1. **What is JavaScript?**

**Ans.** JavaScript is a lightweight, cross-platform, and interpreted compiled programming language which is also known as the scripting language for web pages.

* It also known as object-based scripting language.
* It is well-known for the development of web pages, many non-browser environments also use it.
* In 1993, Mosaic, the first popular web browser, came into existence.
* In the year 1994, Netscape was founded by Marc Andreessen.
* Java script first name is live script.
* Due to trademark and other reason live script renamed as java script.
* Pure JavaScript code also known as Vanila java script.
* JavaScript is used to create interactive websites. It is mainly used for:
* Client-side validation
* Dynamic drop-down menus
* Displaying date and time
* Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box)
* Displaying clocks etc.
* It is written in <script> tag.
* <script> tag is written between <head> or <body> tag.

example,

<script>

document.write("Hello JavaScript by JavaScript");

</script>

* where a Document object represents the HTML document that is displayed in that window. write() is a function to display your content.
* There are three ways to add java script in file :
* Inline - add using event
* Internal - in <script> tag in <body> or <head> tag
* External - create new file using .js extension and add with <script> tag in HTML file.

1. **What is the use of isNaN function?**

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

* **Syntax:**

isNaN( value )

* **Parameter Values:** This method accepts a 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.

1. **What is negative Infinity?**

**Ans.** The **negative infinity** in JavaScript is a constant value that is used to represent a value that 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:

* Negative infinity results in **-0**(different from 0 ) when divided by any other number.
* When divided by itself or positive infinity, negative infinity return NaN
* Negative infinity, when divided by any positive number (apart from positive infinity) is negative infinity.
* Negative infinity, divided by any negative number (apart from negative infinity) is positive infinity.
* If we multiply negative infinity with NaN, we will get NaN as a result.
* The product of 0 and negative infinity is Nan.
* The product of two negative infinities is always a positive infinity.
* The product of both positive and negative infinity is always negative infinity.

**Syntax:**

Number.NEGATIVE\_INFINITY

1. **Which company developed JavaScript?**

**Ans.** JavaScript was created at Netscape Communications by Brendan Eich in 1995. Netscape and Eich designed JavaScript as a scripting language for use with the company's flagship web browser, Netscape Navigator.

1. **What are undeclared and undefined variables?**

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

**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 *run-time error* with the return value as **“undefined”**. The scope of the undeclared variables is always global.

1. **Write the code for adding new elements dynamically?**

**Ans.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        .mydiv {

            height: 200px;

            width: 200px;

            border: 1px solid black;

        }

    </style>

</head>

<body>

    <button onclick="changeDivVal()">Click Here</button>

    <hr>

    <div id="div1" class="mydiv" onmouseover="changeDivVal()" onmouseout="removeDivVal()"></div>

    <script>

        function changeDivVal() {

            let rect = document.getElementById("div1");

            rect.style.backgroundColor = "pink"

            rect.style.border = "4px solid brown"

            rect.style.borderRadius = "40px"

            rect.style.transition = "1s all ease"

        }

        function removeDivVal() {

            let rect = document.getElementById("div1");

            rect.style.backgroundColor = "white"

            rect.style.border = "1px solid black"

            rect.style.borderRadius = "0"

            rect.style.transition = "1s all ease"

        }

    </script>

</body>

</html>

1. **What is the difference between ViewState and SessionState?**

**Ans. Differences 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. |

1. **What is === operator?**

**Ans.** The strict equality ( === ) operator checks whether its two operands are equal, returning a Boolean result. Unlike the equality operator, the strict equality operator always considers operands of different types to be different.

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

**Ans.** We can change, add or remove any CSS property from an HTML element on the occurrence of any event with the help of JavaScript. There are two common approaches that allow us to achieve this task.

* **Style.property**
* **Changing the class itself**

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

**Syntax:**

document.getElementById("id").style.property = new\_style

**Approach 2: Changing the class itself –**We can use two properties that can be used to manipulate the classes.

**The class-list Property:**The **class-list** is a read-only property that returns the CSS class names of an element as a DOMTokenList object.

**Syntax:**

document.getElementById("id").class-list

You can use the below-mentioned methods to add classes, remove classes, and toggle between different classes respectively.

* **The add() method:** It adds one or more classes.
* **The remove() method:**It removes one or more classes.
* **The toggle() method:**If the class does not exist it adds it and returns true. It removes the class and returns false. The second Boolean argument forces the class to be added or removed.

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

**Ans.** 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 java script. 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, Typed Array, or Data View.
* **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.

1. **What are all the looping structures in JavaScript?**

**Ans.** The main looping structures in JavaScript:

1. ****For Loop:**** The **for** loop is a traditional loop that runs a specific number of times.

**Syntax:**

for (initialization; condition; increment/decrement) {

// code to be executed

}

1. ****While Loop:**** The **while** loop runs as long as a specified condition evaluates to **true**.

**Syntax:**

while (condition) {

// code to be executed

}

1. ****do...while Loop:**** The **do...while** loop is similar to the **while** loop, but it ensures that the code block is executed at least once before checking the condition.

**Syntax:**

do {

// code to be executed

} while (condition);

1. ****for...of Loop:**** The **for...of** loop is used to iterate over values of an iterable object like arrays, strings, maps, sets, etc.

**Syntax:**

for (const element of iterable) {

// code to be executed for each element

}

1. ****for...in Loop:**** The **for...in** loop is used to iterate over the properties of an object. Note that this loop is not recommended for iterating over arrays, as it may behave unexpectedly due to enumerating properties.

**Syntax:**

for (const key in object) {

if (object.hasOwnProperty(key)) {

// code to be executed for each property

}

}

1. ****forEach Method (Array):**** This is a method available for arrays that allows you to iterate through each element in the array.

**Syntax:**

array.forEach((element, index) => {

// code to be executed for each element

});

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

**Ans.** In JavaScript [parseInt()](https://www.geeksforgeeks.org/javascript-parseint-function/) function (or a method) is used to convert the passed-in string parameter or value to an integer value itself. This function returns an **integer** of the base which is specified in the second argument of the [parseInt() function](https://www.geeksforgeeks.org/javascript-parseint-function/). [JavaScript parseInt() function](https://www.geeksforgeeks.org/javascript-parseint-function/) returns Nan( not a number) when the string doesn’t contain a number. We can convert a string to java script by the following methods:

* [Using the parseInt() method](https://www.geeksforgeeks.org/javascript-parseint-function/)
* [Using the Number() method](https://www.geeksforgeeks.org/javascript-number-complete-reference/)
* [Using the Unary operator](https://www.geeksforgeeks.org/javascript-arithmetic-unary-plus-operator/)
* **Using the**[parseInt()](https://www.geeksforgeeks.org/javascript-number-parseint-method/)**method:**[JavaScript parseInt()](https://www.geeksforgeeks.org/javascript-parseint-function/)Method is used to accept the string and radix parameter and convert it into an integer.

**Syntax:**

parseInt(Value, radix)

* **Using the**[Number() method](https://www.geeksforgeeks.org/javascript-number-complete-reference/)**:**In Java script, the [Number() method](https://www.geeksforgeeks.org/javascript-number-complete-reference/) is used to convert any primitive data type to a number, if it is not convertible it returns NAN.

**Syntax:**

Number(value)

* **Using the**[Unary Operator](https://www.geeksforgeeks.org/javascript-arithmetic-unary-plus-operator/)**:**In Java script, the [Unary operator(+)](https://www.geeksforgeeks.org/javascript-arithmetic-unary-plus-operator/) is used to convert a string, Boolean, and non-string to a number.

**Syntax:**

+op;

1. **What is the function of the delete operator?**

**Ans.** Delete is comparatively a lesser-known operator in JavaScript. This operator is more specifically used to delete JavaScript object properties.

* 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']

**Parameter:**It does not take any parameter.

**Return type:** This operator returns *true* if it removes a property. While deleting an object property that doesn’t exist will return a *true* but it will not affect the object. Though while trying to delete a variable or a function will return a *false*.

Below are examples of the delete Operator.

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

**Ans.** 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**.

1. **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")

1. **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")

1. **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")

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

**Ans.** 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.

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

**Ans.** 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:**

<script>

window.location = <Path / URL>

</script>

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

**Ans. T**here are some disadvantages to using innerHTML in JavaScript.

* **Disadvantages of using inner html 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.