

Introduction to Web Technology

HTML5: Graphic Canvas, Drag and Drop

Joseph Tonien
School of Computing and Information Technology
University of Wollongong

HTML 5

Canvas

- First introduced in WebKit by Apple for the OS X Dashboard, Graphic Canvas has since been implemented in other major browsers.
- Canvas is used to draw graphics, such as paths, boxes, circles, text, and images, on the fly, via JavaScript.

HTML 5

Drag and Drop

- Drag and Drop enable applications to use drag and drop features in browsers.
- The user can select draggable elements with a mouse, drag the elements to a droppable element, and drop those elements by releasing the mouse button.

Canvas

The `<canvas>` element is used to draw graphics on a web page.

```
<canvas id="mycanvas" width="1000" height="500"  
style="border:1px solid black;">
```

Your browser does not support canvas.

```
</canvas>
```

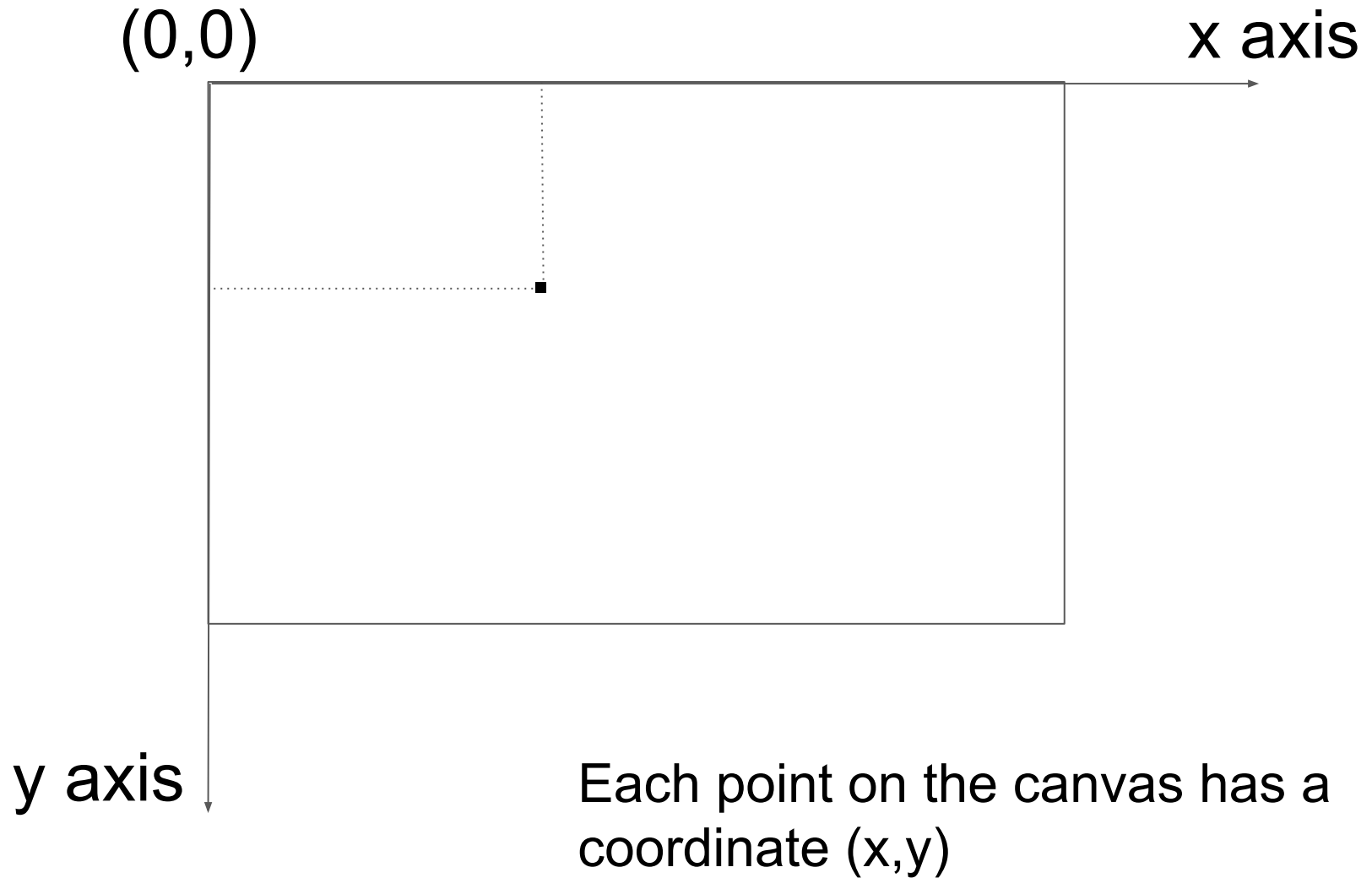
Canvas

The `<canvas>` element is used to draw graphics on a web page.

```
<canvas id="mycanvas" width="1000" height="500"  
style="border:1px solid black;">  
Your browser does not support canvas.  
</canvas>
```

The `<canvas>` element is only a container for the graphics. We must use JavaScript to actually draw the graphics content.

Canvas



Canvas

CanvasRenderingContext2D is used for drawing text, images, shapes and other objects onto the canvas element. It provides the 2D rendering context for the drawing surface of a canvas element.

```
// get the canvas's 2d context
```

```
var canvas = document.getElementById("the-canvas-id");
```

```
var context = canvas.getContext("2d");
```

There are other rendering contexts for canvas that are not covered in this subject:

WebGLRenderingContext,
WebGL2RenderingContext

Hello World

HELLO WORLD

Hello World

Start

Hello World

HELLO WORLD

```
<canvas id="canvas" width="1300" height="500"  
    style="border:1px solid black;">  
// // // ) // // // ) // // //  
Your browser does not support canvas.  
</canvas>
```


Start

```
<button onClick="drawTextHello()">
```

Start

</button>

Hello World

HELLO WORLD

Hello World

```
function drawTextHello() {  
    // get the canvas's 2d context  
  
    // fillText  
  
    // strokeText  
  
}
```

Hello World

```
<canvas id="canvas" width="1300" height="500"
```

```
style="border:1px solid black;">
```

HELLO W Your browser does not support canvas.

```
</canvas>
```

Hello World

Start

```
// get the canvas's 2d context
```

```
var canvas = document.getElementById("canvas");
```

```
var context = canvas.getContext("2d");
```

Hello World

HELLO WORLD

Hello World

```
// fillText
```

```
context.font = "italic small-caps bold 50px Arial";
```

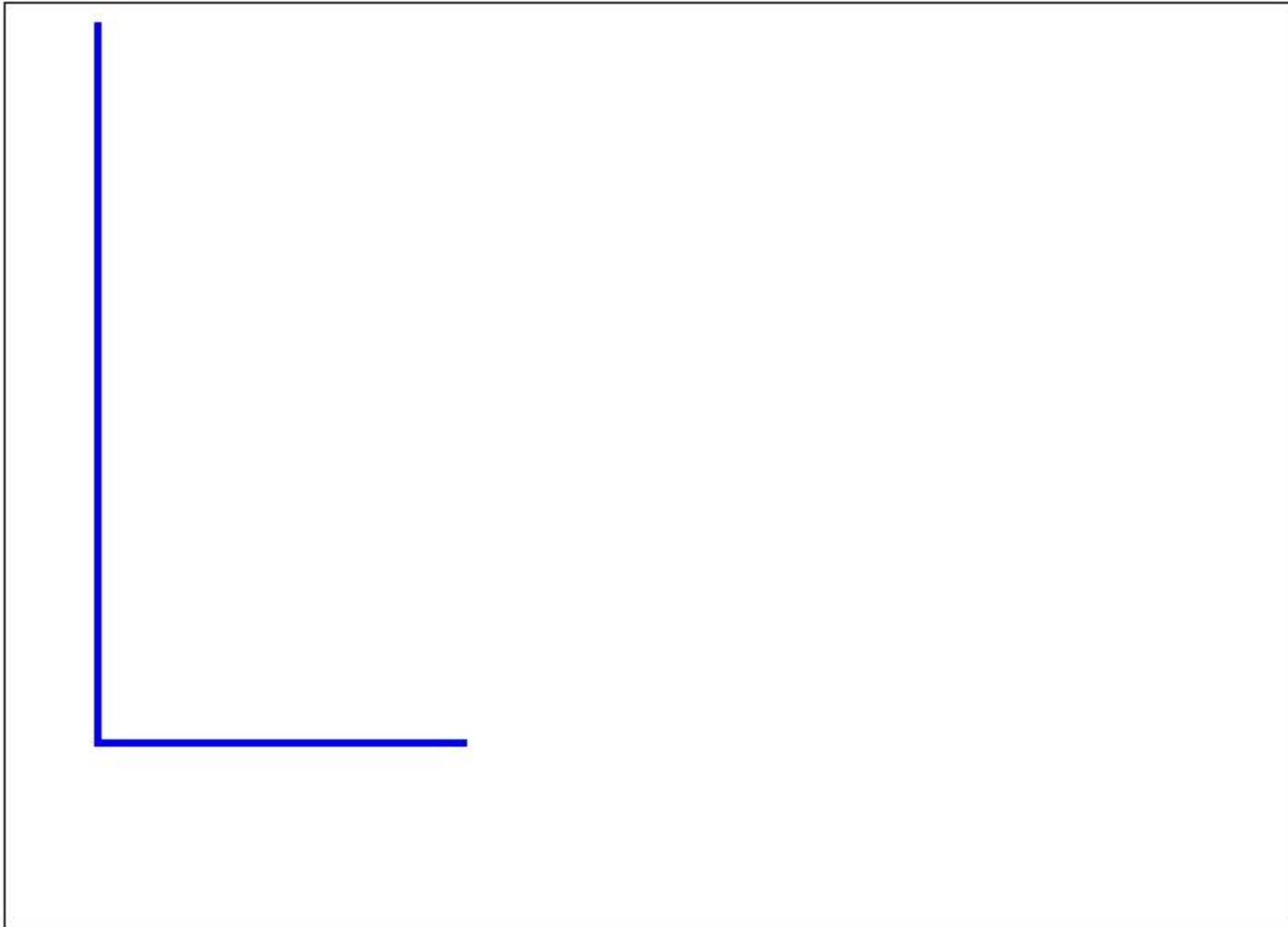
```
context.fillText("Hello World", 200, 100);
```

```
// strokeText
```

```
context.font = "oblique 100px Courier New";
```


```
context.strokeText("Hello World", 250, 300);
```

Stroke Demo 1



Start

Stroke Demo 1



```
<canvas id="canvas" width="700" height="500"  
style="border:1px solid black;">  
Your browser does not support canvas.  
</canvas>
```

```
<br /><br />
```

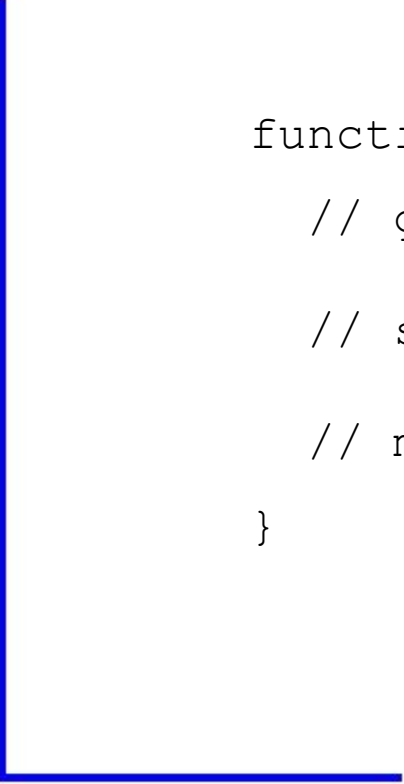
```
<button onClick="strokeDemo()">
```

Start

```
</button>
```

Start

Stroke Demo 1

A blue L-shaped stroke is drawn on a white canvas. The stroke starts at the top left, goes vertically down, and then turns 90 degrees to the right to go horizontally. The code to the right explains how this stroke is created using a canvas's 2D context.

```
function strokeDemo() {  
    // get the canvas's 2d context  
  
    // specify the path  
  
    // make the stroke along the path  
}
```

Start

Stroke Demo 1

```
// get the canvas's 2d context
```

```
var canvas = document.getElementById("canvas");
```

```
var context = canvas.getContext("2d");
```

```
<canvas id="canvas" width="700" height="500"
```

```
style="border:1px solid black;">
```

```
Your browser does not support canvas.
```

```
</canvas>
```

Start

Stroke Demo 1

(0,0)

X

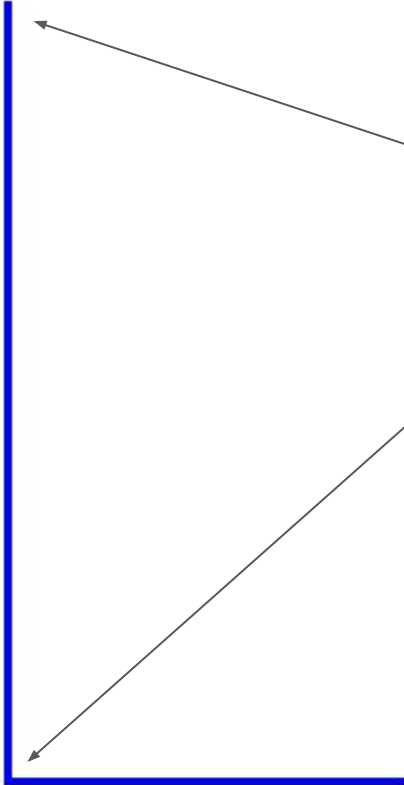
// specify the path

```
context.beginPath();
```

```
context.moveTo(50, 10);
```

```
context.lineTo(50, 400);
```

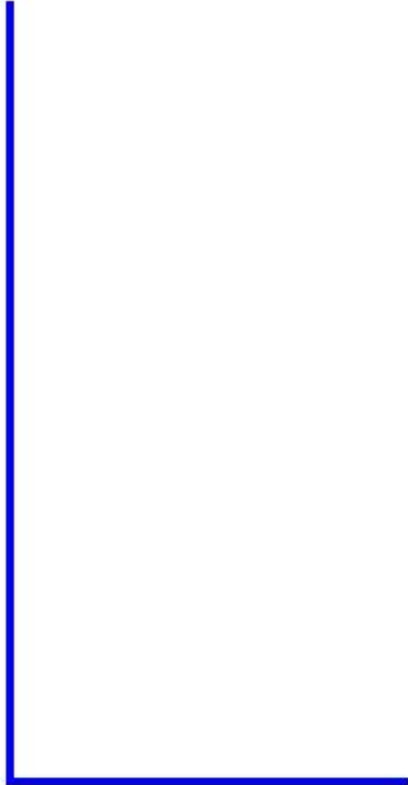
```
context.lineTo(250, 400);
```



Start

Y

Stroke Demo 1

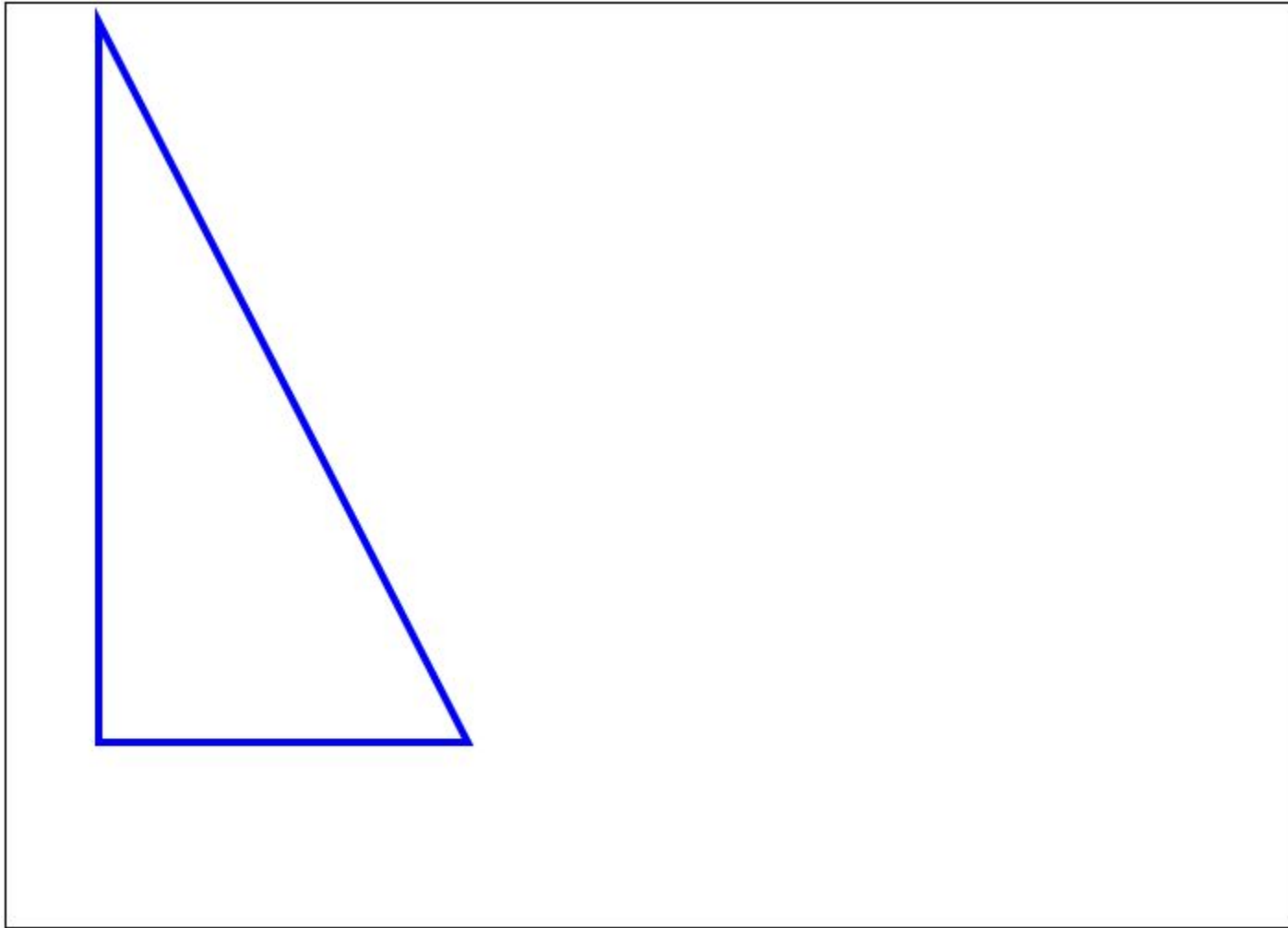


```
// specify the path
context.beginPath();
context.moveTo(50, 10);
context.lineTo(50, 400);
context.lineTo(250, 400);

// make the stroke along the path
context.strokeStyle = "blue";
context.lineWidth = "4";
context.stroke();
```

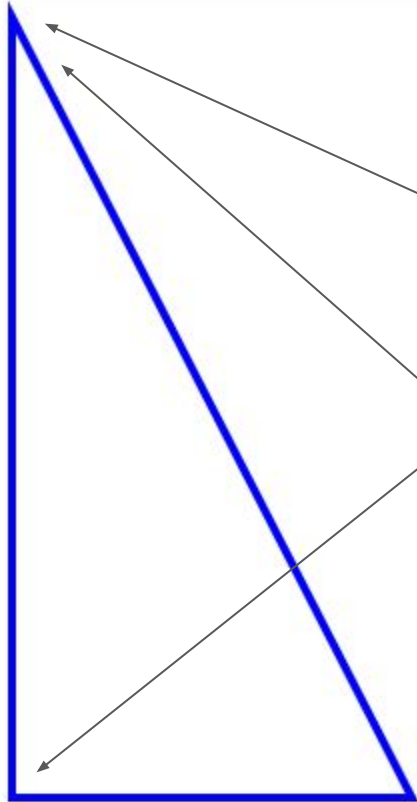
Start

Stroke Demo 2



Start

Stroke Demo 2



// specify the path

```
context.beginPath();
```

```
context.moveTo(50, 10);
```

```
context.lineTo(50, 400);
```

```
context.lineTo(250, 400);
```

```
context.closePath();
```

Start

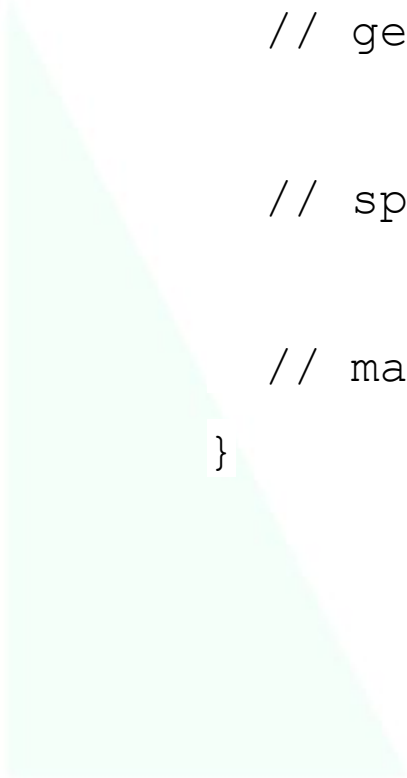
Fill Demo 1



Start

Fill Demo 1

```
function fillDemo() {  
    // get the canvas's 2d context  
  
    // specify the path  
  
    // make the fill of the region enclosed by the path  
}
```



Start

Fill Demo 1



```
// get the canvas's 2d context
```

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

Start

Fill Demo 1



```
// specify the path
```

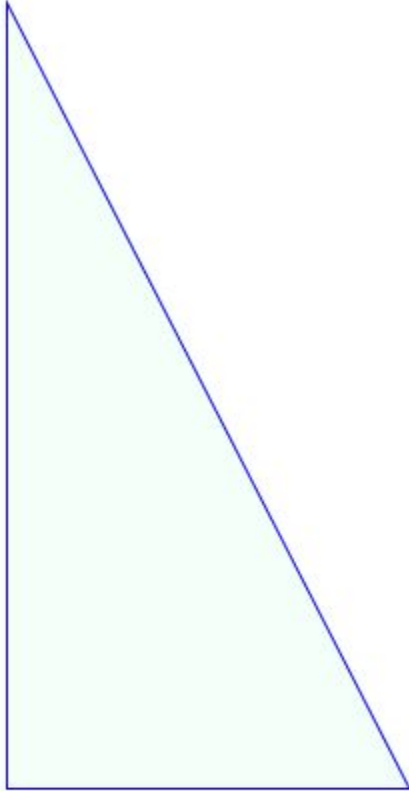
```
context.beginPath();  
context.moveTo(50, 10);  
context.lineTo(50, 400);  
context.lineTo(250, 400);  
context.closePath();
```

```
// make the fill of the region enclosed by the path
```

```
context.fillStyle="#F5FFFA";  
context.fill();
```

Start

Fill Demo 2



```
// specify the path
```

```
context.beginPath();
```

```
context.moveTo(50, 10);
```

```
context.lineTo(50, 400);
```

```
context.lineTo(250, 400);
```

```
context.closePath();
```

```
// make the stroke along the path
```

```
context.strokeStyle = "blue";
```

```
context.lineWidth = "2";
```

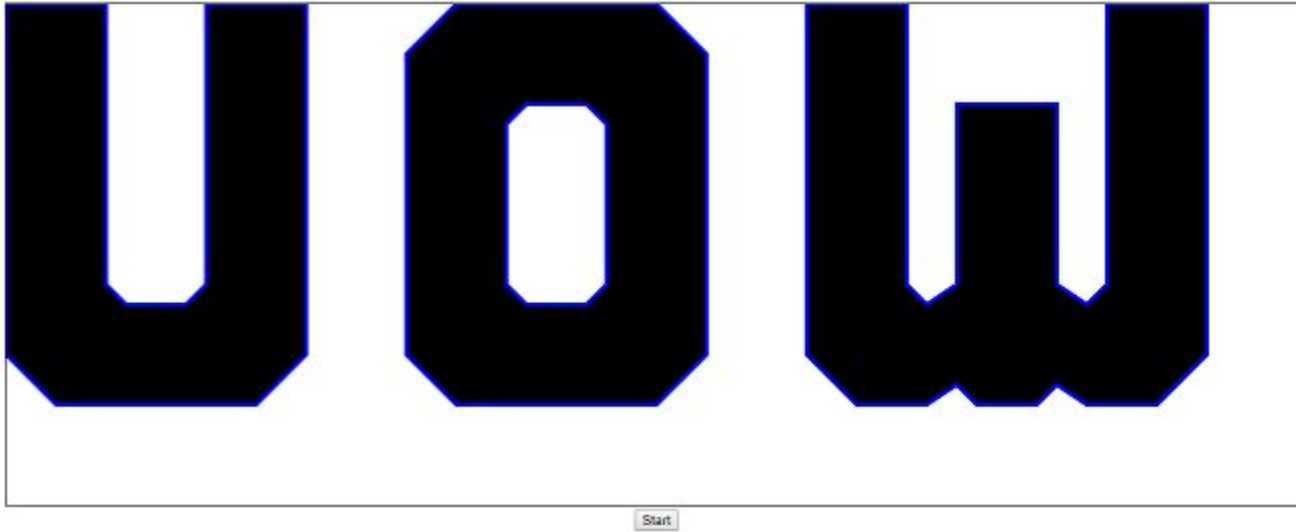
```
context.stroke();
```

```
// make the fill of the region enclosed by the path
```

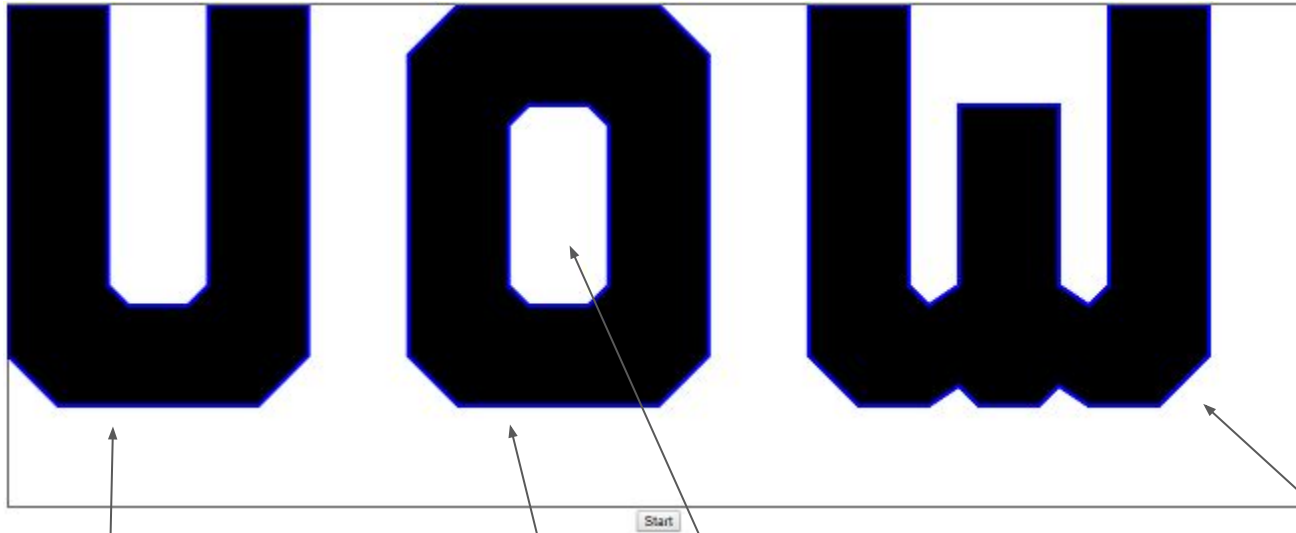
```
context.fillStyle="#F5FFFA";
```

```
context.fill();
```

UOW



UOW



1. letter U
filled with black

2. letter O (**outer**)
filled with black

3. letter O (**inner**)
filled with white

4. letter W
filled with black

UOW



```
<canvas id="canvas" width="1300"  
height="500" style="border:1px solid  
black;">
```

Your browser does not support canvas.

```
</canvas>
```

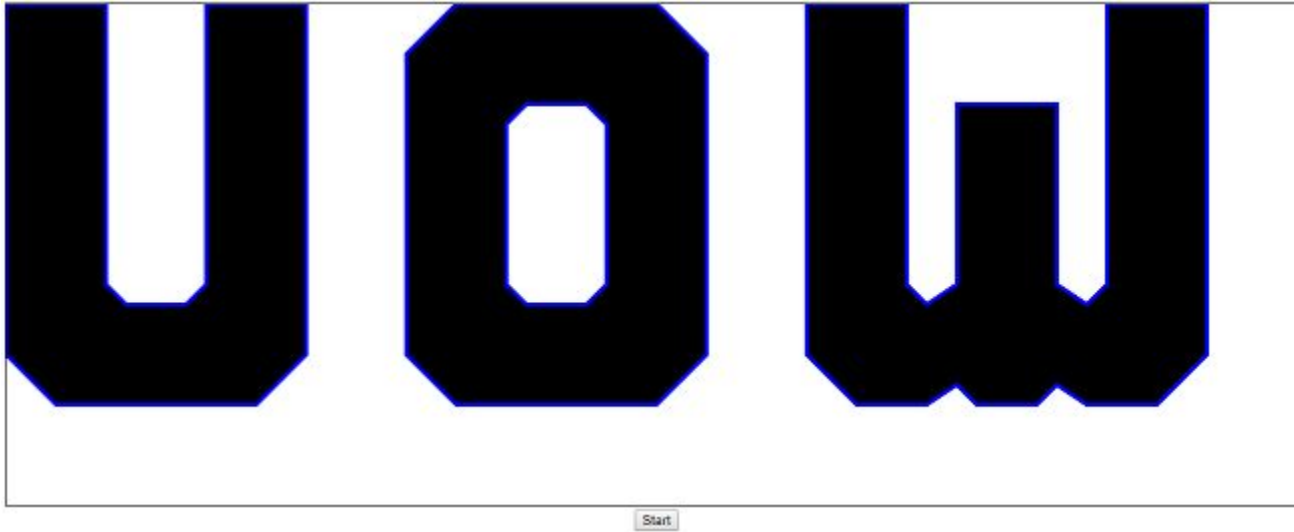
```
<br /><br />
```

```
<button onClick="drawUOW()">
```

Start

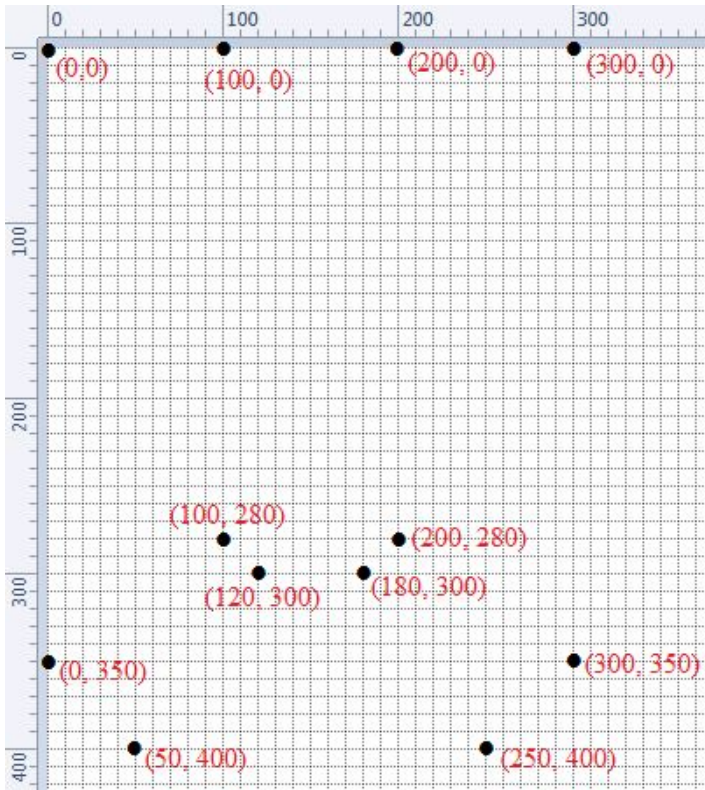
```
</button>
```

UOW



```
function drawUOW() {  
    // get the canvas's 2d context  
    // letter U  
    // letter O (outer)  
    // letter O (inner)  
    // letter W  
}
```

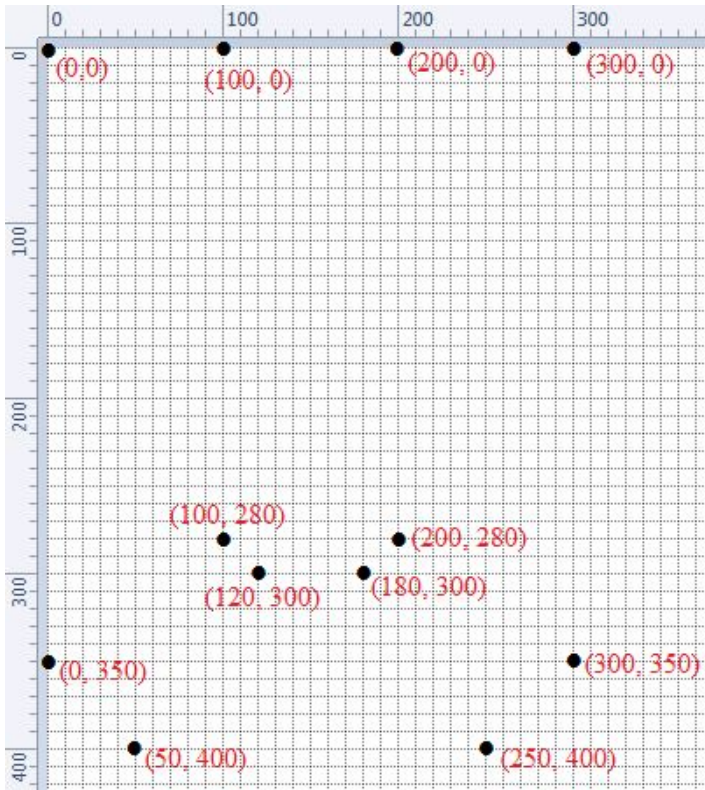
UOW



```
// letter U
```

```
context.beginPath();  
context.moveTo(0, 0);  
context.lineTo(0, 350);  
context.lineTo(50, 400);  
context.lineTo(250, 400);  
context.lineTo(300, 350);  
context.lineTo(300, 0);  
context.lineTo(200, 0);  
context.lineTo(200, 280);  
context.lineTo(180, 300);  
context.lineTo(120, 300);  
context.lineTo(100, 280);  
context.lineTo(100, 0);  
context.closePath();
```

UOW



```
// letter U
```

```
context.beginPath();
```

```
context.moveTo(0, 0);
```

```
...
```

```
context.lineTo(100, 0);
```

```
context.closePath();
```

```
context.fillStyle="black";
```

```
context.fill();
```

```
context.strokeStyle="blue";
```

```
context.lineWidth = "4";
```

```
context.stroke();
```

Drag and Drop

Need to specify 2 types of elements:

- ***Draggable elements:*** *elements that we can be dragged*
- ***Droppable elements:*** *elements that can be dropped on*

The user can select **draggable elements** with a mouse, drag the elements to a **droppable element**, and drop those elements by releasing the mouse button.

Drag and Drop

Need to specify 2 types of elements:

- ***Draggable elements:*** *elements that we can be dragged*
- ***Droppable elements:*** *elements that can be dropped on*

```
<element id="drag-id" draggable="true"  
onDragStart="dragStart(event) " >draggable  
element</element>
```

```
<element id="drop-id" onDrop="drop(event) "  
onDragOver="dragOver(event) ">droppable element</element>
```

Drag and Drop


***Draggable elements:** elements that we can be dragged*

```
<element id="drag-id" draggable="true"
onDragStart="dragStart(event)" >draggable
element</element>
```

```
function dragStart(event) {
  // get the dragged element ID
  var dragId = event.target.id;

  // store the dragged element ID into the
  //dataTransfer object
  event.dataTransfer.setData("dragId", dragId);
}
```

dragStart event is fired when
the user starts dragging an
element



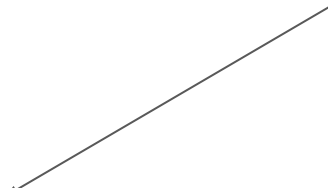
Drag and Drop

***Draggable elements:** elements that we can be dragged*

```
<element id="drag-id" draggable="true"  
onDragStart="dragStart(event)" >draggable  
element</element>
```

```
function dragStart(event) {  
    // get the dragged element ID  
    var dragId = event.target.id;
```

*We need to know what
object we are dragging*



```
    // store the dragged element ID into the dataTransfer object  
    event.dataTransfer.setData("dragId", dragId);  
}
```

The DataTransfer object is used to hold the data that is being dragged during a drag and drop operation.

Drag and Drop

***Droppable elements:** elements that can be dropped on*

```
<element id="drop-id" onDrop="drop(event)"  
onDragOver="dragOver(event)">droppable element</element>
```

```
function drop(event) {
```

```
    // get the drop element ID
```

```
    var dropId = event.target.id;
```

```
    // retrieve the dragged element ID from the dataTransfer object
```

```
    var dragId = event.dataTransfer.getData("dragId");
```

```
    // do the dropping logic
```

```
}
```

*The **drop** event is fired when an element is dropped on a valid drop target.*

Drag and Drop

***Droppable elements:** elements that can be dropped on*

```
<element id="drop-id" onDrop="drop(event)"  
onDragOver="dragOver(event)">droppable element</element>
```

*What is the **dragOver** event for?*

*Calling the `preventDefault()` method during a **dragOver** event will indicate that a drop is allowed at that location.*

```
function dragOver(event) {  
    event.preventDefault();  
}
```

Drag and Drop: Hello World

Drag an **orange** word and drop it on a **red** word.

hello hi bonjour salut

web maze earth world

When “hello” is dropped on “world”, the page displays “hello world”.

hello hi bonjour salut

web maze earth world

hello world


Drag and Drop: Hello World

Drag an **orange** word and drop it on a **red** word.


hello hi bonjour salut

web maze earth world

draggable elements:
elements that we can be dragged



droppable elements:
elements that can be dropped on




Drag and Drop: Hello World

Drag an **orange** word and drop it on a **red** word.

hello hi bonjour salut

web maze earth world

***draggable elements:**
elements that we can drag*



```
<span id="hello" draggable="true"  
onDragStart="dragStart(event)" >hello</span>
```

```
<span id="hi" draggable="true"  
onDragStart="dragStart(event)" >hi</span>
```

```
<span id="bonjour" draggable="true"  
onDragStart="dragStart(event)" >bonjour</span>
```

. . .

Drag and Drop: Hello World

Drag an **orange** word and drop it on a **red** word.

hello hi bonjour salut

web maze earth world

← **draggable elements:**
*elements that can be
dropped on*

```
<span id="web" onDrop="drop(event)"  
onDragOver="dragOver(event)">web</span>
```

```
<span id="maze" onDrop="drop(event)"  
onDragOver="dragOver(event)">maze</span>
```

```
<span id="earth" onDrop="drop(event)"  
onDragOver="dragOver(event)">earth</span>
```


. . .

Drag and Drop: Hello World

Drag an **on** `<span id="hello" draggable="true"`
hello `onDragStart="dragStart(event)"` `>hello`

web **maze** **earth** **world**

dragStart event is fired when
the user starts dragging an
element



```
function dragStart(event) {
```

```
    // get the dragged element ID
```

```
    var dragId = event.target.id;
```

```
    // store the dragged element ID into the dataTransfer object
```

```
    event.dataTransfer.setData("dragId", dragId);
```

```
}
```

Drag and Drop: Hello World

Drag an **on** `<span id="hello" draggable="true"`
hello `onDragStart="dragStart(event)"` `>hello`

web **maze** **earth** **world**

```
function dragStart(event) {
```

```
    // get the dragged element ID
```

```
    var dragId = event.target.id;
```

```
    // store the dragged element ID into the dataTransfer object
```

```
    event.dataTransfer.setData("dragId", dragId);
```

```
}
```

If **hello** is dragged, then
`event.target.id = "hello"`
and we store "hello" into the
dataTransfer object

Drag and Drop: Hello World

Drag an orange word and drop it on a red word.

hello hi horizon robot

```
<span id="world" onDrop="drop(event) "  
  onDragOver="dragOver(event) ">world</span>
```

web maze earth world

```
function drop(event) {
```

```
  // get the drop element ID
```

```
  var dropId = event.target.id;
```

```
  // retrieve the dragged element ID from the dataTransfer object
```

```
  var dragId = event.dataTransfer.getData("dragId");
```

```
  // display the message
```

```
  var messageSpan = document.getElementById("message");
```

```
  messageSpan.innerHTML = dragId + " " + dropId;
```

*The **drop** event is fired when an element is dropped on a valid drop target.*

```
}
```

Drag and Drop: Hello World

Drag an **orange** word and drop it on a **red** word.

hello hi horizon robot

```
<span id="world" onDrop="drop(event) "  
  onDragOver="dragOver(event) ">world</span>
```

web maze earth world

What is the **dragOver** event for?

*Calling the `preventDefault()` method during a **dragOver** event will indicate that a drop is allowed at that location.*

```
function dragOver(event) {  
    event.preventDefault();  
}
```

References

- https://www.w3schools.com/html/html5_canvas.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://www.w3schools.com/html/html5_draganddrop.asp
- https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API