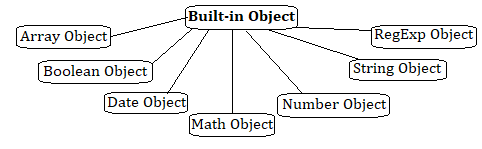
**JavaScript Built in Objects**

JavaScript has several built-in or core language objects. These built-in objects are available regardless of window content and operate independently of whatever page your browser has loaded.



**Array Object**

* Multiple values are stored in a single variable using the Array object.
* In JavaScript, an array can hold different types of data types in a single slot, which implies that an array can have a string, a number or an object in a single slot.
* An Array object can be created by using following ways:

***Using the Array Constructor:***

To create empty array when don’t know the exact number of elements to be inserted in an array

var arrayname = new Array();

To create an array of given size

var arrayname = new Array(size);

To create an array with given elements

var arrayname = new Array(“element 1”,”element 2”,……..,”element n”);

***Using the Array Literal Notation:***

To create empty array

var arrayname =[ ];

To create an array when elements are given

var arrayname =[“element 1”,”element 2”,……..,”element n”];

* Properties of the Array object
  + **Length** - Returns the number of elements in the array.
  + **Constructor** - Returns the function that created the Array object.
  + **Prototype** - Add properties and methods to an object.
* Methods of the Array object
  + **reverse()** - Reverses the array elements
  + **concat()** - Joins two or more arrays
  + **sort()** - Sort the elements of an array
  + **push()** - Appends one or more elements at the end of an array
  + **pop()** - Removes and returns the last element
  + **shift()** - Removes and returns the first element
* unshift(), join(), indexOf(), lastIndexOf(), slice(startindex, endindex) are some of the methods used in Array object.

**Boolean Object**

* The Boolean object is wrapper class and member of global objects.
* It is used to convert a non-Boolean value to a Boolean value (true or false).
* If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!
* Boolean object can be created by using following ways:

***Using Boolean Literal Notation:*** var bool = true;

***Using Boolean Object as Function:*** var bool = Boolean(true);

***Using Testable Expressions:*** if(true) {………….}

* Properties of the Boolean object
  + **Constructor** - Returns the function that created the Boolean object.
  + **Prototype** - Add properties and methods to an object.
* Methods of the Boolean object
  + **toSource()** - Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
  + **toString()** - Returns a string of either "true" or "false" depending upon the value of the object.
  + **valueOf()** - Returns the primitive value of the Boolean object

**Date Object**

* At times when user need to access the current date and time and also past and future date and times. JavaScript provides support for working with dates and time through the Date object.
* The Date object provides a system-independent abstraction of dates and times.
* Date object can be created as : var today = new Date( );
* Dates may be constructed from a year, month, day of the month, hour, minute, and second, and those six components, as well as the day of the week, may be extracted from a date.
* Dates may also be compared and converted to a readable string form. A Date is represented to a precision of one millisecond.
* Properties of Date object
  + **Constructor** - Returns the function that created the Date object.
  + **Prototype** - Add properties and methods to an object.
* Methods of Date object
  + **Date()** - Returns today's date and time
  + **getDate()** - Returns the day of the month for the specified date
  + **getDay()** - Returns the day of the week for the specified date
  + **getFullYear()** - Returns the year of the specified date
  + **getHours()** - Returns the hour in the specified date according to local time.
  + **getMilliseconds()** - Returns the milliseconds in the specified date according to local time.
* getMinute(), getMonth(), getTime(), getTimezoneOffset(), setDate(), setFullYear(), setHours(), setMilliseconds(), setMinutes(), setMonth(), setSeconds(), setTime() are some of the methods used in Date object.

**Math Object**

* The Math object is used to perform simple and complex arithmetic operations.
* The Math object provides a number of properties and methods to work with Number values
* The Math object does not have any constructors. All of its methods and properties are static; that is, they are member functions of the Math object itself. There is no way to create an instance of the Math object.
* Properties of Math object
  + **PI** - The value of Pi
  + **E** - The base of natural logarithm e
  + **LN2** - Natural logarithm of 2
  + **LN10** - Natural logarithm of 10
  + **LOG2E** - Base 2 logarithm of e
  + **LOG10E** - Base 10 logarithm of e
  + **SQRT2** - Square root of 2
  + **SQRT1\_2** - Square root of ½
* Methods of Math object
  + **max(a,b)** - Returns largest of a and b
  + **min(a,b)** - Returns least of a and b
  + **round(a)** - Returns nearest integer
  + **ceil(a)** - Rounds up. Returns the smallest integer greater than or equal to a
  + **floor(a)** - Rounds down. Returns the largest integer smaller than or equal to a
  + **exp(a)** - Returns ea
  + **pow(a,b)** - Returns ab
  + **abs(a)** - Returns absolute value of a
  + **random()** - Returns a pseudo random number between 0 and 1
  + **sqrt(a)** - Returns square root of a
  + **sin(a)** - Returns sin of a (a is in radians)
  + **cos(a)** - Returns cos of a (a is in radians)

**Number Object**

* The Number objects represents numerical date, either integers or floating-point numbers.
* A Number objects are created using the Number() constructor var num = new number(value);
* Properties of Number object
  + **Constructor** - Returns the function that created the Number object.
  + **MAX VALUE** - Returns maximum numerical value possible in JavaScript.
  + **MIN VALUE** - Returns minimum numerical value possible in JavaScript.
  + **NEGATIVE INFINITY** - Represent the value of negative infinity.
  + **POSITIVE INFINITY** - Represent the value of infinity.
  + **Prototype** - Add properties and methods to an object.
* Methods of Number object
  + **toExponential()** - Converts a number into exponential notation.
  + **toFixed()** - Formats a number with a specific number of digits to the right of the decimal.
  + **toLocaleString()** - Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
  + **toPrecision()** - Defines how many total digits to display of a number.
  + **toString()** - Returns the string representation of the number's value.
  + **valueOf()** - Returns the number's value.

**String Object**

* in JavaScript, all strings are represented as instances of the String object.
* The String object is wrapper class and member of global objects.
* String object used to perform operations on the stored text, such as finding the length of the string, searching for occurrence of certain characters within string, extracting a substring etc.
* A String is created by using literals. A string literal is either enclosed within single quotes(‘ ‘) or double quotes(“ “) var string1= “ Ques10“ ;
* Properties of String object
  + **Length** - Returns the length of string.
  + **Constructor** - Returns the function that created the String object
  + **Prototype** - Add properties and methods to an object
* Methods of String object
  + **charAt()** - Returns the character at a specified position.
  + **concat()** - Combines two or more strings.
  + **toLowerString()** - Converts a string to lowercase.
  + **toUpperString()** - Converts a string to uppercase.
  + **indexOf(searchtext, index)** - Searches for the specified string from the beginning of the string.
  + **lastIndexof(searchtext, index)** - Searches for the specified string from the end of the string

**RegExp Object**

* RegExp object used to validate the pattern of characters.
* RegExp define methods that use regular expressions to perform powerful pattern-matching and search and replace functions on text.
* Regular expressions can be defined by using following ways

***Using RegExp() Constructor:*** var RegularExpression = new RegExp(“pattern”,”flag”);

***Using Literal:*** var RegularExpression = /pattern/flag;

* ***Pattern*** – A String that specifies the pattern of the regular expression or another regular expression.
* ***Flag*** – An optional string containing any of the “g”, “I” and “m” attributes that specify global, case-insensitive and multiple matches respectively.
* Properties of RegExp object
  + **Constructor** - Returns the function that created the RegExp object
  + **Global** - Checks whether the "g" modifier is set
  + **ignoreCase** - Checks whether the "i" modifier is set
  + **lastIndex** - Specifies the index at which to start the next match
  + **multiline** - Checks whether the "m" modifier is set
  + **source** - Returns the text of the RegExp pattern
* Methods of RegExp object
  + **exec()** - Tests for a match in a string. Returns the first match.
  + **test()** - Tests for a match in a string. Returns true or false.
  + **toString()** - Returns the string value of the regular expression.
  + **toSource()** - Returns an object literal representing the specified object.