JAVASCRIPT OBJECTS

Unit 2

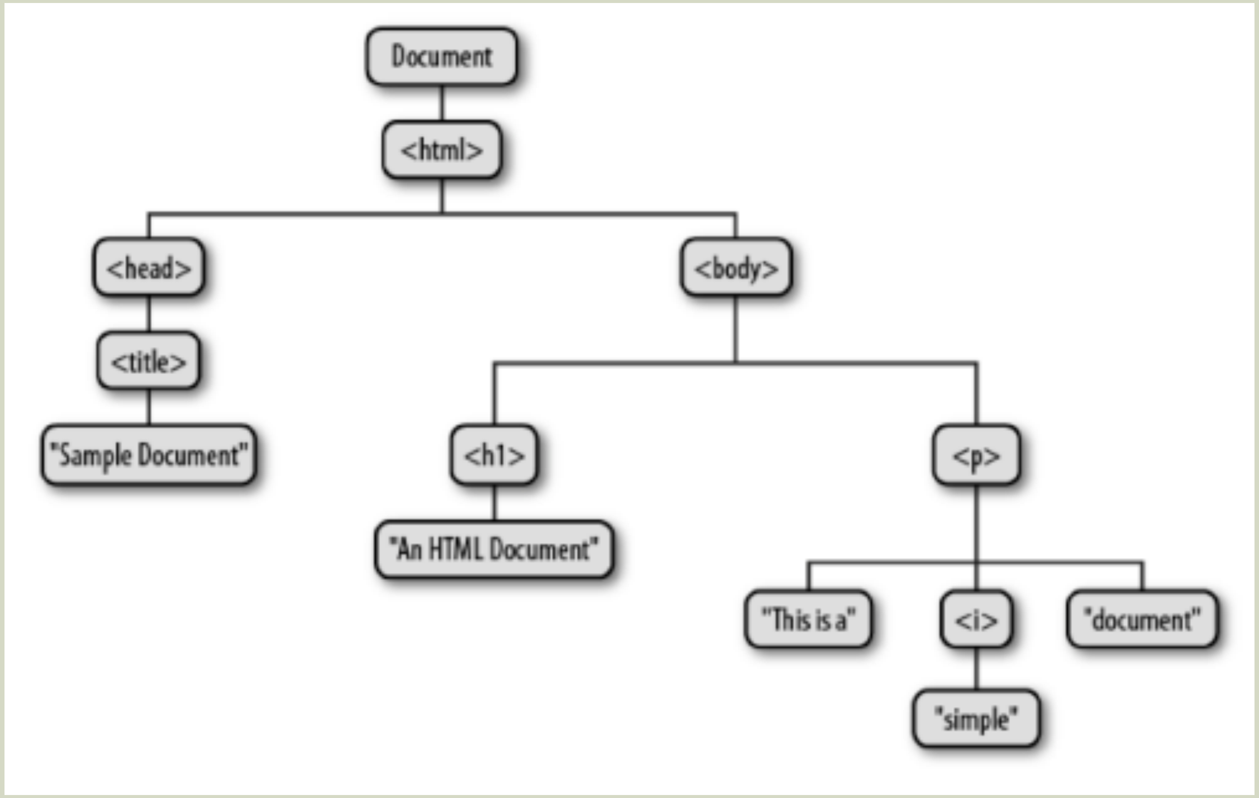
INDEX

1 – WINDOW OBJECT

- Window object
  - The window object represents an open window in a browser.
  - Window object has properties and methods.
  - Properties are barely used.
  - We have already studied some of its methods in the Unit 1. Alert, prompt, parseFloat, parseInt.
  - You can find all the properties and methods in the W3Schools web.
  - http://www.w3schools.com/jsref/obj_window.asp

2 – DOCUMENT OBJECT

- Document object
  - Is the highest object in the DOM structure.



2 – DOCUMENT OBJECTS

- Document object
  - When an HTML document is loaded into a browser, it becomes a document objets.
  - Document object provides properties and methods to access all node objects, from within JavaScript.
  - Is the object that lets you work with Document Object Model (DOM) that represents all of the HTML elements on the page.

2 – DOCUMENT OBJECTS

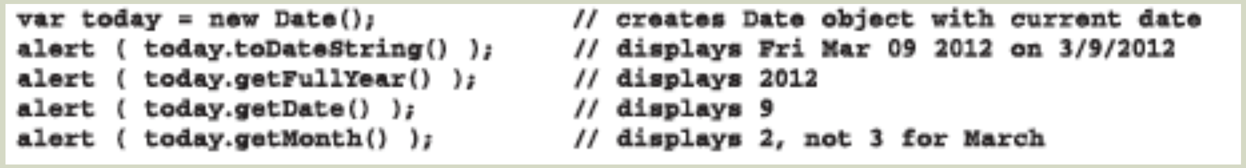
- Document object
  - getElementByID(“introduction”) //Gets the element with id=“introduction)
    - Returns null if no elements with specified id exists
  - write(), getElementByName(), hasFocus(), getElementsByTagName()
  - document.body, document.cookie, document.images
  - http://www.w3schools.com/jsref/dom_obj_document.asp

3 – ELEMENT OBJECT

- Element object
  - Represents an HTML element.
  - Can have child nodes.
  - A NodeList object represents a list of nodes, like an HTML element's collection of child nodes. (We can iterate this list to do operations, Unit 3)
  - Elements can also have attributes. Attributes are attributes nodes.

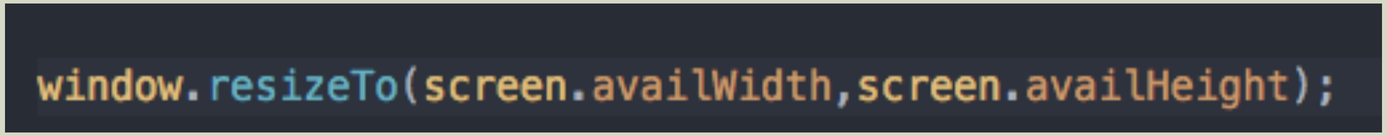
4 – ARE ALSO OBJECTS…

- There more objects in Java. Sobre of them are:
  - Date: To manage dates



```
var today = new Date();              // creates Date object with current date
alert ( today.toDateString() );      // displays Fri Mar 09 2012 on 3/9/2012
alert ( today.getFullYear() );       // displays 2012
alert ( today.getDate() );           // displays 9
alert ( today.getMonth() );          // displays 2, not 3 for March
```

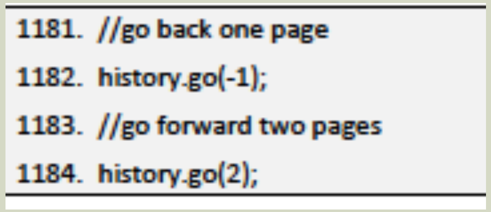
  - Screen: Gives you information about the user's screen resolution.



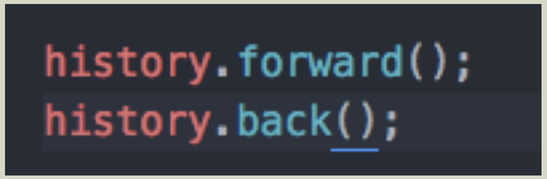
```
window.resizeTo(screen.availWidth,screen.availHeight);
```

4 – ARE ALSO OBJECTS…

- History: Represents the user’s navigation history since the given window was first used.

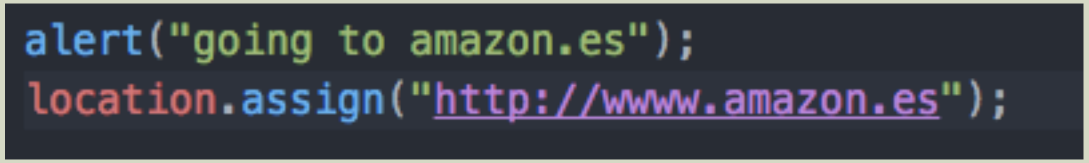


```
//go back one page
history.go(-1);
//go forward two pages
history.go(2);
```



```
history.forward();
history.back();
```

- Location: Gives you information abou the page loaded in the browser.



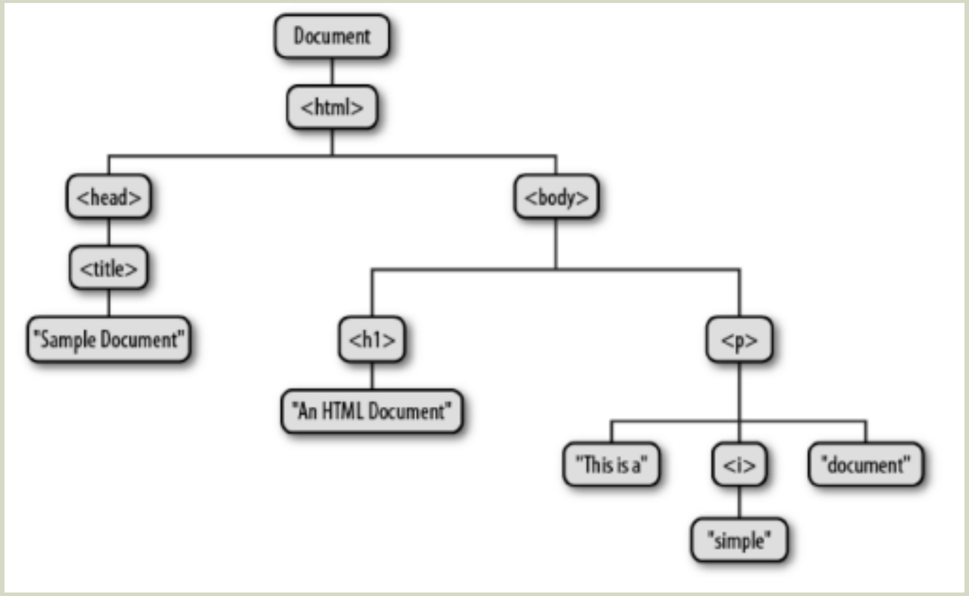
```
alert("going to amazon.es");
location.assign("http://wwww.amazon.es");
```

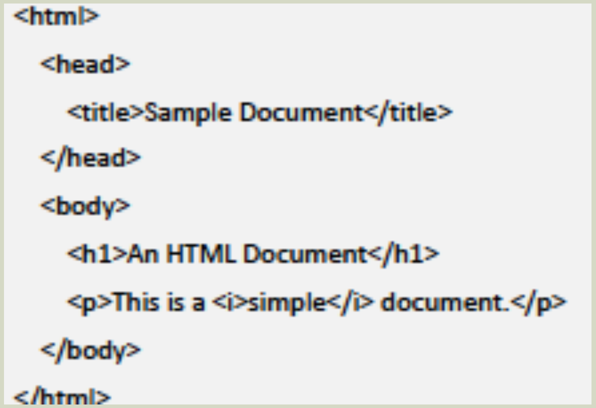
- And much more…

5 – SURFING IN THE DOM

- The Document Object Model is an application programing interface (API) for HTML and XML.
- The DOM represents a document as a hierarchical tree of nodes.
- Any HTML can be represented as a hierarchy of nodes using the DOM.

5 – SURFING IN THE DOM





```
<html>
  <head>
    <title>Sample Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.</p>
  </body>
</html>
```

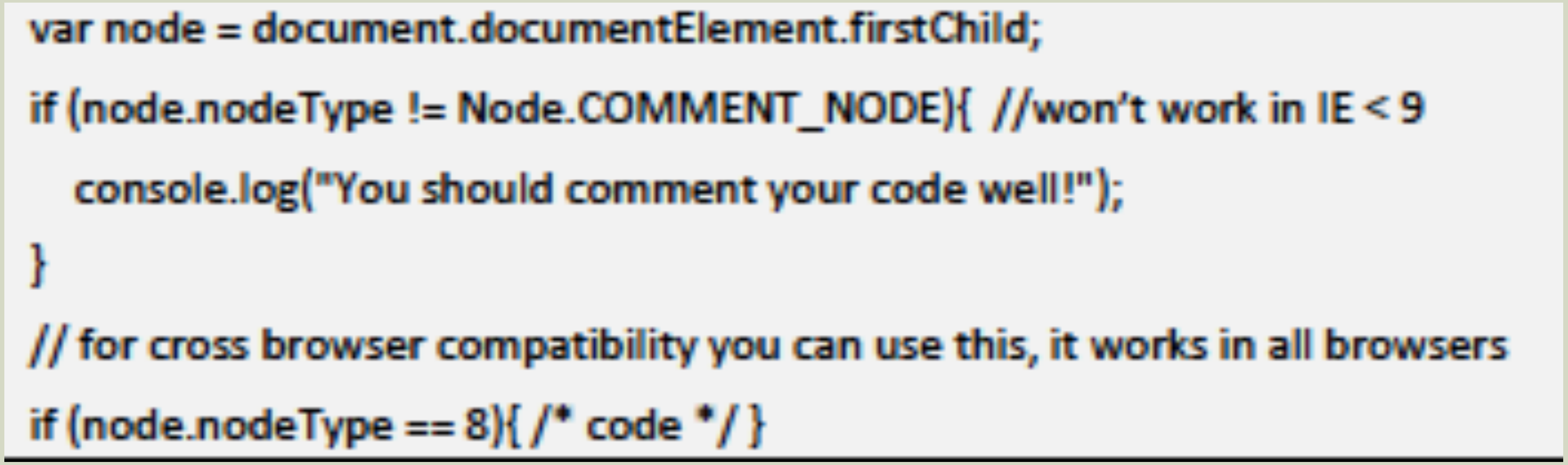
5 – SURFING IN THE DOM

- In total there are 12 node types, all of which inherit from a base type.

- Every node has a nodeType property that indicates the type of node that it is.

- Node.ELEMENT_NODE (1)
- Node.ATTRIBUTE_NODE (2)
- Node.TEXT_NODE (3)
- Node.COMMENT_NODE (8)
- Node.DOCUMENT_NODE (9)

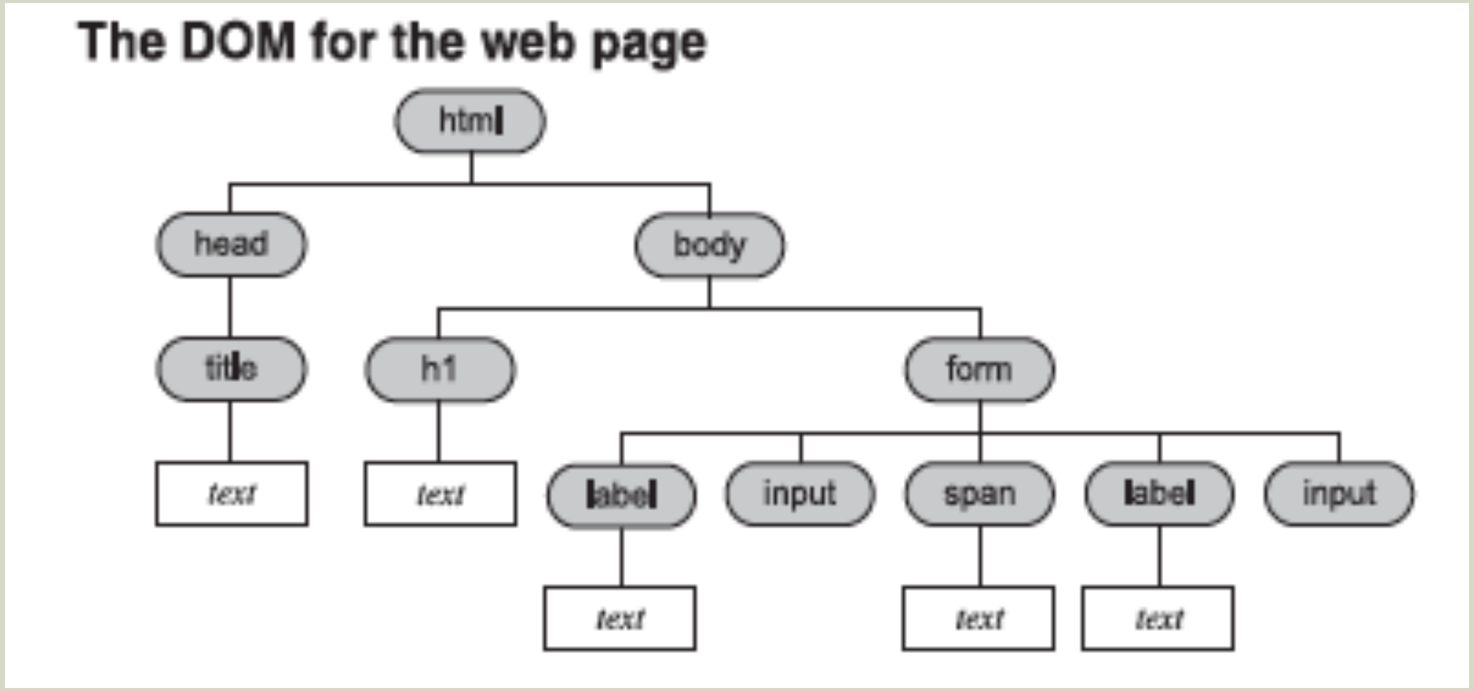
5 – SURFING IN THE DOM



```
var node = document.documentElement.firstChild;
if (node.nodeType != Node.COMMENT_NODE){  //won't work in IE < 9
   console.log("You should comment your code well!");
}
// for cross browser compatibility you can use this, it works in all browsers
if (node.nodeType == 8){ /* code */ }
```

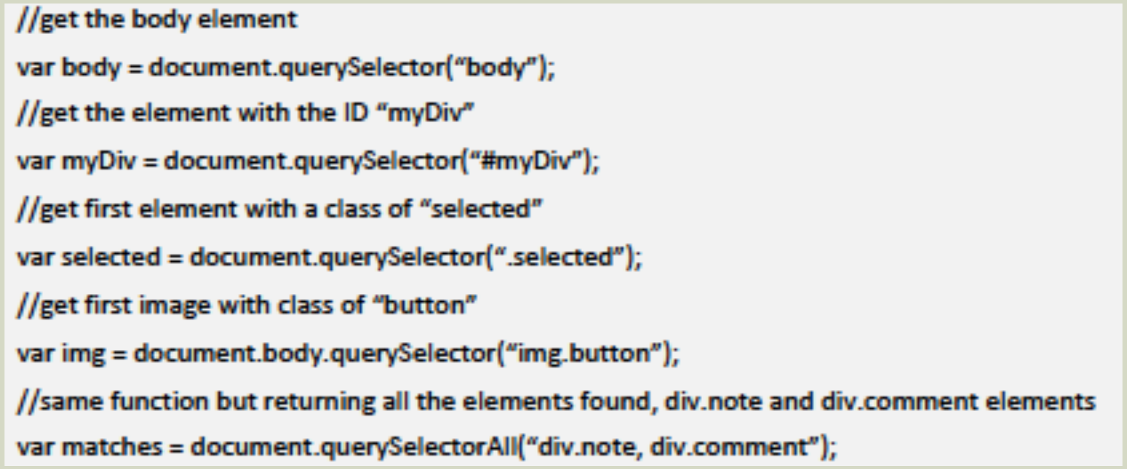
5 – SURFING IN THE DOM

- Create the following HTML.



5 – SURFING IN THE DOM

- Selecting
  - getElementsByTagName()
    - var praragraphs = document.getElementsByTagName(“p”);
  - getElementsByName()
  - getElementsByID()
  - querySelector() and querySelectorAll() ----- HTML5

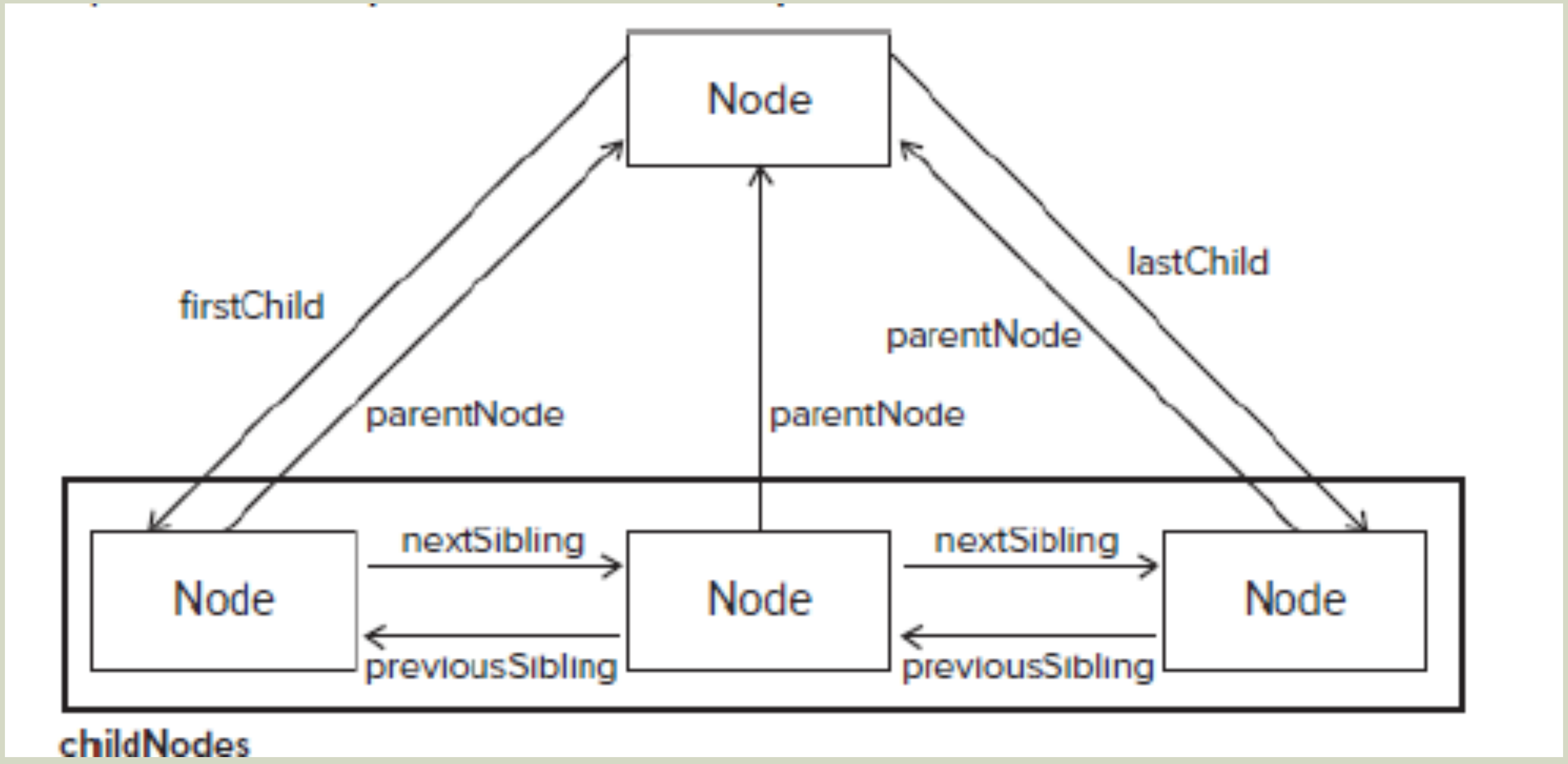


```
//get the body element
var body = document.querySelector("body");
//get the element with the ID "myDiv"
var myDiv = document.querySelector("#myDiv");
//get first element with a class of "selected"
var selected = document.querySelector(".selected");
//get first image with class of "button"
var img = document.body.querySelector("img.button");
//same function but returning all the elements found, div.note and div.comment elements
var matches = document.querySelectorAll("div.note, div.comment");
```

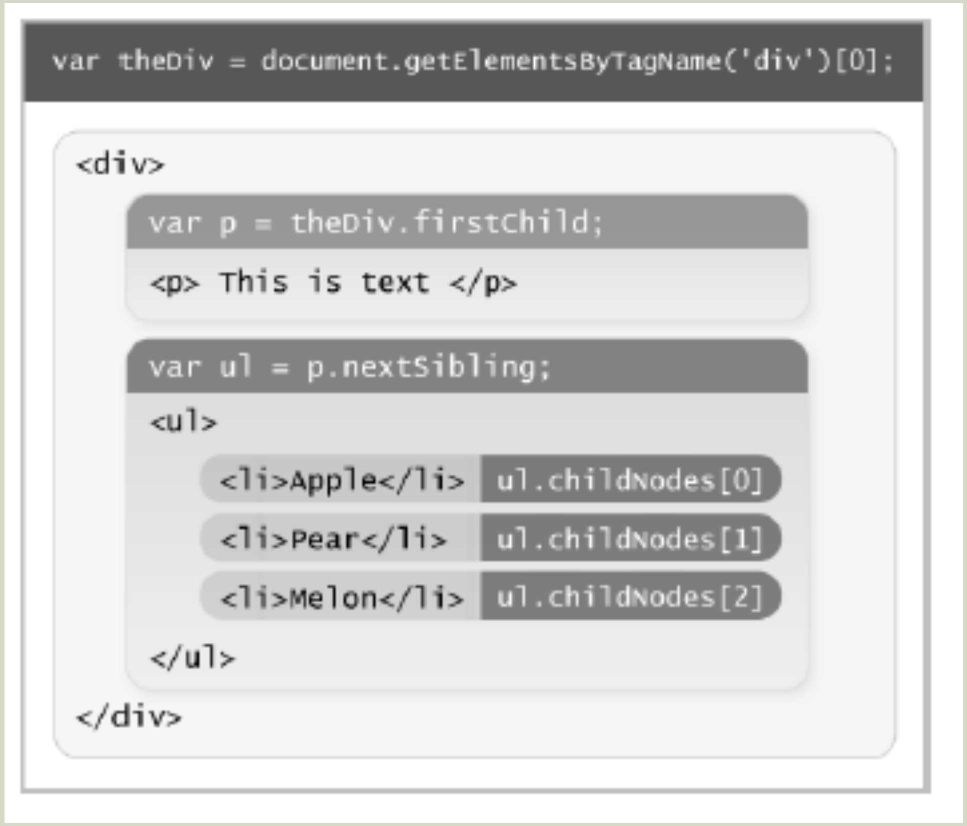
5 – SURFING IN THE DOM

- Node Relationships
  - All nodes in a document have relationships to other nodes.
  - There relationships are described in terms of traditional family.
  - Body element is a child of the html element. head is a sibling of the body element.
  - Each node, has child nodes.
  - Node has some important properies:
    - parentNode.
    - childNodes.
    - firstchild, lastchild
    - nextSigbling,previousSibling
    - nodeType
    - nodeValue
    - nodeName

5 – SURFING IN THE DOM



5 – SURFING IN THE DOM



```
var theDiv = document.getElementsByTagName('div')[0];

<div>
    var p = theDiv.firstChild;
    <p> This is text </p>

    var ul = p.nextSibling;
    <ul>
        <li>Apple</li>  ul.childNodes[0]
        <li>Pear</li>   ul.childNodes[1]
        <li>Melon</li>  ul.childNodes[2]
    </ul>
</div>
```

6 – CHANGING THE DOCUMENT TREE

- Accessing and dynamically changing CSS styles
  - Typical <link> element:

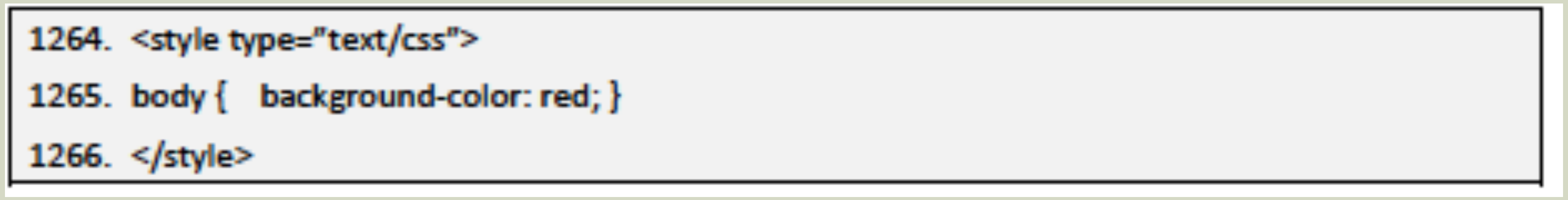
```
<link rel="stylesheet" type="text/css" href="styles.css">
```

  - It can be easily created using the following DOM code:

```
var link = document.createElement("link");
link.rel = "stylesheet";
link.type = "text/css";
link.href = "styles.css";
var head = document.getElementsByTagName("head")[0];
head.appendChild(link);
```

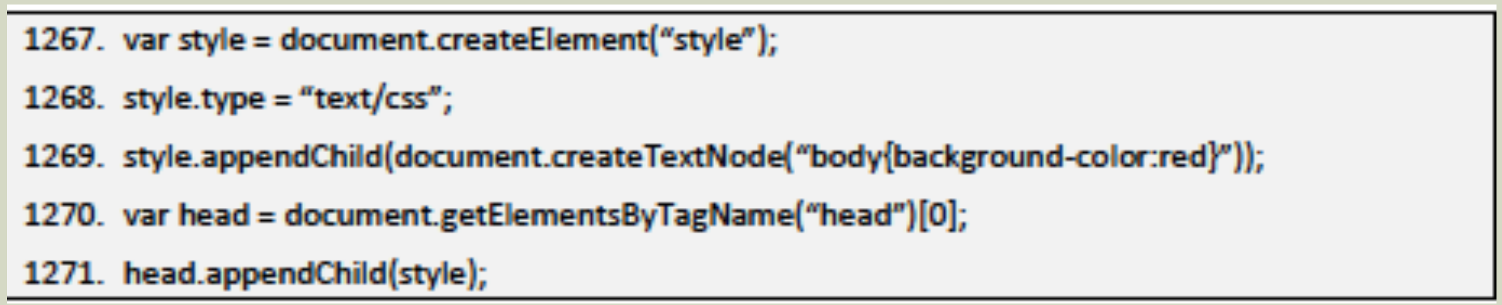
6 – CHANGING THE DOCUMENT TREE

- Accessing and dynamically changing CSS styles
  - Using the <style> and including inline CSS:



```
<style type="text/css">
body {    background-color: red; }
</style>
```

  - Using DOM code:



```
var style = document.createElement(“style”);
style.type = “text/css”;
style.appendChild(document.createTextNode(“body{background-color:red}”));
var head = document.getElementsByTagName(“head”)[0];
head.appendChild(style);
```

6 – CHANGING THE DOCUMENT TREE

- Inserting and removing nodes
  - As the last child → someNode.appendChild(someNode)ç
  - In a specific location → someNode.insertBefore(node, position);
  - Replacing node → someNode.replaceChild(newNode, oldNode);