**(Q.1) What is JavaScript?**

* JavaScript decides how your website behaves. JavaScript is client-side scripting language. It is written in script tag in html.

Ex. <script>

………code

</script

**(Q.2) What is the use of is NaN function?**

* In JavaScript NaN is short for "Not-a-Number.” The isNaN() method returns true if a value is NaN. The isNaN() method converts the value to a number before testing it.
* isNaN() method returns true if a value is Not-a-Number. Number.isNaN() returns true if a number is Not-a-Number.
* In other words:
* isNaN() converts the value to a number before testing it.
* Syntax: isNaN(value)

Ex. (a) This returns true

isNaN(‘Uday’);

(b)This returns false

Number.isNaN(‘Uday’);

**(Q.3) What is negative Infinity?**

* Number.NEGATIVE\_INFINITY returns negative infinity. Number.NEGATIVE\_INFINITY is "a number lower than any other number".
* NEGATIVE\_INFINITY is a property of the JavaScript Number object. You can only use it as Number.NEGATIVE\_INFINITY.

Ex. (a) let x = Number.NEGATIVE\_INFINITY;

(b) Using x.NEGATIVE\_INFINITY, where x is a variable, will return undefined.

let x = 100;  
 x.NEGATIVE\_INFINITY;

**(Q.4) Which company developed JavaScript?**

* JavaScript was invented by Brendan Eich in 1995.
* It was developed for Netscape 2, and became the ECMA-262 standard in 1997.
* After Netscape handed JavaScript over to ECMA, the Mozilla foundation continued to develop JavaScript for the Firefox browser.

**(Q.5) What are undeclared and undefined variables?**

* Undefined: It occurs when a variable has been declared but has not been assigned any value. Undefined is not a keyword.

Ex. var demo;

Undefined

console.log(demo);

* Undeclared: It occurs when we try to access any variable that is not initialized or declared earlier using the var or const keyword. If we use ‘typeof’ operator to get the value of an undeclared variable, we will face the runtime error with the return value as “undefined.” The scope of the undeclared variables is always global.

Ex. ReferenceError: myVariable is not defined

Console.log(myVariable);

**(Q.6) Write the code for adding new elements dynamically?**

<!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>

    <h1>Hello</h1>

    <div id="mydiv"></div>

    <button onclick="AddEvent()">Add Element</button>

</body>

</html>

<script>

    function AddEvent()

    {

        a = document.getElementById('mydiv');

        a.innerHTML = "I am Uday Kaneriya";

    }

</script>

**(Q.7) What is the difference between ViewState and SessionState?**

|  |  |
| --- | --- |
| ViewState | SessionState |
| Maintained at page level only. | Maintained at session level. |
| View state can only be visible from a single page and not multiple pages. | Session state value availability is across all pages available in a user session. |
| It will retain values in the event of a postback operation occurring. | In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration. |
| Information is stored on the client’s end only. | Information is stored on the server. |
| used to allow the persistence of page-instance-specific data. | used for the persistence of user-specific data on the server’s end. |
| ViewState values are lost/cleared when new page is loaded. | SessionState can be cleared by programmer or user or in case of timeouts. |

**(Q.8) What is === operator?**

* === equal value and equal type
* X === 5 true
* X === “5” false

Ex. (a) true

<p id="demo"></p>

<script>

let x = 5;

document.getElementById("demo").innerHTML = (x === 5);

</script>

(b) false

<p id="demo"></p>

<script>

let x = 5;

document.getElementById("demo").innerHTML = (x === "5");

</script>

**(Q.9) How can the style/class of an element be changed?**

<!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>

    <style>

        .x

        {

            background-color: aquamarine;

            color: rebeccapurple;

            padding: 20px 50px;

            font-size: 120px;

            font-style: italic;

        }

    </style>

</head>

<body>

    <h1 id="colorH1" > heading 1 </h1>

    <h4 id="colorH4" > heading 4 </h4>

    <script>

        var h1 = document.getElementById("colorH1");

        var h4 = document.getElementById("colorH4");

        document.write(h1.id)

        h1.style.background = "blue";

        document.getElementById("colorH1").style.color ="#fff"

        h1.style.padding="23px 45px ";

        h1.style.fontSize ="120px"

        h1.classList.add('x');

        h4.classList.add('x');

        h1.classList.remove('x');

        h1.classList.add('x');

    </script>

</body>

</html>

**(Q.10) How to read and write a file using JavaScript?**

* The[fs.readFile()](https://www.geeksforgeeks.org/node-js-fs-readfile-method/) and [rs.writeFile()](https://www.geeksforgeeks.org/node-js-fs-writefile-method/) 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)

* 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)

Ex.

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());

});

}

);

**(Q.11) What are all the looping structures in JavaScript?**

Types:

    1. for loop / for in / for of / forEach

    2. while loop

    3. Do while loop

    4. nested loop

1. For loop

Syntax:

        for(initialization;condition;inc/dec)

        {

            ..code;

        }

2. while

initialization;

while(condition)

{

    ..code;

    inc/dec;

}

3. do while

initialization;

do

{

    ……code;

    inc/dec;

}while(condition);

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

* The parseInt method parses a value as a string and returns the first integer.
* A radix parameter specifies the number system to use:
* 2 = binary, 8 = octal, 10 = decimal, 16 = hexadecimal.
* If radix is omitted, JavaScript assumes radix 10. If the value begins with "0x", JavaScript assumes radix 16.
* Syntax:

parseInt(string, radix)

Ex.

<script>

document.getElementById("demo").innerHTML =

parseInt("10") + "<br>" +

parseInt("10.00") + "<br>" +

parseInt("10.33") + "<br>" +

parseInt("34 45 66") + "<br>" +

parseInt(" 60 ") + "<br>" +

parseInt("40 years") + "<br>" +

parseInt("He was 40");

</script>

**(Q.13) What is the function of the delete operator?**

* The JavaScript [pop()](https://www.geeksforgeeks.org/javascript-array-pop-method/), [shift()](https://www.geeksforgeeks.org/javascript-array-shift-method/), or [splice()](https://www.geeksforgeeks.org/javascript-array-splice-method/) methods are available to delete an element from an array. But because of the key-value pair in an object, deleting is more complicated. Note that, the delete operator only works on objects and not on variables or functions.
* Syntax:

delete object

or

delete object.property

or

delete object['property']

Ex.

let emp = {

firstName: "Uday",

lastName: "Kumar",

salary: 40000

}

console.log(delete emp.salary);

console.log(emp);

**(Q.14) What are all the types of Pop up boxes available in JavaScript?**

* JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.
* Alert Box:

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

* Syntax:

window.alert(“Some text”);

* Confirm Box:

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

* Syntax:

window.confirm(“Some Text”);

* Prompt Box:

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

* Syntax:

window.prompt(“sometext”,”defaultText”);

Ex.

<h3 id="getText"></h3>

<!-- this is internal JS which we need to place just above the closing body tag -->

    <script type="text/javascript">

        document.write("Hello Uday");

        console.log("Hello Patel");

        document.getElementById('getText').innerHTML = " this is h3 tag";

        window.alert("this is alert BOX");

        confirm("are you sure...!");

        prompt("5 + 5 = ?");

        print();

    </script>

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

* You might have occasionally came across “javascript:void(0)” in an HTML Document. It is often used when inserting an expression in a web page might produce some unwanted effect. To remove this effect, “javascript:void(0)” is used. This expression returns undefined primitive value. This is often used with hyperlinks. Sometimes, you will decide to call some JavaScript from inside a link. Normally, when you click a link, the browser loads a brand new page or refreshes the same page (depending on the URL specified). But you most likely do not desire this to happen if you have hooked up some JavaScript thereto link. To prevent the page from refreshing, you could use void(0).

Ex.

<h1 style="color:green">Uday</h1>

<h3>without JavaScript:void(0)</h3>

<a href="#" ondblclick="alert('Welcome to My City')">

Double click on me </a>

<a href="#" ondblclick="MyVoid()">

Double click on me

</a>

<script>

function MyVoid(){

document.getElementById("gfg")

.innerHTML='Welcome to Junagadh';

}

</script>

<p id="gfg"></p>

**(Q.16) How can a page be forced to load another page in JavaScript?**

* We can use [window.location](https://www.geeksforgeeks.org/javascript-window-location-and-document-location-objects/) property inside the script tag to forcefully load another page in Javascript. It is a reference to a Location object that is it represents the current location of the document. We can change the URL of a window by accessing it.
* Syntax:

window.location=<path/URL>

Ex. (a)

<script>

window.location = "https://www.geeksforgeeks.org/"

</script>

(b)

[b-1]

* Force load page

<!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>

    <h2>This is original page</h2>

    <button onclick="Force\_load\_page()">Online</button>

    <button onclick="Force\_load\_local()">Offline</button>

</body>

</html>

<script>

    function Force\_load\_page()

    {

        window.location=("https://www.youtube.com/watch?v=t0Q2otsqC4I")

    }

    function Force\_load\_local()

    {

        window.location=("new\_page.html")

    }

</script>

[b-2]

* New page

<!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>

    <h1>This is New Loaded Page</h1>

</body>

</html>

**(Q.17) What are the disadvantages of using innerHTML in JavaScript?**

* The use of innerHTML very slow: The process of using innerHTML is much slower as its contents as slowly built, also already parsed contents and elements are also re-parsed which takes time.
* Preserves event handlers attached to any DOM elements: The event handlers do not get attached to the new elements created by setting innerHTML automatically. To do so one has to keep track of the event handlers and attach it to new elements manually. This may cause a memory leak on some browsers.
* Content is replaced everywhere: Either you add, append, delete or modify contents on a webpage using innerHTML, all contents is replaced, also all the DOM nodes inside that element are reparsed and recreated.
* Appending to innerHTML is not supported: Usually, += is used for appending in JavaScript. But on appending to an Html tag using innerHTML, the whole tag is re-parsed.
* Old content replaced issue: The old content is replaced even if object.innerHTML = object.innerHTML + ‘html’ is used instead of object.innerHTML += ‘html’. There is no way of appending without reparsing the whole innerHTML. Therefore, working with innerHTML becomes very slow. String concatenation just does not scale when dynamic DOM elements need to be created as the plus and quote openings and closings becomes difficult to track.
* Can break the document: There is no proper validation provided by innerHTML, so any valid HTML code can be used. This may break the document of JavaScript. Even broken HTML can be used, which may lead to unexpected problems.
* Can also be used for Cross-site Scripting(XSS): The fact that innerHTML can add text and elements to the webpage, can easily be used by malicious users to manipulate and display undesirable or harmful elements within other HTML element tags. Cross-site Scripting may also lead to loss, leak and change of sensitive information.