HTML

Sujata Batra

Overview

1. Advance version of HTML.
2. In 2008, the first HTML5 public draft was released
3. HTML5 W3C Final Recommendation was released 28. October 2014.
4. New elements, attributes, and behaviors were introduced.
5. It helps to create more powerful website and interactive web applications.
6. HTML5 comes with XML syntax.
7. HTML5 is to compete with Flash and Silverlight.
8. Empowering Mobile devices.

Sujata Batra

Technical Advantages Over Previous Version.

1. Audio and Videos are integral part of HTML5 specifications e.g. <audio> and<video> tags.
2. Vector graphics is integral part of HTML5 e.g. SVG and canvas.
3. JS GeoLocation API in HTML5 helps identify location of user browsing any website (provided user allows it).
4. Full duplex communication channels can be established with Server using Web Sockets.
5. Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5.
6. Application Cache, Web SQL database and Web storage is available as client side storage.
7. Retain Backward Compatibility with previous versions of HTML5.

HTML5 Technology Functions

Sujata Batra

Semantics: allowing you to describe more precisely what your content is.

Connectivity: allowing you to communicate with the server in new and innovative ways.

Offline & Storage: allowing webpages to store data on the client-side locally and operate offline more efficiently.

Multimedia: making video and audio first-class citizens in the Open Web.

2D/3D Graphics & Effects: allowing a much more diverse range of presentation options.

Performance & Integration: providing greater speed optimization and better usage of computer hardware.

Device Access: allowing for the usage of various input and output devices.

Styling: letting authors write more sophisticated themes.

Elements removed in HTML5

Element	Use instead
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<strike>	CSS
<tt>	CSS

Sujata Batra

HTML5 New Tags and Elements

HTML5 Indroduces 28 New Elements, Some of them are mentioned here.

Navigation:

<article>
<aside>
<header>
<hgroup>
<footer>
<figure>
<figcaption>
<nav>
<section>

Multimedia/Interactivity:

<audio>
<canvas>
<embed>
<source>
<track>
<video>

New <input> types:

color
date
datetime
datetime-local
email
month
number
range
search
tel
time
url
week

Miscellaneous:

<bdi>
<command>
<datalist>
<details>
<mark>
<meter>
<output>
<progress>
<summary>
<rp>
<rt>
<ruby>
<time>
<wbr>

Suja ta Batra

Defining HTML5 Documents

Remember the DOCTYPE declaration-

```
<!DOCTYPE html>
```

Again, HTML5 simplifies this line:

```
<html lang="en">
```

The default character encoding (charset) declaration

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

Semantic Elements

A semantic element clearly describes its meaning to both the browser and the developer.

Eg. of non-semantic elements: ** ** and **** - Tells nothing about its content.

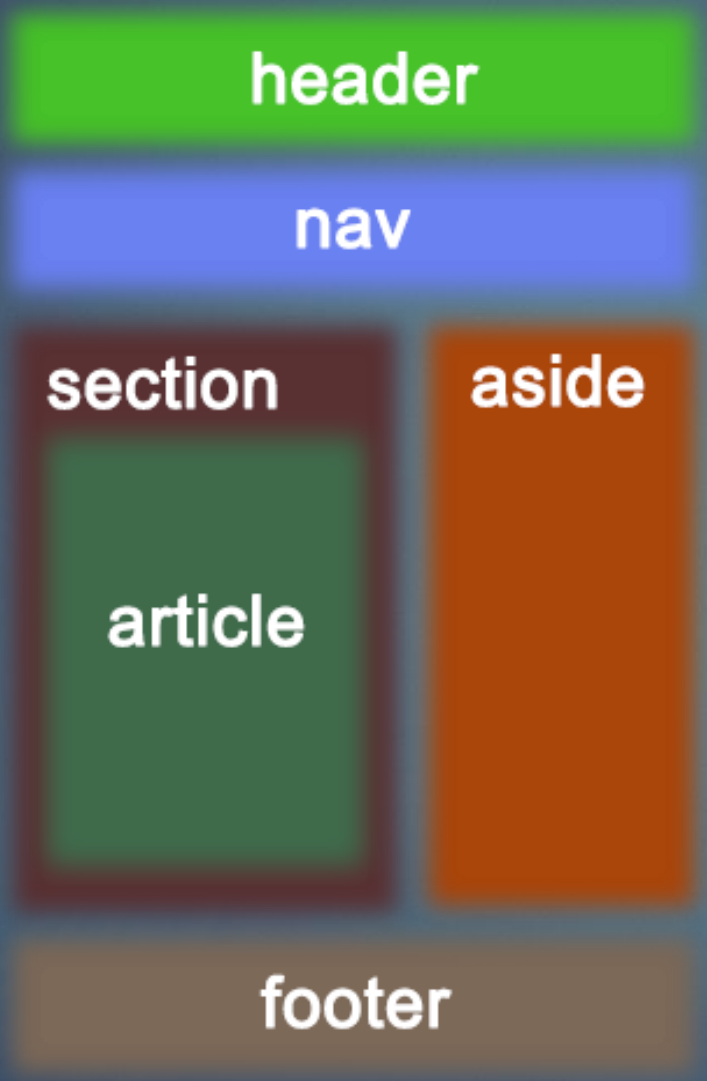
Eg. of semantic elements: **<form>, <table>,** and **<img>** - Clearly defines its content.

Many web sites contain HTML code like: ** ** to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

<article>
<aside>
<details>
<figcaption>
<figure>
<footer>
<header>
<main>
<mark>
<nav>
<section>
<summary>
<time>

Semantic Elements

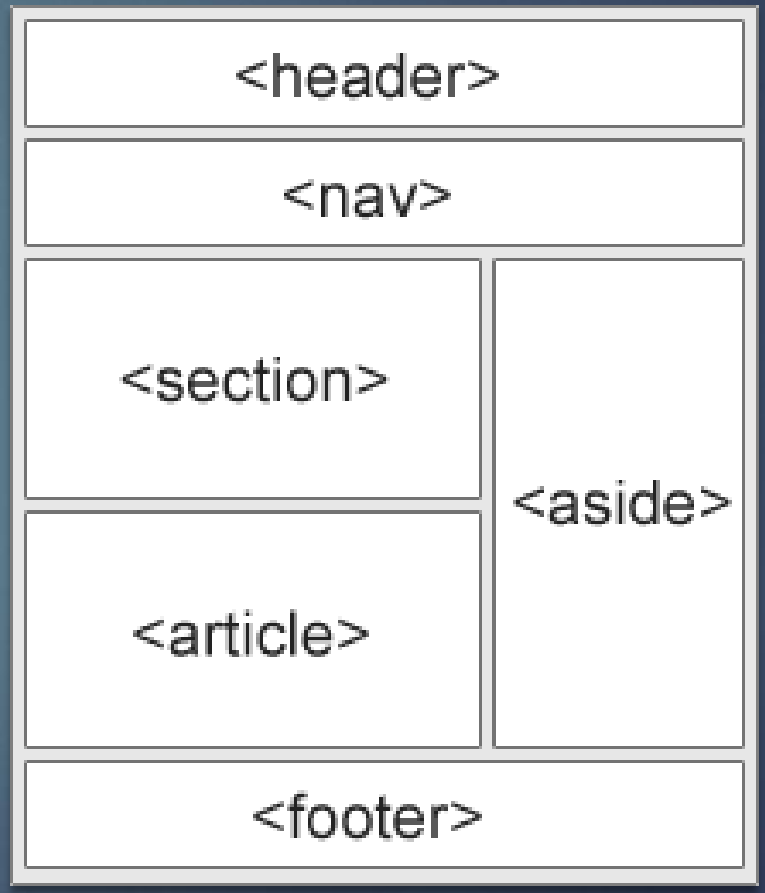



```
<header>
    <h1>title</h1>
    <form>Search</form>
    <nav>
        <ul>Site navigation</ul>
    </nav>
</header>
<section>
    <article>
        <h1>Article title</h1>
        <p>Summary</p>
    </article>
</section>
<aside>
    <section>
        <h2>Blogroll…</h2>
    </section>
</aside>
<footer>
    <h2>Footer</h2>
</footer>
```

Sujata Batra

New Semantic Elements in HTML5

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time



Sujata Batra

HTML5 <section> Element

- A section is a thematic grouping of content, typically with a heading.
- A home page could normally be split into sections for introduction, content, and contact information.

HTML5 <article> Element

- The <article> element specifies independent, self-contained content.
- An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
- Examples of where an <article> element can be used:
  - Forum post
  - Blog post
  - Newspaper article

HTML5 <header> Element

- The <header> element specifies a header for a document or section.
- The <header> element should be used as a container for introductory content.
- You can have several <header> elements in one document.

HTML5 <footer> Element

- The <footer> element specifies a footer for a document or section.
- A <footer> element should contain information about its containing element.
- A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.
- You may have several <footer> elements in one document.

HTML5 <nav> Element

- The <nav> element defines a set of navigation links.
- NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

HTML5 <aside> Element

- The <aside> element defines some content aside from the content it is placed in (like a sidebar).

- The <aside> content should be related to the surrounding content.

HTML5 <figure> and <figcaption> Elements

- The purpose of a figure caption is to add a visual explanation to an image.
- In HTML5, an image and a caption can be grouped together in a <figure> element:

- **Example:**

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

HTML <summary> Tag

- The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.
- Browser Support

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<summary>	12.0	Not Supported	49.0	6.0	15.0

HTML <details> Tag

- The <details> tag can be used to create an interactive widget that the user can open and close.
- The content of a <details> element should not be visible unless the open attribute is set.
- Browser Support

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<details>	12.0	Not Supported	49.0	6.0	15.0

Example

Sujata Batra

```
<details>
  <summary>Copyright 1999-2018.</summary>
  <p> - by Refsnes Data. All Rights Reserved.</p>
  <p>All content and graphics on this web site are the property of the
company Refsnes Data.</p>
</details>
```

HTML <main> Tag

- The <main> tag specifies the main content of a document.
- The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents such as sidebars, navigation links, copyright information, site logos, and search forms.
- Note: There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<main>	6.0	12.0	4.0	5.0	11.1

HTML <mark> Tag

- Use the <mark> tag if you want to highlight parts of your text.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<mark>	6.0	9.0	4.0	5.0	11.1

HTML <time> Tag

Sujata Batra

- The <time> tag defines a human-readable date/time.
- This element can also be used to encode dates and times in a machine-readable way so that user agents can offer to add birthday reminders or scheduled events to the user's calendar, and search engines can produce smarter search results.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<time>	6.0	9.0	4.0	5.0	11.1

- Attribute

Attribute	Value	Description
datetime	*datetime*	Represent a machine-readable date/time of the <time> element

Migration from HTML4 to HTML5

HTML4	HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

Sujata Batra

HTML 5 Form Elements

HTML5 Form Elements

- HTML5 added the following form elements:

- <datalist>
- <output>

HTML5 <datalist> Element

- <datalist> element specifies a list of pre-defined options for an <input> element.
- Users will see a drop-down list of the pre-defined options as they input data.
- The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.
- <form action="/action_page">
  <input **list="browsers"**>
  <datalist **id="browsers"**>
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
  </datalist>
  </form>

HTML5 <output> Element

- <output> element represents the result of a calculation (like one performed by a script).

```
function showResult() {
x = document.forms["myform"]["newinput"].value;
document.forms["myform"]["result"].value = x;
  }
<form action = "/cgi-bin/html5.cgi" method = "get" name = "myform">
 Enter a value : <input type = "text" name = "newinput" />
<input type = "button" value = "Result"  onclick = "showResult();" />
<output name = "result"></output>
</form>
```

HTML 5 Input Types

HTML5 Input Types

- HTML5 added several new input types:
- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Note : New input types that are not supported by older web browsers, will behave as <input type="text">.

- **Input Type Color**
  - The <input type="color"> is used for input fields that should contain a color.
  - Depending on browser support, a color picker can show up in the input field.

Example : <input type="color" name="favcolor" value="#ff0000">

- **Input Type Date**
  - The <input type="date"> is used for input fields that should contain a date.
  - You can also use the min and max attributes to add restrictions to dates.
  - Depending on browser support, a date picker can show up in the input field.

Example : Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">
Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

- **Input Type Datetime-local**
  - The <input type="datetime-local"> specifies a date and time input field, with no time zone.
  - Depending on browser support, a date picker can show up in the input field.

Example : <input type="datetime-local" name="bdaytime">

- **Input Type Email**
  - The <input type="email"> is used for input fields that should contain an e-mail address.
  - Depending on browser support, the e-mail address can be automatically validated when submitted.

- **Input Type File**
  - The <input type="file"> defines a file-select field and a "Browse" button for file uploads.

- **Input Type Month**
  - The <input type="month"> allows the user to select a month and year.
  - Depending on browser support, a date picker can show up in the input field.

- **Input Type Number**
  - The <input type="number"> defines a numeric input field.
  - You can also set restrictions on what numbers are accepted.

Example: <input type="number" name="quantity" min="1" max="5">

HTML 5 Input Attributes

HTML5 added the following attributes for <input>:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

and the following attributes for <form>:

- autocomplete
- novalidate

- **The autocomplete Attribute**
  - The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.
  - When autocomplete is on, the browser automatically completes the input values based on values that the user has entered before.
  - It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.
  - The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

- **The novalidate Attribute**
  - The novalidate attribute is a <form> attribute.
  - When present, novalidate specifies that the form data should not be validated when submitted.
- Example :
```
<form action="/action_page" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

- **The autofocus Attribute**
  - The autofocus attribute specifies that the input field should automatically get focus when the page loads.

- **The form Attribute**
  - The form attribute specifies one or more forms an <input> element belongs to.
  - To refer to more than one form, use a space-separated list of form ids.
- Example :<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname">
  <input type="submit" value="Submit">
  </form>

  Last name: <input type="text" name="lname" form="form1">

- **The formaction Attribute**
  - The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.
  - The formaction attribute overrides the action attribute of the <form> element.
  - The formaction attribute is used with type="submit" and type="image".

Example :<form action="/action_page.php">
First name: <input type="text" name="fname">
Last name: <input type="text" name="lname">
<input type="submit" value="Submit">
<input type="submit" formaction="/action_page2.php"
value="Submit as admin">
</form>

- **The formmethod Attribute**
  - The formmethod attribute defines the HTTP method for sending form-data to the action URL.
  - The formmethod attribute overrides the method attribute of the <form> element.
  - The formmethod attribute can be used with type="submit" and type="image".

- **The formnovalidate Attribute**
  - The formnovalidate attribute overrides the novalidate attribute of the <form> element.
  - The formnovalidate attribute can be used with type="submit".
- Example : <form action="/action_page.php">
  E-mail: <input type="email" name="userid">
  <input type="submit" value="Submit">
  <input type="submit" formnovalidate value="Submit without validation">
  </form>

- **The height and width Attributes**
  - The height and width attributes specify the height and width of an <input type="image"> element.
  - Always specify the size of images. If the browser does not know the size, the page will flicker while images load.

- **The min and max Attributes**
  - The min and max attributes specify the minimum and maximum values for an <input> element.
  - The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

- **The multiple Attribute**
  - The multiple attribute specifies that the user is allowed to enter more than one value in the <input> element.
  - The multiple attribute works with the following input types: email, and file.

- **The pattern Attribute**
  - The pattern attribute specifies a regular expression that the <input> element's value is checked against.
  - The pattern attribute works with the following input types: text, search, url, tel, email, and password.
  - Use the global title attribute to describe the pattern to help the user.
- Example : Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">

- **The placeholder Attribute**
  - The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).
  - The hint is displayed in the input field before the user enters a value.
  - The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

- **The required Attribute**
  - The required attribute specifies that an input field must be filled out before submitting the form.
  - The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

- **The step Attribute**
  - The step attribute specifies the legal number intervals for an <input> element.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

  - The step attribute can be used together with the max and min attributes to create a range of legal values.
  - The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.

HTML 5 Media

HTML Audio Tag

- HTML audio tag is used to define sounds such as music and other audio clips.
- Currently there are three supported file format for HTML 5 audio tag.
  - mp3
  - wav
  - ogg

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<audio>	4.0	9.0	3.5	4.0	10.5

Browser Support Audio File Format

Sujata Batra

Browser	mp3	wav	ogg
Internet Explorer	yes	no	no
Google Chrome	yes	yes	yes
Mozilla Firefox	yes	yes	yes
Opera	no	yes	yes
Apple Safari	yes	yes	no

Attributes of HTML Audio Tag

Attribute	Description
controls	It defines the audio controls which is displayed with play/pause buttons.
autoplay	It specifies that the audio will start playing as soon as it is ready.
loop	It specifies that the audio file will start over again, every time when it is completed.
muted	It is used to mute the audio output.
preload	It specifies the author view to upload audio file when the page loads.
src	It specifies the source URL of the audio file.

Audio Tag Example


```
<audio controls>
   <source src="myaudio.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Video Tag

- The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.
- Three video formats supported for HTML video tag:
  - mp4
  - webM
  - ogg

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<video>	4.0	9.0	3.5	4.0	10.5

Browser Support Video File Format

Browser	mp4	webM	ogg
Internet Explorer	yes	no	no
Google Chrome	yes	yes	yes
Mozilla Firefox	yes	yes	yes
Opera	no	yes	yes
Apple Safari	yes	no	no

Attributes of HTML Video Tag

Attribute	Value	Description
autoplay	autoplay	Specifies that the video will start playing as soon as it is ready
controls	controls	Specifies that video controls should be displayed (such as a play/pause button etc).
height	pixels	Sets the height of the video player
loop	loop	Specifies that the video will start over again, every time it is finished
muted	muted	Specifies that the audio output of the video should be muted
poster	URL	Specifies an image to be shown while the video is downloading, or until the user hits the play button
preload	auto metadata none	Specifies if and how the author thinks the video should be loaded when the page loads
src	URL	Specifies the URL of the video file
width	pixels	Sets the width of the video player

Sujata Batra

Video Tag Example


```
<video width="320" height="240" controls autoplay loop>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the html video tag.
</video>
```

HTML5 Plug-ins

To extend the functionality of the HTML browser. Plug-ins are also known as Helper Applications. Popular example of plug-ins are Java applets. Plug-ins can be added to web pages with the **<object>** tag or the **<embed>** tag.

1. **<object> Element –** It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.

   Eg.: <object width="200" height="60" data="SujataTraining.swf"></object>

2. **<embed> Element –** Used to embed object within an HTML document. It does not have closing tag. It can not contain alternative text.

   Eg.: <embed width="200" height="60" src="SujataTraining.swf">

Playing a YouTube Video in HTML

- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player.

Example:

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

Note: YouTube <object> and <embed> were deprecated from January 2015. You should migrate your videos to use <iframe> instead.

YouTube Autoplay


```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1">
</iframe>
```

- Value 0 (default): The video will not play automatically when the player loads.
- Value 1: The video will play automatically when the player loads.

YouTube Controls


```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

- Value 0: Player controls does not display.
- Value 1 (default): Player controls display.

Graphics API
(Canvas and SVG)

Previously possible with Flash, VML, Silverlight.

Very complex to do in JavaScript without plugins (for example, rounded corners or diagonal lines).

Provide native drawing functionality on the Web.

Completely integrated into HTML5 documents (Part of DOM).

Can be styled with CSS.

Can be controlled with JavaScript.

HTML Canvas

- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<canvas>	4.0	9.0	2.0	3.1	9.0

Canvas Examples

- A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

Example of an empty Canvas

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>
```

Draw on the Canvas With JavaScript

- Step 1: Find the Canvas Element

```
var canvas = document.getElementById("myCanvas");
```

- Step 2: Create a Drawing Object
  - The getContext() is a built-in HTML object, with properties and methods for drawing

```
var ctx = canvas.getContext("2d");
```

- Step 3: Draw on the Canvas
  - Finally, you can draw on the canvas.
  - Set the fill style of the drawing object to the color red:

```
ctx.fillStyle = "#FF0000";
```

  - The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is black.
  - The fillRect(*x,y,width,height*) method draws a rectangle, filled with the fill style, on the canvas:

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

Canvas Coordinates

- The HTML canvas is a two-dimensional grid.
- The upper-left corner of the canvas has the coordinates (0,0)

- Draw a Line
  - Use the following methods.
  - moveTo(x,y): It is used to define the starting point of the line.
  - lineTo(x,y): It is used to define the ending point of the line.
  - If you draw a line which starting point is (0,0) and the end point is (200,100), use the stroke method to draw the line.

Example

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

- Drawing Circle on Canvas
  - beginPath() - begins a path
  - use the arc() method
  - arc(x, y, r, startAngle, stopAngle)

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

Canvas - Gradients

- Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.
- There are two different types of gradients:
  - createLinearGradient(*x,y,x1,y1*) - creates a linear gradient
  - createRadialGradient(*x,y,r,x1,y1,r1*) - creates a radial/circular gradient
    - creates a radial gradient using the size and coordinates of two circles.
- Once we have a gradient object, we must add two or more color stops.
- The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

- To use the gradient, set the fillStyle or strokeStyle property to the gradient, then draw the shape (rectangle, text, or a line).

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

// Create a linear gradient
// The start gradient point is at x=20, y=0
// The end gradient point is at x=220, y=0
var gradient = ctx.createLinearGradient(20,0, 220,0);

// Add three color stops
gradient.addColorStop(0, 'green');
gradient.addColorStop(.5, 'cyan');
gradient.addColorStop(1, 'green');

// Set the fill style and draw a rectangle
ctx.fillStyle = gradient;
ctx.fillRect(20, 20, 200, 100);
```

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

// Create a radial gradient
// The inner circle is at x=110, y=90, with radius=30
// The outer circle is at x=100, y=100, with radius=70
var gradient = ctx.createRadialGradient(110,90,30, 100,100,70);

// Add three color stops
gradient.addColorStop(0, 'pink');
gradient.addColorStop(.9, 'white');
gradient.addColorStop(1, 'green');

// Set the fill style and draw a rectangle
ctx.fillStyle = gradient;
ctx.fillRect(20, 20, 160, 160);
```

Drawing Text on the Canvas

- font - defines the font properties for the text
- fillText(*text,x,y*) - draws "filled" text on the canvas
- strokeText(*text,x,y*) - draws text on the canvas (no fill)

Example

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
```

Or

```
ctx.strokeText("Hello World", 10, 50);
```

Add Color and Center Text


```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "red";
ctx.textAlign = "center";
ctx.fillText("Hello World", canvas.width/2, canvas.height/2);
```

Canvas - Images

- To draw an image on a canvas, use the following method:
  - drawImage(*image,x,y*)

```
window.onload = function() {
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  var img = document.getElementById(“myimage");
  ctx.drawImage(img, 10, 10);
};
```

HTML 5 API

HTML5 Geolocation

- Geolocation API is used to locate a user's position.
- 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.

Note: Geolocation is most accurate for devices with GPS, like smartphone.

Browser Support

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
GeoLocation	5.0-49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0

Note: As of Chrome 50, the Geolocation API will only work on secure contexts such as HTTPS. If your site is hosted on an non-secure origin (such as HTTP) the requests to get the users location will no longer function.

HTML 5 Geolocation

- The geolocation APIs work with a new property of the global navigator object ie. Geolocation object which can be created as follows –
- var geolocation = navigator.geolocation;

HTML 5 Geolocation

- The geolocation APIs work with a new property of the global navigator object ie. Geolocation object which can be created as follows –

```
var geolocation =
navigator.geolocation;
```

Sr.No.	Method & Description
1	getCurrentPosition()This method retrieves the current geographic location of the user.
2	watchPosition()This method retrieves periodic updates about the current geographic location of the device.
3	clearWatch()This method cancels an ongoing watchPosition call.

Using HTML Geolocation

- The **getCurrentPosition()** method is used to return the user's position.

```
<script>
var x = document.getElementById("demo");
function getLocation() {
//Check if Geolocation is supported
  if (navigator.geolocation) {
//If supported, run the getCurrentPosition() method.
//If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
//outputs the Latitude and Longitude
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Handling Errors and Rejections


```
<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition, showError);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
```

```
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}

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;
  }
}
```

The getCurrentPosition() Method - Return Data

- The getCurrentPosition() method returns an object on success.

Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

- watchPosition()

```
<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(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>
```

Note: You need an accurate GPS device to test this (like smartphone)

HTML5 Web Storage

- The **Web Storage API** provides mechanisms by which browsers can store key/value pairs, in a much more intuitive fashion than using cookies.

- Before HTML5, application data had to be stored in cookies, included in every server request.
- Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Web storage allows large amounts of application data to be be stored locally, without affecting your web application's performance.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
Web Storage	4.0	8.0	3.5	4.0	11.5

Web Storage concepts and usage

- The two mechanisms within Web Storage are as follows:
  - **window.sessionStorage** maintains a separate storage area for each given origin that's available for the duration of the page session (as long as the browser is open, including page reloads and restores)
  - **window.localStorage** does the same thing, but persists even when the browser is closed and reopened.
- These mechanisms are available via the Window.sessionStorage and Window.localStorage properties.

The localStorage


```
<div id="result"></div>

<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
  // Store
  localStorage.setItem("lastname", "Smith");
  // Retrieve
  document.getElementById("result").innerHTML = localStorage.getItem("lastname");
} else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";
}
</script>
```

- **Example to count the number of times a user has clicked a button**

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
  if (typeof(Storage) !== "undefined") {
    if (localStorage.clickcount) {
      localStorage.clickcount = Number(localStorage.clickcount)+1;
    } else {
      localStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";
  }
}
</script>
```

HTML5 Web SQL

HTML5 Web SQL Databases introduces a set of APIs to manipulate the client side database and is not a part of HTML5. Following are the concepts covered.

- What is Web SQL Database
- Creating Database
- Execution of Web SQL

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
Web SQL	4.0	Not Supported	Not Supported	3.1	11.5

What is Web SQL Database

Web SQL database is a web page API to store the data in databases.Actually SQL is not a part of HTML5. In order to store the data, HTML5 introduced some methods as follows.

- openDatabase
  - Used to create object of the Database. user can use the existing Database or create new one.
- transaction
  - Used to get the control on a transaction to perform commit or roll back depends on situation.
- executeSql
  - Used to execute the SQL Query.

In order to use Web SQL, initially open the existing data base, if not create the new database. The code below is used to create or open the database.

```
var db = openDatabase( ' mydb ' , ' 1.0 ' , ' Test DB ' , 2 * 1024 *1024);
```

The above code parameters are described below.

- Name of the database
- Version
- Description of text
- Database size
- Callback Creation

Now use the transaction() function which is used to execute the queries and transaction function need only one argument code as shown below.

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);

db.transaction(function (tx) {

tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, user)');

});
```

Then enter the table data by adding simple queries to the code as shown below.

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);

db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, user)');
        tx.executeSql('INSERT INTO LOGS (id, user) VALUES (1, "foobar")');
        tx.executeSql('INSERT INTO LOGS (id, user) VALUES (2, "logmsg")');
        tx.executeSql('INSERT INTO LOGS (id, user) VALUES (3, "lomsg")');
        msg = '<p>User message created and row inserted.</p>';
        document.querySelector('#status').innerHTML =  msg;
    });
```

```
db.transaction(function (tx) {
        tx.executeSql('SELECT * FROM LOGS', [], function (tx, results) {
          var len = results.rows.length, i;
          msg = "<p>Found rows: " + len + "</p>";
          document.querySelector('#status').innerHTML +=  msg;

          for (i = 0; i < len; i++){
            msg = "<p><b>" + results.rows.item(i).user + "</b></p>";
            document.querySelector('#status').innerHTML +=  msg;
          }
        }, null);
      });
```

HTML5 Web Workers

- HTML5 Web workers are used to handle the JavaScript multiple tasks at a time by running multiple threads.
- At the point when executing scripts in a HTML page, the page gets slow until the script is done.
- A Web Worker is a JavaScript which keeps on running in background, autonomously it runs different scripts, without influencing the execution of the page. User can keep on doing whatever need: clicking, selecting things, etc., while the web specialist keeps running out of sight.
- While executing the script in a web worker window does not have the direct access to the webpage and **DOM API**. In order to make the system less responsive Web Workers use the **System CPU cycles**.

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
Web Worker	4.0	10.0	3.5	4.0	11.5

HTML5 Web Workers

- The Web Workers are introduced with the URL of the JavaScript which contains the web worker executable code. The code is used to set the event listener and communicate with the script which is drawn from the main page as follows.
- Syntax

```
var worker = new Worker("bigLoop.js”);
```

- Then it checks if JavaScript file is present, the browser will send a new thread.
- Now A postmessage( ) is used to start a web worker and which also used to establish the communication between the web worker and main page. User can access the message passed by web worker by using the onmessage method.

```
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<script>
var w;

function startWorker() {
  if(typeof(Worker) !== "undefined") {
    if(typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
    }
    w.onmessage = function(event) {
        document.getElementById("result").innerHTML = event.data;
    };
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";
  }
}
function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>

</body>
</html>
```

demo_Worker.js


```
var i = 0;

function timedCount() {
  i = i + 1;
  postMessage(i);
  setTimeout("timedCount()",500);
}

timedCount();
```

- While working with the **Web Workers API** frequently user can get the some errors, to solve those errors HTML5 introduced error codes as shown below.

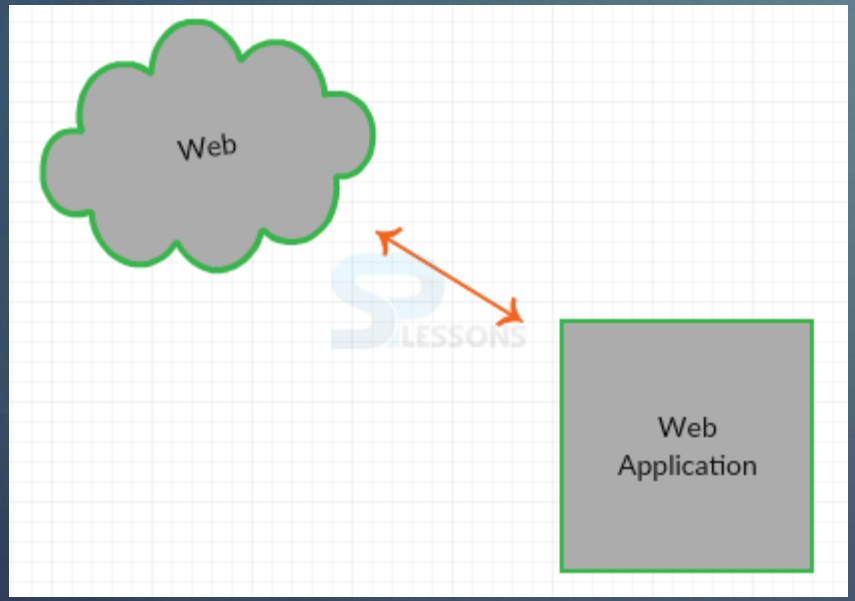
```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Big for loop</title>
    <script>
      var worker = new Worker('bigLoop.js');
      worker.onmessage = function (event) {
        alert("Completed " + event.data + "iterations" );
      };
      worker.onerror = function (event) {
        console.log(event.message, event);
      };
      function sayHello(){
        alert("Hello sir...." );
      }
    </script>
  </head>
  <body>
    <input type="button" onclick="sayHello();" value="Say Hello"/>
  </body>
</html>
```

HTML5 Offline Applications

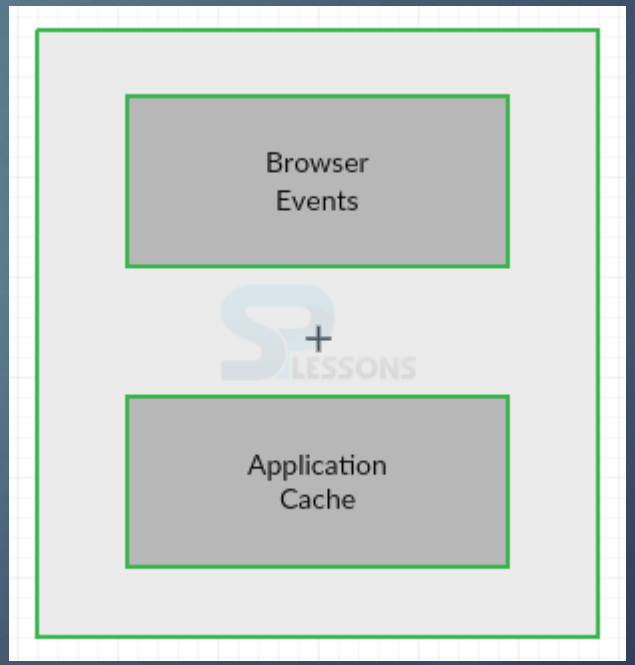
- HTML5 Offline Applications are same as web applications, but offline applications also work even when there is no internet connectivity.

Web Application Vs Offline Application

- Web application, user need to have the internet connectivity.
- User need to interact with page or site.
- Application takes roundtrip requests from the server and gives access to the user. If internet connectivity is lost, web application will not work properly.



- Offline Application will work even when there is no internet connectivity. In order to work certain parts of the application, correctly copy those files and save in a retrievable location.



Application Cache

HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

Application cache gives an application three advantages:

- Offline browsing - users can use the application when they're offline
- Speed - cached resources load faster
- Reduced server load - the browser will only download updated/changed resources from the server

Element	Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
Application Cache	4.0	10.0	3.5	4.0	11.5

- To enable application cache, include the manifest attribute in the document's <html> tag:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

- Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).
- The recommended file extension for manifest files is: ".appcache"

Note :A manifest file needs to be served with the **correct media type**, which is "text/cache-manifest". Must be configured on the web server.

The Manifest File

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The manifest file has three sections:

- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

demo.appcache

CACHE MANIFEST
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html

Updating the Cache

Once an application is cached, it remains cached until one of the following happens:

- The user clears the browser's cache
- The manifest file is modified (see tip below)
- The application cache is programmatically updated

HTML 5 Tags

Tag	Description
<article>	This element is used to define an independent piece of content in a document, that may be a blog, a magazine or a newspaper article.
<aside>	It specifies that article is slightly related to the rest of the whole page.
<audio>	It is used to play audio file in HTML.
<bdi>	The bdi stands for bi-directional isolation. It isolates a part of text that is formatted in other direction from the outside text document.
<canvas>	It is used to draw canvas.

<data>	It provides machine readable version of its data.
<datalist>	It provides auto complete feature for textfield.
<details>	It specifies the additional information or controls required by user.
<dialog>	It defines a window or a dialog box.
<figcaption>	It is used to define a caption for a <figure> element.
<figure>	It defines a self-contained content like photos, diagrams etc.
<footer>	It defines a footer for a section.
<header>	It defines a header for a section.
<main>	It defines the main content of a document.
<mark>	It specifies the marked or highlighted content.
<menuitem>	It defines a command that the user can invoke from a popup menu.
<meter>	It is used to measure the scalar value within a given range.

<nav>	It is used to define the navigation link in the document.
<progress>	It specifies the progress of the task.
<rp>	It defines what to show in browser that don't support ruby annotation.
<rt>	It defines an explanation/pronunciation of characters.
<ruby>	It defines ruby annotation along with <rp> and <rt>.
<section>	It defines a section in the document.
<summary>	It specifies a visible heading for <detailed> element.
<svg>	It is used to display shapes.
<time>	It is used to define a date/time.
<video>	It is used to play video file in HTML.
<wbr>	It defines a possible line break.