# Internet Software Technologies
# Dynamic HTML – *part one*

IMCNE

A.A. 2008/09

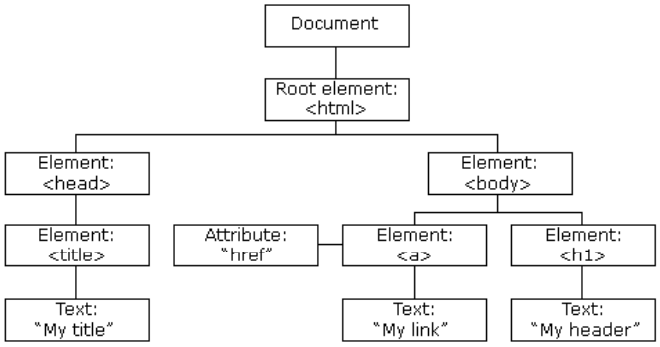
Gabriele Cecchetti

## What is DHTML ?

- DHTML stands for **D**ynamic **HTML**.
- DHTML is NOT a language or a web standard.
- DHTML is a TERM used to describe the technologies used to make web pages dynamic and interactive.
- DHTML = HTML + CSS + JavaScript

# DOM

- The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents.
- D.O.M. is an API for HTML document.
- The DOM presents an HTML document as a tree-structure (a node tree), with elements, attributes, and text.






# What is HTML DOM ? (1/2)

- A standard object model for HTML
- A standard programming interface for HTML
- Platform- and language-independent
- A W3C standard

## What is HTML DOM ? (2/2)

- The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods** (interface) to access them.
- In other words:
- **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**



## HTML DOM Nodes

- According to the DOM, everything in an HTML document is a node.
  - The entire document is a document node
  - Every HTML tag is an element node
  - The text in the HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - Comments are comment nodes
- The text of an element node is stored in a text node.
- Example:

```
<body>
 <p> This a text node </p>
</body>
```

# HTML DOM Node Tree

- The HTML DOM views a HTML document as a tree-structure. The tree structure is called a **node-tree.**
- All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created.
- The nodes in the node tree have a hierarchical relationship to each other:
  - 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



# HTML DOM Properties

- `x.innerHTML` - the inner text value of x
  *(a HTML element)*
  *(NOT STANDARD property !!!)*
- `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 *(it's an array)*
- `x.attributes` - the attributes nodes of x *(it's an array)*

## HTML 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



## innerHTML

- The easiest way to get or modify the content of an element is by using the innerHTML property.
- **`innerHTML`** is not a part of the W3C DOM specification. However, it is supported by all major browsers.
- The **`innerHTML`** property is useful for returning or replacing the content of HTML elements (including `<html>` and `<body>`), it can also be used to view the source of a page that has been dynamically modified.

# innerHTML Example

```
<body>
 <p id="intro"> This is an intro </p>
 <script>
    txt=document.getElementById("intro").innerHTML
 </script>
</body>
```



# *Standard* way to get the content of an element

- Same example:

```
<body>
 <p id="intro"> This is an intro </p>
 <script>
   txt=document.getElementById("intro").childNodes[0].no
   deValue
 </script>
</body>
```

- Usually `nodevalue` property is also faster than `innerHTML` property, so the former should be preferred.

# Accessing DOM Nodes

- You can access a node in three ways:
  1. By using the `getElementById()` method
  2. By using the `getElementsByTagName()` method
  3. By navigating the node tree, using the node relationships.



# The getElementById() Method

- To get a reference to an element:

  *ref* = **document.getElementById**(“*id*”);
- Example: *if HTML document has*

```
<div id="mydiv"> … </div>
```

- *we can write:*

```
ref = document.getElementById(“mydiv”);
```

## The getElementsByTagName() Method

- `getElementsByTagName()` returns all elements with a specified tag name.
- ***Syntax***:

```
node.getElementsByTagName("tagname");
```

- The following example returns a *nodeList* of all <p> elements in the document:

```
x=document.getElementsByTagName("p");
```

where x is an array; to access the second <p> element in the list you can write:

```
y=x[1];
```

and of course `x.length` is the length of the such list.



## firstChild, lastChild

- firstChild and lastChild are properties which link the first and last child nodes of an element.
- *Example*: to access the text of an element having id="intro":

```
var x=document.getElementById("intro");
var text=x.firstChild.nodeValue;
```

- *Example*: to access the text of the last `<p>` element of a the `<body>` element:

```
var b=document.getElementByTag("body");
var z=b.lastChild.nodeValue;
```

## How to Change HTML Elements using DOM

- The HTML DOM and JavaScript can be used to change the inner content and attributes of HTML elements dynamically.



## How to Change an HTML style property

- Example using JavaScript:

```
<body>
<script type="text/javascript">
    document.body.bgColor="yellow";
</script>
</body>
```

- Example using CSS and JavaScript:

```
<body>
<script type="text/javascript">
    document.body.style.backroundColor="yellow";
    document.body.style.fontFamily="Arial";
</script>
</body>
```

# How to change the text of an Element

- Example with innerHTML:

```
<body>
  <p id="p1">Hello World!</p>
  <script type="text/javascript">
  document.getElementById("p1").innerHTML="New text!";
  </script>
</body>
```

- Example with nodeValue:

```
<body>
  <p id="p1">Hello World!</p>
  <script type="text/javascript">
  document.getElementById("p1").firstChild.nodeValue="N
  ew text!"; </script>
</body>
```



# How to create a new element (1/2)

- Example:

```
<body>
  <div id="news"></div>
  <script type="text/javascript">
  newpar=document.createElement("p");
  mydiv=getElementById("news");
  mydiv.appendChild(newpar);
  newtxt=document.createTextElement("A news");
  newpar.appendChild(newtext);
  </script></body>
```

- Note that when you create an element you must append it to something inside the HTML document to give it scope (visibility).

## How to create a new element (2/2)

- The syntax to create new elements is:

```
elemnoderef = document.createElement(htmltag)
txtnoderef = document.createTextNode(textstring)
```

  while the syntax to append or insert these elements in the document is:

```
fatherObj.appendChild(childObj)
parentObj.insertBefore(childObj,brotherObj)
```



## How to create clone nodes

- You can use the method `cloneNode()` which creates a copy of a specified node. Syntax:

```
newNode=oldNode.cloneNode(boolean);
```

  where if the boolean parameter is:

  `true` → the cloned node include all attributes and child nodes of the original node.

  `false` → the cloned node does not include attributes nor child nodes of the original node.

## How to test if a node has child nodes ?

- You can use the method `hasChildNodes()` to test if a node has one or more child nodes. Syntax

```
booleanFlag=aNode.hasChildNodes();
```

- Then you can get the child nodes using childNodes array. Example:

```
<body><p> abc </p> <p> cde </p> <p> fgh </p>
<script>
  body=document.getElementByTag("body");
  if (body.hasChildNodes())
    for (i = 0; i < body.length; i++)
document.writeln(body.childNodes[i].firstChild.nodeValu
  e); </script></body>
```



## How to replace a node ?

- You can use the `replaceChild()` method to replace a node. Syntax:

```
replacedNode = parentNode.replaceChild(newChild,
oldChild);
```

- Example:

```
<body><p> abc </p> <p> cde </p> <p> fgh </p>
<script>
  newpar=document.createElement("p");
  newtxt=document.createTextElement("xyz");
  newpar.appendChild(newtxt);
  firstPar=document.getElementByTag("body").firstChild;
  firstPar.parentNode.replaceChild(newpar,firstPar);
</script></body>
```

# How to remove a node ?

- You can use the `removeChild()` method to remove a node. Syntax:

```
oldChild = element.removeChild(child);
```

- Example:

```
<body><p> abc </p> <p> cde </p> <p> fgh </p>
<script>
  body=document.getElementByTag("body")
  firstPar=body.firstChild;
  body.removeChild(firstPar);
</script></body>
```



# Summary of main DOM methods

- *To create a new element*
  - NewObj= **document.createElement**(“Tag”)
- *To create a new text node*
  - NewObj= **document.createTextNode**(string)
- *To insert an element or a text node previusly created*
  - FatherObj.**appendChild**(childObj)
  - FatherObj.**insertBefore**(childObj,brotherObj)
- *Other methods:*
  - NewObj= Obj.**cloneNode**(flag)
  - Flag= Obj.**hasChildNodes**()
  - Obj.**removeChild**(child)
  - ParentObj.**replace**(newObj,childObj)

## Summary of main DOM properties

- **nodeName**
- **nodeValue**
- **parentNode**
- **childNodes[]**
- **firstChild**
- **lastChild**
- **previousSibling**
- **nextSibling**



## Read and Set HTML attributes

- Two methods:
  **`getAttribute`**`(`*`attribute_name`*`)`
  **`setAttribute`**`(`*`attribute_name, attribute_value`*`)`
- Example:
  var table = document.getElementById("idtable");
  table.**setAttribute**("width","400");
- Example

```
function attribute(obj) {
  var str=””;
  for (var i in obj)
      str += i + ": " + obj.getAttribute(i) + "\n";
  document.writeln(str);
}
```