



Introduction to HTML5





Tag Structure



<meta> Tag

The meta declaration for document written in HTML 4.01

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

In HTML5

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

<script> Tag

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

In HTML5

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



Tag Structure



<link> Tag

```
<link rel="stylesheet" type="text/css" href="file.css">
```

In HTML5

```
<link rel="stylesheet" href="file.css">
```



Obsolete Elements In HTML5

Frame , frameset , noframes , font , big , center and few more.
Attributes like bgcolor, cellspacing, cellpadding, and valign.



Use of <a> Tag in HTML5

```
<a href="/about">
```

```
  <h2>About me</h2>
```

```
  <p>Find out what makes me tick.</p>
```

```
</a>
```



Tag Structure



Organizing Code Using Blocking Elements

-  **SECTION**
-  **ARTICLE**
-  **HEADER**
-  **FOOTER**
-  **ASIDE**
-  **FIGURE**
-  **NAV**



Forms in HTML5



- In addition to GET and POST action , PUT and DELETE are added as an action for form.
- Form elements can be anywhere and can be associated to form by giving form's ID to form attribute of that element.

```
<form id=foo>  
<input type="text">  
...  
</form>  
<textarea form=foo></textarea>
```



Forms in HTML5



New INPUT types



Email input type
`<input type=email>`



URL input type
`<input type=url>`



Date input type
`<input type=date>`



Time input type
`<input type=time>`



datetime input type
`<input type=datetime>`



Month input type
`<input type=month>`

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	26	27	28	29	30	31	1
31	2	3	4	5	6	7	8
32	9	10	11	12	13	14	15
33	16	17	18	19	20	21	22
34	23	24	25	26	27	28	29
35	30	31	1	2	3	4	5

Today None



Offline Web Applications



- **Allow you to keep using web apps and sites without a network connection**
- **Browsers cache data in Application cache**
- **Once resources are cached you can access them very quickly (without network request)**
- **HTML5 also allows online and offline detection**
- **We can use offline mechanism that allows us to easily prefetch site resources**



Geolocation



Geolocation is the art of figuring out where you are in the world and sharing that information with the people you trust.



Geolocation API provides latitude and longitude on the page where this API is used.



The simplest use of the geolocation API looks like this :

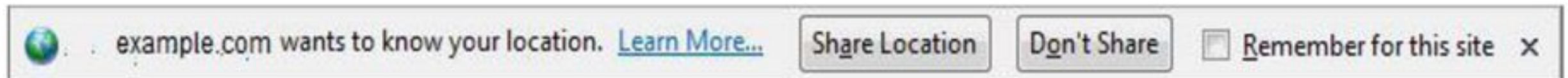
```
function get_location() {  
    navigator.geolocation.getCurrentPosition(show_map);  
}
```




Geolocation



Calling the `getCurrentPosition()` function of the geolocation API will cause the browser to popup an infobar at the top of the browser window.



This infobar says that you ,

- are told that a website wants to know your location**
- are told which website wants to know your location**
- can click through to Learn more link which explains what the is going on**
- can choose to share your location**
- can choose not to share your location**
- can tell your browser to remember your choice so you never see this infobar again on this website**



Geolocation






getCurrentPosition() is single function which takes callback function **show_map**. Callback function looks like :

```
function show_map(position) {  
    var latitude = position.coords.latitude;  
    var longitude = position.coords.longitude;  
    // let's show a map or do something interesting!  
}
```



This position object provides information like latitude , longitude , accuracy , speed etc.

Canvas

-  The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.
-  Canvas gives an easy and powerful way to draw graphics using Javascript.
-  For each canvas element you can use a "context" (think about a page in a drawing pad), into which you can issue JavaScript commands to draw anything you want.



Basics of Canvas

```
<canvas id="myCanvas" width="300" height="150">
Fallback content, in case the browser does not support Canvas.
</canvas>

<script>
// Get a reference to the element.
var elem = document.getElementById('myCanvas');

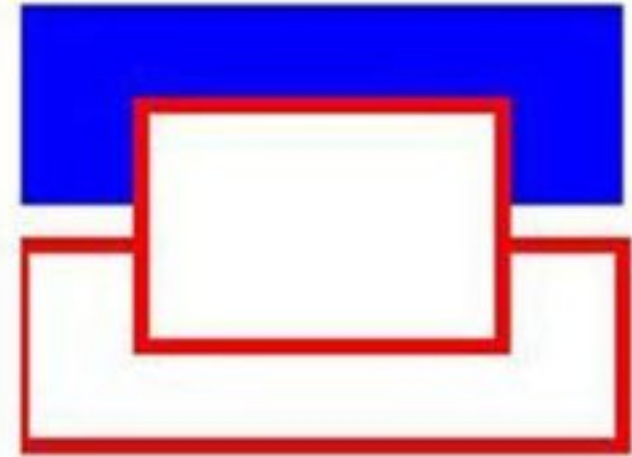
// Always check for properties and methods, to make
// sure your code doesn't break
// in other browsers.
if (elem && elem.getContext) {
    // Get the 2d context.
    // Remember: you can only initialize one context per element.
    var context = elem.getContext('2d');
    if (context) {
        // You are done! Now you can draw your first rectangle.
        // You only need to provide the (x,y) coordinates, followed
by
        // the width and
        // height dimensions.
        context.fillRect(0, 0, 150, 100);
    }
}
</script>
```

Canvas

Basic lines and strokes

```
context.fillStyle    = '#00f'; // blue
context.strokeStyle  = '#f00'; // red
context.lineWidth    = 4;
```

```
// Draw some rectangles.
context.fillRect(0, 0, 150, 50);
context.strokeRect(0, 60, 150, 50);
context.clearRect(30, 25, 90, 60);
context.strokeRect(30, 25, 90, 60);
```





Canvas



Paths

```
// Set the style properties.  
context.fillStyle = '#00f';  
context.strokeStyle = '#f00';  
context.lineWidth = 4;  
  
context.beginPath();  
// Start from the top-left point.  
context.moveTo(10, 10); // give the (x,y) coordinates  
context.lineTo(100, 10);  
context.lineTo(10, 100);  
context.lineTo(10, 10);  
  
// Done! Now fill the shape, and draw the stroke.  
// Note: your shape will not be visible until you call  
any of the two methods.  
context.fill();  
context.stroke();  
context.closePath();
```





Text in Canvas

```
context.fillStyle    = '#00f';  
context.font         = 'italic 30px sans-serif';  
context.textBaseline = 'top';  
context.fillText('Hello world!', 0, 0);  
context.font         = 'bold 30px sans-serif';  
context.strokeText('Hello world!', 0, 50);
```

Hello world!

Hello world!



Data Storage



Current problems with COOKIES

- ❖ **Low size:** Cookies generally have a maximum size of around 4KB, which is not much good for storing any kind of complex data.
- ❖ **It's difficult for cookies to keep track of two or more transactions on the same site, which might be happening in two or more different tabs.**
- ❖ **Cookies can be exploited using techniques such as cross site scripting, resulting in security breaches.**



Data Storage



Storage options

HTML5 gives two options when it comes to storing data on the client side :-



Web Storage – supported in all latest browsers.



Web SQL Database – supported in Opera , Chrome and Safari



Data Storage



Web Storage

- ❖ Web Storage spec was designed as a better way of storing data on the client-side until the end of a session (Session Storage) , or beyond (Local Storage).
- ❖ It has two different types of storage:
 - ❖ Session Storage
 - ❖ Local Storage
- ❖ Able to store around 5MB of data per domain



Data Storage



Session Storage

➤ **Purpose :** To remember all the data in your session, and forget it as soon you close the tab (or window) you are working on.

Setting Data :

```
sessionStorage.setItem(yourkey, yourvalue);
```

Retrieving Data :

```
item = sessionStorage.getItem(yourkey);
```

Removing Data :

```
sessionStorage.removeItem(yourkey);
```

Clearing Data :

```
sessionStorage.clear();
```




Data Storage



Local Storage

- **Local Storage is used if you want the data to be stored for more than a single session.**
- **When a page uses Local Storage, the page (or window) can be closed and re-opened, and still show the saved data — it provides persistent storage.**
- **Local Storage API uses same function names as of Session Storage like `getItem()` , `setItem()` `clear()`, `removeItem()`.**



Data Storage



Using Storage event

The specification also provides for the storage event to be fired whenever the storage area changes. This provides various useful attributes such as:

- ❖ **storageArea:** Tells which kind of storage it is (Session or Local)
- ❖ **key:** The key which is being changed.
- ❖ **oldValue:** The old value of the key
- ❖ **newValue:** The new value of the key
- ❖ **url:** The URL of the page whose key is changed.



Videos



- **HTML5 provides a standardised way to play video directly in the browser, with no plugins required.**
- **No `<object>` , `<embed>` elements required.**
- **`<video>` elements can be styled with CSS**
- **HTML5 can be tweaked and redisplayed onto `<canvas>` with Javascript.**



Videos



Usage of video element

```
<video src=turkish.ogv></video>
```



Fallback markup between the tags, for older Web browsers that do not support native video

```
<h1>Video and legacy browser fallback</h1>
```

```
<video src=leverage-a-synergy.ogv>
```

```
    Download the <a href=leverage-a-synergy.ogv>How to  
    leverage a synergy video</a>
```

```
</video>
```

Video and legacy browser fallback



Videos

Attributes supported



autoplay

```
<video src=leverage-a-synergy.ogv autoplay>  
</video>
```

It tells the browser to play the video or audio automatically.



controls

```
<video src=leverage-a-synergy.ogv controls>  
</video>
```

Provides
play/ pause toggle,
a seek bar, and
volume control.



Videos

Attributes supported



poster

```
<video src=leverage-a-synergy.ogv poster >  
</video>
```

The poster attribute points to an image that the browser will use while the video is downloading, or until the user tells the video to play.



height , width

These attributes tell the browser the size in pixels of the video.

Videos

Attributes supported



loop

```
<video src=leverage-a-synergy.ogv loop >  
</video>
```

It loops the media playback.



preload (preload=auto)

Indicates browser that it should begin downloading the entire file.

Videos

Attributes supported



preload = none

```
<video src=leverage-a-synergy.ogv preload=none >  
</video>
```

This state suggests to the browser that it shouldn't preload the resource until the user activates the controls.



preload = metadata

This state suggests to the browser that it should just prefetch metadata but that it shouldn't download anything.

Videos

Attributes supported

Multiple <source> elements

```
<video controls>
  <source src=leverage-a-synergy.ogv type='video/ogg;
    codecs="theora, vorbis"'>
  <source src=leverage-a-synergy.mp4 type='video/mp4;
    codecs="avc1.42E01E, mp4a.40.2"'>
  <p>Your browser doesn't support video.
  Please download the video in <a href=leverage-a-
  synergy.ogv>Ogg</a> or <a href=leverage-a-
  synergy.mp4>mp4</a> format.</p>
</video>
```