Web Development

Lecture 18 – HTML5 2

http://www.w3schools.com/html5/default.asp

What is HTML5?

HTML5 is the new standard for HTML.

The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then.

Until recently HTML5 was still a work in progress. However, the major browsers have supported many of the new HTML5 elements and APIs for some time.

The W3C plan published a full HTML5 recommendation in Oct 2014.

There was a version of HTML5 which obeyed the stricter rules of all XML documents, called XHTML5. Rather than develop it as a separate language, they seem to be trying to make HTML5 compatible with XML rules.

Topics

Last time:

- Embedding Video
- Embedding Audio

This time:

- Canvas
- SVG
- Geolocation
- Web Storage
- Web Workers

HTML5 Canvas

The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).

The <canvas> element is only a container for graphics, you must use a script to actually draw the graphics.

A canvas is a drawable region defined in HTML code with height and width attributes.

Canvas has several methods for drawing paths, boxes, circles, characters, and adding images.

A canvas is specified with the <canvas> element.

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

The <canvas> element has no drawing abilities of its own.

All drawing must be done inside a JavaScript script:

```
<script type="text/javascript">
  var c=document.getElementById("myCanvas");
  var ctx=c.getContext("2d");
  ctx.fillStyle="#FF0000";
  ctx.fillRect(0,0,150,75);
</script>
```

JavaScript uses the id to find the <canvas> element:

```
var c=document.getElementById("myCanvas");
```

Then, create a context object:

```
var ctx=c.getContext("2d");
```

The getContext("2d") object is a built-in HTML5 object, with many methods to draw paths, boxes, circles, characters, images and more.

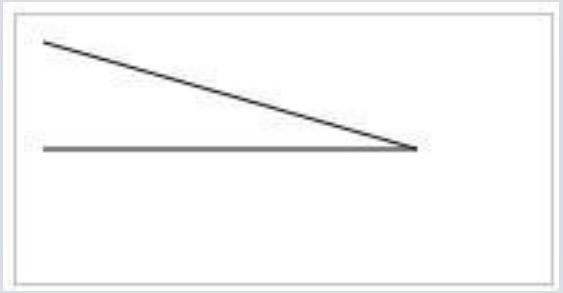
The next two lines draws a red rectangle:

```
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
```

The fillStyle attribute makes it red, and the fillRect attribute specifies the shape, position, and size.

Canvas Example 1

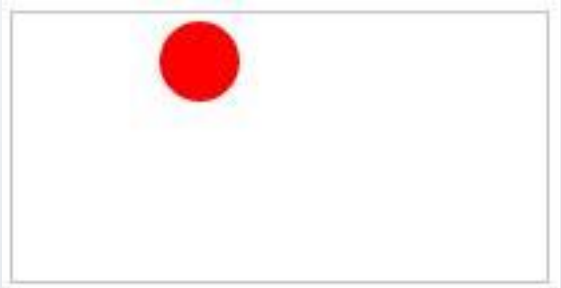
```
<!DOCTYPE html>
<html>
  <body>

    <canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">
      Your browser does not support the canvas element.
    </canvas>

    <script type="text/javascript">

      var c=document.getElementById("myCanvas");
      var ctx=c.getContext("2d");
      ctx.moveTo(10,10);
      ctx.lineTo(150,50);
      ctx.lineTo(10,50);
      ctx.stroke();

    </script>

  </body>
</html>
```



Canvas Examples 2 and 3

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.beginPath();
ctx.arc(70,18,15,0,Math.PI*2,true);
ctx.closePath();
ctx.fill();
```

context.arc(x, y, r, sAngle, eAngle, clockwise);



```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var grd=ctx.createLinearGradient(0,0,175,50);
grd.addColorStop(0,"#FF0000");
grd.addColorStop(1,"#00FF00");
ctx.fillStyle=grd;
ctx.fillRect(0,0,175,50);
```

Canvas Example 4

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var img=new Image();
img.onload = function()
               {
                ctx.drawImage(img,0,0);
               };
img.src="img_flwr.png";
```

The canvas element has many more drawing methods, transformation methods, and can even hold text. See:

http://www.w3schools.com/html5/html5_ref_canvas.asp

HTML5 Inline SVG

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- 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

SVG Advantages over other formats such as JPEG or GIF:

- SVG images can be created and edited with any text editor
- SVG images can be searched, indexed, scripted, and compressed
- SVG images are scalable
- SVG images can be printed with high quality at any resolution
- SVG images are zoomable (and the image can be zoomed without degradation)

SVG Example

```
<!DOCTYPE html>
<html>
<body>

<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     height="190">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>

</body>
</html>
```

SVG elements

Rectangle	**<rect>**
Circle	**<circle>**
Ellipse	**<ellipse>**
Line	**<line>**
Polyline	**<polyline>**
Polygon	**<polygon>**
Path	**<path>**
Text	**<text>**

SVG is an XML based language, separate from HTML, which is designed to encode pictures as a series of text commands.

It has it's own tutorial on W3Schools.

HTML5 Geolocation

```
<button onclick="getLocation()">Try It</button>

<p id="demo"></p>

<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";}
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.c
  "<br />Longitude: " + position.coords.longi
  }
</script>
```

If Geolocation is supported

Run the getCurrentPosition method, or display error message

Display a message showing the lattitude and longitude

The getCurrentPosition() Method - Return Data

The getCurrentPosition() method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned. The other properties below are returned if available.

Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

HTML5 Web Storage

With HTML5, web pages can store data locally within the user's browser.

Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.

The data is stored in key/value pairs, and a web page can only access data stored by itself.

There are two new objects for storing data on the client:

- localStorage - stores data with no expiration date
- sessionStorage - stores data for one session

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
localStorage.lastname="Smith";
document.getElementById("result").innerHTML =
                        "Last name: " + localStorage.lastname;
```

Example explained:

- Create a localStorage key/value pair with key="lastname" and value="Smith"
- Retrieve the value of the "lastname" key and insert it into the element with id="result"

Tip: Key/value pairs are always stored as strings. Remember to convert them to another format when needed.

The sessionStorage Object

The sessionStorage object is the same as the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

The following example counts the number of times a user has clicked a button, in the current session:

Sessionstorage Example

```
<head>
  <script>
    function clickCounter()
      {
        if(typeof(Storage)!=="undefined")
          {
            if (sessionStorage.clickcount)
              {      sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;      }
            else
              {      sessionStorage.clickcount = 1;      }
            document.getElementById("result").innerHTML = "You have clicked the button " +
                                                          sessionStorage.clickcount +
                                                             " time(s) in this session.";
          }
        else
          {
            document.getElementById("result").innerHTML =
                                       "Sorry, your browser does not support web storage...";
          }
      }
  </script>
</head>
<body>
  <p><button onclick="clickCounter()" type="button">Click me!</button></p>
  <div id="result"></div>
  <p>Click the button to see the counter increase.</p>
  <p>Close the browser tab (or window), and try again, and the counter is reset.</p>
</body>
```

HTML5 Web Workers

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page.

You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Web workers are supported in all major browsers.

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

```
if(typeof(Worker)!=="undefined")
  {
    // Yes! Web worker support!
    // Some code.....
  }
else
  {
    // Sorry! No Web Worker support..
  }
```

Create a Web Worker File

Here, we create a web worker script that increments a count variable.

The script is stored in the "demo_workers.js" file:

```
var i=0;

function timedCount()
  {
    i=i+1;
    postMessage(i);
    setTimeout("timedCount()",500);
  }

timedCount();
```

The important part of the code above is the **postMessage()** method - which is used to posts a message back to the HTML page.

Create a Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page.

The following line creates a new web worker object and runs the code in "demo_workers.js":

```
w=new Worker("demo_workers.js");
```

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

```
w.onmessage = function(event)
        {
        document.getElementById("result").innerHTML = event.data;
        };
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

```
<!DOCTYPE html>
<html>
  <body>
    <p>Count numbers: <output id="result"></output></p>
    <button onclick="startWorker()">Start Worker</button>
    <button onclick="stopWorker()">Stop Worker</button>
    <br /><br />

  <script>
    var w;

    function startWorker()
      {
       if(typeof(Worker)!=="undefined")
         {
          w = new Worker("demo_workers.js");
          w.onmessage = function (event) {
                                document.getElementById("result").innerHTML=event.data;
                                        };
         }
       else
        {
          document.getElementById("result").innerHTML =
                                  "Sorry, your browser does not support Web Workers...";
        }
     }

   function stopWorker()
     {
       w.terminate();
     }
  </script>

  </body>
</html>
```

Other HTML5 Features

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard, and any element can be draggable.

All major browsers support Drag and Drop

Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

All major browsers except Internet Explorer support Server-Sent Events

Topics

Last time:

- Embedding Video
- Embedding Audio

This time:

- Canvas
- SVG
- Geolocation
- Web Storage
- Web Workers