UR ENGINEERING FRIEND

#1 stop Destination for Diploma & Degree

Live Seminar

Subject – Client Side Scripting

Date – 25th Dec , 2022

Contents

13 .. Form Components

- **a. Text**
- **b. Text Area**
- **c. Checkbox**
- **d. Radio Button**
- **e. Button**
- **f. List**

14.. Form Events

- **a. Mouse Event**
- **b. Key Event**

15.. Form Objects & Elements

16.. Changing option list dynamically

17.. Manipulating Form elements

18.. Disabling elements & Read-only elements

19.. Basics of cookies

20.. Browser

- **a. Opening a new window**
- **b. Window position**
- **c. Change the contents of a window**
- **d. Closing a window**
- **e. Scrolling a window**
- **f. Multiple windows at a glance**

g. Javascript in URLS

21.. Regular Expression

22.. Frame

23.. Rollover

24.. Status bar

25.. Banner

26.. Slide show

27.. Menu

Client Side Scripting

What is Scripting Language ?

All scripting languages are programming languages. The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted. It brings new functions to applications and glue complex system together. A scripting language is a programming language designed for integrating and communicating with other programming languages.

Advantages of scripting languages -:

- **Easy learning:** The user can learn to code in scripting languages quickly, not much knowledge of web technology is required.
- **Fast editing:** It is highly efficient with the limited number of data structures and variables to use.
- **Interactivity:** It helps in adding visualization interfaces and combinations in web pages. Modern web pages demand the use of scripting languages. To create enhanced web pages, fascinated visual description which includes background and foreground colors and so on.
- **Functionality:** There are different libraries which are part of different scripting languages. They help in creating new applications in web browsers and are different from normal programming languages.

Basics of Java Script -:

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

- It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers.
- With JavaScript, users can build modern web applications to interact directly without reloading the page every time.
- The traditional website uses js to provide several forms of interactivity and simplicity.

Features of JavaScript

1. All popular web browsers support JavaScript as they provide built-in execution environments .
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.

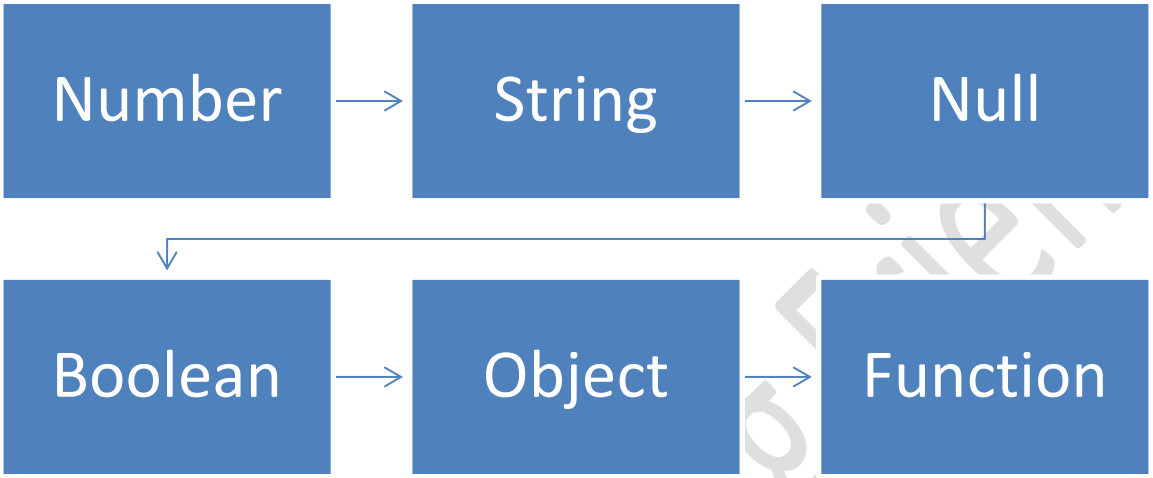
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

How to Write Java Script Code ?

When you see JavaScript code on the Web, you will sometimes see some JavaScript code between the <head></head> tags. Or, you may see it in the <body></body> tags (or even in both places). To separate JavaScript code from HTML code, you need to enclose it within a set of <script></script> tags. The opening <script> tag has one required attribute and one optional attribute. The required attribute is the type attribute, while the optional attribute is **src** (which lets you point to an external script file, covered later in this answer). The value of the type attribute is set to **text/javascript**, as shown below.

```
<script type="text/javascript">
Your JavaScript code goes here.
</script>
```

Data Types Used in Java Script



Variables in Java Script -:

In JavaScript, we can declare a variable in different ways by using different keywords. Each keyword holds some specific reason or feature in JavaScript. Basically, we can declare variables in three different ways by using **var**, **let** and **const** keywords. Each keyword is used in some specific conditions.

1. **JavaScriptvar:** This keyword is used to declare variables globally. If you used this keyword to declare a variable then the variable can accessible

globally and changeable also. It is good for a short length of codes, if the codes get huge then you will get confused.

Syntax:

```
var variableName = "Variable-Value;"
```

2. **JavaScriptlet:** This keyword is used to declare variable locally. If you used this keyword to declare a variable then the variable can accessible locally and it is changeable as well. It is good if the code gets huge.

   Syntax:

   ```
   let variableName = "Variable-Value;"
   ```

3. **JavaScript const:** This keyword is used to declare variable locally. If you use this keyword to declare a variable then the variable will only be accessible within that block similar to the variable defined by using let and difference between let and const is that the variables declared using const values can't be reassigned. So we should assign the value while declaring the variable.

   Syntax:

   ```
   const variableName = "Variable-Value;"
   ```

Operators in JavaScript

JavaScript operators are symbols that are used to perform operations on operands. For example:

1. **var sum=10+20;**

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

(+ , - , * , / , % , ++ , --)

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

(== , !- , >= , <= , > , <)

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

(Logical AND, OR & Not)

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

(= , += , -+ , *= , /= , %=)

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

Conditional Statements

The **if-else** or conditional statement will perform some action for a specific condition. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition. Such control statements are used to cause the flow of execution to advance and branch based on changes to the state of a program.

JavaScript's conditional statements:

- If
- if-else
- nested-if

JavaScript if-statement: It is a conditional statement used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

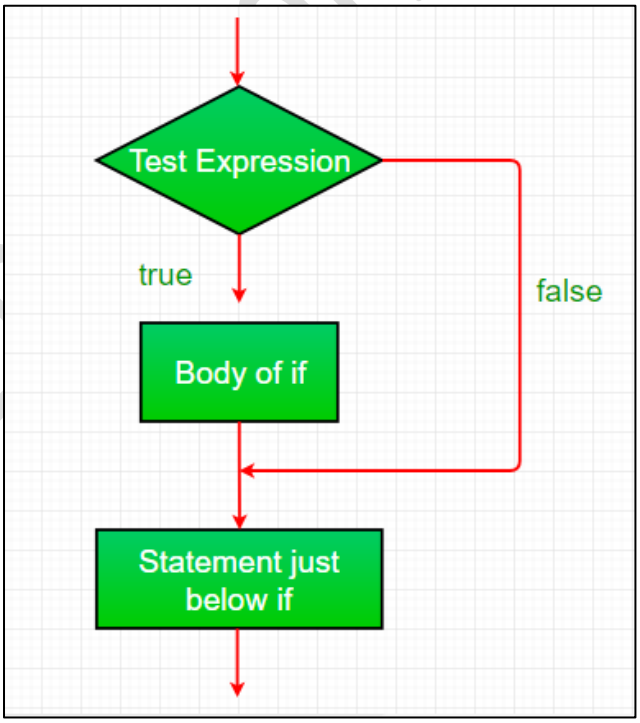
```
if(condition)
{
  // Statements to execute if
  // condition is true
}
```

The if statement accepts boolean values – if the value is true then it will execute the block of statements under it. If we do not provide the curly braces '{' and '}' after **if(condition)** then by default if statement considers the immediate one statement to be inside its block. For example,

```
if(condition)
   statement1;
   statement2;

// Here if the condition is true, if block
// will consider only statement1 to be inside
// its block.
```

Flow chart:

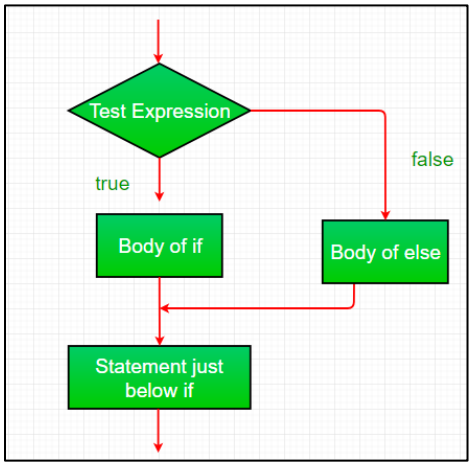


JavaScript if-else statement: The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false? Here comes the else statement. We can use the else statement with the if statement to execute a block of code when the condition is false.

Syntax:

```
if (condition)
{
    // Executes this block if
        // condition is true
}
else
{
        // Executes this block if
        // condition is false
}
```

Flow chart:

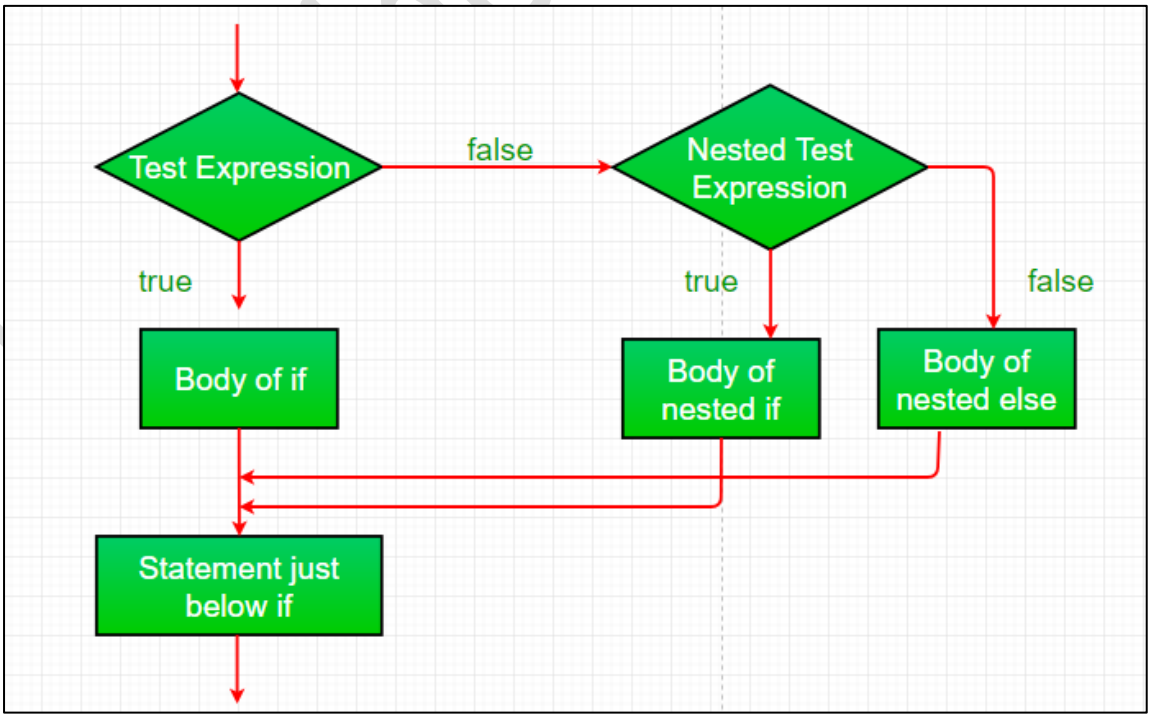


JavaScript nested-if statement: JavaScript allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement. A nested if is an if statement that is the target of another if or else.

Syntax:

```
if (condition1)
{
  // Executes when condition1 is true
  if (condition2)
  {
    // Executes when condition2 is true
  }
}
```

Flow chart:



Switch Case in JavaScript

The **switch case** statement in JavaScript is also used for decision-making purposes. In some cases, using the switch case statement is seen to be more convenient than if-else statements. Consider a situation when we want to test a variable for hundred different values and based on the test we want to execute some task. Using if-else statements for this purpose will be less efficient than switch-case statements and also it will make the code look messy.

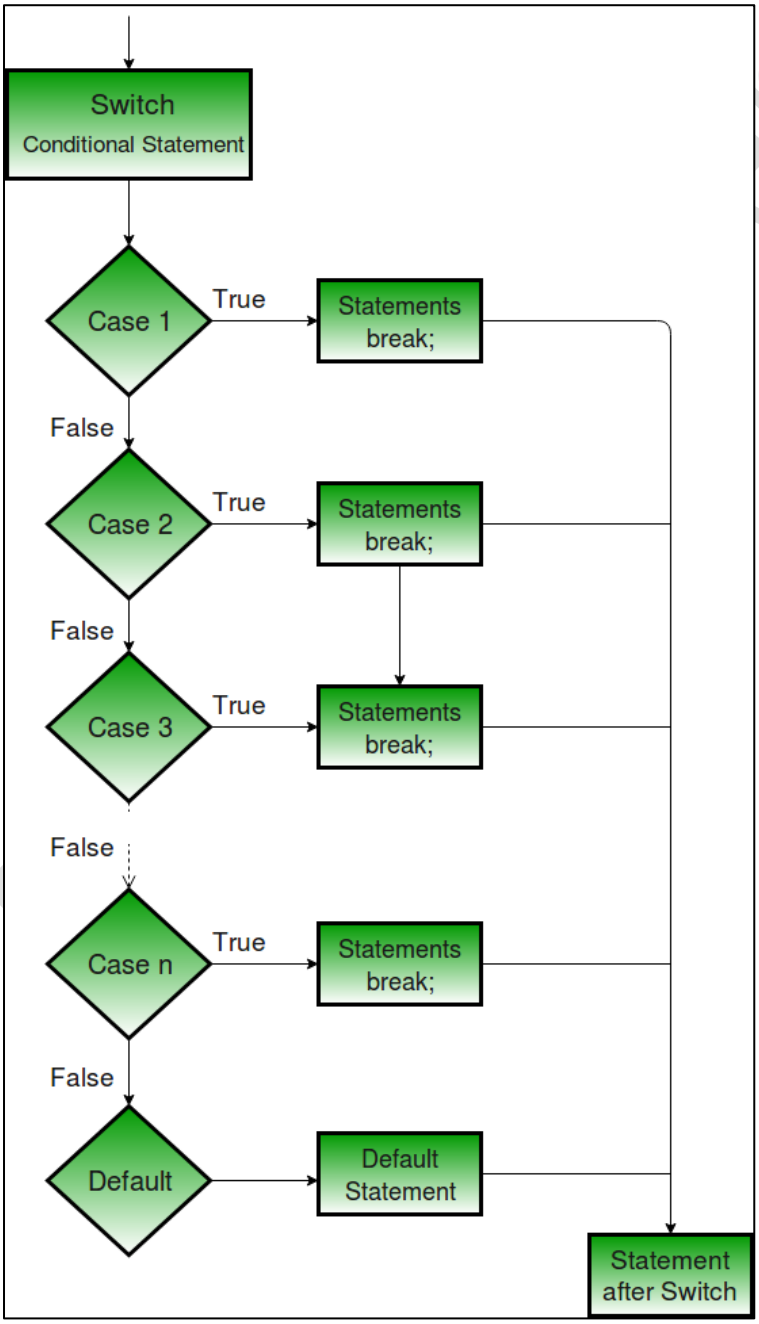
The switch case statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

Syntax:

```
switch (expression)

{
   case value1:
      statement1;
      break;
   case value2:
      statement2;
      break;
   .
  case valueN:
      statementN;
```

```
        break;
    default:
        statementDefault;
}
```



Looping in JavaScript

Looping in programming languages is a feature that facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.

There are mainly two types of loops:

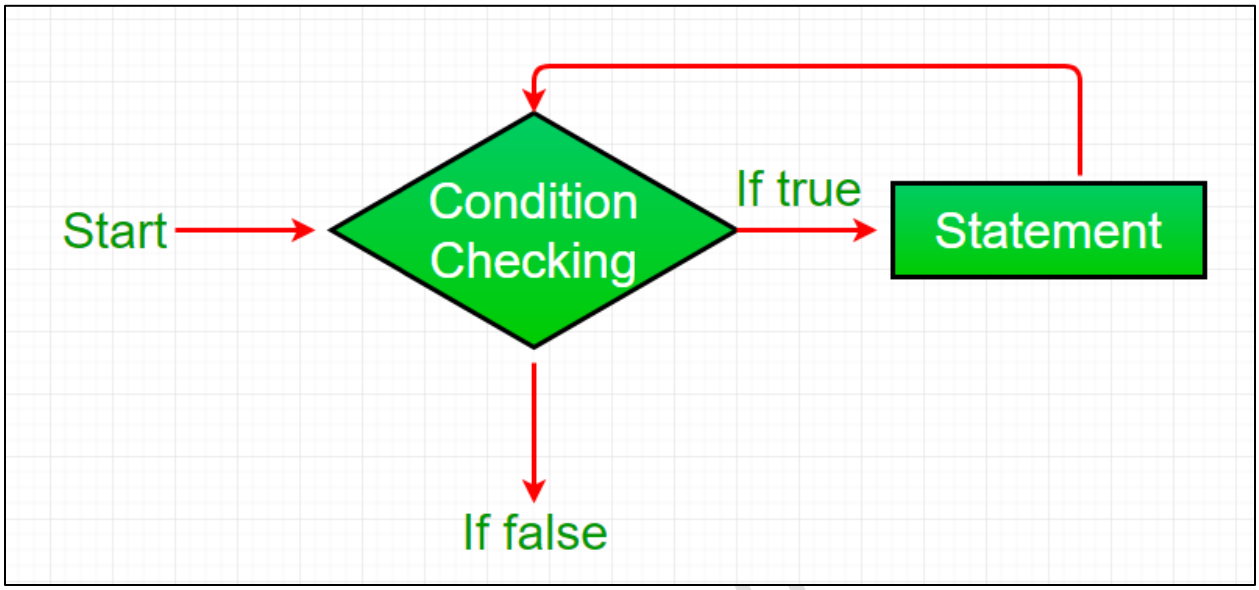
1. **Entry Controlled loops**: In these types of loops, the test condition is tested before entering the loop body. **For Loops** and **While Loops** are entry-controlled loops.
2. **Exit Controlled loops**: In these types of loops the test condition is tested or evaluated at the end of the loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. The **do-while loop** is exit controlled loop.

while loop: A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Syntax :

```
while (boolean condition)
{
  loop statements...
}
```

Flowchart:

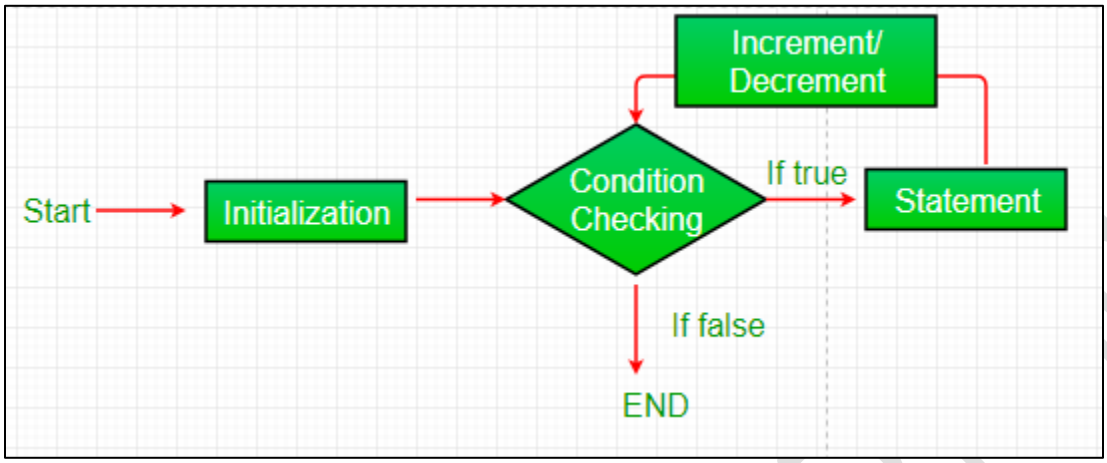


for loop: for loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition, and increment/decrement in one line thereby providing a shorter, easy-to-debug structure of looping.

Syntax:

```
for (initialization condition; testing condition;
                    increment/decrement)
{
   statement(s)
}
```

Flowchart:

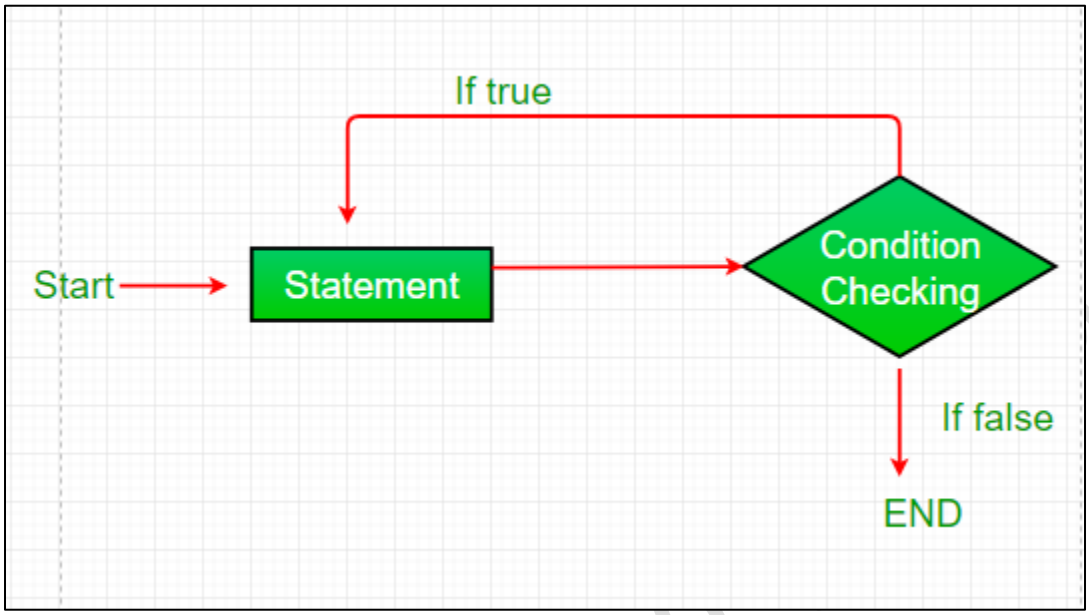


do-while: The do-while loop is similar to the while loop with the only difference that it checks for the condition after executing the statements, and therefore is an example of an **Exit Control Loop.**

Syntax:

```
do
{
   statements..
}
while (condition);
```

Flowchart:



Querying & Handling

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 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")

Array

JavaScript array is a single variable that is used to store different elements. It is often used when we want to store a list of elements and access them by a single variable. Unlike most languages where the array is a reference to the multiple variables, in JavaScript, an array is a single variable that stores multiple elements.

Why do we use arrays?

Arrays help to store multiple values with the same data type in the name of a single variable, otherwise, we have to declare separate variables for each value. However, if you want to loop over a larger number of elements then it is efficient to store the values in an array, instead of writing the same code for each value again and again.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

var arrayname=[value1,value2.....valueN];

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

var arrayname=new Array();

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

var arrayname = new Array(“Ur”, “Engineering”, “Friend”);

Adding an Array Element in JavaScript

Sorting Array in both Order

// program space //

Function

A **function** is a set of statements that take inputs, do some specific computation, and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call that function.

Syntax: The basic syntax to create a function in JavaScript is shown below.

```
function functionName(Parameter1, Parameter2, ...)
{
   // Function body
}
```

Calling function from HTML

String

Will discuss this at the end……..

Events

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

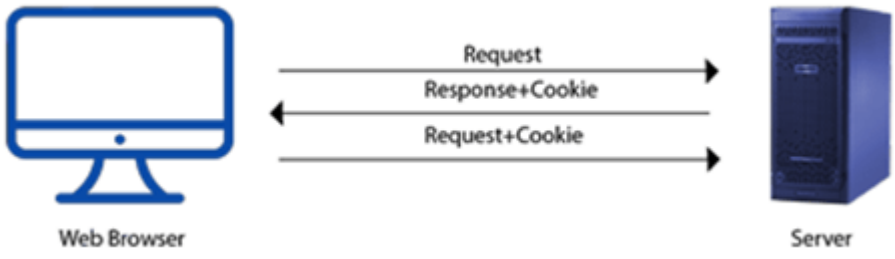
Cookies

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.

How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- So, to recognize the old user, we need to add the cookie with the response from the server.
- browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.



How to create a Cookie in JavaScript?

In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

Cookie Program -:

Window

The open() method opens a new browser window, or a new tab, depending on your browser settings and the parameter values.

JavaScript offers in-built methods to open and close the browser window to perform additional operations like robot window etc. These methods help to open or close the browser window pop-ups. Following are the window methods:

- **open()**
- **close()**

The **window.open** method is used to open a new web page into a new window and **window.close** method to close web page opened by window.open method. See the window.open() method in detail:

Window.open()

It is a pre-defined window method of JavaScript used to open the new tab or window in the browser. This will depend on your browser setting or parameters passed in the window.open() method that either a new window or tab will open.

This method is supported by almost all popular web browsers, like Chrome, Firefox, etc. Following is the syntax and parameters of the window open method

Syntax

This function accepts four parameters, but they are optional.

1. window.open(URL, name, specs, replace);

open() with URL parameter

This is a simple example of window open method having a website URL inside it. We have used a button. By clicking on this button, window.open() method will call and open the website in new browser tab.

Copy Code

1. <html>
2. <body>
3. Click the button to open new window

4. <button onclick="window.open('https://www.urengineeringfriend.in')"> Open Window </button>
5.
6. </body>
7. </html>

Open new window with a name and having a message

We can show any user-defined text or form in new window that we are going to open on button click. For this, we need to provide any name to the new window and write some text into it. This name will pass to the window.open() method. See the code below how it will implement with actual coding.

Copy Code

1. <html>
2. <script>
3. function openWindow() {
4. var newtab = window.open("", "anotherWindow", "width=300,height=150");
5. newtab.document.write(" This is 'anotherWindow'. It is 300px wide and 150px tall new window! ");
6. }
7. </script>
8.
9. <body>
10. Click the button to open the new user-defined sized window
11.


```
12.<button onclick="openWindow()"> Open Window </button>
13.</body>
14.</html>
```

JavaScript Window close method

JavaScript provides an in-built function named **close()** to close the browser window that is opened by using window.open() method. Unlike the window.open() method, it does not contain any parameter. This window.close() method simply close the window or tab opened by the window.open() method.

Remember that - You have to **define a global JavaScript variable** to hold the value returned by window.open() method, which will be used later by the close() method to close that opened window.

Syntax

1. window.close()

JavaScript setTimeout() & setInterval() Method

JavaScript **SetTimeout** and **SetInterval** are the only native function in JavaScript that is used to run code asynchronously, it means allowing the function to be executed immediately, there is no need to wait for the current execution completion, it will be for further execution.

JavaScript setTimeout() Method: This method executes a function, after waiting a specified number of milliseconds.

Syntax:

window.setTimeout(function, milliseconds);

Parameter: There are two parameters accepted by this method

JavaScript setInterval() Method: The setInterval() method repeats a given function at every given time interval.

Syntax:

window.setInterval(function, milliseconds);

Parameter: There are two parameters that accepted by this method

- **function:** the first parameter is the function to be executed
- **milliseconds:** indicates the length of the time interval between each execution.

Regular Expression

A **regular expression** is a sequence of characters that forms a search pattern. The search pattern can be used for text search and text to replace operations. A regular expression can be a single character or a more complicated pattern. Regular expressions can be used to perform all types of text search and text replace operations.

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
g	Perform a global match (find all matches rather than stopping after the first match)
i	Perform case-insensitive matching
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
[x|y]	Find any of the alternatives specified

Frames in JS

HTML Frames are used to divide the web browser window into multiple sections where each section can be loaded separately. A frameset tag is the collection of frames in the browser window.

Creating Frames: Instead of using body tag, use frameset tag in HTML to use frames in web browser. But this Tag is deprecated in HTML 5. The frameset tag is used to define how to divide the browser. Each frame is indicated by frame tag and it basically defines which HTML document shall open into the frame. To define the horizontal frames use row attribute of frame tag in HTML document and to define the vertical frames use col attribute of frame tag in HTML document.

Create Vertical frames:

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <title>Frame tag</title>
5.  </head>
6.    <frameset cols="25%,50%,25%">
7.      <frame src="frame1.html" >
8.      <frame src="frame2.html">
9.      <frame src="frame3.html">
10.   </frameset>
11. </html>
```

Create Horizontal frames:

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <title>Frame tag</title>
5.  </head>
6.    <frameset rows="30%, 40%, 30%">
7.      <frame name="top" src="frame1.html" >
8.      <frame name="main" src="frame2.html">
9.      <frame name="bottom" src="frame3.html">
10.   </frameset>
11. </html>
```

Rollover

Rollover is a JavaScript technique used by Web developers to produce an effect in which the appearance of a graphical image changes when the user rolls the mouse pointer over it. Rollover also refers to a button on a Web page that allows interactivity between the user and the Web page. It causes the button to react by either replacing the source image at the button with another image or redirecting it to a different Web page.

Rollover is triggered when the mouse moves over the primary image, causing the secondary image to appear. The primary image reappears when the mouse is moved away.

Example -:

```
<html>
<head>
<title>JavaScript Image Rollovers</title></head>
<body>
  <a href="link.html"
   onMouseOver="document.image1.src='onImage.gif'"
   onMouseOut="document.image1.src='outImage.gif'">

   <img src="outImage.gif" name="image1">

  </a>
</body>
</html>
```

Status bar

JavaScript gives you the ability to modify the status bar. For example it can be useful to display information about a link, when the user moves his mouse over it or you can display a small amount of information about the page the user is on in the status bar. You can also trick people into clicking a link, so be careful how you use it. If you play too many tricks on your visitors, they might not come back.

Status Bar Example:

```
<html>
  <head>
    <title>JavaScript Status Bar</title></head>
    <body onLoad="window.status='Welcome!';return true">
  </body>
</html>
```

Banner

Displaying banners ads is a common practice for showing advertisements on web pages to the visitors. Banners ads are normally created using standard graphic tools such as Photoshop, Paintbrush Pro, and other software. Banner ads can be static or animated. Animated images are animated GIF files or flash movies. Flash movies are created using Macromedia Flash and the browsers must have installed flash plugin to view the movies. On the other hand, you can create some animated effect using JavaScript, like rotating static banner ads at a certain time interval.

Example

Menu

A website menu is a series of linked items that serve to foster website navigation between the different pages or sections of a site. There are several kinds of menus, depending on the website's content and design. The main types of website menus are:

- **Classic navigation menu:** This most widespread kind of menu is placed in the website's header, typically as a horizontal list.
- **Sticky menu:** Also known as a fixed or floating menu, this menu stays put as visitors scroll down the site. These are ideal for long-scrolling pages.
- **Hamburger menu:** An icon made up of three horizontal stripes that opens up into a menu once clicked. This design convention is rooted in mobile navigation design, but is widely used on desktop as well.
- **Dropdown menu:** A menu in which a list of additional items opens up once visitors click on - or hover over - one of the menu items. This option is suitable for websites with a lot of content.
- **Sidebar menu:** A list of menu items that's located on the left or right side of a webpage.

Example -: