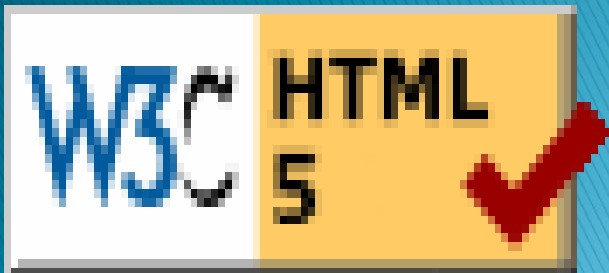
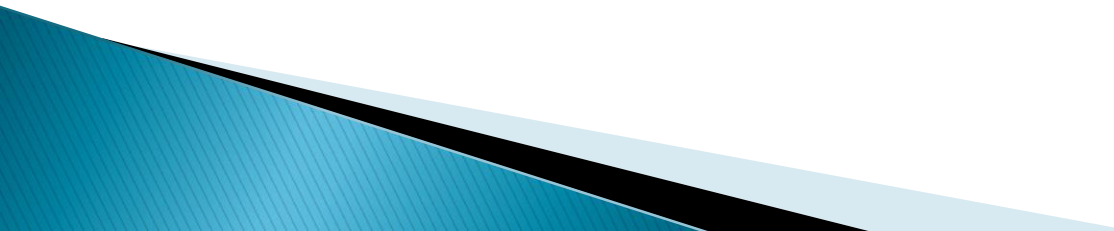


GETTING STARTED WITH HTML5



Topics

- History, Vision & Future of HTML5
 - Getting Started With HTML5
 - Structure of a Web Page
 - Forms
 - Audio and Video
 - HTML5 Canvas
 - Introduction to Data Storage
- 

What is HTML5?

- HTML5 will be the new standard for HTML, XHTML, and the HTML DOM.
- The previous version of HTML came in 1999. The web has changed a lot since then.
- HTML5 is still a work in progress. However, most modern browsers have some HTML5 support.

HTML5 \approx HTML + CSS + JS



How Did HTML5 Get Started

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).


WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like Flash)
- Better error handling
- More markup to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

New Features


Some of the most interesting new features in HTML5 :

- The canvas element for drawing
 - The video and audio elements for media playback
 - Better support for local offline storage
 - New content specific elements, like article, footer, header, nav, section
 - New form controls, like calendar, date, time, email, url, search
- 

Browser Support

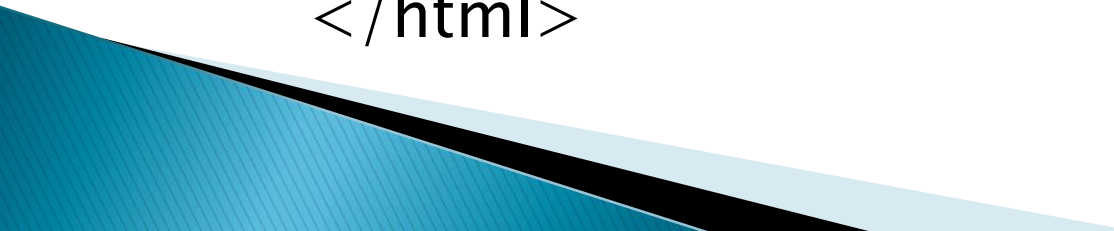
- HTML5 is not yet an official standard, and no browsers have full HTML5 support.
- But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

History, Vision & Future of HTML5

- December 1997: HTML 4.0 is published by the W3C
 - February - March 1998: XML 1.0 is published
 - December 1999 - January 2000: ECMAScript 3rd Edition, XHTML 1.0 (Basically HTML tags reformulated in XML) and, HTML 4.01 recommendations are published
 - May 2001: XHTML 1.1 recommendation is published
 - August 2002: XHTML 2.0 first working draft is released.
 - December 2002: XHTML 2.0 second working draft published.
 - January 2008: First W3C working draft of HTML5 is published!!
- 

Creating your first HTML5 page

```
<!doctype html>
<head>
  <meta charset="utf-8">
  <title>Our First HTML5 Page</title>
</head>
  <body>
    <header>
      <h1>Welcome, one and all to HTML5
      webpage design!</h1>
    </header>
  </body>
</html>
```



New and Updated HTML5 Elements

HTML5 introduces 28 new elements:

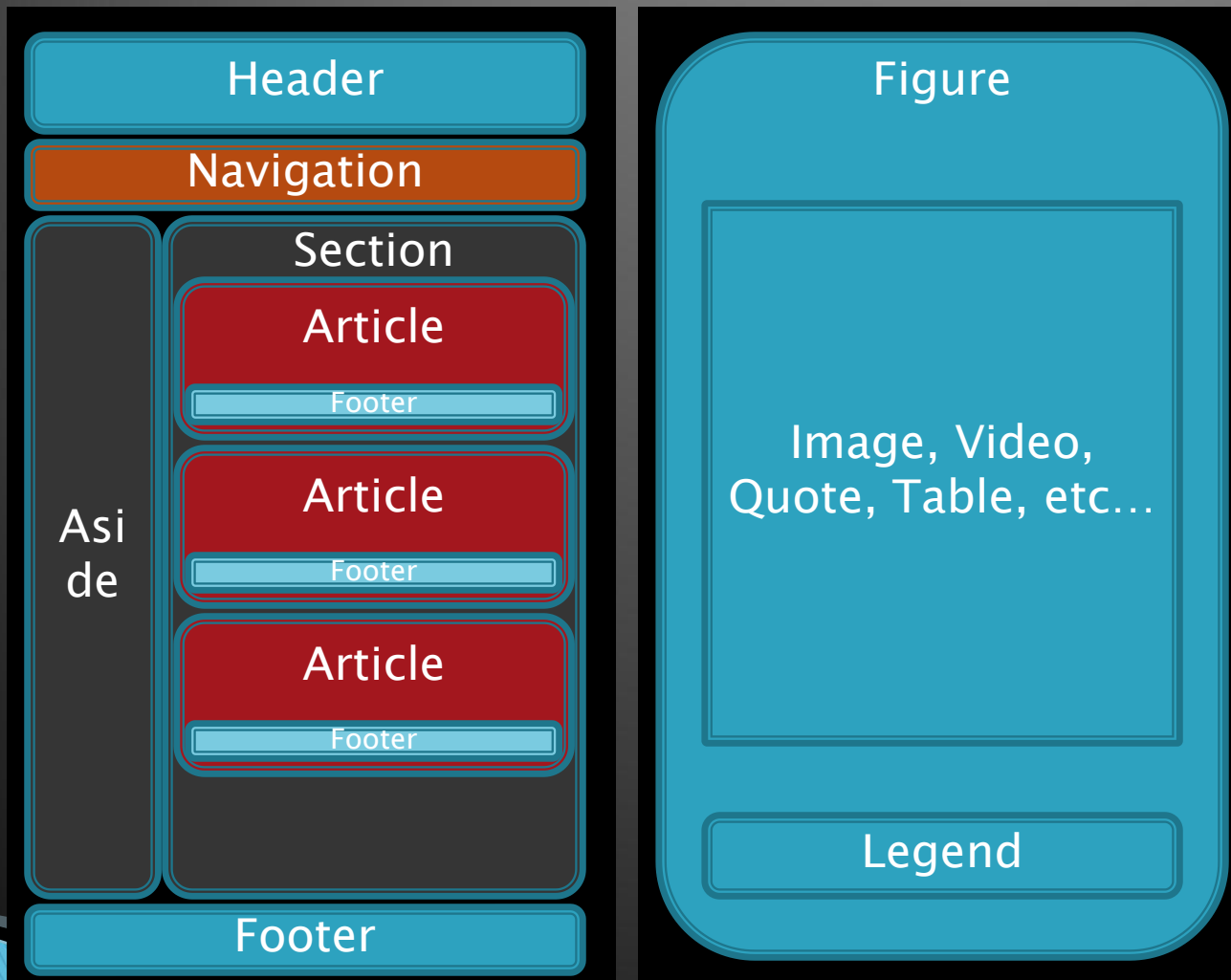
`<section>`, `<article>`, `<aside>`, `<hgroup>`, `<header>`, `<footer>`, `<nav>`,
`<figure>`, `<figcaption>`, `<video>`, `<audio>`, `<source>`, `<embed>`,
`<mark>`, `<progress>`, `<meter>`, `<time>`, `<ruby>`, `<rt>`, `<rp>`, `<wbr>`,
`<canvas>`, `<command>`, `<details>`, `<summary>`, `<datalist>`, `<keygen>`
and `<output>`

An HTML page first starts with the DOCTYPE declaration

HTML5 also update some of the previous existing elements to better reflect how they are used on the Web or to make them more useful such as:

- ❑ The `<a>` element can now also contain flow content instead of just phrasing content
- ❑ The `<hr>` element is now representing a paragraph-level thematic break
- ❑ The `<cite>` element only represent the title of a work
- ❑ The `` element is now representing importance rather than strong emphasis

HTML



Page-Structure Elements

- ▶ HTML5 introduces several new page-structure elements that meaningfully identify areas of the page as headers, footers, articles, navigation areas, asides, figures and more.

Past, Present, & Future of HTML

Review

► **HTML**

vs

XHTML

Uppercase or lowercase tags
All elements don't have to be closed
Empty elements don't require closing
Does not require DOCTYPE declaring
Attribute values do not require quotes

Uppercase or lowercase tags
All elements don't have to be closed
Empty elements don't require closing
Does not require DOCTYPE declaring
Attribute values do not require quotes

HTML

vs

HTML5

Understood by all browsers
Not originally built to handle video
Does not support local offline storage
Not built for today's internet needs
Does not support creation of a canvas for drawing

Not fully functional with all browsers
Built for audio & video
Supports local offline storage
Supports content specific tags
Supports creation of a canvas for drawing

<nav> tag

<nav>: Represents a major navigation block. It groups links to other pages or to parts of the current page.

<nav> does not have to be used in every place you can find links. For instance, footers often contains links to terms of service, copyright page and such, the <footer> element would be sufficient in that case

```
<nav>
<ul>
<li><a href="/">Home</a></li>
<li><a href="/events">Current Events</a></li>
<li><a href="/contact">Contact us</a></li>
</ul>
</nav>
```

<header> tag

<Header>: tag specifies a header for a document or section.

However, we mustn't think that "header" is only for masthead of a website. "header" can be use as a heading of an blog entry or news article as every article has its title and published date and time

```
<article>
<header>
<h1>Military Offers Assurances to Egypt and Neighbors</h1>
Published : <time datetime="2011-02-13" pubdate>February 13 2011</time>
</header>
<p>
CAIRO - As a new era dawned in Egypt on Saturday,
the army leadership sought to reassure Egyptians and
the world that it would shepherd a transition to civilian
rule and honor international commitments like the peace treaty with Israel.
</p>
</article>
```

<article> tag

<article>: The web today contains a ocean of news articles and blog entries. That gives W3C a good reason to define an element for article instead of <div class="article">.

We should use article for content that we think it can be distributable. Just like news or blog entry can we can share in RSS feed

You can have header and footer element in an article. In fact, it is very common to have header as each article should have a title.

```
<body>
<h1>My blog</h1>
<article>

  <header>
    <h1>The Very First Rule of Life</h1>
    <p><time pubdate datetime="2009-10-09T14:28-08:00"></time></p>
  </header>

  <p>If there's a microphone anywhere near you, assume it's hot and
  sending whatever you're saying to the world. Seriously.</p>
  <p>...</p>

  <footer>
    <a href="?comments=1">Show comments...</a>
  </footer>

</article>
</body>
```

<progress> tag

<Progress>: The new "progress" element appears to be very similar to the "meter" element. It is created to indicate progress of a specific task.

The progress can be either determinate OR indeterminate. Which means, you can use "progress" element to indicate a progress that you do not even know how much more work is to be done yet.

Progress EX : `<progress value="60" max="100">60%</progress>`

Progress of Task A : 

Browsers	Progress element support
IE 9 Beta	
Firefox 13	✓
Safari 5	
Chrome 8	✓
Opera 11	✓

Example

```
<!doctype html>
```

```
<body>
```

Your score is:

```
<progress>working...</progress>
```

Download is:

```
<progress value="75" max="100">3/4  
  complete</progress>
```

```
</body>
```

```
<html>
```



<meter>tag

<meter>: "Meter" is a new element in HTML5 which represent value of a known range as a gauge. The keyword here is "known range". That means, you are only allowed to use it when you are clearly aware of its minimum value and maximum value.

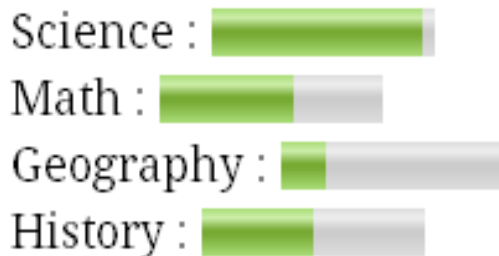
<meter min="0" max="10" value="8">8 of 10</meter>.

```
Science : <meter min="0" max="100" value="95">95 of 100</meter> <br />
```

```
Math : <meter min="0" max="100" value="60">60 of 100</meter><br />
```

```
Geography : <meter min="0" max="100" value="20">20 of 100</meter> <br />
```

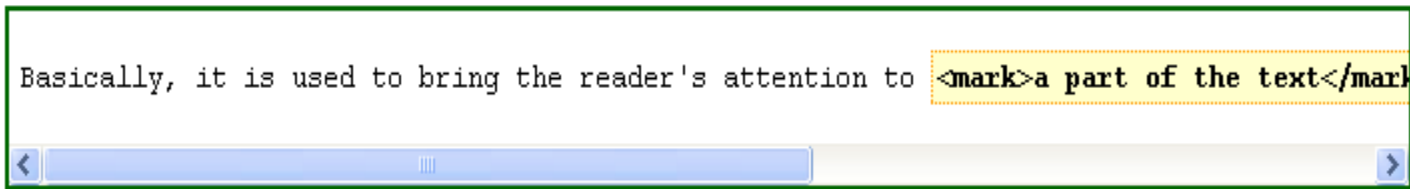
```
History : <meter min="0" max="100" value="50">50 of 100</meter>
```



Browsers	Render meter element as gauge
IE 9 Beta	
Firefox 13	
Safari 5	
Chrome 8	✓
Opera 11	✓

<mark> & <figure> tag

<mark>: The mark <mark> element represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.



<figure>: The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document

```
<figure>
  
</figure>
```

Key Points

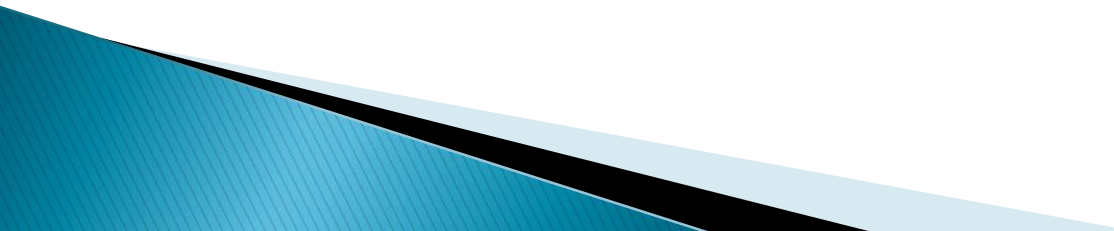
- ▶ To specify HTML5 as the document type, add `<!DOCTYPE html>` at the beginning of the file.
- ▶ All the HTML coding in a document (except the *DOCTYPE*) is enclosed within a two-sided `<html>` tag.
- ▶ The `<html>` and `</html>` tags enclose the `<head>` and `<body>` sections.
- ▶ The `<head>` area contains the page title (`<title>`) and any `<meta>` tags. The `<body>` area contains all the displayable text for the page.
- ▶ Enclose each paragraph in a two-sided `<p>` tag. Most browsers add space between paragraphs when displaying the page.
- ▶ To create a line break without starting a new paragraph, use the one-sided `
` tag.
- ▶ When coding for XHTML, end one-sided tags with a space and a slash (`/`). The space is required for recognition in HTML, and the slash is necessary for recognition in XHTML.
- ▶ Use `<meta>` tags in the `<head>` section to indicate keywords and the document encoding language.
- ▶ Use the `<title>` and `</title>` tags to enclose the text that should appear in the browser's title bar. Place these in the `<head>` section of the file.

HTML 5 Forms

Introduction to HTML5 Forms

HTML5 web forms have introduced new form elements, input types, attributes, and other features. Many of these features we've been using in our interfaces for years: form validation, combo boxes, placeholder text, and the like. The difference is that where before we had to resort to JavaScript to create these behaviors, they're now available directly in the browser; all you need to do is set an attribute in your markup to make them available.

HTML5 not only makes marking up forms easier on the developer, it's also better for the user. With client-side validation being handled natively by the browser, there will be greater consistency across different sites, and many pages will load faster without all that redundant JavaScript.



<form> / <input> 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 complete values based on values that the user has entered before.

```
<!DOCTYPE html>
```

```
<html><body>
```

```
<form action="demo_form.asp" autocomplete="on">
```

```
  First name:<input type="text" name="fname"><br>
```

```
  Last name: <input type="text" name="lname"><br>
```

```
  E-mail: <input type="email" name="email"  
    autocomplete="off"><br>
```

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

```
</form></body></html>
```

The **novalidate** attribute is a boolean attribute. It specifies that the form-data (input) should not be validated when submitted.

```
<form action="demo_form.asp" novalidate>
```

```
E-mail: <input type="email" name="user_email">
```

```
<input type="submit"> </form>
```

a) Form rendered in Firefox before the user interacts with it

Firefox

New HTML5 autocomplete Attribute an... +

file:///C:/books/2011/TW3HTP5/examples/ ☆ Google

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: Select a month

b) autocomplete automatically fills in the data when the user returns to a form submitted previously and begins typing in the First Name input element; clicking Jane inserts that value in the input

Firefox

New HTML5 autocomplete Attribute an... +

file:///C:/books/2011/TW3HTP5/examples/ ☆ Google

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: Select a month

c) autocomplete with a datalist showing the previously entered value (June) followed by all items that match what the user has typed so far; clicking an item in the autocomplete list inserts that value in the input

Firefox

New HTML5 autocomplete Attribute an... +

file:///C:/books/2011/TW3HTP5/examples/ ☆ Google

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: j

- June
- January
- June
- July

datalist values filtered by what's been typed so far

<input> autofocus Attribute

The autofocus attribute is a boolean attribute. It specifies that an <input> element should automatically get focus when the page loads.

Only one form element can have autofocus in a given page.

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<form action="/cgi-bin/html5.cgi" method="get">
```

```
Enter Department : <input type="text" name="dept" />
```

```
Enter email : <input type="text" name="newinput" autofocus />
```

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

```
</form>
```

```
</body>
```

```
</html>
```



<input> 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.
- **Note:** The **formaction** attribute is used with type="submit" and type="image"

<input> 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.
- **Note:** The **formmethod** attribute can be used with type="submit" and type="image".

<input> list Attribute

- Data lists are currently only supported in Firefox and Opera, but they are very cool.
- They fulfill a common requirement: a text field with a set of predefined autocomplete options.

```
<!doctype html>
```

```
<body>
```

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

```
<datalist id="cars">
```

```
  <option value="BMW"/>
```

```
  <option value="Bolero"/>
```

```
  <option value="Ford"/>
```

```
  <option value="Volvo"/>
```

```
</datalist>
```

```
</body>
```

```
</html>
```

<input> required Attribute

- The Boolean required attribute tells the browser to only submit the form if the field in question is filled out correctly. Obviously, this means that the field can't be left empty, but it also means that, depending on other attributes or the field's type, only certain types of values will be accepted.
- If a required field is empty or invalid, the form will fail to submit, and focus will move to the first invalid form element.
- The required attribute can be set on any input type except button, range, color, and hidden, all of which generally have a default value.

```
<!DOCTYPE HTML>
```

```
<html> <body>
```

```
<form action="/cgi-bin/html5.cgi" method="get">
```

```
Enter Department : <input type="text" name="dept" />
```

```
Enter email : <input type="text" name="newinput" required/>
```

```
<p>Try to submit using Submit button</p>
```

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

```
</form> </body> </html>
```



The screenshot shows a web form with two input fields. The first field is labeled "Email:" and contains the placeholder text "(name@domain.com)". The second field is labeled "Month:" and contains the placeholder text "y-mm)". A red error message box is displayed below the "Email:" field, stating "Please fill out this field.".

<input> placeholder Attribute

- The placeholder attribute allows a short hint to be displayed inside the form element, space permitting, telling the user what data should be entered in that field.
- The placeholder text disappears when the field gains focus, and reappears on blur if no data was entered.
- Developers have provided this functionality with JavaScript for years, but in HTML5 the placeholder attribute allows it to happen natively, with no JavaScript required.

<!DOCTYPE HTML>

<html> <body>

<form action="/cgi-bin/html5.cgi" method="get">

**Enter email : <input type="email" name="newinput"
placeholder="email@example.com"/>**

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

</form> </body> </html>

```
$("#input[placeholder], textarea[placeholder]").each(function() {  
    if($(this).val()==""){  
        $(this).val($(this).attr("placeholder"));  
        $(this).focus(function(){  
            if($(this).val()==$(this).attr("placeholder")) {  
                $(this).val("");  
                $(this).removeClass('placeholder');  
            }  
        });  
        $(this).blur(function(){  
            if($(this).val()==""){  
                $(this).val($(this).attr("placeholder"));  
                $(this).addClass('placeholder');  
            }  
        });  
    }  
});
```

<input> placeholder Attribute example

a) Text field with gray placeholder text



b) placeholder text disappears when the text field gets the focus



<input> multiple Attribute

The multiple attribute, if present, indicates that multiple values can be entered in a form control. While it has been available in previous versions of HTML, it only applied to the select element. In HTML5, it can be added to email and file input types as well.

If present, the user can select more than one file, or include several comma-separated email addresses.

At the time of writing, multiple file input is only supported in Chrome, Opera, and Firefox.

Select images: `<input type="file" name="img" multiple="multiple" />`

<form> input types

Introduction

you're familiar with: checkbox, text, password, and submit. Here's the full list of types that were available before HTML5:

■ button ■ checkbox ■ file ■ hidden ■ image ■ password ■ radio ■ reset ■ submit ■ text

HTML5 gives us input types that provide for more data-specific UI elements and native data validation. HTML5 has a total of 13 new input types:

search

email

url

tel

datetime

date

month

week

Time

datetime-

local

number

range

Color

<input type> search

- The search type is used for search fields (a search field behaves like a regular text field).
- Search type is only supported in Chrome, Opera, and safari.

Search Google: <input type="search" name="googlesearch" />

<input type="email"> email

- The email type (`type="email"`) is used for specifying one or more email addresses.
- It supports the Boolean `multiple` attribute, allowing for multiple, comma-separated email addresses.
- Define a field for an e-mail address (will be automatically validated when submitted):



| Validating an e-mail address in Chrome.

<input type="url">

- The url type is used for input fields that should contain a URL address.
- The value of the url field is automatically validated when the form is submitted.

Add your homepage: `<input type="url" name="homepage" />`



Validating a URL in Chrome.

<input type="tel">

For telephone numbers, use the tel input type (type="tel"). Unlike the url and email types, the tel type doesn't enforce a particular syntax or pattern. Letters and numbers—indeed, any characters other than new lines or carriage returns—are valid.

Telephone: <input type="tel" name="usrtel" />

<input type="tel" placeholder="(555) 555-5555"
pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.*?\$" />



`<input type="datetime,datetime-local">`

- ▶ The `datetime-local` type allows the user to select a date and time (no time zone).
- ▶ The `datetime` type allows the user to select a date and time (with time zone).

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<form action="/cgi-bin/html5.cgi" method="get">
```

```
Local Date and Time : <input type="datetime-local"
name="newinput" />
```

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

```
</form>
```

```
</body>
```

```
</html>
```

A date chooser control.



<input type> range, number

The range input type (type="range") displays a slider control in browsers. As with the number type, it allows the min, max, and step attributes.

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<form action="/cgi-bin/html5.cgi" method="get">
```

```
Select Range : <input type="range" min="0" max="10" step="1"
value="5" name="newinput" />
```

```
Select Number : <input type="number" min="0" max="10" step="2"
value="5" name="newinput" />
```

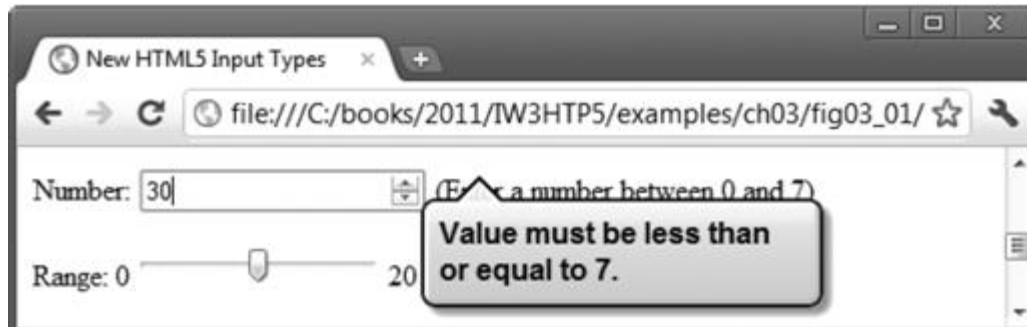
```
<input type="submit" value="submit" />
```

```
</form>
```

```
</body>
```

```
</html>
```

number, range example



<input type> output

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

```
<!DOCTYPE HTML>
<html>
<script type="text/javascript">
function showResult()
{
    x = parseInt(document.forms["myform"]["newinput"].value);
    y = parseInt(document.forms["myform"]["newinput1"].value);
    var sum;
sum = x+y;
    document.forms["myform"]["result"].value=sum;
}
</script>
<body>
<form action="/cgi-bin/html5.cgi" method="get" name="myform">
Enter a value : <input type="text" name="newinput" />
Enter a value : <input type="text" name="newinput1" />
<input type="button" value="Result" onclick="showResult();" />
<output name="result"/>
</form>
</body>
</html>
```

Enter a value :

Enter a value :

55

<input type>keygen

The purpose of the <keygen> element is to provide a secure way to authenticate users.

The keygen element represents a key pair generator control. When the control's form is submitted, the private key is stored in the local keystore, and the public key is packaged and sent to the server.

The public key could be used to generate a client certificate to authenticate the user in the future.

```
<form action="demo_keygen.asp" method="get">  
  Username: <input type="text" name="usr_name" />  
  Encryption: <keygen name="security" />  
  <input type="submit" />  
</form>
```

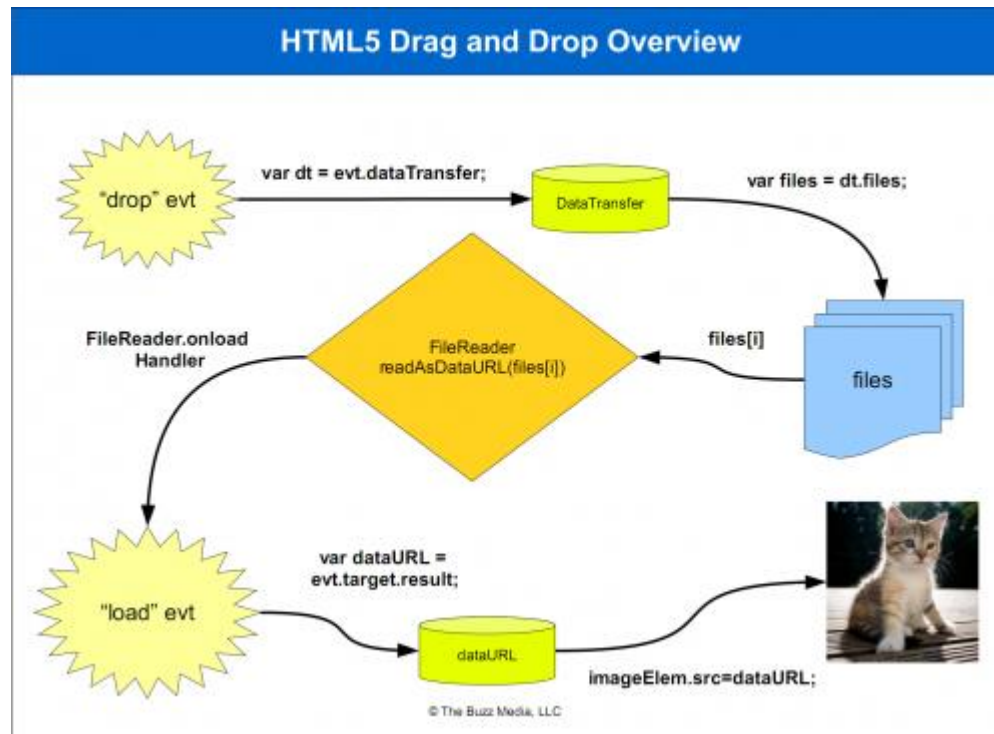
Drag and drop

- ▶ Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

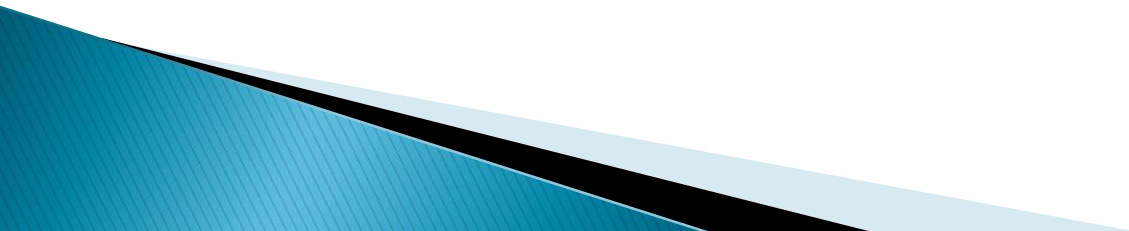
Demo

- ▶ You Drag Files Into the Browser From the Desktop with HTML5
- ▶ [Demo_drag_files](#)

Understanding the HTML5 DnD and File API



HTML 5 Audio + Video



<audio> and <video> tags

- ▶ HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.
- ▶ **Supported audio codecs:**
 - Chrome 3+ supports .ogg and .wav
 - Firefox 3.5+ supports .ogg and .wav
 - Opera 10.5+ supports .ogg and .wav
 - Safari 4+ supports .mp3 and .wav

<audio controls>

 <source src="horse.ogg" type="audio/ogg">

 </audio>

<video width="480" height="270" poster="html5-video-element-test.mp4" controls>

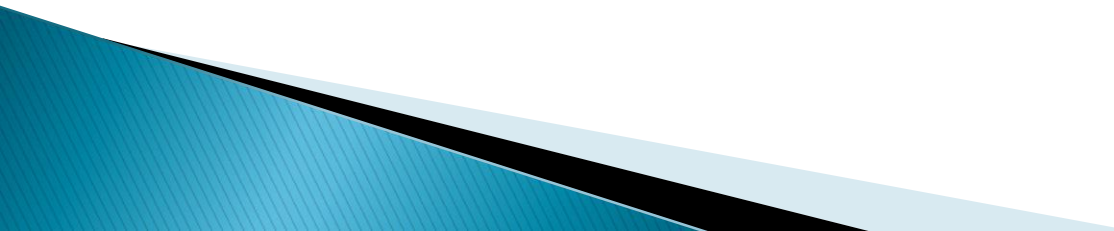
 <source src="html5-video-element-test.mp4"
 type="video/mp4">

</video>

[Demo_audio_video](#)

HTML 5 Canvas

What is Canvas?

- ▶ The HTML5 `<canvas>` element is used to draw graphics, on the fly, via scripting (usually JavaScript).
 - ▶ The `<canvas>` element is only a container for graphics. You must use a script to actually draw the graphics.
 - ▶ Canvas has several methods for drawing paths, boxes, circles, text, and adding images.
- 

Drawing a canvas

We'll add the function to our script element. The first step is to grab hold of the canvas element on our page:

canvas/demo1.html ()

```
<script>
:
function draw() {
  var canvas = document.getElementById("myCanvas");
}
</script>
```

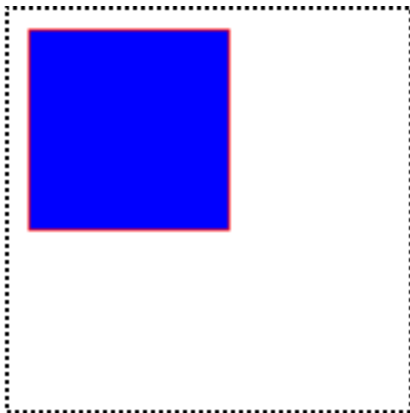
We obtain our drawing context by calling the `getContext` method and passing it the string "2d", since we'll be drawing in two dimensions:

canvas/demo1.html

```
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");
}
```


filling brush with color

```
<script>  
function draw(){  
    var canvas=document.getElementById('myCanvas');  
    var context=canvas.getContext('2d');  
    context.strokeStyle = "red";  
    context.fillStyle = "blue";  
    context.fillRect(10,10,100,100);  
    context.strokeRect(10,10,100,100);  
}  
</script>
```



Example: [demo1.html](#)

Create circle using canvas

```
<script type="text/javascript">  
var c=document.getElementById( "myCanvas" );  
var ctx=c.getContext( "2d" );  
ctx.fillStyle="#FF0000";  
ctx.beginPath();  
ctx.arc(70,18,15,0,Math.PI*2,true);  
ctx.closePath();  
ctx.fill();  
</script>
```

Output



Create linear gradient

```
<script type="text/javascript">
```

```
var c=document.getElementById("myCanvas");  
var ctx=c.getContext("2d");  
var grd=ctx.createLinearGradient(0,0,175,50);  
grd.addColorStop(0,"#FF0000");  
grd.addColorStop(1,"#00FF00");  
ctx.fillStyle=grd;  
ctx.fillRect(0,0,175,50);
```

```
</script>
```

Output



HTML 5 Web Storage

Introduction

The Web Storage API defines a standard for how we can [save simple data locally](#) on a user's computer or device. Before the emergence of the Web Storage standard, web developers often stored user information in [cookies, or by using plugins](#). With Web Storage, we now have a standardized definition for how to store up to [5MB of simple data created by our websites](#) or web applications. Better still, Web Storage already works in Internet [Explorer 8.0!](#)

Web Storage is a great complement to [Offline Web Applications](#), because you need somewhere to store all that user data while you're working offline, and Web Storage provides it.

Two kinds of storage


session Storage

Session storage lets us keep track of [data specific to one window or tab](#). It allows us to isolate information in each window. Even if the user is visiting the same site in two windows, each window will have its own individual session storage object and thus have separate, distinct data.

Session storage [is not persistent—it only lasts for the duration of a user's session](#) on a specific site (in other words, for the time that a browser window or tab is open and viewing that site).

Local Storage

Unlike session storage, local storage allows us to [save persistent data](#) to the user's computer, via the browser. When a user revisits a site at a later date, any data saved to local storage can be retrieved.



local storage v/s cookies

Local storage can at first glance seem to play a similar role to HTTP cookies, but there are a few key differences. First of all, **cookies are intended to be read on the server side**, whereas **local storage is only available on the client side**.

If you need your server-side code to react differently based on some saved values, cookies are the way to go. Yet, cookies are sent along with each HTTP request to your server and this can result in significant **overhead in terms of bandwidth**.

Local storage, on the other hand, just sits on the **user's hard drive** waiting to be read, so it costs nothing to use.

In addition, we have significantly more size to store things using local storage. With **cookies, we could only store 4KB** of information in total. With **local storage**, the maximum is **5MB**.



Session : example

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
  <script type="text/javascript">
```

```
    if( sessionStorage.hits ){
```

```
      sessionStorage.hits = Number(sessionStorage.hits) + 1;
```

```
    }else{
```

```
      sessionStorage.hits = 1;
```

```
    }
```

```
    document.write("Total Hits :" + sessionStorage.hits );
```

```
  </script>
```

```
  <p>Refresh the page to increase number of hits.</p>
```

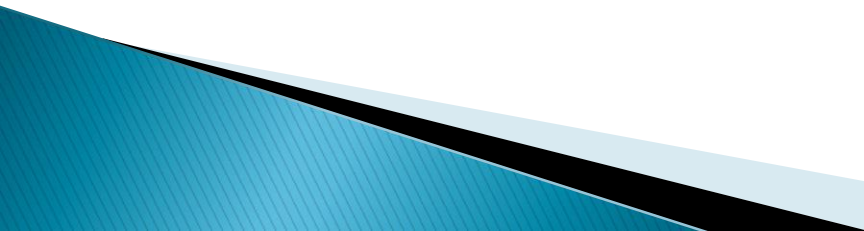
```
</body>
```

```
</html>
```



LocalStorage: example

```
<!DOCTYPE HTML>
<html>
<body>
  <script type="text/javascript">
    if( localStorage.hits ){
      localStorage.hits = Number(localStorage.hits) + 1;
    }else{
      localStorage.hits = 1;
    }
    document.write("Total Hits :" + localStorage.hits );
  </script>
  <p>Refresh the page to increase number of hits.</p>
  <p>Close the window and open it again and check the result.</p>
</body>
</html>
```



Exercise-1

- ▶ Setup the basic HTML document structure (i.e. doctype, html, head, meta, title, body, footer, article etc.)



Exercise-2

- ▶ Your task is to design the form elements used to create the form shown in Figure-1 in the next slide.
- ▶ You must use the HTML5 tags that are appropriate to replicate the form and fulfill all the specifications listed.
- ▶ Code the form with autocomplete active.
- ▶ Given the image shown in Figure 1, it is easy to see that two field sets are used to create the main structure of the form. Your task is to create the field sets, including the names *Customer Info* and *Books*. Don't worry about the content fields for the moment.
- ▶ The **Name** field you create should have autofocus, placeholder text, and be required.
- ▶ The **Telephone** field should have placeholder text, a pattern to restrict entry, and be required.
- ▶ The **Email address** field should have placeholder text and allow multiple entries. This field should also be required.
- ▶ The **Books** field should have a data list. You can select the content you would like to list.
- ▶ The **Quantity (Maximum 5)** field should have a minimum value of 1 and a maximum value of 5.

Form-1

A Simple Form

Form Fundamentals

Customer Info

Name:

Telephone:

Email address:

Books

Quantity (Maximum 5):

© 2011 Acme United. All rights reserved.