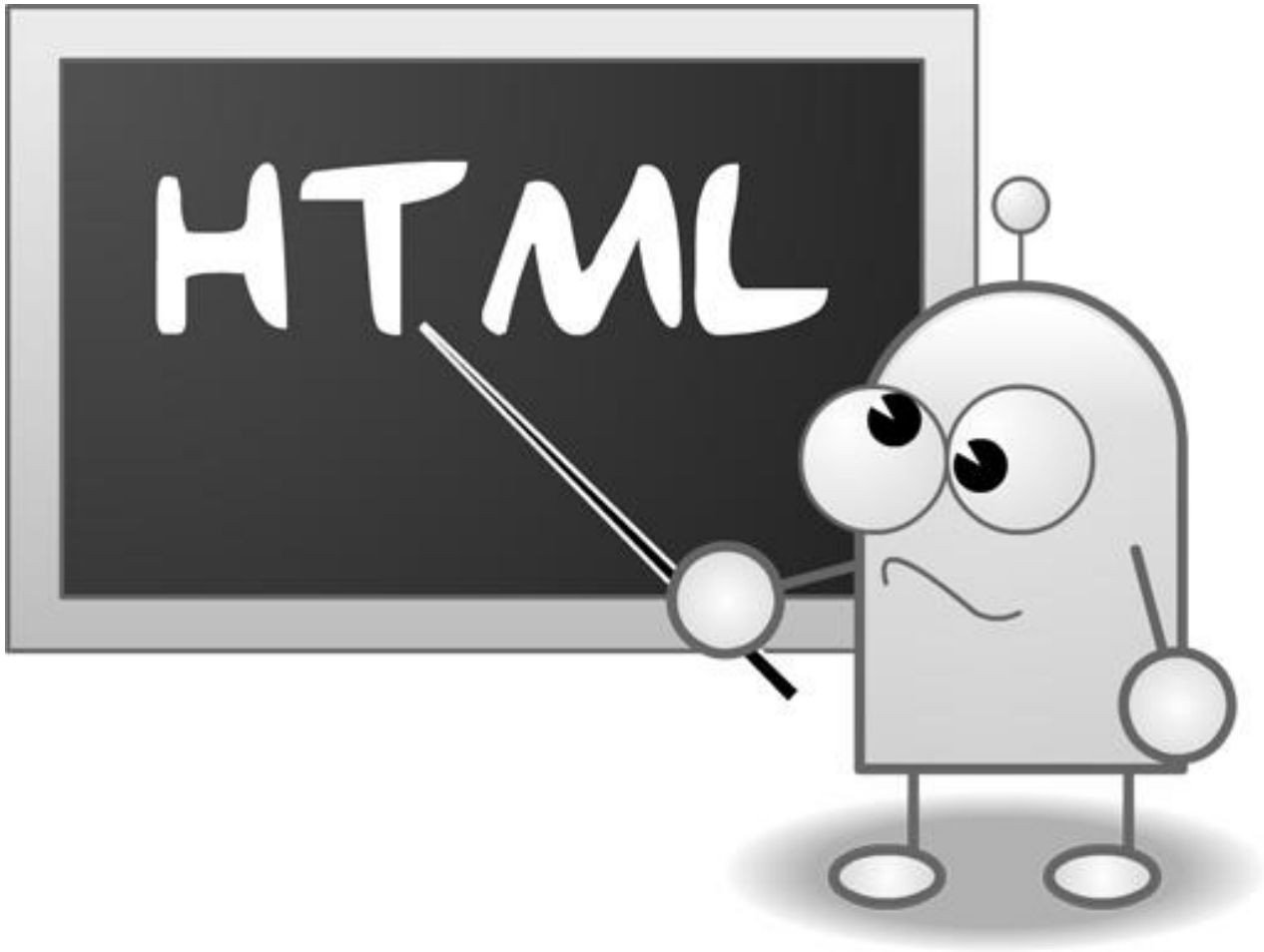
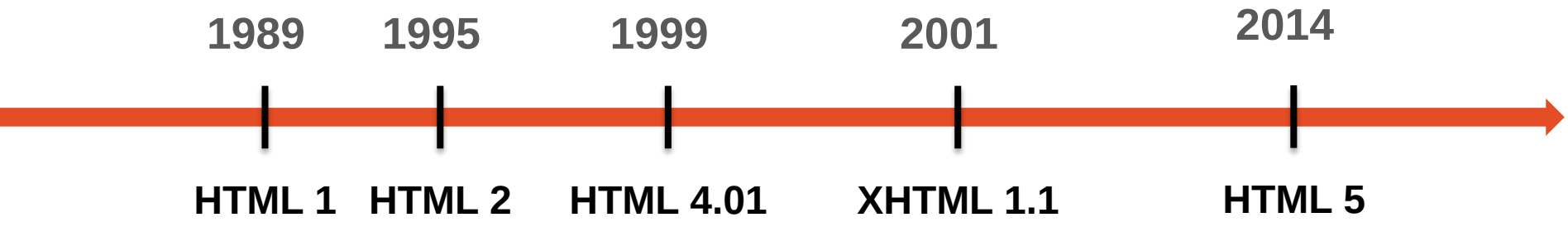

HTML

What is it?

HTML5 is a markup language used for structuring and presenting content on the World Wide Web.



What is HTML

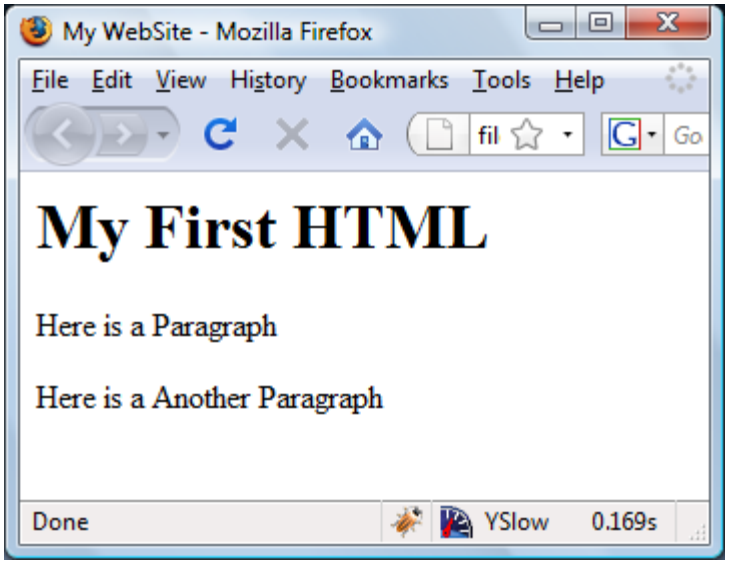
HTML is a language for describing web pages.

HTML stands for **H**yper **T**ext **M**arkup **L**anguage.

```
<html>
<head>
   <title>My WebSite</title>
</head>
<body>
   <h1> My First HTML </h1>
</body>
</html>
```

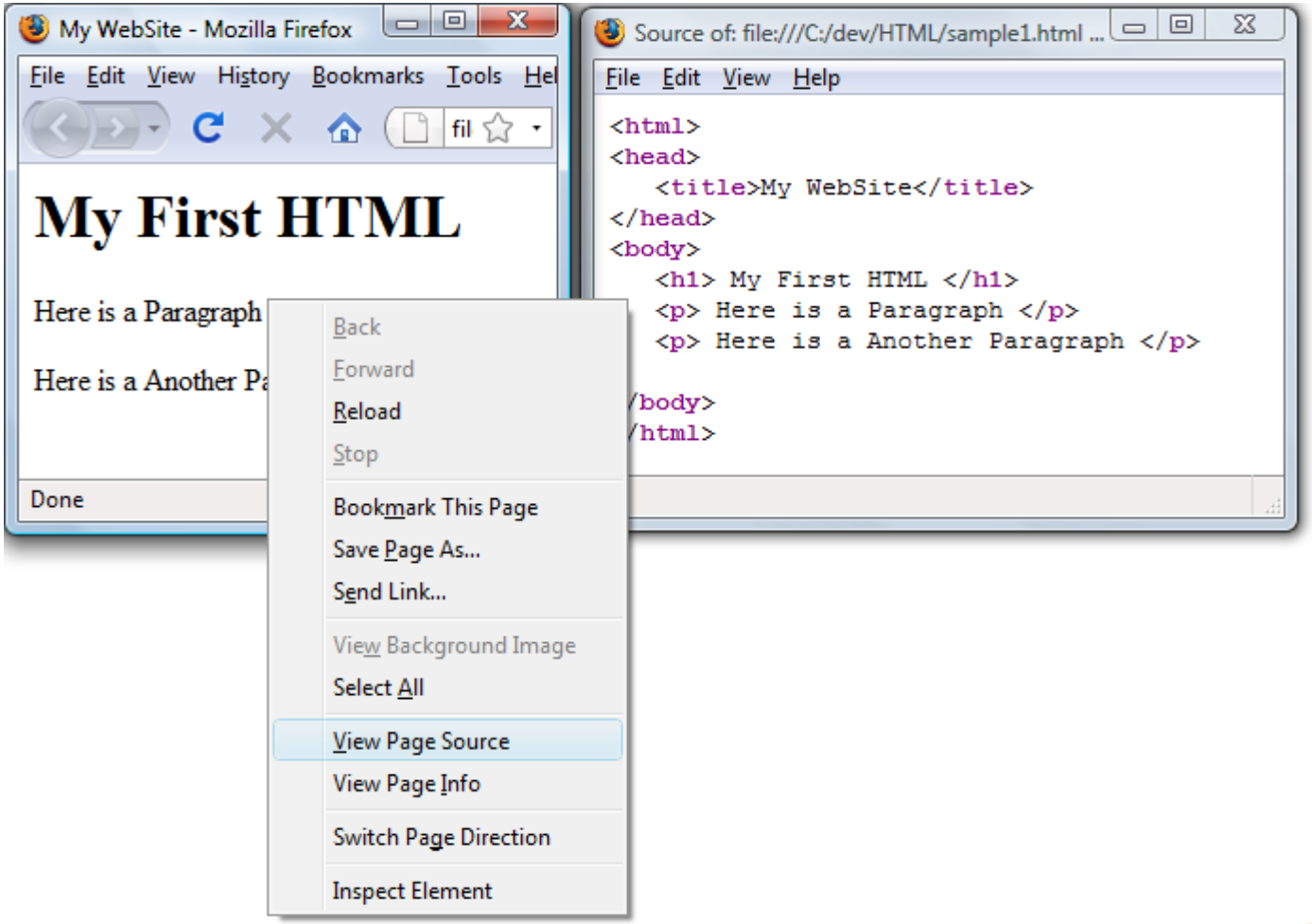
Basic Example

```
<html>
<head>
   <title>My WebSite</title>
</head>
<body>
   <h1> My First HTML </h1>
   <p> Here is a Paragraph </p>
   <p> Here is a Another Paragraph </p>

</body>
</html>
```



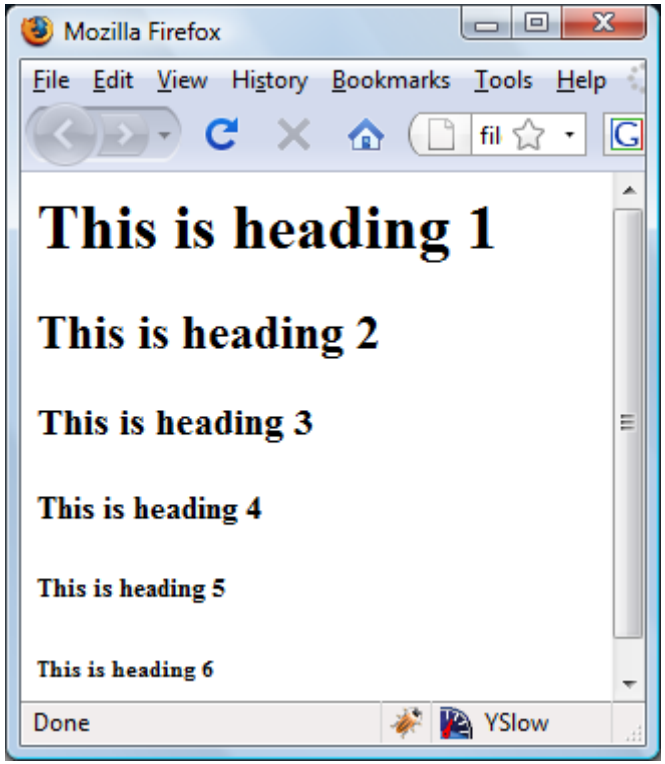
- <html> describes the web page.
- <body> is the visible page content.
- <h1> is displayed as a heading.
- <p> is displayed as a paragraph.
- <head> gives information about the page.
- <title> is the browser's window title.

HTML is the Browser's "Mother-Tongue"



Headings

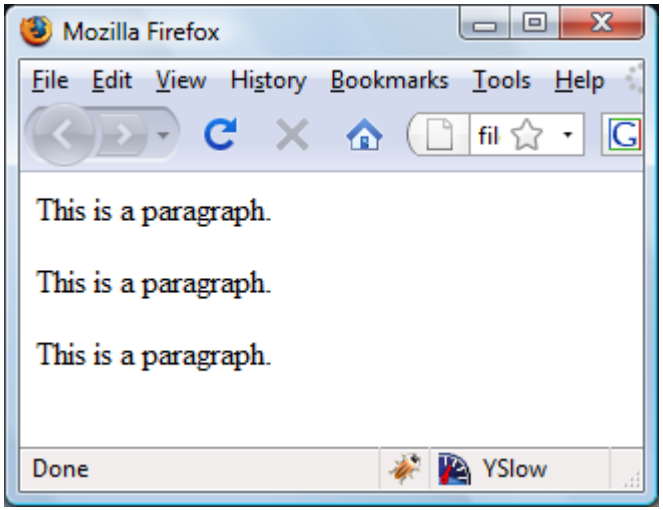
```
<html>
<body>
     <h1>This is heading 1</h1>
     <h2>This is heading 2</h2>
     <h3>This is heading 3</h3>
     <h4>This is heading 4</h4>
     <h5>This is heading 5</h5>
     <h6>This is heading 6</h6>
</body>
</html>
```



- These are the headers provided in HTML.
- All those <h*> elements are block elements
  - So a line-break is added
- Browsers also add some margin at the top and bottom

Paragraphs

```
<html>
<body>

<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>

</body>
</html>
```



- Paragraphs are also block elements
  - So a line-break is added
- Browsers also add some margin at the top and bottom

Images

- Note that *<img>* is an **empty element** – no closing tag is needed.
- The source for the image is provided as an attribute

```
<html>
<body>

    <img src="lala.jpg"   />

</body>
</html>
```

Links

```
<html>
<body>

Go here:
<a href="http://baba.com">Baba Buba</a>

</body>
</html>
```

- Links allow us to navigate in the web.
- Note that the link address is provided as an attribute

More Useful Elements

- **
** - (break-row) used to break lines.
- Note that simple line breaking are translated to spaces.
- **<hr />** - (horizontal rule) used to draw a line.
- You can configure the width of the rule:

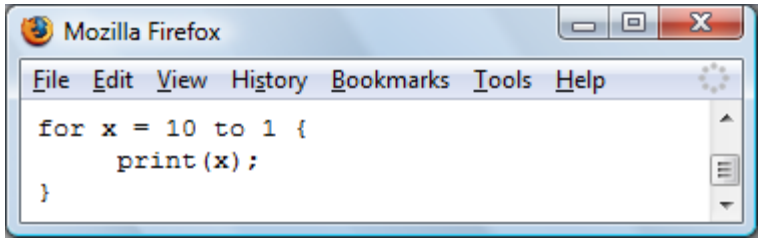
```
<hr  width="150"  />
```

- You can improve readability using comments (comments are ignored by the browser):

```
<!-- This is the lists of states,
it is updated when you select a country -->
```

Preformatted Text

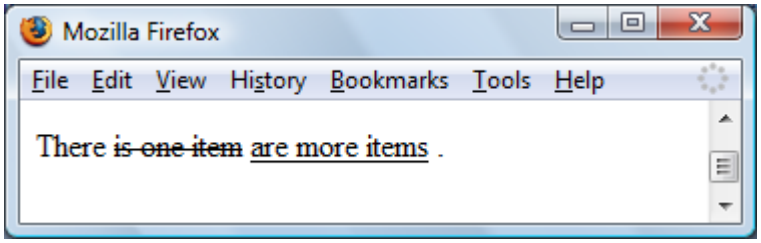
```
<pre>
for x = 10 to 1 {
     print(x);
}
</pre>
```



- Normally, new lines and multiple white spaces are ignored.
- Preformatted text is displayed with respect to white spaces and new lines.

Deleted and Inserted Text

```
<p>
There
<del>is one item</del>
<ins>are more items</ins>
.
</p>
```



- Better done with CSS

Lists

- Unordered List:

```
<h5>Great Carrot Drink</h5>
<ul>
   <li>Carrots - 5 pieces</li>
   <li>Honey - 1 spoon</li>
   <li>Ice - one glass</li>
</ul>
```

Great Carrot Drink

- Carrots - 5 pieces
- Honey - 1 spoon
- Ice - one glass

- Ordered List:

```
<h5>Follow the following steps:</h5>
<ol>
   <li>Go straight to the End</li>
   <li>Turn left</li>
   <li>Leave the case by the tree</li>
   <li>Start running</li>
</ol>
```

Follow the following steps:

1. Go straight to the End
2. Turn left
3. Leave the case by the tree
4. Start running

Nested Lists

- It is possible to create nested lists:

```
<ul>
        <li>Japan</li>
        <li>Israel:
            <ul>
                <li>Tel Aviv:
                    <ul>
                        <li>Florentin</li>
                        <li>HaTikva</li>
                    </ul>
                </li>
                <li>Jerusalem</li>
            </ul>
        </li>
        <li>China</li>
    </ul>
```

- Japan
- Israel:
  - Tel Aviv:
    - Florentin
    - HaTikva
  - Jerusalem
- China

More about Anchors

- To open a link in a new tab, use:

```
<a href="http://www.gmail.com/" target="_blank">
   Open Gmail (opens a popup window)
</a>
```

- The URL can point to any resource available on the web, e.g., picture, movie, etc:

Named Anchors

- Named Anchors are used for linking to a different area in the same page (note the differences):

```
<a name="pageTop"><h1>Samples</h1></a>
...
...
<a href="#pageTop">Goto Top</a>
```

  - Note that the *pageTop* anchor is not displayed differently.

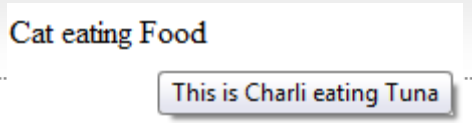
- Linking to a specific section in another page:

```
<a href="http://www.MyDomain.com/MyPage.html#someSection">
Goto Page at Specific Section</a>
```

More About Images

- The <img> tag is an empty element.
- Use the <alt> attribute to improve readability of your page in text-only browsers, or when a browser fails to load an image:

```
<img src="catEats.gif" alt="Cat eating Food"
     title="This is Charli eating Tuna"/>
```



- You can modify the size of an image using the width and the height attribute.
- Note that the image is downloaded from the server in its real size.

```
<img src="cat.jpg" width="48" height="48">
```

Floating

- Elements in HTML are usually rendered In-Line:
- Every element takes a position by the order it appears.
- It is also possible to make the elements float right or left.


Note that the Text
starts where the image ends

```
<img src="pic1.jpg" />
Note that the Text <br/>
starts where the image ends
```

Note that the Text
appears at the same level of the image

```
<img src="pic1.jpg"
style="float:right" />
Note that the Text <br/>
appears at the same level of the
image
```

Tables

- Tables are very useful elements. Here is a simple example:

```
<table border="1">
<tr>
   <td>row: 1, column: 1</td>
   <td>row: 1, column: 2</td>
</tr>
<tr>
   <td>row: 2, column: 1</td>
   <td>row: 2, column: 2</td>
</tr>
</table>
```

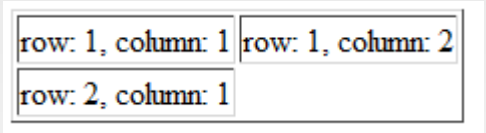
row: 1, column: 1	row: 1, column: 2
row: 2, column: 1	row: 2, column: 2

- <*tr*> represents a table-row
- <*td*> represents a cell: table-data

Empty Cells

- When a cell is empty, on some of the browsers (e.g., FF), the table will look like this:

```
<table border="1">
<tr>
   <td>row: 1, column: 1</td>
   <td>row: 1, column: 2</td>
</tr>
<tr>
   <td>row: 2, column: 1</td>
   <td></td>
</tr>
</table>
```



- Note that the inner line is not drawn. This can be fixed by placing a white space in the *td*:

```
<td> </td>
```

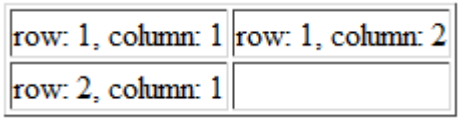


Table Headers

- Tables may have Headers:

```
<table border="1">
<tr>
   <th>header1</th>
   <th>header2</th>
</tr>
<tr>
   <td>row: 1, column: 1</td>
   <td>row: 1, column: 2</td>
</tr>
<tr>
   <td>row: 2, column: 1</td>
   <td>row: 2, column: 2</td>
</tr>
</table>
```

header1	header2
row: 1, column: 1	row: 1, column: 2
row: 2, column: 1	row: 2, column: 2

<*th*> is used for headers.

No-Border Tables

Sometimes, we don't want the table to show borders.
Leave the default or explicitly set the border to 0

```
<table border="0" >
<tr>
  <td><img src="pic1.jpg"></td>
  <td>Waiting for the Daly Lama</td>
</tr>
<tr>
  <td><img src="pic2.jpg"></td>
  <td>Sunset in the Arabian sea</td>
</tr>
</table>
```

Tables

Cells may span more that one row / column, here is how:

```
<table border="1">
<tr>
  <th>Name</th>
  <th colspan="2">Phones</th>
</tr>
<tr>
  <td>Muki D.</td>
  <td>763-8796-980</td>
  <td>763-3746-731</td>
</tr>
</table>
```

Name	Phones	
Muki D.	763-8796-980	763-3746-731

Tables

Cells may span more that one row / column, here is how:

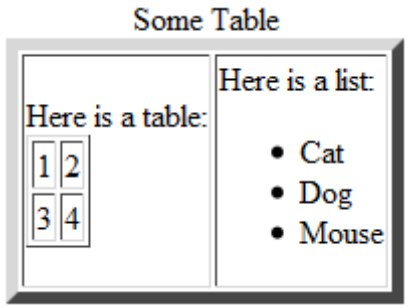
```
<table border="1">
<tr>
  <th>Name:</th>
  <td>Puki G.</td>
</tr>
<tr>
  <th rowspan="2">Phones:</th>
  <td>763-8796-980</td>
</tr>
<tr>
  <td>763-3746-731</td>
</tr>
</table>
```

Name:	Puki G.
Phones:	763-8796-980
	763-3746-731

Nested Tables

Tables may be nested:

```
<table border="6">
<caption>Some Table</caption>
<tr>
  <td>
  Here is a table:
   <table border="1">
   <tr>
     <td>1</td>
     <td>2</td>
   </tr>
   <tr>
     <td>3</td>
     <td>4</td>
   </tr>
    </table>
  </td>
```

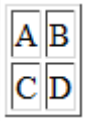



```
<td>Here is a list:
   <ul>
    <li>Cat</li>
    <li>Dog</li>
    <li>Mouse</li>
   </ul>
  </td>
</tr>
</table>
```

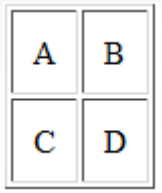
Tables

You may use cellpadding to add padding to the cell

```
<table border="1">
<tr>
  <td>A</td>
  <td>B</td>
</tr>
<tr>
  <td>C</td>
  <td>D</td>
</tr>
</table>
```




```
<table border="1" cellpadding="10">
<tr>
  <td>A</td>
  <td>B</td>
</tr>
<tr>
  <td>C</td>
  <td>D</td>
</tr>
</table>
```



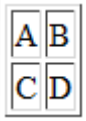
BUT, its done better with CSS:

```
td  {
  padding: 10px;
}
```

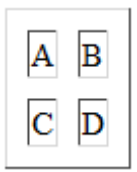
Tables

Use cellspacing to add space between cells.

```
<table border="1">
<tr>
  <td>A</td>
  <td>B</td>
</tr>
<tr>
  <td>C</td>
  <td>D</td>
</tr>
</table>
```




```
<table border="1" cellspacing="10">
<tr>
  <td>A</td>
  <td>B</td>
</tr>
<tr>
  <td>C</td>
  <td>D</td>
</tr>
</table>
```



BUT, its done better with CSS:

```
table {
  border-spacing: 10px;
  border-collapse: separate;
}
```

HTML Entities

- Some characters (like <) cannot be placed inside the text (the browser might mistake them for tags).

- HTML Entities are used to output these special characters. For example: *<* is the entity code for <.

- You can also use entity numbers, such as: ¥
  - Entity names are recommended as they are more readable.
  - However, some Entity names are not supported by all browsers.

HTML Entities

Here are some sample entities:

Entity Output	Description	Code	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;

Head Elements

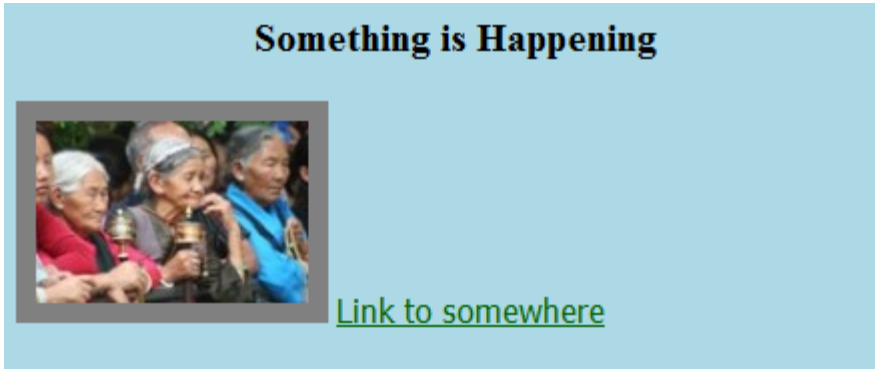
The Head element contains information about the HTML document:

```
<meta charset="utf-8" />
<meta name="title"       content="Frogi Pets Shop" />
<meta name="description" content="Only the Best for your Pet" />
<meta name="keywords"    content="Frogi, Pets Food, miyahu" />
```

Lets look a bit deeper on the <head> - here

Introduction to CSS

- The attribute style can be used on most of the HTML elements.
- This is where CSS – the Cascading Style Sheets standard - meets HTML.
- Using CSS we can describe the look of our web page, separated from its structure and in a standard way.
- Examine the following Formatting:



Introduction to CSS

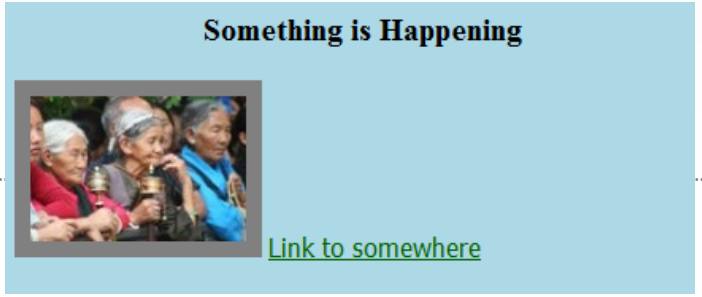
- CSS – Cascading Style Sheets
- We use it to *magically* control the look of our apps

- Be warned, getting hold of the CSS magic will take you further than you think.

Using CSS

- Let's have a look at the backing CSS:

```
<html>
<body style="background-color: lightblue;">

<p style="text-align: center; margin-left: 10px;
          font-size:20px; font-weight: bold;">
Something is Happening
</p>

<img src="pic1.jpg" style="border: 10px gray solid" />

<a href="http://google.com" style="color: darkgreen;
                                   font-family: tahoma;  ">
Link to somewhere
</a>

</body>
</html>
```



Using the Class Attribute

- Using the style attribute is the wrong way to design the look of your pages:
  - Mix of HTML & CSS is **harder to maintain**
  - We cannot **reuse** previous declaration (hard to keep a consistent look)

- This is where the class attribute fits:

```
<head>
<style>
   .niceLink{
      color: darkgreen;
      font-family: tahoma;
   }
</style>
</head>
```

```
<a href="http://google.com" class="niceLink">Link to somewhere</a>
```

Selectors

- We saw how to apply styling to elements using the class attribute, this helps when we need to apply common styling to multiple elements.
- There are more ways to reference elements, such as: by their IDs and by their tag-type:

```
<body>
    <img src="pic1.jpg" id="myImg”/>
</body>
```

```
<style>
    body {
        background-color: lightblue;
    }
    #myImg{
        border: 3px gray solid;
        font-family: tahoma;
    }

</style>
```

Selectors

This selector has the impressive name:
Descendant combinator but its quite simple really:

```
<a href="http://google.com" class="niceLink">
   <img src="pic1.jpg"  />
</a>
```

```
.niceLink img {
   border:10px ridge maroon;
}
```

Only images inside a .*niceLink* are affected.

Selectors

- Using selectors effectively is one of the keys for building professional level CSS
- Here is the List
- Here is a good place to practice

Pseudo Classes

CSS supports basic support for reacting to user interaction:

```
.niceLink{
    text-decoration:none;
    color: green;
}
.niceLink:hover{
    text-decoration:underline;
    color: red;
}
```

Link to somewhere

Link to somewhere

Using a CSS File

- Usually, we need to share the same styling across several pages, so our CSS declarations should reside in a separate file, a CSS File.
- Here is how we reference the CSS file:

```
<head>
    <link rel="stylesheet" href="myStyle.css" />
</head>
```

Deprecated HTML Styling

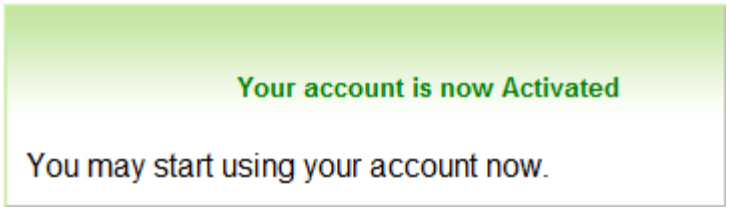
- With the introduction of CSS in HTML 4, some tags became deprecated:

  * , <font>, <u>*
- There are also attributes that are deprecated:

  align, bgcolor, color

The div Element

- The *div* element is commonly used to define a division (area) in the document:
- The div is a block element.



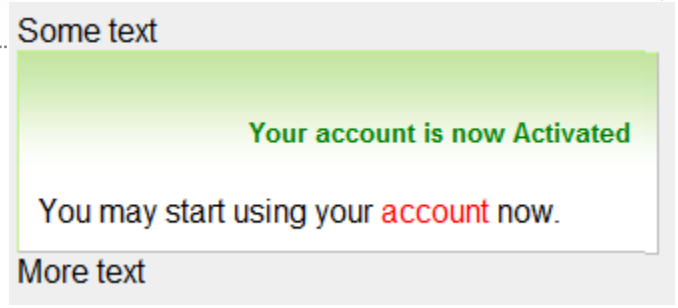

```
<div class="niceBox">
<h5>Your account is now
Activated</h5>
You may start using your
account now.
</div>
```

```
.niceBox {
    padding:10px;
    width:400px;
    border:1px #C0C0C0 solid ;
    border-top:1px #BEF488 solid;
    border-left:1px #BEF488 solid;
    background-color:white;
    background-image:url(greenFadeBG.jpg);
    background-repeat:repeat-x;
}
.niceBox h5{
    text-align:center;
    color:green;
{
```

The *span* Element

The *span* element is used to define an inline span in the document:

```
Some text
<div class="niceBox">
    <h5>Your account is now Activated</h5>
     You may start using your
     <span style="color:red">account</span> now.
</div>
More text
```



Victorious!

You have successfully grasped the basics of

HTML & CSS

Now lets dance

Javascript

In The Browser

Calling Functions from DOM

- Javascript code is usually placed in functions.
  - The code in the function will be executed only when the function is called.
- Take a look at the following code:

```
<button onclick="confirmAndDo()" >Do It!<button>

<script>
function createUserQuestion(quest) {
    return "Dear Mr. Bitof, " + quest + "?";
}
function confirmAndDo() {
    var q = createUserQuestion("Are you sure");
    if (confirm(q)) {
        // Do it!
    }
}
</script>
```

window

- Represents the browser window, used for:
- Getting access to the URL (Location), the previous browsed pages (History), etc.
- Setting timeouts and intervals.
- Opening popup windows.
- **window is the default object, it can be used without specifying its name.**

```
window.setTimeout('alert("aha!")', 3000);

// same as:
setTimeout('alert("aha!")', 3000);
```

window.document

The document object provide access to the current document.

```
document.getElementById('myBox');
document.querySelector('#myBox');
```

Its recommended to stick to querySelector

Opening a window

You can open a window in JS

- Note that popup blockers tends to block such popups, specially when the window is not opened as a result of user click

```
var popup =
window.open('','','width=100,height=80')
popup.document.write("a Popup")
popup.focus()
```

window.navigator

```
appCodeName: "Mozilla"
appName: "Netscape"
appVersion: "5.0 (Windows)"
battery: BatteryManager
buildID: "20160210153822"
cookieEnabled: true
doNotTrack: "unspecified"
geolocation: Geolocation
language: "en-US"
languages: Array[2]
mediaDevices: MediaDevices
mimeTypes: MimeTypeArray
mozApps: DOMApplicationsRegistry
mozContacts: ContactManager
mozPay: null
onLine: true
oscpu: "Windows NT 6.1; WOW64"
platform: "Win32"
plugins: PluginArray
product: "Gecko"
productSub: "20100101"
serviceWorker: ServiceWorkerContainer
userAgent: "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0"
```

Holds Information about the browsers and environment:

For example:

userAgent – which browser is used

OnLine – is there network connectivity

Geolocation – get the current user location

window.history

Using the history object it is possible to simulate a click on the next/previous buttons.

```
<input type="button" value="Go Back"
onclick="window.history.go(-1)“ />
```

window.location

- The Location object holds information and methods regarding the current URL:

```
▼ Location {replace: function, assign: function, ancestorOrigins:
  ▶ ancestorOrigins: DOMStringList
  ▶ assign: function () { [native code] }
    hash: "#/"
    host: "localhost"
    hostname: "localhost"
    href: "http://localhost/academy/app/index.html#/"
    origin: "http://localhost"
    pathname: "/academy/app/index.html"
    port: ""
    protocol: "http:"
  ▶ reload: function reload() { [native code] }
  ▶ replace: function () { [native code] }
    search: ""
  ▶ toString: function toString() { [native code] }
  ▶ valueOf: function valueOf() { [native code] }
  ▶ __proto__: Location
```

Handling Events

- Javascript is usually used to react to events.
- Here are some examples:
  - The Page finished loading.
  - Mouse click.
  - The user selected something (e.g. option in a list box).
  - Keystroke in a textbox.
  - Mouse hovering over an element.
  - Dragging, pinching…

Handling Events

- Have a look at the following HTML code:

```
<body onload="initSelf()">
    Your name: <input type="text"  id="userName" onkeyup="echoInput()" />
    <input type="button" value="Do It!" onclick="confirmAndDo()" />
    <div id="echoArea" style="color:green" ></div>
</body>
```

- It is safe to refer to elements only after the document has loaded, so we used the **onload** event to focus on an element.

```
function initSelf() {
    var elUserName = document.querySelector("#userName");
    elUserName.focus();
}

function echoInput() {
    var elUserName = document.querySelector("#userName");
    var elEchoArea = document.querySelector("#echoArea");

    elEchoArea.innerHTML = elUserName.value;
}
```

Some More Events

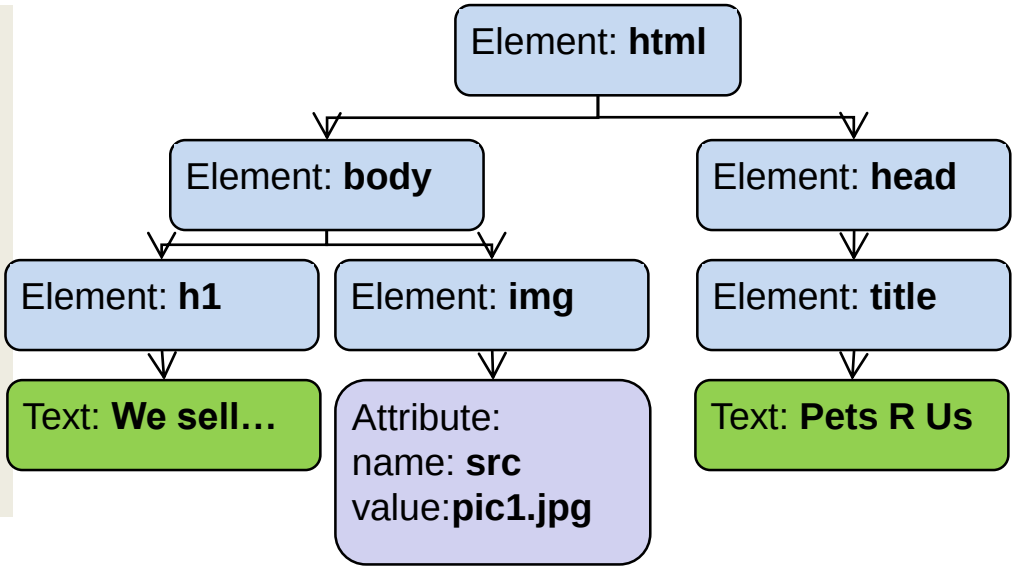
Event	Occurs	Sample usage
onFocus	The field got focus	Show relevant help tooltip
onBlur	The field lost focus	Form element validation
onChange	The field was change	Form element validation
onSubmit	The submit button was clicked	Entire form validation
onMouseOver	The mouse entered the area of the element	Try using CSS :hover when possible
onMouseOut	The mouse has left the area of the element	Try using CSS when possible

And many more…

DOM – document object model

- The DOM is a tree representation of our HTML document.
- It is composed from nodes
- See the following example:

```
<html>
  <head>
    <title>Pets R Us</title>
  </head>
  <body>
    <h1>We sell Pets</h1>
    <img src="pic1.jpg"/>
</body>
</html>
```



DOM – node's properties

- Every node in the DOM tree supports the following attributes:
  - e.parentNode - the parent node of e.
  - e.childNodes - the child nodes of e.
  - e.attributes - the attributes nodes of e.
  - e.innerHTML - the inner text value of e.
  - e.nodeName – read-only, the name of e.
    - For element – the tag name.
    - For attribute – the attribute name.
    - For text - #text.
  - e.nodeValue - the value of e.
    - For element – undefined.
    - For attribute – the attribute value.
    - For text – the text itself.
  - e.nodeType
    - The most useful types: 1 – element, 2 – attribute, 3 – text.

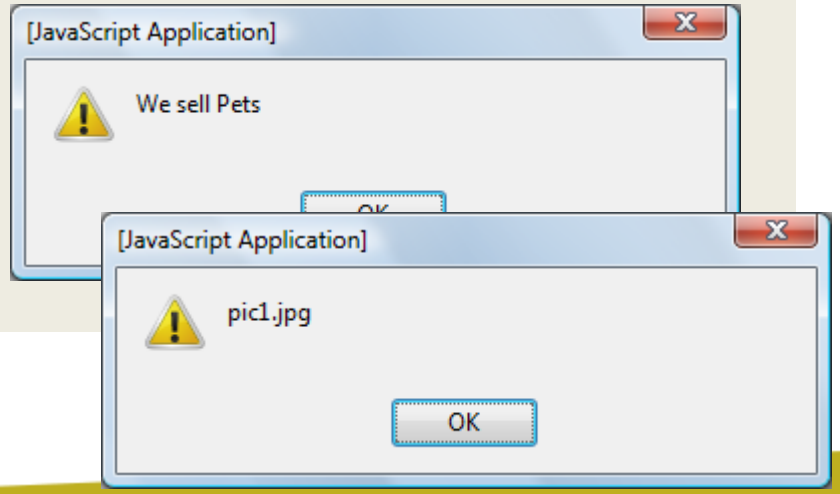
DOM – node's methods

- Every node in the DOM tree supports the following methods:
  - e.querySelector(selector)
  - e.getElementById(id) - get the element with a specified id under e.
  - e.getElementsByTagName(name) – get all elements of a specified tag under e.
  - e.appendChild(node) – adds a child node.
  - e.removeChild(node) – removes a child node.

Navigating in the DOM tree

Note: We will not write code like that, this is just to help visualize the DOM

```
<html>
<head>
<title>Pets R Us</title>
<script type="text/javascript">
function initSelf() {
    alert(document.documentElement.childNodes[1].childNodes[1]
        .firstChild.nodeValue);
    alert(document.documentElement.childNodes[1].childNodes[3]
        .attributes[0].nodeValue);
}
</script>
</head>
<body onload="initSelf()">
    <h1>We sell Pets</h1>
    <img src="pic1.jpg"/>
</body>
</html>
```



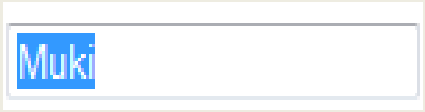
HTML DOM Objects

- The HTML DOM defines a standard set of objects for HTML, and a way to access and manipulate HTML elements.
- Here is a partial list of the objects:
  - Document - used to access all elements in a page
  - Anchor - Represents an <a> element
  - Image Represents an <img> element
  - Table Represents a <table> element
  - TableData Represents a <td> element
  - TableRow Represents a <tr> element
  - Form Represents a <form> element
  - Textarea Represents a <textarea> element
  - Etc.

Example: the Text object

The Text object represents a textbox (an <input> element):

```
var txt = document.getElementById("txt");
txt.value = "Muki";
txt.maxLength = 10;
txt.readOnly = true;
txt.focus();
txt.select();


<input id="txt"  />
```



Example: the Image object

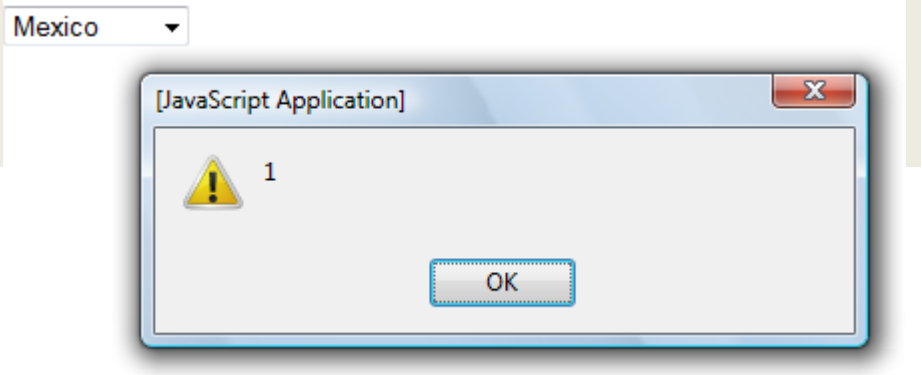
The Image object represents, well, an image:

```
setInterval(switchImage, 2000);

var IMG1 = "pic1.jpg";
var IMG2 = "pic2.jpg";
var gSelectedImg = IMG1;
function switchImage() {
   if (gSelectedImg == IMG1) gSelectedImg = IMG2;
   else gSelectedImg = IMG1;
   document.querySelector("#myImg").src = gSelectedImg;
{


<img id="myImg" src="pic1.jpg"  />
```

Example: the Select object

The Select object represents a dropdown:

```
function echoCountry() {
    var elCountryList = document.getElementById("country");
    alert(elCountryList.selectedIndex);
}


<select id="country" onchange="echoCountry()">
    <option>Guatemala</option>
    <option>Mexico</option>
    <option>Beliz</option>
</select>
```



innerHTML

Using the innerHTML attribute you may alter the content of an element.

- This means that you create new elements on the fly.

```
function changeLink(){
   document.getElementById('myLink').innerHTML="<b>Go Now</b>";
   document.getElementById('myLink').href="http://misterBIT.co.il";
}


<a id="myLink" href="http://g.com" onmouseover="changeLink()"">Go Search</a>
```

Controlling DOM events

Two most important methods of the event are:

- event.preventDefault() – prevents the browser from doing its default behavior – here is an example
- event.stopPropagation() - Prevents further propagation of the current event in the capturing and bubbling phases.
  Here is an example.

Data- attributes

Sometimes its useful to keep some data on the DOM elements, this is done by setting "data-" attributes:

```
function foo(el){
    window.location = el.getAttribute('data-where');
}


<a href="#" data-where="other.html"
onmouseover="foo(this)">Go There</a>
```

Its all in the DOM

- We saw several examples for objects representing elements in the page.
- Remember that everything is in the DOM, so its very easy to access and manipulate any section of your page.

Summary

- Javascript is the one and only scripting language for the web.
- Use Javascript to:
  - Code HTML events,
  - Create a dynamic and responsive GUI,
  - Dynamically manipulate your HTML elements.
- HTML documents are available as DOM, defining a standard way to access and modify elements.

HTML Input Elements

- Input Elements are used to get data from the user:

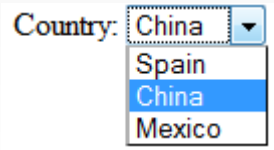
```
Full Name:<input type="text" name="fname" size="20"/>
```

Full Name:

```
Password:<input type="password" name="pass" size="20"/>
```

Password: •••••

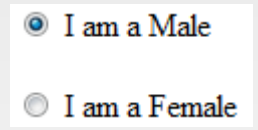
```
Country:
<select name="country" >
   <option id="1">Spain</option>
   <option id="2" selected="selected">China</option>
   <option id="3">Mexico</option>
</select>
```



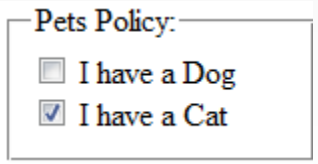
HTML Input Elements

More input Elements:

```
<input type="radio" id="gender1" name="gender" value="m" checked="checked" />
<label for="gender1">I am a Male</label>
<input type="radio" id="gender2" name="gender" value="f" />
<label for="gender2">I am a Female</label>
```

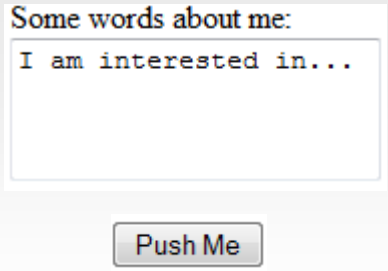



```
<fieldset>
   <legend>Pets Policy:</legend>

   <input type="checkbox" name="dog" id="dog" />
   <label for="dog">I have a Dog</label><br/>

   <input type="checkbox" name="cat" id="cat" checked="checked"/>
   <label for="cat">I have a Cat</label>
</fieldset>
```



HTML Input Elements

```
Some words about me:<br/>
<textarea name="desc"  rows="3" cols="20">
   I am interested in...
</textarea>

<input type="button" value="Push Me" onclick="alert('Hello')" />
```



HTML Forms

- Form is a logical element, uniting input elements:

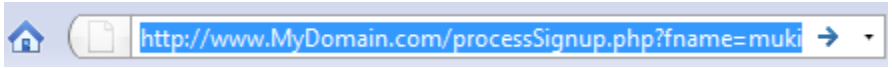
```
<form action="processSignup.php" method="get">
   <input type="text" name="fname" size="20"/>
   …
   <input type="submit" value="Send to Server" />

</form>
```

- The *action* element specifies a URL for a server-side file to process the data sent from the browser.

HTML Forms

```
<form action="processSignup.php" method="get">
   <input type="text" name="fname" size="20"/>
   …
   <input type="submit" value="Send to Server" />

</form>
```

- The *method* can be:
- **get** – data is sent as part of the URL and visible to the user, limited by size (1024 characters).



- **post** – data is sent hidden as part of the http request, this is the recommended way.

HTML Forms

- Let's put it all together:

Full Name: Ronaldo

Password: ••••

Country: China

I am a Male
I am a Female

Pets Policy:
I have a Dog
I have a Cat

Some words about me: I am interested in...

Send to Server

http://www.mydomain.com/processSignup.php?
fname=Ronaldo&
Pass=poli&
country=Mexico&
gender=m&
cat=on&
desc=I+am+interested+in+musica

Server side code will process the parameters, save to DB and redirect to the next page.

Form Hidden Elements

- Hidden input elements are not shown to the user.
- These are very useful when sending information from the server to the browser and back. Examine the following form:

```
<form action="processSignup.php" method="get">
   <input type="hidden" name="timestampFromServer" value="112341433"/>
   …
   <input type="submit" value="Send to Server" />

</form>
```

- When the form gets submitted, the server-side code may check the *timestampFromServer* parameter to figure out how much time it took the user to fill the form.

Victorious!

You have successfully grasped the basics of

HTML & CSS & JS in the Browser!

Now lets build something great