

**Name: Sciddhanto Sinha**

**Roll no: 2213111**

**Batch: A**

### Experiment No:4

4) (e) Create an array using JavaScript and display the occurrences of a specific character [For example; arr = ['a', 'b', 'a', 'c', 'z'] Output should be occurrences of a is 2]

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 used 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 node 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.

### Code:

```
<html>
<head>
  <title>Character Occurrences</title>
</head>
<body>
<script>
  var array = ['a', 'b', 'a', 'c', 'z'];
  var targetCharacter = 'a';

  var occurrences = array.reduce(function(count, char) {
    if (char === targetCharacter) {
      return count + 1;
    }
    return count;
  }, 0);

  document.write("<h2>Occurrences of '" + targetCharacter + "' is " + occurrences + "</h2>");
</script>
</body>
</html>
```

### Output(Screenshots):

---

**Occurrences of 'a' is 2**

