DOM

Document Object Model

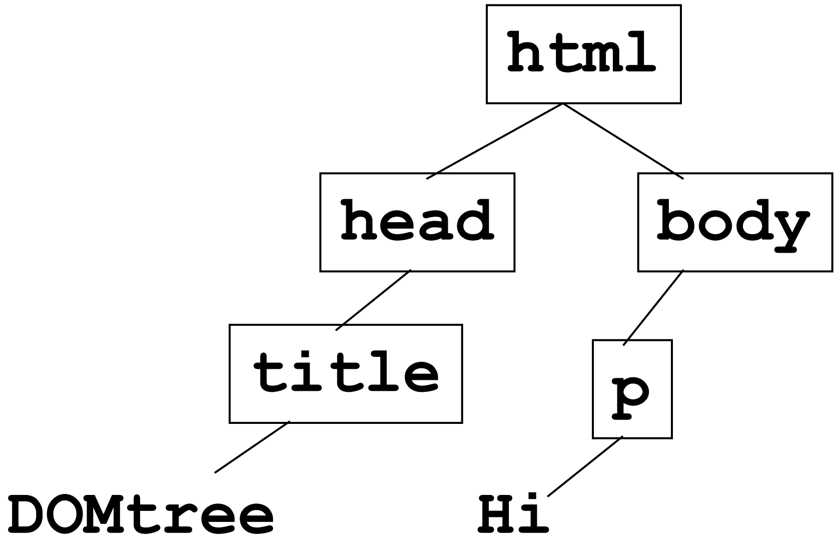
DOM

- W3C definition
  - The Document Object Model (DOM) is an application programming interface (*API*) for valid *HTML* and well-formed *XML* documents.
- It is a way in which elements of HTML, XHTML and XML can be parsed, accessed and modified.
- JavaScript provides API for DOM using which we can access HTML elements.

Nodes

- Every element/tag is considered as a NODE.
- A DOM tree can be built using the nodes.
- Example:

```
<html>
   <head>
    <title>DOMtree</title>
  </head>
   <body>
   <p> Hi</p>
   </body>
</html>
```



Traversing the DOM tree

- Getting an element:
  - **`document.getElementsByTagName('tag')[index]`**
  - Example: to get to the **`p`** tag

  `document.getElementsByTagName('p')[0]`
- On any node following properties can be used to traverse
  - **`firstChild`**
  - **`lastChild`**
  - **`childNodes[index]`**
- To get to the first node (that is **`<html>`**)
  - **`document.firstChild`**
- To get the value of the node: **`nodeValue`**
- To get the name of the node: **`nodeName`**

We write **`document.getElementsByTagName.`** Is **`document`** an object in JavaScript?

That is right. **`document`** is a predefined object available to JavaScript which indicates the current document.

Getting an element by tag name

```
<html>
     <head>
          <title>DOMtree</title>
     </head>
     <body>
          <p> Hi</p>
     </body>
</html>
```

Ways to get to **`<p>`** node:

- **`document.childNodes[0].childNodes[1].childNodes[0];`**
- **`document.getElementByTagName(‘p’)[0];`**
- **`document.body.childNodes[0];`**

Example to print Hi

```
<html>
  <head><title>DOMtree</title></head>
  <body>
<p>Hi</p>
<script>
function f(){
var x =
  document.getElementsByTagName(‘p')[0].first
  Child;
 alert("Says "+ x.nodeValue);
}
f();
</script>
</body>
</html>
```

Case insensitive

Tags must be available before the call.
`<p>Hi</p>` must appears before the call

Referencing attributes

- **`attributes`** property of the node can be used to get all the attributes in the form of array.
- **`nodeName`** and **`nodeValue`** is used to get the values name-value pair of the attribute.
- Based on the browser, **`attributes`** array return either
  - all empty attributes of a particular node (whether explicitly defined or not) →IE 7
  - Or only the explicitly defined attribute → Chrome
- The case of the attribute, node names also differ from browser to browser.

```
<html>
     <body bgColor="red"><p> Hi</p>
<script>
function f(){
var x2 = document.body;
for( var x = 0; x < x2.attributes.length; x++ )
{
if( x2.attributes[x].nodeName=='bgColor' ||
x2.attributes[x].nodeName=='bgcolor')
 {
    alert( 'The page is of '+
x2.attributes[x].nodeValue+' color' );
  }
}
}
f();
</script></body></html>
```

IE

Chrome

Get, set and remove attribute

- Another way to get the value of an attribute is using **`getAttribute()`** method
  - **`getAttribute('<attributeName>')`**
- Attributes can also be set using **`setAttribute()`** method
  - **`setAttribute ('<attributeName>', '<attributeValue>')`**
- To remove an attribute
  - **`removeAttribute('<attributeName>')`**

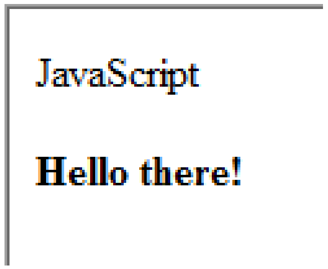
IE7- (and some minor browsers) cannot set values for style,class or event handlers, using setAttribute. IE8 has fixed most of these, but still cannot set event handlers. A few more browsers also have trouble reading these attributes using getAttribute.

```
<html>
    <body bgColor="red"><p> Hi</p>
<script>
function f(){
var x2 =
document.body.getAttribute("bgColor");
alert("color "+ x2+ " is changed to
green");
document.body.setAttribute("bgColor","g
reen");
}
f();
</script>
</body></html>
```

Creating element and adding, removing, replacing new nodes

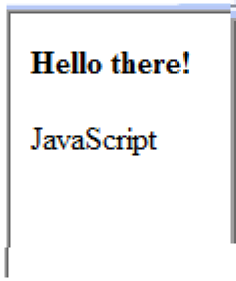
- **`createElement()`** and **`createTextNode`** can be used to create new elements (tags)or text nodes.
- **`appendChild()`** and **`insertBefore()`** are used to append the node and insert before a specified node
- **`replaceChild()`** and **`removeChild()`** are used to replace an existing node with a new node and remove a particular node.
- **`hasChildNodes()`** returns **`true`** if the node has child nodes.

```
<html>
     <head><title>DOMtree</title></head>
     <body><p>JavaScript</p>
<script>
function f(){
var
epar=document.body.getElementsByTagName('p')[0];
var para = document.createElement('b');
var text = document.createTextNode('Hello
there!');
para.appendChild(text);
// document.body.replaceChild(para,epar);
 //document.body.appendChild(para);
 document.body.insertBefore(para,epar);
}
f();
</script></body>
</html>
```



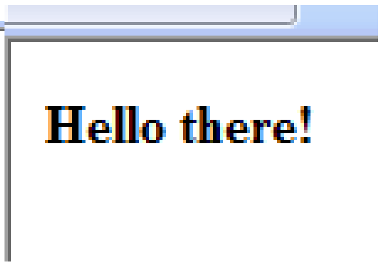
JavaScript

Hello there!



Hello there!

JavaScript



Hello there!

Dynamically change sections of HTML

```
<html><head><title>change</title>
<script>
<!--
function change(){
var x1=prompt("Enter your name","");
var y1=prompt("Enter your favourite food","");

var
es=document.getElementsByTagName(‘div')[0];

while (es.hasChildNodes()){
        es.removeChild(es.firstChild);
      }
p1=document.createElement('p');
node=document.createTextNode("Name:  " +x1);
p1.appendChild(node);
es.appendChild(p1);
p2=document.createElement('p');
node=document.createTextNode("Food:  " +y1);
p2.appendChild(node);
es.appendChild(p2);
}
//-->
</script></head>
<body>
<div id="sec">
<p>Name: XXXX</p>
<p>Food: YYYY</p>
</div>
<script> change()</script></body></html>
```

`getElementById`

- To create a section or division in HTML **`div`** tag is used.
- **`div`** tag can be associated with an **`id`**.
- **`document.getElementById('name')`** can be used to get to the section referred to by the **`div.`**
- (It can also be used on any html element which is associated with id like **`<input>`** tag etc.)
- To change the content of **`div`**, **`innerHTML`** propery comes very handy.

```
<html>
      <head><title>ID</title></head>
      <body>
      Changing the below section:
      <div id="change">
      <p> This section is going to change</p>
      </div>
<script>
function f(){
var name=prompt("enter your name","");
var epara=document.getElementById('change');
epara.innerHTML ="<p><b>Hello, "+
name+"</b></p>";
epara.setAttribute('align',"center");
}
f();
</script>
</body></html>
```