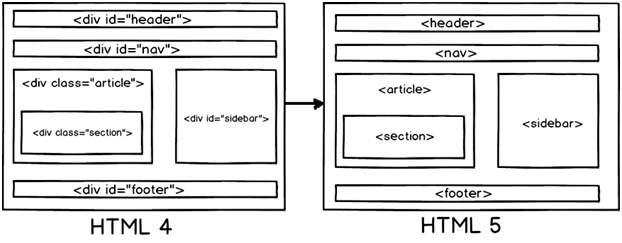
**How is the page structure of HTML 5 different from HTML 4 or previous HTML?**

A typical web page has headers, footers, navigation, central area and side bars. Now if we want to represent the same in HTML 4 with proper names to the HTML section we would probably use a DIV tag.  
But in HTML 5 they have made it more clear by creating element names for those sections which makes your HTML more readable.



Below are more details of the HTML 5 elements which form the page structure.

* <header>: Represents header data of HTML.
* <footer>: Footer section of the page.
* <nav>: Navigation elements in the page.
* <article>: Self-contained content.
* <section>: Used inside article to define sections or group content in to sections.
* <aside>: Represent side bar contents of a page.

**Question: What are the new features in HTML5?**  
Following are new features in HTML5  
Local storage.  
New form controls like calendar, date, time, email, URL and search etc.  
canvas element is provided for 2D drawing.  
video and audio elements for media playback.  
New elements are provided. For e.g. article, header, footer, nav, section.  
  
  
**Question: What are the various elements provided by HTML 5 for media content?**  
**audio**- It defines sound content.  
**video**- It defines a video.   
**source**- This tag defines the source of video and audio.  
**embed**- It provides a container for an external application.   
**track**- It defines text tracks for videoand audio.  
  
  
**Question: What are the new Form elements made available in HTML5?**  
**datalist**- It specifies a list of options for input controls   
**keygen**- This tag defines a key-pair generator field.   
**output**- It defines the result of a calculation.  
  
  
**Question: What are the various tags provided for better structuring in HTML5?**  
**article**- This tag defines an article.   
**aside**- It defines content other than the page content.  
**bdi**- This tag isolates a part of text for formatting.   
**command**- It defines a command button to be invoked by the user.   
**details**- It defines additional details that can be viewed .   
**dialog**- It defines a dialog box.   
**figure**- This tag specifies content like illustrations, diagrams, photos, code listings etc.   
**figcaption**- It is used to provide a caption for a figure element .  
**footer**- This tag defines a footer for a document or section.   
**header**- This tag is used to define a header for a document .   
**hgroup**- When there are multiple levels in a heading, it groups a set of h1 to h6 elements.   
**mark**- It defines highlighted text.   
**meter**- It defines a scalar measurement within a known range.   
**nav**- It defines links for navigation.   
**progress**- This tag exhibits the progress of a task.   
**ruby**- It defines a ruby annotation for East Asian typography.   
**rt**- It defines an explanation/pronunciation of characters for East Asian typography.  
**rp**- This tag tells the system what to display in browsers that do not support ruby annotations.   
**section**- It defines a section in a document.   
**summary**- It provides a visible heading for a details element.   
**time**- This tag defines a date/time.   
**wbr**- This tag defines a line-break.  
  
  
**Question: What is SVG?**  
SVG is the abbreviation for **Scalable Vector Graphics** and is recommended by W3C.   
It is used to define vector-based graphics for the Web  
  
  
**Question: What is a Canvas? What is the default border size of a canvas?**  
Canvas is a rectangular area on a HTML page, specified with the canvas tag.   
By default, It has no border. To get a border style attribute can be used.  
  
  
**Question: Differentiate between Canvas and SVG?**  
Canvas is resolution dependent while SVG is not.   
Canvas does not provide any support for event handlers while SVG does provide the support for event handlers.   
Canvas is suitable for graphic-intensive games while SVG is not suitable for gaming.  
Canvas is suitable for small rendering areas while SVG is suitable for large rendering areas like Google maps.  
  
  
**Question: HTML 5 provides drag and drop facility. How do you make an image draggable?**

<img draggable="true" />

**Question: What is HTML5 Web Storage?**  
It store the data locally in the user's browser  
  
  
**Question: Differentiate between session Storage and local Storage objects?**  
Session Storage object stores the data only for one session while local Storage object stores the data without an expiry date.  
  
  
**Question: What is a Manifest file?**  
A Manifest file is a simple text file that tells the browser what to cache and what not to cache.   
  
  
**Question: What is a Web Worker?**  
A web worker is a JavaScript which runs in the background.

Consider the below heavy for loop code which runs above million times.

function SomeHeavyFunction()

{

for (i = 0; i < 10000000000000; i++)

{

x = i + x;

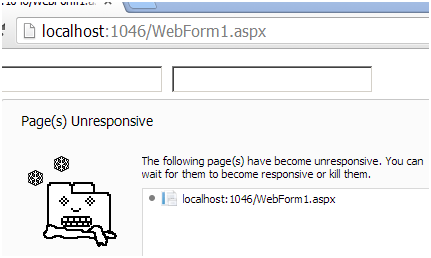
}

}

Let’s say the above for loop code is executed on a HTML button click. Now this method execution is synchronous. In other words the complete browser will wait until the for loop completes.

<input type="button" onclick="SomeHeavyFunction();" />

This can further lead to browser getting freezed and unresponsive with an error message as shown in the screen below.



So if we can move this heavy for loop in a JavaScript file and run it asynchronously that means the browser does need to wait for the loop then we can have a more responsive browser. That’s what web worker are for.

Web worker helps to execute JavaScript file asynchronously.

**Question: What is the purpose of HTML5 versus XHTML?**  
HTML5 is the next version of HTML 4.01, XHTML 1.0 and DOM Level 2 HTML. Its aim to reduce the need for proprietary plug-in-based rich internet application (RIA) technologies such as Adobe Flash, Microsoft Silverlight etc.  
  
  
**Question: WHAT are some other advantages of HTML5?**  
Cleaner markup than earlier versions of HTML  
Additional semantics of new elements like header, nav, and time  
  
  
**Question: What is the !DOCTYPE? Is it mandatory to use in HTML5?**  
The !DOCTYPE is an instruction to the web browser about what version of HTML the page is written in. The !DOCTYPE tag does not have an end tag. It is not case sensitive.  
  
  
  
**Question: What are various tags which are not available in HTML5?**

acronym

applet

basefont

big

center

dir

font

frame

frameset

noframes

strike

tt

**Question: How to link an email address?**

<a href="mailto:myemialid@wten.in">Email Me</a>

**What is local storage concept in HTML 5?**

Many times we would like to store information about the user locally in the computer. For example let’s say user has half-filled a long form and suddenly the internet connection breaks off. So the user would like you to store this information locally and when the internet comes back.He would like to get that information and send it to the server for storage.

Modern browsers have storage called as “Local storage” in which you can store this information.

**How can we add and remove data from local storage?**

Data is added to local storage using “key” and “value”. Below sample code shows country data “India” added with key value “Key001”.

localStorage.setItem("Key001","India");

To retrieve data from local storage we need to use “getItem” providing the key name.

var country = localStorage.getItem("Key001");

You can also store JavaScript object’s in the local storage using the below code.

var country = {};

country.name = "India";

country.code = "I001";

localStorage.setItem("I001", country);

var country1 = localStorage.getItem("I001");

If you want to store in JSON format you can use “JSON.stringify” function as shown in the below code.

localStorage.setItem("I001",JSON.stringify(country));

**What is the lifetime of local storage?**

Local storage does not have a life time it will stay until either the user clear it from the browser or you remove it using JavaScript code.

**What is the difference between local storage and cookies?**

|  |  |  |
| --- | --- | --- |
|  | **Cookies** | **Local storage** |
| **Client side / Server side.** | Data accessible both at client side and server side. Cookie data is sent to the server side with every request. | Data is accessible only at the local browser side. Server cannot access local storage until deliberately sent to the server via POST or GET. |
| **Size** | 4095 bytes per cookie. | 5 MB per domain. |
| **Expiration** | Cookies have expiration attached to it. So after that expiration the cookie and the cookie data get’s deleted. | There is no expiration data. Either the end user needs to delete it from the browser or programmatically using JavaScript we need to remove the same. |

**<a name="Whatissessi>What is session storage and how can you create one?</a></h2> <p>Session storage is same like local storage but the data is valid for a session. In simple words the data is deleted as soon as you close the browser.</p> <p>To create a session storage you need to use “sessi>What is difference between session storage and local storage?**

Local storage data persists forever but session storage is valid until the browser is open, as soon as the browser closes the session variable resets.

**What is WebSQL?**

WebSQL is a structured relational database at the client browser side. It’s a local RDBMS inside the browser on which you can fire SQL queries.

**Is WebSQL a part of HTML 5 specification?**

No, many people label it as HTML 5 but it’s not part of HTML 5 specification. The specification is based around SQLite.

**So how can we use WebSQL?**

The first step we need to do is open the database by using “OpenDatabase” function as shown below. The first argument is the name of the database, the next is the version, then a simple textual title and finally the size of the database.

var db=openDatabase('dbCustomer','1.0','Customer app’, 2 \* 1024 \* 1024);

To execute SQL we then need to use “transaction” function and call “executeSql” function to fire SQL.

db.transaction(function (tx)

{

tx.executeSql('CREATE TABLE IF NOT EXISTS tblCust(id unique, customername)');

tx.executeSql('INSERT INTO tblcust (id, customername) VALUES(1, "shiv")');

tx.executeSql('INSERT INTO tblcust (id, customername) VALUES (2, "raju")');

}

In case you are firing “select” query you will get data is “results” collection which we can loop and display in the HTML UI.

db.transaction(function (tx)

{

tx.executeSql('SELECT \* FROM tblcust', [], function (tx, results) {

for (i = 0; i < len; i++)

{

msg = "<p><b>" + results.rows.item(i).log + "</b></p>";

document.querySelector('#customer).innerHTML += msg;

}

}, null);

});

**<a name="Whatisapplicati>What is application cache in HTML5?</a></h2> <p>One of the most demanded things by end user is offline browsing. In other words if internet c>So how do we implement application cache in HTML 5?**

The first thing in we need to specify is the “manifest” file. “manifest” file helps you to define how your caching should work. Below is the structure of the manifest file :-

CACHE MANIFEST

# version 1.0

CACHE :

Login.aspx

* All manifest file starts with CACHE MANIFEST statement.
* #( hash tag) helps to provide the version of the cache file.
* CACHE command specifies which files needs to be cached.
* The content type of the manifest file should be “text/cache-manifest”.

Below is how cache manifest has been provided using ASP.NET C#.

Response.ContentType = "text/cache-manifest";

Response.Write("CACHE MANIFEST \n");

Response.Write("# 2012-02-21 v1.0.0 \n");

Response.Write("CACHE : \n");

Response.Write("Login.aspx \n");

Response.Flush();

Response.End();

One the cache manifest file is created the next thing is to provide the link of the manifest file in the HTML page as shown below.

<html manifest="cache.aspx">

When the above file runs first time it gets added in the browser application cache and in case server goes down the page is served from the application cache.

**What is fallback in Application cache?**

<a name="WhatisfallbackinApplicati>FALLBACK:

/home/ /homeoffline.html </a></pre>

<h2><a name=" whatisfallbackinapplicati=""></a>