CSIT128 / CSIT828

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>
```

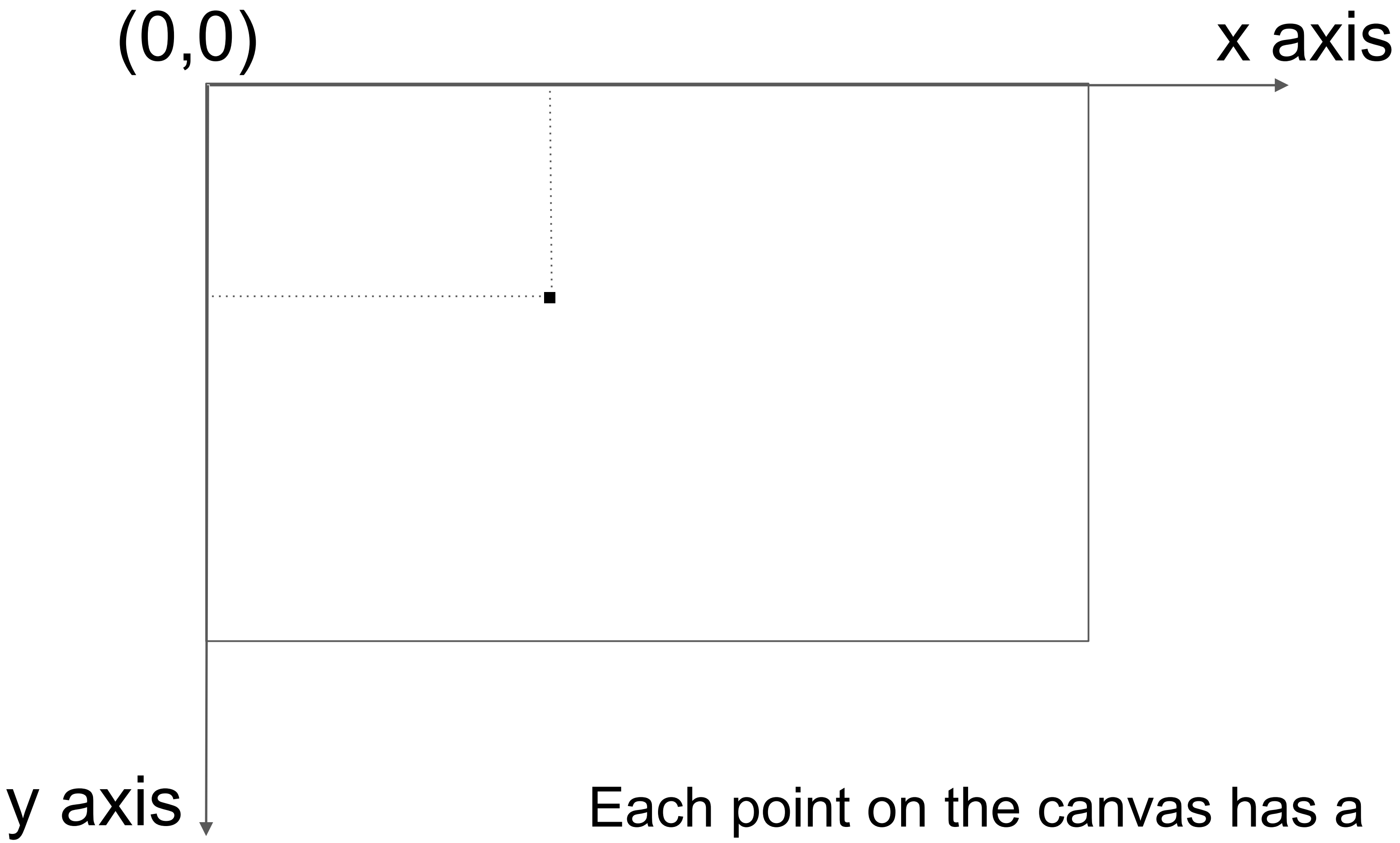
Canvas

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



Each point on the canvas has a coordinate (x,y)

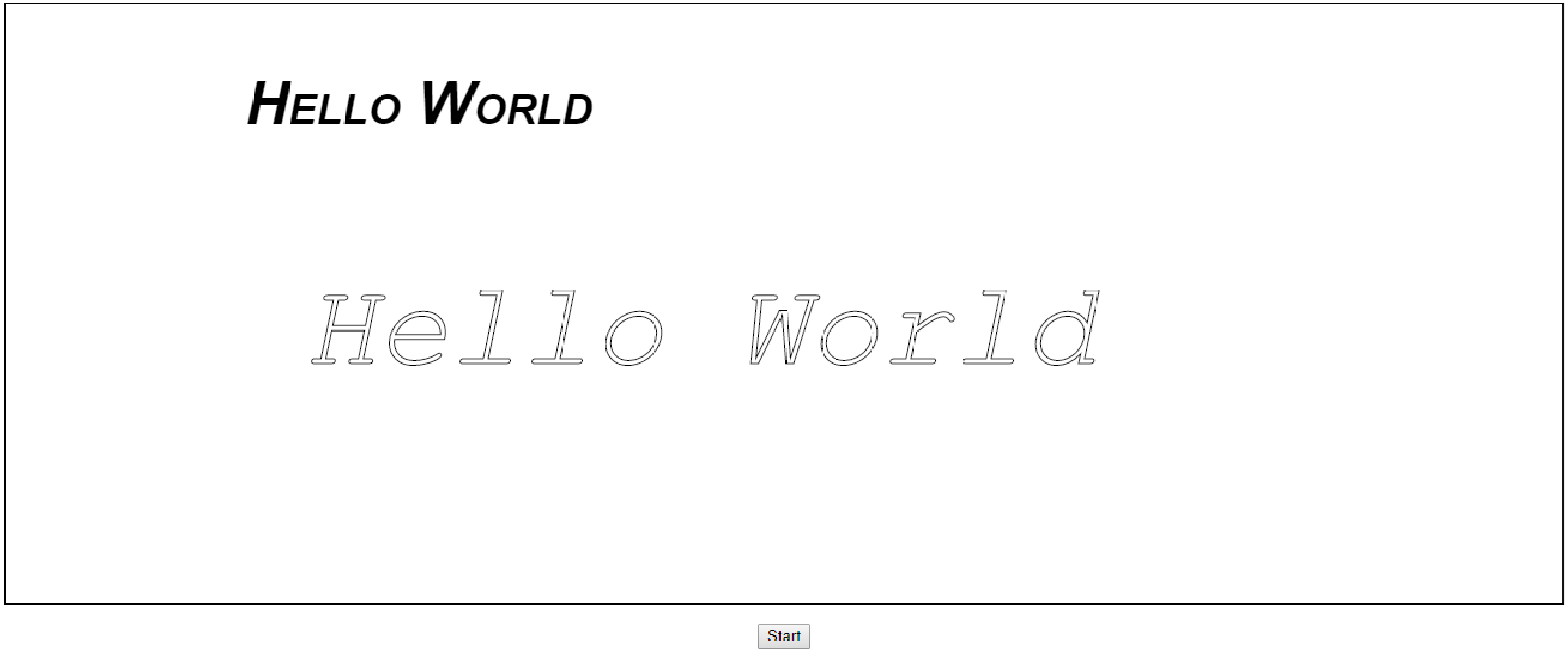
Canvas

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.

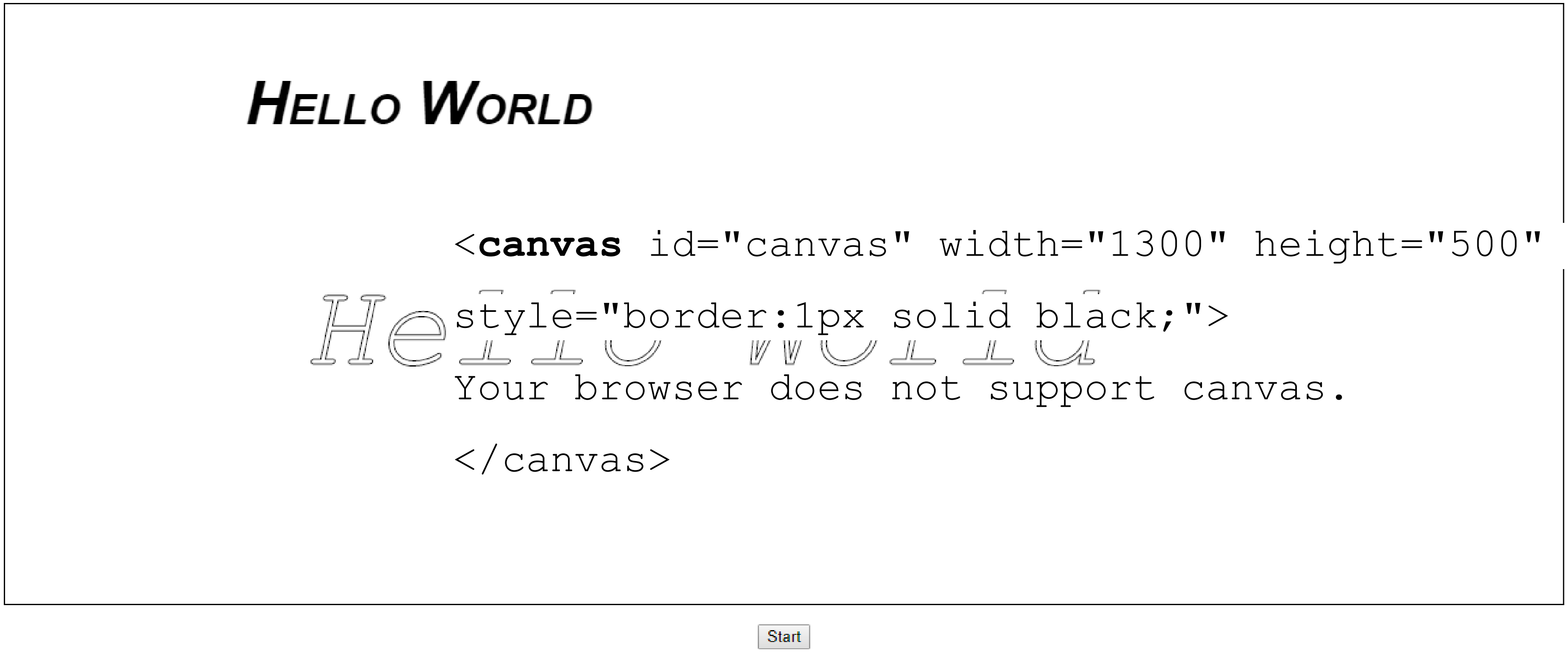
```
// get the canvas's 2d context
var canvas = document.getElementById("the-canvas-id");
var context = canvas.getContext("2d");
```

There are other rendering contexts for canvas that are not covered in this subject:
WebGLRenderingContext,
WebGL2RenderingContext

Hello World

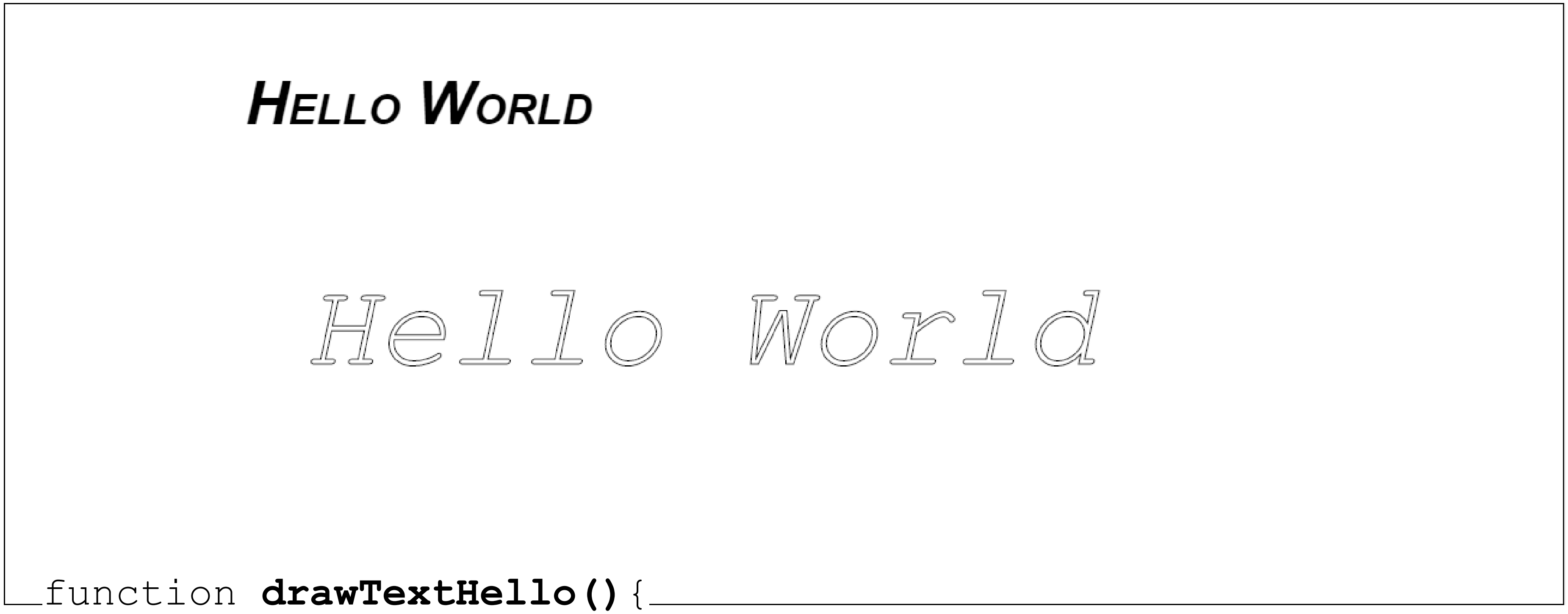


Hello World



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

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

Hello World

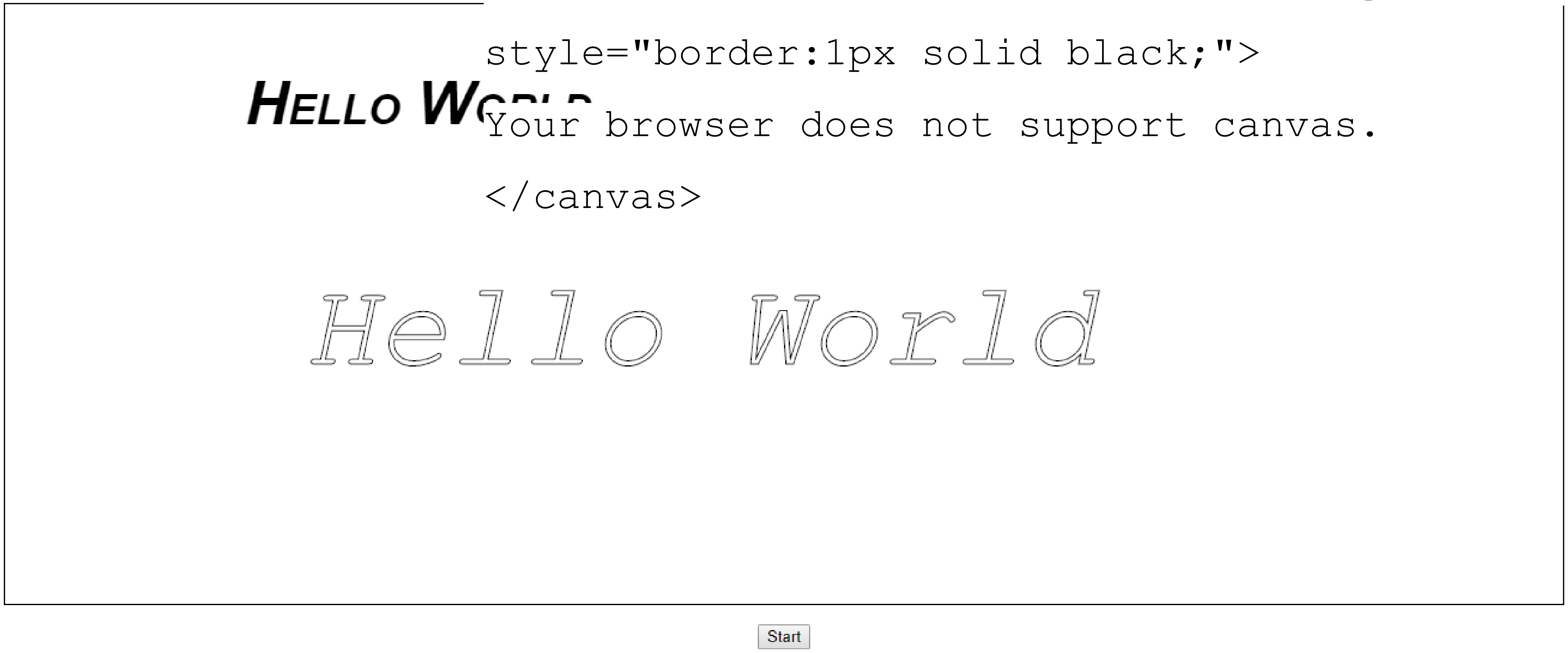


Start

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

  // fillText


  // strokeText

}
```

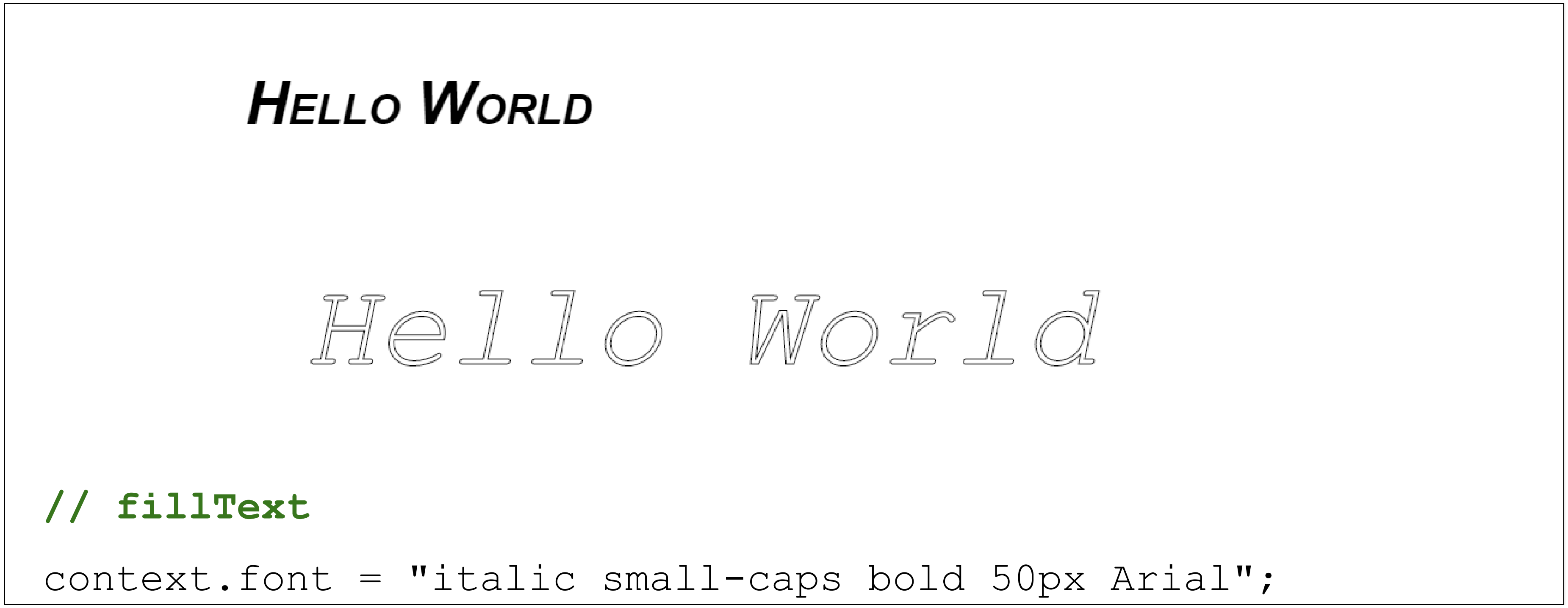
Hello World

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



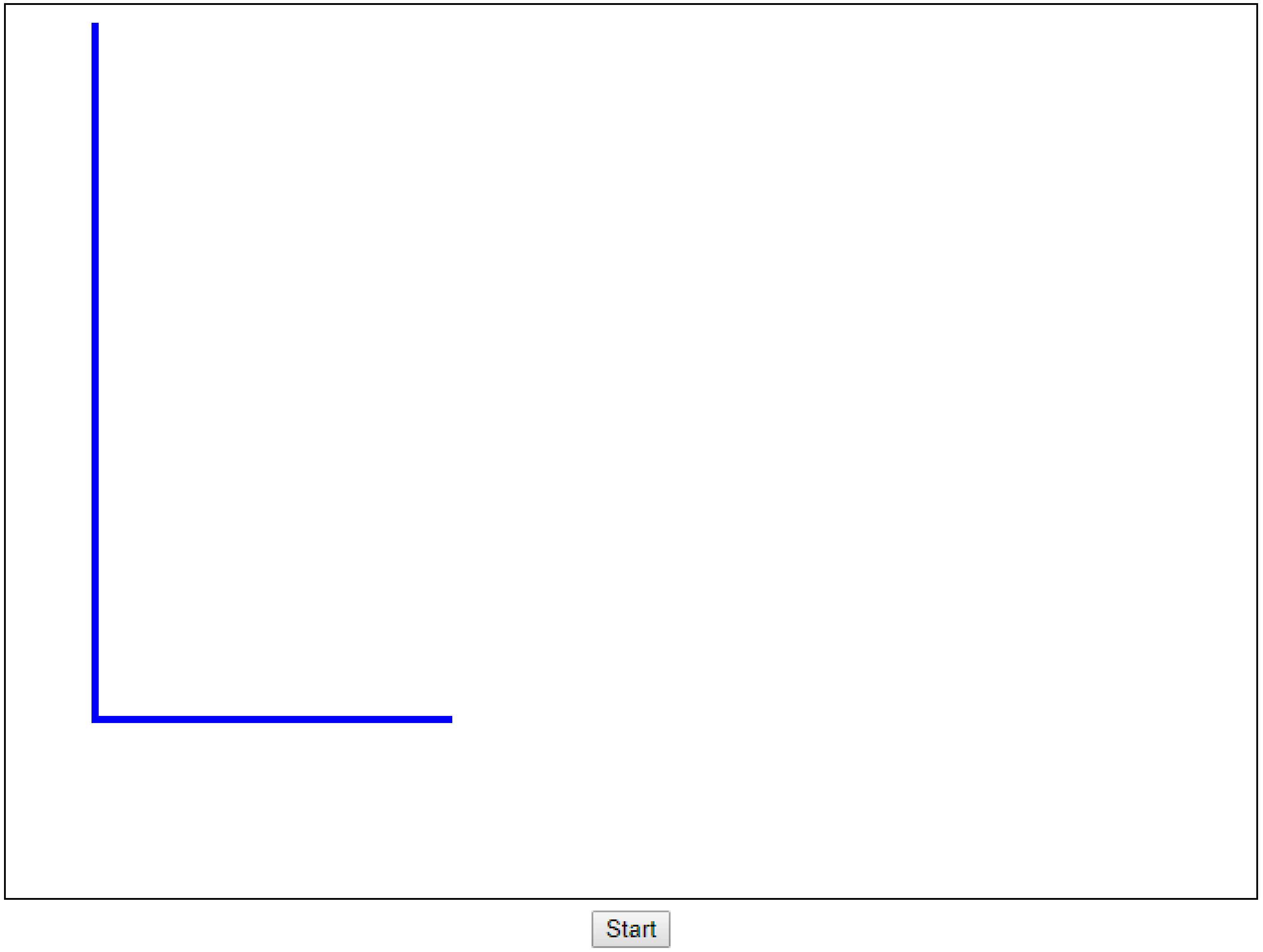
```
// get the canvas's 2d context
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
```

Hello World

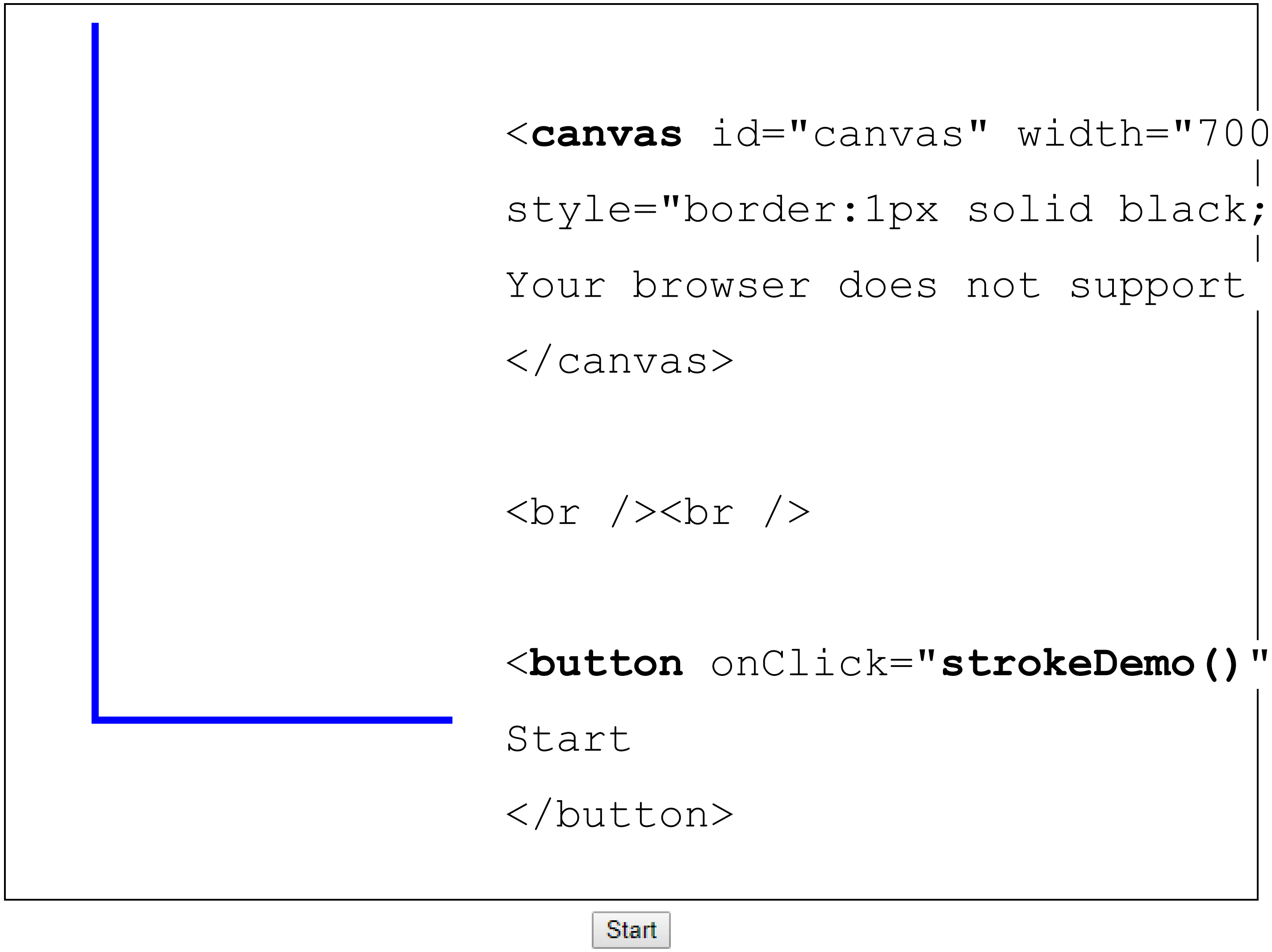


```
// fillText
context.font = "italic small-caps bold 50px Arial";
context.fillText("Hello World", 200, 100);

// strokeText
context.font = "oblique 100px Courier New";
context.strokeText("Hello World", 250, 300);
```

Stroke Demo 1

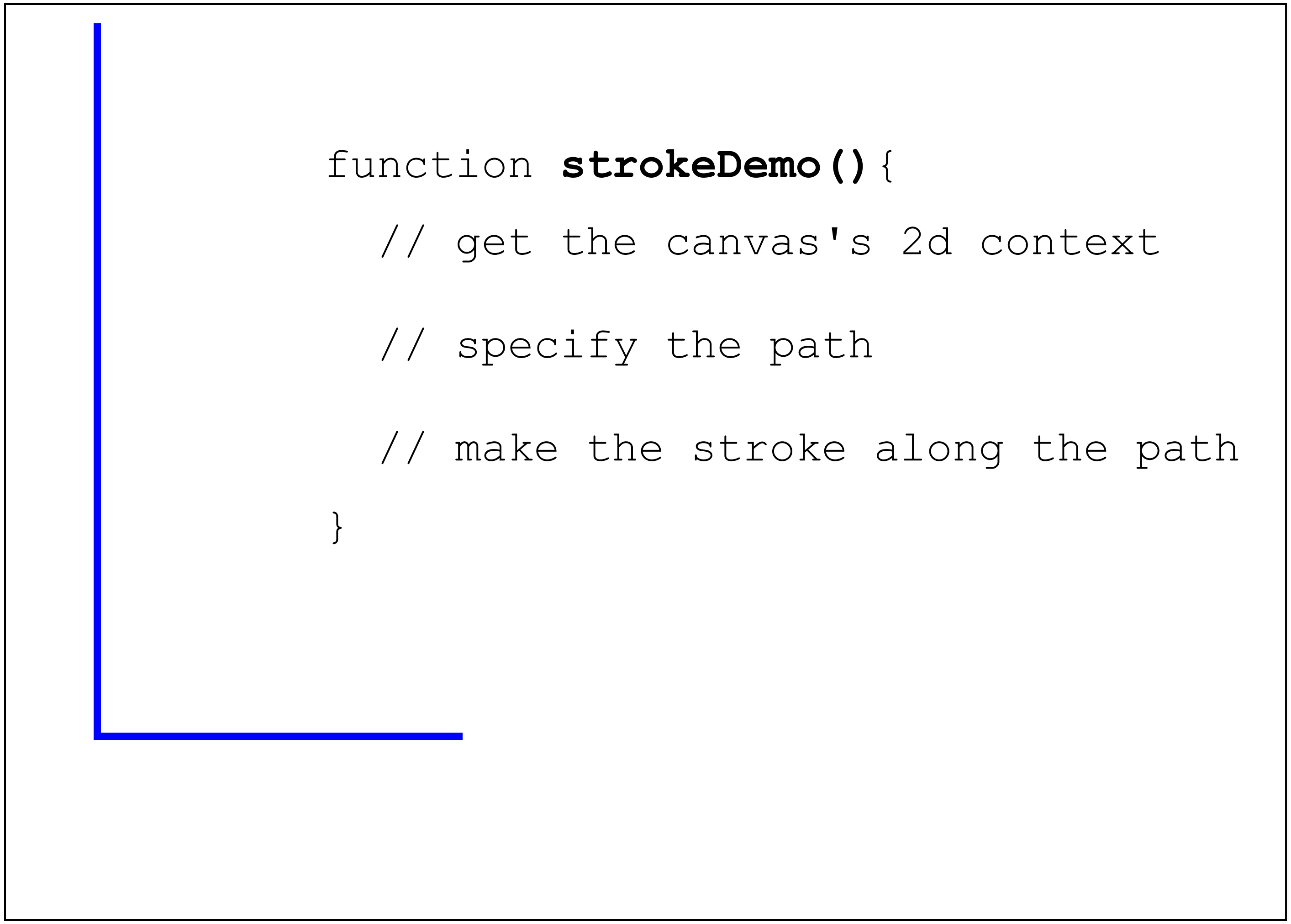


Stroke Demo 1



```
<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>
```

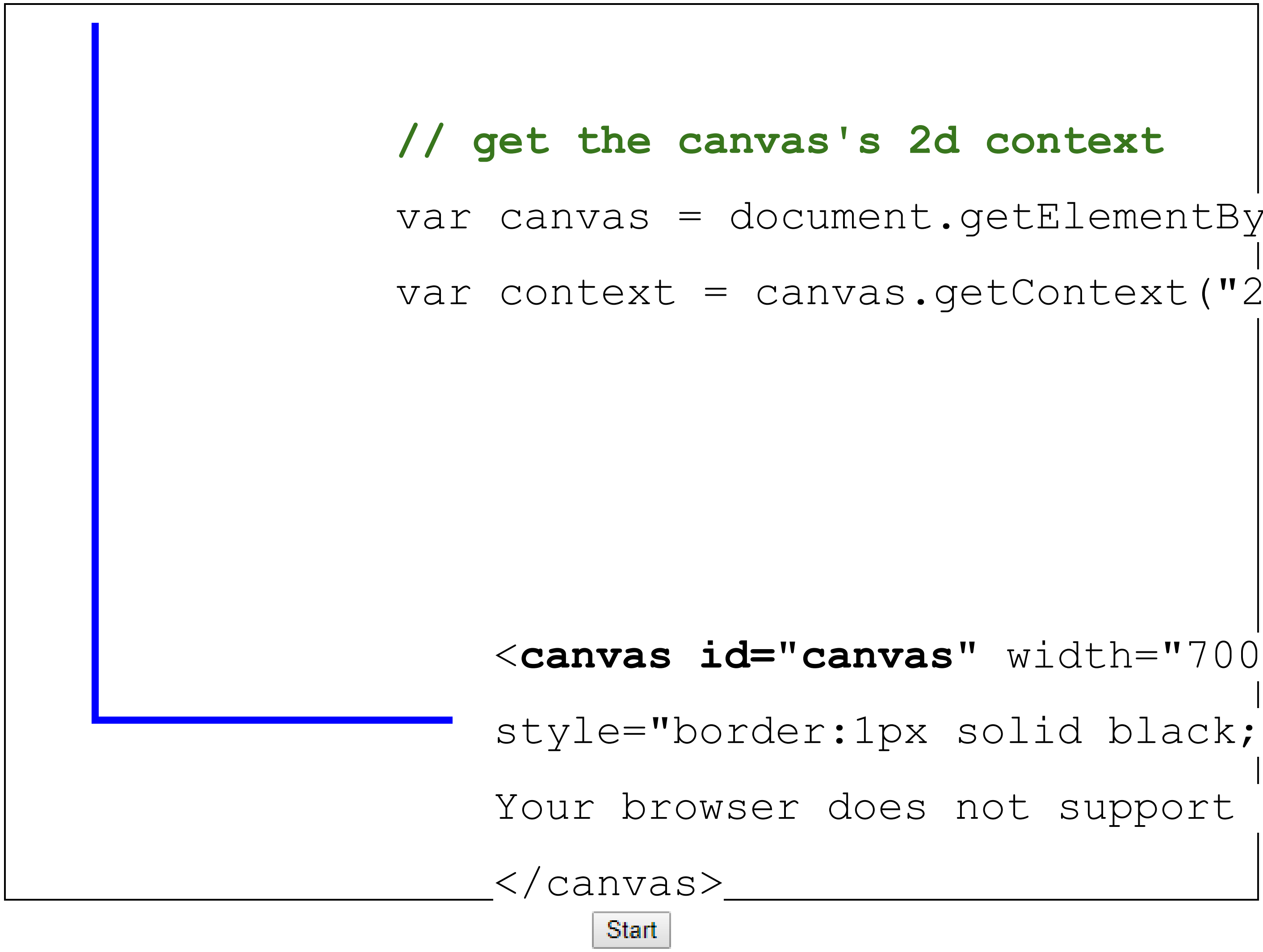
Stroke Demo 1



```
function strokeDemo(){
  // get the canvas's 2d context

  // specify the path

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

Start

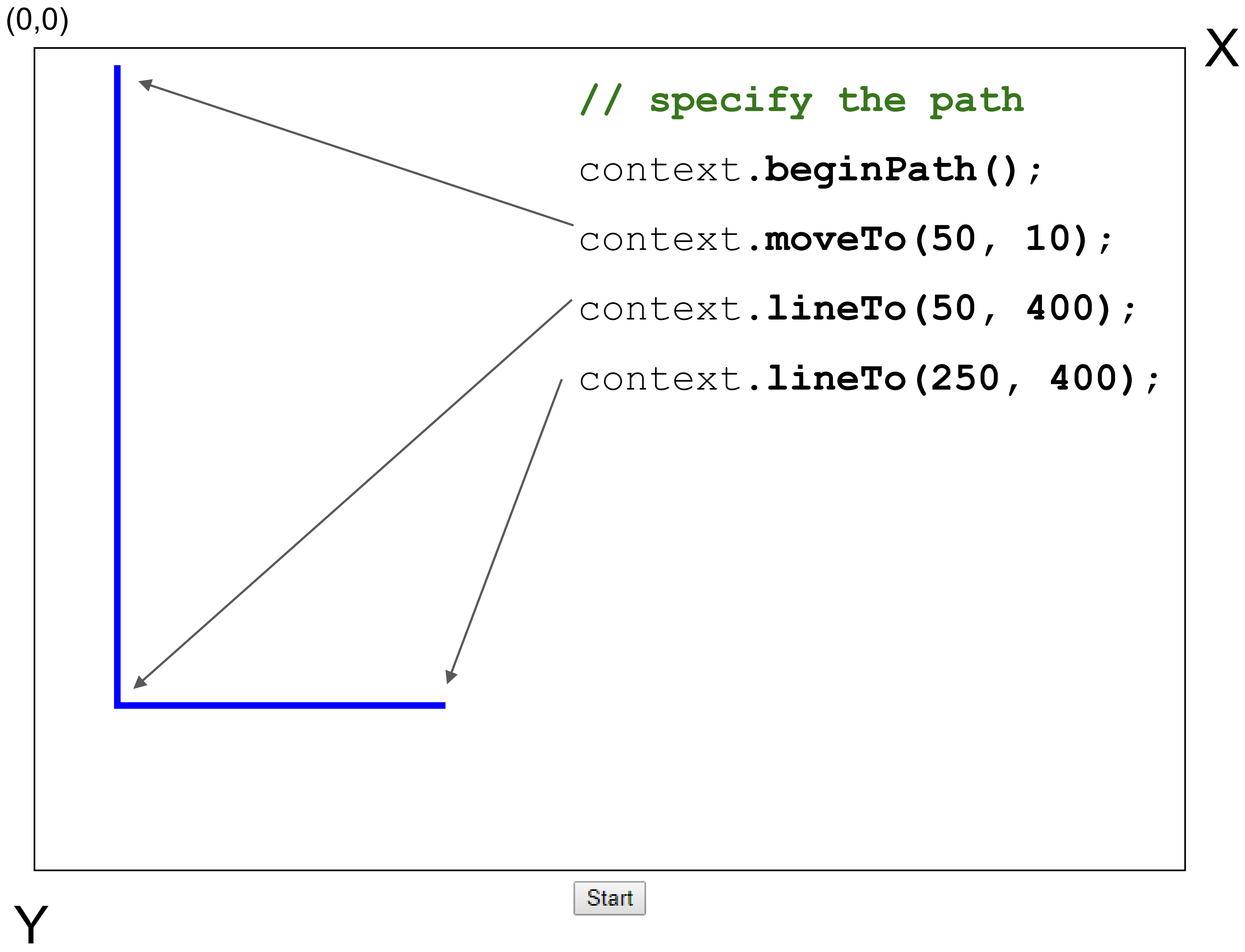
Stroke Demo 1



```
// get the canvas's 2d context
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
```

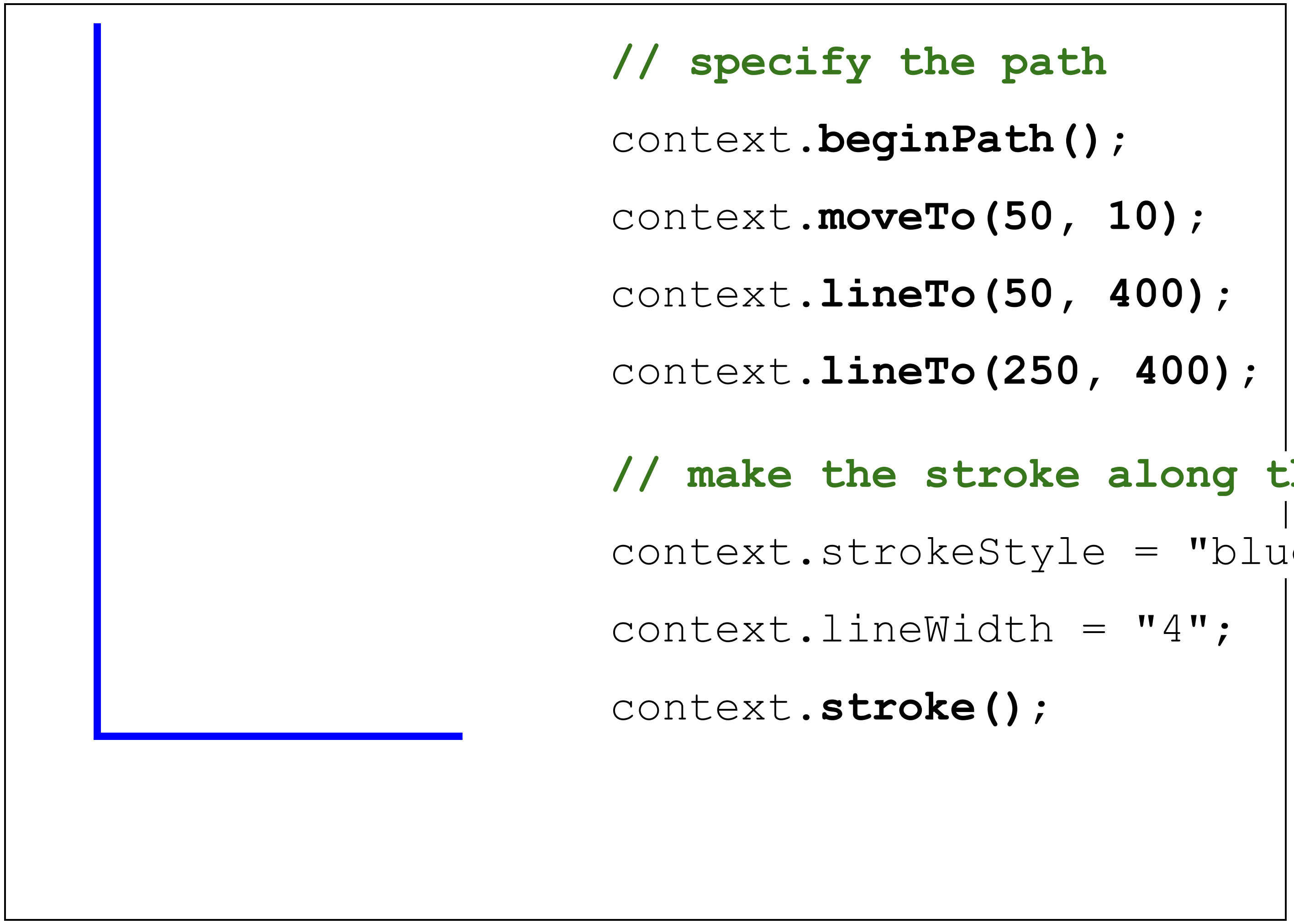
```
<canvas id="canvas" width="700" height="500"
style="border:1px solid black;">
Your browser does not support canvas.
</canvas>
```

Stroke Demo 1



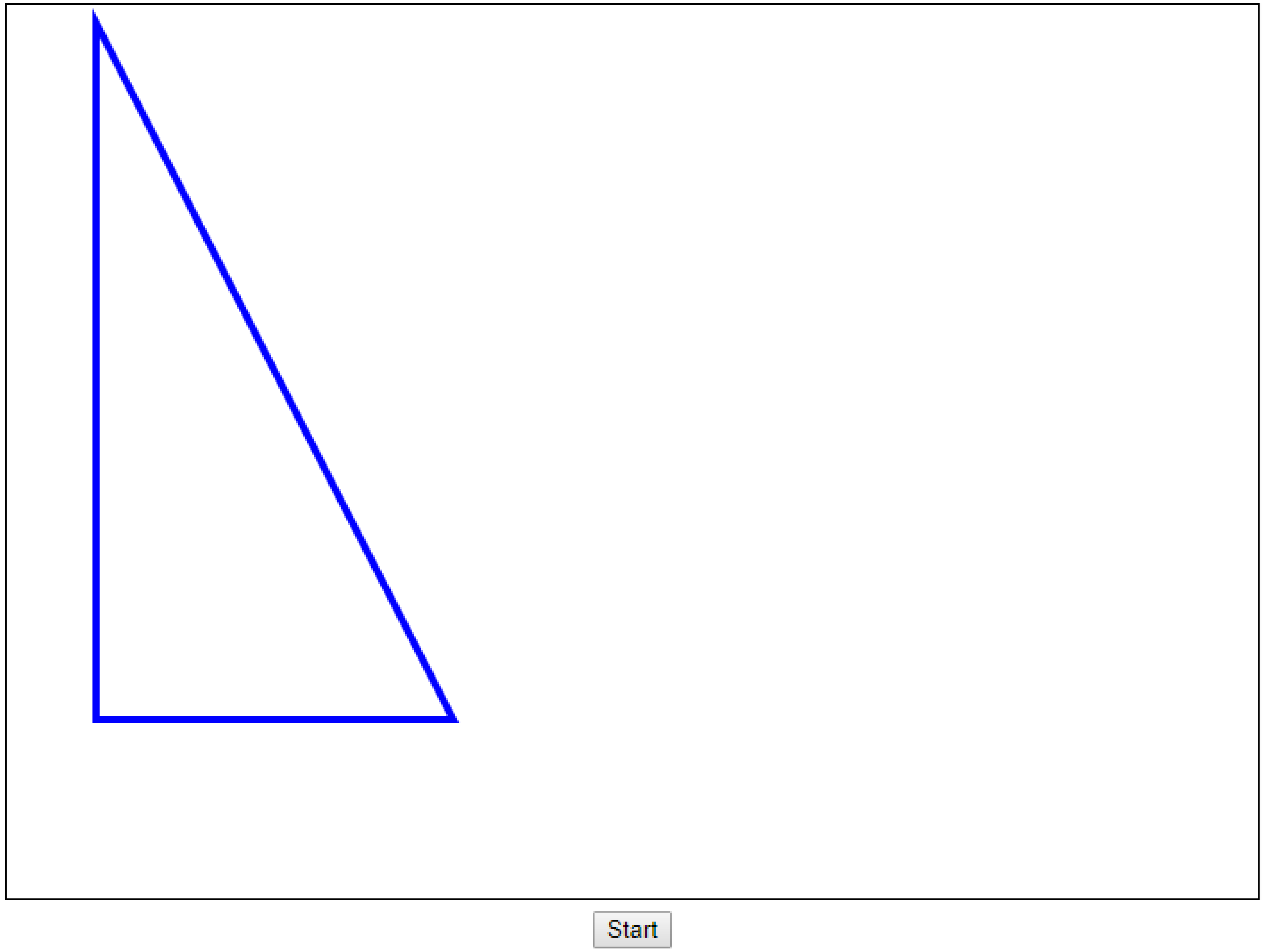
```
// specify the path
context.beginPath();
context.moveTo(50, 10);
context.lineTo(50, 400);
context.lineTo(250, 400);
```

Stroke Demo 1

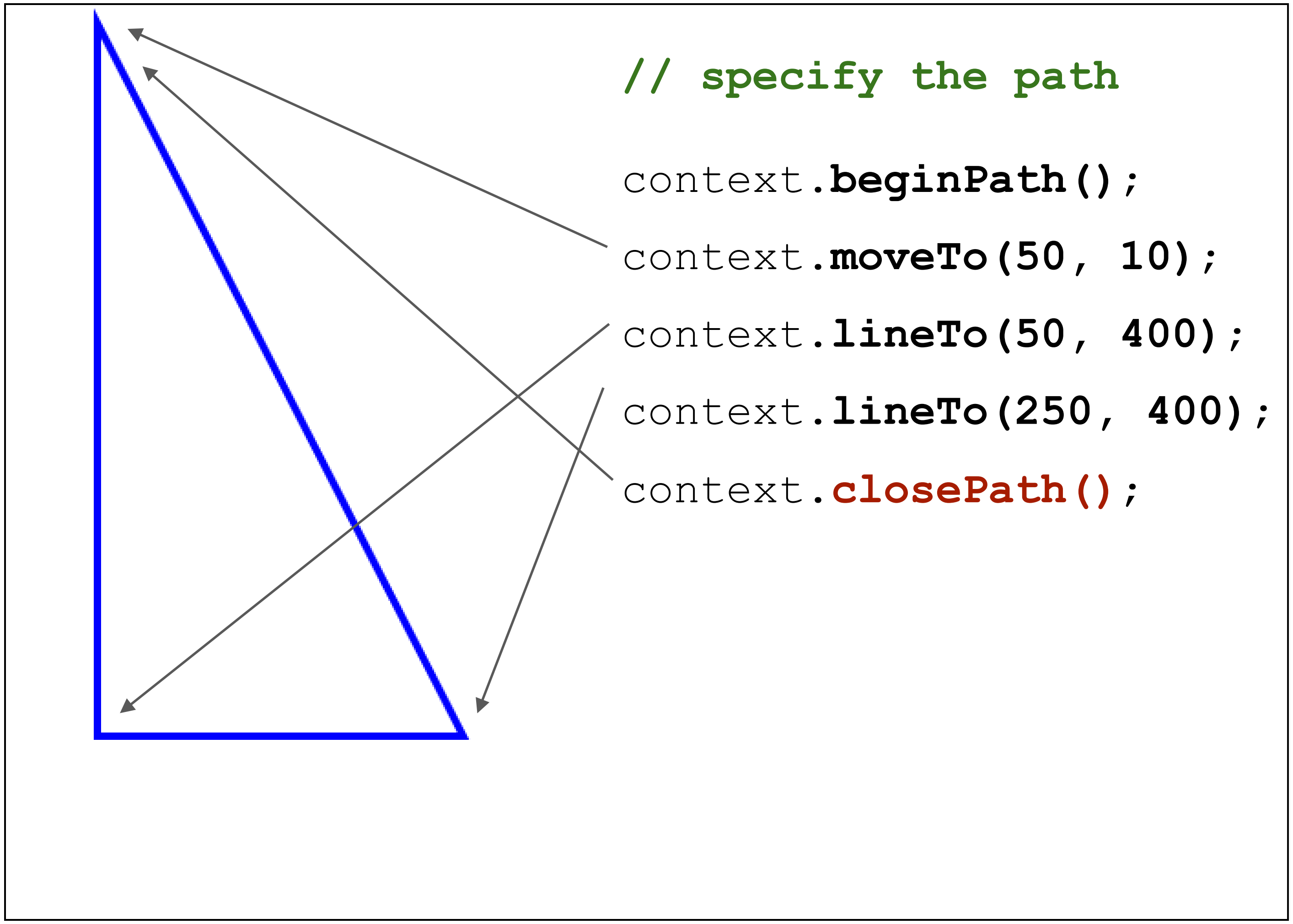


```
// specify the path
context.beginPath();
context.moveTo(50, 10);
context.lineTo(50, 400);
context.lineTo(250, 400);

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

Stroke Demo 2

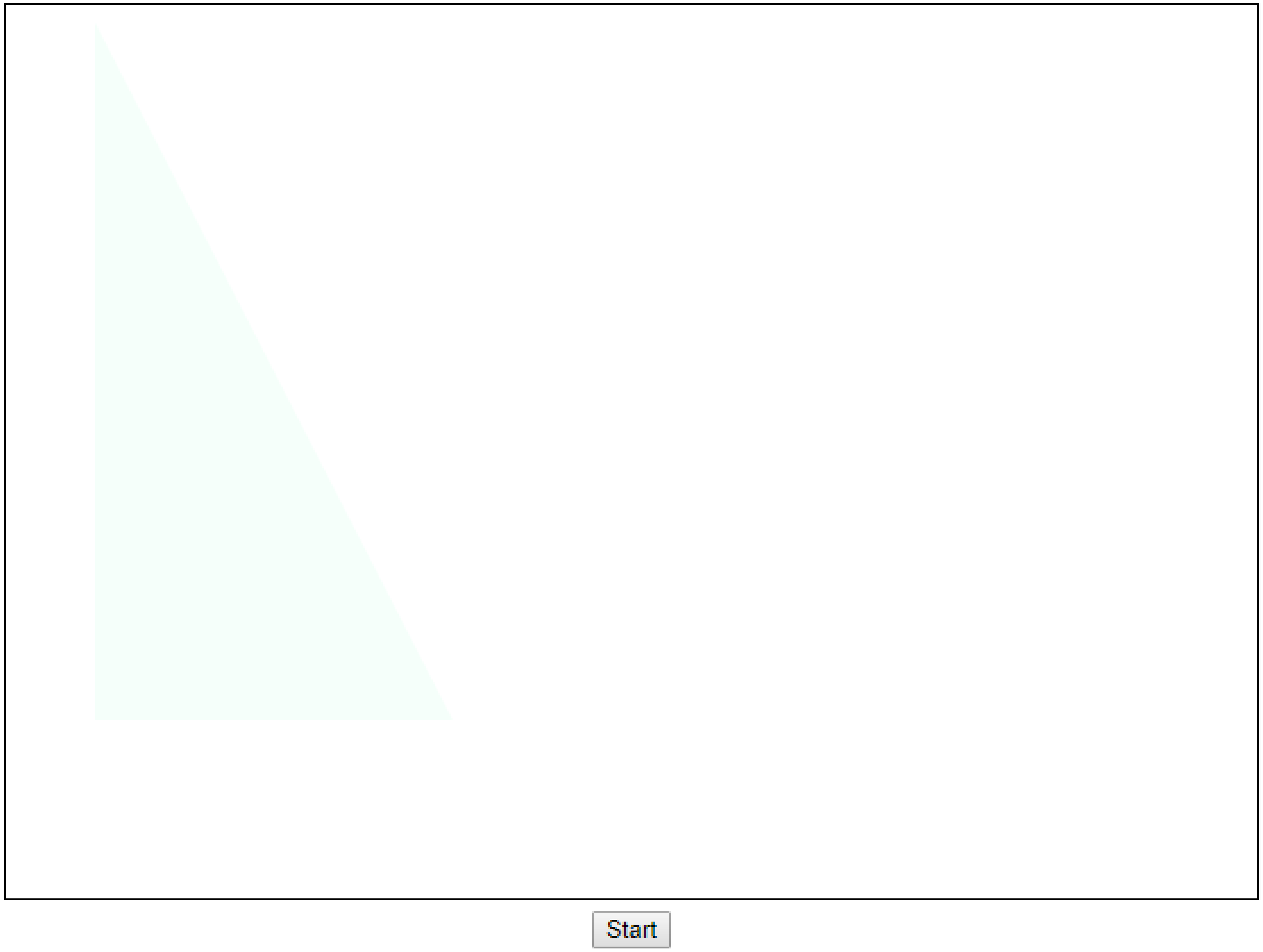


Stroke Demo 2



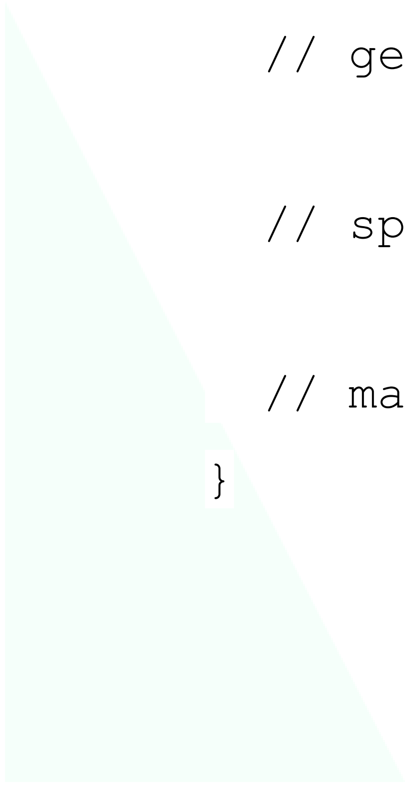
```
// specify the path

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

Fill Demo 1



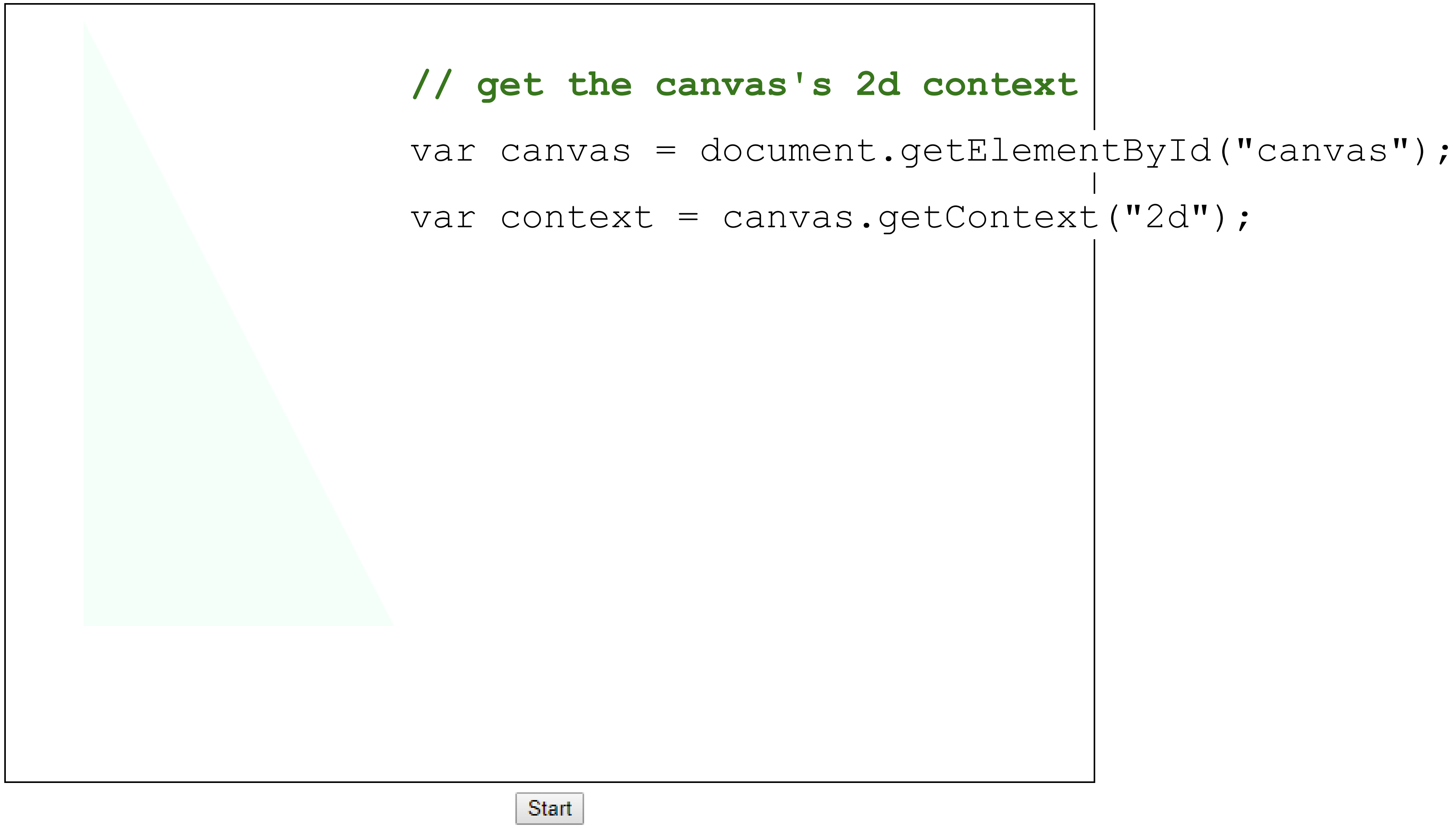
Fill Demo 1

```
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



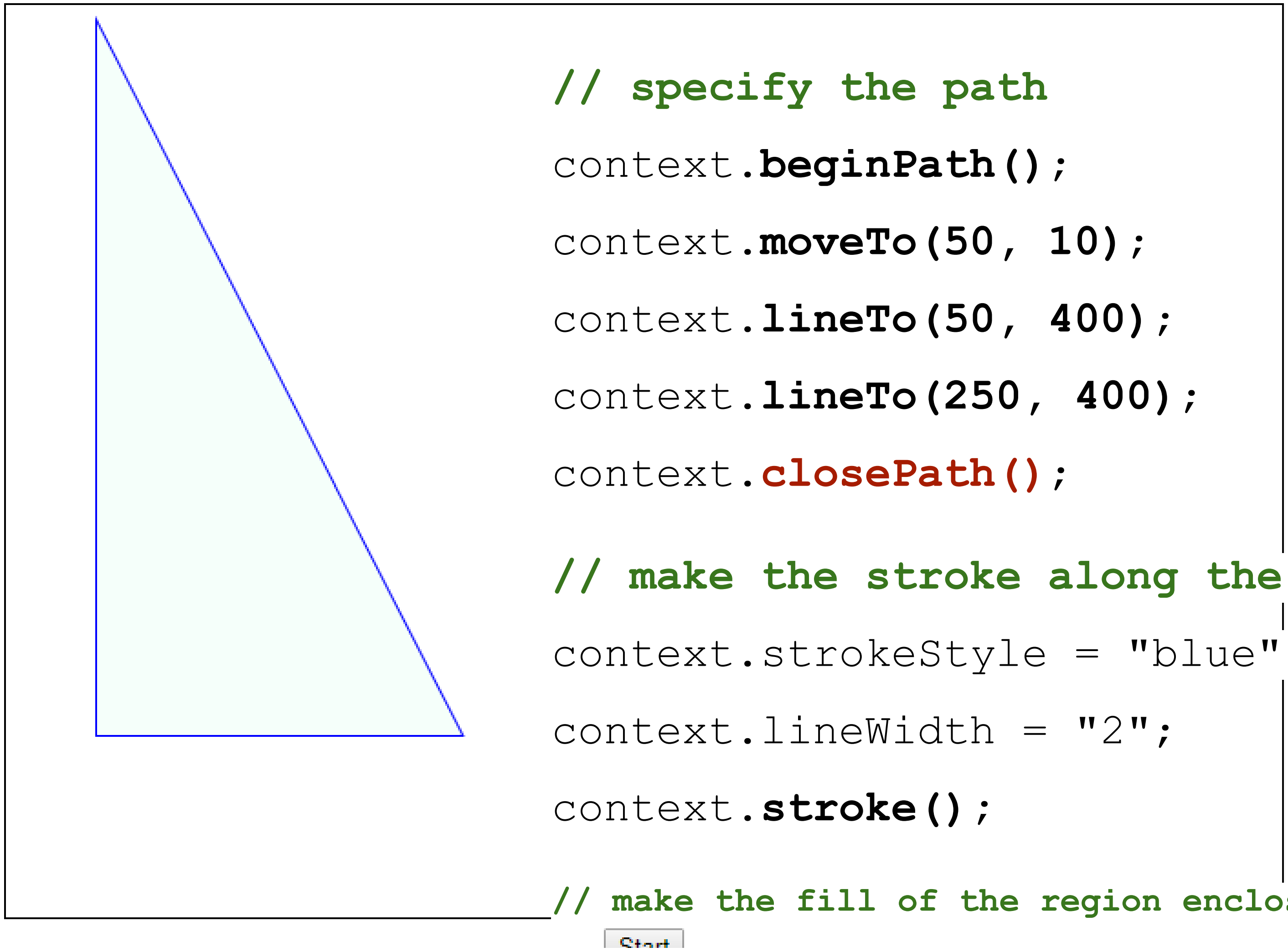
```
// get the canvas's 2d context
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
```

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();
```

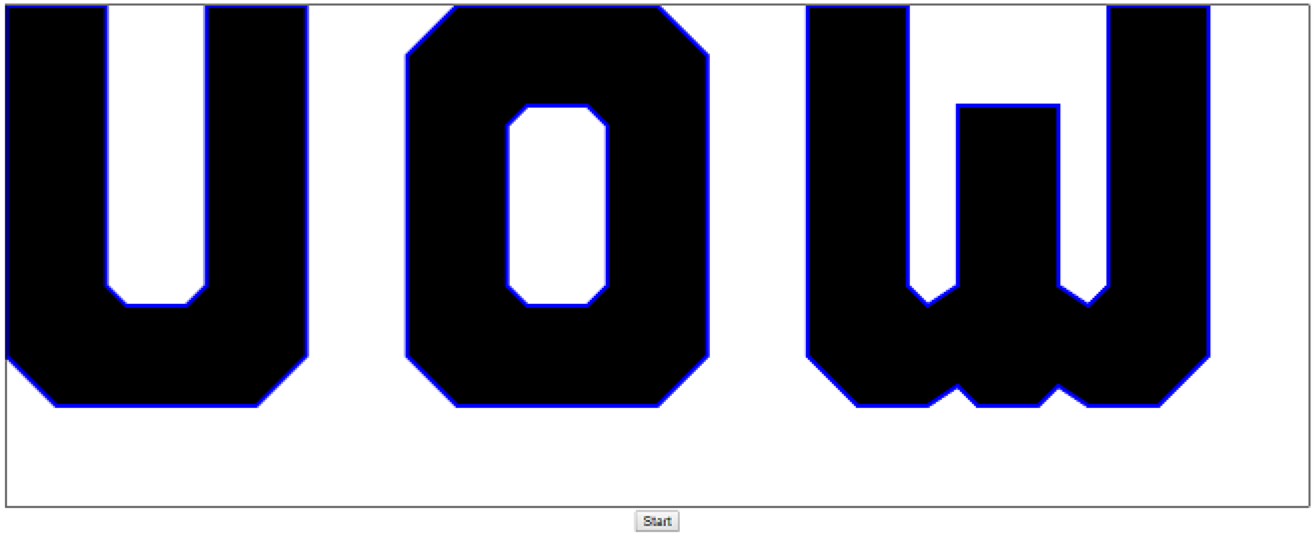
Start

Fill Demo 2

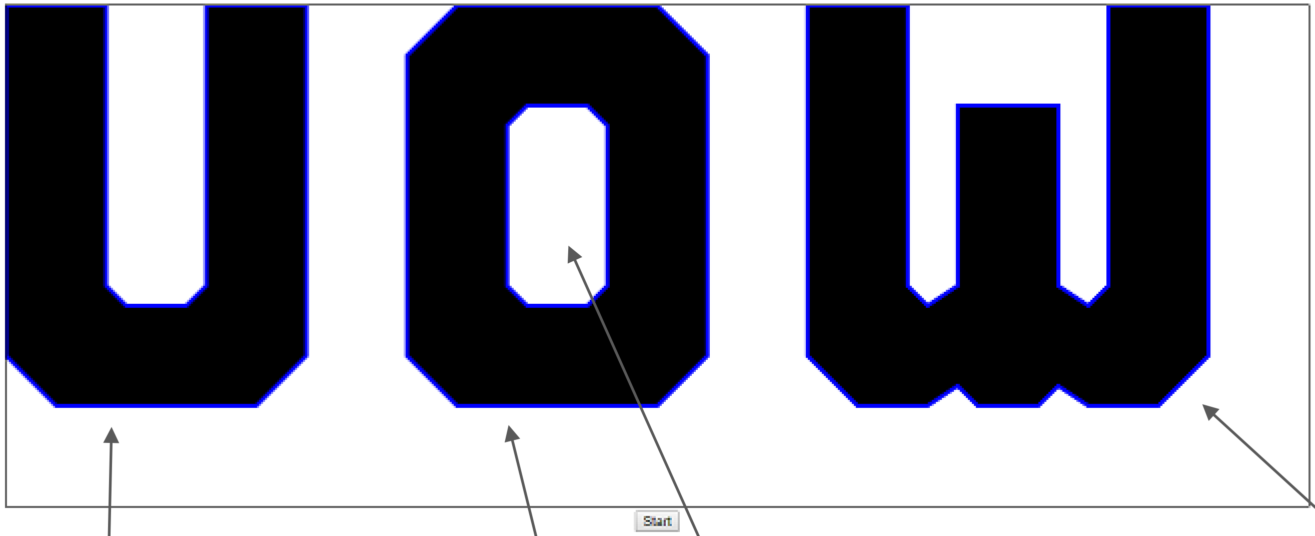


```
// specify the path
context.beginPath();
context.moveTo(50, 10);
context.lineTo(50, 400);
context.lineTo(250, 400);
context.closePath();

// 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 1



UOW 1



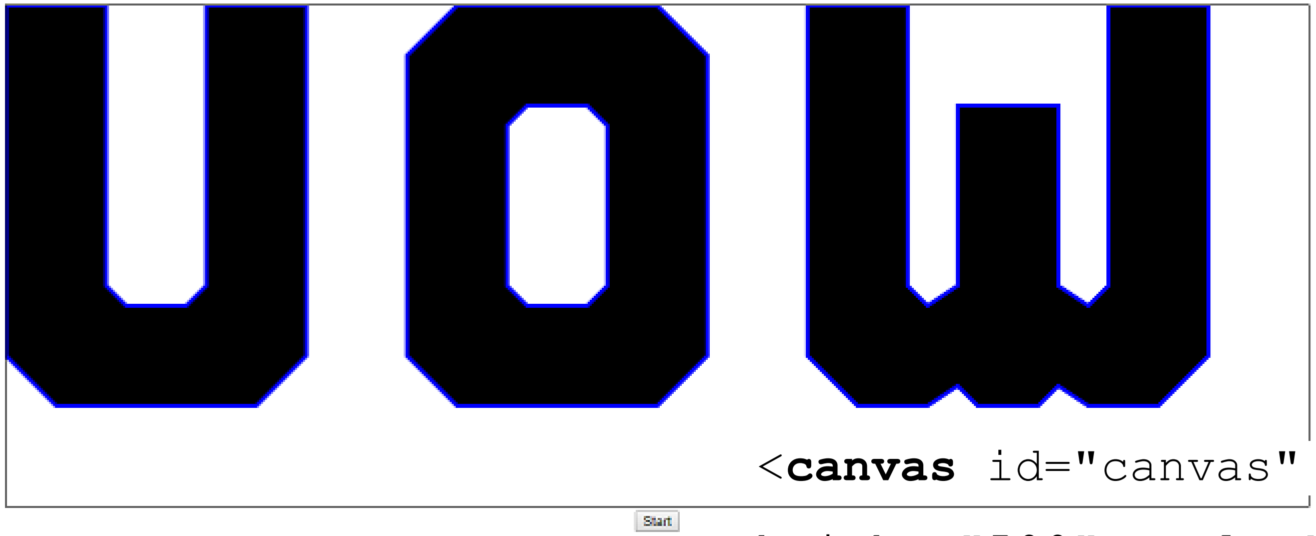
1. letter U filled with black

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

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

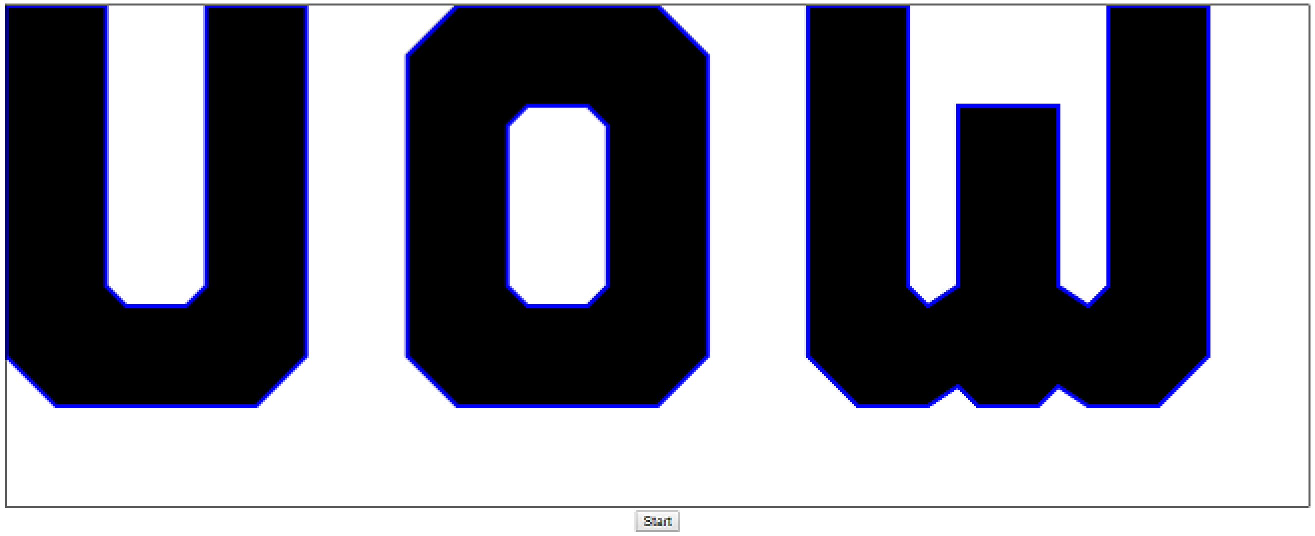
4. letter W filled with black

UOW 1



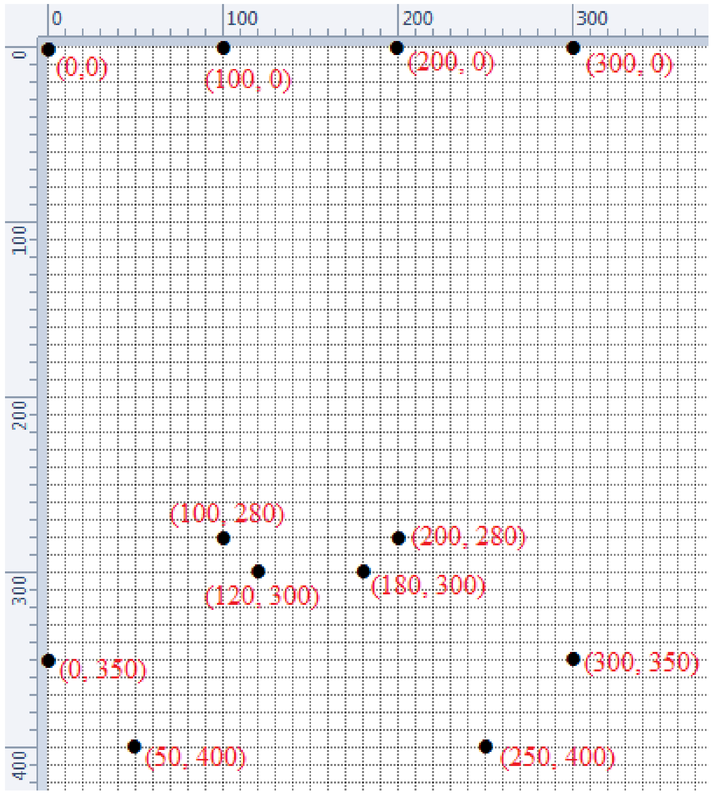
```
<canvas id="canvas" width="1300"
height="500" style="border:1px solid
black;">
Your browser does not support canvas.
</canvas>

<br /><br />

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

UOW 1



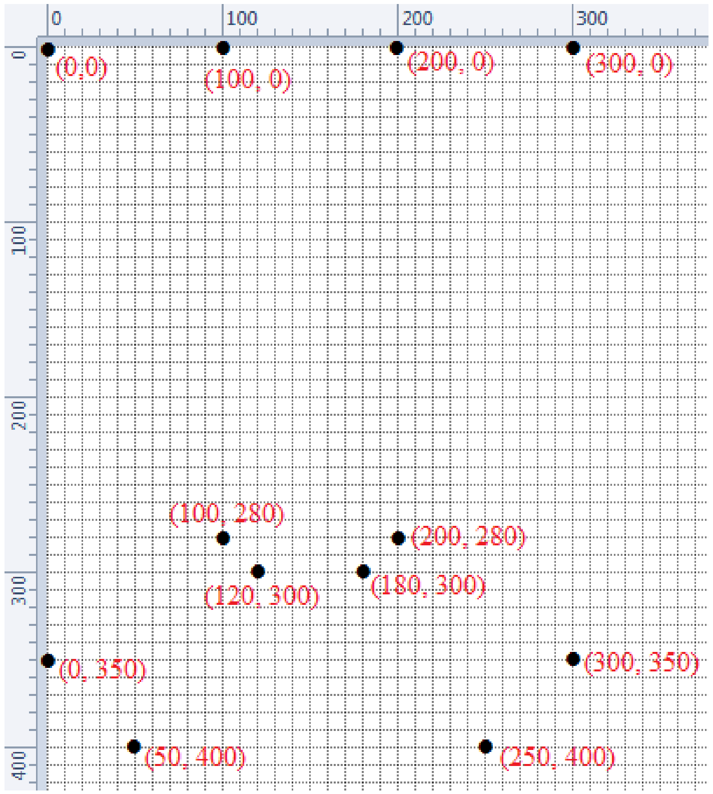
```
function drawUOW(){
  // get the canvas's 2d context
  // letter U
  // letter O (outer)
  // letter O (inner)
  // letter W
}
```

UOW 1



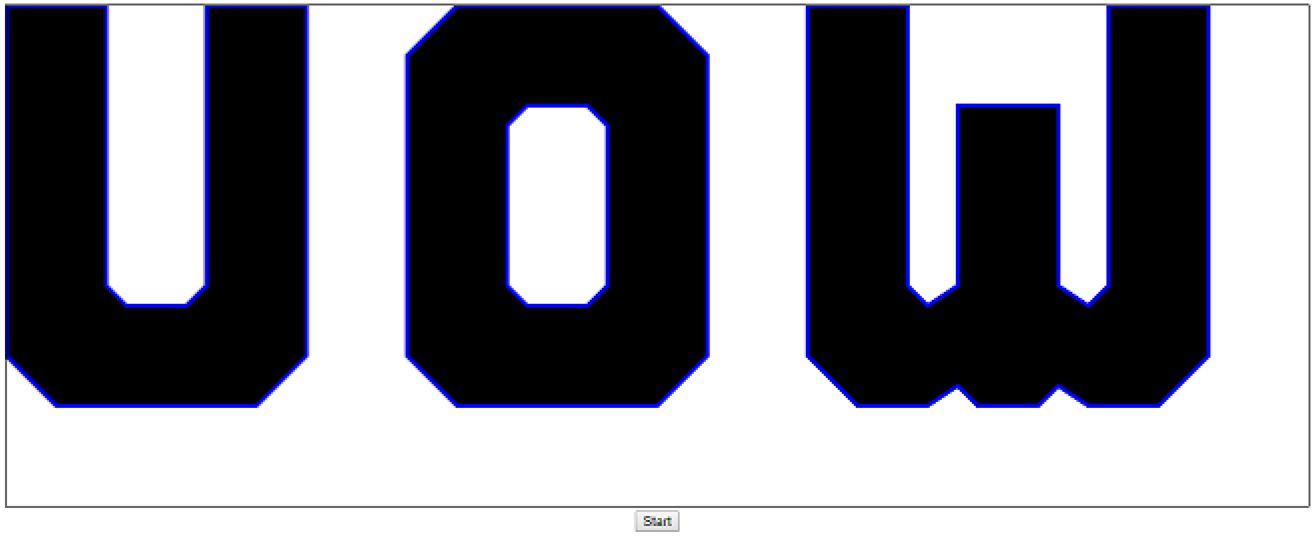
```
// letter U
context.beginPath();
context.moveTo(0, 0);
context.lineTo(0, 350);
context.lineTo(50, 400);
context.lineTo(250, 400);
context.lineTo(300, 350);
context.lineTo(300, 0);
context.lineTo(200, 0);
context.lineTo(200, 280);
context.lineTo(180, 300);
context.lineTo(120, 300);
context.lineTo(100,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

UOW 1

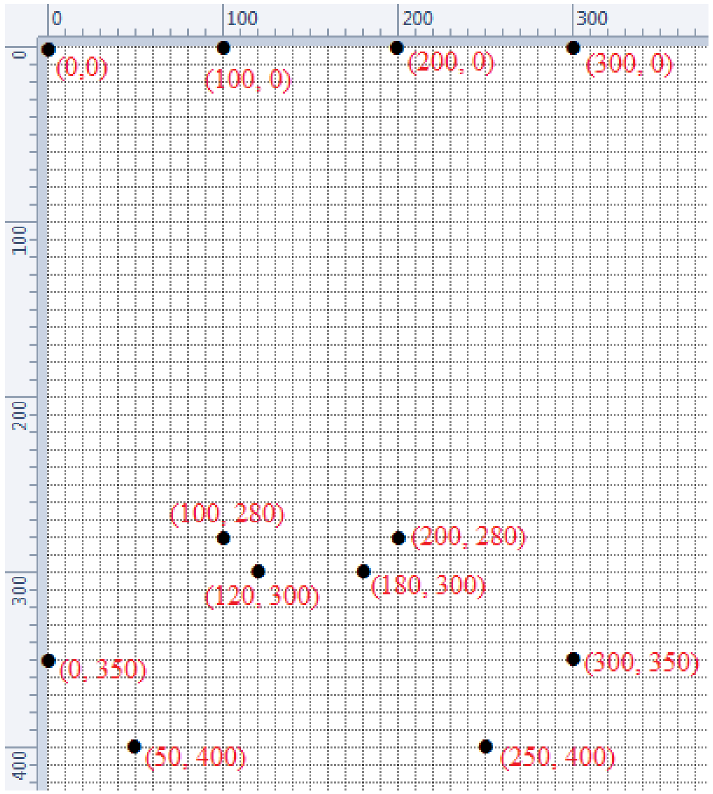


```
// letter U
context.beginPath();
context.moveTo(0, 0);
...
context.lineTo(100, 0);
context.closePath();

context.fillStyle="black";
context.fill();

context.strokeStyle="blue";
context.lineWidth = "4";
context.stroke();
```

UOW 2 - using object



UOW 2



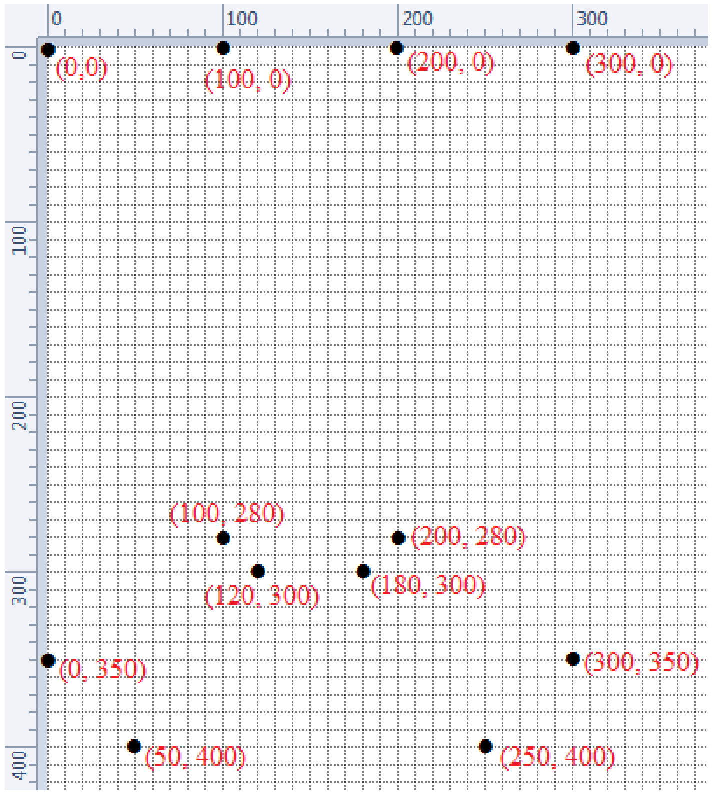
```
// letter U
context.beginPath();
context.moveTo(0, 0);
...
context.lineTo(100, 0);
context.closePath();

context.fillStyle="black";
context.fill();

context.strokeStyle="blue";
context.lineWidth = "4";
context.stroke();
```

Positions: is an array of coordinates

*Using **object** to store the letter setting:*

- *Positions*
- *Fill style*
- *Stroke style*
- *Line width*

UOW 2

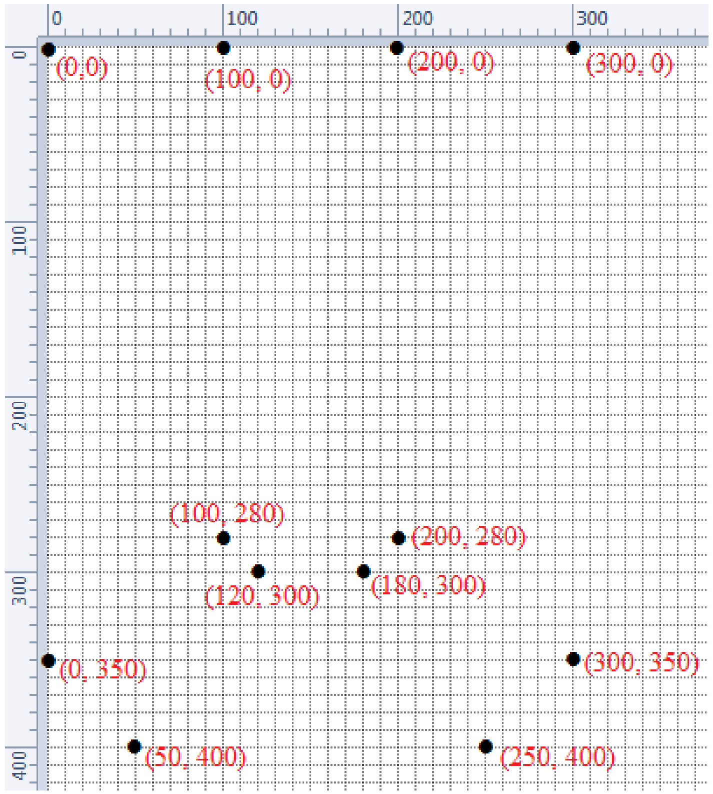


```
// letter U
var letterU = {
  positions: [ [0, 0], …, [100, 0] ],
  fillStyle: "black",
  strokeStyle: "blue",
  lineWidth: "4"
};
```

Using object to store the letter setting:

- *Positions*
- *Fill style*
- *Stroke style*
- *Line width*

UOW 2



```
// letter U
var letterU = {
  positions: [ [0, 0], …, [100, 0] ],
  fillStyle: "black",
  strokeStyle: "blue",
  lineWidth: "4"
};
// letter O outer
// letter O inner
// letter W
```

```
// array of letter settings
var letters = [letterU, letterOouter, letterOinner, letterW];
```

Using object to store the letter setting:

- *Positions*
- *Fill style*
- *Stroke style*
- *Line width*

UOW 2

```
function drawUOW(){
  // objects contains letter's drawing setting
  // letter U object
  // letter O outer object
  // letter O inner object
  // letter W object
  // array of letter settings
  var letters = [letterU, letterOouter, letterOinner, letterW];
  // get the canvas's 2d context
  var canvas = document.getElementById("canvas")
  var context = canvas.getContext("2d");
  // drawing each letter in the array
  for(var i=0; i < letters.length; i++){
    drawLetter(context, letters[i]);
  }
}
```

UOW 2

```
function drawLetter(context, letter){
  // start a new path

  // move to the first position

  // then make a line to other positions

  // finally close the path

  // fill

  // stroke

}
```

UOW 2

```
// letter U
var letterU = {
  positions: [ [0, 0], …, [100, 0] ],
  fillStyle: "black",
  strokeStyle: "blue",
  lineWidth: "4"
};
```

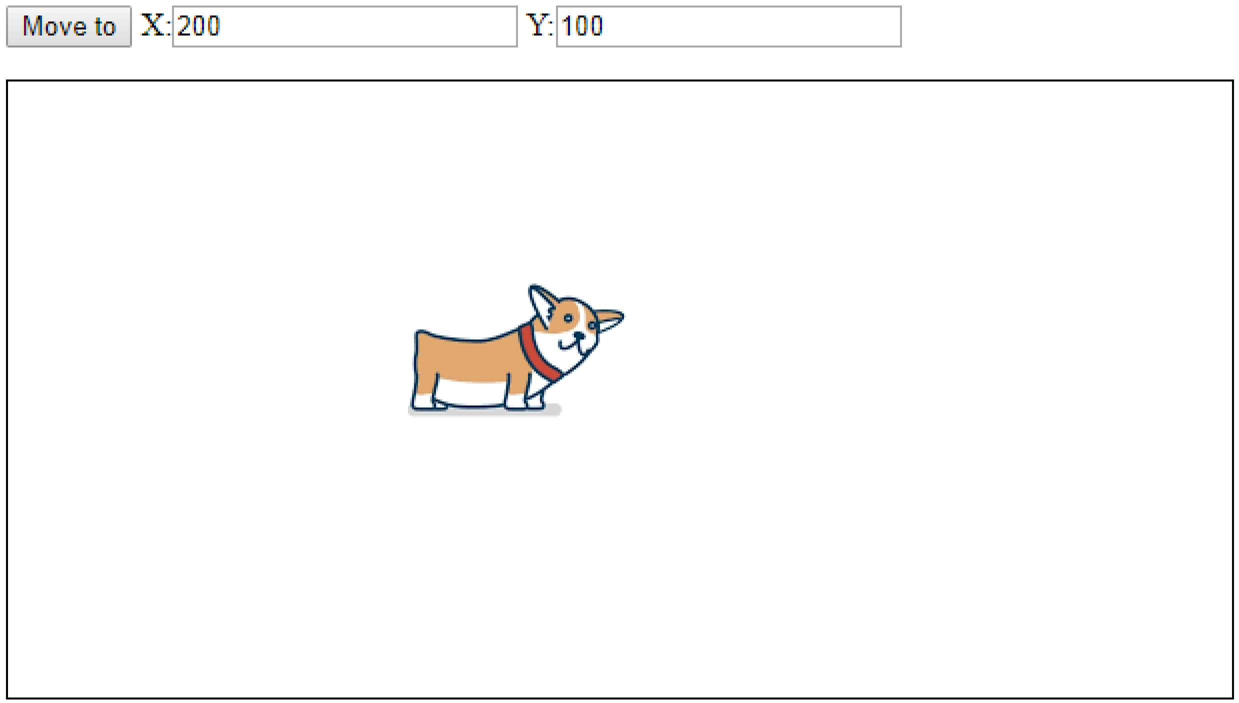
```
// start a new path
context.beginPath();

// move to the first position
var firstPosition = letter.positions[0];

context.moveTo(firstPosition[0], firstPosition[1]);

// then make a line to other positions
for(var j=1; j < letter.positions.length; j++){
  // get the jth position
  var position = letter.positions[j];

  context.lineTo(position[0], position[1]);
}
// finally close the path
context.closePath();
```

UOW 2

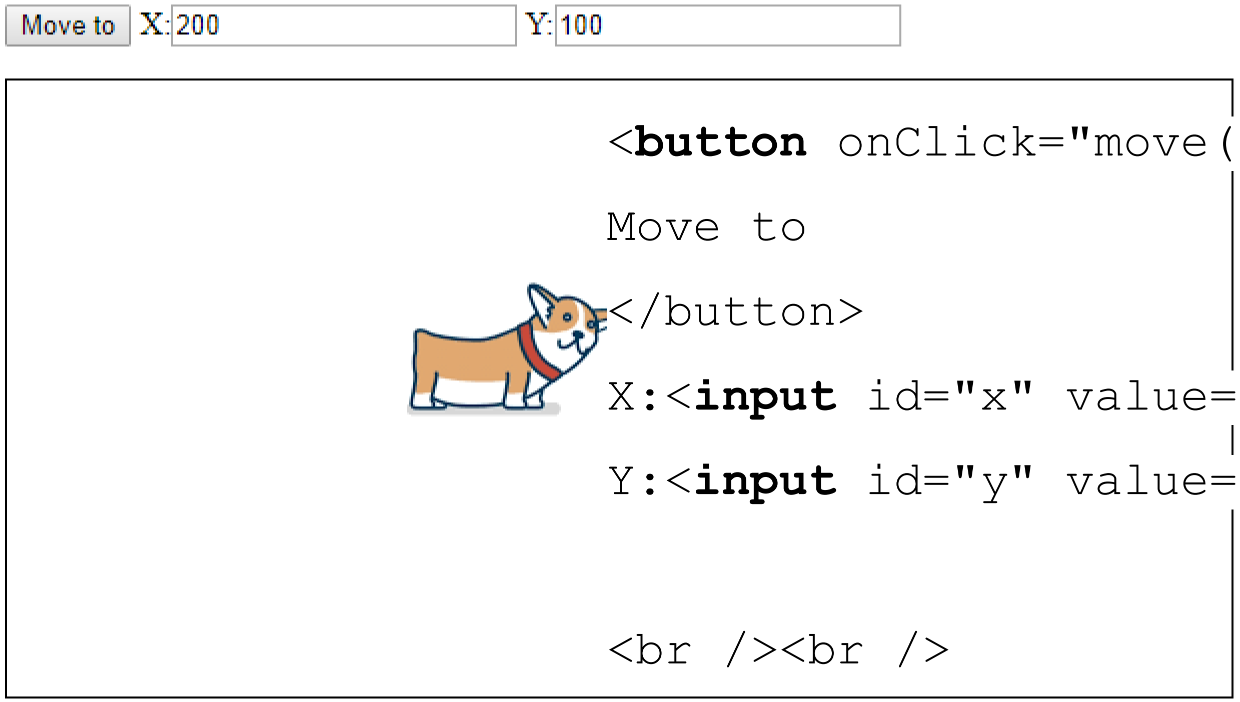
```
// fill
context.fillStyle = letter.fillStyle;
context.fill();


// stroke
context.strokeStyle = letter.strokeStyle;
context.lineWidth = letter.lineWidth;
context.stroke();
```

```
// letter U
var letterU = {
  positions: [ [0, 0], …, [100, 0] ],
  fillStyle: "black",
  strokeStyle: "blue",
  lineWidth: "4"
};
```

Move the Dog

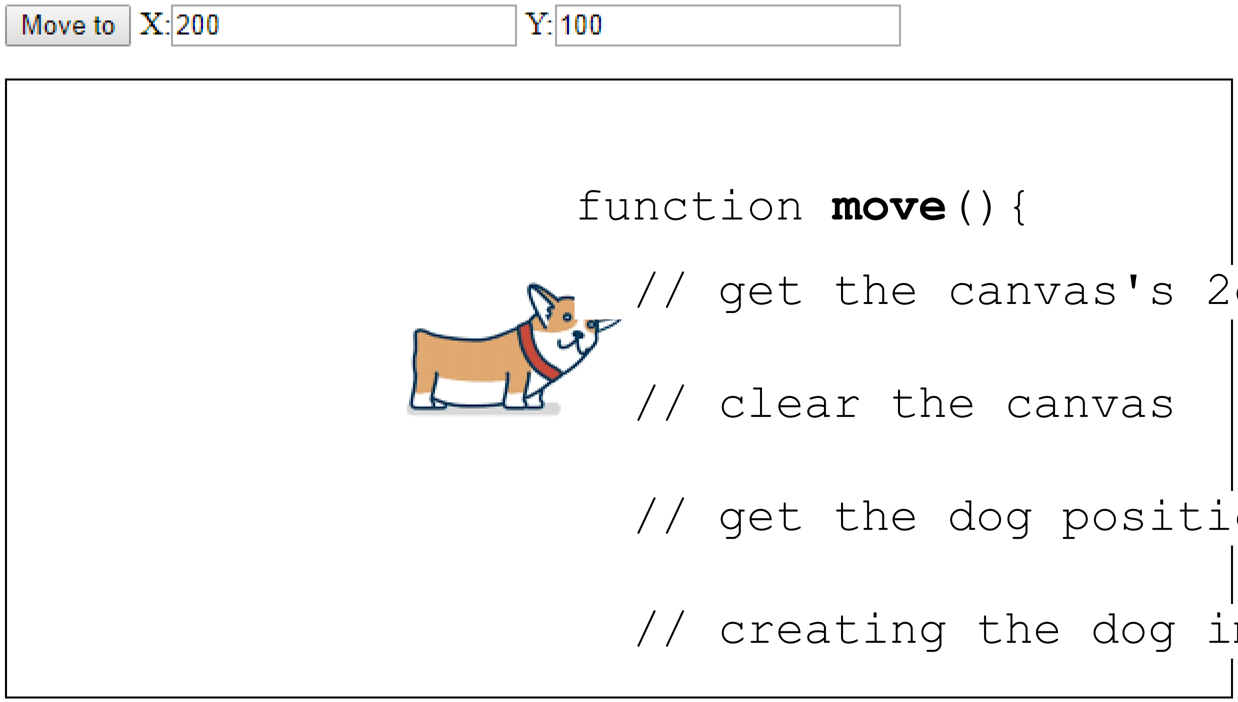


Move the Dog



```
<button onClick="move()">
Move to
</button>
X:<input id="x" value="200"/>
Y:<input id="y" value="100"/>

<br /><br />

<canvas id="canvas" width="800" height="500"
style="border:1px solid black;">
Your browser does not support canvas.
</canvas>
```

Move the Dog



```
function move(){
  // get the canvas's 2d context

  // clear the canvas

  // get the dog position

  // creating the dog image

  // when the image are loaded

  // draw the image at the specified position

}
```

Move the Dog

```
// get the canvas's 2d context
var canvas = document.getElementById("canvas")
var context = canvas.getContext("2d");

// clear the canvas
context.clearRect(0, 0, canvas.width, canvas.height);
```

What would happen if the canvas not cleared

Move the Dog

```
// get the dog position
var x = Number(document.getElementById("x").value);
var y = Number(document.getElementById("y").value);

// creating the dog image
var image = new Image();
image.src = "dog.png";


// when the image are loaded
// draw the image at the specified position
image.onload = function() {
  context.drawImage(image, x, y);
};
```

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>
```

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

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

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

We need to know what object we are dragging

```
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);
}
```

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>
```

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

```
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
}
```

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?*

https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API/Drag_operations#droptargets

*A listener for the **dragEnter** and **dragOver** events are used to indicate **valid drop targets**.*

Most areas of a web page are not valid places to drop data.
Thus, the default handling of these events is not to allow a drop.

*If you want to **allow a drop**, you must **prevent the default handling** by cancelling the event. Calling the preventDefault() method during both a **dragEnter** and **dragOver** event will indicate that a drop is allowed at that location.*

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?*

https://developer.mozilla.org/en-US/docs/Web/Events/dragenter

DragEnter** ***BUG ALERT (found on chrome)*

The target property is broken for this event (dragEnter)
Instead of pointing on "The element underneath the element being dragged."
*it points to itself which explains why people use **dragOver** to **allow the drop**.*

Drag and Drop

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

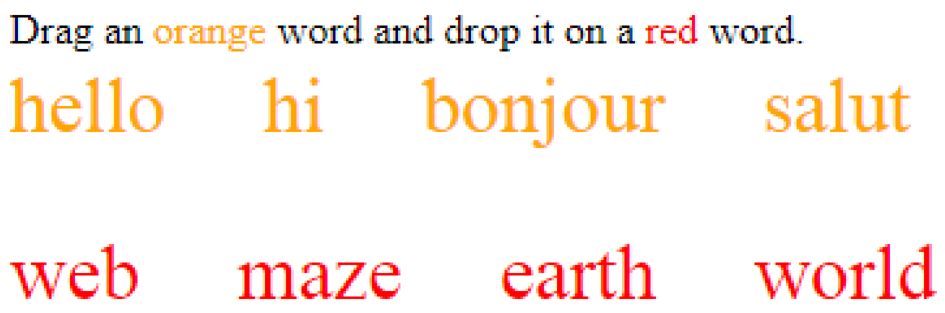
```
<element id="drop-id" onDrop="drop(event)"
onDragOver="dragOver(event)">droppable element</element>
```

*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

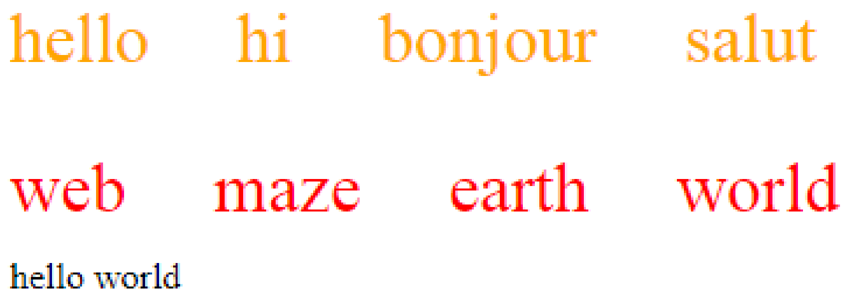


Drag an orange word and drop it on a red word.

hello hi bonjour salut

web maze earth world

When "hello" is dropped on "world", the page displays "hello world".

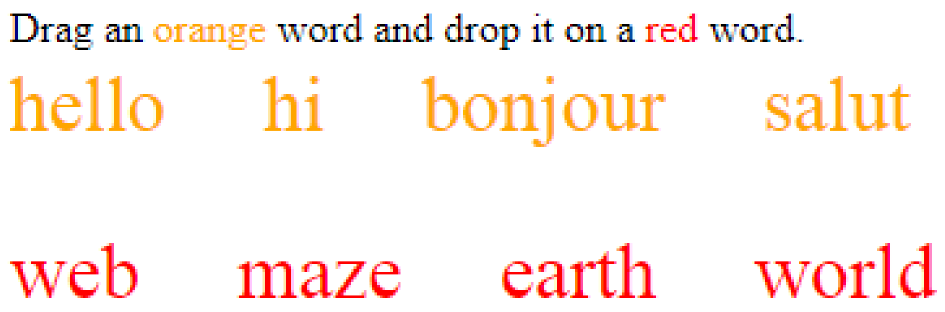


hello hi bonjour salut

web maze earth world

hello world

Drag and Drop: Hello World



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

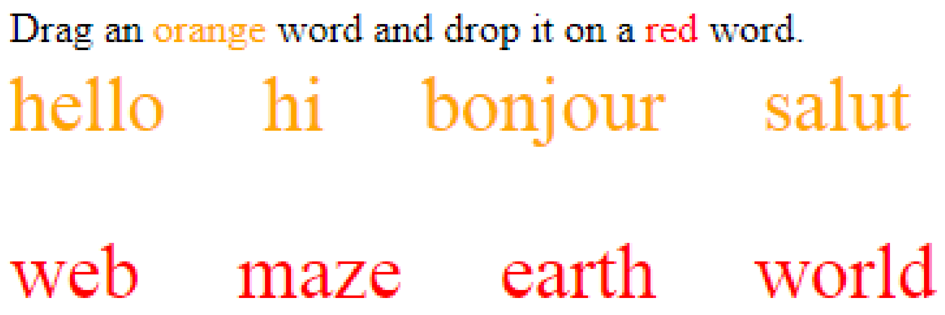
hello hi bonjour salut

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.

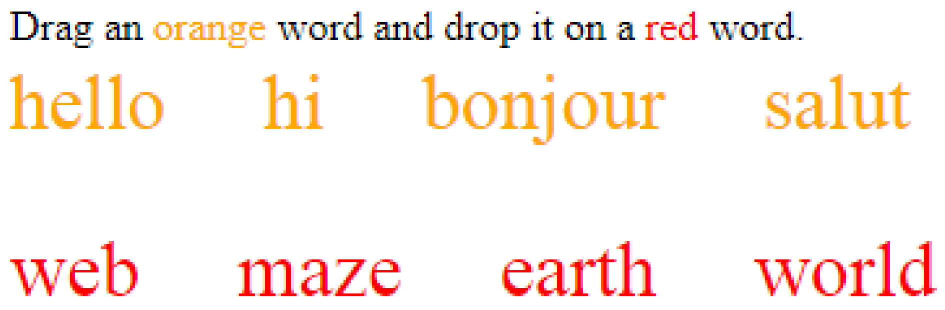
hello hi bonjour salut

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.

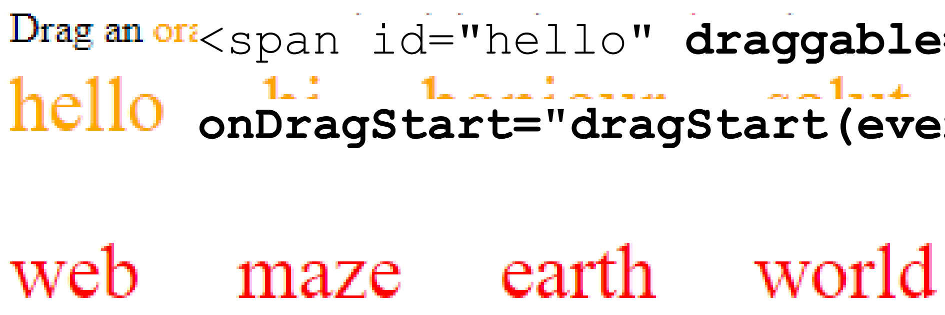
hello hi bonjour salut

web maze earth world

droppable 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

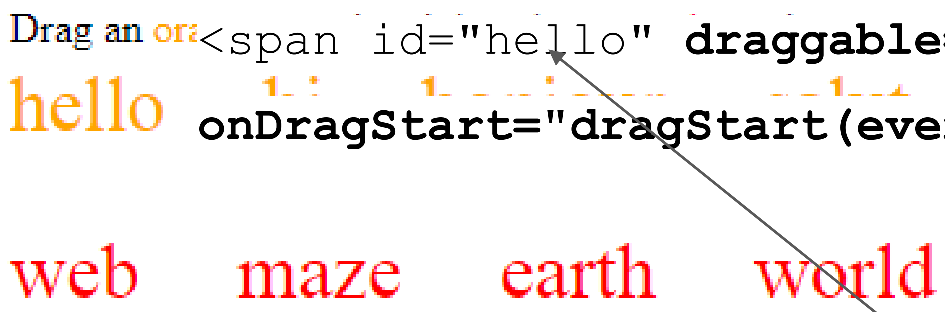


```
<span id="hello" draggable="true"
onDragStart="dragStart(event)" >hello</span>
```

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 ora

hello

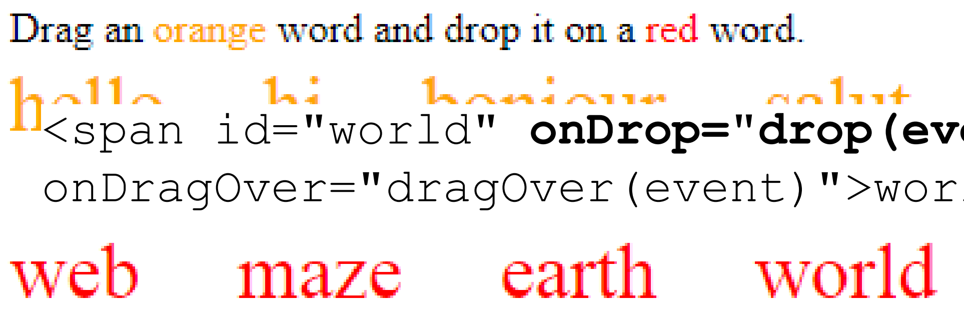
web maze earth world

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

```
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 bonjour salut

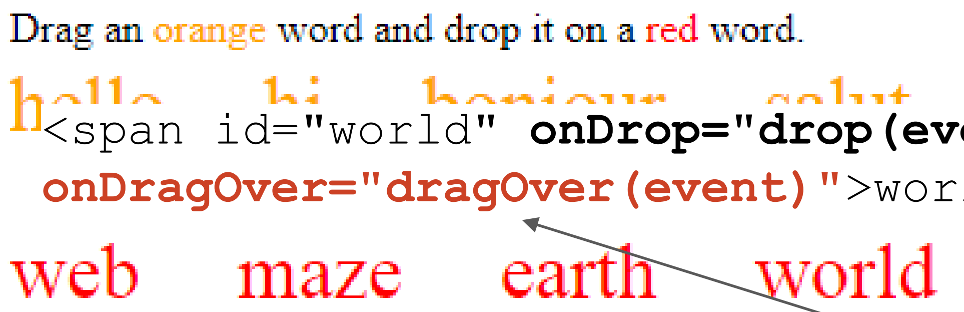
web maze earth world

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

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

```
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;
}
```

Drag and Drop: Hello World



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

hello hi bonjour salut

web maze earth world

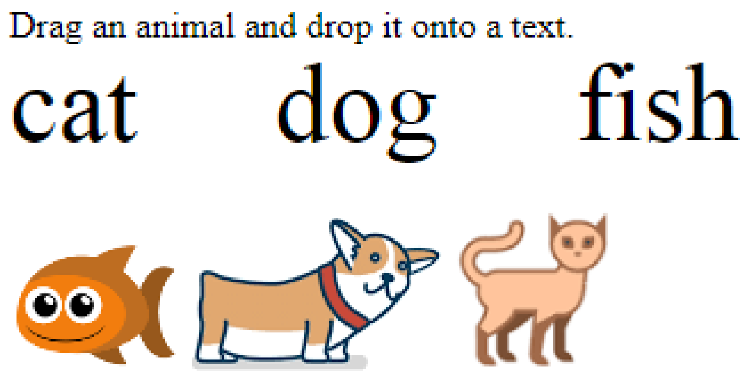
```
<span id="world" onDrop="drop(event)"
onDragOver="dragOver(event)">world</span>
```

*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();

}
```

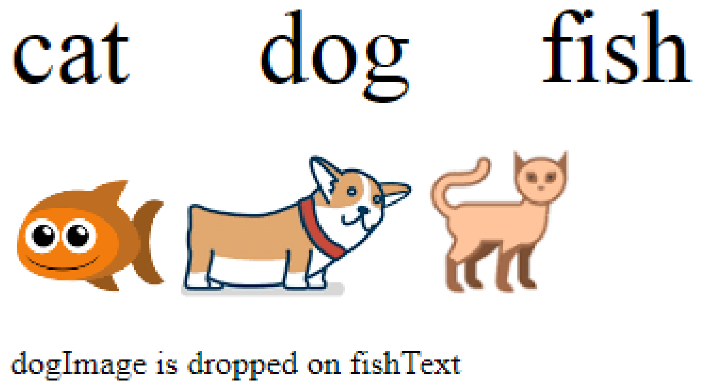
Cat, Dog, and Fish 1



Drag an animal and drop it onto a text.

cat dog fish

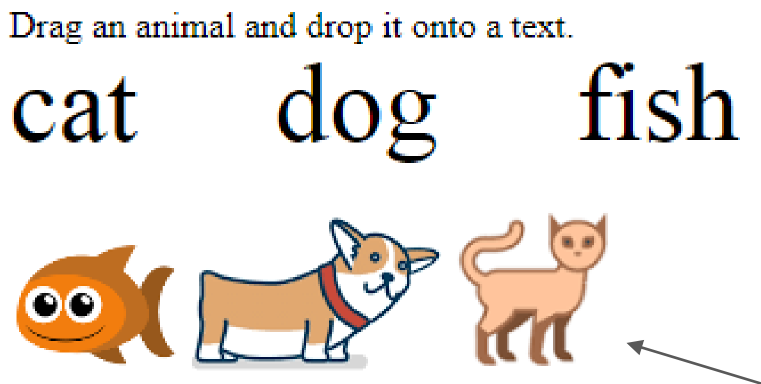
When an animal image is dropped onto a text, a message is displayed.



cat dog fish

dogImage is dropped on fishText

Cat, Dog, and Fish 1



Drag an animal and drop it onto a text.

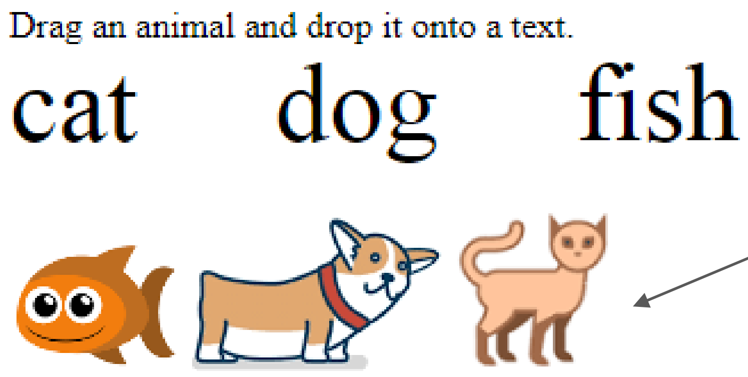
cat dog fish

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

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

Cat, Dog, and Fish 1

DRAGGABLE ELEMENTS

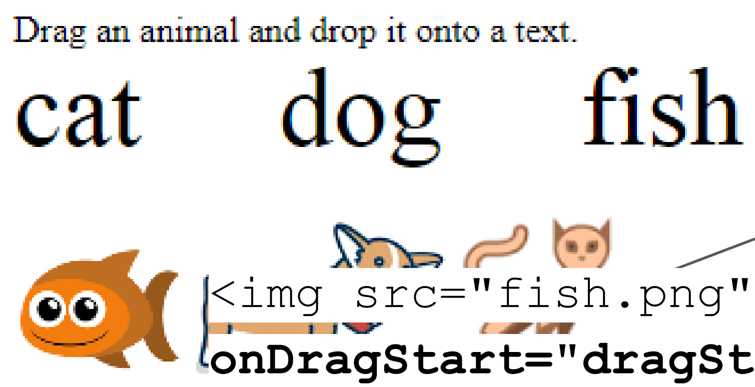

Drag an animal and drop it onto a text.

cat dog fish

```
<img src="fish.png" draggable="true"
onDragStart="dragStart(event)" id="fishImage" />

<img src="dog.png" draggable="true"
onDragStart="dragStart(event)" id="dogImage" />

<img src="cat.png" draggable="true"
onDragStart="dragStart(event)" id="catImage" />
```

Cat, Dog, and Fish 1

DRAGGABLE ELEMENTS



Drag an animal and drop it onto a text.

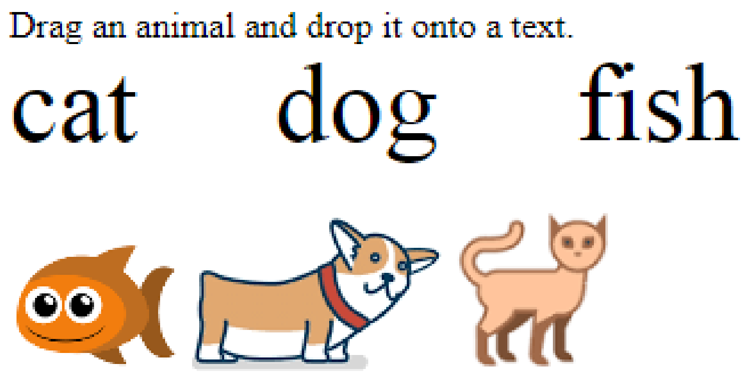
cat dog fish

```
<img src="fish.png" draggable="true"
onDragStart="dragStart(event)" id="fishImage" />
```

```
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);
}
```

Cat, Dog, and Fish 1

DROPPABLE ELEMENTS

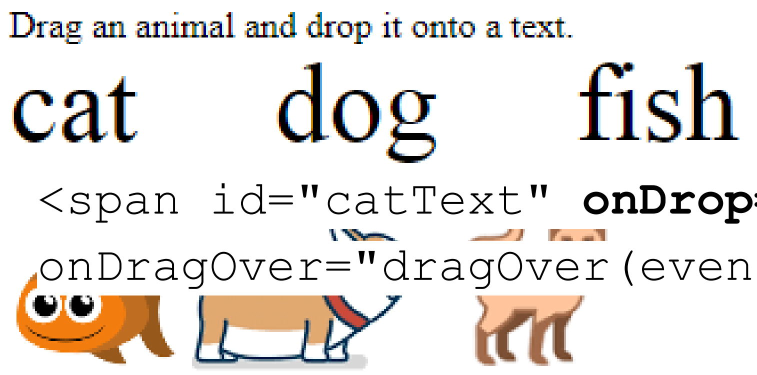

Drag an animal and drop it onto a text.

cat dog fish

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

<span id="dogText" onDrop="drop(event)"
onDragOver="dragOver(event)">dog</span>

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

Cat, Dog, and Fish 1

DROPPABLE ELEMENTS



Drag an animal and drop it onto a text.

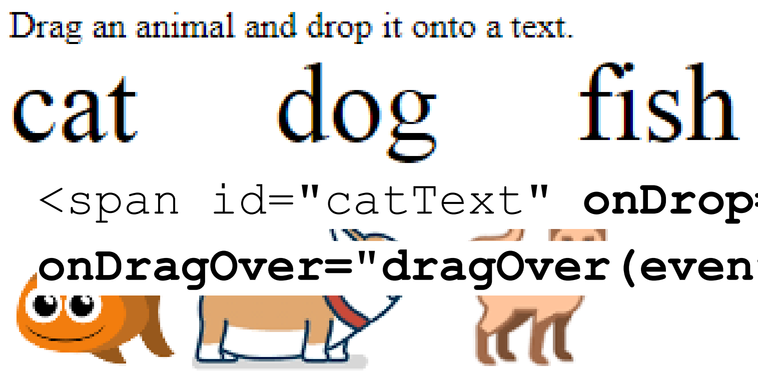
cat dog fish

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

```
function drop(event){

  // get the drop element ID

  var dropId = event.target.id;

  // retrieve the dragged element ID from the dataTransfer objec

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

  // display the message

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

  messageSpan.innerHTML = dragId + " is dropped on " + dropId;

}
```

Cat, Dog, and Fish 1

DROPPABLE ELEMENTS

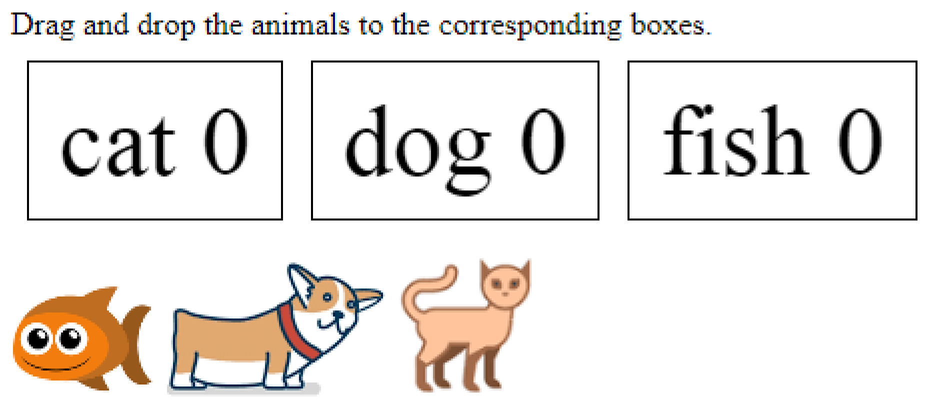

Drag an animal and drop it onto a text.

cat dog fish

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

```
/*
Calling the preventDefault() method during the dragOver event to
indicate that a drop is allowed at that location.
*/
function dragOver(event){
  event.preventDefault();
}
```

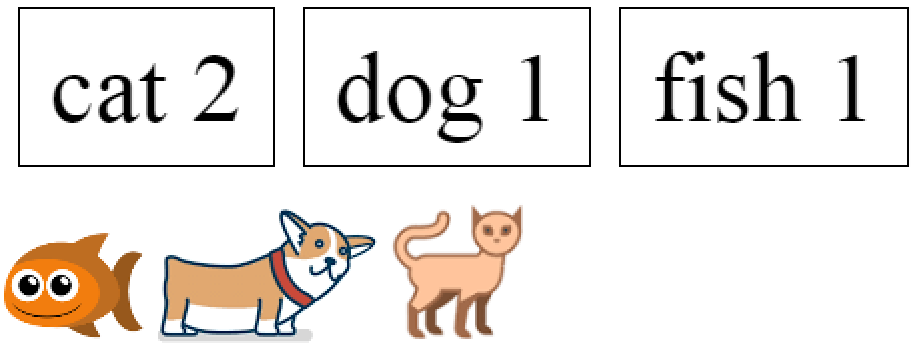
Cat, Dog, and Fish 2



Drag and drop the animals to the corresponding boxes.

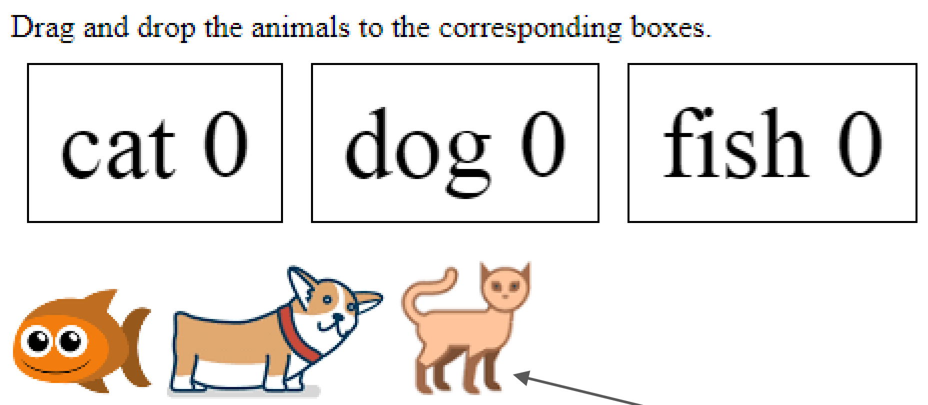
cat 0	dog 0	fish 0

When the animals are dropped into correct boxes, the counters will be increased.



cat 2	dog 1	fish 1

Cat, Dog, and Fish 2



Drag and drop the animals to the corresponding boxes.

cat 0

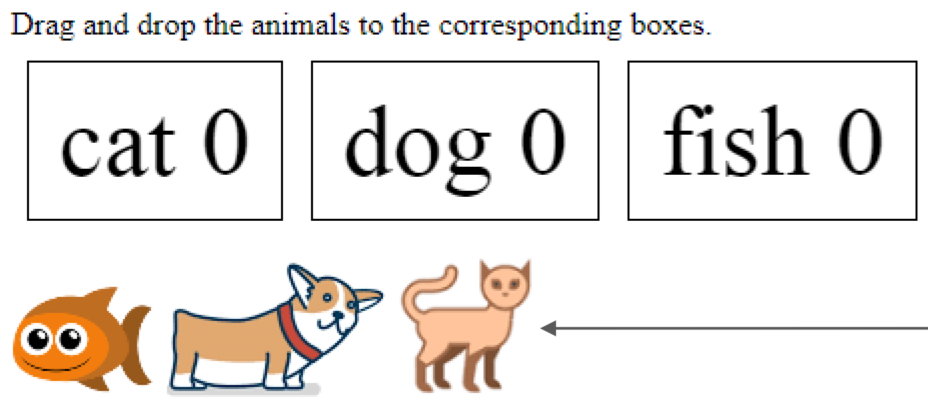
dog 0

fish 0

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

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

Cat, Dog, and Fish 2

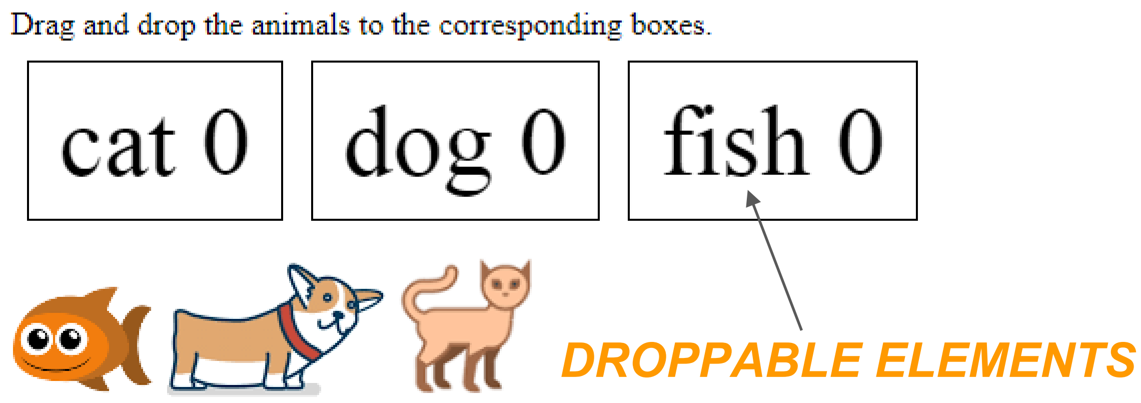

Drag and drop the animals to the corresponding boxes.

cat 0 | dog 0 | fish 0

DRAGGABLE ELEMENTS

```
<img src="fish.png" draggable="true"
onDragStart="dragStart(event)" id="fishImage" />

<img src="dog.png" draggable="true"
onDragStart="dragStart(event)" id="dogImage" />

<img src="cat.png" draggable="true"
onDragStart="dragStart(event)" id="catImage" />
```

Cat, Dog, and Fish 2


Drag and drop the animals to the corresponding boxes.

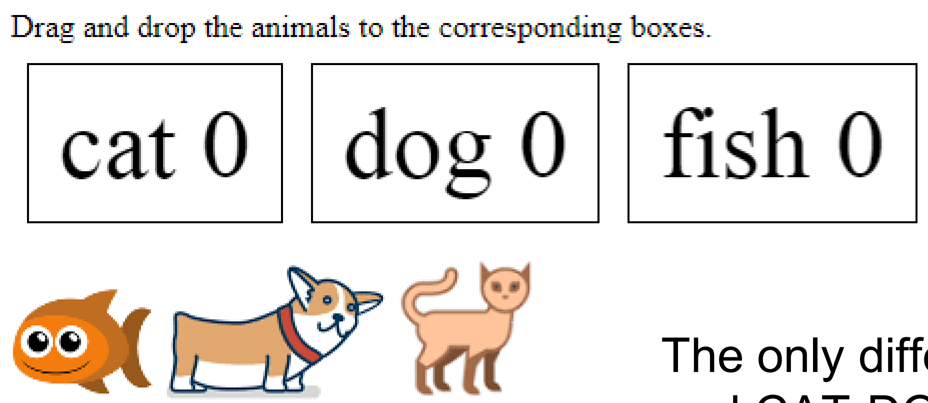
cat 0 | dog 0 | fish 0

```
<style>
#catDiv, #dogDiv, #fishDiv {
  border: 1px solid black;
  display: inline-block;
  font-size: 50px;
  text-align: center;
  text-decoration: none;
  padding: 10px 15px;
  margin-left: 10px;
  margin-top: 20px;
}
</style>
```

```
<div id="catDiv" onDrop="drop(event)" onDragOver="dragOver(event)">
cat <span id="catCount">0</span>
</div>

<div id="dogDiv" onDrop="drop(event)" onDragOver="dragOver(event)">
dog <span id="dogCount">0</span>
</div>

<div id="fishDiv" onDrop="drop(event)" onDragOver="dragOver(event)">
fish <span id="fishCount">0</span>
</div>
```

The only difference between CAT-DOG-FISH (1) and CAT-DOG-FISH (2) is the implementation of the function **`drop(event)`**

Cat, Dog, and Fish 2



Drag and drop the animals to the corresponding boxes.

cat 0 | dog 0 | fish 0

The only difference between CAT-DOG-FISH (1) and CAT-DOG-FISH (2) is the implementation of the function **`drop(event)`**

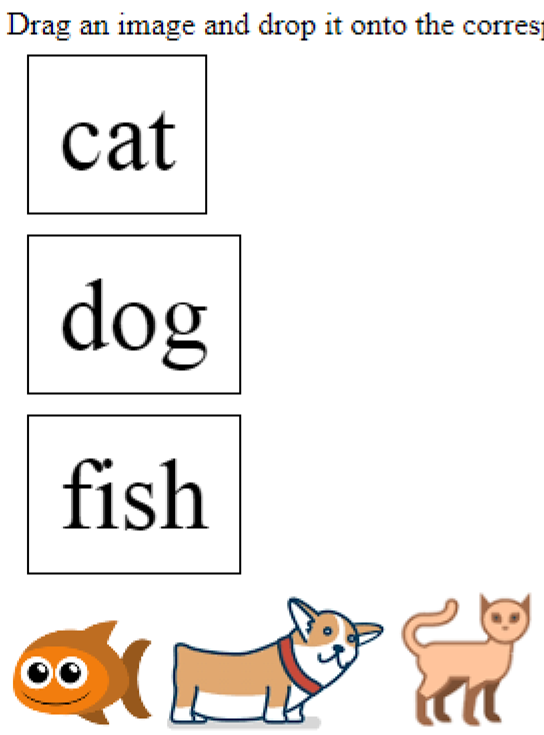
```
var dogCount = 0;
var catCount = 0;
var fishCount = 0;

function drop(event){
  // get the drop element ID

  // retrieve the dragged element ID from the dataTransfer object

  // display the count

}
```

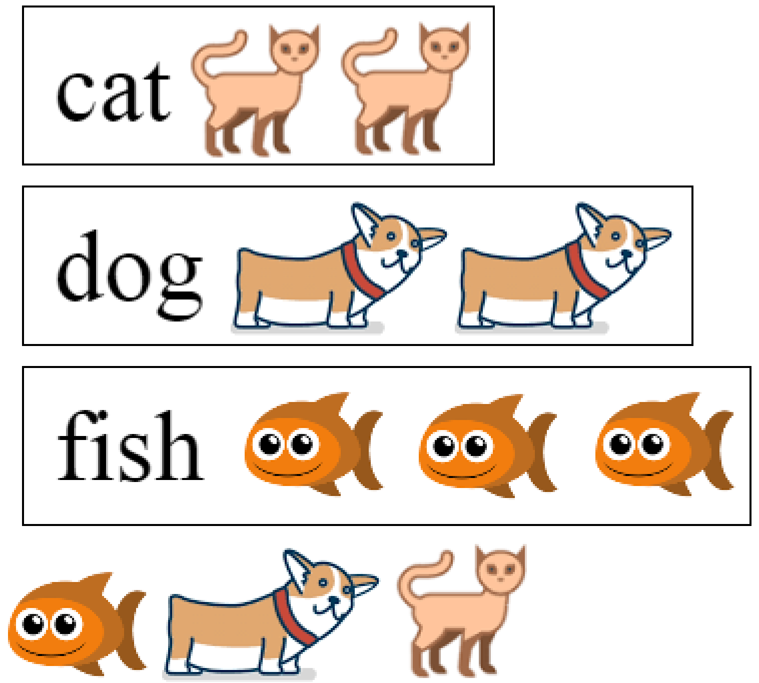
Cat, Dog, and Fish 2

The only difference between CAT-DOG-FISH (1) and CAT-DOG-FISH (2) is the implementation of the function **`drop(event)`**

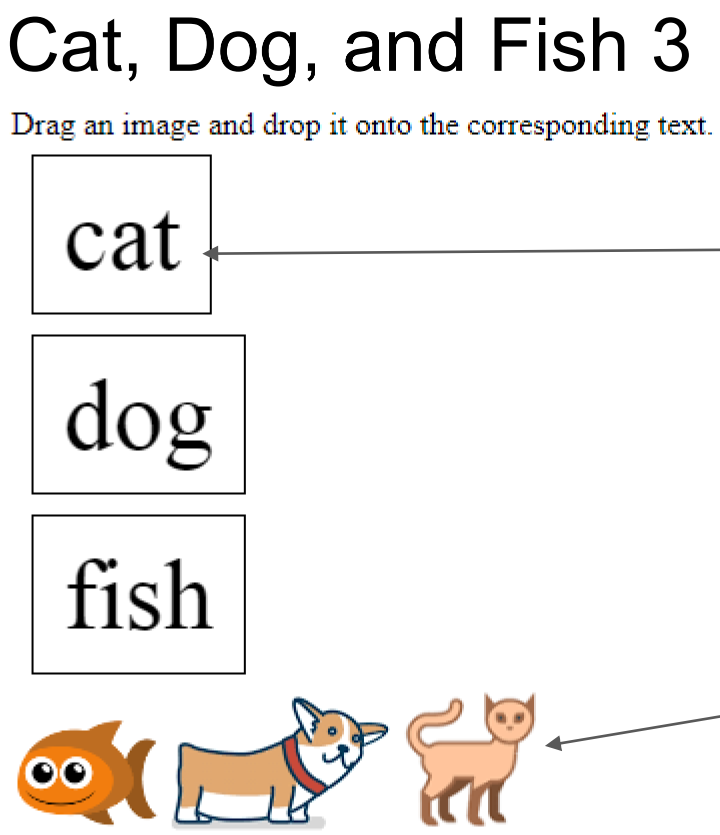
```
// get the drop element ID
var dropId = event.target.id;

// retrieve the dragged element ID from the dataTransfer object
var dragId = event.dataTransfer.getData("dragId");

// display the count

if((dragId == "catImage") && (dropId == "catDiv")){
  catCount = catCount + 1;
  var catCountSpan = document.getElementById("catCount");
  catCountSpan.innerHTML = catCount;
}



if((dragId == "dogImage") && (dropId == "dogDiv")) ...

if((dragId == "fishImage") && (dropId == "fishDiv")) ...
```

Cat, Dog, and Fish 3


Drag an image and drop it onto the corresponding text.



When the animals are dropped into correct boxes, the images are added into the boxes.



Cat, Dog, and Fish 3

Drag an image and drop it onto the corresponding text.

cat

dog

fish

droppable elements*: elements that can be dropped on*

draggable elements*: elements that we can be dragged*

The only difference between CAT-DOG-FISH (2) and CAT-DOG-FISH (3) is the implementation of the function **`drop(event)`**

Cat, Dog, and Fish 3

The only difference between CAT-DOG-FISH (2) and CAT-DOG-FISH (3) is the implementation of the function **drop(event)**

```
function drop(event){
  // get the drop element ID

  // retrieve the dragged element ID from the dataTransfer object

  // if correct drop then create image and put it in the div

}
```

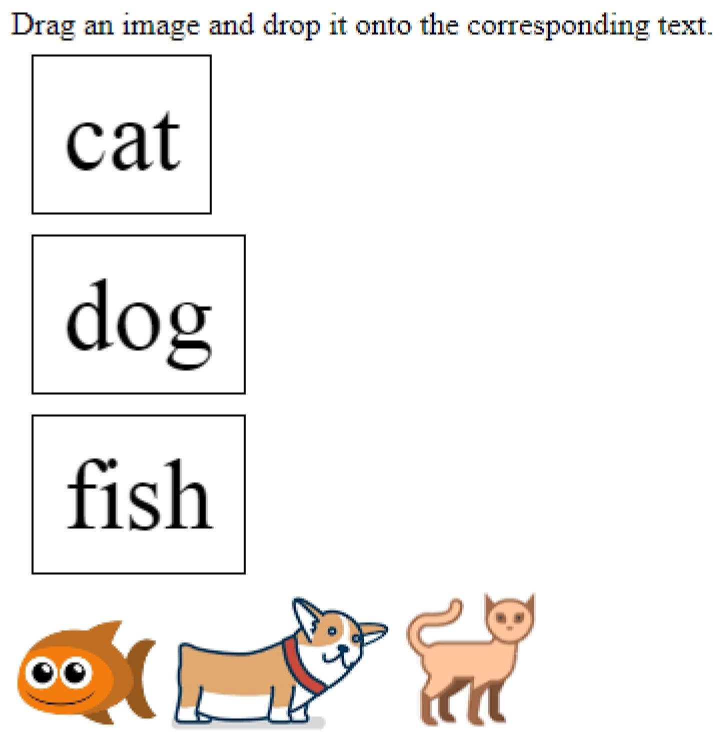
Cat, Dog, and Fish 3

The only difference between CAT-DOG-FISH (2) and CAT-DOG-FISH (3) is the implementation of the 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");

// if correct drop then create image and put it in the div

if((dragId == "catImage") && (dropId == "catDiv")){
  var img = document.createElement("img");
  img.setAttribute("src", "cat.png");

  var catDiv = document.getElementById("catDiv");
  catDiv.appendChild(img);
}

if((dragId == "dogImage") && (dropId == "dogDiv")) ...
if((dragId == "fishImage") && (dropId == "fishDiv")) ...
```

```
<div id="catDiv"
onDrop="drop(even
)"
onDragOver="dragO
er(event)">
cat <span
id="catCount">0</
pan>
</div>
```

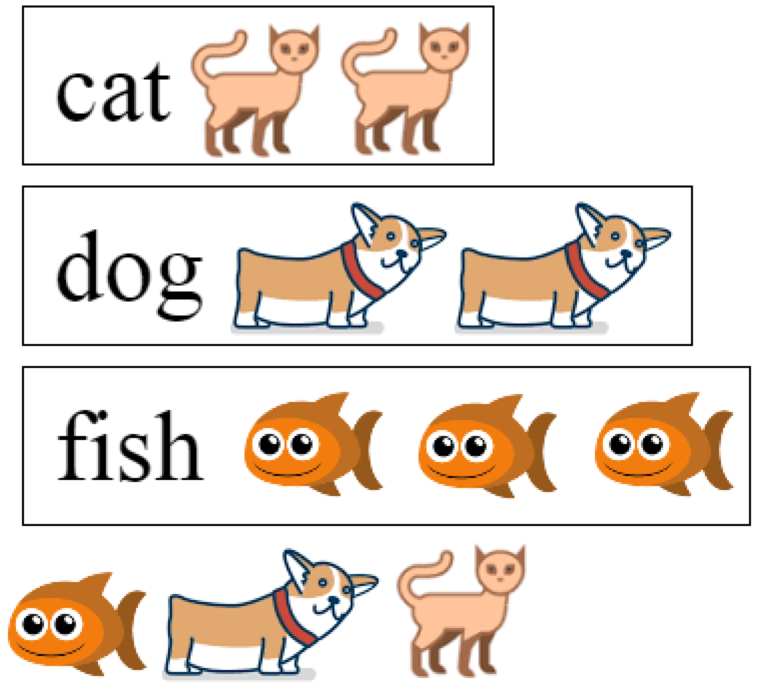
Cat, Dog, and Fish 4


Drag an image and drop it onto the corresponding text.

cat

dog

fish


cat

dog

fish

CAT-DOG-FISH (4) is similar to CAT-DOG-FISH (3), with additional feature: ***click on the animal image in the boxes to make it disappear***.

Cat, Dog, and Fish 4

```
// if correct drop then create image and put it in the div

if((dragId == "catImage") && (dropId == "catDiv")){
  var img = document.createElement("img");
  img.setAttribute("src", "cat.png");

  // when the image is clicked, it will be hidden
  img.addEventListener(
    "click",
    function(){
      img.style.display = "none";
    }
  );

  var catDiv = document.getElementById("catDiv");
  catDiv.appendChild(img);
}

if((dragId == "dogImage") && (dropId == "dogDiv")) ...
if((dragId == "fishImage") && (dropId == "fishDiv")) ...
```

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