MODULE: 4 (JavaScript Basic & DOM)
(Assignment)

1.What is JavaScript?

JavaScript is a cross-platform, object-oriented scripting language developed by Netscape. JavaScript was created by Netscape programmer Brendan Eich.

It was first released under the name of LiveScript as part of Netscape Navigator 2.0 in September 1995. It was renamed JavaScript on December 4, 1995. As JavaScript works on the client side, It is mostly used for client-side web development.

JavaScript is designed for use on web pages and closely integrated with HTML. JavaScript can create applications which run in the browsers such as IE, Opera, FireFox, Google Chrome and other. Netscape submitted JavaScript to ECMA International for standardization resulting in the standardized version named ECMAScript.

2.What is the use of isNaN function

The isNaN() function is used to check whether a given value is an illegal number or not. It returns true if value is a NaN else returns false. It is different from the Number.isNaN() Method.

Syntax:

```
isNaN( value
```

Parameter Values: This method accepts single parameter as mentioned above and described below:

- value: It is a required value passed in the isNaN() function.

Return Value: It returns a Boolean value i.e. returns true if the value is NaN else returns false.

```
<script>
    document.write(isNaN(12) + "<br>");
    document.write(isNaN(0 / 0) + "<br>");
    document.write(isNaN(12.3) + "<br>");
    document.write(isNaN("Geeks") + "<br>");
    document.write(isNaN("13/12/2020") + "<br>");
    document.write(isNaN(-46) + "<br>");
    document.write(isNaN(NaN) + "<br>");
</script>
```

Output:

```
false
true
false
true
True
```

```
false
```

```
true
```

3.What is negative Infinity?

The negative infinity in JavaScript is a constant value which is used to represent a value which is the lowest available. This means that no other number is lesser than this value. It can be generated using a self-made function or by an arithmetic operation.

Negative infinity is different from mathematical infinity in the following ways:

1. Negative infinity results in 0 when divided by any other number.
2. When divided by itself or positive infinity, negative infinity return NaN
3. Negative infinity, when divided by any positive number (apart from positive infinity) is negative infinity.
4. Negative infinity, divided by any negative number (apart from negative infinity) is positive infinity.
5. If we multiply negative infinity with NaN, we will get NaN as a result.
6. The product of NaN and negative infinity is 0.

```
<!DOCTYPE html>
<html>
<body>
        <style>
                h1 {
                color: green }
```

```
</style>
<h1>GeeksforGeeks</h1>
        <h1>
        What is negative infinity in JavaScript?
</h1>
        <button onclick="geekNegativeInfinity()">
        Generate negative infinite
</button>
<p id="geek"></p>
<script>
function geekNegativeInfinity() {
                        //negative value greater than the
                        //largest representable number in JavaScript
                        var n = (-Number.MAX_VALUE) * 2;
                        document.getElementById("geek").innerHTML = n;
                }
        </script>
</body>
</html>
```

4.Which company developed JavaScript?

Netscape is a software company who developed JavaScript.

5.• What are undeclared and undefined variables?

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

6.Write the code for adding new elements dynamically?

```
<html>
    <head>
        <title>t1</title>
        <script type="text/javascript">
            function addNode() {
                var newP = document.createElement("p");
                var textNode = document.createTextNode(" This
is a new text node");
                newP.appendChild(textNode);

document.getElementById("firstP").appendChild(newP);
            }
    </script>
    </head>
    <body>
        <p id="firstP">firstP<p>
    </body>
</html>
```

7.What is the difference between ViewState and SessionState

ViewState' is specific to a page in a session.

'SessionState' is specific to user specific data that can be accessed across all pages in the web application.

8.- What is === operator?

=== is called as strict equality operator which returns true when the two operands are having the same value without any type conversion.

9. How can the style/class of an element be changed?

It can be done in the following way:

```
document.getElementById("myText").style.fontSize = "20?;
```

or

```
document.getElementById("myText").className = "anyclass";
```

10.How to read and write a file using JavaScript?

On the client side, you can't read or write files in JavaScript browsers. The fs module in Node.js may be used to accomplish this on the server-side. It has methods for reading and writing files on the file system that are both synchronous and asynchronous. Let's demonstrate some examples of reading and writing files with the node.js fs module.

The fs.readFile() and rs.writeFile() methods are used to read and write of a file using javascript. The file is read using the fs.readFile() function, which is an inbuilt method. This technique reads the full file into memory and stores it in a buffer.

Syntax:

```
fs.readFile( file_name, encoding, callback_function )
```

Parameters:

- filename: It contains the filename to be read, or the whole path if the file is saved elsewhere.
- encoding: It stores the file's encoding. 'utf8' is the default setting.
- callback function: This is a function that is invoked after the file has been read. It requires two inputs:
- err: If there was an error.
- data: The file's content.
- Return Value: It returns the contents contained in the file, as well as any errors that may have occurred.

The fs.writeFile() function is used to write data to a file in an asynchronous manner. If the file already exists, it will be replaced.

Syntax:

```
fs.writeFile( file_name, data, options, callback )
```

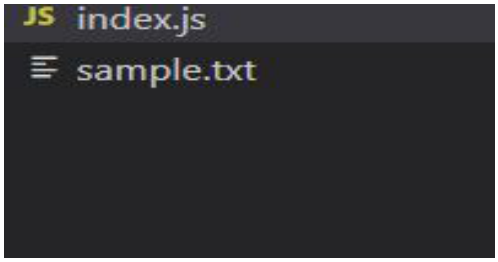
Parameters:

- file_name: It's a string, a buffer, a URL, or a file description integer that specifies the location of the file to be written. When you use a file descriptor, it will function similarly to the fs. write() method.
- data: The data that will be sent to the file is a string, Buffer, TypedArray, or DataView.
- options: It's a string or object that may be used to indicate optional output options. It includes three more parameters that may be selected.
- encoding: It's a string value that indicates the file's encoding. 'utf8' is the default setting.
- mode: The file mode is specified by an integer number called mode. 0o666 is the default value.

- flag: This is a string that indicates the file-writing flag. ‘w’ is the default value.
- callback: This function gets invoked when the method is run.
- err: If the process fails, this is the error that will be thrown.

Let’s understand how to write and read files using an example:

In the below example, we are creating a file using the writeFile() method and reading the content of the file readFile() method.




```
var fs = require("fs");
console.log(" Writing into an file ");

// Sample.txt is an empty file
fs.writeFile(
"sample.txt",
"Let's write a few sentences in the file",
function (err) {
        if (err) {
        return console.error(err);
        }

        // If no error the remaining code executes
        console.log(" Finished writing ");
        console.log("Reading the data that's written");

        // Reading the file
        fs.readFile("sample.txt", function (err, data) {
        if (err) {
                return console.error(err);
        }
        console.log("Data read : " + data.toString());})});
```

11.What are all the looping structures in JavaScript?

Following are looping structures in Javascript:

- For
- While
- do-while loops

12.How can you convert the string of any base to an integer in JavaScript?

The parseInt function available in JavaScript has the following signature –

```
parseInt(string, radix);
```

Where, the paramters are the following –

String – The value to parse. If this argument is not a string, then it is converted to one using the ToString method. Leading whitespace in this argument is ignored.

Radix – An integer between 2 and 36 that represents the radix (the base in mathematical numeral systems) of the string.

So we can pass the string and the radix and convert any numbner with base from 2 to 36 to integer using this method.

Example

```
console.log(parseInt("100", 10))

console.log(parseInt("10", 8))

console.log(parseInt("101", 2))

console.log(parseInt("2FF3", 16))
```

```
console.log(parseInt("ZZ", 36))
```

Output

```
100


8


5


12275


1295
```

13• What is the function of the delete operator?

The delete operator removes a property from an object. If the property's value is an object and there are no more references to the object, the object held by that property is eventually released automatically.

```
const Employee = {
  firstname: 'John',
  lastname: 'Doe'
};

console.log(Employee.firstname);
// expected output: "John"

delete Employee.firstname;

console.log(Employee.firstname);
// expected output: undefined
```

Output

```
> "John"
> undefined
```

Parameters

object
The name of an object, or an expression evaluating to an object.

property
The property to delete.

Return value

true for all cases except when the property is an own non-configurable property, in which case false is returned in non-strict mode.

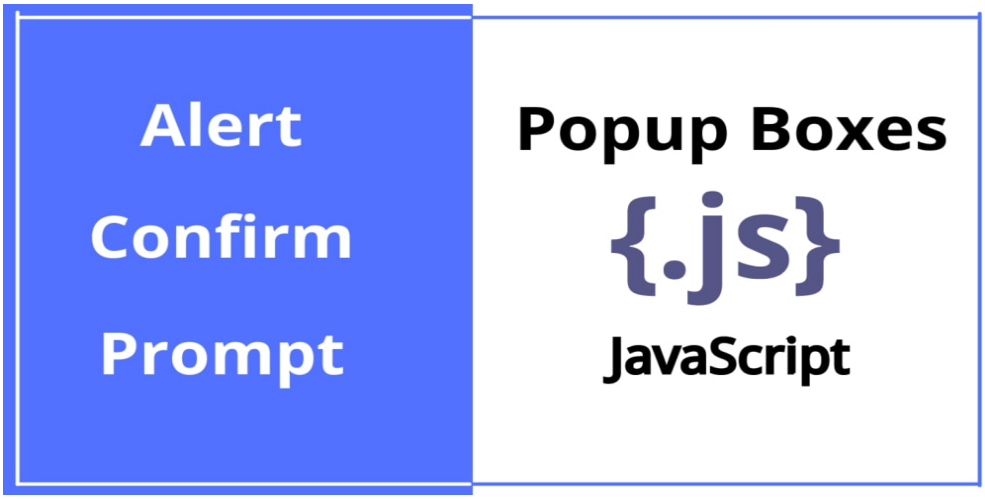
Exceptions

TypeError
Thrown in strict mode if the property is an own non-configurable property.

ReferenceError
Thrown if object is super.

14.What are all the types of Pop up boxes available in JavaScript?

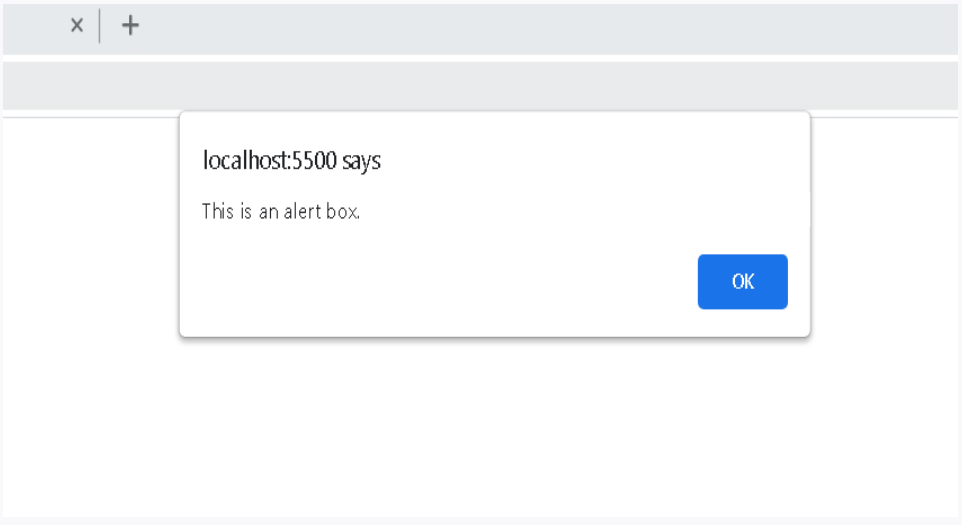
- Alert
- Confirm and
- Prompt



`alert(message);`

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta
http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport"
content="width=device-width,
initial-scale=1.0">
<title>Document</title>
</head>
<body>
<script>
window.alert("This is an
alert box.");
</script>
</body>
</html>
```

Output



15.What is the use of Void (0)?

Void(0) is used to prevent the page from refreshing and parameter “zero” is passed while calling.

Void(0) is used to call another method without refreshing the page.

16.How can a page be forced to load another page in JavaScript

The following code has to be inserted to achieve the desired effect:

```
<script language="JavaScript" type="text/javascript" >
  <!-- location.href="http://newhost/newpath/newfile.html";
//-->
</script>
```

17.What are the disadvantages of using innerHTML in JavaScript?

If you use *innerHTML* in JavaScript the disadvantage is

Content is replaced everywhere

We cannot use like “appending to *innerHTML*“

Even if you use +=like “*innerHTML = innerHTML + ‘html’*” still the old content is replaced by html

The entire *innerHTML* content is re-parsed and build into elements, therefore its much slower

The *innerHTML* does not provide validation and therefore we can potentially insert valid and broken HTML in the document and break it