Client Side Technologies

HTML DOM

ITI – Assiut Branch
Eng. Hany Saad

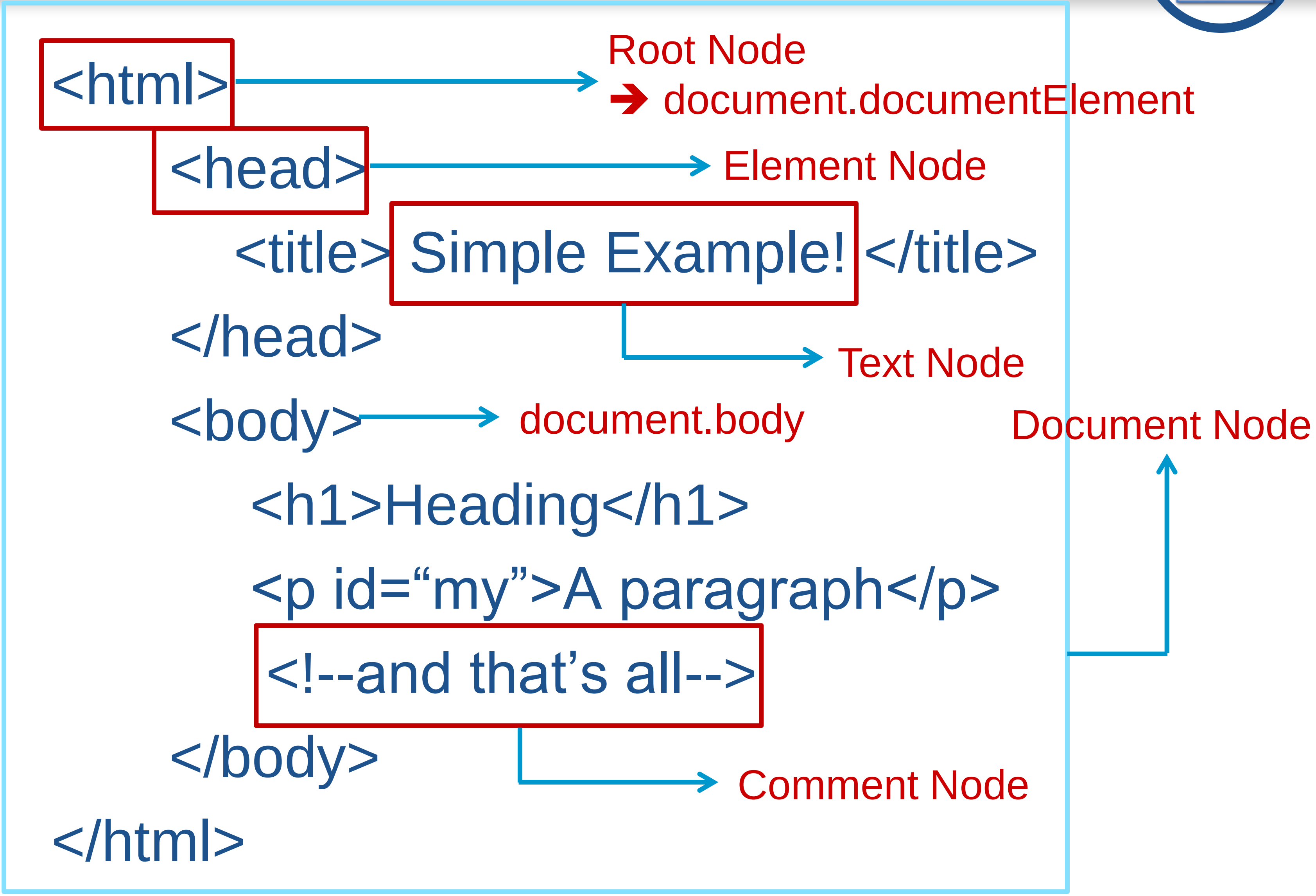
HTML DOM

- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.
- It is a hierarchy of data types for HTML documents, links, forms, comments, and everything else that can be represented in HTML code.
- The general data type for objects in the DOM are *Nodes*. They have *attributes*, and some nodes can contain other nodes.
- There are several node types, which represent more specific data types for HTML elements. Node types are represented by numeric constants.

HTML DOM (Cont.)

- According to the DOM, everything in an HTML document is a node.
- The DOM says:
  - The entire document is a document node
  - Every HTML element is an element node
  - The text in the HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - Comments are comment nodes

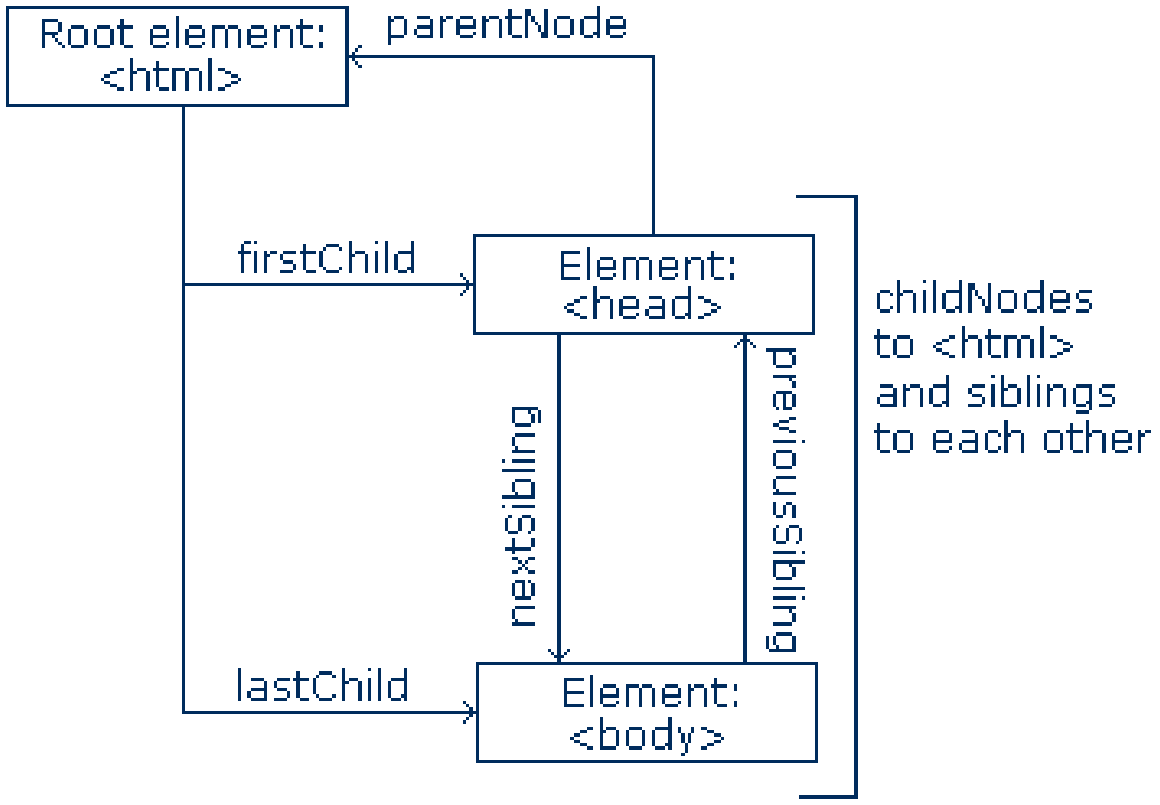
Simple Example



```
<html>
    <head>
        <title>Simple Example!</title>
    </head>
    <body>
        <h1>Heading</h1>
        <p id="my">A paragraph</p>
        <!--and that's all-->
    </body>
</html>
```

Node Tree

- The HTML DOM views a HTML document as a node-tree.
- All the nodes in the tree have relationships to each other.
  - Parent
  - Child
    - firstChild
    - lastChild
  - Sibling
    - nextSibling
    - previousSibling

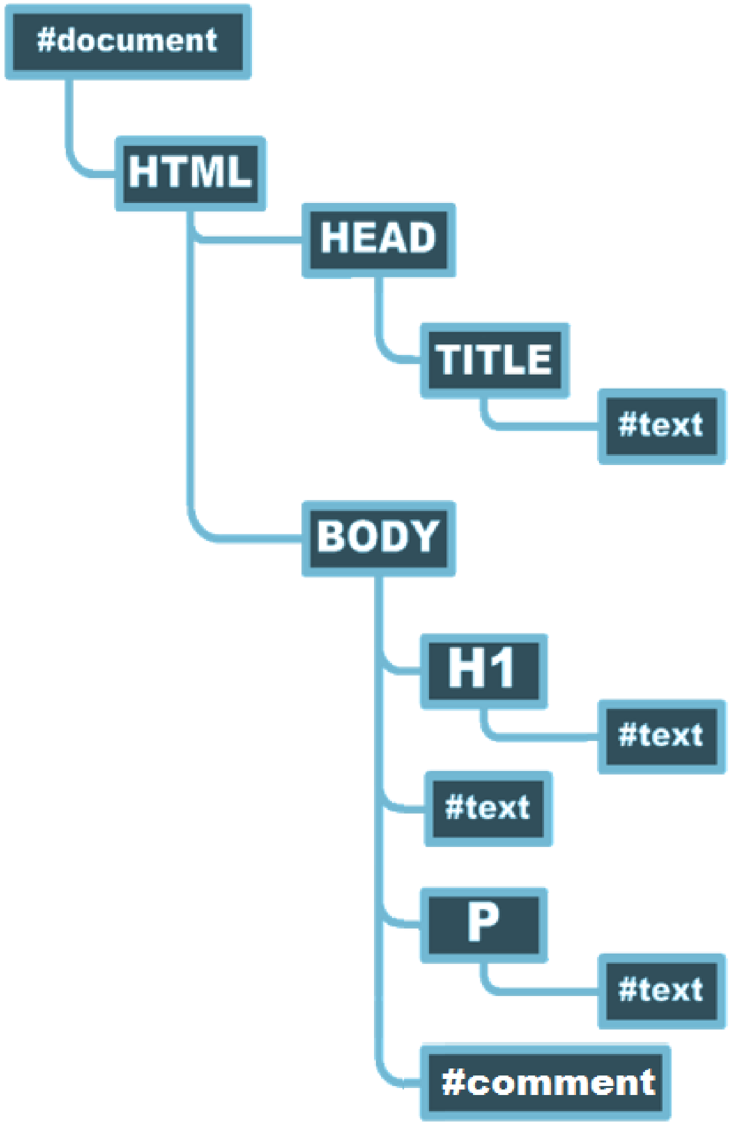


Nodes Relationships

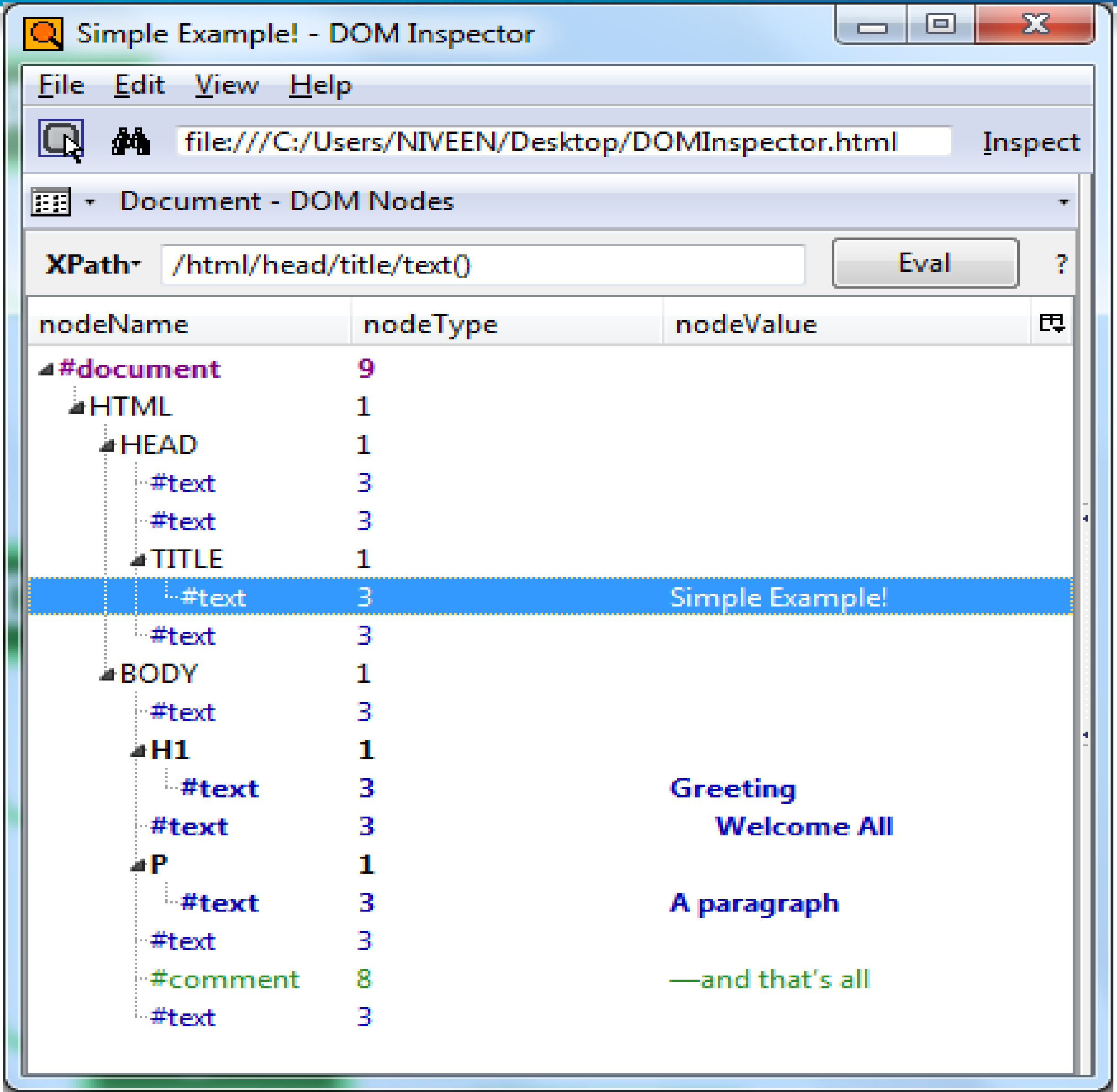
- The terms **parent**, **child**, and **sibling** are used to describe the relationships.
  - Parent nodes have children.
  - Children on the same level are called siblings (brothers or sisters).
- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent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

Simple Example

```
<html>
        <head>
            <title>Simple Example!</title>
        </head>
        <body>
            <h1>Greeting</h1>
             Welcome All
            <p>A paragraph</p>
            <!—and that's all-->
        </body>
 </html>
```



Simple Example



file:///C:/Users/NIVEEN/Desktop/DOMInspector.html

XPath: /html/head/title/text()

nodeName	nodeType	nodeValue
#document	9	
HTML	1	
HEAD	1	
#text	3	
#text	3	
TITLE	1	
#text	3	Simple Example!
#text	3	
BODY	1	
#text	3	
H1	1	
#text	3	Greeting
#text	3	Welcome All
P	1	
#text	3	A paragraph
#text	3	
#comment	8	—and that's all
#text	3	

Node Properties

- All nodes have three main properties

Property	Description
nodeName	Returns HTML Tag name in uppercase display
tagname	
nodeType	returns a numeric constant to determine node type. There are 12 node types.
nodeValue	returns null for all node types except for text and comment nodes.

Using nodeName
If node is text it returns #text
For comment it returns #comment
For document it returns #document

Value	Description
1	**Element Node**
2	**Attribute Node**
3	**Text Node**
8	**Comment Node**
9	**Document Node**

To get the Root Element: document.documentElement.

Node Collections

- Node Collections have One Property
  - **length : gives the length of the Collection.**
    - **e.g. childNodes.length:** returns number of elements inside the collection
- We can check if there is child collection using
  - **hasChildNodes():** Tells if a node has any children
- We can check if there is attribute collection using
  - **hasAttributes():** Tells if a node has any attributes

Collection	Description
childNodes[ ]	**Collection of element's children**
attributes[ ]	**Returns an array of the attributes of an element**

Dealing Nodes

- Dealing with nodes fall into four main categories:
  - Accessing Node
  - Modifying Node’s content
  - Adding New Node
  - Remove Node from tree

Accessing DOM Nodes

- You can access a node in 5 ways:
  - [window.]document.getElementById(“id”)
  - [window.]document.getElementsByName(“name”)
  - [window.]document.getElementsByTagName(“tagname”)
  - [window.]document.getElementsByClassName(“Classname”)
  - By navigating the node tree, using the node relationships
  - New HTML5 Selectors.

Note:

- [window.]document.all.elementID → Only in IE

Example!

Accessing DOM Nodes (Cont.)

- Navigating the node tree, using the node relationships

firstChild	Move direct to first child
lastChild	Move direct to last child
parentNode	To access child’s parent
nextSibling	Navigate down the tree one node step
previousSibling	Navigate up the tree one node step
Using children collection → childNodes[ ]	

Example!

Accessing DOM Nodes (Cont.)

- **Accessing DOM Nodes using HTML5 New selectors:**

  Using HTML5 New selector methods *querySelector(), querySelectorAll(),* that takes any CSS rule.

  – **Selecting the first div met**

    var elements = document.querySelector("div");

  – **Selecting all the divs in the current container**

    var elements = document.querySelectorAll("div");

  – **Selecting the first item with class SomeClass**

    var elements = document.querySelector(".SomeClass");

  – **Selecting the first item with id someID**

    var elements = document.querySelector("#someID");

Example!

Modifying Node's Content

- Changing the Text Node by using

innerHTML	Sets or returns the HTML contents (+text) of an element
innerText	Sets or returns the text of an element
textContent	Equivalent to innerText.
nodeValue → with text and comment nodes only	
setAttribute()	Modify/Adds a new attribute to an element
just using attributes as object properties	

- Modifying Styles
  – Node.style

Example!

Creating & Adding Nodes

Method	Description
createElement()	To create new tag element
createTextNode()	To create new text element
createAttribute()	To creates an attribute element
createComment()	To creates an comment element
appendChild()	To add new created node to DOM Tree at the end of the selected element.
cloneNode(true|false)	Creating new node a copy of existing node. It takes a Boolean value true: Deep copy with all its children or false: Shallow copy only the node
insertBefore()	Similar to appendChild() with extra parameter, specifying before which element to insert the new node.

Removing DOM Nodes

Method	Description
removeChild()	To remove node from DOM tree
replaceChild()	To remove node from DOM tree and put another one in its place
removeAttribute()	Removes a specified attribute from an element

- A quick way to wipe out all the content of a subtree is to set the innerHTML to a blank string. This will remove all of the children of <body>

    document.body.innerHTML="";

Example!

addEventListener() method

- The addEventListener() method attaches an event handler to the specified element.
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.
- You can add event listeners to any DOM object not only HTML elements. i.e the window object.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.
- When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

addEventListener() method (cont.)

- Syntax:

```
element.addEventListener(event, function, useCapture);
```

```
document.getElementById(“b1”).addEventListener("click",
myFunction);
function myFunction() {
    alert (“Button Clicked");
}
```

- The first parameter is the type of the event (like "click" or "mousedown").
- The second parameter is the function we want to call when the event occurs.
- The third parameter (optional parameter): is a boolean value specifying whether to use event bubbling or event capturing. Possible values:

  true - The event handler is executed in the capturing phase

  false- Default. The event handler is executed in the bubbling phase

addEventListener() method (cont.)

- You can easily remove an event listener by using the removeEventListener() method.

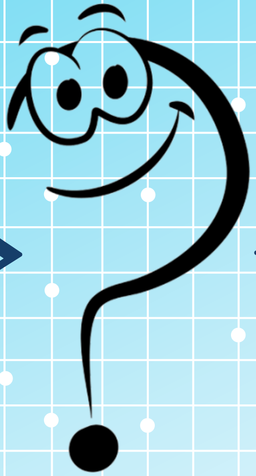
```
element.removeEventListener("mousemove", myFunction);
```

- Note: The addEventListener() and removeEventListener() methods are not supported in IE 8 and earlier versions and Opera 6.0 and earlier versions. However, for these specific browser versions, you can use the attachEvent() method to attach an event handlers to the element, and the detachEvent() method to remove it.

```
element.attachEvent(event, function);
element.detachEvent(event, function);
```

Summery

- Access nodes:
  - Using parent/child relationship properties parentNode, childNodes, firstChild, lastChild, nextSibling, previousSibling
  - Using getElementsById(), getElementsByTagName(), getElementsByName()
  - Using HTML5 New selectors.
- Modify nodes:
  - Using innerHTML or innerText/textContent
  - Using nodeValue or setAttribute() or just using attributes as object properties
- Remove nodes with
  - removeChild() or replaceChild()
- And add new ones with
  - appendChild(), cloneNode(), insertBefore()



<SCRIPT > </SCRIPT>

<script>document.writeln("Thank You!")</script>