**JAVASCRIPT BASIC & DOM**

1. **What is JavaScript?**

**Ans.** JavaScript is a lightweight, cross-platform, and interpreted scripting language. it is well-known for the development webpages,

many non- browser environments also use it.

JavaScript is used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.)

JavaScript can be used for client-side development as well as server-side developments

JavaScript contains a standard library of object like array, date, and math and core set of language element like operator

control structure and statements.

**2.What is the use of is NaN function?**

## **Ans.**

## **Definition and Usage**

## In JavaScript Nan is short for “Not-a-Number”.

## The is NaN () method returns true if a value is NaN.

## The is NaN () method converts the value to a number before testing it.

## **3. What is negative Infinity?**

**Ans.** 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.

**Note:** JavaScript shows the NEGATIVE\_INFINITY value as -Infinity.

**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 returns 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.
7. The product of two negative infinities is always a positive infinity.
8. The product of both positive and negative infinity is always negative infinity.

## **4. Which company developed JavaScript?**

## **Ans.** JavaScript is a scripting language developed by Netscape. It can be used to program web browser or even servers. It can dynamically update the contents of the webpage, which is the beauty of JavaScript

## **5. What are undeclared and undefined variables?**

## **Ans.**

* [**Undefined**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) variable means a variable has been declared but does not have a value.
* **Undeclared** variable means that the variable does not exist in the program at all.

## **6. Write the code for adding new elements dynamically?**

## **Ans.** New elements can be dynamically created in JavaScript with the help of createElement() method. The attributes of the created element can be set using the setAttribute() method

## **7. What is the difference between View State and Session State?**

## **Ans.**

|  |  |
| --- | --- |
| **View State** | **Session State** |
| 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 post back 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. |
| View State values are lost/cleared when new page is loaded. | Session State can be cleared by programmer or user or in case of timeouts. |

## **8. What is === operator?**

## **Ans.** The strict equality operator (===) checks whether its two operands are equal, returning a Boolean result. Unlike the [equality](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Equality) operator, the strict equality operator always considers operands of different types to be different.

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

## **Ans.**

## **1**: **Changing CSS with the help of the style property:**

**Syntax:**

document. getElementById("id"). style. Property = new style

**2. The className Property:**This property is used to set the current class of the element to the specified class.

**Syntax:**

document. getElementById("id").class Name = class

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

## **Ans.** **Reading from the file**

After the File System module is imported, the reading of the file in JavaScript can be done by using the readFile() function.

### **Syntax**

The syntax to read from a file is as follows −

readFile(path, format, callBackFunc)

The readFile() function accepts three parameters including one optional parameter.

* **Path** − The first parameter is the path of the test file from which the contents are to read. If the current location or directory is the same directory where the file which is to be opened and read is located then, only the file name has to be given.
* **Format** − The second parameter is the optional parameter which is the format of the text file. The format can be ASCII, utf-8 etc.
* **CallBackFunc** − The third parameter is the call back function which takes the error as the parameter and displays the fault is any raised due to the error.

## **Write operation on a file**

After the File System file is imported then, the writeFile() operation is called. The writeFile() method is used to write into the file in JavaScript. The syntax of this method is as follows −

writeFile(path,inputData,callBackFunction)

The writeFile() function accepts three parameters −

* Path − The first parameter is the path of the file or the name of the file into which the input data is to be written.

If there is a file already, then the contents in the file are deleted and the input which is given by the user will get updated or if the file is not present, then the file with that will be created in the given path and the input information is written into it.

* inputData − The second parameter is the input data which contains the data to be written in the file that is opened.
* callBackFuntion − The third parameter is the function which is the call back function which takes the error as the parameter and shows the fault if the write operation fails.

## **11. What are all the looping structures in JavaScript?**

## **Ans.**

**JavaScript supports different kinds of loops:**

* for - loops through a block of code a number of times.
* for/in - loops through the properties of an object.
* for/of - loops through the values of an iterable object.
* while - loops through a block of code while a specified condition is true.

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

**Ans.** Given a string containing an integer value and along with that user passes a base value. We need to convert that string of any base value to an integer in JavaScript.

String Integer

"1002" 1002

For performing the above-illustrated task, we would be using a method (or a function) provided by JavaScript called as **parseInt().**

This is a special method, provided by JavaScript, that takes an integer value (of any base which is either specified or not) and further converts the string into an integer value.

**Syntax:**

* Following is the syntax that a user may use to convert a string into an integer value (of any base)-

parseInt(string\_value, base)

* Alternatively, if we don’t want to specify the base value and just want to convert our string value into an integer value itself, then we may use the following syntax also-

parseInt(string\_value)

Default value returned by base or radix of parseInt() method is **10.**In other words, if we don’t specify any base or radix value then it by default converts the string value to an integer value by taking into regard the base or radix value as 10.

Let us visualize all of the above-illustrated facts with some of the following examples-

## **13. What is the function of the delete operator?**

**Ans.** The **delete operator** in JavaScript is used to delete an object’s property.

If it is used to delete an object property that already exists, it returns true and removes the property from the object. However, deleting an object property that doesn’t exist will not affect the object, but will still return true.

The only time false will be returned is when the delete operator is used to delete a variable or a function.

### Syntax

The syntax for using the delete operator is as follows:

delete object.property;  
// OR  
delete object["property"];

### Parameters

**object**: This is the object whose property we want to delete.

**property**: This is the property to be deleted.

### Return value

The delete operator returns true if the specified property is deleted, or false if the property is not deleted.

### Code

In the code below, an object is created and the delete operator is used to delete some of its properties:

let human = {

    name: "John Doe",

    age: 15,

    country: "Nigeria"

}

let dog = {

    name: "Buddy",

    age: 2,

    country : "Germany"

}

// log retured values after delete

console.log(delete human["country"]) // same as human.country

console.log(delete dog.country) // same as dog["country"]

// log affected objects

console.log(human)  // "country" property deleted

console.log(dog)    // "country" property deleted

**Output**

**true**

**true**

**{ name: 'John Doe', age: 15 }**

**{ name: 'Buddy', age: 2 }**

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

## **Ans.**

In Javascript, popup boxes are used to display the message or notification to the user. There are three types of [pop-up boxes in JavaScript](https://www.geeksforgeeks.org/javascript-dialogue-boxes/) namely**Alert Box**,**Confirm Box** and**Prompt Box**.

**Alert Box:** It is used when a warning message is needed to be produced. When the alert box is displayed to the user, the user needs to press ok and proceed.

**Syntax:**

alert("your Alert here")

**Prompt Box:** It is a type of pop up box which is used to get the user input for further use. After entering the required details user have to click ok to proceed next stage else by pressing the cancel button user returns the null value.

**Syntax:**

prompt("your Prompt here")

**Confirm Box:** It is a type of pop-up box that is used to get authorization or permission from the user. The user has to press the ok or cancel button to proceed.

**Syntax:**

confirm("your query here")

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

## **Ans.**

**Using “javascript:void(0);” in anchor tag:**Writing “javascript:void(0);” in anchor tag can prevent the page to reload and JavaScript functions can be called on single or double clicks easily.

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

## **Ans.**

**Step 1:** Create a file named ***index.html***. Add a heading and two buttons to it. One button forcefully loads a page with a live URL and the other button loads a local HTML page. In the *<script>* tag we have two functions, one loads gfg home page, and the second loads a local HTML page using ***window.location*** property.

index.html

**Input:**

|  |
| --- |
| <!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">  </head>    <body>      <h3>This is the original page</h3>      <br>        <button onclick="force\_load\_top()">          Force Load Top Page      </button>     <br><br>        <button onclick="force\_load\_local()">          Force Load Local HTML page      </button>        <script>          // top technologies link page          function force\_load\_top() {              window.location =                  "<https://www.tops-int.com/>"          }            function force\_load\_local() {              window.location =                  "F:/top/PageRedirect/newPage.html"          }      </script>  </body>    </html>  **Output:**  Image  Image |

**Step 2:** Create a file named ***newPage.html***. This is the local HTML page that would be loaded by Javascript.

newPage.html

**Input:**

|  |
| --- |
| <!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> New Page </title>  </head>    <body>      <!-- new loaded page -->      <h3>This is the new loaded page</h3>  </body>    </html>  **Output:**  Image |

## **17. What are the disadvantages of using inner HTML in JavaScript?**

## **Ans.**

**Disadvantages of using innerHTML property 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.

**Example:**

<p id="geek">Geeks</p>

title = document.getElementById('#geek')

// The whole "geek" tag is reparsed

title.innerHTML += '<p> forGeeks </p>'

* **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.

**Example:**

|  |
| --- |
| <!DOCTYPE html>  <html>    <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>          Using innerHTML in JavaScript      </title>  </head>    <body style="text-align: center">        <h1 style="color:green">          Top Technology      </h1>        <p id="P">          A computer science          portal for tops.      </p>        <button onclick="tops()">          Try it      </button>        <p id="p"></p>        <script>      // try it function start          function tops() {              var x = document.getElementById("P")                          .innerHTML;                document.getElementById("p")                          .innerHTML = x;                document.getElementById("p")                          .style.color = "green";          }      </script>  </body>    </html>  **Output:**  Image  Image |