Introduction to HTML5
Lesson 6
Vector Graphics and SVG Files
HTML5 - Lesson 6

Objectives

- Animation with JavaScript
  - Tracking values with variables
  - With setInterval
  - With setTimeout
  - With requestAnimationFrame method

- SVG
  - Discuss raster vs vector images
  - Inkscape’s use in making SVGs
  - SVG code
  - Animation and Beyond
Intro to SVG and Canvas Animations

The **HTML5 canvas element** is used everywhere.

Web developers are taking advantage of the benefits of what the canvas element has to offer. It allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a procedural model that updates a bitmap and does not have a built-in scene graph at a low level.

<canvas>

SVG (**Scalable Vector Graphics**) - An image format for vector graphics such as:

- Adobe Illustrator
- Inkscape
- Scribus
- other software that create vector images
Animation with JavaScript

Animating with JavaScript is more complicated than using CSS transitions or animations but as difficult as it is, it provides much more power to the developer.

A number of DOM elements are placed within the page and elements are moved at a given interval or frame rate using `setInterval()` or `setTimeout()`.

`setInterval()` - Method calls function over and over again at specific time intervals while evaluating an expression.

```html
<!DOCTYPE html>
<html>
<body>
<p>Start the clock:</p>
<p id="timer"></p>
<script>
var myVar=setInterval(function(){myTimer()},1000);
function myTimer() {
    var d = new Date();
    document.getElementById("timer").innerHTML = d.toLocaleTimeString();
}
</script>
</body>
</html>
```

This code displays the time every second

1000 milliseconds = 1 second
Animation with JavaScript

setTimeout() - calls a function after a specific number of milliseconds. If you need to repeat the call to the function, you use setInterval() instead.

When the buttons is pushed, an alert will pop up in 5 seconds

```html
<!DOCTYPE html>
<html>
<body>
<button onclick="setTimeout(function(){alert('Hello')},5000);">
Try it</button>

</body>
</html>
```
Intro to SVG and Canvas Animations

Animation with JavaScript

```html
<!DOCTYPE html>
<style>
div { position: absolute; left: 10px; padding: 50px; background: crimson; color: white; }
</style>
<script>
var requestId = 0;

function animate(time) {
    document.getElementById("animated").style.left = (time - animationStartTime) % 2000 / 4 + "px";
    requestId = window.requestAnimationFrame(animate);
}

function start() {
    animationStartTime = window.performance.now();
    requestId = window.requestAnimationFrame(animate);
}

function stop() {
    if (requestId)
        window.cancelAnimationFrame(requestId);
    requestId = 0;
}
</script>

<button onclick="start()">Click me to start!</button>
<button onclick="stop()">Click me to stop!</button>
<div id="animated">Hello there.</div>
```

requestAnimationFrame() - Used to perform a task with animation by requesting the browser to call an update.
Intro to SVG and Canvas Animations

Benefits of using SVG formats:

● Small file sizes compress well
● Without losing quality, SVG scales to any size
● Have design control like interactivity and filters using XML code
● keeping everything working well together

SVG developments continue to advance. For CSS backgrounds and inline integration, browser vendors are adding support as well as implementing mobile engines, animations, transforms and filters.

Canvas will always impose limitations and is evolving but remains a self-contained programmable bit-mapped image element.
Intro to SVG and Canvas Animations

Vector graphics

A vector based graphic is a group of paths made up of the basic geometric shapes that are formed using points, lines and curves. They produce the most clear and precise graphic as they use mathematically-defined areas to produce any shape, line and curves.

Advantages - scalable, resolution independent, no background, cartoon-like, reducing HTTP Request, styling and scripting through CSS3, can manipulate graphic via JavaScript, can be animated and edited, smaller file size, metafiles contain both raster and vector data, no limit on output size, can be scaled to any size with no quality loss

Disadvantages - inappropriate for photo-realistic images, More complicated to create and edit, requires advanced knowledge and software. can be costly and time consuming.

Popular File Formats:

- SVG
- EPS
- AI
- PDF
- XAML
- CDR
- WMF/EMF
- VML
- .blend
Intro to SVG and Canvas Animations

Raster images

A raster image aka bitmap graphics are made of many tiny squares of color information and are referred to as pixels or dots arranged to display an image. They are resolution dependent and are measured in dot-per-inch (dpi).

Advantages:

- Easy to use
- Can be edited using paint
- Best for photos

Disadvantages:

- Blowing image size up creates degradation in quality
- Large dimensions and details equal large file sizes
- Slower to display and print

Popular File Formats:

- JPG
- PSD
- PNG
- TIFF
- GIF
- BMP

For a complete list of image file formats visit: http://en.wikipedia.org/wiki/Image_file_formats
Intro to SVG and Canvas Animations

Examples of Raster images

A vector graphic consists of located points (nodes), lines (connected series of points), and areas (closed, connected series of points, also called polygons). Attribute information can be attached to a point, line, or area. A line representing a road could include attributes such as name, width, surface, etc.

The example below shows the difference between a vector image and a raster image.
In this course we will be using Inkscape, an open-source vector graphics editor similar to Adobe Illustrator, Corel Draw, Freehand, or Xara X.

Inkscape is used to make Scalable Vector Graphics (SVG), an open XML-based W3C standard. Vector designing is a method mostly used for the creation of:

- Logos
- Illustrations
- Art

They all require high scalability however editing photos cannot be done with this type of program.
Intro to SVG and Canvas Animations

Terms

Object = a vector line or shape created with the Pen or Preset Shape tool.

Path = all of the contours that make up an object. A path goes in one direction from 'start' to 'end'. Moving a path will move the entire object.

Contour = contains one or more line segments.

Line Segment = the straight or curved line between two nodes.

Node = Nodes are the square points on a path or contour that allow you to change the shape of a line or curve. You alter a vector object by moving, adding, merging or deleting nodes or by rotating the node handles.

Example

Find out everything you want to know about working with vector graphic tools from the Inkscape website. https://inkscape.org/en/
Intro to SVG and Canvas Animations

**SVG Basics**

With SVG, you can draw 2D vector graphics using XML. An svg file can be opened in most browsers to view the image and can be used in an image tag or you can actually put the SVG code directly in the page.

SVG builds an object model of styles, elements, and attributes.

When the `<svg>` element appears, it acts like an inline block that is part of the HTML document tree.

With SVG, you can fill or stroke along the outline of the given graphical element.

You can style any shape and text using:

- color `<color>`,
- gradients (linear or radial)
- patterns (vector or image)
Intro to SVG and Canvas Animations

**SVG Basics**
- Object Model-based (Similar to HTML elements)
- Multiple graphical elements which become part of the DOM
- Visual presentation created HTML and modified by CSS or programmatically through script (JavaScript)
- Event model/user interaction is object-based at the level of primitive graphic elements (lines, rectangles, paths)
- Resolution independent, based on XML, Good support for animations via scripting, CSS animations and SMIL
- SVG markup and object model supports accessibility directly.

**SVG Code**

```
<svg width="300px" height="300px">
  <text x="25" y="50" font-size="24">SVG Circle Element</text>
  <text x="25" y="275">Click the circle to change its size.</text>
  <circle cx="125" cy="150" r="50"
    fill="pink" stroke="green" stroke-width="5"
    onclick="doCircle(evt)" />
</svg>
```

**Browser view**

SVG Circle Element
Intro to SVG and Canvas Animations

<table>
<thead>
<tr>
<th>Elements</th>
<th>Properties</th>
<th>Attributes</th>
<th>Attributes</th>
<th>Attributes</th>
<th>Attributes</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>clip</td>
<td>clip</td>
<td>font-family</td>
<td>offset</td>
<td>stroke-width</td>
<td>onAbort</td>
</tr>
<tr>
<td>animate</td>
<td>clip-path</td>
<td>clip-path</td>
<td>font-size</td>
<td>opacity</td>
<td>onActivate</td>
<td>onActivate</td>
</tr>
<tr>
<td>filter</td>
<td>clip-rule</td>
<td>clip-rule</td>
<td>font-stretch</td>
<td>operator</td>
<td>onBegin</td>
<td>onBegin</td>
</tr>
<tr>
<td>linearGradient</td>
<td>color</td>
<td>color</td>
<td>font-style</td>
<td>order</td>
<td>targetX</td>
<td>onEnd</td>
</tr>
<tr>
<td>radialGradient</td>
<td>fill</td>
<td>cx</td>
<td>font-variant</td>
<td>origin</td>
<td>targetY</td>
<td>onEnd</td>
</tr>
<tr>
<td>view</td>
<td>filter</td>
<td>cy</td>
<td>font-weight</td>
<td>overflow</td>
<td>textAnchor</td>
<td>onEnd</td>
</tr>
<tr>
<td>a</td>
<td>display</td>
<td>d</td>
<td>height</td>
<td>path</td>
<td>textDecoration</td>
<td>onEnd</td>
</tr>
<tr>
<td>marker</td>
<td>font</td>
<td>dx</td>
<td>id</td>
<td>pathLength</td>
<td>textDecoration</td>
<td>onEnd</td>
</tr>
<tr>
<td>clipPath</td>
<td>font-family</td>
<td>dx</td>
<td>local</td>
<td>points</td>
<td>textDecoration</td>
<td>onEnd</td>
</tr>
<tr>
<td>mask</td>
<td>mask</td>
<td>dy</td>
<td>marker-end</td>
<td>radius</td>
<td>textRendering</td>
<td>onload</td>
</tr>
<tr>
<td>pattern</td>
<td>opacity</td>
<td>dy</td>
<td>marker-mid</td>
<td>result</td>
<td>textLength</td>
<td>onload</td>
</tr>
<tr>
<td>circle</td>
<td>stop-color</td>
<td>fill</td>
<td>marker-start</td>
<td>rotate</td>
<td>transform</td>
<td>onMousedown</td>
</tr>
<tr>
<td>ellipse</td>
<td>stop-</td>
<td>fill-opacity</td>
<td>mask</td>
<td>rx</td>
<td>type</td>
<td>onMousedown</td>
</tr>
<tr>
<td>line</td>
<td>opacity</td>
<td>dx</td>
<td>name</td>
<td>ry</td>
<td>viewbox</td>
<td>onMouseover</td>
</tr>
<tr>
<td>polygon</td>
<td>stroke</td>
<td>fill-rule</td>
<td>scale</td>
<td>visibility</td>
<td>width</td>
<td>onMouseover</td>
</tr>
<tr>
<td>polyline</td>
<td>stroke</td>
<td>filter</td>
<td>stop-color</td>
<td>x</td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>rect</td>
<td>opacity</td>
<td></td>
<td>stop-opacity</td>
<td>x1</td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>g</td>
<td>stroke</td>
<td></td>
<td>stroke</td>
<td>y</td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>image</td>
<td>width</td>
<td></td>
<td>dasharray</td>
<td>y1</td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>use</td>
<td>visibility</td>
<td></td>
<td>stroke</td>
<td>y2</td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>style</td>
<td>writing-mode</td>
<td></td>
<td>dashoffset</td>
<td></td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>text</td>
<td>mode</td>
<td></td>
<td>stroke-opacity</td>
<td></td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
<tr>
<td>textPath</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>onWheel</td>
<td>onMouseover</td>
</tr>
</tbody>
</table>

For more a complete list of SVG elements, please visit the W3Schools site at: [http://www.w3schools.com/svg/svg_reference.asp](http://www.w3schools.com/svg/svg_reference.asp)
**SVG Animation**
You may have noticed on the previous slide that there was an animate element. SVG can animate based on the use of that tag. To animate you need to make the animate tag within the element being animated, specify what attributes to animate, what values to go between, the duration of the animation and how many times to repeat it.

```xml
<ellipse cx="100" cy="50" rx="85" ry="40" fill="url(#colorfulGradient)" stroke="#ff0000">
  <animate attributeName="cx" dur="1s" values="100; 300; 100" repeatCount="indefinite"/>
</ellipse>
```

The SVG can also animated programmatically with JavaScript by using the same animation methods discussed earlier in these slides.

**Beyond**
Writing SVG code manually is unreasonably tedious, but that’s what is needed in order to accomplish animation with SVG. And you often want change the SVG through some use of JavaScript. To simplify this there are a number of JavaScript libraries to make working with SVGs easier. For example, Snap SVG.

http://snapsvg.io/
HTML5 - Lesson 6
Objectives

- **Animation with JavaScript**
  - Tracking values with variables
  - With `setInterval`
  - With `setTimeout`
  - With `requestAnimationFrame` method
- **SVG**
  - Discuss raster vs vector images
  - Inkscape’s use in making SVGs
  - SVG code
  - Animation and Beyond
Sources


