3/4/20 HTML5

local storage & session storage objects

The two pre-defined objects supported in a HTML5 based browsers using which we could able to store user preferences within the browser cache itself.

following are the pre-defined methods being supported or can be applied on both local storage and session storage object through which we could able to add, remove or update data.

setItem("Key", <value>) – To set a value in cache

getItem("Key") – Returns value been stored on a Key.

- removeItem("Key") – Removes a Key value
- removeAll() – removes all values inside object

4/4/20

Difference b/w local and session storage objects

These two objects are almost similar used to store user preferences within browser cache which can be accessible even on reload or reopen, the only difference b/w these two objects is data stored under local storage object will be available even after reloading or reopen where as data stored under session object will be only available on reload of the

page, will be flushed out automatically while closing the page.

Note: These two objects has same set of pre-defined methods can be applied on them.

Sematic tags of HTML 5:

following are the pre-defined semantic tags supported in Html5. The name of semantic tags describe the purpose of the element and type of content that is within the tag.

1. article
2. aside
3. details
4. figure
5. footer
6. header
7. main
8. maincontainer
9. section
10. menu
11. navbar
12. nav
13. summary
14. title

etc.

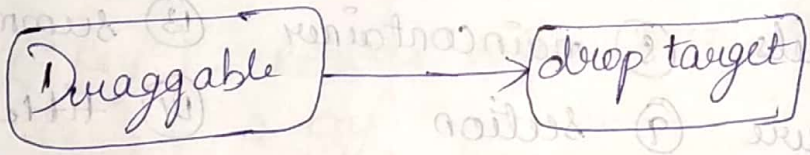
following are the set of input elements supported in Html5 to read different types of data from user.

1. input type = "color"
2. input type = "date"
3. input type = "datetime-local"
4. input type = "email"
5. input type = "search"
6. input type = "month"
7. input type = "number"

8. input type = "range"
9. input type = "tel"
10. input type = "url"
11. input type = "week"
12. input type = "time"
etc.

6/7/20

HTML5 drag & drop Events: HTML5 supports handling new set of event types like drag & drop. We could able to invoke call back methods when an element is getting dragged (or) an element gets droped.



Events:
→ dragstart
→ dragend

Attributes:
→ draggable = "true"

Events:
→ dragenter
→ dragover
→ dragleave
→ drop

HTML5 Audio Vedio Tag Support

HTML5 provides a feature of adding audio or vedio files without any dependency of third party plug ins

Syntax: <audio autoplay controls>
<source type = "audio/<file extension>" src = "<path of audio file>"
</audio>

```
<vedio autoplay controls>
    <source type = "vedio/<file extension>"
            src = "<path of vedio file">
</vedio>
```

Attributes can be added to audio or vedio tags

① controls - specifies that whether audio/vedio controls like play, pause etc to be show or not.

② autoplay - attribute ~~Boolean~~ value specifies whether audio/vedio to be played on load of page itself.

③ height/width - To set height and width of the player.

④ loop - Boolean value specifies to keep playing audio/vedio file once it finishes.

⑤ muted - specifies audio/vedio output to be muted.

⑥ poster - Takes an image url as input and shows the image before the vedio gets played.

⑦ preload - automatically buffers the vedio/audio file.

7/4/20 HTML5 Web Workers

Multi-tasking & Multi-threading.

The concept of process of executing multiple jobs at a time to increase the performance of the page is called multi-tasking or multi threading.

Note:

java script directly does not support the threading concept in it.

Web Workers

It is a new feature been supported from HTML5 through which we indirectly achieve the feature of multi threading and multi-tasking.

(1) workers are a seperate js files gets initiated through the main thread, executes parallely to the main thread.

(2) Even though a worker gets executed independent through the main thread it can still pass or communicate to the main thread.

(3) Even the main thread is capable of recieving msgs from a workers.

④ In a single page, we can instantiate any no. of web workers.

following are the steps to be followed to implement web workers in a application

Step 1: create an External js file (a web worker) which executes independent to the main thread, communicates through the main thread using post message method.

Step 2: Instantiate a web worker from the main thread through pre-defined worker class

Syntax: var worker = new worker ("<worker js file path>");

Step 3: add the onmsg Event handler on worker object which gets invoked automatically when there is a response from corresponding web worker

Syntax: worker.onmessage = function(event) {

...

}

// call back methods get fired automatically when there is a message from web worker.

// Event holds the data been passed by web worker.

8/4/20 Note:

As the web worker gets executed independent to the main thread, it is not included within the main page, it can never access the dom structure of the web page. (document object).

Application Cache:

Its an new feature being supported in Html5 through which we could make the web page resources to be accessable even while offline.

following are the steps to be followed to implement application cache to a web page.

Step 1: create an External app cache file (recommended file extention is '.appcache')

Step 2: define set of rules within the Appcache files which specifies which resources to be available offline, for which resources network connection is mandatory.

Step 3: within the HTML tag through manifest attribute specify the app cache file to be used to the current file.

Syntax:

```
<Html manifest = "../sample.appcache">
...
</Html>
```

Creating an Appcache file

Any appcache file contains following 3 blocks.

1. CACHE MANAGEMENT / MANIFEST
2. NETWORK
3. FALLBAC.

(1) Cache Management Block

under this block we specify the list of all the resources (HTML files, js files, images, css files) which needs to be accessed even when there is no n/w connection.

(2) Network Block.

under this, we specify the list of resources which should never be cached. should have network connection to access these resources (ex: loginpage paymentpage.html

(3) fallback Block.

under this we specify a custom user-defined page not found file which will be automatically thrown when the user tries to access the resource which need an internet connection

Sample appcache file

The following is a more complete cache manifest file for the imaginary web site at www.example.com.

```
CACHE MANIFEST
# V1 2011-08-14
# This is another comment.
index.html
cache.html
style.css
image1.png

# use from network if available
NETWORK:
network.html

# Fallback Content
FALLBACK:
/ fallback.html
```

This Example uses NETWORK and FALLBACK sections to specify that the network.html page must always be retrieved from the network, and that the fallback.html page should be served as a fallback resource (eg., in case a connection to the server cannot be established)

9/4/20 Html5 'Canvas' Tag

a pre-defined tag been supported in Html5 using which we could able to draw graphical objects within the canvas container

* It supports set of pre-defined java script methods through which we could draw objects within canvas container.

Following are the steps to be followed to draw graphical objects within canvas container

① create a canvas container with unique ref. ~~container~~. Eg: <canvas id = "userCanvastag"></canvas>

② create a context object of canvas container.

Eg: var element = document.querySelector("#userCanvastag");

// context object

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

③ use the pre-defined methods on context object through objects can be drawn on container.

Drawing Rectangle:

```
ctx.fillStyle = "red"; // setting background
ctx.fillRect(0,0,150,100); // x, y, width, height
```

Drawing lines:

```
ctx.moveTo(20,20); // x, y
ctx.lineTo(80,80);
ctx.stroke();
```

Drawing circle:

```
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2*Math.PI);
ctx.stroke();
```

Adding Text:

```
ctx.font = "20px Arial";
ctx.fillText("Hello...", 10, 50);
(or)
ctx.strokeText("msg", 10, 50);
```

etc...

SVG (Scalable Vector Graphics)

Another way of creating graphical objects within the html5 container, drawing graphical objects within SVG is almost like canvas tag. where the only difference is to draw objects within canvas we make use of js methods where as in SVG we use pre-defined html tags to draw objects.

following are pre-defined graphical object tags can be used only within the SVG container to draw objects

Syntax: <svg>...
</svg>

drawing circle:

```
<circle cx="50" r="20" cy="60"></circle>
```

drawing Rectangle:

<rect width = "100" height = "200"></rect>

drawing ellipse:

<ellipse cx = "200" cy = "200" rx = "100" ry = "150">

// rx - Horizontal radius, ry is Vertical radius.

drawing line:

<line x1 = "100" y1 = "100" x2 = "120" y2 = "120"></line>

Adding TEXT:

<text x = "20" y = "90">

...

...

</tex>

Note:

css properties can be applied to tags under svg container:

stroke - to set color → stroke: red;

stroke-width - line size → stroke-width: "4"

fill - to set background → fill: blue;

10/4/20 Navigator Object

A pre-defined object by default available in html5 based page which holds extra information of the current browser and operating system of the current machine.

Eg: * list of languages the browser is supporting whether.

* whether system connected to internet or not.
* appcode and appname of current browser
* vendor name and version number of the current browser.
* list of plugins being installed by the browser
* list of media devices being connected
* Events like onconnect and ondisconnect of USB device
* list of media devices being connected
* Bluetooth info of current machine.
* geolocation of current machine etc.