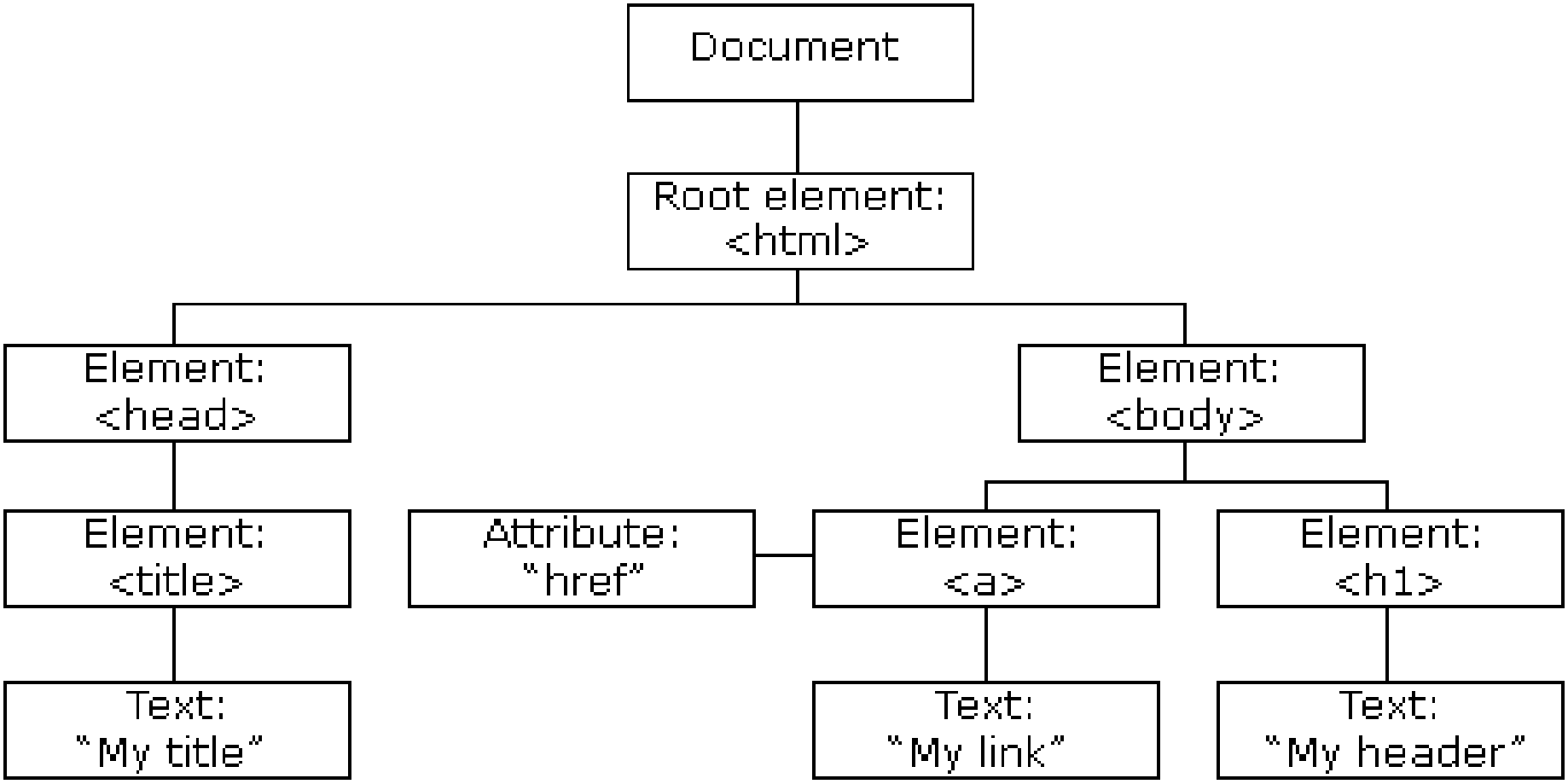
The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM

- The DOM is a W3C (World Wide Web Consortium) standard. The DOM defines a standard for accessing documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- **HTML DOM** - standard model for HTML documents

- **What is the HTML DOM?**
- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

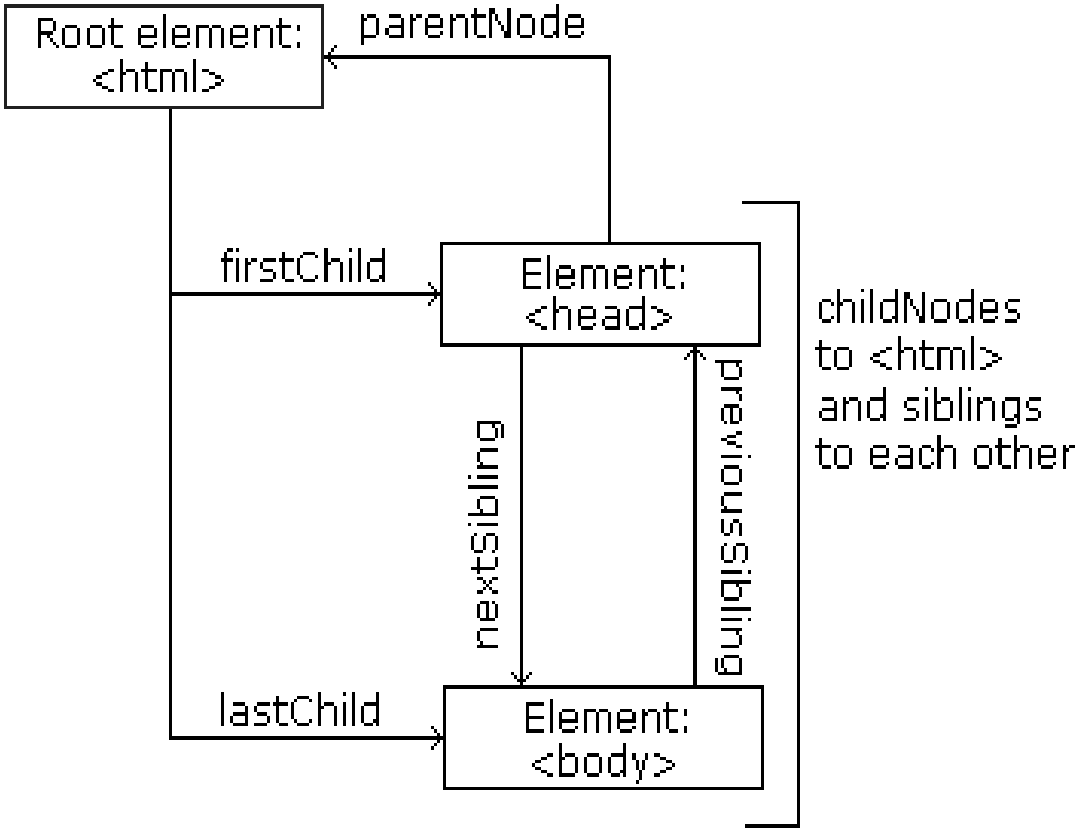
- **HTML DOM Nodes:** In the HTML DOM (Document Object Model), everything is a **node**:
- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

Node Relationships: The nodes in the node tree have a hierarchical relationship to each other.

- The terms parent, child, and sibling are used to describe the relationships.
- In a node tree, the top node is called the root (or root node)
- Every node has exactly one parent, except the root (which has no parent)
- A node can have a number of children
- Siblings (brothers or sisters) are nodes with the same parent

```
<html>
<head>
<title>DOM Tutorial</title>
</head>
<body>
<h1>DOM Lesson one</h1>
<p>Hello world!</p>
</body>
</html>
```

<html> is the root node
<html> has no parents
<html> is the parent of <head> and <body>
<head> is the first child of <html>
<body> is the last child of <html>
and:
<head> has one child: <title>
<title> has one child (a text node): "DOM Tutorial"
<body> has two children: <h1> and <p>
<h1> has one child: "DOM Lesson one"
<p> has one child: "Hello world!"
<h1> and <p> are siblings


Root element:
<html>
parentNode
firstChild
Element:
<head>
childNodes
to <html>
and siblings
to each other
nextSibling
previousSibling
lastChild
Element:
<body>

Example

WAP to create <H1> to display Hello World using the concept of DOM.

Solution:

- HTML elements often consists of both an element node *and* a text node.
- To create a header (e.g. <h1>), you must create both an <h1> element *and* a text node

This would require 3 steps as:

1. createElement()

it is used to create an Element Node with the specified name.

2. The **createTextNode()** method creates a Text Node with the specified text.

3. After the Text Node is created, use the *element*.appendChild() method to append it to an element.

code:

```
<html>
<body>

<script>

    var h = document.createElement("H1");
    var t = document.createTextNode("Hello World");
    h.appendChild(t);
    document.body.appendChild(h);
</script>

</body>
</html>
```

Creates a new <h1> element

To add some text to <h1>

Append <h1> to <body>

output

Hello World

HTML DOM parentNode Property

parentNode Property
returns the parent node of the specified node, as a Node object.

nodeName property returns the name of the specified node.

Example:

Question:

Example list:

- Coffee
- Tea

Click on a button to get the node name of the parent node of the li element in the list.

Solution:

```
<html><body>

<p>Example list:</p>

<ul>
  <li id="myLI">Coffee</li>
  <li>Tea</li>
</ul>

<p>Click the button to get the node name of the parent node of the li element in the list.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    var x = document.getElementById("myLI").parentNode.nodeName;
    document.getElementById("demo").innerHTML = x;
}
</script></body>
```

output

Example list:

- Coffee
- Tea

Click the button to get the node name of the parent node of the li element in the list.

Try it

UL

nextSibling Property

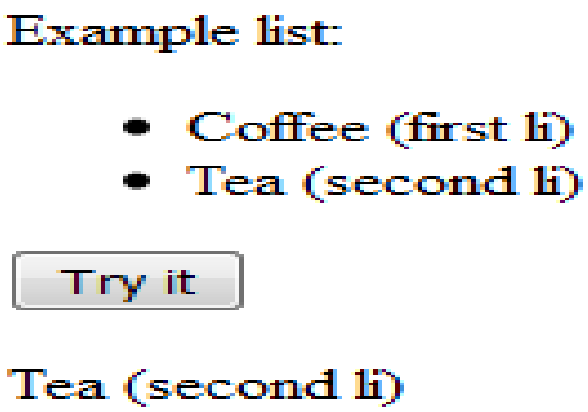
- returns the node immediately following the specified node, in the same tree level
- **Syntax**

node.nextSibling

example

```
<html><body>

<p>Example list:</p>

<ul><li id="item1">Coffee (first li)</li><li id="item2">Tea (second li)</li></ul>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var x = document.getElementById("item1").nextSibling.innerHTML;
   document.getElementById("demo").innerHTML = x;
}
</script></body></html>
```

output


Example list:
Coffee (first li)
Tea (second li)
Try it
Tea (second li)

HTML DOM previousSibling Property

- returns the previous node of the specified node, in the same tree level.
- The returned node is returned as a Node object.

Example: previousSibling

```
<html><body>

<p>Example list:</p>

<ul><li id="item1">Coffee (first li)</li><li id="item2">Tea (second li)</li></ul>


<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var x = document.getElementById("item2").previousSibling.innerHTML;
   document.getElementById("demo").innerHTML = x;
}
</script></body></html>
```

output

Example list:

- Coffee (first li)
- Tea (second li)

Try it

Coffee (first li)

HTML DOM childNodes Property

- The childNodes property returns a collection of a node's child nodes, as a NodeList object.
- The nodes in the collection are sorted as they appear in the source code and can be accessed by index numbers. The index starts at 0.

Syntax

element.childNodes

Example

```
<html>
<body><!-- This is a comment node! -->

<p>Click the button get info about the body element's child nodes.</p>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> Whitespace inside elements is considered as text, and text
is considered as nodes. Comments are also considered as nodes.</p>

<p id="demo"></p>

<script>
function myFunction() {
   var c = document.body.childNodes;
   var txt = "";
   var i;
   for (i = 0; i < c.length; i++) {
      txt = txt + c[i].nodeName + "<br>";
   }

   document.getElementById("demo").innerHTML = txt;
}
</script></body></html>
```

output

Click the button get info about the body element's child nodes.

Try it

Note: Whitespace inside elements is considered as text, and text is considered as nodes. Comments are also considered as nodes.

#comment
#text
P
#text
BUTTON
#text
P
#text
P
#text
SCRIPT
#text

HTML DOM firstChild Property

- The firstChild property returns the first child node of the specified node, as a Node object.

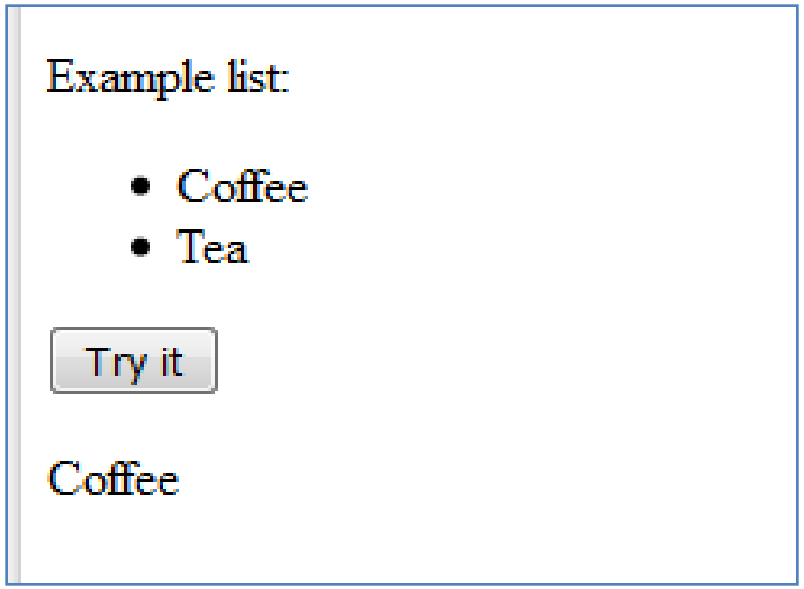
Syntax

node.firstChild

example

```
<html>
<body>

<p>Example list:</p>
<ul id="myList"><li>Coffee</li><li>Tea</li></ul>


<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var list = document.getElementById("myList").firstChild.innerHTML;
   document.getElementById("demo").innerHTML = list;
}
</script>

</body>
</html>
```

output



HTML DOM lastChild Property

```
<html><body>

<p>Example list:</p>
<ul id="myList"><li>Coffee</li><li>Tea</li></ul>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var list = document.getElementById("myList").lastChild.innerHTML;
   document.getElementById("demo").innerHTML = list;
}
</script></body></html>
```

output

Example list:

- Coffee
- Tea

Try it

Tea

HTML DOM createElement() Method

The createElement() method creates an Element Node with the specified name.

Question:

Create a button, and if you click on this button, create another button.. And so on

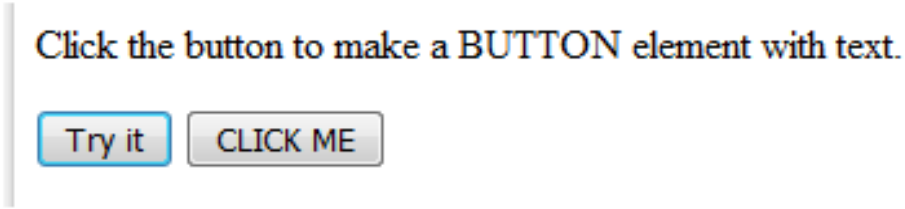
```
<html>
<body>

<p>Click the button to make a BUTTON element with text.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    var btn = document.createElement("BUTTON");
    var t = document.createTextNode("CLICK ME");
    btn.appendChild(t);
    document.body.appendChild(btn);
}
</script>

</body>
</html>
```

output

Click the button to make a BUTTON element with text.

Try it

when you click on this button, output will be:



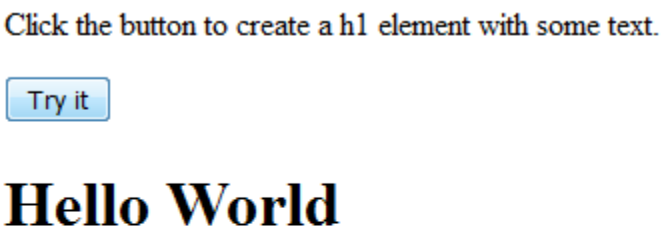
Another example:

HTML elements often consists of both an element node *and* a text node.

To create a header (e.g. <h1>), you must create both an <h1> element *and* a text node:

```
<html>
<body>

<p>Click the button to create a h1 element with some text.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   var h = document.createElement("H1");
   var t = document.createTextNode("Hello World");
   h.appendChild(t);
   document.body.appendChild(h);
}
</script>

</body>
</html>
```

output


Click the button to create a h1 element with some text.
Try it
Hello World

- **The nodeType Property**: The nodeType property returns the type of node. nodeType is read only.

The most important node types are:

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

- **HTML DOM nodeType Property:** The nodeType property returns the node type, as a number, of the specified node.
- If the node is an element node, the nodeType property will return 1.
- If the node is an attribute node, the nodeType property will return 2.
- If the node is a text node, the nodeType property will return 3.
- If the node is a comment node, the nodeType property will return 8.This property is read-only.

Get the node type of the element:

```
<html> <body>
<p id="myP">Click the button to get the node type of this
    element.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
 {
   var x = document.getElementById("myP").nodeType;

   document.getElementById("demo").innerHTML = x;
}
</script></body>
</html>
```

Output: 1

nodeValue Property

- The nodeValue property sets or returns the node value of the specified node.
- If the node is an element node, the nodeValue property will return null.
- If you want to return the text of an element, remember that text is always inside a *Text node*, and you will have to return the *Text node's* node value **(*element*.childNodes[0].nodeValue**).
- For other node types, the nodeValue property will return different values for different node types.
- **Syntax**

Return the node value:

node.nodeValue

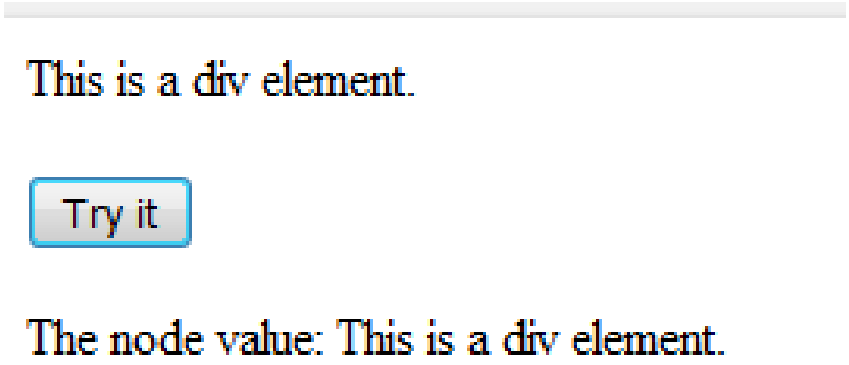
Set the node value:

- *node*.nodeValue=*value*

WAP to get the node value of the element's first child node

```
<html>
<body>


<div id="myDIV">This is a div element.</div>
<br>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var x = document.getElementById("myDIV").firstChild;

   var txt = "The node value: " + x.nodeValue + "<br>";

   document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

output


This is a div element.
Try it
The node value: This is a div element.

DOM nodeName Property

- The nodeName property returns the name of the specified node.
- If the node is an element node, the nodeName property will return the tag name.
- If the node is an attribute node, the nodeName property will return the name of the attribute.
- For other node types, the nodeName property will return different names for different node types.

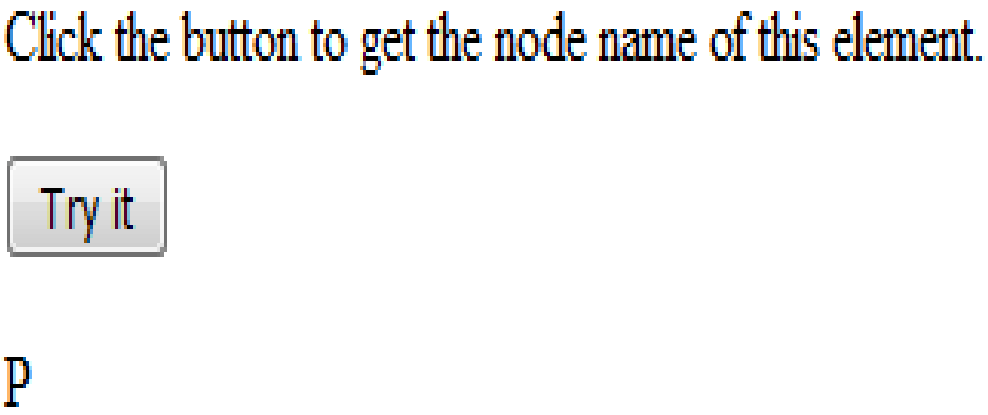
Possible values:

- Returns the tagname for element nodes, in uppercase
- Returns the name of the attribute for attribute nodes
- Returns "#text" for text nodes
- Returns "#comment" for comment nodes
- Returns "#document" for document nodes

example

```
<html>
<body>

<p id="myP">Click the button to get the node name of this element.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
   var x = document.getElementById("myP").nodeName;
   document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

output



Child Nodes and Node Values

- In addition to the innerHTML property, you can also use the childNodes and nodeValue properties to get the content of an element.

Syntax

element.childNodes

Question: WAP that collects the node value of an <h1> element and copies it into a <p> element:

Solution

```
<html> <body>
<h1 id="intro">My First Page</h1>
<p id="demo">Hello World!</p>
<script>
var myText = document.getElementById("intro").childNodes[0].nodeValue;

document.getElementById("demo").innerHTML = myText;

</script></body> </html>
```

Output:

My First Page

My First Page

**Using the firstChild property is the same as using childNodes[0]

Example:

```
<html> <body>
<h1 id="intro">My First Page</h1>
<p id="demo">Hello World!</p>
<script>
myText =
   document.getElementById("intro").firstChild.nodeValue;

document.getElementById("demo").innerHTML = myText;
</script></body></html>
```

Output:

My First Page

My First Page

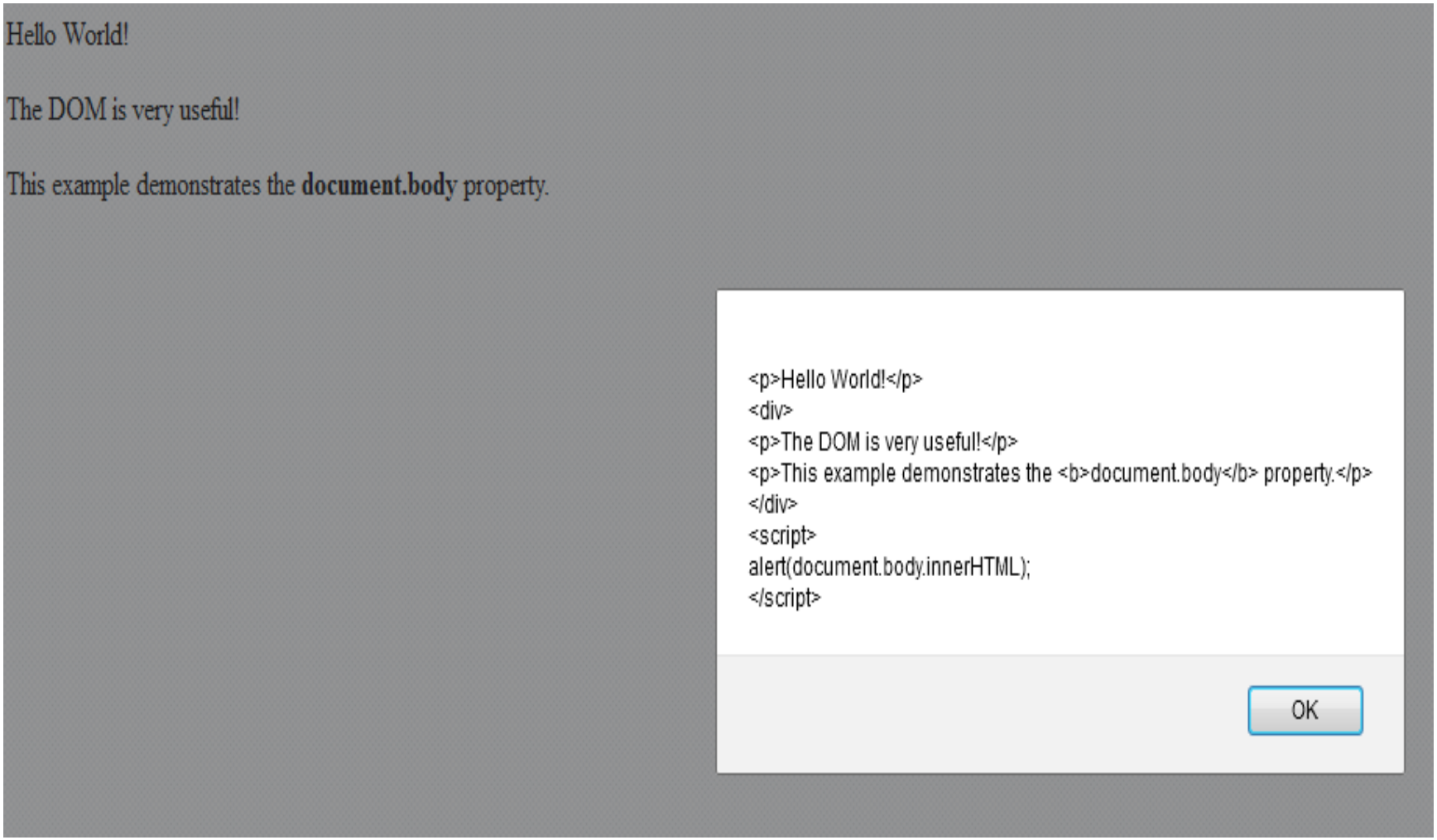
- **DOM Root Nodes**

There are two special properties that allow access to the full document:

document.body - The body of the document
document.documentElement - The full document

```
<html>  <body>
<p>Hello World!</p>
<div>
<p>The DOM is very useful!</p>
<p>This example demonstrates the <b>document.body</b>
    property.</p>
</div>
<script>
alert(document.body.innerHTML);
</script></body></html>
```

OUTPUT: will display body of document



alert(document.documentElement.innerHTML): displays full document

appendChild:

Question :WAP to create a div tag with 2 paragraphs. Now append a 3rd paragraph at the end of this document

Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<html> <body>
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para);
</script> </body> </html>
```

Output:

This is a paragraph.
This is another paragraph.
This is new.

* **The explanation of this code is provided in next slide**

Example Explained

- This code creates a new <p> element:

  var para = document.**createElemen**t("p");

- To add text to the <p> element, you must create a text node first. This code creates a text node:

  var node = document**.createTextNode**("This is a new paragraph.");

- Then you must append the text node to the <p> element:

  para**.appendChild(**node);

Finally you must append the new element to an existing element.

- This code finds an existing element: firstly, access the parent tag i.e. div

var element = document.getElementById("div1");

- This code appends the new element to the existing element:

element.**appendChild**(para);

insertBefore()

insertBefore():

- The insertBefore() method inserts a node as a child, right before an existing child, which you specify.

But the appendChild() method in the previous example, appended the new element as the last child of the parent.

Syntax

node.insertBefore(*newnode, existingnode*)

insertBefore()

If you want to insert a new element at some desired position, then you can use the insertBefore() method:

```
<html><body>
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var element = document.getElementById("div1");
var child = document.getElementById("p1");

element.insertBefore(para,child);
</script></body></html>
```

Output:

This is new.
This is a paragraph.
This is another paragraph.

- **removeChild: Removing Existing HTML Elements.**

Syntax: node.removeChild(node)

Q. Write a program to remove the first paragraph of a document

Solution:

To remove an HTML element, you must know the parent of the element:

```
<html><body>
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script></body></html>
```

Output:

This is another paragraph.

replaceChild() Method

- The replaceChild() method replaces a child node with a new node.
- The new node could be an existing node in the document, or you can create a new node.

Syntax

node.replaceChild(*newnode*,*oldnode*)

Where

newnode: The node object you want to insert

***Oldnode*:** The node object you want to remove

Practice Question 1

Create a list:

- Coffee
- Tea
- Milk

Then create a button. If user clicks the button, the first element of the list should be replaced by “water”. i.e. the new list should appear as:

- Water
- Tea
- Milk

Solution

**here // represents comments

```
<html><body>

<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
   var textnode = document.createTextNode("Water"); // Create a new text node called
    "Water"

   var item = document.getElementById("myList").childNodes[0]; // Get the first child
    node of an <ul> element

   item.replaceChild(textnode, item.childNodes[0]); // Replace the first child node of
    <ul> with the newly created text node
}
</script></body></html>
```

output

- Coffee
- Tea
- Milk

Try it

After clicking on this button: new list will be

- Water
- Tea
- Milk

Practice question 2

Write a program to remove first list element using removeChild() Method

The list before removing is:

- Coffee
- Tea
- Milk

The list after removing is:

- Tea
- Milk

solution

```
<html><body>

<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
   var list = document.getElementById("myList"); // Get the <ul> element with id="myList"

   list.removeChild(list.childNodes[0]); // Remove <ul>'s first child node (index 0)
}
</script></body></html>
```

Practice ques. 3

Using appendChild() Method, modify the following list:

- Coffee
- Tea

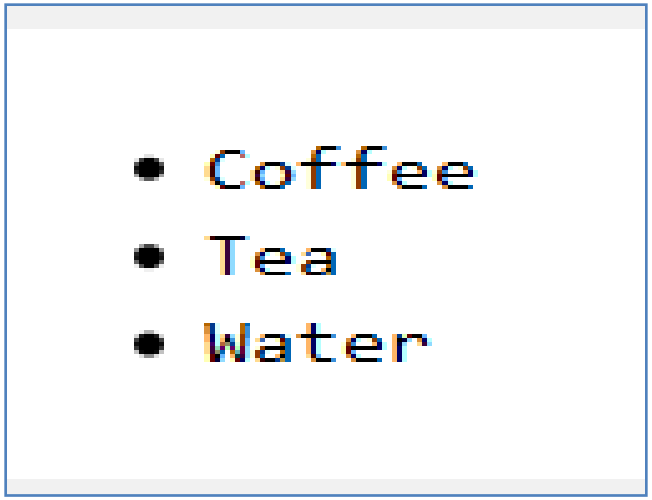
Then Click a button,

The new list must appear as:

- Coffee
- Tea
- Water

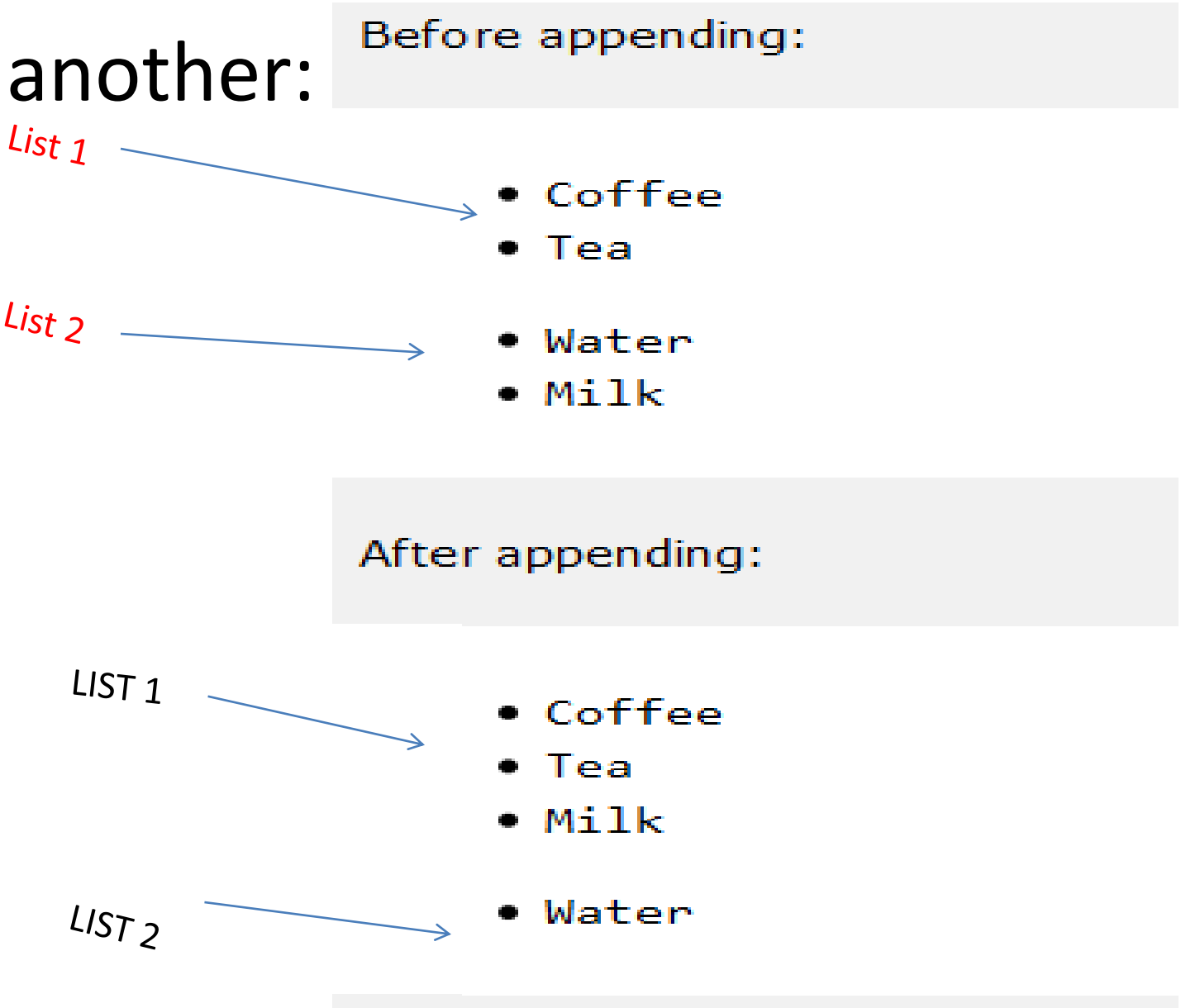
```
<html><body>
<ul id="myList">  <li>Coffee</li>  <li>Tea</li></ul>

<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
   var node = document.createElement("LI"); // Create a <li> node

   var textnode = document.createTextNode("Water"); // Create a text node

   node.appendChild(textnode); // Append the text to <li>

   document.getElementById("myList").appendChild(node); // Append <li> to
    <ul> with id="myList"
}
</script></body></html>
```

output



Practice question 4:

WAP to Move a list item from one list to another:



Solution:

```
<html><body>

<ul id="myList1"><li>Coffee</li><li>Tea</li></ul>

<ul id="myList2"><li>Water</li><li>Milk</li></ul>

<p>Click the button to move an item from one list to another.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   var node = document.getElementById("myList2").lastChild; //picking the
    last item of mylist2 i.e. list item named “milk”

   document.getElementById("myList1").appendChild(node);// appending the
    list item named “milk” it as the last list item of mylist1
}

</script></body> </html>
```