JAVASCRIPT - DOCUMENT OBJECT MODEL (DOM)

Asst. Lect. Ali Al-khawaja

INTRODUCTION

- **The Document Object Model (DOM) specifies how browsers should create a model of an HTML page and how JavaScript can access and update the contents of a web page while it is in the browser window.**

- The DOM is neither part of HTML, nor part of JavaScript; it is a separate set of rules. It is implemented by all major browser makers, and covers two primary areas:
  1. MAKING A MODEL OF THE HTML PAGE
  2. ACCESSING AND CHANGING THE HTML PAGE

THE DOM TREE IS A MODEL OF A WEB PAGE

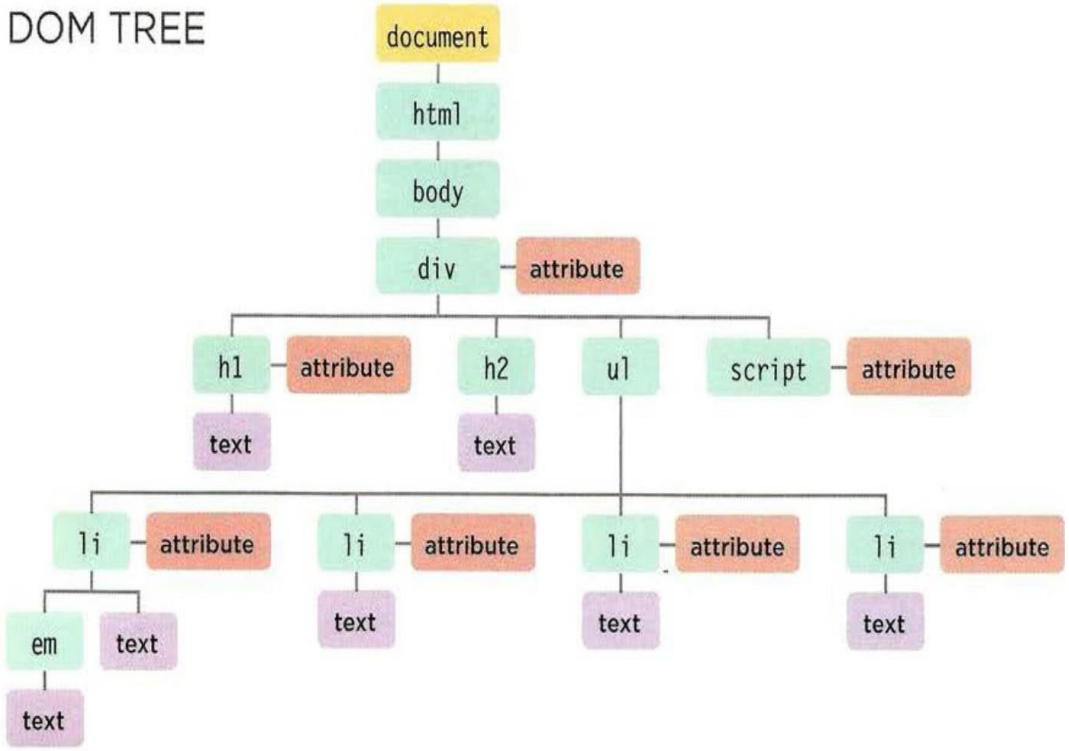
As a browser loads a web page, it creates a model of that page. The model is called a DOM tree, and it is stored in the browsers' memory. It consists of four main types of nodes.

1 THE DOCUMENT NODE

2 ELEMENT NODES

3 ATTRIBUTE NODES

4 TEXT NODES

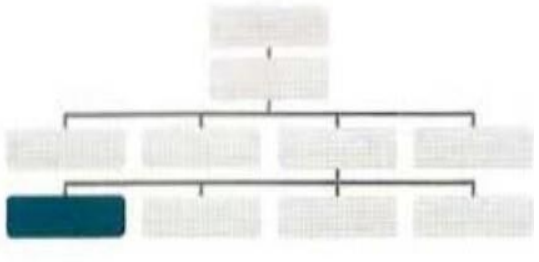


WORKING WITH THE DOM TREE

- Accessing and updating the DOM tree involves two steps:

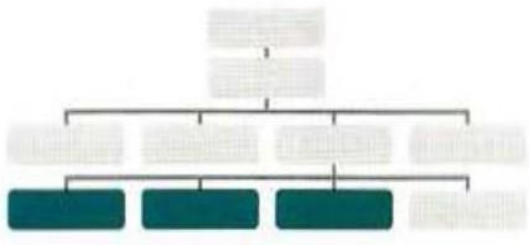
STEP 1:ACCESS THE ELEMENTS

1- SELECT AN INDIVIDUAL ELEMENT NODE



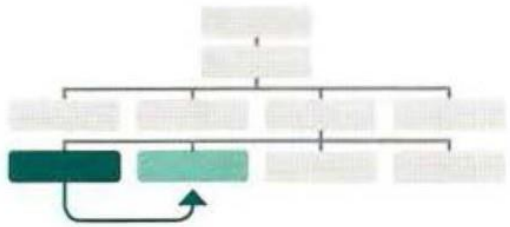
getElementById()
querySelector()

2- SELECT MULTIPLE ELEMENTS (NODELISTS)



getElementsByClassName()
getElementsByTagName()
querySelectorAll()

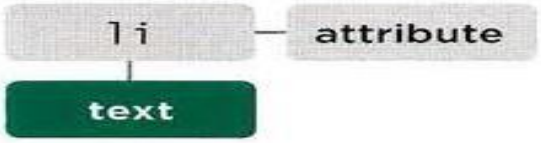
3- TRAVERSING BETWEEN ELEMENT NODES



parentNode
previousSibling/nextSibling
firstChild /lastChild

STEP 2: WORK W IT H THOSE ELEMENTS

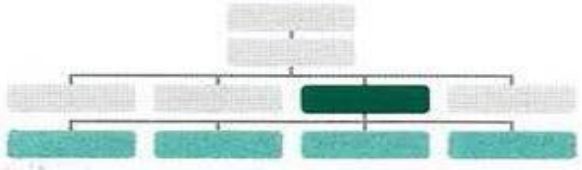
ACCESS/ UPDATE TEXT NODES



The text inside any element is stored inside a text node. To access the text node above:

1. Select the element
2. Use the firstChild property to get t he text node.
3. Use the text node's only property (nodeValue) to get the text from t he element

WORK W ITH HTML CONTENT



One property allows access to child elements and text content: innerHTML

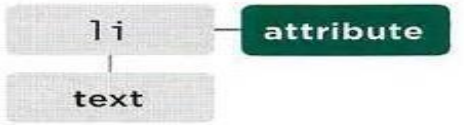
Another just the text content: . textContent

Several methods let you create new nodes, add nodes to a tree, and remove nodes from a tree:

create Element() createTextNode()
appendChild()/removeChild()

This is called D O M manipulation.

ACCESS OR UPDATE ATTRIBUTE VALUES



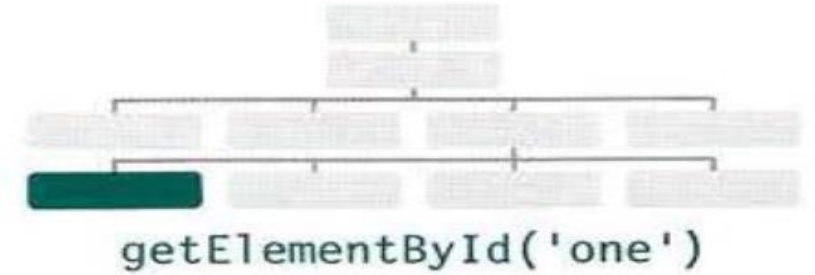
Methods work with attributes:
className/id
Lets you get or update the value of the class and id attributes.

hasAttribute() checks if an attribute exists
getAttribute() gets its value
setAttribute() updates t he value
removeAttribute() removes an attribute

METHODS THAT RETURN A SINGLE ELEMENT NODE

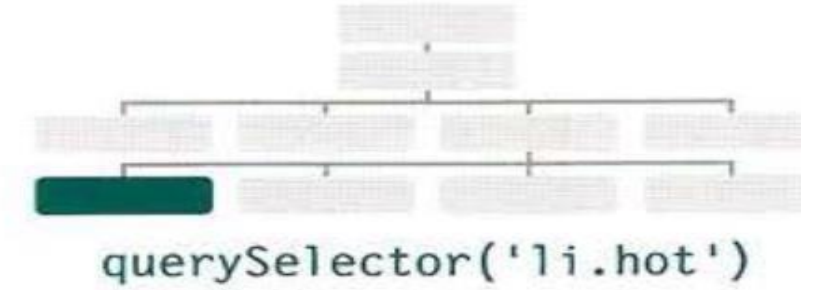
1 getElementByld('id')

- Selects an individua l element give n the value of its id attribute .
  The HTML must have an id attribute in order for it to be selectable.



2 querySelector('css selector')

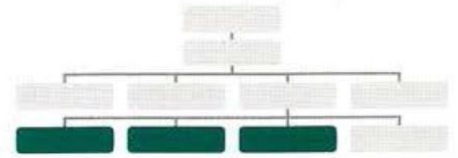
Uses CSS selector syntax t hat would select one or more elements.This method returns only the first of the matching elements.



M ETHODS THAT RETURN ONE OR MORE ELEMENTS (AS A NODELIST):

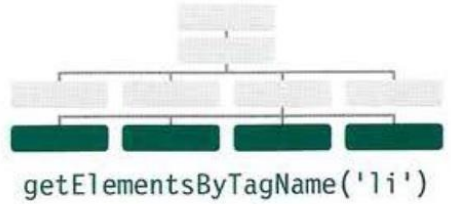
1 getElementsByClassName('class')

Selects one or more elements given the value of their class attribute.
The HTML must have a class attribute for it to be selectable.
This method is faster than querySelectorAll().



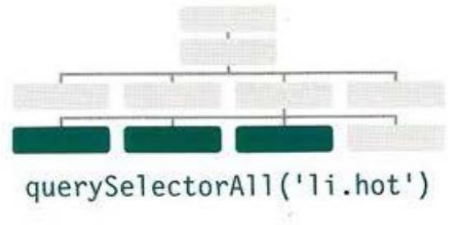
2 getElementsByTagName('tagName')

Selects all elements on t he page with the specified t ag name.
This method is faster than querySe1ectorAll().



3 querySelectorAll('css selector')

Uses CSS se lector syntax to select one or more element s and returns all of those t hat match.



SELECTING ELEMENTS USING ID ATTRIBUTES

HTML

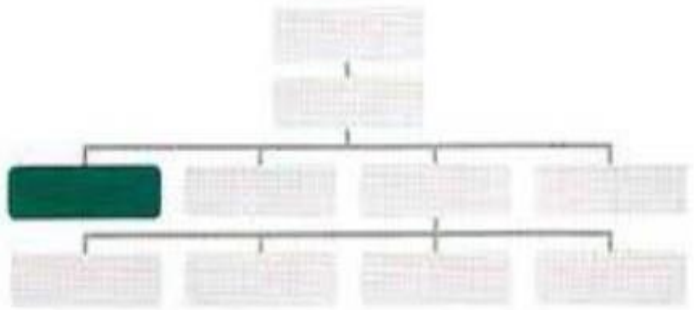
```
<h1 id="header">List King</h1>
<h2>Buy groceries</h2>
<ul>
  <li id="one" class="hot"><em>fresh</em>
    figs</li>
  <li id="two" class="hot">pine nuts</li>
  <li id="three" class="hot">honey</li>
  <li id="four">balsamic vinegar</li>
</ul>
```

JAVASCRIPT

```
// Select the element and store it in a variable.
var el = document.getElementById('one');

// Change the value of the class attribute.
el.className = 'cool';
```

RESULT

DOM QUERIES EXAMPLES

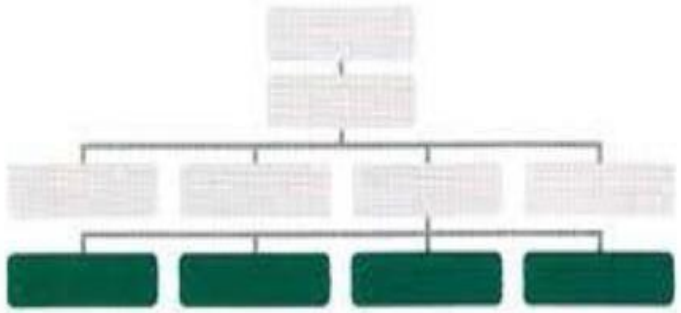


getElementsByTagName('h1')

Even though this query only returns one element, the method still returns a NodeList because of the potential for returning more than one element.

INDEX NUMBER & ELEMENT
0 <h1>

DOM QUERIES EXAMPLES

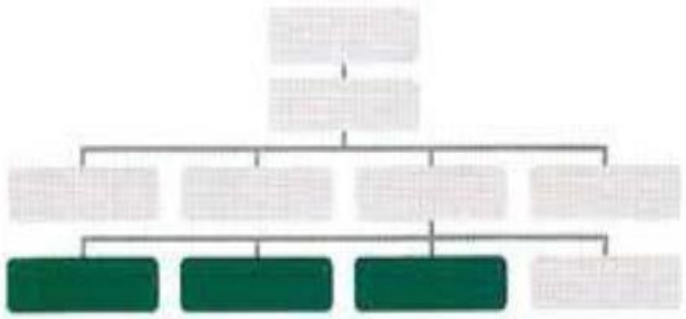


getElementsByTagName('li')

This method returns four elements, one for each of the <li> elements on the page. They appear in the same order as they do in the HTML page.

INDEX NUMBER & ELEMENT	
0	<li id="one" class="hot">
1	<li id="two" class="hot">
2	<li id="three" class="hot">
3	<li id="four">

DOM QUERIES EXAMPLES

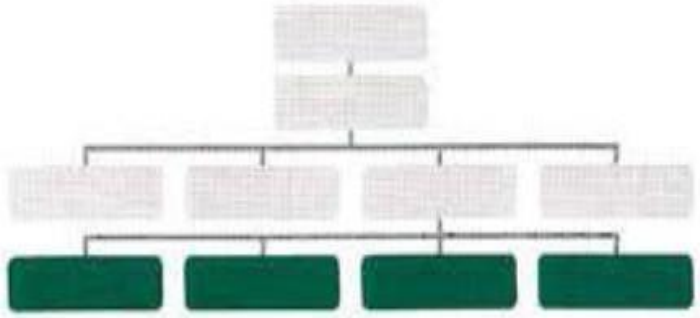


```
getElementsByClassName('hot')
```

This NodeList contains only three of the `<li>` elements because we are searching for elements by the value of their `class` attribute, not tag name.

INDEX NUMBER & ELEMENT	
0	`<li id="one" class="hot">`
1	`<li id="two" class="hot">`
2	`<li id="three" class="hot">`

DOM QUERIES EXAMPLES



```
querySelectorAll('li[id]')
```

This method returns four elements, one for each of the <li> elements on the page that have an id attribute (regardless of the values of the id attributes).

INDEX NUMBER & ELEMENT	
0	<li id="one" class="hot">
1	<li id="two" class="hot">
2	<li id="three" class="hot">
3	<li id="four">

DOM QUERIES EXAMPLES

JAVASCRIPT

```
var elements = document.getElementsByClassName('hot'); // Find hot items

if (elements.length > 2) {                              // If 3 or more are found

    var el = elements[2];           // Select the third one from the NodeList
    el.className = 'cool';          // Change the value of its class attribute

}
```

RESULT

DOM QUERIES EXAMPLES

JAVASCRIPT

```
var elements = document.getElementsByTagName('li');    // Find <li> elements

if (elements.length > 0) {                              // If 1 or more are found

    var el = elements[0];              // Select the first one using array syntax
    el.className = 'cool';             // Change the value of the class attribute

}
```

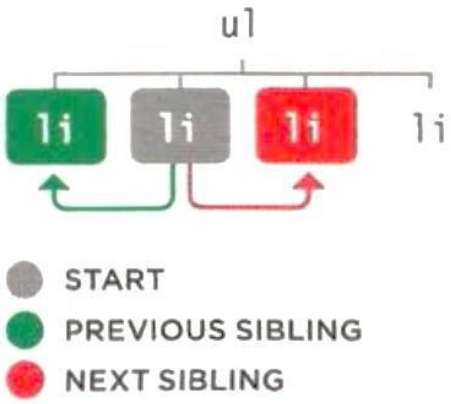
RESULT

SELECTING ELEMENTS USING CSS SELECTORS

```
// querySelector() only returns the first match
var el = document.querySelector('li.hot');
el.className = 'cool';

// querySelectorAll returns a NodeList
// The second matching element (the third list item) is selected and changed
var els = document.querySelectorAll('li.hot');
els[1].className = 'cool';
```

PREVIOUS & NEXT SIBLING

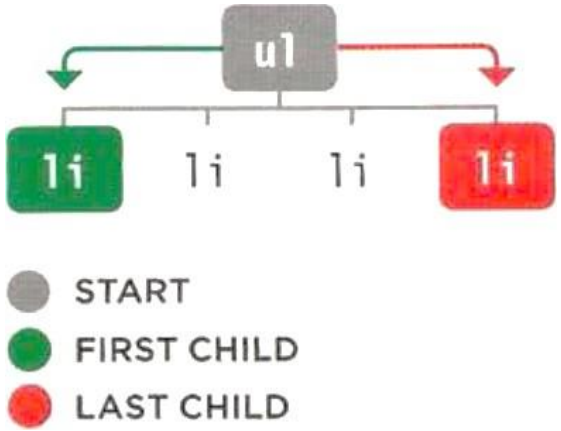
```
<ul>
    <li id="one" class="hot"><em>fresh</em> figs </li>
    <li id="two“ class="hot">pine nuts</li>
    <li id="three" class ="hot">honey</li>
    <li id="four ">balsamic vinegar</li>
</ul>
```




```
//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
    prevltem.className='complete ' ;
    nextltem.className='cool';
```

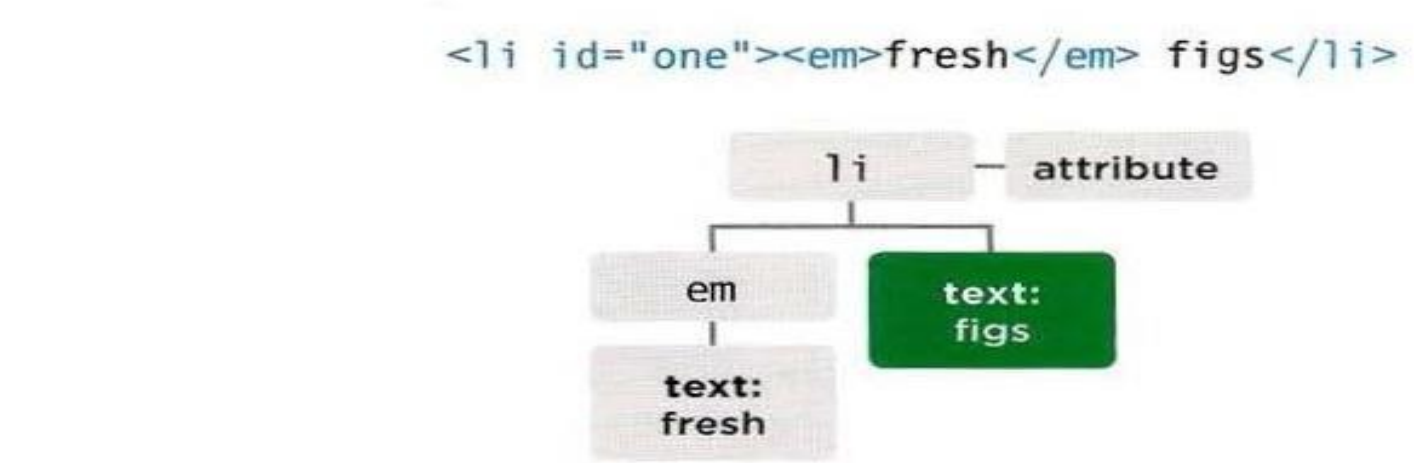
FIRST & LAST CHILD

```
<ul>
    <li id="one" class="hot"><em>fresh</em> figs </li>
    <li id="two“ class="hot">pine nuts</li>
    <li id="three" class ="hot">honey</li>
    <li id="four ">balsamic vinegar</li>
</ul>
```




```
//Select the starting point and find its children
    var startItem = document.getElementsByTagName('ul ') [0] ;
    var firstItem = startIltem.firstChild;
    var lastItem = startItem.lastChild;
//Change the values of the children's class attributes
    firstItem.setAttribute('class ' , 'complete');
    lastItem.setAttribute('class', ' cool');
```

ACCESS & UPDATE A TEXT NODE WITH NODEVALUE




```
document.getElementById('one').firstChild.nextSibling.nodeValue;
```

① ② ③ ④

ACCESSING & CHANGING A TEXT NODE

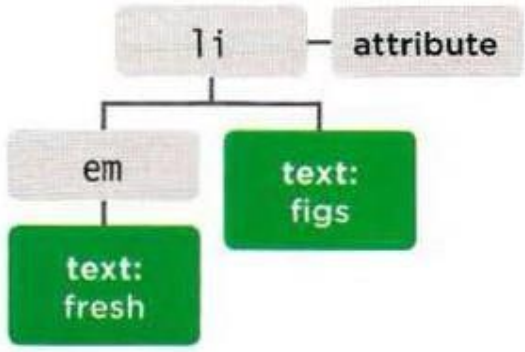
JAVASCRIPT

```
var itemTwo = document.getElementById('two');        // Get second list item

var elText  = itemTwo.firstChild.nodeValue;          // Get its text content

elText = elText.replace('pine nuts', 'kale');        // Change pine nuts to kale

itemTwo.firstChild.nodeValue = elText;               // Update the list item
```

RESULT

ACCESS & UPDATE TEXT WITH TEXTCONTENT (& INNERTEXT)


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




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

To collect the text from the <l i> elements in our example (and ignore any markup inside the element) you can use the textContent property on the containing <l i > element. In this case it would return the value: fresh figs.

You can also use this property to update the content of the element; it replaces the entire content of it (including any markup).

JAVASCRIPT

```
var firstItem = document.getElementById('one');    // Find first list item
var showTextContent = firstItem.textContent;       // Get value of textContent
var showInnerText = firstItem.innerText;           // Get value of innerText

// Show the content of these two properties at the end of the list
var msg = '<p>textContent: ' + showTextContent + '</p>';
    msg += '<p>innerText: ' + showInnerText + '</p>';
var el = document.getElementById('scriptResults');
el.innerHTML = msg;

firstItem.textContent = 'sourdough bread';         // Update the first list item
```