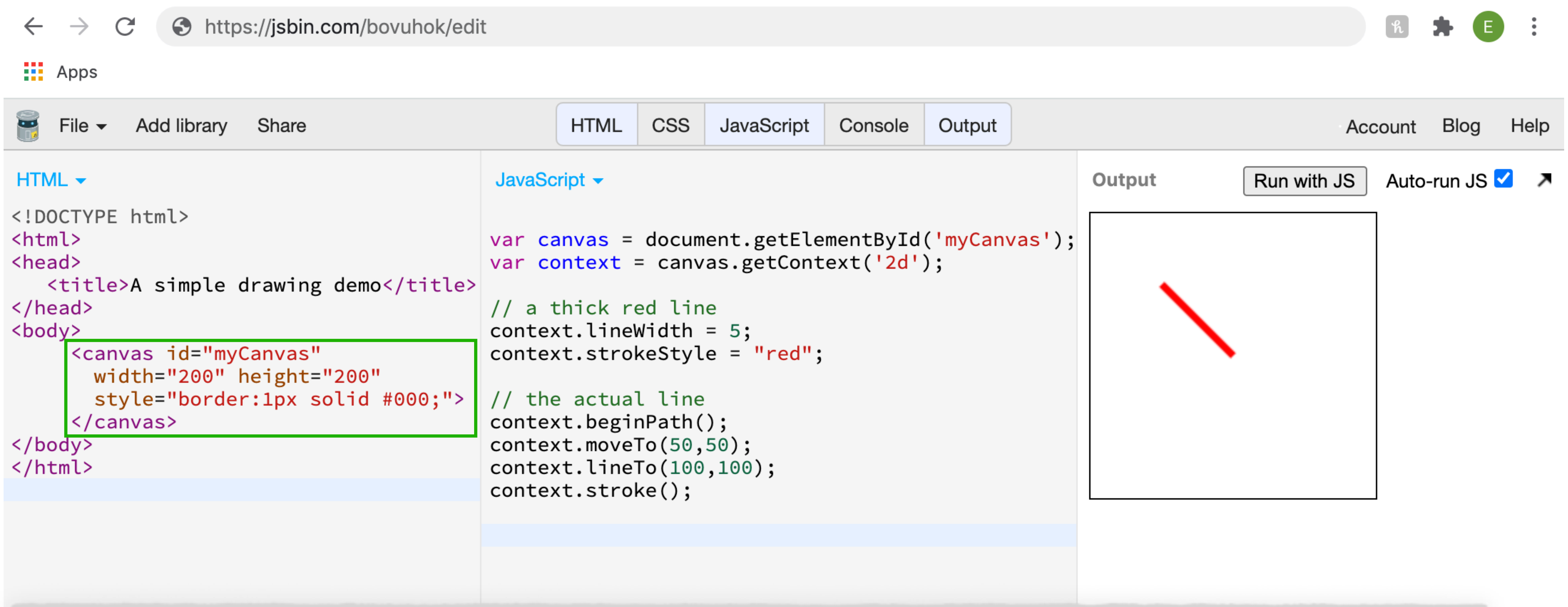
A walk-through of the HTML component of this webpage (again, by example …)

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



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();
```

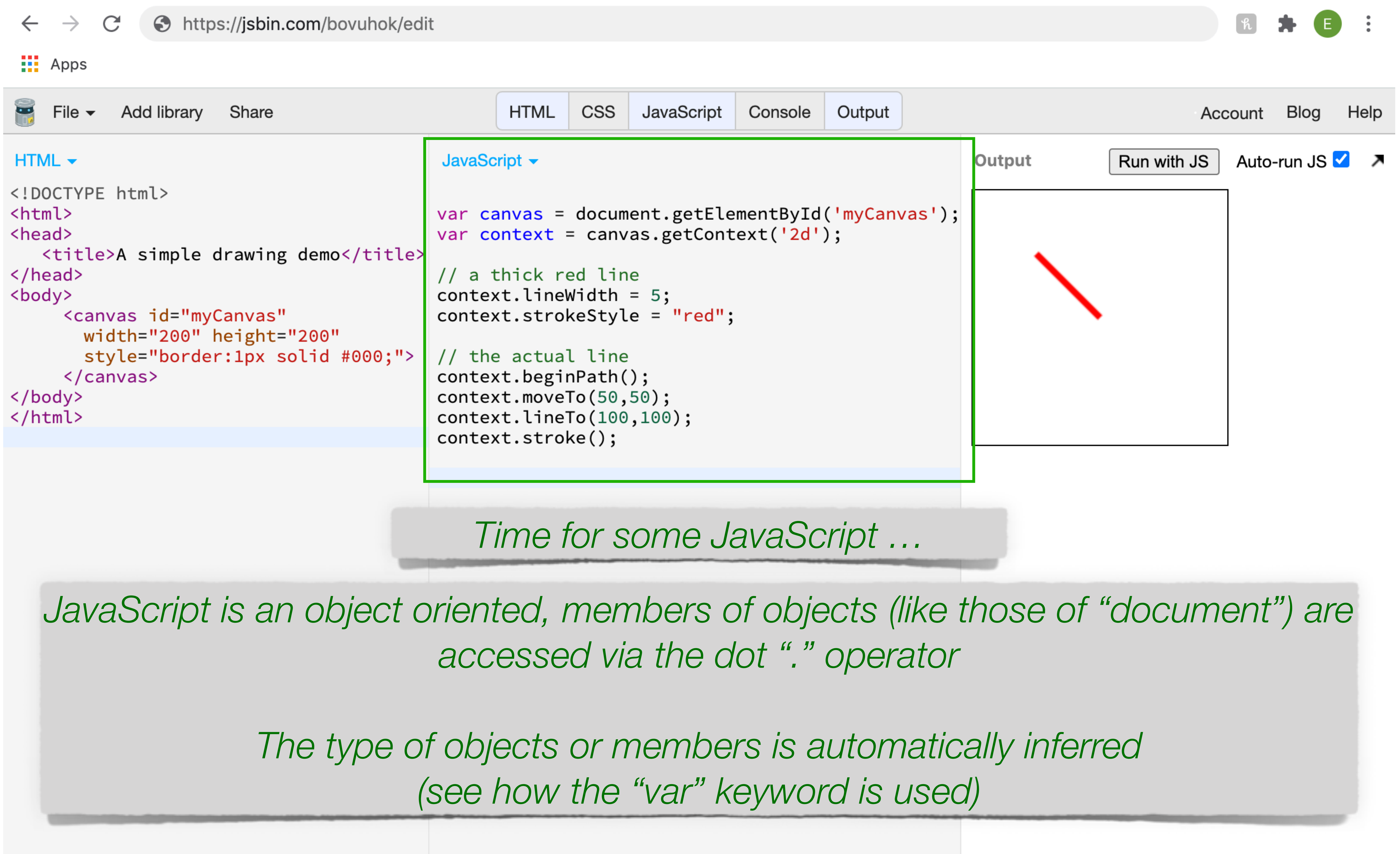
A walk-through of the HTML component of this webpage (again, by example …)

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



```
<!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>
```

```
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();
```

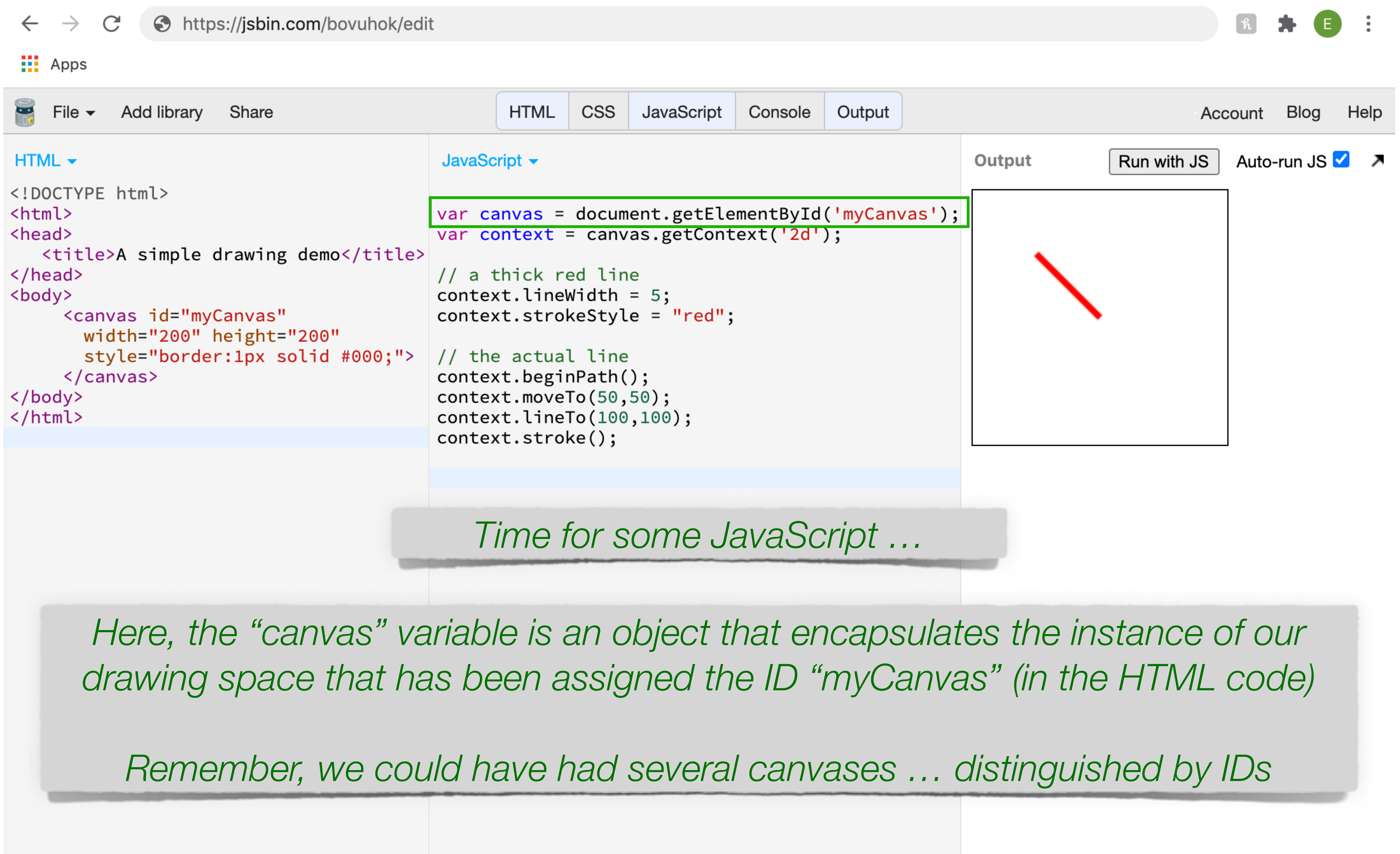
Time for some JavaScript …

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



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

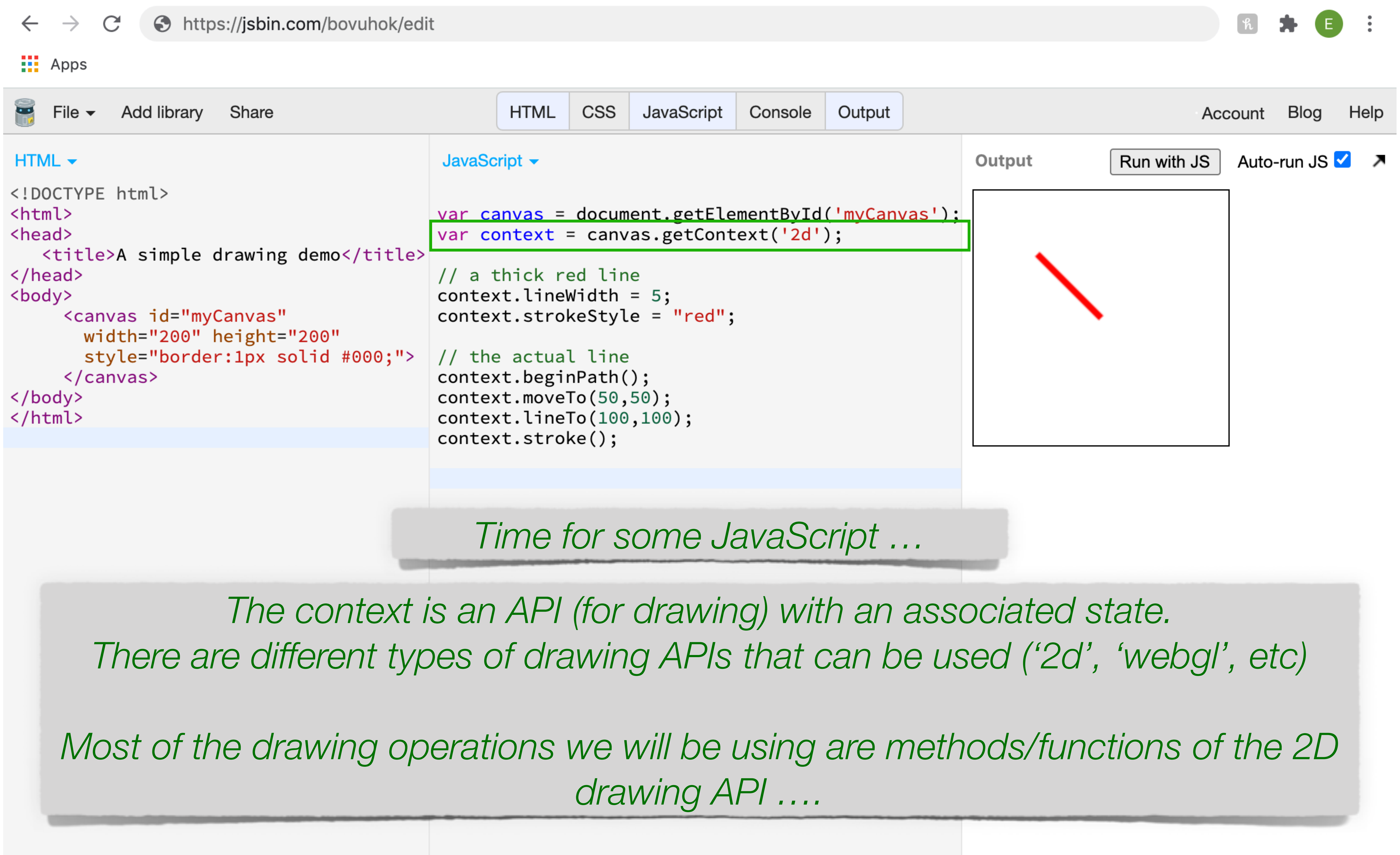
Time for some JavaScript …

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



```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();
```

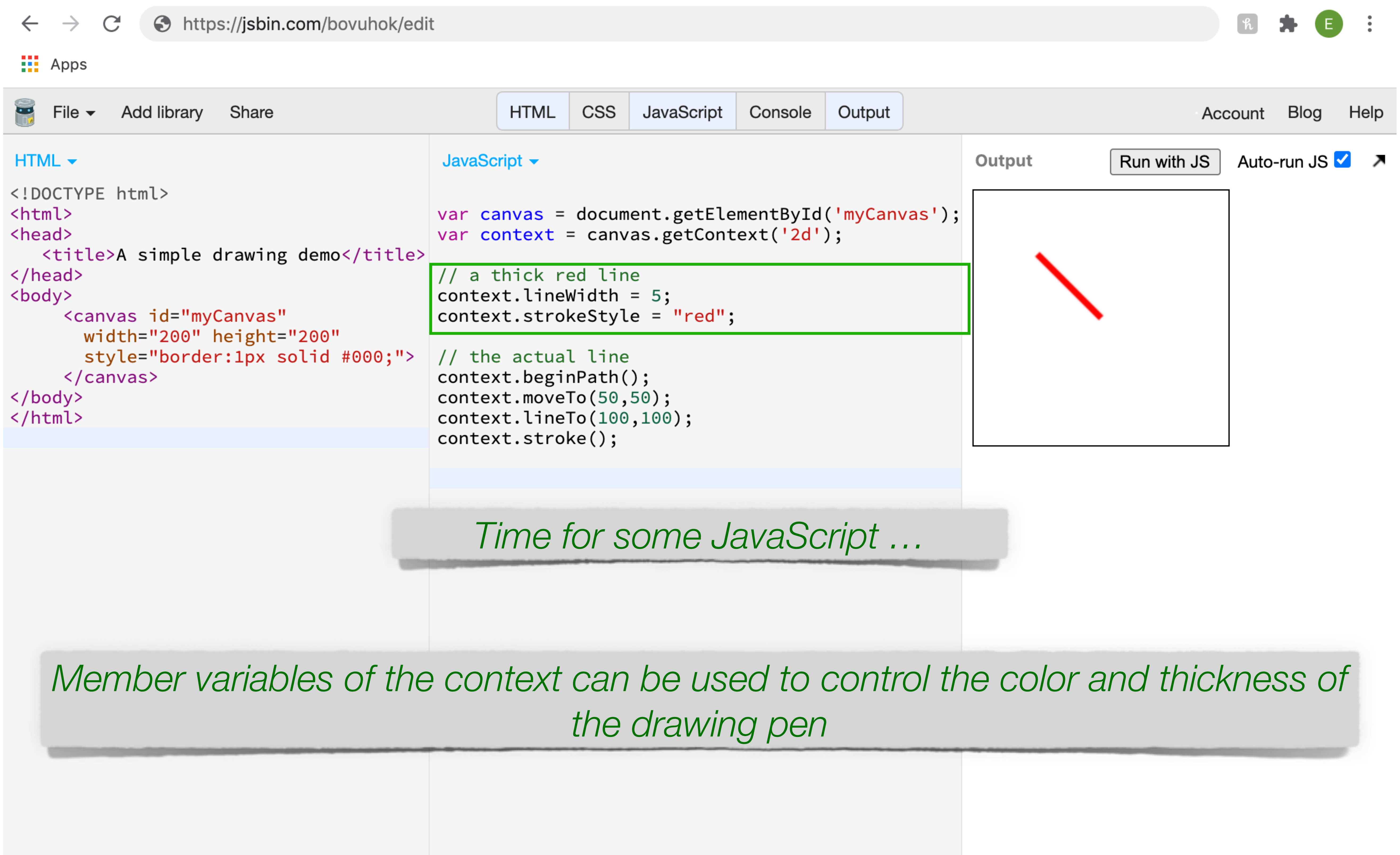
Time for some JavaScript …

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



```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>
```

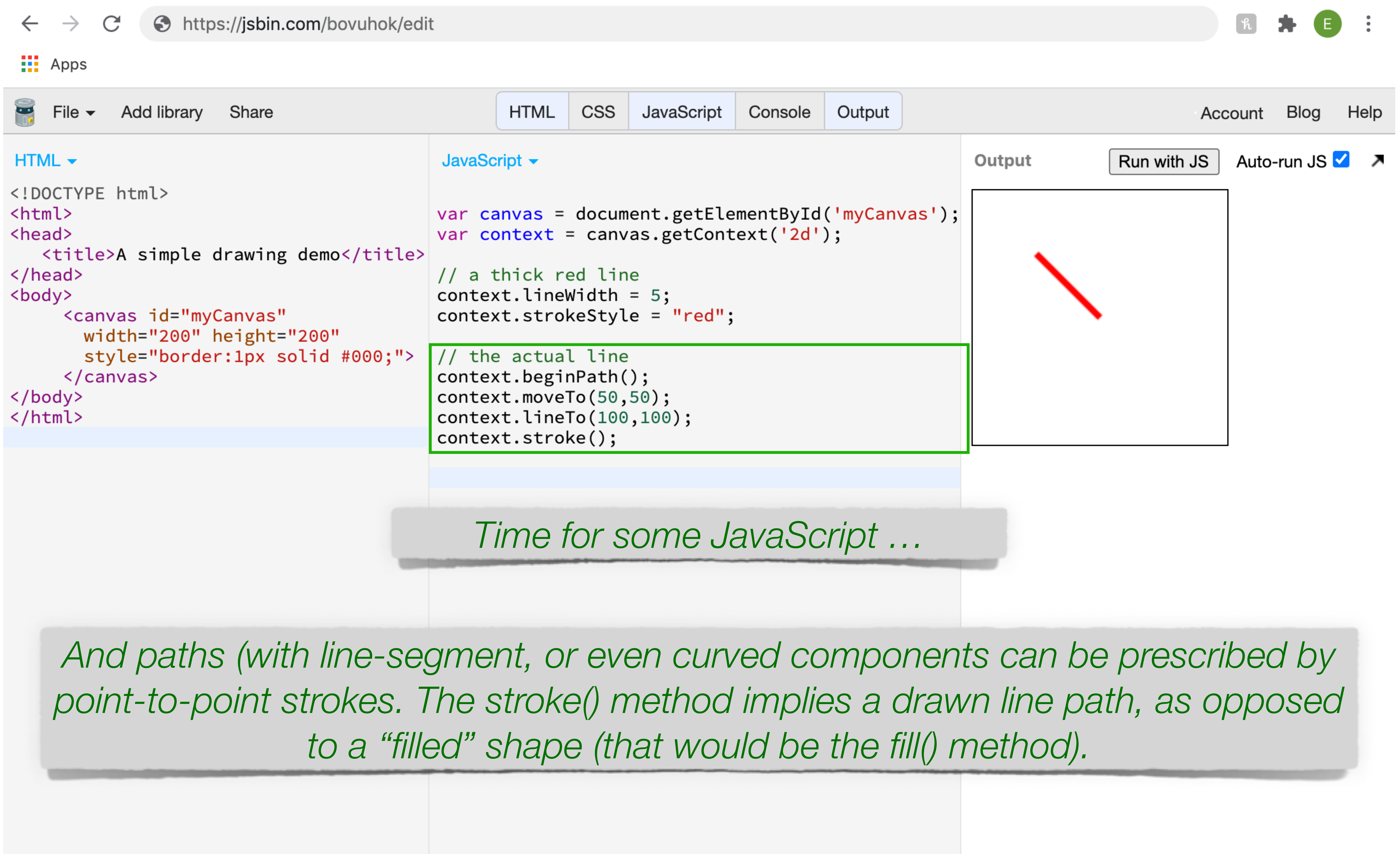
```javascript
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

// 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();
```

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



```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();
```

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)

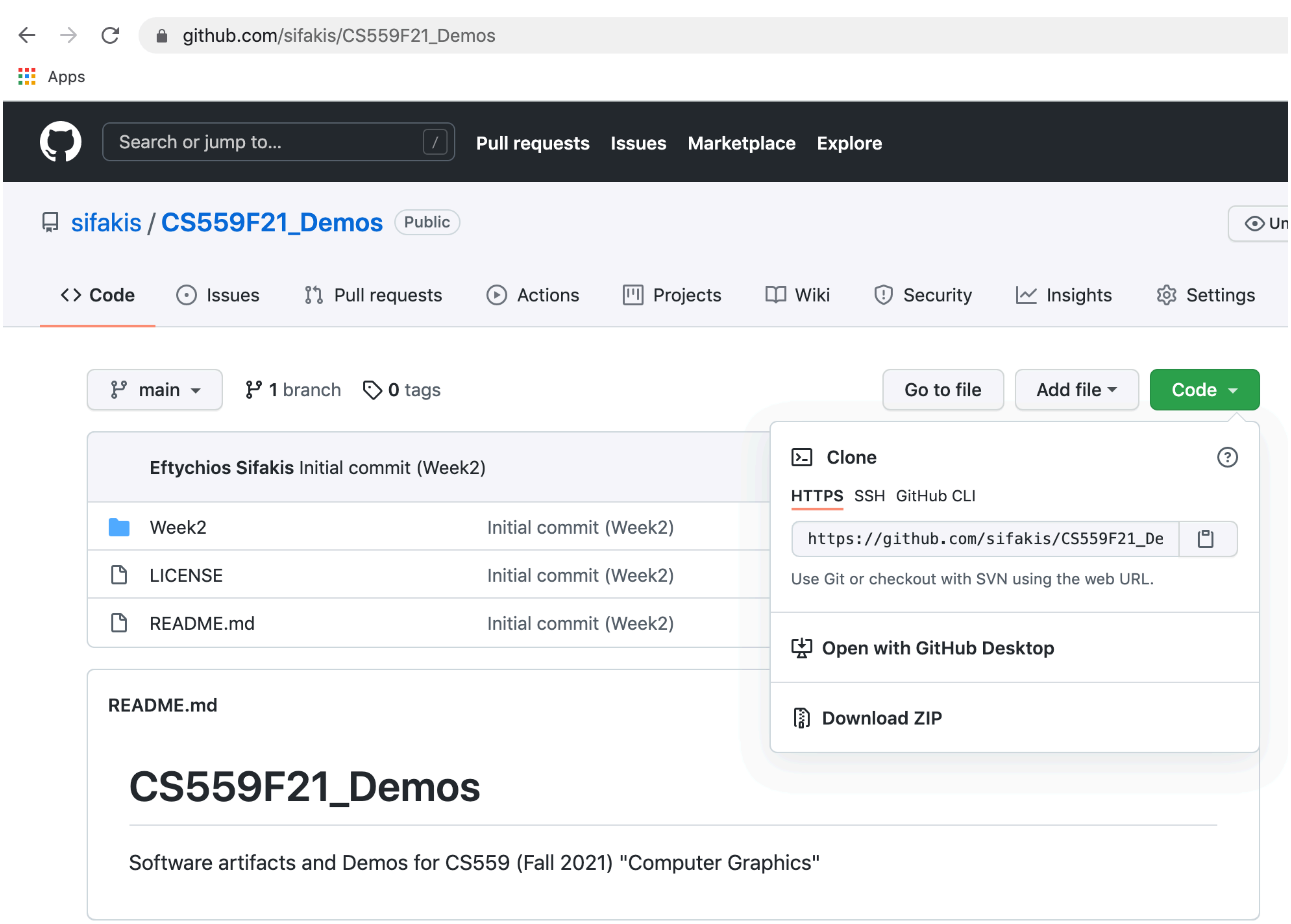
jsbin.com/suhujar

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)



Eftychios Sifakis Initial commit (Week2)

Week2	Initial commit (Week2)
LICENSE	Initial commit (Week2)
README.md	Initial commit (Week2)

```
https://github.com/sifakis/CS559F21_De
```

Use Git or checkout with SVN using the web URL.

README.md

CS559F21_Demos

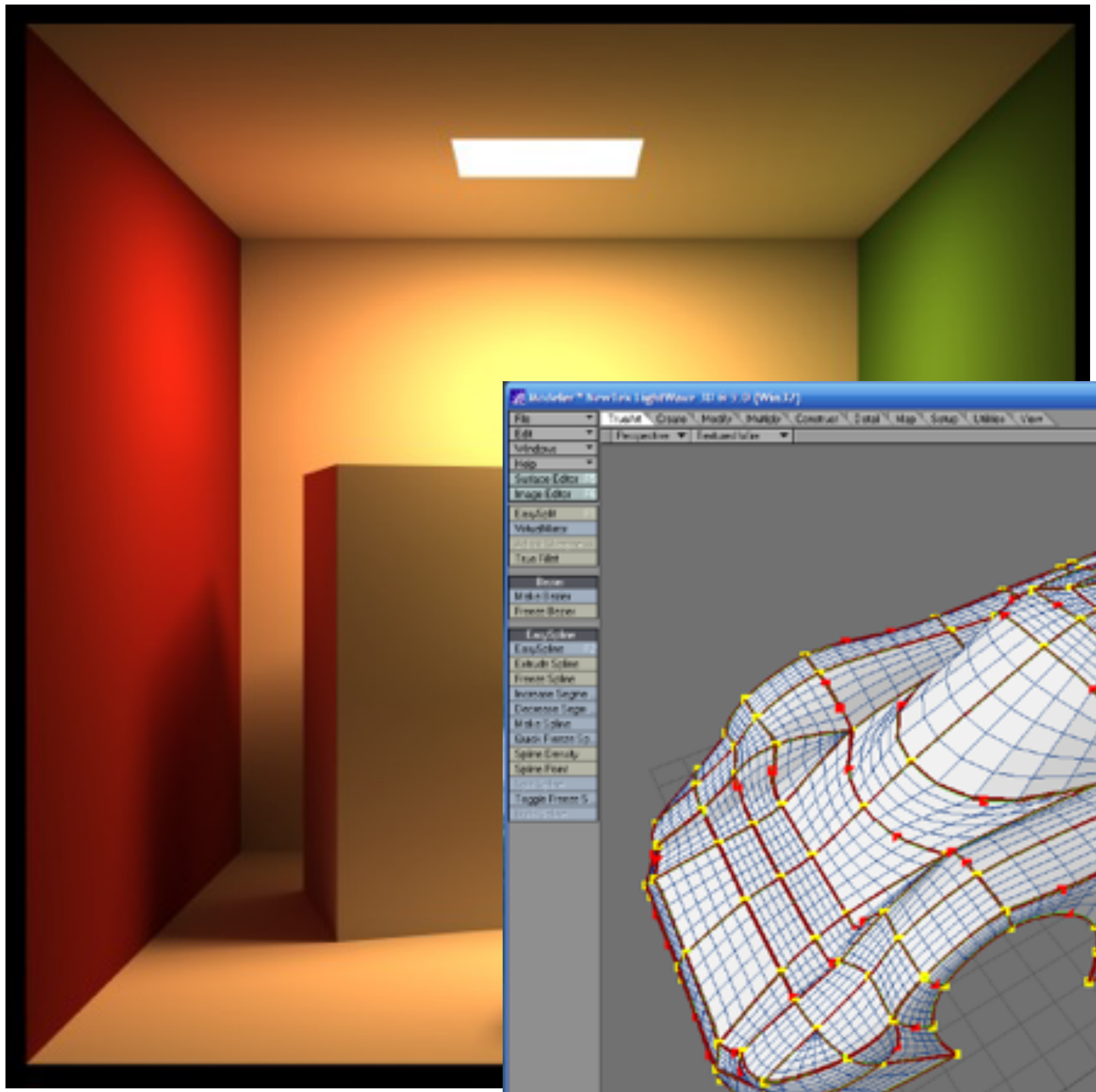
Software artifacts and Demos for CS559 (Fall 2021) "Computer Graphics"

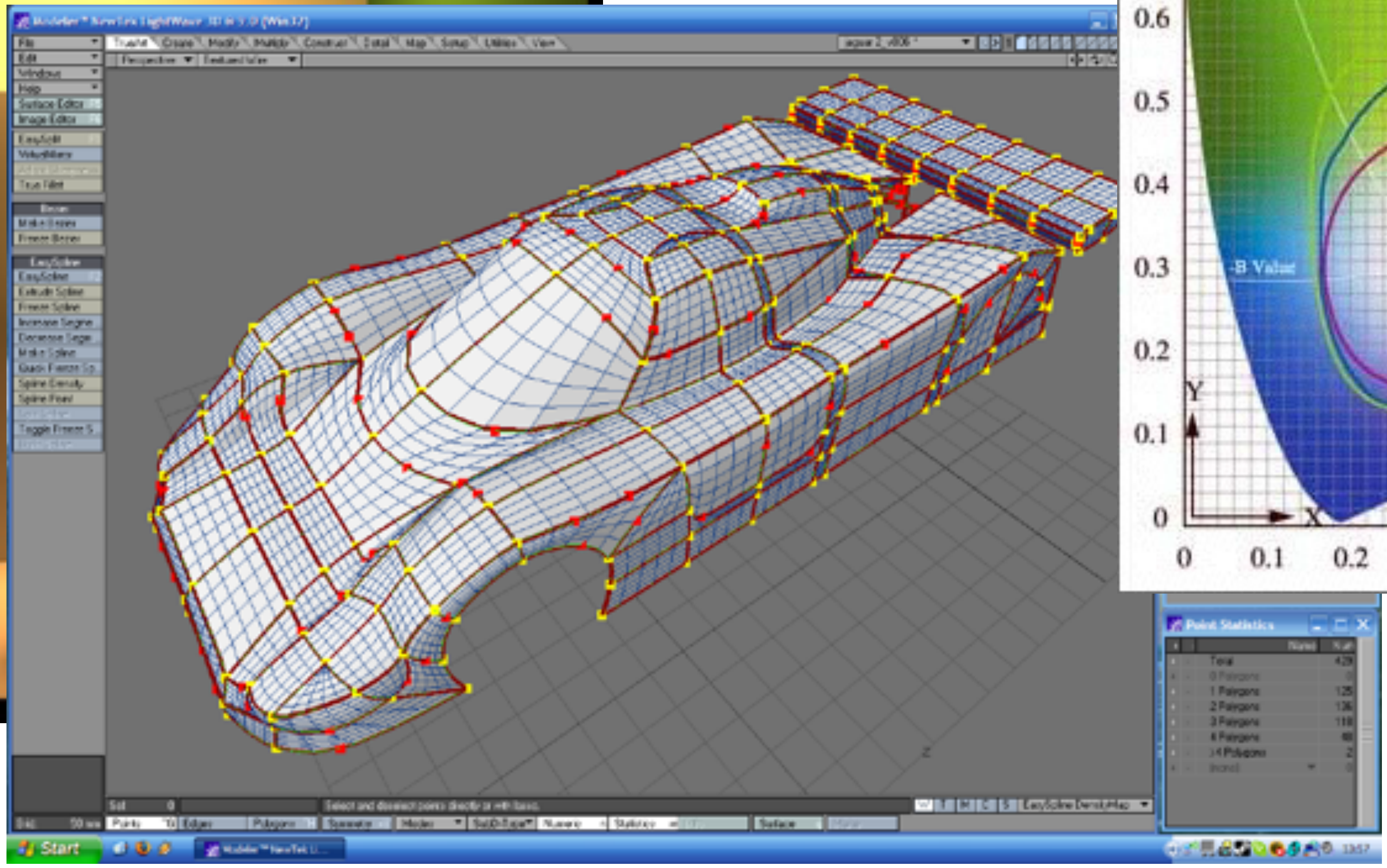
Additional examples (flash preview)

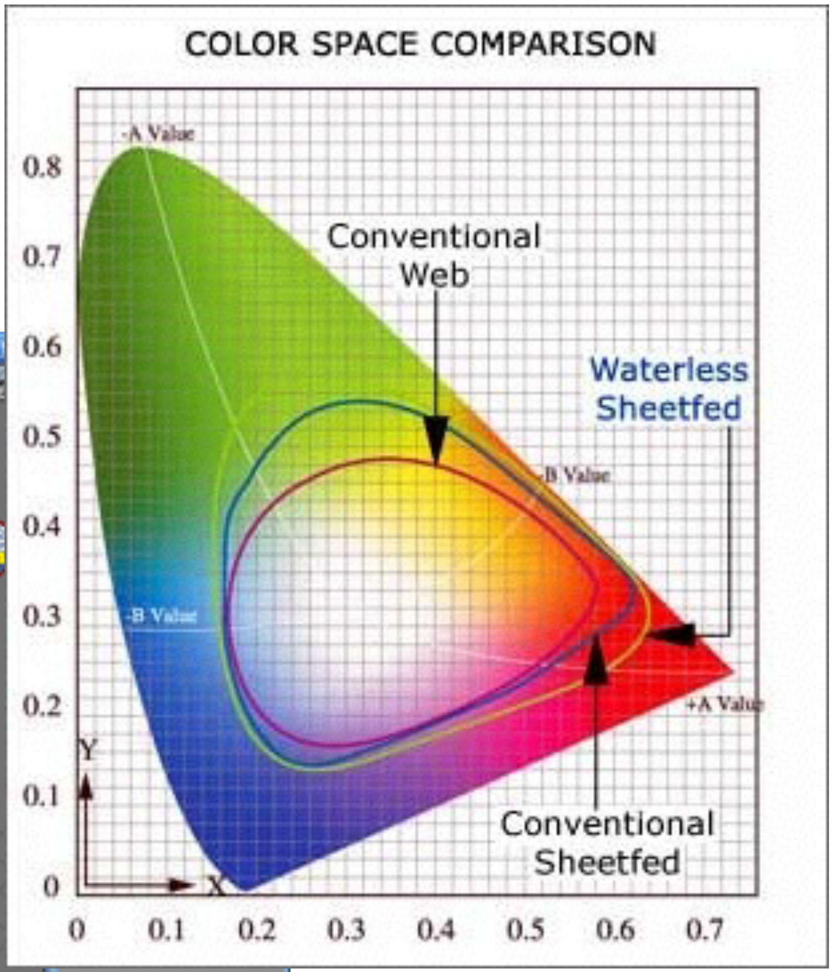
jsbin.com/fesukexori

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 14th 2021