HTML 5

Introduction to HTML

HTML5 is the new and improved version of HTML4.

HTML5 now supports many inbuilt features which in the past were achieved using JavaScript and flash.

HTML5 has new tags and input types for faster application development.

Html5 has inbuilt support for data storage which enables storing data at front end.

Html5 exposes many functionalities like geolocation etc with the help of API functions that can be called using JavaScript.

Html5 is no longer a presentation language.

It's now a complete application development platform for rich and dynamic applications

New Features of Html5

Symantic Tags: Html5 comes up with new symantic tags that help in organizing the content.

Example is **<nav> tag** that is used to enclose the navigation code. Note that the new symantic tags only attach meaning to tags, they don't provide any new behavior.

New Input Types: Html5 supports new input types for forms. The new input types come up with new features for better user experience.

Example: the input type email now supports inbuilt validation for email types.

Support for video and Audio: Html5 now has inbuilt support for video and audio. We no longer need flash to play videos and audios.

Database support: Html5 has inbuilt database to store data. The different types of database support enables developers to now store data like files, image along with relational database support.

Device Interactions: Html5 can now interact with the host device like battery API and Geolocation API through JavaScript.

Animation and Drawing support: Html5 has inbuilt support for drawing images using canvas API and SVG. This helps to create images dynamically using JavaScript.

Beginning with HTML5

Doctype: We no longer need the long Doctype Declaration. For declaring a document for html5, use the following doctype

```
<!DOCTYPE HTML>
```

Meta Charset: For meta declaration for charset we no longer need the older long version in html4. Just as reminder below is the html4 version of meta tag for charset

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">.
```

For html5 simply use <meta charset = “UTF8”>.

Declaring styles and scripts

Styles declaration. The Style can be linked using <link rel="stylesheet" href="style.css">. We no longer need to specify the type for html5.

Script Declaration To link a script just we need use below format of html5. The type is no longer required.

```
<script src="page.js"></script>
```

Deprecated Tags in HTML5

Following are the tags which should not be used in HTML5.

<acronym>	Previously Used For an acronym
<applet>	Previously Used For an applet
<basefont>	Previously Used For base font
<big>	Previously Used For big text
<center>	Previously Used For centered text

<dir>	Previously Used For a directory list
<font>	Previously Used For text font, size, and color
<frame>	Previously Used For a frame
<frameset>	Previously Used For frames
<isindex>	Previously Used For a single-line input field
<noframes>	Previously Used For a noframe section
<s>	Previously Used For strikethrough text
<strike>	Previously Used For strikethrough text
<tt>	Previously Used For teletype text
<u>	Previously Used For underlined text

Deprecated Attributes in HTML5

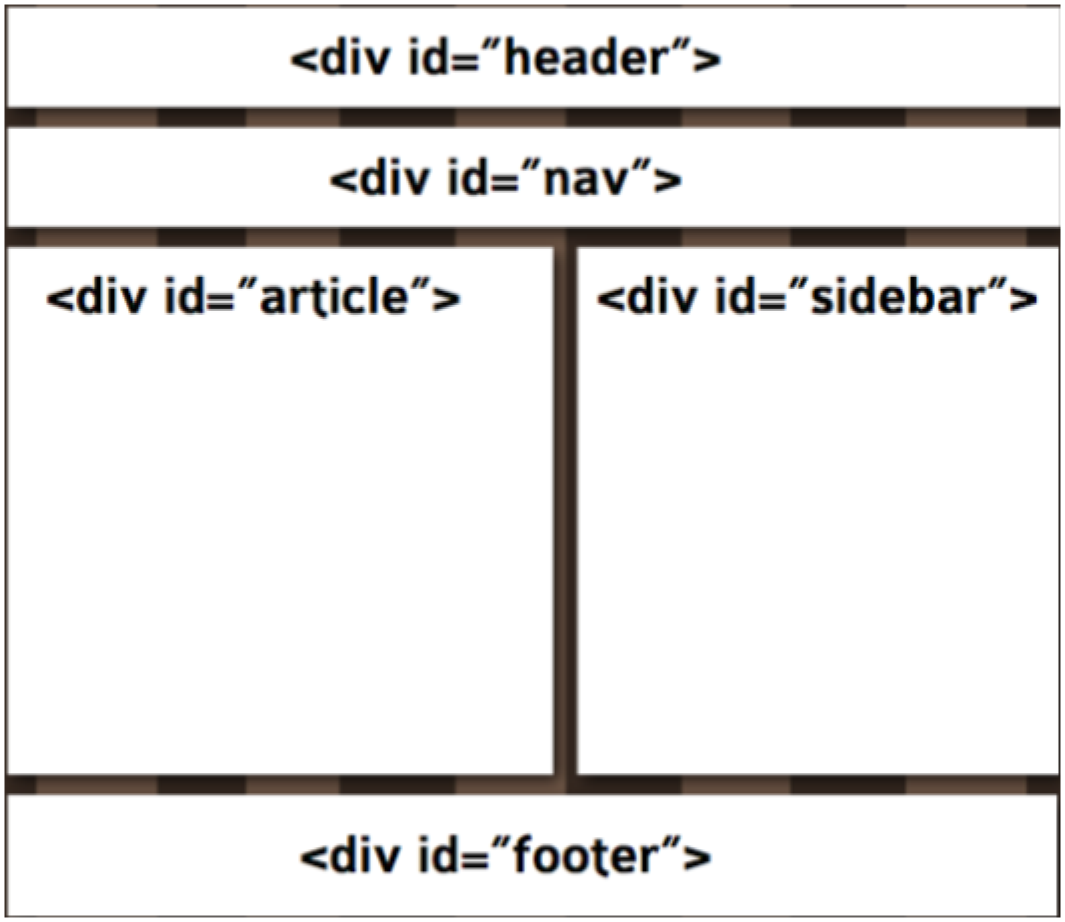
Following are the attributes which should not be used in HTML5.

rules, scrolling, size, type, valign, width, scheme, archive, align, alink, link, vlink, revcharset, shape, coords, longdesc, target, nohref, profile, version, name, scope, text, background bgcolor, border, cellpadding, cellspacing, char, charoff, clear, compact, frame, frameborder hspace, vspace, marginheight, marginwidth, noshade, nowrap, classid, codebase, codetype, declare, standby, valuetype, type, axis, abbr.

HTML5 Symantic Tags

Layouts Without Semantic tags :

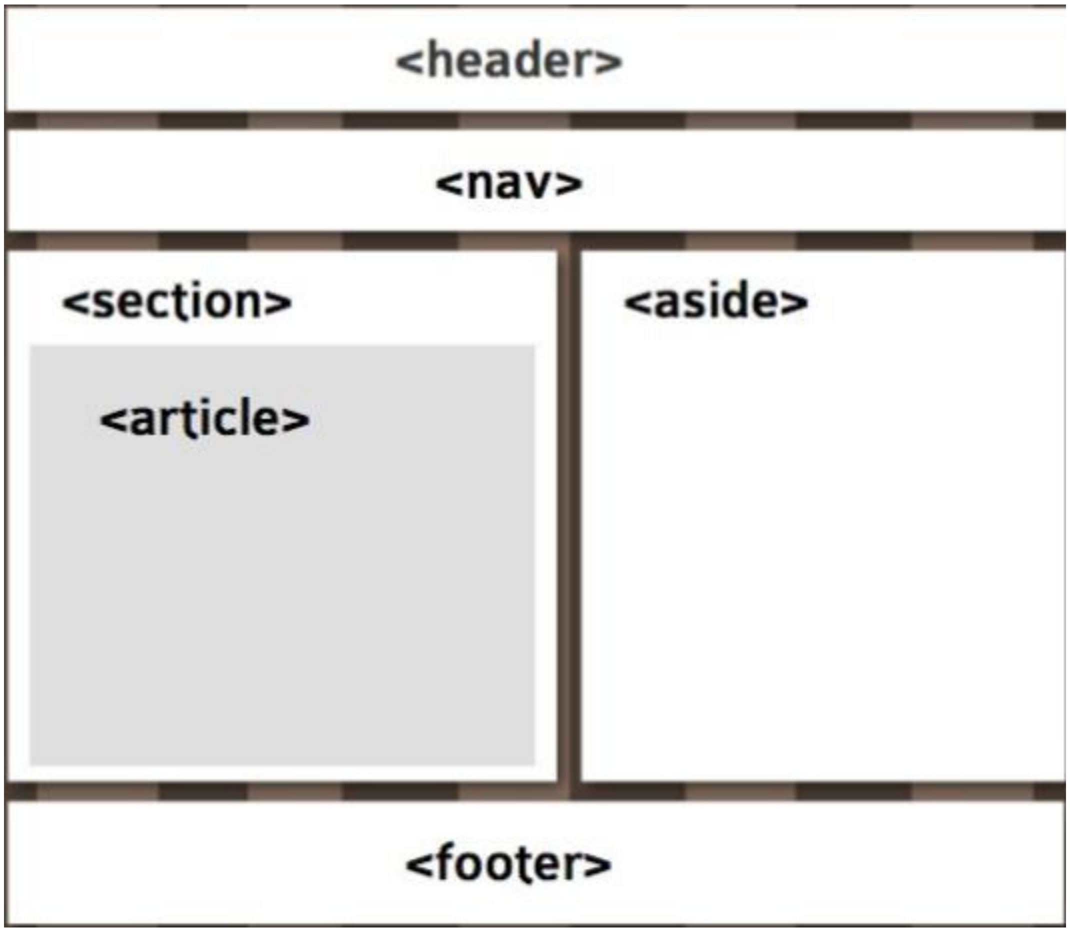
Let's consider an example where we are creating a website with the layout as shown below. As usual we will like to classify the div containers based on either IDs or classes as shown below. The naming convention looks good but is no standard and can't be used to classify the containers.



HTML 5

Layouts with HTML5 Semantic tags

Lets create the same layout using semantic tags. Let's see how the example looks now. As shown below, now the div containers have a semantic tags.



Section tag : The section tag is a block level tag. It is used to group similar contents. If there is a news website then the section tag can be used to group different types of news together. For example, top news can come under section. Similarly sports news can also be grouped inside the section tags. Below is the example usage.

Article tag : The article tag is used to organize independent content. Continuing the previous example of news site, a single news is an independent content and is ideal for organizing within article tags. For a product display site, a product is an independent content which again can be organized inside an article tag. Below is the example usage

```
<section>

        <article>
```

```
                    <h3> Lets save the environment</h3>

                    <p> ……..News goes here.......... </p>

          </article> ….Other news items written..

</section>
```

Header tag : The header tag is used to set the header details like navigation for the page, article or section. It is the head section of a container. Ex: For the page, we can keep the navigation inside the header tag. For an article the header can contain the heading part of an article.

Example: the heading for a news item can be kept within the header tag. Note that there can be multiple header tags within a page.

```
<section>

          </header>    Sports new Section  </header>

</section> ..
```

Footer tag : The Footer tag is used to organize the footer or ending area of a parent container. Similar to header , we can have multiple footer tags. A footer for an article tag that contains a news item can be used to provide the author details which is usually at the end section. For a layout, a footer tag contains the footer information like copyright and other related data.

Example:

```
<footer> news by ..Author.. </footer>
```

nav tag : The nav tag is used to represent the navigation links for the page, article or section. If used for the page layout <nav> tag can be used to contain the menu items. nav can be used to represent the menu items for a particular section or even article if required.

```
<nav> ...the menu items go here </nav>
```

main tag : <main> tag represents the main content of the document or application. The main content area consists of content that is directly related to, or expands upon the central topic of a document. The entire page can not contain more than one main tag. <main> tag must not be inside of an <article>, <aside>, <footer>, <header>, or <nav> element.

Example:

```
<body>
```

```
        <nav>    </nav>

        <main>   ...main content to be written inside  </main>

    </body>
```

figure tag : A figure tag is used to store an image, an illustration, a diagram, a code snippet, or a schema that is referenced in the main text. A caption can be associated with the <figure> element using a <figcaption> inside the <figure> tag.

Example:

```
    <figure>

        <img src="test.png" alt="test image">

        <figcaption>    Caption for the test image  </figcaption>

     </figure>
```

Form Handling and New Input Types

Html5 comes up with a fantastic support for user input types. A lot of new input types are available for better user experience. Prior to Html5 we used to use JavaScript for basic form validation. Now we don't need to use JavaScript for basic form validation. Most of the validations are now inbuilt.

input type = “email”

Input type = “email” has 2 functionalities.

1)For mobile devices, when the user types in the email type field, the keyboard layout changes accordingly. Ex in safari the keyboard input displays the dot(.) and @ symbol for easy input.

2)It also validates if the entered value is a valid email. If the value is not a valid email, the form doesn't submit and displays an error message Usage: <input type = “email”/>

input type = “url”

Input type = “url” has 2 functionalities.

1)For mobile devices, when the user types in the url type field, the keyboard layout changes accordingly . Ex in safari the keyboard input displays the dot(.) and com symbol for easy input.

2)It also validates if the entered value is a valid url. If the value is not a valid email, the form doesn't submit and displays an error message Usage: <input type = “url” />

input type = “date”

Input type = “date” displays a calendar input in the page. This helps user input the date with a calendar input. Prior to html5 we had to use jquery plug-in to display a calendar.

1) Again on mobile devices, when the user types in the calendar type field, the keyboard layout changes accordingly to date input We can specify min and max values as attributes for the date type also

Example: <input min="2012-01-01" max="2013-01-01" type="date" />

As of now, Feb 2015, input type = date is only supported in chrome 5.0 onwards, opera 10.62 onwards properly. In Firefox, it's not working currently due to a bug.

input type = “month”

This is a variant of date type. Instead of the full date, the month is displayed.

Example: <input id="expiry" name="expiry" type="month" />

input type = “week”

This is a variant of date type. If we want to display the user to only input week instead of month we can use it. Below is the example usage

```
<input  name="holiday” type="week">
```

input type = “time”

This is a variant of date type. If we want to display the user to only input time we can use this. Below is the example usage

```
<input id="exit-time" name="exittime" type="time">

<input  name="meeting-time” type="time" />
```

input type = “datetime”

This is a variant of date type. This gives the control for entering a date and time in hour, minute, second, and second format based on UTC time zone.

Example: <input name="leaving-time" type="datetime" />

input type = “color”

This input type allows the user to select a color and returns the hex value for that color

Example: <input id="color" name="color" type="color">

Note that the color input look varies in different browsers.

input type = “tel”

This input type doesn't validate if the field is a telephone no or number. It is useful when used for mobile sites. Mobile OS are able to indentify this field input and display

Ex: <input name="telephone" type="tel" />

input type = “number”

This input type number is used to input a number value. This field also has validation inbuilt. If the user enters a value other than number, the validation fails and the form is not submitted. It is useful when used for mobile sites. Mobile OS are able to identify this field input and display the keyboard input accordingly.

Ex: <input type = "number" name = "num">

input type = “number”

This input type number is used to input a number value. This field also has validation inbuilt. If the user enters a value other than number, the validation fails and the form is not submitted. It is useful when used for mobile sites. Mobile OS are able to identify this field input and display the keyboard input accordingly.

Ex: <input type = "number" name = "num">

input type = “range”

This input type provides a slider with predefined values to input. For example if you want to enter the user his age, you can use the slider. You can set the min and max value for the age.

Ex: <input type = "range" min = 40 max = 140 name = "num">

Note that the problem with slider is that it doesn't display the value that the user enters. For this you can use bit of JavaScript and other input type called output to achieve the desired result. Below is the code.

```
<form oninput="display.value=rangeval.value" action="bla.php" method="GET">

<input type="range" value=0 id="rangeInput" name="rangeval" min="20" max="100">

 <output name="display" for="rangeInput">20</output>

 <input type = "submit" value = "submit">
```

input type = “search”

This input type provides a text box for entering text search values. Chrome displays a cross symbol to clear the items entered. For Mobile, the user's keyboard layout changes and shows a search button with supported platforms.

Note: Currently(Feb2015) this input type just appears as plain text field in mozilla

Ex: <input type = "search" name = “search-text” >

DataList

Datalist shows a list of pre-defined options to suggest to the user Have a look at the example below.

Datalist provides the values list to attach to the input type.

Note that input type can have a type like email in case we want to show a list of email values.

Example:

```
<input list="hobbies"/>

<datalist id="hobbies">

        <option value="chess">

        <option value="coding">

        <option value="travelling">

        <option value="skiing">

        <option value="Cricket">

</datalist>
```

progress

This element is used to view the completion progress of a task. The output is displayed as a progress bar. Note that the output of the progress bar . The formatting of progress bar is different in different browser.

Example usage: <progress value="80" max="100">80 %</progress>

Form Handling in HTML5

Now we know how to create new input types and use them. Let's explore additional form handling features that html5 provides.

novalidate : Validation for input type email, number now comes inbuilt. If you want to turn off the html5 validation use the novalidate attribute for a form. Using novalidate the html5 validation is turned off.

Example: <form action = “register.php” novalidate> ..form elements here ...</form>

autocomplete = on/off Browsers remember the values entered in the forms. It is a useful features but at times we don't want browser to remember the values entered in the form. For this feature we can use the autocomplete attribute. Setting autocomplete value as off will force the browser not to display the previously entered values.

Example: <form action = “register.php” autcomplete = “off” > ..form elements here ...</form>

required : If we want an element within a form as required, we can give required attribute to the element. This will enable the form validation and html5 will check if the value is entered by the user. On submit, if the value is not entered, an error will be thrown to the user. In the example below in from for the text field named first-name will throw error if the value is empty.

Example:

<form action = “register.php” >

<input type = “text” name = “first-name” required> ...</form>

autofocus : The attribute autofocus is an attribute of form elements. The element with this attribute get the focus automatically when the form loads.

Example:

<form action = “register.php” >

<input type = “text” name = “first-name” autofocus > ...</form>

placeholder : The placeholder attribute provides the text that will appear within the input element. Previously developers used to use JavaScript to achieve this functionality.

Example:

<form action = “register.php” >

<input type = “text” name = “first-name” placeholder = “Enter First name” > ...</form>

pattern attribute

The pattern attribute has a value of a JavaScript regular expression. It is used to validate a form field to match the pattern provided as attribute The pattern in a very powerful feature. We always encounter requirements where we have to validate the form based on different business requirements. Without pattern attribute we will have use custom JavaScript functions or jquery plugins.

See the example below for validation for a telephone field The patter is a JavaScript regular expression. It will only submit the value if the number is a 10 digit number .

```
<input type="tel" name = “phone-number”  pattern="[0-9]{10}" />
```

Audio Video Support in HTML5

HTML5 has a fantastic support for video and audio. Prior to html5 developers used to use flash to display video and audio in webpages. html5 now comes with its own tags to display audio and video. Since audio and video are now displayed in tags, using JavaScript and DOM we can do a lot more operations very easily

Video tag : html5 video tag is used to display videos. Below is the example.

We can specify the source of the video inside the source tag . Note that we can specify multiple sources. If browser is able to play the video, it plays it and ignores the rest source tags. We need to specify multiple source tags because different browsers support different types of video format. Hence we have to give multiple video formats for different browsers. The video tag optional attribute type to specify the video format. Though optional, we should use it as it makes the video display faster. Without this attribute the browser has to do the task of finding out the video that plays.

control attribute: the control attribute is used to display the controls in the video (ie play, pause, volume etc). With type we can specify the codecs if the video needs codecs to play.

```
<video controls>

<source src="junglebook.webm" type='video/webm;codecs="vp8, vorbis"'/>

<source src="junglebook.mp4" type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"'/>

</video>
```

The above example will play video if browser supports web format and vp8 and vorbis codecs installed, else will move to the next video. Always include the type of video format else browser keeps on loading the files which one it can play. This slowdowns the performance.

Media Fragments : In video source tag we can also specify the times in hours:minutes:seconds, such as #t=00:05:05 to start the video at Five minute, five seconds. With this we can decide the time range to play the video.

Note: its important to see that sure Range Requests are supported by the server:

Ex <source src = "junglebook.mp4#t=5,10" type = "video/mp4" >

Poster Attribute : Poster attribute has URL indicating a frame to show until the user plays the video. If poster attribute isn't specified, the first available frame is shown.

Ex: <video width="700" height="460"poster="/img/poster.gif">

Displaying Subtitles : We can specify the tracks also inside the video tag. For displaying subtitles we use the <track> tag. See the example for the track tag.

The srclang attribute specifies the language. We can specify multiple track tags for multiple languages. The src attribute specifies the source of the tracks file that contains the subtitles.

```
<video controls>

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

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

<track src="deloitte-en.vtt" label="English subtitles" kind="subtitles" srclang="en"
default> </track>

No Support for video tag </video>
```

For displaying subtitles, we need to create a file in WebVTT format. It's simple.

Create a file with .vtt extension and keep it in the server. Below is an example of webvtt file for displaying subtitles

Ex:

```
WEBVTT FILE

1

00:00:00.500 --> 00:00:03.000 D:vertical A:start

Some text

2

00:00:03.500 --> 00:00:05.000

Other text

3 00:00:05.000 --> 00:00:07.000

Another text
```

Attributes for video tag : The video tag has attributes which add new features for playing the video. Below are the attributes

autoplay : To immediately start downloading the video and play it as soon as it can. This property behavior may be differ for mobile browsers.

preload: The preload attribute has three values none, metadata and auto

none: This value means do not download video and related metadata.

metadata: This value means download the meta data like dimensions, first frame, track list, duration, etc so that the user may later watch the video.

auto: means that download the entire video .

loop(Binary) : This attribute tells the browser to keep playing in loop

Muted (Binary): This attribute tells the browser the mute the video by default.

Methods video DOM Object

The video tag DOM object has methods that further help us in changing video properties. Note that we need to get reference to the video element to run these methods.

load(): This function Loads the video and resets the play head to the beginning of the video.

Play(): Plays the video from the current location pause() Pauses the video at the current playing location canPlayType

(video-format): This method tests whether the browser can play a specific type of video. Following are the return values

probably -most likely the video file can be played

maybe - The video might be playable

[empty string] – Indicates that the video file is cannot be played

addTextTrack() : Adds a new text track to the playing audio/video

Events on video DOM Object

The video tag, DOM object comes up with events to respond to video events. Following are the available events.

canplaythrough: This event is fired when enough data is available that the browser believes it can play the video completely without interruption.

ended: Fired when the video has finished playing

error: Fired if an error occurs

playing: Fired when the video starts playing after it was paused or when restarting

progress: This event is fired periodically to indicate the progress of downloading the video

waiting : Fired when an action is delayed pending the completion of another action

loadedmetadata: This even is fired when the browser has finished loading the metadata for the video and all attributes have been populated.

GeoLocation API

As the name suggests, the GeoLocation API provides set of functions to get the user location details. To note that GeoLocation works for mobiles, tablets as well as PC! Yes..it even works for comp. As the name suggests, The GeoLocation API provides set of functions to get the user location details.

Below are the browsers that support it

Opera 10.60+ Internet Explorer 9.0+ Firefox 3.5+ Chrome 5.0+ Safari 5.0+

For Mobile Browsers

Opera Mobile 10.1+ Symbian (S60 3rd & 5th generation) Blackberry OS 6 Android 2.0+ iPhone 3.0+

How GeoLocation Works for computers

Lets divide it into 2 parts,

1. First for PC

If your computer has internet connectivity via wifi, then wifi device WiFi readings are sent to various web services which convert them into latitude and longitude and return us the data. Note that different browsers use different types of web services. Mozilla and chorome use the google web service to get the address details. If your computer is not connected via Wifi, then IP address details are sent and approximate location is sent via IP address mapping to location database.

2. **Geolocation works for Mobiles**

For Mobiles variety of techniques are implemented GPS. This technique works by detecting the signal from a number of satellites present in the sky; Usually four or more satellites. Using a mathematical technique called trilateration the device is able determine its location based on the timing of the satellite signals. The main issue with this approach is that it takes time.

A-GPS: (Assisted GPSs): This approach is better than GPS. It gets address by augmenting the GPS receiver's capabilities by supplying data from the service-provider network which helps to locate the device. Wi-Fi based positioning: By using Wi-Fi signals with GPS and other location data, details to the device's location is provided.

Cell Triangulation: This technique uses an approximate location that is calculated by using information about the location of the cell towers that a mobile device is connected to at any time.

Using the API, we need to check if our browser actually supports GeoLocation. The API provides a function for this which can be called as follows:

if (navigator.geolocation) { // means it works }

If navigator.geolocation returns false,

then we have to tell user that browser doesn't support it.

getCurrentPosition() : This methods return immediately, and then asynchronously attempt to obtain the current location. The getCurrentPosition function gets the user location one time.

It takes two arguments in the form of callbacks: one for a successful location query and one for a failed location query. The success callback takes a Position object as an argument. It optionally takes a third argument in the form of a PositionOptions object.

Example:

```
navigator.geolocation.getCurrentPosition(locationSuccess, locationFail);

function locationSuccess(position) {

        var latitude = position.coords.latitude;

        var longitude = position.coords.longitude;

        var altitude = position.coord.altitude;

}

function locationFail() {

        alert('Oops, could not find you.');

}
```

Position Object Properties

coords. latitude	Decimal degrees of the latitude
coords.longitude	Decimal degrees for longitude
coords.altitude	Height(metres) of the position above the reference.
coords.accuracy	The property gives the accuracy in metres of the returned result.
coords.altitude	Accuracy Accuracy in metres of the returned altitude
coords.heading	Provides the direction of travel of the hosting device, clockwise from true north
coords.speed	The current device speed in metres per second
timestamp	Timestamp of when the position was acquired.

GeoLocation Data Format

We have 2 data formats, **Goedetic and Civic**.

The geodetic way of describing position refers directly to latitude and longitude The civic representation of location data is a human readable. This data shows the address in a proper format. Ie.. like your postal address

Geolocation API Example

```
navigator.geolocation.getCurrentPosition(

processGeolocation,    // Optional settings below

geolocationError,    {

        timeout: 0,

        enableHighAccuracy: true,

        maximumAge: Infinity
} );

//For tracking users position

watchId = navigator.geolocation.watchPosition( processGeolocation, geolocationError,
{

        timeout: 0,

        enableHighAccuracy: true,

        maximumAge: Infinity

} );
```

Enabling HighAccuracy

To Enable high Accuracy for getting location on a phone, we use the **enableHIghAccuracy** property value as true. This setting gives accuracy value with 3 metres. Typical time taken in within 10 to 35 seconds.

Note that appropriate permissions should be granted by the user for it. We can give the follwing values with the enableHighAccuracy Mode.

timeout(milliseconds) – this indicates the maximum length of time to wait for a response. Giving a value of 0 means infinite.

maximumAge(milliseconds) – This value denotes the maximum age of a cached position that the application will accept. The default value is 0, which means an attempt must be made to obtain new position object immediately.

Canvas API

What is canvas?

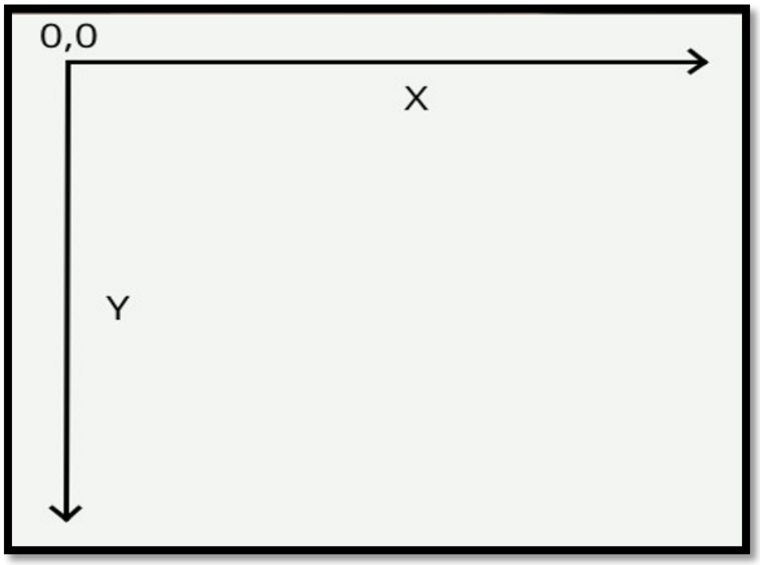
Canvas is a drawing API provided in HTML5 . It was originally created by Apple for its Dashboard widgets. Canvas is used for drawing pixels. Note that It just draws pixels to the screen. Let's have a look at simple example using canvas API.

Example:

```
<html>

        <body>

         <canvas width="500" height="300" id="canvas"></canvas>

        <script> var canvas = document.getElementById('canvas');

                var c = canvas.getContext('2d');

                c.fillStyle = "Blue";

                c.fillRect(50,50,400,400);

        </script>

        </body>

</html>
```

Understanding the coordinate system for browsers

The coordinate system for browsers is different than the normal Cartesian coordinate system. It starts from the top left. The y axis goes from top to down and x axis starts from left to right . Below is the diagram showing the web coordinate system.



Drawing shapes using canvas

Let's see how we can draw different types of shapes using the canvas API

```
<body>

<canvas width="600" height="400" id="canvas"></canvas>

<script> var canvas = document.getElementById('canvas');

        var c = canvas.getContext('2d');

        c.fillStyle = ' #8A2BE2f';

        c.beginPath();

        c.moveTo(30,20);

        c.lineTo(200,50);

        c.lineTo(126,90);

        c.closePath();

        c.fill();

         c.strokeStyle = 'rgb(0,128,0)';

        c.lineWidth = 5;

        c.stroke();

</script>

 </body>
```

Drawing Images using canvas

Canvas can draw images with the **drawImage** function. With **drawImage** we can stretch, slice or scale the images too. See the example below. Here ctx is the canvas context ctx.drawImage(img, 0,0);

```
//Stretching  image

ctx.drawImage(img,    0,0,100,100,    200,0,200,200    );
ctx.drawImage(img, //draw a slice

5,5,20,20, //source coords (x,y,w,h)

250,0,250,70//destination coords (x,y,w,h)    );
```

Drawing text with Canvas API

We can draw texts within the canvas using the **fillText** function.

See the example code for usage below

```
        var ctx = document.getElementById('canvas').getContext('2d');

        ctx.fillStyle = "black";

        ctx.font = "bold "+56+"pt Arial ";

        ctx.fillText("I miss the Childhood Days.!", 20,150);
```

Storage API in HTML5

HTML5 comes up with inbuilt storage capabilities that further makes html5 a complete application development environment. Prior to html5 developers used to use cookies or flash storage to store data.

Storages in HTML5

HTML5 comes up with the following storage types

1. **localStorage**
2. **sessionStorage**
3. **IndexedDB**
4. **Application Cache**

Local Storage

Local storage is Key/value hash table to store date This storage is persistent and remains even on page reloads or after closing the browser It avoids cookies limitations Its important to note that its scoped to a domain name. The data stored at one domain will not be available to other domains/webpagesKey/value hash table Persistent on page reloads Avoids cookies' limitations Note that localstorage can only store string values.

Local Storage API function

localStorage.setItem('key', someValue);

method stores the vaue with the key provided

Ex: localStorage.setItem('name, 'smith');

To get the item use the **getItem** function with the same key localStorage.getItem('key');

Ex: localStorage.getItem('name');

localstorage.length: returns the number of items stored

Ex :localStorage.length

localStorage.key(index); This method returns the key stored at the index. The index starts at 0.
Ex: localStorage.key(0) returns name

localStorage.removeItem(key): This method removes the particular key form localstorage.

Ex localStorage.removeItem('name').

localStorage.clear(); This function clears all the storage values.

Session Storage

Session storage is Key/value hash table to store date This storage is not persistent and goes of if the tab is closed or browser is closed. Its important to note that its scoped to a domain name. The data stored at one domain will not be available to other domains/webpagesKey/value hash table Note that session storage can only store string values.

Session Storage API function

sessionStorage.setItem('key', someValue); This method stores the vaue with the key provided
Example: sessionStorage.setItem('name, 'smith');

To get the item use the **getItem** function with the same key **sessionStorage.getItem('key');**
Example:sessionStorage.getItem('name');

sessiontorage.length: returns the number of items stored.

Ex :sessionStorage.length

sessionStoragekey(index): This method returns the key stored at the index.

The index starts at 0.

Ex: sessionStorage.key(0) returns name

sessionStorage.removeItem(key): This method removes the particular key form localstorage.
Ex: sessionStorage.removeItem('name').

sessionStorage.clear(); This function clears all the storage values.

Storing arrays/object values in localStorage/SessionStorage

To store the object,array value, convert to json string and then store it

var userOb = {user: 'john',id: 10}; //convert the object to string value using JSON.stringify

Var stringValue = JSON.stringify(userOb); //storing the string value

localStorage.setItem('user',stringValue);

//Parsing the JSON string value and converting to object

var user = JSON.parse(localStorage.getItem('user'));

Indexed DB

IndexedDB is HTML5 API for client-side storage. The data stored is structured. It also comes up with a powerful search API for quickly searching. Unlike other data storage like local storage or session storage, indexed db doesn't have a limit and can store huge amounts of data. Unlike local/session storage, we can also store JavaScript objects directly to the data store.Using IndexedDB are done asynchronously.

Following are the important points of indexed Db

1. IndexedDB databases store key-value pairs
2. IndexedDb is availble per Domain. Means data is accessible to the same origin that creates it.
3. IndexedDB is built on a transactional database model
4. IndexedDB is object based

Using Indexed DB

window.indexedDB.open(<db>,<version no>) is the function used to open the database. Note that it is a asynchronous operation. The first parameter is the db name and the second is the version.

```
var db;

function openDatabase()

{ var request = window.indexedDB.open("students",3);

request.onsuccess = function(e) {

db = request.result; } }
```

Note that opening the DB may fire the following events:

success:fired when the database is successfully opened

error:when an error occurs upgrade

needed: This event is used when the user first opens the database as well as when the database version is changed

blocked: May fire if a previous connection was not closed.

Working with database

objectStore.add(myItem, optionalKey) : This function is used to create the object store.

Following is the example showing its usage

```
var data = { name: "smith", age: 29} ;  // open a read/write db transaction for adding the data

var transaction = db.transaction(["students"], "readwrite");  // on the success of transaction

 transaction.oncomplete = function(event) {

console.log("transaction complete");

 };

 transaction.onerror = function(event) {

console.log("transaction error");

};  // creating  object store on the transaction

var objectStore = transaction.objectStore("students",{ autoIncrement: true });

// add our newItem object to the object store

var objectStoreRequest = objectStore.add(data);

objectStoreRequest.onsuccess = function(event) {

console.log("item added");  };
```

Reading database

Following is the code to read the data

```
        var key = 1; //key correspoding to the autoincrement key

        var transaction = db.transaction(["students"],"readonly");

        var store = transaction.objectStore("students");

        var request = store.get(key);

        request.onsuccess = function(event) {

        var result = event.target.result;

        if(result) { for(var field in result) {
```

```
console.log(result[field]); }

else {  console.log("Not match found");

}
```

Modernizr

Modernizr, a tool for HTML5 verification of tags in HTM5 and features. Modernizr is a small JavaScript Library that detects the availability of native implementations for HTML5 Features and CSS Modernizr provides an easy way to detect any new feature so that you can take corresponding action Just load the Modernizr script at the head section of DOM

```
<script src="modernizr.min.js" type="text/javascript"></script>
```

Feature Detection using Modernizr

Below is the code for detecting support of audio tag.

the no-audio is the css implementation if audio support is not present.

```
/* In your CSS: */

        .no-audio #playback-music {

        display: none; /* Don't show Audio options */ }

        .audio #playback-music button {

/* Style the Play and Pause buttons nicely */ }

<!-- In your HTML: -->

        <div id="playback-music">

                <audio>

                <source src="audio.ogg" />

                <source src="audio.mp3" />
```

```
            </audio>

        <button id="play">Play</button>

         <button id="pause">Pause</button>

        </div>
```

For detecting support for CSS3, Modernizr requires no knowledge of JavaScript. we simply attach the file to the web page, and it dynamically adds a set of classes to the <html> element depending on the browser's environment.

Note that The class names are standardized.

For example, the boxshadow class is added if the browser supports the boxshadow property; otherwise no-boxshadow is added instead. But Modernizr uses JavaScript to detect which features a browser supports, but rather than using JavaScript to load different style sheets dynamically. It uses the simple technique of adding classes to the page's <html> tag. It up to the designer to use the CSS cascade to serve the appropriate styles for HTML5 elements.

Example: if the box-shadow property is supported, Modernizr adds the boxshadow class. If not then it adds the no-boxshadow class instead.

    Ex:. no-boxshadow img { /* styles for browsers that don't support box-shadow */ }

no-js class Have a look on the code below:

```
<!DOCTYPE HTML>

<html class="no-js">
```

Modernizr depends on JavaScript being enabled in the browser. If it is, this class is automatically removed. If JavaScript is not enabled, serve an alternate style sheet with class no-js Note that add modernizr lib before the style sheets are loaded.

Detecting attribute support

Let's find if required and autofocus property is available for us to use Have a look at the code below

```
<script src="js/modernizr.js">

</script> <script> window.onload = function() { // code to execute when page loads };

 </script> </head window.onload = function() { // getting form and  input elements

var form = document.forms[0],

inputs = form.elements; // if no autofocus, put the focus in the first field

if (!Modernizr.input.autofocus) { inputs[0].focus(); }

// if required not supported, we will emulate it ourself }
```