JavaScript

JS HTML DOM

JS HTML DOM

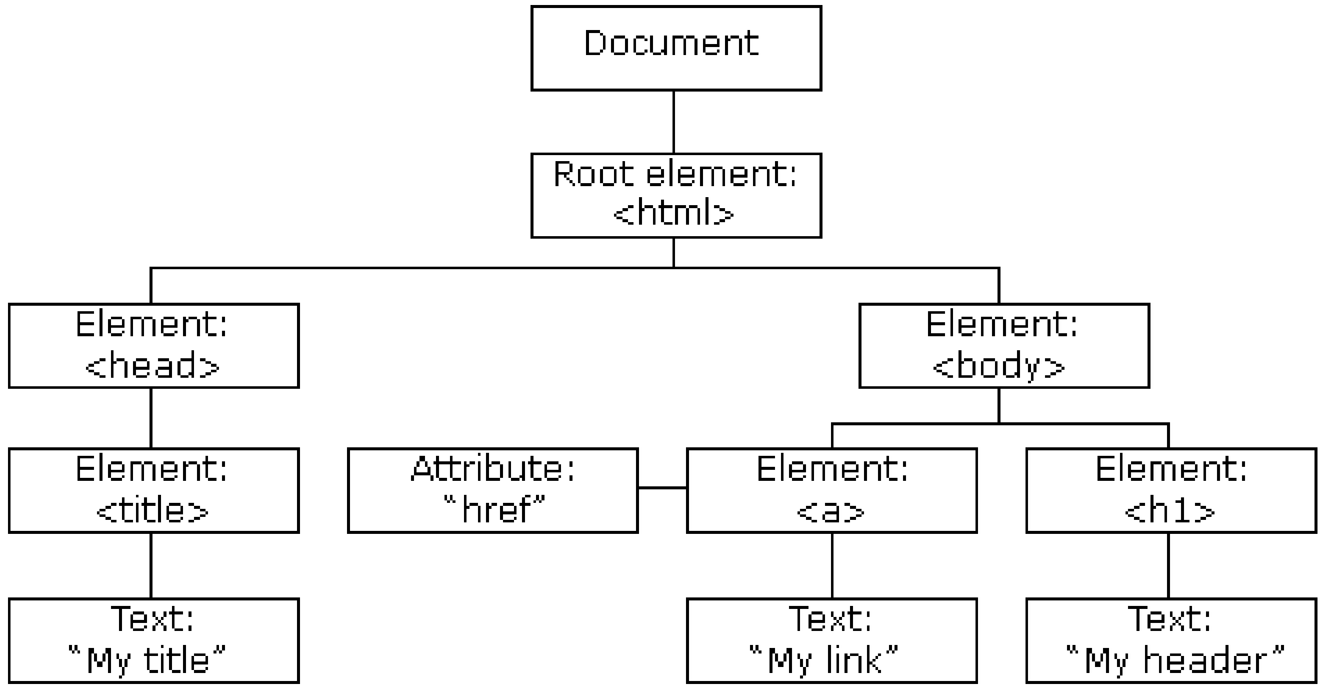
- The Document Object Model (DOM) is an internal representation of the HTML elements on a web page.
- When a web page is loaded, the browser creates a Document Object Model of the page.
- DOM is a hierarchical collection of nodes in the web browser's memory that represents the current web page. It defines:
    - The HTML elements as objects.
    - The properties of all HTML elements.
    - The methods to access all HTML elements.
    - The events for all HTML elements.

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

- With the HTML DOM, JavaScript can access all the elements of an HTML document.

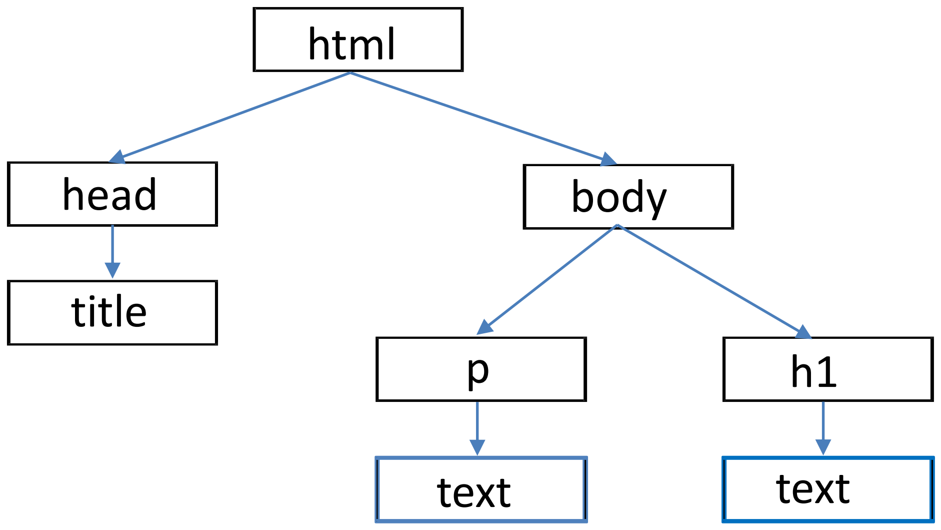
JS HTML DOM: The Document Tree

- Nodes in the DOM tree represent elements and are related to each other through child-parent relationships
- A node may have multiple children, but only one parent.
- Nodes with the same parent node are referred to as siblings.



DOM Tree: Example

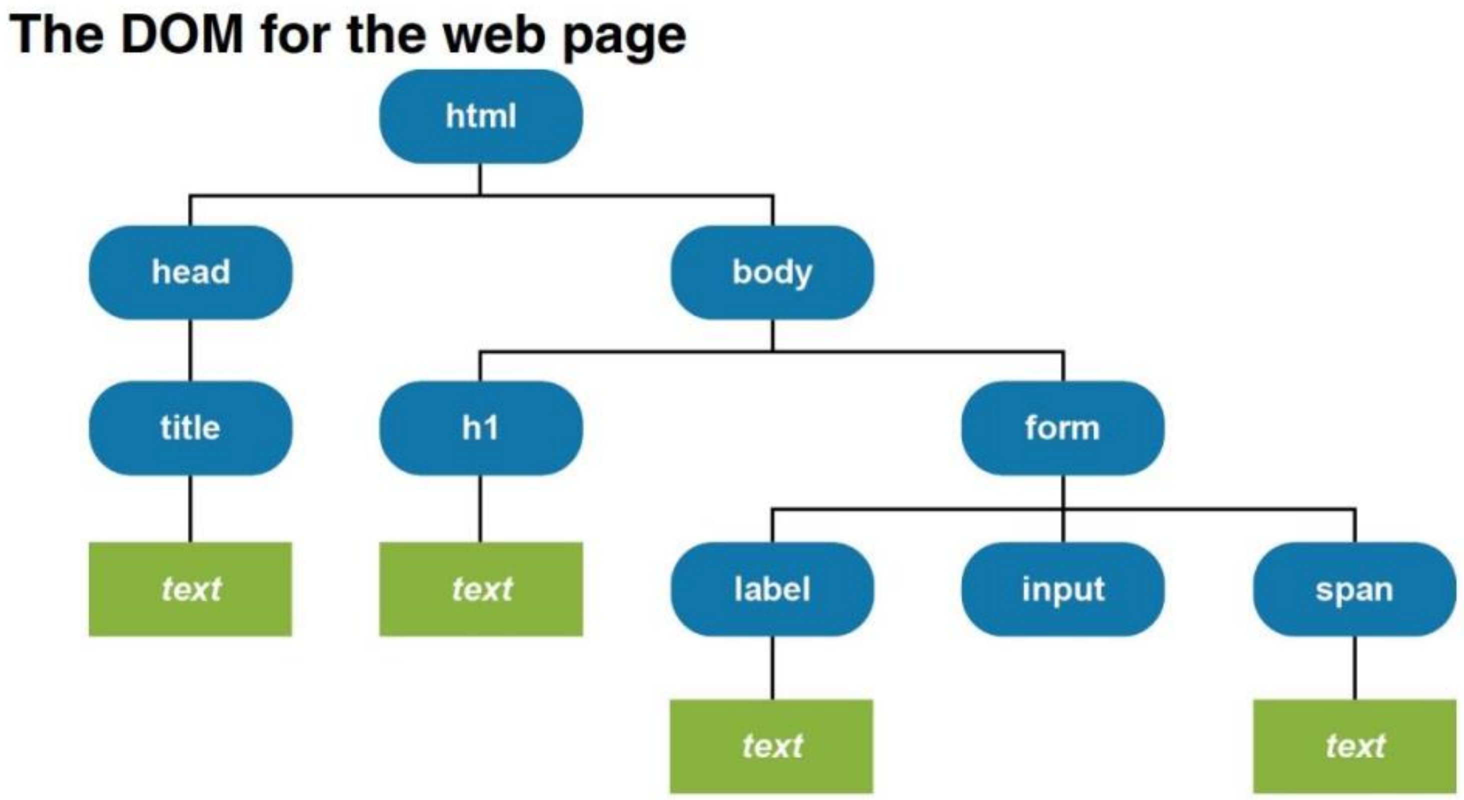
```
<html>
<head>
    <title>   </title>
</head>
<body>
   <p>This is paragraph number one</p>
   <h1>heading 1 is the largest level</h1>
</body>
```



DOM Tree: Example

```
<!DOCTYPE html>
<html>
<head>
        <title>Join Email List</title>
</head>
<body>
   <hl>Please join our email list</hl>
   <form id= "email_ form" name= "email form" action= " join.html"
      method= "get" >
   <label for= "email_address" >Email Address:</label>
   <input type= "text " id= "email_address" >
   <span id= "email_error " >*</span>
</form>
</body>
</html>
```

DOM Tree: Example



DOM Tree:

```
<a href="index.html">Home</a>
```

<a> is an element, which is a link to index.html.

a: is the tag
href: is the attribute
index.html: is the attribute value
Home: is the text.

HTML DOM Object Properties

- HTML DOM properties are values of HTML Elements that you can set or change.
- Html DOM properties:
  - x.innerHTML - the inner text value of x (a HTML element)
  - x. textContent- the text content of x
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x.

HTML DOM Object Properties

```
<body>
 <ul>
     <li id="myLI">Coffee</li>
     <li>Tea</li>
 </ul>
<p>The node name of the parent node of "myLI" is:</p>
<p id="p1"></p>
<script>
    let name =
    document.getElementById("myLI").parentNode.nodeName;
    document.getElementById("p1").innerHTML = name;
</script>
</body>
```

HTML DOM Object Properties

- The **childNodes** property will return a list of every child of a node.

```
<body>
<ul id="uu">
  <li id="myLI">Coffee</li>
  <li>Tea</li>
</ul>
<p>The children node name of the body are :</p>
<p id="p1"></p>
<script>
    let children="   "
    let childrenList=document.body.childNodes;
    for (let i=0; i<childrenList.length; i++){
      children+=childrenList[i].nodeName+ "<br>";  }
      document.getElementById("p1").innerHTML =children;
</script>
</body>
```

HTML DOM Object Methods

- HTML DOM methods are actions you can perform on HTML Elements. (like add or deleting an HTML element).
- Html DOM methods:
  - x.getElementById(id) - get the element with a specified id
  - x.getElementsByTagName(name) - get all elements with a specified tag name
  - x. createElement(element) – create an HTML element
  - x.appendChild(node) - insert a child node to x
  - x.removeChild(node) - remove a child node from x

Accessing Nodes

- You can access a node in three ways:

1. By using the getElementById() method
2. By using the getElementsByTagName() method
3. By navigating the node tree, using the node relationships

   - Use parentNode, childNodes, firstChild, lastChild properties.

```
var x=document.getElementById("intro");
var text=x.firstChild.nodeValue;
```

Accessing Nodes - Examples

```
x=document.getElementsByTagName("p");
for (i=0;i<x.length;i++) {
    document.write(x[i].innerHTML);
    document.write("<br />");  }
```

```
<h1 id="id1">My First Page</h1>
<p id="id2"></p>
<script>
document.getElementById("id2").innerHTML =
document.getElementById("id1").firstChild.nodeValue;
</script>
```

```
x=document.getElementsByTagName("p");
document.write(“Paragraph text: " + x[1].innerHTML);
```

Changing HTML - Examples

Change the background color

```
<html>
<body>
     <script>
          document.body.bgColor="yellow";
      </script>
</body>
</html>
```

Change the text of an element

```
<html>
<body>
   <p id="p1">Hello World!</p>
   <script>
             document.getElementById("p1").innerHTML="New text!";
    </script>
</body>
</html>
```

Adding HTML Elements

- use the “appendChild()” method for adding the HTML elements through JavaScript.

```
<body>

 <ul id="myList">
     <li>Coffee</li>
     <li>Tea</li>
 </ul>

<script>
  // Create an "li" node:
     const node = document.createElement("li");
  // Create a text node:
  const textnode = document.createTextNode("Water");
  // Append the text node to the "li" node:
  node.appendChild(textnode);
  // Append the "li" node to the list:
  document.getElementById("myList").appendChild(node);

</script>
```

Deleting HTML Elements

- Child nodes can be removed from a parent with **removeChild()**, and a node itself can be removed with **remove()**..

```
<body>
<button onclick="myFunction()">Remove</button>
<ul id="myList">
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
  </ul>

function myFunction() {
    const list = document.getElementById("myList");
    list.removeChild(list.firstElementChild); }
  </script>
```

```
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
<script>
const elmnt = document.getElementById("p1");
elmnt.remove();
</script>
```