



JAVASCRIPT

Haider M. Habeeb

JAVASCRIPT

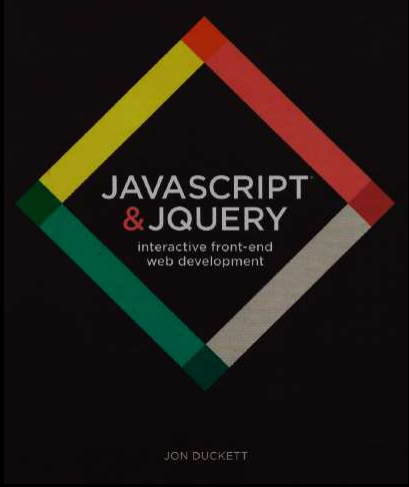

Document Object Model

DOM

Part I

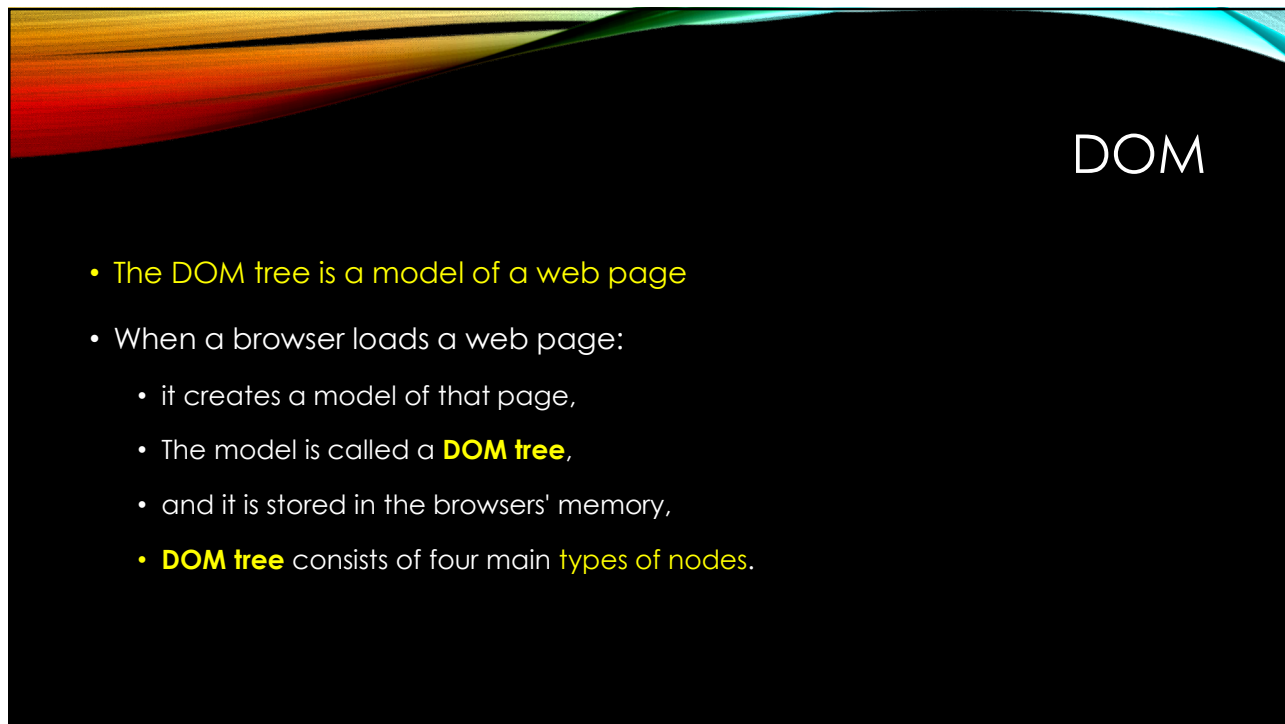
ATTENTION

- You are required to read chapter four of the reference.



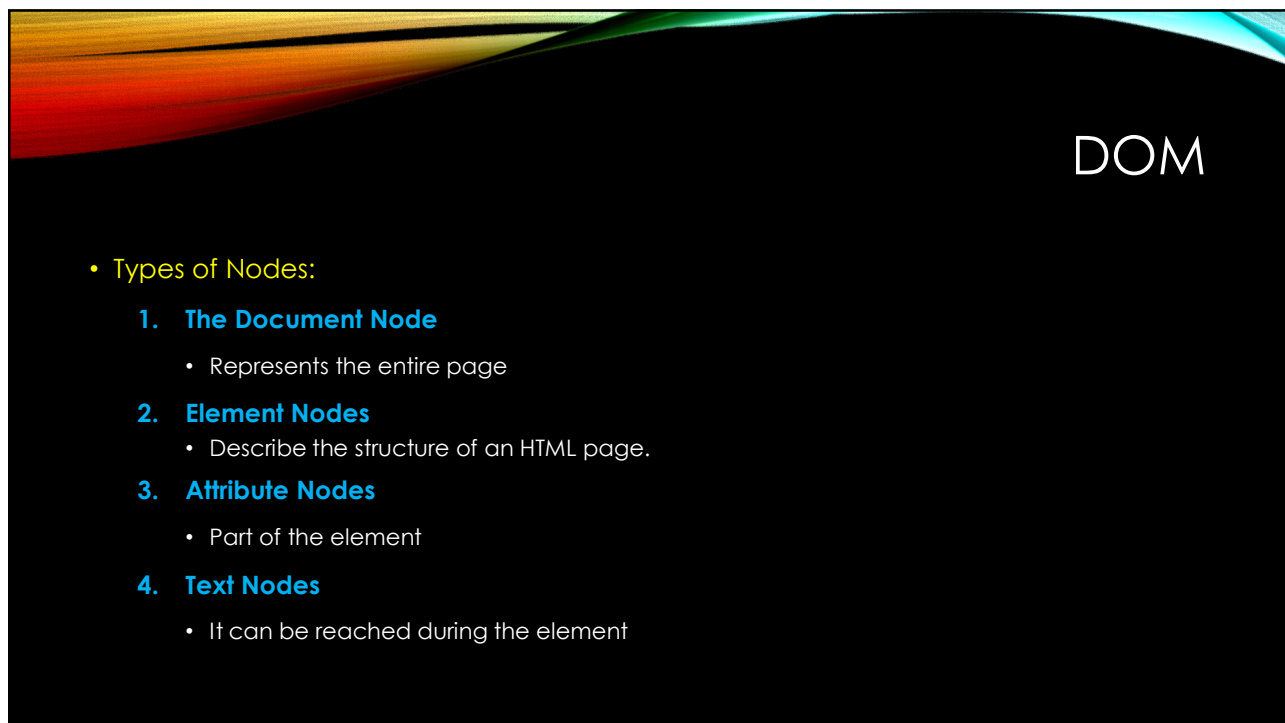
DOM

- The Document Object Model (**DOM**) specifies:
 - how browsers should create a model of an HTML page.
 - 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;
- **DOM** is a separate set of rules.

A presentation slide with a dark background and a colorful abstract header. The word "DOM" is written in white in the top right corner. The slide contains a bulleted list of information about the DOM tree.

DOM

- The DOM tree is a model of a web page
- When 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,
 - **DOM tree** consists of four main types of nodes.

A presentation slide with a dark background and a colorful abstract header. The word "DOM" is written in white in the top right corner. The slide contains a numbered list of node types, each with a sub-bulleted list of descriptions.

DOM

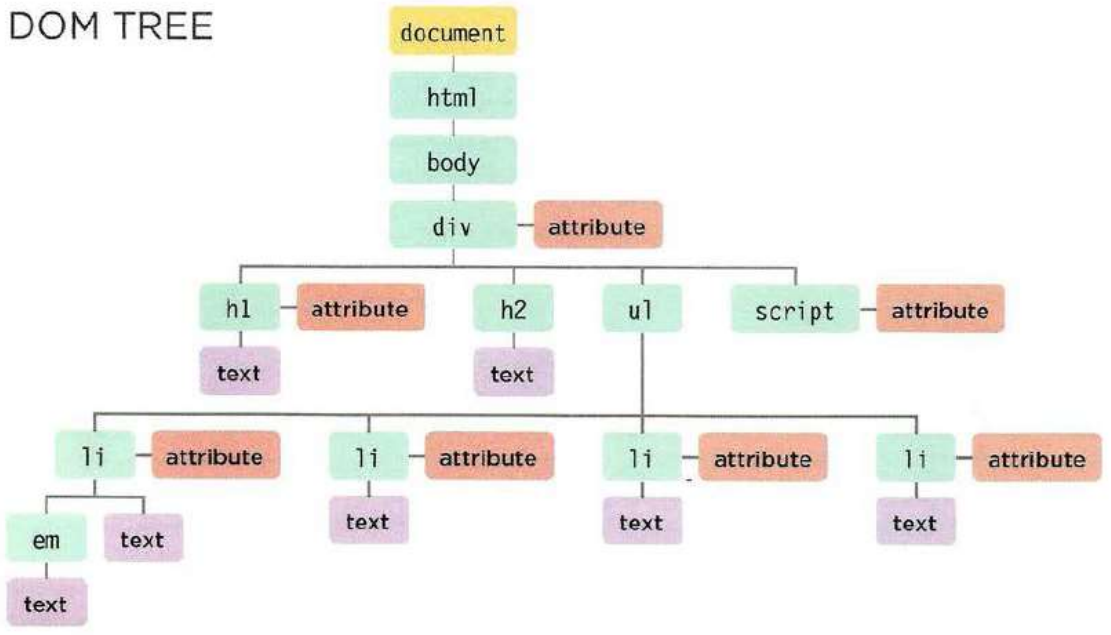
- Types of Nodes:
 1. **The Document Node**
 - Represents the entire page
 2. **Element Nodes**
 - Describe the structure of an HTML page.
 3. **Attribute Nodes**
 - Part of the element
 4. **Text Nodes**
 - It can be reached during the element

DOM TREE

- Body Of HTML Page

```
<html>  
  <body>  
    <div id="page">  
      <h1 id="header">List</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>  
      <script src="js/list.js"></script>  
    </div>  
  </body>  
</html>
```

DOM TREE



WORKING WITH THE DOM TREE

- Accessing and updating the DOM tree involves two steps:
 1. Locate the **node** that represents the element you want to work with.
 2. Use its **text** content, **child** elements, and **attributes**.

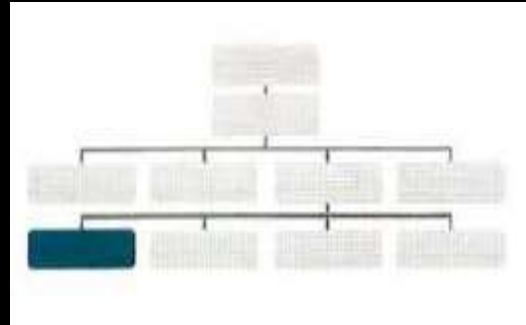
WORKING WITH THE DOM TREE

- **STEP 1: ACCESS THE ELEMENTS**
- DOM Queries
 1. Select an Individual Element Node
 2. Select Multiple Elements (Nodelists)
- Traversing the DOM.
 3. Traversing Between Element Nodes

DOM TREE

- **SELECT AN INDIVIDUAL ELEMENT NODE**

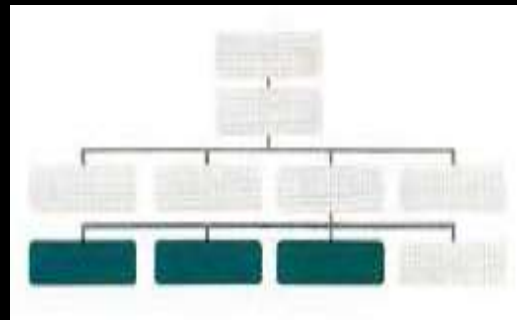
1. `getElementById()`
 - Uses the value of an element's id attribute
2. `querySelector()`
 - Uses a CSS selector, and returns the first matching element.
3. Traversing from one element to another within the DOM tree



DOM TREE

- **SELECT MULTIPLE ELEMENTS (NODELISTS)**

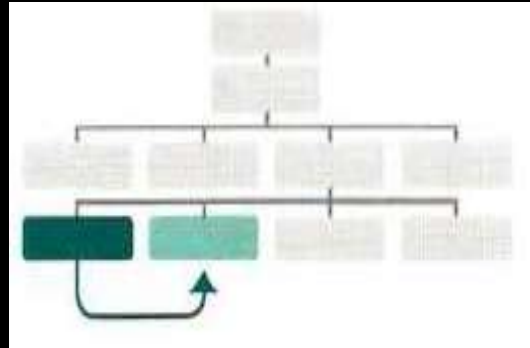
1. `getElementsByClassName()`
 - Selects all elements that have a specific value for their class attribute.
2. `getElementsByTagName()`
 - Selects all elements that have the specified tag name.
3. `querySelectorAll()`
 - Uses a CSS selector to select all matching elements.



DOM TREE

- **TRAVERSING BETWEEN ELEMENT NODES**

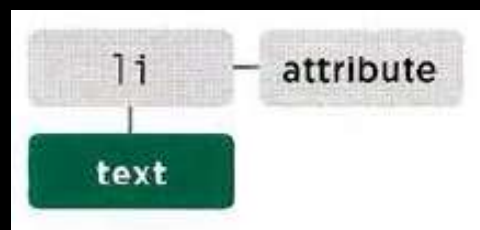
- **parentNode**
 - Selects the parent of the current element node.
- **previousSibling / nextSibling**
 - Selects the previous or next sibling from the DOM tree.
- **firstChild / lastChild**
 - Select the first or last child of the current element.



DOM TREE

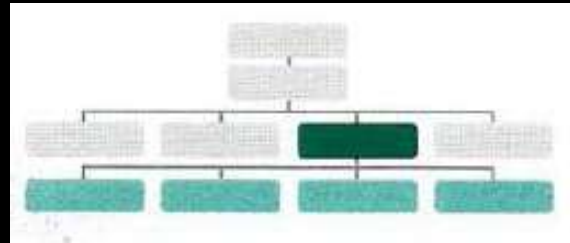
- **STEP 2: WORK WITH THOSE ELEMENTS**

- Access / Update Text Nodes
- **nodeValue**
 - This property lets you access or update contents of a text node.
- To access the text node on the right:
 1. Select the `<i>` element
 2. Use the `firstChild` property to get the text node
 3. Use the text node's only property (`nodeValue`) to get the text from the element



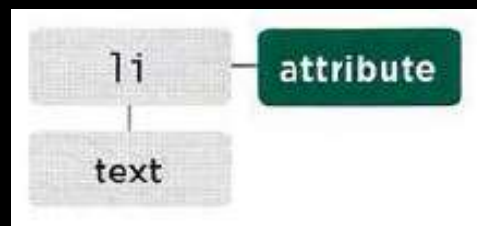
DOM TREE

- **STEP 2: WORK WITH THOSE ELEMENTS**
- Work With HTML Content
- `innerHTML`
- `textContent`



DOM TREE

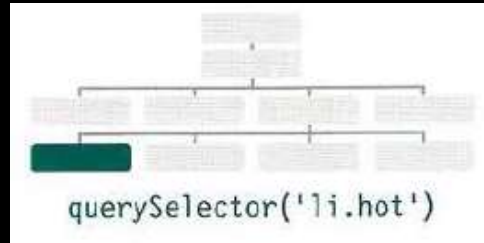
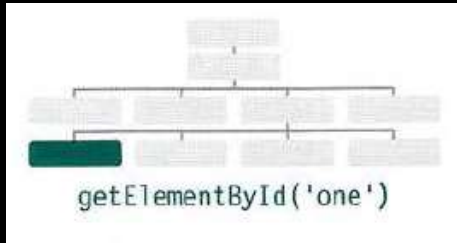
- **STEP 2: WORK WITH THOSE ELEMENTS**
- Access Or Update Attribute Values
- `className /id`
 - Lets you get or update the value of the class and id attributes.
- `hasAttribute()`
 - checks if an attribute exists.
- `getAttribute()`
 - gets attribute's value.
- `setAttribute()`
 - updates the value.
- `removeAttribute()`
 - removes an attribute.



DOM TREE

- Methods that return a single element node:

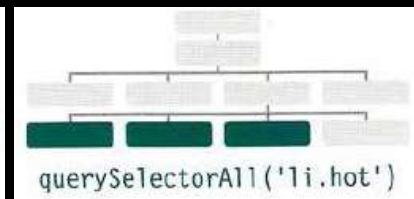
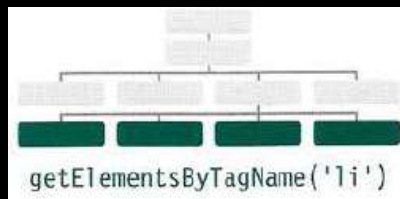
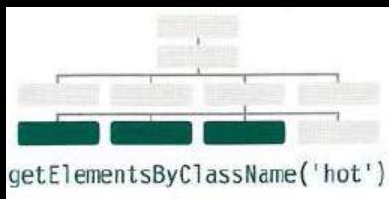
- `getElementById('id')`
- `querySelector('css selector')`



DOM TREE

- Methods that return one or more elements (as a nodelist):

- `getElementsByClassName('class')`
- `getElementsByTagName('tagName')`
- `querySelectorAll('css selector')`



DOM TREE

- Selecting elements using id attributes

```
// Select the element and store it in a variable.  
var e1 = document.getElementById('one');  
  
// Change the value of the class attribute.  
e1.className = 'cool';
```

```
<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>
```

DOM TREE

- NodeLists: DOM queries that return more than one element
- When a DOM method can return more than one element, it returns a Nodelist (even if it only finds one matching element).
- **Nodelists** look like arrays and are numbered like arrays, but they are not actually arrays; they are a type of object called a **collection**.
- Nodelist has properties and methods, notably:
 - `length` property
 - `item()` method

DOM TREE

`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 TREE

`getElementsByTagName('li')`

This method returns four elements, one for each of the 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 TREE

`getElementsByClassName('hot')`

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

INDEX NUMBER & ELEMENT

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

DOM TREE

`querySelectorAll('li[id]')`

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

INDEX NUMBER & ELEMENT

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

DOM TREE

- **Selecting An Element From A NodeList**
- There are two ways to select an element from a NodeList:
 - The item() method.
 - Array syntax.
- Both require the index number of the element you want.

DOM TREE

- The Item() Method:

```
var elements = document.getElementsByClassName('hot')
if (elements.length >= 1) {
    var firstItem = elements.item(0);
}
```

DOM TREE

- Array Syntax

```
var elements = document.getElementsByClassName('hot');  
if (elements.length >= 1) {  
    var firstItem = elements[0];  
}
```

DOM TREE

- SELECTING ELEMENTS USING CLASS ATTRIBUTES

```
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  
  
}
```


DOM TREE

- SELECTING ELEMENTS BY TAG NAME

```
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

}
```

DOM TREE

- 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';
```

DOM TREE

- LOOPING THROUGH A NODELIST

```
var hotItems = document.querySelectorAll('li.hot'); // Store NodeList in array
if (hotItems.length > 0) {                       // If it contains items
    for (var i=0; i<hotItems.length; i++) {      // Loop through each item
        hotItems[i].className = 'cool';         // Change value of class attribute
    }
}
```



Thank you!