**JAVA SCRIPT**

1. **OVERVIEW**

**Java script:**

**dynamic computer** programming language

implementations allow client-side script (add interactivity of the webpage)

**interpreted** programming language ( object-oriented)

**Live Script (1995)--- Netscape change to ---Java script**

**Year -**

**Extension -** .js

**Developer -**

**Advantages:**

lightweight, interpreted programming language

network-centric applications

integrated with Java

integrated with HTML

open and cross-platform

**Client side JavaScript:**

Client side js **referenced by html**

It is does **not need static html** , but this way interact user easily in browser

provides **CGI server-side scripts**(like email validation)

**Advantages:**

**Less server interaction**:

validate user input before sending the page to server

**Immediate feedback to the visitors**:

**Increased interactivity:**

create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard

**Richer interfaces:**

dragand-drop components and sliders to give a Rich Interface to your site visitors.

**Limitation**:

does not allow the reading or writing of files(security reason)

cannot be used for networking applications( no such support available)

doesn't have any multithreading or multiprocessor capabilities.

1. **SYNTAX**

**Syntax:**

<script language="java script" type="text/java script">

JavaScript code

</script>

Language:

Not necessary in advanced html.

Type:

indicate the scripting language

**document.write** :

which writes a string into our HTML document

Ex:

document.write ("Hello World!")

**Whitespace and Line Breaks:**

Not necessary space , next line , tab (these are used to get great indentation for understanding purpose )

**Semicolons are Optional:**

Ex:

var1 = 10

var2 = 20 (one way)

var1 = 10; var2 = 20;(another way)

**Case Sensitivity:**

Ex:

Time - TIME - time (different meanings in JS)

**Comments in JavaScript:**

Ex:

<!--

// This is a comment. It is similar to comments in C++

/\*

\* This is a multiline comment in JavaScript

\* It is very similar to comments in C Programming

\*/

//-->

1. **PLACEMENT**

* <head>...</head> section

<head>

Ex:

<head>

<script type="text/javascript">

<!--

function sayHello() {

alert("Hello World")

}

//-->

</script>

</head>

* <body>...</body> section

Ex:

<body>

<script type="text/javascript">

<!--

function sayHello() {

alert("Hello World")

}

//-->

</script>

</body>

* <body>...</body> and <head>...</head> sections

Ex:

<head>

<script></script>

</head>

<body>

<script></script>

</body>

* external file and then include in <head>...</head> section

Ex:

<script type=”text/javascript” src=”external.js”></script>

1. **VARIABLES**

Ex:

var money;

**Three primitive data type: (single value)**

Numbers, e.g., 123, 120.50 etc

Strings of text, e.g. "This text string" etc

Boolean, e.g. true or false

Ex:

var money = 2000.04;

var name =”optimus”;

var flag=true;

**Two trivial data types**: (single value)

null

undefined

**composite data type (multi value)**

object

**JavaScript Variable Scope:**

**Global Variables :**

defined anywhere in your JavaScript code

**Local Variables:**

visible only within a function

function parameter always local variables

Ex:

**var myVar = "global"; // Declare a global variable**

function checkscope( ) {

**var myVar = "local"; // Declare a local variable**

document.write(myVar**); //local**

}

document.write(myVar**); //global**

**Keywords:**

Support all java keywords with some external**(59 keywords)**

Variable name declaration also same in java and java script.

1. **OPERATORS**

* Arithmetic Operators

A holds 10 and variable B holds 20,

A+B = 30

A-B = -10

A\*B = 60

B/A = 2

B%A = 0

A++ = 11

A-- = 9

* Comparison Operators( **return only boolean**)

A=10 B=20

**(A == B)** is not true

**(A != B)** is true

**(A > B)** is not true

**(A < B)** is true

**(A >= B)** is not true

**(A <= B)** is true

* Logical (or Relational) Operators

**Logical operator**

A=10 B=20

&& (Logical AND)

Ex:

(A && B) is true.

|| (Logical OR)

Ex:

(A || B) is true.

! (Logical NOT)

Ex:

! (A && B) is false

**Bitwise Operator**

A=2 B= 3

& (Bitwise AND)

Ex:

(A & B) is 2.

| (BitWise OR)

Ex:

(A | B) is 3.

^ (Bitwise XOR)

Ex:

(A ^ B) is1.

~ (Bitwise Not)

Ex:

(~B) is -4.

<< (Left Shift)

Ex:

(A << 1) is 4.

>> (Right Shift)

Ex:

(A >> 1) is 1.

>>> (Right shift with Zero)

Ex:

(A >>> 1) is 1.

* Assignment Operators

= (Simple Assignment )

Ex:

C = A + B will assign the value of A + B into C

+= (Add and Assignment)

Ex:

C += A is equivalent to C = C + A

-= (Subtract and Assignment)

Ex:

C -= A is equivalent to C = C - A

\*= (Multiply and Assignment)

Ex:

C \*= A is equivalent to C = C \* A

/= (Divide and Assignment)

Ex:

C /= A is equivalent to C = C / A

%= (Modules and Assignment)

Ex:

C %= A is equivalent to C = C % A

**Note:** Same logic applies to Bitwise operators, so they will become <<=, >>=, >>=, &=, |= and ^=.

* Conditional (or ternary) Operators

**conditional operator (? :)**

Ex:

(a > b) ? 100 : 200

**typeof operator**

**Datatype Return**

Number "number"

String "string"

Boolean "boolean"

Object "object"

Function "function"

Undefined "undefined"

Null "object"

**7. IF-ELSE**

**if Statement**

if (expression){

Statement(s) to be executed if expression is true

}

**if...else Statement**

if (expression){

Statement(s) to be executed if expression is true

}else{

Statement(s) to be executed if expression is false

}

**if...else if... Statement**

if (expression 1){

Statement(s) to be executed if expression 1 is true

}else if (expression 2){

Statement(s) to be executed if expression 2 is true

}else if (expression 3){

Statement(s) to be executed if expression 3 is true

}else{

Statement(s) to be executed if no expression is true

}

**8. SWITCH-CASE**

var grade='A';

switch (grade)

{

case 'A': document.write("Good job<br />");

case 'B': document.write("Pretty good<br />");

case 'C': document.write("Passed<br />");

case 'D': document.write("Not so good<br />");

case 'F': document.write("Failed<br />");

default: document.write("Unknown grade<br />")

}

**9. WHILE LOOP**

**The while Loop:**

var count = 0;

while (count < 10){

count++;

}

**The do...while Loop:**

var count = 0;

do{

count++;

}while (count < 5);

**10. FOR LOOP**

var count;

for(count = 0; count < 10; count++){

}

**11. FOR-IN LOOP**

for (variablename in object){

statement or block to execute

}

Ex:

var aProperty;

for (aProperty in navigator) {

}

**12. LOOP CONTROL**

**The break Statement :**

var x = 1;

while (x < 20)

{

f (x == 5){

**break;** // breaks out of loop completely

}

x = x + 1;

}

**The continue Statement:**

var x = 1;

while (x < 10)

{

x = x + 1;

if (x == 5){

continue; // skill rest of the loop body

}

}

**Using Labels to Control the Flow**

outerloop: // This is the label name

for (var i = 0; i < 5; i++)

{

document.write("Outerloop: " + i + "<br />");

innerloop:

for (var j = 0; j < 5; j++)

{

if (j > 3 ) break ; // Quit the innermost loop

if (i == 2) break innerloop; // Do the same thing

if (i == 4) break outerloop; // Quit the outer loop

document.write("Innerloop: " + j + " <br />");

}

}

**13. FUNCTIONS**

**Function Definition:**

**Syntax:**

function functionname(parameter-list)

{

statements

}

Ex:

function sayHello()

{

alert("Hello there");

}

**Calling a Function:**

**Script:**

<script type="text/javascript">

function **sayHello()**

{

document.write ("Hello there!");

}

</script>

**html:**

<form>

<input type="button" **onclick="sayHello()"** value="Say Hello">

</form>

**Function Parameters:**

**Script:**

function **sayHello(name, age)**

{

document.write (name + " is " + age + " years old.");

}

**html:**

<form>

<input type="button" **onclick="sayHello('Zara', 7)"** value="Say Hello">

</form>

**The return Statement:**

function concatenate(first, last)

{

var full;

full = first + last;

**return full;**

}

function secondFunction()

{

var result;

result = **concatenate('Zara', 'Ali');**

document.write (result );

}

**Nested Functions:**

function **hypotenuse(a, b)** {

function **square(x)** { return x\*x; }

return Math.sqrt(**square(a)** + **square(b)**);

}

function **secondFunction(){**

var result;

result = **hypotenuse(1,2)**;

document.write ( result );

}

**Html:**

<form>

<input type="button" **onclick="secondFunction()"** value="Call Function">

**</**form>

**Function () Constructor**

**new** operator

Note:

Constructor is a terminology from **Object Oriented Programming**

**Syntax** :

<script type="text/javascript">

<!--

**var variablename = new Function(Arg1, Arg2..., "Function Body");**

//-->

</script>

Ex:

**script**

**var func = new Function("x", "y", "return x\*y;");**

function secondFunction(){

var result;

result = **func(10,20);**

document.write ( result ); }

**html**

<form>

<input type="button" **onclick="secondFunction()"** value="Call Function">

</form>

**Function Literals:**

**Syntax 1:**

<script type="text/javascript">

<!--

var variablename = function(Argument List){

Function Body

};

//-->

</script>

Ex:

**var func = function(x,y){ return x\*y };**

function secondFunction(){

var result;

result = **func(10,20);**

document.write ( result );

}

**Syntax 2:**

<script type="text/javascript">

<!--

var variablename = function FunctionName(Argument List){

Function Body

};

//-->

</script>

**14. EVENTS**

**Event:**

user or the browser **manipulates a page**

Events are a part of the **Document Object Model** (DOM)

**onclick Event Type:**

<input type="button" onclick="sayHello()" value="Say Hello" />

**onsubmit Event Type:**

**script:**

function validation() {

all validation goes here

.........

**return either true or false**

}

**Html:**

<form method="POST" action="t.cgi" **onsubmit="return validate()"**>

.......

<input type="submit" value="Submit" />

</form>

**onmouseover and onmouseout:**

<div onmouseover="over()" onmouseout="out()">

**15. COOKIES**

**16. PAGE REDIRECT**

Process Redirect from one page to another page

**JavaScript Page Refresh:**

<a href="javascript:**location.reload(true)"**>Refresh Page</a>

**Auto Refresh:**

**setTimeout()** is a built-in JavaScript function which can be **used to execute another function after a given time interval**

**Syntax:**

function AutoRefresh( t ) {

**setTimeout("location.reload(true);", t)**;

}

<body onload="JavaScript:**AutoRefresh(5000)**;">

**How Page Re-direction Works?**

**Example 1:**

function Redirect() {

**window.location="http://www.tutorialspoint.com";**

}

<form>

<input type="button" value="Redirect Me" **onclick="Redirect();"** /> </form>

**Example 2:**

function Redirect() {

**window.location="http://www.tutorialspoint.com";**

}

document.write ("You will be redirected to our main page in 10 seconds!");

**setTimeout('Redirect()', 1000);**

**Example 3:**

var browsername=**navigator.appName**;

if( browsername == "Netscape" )

{

window.location="http://www.location.com/ns.htm";

}

else if ( browsername =="Microsoft Internet Explorer")

{

window.location="http://www.location.com/ie.htm";

}

else

{

window.location="http://www.location.com/other.htm"; }

**17. DIALOG BOX**

**Alert Dialog Box:**

function Warn() {

**alert ("This is a warning message!");**

document.write ("This is a warning message!");

}

<form>

<input type="button" value="Click Me" onclick="Warn();" />

</form>

**Confirmation Dialog Box:**

function getConfirmation(){

var retVal = **confirm("Do you want to continue ?");**

if( retVal == true ){

document.write ("User wants to continue!");

return true;

}else{

Document.write ("User does not want to continue!");

return false;

}

}

<form>

<input type="button" value="Click Me" onclick="getConfirmation();" />

</form**>**

**Prompt Dialog Box:**

function getValue(){

var retVal = **prompt("Enter your name : ", "your name here");**

document.write("You have entered : " + retVal);

}

<form>

<input type="button" value="Click Me" onclick="getValue();" />

</form>

**18. VOID KEYWORD**

* void is an **important keyword** in JavaScript
* **without** returning a value

**Syntax** :

<head>

<script type="text/javascript">

<!--

void func()

javascript:void func()

OR

void(func())

javascript:void(func())

//-->

</script>

</head>

**Example 1:**

<a href="javascript:**void(document.write(“Hello : 0”))**">Click me!</a>

**Example 2:**

<a href="javascript:**void(0)**">Click me!</a>

**Example 3:**

function getValue(){

var a,b,c;

a = **void ( b = 5, c = 7 );**

document.write('a = ' + a + ' b = ' + b +' c = ' + c );

}

<input type="button" value="Click Me" onclick="getValue();" />

**19. PAGE PRINTING**

* Print screen window will be opened

**Syntax:**

<input type="button" value="Print" onclick="**window.print()"** />

**Part 2: JavaScript Objects**

**20. OBJECTS**

* **Encapsulation:**

store related information, whether **data or methods**

* **Aggregation:**

to store one object inside another object

* **Inheritance:**

One class derived from another class

* **Polymorphism**:

to write **one function or method** that works in a variety **of different ways**

**Object Properties:**

**syntax :**

objectName.objectProperty = propertyValue;

**example:**

var str = document.title;

**Object Methods:**

document.write ("This is test");

**User-Defined Objects:**

All **user-defined objects** and **built-in objects** are **descendants** of an object called Object

**The new Operator:**

create an instance of an object

Ex:

var employee = **new Object()**;

var books = **new Array("C++", "Perl", "Java")**;

var day = **new Date("August 15, 1947")**;

**The Object ( ) Constructor:**

function that **creates and initializes** an object

**Ex 1:**

var book = new Object(); // Create the object

book.subject = "Perl"; // Assign properties to the object

book.author = "Mohtashim";

**Ex 2:**

function book(title, author){

this.title = title;

this.author = author;

}

var myBook = new book("Perl", "Mohtashim");

myBook .title //Perl

myBook .author //Mohtashim

**Defining Methods for an Object:**

function addPrice(amount){

this.price = amount;

}

function book(title, author){

this.title = title;

this.author = author;

**this.addPrice = addPrice;** // Assign that method as property.

}

var myBook = new book("Perl", "Mohtashim");

myBook.addPrice(100);

myBook.title //Perl

myBook.author //Mohtasium

myBook.price //100

**The ‘with’ Keyword :**

* + - shorthand for **referencing an object's** properties or methods
    - **default object** for the duration of the block
    - methods for the object can be used **without naming** the object.

Ex:

function addPrice(amount){

**with(this){**

**price = amount;**

**}**

}

function book(title, author){

this.title = title;

this.author = author;

this.price = 0;

this.addPrice = addPrice; // Assign that method as property.

}

var myBook = new book("Perl", "Mohtashim");

myBook.addPrice(100);

myBook.title//

myBook.author//

myBook.price //100

**21. NUMBER**

* represents numerical **date**, either **integers** or **floating-point** numbers

**Syntax:**

var val = **new Number(number)**;

**Note :** non-number argument, then the argument **cannot be converted into a number**, it returns **NaN (Not-a-Number)**

**Property:**

|  |  |
| --- | --- |
| **MAX\_VALUE** | 1.7976931348623157 x 10^308 |
| var val = Number.MAX\_VALUE; | |

|  |  |
| --- | --- |
| **MIN\_VALUE** | 5 x 10^-324 |
| var val = Number.MIN\_VALUE; | |

|  |  |
| --- | --- |
| **NaN** | NaN |
| var val = Number.NaN; | |
| **NEGATIVE\_INFINITY** | -Infinity |
| var val = Number. NEGATIVE\_INFINITY; | |

|  |  |
| --- | --- |
| **POSITIVE\_INFINITY** | Infinity |
| var val = Number. POSITIVE\_INFINITY; | |

**Prototype:**

* add **properties and methods** to any object (Number, Boolean, String and Date etc.)

**Note:** Prototype is a global property which is available with almost all the objects.

**Syntax**: object.prototype.name = value;

**Ex:**

var myBook = new book("Perl", "Mohtashim");

**book.prototype.price = null;**

myBook.price = 100;

myBook.title //Perl

myBook.author //Mohtasium

myBook.price //100

**constructor:**

**Syntax:** number.constructor()

**Ex:**

var num = new Number( 177.1234 );

**Number Methods :**

|  |  |
| --- | --- |
| **Method** | |
| **toExponential ()** | Like (7.71234e+1) |
| var num=77.1234;  a) var val = num.toExponential();  b) val = num.toExponential(4);  c) val = 77.1234.toExponential(); | |

|  |  |
| --- | --- |
| **Method** | |
| **toFixed ()** | Like (7.71234e+1) |
| var num=77.1234;  a) num.toFixed() //77  b) num.toFixed(6) //77.123400  c) num.toFixed(1) //77.1 | |

|  |  |
| --- | --- |
| **Method** | |
| **toLocaleString ()** | 177.123 |
| var num = new Number(177.1234);  document.write( num.toLocaleString()); | |

|  |
| --- |
| **Method** |
| **toPrecision ()** |
| var num = new Number(7.123456);  num.toPrecision() //7.123456  num.toPrecision(4) // 7.123  num.toPrecision(2) // 7.1  num.toPrecision(1) // 7 |

|  |
| --- |
| Method |
| toString () |
| var num = new Number(15);  num.toString() is 15  num.toString(2) is 1111  num.toString(4) is 33 |

|  |
| --- |
| **Method** |
| **valueOf ()** |
| var num = new Number(15.11234);  num.valueOf() is 15.11234 |

**22. BOOLEAN**

**TRUE -** true

**FALSE -** 0, -0, null, false, NaN, undefined, or the empty string ("")

**Syntax:**

var val = new Boolean(value);

**Note :** prototype and constructor same in Boolean and number

**Boolean Methods:**

|  |
| --- |
| Method |
| toSource () |
| function book(title, publisher, price)  {  this.title = title;  this.publisher = publisher;  this.price = price;  }  var newBook = new book("Perl","Leo Inc",200);  document.write("newBook.toSource() is : "+ newBook.toSource()); |
| ({title:"Perl", publisher:"Leo Inc", price:200}) |

|  |
| --- |
| **Method** |
| **toString ()** |
| var flag = new Boolean(false);  document.write( "flag.toString is : " + flag.toString() ); |
| **flag.toString is : false** |

|  |
| --- |
| **Method** |
| **valueOf ()** |
| var flag = new Boolean(false);  document.write( "flag.valueOf is : " + flag.valueOf() ); |
| **flag.toString is : false** |

**23. STRING**

**Syntax:**

var str = new String( "This is string" );

**String Properties:**

Constructor **-** str.constructor

Length - str.length

Prototype - book.prototype.price = null;

**String Methods:**

|  |
| --- |
| **Method** |
| **charAt()** |
| var str = new String( "This is string" );  document.writeln("str.charAt(0) is:" + str.charAt(0));  document.writeln("<br />str.charAt(1) is:" + str.charAt(1)); |
| **str.charAt(0) is:T**  **str.charAt(1) is:h** |

|  |
| --- |
| Method |
| charCodeAt () |
| var str = new String( "This is string" );  document.write("str.charCodeAt(0) is:" + str.charCodeAt(0));  document.write("<br />str.charCodeAt(1) is:" + str.charCodeAt(1)); |
| str.charCodeAt(0) is:84  str.charCodeAt(1) is:104 |

|  |
| --- |
| **Method** |
| **contact ()** |
| var str1 = new String( "This is string one" );  var str2 = new String( "This is string two" );  var str3 = str1.concat( str2 ); |
| **This is string one This is string two** |

|  |
| --- |
| **Method** |
| **indexOf ()** |
| var str1 = new String( "This is string one" );  var index = str1.indexOf( "string" );  document.write("indexOf found String :" + index ); |
| **indexOf found String :8** |

|  |
| --- |
| **Method** |
| **lastIndexOf ()** |
| var str1 = new String( "This is string one and again string" );  var index = str1.lastIndexOf( "string" );  document.write("lastIndexOf found String :" + index ); |
| **lastIndexOf found String :29** |

**localeCompare ()**

|  |
| --- |
| Method |
| match () |
| var str = "For more information, see Chapter 3.4.5.1";  var re = /(chapter \d+(\.\d)\*)/i;  var found = str.match( re );  document.write(found ); |
| Chapter 3.4.5.1,Chapter 3.4.5.1,.1 |

|  |
| --- |
| **Method** |
| **replace ()** |
| var re = /apples/gi;  var str = "Apples are round, and apples are juicy.";  var newstr = str.replace(re, "oranges");  document.write(newstr ); |
| oranges are round, and oranges are juicy. |

|  |
| --- |
| **Method** |
| **Search ()** |
| var re = /apples/gi;  var str = "Apples are round, and apples are juicy.";  if ( str.search(re) == -1 ){  document.write("Does not contain Apples" );  }else{  document.write("Contains Apples" ); } |
| Contains Apples |

|  |
| --- |
| Method |
| slice () |
| var str = "Apples are round, and apples are juicy.";  var sliced = str.slice(3, -2);  document.write( sliced ); |
| les are round, and apples are juic |

|  |
| --- |
| **Method** |
| **split ()** |
| var str = "Apples are round, and apples are juicy.";  var splitted = str.split(" ", 3);  document.write( splitted ); |
| Apples,are,round, |

|  |
| --- |
| **Method** |
| **substr ()** |
| var str = "Apples are round, and apples are juicy.";  document.write("(1,2): " + str.substr(1,2));  document.write("<br />(-2,2): " + str.substr(-2,2)); |
| (1,2): pp  (-2,2): y. |

|  |
| --- |
| **Method** |
| **substring ()** |
| var str = "Apples are round, and apples are juicy.";  document.write("(1,2): " + str.substring(1,2));  document.write("<br />(0,10): " + str.substring(0, 10));  document.write("<br />(5): " + str.substring(5)); |
| (1,2): p  (0,10): Apples are  (5): s are round, and apples are juicy. |

|  |
| --- |
| **Method** |
| **toLocaleLowerCase()** |
| var str = "Apples are round, and Apples are Juicy.";  document.write(str.toLocaleLowerCase( )); |
| apples are round, and apples are juicy |

|  |
| --- |
| **Method** |
| **toLocaleUppereCase ()** |
| var str = "Apples are round, and Apples are Juicy.";  document.write(str.toLocaleUpperCase( )); |
| APPLES ARE ROUND, AND APPLES ARE JUICY |

|  |
| --- |
| **Method** |
| **toLowerCase ()** |
| var str = "Apples are round, and Apples are Juicy.";  document.write(str.toLowerCase( )); |
| apples are round, and apples are juicy. |

|  |
| --- |
| **Method** |
| **toString ()** |
| var str = "Apples are round, and Apples are Juicy.";  document.write(str.toString( )); |
| Apples are round, and Apples are Juicy. |

|  |
| --- |
| **Method** |
| **toUpperCase ()** |
| var str = "Apples are round, and Apples are Juicy.";  document.write(str.toUpperCase( )); |
| APPLES ARE ROUND, AND APPLES ARE JUICY. |

|  |
| --- |
| **Method** |
| **valueOf ()** |
| var str = new String("Hello world");  document.write(str.valueOf( )); |
| Hello world |

**String HTML Wrappers:**

|  |
| --- |
| **Method** |
| **anchor()** |
| var str = new String("Hello world");  alert (str.anchor( "myanchor" )); |
| <a name="myanchor">Hello world</a> |

**Method Description**

**anchor()** Creates an HTML anchor that is used as a hypertext target.

**big()** Creates a string to be displayed in a big font as if it were in a <big> tag.

**blink()** Creates a string to blink as if it were in a <blink> tag.

**bold()** Creates a string to be displayed as bold as if it were in a <b> tag.

**fixed()** Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag

**fontcolor()** Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.

**fontsize()** Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.

**italics()** Causes a string to be italic, as if it were in an <i> tag.

**link()** Creates an HTML hypertext link that requests another URL.

**small()** Causes a string to be displayed in a small font, as if it were in a <small> tag.

**strike()** Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.

**sub()** Causes a string to be displayed as a subscript, as if it were in a <sub> tag

**sup()** Causes a string to be displayed as a superscript, as if it were in a <sup> ta

**24. ARRAYS**

* Array object lets you **store multiple values in a single variable**

**Syntax:**

var fruits = **new Array**( "apple", "orange", "mango" );

var fruits = [ "apple", "orange", "mango" ];

**Properties:**

var arr = new Array( 10, 20, 30 );

constructor - arr.constructor

index - arr[0]

input -

length - arr.length

prototype - book.prototype.price = null;

**Array Methods:**

|  |
| --- |
| **Method** |
| **concat ()** |
| var alpha = ["a", "b", "c"];  var numeric = [1, 2, 3];  var alphaNumeric = alpha.concat(numeric);  document.write("alphaNumeric : " + alphaNumeric ); |
| alphaNumeric : a,b,c,1,2,3 |

**every () lastindexof()**

**filter () reduce()**

**indexof() reduceright()**

**forEach() same()**

**map()**

|  |
| --- |
| **Method** |
| **Join ()** |
| var arr = new Array("First","Second","Third");  var str = arr.join(" + ");  document.write("<br />str : " + str ); |
| str : First + Second + Third |

|  |
| --- |
| **Method** |
| **pop ()** |
| var numbers = [1, 4, 9];  var element = numbers.pop();  document.write("element is : " + element ); |
| element is : 9 |

|  |
| --- |
| **Method** |
| **push ()** |
| var numbers = new Array(1, 4, 9);  var length = numbers.push(10);  document.write("new numbers is : " + numbers ); |
| new numbers is : 1,4,9,10 |

|  |
| --- |
| **Method** |
| **reverse ()** |
| var arr = [0, 1, 2, 3].reverse();  document.write("Reversed array is : " + arr ); |
| Reversed array is : 3,2,1,0 |

|  |
| --- |
| **Method** |
| **shift ()** |
| var element = [105, 1, 2, 3].shift();  document.write("Removed element is : " + element ); |
| Removed element is : 105(remove first element) |

|  |
| --- |
| **Method** |
| **slice ()** |
| var arr = ["orange", "mango", "banana", "sugar", "tea"];  document.write("arr.slice( 1, 2) : " + arr.slice( 1, 2) );  document.write("<br />arr.slice( 1, 2) : " + arr.slice( 1, 3) ); |
| arr.slice( 1, 2) : mango  arr.slice( 1, 2) : mango,banana |

|  |
| --- |
| **Method** |
| **sort ()** |
| var arr = new Array("orange", "mango", "banana", "sugar");  var sorted = arr.sort();  document.write("Returned string is : " + sorted ); |
| Returned string is : banana,mango,orange,sugar |

|  |
| --- |
| **Method** |
| **splice ()** |
| var arr = new Array("orange", "mango", "banana", "sugar");  var sorted = arr.sort();  document.write("Returned string is : " + sorted ); |
| After adding 1: orange,mango,water,banana,sugar,tea removed is: After adding 1: orange,mango,water,sugar,tea removed is: banana |

|  |
| --- |
| **Method** |
| **sort ()** |
| var arr = new Array("orange", "mango", "banana", "sugar");  var sorted = arr.sort();  document.write("Returned string is : " + sorted ); |
| Returned string is : banana,mango,orange,sugar |

|  |
| --- |
| **Method** |
| **toString ()** |
| var arr = new Array("orange", "mango", "banana", "sugar");  var str = arr.toString();  document.write("Returned string is : " + str ); |
| Returned string is : orange,mango,banana,sugar |

|  |
| --- |
| **Method** |
| **unshift ()** |
| var arr = new Array("orange", "mango", "banana", "sugar");  var length = arr.unshift("water");  document.write("Returned array is : " + arr ); |
| Returned array is : water,orange,mango,banana,sugar |

**25. DATE**

**Syntax** :

new Date( )

new Date(milliseconds)

new Date(datestring)

new Date(**year,month,date[,hour,minute,second,millisecond** ])(7 args)

**Date Methods** :

**Method**  **Description**

**Date()** Returns **today's date and time**

**Syntax:**

var dt = Date();

document.write("Date and Time : " + dt );

**output:** Date and Time : Wed Mar 25 2015 15:00:57 GMT+0530 (India Standard Time)

**getDate()** (1 to 31) Return **day of the month**

**Syntax:**

var dt = new Date("December 25, 1995 23:15:00");

document.write("getDate() : " + dt.getDate() );

**output:** getDate() : 25

**getDay() (0 to 6)**  Returns **the day of the week**

**Syntax:**

var dt = new Date("December 25, 1995 23:15:00");

document.write("getDay() : " + dt.getDay() );

**output:** getDay() : 1

**getFullYear()** (1000 to 9999) Returns the **year of the specified**

**getHours() (0 to 23)`**  Returns the **hour in the specified date**

**getMilliseconds()** (0 to 999)Returns the **milliseconds in the specified date**

**getMinutes()** (0 to 59) Returns the **minutes in the specified date**

**getMonth()** (0 to 11) Returns the **month in the specified date**

**getSeconds()** (0 to 59) Returns the **seconds in the specified date**

**getTime()** (no of milliseconds)Returns the **numeric value of the specified date**

**getTimezoneOffset()** Return **time-zone** offset in minutes.

**getUTCDate() (1- 31)** Returns the **day of the month** in the specified date

**getUTCDay() (0-6)**  Returns the **day of the week** in the specified date

**getUTCFullYear()** Returns the **year in the specified date**

**getUTCHours()** Returns the **hours in the specified date**

**getUTCMilliseconds()**  Returns the **milliseconds in the specified date**

**getUTCMinutes()** Returns the **minutes in the specified date**

**getUTCMonth()** Returns the **month in the specified date**

**getUTCSeconds()** Returns the **seconds in the specified date**

**getYear()** Deprecated **- Returns the year in the specified date**

**setDate()** Sets the **day of the month** for a specified

**Syntax:**

var dt = new Date( "Aug 28, 2008 23:30:00" );

dt.setDate( 24 );

document.write( dt );

**setFullYear()** Sets the **full year for a specified**

**dt.setFullYear( 2000 );**

**setHours()** Sets the **hours for a specified date**

**dt.setHours( 02 );**

**setMilliseconds()** Sets the **milliseconds for a specified date**

**setMinutes()** Sets the **minutes for a specified date**

**setMonth()** Sets the **month for a specified date**

**setSeconds()** Sets the **seconds for a specified date** .

**setTime()** Sets the **Date object to the time**

**setUTCDate()** Sets the **day of the month for a specified date**

**setUTCFullYear()** Sets the **full year for a specified date**

**setUTCHours()** Sets the **hour for a specified date**

**setUTCMilliseconds()** Sets the **milliseconds for a specified**

**setUTCMinutes()** Sets the **minutes for a specified date**

**setUTCMonth()** Sets the **month for a specified date**

**setYear()** Deprecated - **Sets the year for a specified date**

**toDateString()** Returns the "**date" portion of the Date** as a **humanreadable string**.

**Syntax:**

var dt = new Date(1993, 6, 28, 14, 39, 7);

document.write( "Formated Date : " + dt.toDateString() );

**Output :**

Formated Date : Wed Jul 28 1993

**toGMTString()** Deprecated - **Converts a date to a string**

**Output :**

Formated Date : Wed, 28 Jul 1993 09:09:07 GMT

**toLocaleDateString()**  Returns the "date" **portion of the Date**

**Output :**

Formated Date : 7/28/1993

**toLocaleFormat()** Converts a **date to a string**

**Output :**

Formated Date : Wed Jul 28 1993 14:39:07 GMT+0530 (India Standard Time)

**toLocaleString**() Converts a **date to a string**

**Output:**

Formated Date : 7/28/1993, 2:39:07 PM

**toLocaleTimeString()** Returns the "**time" portion of the Date**

**Output:**

Formated Date : 2:39:07 PM

**toSource()** Returns a **string representing the source for an equivalent Date**

**Output:**

Formated Date : (new Date(743850547000))

**toString()** Returns a string **representing the specified Date object**.

**Output:**

String Object : Wed Jul 28 1993 14:39:07 GMT+0530 (India Standard Time)

**toTimeString()**  Returns the "**time" portion of the Date** 14:39:07

**Output:**

GMT+0530 (India Standard Time)

**toUTCString()** Converts a **date to a string**

**Output:**

Wed, 28 Jul 1993 09:09:07 GMT

**valueOf()**  Returns the **primitive value of a Date object**.

**Output:**

743850547000

**26. MATH**

**Math Properties** :

**Property Description**

Math.E Euler's constant //2.718.

Math.LN2 Natural logarithm of 2// 0.693.

Math.LN10 Natural logarithm of 10// 2.302.

Math.LOG2E Base 2 logarithm of E// 1.442

Math.LOG10E Base 10 logarithm of E// 0.434.

Math.PI approximately 3.14159.

Math.SQRT1\_2 Square root of 1/2; 0.707

Math.SQRT2 Square root of 2// 1.414

**Methods:**

|  |
| --- |
| **Method** |
| **ceil ( )** |
| var value = Math.ceil(45.95); // 46  var value = Math.ceil(45.20); // 46  var value = Math.ceil(-45.95); // -45  var value = Math.ceil(-45.20); // -45 |

|  |
| --- |
| **Method** |
| **floor ( )** |
| var value = Math.floor(10.3); // 10  var value = Math.floor(30.9); // 30  var value = Math.floor(-2.9); // -3  var value = Math.floor(-2.2); // -3 |

|  |
| --- |
| **Method** |
| **min ( )** |
| var value = Math.min(10, 20, -1, 100); // -1 |

|  |
| --- |
| **Method** |
| **max ( )** |
| var value = Math.min(10, 20, -1, 100); // 100 |

**27. REGEXP**

**Brackets:**

**[...] Any one character between the brackets**

**[^...] Any one character not between the brackets**

**[0-9] It matches any decimal digit from 0 through 9.**

**[a-z] any character from lowercase a through lowercase z**

**[A-Z] any character from uppercaseA through uppercase Z**

**[a-Z] any character from lowercase a through uppercase Z.**

**Quantifiers:**

**p+ It matches any string containing at least one p**

**p\* It matches any string containing zero or more p's**

**p? It matches any string containing one or more p's**

**p{N} It matches any string containing a sequence of N p's**

**p{2,3} It matches any string containing a sequence of two or three p's.**

**p{2, } It matches any string containing a sequence of at least two p's.**

**p$ It matches any string with p at the end of it**

**^p It matches any string with p at the beginning of it**

**[^a-zA-Z] any of the characters ranging from a through z and A through Z**

**p.p It matches any string containing p, followed by any character, in turn followed by another p.**

**^.{2}$ It matches any string containing exactly two characters.**

**<b>(.\*)</b> It matches any string enclosed within <b> and </b>.**

**28. DOM**

**Window object**:

Top of the hierarchy. It is the **outmost element of the object** hierarchy.

**Document object**:

Each HTML document **that gets loaded into a window** becomes a document object

**Form object**:

Everything enclosed in the <form>...</form> tags sets the form object.

**Form control elements**:

The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes

**Document Properties in Legacy DOM:**

**specifies the color of activated links.**

**Ex: document.alinkColor**

**2**

**anchors[ ] An array of Anchor objects, one for each anchor that appears in the document**

**Javascript**

**327**

**Ex: document.anchors[0], document.anchors[1] and so on**

**3**

**applets[ ] An array of Applet objects, one for each applet that appears in the document**

**Ex: document.applets[0], document.applets[1] and so on**

**4**

**bgColor Deprecated - A string that specifies the background color of the document.**

**Ex: document.bgColor**

**5**

**Cookie A string valued property with special behavior that allows the cookies associated with this document to be queried and set.**

**Ex: document.cookie**

**6**

**Domain A string that specifies the Internet domain the document is from. Used for security purpose.**

**Ex: document.domain**

**7**

**embeds[ ] An array of objects that represent data embedded in the document with the <embed> tag. A synonym for plugins []. Some plugins and ActiveX controls can be controlled with JavaScript code.**

**Ex: document.embeds[0], document.embeds[1] and so on**

**8**

**fgColor A string that specifies the default text color for the document**

**Ex: document.fgColor**

**Javascript**

**328**

**9**

**forms[ ] An array of Form objects, one for each HTML form that appears in the document.**

**Ex: document.forms[0], document.forms[1] and so on**

**10**

**images[ ] An array of Image objects, one for each image that is embedded in the document with the HTML <img> tag.**

**Ex: document.images[0], document.images[1] and so on**

**11**

**lastModified A read-only string that specifies the date of the most recent change to the document**

**Ex: document.lastModified**

**12**

**linkColor Deprecated - A string that specifies the color of unvisited links**

**Ex: document.linkColor**

**13**

**links[ ] It is a document link array.**

**Ex: document.links[0], document.links[1] and so on**

**14**

**Location The URL of the document. Deprecated in favor of the URL property.**

**Ex: document.location**

**15**

**plugins[ ] A synonym for the embeds[ ]**

**Javascript**

**329**

**Ex: document.plugins[0], document.plugins[1] and so on**

**16**

**Referrer A read-only string that contains the URL of the document, if any, from which the current document was linked.**

**Ex: document.referrer**

**17**

**Title The text contents of the <title> tag.**

**Ex: document.title**

**18**

**URL A read-only string that specifies the URL of the document.**

**Ex: document.URL**

**19**

**vlinkColor Deprecated - A string that specifies the color of visited links.**

**Ex: document.vlinkColor**