CS559 Spring 2019

Lecture 4 (week 3, lecture 1)

February 5, 2019

Canvas and SVG

Michael Gleicher

A Simple Example

```html
<html><body>
  <canvas id="acanvas" height="600px" width="600px"></canvas>
  <script src="myfile.js" type="module"></script>
</body></html>
```

```javascript
window.onload = function () {
  let canvas = document.getElementById("acanvas");
  let context = canvas.getContext("2d");

  context.fillRect(20,20,40,80);
  context.fillStyle = "red";
  context.fillRect(40,60, 40,80);
}
```

```javascript
window.onload = function () {
    let canvas = document.getElementById("mycanvas");
    let context = canvas.getContext("2d");
    function draw() {
        context.clearRect(0,0, canvas.width, canvas.height);
        context.fillStyle = "black"
        context.fillRect(20,20,80,40);
        let xp = performance.now() % canvas.width;
        context.fillStyle = "red";
        context.fillRect(xp,40, 80,40);
        window.requestAnimationFrame(draw);
    }
    draw();
}
```

Where do I draw?

Points (x,y) are interpreted in the **current coordinate system**

```
context.fillRect(40,60,80,50);
```

Canvas coordinates:

- origin at top left
- x to the right in "html pixels"
- y down in "html pixels"

Canvas Coordinates

```
<canvas width="400px" "height=200px"></canvas>
```

(0,0) is top left

`canvas.width,canvas.height` is bottom right

Stroke and Fill

```
context.fillStyle = "yellow";
context.strokeStyle = "goldenrod";

context.fillRect(30,30,30,30);
context.strokeRect(30,30,30,30);
```

Beyond Rectangles: Paths

```
context.beginPath();
context.moveTo(x,y);
context.lineTo(x2,y2);
context.lineTo(x3,y3);
context.fill();
context.stroke();
```

Open, Closed, Disconnected …

```
context.beginPath();
context.moveTo(100,100);
context.lineTo(110,120);
context.lineTo(120,100);
context.closePath();
context.moveTo(150,100);
context.lineTo(160,120);
context.lineTo(170,100);
context.fill();
context.stroke();
```

Save and Restore

```
context.save();
context.fillStyle="red";
context.fillRect(40,40,20,20);
context.restore();
context.fillRect(50,50,20,20);
```

`save` and `restore` capture most (all?) context information

Canvas "Events"

Only the “canvas” is an HTML element

Only the “canvas” gets events

The graphics are represented in code

There is no object to get an event

Click in a rectangle

```
canvas.fillRect(20,20, 60,60);

canvas.onclick = function(event) {
    let mouseX = getXposition(event);
    let mouseY = getYposition(event);
    // check if event is inside of rectangle
    if ( (x>=20) and (x<=60) and (y>=20) and (y<=60)) {
        console.log("rectangle was clicked")
    }
}
```

Remember the rectangle?

```
rects = [];

canvas.fillRect(20,20, 60,60);
rects.push( { x:20, y:20, w:60, h:60} );
```

In immediate mode, the shapes are in the code - not data structures.

Immediate-Mode vs. Retained-Mode

Immediate mode

Drawing commands draw

Nothing is kept around

Retained Mode

Drawing commands create objects

Objects are drawn (when appropriate)

An Example

```html
<html>
<body>
  <canvas width="600px" height="400px" id="acanvas"> </canvas>

  <svg width="600px" height="400px" id="mysvg">
      <rect x="20" y="20" width="40" height="40" fill="red">
      </rect>
  </svg>
</body>
```

SVG

Graphics objects are elements (in the DOM tree)

Graphics objects are just like HTML elements

- handle events
- can be styled
- altered by style sheets (CSS)
- can be accessed by JavaScript

SVG with Style

```
<style>
    .st1 {
        fill:aqua;
    }
</style>

<svg width="600px" height="400px" id="mysvg">
    <rect x="20" y="20" width="40" height="40" class="st1">
    </rect>
</svg>
```

SVG with JavaScript

```
<rect x="20" y="20" width="40" height="40" id="r1"></rect>
```

```
let r = document.getElementById("r1");
r.setAttribute("fill","purple");
```

SVG Events

```
<rect x="20" y="20" width="40" height="40" id="r1"></rect>
```

```
let r = document.getElementById("r1");
r.onmouseenter = function() {
    r.setAttribute("fill","green");
}
r.onmouseleave = function() {
    r.setAttribute("fill","blue");
}
```

Drawing Order (in SVG and Canvas)

Things drawn in order

Things drawn on top of previously drawn things

Painters-Algorithm

Transparency (in SVG and Canvas)

Alpha-Blending

result = (1-α)*old + α*new

Drawing (with Canvas)

```
window.onload = function () {
let canvas = document.getElementById("mycanvas");
let context = canvas.getContext("2d");
canvas.onmousemove = function (event) {
    let box = event.target.getBoundingClientRect();
    let x = event.clientX - box.left;
    let y = event.clientY - box.top;
    context.fillStyle = "#FF00FF7F";
    context.fillRect(x-5,y-5,10,10);
};
```

```
let dots = [];
canvas.onmousemove = function (event) {
  let box = event.target.getBoundingClientRect();
  let x = event.clientX - box.left;
  let y = event.clientY - box.top;
  dots.push( { x:x , y:y });
};
function draw() {
  context.clearRect(0,0, canvas.width, canvas.height);
  context.fillStyle = "#8000F080";
  dots.forEach(dot => context.fillRect(dot.x-5,dot.y-5,5,5));
  dots.forEach(dot => dot.y += 1);
  dots = dots.filter(dot => dot.y < canvas.height);
  window.requestAnimationFrame(draw);
};
draw();
```