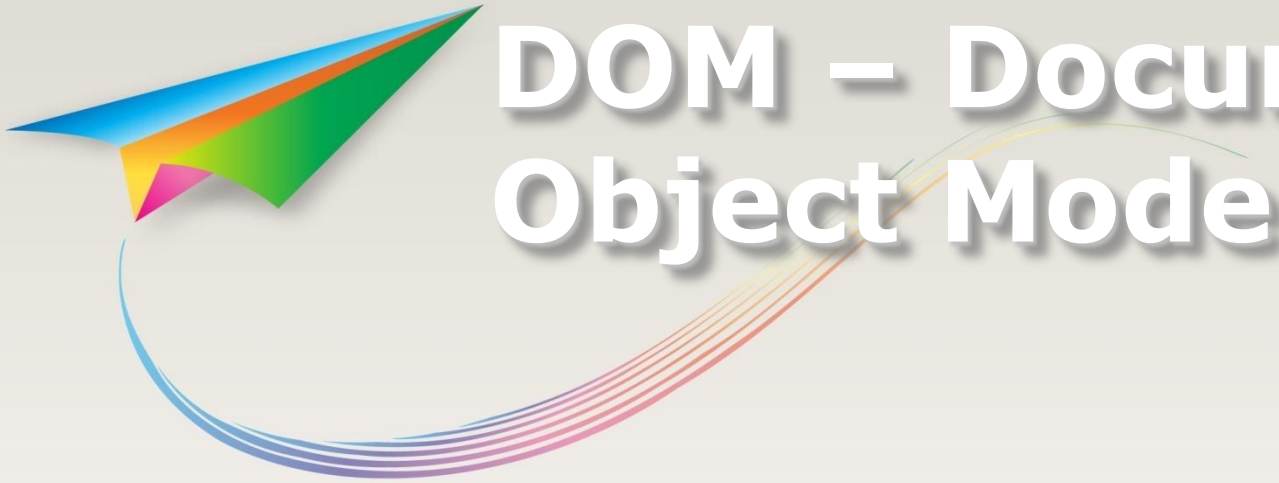


DOM – Document Object Model

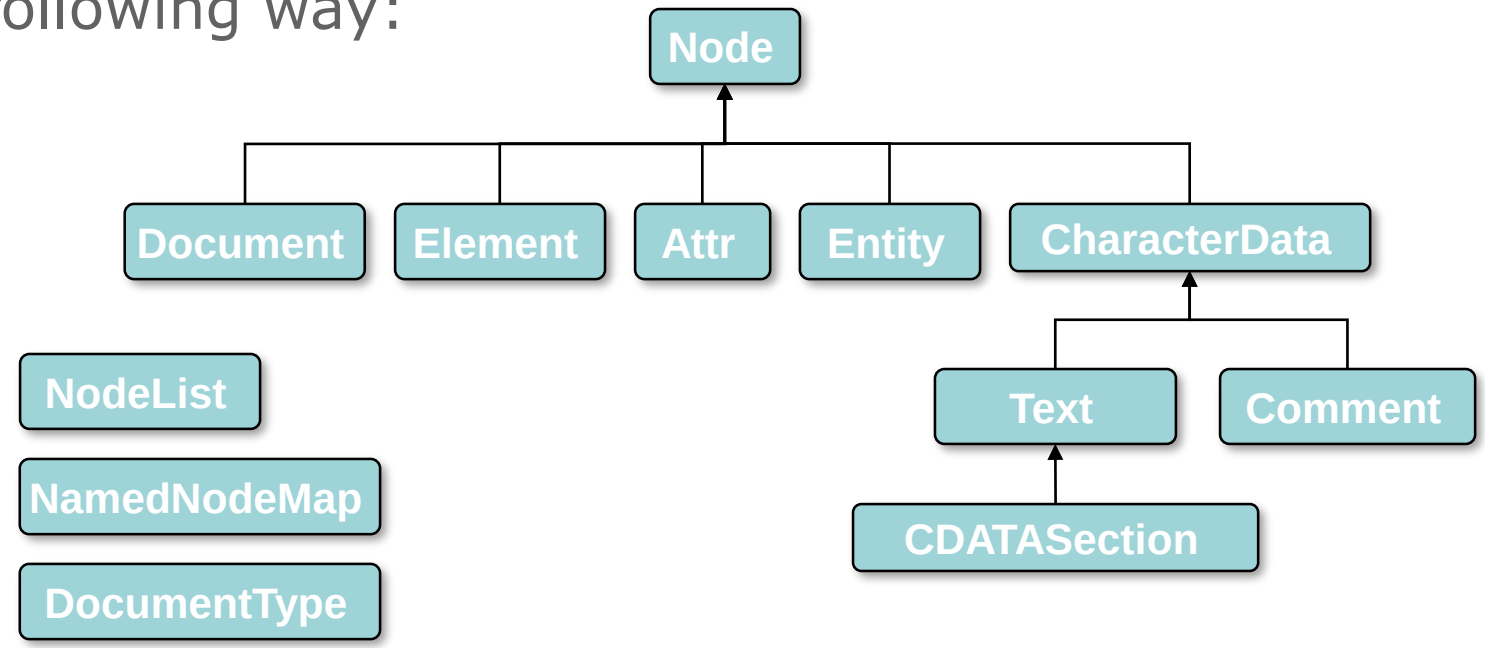
trainologic

DOM

- Document Object Model
- Handling Events
- Built-in classes and objects

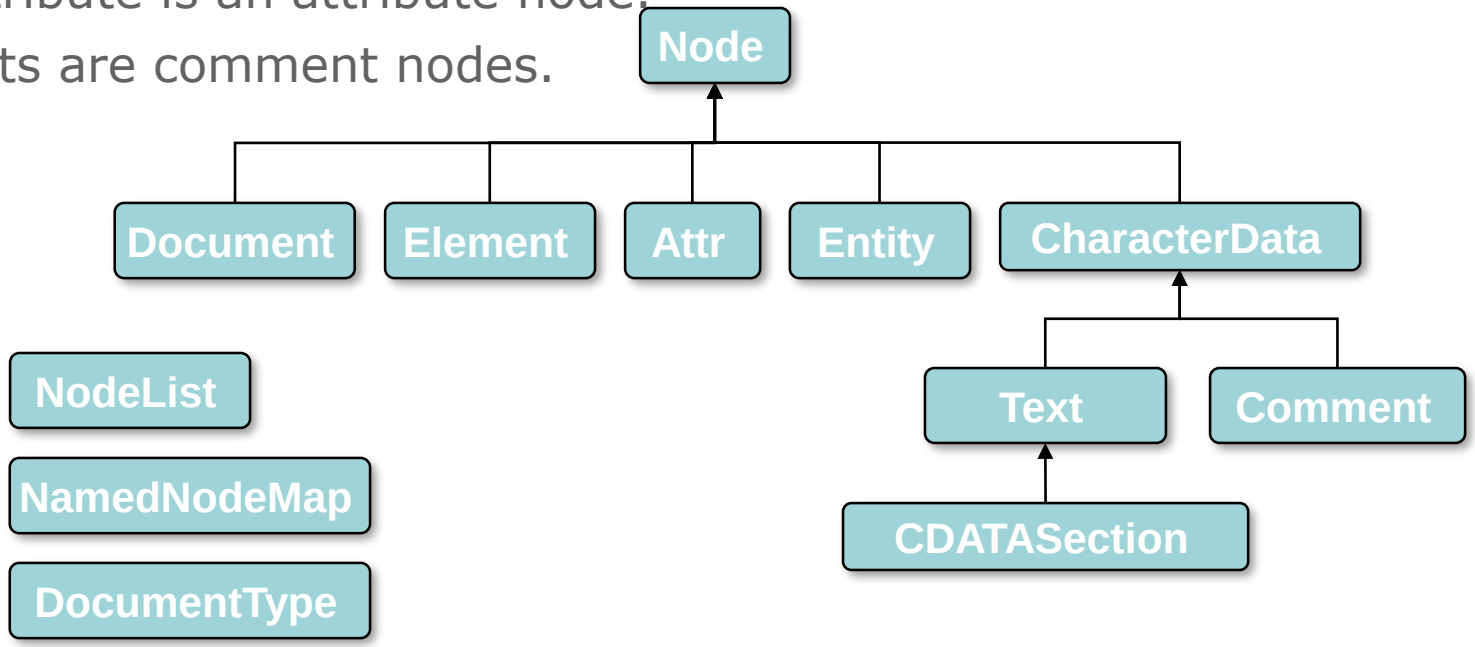
DOM

- DOM is a standard (by W3C) for accessing documents such as HTML and XML.
- It defines the different objects in the document in the following way:



DOM

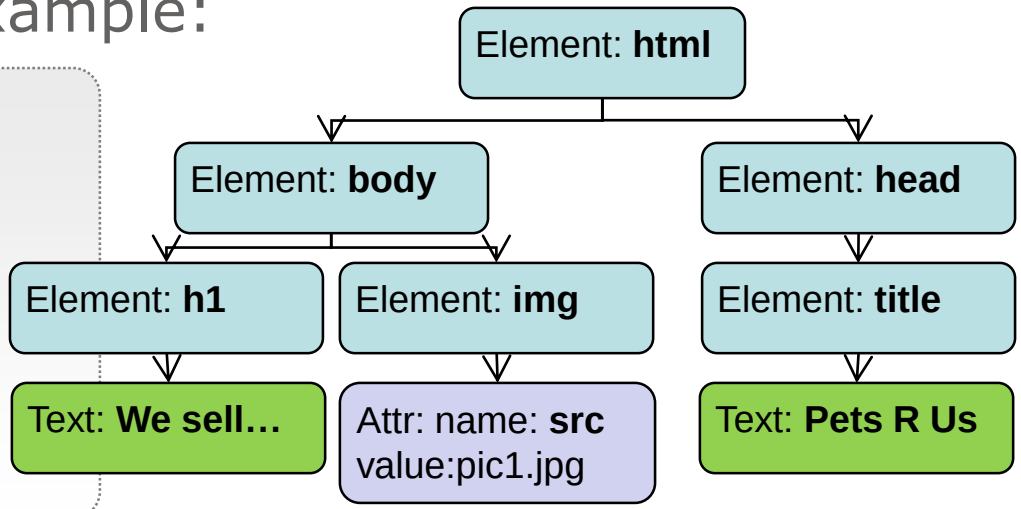
- The entire document is a document node .
- HTML tags are element nodes.
- The text in the element is a text node.
- Every attribute is an attribute node.
- Comments are comment nodes.



Example

- The DOM is a tree representation of our HTML document.
- 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.

DOM Node's Properties

- More attributes:
- 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.

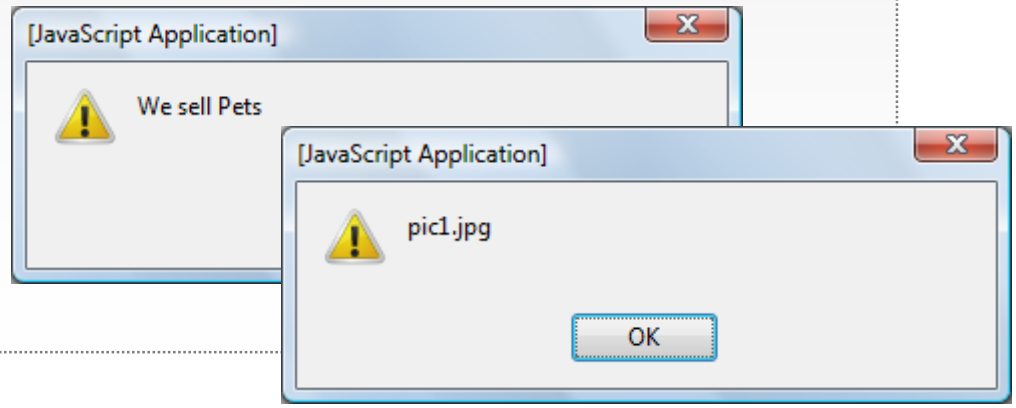
DOM Node's Methods

- Every node in the DOM tree supports the following methods:
- e.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.

Moving in the Tree

- Accessing elements by the tree structure:
  - Note that the "\n"s are represented in the tree.

```
<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

- 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

HTML DOM

- More objects:
  - Form Represents a <form> element
  - Input text Represents a textbox
  - Input button Represents a button
  - Textarea Represents a <textarea> element
  - Frame Represents a <frame> element

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://www.w3schools.com";
{


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

Go Search **Go Now**

The Document

- Represents the entire document, here are some useful methods and attributes:

```
var txt = document.getElementById("myTxt");
txt.innerHTML = "<b>Halo</b>"


var divs = document.getElementsByTagName("div");
var images = document.images;
var forms = document.forms;
alert(divs.length);
```

Switching the Element's Class

- One useful technique to create a responsive GUI is to handle the mouse events:

```
<style>
.nav {
    border: 1px gray solid;
    width: 100px;
{
.navHover {
    border: 2px green solid;
    cursor: pointer;
    width: 100px;
{

</style>
```

```
function switchMyClass(e) {
    if (e.className == "navHover")
     e.className = "nav";
    else e.className = "navHover";
{
```

```
<div class="nav" onmouseover="switchMyClass(this)"
                 onmouseout="switchMyClass(this)">
Go Fetch
</div>
```

Hiding/Showing Elements

- A common situation is when you need to hide and show a section of your document

```
function showStuff(id) {
   document.getElementById(id).style.display = 'block';
}

function hideStuff(id) {
   document.getElementById(id).style.display = 'none';
}
```

The Text Object

- Represents a textbox:

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


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

The Image Object

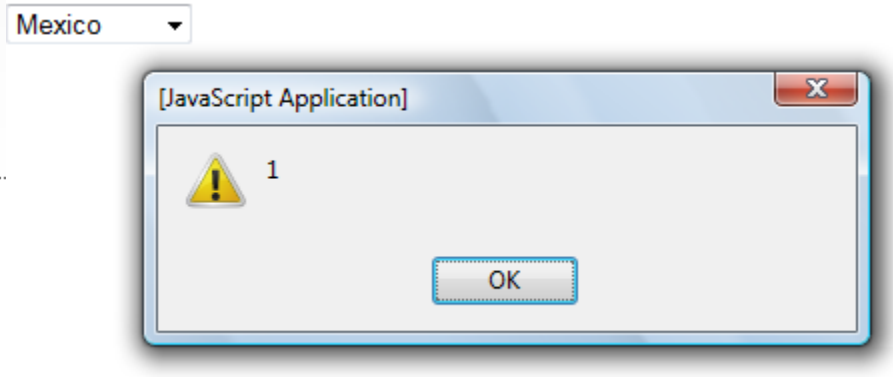
- Represents an Image:

```
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.getElementById("myImg").src = gSelectedImg;
{


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

The Select Object

- Represents a combo box.

```
function echoCountry() {
   countries = document.getElementById("country");
   alert(countries.selectedIndex);
{


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



Everything is 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.

DOM

- Document Object Model
- Handling Events
- Built-in classes and objects

Events

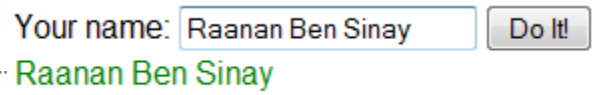
- Javascript is usually used to react to events.
- Here are some examples:
  - The Page finished loading.
  - Mouse click.
  - The user selected something (e.g. option in a list box).
  - Keystroke in a textbox.
  - Mouse hovering over an element.
- Have a look at the following HTML code:

```
<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>
```

Events

- The following JS functions handle the events.
  - It is safe to refer to elements only after the document has loaded, so we used the *onload* event to focus on an element.
  - We see here the document object, we will examine it more closely later:

```
function initSelf() {
   var txtUserName = document.getElementById("userName");
   txtUserName.focus();
{

function echoInput() {
   var txtUserName = document.getElementById("userName");
   var divEchoArea = document.getElementById("echoArea");

   divEchoArea.innerHTML = txtUserName.value;
}
```

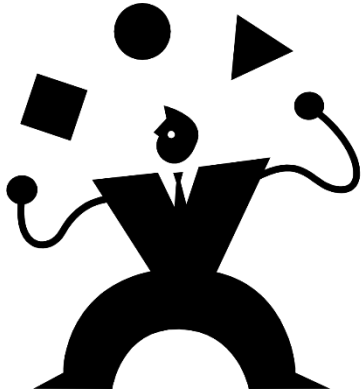


More Events

Event	Occurs	Usage
onunload	The user left the page (e.g., via a link)	Unload resources
onFocus	The field got focus	
onBlur	The field lost focus	
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	Give the user an indication that he's on a hot spot
onMouseOut	The mouse has left the area of the element	Change the mouse cursor to regular arrow

Events

- There are many different events you can code.
- We will see more events examples in the following slides.



DOM

- Document Object Model
- Handling Events
- Built-in classes and objects

Built-in Classes and Objects

- Javascript supplies some built-in classes for your use:
  - String, Date, Array, Math, etc.
- Javascript comes with several pre-initialized objects to interact with the environment:
  - window, document, location, etc.
- Lets examine some of them closely.

The Date Object

- Javascript supplies a strong Date object, inspired by the Java Date Object.
- Lets see some of its power.

The Date Object

- The following code shows a clock:

```
function initSelf() {
   var today = new Date();
   var h= today.getHours();
   var m= today.getMinutes();
   var s= today.getSeconds();
   // add leading zeros
   m = formatTime(m);
   s = formatTime(s);
   document.getElementById('showTime').innerHTML = h+":"+m+":"+s;

   window.setTimeout('initSelf()',400);
{

function formatTime(n) {
   if (n<10) n = "0" + n;
   return n;
{
```

14:18:25

Manipulating Dates

- The following code checks whether my birthday already occurred in the current year.
  - Note that the month starts from 0, so 8 is September.

```
var now = new Date();
var myBirthday = new Date();
var inOneWeek = new Date();

myBirthday.setFullYear(now.getFullYear(), 8, 24);
inOneWeek.setDate(now.getDate()+5);

if (now > myBirthday) {
   alert('After Birthday');
} else if (inOneWeek > myBirthday ) {
   alert('Get Ready, birthday in 5 days');
} else {
   alert('Before Birthday');
}
```

Math

- Use the Math static methods to do mathematical calculations:
  - *max* receives an array of numbers.
  - *random* returns a real number between 0-1.

```
document.write("PI: " + Math.PI + "<br/>");
document.write("Random: " + Math.random() + "<br/>");
document.write(Math.pow(2, 10) + "<br/>");
document.write(Math.max(7, 9, 2) + "<br/>");
document.write(Math.round(7.51) + "<br/>");
```

PI: 3.141592653589793
Random: 0.1014029317983347
1024
9
8

The 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.

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

Using the Navigator

- Get information about the user's browser and OS:

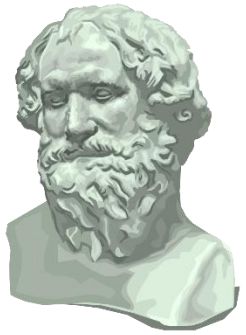
```
var n = navigator;
document.write("<table>");
document.write("<tr><td>UA</td><td>" + n.userAgent + "</td></tr>");
document.write("<tr><td>Platform</td><td>" + n.platform + "</td></tr>");
document.write("<tr><td>CodeName</td><td>" + n.appCodeName + "</td></tr>");
document.write("<tr><td>Name</td><td>" + n.appName + "</td></tr>");
document.write("<tr><td>Version</td><td>" + n.appVersion + "</td></tr>");
document.write("<tr><td>Cookies?</td><td>" + n.cookieEnabled+ "</td></tr>");
document.write("</table>");
```

UA	Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.8) Gecko/2009032609 Firefox/3.0.8 (.NET CLR 3.5.30729)
Platform	Win32
CodeName	Mozilla
Name	Netscape
Version	5.0 (Windows; en-US)
CookieEnabled	true

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)“ />
```



Location

- The Location object holds information about the current URL:

```
alert(location.hostname);
```

Summary

- Javascript is the leading 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.