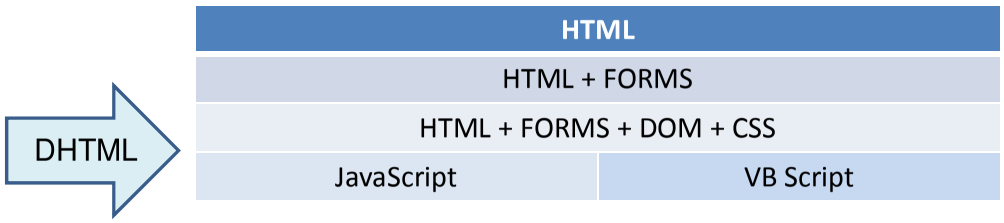
Content

- Introduction
- **User Interaction Client-Side**
  - **HTML and DHTML**
  - **DOM, Scripting**
- Building ASP.NET Web Pages
- ASP.NET Web Controls, Themes and Master Pages
- ASP.NET State Management Techniques
- ADO.NET
- AJAX
- Validating Data
- Deploying Your Application

HTML and DHTML

Moving to the interaction



Dynamic HTML (DHTML)

- According to the World Wide Web Consortium (W3C): *"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."*
- The W3C HTML 4 standard has rich support for dynamic content:
  - **HTML** supports **JavaScript**
  - **HTML** supports the Document Object Model (**DOM**)
  - **HTML** supports HTML **Events**
  - **HTML** supports Cascading Style Sheets (**CSS**)

HTML: HyperText Markup Language

- HTML is a language for describing web pages. It was created by Tim Berners-Lee, who wrote the browser and server software in the last part of 1990. It is not a programming language, it is a **markup language**.

- HTML is the predominant markup language for web pages. It provides a means to create structured documents by denoting structural semantics for text such as **headings**, **paragraphs**, **lists** etc as well as for links, quotes, and other items. It allows images and objects to be embedded and can be used to create interactive forms. It is written in the form of elements consisting of **"tags" surrounded by angle brackets** (<, >) within the web page content.

- HTML can include or can load scripts in languages such as JavaScript which affect the behavior of HTML processors like Web browsers; and Cascading Style Sheets (**CSS**) to define the appearance and layout of text and other material. The **W3C**, maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup.

HTML: New Version…

- **HTML5** is the next major revision of HTML. The Web Hypertext Application Technology Working Group (WHATWG) started work on the specification in June 2004 under the name Web Applications 1.0.

- HTML5 is the proposed next standard for HTML 4.01, XHTML 1.0 and DOM Level 2 HTML. HTML5 is expected to be a game-changer in Web application development, **making obsolete** such **plug-in-based** rich Internet application (RIA) technologies as **Adobe Flash**, Microsoft **Silverlight**, and Sun **JavaFX**.

- The HTML5 specification was adopted as the starting point of the work of the new HTML working group of the W3C in 2007. The working group published the **First Public Working Draft of the specification on January 22, 2008**. The specification is an ongoing work, and is expected to remain so for many years, although parts of HTML5 are going to be finished and implemented in browsers before the whole specification reaches final Recommendation status. The editors are Ian Hickson of Google, Inc. and David Hyatt of Apple, Inc.

- We will be working with **HTML 4.0**

HTML: Example

```
<html>
 <body>
  <h1>My First Heading1 and Page</h1>
  <hr />  <!-- This is a line  -->
  <h2>My First Heading2 and Page</h2>
  <hr />
  <p>My first paragraph</p>
  <p>My second parragraph <br /> with two lines </p>
  <p><b>This text is bold</b></p>
  <p><strong>This text is strong</strong></p>
  <p><big>This text is big</big></p>
  <p><em>This text is emphasized</em></p>
  <p><i>This text is italic</i></p>
  <p><small>This text is small</small></p>
  <p>This is<sub> subscript</sub> and
<sup>superscript</sup></p>
  <p style="font-family:times;color:green">Green Text</p>

  <p><a href="page1.htm">This is a link to Page 1</a></p>
  <p><a href="page2.htm">This is a link to Page 2</a></p>
  <img src="Milhouse.jpg" width="104" height="142" />
 </body>
</html>
```

```
<html>
<body>

<h1>This is Page1</h1>
<p>This is some text.</p>

</body>
</html>
```

```
<html>
<body>

<h1>This is Page2</h1>
<p>This is some text.</p>

</body>
</html>
```


HTML: Tables and Lists

Tables and list are the typical HTML elements to show data:

Table

```
<html>
<body>

<h4>Table headers:</h4>
<table border="1">
<tr>
  <th>Name</th>
  <th>Telephone</th>
  <th>Telephone</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
<tr>
  <td>Victor Sosa</td>
  <td>834 316 66 00</td>
  <td>834 316 66 01</td>
</tr>

</table>
```

Unordered list

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Ordered list

```
<h4>Numbered list:</h4>
<ol>
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```

```
<h4>Letters list:</h4>
<ol type="A">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```

Definition list

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Let's see how these web pages look like

HTML: Forms

- Forms were introduced in HTML 2.0, now are included in all HTML versions, allowing information exchange between users and web servers.
- The user enters information by filling in fields or clicking on buttons, then pressing a submit button to send it to a URL, normally one pointing to a dynamic web script (like ASP, JSP, PHP or CGI script) or an e-mail address .

- Forms are delimited with the **<FORM> ... </FORM>** tag, which contains several form elements, such as buttons and text boxes, and which must possess the following attributes:
  - **METHOD** indicates how replies will be sent "**POST**" is the value that sends the data to the processing agent by storing it in the body of the form, while "**GET**" sends the data by adding it to the URL, separating it from the address with an exclamation mark.

  - **ACTION** indicates the address that the information will be sent to (a CGI script or e-mail address (mailto:email.address@machine)) . An **optional attribute** of the FORM tag is ***ENCTYPE***, which specifies how the form data is encoded in the URL. However, this does not need to be specified, since the default value (application/x-www-form-urlencoded) is the only legal value. The optional attribute ***ACCEPT*** is used to set **MIME types** for the data which the form can send.

HTML: Forms

- This is the **syntax for the FORM tag**:

<FORM METHOD="POST" or "GET" ACTION="url" ENCTYPE="x-www-form-urlencoded">
...
</FORM>

Here are a few examples of FORM tags:

<FORM METHOD=POST ACTION="mailto:webmaster@kioskea.net">

<FORM METHOD=GET ACTION="http://en.kioskea.net/cgi-bin/script.cgi"> **Inside the FORM Tag**

FORMS: Elements

<form name="input" action="" method="get">

Text Fields

```
First name: <input type="text" name="firstname" /> <br />
Last name: <input type="text" name="lastname" />  <br />
```

Radio Buttons

```
<input type="radio" name="sex" value="male" /> Male <br />
<input type="radio" name="sex" value="female" /> Female <br />
```

Checkboxes

```
I have a bike: <input type="checkbox" name="vehicle" value="Bike" /> <br />
I have a car: <input type="checkbox" name="vehicle" value="Car" /> <br />
I have an airplane: <input type="checkbox" name="vehicle" value="Airplane" /> <br />
```

Dropdown Box

```
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```

Submit Button

```
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

Example: Forms_Elements

HTML: CSS

- Cascading Style Sheets (CSS) is a **style sheet language** used to describe the presentation semantics (that is, the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document, including SVG and XUL.

External Style Sheet:

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

Internal Style Sheet:

```
<head>
<style type="text/css">
        body {background-color: red}
        p {margin-left: 20px}
</style>
</head>
```

Inline Style Sheet:

```
<p style="color: red; margin-left: 20px">
This is a paragraph
</p>
```

For a complete CSS tutorial: http://www.csstutorial.net/

HTML: JavaScripts

- Scripts make HTML pages more dynamic and interactive.

```
<html>
<body>
    JavaScript:
    <script type="text/javascript">
        document.write("Hello World!") /
    </script>

VBScript:
<script type="text/vbscript">
    <!-- document.write("Hello World!") -->
</script>

</body>
</html>
```

For old Browsers

```
JavaScript:
<script type="text/javascript">
  <!-- document.write("Hello World!") -->
</script>
<noscript> Your browser does not support JavaScript!</noscript>

 VBScript:
<script type="text/vbscript">
    <!-- document.write("Hello World!") -->
</script>
<noscript> Your browser does not support VBScript!</noscript>
```

More information comes later…

HTML: Character entities

- Some **characters are reserved** in HTML. For example, you cannot use the **greater than** or **less than** signs within your text because the browser could mistake them for markup.
- If we want the browser to actually display these characters we must insert **character entities** in the HTML source.
- A character entity looks like this: ***&entity_name***; OR ***&#entity_number***;
- To display a **less than** sign we must write: **<** or **<**
- The advantage of using an entity name instead of a number is that the name often is easier to remember. However, the disadvantage is that browsers may not support all entity names (while the support for entity numbers is very good).

HTML: Common character entities

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;

Note1: Entity names are case sensitive!
Note2: For a complete reference: http://www.w3.org/TR/html4/sgml/entities.html

DOM: Document Object Model

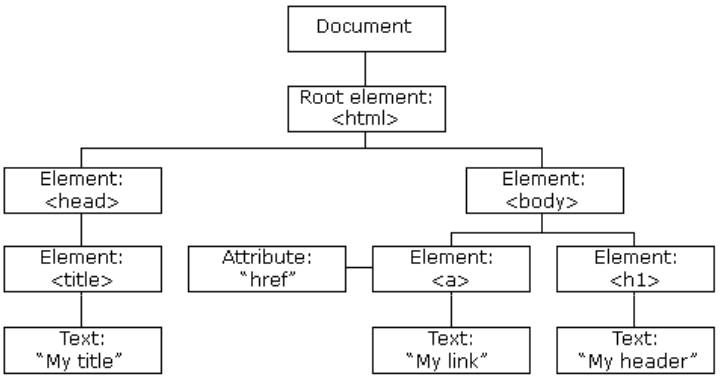
- The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.
- Interprets XML/HTML as a tree of nodes
- Establishes an object interface on a XML/HTML document
- Builds data model in memory
- Enables random access to data
- Good choice when data model has natural tree structure

DOM: Levels

The DOM is separated into 3 different parts / levels:

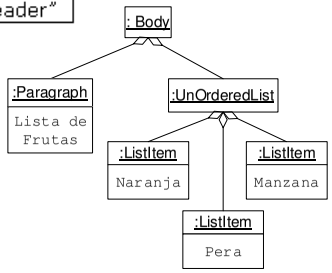
- **Core DOM** - standard model for any structured document
- **XML DOM** - standard model for XML documents
- **HTML DOM** - standard model for HTML documents

HTML DOM Node Tree



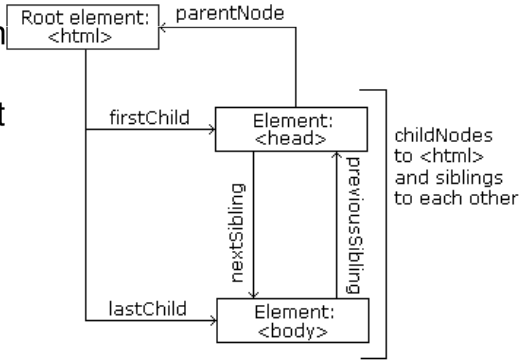
HTML Document:

```
<p>Lista de frutas:</p>
<ul>
        <li>Naranja</li>
        <li>Pera</li>
        <li>Manzana</li>
</ul>
```



Node Parents, Children, and Siblings

- The nodes in the node tree have a hierarchical relationship to each other.
- The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).
- In a node tree, the top node is called the root. Every node, except the root, has exactly one parent node.
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent



DOM: Programming Interface

- Since in the DOM, HTML documents consist of a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we will use JavaScript.
- The programming interface of the DOM is defined by standard properties and methods.:
  - **Properties** are often referred to as something that is (i.e. the name of a node).
  - **Methods** are often referred to as something that is done (i.e. remove a node).
- Some DOM properties:
  - x.innerHTML - the text value of x
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x
  - **Note:** In the list above, x is a node object (HTML element).

HTML DOM Methods

- Some DOM methods:
  - **x.getElementById(*id*)** - get the element with a specified id
  - **x.getElementsByTagName(*name*)** - get all elements with a specified tag name
  - **x.appendChild(*node*)** - insert a child node to x
  - **x.removeChild(*node*)** - remove a child node from x

Example

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
txt=document.getElementById("intro").innerHTML;
document.write("<p>The text from the intro paragraph: " + txt + "</p>");
</script>

</body>
</html>
```

More examples: DOM02, DOM03, DOM04

HTML DOM Properties

- In the HTML DOM, each node is an **object**.
- Objects have methods and properties that can be accessed and manipulated by JavaScript.
- Three important node properties are:
  - nodeName
  - nodeValue
  - nodeType

The most important node types are:

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

See examples:, Dom05

HTML DOM: Using Forms

DOM6a_UsingForms.htm

```
<html>
<head>
<script type="text/javascript">
<!– HERE:  function validate()   - - ->
</head>

<body>
<form action="DOM6b_SummitedForm.htm" onsubmit="return validate()">
Name (max 10 characters): <input type="text" id="fname" size="20"><br />
Age (from 1 to 100): <input type="text" id="age" size="20"><br />
E-mail: <input type="text" id="email" size="20"><br />
<br />
<input type="submit" value="Submit">
</form>
</body>

</html>
```

DOM6b_SummitedForm.htm

```
<html>
<head>
<title>Submit page</title>
</head>
<body>
<h1>Your input has been  submitted</h1>
</body> </html>
```

```
function validate()
{
var at=document.getElementById("email").value.indexOf("@");
var age=document.getElementById("age").value;
var fname=document.getElementById("fname").value;
submitOK="true";

if (fname.length>10)
 {
 alert("The name may have no more than 10 characters");
 submitOK="false";
 }
if (isNaN(age)||age<1||age>100)
 {
 alert("The age must be a number between 1 and 100");
 submitOK="false";
 }
if (at==-1)
 {
 alert("Not a valid e-mail!");
 submitOK="false";
 }
if (submitOK=="false")
 {
 return false;
 }
}
</script>
```

HTML DOM: Table Objects

- The Table object represents an HTML table. For each instance of a <table> tag in an HTML document, a Table object is created. **IE:** Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Table Object Collections

Collection	Description	IE	F	O	W3C
cells[]	Returns an array containing each cell in a table	5	1	1	No
rows[]	Returns an array containing each row in a table	4	1	9	Yes
tBodies[]	Returns an array containing each tbody in a table	4			Yes

HTML DOM: Table Objects

Table Object Properties

Property	Description	IE	F	O	W3C
border	Sets or returns the width of the table border	4	1	9	Yes
caption	Sets or returns the caption of a table	4	1	9	Yes
cellPadding	Sets or returns the amount of space between the cell border and cell content	4	1	9	Yes
cellSpacing	Sets or returns the amount of space between the cells in a table	4	1	9	Yes
frame	Sets or returns the outer-borders of a table	4	1	9	Yes
id	Sets or returns the id of a table	4	1	9	Yes
rules	Sets or returns the inner-borders of a table	4	1	9	Yes
summary	Sets or returns a description of a table	6	1	9	Yes
tFoot	Returns the TFoot object of a table	4	1	9	Yes
tHead	Returns the THead object of a table	4	1	9	Yes
width	Sets or returns the width of a table	4	1	9	Yes

HTML DOM: Table Objects

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Table Object Methods

Method	Description	IE	F	O	W3C
createCaption()	Creates a caption element for a table	4	1	9	Yes
createTFoot()	Creates an empty tFoot element in a table	4	1	9	Yes
createTHead()	Creates an empty tHead element in a table	4	1	9	Yes
deleteCaption()	Deletes the caption element and its content from a table	4	1	9	Yes
deleteRow()	Deletes a row from a table	4	1	9	Yes
deleteTFoot()	Deletes the tFoot element and its content from a table	4	1	9	Yes
deleteTHead()	Deletes the tHead element and its content from a table	4	1	9	Yes
insertRow()	Inserts a new row in a table	4	1	9	Yes

See Examples: DOM08, DOM09

DHTML: Example (All together)

```
<html>
<head>
<script type="text/javascript">
 function bgChange(bg) {   document.getElementById('x').style.background="url(" + bg + ")";    }
</script>
</head>
<body>
<p>This example demonstrates how to insert a background image to a textarea</p>
<p>Mouse over these images and the textarea will get a background image.</p>
<table width="300" height="100">
<tr>
 <td onmouseover="bgChange('images/paper.jpg')" background="images/paper.jpg"></td>
 <td onmouseover="bgChange('images/bgdesert.jpg')" background="images/bgdesert.jpg"></td>
</tr>
</table>

<form>
<textarea id="x" rows="5" cols="20"> This example can not be edited because
our editor uses a textarea for input, and your browser does not allow
a textarea inside a textarea. </textarea>                    See another example (animation)
</form>                                                                    image_as_a_link
                                                                           image_as_a_map
</body>                                                                 Go_WhereWithSelect
</html>
```


DHTML: Examples

- See another examples:
  - image_as_a_link
  - image_as_a_map
  - DHTML00_MovingImage
  - DHTML01_InsertImageInTextArea
  - DHTML02_ChangeTextColorInOptionList
  - DHTML03_GoWhereWithSelect
  - DHTML04_DigitalClock
  - DHTML05_MovingImage.html