HTML5 & CSS3

A chance to Do things Differently

Eng. Niveen Nasr El-Den SD
& Gaming CoE
iTi

Day 2

New Element Enable & Feature Detection

New Element Enable

- Earlier IE doesn’t know how to render CSS on elements that it doesn’t recognize
- HTML5 Shiv or Shim by John Resig document.createElement(“….”) for all of the used tag

https://github.com/aFarkas/html5shiv/blob/m aster/src /html5shiv.js

API Feature Detection

- Modernizr.js
  - Implement HTML5 Shim
  - Apply classes to <html> based on what the browser support
  - Better place its script within <head> and after<style>
  - if(!Modernizr.localstorage){
- //provide polyfill}
- https://github.com/Modernizr/Modernizr/wiki/
  - HTML5-Cross-browser-Polyfills

http://html5please.com/#polyfill

API Feature Detection

- Modernizr.js
  - ▹Runs automatically, creating a ***global*** object called ***Modernizr*** that contains a set of Boolean properties for each feature it can detect.
- Example:
  - if your browser supports the video API , the Modernizr.video
  - property will be true.
  - else, the Modernizr.video property will be false
  - ▹By default, ***Modernizr*** sets classes for all of tests on the root element.
- i.e. adding the class for each feature when it is supported,
- and adding it with a no- prefix when it is not.
- ▹ It is recommended to add no-js class to root element

API Feature Detection

- Conditionally loading .js file
  - Conditionizr library
    - https://conditionizr.github.io/
    - https://github.com/conditionizr/conditionizr
  - Conditionizr jQuery Plugin
    - https://github.com/renvrant/conditionize.js/tree/master
    - https://www.jqueryscript.net/form/jQuery-Plugin-For-Conditional-Form-Fields-conditionize-js.html
  - Head.js
    - http://headjs.com/

Demo

Modernizr.js

http://fmbip.com/litmus/
http://www.wufoo.com/html5/
http://html5readiness.com/
http://caniuse.com/

MathML

MathML

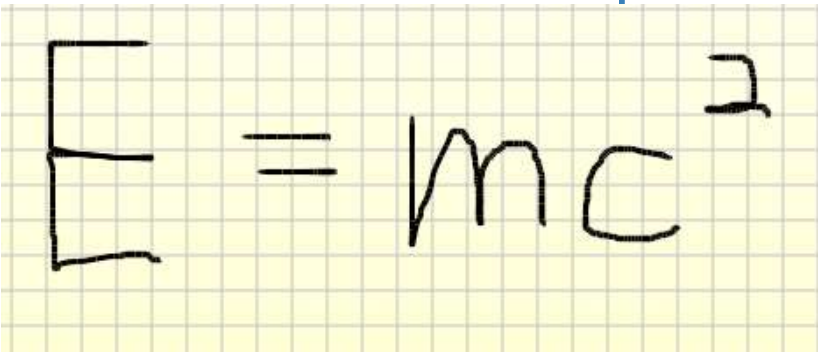
- MathML is an XML vocabulary for representing mathematical expressions
- The HTML5 specification provides native support for MathML in HTML documents
- MathML provides both Presentation and Content Markup models.
  - Presentation markup tags math expressions based on how they should be displayed
    - e.g., “superscripted 2”
  - Content markup tags expressions based on the mathematical operations performed
    - e.g., “taken to the 2nd power”

MathML Presentation Markup Glossary

- <math> -- Root element for a mathematical expression
- <mrow> -- Element for grouping subexpressions
- <mo> -- Math operator (e.g., +, -)
- <mi> -- Math identifier (e.g., variable or constant)
- <mn> -- Number
- <mfrac> -- Fraction
- <msqrt> -- Square root
- <msup> -- Superscript
- <msub> -- Subscript
- <mfenced> -- Parentheses or braces

Converting Famous Eqn. to MathML

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi> E </mi>
    <mo> = </mo>
    <mi> m </mi>
    <msup>
      <mrow>
        <mi> c </mi>
      </mrow>
      <mrow>
        <mn> 2 </mn>
      </mrow>
    </msup></math>
```



SVG

SVG

- SVG stands for **S**calable **V**ector **G**raphics and it is a language for describing 2D-graphics and graphical applications in XML
- SVG is W3C standard
- HTML5 allows embedding SVG directly using **<svg>...</svg>**

SVG

- SVG would draw
  - Rectangle using

    <rect x="" y="" width=“" height=“" style=“”>

  - line using
    - <line x1=“" y1=“” x2=“" y2=““ style=“”>
  - circle using
    - <circle cx="" cy=“" r=“" stroke=““ stroke-width=“" fill=““>
  - ellipse using
    - <ellipse cx=“" cy=“" rx=“" ry="“ style=“”>

SVG

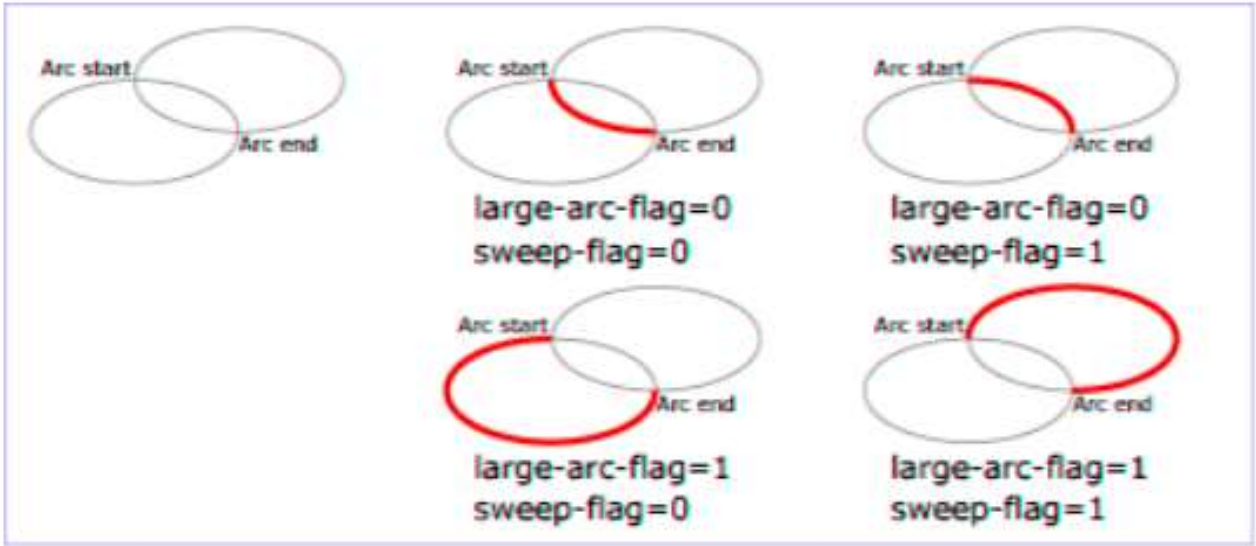
- SVG would draw
  - path
    - <path d=“”>
  - polygon using
    - <polygon points=“”> tag
  - polyline using
    - <polyline points=“”> tag

http://www.svgbasics.com/index.html

SVG - Path

- The shape of a <path> element is defined by one parameter d.
- MoveTo: M, m (M x y)
- LineTo: L, l, H, h, V, v
  - L x y
  - H x
  - V y
- Elliptical Arc Curve: A, a
  - A rx ry x-axis-rotation large-arc-flag sweep-flag
- ClosePath: Z, z

SVG


Arc start
Arc end
Arc start
Arc end
large-arc-flag=0
sweep-flag=0
Arc start
Arc end
large-arc-flag=0
sweep-flag=1
Arc start
Arc end
large-arc-flag=1
sweep-flag=0
Arc start
Arc end
large-arc-flag=1
sweep-flag=1

Canvas

Canvas

- Canvas is a new HTML element
- A canvas is a rectangular area, that you control every pixel of it.
- The canvas element has several methods for drawing paths, boxes, circles, characters, and adding images...

Canvas

- <canvas> element is an HTML tag, with the exception that its contents are rendered with JavaScript.
- It creates a fixed size drawing surface that exposes one or more *rendering contexts* using canvas context object.
- Each canvas element can only have one context that can be “2d”.

Canvas

- Draw dynamic and interactive graphics
- Draw images using 2D drawing API
  - Lines, curves, paths, shapes, fill styles, etc.
- Useful for:
  - Graphs
  - Applications
  - Games and Puzzles
  - And more…

Steps to follow

- Place the canvas tag somewhere inside the HTML document,
- Access the canvas tag with JavaScript,
- Create a 2D context, and then
- Utilize the HTML5 Canvas API to draw visualizations.

```
<canvas id="myCanvas" width="300" height="150"></canvas>

<script>
    var canvas = document.getElementById('myCanvas'); var
    context = canvas.getContext('2d');
    // do stuff here
</script>
```

Canvas Element & Canvas Context

- The canvas element is an actual DOM node that's embedded in the HTML page.
- The canvas context is an object with properties and methods that you can use to render graphics inside the canvas element.
- The context is *2d*.

Canvas Context Properties & Methods

▪	Color &Fill Styles	▪	Shadows
▪	Line	▪	Images/Videos
▪	Path	▪	Clipping
▪	Curve	▪	Transforms
	▷ Besier		▷ Scale
	▷ Quadratic		▷ Translate
▪	Shapes		▷ Rotate
	▷ Rectangle	▪	Patterns
	▷ Circle	▪	Gradients

Line using HTML5 Canvas

- To draw a line using HTML5 canvas

http://www.w3.org/TR/2dcontext/#building-paths

- First, use the *beginPath()*
- method to declare that we are about to draw a new path.
- Next, use the *moveTo()*
- method to position the context point (i.e. drawing cursor
- Then, use the *lineTo()*
- method to draw a straight line from the starting position to a new position.
- Finally, to make the line visible, we can apply a stroke to the line using *stroke()*.
- Note:
  - without declaring strokeStyle property before
  - using stroke(), the stroke default color is black

Line useful Properties & Methods

- lineWidth
  - used to define width of the required line to be drawn in px,
- should be declared before strokeStyle property.

- lineCap = square | round | butt
  - declares how the drawn line ends look

- lineJoin = bevel | round | miter
  - declares how two lines are joined together

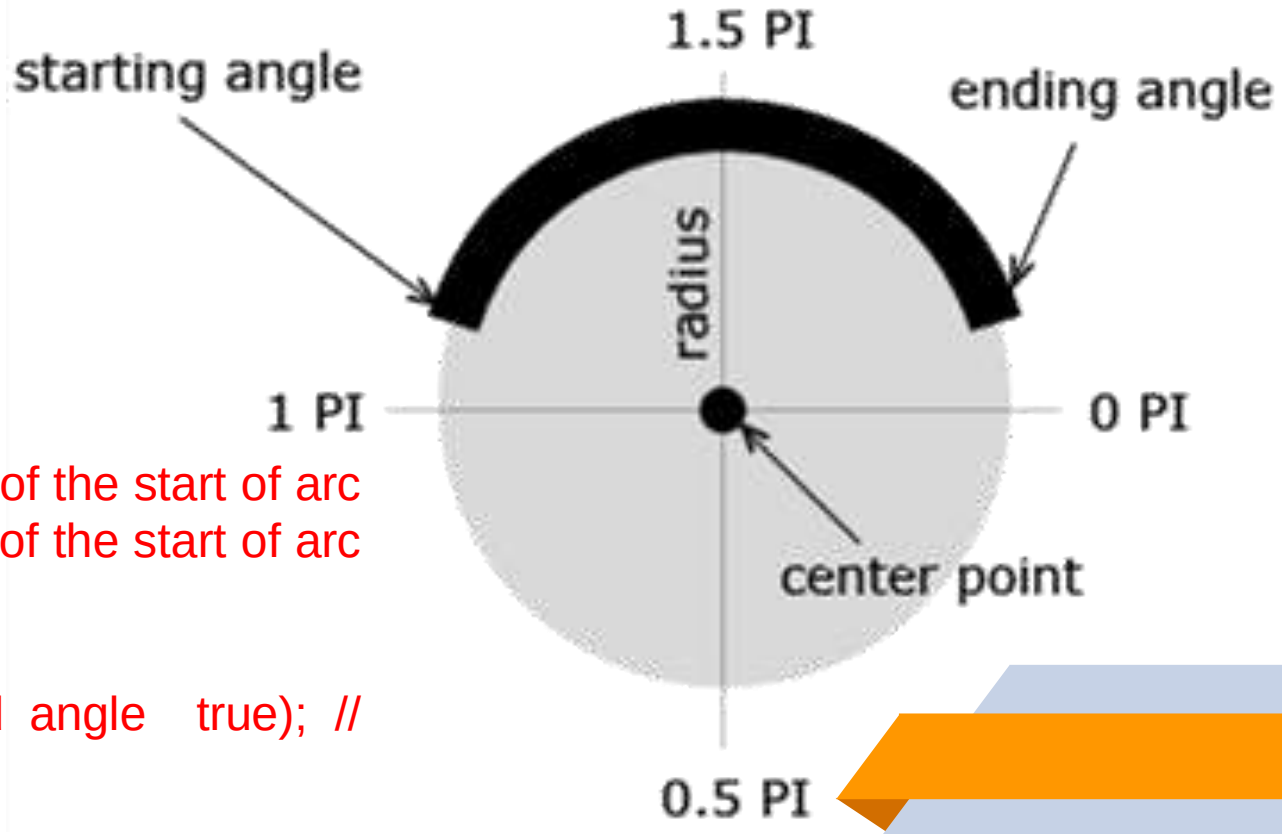
Curves & Arcs Using HTML5 Canvas

arc(x, y, radius, startAngle, endAngle, antiClockwise);

- An arc is nothing more than a section of the circumference of an imaginary circle that can be defined by *x*, *y*, and *radius*.
- *startAngle* and *endAngle*. These two angles are defined in radians.
- *antiClockwise* boolean value which defines the direction of the arc path between its two ending points, its default is false
  - i.e. the arc to be drawn is clockwise

Curves & Arcs Using HTML5 Canvas

- arc(x, y, radius, startAngle, endAngle, antiClockwise);
- arcTo(controlX,controlY,endX,endY,radius);



Example:

```
arc(
    210, // X coordinate of the start of arc
    210, // Y coordinate of the start of arc
    200, // Radius
    0,   // Start angle
    Math.PI * 2, // End angle  true); // Anticlockwise
```

Circle & Semi-Circle using HTML5 Canvas

- To draw a circle
  - Use *arc()* method and define its starting angle as 0 and the ending angle as 2 * PI.

    arc(x, y, radius, 0, 2*Math.PI, anticlk);

- To draw a semi-circle
  - Use *arc()* method and define its ending angle has *startAngle* + PI.

    arc(x, y, radius, sAngle, sAngel+Math.PI, anticlk);

Rectangle using HTML5 Canvas

rect(x, y, width, height) fillRect(x, y, width, height) strokeRect(x, y, width, height) clearRect(x, y, width, height)

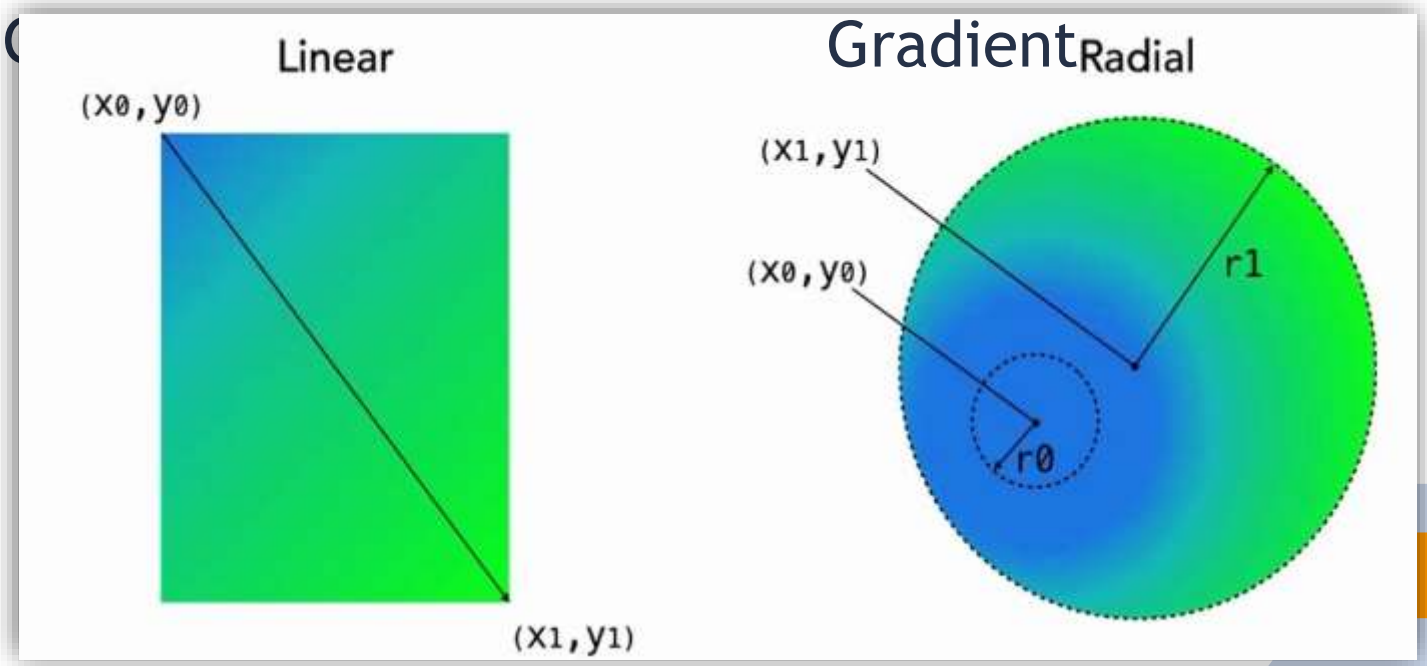
- An HTML5 Canvas rectangle is positioned with *x* and *y* parameters, and is sized with *width* and *height* parameters.
- The rectangle is positioned about its top left corner.

Paths & shapes using HTML5 Canvas

- To create a path with HTML5 Canvas, connect multiple subpaths using
  - *lineTo()*,
  - *arcTo()*,
  - *quadraticCurveTo()*, and
  - *bezierCurveTo()*
- To create a custom shape
  - First create a path and mentioned above
  - Then, close it using the *closePath()*
- *Note:* *beginPath()* is used in the beginning to start drawing a new path.
- fillStyle property & fill() can be used to fill in color within drawn shape.

Gradient

- Gradient can be used to fill rectangles, circles, lines, text, etc..
- Two types of gradient
  - Linear
  - Radial



Assignments