Name: Sciddhanto Sinha

Roll no: 2213111

Batch: A

Experiment No:5

5(a) Write a Javascript code to append static row to the table on click of button.

Objective: To learn about concept and implementation of JAVASCRIPT

Theory:

What is Javascript. Why it is called Client side Scripting language Client side: JavaScript is a client-side language, which means it gets executed at the client side (i.e, user side). On contrary, PHP is a server-side scripting language, as it gets executed at server. Whenever you browse the web, all the HTML,CSS & JS files are fetched from the server & then executed/interpreted at your side by your browser.Scripting language: Since it is interpreted rather than compiled & are úsed to execute tasks one-by-one. More professionally, A scripting or script language is a programming language that supports scripts: programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted

Explain Javascript functions used in above programs like with syntax and example:-

getElementByID()

The getElementById() method returns an element with a specified value. The getElementById() method returns null if the element does not exist.
document.getElementById("demo").style.color
= "red";

•innerHTML()

The innerHTML property sets or returns the HTML content (inner HTML) of an element.
let html
= document.getElementById("myList").innerHTML;

value()

The Object.values() method returns an array of a given object's own enumerable property values, in the same order as that provided by a for...in loop. (The only difference is that a for...in loop enumerates properties in the prototype chain as well.)
console.log(Object.values(object1))

•parseInt()

The parseInt() function parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems). function roughScale(x, base) { const parsed = parseInt(x, base); if (isNaN(parsed)) { return 0; } return parsed * 100;

}

DOM

- It is an application programming interface.
- Dom represents a document as a hierarchical tree of nodes allowing dev to add, remove, and modify individual part of the page.
- Document node represents every document as a root. In this example only child of document node is html which is called as document element.
- Every piece of markup can be represented by a node in the tree: HTML elements are represented by element nodes, attributes are represented by attribute nodes, the document type is represented by a document type node, and comments are represented by comment nodes.
- In total, there are 12 node types, all of which inherit from a base type.
- DOM Level 1 describes an interface called Node that is to be implemented by all node types in the DOM.
- Every node has a nodeType property that indicates the type of node that it is. Node types are represented by one of the following 12 numeric constants on the Node type:

a. Node.ELEMENT_NODE (1)
b. Node.ATTRIBUTE_NODE (2)
c. Node.TEXT_NODE (3)
d. Node.CDATA_SECTION_NODE (4)
e. Node.ENTITY_REFERENCE_NODE (5)
f. Node.ENTITY_NODE (6)
g. Node.PROCESSING_INSTRUCTION_NODE (7)
h. Node.COMMENT_NODE (8)
i. Node.DOCUMENT_NODE (9)
j. Node.DOCUMENT_TYPE_NODE (10)
k. Node.DOCUMENT_FRAGMENT_NODE (11)
l. Node.NOTATION_NODE (12)

```
    if (someNode.nodeType == Node.ELEMENT_NODE)
    {
      alert(“Node is an element.”); //won’t work in IE < 9
}
```

This example compares the someNode.nodeType to the Node.ELEMENT_NODE constant. If they’re equal, it means someNode is actually an element. Unfortunately, since Internet Explorer 8 and earlier doesn’t expose the Node type constructor, this code will cause an error. For cross-browser compatibility, it’s best to compare the nodeType property against a numeric value, as in the following:

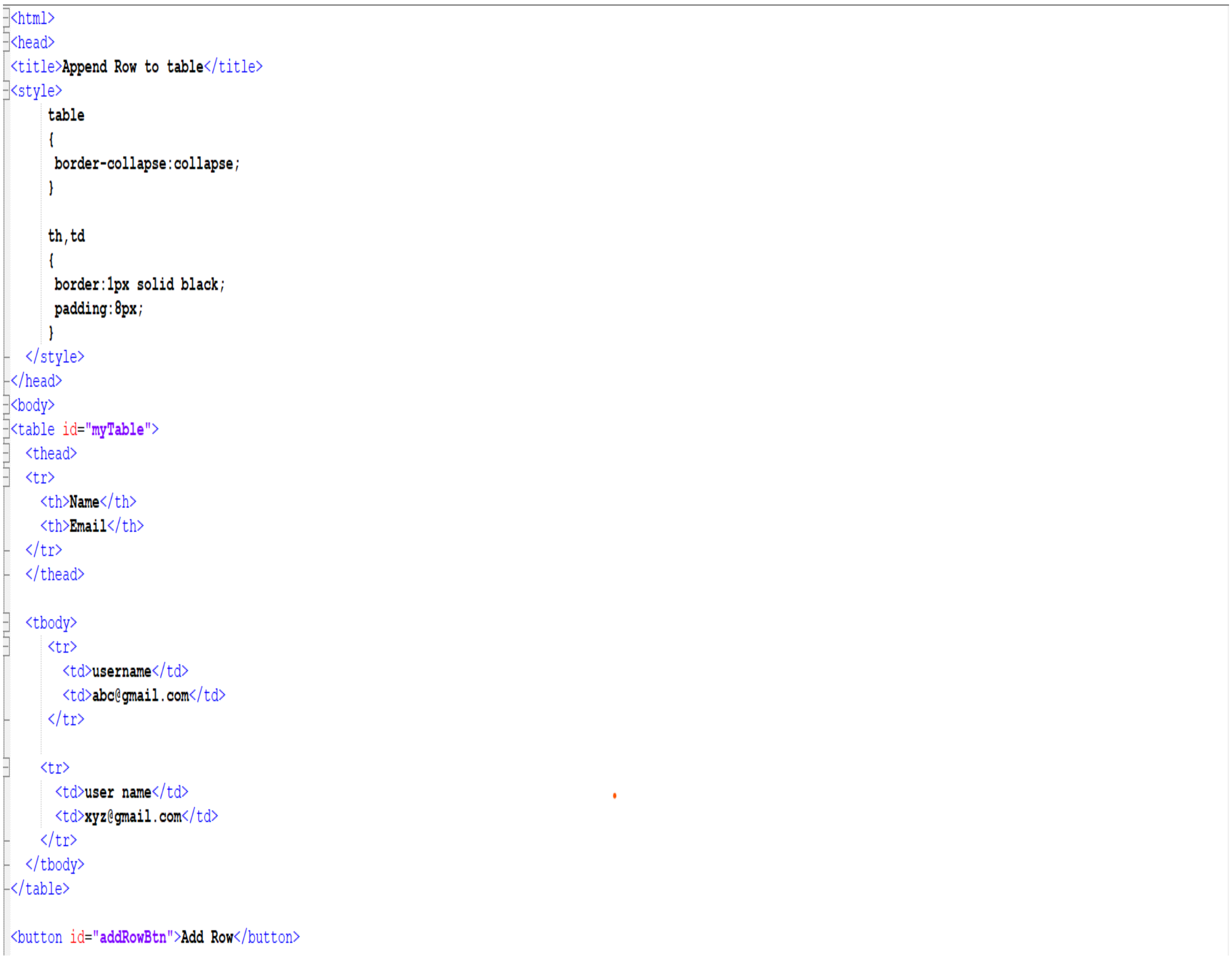
```
        if (someNode.nodeType == 1)
             {
                 alert(“Node is an element.”); //works in all browsers

  }
```

Properties of note type

- Two properties, nodeName and nodeValue, give specific information about the node. The values of these properties are completely dependent on the node type.
- nodeName is always equal to the element’s tag name, and nodeValue is always null.
- Each node has a childNodes property containing a NodeList. A NodeList is an array- like object used to store an ordered list of nodes that are accessible by position. Keep in mind that a NodeList is not an instance of Array even though its values can be accessed using bracket notation and the length property is present.
- Node Relationship child parent

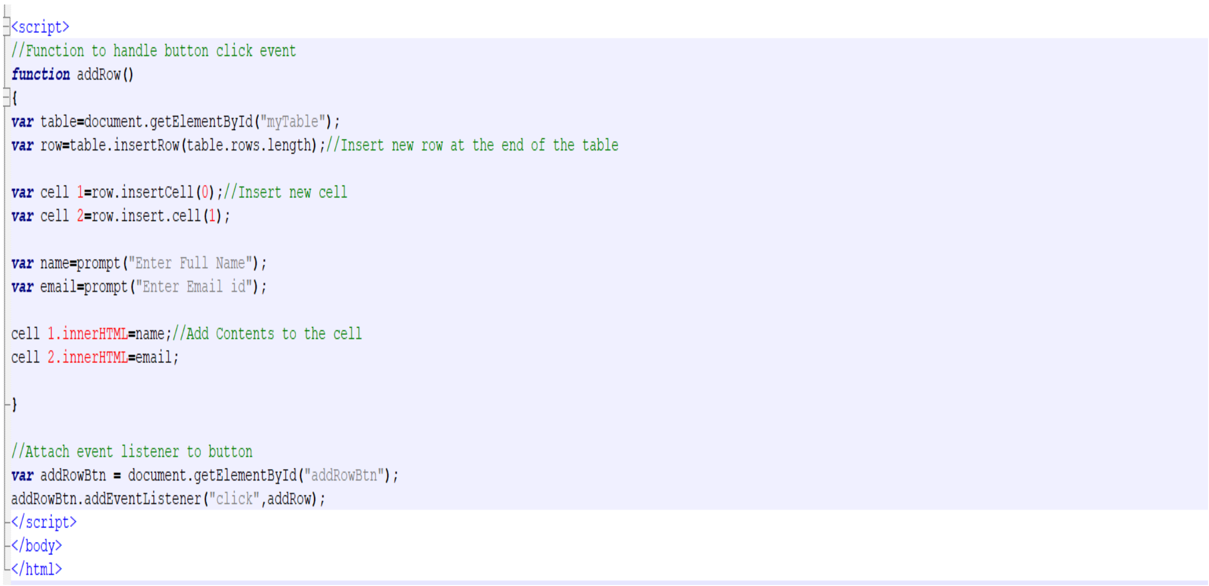
Conclusion:

Thus, we have successfully implemented the program



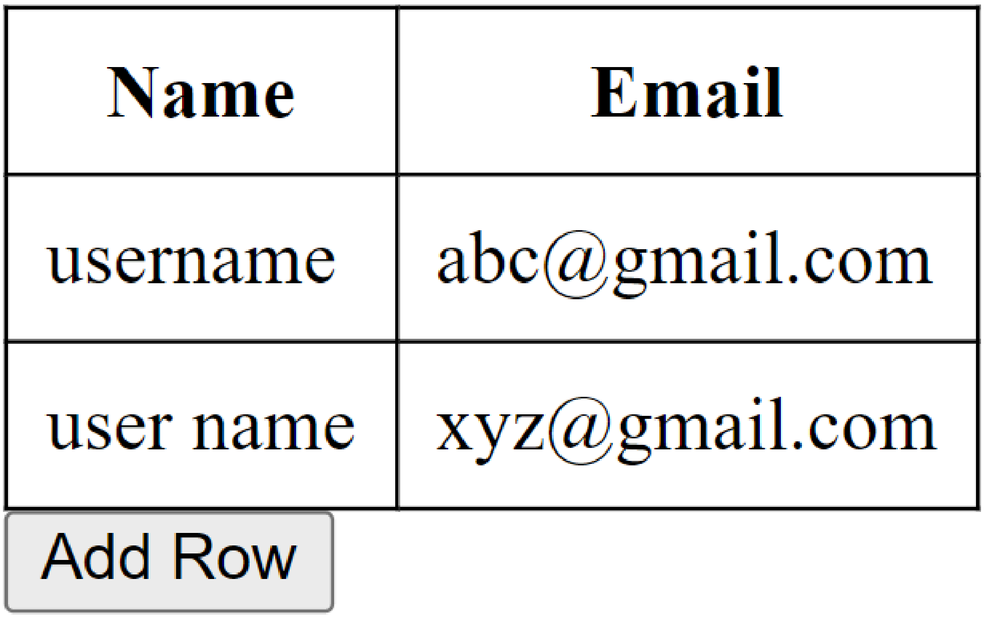
```
<html>
<head>
<title>Append Row to table</title>
<style>
     table
     {
      border-collapse:collapse;
     }

     th,td
     {
      border:1px solid black;
      padding:8px;
     }
  </style>
</head>
<body>
<table id="myTable">
  <thead>
  <tr>
    <th>Name</th>
    <th>Email</th>
  </tr>
  </thead>

  <tbody>
     <tr>
       <td>username</td>
       <td>abc@gmail.com</td>
     </tr>

    <tr>
      <td>user name</td>
      <td>xyz@gmail.com</td>
    </tr>
  </tbody>
</table>

<button id="addRowBtn">Add Row</button>
```

Code:



```
<script>
//Function to handle button click event
function addRow()
{
var table=document.getElementById("myTable");
var row=table.insertRow(table.rows.length);//Insert new row at the end of the table

var cell 1=row.insertCell(0);//Insert new cell
var cell 2=row.insert.cell(1);

var name=prompt("Enter Full Name");
var email=prompt("Enter Email id");

cell 1.innerHTML=name;//Add Contents to the cell
cell 2.innerHTML=email;

}

//Attach event listener to button
var addRowBtn = document.getElementById("addRowBtn");
addRowBtn.addEventListener("click",addRow);
</script>
</body>
</html>
```

Output (Screenshots):



Name	Email
username	abc@gmail.com
user name	xyz@gmail.com