COMP 4021
Internet Computing

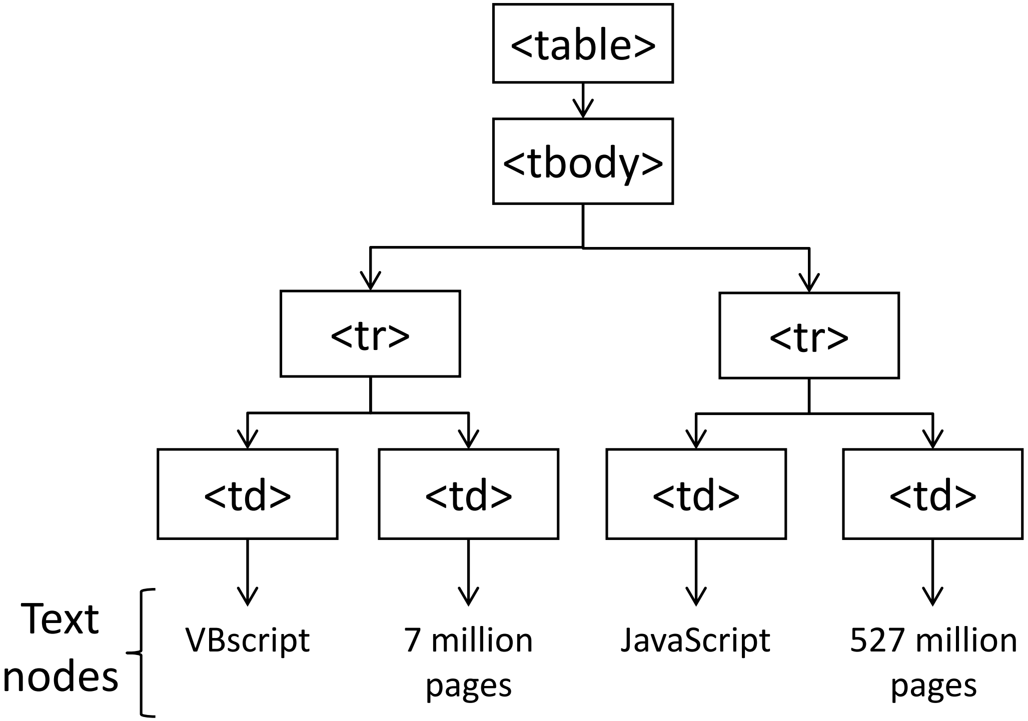
Document Object Model (DOM)

David Rossiter

This Presentation

- This presentation considers the following:
  - Simple DOM example
  - DOM representation
  - Flash DOM
  - Using relations to traverse the tree – examples
  - Referring to nodes - three methods

Simple DOM Example



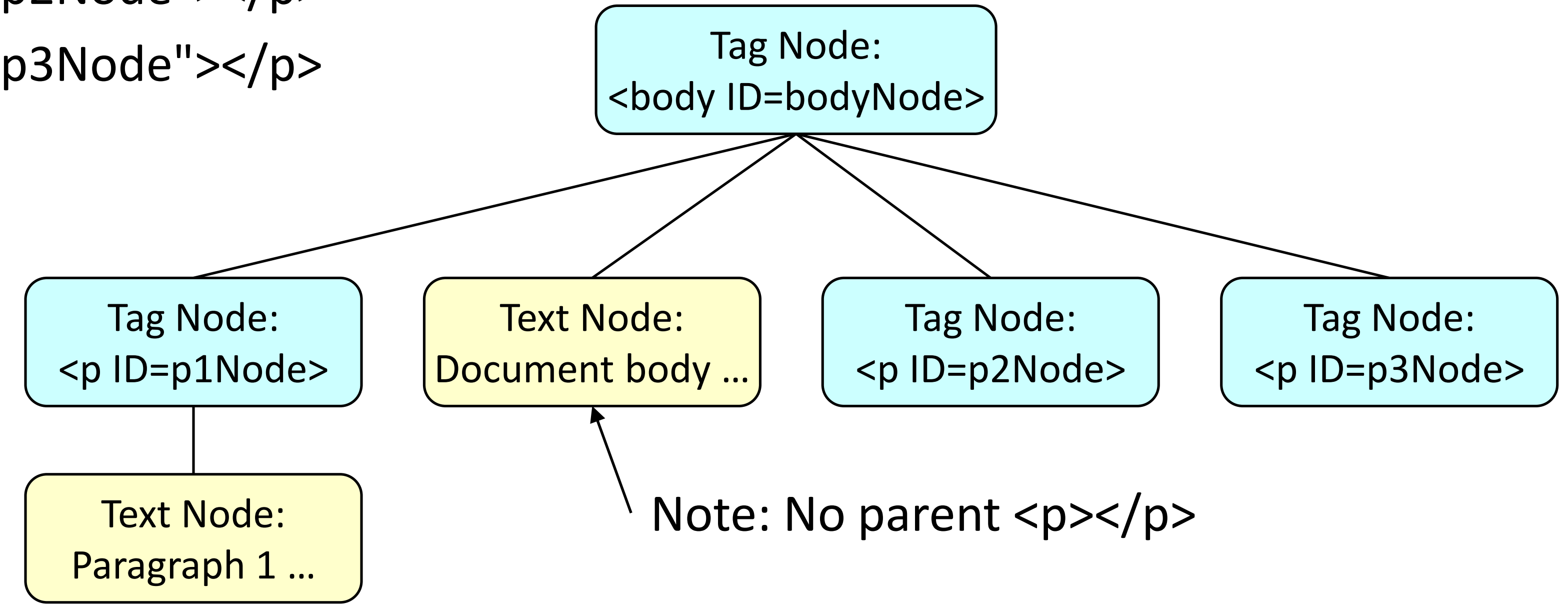
```
<table>
    <tbody>
    <tr>
        <td>VBscript</td>
        <td>7 million pages</td>
    </tr>
    <tr>
        <td>JavaScript</td>
        <td>527 million pages</td>
    </tr>
    </tbody>
</table>
```

The DOM Standard

- Scripting languages (not only JavaScript) can access any part of the DOM including relationships (parent/sibling, etc.)
- You can actively alter, create and destroy *any* part of the DOM structure, at *any* time
- The same code will work for all browsers, e.g., IE, Firefox and Opera without any changes
- The same techniques can also be used in lots of other languages i.e. Java, C++, PHP, etc.
- Flash also has its own DOM but it is not the same as the W3C DOM standard

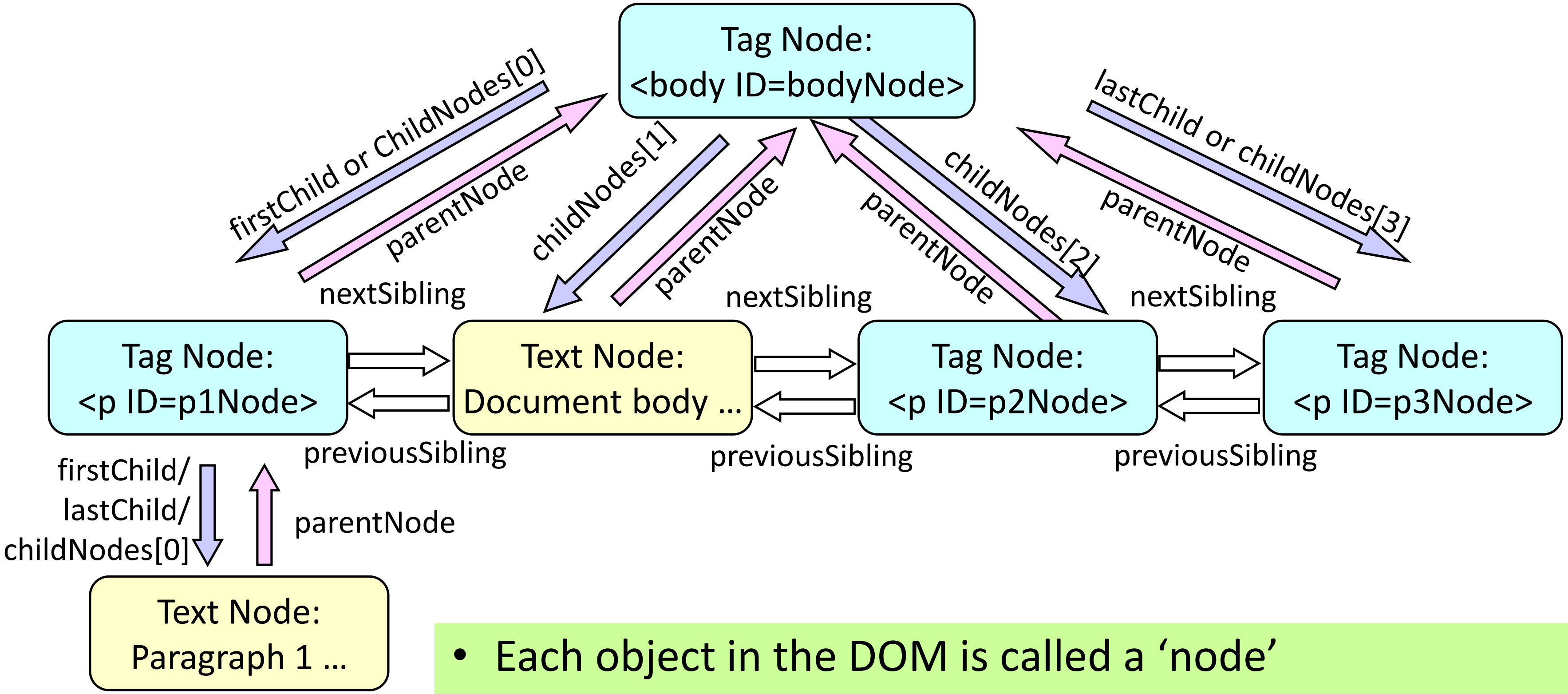
Detailed DOM Example

```
<body id="bodyNode">
<p id = "p1Node">Paragraph 1 …</p>
Document body …
<p id = "p2Node"></p>
<p id = "p3Node"></p>
</body
```



Note: No parent <p></p>

Detailed DOM Example



- Each object in the DOM is called a ‘node’
- Both nodes and relationships between nodes are shown
- Any node can be given a name (the ID attribute) for reference by other nodes and scripts

Using Node Relations

- Scripts can access all of these relations between nodes:
  - parentNode
  - childNodes[], firstChild, lastChild
  - previousSibling, nextSibling
  - and more...

- There is more than one way to write some things i.e. childNodes[0] is the same as firstChild
- childNodes.length returns the number of child nodes
  - So childNodes[childNodes.length-1]equals lastChild

Using Relations to Traverse the Tree - 1

- The code below starts with any node ‘node’ in the DOM, and then traverses up the branches, each time adding the name of the parent to a string, until the root is reached
- The result is to create a string which contains the path from the root to the starting ‘node’,
  - e.g., #document->HTML->BODY->UL->LI->A

```
function click()   {
      var node=this;  // this = current object

      tree=node.nodeName;
      while (node.parentNode) {
              node = node.parentNode;
              tree = node.nodeName + " -> " + tree;    }
       alert(tree);     }
```

Using Relations to Traverse the Tree - 2

- This example is more advanced, using recursion
- It shows how code can be written to access every single element in the DOM (i.e., everything in the web page)
- It goes to every node and instructs that when an *onmouseover* event occurs to that node, the function *do_someth* will be executed
    - The exact purpose of the *do_someth* is not important for this demo; it could be as simple as changing the colour of the node to red

Using Relations to Traverse the Tree - 2

```
function processChildren(node) {
    var currentNode = node.firstChild;   // start with the first child
    do {
        currentNode.onmouseover = do_someth;    // do something with node
        if (currentNode.hasChildNodes) {            // if node has children
            processChildren(currentNode);    }      // process them (recursive)

    currentNode = currentNode.nextSibling;       // move to the next sibling

    } while (currentNode != node.lastChild       // repeat until last child
            && currentNode != null)              // or until nothing more    }
```

- Traversal of the entire DOM can be done in different ways
- Upon reaching a node, attach an event handler **do_someth** (function not shown, e.g., change the background colour of the node)

How to Locate One Particular Thing?

- Method 1: Use the exact DOM path
  - May be hard to work out the exact position
  - Easy to make mistakes
  - Load into another browser – DOM may be a bit different, not work!
- Method 2: Use getElementsByTagName()
  - Require you to know the exact tag name (I.e. is it h2 or h3?)
  - Also, there might be several nodes of that type, so you have to know exactly which one it is (I.e. first one? second one?)
- Method 3: Use getElementById()
  - If you give the nodes unique names then this method is the easiest to refer to them

Methods 1, 2, 3 - Examples

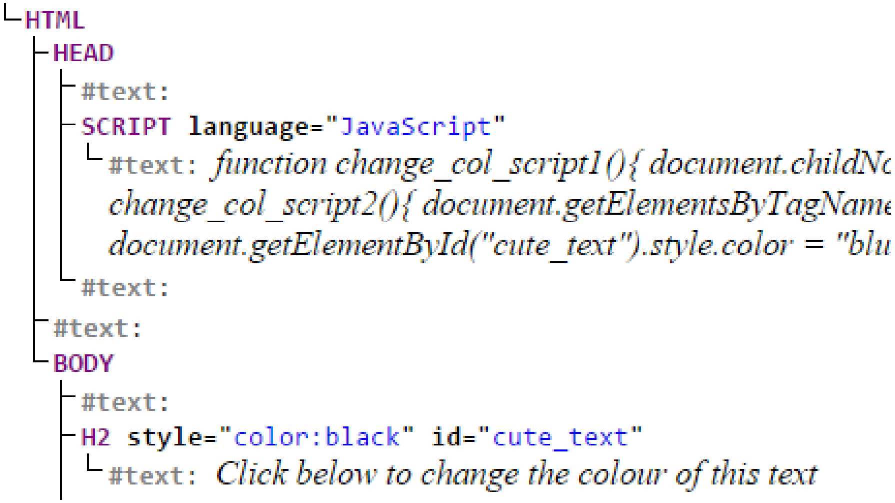
Address DOM by absolute path; why doesn’t it work? Check DOM examples1

```
<html>  <head>  <script language="JavaScript">
    function change_col_script1() {
        document.childNodes[0].childNodes[1].childNodes[0].style.color="red";     }
    function change_col_script2() {
        document.getElementsByTagName("h2")[0].style.color = "yellow";      }
    function change_col_script3() {
        document.getElementById("cute_text").style.color = "blue";      }
</script>   </head>
<body>
<h2 id="cute_text">
Click below to change the colour of this text
</h2>
<form>
    <input onclick="change_col_script1()" type="button" value="Change using method 1">
    <input onclick="change_col_script2()" type="button" value="Change using method 2">
    <input onclick="change_col_script3()" type="button" value="Change using method 3">
</form>    </body>    </html>
```

Why Absolute Addressing does not Work?

```
<html>
<head>  <script language="JavaScript">
   function change_col_script1() {
      document.childNodes[0].childNodes[1].childNodes[0].style.color="red";     }
   … …


</script>
</head>
<body>
<h2 id="cute_text">
Click below to change the colour of this text
</h2>
… …
```



```
└HTML
  ├HEAD
  │ ├#text:
  │ ├SCRIPT language="JavaScript"
  │ │ └#text: function change_col_script1(){ document.childNc
  │ │   change_col_script2(){ document.getElementsByTagName
  │ │   document.getElementById("cute_text").style.color = "blu
  │ └#text:
  ├#text:
  └BODY
    ├#text:
    ├H2 style="color:black" id="cute_text"
    │ └#text: Click below to change the colour of this text
    │
```

Draw the DOM graphically

Outline

- This presentation considers the following:
  - Creating and adding nodes to the DOM
    - HTML example
    - SVG example
  - Deleting nodes in the DOM
    - HTML example
    - SVG example

  - Old style DOM code: document.all

Creating and Adding Nodes to DOM

1. Create a node

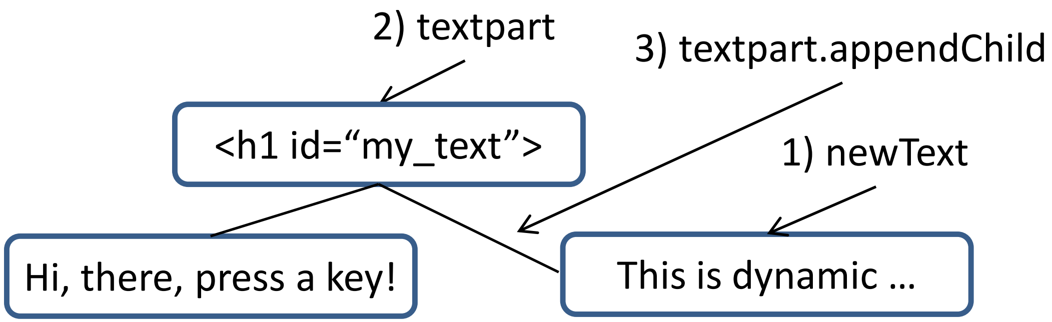
2. Add it to the DOM at an appropriate place

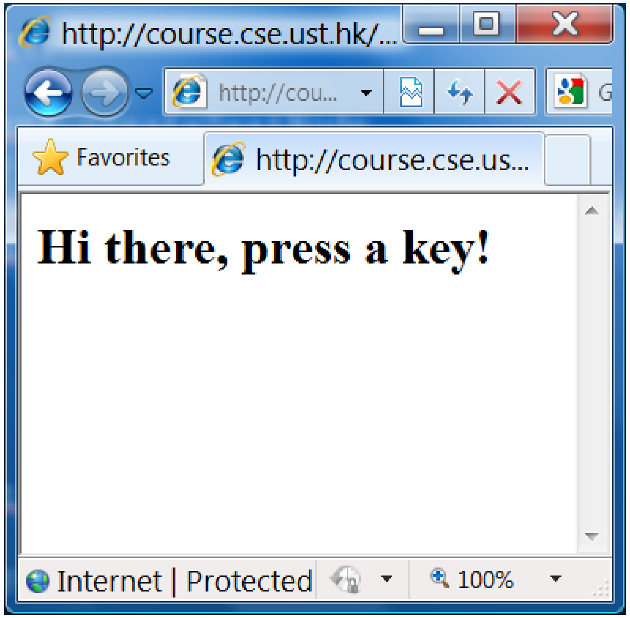
- Right after you created a node (step 1), the node is not actually part of the DOM yet
- You need to attach it to an existing node in the DOM
- For visual languages such as HTML and SVG, you won’t actually see the node until it is added to the DOM

Dynamic HTML Node Creation – Example

```
<html> <head> <script type="text/javascript">
function insert_new_text()    {
    var newText = document.createTextNode("This is dynamically added text!");
    var textpart = document.getElementById("my_text");
    textpart.appendChild(newText) ;    }      </script>     </head>

<body onkeypress="insert_new_text()">
<h1 id="my_text" >Hi there, press a key! </h1>
</body>
```





Hi there, press a key!

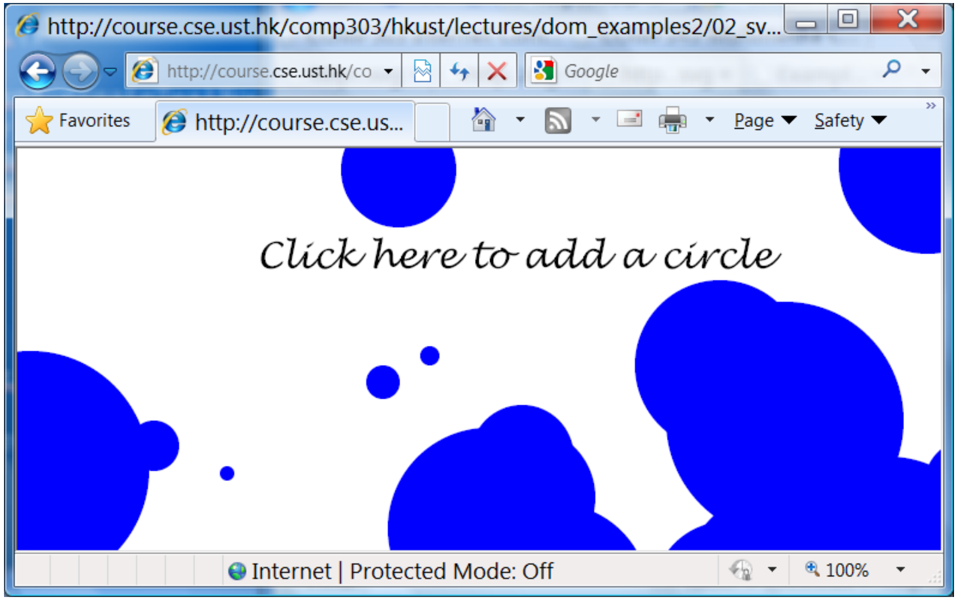
Dynamic Node Creation – SVG Example 1/2

- Dynamic creation of an SVG node

```
<svg width="1000" height="800" onclick="insert_a_circle(evt)" >

<text x="200" y="100" style="font-size:30px;font-family:Lucida Handwriting">
    Click here to add a circle
</text>
```

Example display after many clicks



http://course.cse.ust.hk/comp303/hkust/lectures/dom_examples2/02_sv...
http://course.cse.ust.hk/co
Google
Favorites
http://course.cse.us...
Page
Safety
Click here to add a circle
Internet | Protected Mode: Off
100%

Dynamic Node Creation – SVG Example 2/2

```
<script type="text/javascript">
var SVGDocument = null,   SVGRoot = null;

function insert_a_circle(event)    {
   SVGDocument = event.target.ownerDocument;
   SVGRoot = SVGDocument.documentElement;

   var newnode=SVGDocument.createElementNS(
            "http://www.w3.org/2000/svg","circle");
   var cx=Math.floor(Math.random() * 1000);
   var cy=Math.floor(Math.random() * 800);
   var r=Math.floor(Math.random() * 100);
   newnode.setAttribute('cx', cx);     newnode.setAttribute('cy', cy);
   newnode.setAttribute('r', r);       newnode.setAttribute('fill', "blue");

   SVGRoot.appendChild(newnode);    }    </script>
```

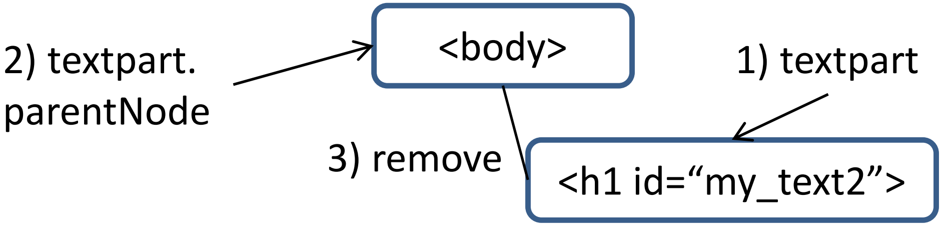
Deleting Nodes

- To delete a node in the DOM, you cannot simply point to a node and say ‘delete this’
- Instead, you have to ask the parent node to delete that child node
- The parent node may have many children, so you have to specify exactly which child you want the parent to delete

Dynamic Node Deletion – HTML Node

```
function delete_text()    {
   var textpart = document.getElementById("my_text2");
   textpart.parentNode.removeChild(textpart);    }
```

```
<body onkeypress="delete_text() ">
<h1 id="my_text1">Paragraph 1</h1>
<h1 id="my_text2">Paragraph 2</h1>
<h1 id="my_text3">Paragraph 3</h1>
<h1 id="my_text4">Paragraph 4</h1>
</body>
```



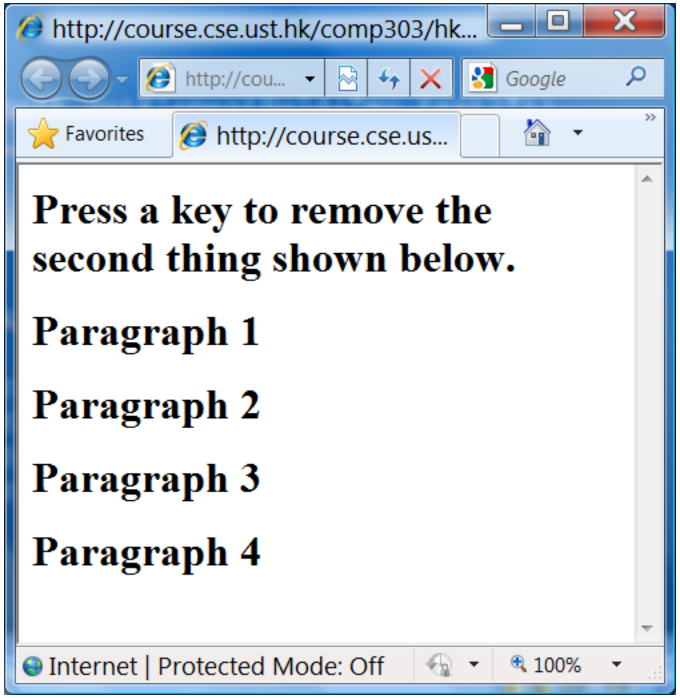
2) textpart. parentNode

<body>

1) textpart

3) remove

<h1 id=“my_text2”>



http://course.cse.ust.hk/comp303/hk...
http://cou...
Google
Favorites
http://course.cse.us...
Internet | Protected Mode: Off
100%

Press a key to remove the second thing shown below.

Paragraph 1

Paragraph 2

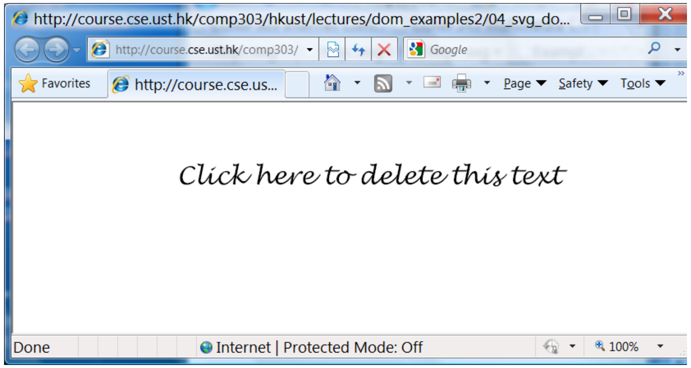
Paragraph 3

Paragraph 4

- Always deletes the 2nd paragraph; change it to delete the paragraph clicked

Dynamic Node Deletion – SVG Node

```
<svg width="1000" height="800“  onclick="delete_text(evt)">
<script type="text/javascript">
var SVGDocument = null, SVGRoot = null;
var node = null;
```



Click here to delete this text

http://course.cse.ust.hk/comp303/hkust/lectures/dom_examples2/04_svg_do...
http://course.cse.ust.hk/comp303/
Google
Favorites
http://course.cse.us...
Page
Safety
Tools
Done
Internet | Protected Mode: Off
100%

```
function delete_text(event)     {
    SVGDocument = event.target.ownerDocument;

    node = SVGDocument.getElementById("nice_text");
    if (node) node.parentNode.removeChild(node);     }      </script>

<text id="nice_text" x="200" y="100"
   style="font-size:30px;font-family:Lucida Handwriting">
   Click here to delete this text</text>     </svg>
```

document.all[]

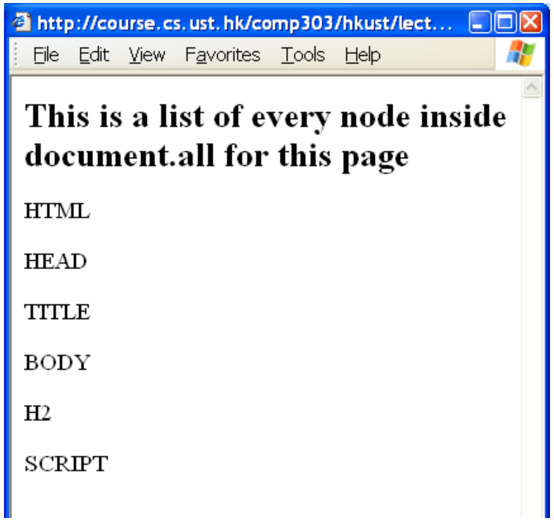
- Another way to access ‘anything’ in the DOM is by using document.all
    - document.all["ugly_paragraph"].style.color="black";
- document.all was created by Microsoft before all the proper DOM existed and is not part of the DOM standard
    - Produce different results in different browsers, and it does not seem to be able to access all nodes in the DOM
    - Please do not use document.all[ ]
- However, the examples in the next few slides do give further insight into how DOM works dynamically

.all[] Example 1

```
<html>  <head><title></title></head>
<body>
<h2>This is a list of every node inside
    document.all for this page</h2>

<script language="JavaScript">
var list="";
for (i = 0; i < document.all.length; i++){
  list = list + "<p>" +
          document.all(i).tagName + "</p>";
}
document.write( list );

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


http://course.cs.ust.hk/comp303/hkust/lect...
File Edit View Favorites Tools Help

This is a list of every node inside document.all for this page

HTML

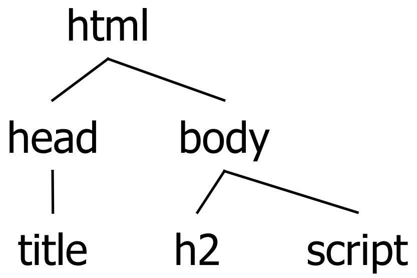
HEAD

TITLE

BODY

H2

SCRIPT



list

<p>HTML</p><p>HEAD</p> .... <p>SCRIPT</p>

.all[] Example 2: List tag properties and values

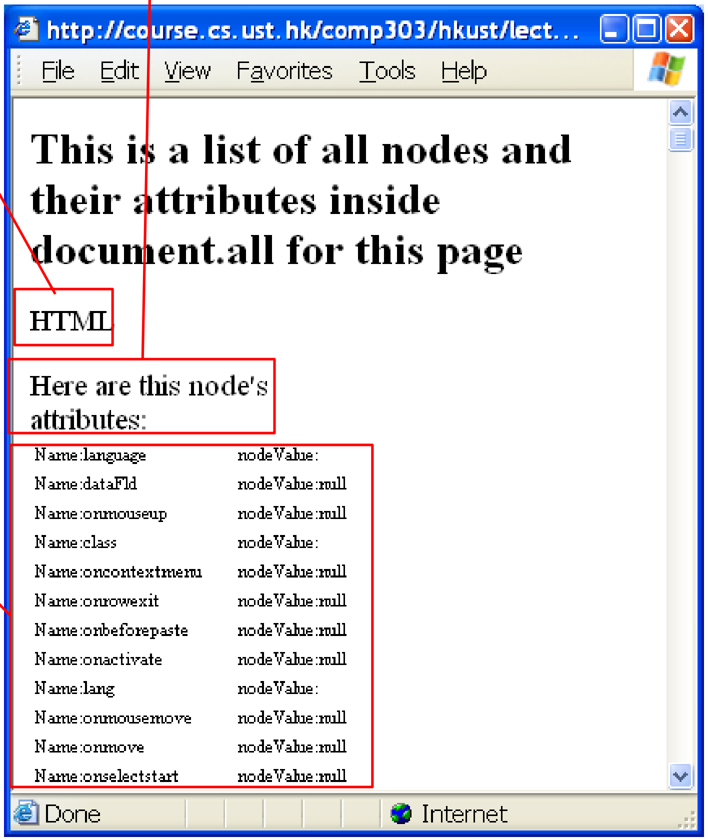
```
for(i = 0; i < document.all.length; i++)  {
   list = list + "<p>" + document.all(i).tagName + "</p>";

   list=list + "<table style='font-size:8pt'><thead>Here are this node's
     attributes:</thead>";

   for (j=0; j< document.all(i).attributes.length; j++)
      list = list + "<tr> <td>Name:" +
      document.all(i).attributes[j].nodeName +
      "</td> <td>nodeValue:" +
      document.all(i).attributes[j].nodeValue +
      "</td> </tr>";      }
   list=list + "</table>";      }
document.write( list );
```

list

```
…<p>HTML</p><table …><thead> …</thead>
<tr><td>Name: Name.language</td>
<td>nodeValue:</td></tr>… …</table>
```



This is a list of all nodes and their attributes inside document.all for this page

HTML

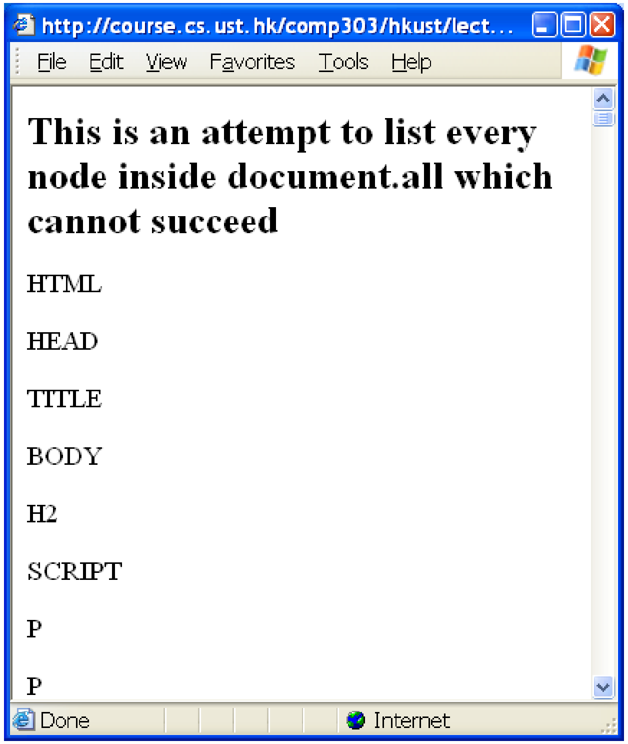
Here are this node's attributes:

```
Name:language          nodeValue:
Name:dataFld           nodeValue:null
Name:onmouseup         nodeValue:null
Name:class             nodeValue:
Name:oncontextmenu     nodeValue:null
Name:onrowexit         nodeValue:null
Name:onbeforepaste     nodeValue:null
Name:onactivate        nodeValue:null
Name:lang              nodeValue:
Name:onmousemove       nodeValue:null
Name:onmove            nodeValue:null
Name:onselectstart     nodeValue:null
```

.all[] Example 3: Infinite DOM

```
<html>  <head></head>
<body>
<h2>This is an attempt to list every  node
inside document.all which cannot succeed</h2>

<script language="JavaScript">
for (i = 0; i < document.all.length; i++) {
 document.write("<p>" +
   document.all(i).tagName + "</p>");
}
</script>  </body>   </html>
```


http://course.cs.ust.hk/comp303/hkust/lect...
File Edit View Favorites Tools Help
Done
Internet

This is an attempt to list every node inside document.all which cannot succeed

HTML

HEAD

TITLE

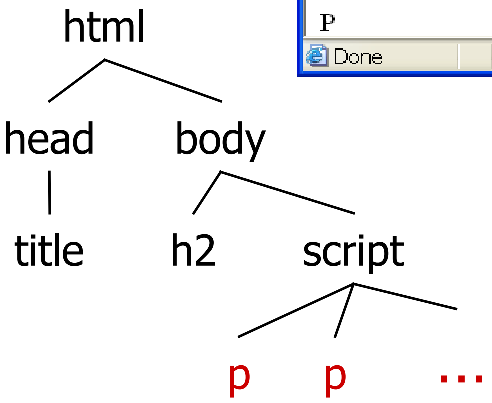
BODY

H2

SCRIPT

P

P



Take Home Message

- DOM captures everything on a webpage, including all the attributes defined for each tag
- Dynamic update to any part of a DOM is supported
  - Insertion and deletion of tag nodes
  - Update to any properties (attribute values)