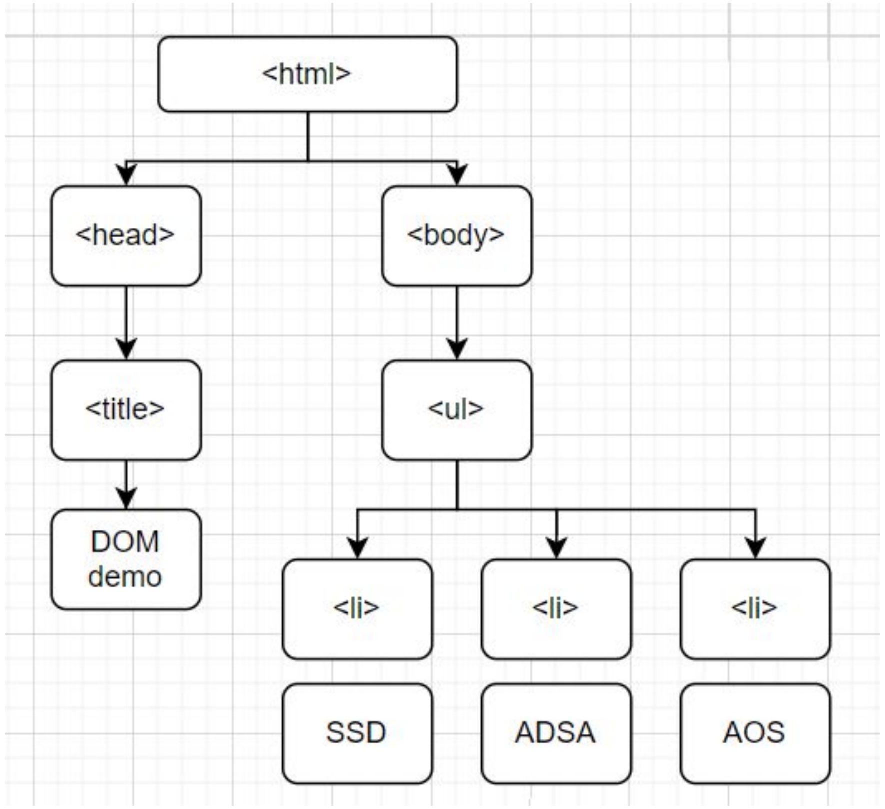
SSD LAB

DOM

What is DOM?

- Document Object Model
- API for HTML and XML documents
- Language neutral
- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- **HTML DOM - standard model for HTML documents**

DOM Visualization



Element Nodes : html, head, body, title, ul, li
Can have further child nodes

Text Nodes : DOM demo, SSD, ADSA, AOS
Cannot have further child nodes

DOM Methods

getElementById

Syntax:

```
document.getElementById( elementID)
```

- Returns the element that has the ID attribute with the specified value.
- Returns null if no matching element found
- elementID is case-sensitive
- document.getElementByID(“Main”) for accessing an element with id “main” will return null

getElementsByTagName

Syntax:

```
document.getElementsByTagName( tagname)
```

- Returns the collection of all the elements in the document with the given tag name
- The order of the elements in the collection is same as they appear in the tree.
- E.g. to find out how many list items are there in a list:
    - `var x = document.getElementsByTagName( "li").length;`

getElementsByClassName

Syntax:

```
document.getElementsByClassName(classname1 classname1 ...)
```

- Returns an HTMLCollection object, representing a collection of elements with the specified class name
- To search for multiple class names, separate them with spaces, like "class1 class2”
- E.g. to change the font color of all the element with class “example” to red

```
var exampleElements = document.getElementsByClassName("example");

var i;

for (i = 0; i < exampleElements.length; i++) {

  exampleElements[i].color = "red";

}
```

querySelectorAll

Syntax:

```
document.querySelectorAll(CSS selectors)
```

- Returns a **NodeList object**, representing all elements in the document that matches the specified CSS selector(s)..
- E.g. Get all <p> elements in the document with class="example"
  - `var x = document.querySelectorAll("p.example");`

document.createElement

Syntax:

```
document.createElement(nodename)
```

- Creates an Element Node with the specified name.
- E.g. Insert a new <p> node and some text to it:
  - `var newNode = document.createElement(“p”)`
  - `newNode.innerHTML = “Newly Created Node.”`
- Note that we also need to append the new node to the document object:
  - `document.body.appendChild(newNode);`

document.appendChild

Syntax:

```
node.appendChild(node)
```

- Adds a node to the end of the list of children of a specified parent node
- E.g. Add a new list item to the existing list
  - `var item = document.createElement("li")`
  - `item.textContent = "item2";`
  - `document.getElementById("list").appendChild(item);`

document.replaceChild

Syntax:

```
node.replaceChild(newnode, oldnode)
```

- Replaces a child node with a new node.
- E.g. Replace first item in a list with another item
    - `var list = document.getElementById('list');`
    - `var newItem = document.createElement('li');`
    - `newItem.textContent = 'New item';`
    - `list.replaceChild(newItem, menu.firstElementChild);`

document.removeChild

Syntax:

`node.removeChild(node)`

- Removes a specified child node of the specified element.
- E.g. Remove all the items in a list
    - `var list = document.getElementById('list');`
    - `while (list.firstChild) {`
    - `    list.removeChild(list.firstChild);`
    - `}`

Event Attributes

Onload

Syntax:

```
<element onload="myFun()">
```

- The onload event occurs when an object has been loaded.
- E.g. To check whether an image has been loaded or not
  - `<img src="img.jpg" onload="loadImage()">`
  - `<script>`
  - `function loadImage() {`
  - `  alert("The image is loaded");`
  - `}`
  - `</script>`

OnUnload

Syntax:

```
<element onunload="myFun">
```

- The onunload event occurs once a page has unloaded
- Triggers when a user:
  - Closes the browser
  - Reloads a page
  - Navigates away by clicking on a link, etc

Onchange

Syntax:

```
<element onchange="myFun()">
```

- Onchange event occurs when the value of an element has been changed.
- E.g. to alert when an option is changed:
    - `<select id="list" onchange="myFun()">`
    - `  <option value="Option1">Option1</option>`
    - `  <option value="Option2">Option2</option>`
    - `</select>`
    - `function myFun() {`
    - `alert(“ You changed the option “)`
    - `}`

onmouseover

Syntax:

```
<element onmouseover="myFun()">
```

- Occurs when the mouse pointer is moved onto an element, or onto one of its children.
- E.g. change the color of a div on mouse over
  - `<div width="32" height="32" class="div" onmouseover=myFun(this)>`
  - `</div>`
  - `function myFun(x) {`
  - `x.style.background="red"`
  - `}`

onmouseout

Syntax:

```
<element onmouseout="myFun()">
```

- occurs when the mouse pointer is moved out of an element, or out of one of its children.
- E.g. change the color of a div back to white on mouse out
    - `<div width=”32” height=”32” class=”div” onmouseout=myFun(this)>`
    - `</div>`
    - `function myFun(x) {`
    - `this.style.background=”white”`
    - `}`

onmousedown

Syntax:

```
<element onmousedown="myFun()">
```

- occurs when a user **presses** a mouse button over an element.

onmouseup

Syntax:

```
<element onmouseup="myFun()">
```

- occurs when a user **releases** mouse button over an element.

onclick

Syntax:

```
<element onclick="myFun()">
```

- occurs when the user clicks on an element
- E.g. create a button that alerts today's time and date on clicking
  - `<button onclick="myFun()">Click Me</button>`
  - `Function myFun(){`
  - `alert(Date())`
  - `}`
- Note the difference between onmouseup, onmousedown and onclick