HTML 5

The New Standard For HTML

mos.bayomi@gmail.com

SVG

Vector-based graphics for the Web

What is SVG?

- Scalable Vector Graphics (SVG) is an XML language for displaying vector graphics.
- SVG graphics do NOT lose any quality if they are zoomed or resized.
- Every element and every attribute in SVG files can be animated.
- SVG is a W3C recommendation.
- Many of the things that can be achieved with **<canvas>** can also be easily achieved with SVG.
- Where <canvas> requires JavaScript, SVG requires markup, much like HTML, and it can be included directly in HTML5

Create SVG

- Create SVG:

```
<svg viewBox="0 0 320 240“ style=“border: 1px solid #999;
     width: 320px; height:240px;">
</svg>
```

- Note that the size of the element on the page is determined by CSS in the style attribute.

- Because SVG is a vector format, you can use viewBox to define a mapping between the physical dimensions of the element, defined in CSS , and the logical coordinates of everything displayed within.

Draw shapes

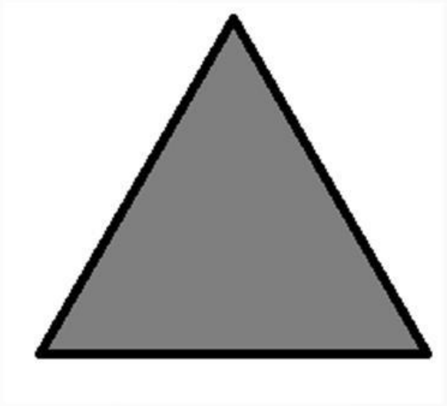
```
<svg id="mysvg" viewBox="0 0 320 240" style="border: 1px
solid #999; width: 320px; height:240px;">

  <rect x="50" y="50" width="100" height="100“
             style="fill: rgb(255,0,0)">
  </rect>
   <line x1="50" y1="50" x2="150" y2="150"
          style="stroke: rgb(0,127,127); stroke-width: 5;">
   </line>
   <circle cx="165" cy="100" r="50"
          style="fill:rgb(0,255,0);">
   </circle>
</svg>
```

Draw polygons.

- A polygon is slightly more complex; it has an attribute points that you use to supply a space-separated list of x,y coordinates.

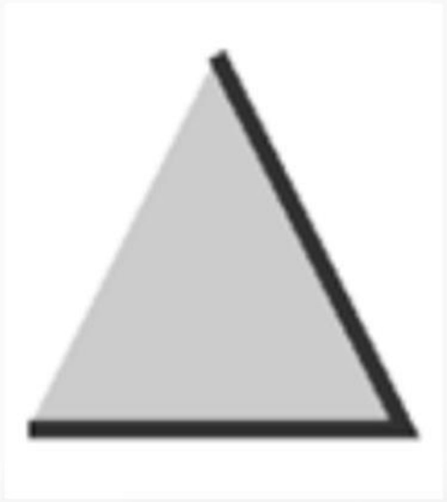
```
<polygon points="265,50  315,150  215,150"
  style="stroke: rgb(51,51,51); fill: rgb(204,204,204);
  stroke-width: 5;">
</polygon>
```



Draw polygons (cont'd).

- With the polygon element you don't have to provide the starting point again; it assumes the shape is closed, and the path drawn returns to the first point.

- If you want to draw an open shape, you can use the `<polyline>` element instead; it uses an identical points attribute but doesn't close the path around the shape.



Applying styles to SVG

- The previous example used an inline style to apply colors and stroke thicknesses.
- Those properties can also be applied directly to the elements.

```
<rect x="50" y="50" width="100" height="100"
      fill="rgb(255,0,0)"></rect>

<line x1="50" y1="50" x2="150" y2="150"
      stroke="rgb(0,127,127)" stroke-width="5"></line>
```

Applying styles to SVG (cont'd)

- But you can alternatively leave off the style and inline attributes and use this in your CSS file, and achieve the same results:

```
rect {
      fill: rgb(255,0,0);
}

line {
      stroke: rgb(0,127,127);
      stroke-width: 5;
}
```

- It looks much like any other CSS ,but with some unusual properties.

Images in SVG

- Images are easy to embed within your SVG drawing. The syntax is similar to that of HTML, and the only additional information you need to provide over the <image> element are the coordinates of the upper-left corner:

```
<image x="10" y="10"
       width="236" height="260"
       xlink:href="example.png">
</image>
```

- You use an `xlink:href` to link to the image.

Text in SVG

- Text is handled a little differently in SVG compared to
- HTML. In HTML, any text within the body is rendered to the screen ,no special wrapping is required.
- In SVG, text has to be explicitly wrapped within a containing element.

```
<text x="10" y="20">
    Hi everybody , everything is OK!
</text>
```

Many things in SVG

- Transforms, gradients, patterns, and declarative animation
  - SVG is a huge topic, worthy of a book by itself, and we've barely scratched the surface so far.

HTML 5 AUDIO

No Plugin?? -- Yes -- no plugin

HTML 5 Audio

- HTML5 provides tags that quickly convey an understanding of their content.
- Media elements, such as music and video that were previously only embedded, can now be identified more precisely.

- **The `<audio>` Element:**
  - The `<audio>` tag identifies sound content, such as music or any other audio streams.
  - The `<audio>` tag has attributes that control **what**, **when**, and **how** audio will be played. The attributes are *src*, *preload*, *controls*, *loop*, and *autoplay*.

```
<audio src="MyFirstMusic.ogg" controls autoplay loop>
    Your browser does not support the audio tag.
</audio>
```

The HTML5 <source> Element

- **The <source> Element:**
  - The <video> and <audio> tags can contain the *<source>* tag, which defines multimedia resources for <video> and <audio>tags.
  - With this element, you specify alternative video and audio files from which the browser can then choose based on its media type or codec support.

```
<audio controls>
    <source src="MyFirstAudio.ogg" />
    <source src=" MyFirstAudio.mp3" />
        <p>Your browser does not support the audio tag</p>
</audio>
```

Audio Formats and Browser Support

Currently, there are 3 supported file formats for the `<audio>` element: MP3, Wav, and Ogg:

Browser	MP3	Wav	Ogg
Internet Explorer 9	YES	NO	NO
Firefox 4.0	NO	YES	YES
Google Chrome 6	YES	YES	YES
Apple Safari 5	YES	YES	NO
Opera 10.6	NO	YES	YES

HTML 5 Audio APIs

- If the built-in controls do not suit the layout of your user interface, or if you need to control the media element using calculations or behaviors that are not exposed in the default controls.

- There are built-in JavaScript ***functions*** and ***attributes*** to help you.

Audio APIs *(Common control functions)*

1. `load()`: Loads the media file and prepares it for playback. Normally does not need to be called unless the element itself is dynamically created.

2. `play()`: Loads (if necessary) and plays the media file. Plays from the beginning unless the media is already paused at another position.
3. `pause()`: Pauses playback if currently active.

4. `canPlayType(type)`: Tests to see whether the video element can play a hypothetical file of the given MIME type.

Audio APIs *(Read-only media attributes)*

1. `Duration` : The duration of the full media clip, in seconds. If the full duration is not known, NaN is returned.

2. `Paused:` Returns true if the media clip is currently paused. Defaults to true if the clip has not started playing.

3. `ended` : Returns true if the media clip has finished playing.

4. `startTime` : Returns the earliest possible value for playback start time. This will usually be 0.0 unless the media clip is streamed and earlier content has left the buffer.
5. `error` : An error code, if an error has occurred.
6. `currentSrc` : Returns the string representing the filethat is currently being displayed or loaded.

Audio APIs *(Scriptable attribute values)*

1. `autoplay` : Sets the media clip to play upon creation or query.
2. `loop` : Returns trueif the clip will restart upon ending.
3. `currentTime` : Returns the current time in seconds that has elapsed since the beginning of the playback.
4. `controls` : Shows or hides the user controls, or queries whether they are currently visible.
5. `volume` : Sets the audio volume to a relative value between 0.0 and 1.0.
6. `muted` : Mutes or unmutes the audio, or determines the current mute state.

VIDEO

Problems, Problems, and Solutions

HTML5 video

- Displaying videos in HTML is not easy!

- You must add a lot of tricks to make sure your video will play in all browsers (*Internet Explorer, Chrome, Firefox, Safari, Opera*) and on all hardware (*PC, Mac , iPad, iPhone*).

The `<embed>` Element

- The purpose of the `<embed>` tag is to embed multimedia elements in HTML pages.
- The following HTML fragment displays a Flash video embedded in a web page:

```
<embed src="intro.swf" height="200" width="200">
```

- **Problems:**
  - If the browser does not support Flash, the video will not play
  - iPad and iPhone **do not** support Flash videos
  - If you convert the video to another format, it will still not play in all browsers

The `<object>` Element

- The purpose of the `<object>` tag is to embed multimedia elements in HTML pages.
- The following HTML fragment displays a Flash video embedded in a web page:

```
<object data="intro.swf" height="200" width="200">
```

- **Problems:**
  - If the browser does not support Flash, the video will not play
  - iPad and iPhone do not support Flash videos
  - If you convert the video to another format, it will still not play in all browsers

The HTML5 <video> Element

- **The `<video>` Element:**
  - allows you to broadcast video clips or streaming visual media.
  - It has all the attributes of the `<audio>` tag plus three more: *`poster`*, *`width`*, and *`height`*.
  - The ***`poster`*** attribute lets you identify an *image* to be used while the video is **loading** or in the case when the video won't load at all.

```
<video src="MyFirstMovie.ogg" controls="controls">
    Your browser does not support the video tag
</video>
```

- **Problems:**
  - You must convert your videos to many different formats.
  - The `<video>` element does not work in older browsers.

The Best HTML Solution

- The example below uses 4 different video formats.
- The HTML 5 `<video>` element tries to play the video either in MP4, OGG, or WEBM format.
- If this fails, the code "falls back" to try the <object> element. If this also fails, it "falls back" to the <embed> element:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.webm" type="video/webm">
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240">
  </object>
</video>
```

HTML 5 Video APIs *(Additional video attributes)*

1. `poster` : The URL of an image file used to represent the video content before it has loaded.

2. `width, height` : Read or set the visual display.

3. `videoWidth, videoHeight` : Return the intrinsic or natural width and height of the video. They cannot be set.