INT222 - Internet Fundamentals

Week 11: HTML5 Media

Agenda

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

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.

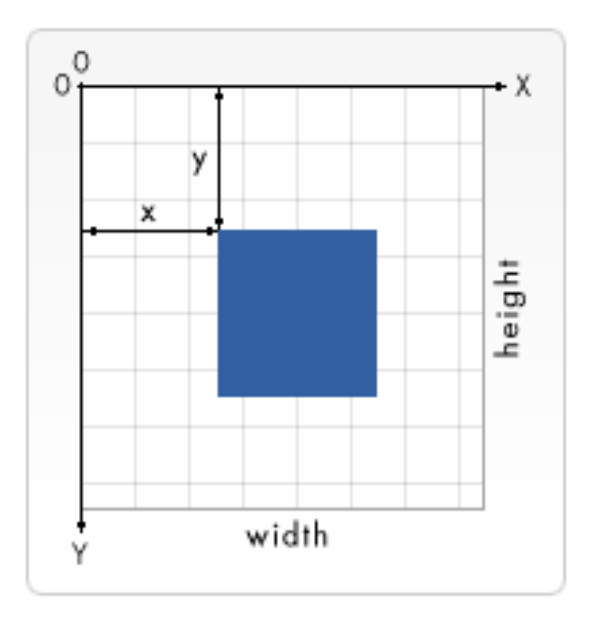
```
var canvas = document.getElementById(‘myCanvas');
if (canvas.getContext){
    var ctx = canvas.getContext('2d');
    // drawing code here
} else {
    // canvas-unsupported code here
}
```

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).

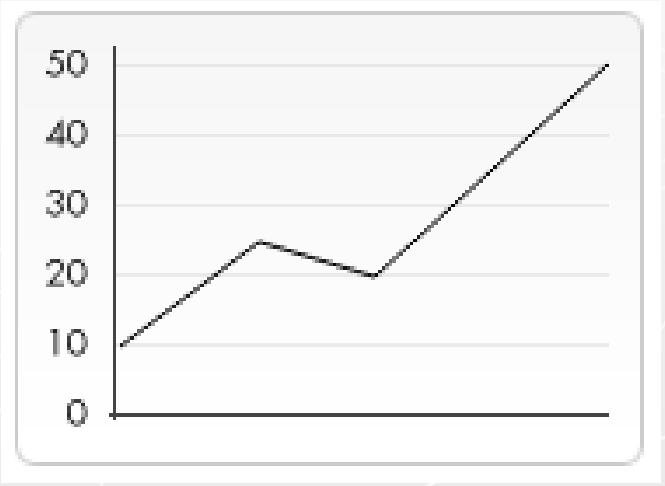
  drawImage(*image*, *x*, *y*, *width*, *height*)
  - This adds the width and height parameters, making the image scalable.

- 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

Image gallery

Creating an Image Gallery

- Wrapping up an image
  - In image galleries, an image is not only a photo or picture,
  - it may have a caption / description, or an action which is performed when the image is clicked.

- Example 1:

```
<div class="img">
  <a target="_blank" href="images/travel-1.jpg">
    <img src="images/travel-1.jpg" alt="Travel">
  </a>
  <div class="desc">Travel</div>
</div>
```

More Examples of Wrapped Images

- Example 2:

```
<div class="img">
  <input type="image" src="images/park-1.jpg" name="park"
         onclick="show_gallery(1); " />
  <div class="desc">Park</div>
</div>
```

- Example 3:

```
<div class=imageRow>
  <img src='images/mednatgeo201201.jpg' alt=''
       onclick='imageView("images/natgeo201201.jpg");'>
  <p>The Matterhorn: Night Clouds #2 -- The Matterhorn, 4478 m, at full
   moon. (&copy; Nenad Saljic/National Geographic Photo Contest)</p>
</div>
```

More Examples of Wrapped Images

- Example 4:

```
<figure>
    <img src='images/tnnatgeo201201.jpg' alt="
        onclick='imageView("images/natgeo201201.jpg");'>
    <figcaption>The Matterhorn: Night Clouds #2 -- The Matterhorn,
      4478 m, at full moon. (&copy; Nenad Saljic/National Geographic
      Photo Contest)</figcaption>
</figure>
```

Creating an Image Gallery (cont')

- Putting all images into a container. For example:

```
<div class=imageGrid>
  …. Wrapped images …
</div>
```

- Setting up CSS for the wrapped images or image boxes. For example:

```
figure
{
  float: left; // for grid galleries
  height: 175px; // size of image boxes
  margin: 1em 2em 0 0;
}
```

Creating an Image Gallery (cont')

- Setting up CSS for images. For example:

```
figure img
{
    padding: 10px;
    border: 1px solid black;
    border-radius: 5px;
    margin: 10px;
    cursor: pointer;
}
```

Creating an Image Gallery (cont')

- Setting up CSS for captions, descriptions. For example:

```
figure figcaption
 {
    width: 200px;
    font-size: 0.7em;
    padding: 5px;
    margin-left: -1000em;
    margin-top: -20px;
    background-color: #ffa;
    border: 1px solid #ffad33;
    border-radius: 5px;
    position: absolute;
 }
```

```
figure:hover  figcaption
{
    margin-left: 50px;
}
```

Creating an Image Gallery (cont')

- Adding two more div elements on the page for showing up full-size images.

```
<div id="popup">
    <!-- large version of the image -->
    <img class="image" id="popupImage" alt="Loading..."
            src="nothing.jpg" />
    <div class="close" onclick="imageClose();"> X 
    </div>
</div>

<div id="popupbg"></div>
```

JS for showing up full-size images

```
function imageView(bigImage) {

  // on the full-size image, set the 'src' attribute that's passed in
  // the '#popupImage' element is an img
  document.querySelector('#popupImage').setAttribute('src',bigImage);

  // show the full-size image
  // the '#popup' element is a div
  document.querySelector('#popup').style.display = 'block';

  // show the faded background image
  // the '#popupbg' element is a div
  document.querySelector('#popupbg').style.visibility = 'visible';
}
```

JS for closing full-size images

```
function imageClose() {
   // hide the full-size image
   document.querySelector('#popup').style.display = 'none';

   // hide the faded background image
   document.querySelector('#popupbg').style.visibility = 'hidden';
}
```

Style rules for the popup full-size image

- Here's a brief explanation of the elements that we need to style:
  - The popup div is not displayed initially, but its other properties are set
  - The image inside the popup div has (for this example) a maximum width setting
  - A 'close' button needs to be added to the top-right corner of the popup div
  - A dark (but translucent) background needs to be placed on the page

CSS rules for the popup full-size image

```
#popup {
    display: none;        /* initial setting*/
    position: fixed;      /* absolute? fixed? they have different results... */
    top: 50px;              /* position from the top edge of the window/viewport */
    left: 50px;             /* position from the left edge of the window/viewport */
    background-color: white;    /* white background... */
    padding: 20px;       /*...and padding together create a 'snapshot' appearance */
    border: 1px solid black;    /* nice crisp border */
    z-index: 1000;  /* this is a big number, we want it on top of the other content */
}

#popup .image {
    max-width: 1000px;          /* limit the width for this code example */
}
```

The rule set for the background

- This effectively enables the popup image to behave like a modal window.

```
#popupbg {
    visibility: hidden;      /* initial setting */
    background: rgba(0,0,0,0.5);/*black background, but translucent(opacity) */
    position: fixed;         /* anchor it to the upper-left of the window/viewport */
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: 999;            /* layer this just below the popup layer */
}
```

- grid-gallery.html

Resourceful Links

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

Thank You!