Demo 1 – Showing some of the new HTML5 elements

There are quite a few new HTML5 elements. In this demo some of them are shown and then styled to see what happens in browsers that understand the new elements and what happens when they do not… with particular attention to older versions of Internet Explorer

1) Navigate to page:

http://html5demo1.azurewebsites.net/Demo01a-BunchOfNewHTML5Elements.html

2) View the page source to see a bunch of the HTML5 elements in Internet Explorer
3) View the page in the browser itself and notice how the elements are all stylized green based on the stylesheet associated with the page
4) Now, put Internet Explorer into compatibility mode
5) Note how the HTML5 elements are no longer stylized. Unlike other browsers, if older versions of IE did not understand an element it would simply not style them (assuming you had a style that styled them of course)
6) Now, navigate to page

http://html5demo1.azurewebsites.net/Demo01b-BunchOfNewHTML5ElementsWithHTML5Shiv.html

7) Notice how this version of the page is stylized regardless of if IE is in compatibility mode or not
8) Viewing the page source, notice that it references something called HTML5Shiv which takes care of "telling" the browser about all of the new elements plus does some basic styling too

**NOTE: at a high level, what this JavaScript file is doing is a document.createElement for each of the HTML5 tags… among other things

9) Navigate to page

http://html5demo1.azurewebsites.net/Demo01c-BunchOfNewHTML5ElementsWithCromeFrame.html

10) Notice how this version of the page is stylized as well
11) View the page source to see how the "Google Chrome Frame" meta tag was applied within the "<head>" section of the page

```
<meta http-equiv="X-UA-Compatible" content="chrome=1">
```

**Notice how the "compatibility mode" button disappears in IE and right-clicking the page you will see "About Chrome Frame…" which is an indication right away of its use

Demo 2 – Transforming a page to HTML5

1) Open up the below page… this is a page written before HTML5

http://html5demo1.azurewebsites.net/Demo02a-PagePreHTML5.html

2) Next, open up the below page… this is the same page updated to use HTML5 semantic tags

http://html5demo1.azurewebsites.net/Demo02b-PageusingHTML5.html

3) Open up the following 2 files side by side in Visual Studio

Demo02a-PagePreHTML5.html and Demo02b-PageusingHTML5.html

4) Note that yes, there has been some simplification to the beginning tags, but overall the page has not got any smaller from a coding perspective… it's just that it now has more semantic meaning. What is one implication to this? Well, if you can imagine, screen readers for blind people or anything that is trying to organize content into something like a table of contents or search engine bots, now can much better understand the intent behind your web page. Where before meaning was coming from "id's" there is now meaning coming from standardized tags.

5) Note also that the stylesheet had to be adjusted accordingly as well… for example:
   instead of:
   ```
   #header {
   ```
   it becomes
   ```
   header {
   ```

Credit for this demo goes to http://www.webucator.com/self-paced-courses/course/comprehensive-introduction-html5.cfm

Demo 2 Bonus! – CSS3 Media queries

Expanding on the previously demonstrated page… I thought it would be interesting to show something called CSS3 media queries too… another component to HTML5.

1) Navigate to the below and review the contents:

http://html5demo1.azurewebsites.net/Demo02c-HTML5WithMediaQuery.html

   a. Notice the following entry:

```
@media only screen and (max-width: 480px) {
```

…which says that while displaying on a screen up to 480 pixels in width, change the background color, remove the background image, remove the logo and remove the sidebar. This could be seen as a "mobile" version of your web page.

   b. Similarly, notice the following entry:

```
@media only screen and (min-width: 481px) and (max-width: 768px) {
```

…which applies the styles beneath it when the screen is between 481px and 768px in width which could be seen as a "tablet" or "iPad" version of your web page.

   c. Lastly, notice the following entry:

```
@media only screen and (min-width: 769px) {
```

…which applies the style beneath it once the screen becomes 769px in width... which could be seen as a "desktop" version of your web page.

2) Now, resize the screen and watch as the content visually changes

Demo 3 – New Input Types

1) Open up the below demo page and review the different input types

http://html5demo1.azurewebsites.net/Demo03-NewInputTypes.html

2) Now, open up in IE and put it into "compatibility mode" to view it in "old-IE"… oh wait, on my machine, which is running IE 9, NONE OF THE NEW FORM TAGS WORK!

- Easy solution to this is, as before, "Google Chrome Frame"

http://html5demo1.azurewebsites.net/Demo03b-NewInputTypesWithChromeFrame.html

- There are other options out there too, but… will leave this as an exercise for the curious… but as you probably can guess, there are JavaScript libraries out there that will use native HTML5 when available and then fall back to JavaScript when they are not available

3) Let's leave IE… open up in Chrome, Safari, Firefox, Sea Monkey, Maxthon, Opera Emulator, etc and see controls in action. Note that this is with no added JavaScript!

   Some notable things:

   a. Safari… "Search field" will have an "x" so can clear contents
   b. Implementations of "time" varies… Maxthon vs. Chrome for example

4) Now use a simulator to see how these behave on mobile devices. Probably best resource is http://www.browserstack.com which although not free, does have a trial that I am using for this purpose
   a. Notice how the keyboard actually changes when in Url field vs Email field
   b. Number uses a keypad

5) Try out the contenteditable areas at the bottom of the page too!

Demo 4 – Multimedia

1) Open up the demo page

http://html5demo1.azurewebsites.net/Demo05-Multimedia.html

2) First link brings you to Solvera's LipDub preview video in YouTube… nothing special here: https://www.youtube.com/watch?v=JlxISJcsqxs

3) Second link navigates to YouTube with HTML5 "switch" set to true https://www.youtube.com/watch?v=JlxISJcsqxs&html5=1
NOTE: If use IE it may still work… because YouTube is using Google Chrome Frame!

4) Note the video on the page… this is using the new <video> HTML5 element and uses within it the "poster" attribute to display an image before it has started playing… in this case a Solvera logo that I had on my machine

5) Note also how easy it is to write JavaScript to play, pause and modify the volume of the video

Demo 5 – Show off Azure

1) Navigate to https://manage.windowsazure.com and login
2) From the main dashboard you can in a couple of links do things such as:
    a. Create a new Website (show on menu)
    b. Create a new VM (show on menu)
    c. Create a new SQL Database (show on menu)
    d. Create something called a mobile service (show on menu)

**Note that for those that are looking to do something for the ultrahackathon, you might be interested in looking into this one… from the dashboard it will even walk you through connecting it to a Windows Phone, iOS or Android device. You can even let the wizard create you a demo app which is a fully functioning "TODO list" application!

3) Drilling into creating a new website… some interesting options here:
    a. Can pick from gallery
    b. Can get code from GitHub, Bitbucket, Codeplex

**Note that my main demo website is actually setup with GitHub interaction. I can make a change and the commit and push it into GitHub and then Azure will automatically detect that a change is done and will deploy it.

4) To demonstrate GitHub…
    a. View the HTML5Demo1 deployment dashboard in Windows Azure
    b. Navigate to Cloud9 IDE at https://c9.io/
    c. Get the latest version from git using...
       git pull
    d. Make a change to the "Index.html" file
    e. Commit the change to git using...
       git commit -a -m "c9 commit"
    f. Push the changes to github using...
       git push origin master

5) You can also do something called "Publish" to Windows Azure using Visual Studio as well… which is how I setup the SignalRDemo website

https://manage.windowsazure.com/#Workspace/WebsiteExtension/Website/SignalRDemo1/quickstart

Demo 6 – SignalR

http://www.asp.net/signalr/

1) Start-up demo site on Azure
2) Navigate to the HitDemo
3) Go to the site in many different browsers… or tabs within a single browser
4) Notice that the counter keeps increasing or decreasing as more and more browsers are hit
5) Review the code by opening up Visual Studio (if not already open)
6) Code reviewed below… from JavaScript:

```
$(function () {
    /* using SignalR JavaScript library, setup the connection */
    var cn = $.hubConnection();
    /*
    next, using SignalR JavaScript library, create a "proxy" that
    you can either send stuff to or receive stuff from
    */
    var hub = cn.createHubProxy('hitCounter');
    /*
    setup a listener from the hub to this web page...
    so, whenever the server raises an event called 'updateHitCount'
    the text for the div 'recordHit' will be adjusted
    */
    hub.on('updateHitCount', function(i) {
        $('#hitCount').text(i);
    });
    /*
    start the connection...
    this creates and maintains a real-time connection
    */
    cn.start(function() {
        hub.invoke('recordHit');
    });
});
```

7) Code reviewed below… from C#:

```
//attribute sets what "serviceProxy" that can create
//default is class name
[HubName("hitCounter")]
public class HitcounterHub : Hub
{
    //holds the number of connected browsers
    private static int _hitCount;

    //server method that can be invoked by JavaScript
    public void RecordHit()
    {
        _hitCount += 1;
        //note... this is a dynamic method... think of the method as an event
        Clients.All.updateHitCount(_hitCount);
        //when this event fires on the server there is going to be a JavaScript
        //event that will fire
        //NOTE:
        //- "All" means All callers will see the response
        //there are other more secure options here as well
    }

    //SignalR will communicate automatically when the browser is disconnected
    //For example, if close the browser window / tab, this will automatically
    //be called!
    public override Task OnDisconnected()
    {
        //decrease the counter
        _hitCount -= 1;
        //send a message back to all of the connected clients
        Clients.All.updateHitCount(_hitCount);
        return base.OnDisconnected();
    }
}
```