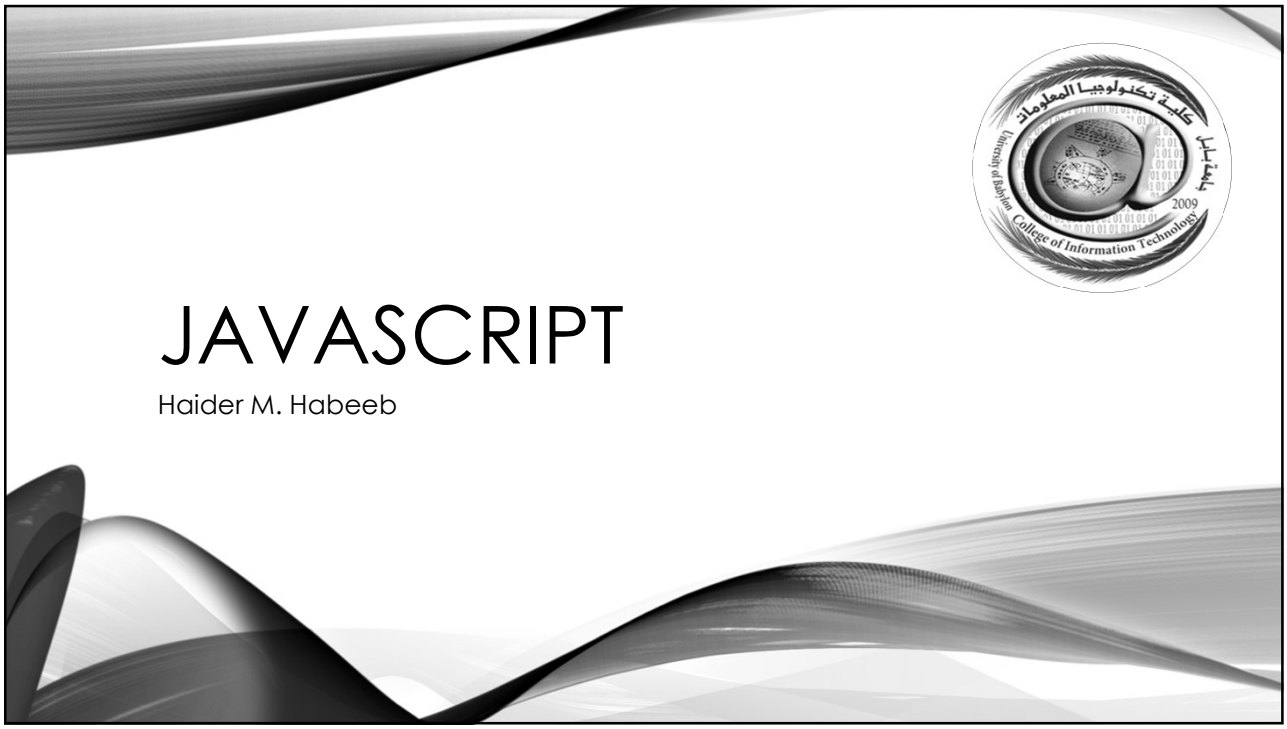
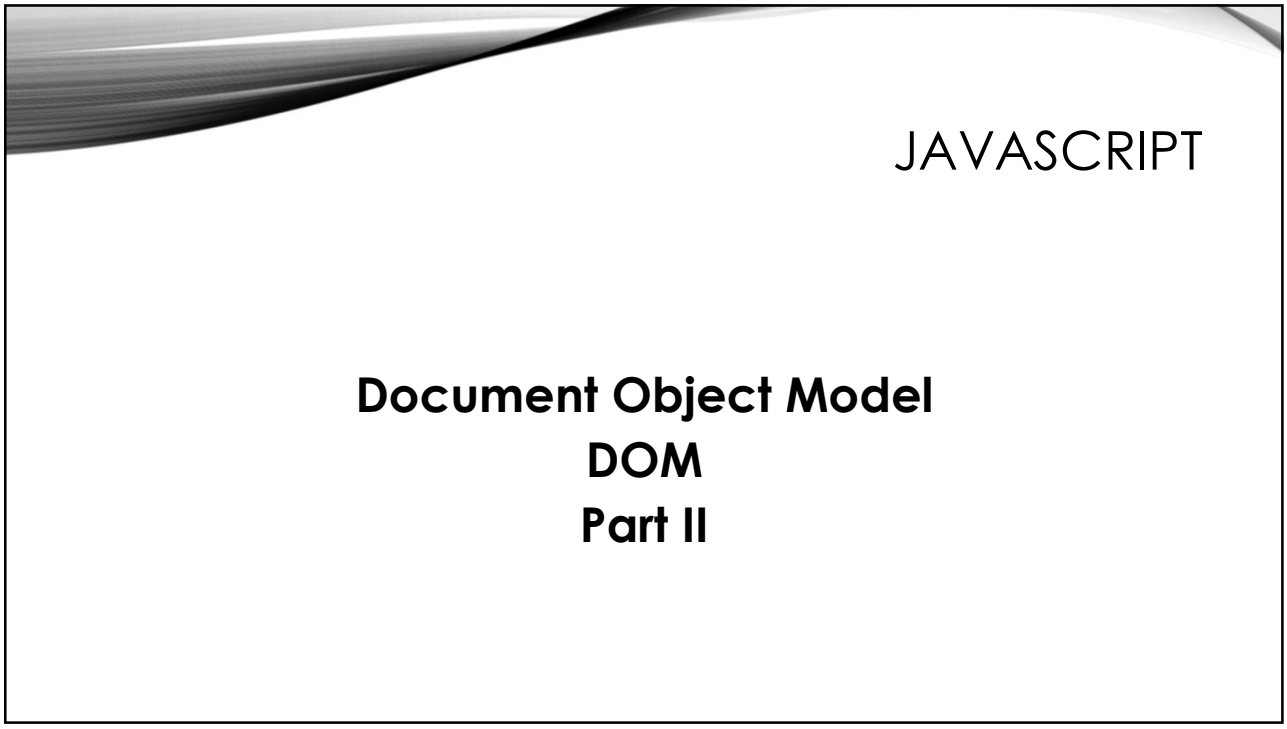
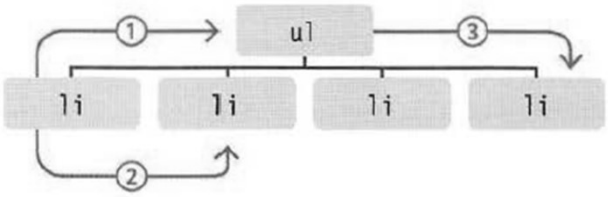

JAVASCRIPT
Haider M. Habeeb


JAVASCRIPT
Document Object Model
DOM
Part II

DOM TREE

- TRAVERSING THE DOM
  1. parentNode
  2. previousSibling / nextSibling
  3. firstChild / lastChild



DOM TREE

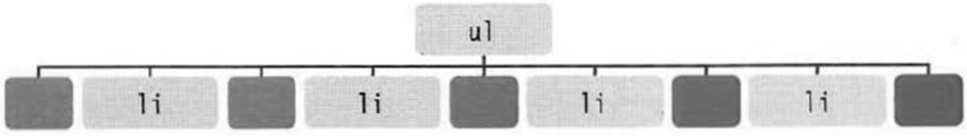
- PREVIOUS & NEXT SIBLING

```
// Select the starting point and find its siblings
var startItem = document.getElementById('two');
var prevItem = startItem.previousSibling;
var nextItem = startItem.nextSibling;

// Change the values of the siblings' class attributes
prevItem.className = 'complete';
nextItem.className = 'cool';
```

DOM TREE

- WHITESPACE NODES



Chrome, Firefox, Safari, and Opera create text nodes from whitespace (spaces and carriage returns).

DOM TREE

- PREVIOUS & NEXT ELEMENT SIBLING

```
var  statrtItem = document.getElementById('two');
var prevItem = startItem.previousElementSibling;
var nextItem = startItem.nextElementSibling;

prevItem.className = 'complete';
nextItem = className = 'cool';
```

DOM TREE

- FIRST & LAST CHILD

```
// Select the starting point and find its children
var startItem = document.getElementsByTagName('ul')[0];
var firstItem = startItem.firstChild;
var lastItem = startItem.lastChild;

// Change the values of the children's class attributes
firstItem.setAttribute('class', 'complete');
lastItem.setAttribute('class', 'cool');
```

DOM TREE

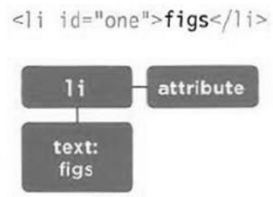
- FIRST & LAST ELEMENT CHILD

```
var  statrtItem = document.getElementByTagName('ul')[0];
var firstItem = startItem.firstElementChild;
var lastItem = startItem.lastElementChild;

firstItem.setAttribute('class', 'complete');
lastItem.setAttribute('class', 'cool';
```

DOM TREE

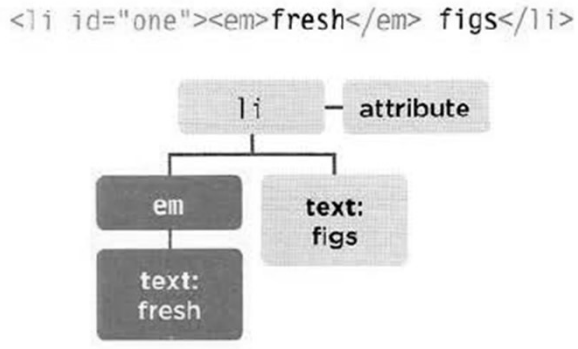
- HOW TO GET/UPDATE ELEMENT CONTENT
- Your choice of techniques depends upon what the element contains.



Above, the `<li>` element has:

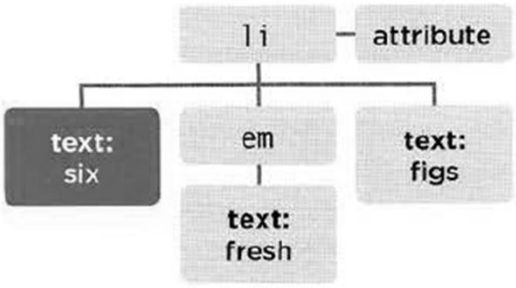
- One *child* node holding the word that you can see in the list item: `figs`
- An attribute node holding the `id` attribute.

DOM TREE



DOM TREE

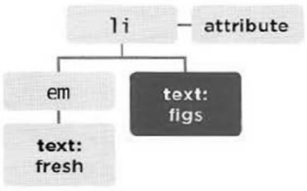
```
<li id="one">six <em>fresh</em> figs</li>
```



DOM TREE

- ACCESS & UPDATE A TEXT NODE WITH NODEVALUE

```
<li id="one"><em>fresh</em> figs</li>
```



The code below shows how you access the second text node. It will return the result: figs

```
document.getElementById('one').firstElementChild.nextElementSibling.nodeValue;
```

DOM TREE

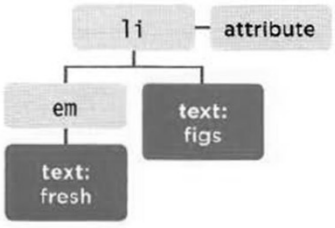
- ACCESSING & CHANGING A TEXT NODE

```
var itemTwo = document.getElementById('two');          // Get second list item
var  elText = itemTwo.firstElementChild.nodeValue;     // Get its text content
elText = elText.replace('pine nuts', 'kale');          // Change pine nuts to kale
itemTwo.firstChild.nodeValue = elText;                 // Update the list item
```

DOM TREE

- ACCESS & UPDATE TEXT WITH TEXTCONTENT

```
<li id="one"><em>fresh</em> figs</li>
```




```
document.getElementById('one').textContent;
```

DOM TREE

- Adding and Removing HTML Content:
  - innerHTML Property
  - DOM Manipulation

DOM TREE

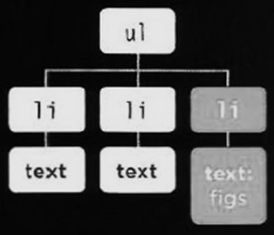
EXAMPLE: CHANGING A LIST ITEM

1: Create variable holding markup

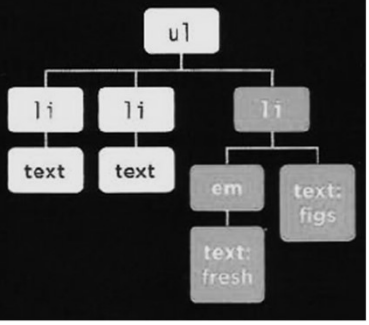
```
var item;
item = '<em>Fresh</em> figs';
```

You can have as much or as little markup in the variable as you want. It is a quick way to add a lot of markup to the DOM tree.

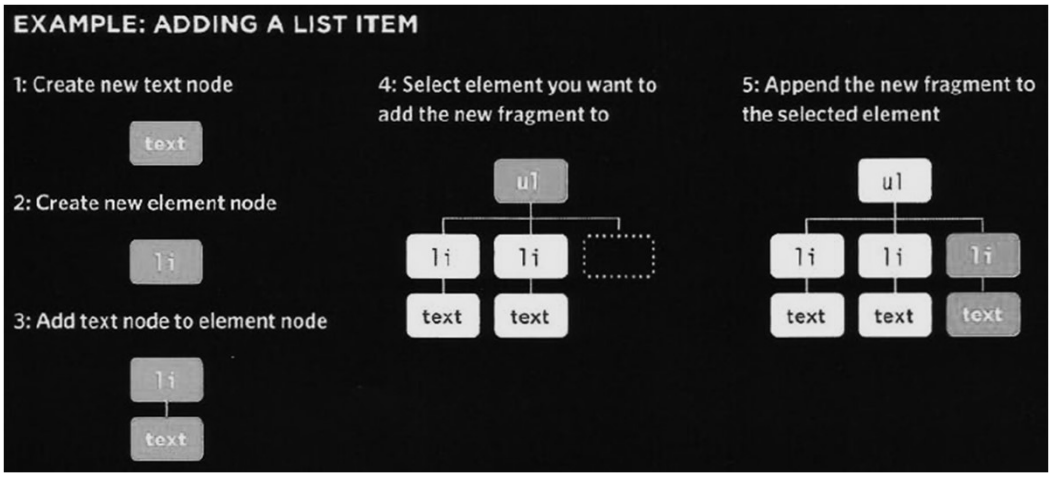
2: Select element whose content you want to update



3: Update content of selected element with new markup



DOM TREE



DOM TREE

- ADDING ELEMENTS USING DOM MANIPULATION

1. CREATE THE ELEMENT
   - createElement ()
2. GIVE IT CONTENT
   - createTextNode()
3. ADD IT TO THE DOM
   - appendChild()

DOM TREE

- ADDING AN ELEMENT TO THE DOM TREE

```
// Create a new element and store it in a variable.
var newEl = document.createElement('li');

// Create a text node and store it in a variable.
var newText = document.createTextNode('quinoa');

// Attach the new text node to the new element.
newEl.appendChild(newText);

// Find the position where the new element should be added.
var position = document.getElementsByTagName('ul')[0];

// Insert the new element into its position.
position.appendChild(newEl);
```

DOM TREE

- REMOVING ELEMENTS VIA DOM MANIPULATION
- Store the element to be removed in a variable
- Store the parent of that element in a variable
- Remove the element from its containing element
  - removeChild()

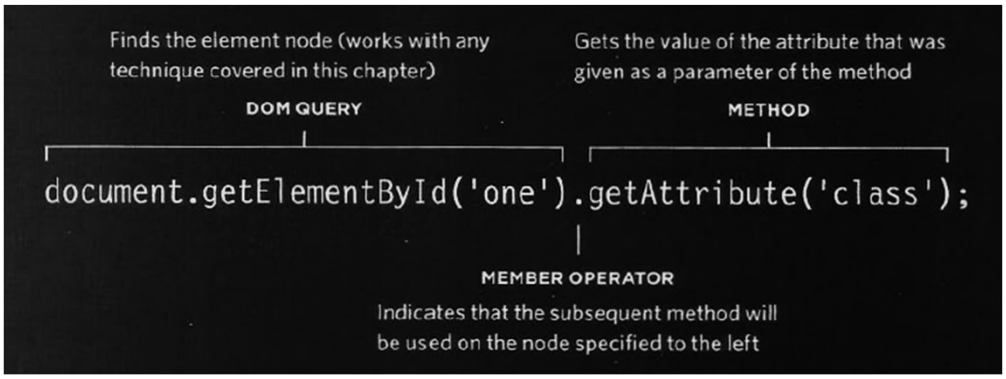
DOM TREE

- REMOVING AN ELEMENT FROM THE DOM TREE

```
var removeEl = document.getElementsByTagName('li')[3]; // The element to remove
var containerEl = removeEl.parentNode;                 // Its containing element
containerEl.removeChild(removeEl);                     // Removing the element
```

DOM TREE

- ATTRIBUTE NODES



DOM TREE

- ATTRIBUTE NODES

METHOD	DESCRIPTION
getAttribute()	gets the value of an attribute
hasAttribute()	checks if element node has a specified attribute
setAttribute()	sets the value of an attribute
removeAttribute()	removes an attribute from an element node

PROPERTY	DESCRIPTION
className	gets or sets the value of the class attribute
id	gets or sets the value of the id attribute

DOM TREE

- CHECK FOR AN ATTRIBUTE AND GET ITS VALUES

```
var firstItem = document.getElementById('one');    // Get first list item

if (firstItem.hasAttribute('class')) {             // If it has class attribute
    var attr = firstItem.getAttribute('class');    // Get the attribute

    // Add the value of the attribute after the list
    var el = document.getElementById('scriptResults');
    el.innerHTML = '<p>The first item has a class name: ' + attr + '</p>';

}
```

DOM TREE

- CREATING ATTRIBUTES & CHANGING THEIR VALUES

```
var firstItem = document.getElementById('one'); // Get the first item
firstItem.className = 'complete';               // Change its class attribute

var fourthItem = document.getElementsByTagName('li').item(3);// Get fourth item
el2.setAttribute('class', 'cool');              // Add an attribute to it
```

DOM TREE

- REMOVING ATTRIBUTES

```
var firstItem = document.getElementById('one'); // Get the first item
if (firstItem.hasAttribute('class')) {          // If it has a class attribute
    firstItem.removeAttribute('class');         // Remove its class attribute
}
```

SUMMARY: DOM TREE

- The browser represents the page using a DOM tree.
- DOM trees have four types of nodes: document nodes, element nodes, attribute nodes, and text nodes.
- You can select element nodes by their id or class attributes, by tag name, or using CSS selector syntax.
- Whenever a DOM query can return more than one node, it will always return a Nodelist.
- From an element node, you can access and update its content using properties such as textContent and innerHTML or using DOM manipulation techniques.
- An element node can contain multiple text nodes and child elements that are siblings of each other.