HTML5 Learning Note

Tuesday, April 29, 2014 13:44

1. **A more perfect DOCTYPE**
   <!DOCTYPE html>

2. **Declaring language in HTML5**
   <html lang="en">
   <html lang="en-US"> <!-- US English -->
   <html lang="en-GB"> <!-- UK English -->

3. **Character encoding**
   <meta charset="UTF-8" />

4. **How do browsers deal with unknown elements? (Meet the shiv)**
   <!--[if lt IE 9]>
   <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
   <![endif]-->

5. **Declaring block-level elements**
   when it comes to styling, browsers treat unknown elements as anonymous inline elements.
   HTML5 introduces a number of new block level elements: if these elements aren't included in the browser's lookup table of known elements, they will be treated as inline. We therefore need to add a CSS rule declaring them as block-level elements.

```
<style>
    article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section {
    display: block;
    }
</style>
```

6. **An HTML5 boilerplate page**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Insert Your Title Here</title>
    <!--[if lt IE 9]>
        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>   <!-- no more type attribute '"style type="text/css" "' -->
    <![endif]-->
    <style>
        article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section
        {
        display: block;
        }
    </style>
</head>
<body>
    <p>Insert your content here.</p>
</body>
</html>
```

7. **New Elements**
   a. **The **: In writing POSH (plain old semantic HTML), we should use the most suitable or semantically accurate element. While has the semantics of a general flow container element, it doesn't have any semantic meaning beyond this, and it's used when there are no elements that are more appropriate (i.e., all the time in HTML 4). There is no requirement for the contents of a to be related to each other.
   b. **The <section>:** The new HTML5 <section> element is similar to as a general container element, but it does have some additional semantic meaning—the things it contains are a logical group of related content. <section> is also a sectioning content element. Along with <article>, <nav>, and <aside>, it indicates a new section in the document. Imagine making your page into a nested list of related parts, or using a word processor's outline view: the heading of each sectioning content element is a new item and indentation reflects nesting.
   c. **The <article>:** The new HTML5 <article> element is like a specialized kind of <section>; it has the more specific semantic meaning that it is an independent, self-contained part of the page. We could use <section>, but using <article> gives more semantic meaning.
   d. **The <header>**: Used for introductory and navigational content of a sectioning element. This typically includes the heading (an <h1>–<h6> element or an <hgroup> element), but can also contain other content, such as a <nav> element or navigational links, a table of contents, a search form, or any relevant logos. It can't contain <footer> or another <header>.
   e. **The <footer>:** Used for additional information about the content, such as who wrote it, links to related documents, copyright data, a link to the top of the page, etc., and usually appears at the end of the content. Like <header>, it can't contain <header> or <footer> elements.
   f. **<hgroup>:** A specialized form of <header> that can contain only <h1>–<h6> elements. It is for grouping a heading with subheading(s).
   g. **<h1>–<h6>**: The heading elements from HTML 4 are back and basically unchanged, except for HTML5's stronger guidance on using them correctly (generally, don't skip levels), and the interesting addition of HTML5-style heading levels.
   h. **<nav>:** A section of navigational links, either to other pages (generally site navigation) or to sections on the same page (such as a table of contents for long articles). This is for major navigation blocks, not just for any group of links.
   i. **<aside>:** A section of a page that consists of content that is tangentially related to—but separate from—the surrounding content. In print, this would be a sidebar, pull quote, or footnote. In a weblog article, this could be related information about the article, extra information in the margin, or the comments section.
   j. **<figure>:** For content that is essential to understanding but can be removed from the document's flow (moved to a different place) without affecting the document's meaning. This can be used for images or video, but it can also be used for any other content including a graph, code sample, or other media. Use the optional (and delicious) child <figcaption> to provide a label.


NewElement


canvas


SVG

h. **<mark>:** The mark element represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
i. And so on… details to see the "A Richer Approach to Marking Up Content" in "Beginning HTML5 and CSS3"

9. Rich Media (Canvas)

a. Simple example

```
<!DOCTYPE HTML>
<html lang="en_UK">
<head>
	<meta charset="UTF-8">
	<title>Canvas</title>
	<!--[if lt IE 9]>
		<script src="excanvas.js"></script>
	<![endif]-->
	<script>
		<!-- The 2D context -->
		window.onload = function() {
			var canvas = document.getElementById('outerspace');
			if (canvas.getContext){
				var ctx = canvas.getContext('2d');

				// set the color of rectangle
				ctx.fillStyle="rgba(0,149,197,0.8)";
				//set the size and shape (x, y, width, height)
				ctx.fillRect(10, 10, 240, 120);

				// draw another rectangle has an outline insteasd of being filled
				ctx.fillStyle="rgba(0,149,197,0.8)";
				ctx.strokeRect(10, 150, 240, 120);

				// draw cycle
				ctx.fillStyle = "rgba(0,149,197,0.8)";
				var mygrad = ctx.createLinearGradient(300,100,640,480);
				mygrad.addColorStop(0, '#6c88ba');
				mygrad.addColorStop(0.9, 'rgba(0,0,0,0.5)');
				mygrad.addColorStop(1, 'rgba(0,0,0,1)');

				ctx.fillStyle = mygrad;
				ctx.beginPath();
				// begin the path
				ctx.arc(90,370,80,0,Math.PI*2,true);
				// draw an arc - (x, y, radius, startAngle, endAngle, anticlockwise)
				ctx.closePath();
				// close path
				ctx.fill();
			} else {
				// canvas not supported do something
			}
		}
	</script>
</head>
<body>
	<canvas width="640" height="480" id="outerspace">
		Sorry, your browser doesn't support canvas.
	</canvas>
</body>
</html>
```

10. Scalable Vector Graphics (SVG)

a. Scalable Vector Graphics is a technology for creating vector graphics in the browser based on the XML file format. It is an open standard that has been actively developed by the W3C since 1999 (www.w3.org/TR/SVG).
b. SVG is not part of HTML5.
c. SVG files are vector based, meaning they can be scaled up or down without the loss of quality. This is in direct contrast to canvas, which is pixel (raster) based and doesn't scale gracefully.
d. Basic SVG is supported by all major browsers (with the exception IE8 and below). SVG embedded in HTML5, also known as "inline SVG" (which we'll be concentrating on), is supported by Chrome 7+, Internet Explorer 9+, Firefox 4+, Opera 11.5+ and Safari 5.1+.
e. Compare with Canvas:
   i. A major difference between SVG and Canvas is that every SVG element becomes part of the DOM (remember we said that canvas is always empty), allowing each object to be indexed and providing a more accessible solution.
   ii. SVG content can be static, animated, or interactive. You can style it with CSS or add behavior with the SVG DOM. Because each element becomes part of the DOM, this means that SVG is relatively accessible. It is also resolution independent and can scale to any screen size. It can, however, be slow to render with more complex shapes.
   iii. canvas, in contrast, uses a JavaScript API that allows us to draw programmatically. It has two context objects, 2D or 3D. There is no specific file format, and you can only draw

using script. Because there are no DOM nodes (unlike SVG), you can concentrate on drawing (scripting) without taking a performance hit as the image complexity increases. You can also save the resulting images as a PNG or JPG.

iv. SVG is good for data charts and resolution-independent graphics, interactive animated user interfaces, or vector image editing. canvas is suited for games, bitmap image manipulation (like the Darkroom example we saw earlier), generating raster graphics (data visualizations, fractals, etc.), and image analysis (histograms, etc.).

f. Simple Example

```
<!doctype html>
<html lang="en_UK">
<head>
	<meta charset="UTF-8">
	<title>SVG</title>
</head>
<body>
	<svg xmlns="http://www.w3.org/2000/svg" width="400px" height="400px">
		<rect fill="black" stroke="red" x="10" y="10" width="100" height="50"/>

		<circle fill="#6c88ba" stroke="red" cx="60" cy="130" r="50"/>
	</svg>
</body>
</html>
```

11. Form

a. HTML5 forms attributes

i. Placeholder: placeholder attribute, which allows us to set placeholder text as we would currently do in HTML4 with the value attribute. It should only be used for short descriptions. For anything longer, use the title attribute. The difference from HTML4 is that the text is only displayed when the field is empty and hasn't received focus. Once the field receives focus (e.g., you click or tab to the field), and you begin to type, the text simply disappears.

ii. Autofocus

iii. Autocomplete

iv. required

v. Pattern: `<input pattern="[0-9][A-Z]{3}" name="product" type="text" title="Single digit followed by three uppercase letters."/>`

vi. list and the datalist element

```
<label>Your favorite fruit:
<datalist id="fruits">
<option value="Blackberry">Blackberry</option>
<option value="Blackcurrant">Blackcurrant</option>
<option value="Blueberry">Blueberry</option>
<!-- … -->
</datalist>
If other, please specify:
<input type="text" name="fruit" list="fruits">
</label>
```

vii. Multiple

viii. novalidate and formnovalidate: The novalidate and formnovalidate attributes indicate that the form shouldn't be validated when submitted. They are both Boolean attributes. formnovalidate can be applied to submit or image input types. The novalidate attribute can be set only on the form element.

ix. form

x. formaction, formenctype, formmethod, and formtarget

xi. Formaction

xii. Formenctype

xiii. Formmethod

xiv. Formtarget

xv. Form attributes summary

b. New input types

i. Search

ii. Email

iii. Url

iv. Tel

v. Number

vi. Range

vii. Date

viii. Month

ix. Week

x. Time

xi. Datetime

xii. datetime-local

xiii. color

c. Feature detection with JavaScript

we can use JavaScript to detect "support" for a particular feature in a few different ways.

i. Check whether a property exists on a global object like window.

ii. Create an element and test to see if a property exists on it.

```
var input = document.createElement('input');
return 'placeholder' in input;
```

iii. Create an element, test to see if a method exists on that element, and then call the method to test the value it returns.

iv. Create an element, set a property to a value, and check the retained value.

```
var input = document.createElement('input');
input.setAttribute('type','email');
return input.type !== 'text';
<!--
	This piece of JavaScript tells the browser to create an input element and set the type attribute to the one
	you are detecting support for (in this case email). For browsers that support the input type, this value is
	retained. For those that don't, the value we set will be ignored and the type property will default to "text".
-->
```

12. Other New Features

a. Keygen: the keygen element represents a control for generating a public-private key pair and for submitting the public key from that key pair. When a form is submitted, a public key and a private key are generated. A private key is stored on the client side and the public key is sent to the server. The public key can then be used with secure servers to generate certificates, making it more secure and preventing your site from being hacked.
Here is an example of keygen in action.

```
<form action="process.php" method="post" enctype="multipart/form-data">
	<keygen name="key">
	<input type=submit value="Submit">
</form>
```

b. Output: The output element is rather simple; it represents the result of a calculation or user action. Using the for attribute, a relationship is created between the result of the calculation and the value(s) that went into the calculation. The output element is supported in Opera 9.5+, Firefox 4+, Chrome, Safari 5+, and Internet Explorer 10. The following example (depicted in Figure 6-27) adds the values of two inputs together to create the output. This is triggered by the oninput event placed on the output lement and handled by the browser using the new valueAsNumber function—no complex scripting required! For more information, see a detailed write up on output by author Richard Clark (http://j.mp/outputelement21).

2 + 2 = 4

```
<form onsubmit="return false" oninput="o.value = a.valueAsNumber + b.valueAsNumber">
	<input name="a" id="a" type="number" step="any"> +
	<input name="b" id="b" type="number" step="any"> =
	<output name="o" for="a b"></output>
</form>
```

c. Progress

Progress:

```
<p>Percent Downloaded:
<progress id="p" max="100" value="0"><span id="completed">0</span>%</progress>
</p>
<script>
	var progress = document.getElementById('p');
	function updateProg(newValue) {
		progress.value = newValue;
		progress.getElementById('completed')[0].textContent = newValue;
	}
</script>
```

d. Meter

Your Profile is 40% complete.

```
<p>Your profile is <span id="completed">40</span>% complete.</p>
```

13. Introduction to HTML5-related APIs

a. there are two versions of the HTML5 specification, one published by the W3C and another by the WHATWG. The living HTML specification maintained by the WHATWG contains additional APIs to those in the W3C HTML5 spec (although generally they are also maintained by the W3C but in separate specifications).

b. Alongside those in the specification are a number of related APIs that form part of the standards stack and are often grouped under the "HTML5" umbrella term. In some cases, the APIs have been around and implemented for a while, but they've never been documented (something that HTML5 has set out to change). Following figure gives an indication of how each of the APIs relates to HTML5 but is by no means comprehensive.