*Java Script*

*Fundamentals of Java Script*

* *Java Script is a dynamic computer programming language used for creating interactive and dynamic webpages.*
* *It is often used with HTML and CSS to add functionality to web pages and create complex web applications*
* *It is an interpreted programming language with object-oriented capabilities.*
* *It is a light weight and most commonly used as a part of webpages, whose implementations allow client-side script to interact with the user and make dynamic pages.*
* *The Java script should be included in or referenced by an HTML document for the code to be interpreted by the bowser.*
* *Java Script can be used to perform variety of tasks, such as*
* *When the user submits the form, and only if all the entries are valid, they would be submitted to the web server.*
* *For user-initiated events such as button clicks, link Navigations etc.*
* *It is used for manipulating the content of webpage.*
* *It is a client side language, means that it is executed in the user’s browser rather than on a webserver, which allows for real-time updates and interacting.*

*Features of JavaScript*

* *JavaScript has a wide range of features that make it a popular and powerful programming language for web development.*
* *Here are some of its key features*

1. *Client-side scripting: JavaScript runs on the client-side which means it is executed by the user's web browser. This enables it to interact with the user, manipulate the webpage's content and create dynamic, interactive web pages.*
2. *Object-oriented programming: JavaScript is a fully object-oriented language, meaning it supports the four fundamental principles of OOP - Encapsulation, abstraction, Inheritance and polymorphism.*
3. *Platform-independent: JavaScript code can run on any platform or device, as long as there is a compatible web browser available*
4. *light Weight: As JavaScript is not a compiled language, Source code never changes to byte code before running time. low end device can also run javascript because of its light weight feature.*
5. *Case Sensitive: In javascript names, variables, keywords and functions are case-sensitive.*
6. *Dynamic typing: Javascript is dynamically typed, which means variable do not have to be declared before use. This makes it easier and quicker to write code*
7. *Built-in debugging tools: Javascript comes with built in debugging tools like console and debugger, which makes it easier to identify and fix error in your code.*
8. *Structured: Javascript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.*
9. *Weakly Typed: Javascript is a weakly typed language where Certain types are implicitly cast*

*HTML <script>tag*

* *The <script> tag is an HTML element used to include Javascript code within an HTML document.*
* *The <script> tag specifies that we are using Javascript*
* *The < Script > tag can be placed in the <head> or <body> Section of the HTML document, depending on the purpose and requirements of the code.*
* *If the script is placed in the <head> section, it will be loaded before the page content, which may cause a delay in rendering the page*
* *If the Script is place in the <body> section, it will be loaded after the page content, which may result in a delay in the execution of the script.*
* *A Simple Syntax for including javascript code in an HTML document using the <script> tag is as follows:*
* *Syntax*

*<script>*

*//Javascript code.*

*</script>*

*Attributes of <script> tag*

* *The <Script> tag takes two important attributes*

1. *Language: This attribute specifies what scripting language you are using. Typically, its value will be javascript*
2. *Type: this attribute specifies the content type that provides Information to the browser about the data.*

* *The <script>tag has several other attributes that can be used to customize its behaviour, such as the src attribute, which is used to include an extend javascript file.*
* *Syntax*

*<script language="javascript" type="text/javascript">*

*//Java script code.*

*</script>*

*WHERE TO WRITE JAVA SCRIPT CODE*

* *JavaScript provides three places to put Javascript code*

1. *Between the body tag of HTML*
2. *Between the head tag of HTML*
3. *External Javascript file*
4. *Between the Body tag of HTML [Inline Javascript]*

* *Example:*

*<html>*

*<head>*

*<title> Example of Javascript </title>*

*<head>*

*<body>*

*<script>*

*console.log("Hello");*

*document. write("World!");*

*</script>*

*</body>*

*</html>*

* *Here- document. write () = function is used to display dynamic content through Javascript.*
* *It is used to write HTML to the document using Javascript*
* *Console.log () function is used to display something to the javascript Console.*

1. *Between the head bag of the HTML [internal javascript]*

* *Example:*

*<html>*

*<head>*

*<title> Head tag </title>*

*<script>*

*document. write ("Hello World!");*

*</script>*

*</head>*

*<body>*

*<p> Example to add Javascript in the head section </p>*

*</body>*

*</html>.*

1. *External Javascript file [External javascript]*

* *We can create external javascript file and embed it in many html pages.*
* *It provides code reusability because single Javascript file can be used in several html pages*
* *An external javascript file must be saved by .js extension.*
* *It is recommended to embed all javascript files into a single file. It increases the speed of the webpage (caching)*
* *Example*

*<html>*

*<head>*

*<title> Java Script </title>*

*</head>*

*<body>*

*<script type="text/javascript" src="C:/Users/admin/ Desktop/ext.js”>*

*</script>*

*</body>*

*</html>*

*ext.js file*

*document. write ("External file");*

*Javascript Variables*

* *Variables are used to stoop data values.*
* *Here are the main rules when naming variable in Javascript*

1. *Variable names are case sensitive. i.e., y and Y are two different variables*
2. *Name must start with a letter (a to z or A to Z), underscore (\_) or dollar ($) sign.*
3. *Variable names should not begin with numbers, and it cannot contain white space characters*
4. *Variable names cannot contain any keywords*

* *In Java script there are 4 ways to declare a variable*

1. *Using var keyword*
2. *using Let keyword*
3. *using const keyword*
4. *Using Var Keyword*

* *The var keyword can be used to declare a variable that is accessible throughout a program, and can be changed.*
* *var is used in the older versions of javascript*
* *Using var we can declare the same variable any number of times*
* *Using var keyword a variable can be declared after it has been used, which means that you can use a variable before it has been declared. This is called as hoisting.*
* *Variable declaration.*

*Syntax: Var variable\_name;*

*e.g., var num;*

* *Variable initializing*

*Syntax: Variable\_name = value; (or) Var Variable\_name = value;*

*e.g., address =`abc'; (or) var age=20;*

1. *Using let Keyword*

* *Let is the new way of declaring variable*
* *Let is block scoped i.e., variable that is defined by let is limited to the block in which it is declared.*
* *We cannot declare the same variable multiple times if it is declared using Let*
* *The Hoisting cannot be done using let.*
* *Variable declaration*

*Syntax: let Variable\_name;*

*e.g.: let address;*

* *Variable initialization*

*Syntax: Variable\_name = value;*

*e.g.: address = "abc";(or) let address = ‘abc';*

1. *Using Const Keyword*

* *Constant is a type of variable whose value cannot be changed i.e., once a constant is initialized, we cannot change its value, and we cannot declare a constant without initializing it.*
* *Syntax: const variable\_name= value;*

*e.g., Const x = 5;*

* *Constant is block Scoped*

*Javascript Data Types*

* *Javascript provides different datatypes to hold different types of values. There are two types of data type in javascript*

1. *Primitive data type*
2. *Non-primitive data type*

* *Javascript is a dynamic type language, means you don't need to specify type of the variable because if is dynamically used by javascript engine.*
* *To specify datatype we use var to specify datatype*

*Primitive datatypes*

|  |  |
| --- | --- |
| *Datatype* | *Description* |
| *String* | *represents sequence of characters e.g., "hello"* |
| *Number* | *represents numeric value eg:100* |
| *Boolean* | *represents boolean value either false or true* |
| *Undefined* | *represents undefined value or un initialized variable.* |
| *Null* | *represents null i.e., no value at all.* |

*Non primitive data types*

|  |  |
| --- | --- |
| *Datatype* | *Description* |
| *object* | *represents instance through which we can access members.* |
| *array* | *represents group of similar values* |
| *Reg Exp* | *represents regular expression.* |

*Javascript Operators.*

* *Java script operators are symbols that are used to perform operations on operands (variables, values and expressions)*
* *for example, Var Sum=10+20;*
* *Here, “+” is the arithmetic operator and = is the assignment operator.*

*10 and 20 are operands*

* *Java script supports the following types of operators*

1. *Arithmetic operators*
2. *Comparison operators*
3. *logical operators*
4. *Assignment operators*
5. *Conditional (ternary) operators*
6. *Bitwise operators*
7. *Arithmetic operators*

* *Arithmetic operators are used to perform arithmetic calculations on numeric values. The most common arithmetic operators in javascript are:*

|  |  |  |
| --- | --- | --- |
| *Operator* | *Description* | *Example a=10, 6=5* |
| *+* | *Addition* | *10+20=30; a+b=15;* |
| *-* | *Subtraction* | *20-10=10; a-b=5;* |
| *\** | *multiplication* | *10\*20= 200; a\*b=50;* |
| */* | *Division* | *20/10=2; a/b = 2;* |
| *%* | *Modulus (Remainder)* | *20%10=0; 10%5=0;* |
| *++* | *increment* | *a++=11;* |
| *--* | *Decrement* | *b--=4;* |

1. *Comparison Operators*

* *Comparison operators are used to compare two values or variable and return a boolean value (true or false)*
* *The most common comparison operators in javascript are.*

|  |  |  |
| --- | --- | --- |
| *Operator* | *Description* | *Example* |
| *==* | *equal to* | *10==20 (false)* |
| *===* | *identical (equal and of same type)* | *10 = 10.00(false)* |
| *!=* | *Not equal to* | *10! =20(true)* |
| *!==* | *not identical* | *20! ==20 (false)* |
| *>* | *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)* |

1. *Logical Operators*

* *logical operators are used to combine two or more boolean values and return a Boolean result. The most common logical operators are:*

|  |  |  |
| --- | --- | --- |
| *operators* | *Description* | *Examples* |
| *&&* | *logical AND* | *(10=20&&20==33)- false* |
| *||* | *logical OR* | *(10==20||20==33)- false* |
| *!* | *logical NOT* | *! (10==20) -true* |

1. *Assignment operators*

* *Assignment operators are used to assign a value to a variable. The most common assignment operators are:*

|  |  |  |
| --- | --- | --- |
| *Operators* | *Description* | *Example a=20* |
| *=* | *assignment* | *a=20* |
| *+=* | *add and assign* | *a+=20; a=40* |
| *-=* | *Subtract and assign* | *a-= 20; a=0* |
| *Operators* | *Description* | *example a=20* |
| *\*=* | *multiply and assign* | *a\*= 20; a=400* |
| */=* | *divide and assign* | *a = 10; a/= 2; a = 5* |
| *%=* | *modules and assign* | *a = 10; a % = 2; a = 0* |

1. *Conditional (ternary) operators*

* *Conditional operator returns value based on the condition*
* *operators - (? :)*
* *Syntax: exp1? Exp2: exp 3;*

1. *Bitwise operators*

* *Bitwise operators are used to perform bitwise operations on integer values. The most common bitwise operators are:*

|  |  |  |
| --- | --- | --- |
| *Operator* | *Description* | *Example* |
| *&* | *Bitwise AND* | *(10==20&20==33)-false* |
| *|* | *Bitwise OR* | *(10==20|20==33)- false* |
| *^* | *Bitwise XOR* | *(10==20^20==33)-false* |
| *~* | *Bitwise NOT* | *(~10) =-10* |
| *<<* | *Bitwise left shift* | *(10<<2) =40* |
| *>>* | *Bitwise right shift* | *(10 >>2) =2* |
| *>>>* | *Bitwise right shift with zero* | *(10>>>2) =2* |

*Javascript Control Statements*

* *Java Script provides various decision Statements that help us to Control the flow of the program based on certain conditions.*
* *The most commonly used decision statements are as follow:*

1. *if statement*

* The if statement evaluates the given test expression inside the Parenthesis ()
* The expression may represent a relations expression, a logical expression, a numeric variable or constant.
* The expression evaluates to a zero (false) or non-zero (True) value.
* If the test expression is evaluated to true, then statements Inside the body of if are executed.
* If the test expression is evaluated to false, then statements inside the body of it are not executed.
* The condition in if can be a simple expression or compound expression.

Syntax

if (this condition is true)

{

execute this block of statements;

}

* Note that opening and closing Braces are required only when code after if statement occupies multiple lines.
* Note that if is in lowercase letters. Uppercase If will generate an error.

2) if else statement

* if else statement is a combination of both if and else statements.
* Using if statement we execute one group of statements if the expression evaluates to true and
* using else statement, we execute another group of statements if the expression evaluates to false.
* The if statement executes a statement, when the specified expression evaluates to a non-zero value. It does nothing when it evaluates to a zero value.
* Where as in if else statement there are two types of blocks: true and false.
* If the condition given in if evaluates to a non-zero value or evaluated to true the statements inside the body of if are executed and the else part is bypassed.
* If the condition given in it evaluates to a zero or evaluated to false the statement inside the body of else are executed and the if part is by passed.
* The group of statements after the if up to and not including the else is called an ‘if block’. Similarly, the statements after the else form the ‘else block’.
* Note: the else is written exactly below the if.

Syntax

If (this condition is true)

{

execute this block of statements;

}

else

{

execute this block of statements;

}

3) Nested if statement

* It is perfectly all right if we write an entire if-else construct within either the body of the if statement or the body of an else statement. This is called ‘nesting’ of ifs.
* Nested if statements refer to it statements that are contained within Other if statements.
* If statement can be contained within another if statement
* The inner statement will be executed if expression of outer if statement evaluates to non-zero value.

Syntax

If (Condition 1)

{

If (Condition 2)

{

Block of statements;

}

else {

Block of statements;

}

}

else

{

Block of statements;

}

4) if-else ladder (if-else if ladder)

* The if else statement executes two different codes depending upon whether the test expression is true or false. Sometimes, a choice has to be made from more than 2 possibilities.
* The If else ladder allows you to check between multiple test expressions and execute different statements.
* The expressions are evaluated in order, and if any expression is true then the statement associated with the block is executed, and terminate whole loop.
* If none of the specified expressions are satisfied then the last else part or default case is executed.

Syntax

If (condition 1)

Statements 1;

else if (condition 2)

Statements 2;

else if (condition 3)

Statements 3;

-

-

else if (condition n)

Statements n;

else

Statements;

5) Switch statement

* The control statement that allows us to make a decision from the number of choices is called a switch, or more correctly a switch-case default, since these three keywords go together to make up the control statement.
* The switch statement provides an alternative to else if statement.
* A case expression can be repeatedly used in a switch statement.
* Switch statement works by testing desired case among all the given Cases
* When the case is found the block of statement associated with that Case is executed.
* The use of break statement in every case is used to quit the switch Statement after a particular case is matched. Thus, only one case gets executed because break takes control out of the loop.
* If The break statement is not used, then all the statements following the matched cave will get executed.
* If the value of expression does not match any of the case, control goes to the default keyword. And execute the statements.
* If there is no default keyword, the whole switch statement Simply terminates when there is no match.

Syntax

switch (integer expression)

{

case constant 1: Statement 1;

break;

case constant 2: Statement 2;

break;

default: Statements;

}

* The integer expression following the keyword switch is any expression that will yield an integer value. It could be an integer constant like 1, 2 or 3, or an expression that evaluates to an integer.
* The keyword case is followed by an integer or a character constant. Each constant in each case must be different from all the others.
* When we run a switch case statement, First, the integer expression following the keyword switch is evaluated.
* The value it gives is then matched, one by one, against the constant values that follow the case statements. When a match is found, the program executes the statements following that case,
* If no match is found with any of the case statements, only the statements following the default are executed.

Note:

* there is no need for a break statement after the default, since the control comes out of the switch anyway.
* Even if there are multiple statements to be executed in each case there is no need to enclose them within a pair of braces.
* Every statement in a switch must belong to some case or the other. If a statement doesn’t belong to any case the compiler won’t report an error. However, the statement would never get executed.
* If we have no default case, then the program simply falls through the entire switch and continues with the next instruction (if any,) that follows the closing brace of switch.

*Javascript loops*

* *Javascript looping statements are used for running a particular set of code for any number of times.*
* *Loops are used to repeat a block of code until a specified condition is met.*
* *The various looping statement supported are as follows*

1. *while loop*
2. *for loop*
3. *do while loop*

* Three things loop contain

1. Setting a loop counter to an initial value. (Initialization of variable)
2. Testing the loop counter to determine whether its value has reached the number of repetitions desired. (Testing the condition)
3. Increasing the value of loop counter each time the program segment within the loop has been executed. (Incrementation/decrementation)

1) while loop

* It is often the case in programming that you want to do something a fixed number of times. The while loop is suited for such cases.
* The while statement used when it is not known in advance that how many times a Statement or statement-block will be executed.
* The main function of the while statement is to repeat a particular set of statements. For a fixed number of times till condition is met
* In the while statement the initialization of looping variables, the testing of a condition and incrementation of loop variable occurs at different place in the program.
* There must be a statement prior to while statement that initializes the expression and, in the statement block, there must be Statement that modifies (i.e., Increment /decrement) the expression.

Syntax

initialise loop counter;

while (test loop counter using a condition)

{

Statements;

Modifications loop counter;

}

* The while loop contains expression or test condition
* The statement is executed repeatedly till the expression evaluate to a non-Zero value
* firstly, the value of variable (say i) is set to an initial value 1. When the while statement is executed for the first time, the condition is tested.
* If the condition is satisfied and then the body of the loop is executed for the first time.
* Then the value of I variable gets incremented/decremented
* Upon reaching the closing brace of while, control is sent back to the while statement.
* Again, the test is performed to check whether the new value of i satisfy the condition
* If the value of i satisfy the condition, the statements within the braces of while are executed again. The body of the while loop continues to get executed till the condition being tested remains true. when the condition becomes false, the control passes to the first statement that follows the body of the while loop.
* NOTE

1. The variable 'i' is many a times called either a ‘loop counter’ or an ‘index variable’.
2. Other than condition in place of the condition there can be any other valid expression. So long as the expression evaluates to a non-zero value the statements within the loop would get executed.
3. The condition being tested may use relational or logical operators
4. The statements within the loop may be a single and a block of statements
5. As a rule, the while must test a condition that will eventually become false otherwise the loop would be executed forever, indefinitely.
6. It is not necessary ta loop counter must only be an int. It can also be a float. Even floating-point loop counters can be decremented.
7. Once again, the increment and decrement could be by any value, not necessarily 1.
8. if we use i++ ' ++' operator is called a post incrementation operator. With this, incrementation of i happens after its usage or comparison.
9. if we use ++i '++' operator is called a pre incrementation operator. With this, incrementation of i happens before its usage or comparison.

2) for loop

* The for statement is used for problems where the number of times a statement or statement -block will be executed is known in advance
* For statement can be treated as a compact form of while statement.
* The for allows us to specify three things about a loop in a single line:

1. Initialization of variable

2. Testing the condition

3. Incrementation/decrementation.

Syntax

for (initialise counter; test counter; increment counter)

{

Statements;

}

* When the for statement is executed for the first time, the value of variable (say i) is set to an initial value 1.
* Then the condition is tested. If the condition is satisfied and the body of the loop is executed for the first time.
* Upon reaching the closing brace of for, control is sent back to the for statement, where the value of i variable gets incremented/decremented
* Again, the test is performed to check whether the new value of i satisfy the condition. If the value of i satisfy the condition, the statements within the braces of for are executed again.
* The body of the for loop continues to get executed till the condition being tested remains true. when the condition becomes false, the control passes to the first statement that follows the body of the for loop.
* In the initialization and update expression parts can have more than one statement separated by a comma.

Note:

* The initialisation expression of the for loop can contain more than one statement separated by a comma.

For example,

for (i = 1, j = 2; j <= 10; j++)

* Multiple statements can also be used in the incrementation expression of for loop;

i.e., you can increment (or decrement) two or more variables at the same time.

3) do while loop.

* The do-while statement is also used when it is not known in advance that how many time a statement will be executed.
* The do-while bop is similar to the while statement, but the only difference between the two is that the testing condition is present at the end of do while statement; whereas the while statement in which It is present at the beginning.
* The body of do while loop is executed at least once only then, the test expression is evaluated.

i.e., the body of loop is executed once though the condition is false.

Syntax

do

{

//statements;

} while (test expression);

* The do while loop executes at least once i.e., the first iteration runs without checking the condition. The condition is checked only after the first iteration has been executed.
* If the condition is true, then body of loop is executed again. The Statements are executed repeatedly till the exp evaluates to non-zero value
* When condition is false the loop is terminated.

*Javascript functions:*

* *A function is a block of code that performs a specific task and can be called and reused throughout a program.*
* In other words, A function is a block of code which only runs when it is called.

*Advantages of functions*

* Increases program readability: separating the code into smaller parts or modules makes the program easter to understand
* Avoid repetition of codes: writing functions avoids rewriting the same code over and over,it saves time and memory.
* Reduces chances of error: Using functions it becomes easier to write programs and keep track of what they are doing
* Reusability of code: using function we define the code Once and use it many times it saves time and memory.
* Modifying a program becomes easier by using function.

*Function declaration*

* *Functions in Javascript can be defined using the "function" Keyword followed by the name of the function and a set of parentheses that may contain parameters.*
* *The function body is enclosed in curly braces { } and contain the code that will be executed when the function is called.*
* *Note the functions can have 0 or more parameters.*
* *The following is the syntax of declaring function in Javascript.*
* *Syntax:*

*function function\_name ([arg1, arg2, - - argn])*

*{*

*//local declarations*

*//Code to be executed.*

*}*

* *Note: The variables in function call are called arguments and variables in function definition are called parameters.*
* *Example of function with no parameters.*

*<script>*

*function msg ()*

*{*

*alert ("hello!");*

*}*

*</script>*

*msg ();*

* *Example of function with parameter and arguments.*

*<script>*

*function getcube (num)*

*{*

*alert (num\*num\*num);*

*}*

*</script>*

*getcube (4);*

*Java Script Array*

* *Array is a derived datatype that is a collection of similar type of elements.*
* *In Javascript the Array allows you to store multiple values of any datatype in single variable.*
* *In Javascript Arrays can be constructed in two ways.*

1. *By array literal*
2. *By creating Instance of Array directly using new keyword.*
3. *By array Literal*

* *To define an array by literal, we use the square bracket notation [], and separate each value with a comma.*
* *Syntax*

*Var array\_Name = [value1, Value 2, - - - Value N];*

*E.g.: var sections = ["A", "B", "C", "D"];*

* *By new keyword*
* *Syntax of creating array directly*

*Var array\_Name = new Array ();*

*Accessing Element of an array*

* *Arrays in Javascript are zero-indexed, that means that the first element in the array has an index of 0, the second element has an of 1 and so on and the last index value is n-1.*
* *You can access an element in an array by referring to its index in square brackets.*
* *Syntax: array\_Name [index value];*

*Array methods*

* *Arrays in javascript have a number of built-in methods that allow you to manipulate and transform the data they contain.*
* *Some of the most commonly used array methods are as follows:*

1. *push ()*
2. *pop ()*
3. *shift ()*
4. *unshift ()*
5. *slice ()*
6. *splice ()*
7. *is Array ()*
8. *index of ()*
9. *push (): By using push function we can add one or more elements to the end of an array.*

* *Syntax: array\_Name. push ();*
* *e.g., sections. push ();*

1. *pop (): By using pop function we can delete the last element of an array*

* *Syntax: arrgy\_Name.pop ();*
* *e.g., sections. pop ();*

1. *shift (): By using shift function, we can delete the first element of an array.*

* *Syntax: array\_Name. shift ();*
* *e.g., Sections. shift ();*

1. *Unshift (): By using unshift function, we can add one or more elements in the beginning of the array.*

* *Syntax: array\_name. unshift ();*
* *E.g., section. unshift ();*

1. *Slice (): This function is used to get a new array by selecting a subarray of a given array.*

* *Syntax: array\_name. Slice (Starting index, ending index);*
* *e.g., Section. Slice (0,2);*

1. *Splice () ÷ This method is used to add/remove an item from given array*

* *Syntax: array\_name. Splice (Starting index, no. of items to be removed, item1, item 2, - -, item n);*
* *E.g., section. splice (1,2, 'E', 'F');*

1. *isArray (): This function is used to test if the passed value is an array or not.*

* *Syntax: Array. isArray (Array\_name);*
* *e.g., Array. isArray (section);*

1. *indexof (): This method is used to search the specified element In the given array and returns the index of the first match.*

* *Syntax: array\_name, index of (value);*
* *e.g.: section. indexof ("c");*

*Javascript events*

* *The change in the state of an object is known as an event.*
* *Javascript interaction with HTML is handled through events that occur when the user or the browser manipulates a page*
* *when Javascript code is included in HTML, Javascript react over these events and allow the execution.*
* *The process of reacting over the event is called Event Handling.*
* *Javascript handles the HTML events via Event Handlers.*
* *Event handles can be added to HTML element using the attribute, followed by the name of the event and the code to be executed when the event occurs.*
* *The following are the most commonly used event handlers.*

|  |  |
| --- | --- |
| *Event Handler* | *Description* |
| *Mouse events*  *onclick* | *when mouse click on an element.* |
| *Onmouseover* | *when the cursor of the mouse is moved over an element.* |
| *Onmouseout* | *when the cursor of the mouse is moved out from an element.* |
| *Onmousedown* | *when mouse button is pressed over the element.* |
| *Onmouseup* | *when mouse button is released from an element.* |
| *Onmousemove* | *when mouse movement takes place.* |
| *Keyboard events*  *onkeydown* | *when key is being pressed.* |
| *Onkeyup* | *when key is released.* |
| *Onkeypress* | *when user presses the key* |
| *form Event*  *onfocus* | *when the user focusses on an element.* |
| *Onsubmit* | *when the user submits the form.* |
| *Onreset* | *when the user reset the form.* |
| *Onblur* | *when the focus is away from a form element.* |
| *Onchange* | *when the user changes the value of form element.* |

*Browser Object Model*

* *Browser object Model (BOM) is used to interact with the browser.*
* *The default object of browser is window.*
* *i.e., you can call all the functions of window by specifying window or directly.*
* *There are lot of properties defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerwidth.*

1. *window object*

* *The window object represents a window in browser*
* *methods of window object*
* *The important methods of window object are as follows*

|  |  |
| --- | --- |
| *Method* | *Description* |
| *alert ()* | *displays the alert box containing message with ok button.* |
| *confirm ()* | *displays the confirm dialog box containing message with ok and cancel button.* |
| *prompt ()* | *displays a dialog box to get input from the user.* |
| *open ()* | *opens the new window [or the window whose link is provided].* |
| *close ()* | *closes the current window.* |
| *Set Timeout ()* | *Performs action after specified time*  *Like calling function, evaluating expressions etc.* |
| *Set Interval ()* | *performs action with a specific interval.* |

*Document object/Document object Model (DOM)*

* *A document object represents the HTML document that is displayed In that window.*
* *Document object model defines the way a document is accessed and manipulated.*
* *The document object has various properties that refer to other objects which allow access to and modification of document content.*
* *By the help of document object, we can add dynamic content to our webpage.*

*methods of Document objects*

|  |  |
| --- | --- |
| *method.* | *Description* |
| *document.write()* | *writes the given string on the document.* |
| *document.getElementById()* | *returns the element having the given id value.* |
| *document.getElementsByName()* | *returns all the elements having the given name value.* |
| *document.getElementsByTagName()* | *returns all the elements having the given tag name.* |
| *document.getElementsByClassName()* | *returns all the elements having the given class name.* |