**34. HTML5 - How to Create a Visual Library of Images in HTML5 Canvas ?**

**35. HTML5 - HTML5 Apps: Positioning with Geolocation ?**

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

## Using HTML Geolocation

The getCurrentPosition() method is used to get the user's position.

The example below returns the latitude and longitude of the user's position:

<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.coords.latitude +   
    "<br>Longitude: " + position.coords.longitude;   
}  
</script>

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

function showError(error) {  
    switch(error.code) {  
        case error.PERMISSION\_DENIED:  
            x.innerHTML = "User denied the request for Geolocation."  
            break;  
        case error.POSITION\_UNAVAILABLE:  
            x.innerHTML = "Location information is unavailable."  
            break;  
        case error.TIMEOUT:  
            x.innerHTML = "The request to get user location timed out."  
            break;  
        case error.UNKNOWN\_ERROR:  
            x.innerHTML = "An unknown error occurred."  
            break;  
    }  
}

**36. HTML5 - How to Build Cross-Browser HTML5 Forms ?**

Modernizer: Modernizr is a small JavaScript library that tests the current browser against a plethora of HTML5 and CSS3 features.

**37. HTML5 - How to create a Progress bar with HTML5 Canvas ?**

**<!DOCTYPE html>**

**<html>**

**<head lang="en">**

**<meta charset="UTF-8">**

**<title></title>**

**</head>**

**<body>**

**<canvas id="myCanvas" width="500" height="150">Your browser does not have support for canvas.</canvas>**

**<script>**

**function *roundRect*(ctx, x, y, width, height, radius) {**

**ctx.beginPath();**

**ctx.moveTo(x + radius, y);**

**ctx.lineTo(x + width - radius, y);**

**ctx.arc(x+width-radius, y+radius, radius, -Math.PI/2, Math.PI/2, false);**

**ctx.lineTo(x + radius, y + height);**

**ctx.arc(x+radius, y+radius, radius, Math.PI/2, 3\*Math.PI/2, false);**

**ctx.closePath();**

**ctx.fill();**

**}**

**function *progressLayerRect*(ctx, x, y, width, height, radius) {**

**ctx.save();**

***// Define the shadows***

**ctx.shadowOffsetX = 2;**

**ctx.shadowOffsetY = 2;**

**ctx.shadowBlur = 5;**

**ctx.shadowColor = '#666';**

***// first grey layer***

**ctx.fillStyle = 'rgba(189,189,189,1)';**

***roundRect*(ctx, x, y, width, height, radius);**

***// second layer with gradient***

***// remove the shadow***

**ctx.shadowColor = 'rgba(0,0,0,0)';**

**var lingrad = ctx.createLinearGradient(0,y+height,0,0);**

**lingrad.addColorStop(0, 'rgba(255,255,255, 0.1)');**

**lingrad.addColorStop(0.4, 'rgba(255,255,255, 0.7)');**

**lingrad.addColorStop(1, 'rgba(255,255,255,0.4)');**

**ctx.fillStyle = lingrad;**

***roundRect*(ctx, x, y, width, height, radius);**

**ctx.restore();**

**}**

**function *progressBarRect*(ctx, x, y, width, height, radius, max) {**

***// deplacement for chord drawing***

**var offset = 0;**

**ctx.beginPath();**

**if (width<radius) {**

**offset = radius - Math.*sqrt*(Math.*pow*(radius,2)-Math.*pow*((radius-width),2));**

***// Left angle***

**var left\_angle = Math.*acos*((radius - width) / radius);**

**ctx.moveTo(x + width, y+offset);**

**ctx.lineTo(x + width, y+height-offset);**

**ctx.arc(x + radius, y + radius, radius, Math.PI - left\_angle, Math.PI + left\_angle, false);**

**}**

**else if (width+radius>max) {**

**offset = radius - Math.*sqrt*(Math.*pow*(radius,2)-Math.*pow*((radius - (max-width)),2));**

***// Right angle***

**var right\_angle = Math.*acos*((radius - (max-width)) / radius);**

**ctx.moveTo(x + radius, y);**

**ctx.lineTo(x + width, y);**

**ctx.arc(x+max-radius, y + radius, radius, -Math.PI/2, -right\_angle, false);**

**ctx.lineTo(x + width, y+height-offset);**

**ctx.arc(x+max-radius, y + radius, radius, right\_angle, Math.PI/2, false);**

**ctx.lineTo(x + radius, y + height);**

**ctx.arc(x+radius, y+radius, radius, Math.PI/2, 3\*Math.PI/2, false);**

**}**

**else {**

**ctx.moveTo(x + radius, y);**

**ctx.lineTo(x + width, y);**

**ctx.lineTo(x + width, y + height);**

**ctx.lineTo(x + radius, y + height);**

**ctx.arc(x+radius, y+radius, radius, Math.PI/2, 3\*Math.PI/2, false);**

**}**

**ctx.closePath();**

**ctx.fill();**

***// shadow on the right***

**if (width<max-1) {**

**ctx.save();**

**ctx.shadowOffsetX = 1;**

**ctx.shadowBlur = 1;**

**ctx.shadowColor = '#666';**

**if (width+radius>max)**

**offset = offset+1;**

**ctx.fillRect(x+width,y+offset,1,*total\_height*-offset\*2);**

**ctx.restore();**

**}**

**}**

**function *progressText*(ctx, x, y, width, height, radius, max) {**

**ctx.save();**

**ctx.fillStyle = 'white';**

**var text = Math.*floor*(width/max\*100)+"%";**

**var text\_width = ctx.measureText(text).width;**

**var text\_x = x+width-text\_width-radius/2;**

**if (width<=radius+text\_width) {**

**text\_x = x+radius/2;**

**}**

**ctx.fillText(text, text\_x, y+22);**

**ctx.restore();**

**}**

***// Define the size and position of indicator***

**var *i* = 0;**

**var *res* = 0;**

**var *context* = null;**

**var *total\_width* = 300;**

**var *total\_height* = 34;**

**var *initial\_x* = 20;**

**var *initial\_y* = 20;**

**var *radius* = *total\_height*/2;**

**window.onload = function() {**

***// Get the canvas element***

**var elem = document.getElementById('myCanvas');**

***// Check the canvas support with the help of browser***

**if (!elem || !elem.getContext) {**

**return;**

**}**

***context* = elem.getContext('2d');**

**if (!*context*) {**

**return;**

**}**

***// Text’s font of the progress***

***context*.font = "16px Verdana";**

***// Gradient of the progress***

**var progress\_lingrad = *context*.createLinearGradient(0,*initial\_y*+*total\_height*,0,0);**

**progress\_lingrad.addColorStop(0, '#4DA4F3');**

**progress\_lingrad.addColorStop(0.4, '#ADD9FF');**

**progress\_lingrad.addColorStop(1, '#9ED1FF');**

***context*.fillStyle = progress\_lingrad;**

***// Create the animation***

***res* = setInterval(*draw*, 50);**

**}**

**function *draw*() {**

***// augment the length on 1 for every iteration***

***i*+=1;**

***// Clear the layer***

***context*.clearRect(*initial\_x*-5,*initial\_y*-5,*total\_width*+15,*total\_height*+15);**

***progressLayerRect*(*context*, *initial\_x*, *initial\_y*, *total\_width*, *total\_height*, *radius*);**

***progressBarRect*(*context*, *initial\_x*, *initial\_y*, *i*, *total\_height*, *radius*, *total\_width*);**

***progressText*(*context*, *initial\_x*, *initial\_y*, *i*, *total\_height*, *radius*, *total\_width* );**

***// stop the animation when it reaches 100%***

**if (*i*>=*total\_width*) {**

**clearInterval(*res*);**

**}**

**}**

**function *roundInsetRect*(ctx, x, y, width, height, radius) {**

**ctx.beginPath();**

***// draw the outward rectangle in an anti-clockwise direction***

**ctx.moveTo(1000, 1000);**

**ctx.lineTo(1000, -1000);**

**ctx.lineTo(-1000, -1000);**

**ctx.lineTo(-1000, 1000);**

**ctx.lineTo(1000, 1000);**

**ctx.moveTo(x + radius, y);**

**ctx.lineTo(x + width - radius, y);**

**ctx.arc(x+width-radius, y+radius, radius, -Math.PI/2, Math.PI/2, false);**

**ctx.lineTo(x + radius, y + height);**

**ctx.arc(x+radius, y+radius, radius, Math.PI/2, 3\*Math.PI/2, false);**

**ctx.closePath();**

**ctx.fill();**

**}**

**function *progressLayerRect*(ctx, x, y, width, height, radius) {**

**ctx.save();**

***// define the shadows***

**ctx.shadowOffsetX = 2;**

**ctx.shadowOffsetY = 2;**

**ctx.shadowBlur = 5;**

**ctx.shadowColor = '#666';**

***// first gray layer***

**ctx.fillStyle = 'rgba(189,189,189,1)';**

***roundRect*(ctx, x, y, width, height, radius);**

**ctx.fillStyle = 'white';**

***roundInsetRect*(ctx, x, y, width, height, radius);**

**ctx.restore();**

**}**

**</script>**

**</body>**

**</html>**

**38. HTML5 - Create a Bar Graph with HTML5 Canvas and JavaScript ?**

**<!DOCTYPE html>**

**<html>**

**<head lang="en">**

**<meta charset="UTF-8">**

**<title></title>**

**</head>**

**<body>**

**<canvas id="graphSpace" width="800" height="400"></canvas>**

**<script>**

**var *graphCanvas* = document.getElementById('graphSpace');**

***// Ensure that the element is available within the DOM***

**if (*graphCanvas* && *graphCanvas*.getContext) {**

***// Open a 2D context within the canvas***

**var *context* = *graphCanvas*.getContext('2d');**

***// Bar chart data***

**var *data* = new *Array*(5);**

***data*[0] = "apples,200";**

***data*[1] = "oranges,120";**

***data*[2] = "bananas,80";**

***data*[3] = "kiwis,230";**

***data*[4] = "tangarines,340";**

***// Draw the bar chart***

***drawBarChart*(*context*, *data*, 50, 100, (*graphCanvas*.height - 20), 50);**

**}**

**function *drawBarChart*(context, data, startX, barWidth, chartHeight, markDataIncrementsIn) {**

***// Draw the x and y axes***

**context.lineWidth = "1.0";**

**var startY = 380;**

***drawLine*(context, startX, startY, startX, 30);**

***drawLine*(context, startX, startY, 570, startY);**

**context.lineWidth = "0.0";**

**var maxValue = 0;**

**for (var i=0; i < data.length; i++) {**

***// Extract the data***

**var values = data[i].split(",");**

**var name = values[0];**

**var height = *parseInt*(values[1]);**

**if (*parseInt*(height) > *parseInt*(maxValue)) maxValue = height;**

***// Write the data to the chart***

**context.fillStyle = "#b90000";**

***drawRectangle*(context,startX + (i \* barWidth) + i,(chartHeight - height),barWidth,height,true);**

***// Add the column title to the x-axis***

**context.textAlign = "left";**

**context.fillStyle = "#000";**

**context.fillText(name, startX + (i \* barWidth) + i, chartHeight + 10, 200);**

**}**

***// Add some data markers to the y-axis***

**var numMarkers = Math.*ceil*(maxValue / markDataIncrementsIn);**

**context.textAlign = "right";**

**context.fillStyle = "#000";**

**var markerValue = 0;**

**for (var i=0; i < numMarkers; i++) {**

**context.fillText(markerValue, (startX - 5), (chartHeight - markerValue), 50);**

**markerValue += markDataIncrementsIn;**

**}**

**}**

***// drawLine - draws a line on a canvas context from the start point to the end point***

**function *drawLine*(contextO, startx, starty, endx, endy) {**

**contextO.beginPath();**

**contextO.moveTo(startx, starty);**

**contextO.lineTo(endx, endy);**

**contextO.closePath();**

**contextO.stroke();**

**}**

***// drawRectangle - draws a rectangle on a canvas context using the dimensions specified***

**function *drawRectangle*(contextO, x, y, w, h, fill) {**

**contextO.beginPath();**

**contextO.rect(x, y, w, h);**

**contextO.closePath();**

**contextO.stroke();**

**if (fill) contextO.fill();**

**}**

**</script>**

**</body>**

**</html>**

**39. HTML5 - what is Content editable ?and Features of content editable in HTML5?**

**The contenteditable attribute is an enumerated attribute whose keywords are the empty string, true, and false. The empty string and the true keyword map to the true state. The false keyword maps to the false state. In addition, there is a third state, the inherit state, which is the missing value default (and the invalid value default).**

**40. HTML5 - Can you give an example of Canvas element how it can be used?**

<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">

Your browser does not support the HTML5 canvas tag.</canvas>

<script>

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

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

ctx.beginPath();

ctx.arc(95,50,40,0,2\*Math.PI);

ctx.stroke();

</script>

</body>

</html>

**41. HTML5 - write the Code for To Show the User?s Location ?**

<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.coords.latitude +

"<br>Longitude: " + position.coords.longitude;

}

</script>

**42. HTML5 - How to draw rectangle using Canvas and SVG using HTML 5 ?**

<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">

Your browser does not support the HTML5 canvas tag.

</canvas>

<script>

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

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

ctx.fillStyle = "#FF0000";

ctx.fillRect(0,0,150,75);

</script>

</body>

</html>

**43. HTML5 - So how do we implement application cache in HTML 5 ?**

**44. HTML5 - What is datalist in HTML 5? And What is feature detection and types of feature detection ?**

The Datalist element is used to suggest input values to the user, thereby providing an "autocomplete" feature on form elements. This is especially useful for long lists, such as countries or clothing manufacturers. Rather than scan through the entire list, the input control can suggest some items as soon as the user has typed in some characters. Thus, it behaves as a sort of combobox, possessing both a textbox and list component.

<label>Choose a browser from this list:

<input list="browsers" name="myBrowser" /></label>

<datalist id="browsers">

<option value="Chrome">

<option value="Firefox">

<option value="Internet Explorer">

<option value="Opera">

<option value="Safari">

<option value="Microsoft Edge">

</datalist>

**45. HTML5 - What is the difference between HTMl5 Application cache and regular HTML browser cache?**

**46. HTML5 - WHAT ARE THE NEW APIS PROVIDED BY THE HTML 5 STANDARD? GIVE A BRIEF DESCRIPTION OF EACH?**

01. Media API

The media API is part of the media element which includes two of HTML5's poster children, the video and audio elements. The elements themselves are simple to implement but what's less well known are the JavaScript methods available within the associated API. There are a number of methods including play() and pause() as well as load() and canPlayType(). Many of the methods are shared between both media types with a subset of additional properties (eg poster) specifically related to the video element. Combined with additional events and attributes the API allows us to, amongst other things, create custom controls.

02. Text Track API

The text track API leads on nicely from the media API. It is designed to allow us to interact with text tracks (subtitles or captions for example) for the audio and video elements.

You can return the number of text tracks and their length associated with a media element, the kind of text track (subtitles, captions, descriptions, chapters and metadata), language, readyState, mode and label.

This API will have far more support when browsers begin to implement native subtitling, using WebVTT for example.

03. Drag and Drop

It brings native drag and drop support to the browser. By adding a draggable attribute set to true, the user has the ability to move any element. You then add some event handlers on a target drop zone to tell the browser where the element can be dropped.

The API's real muscles are flexed when you start to think outside of the browser. Using drag and drop, a user could drag an image from the desktop into the browser or you could create an icon that gets loaded with content when dragged out of the browser by the user to a new application target.

04. Offline Web Applications/Application Cache

Application caching is carried out by creating a simple manifest file which lists the files that are required for the application to work offline. Authors can then ensure their sites function offline. The manifest causes the user’s browser to keep a copy of the files for use offline later. When a user views the document/application without network access, the browser switches to use the local copies instead. So in theory, you should be able to finish writing that important email or playing the web version of Angry Birds while you're on the underground/subway.

05. User Interaction

Like offline, user interaction is part of the primary HTML5 specification. It's worth mentioning here because some of its features, such as the contenteditable attribute, are extremely useful when you're creating web applications. contenteditable has been around in internet Explorer since version 5.5 and works in all five major browsers. Setting the attribute to true indicates that the element is editable. Authors could then, for example, combine this with local storage to track changes to documents.

06. History

The pre-HTML5 History API allowed us to send users forward or back, and check the length of the history. What HTML5 brings to the party are ways to add and remove entries in the user's history, hold data to restore a page state and update the URL without refreshing the page. The scripting is fairly straightforward and will help us build complex applications that don't refresh the page from which we can continue to share URLs as we've always done.

07. MIME type and protocol handler registration

This API allows sites to register themselves as handlers for certain schemes. By using the registerProtocolHandler method, an example use case could be:

an online telephone messaging service could register itself as a handler of the sms: scheme, so that if the user clicks on such a link, he is given the opportunity to use that Web site (W3C HTML Spec)

Certain schemes are whitelisted such as sms, tel and irc. In addition there is a registerContentHandler method that allows sites to register as handlers for content with a certain mime type.

Canvas 2D Context — allows us draw natively in the browser. Using canvas without the 2D Context API we wouldn't be able to draw. It's our brushes, palette and paint all rolled into one.

Cross document and channel messaging — cross document messaging defines a way for documents to communicate with one-another regardless of their source domain without enabling cross-site attacks. In a similar vein, channel messaging uses independent pieces of code to communicate directly.

Web Workers — an API for running JavaScript in the background independent of any user scripts. Allows for long running tasks to be completed without preventing the page from becoming unresponsive.

**47. HTML5 - How do you play a Video using HTML5? With an example ?**

<video width="320" height="240" controls>

<source src="movie.mp4" type="video/mp4">

<source src="movie.ogg" type="video/ogg">

Your browser does not support the video tag.

</video>

**48. HTML5 - What purpose does HTML5 serve?**

HTML5 is the proposed next standard for HTML 4.01, XHTML 1.0 and DOM Level 2 HTML. It aims to reduce the need for proprietary plug-in-based rich internet application (RIA) technologies such as Adobe Flash, Microsoft Silver light, Apache Pivot, and Sun JavaFX.

**49. HTML5 - Do you know What is the sessionStorage Object in html5? How to create and access?**

The sessionStorage object exists as a property of the window object in supporting browsers (currently Firefox 3+, Safari 4+, and Internet Explorer 8+). You can place data onto the sessionStorage object and that data persists for as long as that window (or tab) is open. Even if you navigate away from the page that stored the data and then navigate back, the data saved to sessionStorage is still available. Any data stored in sessionStorage is tied to the protocol, hostname, and port of the page that saved the information and only pages sharing the same protocol, hostname, and port can access that data later.

Making things more interesting, sessionStorage is unique to a particular window or tab (what the specification refers to as a “top-level browsing context”).

Data stored to sessionStorage is saved in key-value pairs where both the key and the value are strings. Non-string values are automatically converted into strings before being stored.

The data in sessionStorage is deleted once the window or tab is closed, or if the user requests that the browser do so. This behavior, combined with tying the data to a particular window or tab, ensures that data doesn’t get accidentally exposed or stored indefinitely.

Usage

The sessionStorage object has five methods:

getItem(key) – retrieves the value for the given key or null if the key doesn’t exist.

setItem(key, value) – sets the value for the given key.

removeItem(key) – removes the key completely.

key(position) – returns the key for the value in the given numeric position.

clear() – removes all key-value pairs.

There is also a single property, length, which indicates how many key-value pairs are currently stored in sessionStorage.

Whenever a change is made to sessionStorage, a storage event is fired on the document object. The event object for this event contains the following properties:

key – the key that was changed.

oldValue – the value before the operation.

newValue – the value after the operation.

url – the URL of the page that performed the operation.

source – the window object representing the owner of the sessionStorage object.

The specification is unclear as to whether this event should be fired for sessionStorage or not. My testing shows that Internet Explorer fires the event for sessionStorage but Firefox and Safari do not. If anyone has other details, please chime in.

The reason I really like sessionStorage is that it keeps security in mind. By limiting data access to a single window or tab, tying that data to the protocol, domain, and port, and then deleting the data when the window or tab is closed, the implementation really makes sure that data can’t be accessed in harmful ways.

**50. HTML5 - Explain What is the use of localStorage in HTML5?**

HTML5 local storage makes it possible to store values in the browser which can survive the browser session.

HTML5 local storage is similar to cookies in that both mechanisms can be used to store data in the browser between HTTP requests. But there is a difference between HTML5 local storage and cookies.

Cookies are small pieces of data which a server can store in the browser. The cookie is sent by the browser along with all future HTTP requests to the server that set the cookie. Cookies cannot be bigger than 4KB in total.

HTML5 local storage is set via JavaScript executed in the browser. HTML5 local storage properties are never sent to any server - unless you explicitly copy them out of the local storage and appends them to an AJAX request. HTML5 local storage can store somewhere between 2MB and 10MB data in the browser (per origin - domain name). Exactly how much data is allowed depends on the browser. A limit of 5MB to 10MB is most common.

**51. HTML5 - Explain How many New Markup Elements you know in HTML5**

Below are the New Markup Elements added in HTML5

Tag Description

<article>:Specifies independent, self-contained content, could be a news-article, blog post, forum post, or other articles which can be distributed independently from the rest of the site.

<aside> For content aside from the content it is placed in. The aside content should be related to the surrounding content

<bdi>:For text that should not be bound to the text-direction of its parent elements

<command>:A button, or a radiobutton, or a checkbox

<details>:For describing details about a document, or parts of a document

<summary>:A caption, or summary, inside the details element

<figure>:For grouping a section of stand-alone content, could be a video

<figcaption>:The caption of the figure section

<footer>:For a footer of a document or section, could include the name of the author, the date of the document, contact information, or copyright information

<header>:For an introduction of a document or section, could include navigation

<hgroup>:For a section of headings, using <h1> to <h6>, where the largest is the main heading of the section, and the others are sub-headings

<mark>:For text that should be highlighted

<meter>:For a measurement, used only if the maximum and minimum values are known

<nav>:For a section of navigation

<progress>:The state of a work in progress

<ruby>:For ruby annotation (Chinese notes or characters)

<rt>:For explanation of the ruby annotation

<rp>:What to show browsers that do not support the ruby element

<section>:For a section in a document. Such as chapters, headers, footers, or any other sections of the document

<time>:For defining a time or a date, or both

<wbr>:Word break. For defining a line-break opportunity.

**52. HTML5 - Can a web page contain multiple elements or multiple elements?**

Yes. A web page needs a minimum of 4 tags - <html>, <head>, <title>, <body>

**53. HTML5 - What is the purpose of HTML5 versus XHTML?**

XHTML stands for Extensible Hyper Text Markup Language. It is basically a web document standard that came into being almost a decade back soon after the HTML4 was standardized in 1997. In short it is a combination of HTML4 with elements of XML (Extensible Markup Language) 1.0. XHTML was developed to bypass the loopholes HTML4 that was known to generate lenient and sloppy syntax. XHTML was developed to make HTML more extensible and increase interoperability with other data formats. HTML4 was basically an application of Standard Generalized Markup Language (SGML). SGML was complex, thus web browsers and other parsers weren’t fully conformant to it. To overcome these strict XML standards were incorporated into XHTML. This allowed documents to be parsed by existing XML parsers allowing web browsers to easily understand and represent the data.

HTML5 or Hyper Text Markup Language 5 is the latest version of the famous markup language that is taking web development to the next level. This language has been under constant development since its introduction a few years ago and is expected to be finalized in 2014. It brought in the much needed developments in the legendary language which was failing to meet the expectations of the web developers as its predecessor was developed more than a decade back. HTML5 comes with many new features that allow developers to incorporate richness to their website. It is an advanced version of HTML4 markup and also has some of the features of XHTML 1.0.

If we compare HTML5 with its predecessor we will see that it brings in many new tags such as those used for menus, headers, navigation and other elements of the website. Apart from this HTML5 has introduced native support for Audio, Video and SVG images thus allowing web developers to create multimedia based websites without having to depend on technologies like Flash. It has also made websites faster and mobile friendly which is the needs of the hour. Developers are making use of this markup language along with CSS3 to create robust web applications.

HTML5 and XHTML 1.0

Ease of Use – XHTML is very easy to learn and apply on the websites compared to HTML5 which has a steeper learning curve. Most developers know XHTML like the palm of their hand while it takes time for a developer to get used to HTML5.

Basic Website – There is always a great demand for basic websites which purely display information. For such sites that are seen on the computer and laptop screens XHTML is a better language as it takes less time to create such a website.

Web 2.0 – This is one environment where social media and multimedia elements are expected to rule the market. HTML5 has been designed to take web development to the new level. XHTML 1.0 is a decade old technology and hence was never created keeping this kind of web development challenges in mind.

Device Support –Creating a website that is supported across computers, smartphones and tablets is very important in a multi-device world and this is where HTML5 comes good. It allows you to create websites that reach out to the masses. XHTML 1.0 on the other hand was never conceptualized to work in such an environment.

Browser Support – When we factor browser support HTML5 emerges as the clear winner as almost all the web and mobile browsers support this markup language. You might face some issues with the older versions of the browsers but such problem is miniscule. Though XHTML also enjoys browser support it does create some issues with mobile browsers.