**ASSIGNMENT**

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

**• What is JavaScript?**

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.

* **Example**

<script>  
document.getElementById("demo").innerHTML = "Hello JavaScript!";  
</script>

**• What is the use of isNaN function?**

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.

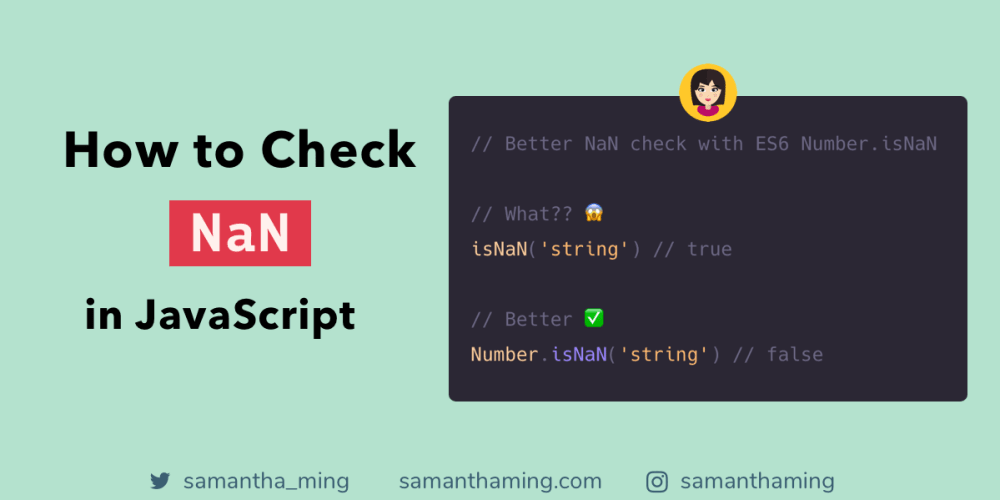
* isNaN function use:

isNaN() is a function property of the global object. For number values, isNaN() tests if the number is the value NaN . When the argument to the isNaN() function is not of type Number, the value is first coerced to a number, and the resulting value is then compared against NaN .

In JavaScript NaN is short for "Not-a-Number".

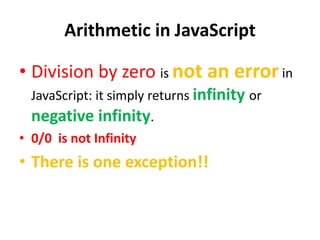
### Example

isNaN(123);  
isNaN(-1.23);  
isNaN(5-2);  
isNaN(0);



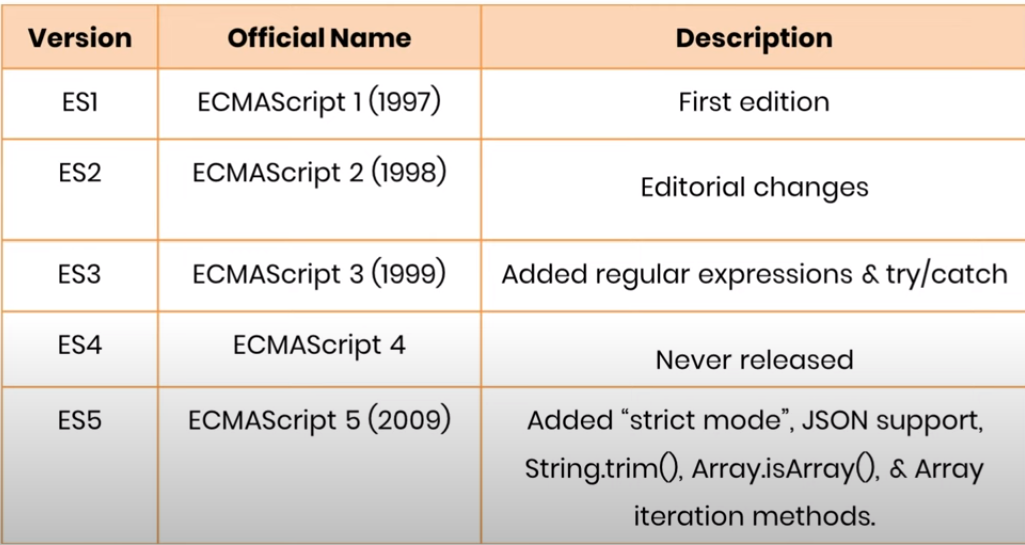
**• What is negative Infinity?**

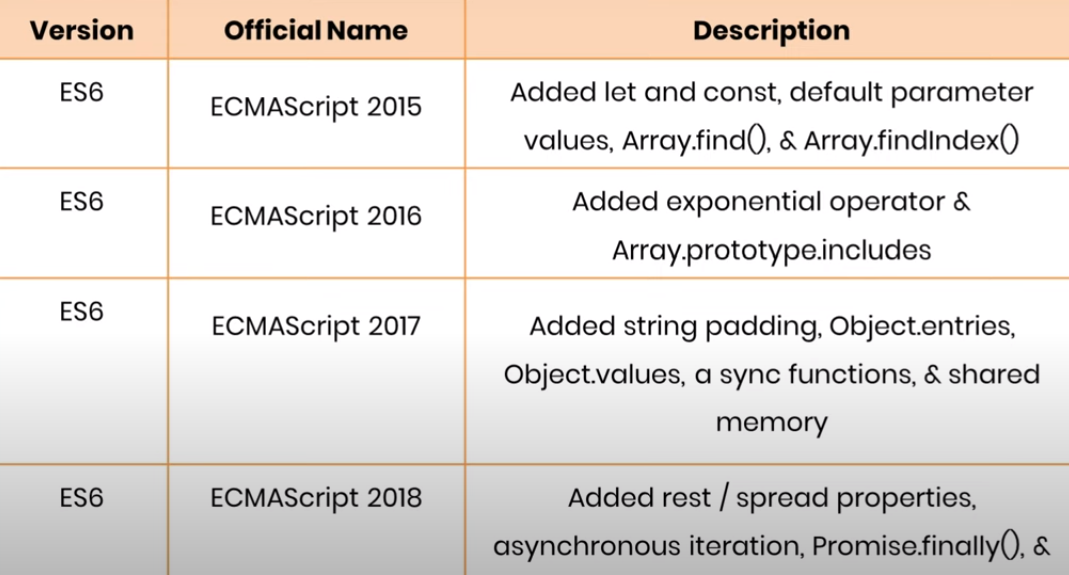
NEGATIVE\_INFINITY is a special numeric value that is returned when an arithmetic operation or mathematical function generates a negative value greater than the largest representable number in JavaScript (i.e., more negative than -Number. MAX\_VALUE) . JavaScript displays the NEGATIVE\_INFINITY value as -Infinity .



**• Which company developed JavaScript?**

* It was created in 1995 by Brendan Eich while he was an engineer at Netscape. It was originally going to be named LiveScript but was renamed. Unlike most programming languages, the JavaScript language has no concept of input or output.
* It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanism for communicating with outside world. The most common host environment is the browser.

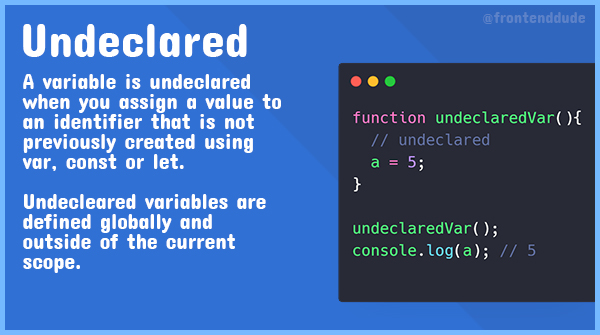


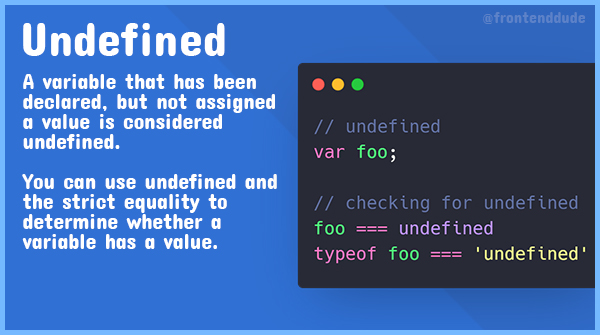


**• What are undeclared and undefined variables?**

**Undeclared** − It occurs when a variable which hasn't been declared using var, let or const is being tried to access.

**Undefined** − It occurs when a variable has been declared using var, let or const but isn't given a value.





**• Write the code for adding new elements dynamically?**

Javascript is a very important language when it comes to learning how the browser works. Often there are times we would like to add dynamic elements/content to our web pages. This post deals with all of that.

**Creation of new element:** New elements can be created in JS by using the **createElement()** method.

**Syntax:**

document.createElement("<*tagName*>");

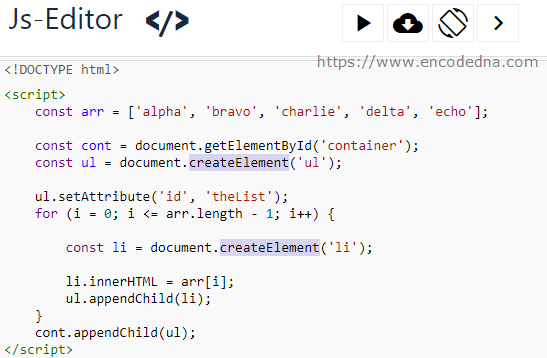
// Where <*tagName*> can be any HTML

// tagName like div, ul, button, etc.

// newDiv element has been created

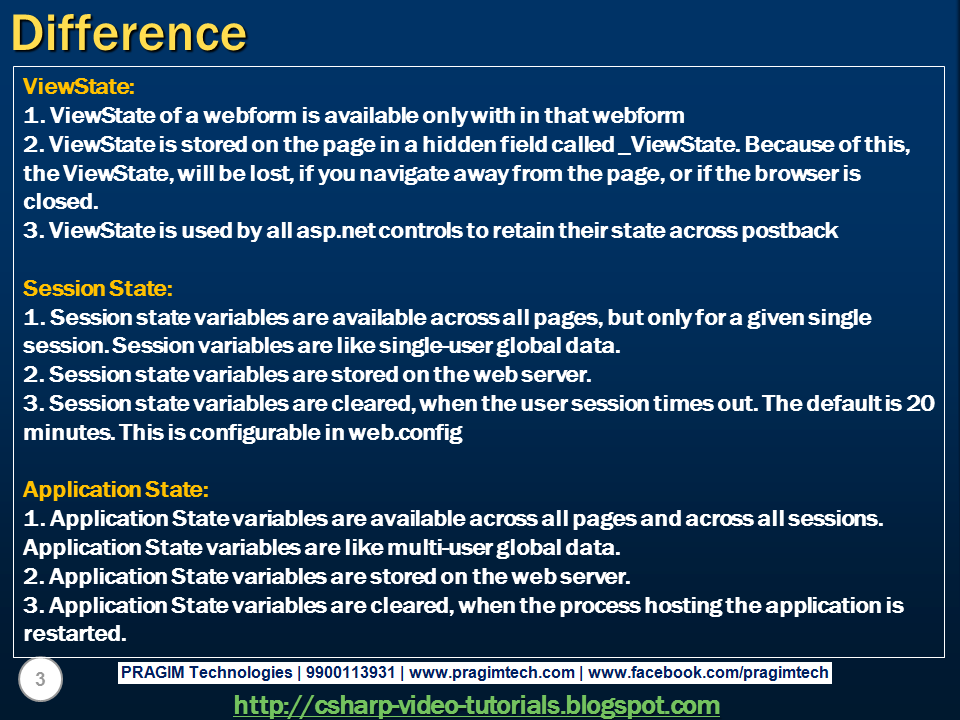
For Eg: **let newDiv = document.createElement("div");**

Once the element has been created, let’s move on to the setting of attributes of the newly created element.



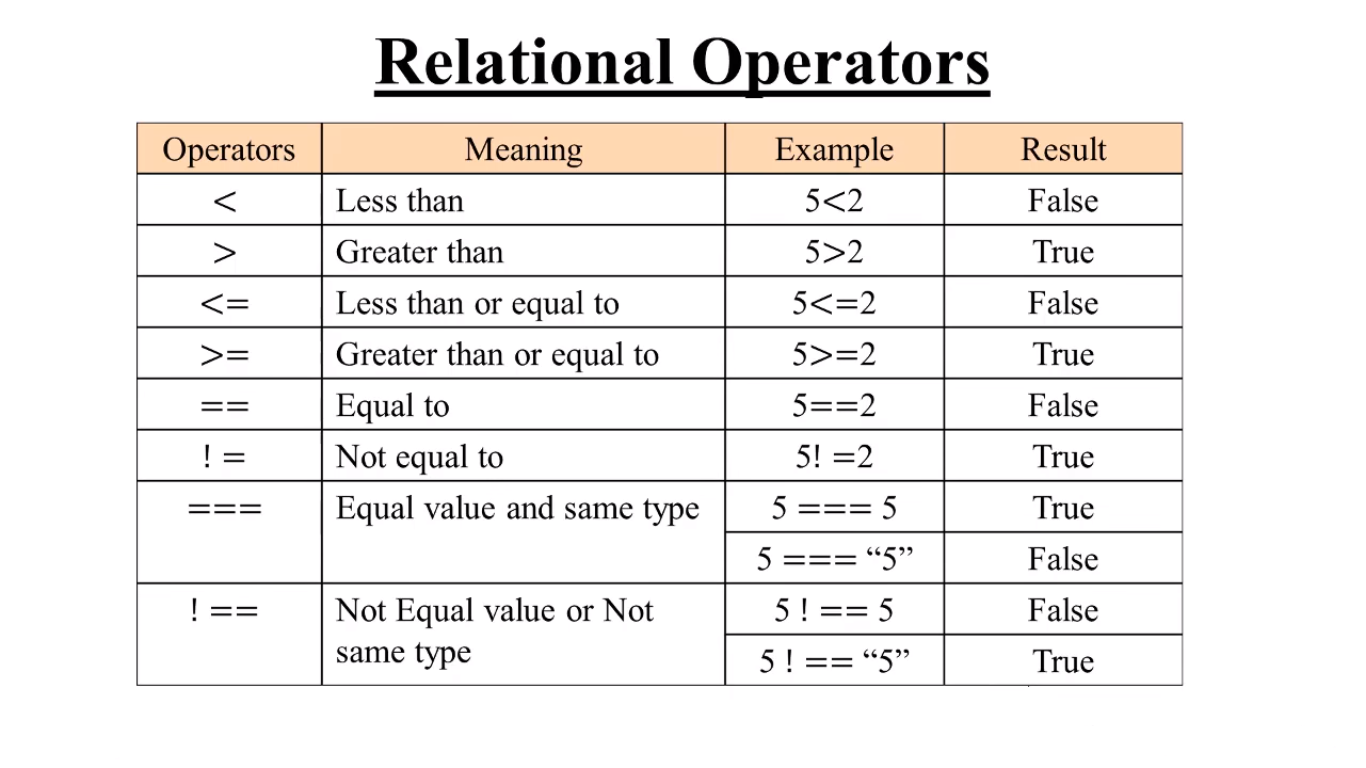
**• What is the difference between ViewState and SessionState?**

The basic difference between these two is that the **ViewState is to manage state at the client's end, making state management easy for end-user while SessionState manages state at the server's end**, making it easy to manage content from this end too.



**• What is === operator?**

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.



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

 If you want to build a cool website or app then UI plays an important role. 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 approaches that allow us to achieve this task.

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

* **Syntax:**

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

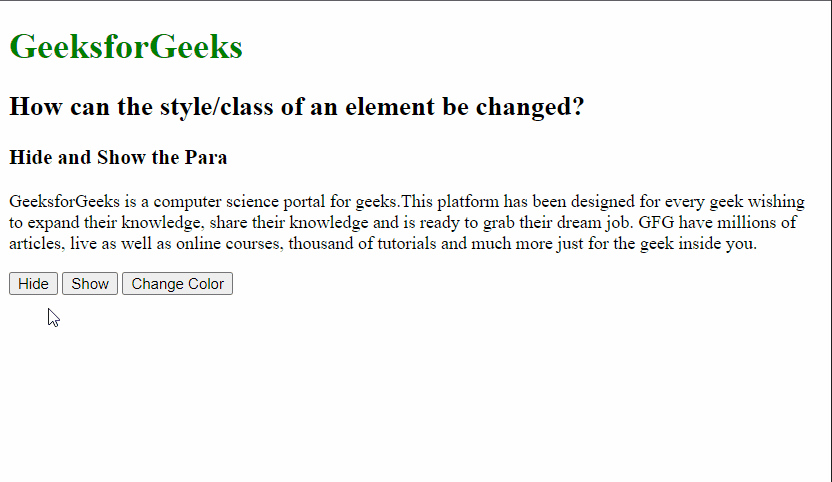
**Approach 2: Changing the class itself –**We can use two properties that can be used to manipulate the classes.

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

document.getElementById("id").classList

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.



**• 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 )

**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.

**• What are all the looping structures in JavaScript?**

**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.
* **EXAMPLE:**

Instead of writing:

text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";

You can write:

for (let i = 0; i < cars.length; i++) {  
  text += cars[i] + "<br>";  
}

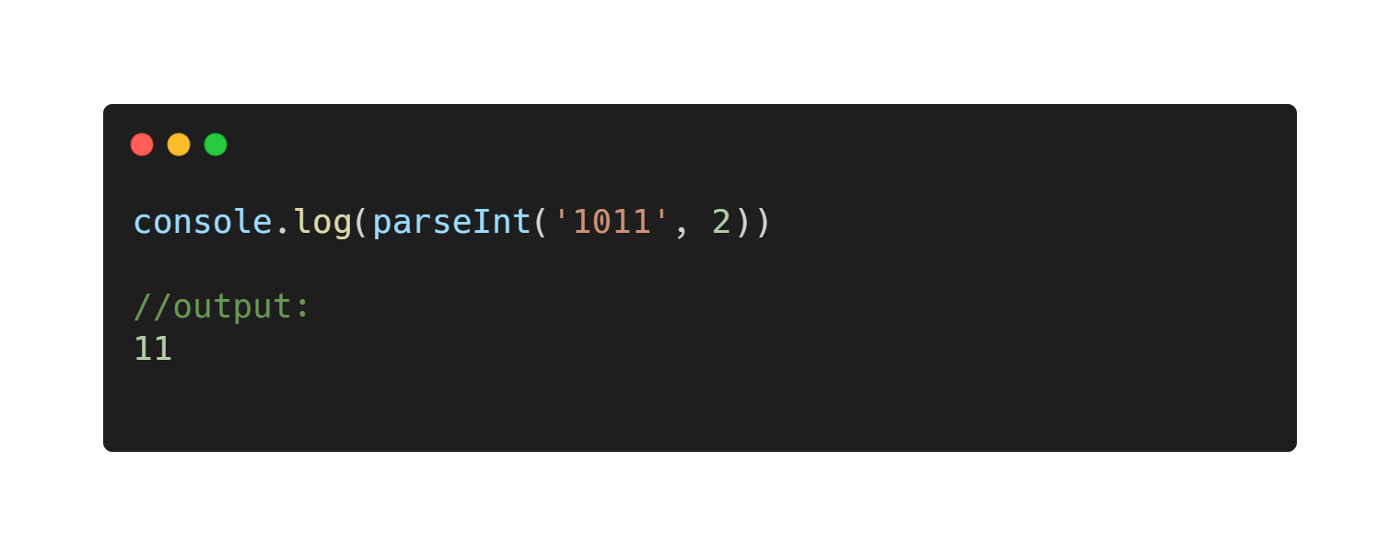
**• How can you convert the string of any base to an integer in JavaScript?**

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 base which is specified in second argument of **parseInt() function**. JavaScript parseInt() function returns Nan( not a number) when the string doesn’t contain number.

* **Syntax:**

parseInt(Value, radix)

It accepts a string as a value and converts it to specified radix system (any desired numerical value passed by a user) and returns an integer (corresponding to the passed in numerical radix value). Program to convert string to integer:



**• 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.

#### **delete operator**

const Employee = {

firstname: 'John',

lastname: 'Doe'

};

console.log(Employee.firstname);

// expected output: "John"

delete Employee.firstname;

console.log(Employee.firstname);

// expected output: undefined

**• 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("*sometext*");

The window.alert() method can be written without the window prefix.

### Example

alert("I am an alert box!");

## **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("*sometext*");

The window.confirm() method can be written without the window prefix.

### Example

if (confirm("Press a button!")) {  
  txt = "You pressed OK!";  
} else {  
  txt = "You pressed Cancel!";  
}

## **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.

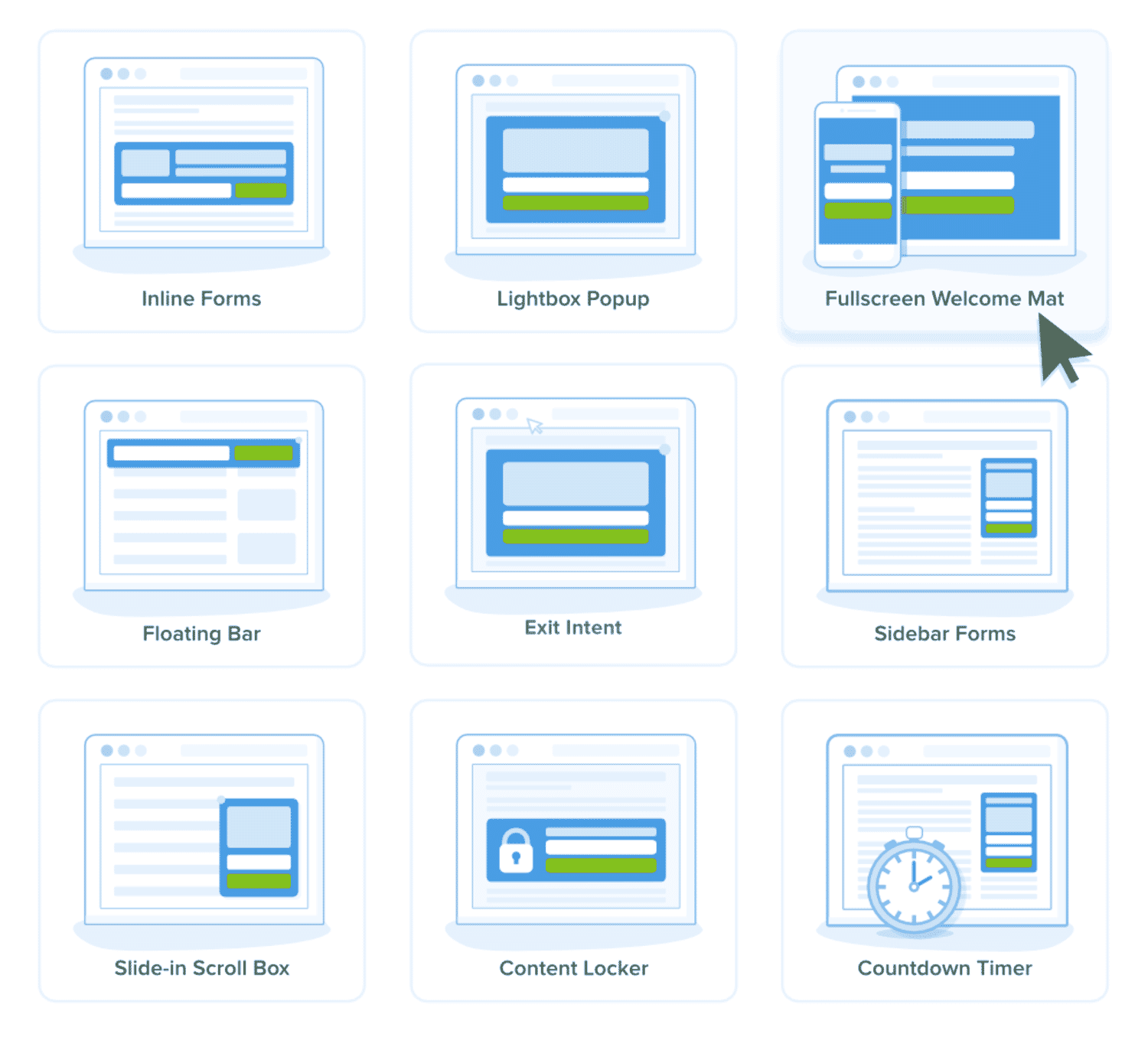
### Syntax

window.prompt("*sometext*","*defaultText*");

The window.prompt() method can be written without the window prefix.

### Example

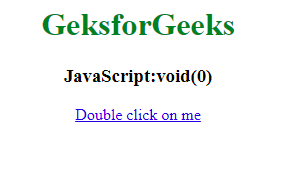
let person = prompt("Please enter your name", "Harry Potter");  
let text;  
if (person == null || person == "") {  
  text = "User cancelled the prompt.";  
} else {  
  text = "Hello " + person + "! How are you today?";  
}



**• What is the use of Void (0)?**

JavaScript void 0 means returning undefined (void) as a primitive value. You might come across the term “JavaScript:void(0)” while going through [HTML](https://www.simplilearn.com/tutorials/html-tutorial/what-is-html) documents. It is used to prevent any side effects caused while inserting an expression in a web page. For instance, URLs or hyperlinks are the common examples of using JavaScript void 0. Suppose you insert a link and want to call some JavaScript through it. Usually, when you click on a link, the browser will either reload or open a new page. However, if you just want to call JavaScript through that link, you would not want the entire page to refresh. This is where the JavaScript:void(0) will come in handy.

When you use JavaScript void 0, it will return an undefined primitive value. This will prevent the browser from opening a new or reloading the web page and allowing you to call the JavaScript through it.



**• 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:**

<script>

window.location = <Path / URL>

</script>

* **Example:**

<script>

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

</script>

So in the above example, we see that by changing the window.location Object inside Javascript we can change the URL of our window and thus successfully load any page forcibly from our Javascript without any href tag. We will build a small working sample to learn it practically.

**• What are the disadvantages of using innerHTML in JavaScript?**

The [innerHTML property](https://www.geeksforgeeks.org/html-dom-innerhtml-property/" \t "_blank) is a part of the Document Object Model (DOM) that is used to set or return the HTML content of an element. Where the return value represents the text content of the HTML element. It allows JavaScript code to manipulate a website being displayed. More specifically, it sets or returns the HTML content (the inner HTML) of an element. The innerHTML property is widely used to modify the contents of a webpage as it is the easiest way of modifying DOM. But there are some disadvantages to using innerHTML in JavaScript.

* **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.

