Scripting the DOM

Terry Marris September 2011

2 Document Object Model Tutorial

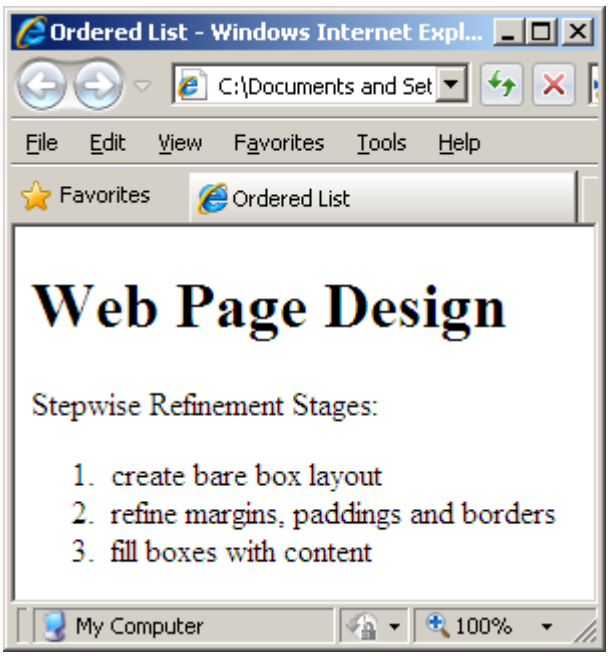
The Document Object Model, DOM provides an interface that allows a scripting language such as JavaScript to interact with HTML components.

2.1 Web Page Structure

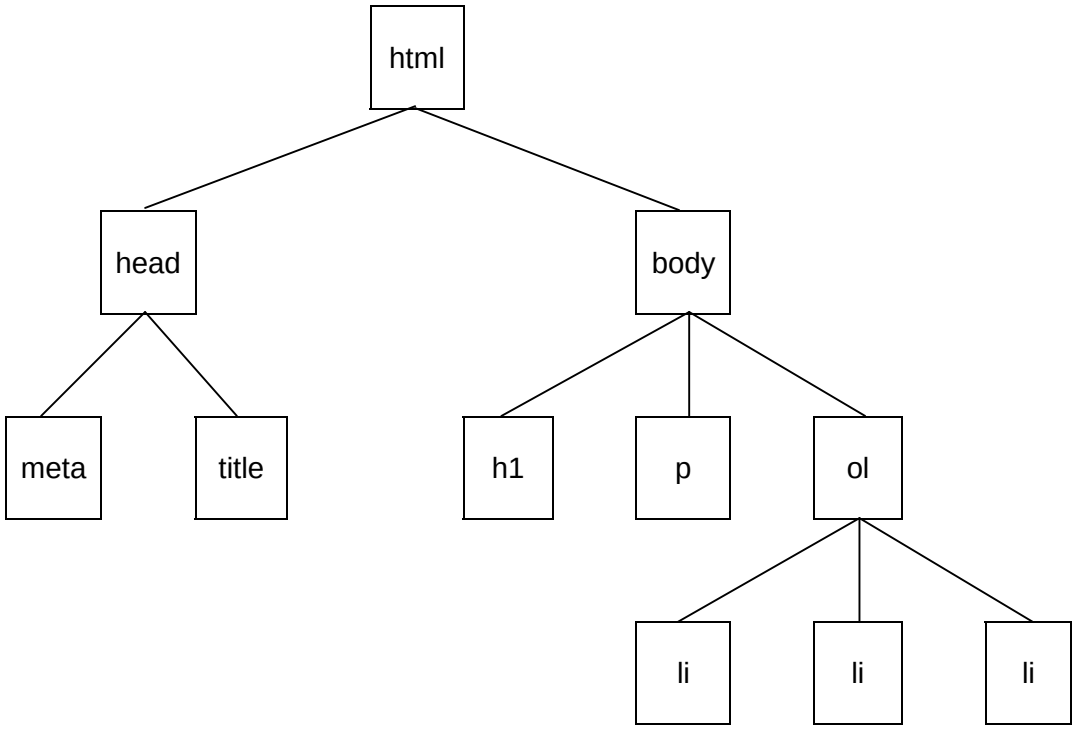
The HTML shown below displays a simple ordered list.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Ordered List</title>
</head>

<body>
<h1>Web Page Design</h1>
<p title="listHeading">Stepwise Refinement Stages:</p>
<ol id="stepwiseRefinement">
<li>create bare box layout</li>
<li>refine margins, paddings and borders</li>
<li>fill boxes with content</li>
</ol>
</body>
</html>
```



We can visualise the HTML document as a hierarchical structure of *nodes*.



<*html*> is the root node.

<*html*> has two children: <*head*> and <*body*>. <*head*> and <*body*> have the same parent, <*html*>. <*head*> and <*body*> are siblings since they both have the same parent and are both on the same level. <*head*> and <*body*> are also parent nodes because they both have children.

<*ol*> is a child of <*body*>. And <*body*> is the parent of <*ol*>.

2.2 Node Types

Element Nodes

The basic elements of a web page, such as <*body*>, <*h1*>, <*p*>, <*ol*>, <*li*>, .. are known as *element nodes*.

An element is named by its *tag*. Paragraph elements have the tag *p*, ordered lists have the tag *ol*, list item elements have the tag *li*, ...

Text Nodes

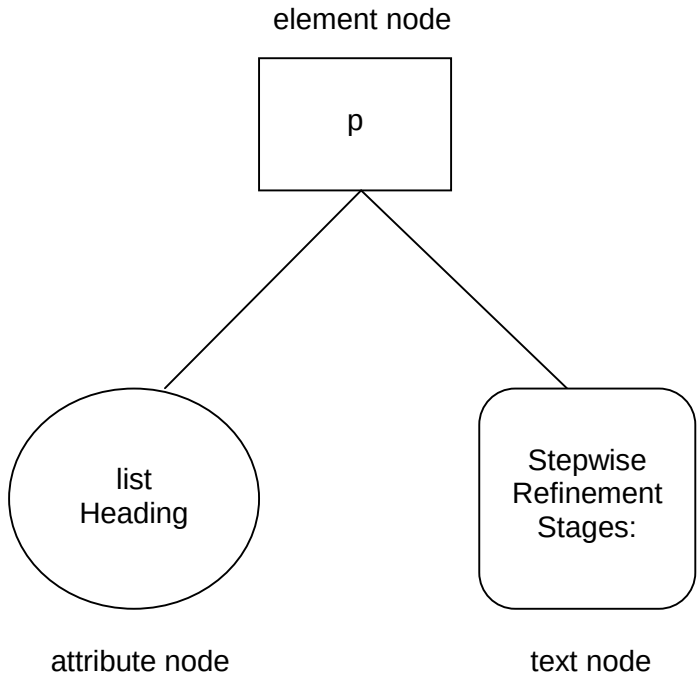
A *text* node contains text. The <*p*> element contains the text *Stepwise Refinement Stages:* This text is a text node.

Attribute Nodes

id, href, *title*, *src* and *alt* are all examples of *attributes*. An attribute provides information about its element.

```
<p title = "listHeading">Stepwise Refinement Stages:</p>
```

Here, *listHeading* is an *attribute* node. Attribute nodes are always contained within element nodes.



In the following sections we see how to access the various nodes of the HTML document.

2.3 getElementById

getElementById() is a DOM method.

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

returns the unique element with the HTML id attribute *stepwiseRefinement*. Looking at the HTML you will see it is the <*ol*> element.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Ordered List</title>
</head>

<body>
<h1>Web Page Design</h1>
<p title="listHeading">Stepwise Refinement Stages:</p>
<ol id="stepwiseRefinement">
<li>create bare box layout</li>
<li>refine margins, paddings and borders</li>
<li>fill boxes with content</li>
</ol>
</body>
</html>
```

Providing an HTML element with a unique id is perhaps the easiest way of accessing that element.

2.4 getElementsByTagName

getElementsByTagName() is another DOM method.

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

returns an array of all list items in the document.

```
var items = document.getElementsByTagName("li");
```

Looking at the HTML you can see there are just three list items.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Ordered List</title>
</head>

<body>
<h1>Web Page Design</h1>
<p title="listHeading">Stepwise Refinement Stages:</p>
<ol id="stepwiseRefinement">
<li>create bare box layout</li>
<li>refine margins, paddings and borders</li>
<li>fill boxes with content</li>
</ol>
</body>
</html>
```

You can visit each list item in turn:

```
for (var i = 0; i < items.length; i++) {
  ...
}
```

You can retrieve every element in a document:

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

You can retrieve just those items inside the element with *id = "stepwiseRefinement"* :

```
var steps = document.getElementById("stepwiseRefinement");
var items = steps.getElementsByTagName("*");
```

This would leave the array *items* containing the three list items.

2.5 getAttribute

getAttribute() is yet another DOM method. With it you can retrieve the value of an element's attribute.

```
var paras = document.getElementsByTagName("p");
for (var i = 0; i < paras.length; i++) {
  var titleText = paras[i].getAttribute("title");
  ...
}
```

Since in our example HTML there is just one paragraph, *titleText* has the value *listHeading*.

If there was no element with the given attribute *getAttribute()* returns *null*.

```
var paras = document.getElementsByTagName("p");
for (var i = 0; i < paras.length; i++) {
  var titleText = paras[i].getAttribute("peanuts");
  if (titleText == null) {
    ...
  }
  ...
}
```

Incidentally, *if (titleText != null) ...* is usually written *if (titleText) ...*

2.6 setAttribute

The *get...* methods listed above retrieve elements and attributes. *setAttribute()* is a DOM method that temporarily overwrites the value of an attribute node. *setAttribute()* has two parameters.

setAttribute(attribute, newValue)

We change the *listHeading* title attribute to *Ordered List*.

```
var paras = document.getElementsByTagName("p");
for (var i = 0; i < paras.length; i++) {
  var titleText = paras[i].getAttribute("title");
  if (titleText === "listHeading") {
    paras[i].setAttribute("title", "Ordered List");
    ...
  }
  ...
}
```

We see a useful application of *setAttribute()* in Chapter 3 Image Gallery.

We have introduced four DOM methods: *getElementById()*, *getElementsByTagName()*, *getAttribute()* and *setAttribute()*. **Next** we see how these four DOM methods can be used to implement an image gallery.

Bibliography

Keith J *DOM Scripting: Web Design with JavaScript and the Document Object Model* Apress 2005
Haverbeke M *Eloquent JavaScript* William Pollock 2011
www.tutorialpoint.com/javascript accessed April 2011
https://developer.mozilla.org accessed June 2011