Other HTML5 Feature Overview

- Microdata
- Web storage
- Video and audio
- Canvas
- Geolocation (not actually in HTML5 specs)

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Microdata

- **Microdata** lets you add information about a web page to the page (metadata)
- This metadata is useful for search engines and other applications
- HTML5 provides a standard way to specify microdata

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Microdata Vocabularies

- A **vocabulary** is a set of name/values that describe "things" in Web pages, e.g. Person, Organization, Event and so forth
- There is a "standard" **vocabulary** that defines the HTML elements that you can use to define microdata that defines **itemscope**, **itemtype** and **itemprop**

test-microdata.html

The standard microdata vocabulary homepage is at: schema.org. It defines types such as "Event", "Organization", "Person", "Product" and so forth. It builds on earlier work such as data-vocabulary.org from Google.

The microdata spec is at: http://dev.w3.org/html5/md-LC/.

HTML5 also defines a DOM interface so you can read/write the microdata from JavaScript, though it's not yet widely adopted by browsers.

Google provides a "RichSnippet" tool at: http://www.google.com/webmasters/tools/richsnippets that you can use to test your microdata.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>About Penelope</title>
  </head>
  <body>
    <section itemscope itemtype="http://schema.org/Person">
      <h1>About Penelope</h1>
      <p>
        Penelope has been baking excellent pizza since 1945.
        She graduated with a PhD (Pizza-hurling Doctor) degree
        from the <span itemtype="alumniOf">
        Culinary Institute of the Midwest</span>
        in 1943.
      </p>
      <h2>Contact Info</h2>
      <div itemprop="name">Penelope Goodpie</div>
      <div itemscope itemtype="http://schema.org/PostalAddress">
          <div itemprop="streetAddress">123 Elm Street</div>
          <div>
             <span itemprop="addressLocality">Anytown</span>
             <span itemprop="addressRegion">IA</span>
             <span itemprop="postalCode">52402</span>
          </div>
      </div>
      <div itemprop="email">penelope@penelopepizza.com</div>
    </section>
  </body>
</html>
```

Statefulness: Issues with Cookies

- To maintain state across requests, applications historically have used **browser cookies**
- Though cookies have served the Web well, they have some shortcomings:
    - Browser limit cookie size, generally to 4KB max
    - Browsers send the cookie on all subsequent requests until the cookie expires

There have been several attempts to address the issues with cookie, including using Flash "local shared objects", but all have been either proprietary or not widely adopted by browsers.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Web Storage

- **Web storage** is an API for client-side storage that's widely adopted by browsers
- Most browsers limit Web storage to 5MB for each domain, and a domain cannot access storage from another domain
- Storage comes in two flavors:
    - **Local storage** – Data persists "forever"
    - **Session storage** – Data persists until browser closed

---

Unlike with cookies, browsers never automatically send Web storage data to the server.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Using Web Storage

- The Web Storage API provides methods for defining and retrieving data using key/value pairs

test-webstorage-page1.html

test-webstorage-page2.html

Currently, the "value" must be a JavaScript string - to store objects or arrays, you can convert them to a JSON string before storing and then convert them back when you retrieve the strings. JavaScript provides the JSON.stringify() and JSON.parse() methods to convert to/from JSON.

Note that there's also a JavaScript event that programs can handle to monitor the storage.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

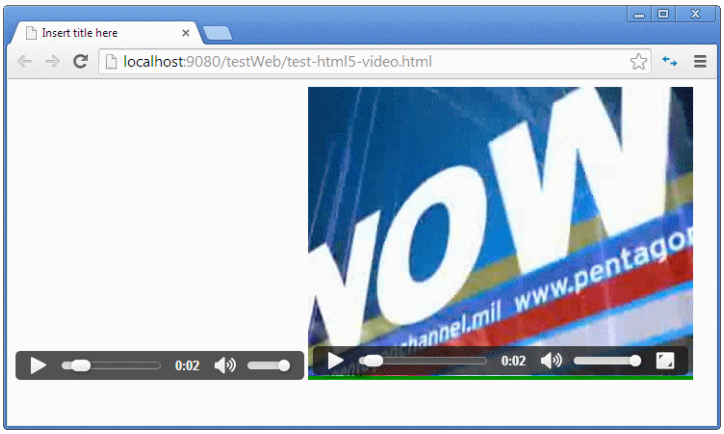
```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <title>Test Web Storage, Page 1</title>
5        <script>
6          function saveIt()
7          {
8            sessionStorage.setItem("mydata1", "Some good data");
9            localStorage.setItem("mydata2", "Some more good data");
10           return true;
11         }
12       </script>
13     </head>
14     <body>
15       <form action="test-webstorage-page2.html"
16         onsubmit="return saveIt();">
17         <input type="submit" value="Next">
18       </form>
19     </body>
20   </html>
```

```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <title>Test Web Storage, Page 2</title>
5        <script>
6          function getIt()
7          {
8            var d1 = sessionStorage.getItem("mydata1");
9            alert("Session var: " + d1);
10           var d2 = localStorage.getItem("mydata2");
11           alert("local var: " + d2);
12         }
13       </script>
14     </head>
15     <body onload="getIt();">
16     </body>
17   </html>
```

Audio and Video

- HTML5 provides standards-based audio and video playback using the <audio> and <video> tags
- Browser support for the tags is quite good, but support for all audio and video formats is not universal - you can list multiple sources and let the browser choose

test-audio-video.html



Common audio formats include MP3, and Ogg Vorbis, while common video formats include Ogg Theora and MP4. Some of the formats are free and unencumbered by patents, while others potential require royalty payments.

The "controls" attribute tells the browser, if possible, to display an interface for starting, pausing and stopping the playback. Note that there are other options, such as "autoplay" (which starts playback on page load) and that you can use JavaScript to control the playback.

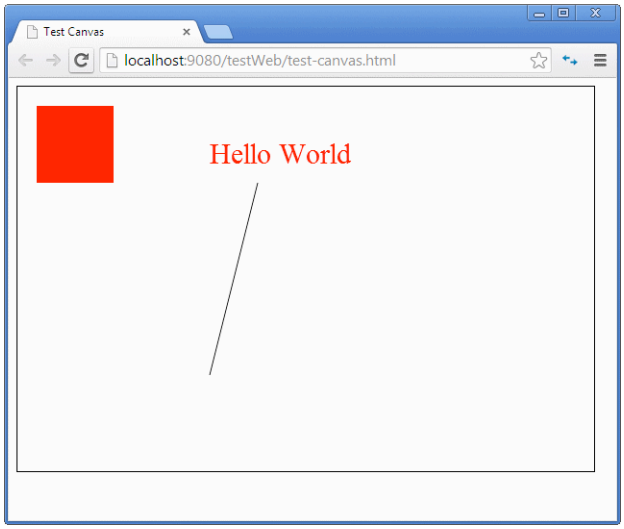
Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <title>Test Audio/Video</title>
5      </head>
6      <body>
7        <audio controls>
8          <source src="ACDC_-_Back_In_Black-sample.ogg"
9            type="audio/ogg">
10         <source src="ACDC_-_Back_In_Black-sample.mp3"
11           type="audio/mpeg">
12         <p>Audio not supported fallback content</p>
13       </audio>
14       <video controls>
15         <source src="Pentagon_News_Sample.ogg"
16          type='video/ogg;'>
17         <p>Video not supported fallback content</p>
18       </video>
19     </body>
20   </html>
```

Canvas

- The HTML5 **canvas** API defines a standard way for JavaScript programs to draw to a rectangle on a Web page

test-canvas.html



The coordinate system is anchored at the top-left of the canvas and the units are generally in pixels (browsers running on high-resolution devices might scale pixel coordinates).

Canvas supports a wide range of drawing primitives. The current spec is at http://www.w3.org/TR/2014/WD-2dcontext-20140520/.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <title>Test Canvas</title>
5        <style>
6          canvas {border: solid black 1px;}
7        </style>
8        <script>
9          function drawIt()
10         {
11           var canvas = document.getElementById("mycanvas");
12           var ctx = canvas.getContext("2d");
13
14           ctx.fillStyle = "rgb(255, 0, 0)";
15
16           ctx.fillRect(20, 20, 80, 80);
17
18           ctx.moveTo(250, 100);
19           ctx.lineTo(200, 300);
20           ctx.stroke();
21
22           ctx.font = "30px Times";
23           ctx.fillText("Hello World",200,80);
24
25         }
26       </script>
27     </head>
28     <body onload="drawIt()">
29
30       <canvas width="600" height="400" id="mycanvas">
31
32         <p>No canvas support placeholder</p>
33       </canvas>
34
35     </body>
36   </html>
```

Geolocation

- The **geolocation** API defines a standard way for JavaScript programs to obtain positional information about the user
- Browsers generally require the the user to consent before providing the information

test-geolocation.html

The Geolocation specification is at: http://dev.w3.org/geo/api/spec-source.html.

There is an optional third parameter to getCurrentPosition() that lets you specify options, for example, enableHighAccuracy, which requests GPS location on devices that support it. That parameter also lets you configure the timeout period.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <meta charset="ISO-8859-1">
5        <title>Test Geolocation</title>
6        <script>
7          function findMe()
8          {
9             navigator.geolocation.getCurrentPosition(
10               foundMe, error);
11         }
12
13         function foundMe(position)
14         {
15           var latitude = position.coords.latitude;
16           var longitude = position.coords.longitude;
17           document.write("Latitude: " + latitude + "<br>");
18           document.write("Longitude: " + longitude);
19         }
20
21         function error(errorObj)
22         {
23           if(errorObj.code == 1) document.write("Denied");
24           else if(errorObj.code == 2) document.write("Unavailable");
25           else if(errorObj.code == 3) document.write("Timeout");
26         }
27       </script>
28     </head>
29     <body onload="findMe()">
30
31     </body>
32   </html>
```

Chapter Summary

In this chapter, you learned:

- About HTML5 features, such as canvas, microdata, geolocation and Web storage

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum