Web Development

Lecture 12 – JavaScript 3
Document Object Model

JavaScript: The Definitive Guide by David Flanagan pub:O'Reilly

Object Oriented Software

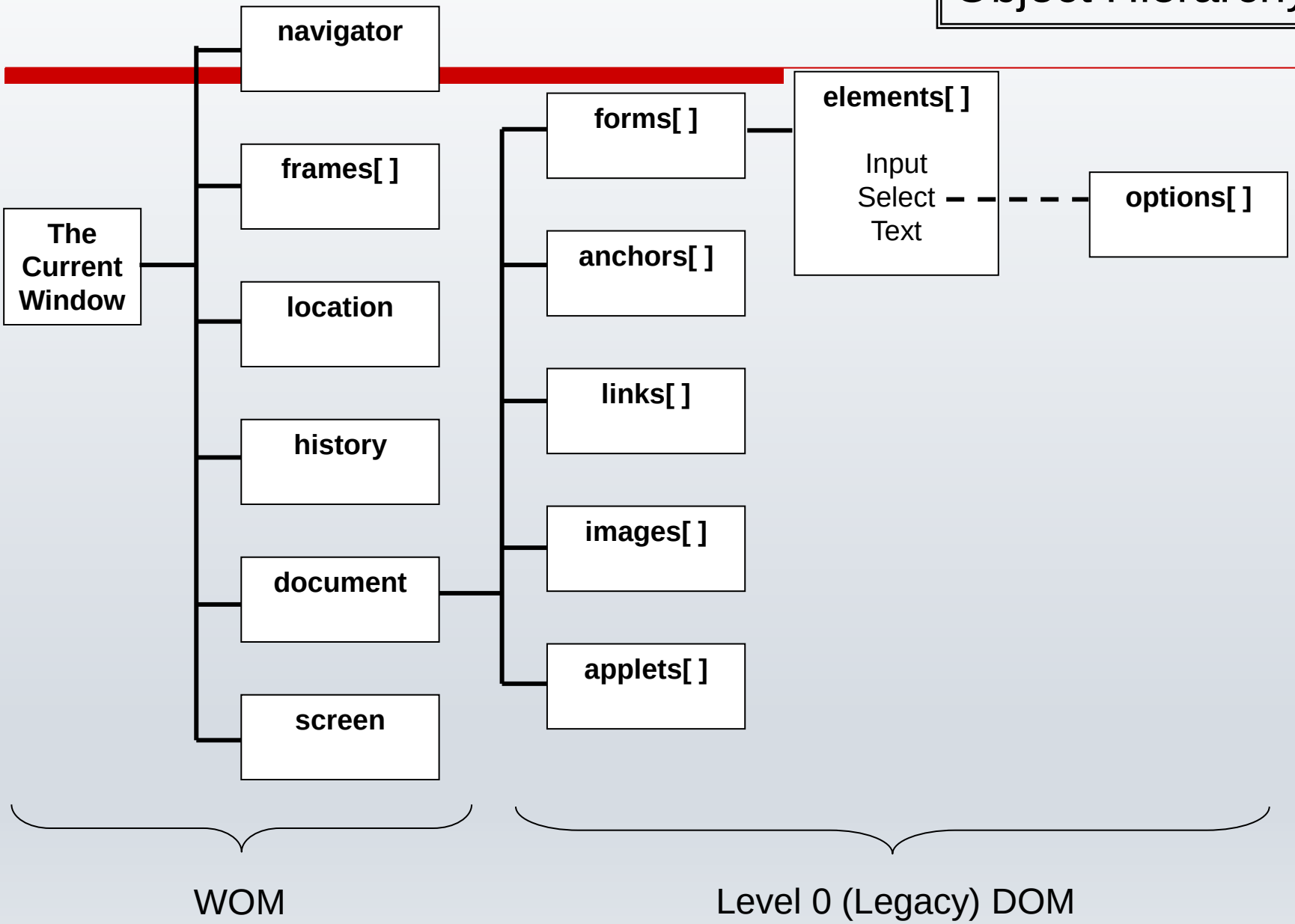
The Browser is constructed as a collection of objects.

An object can have:

Properties	A set of variables which describe the state of the object.
Methods	A set of functions which can be carried out on the object.
Events	These are triggered by the object when some action takes place.

This collection of objects is known as the Window Object Model (WOM).

Object Hierarchy



Level 0 (Legacy) DOM

The previous diagram describes the Level 0 DOM – what was left after the browser wars.

Netscape 3 allowed script access to certain elements such as links, images, forms. This DOM was copied by all other browsers.

IE4 allowed script access to all elements in a document but the DOM was never standardised. Parts of this DOM were copied by other browsers.

Netscape 4 had a strange DOM based on layers. No one copied it – not even Mozilla and Firefox.

W3C bring out a standardised Level 1 DOM in 1998

IE 5 & IE6 supported large parts of the W3C DOM whilst retaining support for the IE4 DOM.

Legacy DOM methods

```
document.write("sometext");
```

This was used to output text into the page whilst it was being parsed.

If you try to use it to output text to a page after it has been parsed (and displayed), it will erase the document (and any scripts it may contain).

The W3C Level 1 DOM provides much better ways of inserting text into a webpage, even after it has been parsed.

Legacy DOM: Object Collections

A set of arrays (images[ ], forms[ ]) which allowed script access to a limited number of the elements on the page. For example:

```
document.image[0].src = "image.gif";
```

would change the value of the **`src`** attribute of the first **`<img>`** tag.

i.e. replace the first image on the page with a different picture.

Legacy DOM: Naming objects

The trouble with numerically indexed object collections is that if the document structure changes, the scripts will no longer work.

The Legacy DOM allowed you to name objects:

```
<forms name="f1">##other stuff##</form>
```

so that no matter where they were in the page, you could refer to them by name:

```
document.forms[0]        // Refer by position
document.forms.f1        // Refer by name
document.forms["f1"]     // Use name as index
```

W3C DOM

In 1998, W3C published the Level 1 DOM specification. This specification allowed access to and manipulation of every single element in an HTML page.

The W3C DOM has different versions:

Level 1 - Defined the tree structure, nodes, elements, attributes
Level 2 – Support for document events, CSS stylesheets
Level 3 – Split into modules:

Core DOM - standard set of objects for any structured document
XML DOM - standard set of objects for XML documents
HTML DOM - standard set of objects for HTML documents
CSS DOM – standard set of objects for CSS stylesheets

etc

Compatibility Tables

There are various sites on the net which attempt to summarise how compatible the various browsers are with the DOMs that are in use.

http://www.quirksmode.org/dom/contents.html

http://www.webdevout.net/

http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(DOM)

Also the reference pages on the www.w3schools.com website.

Accessing The DOM

Just as with other XML-type documents, an XHTML document is thought of as a tree, which is made up of nodes.

Nodes

According to the DOM, every tag in an HTML document is stored as a node.

The DOM says that:

- The entire document is a **document** node
- Every HTML tag is an **element** node
- The texts contained in the HTML elements are **text** nodes
- Every HTML attribute is an **attribute** node
- Comments are **comment** nodes

Node Information

Every node has some properties that contain some information about that node.

The properties are:

- nodeType
- nodeName
- nodeValue

The nodeType property

This contains a numeric code which represents the nodetype

Element	1
Attribute	2
Text	3
Comment	8
Document	9

The nodeName property

nodeType	**nodeName**
element	*the element name*
attribute	*the attribute name*
text	#text
document	#document

The nodeValue property

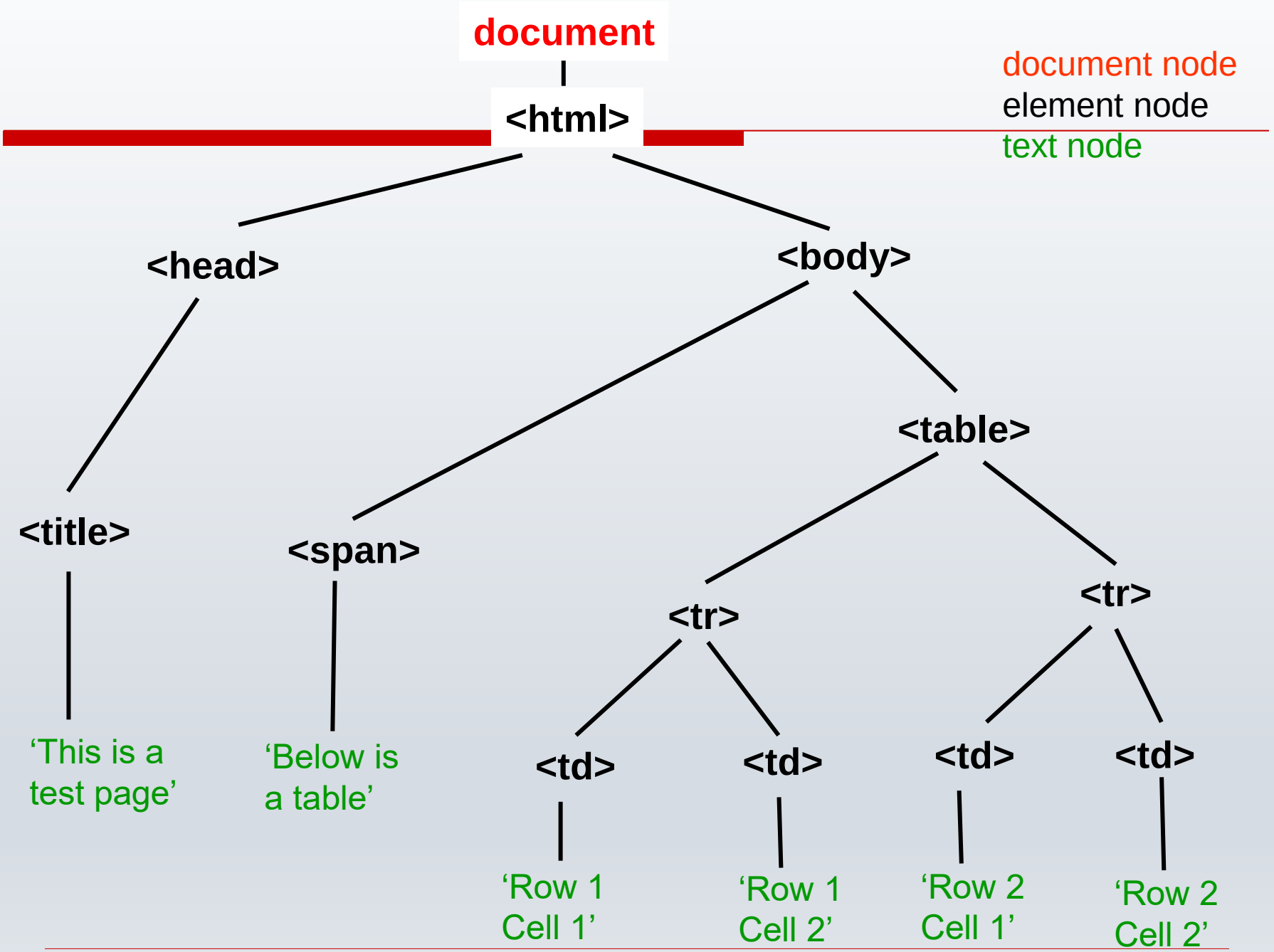
Type of node	**nodeValue**
text	*the text*
attribute	*the attribute value*
document	not available
element	not available

Document Tree

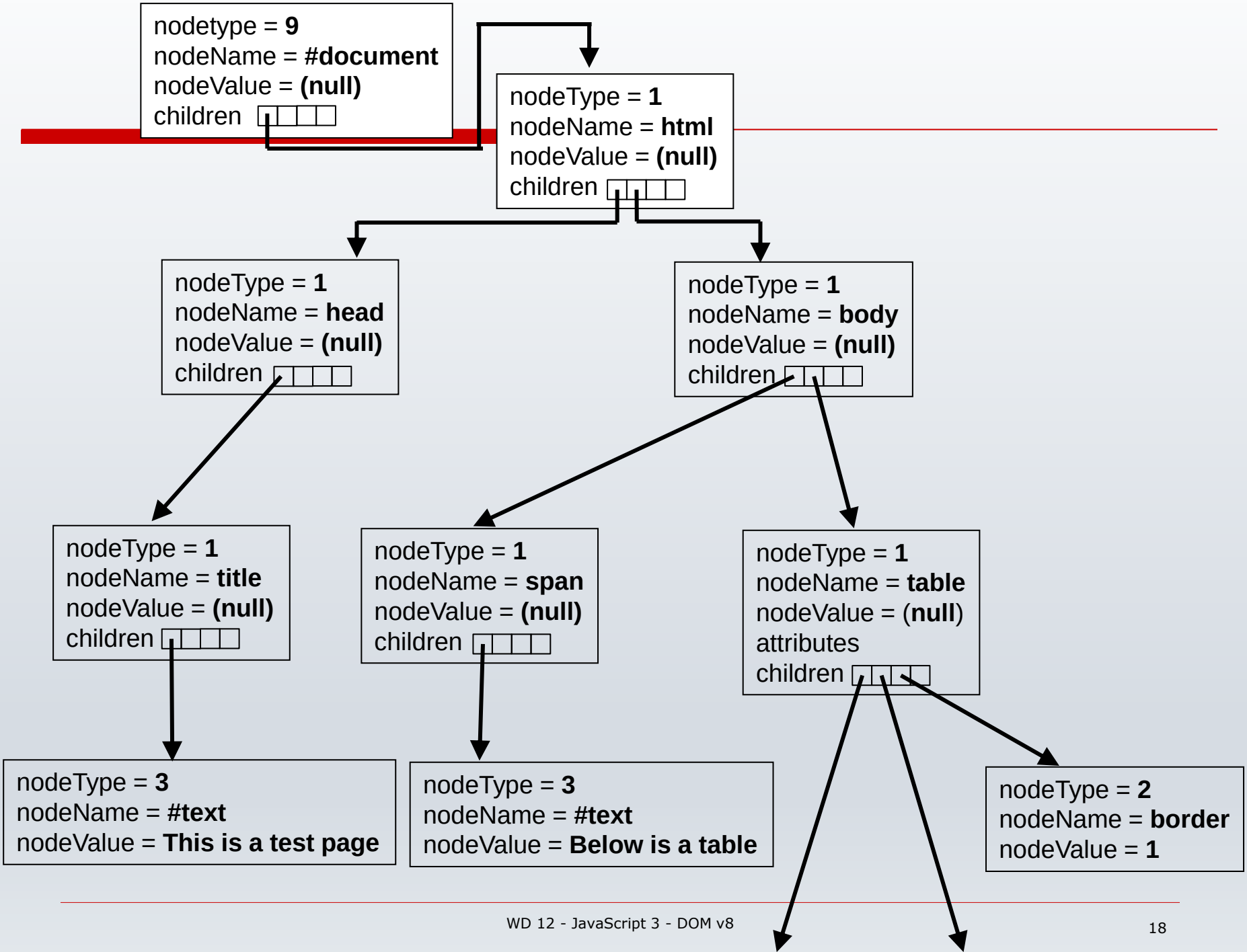
```
<html>
  <head>
    <title>
       This is a test page
    </title>
  </head>

<body>
  <span>Below is a table</span>
  <table border="1">
    <tr>
      <td>Row 1 Cell 1</td>
      <td>Row 1 Cell 1</td>
    </tr>
    <tr>
      <td>Row 1 Cell 1</td>
      <td>Row 1 Cell 1</td>
    </tr>
  </table>
</body>
</html>
```

A HTML page can be thought of as a tree structure:


document
<html>
document node
element node
text node
<head>
<body>
<title>

<table>
<tr>
<tr>
'This is a test page'
'Below is a table'
<td>
<td>
<td>
<td>
'Row 1 Cell 1'
'Row 1 Cell 2'
'Row 2 Cell 1'
'Row 2 Cell 2'


nodetype = 9
nodeName = #document
nodeValue = (null)
children
nodeType = 1
nodeName = html
nodeValue = (null)
children
nodeType = 1
nodeName = head
nodeValue = (null)
children
nodeType = 1
nodeName = body
nodeValue = (null)
children
nodeType = 1
nodeName = title
nodeValue = (null)
children
nodeType = 1
nodeName = span
nodeValue = (null)
children
nodeType = 1
nodeName = table
nodeValue = (null)
attributes
children
nodeType = 3
nodeName = #text
nodeValue = This is a test page
nodeType = 3
nodeName = #text
nodeValue = Below is a table
nodeType = 2
nodeName = border
nodeValue = 1

Firefox DOM Inspector Add-on

DOM Inspector

File Edit Search View Help

file:///E:/3%20BournemouthPoole%20College/FDSc%20BitCit/Modules/Web%20Development%20C1/Exercises/11%20JavaScr Inspect

Document - DOM Nodes

nodeName	nod...	nodeValue
#document	9	
HTML	1	
HEAD	1	
TITLE	1	
#text	3	This is a test page
BODY	1	
#text	3	
SPAN	1	
#text	3	Below is a table
#text	3	
TABLE	1	
#text	3	
TBODY	1	
TR	1	
#text	3	
TD	1	
#text	3	Row 1 Cell 1
#text	3	
TD	1	
#text	3	Row 1 Cell 1
#text	3	
#text	3	
TR	1	
#text	3	
TD	1	
#text	3	Row 1 Cell 1
#text	3	
TD	1	
#text	3	Row 1 Cell 1
#text	3	
#text	3	
#text	3	

Object - Javascript Object

Property	Value
Subject	[object HTMLTableElement]
nodeName	"TABLE"
nodeValue	(null)
nodeType	1
parentNode	[object HTMLBodyElement]
childNodes	[object NodeList]
firstChild	[object Text]
lastChild	[object HTMLTableSectionElem...
previousSibling	[object Text]
nextSibling	[object Text]
attributes	[object NamedNodeMap]
ownerDocument	[object HTMLDocument]
insertBefore	function insertBefore() { [na...
replaceChild	function replaceChild() { [na...
removeChild	function removeChild() { [na...
appendChild	function appendChild() { [na...
hasChildNodes	function hasChildNodes() { [...
cloneNode	function cloneNode() { [nati...
normalize	function normalize() { [nativ...
isSupported	function isSupported() { [na...
namespaceURI	(null)
prefix	(null)
localName	"TABLE"
hasAttributes	function hasAttributes() { [n...
ELEMENT_NODE	1
ATTRIBUTE_NODE	2
TEXT_NODE	3
CDATA_SECTION_NODE	4
ENTITY_REFERENCE_NODE	5
ENTITY_NODE	6
PROCESSING_INSTRUCTIO...	7
COMMENT_NODE	8
DOCUMENT_NODE	9

Note the extra text nodes

The DOM of a slightly different page...

DOM Inspector

File Edit Search View Help

file:///E:/3%20BournemouthPoole%20College/FDSc%20BitCit/Modules/Web%20Development%20C1/Exercises/11%20JavaScrip Inspect

Document - DOM Nodes

nodeName	nod...	nodeValue
#document	9	
HTML	1	
HEAD	1	
TITLE	1	
#text	3	This is a test page
BODY	1	
SPAN	1	
#text	3	Below is a table
TABLE	1	
TBODY	1	
TR	1	
TD	1	
#text	3	Row 1 Cell 1
TD	1	
#text	3	Row 1 Cell 1
TR	1	
TD	1	
#text	3	Row 1 Cell 1
TD	1	
#text	3	Row 1 Cell 1

Object - Javascript Object

Property	Value
Subject	[object HTMLDocument]
location	file:///E:/3%20BournemouthPo...
nodeType	9
nodeValue	(null)
documentElement	[object HTMLHtmlElement]
title	"This is a test page"
referrer	""
styleSheets	[object StyleSheetList]
baseURI	"file:///E:/3%20BournemouthP...
compareDocumentPosition	function compareDocumentPosi...
textContent	(null)
isSameNode	function isSameNode() { [na...
lookupPrefix	function lookupPrefix() { [na...
isDefaultNamespace	function isDefaultNamespace() ...
lookupNamespaceURI	function lookupNamespaceURI(...
isEqualNode	function isEqualNode() { [na...
getFeature	function getFeature() { [nati...
setUserData	function setUserData() { [na...
getUserData	function getUserData() { [na...
DOCUMENT_POSITION_DIS...	1
DOCUMENT_POSITION_PR...	2
DOCUMENT_POSITION_FO...	4

....this one.

```
<html><head><title>This is a test page</title></head>
<body><span>Below is a table</span> <table border="1">
<tr><td>Row 1 Cell 1</td><td>Row 1 Cell 1</td></tr>
<tr><td>Row 1 Cell 1</td><td>Row 1 Cell 2</td></tr>
</table></body></html>
```

When doing DOM scripting, be aware of the effect that indentation can have on the DOM tree.

Firefox, Chrome, and Internet Explorer 10 or above will include whitespaces (carriage return/linefeed) in the tree.

Internet Explorer up to and including IE 9 will not.

See http://www.w3schools.com/dom/dom_mozilla_vs_ie.asp

Microsoft have produced a similar add-on

This is a test page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites

Address E:\3 BournemouthPoole College\FDSc BitCit\Modules\Web Development C1\Exercises\11 JavaScript 3 DOM\ExW Go

Below is a table

Row 1 Cell 1	Row 1 Cell 1
Row 1 Cell 1	Row 1 Cell 1

IE Developer Toolbar

File Find Disable View Outline Images Cache Tools Validate

- <HTML>
  - <HEAD>
    - <TITLE>
  - <BODY>
    - <SPAN>
      - #text
    - #text
    - <TABLE>
      - <TBODY>
        - <TR>
          - <TD>
            - #text
          - <TD>
            - #text
        - <TR>
          - <TD>
            - #text
          - <TD>
            - #text

Attribute: Node: TABLE

Name	Value
border	1

Show Read-Only Properties

Current Style:

Property	Current Value
border-bottom-color	#d4d0c8
border-bottom-style	outset
border-color	#d4d0c8
border-left-color	#d4d0c8
border-left-style	outset
border-right-color	#d4d0c8
border-right-style	outset
border-style	outset
border-top-color	#d4d0c8
border-top-style	outset
display	block
hasLayout	-1

Show Default Style Values

Done My Computer

Document Object

This is where the Window Object Model overlaps with the Document Object Model.

It has a property called:

documentElement

which acts as a reference to the topmost element of the document (eg <html>)

var Container = document.documentElement.

Document Object: Methods

There are two useful methods:

`document.getElementById( )`

Finds an element with the specified ID

`document.getElementsByTagName( )`

Returns a node list of all the elements with the specified tag name.

Example

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

will return a list of all the <p> elements in the document.

```
document.getElementsById('maindiv').getElementsByTagName("p");
```

will return a list of all <p> elements that are descendants of the element with the id="maindiv"

innerHTML

Every HTML element has got an innerHTML property, which refers to the text contained between the open and close tags.

If you assign it a value, you can change the text that appears on the page.

Assume that the HTML page contains the following tag:

```
<span id="result"></span>
```

And the Javascript code contains the following:

```
var resultnode = document.getElementById("result");
resultnode.innerHTML = "This is the result";
```

Then what appeared to be a blank space on the web page will suddenly be replaced by the text:

```
This is the result.
```

Nodelists

You would usually use a loop to process a nodelist:

```
var imglist = document.getElementsByTagName("img");

var limit = imglist.length;

for (var i=0; i<limit; i++)
{
  resize(imglist[i], 50%);
}
```

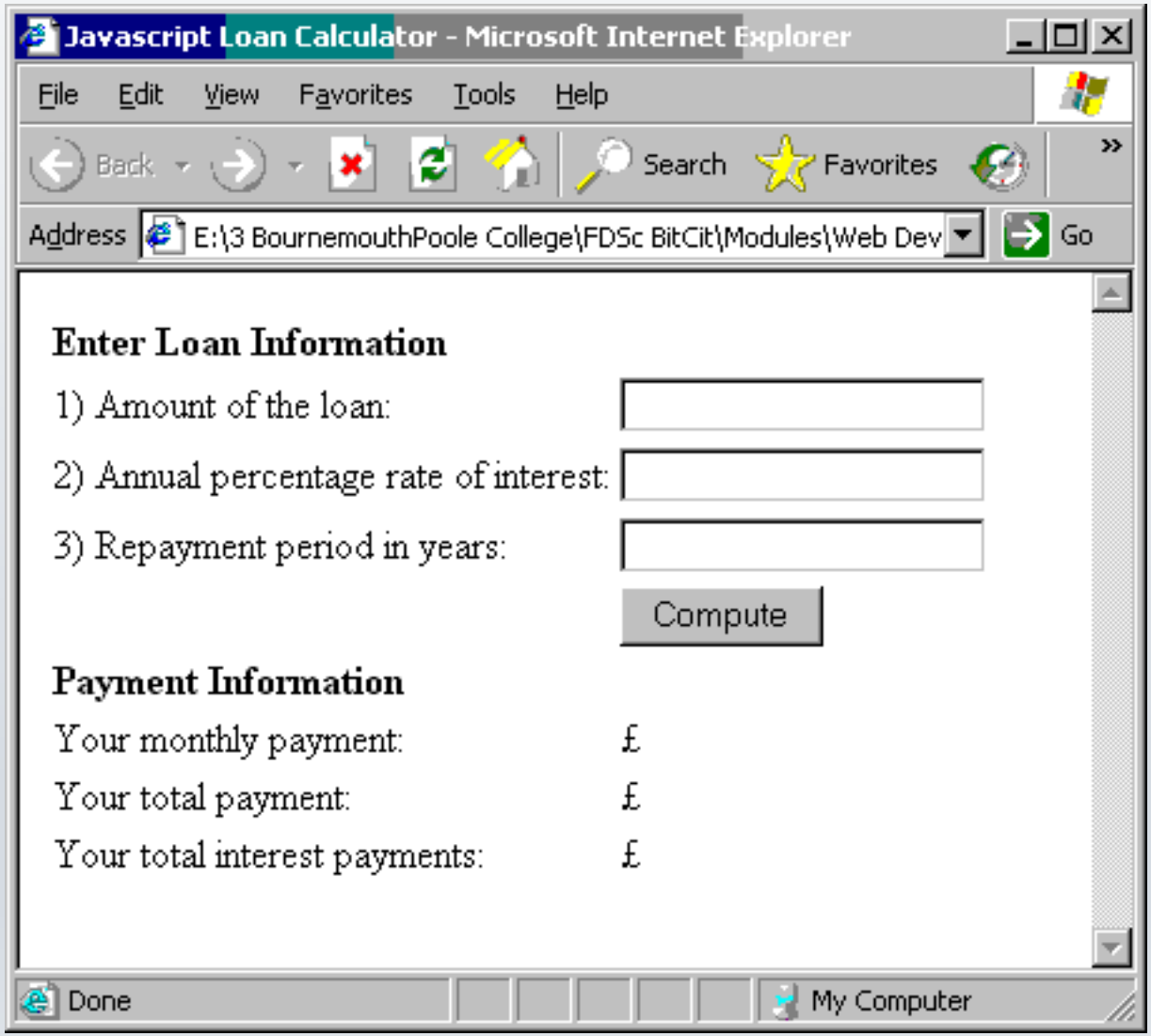
N.B. resize isn't a proper Javascript method – I've just made that up.

Node Object: Properties

Typical properties of a node include:

childNodes[ ]	List of child nodes
firstChild	
lastChild	
nextSibling	
previousSibling	
parentNode	
nodeValue	
nodeName	
nodeType	

Exercise Interest



Extract from Interest.htm

```
<body>
<form name="loandata">

<input type="text" id="principal" onchange="calculate();" />
<input type="text" id="interest" onchange="calculate();" />
<input type="text" id="years" onchange="calculate();" />

<input type="button" value="Compute" onclick="calculate();" />

£<span class="result" id="payment"></span>
£<span class="result" id="total"></span>
£<span class="result" id="totalinterest"></span>

</form>
</body>
```

Interest.js part 1

```
function calculate()
{
  // Get users input from form

  var principal = document.getElementById("principal").value;
  var rawinterest = document.getElementById("interest").value;
  var rawpayments = document.getElementById("years").value;

  // Convert interest from a percentage to a decimal,
  // and annual rate to monthly
  var interest = rawinterest / 100 / 12;
  var payments = rawpayments * 12;

  // Compute monthly payment figure using Javascript maths functions

  var x = Math.pow( 1 + interest, payments);
  var monthly = (principal * x * interest)/(x-1);
```

Interest.js part 2

```
// Get named <span> elements from the form
  var payment = document.getElementById("payment");
  var total = document.getElementById("total");
  var totalinterest = document.getElementById("totalinterest");

  // Check that result is a finite number and if so display it with 2 decimal places
  if (isFinite(monthly))
    {
      payment.innerHTML = monthly.toFixed(2);
      total.innerHTML = (monthly * payments).toFixed(2);
      totalinterest.innerHTML = ((monthly * payments) - principal).toFixed(2);
    }
  else
    {
      payment.innerHTML = " ";
      total.innerHTML = " ";
      totalinterest.innerHTML = " ";
    }
}
```