

# JavaScript Document **O**bject **M**odel & Events

# Document Object Model (DOM)


- JavaScript access to the elements of an HTML document.
- An object hierarchy
- Provides access to elements via:
  - ID (**ID** attribute of HTML tags)
  - Order of elements in document
  - Order of element in collection of similar elements.

# Warning!

- Various browsers support various version of the DOM.
- Old browsers may not support the DOM described here...

this will be the ID we use to locate the object that represents this paragraph

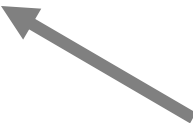
# First Example



```
<P ID=bestparagraph>This is the best paragraph in the document. It is the best because it doesn't contain any words that have the letter 'e'. If you think it is impossible, load me and find out!</P>
```

```
<SCRIPT>  
b = document.getElementById("bestparagraph");  
b.innerHTML="Hi world!";  
</SCRIPT>
```

```
<P>Told you so</P>
```



`innerHTML` is an attribute of an object that corresponds to an HTML tag. It's value is the stuff between the start tag and the end tag.

[dom1.html](#)

# Using IDs

- If you assign an **ID** attribute to all your HTML tags, you can access the objects that correspond to those elements directly.
- The JavaScript **document** object supports a method that allows you to get at the object that represents an HTML tag with a specific ID.

```
document.getElementById("foo");
```

- You must use unique names!

# Important Note

- Generally you should not have JavaScript variables with the same name as an ID you use to identify an HTML element.
  - Some browsers automatically create an variable with the same name as an ID attribute.
  - Some browsers don't.

# Collections

- DOM also supports *collections* of objects,
  - in Javascript these collections are represented as arrays of objects.
- Every HTML element has a **childNodes** collection.
- The document object has collections **forms**, **images**, and **links** in addition to **childNodes**.

# `document.all`

- IE supports the `document.all` collection, which includes an object for every element (tag) in the document.
  - this is not supported by the W3C DOM standard and is no longer supported by Netscape/Mozilla.
- It's best to avoid using `document.all`, but you are likely to see it used in scripts...

# Viewing the **images** collection

```
<SCRIPT>
var txt="";
for (var i=0; i<document.images.length;i++) {
    image = document.images[i];
    name = image.getAttribute("src");
    txt = txt + name + "<BR>";
}
document.writeln("Here are the images found:<BR>\n");
document.write(txt);
</SCRIPT>
```

[dom2.html](#)

← Add this to the bottom of  
any HTML document

# **childNodes** collections

- **childNodes**: just immediate descendants (so subelements are not in the collection).
- Each member of the collection has its own **childNodes** collection!
- You can write a recursive function that can be used to display the structure of a document by looking through the **childNodes** collections.

[recursion.html](#)

# CSS properties

- Style properties can be accessed through the DOM.
  - and can be changed in response to user generated events.

```
document.body.style.backgroundColor="blue";
```

```
for (i=0;i<document.childNodes.length;i++) {  
    document.childNodes[i].style.fontFamily=  
        "Courier";  
}
```

# Element CSS position properties

- You can also mess with CSS position properties

```
<H3 ID=joe STYLE="position:absolute;  
  left:0">
```

```
Hi Joe</H3>
```

```
<SCRIPT>
```

```
document.getElementById('joe').style.left  
  = 200;
```

```
</SCRIPT>
```

[dom4.html](#)

# DOM Issues

- How to access objects that correspond to elements of the document.
  - by **ID**, or in collections.
- How to know what the various object attributes are that change document element properties
  - need to look up the names in a DOM reference.

# JavaScript Events

- It is possible to have the browser run JavaScript programs in response to certain events:
  - user events (moving mouse, clicking, keypress, copy/paste, etc).
  - document/window events (loading, unloading, resizing, scrolling, etc).
  - form submission
  - timed events

# ONLOAD event

- The **onLoad** event is triggered when an element is loaded by the browser.
- You can put an **onLoad** *handler* in the BODY tag:

```
<BODY onLoad="alert ( 'Welcome' ) ;">
```

```
<H1>A title</H1>
```

```
<P>Here is a paragraph
```

...

[dom5.html](#)

# Another onLoad example

```
<SCRIPT>
function start() {
    window.setInterval("updateTime()",1000);
}
var seconds=0;

function updateTime() {
    seconds++;
    document.getElementById("time").innerHTML = seconds;
}
</SCRIPT>

<BODY onLoad="start();">
<H1>Sample Document</H1>

<H3>You have been here <B ID=time>0</B> seconds.</H3>
```

**window.setInterval(*prog*, *delay*)**

- Schedules the JavaScript program ***prog*** to run at a time ***delay*** ms. in the future and at regular intervals after that.
- ***prog*** must be a string (that contains JavaScript code).
- You can stop the code from running with the help of **window.clearInterval()**

# onMouseMove event

- The onMouseMove event is triggered whenever the mouse is moving.
- Can get at the x,y position using the **event** object.
- Here is the example that uses this event to display current mouse coordinates when over a specific element of the document: [mousemove.html](#)

# event object

- Whenever an event causes JavaScript code to run, an object named **event** is created.
- You can find out about the event using this object.
  - the previous example used **clientX** and **clientY**.

# event object attributes

- **clientX, clientY** : coordinates of the mouse.
- **currentTarget**: the object that caused the event (not supported by IE).
- **type**: the name of the event
- **timeStamp**

[eventobject.html](#)

# More events

- **onMouseOver , onMouseOut**
  - mouse moves over or leaves an element.
- **onHelp**
  - user asks for help (F1 or browser help button).
- **onKeyDown , onKeyUp , onKeyPress**
  - keyboard events
- **onUnLoad**
  - element (could be the document) is unloaded (closed).

# Form Fields

- You can use the DOM to get the value of a form field, or to set the value of a form field.
- Assign an ID to each form field.
  - use the **value** attribute to access the current value.

# Form Field Example

```
Name: <INPUT ID=nfield TYPE=TEXT><BR>  
<INPUT TYPE=BUTTON VALUE="Check"  
      onClick="validate();">
```

```
<SCRIPT>  
function validate() {  
  fld = document.getElementById("nfield");  
  if (fld.value != "Dave") {  
    alert("The name you typed is wrong");  
    fld.value="Dave";  
  }  
}
```

[formexample.html](#)