# JavaScript

## Topics to be discussed.....

- Introduction Java script
- Variables, java script operators
- conditional statements and loops
- JavaScript break and continue statement,
- Dialog boxes
- JavaScript Arrays,
- JavaScript Events
- JavaScript User Define Function
- JavaScript Built in Functions.

# INTRODUCTION

- ❑ JavaScript is a scripting language which is used to create web site on internet.
- ❑ It’s a lightweight programming language.
- ❑ It is interpreted by the browser engine when the web page is loaded.
- ❑ JavaScript was created by “Brendan Eich “at Netscape.
- ❑ It was first introduced in December 1995 under the name of Live Script.
- ❑ Java script is best used in Netscape Navigator web browser.
- ❑ JavaScript is an interpreted language that everyone can use JavaScript without purchasing a license.

## INTRODUCTION

- It’s  a case sensitive language.
- It provides an easy development process for program.
- We can write java script program in notepad & also add in html file.
- We can run java script program on any browsers like internet explorer, Netscape navigator etc.
- JavaScript is a cross-platform, object-oriented scripting language.
- JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements

# <Script> Tag

- <SCRIPT> tag is used to write program of java script.
- We can create java script program in <HEAD> part or <BODY> part using <SCRIPT> tag.
- **Syntax:**

- The code must be placed between <script> ….</script>
- In java script each line end with (;)

```
<script language="JavaScript"
type="text/JavaScript" src="path of external js
file">
```

## Types of Java Script

- There are following two ways in which users can add JavaScript to HTML pages.
    - Inline JavaScript
    - Internal JavaScript
    - External file

# Inline Java Script

- JavaScript code embedded directly within HTML elements using the onclick, onmouseover, or other event attributes.
- This allows you to execute JavaScript code in response to user interactions.
- **Example:**

```
<html>
<head>
<script>
            alert(‘welcome’);
</script>
</head>
<body>
     <a href="#" onClick="document.write('Welcome
!');"> Click Here </a>
</body>
</html>
```

# Internal Java Script

- Internal java script are placed in <HEAD> or <BODY> section of a particular web page using <script> tag.
- These styles can be used only for the web page in which they are embedded so its called embedded java script.
- **Example:**

```
<html>
<head>
<script>
            alert('welcome');
</script>
</head>
<body>
        <p>This is a Demo.</p>
</body>
</html>
```

# External Java Script

- External style sheets are separate files full of JavaScript code with **.js** extension.
- It becomes very helpful if we want to use same code in multiple HTML documents.
- **Example:**

```
Alert('welcome');
```

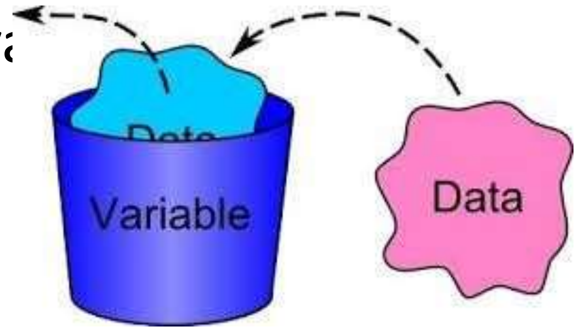
Message.css

```
<html>
<head>
		<script type="text/JavaScript"
src="message.js"></script>
</head>
<body></body>
</html>
```

## Variables

# Variable

- Variables are containers that store values.
- A JavaScript variable is simply a name of storage location.
- It is mainly used to hold values that we want to use during the script.
- Before you use a variable in a JavaScript program, you must declare it.
- Variables are declared with the ***var*** keyword , ***let or const keyword.***
- ***Assignment operator "=“*** used to assign va



# Naming Rules for Variable

- Variable names must start with a letter, an underscore (_) or a dollar sign ($).
- Variable names cannot contain spaces.
- You can not use any Reserve word.
- Variables should be given descriptive names that indicate their content and usage
- **JavaScript variables are case sensitive**
- You can not use any special sign

## Declaration of variables

```
var  123 =30;
```

```
var x = 10;
```

```
var *aa = 320;
```

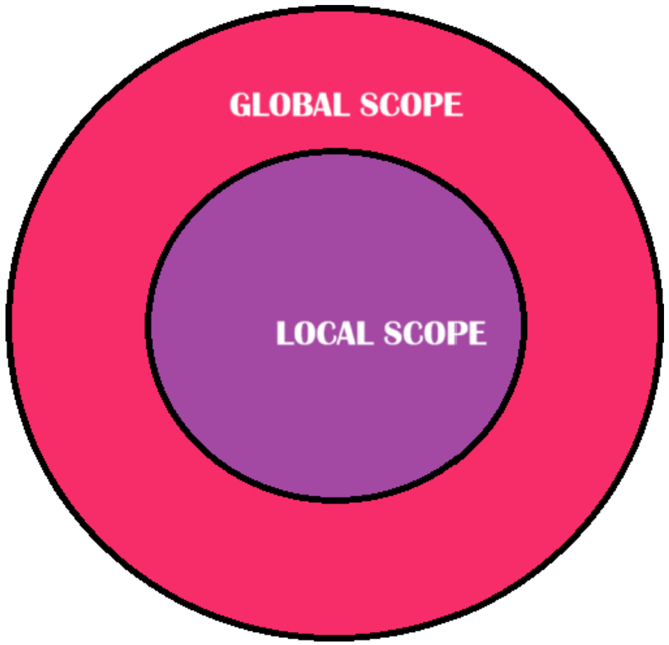
```
var _value =
"abc";
```

```
var 1A = “Aa”;
```

```
var $a = 99.9;
```

## Scope of variable

- **Scope** is like the area/section where a variable can be **seen or used** in your code.



## Scope of variable

**LOCAL SCOPE**

- ❑ Variable declared within programming block.
- ❑ It can only be used inside the code block in which it is declared.
- ❑ You can only use it **inside that function.**

**GLOBAL SCOPE**

- ❑ Variable defined outside programming block.
- ❑ It can be accessed throughout the program.
- ❑ You can use it **anywhere** in your code.

## Data Type

# Data type

- Data type is a specification that shows what kinds of data a variable can hold.
- In java script when we store data inside the variable according to the type of data, data type will be automatically defined.
- There are two types of data types in JavaScript.
  - **Primitive data type**
  - **Non-primitive (reference) data type**

## Primitive Data Type

- These are basic data types that store **single values**

| | |
|---|---|
| **String** | Used to store textual value. <br><br> **Example:** var  name = “Shivam“; |
| **Number** | Used to store Integers or floating-point numbers. <br><br> **Example:** var per=80; var pi=3.14; |
| **Boolean** | Represent Boolean values. <br><br> Example: var isactive=true; |
| **Undefined** | A variable that has been declared but not assigned a value. <br><br> **Example:**  var marks; |
| **Null** | Represents intentional "no value” <br><br> **Example:** var icode=null; |

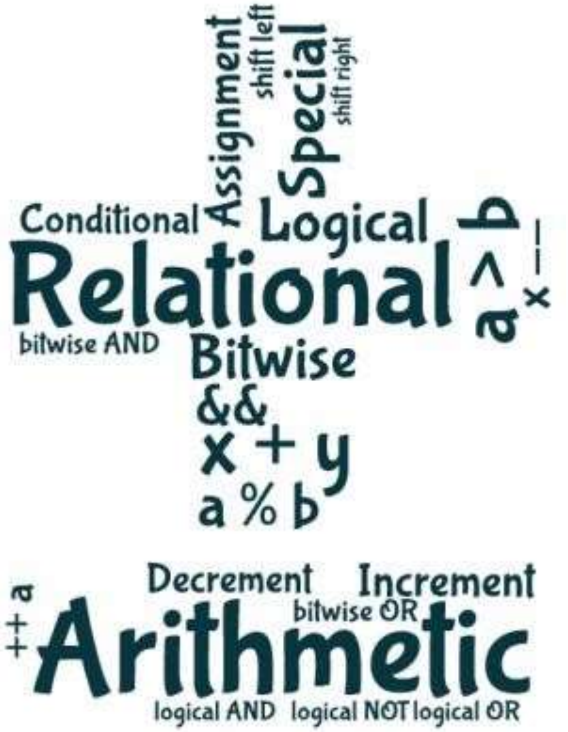
## Non-Primitive Data Type

- These hold collections or more complex entities.

| | |
|---|---|
| **object** | A collection of key-value pairs<br><br>**Example:** var  student= ,name:’Shivam’, Id:’A123’-; |
| **array** | Used to store Ordered list of values<br><br>**Example:** var color=*‘red’,’green’,’blue’+; |

## Operators

- Operators are symbols that used to perform operations on variable's value.
- Java script provides following operators:
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - Logical Operators
  - Conditional Operators



# Arithmetic operators

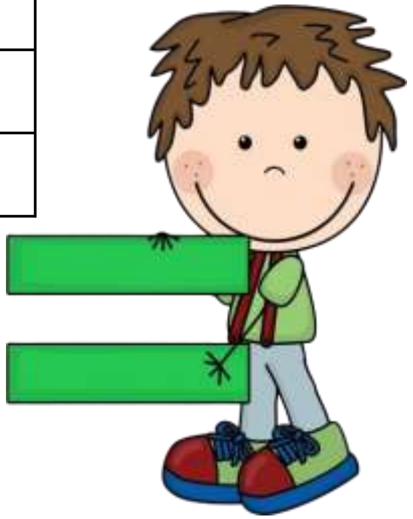
- Arithmetic operators are used to perform arithmetic operations on the operands.

| Operator | Description | Example |
|---|---|---|
| + | Addition | 10+20 = 30 |
| - | Subtraction | 20-10 = 10 |
| * | Multiplication | 10*20 = 200 |
| / | Division | 20/10 = 2 |
| % | Modulus (Remainder) | 20%10 = 0 |

# Assignment operators

- Assignment operators are used to assign the value in variable.

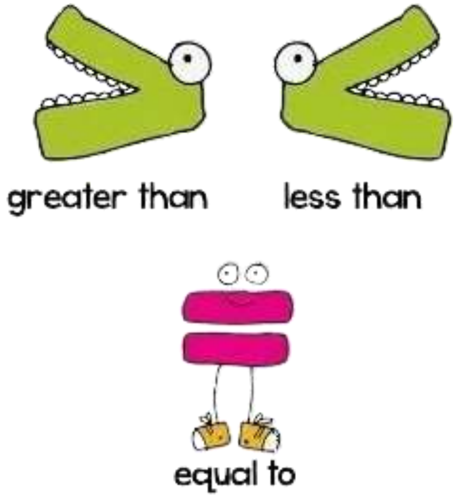
| Operator | Description | Example |
|---|---|---|
| = | Assign | 10+10 = 20 |
| += | Add and assign | var a=10; a+=20; Now a = 30 |
| -= | Subtract and assign | var a=20; a-=10; Now a = 10 |
| *= | Multiply and assign | var a=10; a*=20; Now a = 200 |
| /= | Divide and assign | var a=10; a/=2; Now a = 5 |
| %= | Modulus and assign | var a=10; a%=2; Now a = 0 |



# Comparison operators

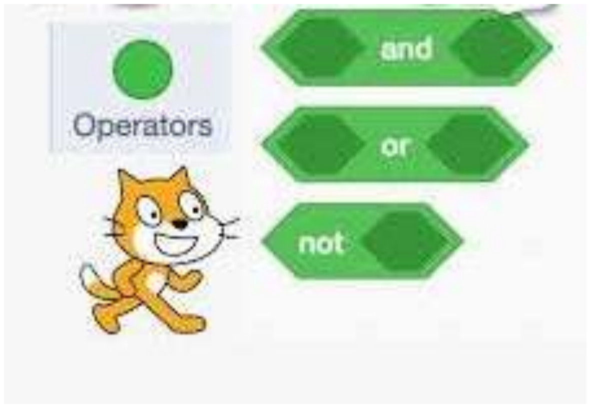
- comparison operators are used to compare one value or variable with something else.

| Operator | Description | Example |
|---|---|---|
| == | Is equal to | 10==20 = false |
| != | Not equal to | 10!=20 = true |
| > | Greater than | 20>10 = true |
| >= | Greater than or equal to | 20>=10 = true |
| < | Less than | 20<10 = false |
| <= | Less than or equal to | 20<=10 = false |



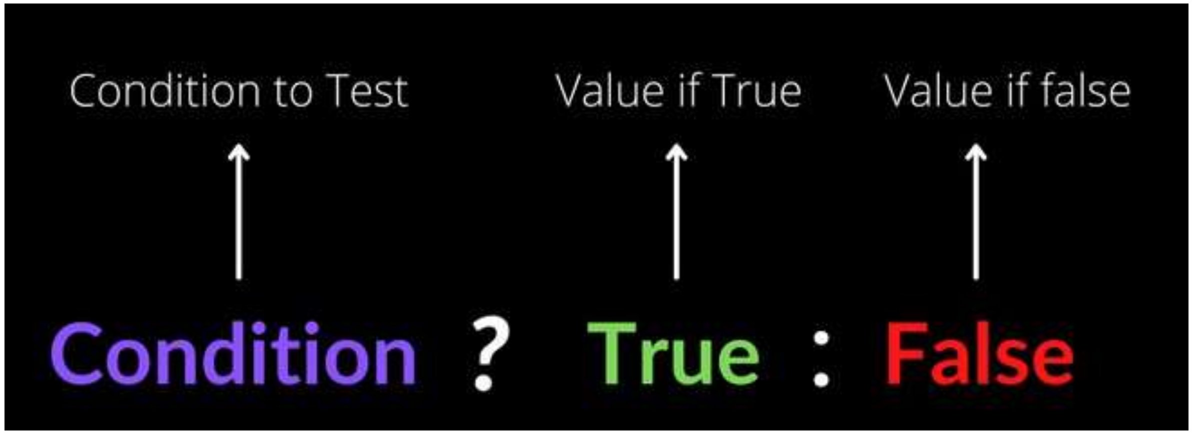
# Logical operators

| Operator | Description | Example |
|---|---|---|
| **&&** | Logical AND | (10==20 && 20==33) = false |
| **\|\|** | Logical OR | (10==20 \|\| 20==33) = false |
| **!** | Logical Not | !(10==20) = true |



## Conditional Operator

- The conditional operator is the only JavaScript operator that takes three operands.
- The operator can have one of two values based on a condition.
- **Syntax :** condition ? val1 : val2



## Conditional Statements

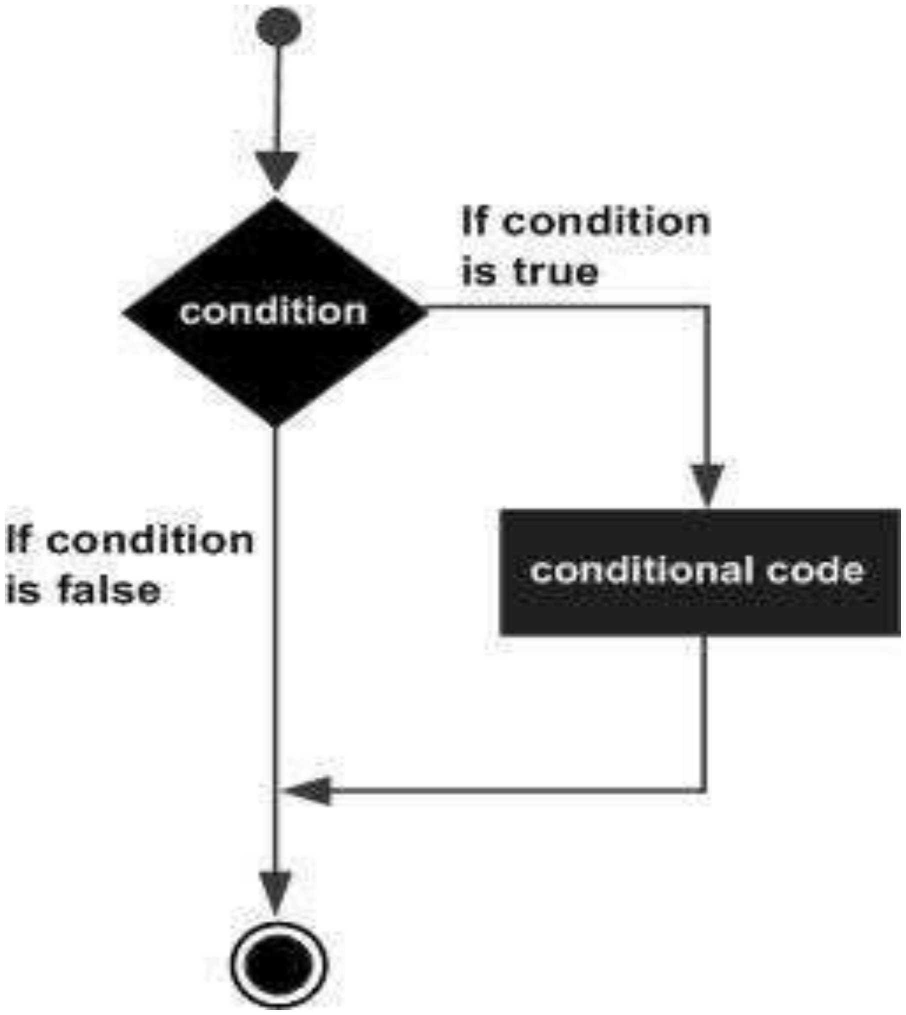
## Conditional Statements

- Conditional statements in JavaScript allow to execute specific blocks of code based on conditions.
- There are three forms of if statement in JavaScript.
  - If Statement
  - If else statement
  - if else if statement
  - Switch case

# Simple if

- **Syntax :**

```
if (condition)
{
        //content to be evaluated
      if condition is true
}
```

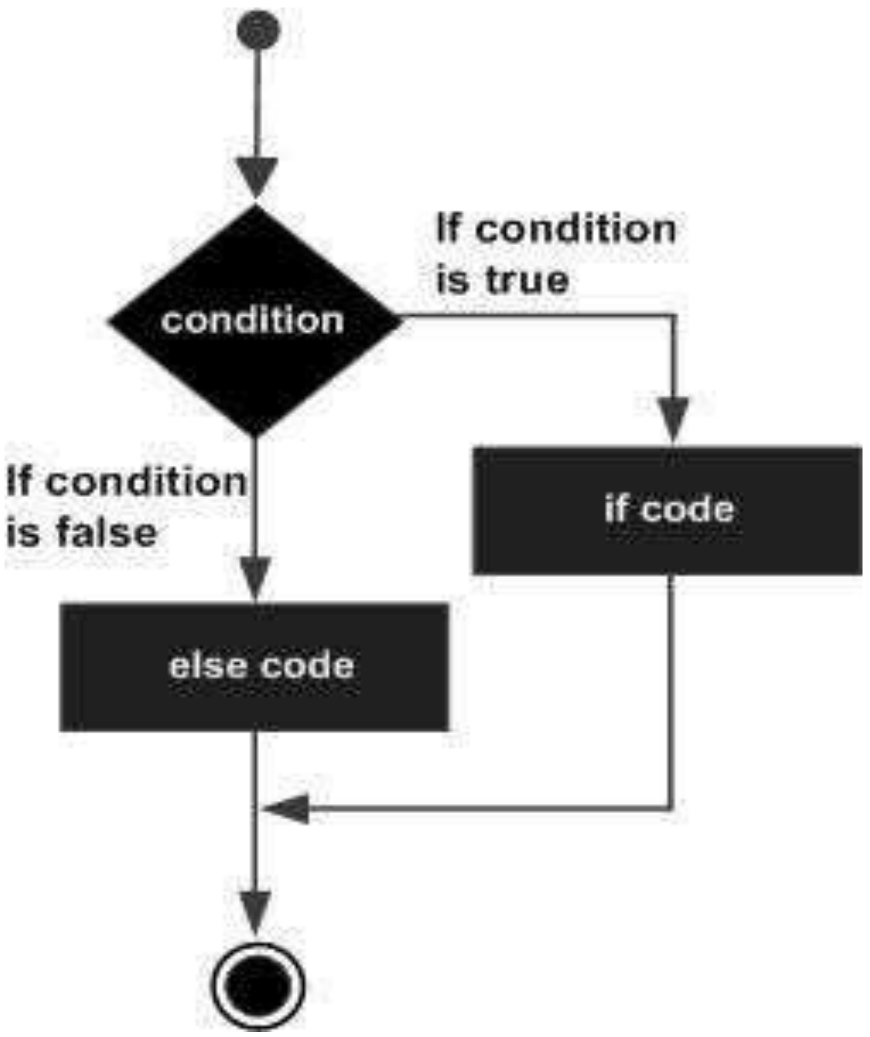


## Simple if

- **Syntax :**

```
if (condition)
{
            //content to be evaluated
        if condition is true
K
```

# If...Else



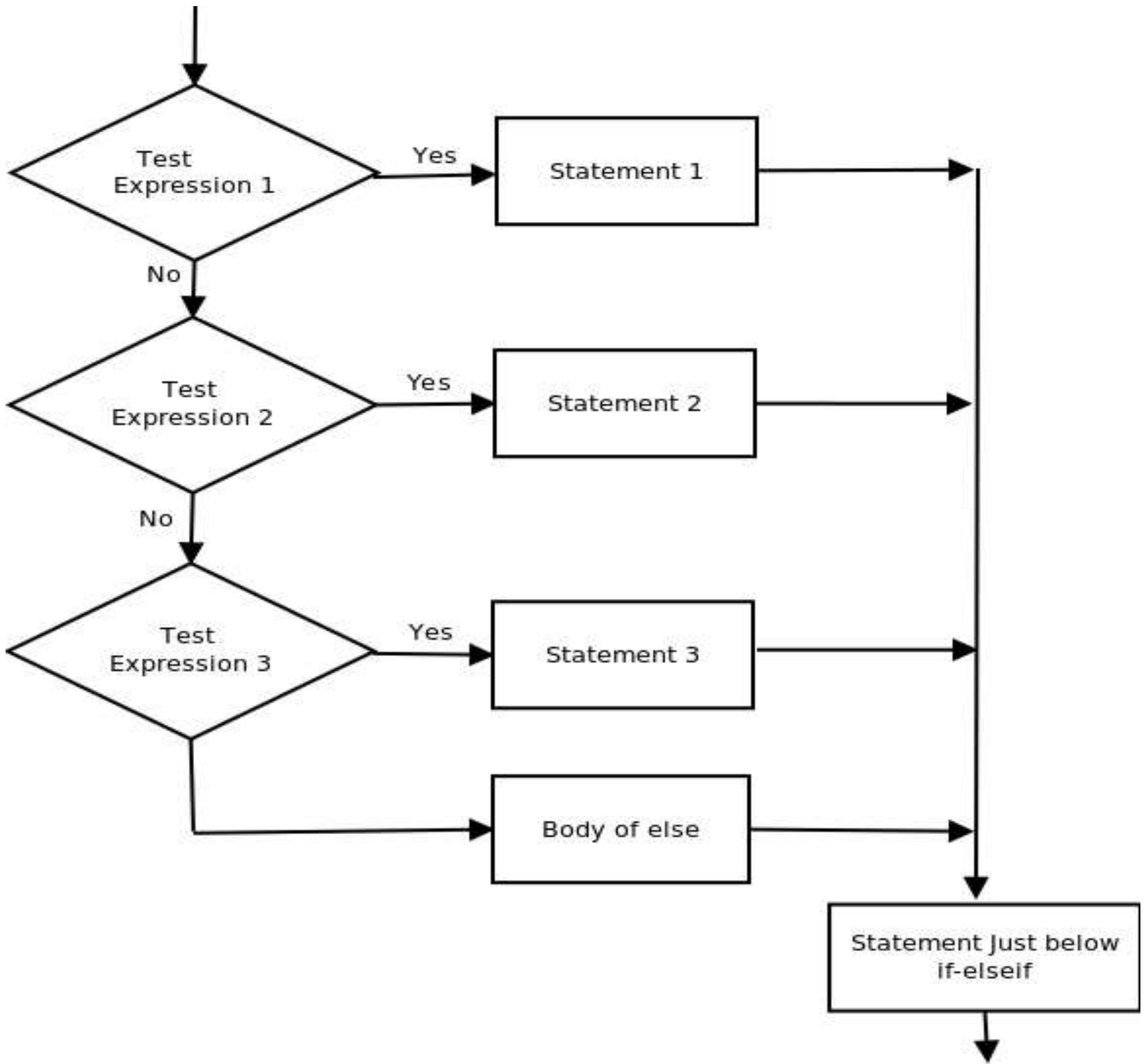
condition
If condition is true
If condition is false
if code
else code

# If...Else

- **Syntax :**

```
if (condition)
{
//content to be evaluated
if condition is true
K
Else
{
//content to be evaluated
          if condition is true
K
```

# If...Else...if statement

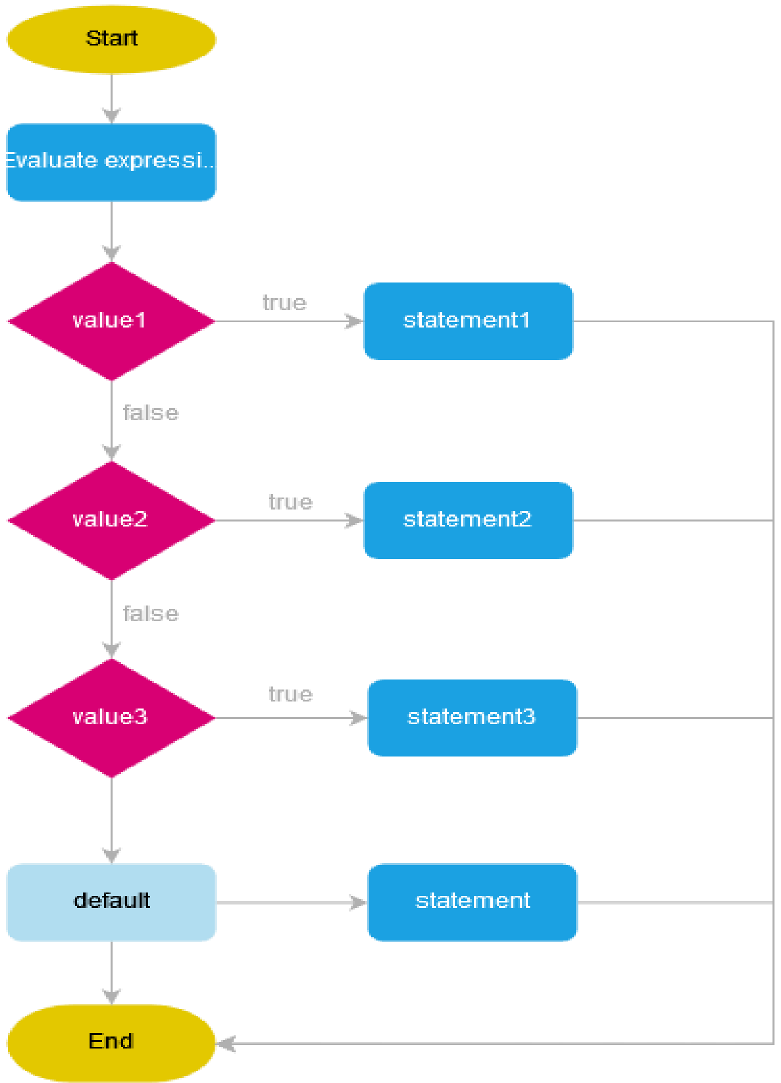


## If...Else...if statement

- **Syntax:**

```
If <Condition> then
            //content to be evaluated
 Else if <condition2>
then
            //content to be evaluated
Else if <condition3>
Then
            //content to be evaluated
Else
            //content to be evaluated
End if;
```

# SWITCH...CASE


Start
Evaluate expressi...
value1
true
statement1
false
value2
true
statement2
false
value3
true
statement3
default
statement
End

## SWITCH...CASE

- SYNTAX:

```
switch(variable/expression) {
                    case value1:
                              // body of case 1,
                              break;
                    case valueN:
                              // body of case N,
                              break;
                     default:
                              // body of default
          }
```

# Looping Statements

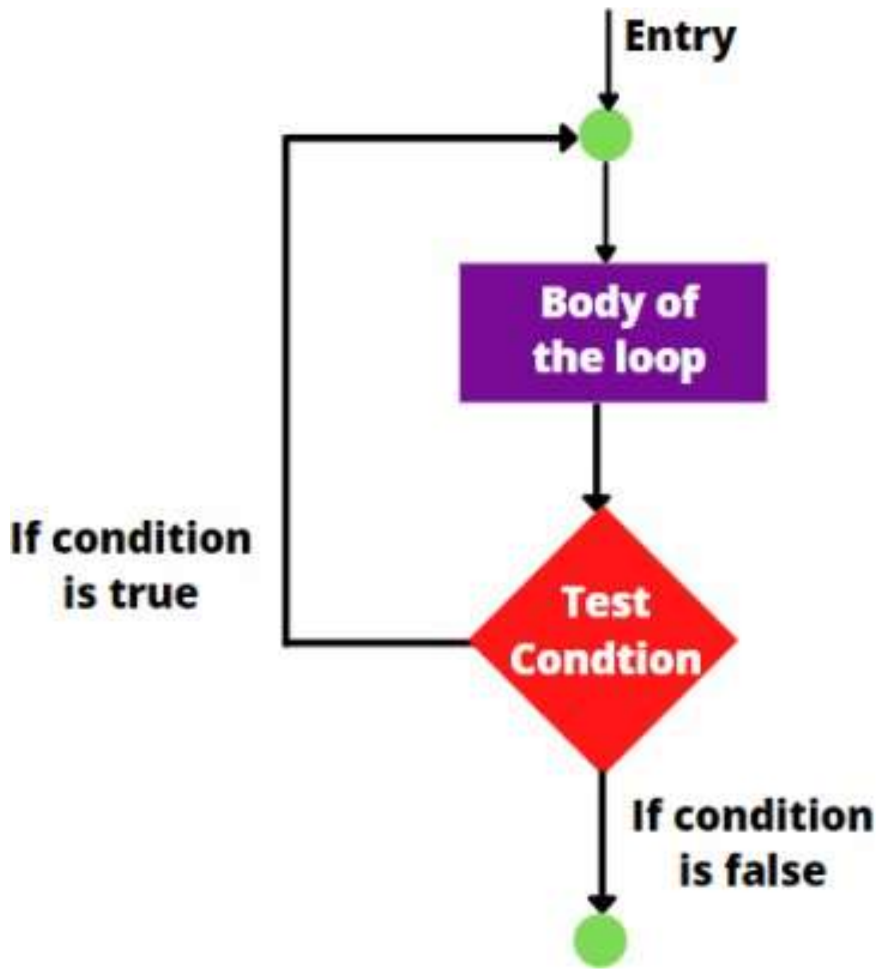
## Looping statements

- In programming, loops are used to repeat a block of code.
- The loops are used to iterate the piece of code using looping statements.
- A block of statements executes sequentially in the loop until a specific condition for the termination of the loop met.
- Therefore, a loop control structure comprises two parts:
  - Control statement
  - Body of the loop

## Looping statements

- There are mainly three types of loops in JavaScript :
  - While loop
  - Do .. While loop
  - For loop

# While Loop

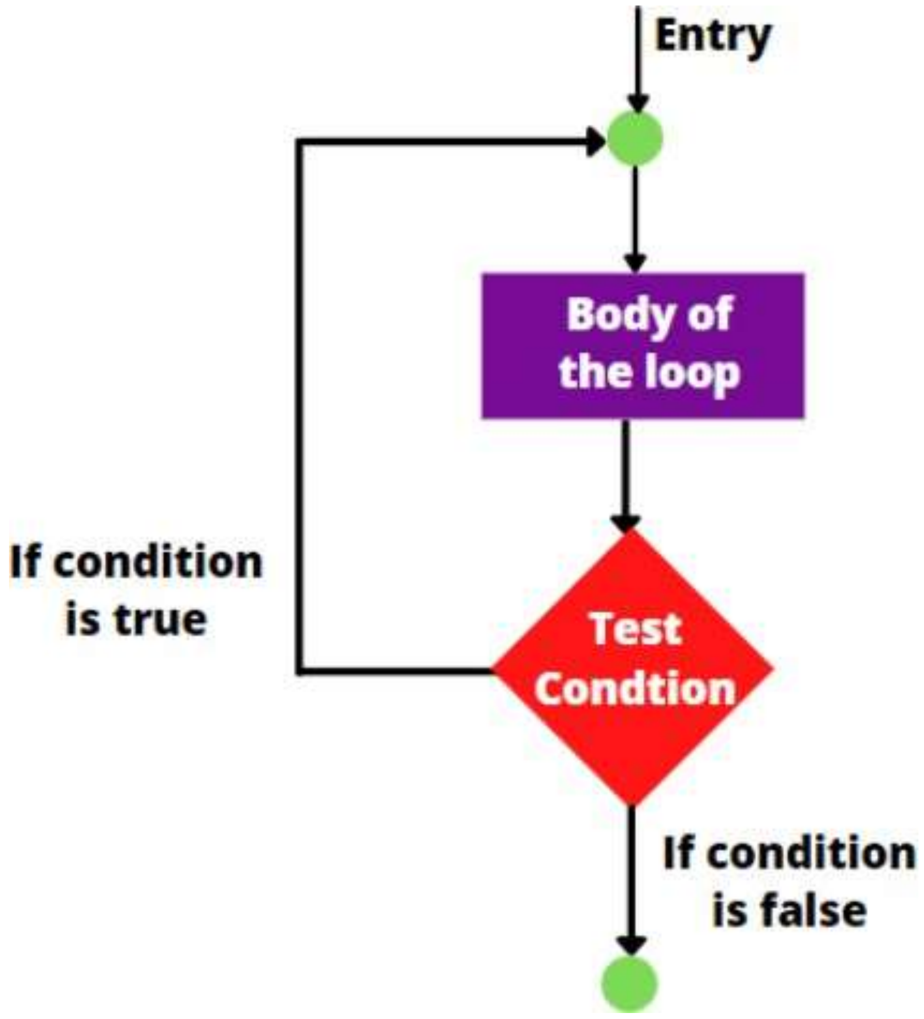

Entry
Body of the loop
If condition is true
Test Condtion
If condition is false

## While Loop

## Syntax:

- while (expression)
- {
- Statement(s) to be executed if
- expression is true; K

## Do ... while

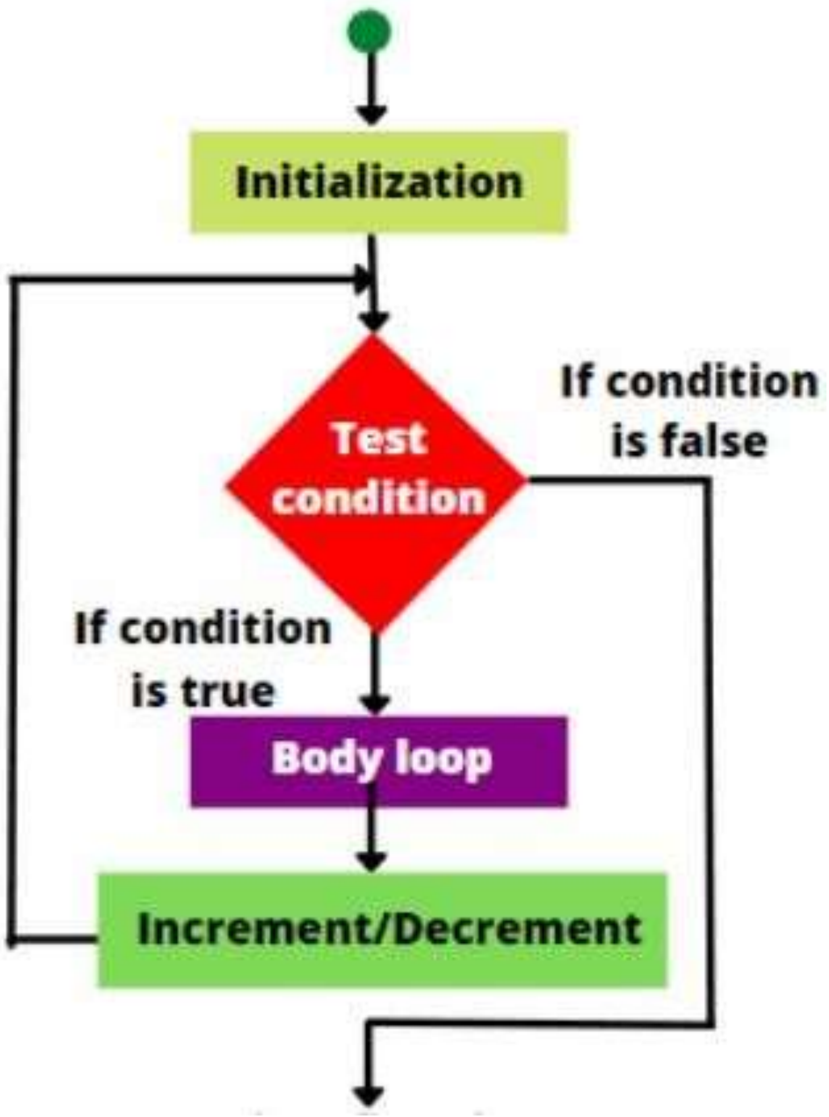

Entry
Body of the loop
If condition is true
Test Condtion
If condition is false

## Do…While

## Syntax:

- do {
- Statement(s) to be executed if expression is true ;
- Kwhile ((expression) ;

# For Loop


Initialization
Test condition
If condition is false
If condition is true
Body loop
Increment/Decrement

## For Loop

## Syntax:

```
for (initialization; condition; iteration)
{
Statement(s) to be executed if condition

is true;

K
```

# BREAK

- The break statement is used to “**jump out**” of a loop or a switch() statement.
- It breaks the loop and continues executing the code after the loop.
- Syntax :
  - ❑ **break [label];** (*label is optional)

## CONTINUE

- The continue statement “**jumps over**” one iteration in loop.
- It breaks iteration in loop and continues executing next iteration in loop.
- Syntax :
    - ❑ **continue [label];** (*label is optional)

## Dialog Box

## Dialog Boxes

- Dialogue boxes are a kind of popup notification, this kind of informative functionality is used to show success, failure, or any particular/important notification to the user.
- JavaScript uses 3 kinds of dialog boxes :
  - Alert
  - Prompt
  - Confirm

## Alert

- An alert dialog box is mostly used to give a warning message to users.
- An alert box is often used if you want to make sure information comes through to the user.
- It has only one 'OK' button to continue and select next task.
- Syntax :

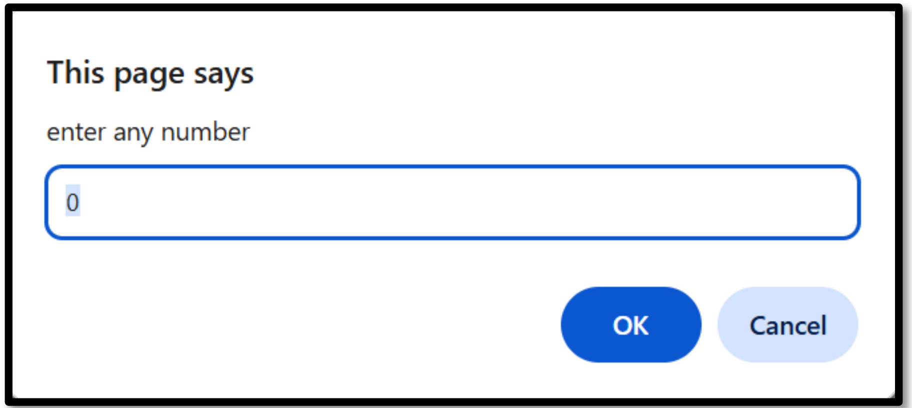
**alert (message);**

This page says

# Prompt Dialog box

- ❑ Prompt dialog box is used when required to pop-up a text box for getting user input.
- ❑ Thus, **it enables interaction with user**.
- ❑ The prompt dialog box also has two buttons, which are OK and Cancel.
- ❑ User needs to provide input in textbox and then click OK.
- ❑ **Syntax :**

**prompt (message, default string);**


This page says

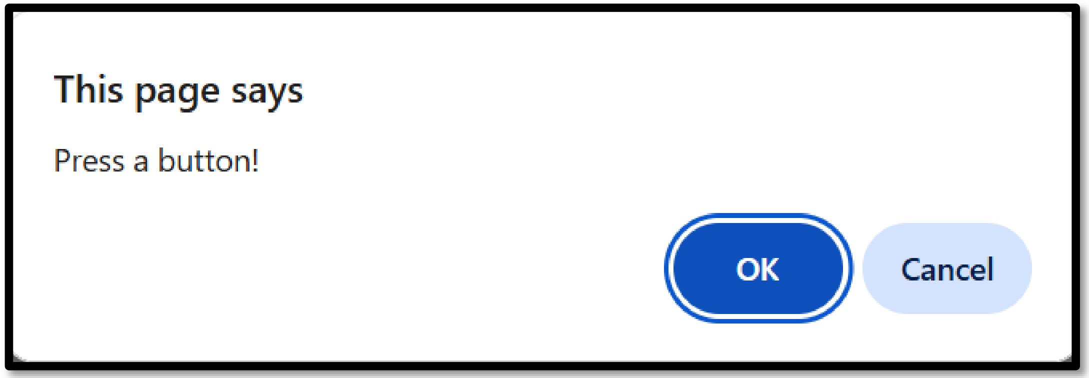
enter any number

0

# Confirm Dialog box

- A confirm box is used for taking opinion from user on the specific option, verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If user clicks on OK button, window method confirm() will return true, & for cancel return false.
- **Syntax :**

**window.confirm("*sometext*");**


This page says

Press a button!

## array

- The Array object lets you store multiple values in a single variable.
- It stores a collection of multiple values of the same type.
- We can create an array using NEW keyword.
- Array use index number for storing set of values & index always begin with 0.
- There are two ways of declaring a JavaScript Array.
  - Using JavaScript Array Literal : let arr = new Array();
  - Using JavaScript new Keyword (array constructor) : let arr = [];

# Using Array Literal

- Array literal way allows us to add elements separated by a comma and enclosed inside square brackets.
- This is the easy as well as the preferred way of declaring JavaScript Arrays.
- **Syntax :**

  **var arrayName = [value1, value2, ...];** or

  **let arrayName = [value1, value2, ...];**

## Using new Keyword

- The arrays are objects in JavaScript; hence they can be created using **new** keyword also.
- Syntax :

❑ **var arrayname;**

❑ **arrayname = new Array();** or

❑ **var arrayname=new Array();**

## Functions

- JavaScript functions are used to perform operations.
- We can call JavaScript function many times to reuse the code.
- JavaScript have two types of functions : **UDF(User Define Function)** and **Built-in**

# Built in functions-string

| FUNCTION NAME | DESCRIPTION |
|---|---|
| **Big()** **SYNTAX:** **string.big( )** | String to be displayed in a big font as if it were in a BIG tag. |
| **Small()** **SYNTAX:** **string.small( )** | string to be displayed in a small font as if it were in a <SMALL> |
| **Bold()** **SYNTAX:** **string.bold( )** | string to be displayed in a bold font |

## Built in functions-string

| FUNCTION NAME | DESCRIPTION |
|---|---|
| ToUpperCase () SYNTAX: String. Touppercase () | Converts string into lowercase. |
| Length() SYNTAX: String.length () | Returns the length of given string |
| ToLowerCase SYNTAX: String. toLowerCase () | Converts string into uppercase. |

# Built in functions-math

| FUNCTION NAME | DESCRIPTION |
|---|---|
| abs()<br>SYNTAX:<br>Math.abs( x ) ; | Returns the absolute value of a number. |
| ceil()<br>SYNTAX:<br>Math.ceil(x) | returns the smallest<br>integer greater than or equal to a number. |
| floor()<br>SYNTAX:<br>Math.floor(x) | returns the largest integer less than or equal to a number. |
| Pow() SYNTAX:<br>Math.pow(base, exponent ) ; | returns the base to the exponent power, that is,<br>base exponent |

## Built in functions-math

| FUNCTION NAME | DESCRIPTION |
|---|---|
| max()<br>SYNTAX:<br>Math.max(value1, value2, ... valueN )<br>; | Returns the maximum no from given numbers |
| min()<br>SYNTAX:<br>Math.min(value1, value2,<br>n... valueN<br>) ; | Returns the maximum no from given numbers |

## Built in functions-math

| FUNCTION NAME | DESCRIPTION |
|---|---|
| parseInt()<br>SYNTAX:<br>parseInt("string"); | This function is intended to converts string value to integer. |
| parseFloat()<br>SYNTAX:<br>parseInt("string"); | This function is intended to converts string value to floating point number. |

# Array functions

| FUNCTION NAME | DESCRIPTION |
|---|---|
| join()<br>SYNTAX:<br>array.join (separator); | joins all the elements of an array into a string. |
| reverse()<br>SYNTAX:<br>Array.reverse() | By using this function we can print elements of array in reverse order. |
| pop()<br>SYNTAX: Array.pop( ) | Remove & return last element of array |
| push()<br>SYNTAX:<br>Array.push (value ) | Add one more element to the end of array & return new length |

# Array functions

| FUNCTION NAME | DESCRIPTION |
|---|---|
| push()<br>SYNTAX:<br>Array.push (value ) | Add one more element to the end of array & return new length |
| sort() SYNTAX:<br>Array.sort | Sorts elements of an array alphabetical order |

## User define Functions

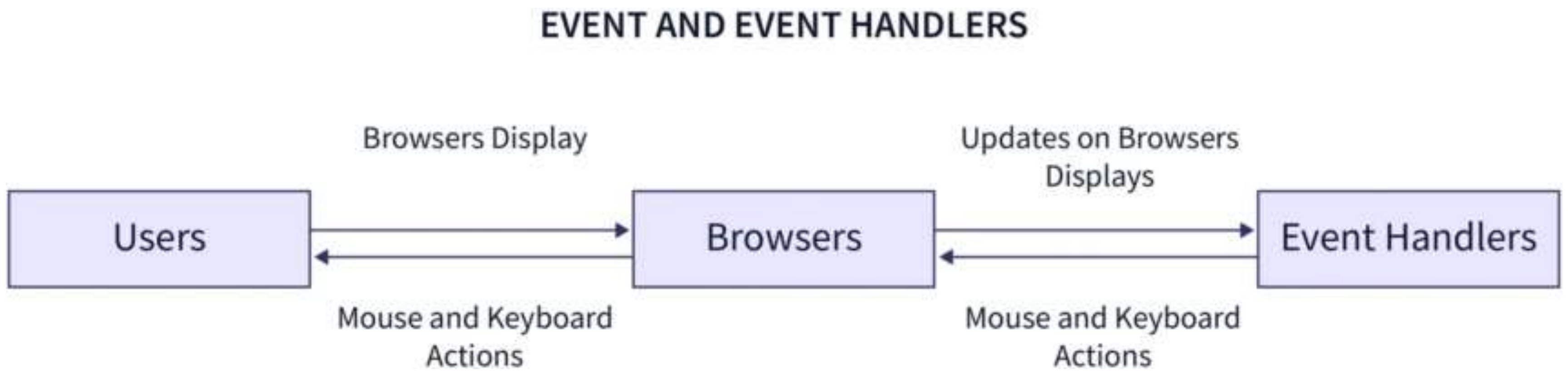
- JavaScript function is a block of code designed to perform a particular task.
- We can call JavaScript function many times to reuse the code.
- JavaScript function is executed when "something" invokes it (calls it).
- Before we use a function, we need to define it.

❑ Syntax:

```
function FunName ([arg1, ...argN]) {
                //some code here....
    }
```

## Event

- Events are actions that happen when a user interacts with page - like clicking an element, typing in a field, or loading a page.


EVENT AND EVENT HANDLERS
Browsers Display
Updates on Browsers Displays
Users
Browsers
Event Handlers
Mouse and Keyboard Actions
Mouse and Keyboard Actions

## events

| EVENT NAME | DESCRIPTION | SUPPORTED BY JAVASCRIPT OBJECTS |
|---|---|---|
| **onclick** | Occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type. | Button, Checkbox, Radio, Reset, Submit |
| **ondblclick** | Occurs when a user double clicks the left button of his mouse. You can put your validation, warning etc., against this event type. | Document, Link |
| **onfocus** | This event occurs when input element get focus | Button, Checkbox,Frame, Password, Radio, Reset, Submit, Text, Textarea, Layer |
| **onblur** | This event Triggers when the window loses focus | Button, Checkbox, Frame, Password, Radio, Reset, Submit, Text, Textarea |

# events

| EVENT NAME | DESCRIPTION | SUPPORTED BY JAVASCRIPT OBJECTS |
|---|---|---|
| **onmouseover** | These two event types will help you create nice effects with images or even with text as well. The onmouseover event triggers when you bring your mouse over any element | Document, Button, Link |
| **onmouseout** | These two event types will help you create nice effects with images or even with text as well. The onmouseout event triggers when you bring your mouse out any element | Document, Button, Link |
| **onmousemove** | This event triggers when use moves mouse pointer | Document, Button, Link |
| **onmouseover** | These two event types will help you create nice effects with images or even with text as well. The onmouseover event triggers when you bring your mouse over any element | Document, Button, Link |

## JQUERY

## WHAT IS JQUERY?

- ❑ **jQuery** is a lightweight, “**write less, do more**” JavaScript library that simplifies web development.
- ❑ jQuery is a lightweight Javascript library which is blazing fast and concise.
- ❑ This library was created by John Resig in 2006.
- ❑ jQuery can be used to find a particular HTML element in the HTML document with a certain ID, class or attribute and later we can use jQuery to change one or more of attributes of the same element like color, visibility etc.
- ❑ jQuery can also be used to make a webpage interactive by responding to an event like a mouse click.

# Download jquery

- There are several ways to start using jQuery on your web site. You can:
  - Download the jQuery library from jQuery.com
  - Use CDN to execute jQuery
- Place jQuery in the same folder where you want to use.
- Use <script> tag  should be be inside the <head> section or <Body> section
- Example :
  - <head>
  - <script src="jquery-3.6.3.min.js"></script>
  - </head>

## JQUERY SYNTAX

- SYNTAX:

```
$(document).ready(function(){
        $(selector).action()
});
```

# JQUERY SYNTAX

- $ sign to define/access jQuery
- (selector) to "query (or find)" HTML elements
- jQuery action() to be performed on element(s)
- Example :
  - $(this).hide() - hides the current element.
  - $("p").hide() - hides all <p> elements.

## Document ready event

- The Document Ready event happens when the web page’s HTML is fully loaded and ready to use, but before things like images or videos finish loading.
- Syntax : $(document).ready()

## Selector

- jQuery Selectors are used to select and manipulate HTML elements.
- They are very important part of jQuery library.
- With jQuery selectors, you can find or select HTML elements based on their id, classes, attributes, types and much more from a DOM.
- All jQuery selectors start with a dollor sign and parenthesis e.g. $().

## Methods

| Name | Description |
|---|---|
| **$(document).ready()** | The $(document).ready() method allows us to execute a function when the document is fully loaded |
| **click()** | The click() method attaches an event handler function to an HTML element.<br>The function is executed when the user clicks on the HTML element. |
| **dblclick()** | The dblclick() method attaches an event handler function to an HTML element. |
| **hover()** | The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods. |
| **focus()** | The focus() method attaches an event handler function to an HTML form field. |
| **blur()** | The blur() method attaches an event handler function to an HTML form field. |

## Events

- ❑ jQuery events are the actions that can be detected by your web application.
- ❑ They are used to create dynamic web pages.
- ❑ An event shows exact moment when something happens.

| Mouse Events | Keyboard Events | Form Events | Document/Window Events |
|---|---|---|---|
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave | | blur | unload |