INT222 - Internet Fundamentals

Week 11: HTML5 Media

Agenda

- HTML5 Review:
  - Image, audio and video with figure/figcaption
- HTML5 Canvas

Image with figure/figcaption tags

- The HTML <figure> Element represents self-contained content, frequently with a caption (<figcaption>), and is typically referenced as a single unit.
- e.g.

```
<div class="picture">
  <figure>
    <figcaption>
        this is a figure caption blah  blah blah blah blah blah blah
    </figcaption>
    <img src="image-01.jpg" alt="landscape 1" title="landscape 1" />
  </figure>
 </div>
```

- html5figure-1.html
- html5figure-2.html

Audio with figure/figcaption tags

- e.g.

```
<figure>
  <audio controls="controls">
    <source src="Track03.mp3" type="audio/mpeg" />
    <source src="Track03.ogg" type="audio/ogg" />
    Your browser does not support the audio tag used.
  </audio>
  <figcaption>Audio Caption</figcaption>
</figure>
```

- autoplay and loop are additional attributes that can be used with the video tag
- html5_audio.html

Video with figure/figcaption tags

- e.g.

```
<figure>
  <video  controls="controls">
     <source src="movie.mp4"   type="video/mp4"/>
     <source src="movie.ogg"    type="video/ogg" />
     <source src="movie.webm" type="video/webm" />
     Your browser does not support the video tag / type
  </video>
  <figcaption>Video Caption</figcaption>
</figure>
```

- autoplay and loop are additional attributes that can be used with the video tag
  - html5_video.html

HTML5:
The <canvas> Element

<canvas>

- <canvas> - an HTML5 element to give you a drawing space in JavaScript on your Web pages.
- It allows for dynamic, scriptable rendering of 2D shapes and bitmap images.
- It is only a container for graphics. You must use a script to actually draw the graphics.
- Canvas consists of a drawable region defined in HTML code with *height* and *width* attributes.

The <canvas> Element

- Example:

  <canvas id="myCanvas" width="150" height="150"></canvas>
- Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.
- You can have multiple <canvas> elements on one HTML page.
- Don not use CSS for width and height.

By default, the <canvas> element has no border and no content

Fallback content

- Providing alternate content inside the <canvas> element, in case of browsers don't support <canvas>.
  - text description

```
<canvas id="myCanvas" width="200" height="100"
  style="border:10px solid #000000;">
      Your browser does not support the HTML5 canvas tag.
  </canvas>
```

  - static image

```
<canvas id="clock" width="150" height="150">
   <img src="images/clock.png" width="150" height="150" alt=""/>
</canvas>
```

Checking for support in Scripts

- By testing for the presence of the getContext() method.

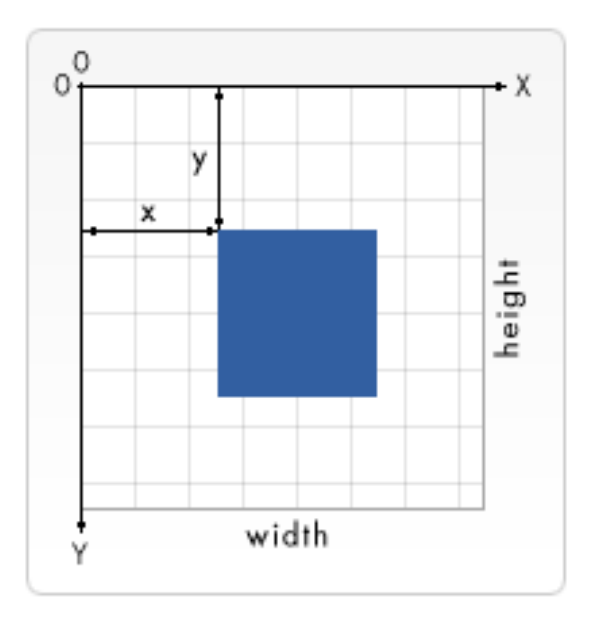
```
var canvas = document.getElementById(‘myCanvas');
if (canvas.getContext){
    var ctx = canvas.getContext('2d');
    // drawing code here
} else {
    // canvas-unsupported code here
}
```

A skeleton template

```
<!DOCTYPE html>
<html>
 <head>
 <title>Canvas Test</title>
  <script type="application/javascript">
    function draw() {
      var canvas = document.getElementById("my-canvas");
      if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.....// drawing code goes here.
      }
    }
  </script>
  <style type="text/css">
                canvas { border: 1px solid black; }
  </style>
 </head>
 <body onload="draw();">
   <canvas id="my-canvas" width="150" height="150"></canvas>
 </body>
</html>
```

Drawing rectangles

- Three functions
  - fillRect(x, y, width, height)
    - Draws a filled rectangle.
  - strokeRect(x, y, width, height)
    - Draws a rectangular outline.
  - clearRect(x, y, width, height)
    - Clears the specified rectangular area, making it fully transparent.



- canvas_test_rect.html

Drawing paths

beginPath()

- Creates a new path. Once created, future drawing commands are directed into the path and used to build the path up.

closePath()

- Closes the path so that future drawing commands are once again directed to the context.

stroke()

- Draws the shape by stroking its outline.

fill()

- Draws a solid shape by filling the path's content area.
- When you call fill(), any open shapes are closed automatically, so you don't have to call closePath().

Moving the Pen & Drawing Lines

moveTo(x, y)

- Moves the pen to the coordinates specified by x and y.

lineTo(x, y)

- Draws a line from the current drawing position to the position specified by x and y.

Drawing a Triangle

```
function draw() {
    var canvas = document.getElementById('canvas');
    if (canvas.getContext){
        var ctx = canvas.getContext('2d');
        ctx.beginPath();
        ctx.moveTo(75,50);
        ctx.lineTo(100,75);
        ctx.lineTo(100,25);
        ctx.fill();
    }
}
```

- canvas_test_tri.html

Drawing Arcs

arc(*x*, *y*, *radius*, *startAngle*, *endAngle*, *anticlockwise*)

- Draws an arc.
- x, y: coordinate
- Start: starting angle (e.g. 0)
- Stop: stopping angle (e.g., 2 * Matho.PI)

➢ To actually draw the circle, use stroke() or fill().

❑ canvas_test_arcs.html

Using images

- One of the more exciting features of <canvas> is the ability to use images.
  - These can be used to do dynamic photo compositing or as backdrops of graphs, for sprites in games, and so forth.
- Drawing images

  drawImage(*image*, *x*, *y*)
  - Draws the CanvasImageSource specified by the image parameter at the coordinates (x, y).

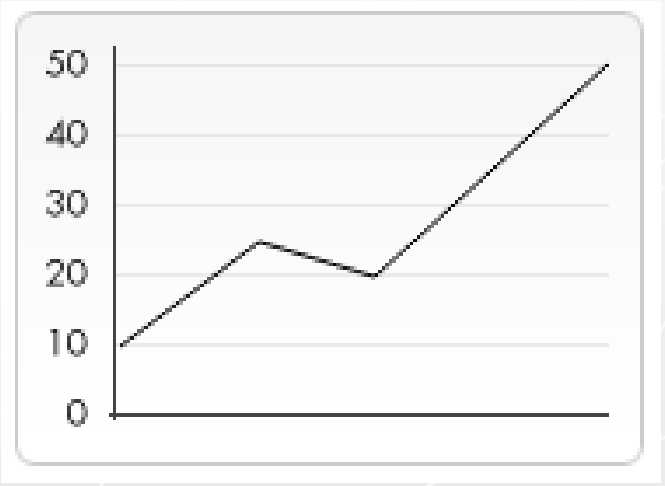
  drawImage(*image*, *x*, *y*, *width*, *height*)
  - This adds the width and height parameters, making the image scalable.

- backdrop.png
- canvas_test_using_img.html

Using images

```
function draw() {
    var canvas = document.getElementById("my-
  canvas");
    if (canvas.getContext) {
      var ctx = canvas.getContext("2d");

      var img = new Image();
     img.onload = function(){
                ctx.drawImage(img,10,10);
                ctx.beginPath();
                ctx.moveTo(40,106);
                ctx.lineTo(80,76);
                ctx.lineTo(113,86);
                ctx.lineTo(180,25);
                ctx.stroke();
      };
      img.src = 'backdrop.png';
    }
 }
```



Filling Text

- To draw text on a canvas, the most important property and methods are:
  - font
    - defines the font properties for text
  - fillText(*text,x,y*)
    - Draws "filled" text on the canvas
  - strokeText(*text,x,y*)
    - Draws text on the canvas (no fill)
- Example

```
ctx.font="30px Arial";
ctx.fillText("Hello World",10,50);
```

  - canvas_test_text.html

Gradients

- Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.
- There are two different types of gradients:

createLinearGradient(*x,y,x1,y1*)

  - Creates a linear gradient

createRadialGradient(*x,y,r,x1,y1,r1*)

  - Creates a radial/circular gradient

- canvas_test_grad.html

Canvas Example

- The Last Canvas Example
  - canvas_test_ball.html

Resourceful Links

- Canvas - Web API Interfaces | MDN
- CSS Media Queries
- CSS Responsive Navigation Menu

Thank You!