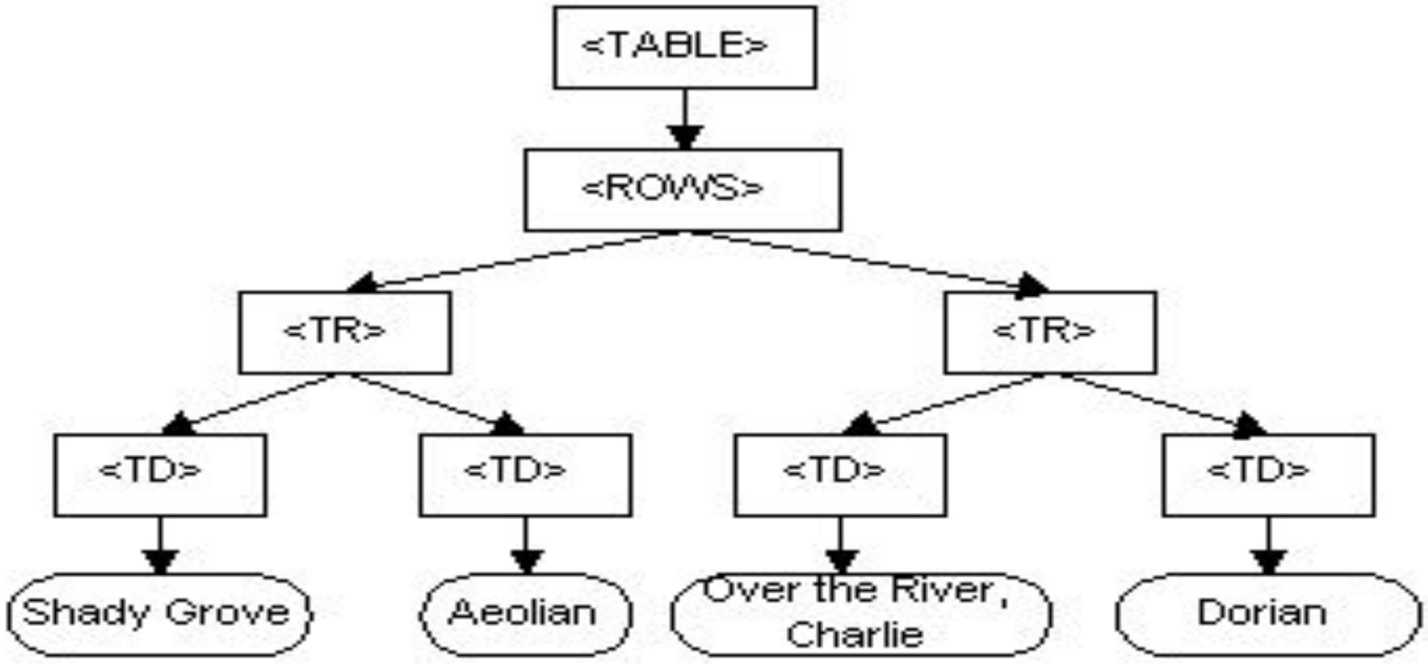
Document object model

- We have studied HTML, created your first tags, learned about CSS, made beautiful forms, amazing buttons, responsive pages and have started to show everyone how amazing all that was.
- But then you decide that you want to take another step in your learning, and you've started wonder to yourself: "How can I add animation to my page? I wish that button made some animation on my page when I clicked it!"
- Well, that's where the DOM comes to solve your problem. You've probably heard a lot about it, but you might not know yet what is it and what problems it solves. So let's dig in.

Definition

- The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.
- The Document Object Model is a **cross-platform** and **language-independent** application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document. The DOM model represents a document with a logical tree.

- <TABLE>
- <ROWS>
- <TR>
- <TD>Shady Grove</TD>
- <TD>Aeolian</TD> </TR>
- <TR> <TD>Over the River, Charlie</TD> <TD>Dorian</TD>
- </TR>
- </ROWS>
- </TABLE>


<TABLE>
<ROWS>
<TR>
<TR>
<TD>
<TD>
<TD>
<TD>
Shady Grove
Aeolian
Over the River, Charlie
Dorian

DOM tree structure

- The **document object** represents the whole html document.
- When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

- As mentioned earlier, it is the object of window. So
- **window.document**
- Is same as
- **document**

- **Window Object:** Window Object is at always at top of hierarchy.
- **Document object:** When HTML document is loaded into a window, it becomes a document object.
- **Form Object:** It is represented by ***form*** tags.
- **Link Objects:** It is represented by ***link*** tags.
- **Anchor Objects:** It is represented by ***a href*** tags.
- **Form Control Elements:**: Form can have many control elements such as text fields, buttons, radio buttons, and checkboxes, etc.

- ***Methods* of Document Object**:
- **write("string"):** writes the given string on the document.
- **getElementById():** returns the element having the given id value.
- **getElementsByName():** returns all the elements having the given name value.
- **getElementsByTagName():** returns all the elements having the given tag name.
- **getElementsByClassName():** returns all the elements having the given class name.

What is DOM in JavaScript?

- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like discussed earlier.
-

```
<html>
<head>
  <title>DOM!!!</title>
</head>
<body>
  <h1 id="one">Welcome</h1>
  <p>This is the welcome message.</p>
  <h2>Technology</h2>
  <p>This is the technology section.</p>
  <script type="text/javascript">
        var text = document.getElementById("one").innerHTML;
        alert("The first heading is " + text);
  </script>
</body>
</html>
```

Javascript - document.getElementById() method

- <html><body>
- <script type="text/javascript">
- function getcube(){
- var number=document.getElementById("number").value;
- alert(number*number*number);
- }
- </script>
- <form>
- Enter No:<input type="text" id="number" name="num"/><br/>
- <input type="button" value="cube" onclick="getcube()"/>
- </form> </body></html>

Javascript - document.getElementsByName() method

- The **document.getElementsByName()** method returns all the element of specified name.
- The syntax of the getElementsByName() method is given below:
- document.getElementsByName("name")

```
<html><body>
<script type="text/javascript">
function totalelements()
{
var allgenders=document.getElementsByName("gender");
alert("Total Genders:"+allgenders.length);
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">

<input type="button" onclick="totalelements()" value="Total Genders">
</form> </body>
</html>
```

Output= 2

document.getElementsByTagName() method

```
<html><body>
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);

}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs.

method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button> </body>
</html>
```

Output=3

Example

- <!DOCTYPE html>
- <html>
- <body>
- <h2>What Can JavaScript Do?</h2>
- <p>JavaScript can change HTML attribute values.</p>
- <p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>
- <img id="myImage" src="bulboff.gif" style="width:100px">
- <button onclick="document.getElementById('myImage').src='bulbon.gif'">Turn on the light</button>
- <button onclick="document.getElementById('myImage').src='bulboff.gif'">Turn off the light</button>
- </body>
- </html>

HTML/DOM events for JavaScript

- **Onclick**==occurs when element is clicked
- **clicked.ondblclick**==occurs when element is double-clicked.
- **onmouseover**==occurs when mouse is moved over an element.
- **onmouseou**t==occurs when mouse is moved out from an element (after moved over).
- **onmousedown**==occurs when mouse button is pressed over an element.
- **onmouseup**==occurs when mouse is released from an element (after mouse is pressed).

Ommouseover and onmouseout

- <html>
- <body>

- <h1 onmouseover="style.color='red'"
- onmouseout="style.color='black'" >Mouse over this text</h1>

- </body>
- </html>

Onmousedown AND onmouseup

- **<!DOCTYPE html>**
- **<html>**
- **<body>**
- **<p id="myP" onmousedown="mouseDown()" onmouseup="mouseUp()">**
- **Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph, and sets the color of the text to red. The mouseUp() function is triggered when the mouse button is released, and sets the color of the text to green.**
- **</p>**
- **<script>**
- **function mouseDown() {**
- **document.getElementById("myP").style.color = "red";**
- **}**
- **function mouseUp() {**
- **document.getElementById("myP").style.color = "green";**
- **}**
- **</script>**
- **</body>**

What is Lazy Loading?

- The concept of lazy loading assists in loading only the required section and delays the remaining, until it is needed by the user.
- For example, say a user requests for the logo of GeeksForGeeks from a search engine. The entire web page, populated with the requested content, is loaded. Now if the user opens the first image and is satisfied with it, he will probably close the web page thus, rest of the images so loaded will be left unseen. This will result in the wastage of the resources so consumed in the bulk load of that page. Thus the solution to this is Lazy Loading.

Advantages of Lazy loading:

- On-demand loading reduces time consumption and memory usage thereby optimizing content delivery. As only a fraction of the web page, which is required, is loaded first thus, the time taken is less and the loading of rest of the section is delayed which saves storage. All of this enhances the user's experience as the requested content is fed in no time.
- Unnecessary code execution is avoided.
- Optimal usage of time and space resources makes it a cost-effective approach from the point of view of business persons. (website owners)

Disadvantages of Lazy loading:

- Firstly, the extra lines of code, to be added to the existing ones, to implement lazy load makes the code a bit complicated.
- Secondly, lazy loading may affect the website’s ranking on search engines sometimes, due to improper indexing of the unloaded content.