Unit-II

Array, Function and String

14 Marks [2R, 4U, 8A]

(2.1) Array –

Array is a collection of elements of similar data types. An array has one name and can hold as many required values. An array is identified by unique name & a number called an index identifies an array element.

⋆ Declaring an Array –

JavaScript arrays are used to store multiple values in a single variable as like arrays in C programming. The declaration of an array is same as variable. This declaration has five parts :

a) the first part is var syntax
b) the second part is the array name
c) the third part is the assignment operator
d) the fourth part is the new operator &
e) the fifth part is the Array() constructor.

```
var Products = new Array();
```

Eg. ① var x = new Array();
② var z = new Array(5);
③ var h = new Array(21, 3, "xyz");

⋆ Initializing an Array –

Initialization is the process of assigning a value to a variable or an array. We can assign the values that must be specified in parenthesis and elements should be seperated by comma.

```
var products = new Array ('Apple')
var fruits = new Array ('Apple', 'Banana',
                        'Orange')
```

The length property is used to determine the size of an array.

// Program to demonstrate array

```
<html>
<head>
  <title> Array Demo </title>
</head>
<body>
  <script>
    var fruits = new Array('Apple','Banana',
                           'Orange','Mango');
    document.write("Length of array:"+
                   fruits.length);
  </script>
</body>
</html>
```

Output –
Length of array: 4

⊛ Defining an array elements —

An array is a list of same kind of things like list of students, list of products name, list of customer names, etc. Each item (element) on the list is identified by the number called as index. The first item is number 0, second item is 1, then 2, then 3 and so on.

After declaring an array we can define its elements by using index in square brackets as follows:

```
var items = new Array(4);
items[0] = "Pen";          // items[0] holds "Pen"
items[1] = "Pencile";      // items[1] holds "Pencile"
items[2] = "Marker";       // items[2] holds "Marker"
items[3] = "Eraser";       // items[3] holds "Eraser"
```

Eg.

```
<script>
    var cars = new Array(4);
    cars[0] = "Bmw";
    cars[1] = "ford";
    cars[2] = "Toyota";
    cars[3] = "Volkswagen";
    for(var i=0; i< cars.length; i++)
    {
        document.write(cars[i] + "<br>");
    }
</script>
```

Output

```
Bmw
Ford
Toyota
Volkswagen
```

⊛ Looping an Array –

We can display all elements of an array by looping through each element of an array. For that we have to know array length by using array_name.length.

```
var programming_languages=new Array();
programming_languages[0]="C";
programming_languages[1]="C++";
programming_languages[2]="Java";
programming_languages[3]="JSP";

for(i=0;i<programming_languages.length;i++)
{
document.write(programming_languages[i]+"<br>");
}
```

Two-dimensional arrays –

```
var matrix=new Array(3);
for(i=0;i<3;i++)
{
  matrix[i]=new Array(3);
}
matrix[0][0]=1; | matrix[0][1]=2; | matrix[0][2]=3;
matrix[1][0]=4; | matrix[1][1]=5; | matrix[1][2]=6;
matrix[2][0]=7; | matrix[2][1]=8; | matrix[2][2]=9;
```

⭐ Adding an Array element –

In some situation JavaScript program we will need to increase the size of the array. This can be done using two ways:

① Using push method of array

```
arrObject.push(elem1, elem2, ..., elemn)
```

② Using length property of array

```
arrObject[arrObject.length] = element;
```

Eg –

```
<script>
    var a = new Array(3);
    a[0] = "One";
    a[1] = "Two";
    a[2] = "Three";
    a.push("Four");
    a.push("Five", "six");
    a[a.length] = "seven";
    a[a.length] = "Eight";
</script>
```

To add items at beginning of an array use unshift() method.

⑧ Sorting Array Elements –

Sometimes we want values to appear in sorted order, which means strings will be presented alphabetically and numbers will be displayed in ascending order.

The sort() method reorders values assigned to elements of an array. We have to call this method using array object.

1) Declare array
2) Assign values to elements of an array.
3) Call the sort() method

Eg –

```
var colors = ["Red", "Green", "Pink",
              "Black", "Cyan", "Blue"];
document.write("Before sorting:" + colors);
document.write("After Sorting:" + colors.sort());
```

By default, the sort() method arrange the elements in ascending order. If you want to sort element in descending order then you have to call reverse() method.

```
colors.sort();      → Asceding order
colors.reverse();   → Descending order
```

⭐ Combining an Array elements into a String –

Sometimes we want to combine values of an array element into one string

For ex– var a = new Array(3);
a[0] = "PHP".
a[1] = "Asp".
a[2] = "JSP";

By combining the above array elements to display the string.

"PHP, ASP, JSP"

Array elements can be combined in two ways –

① Using the concat() method –

This method ~~spe~~ seperates each value with a comma. We can combine two or more array. The existing array does not change but return a new array.

Syntax– arrObj1.concat() OR
arrObj1.concat(arrObj2, arrObj3, arrObj4)

② Using the join() method –

This method adds all elements of an array ~~se~~ seperated by the specified seperator string.

Syntax–
arrObj.join([seperator])

Here, seperator is optional, if not give by detault comma is used by JavaScript.

Example -

```
<html>
<head>
    <title> Demonstrate concat() and join() </title>
</head>
<body>
<script>
    var fruits = ['Apple', 'Orange', 'Banana']
    var s1 = fruits.join ('→');
    var s2 = fruits.concate();

    document.write ("Using concat () " + s2);
    document.write ("Using join () " + s1);
</script>
</body>
</html>
```

Output

Using concat()	Apple, Orange, Banana
using join ()	Apple → Orange → Banana

Slice() method —

The slice method of Array object return a section of an array.

Syntax - arrObj.slice (start, [end])
↳ optional

The slice() method copies a sequential no. of array elements from one array into new array.

Note - 1) If start is negative it is treated as length+start
2) If end is negative, treated as length+end
3) If end is ommitted, extraction continues to end.
4) If end occurs before start, no elements are copies to array

For example -
Let us consider
var arr = [23,56,87,91,58,13]; then

i) var new_arr = arr.slice(); → [23,56,87,91,58,13]

ii) var new_arr = arr.slice(2); → [87,91,58,13]

iii) var new_arr = arr.slice(2,4); → [87,91]

iv) var new_arr = arr.slice(2,10); → [91,58,13]
3 to last since upper limit

⊛ Changing elements of an Array -

An array can be used to implement a to-do list. In that elements are added at end and as the first task is completed it come out and second task takes its position.

The JavaScript's shift() method is used to remove first element of the array then moves to other & it returns the removed item.

Similarly the pop() method is used to remove the item from ~~last~~ end of an array which also returns removed item.

```
var fruits = ['Apple', 'Mango', 'Orange', 'Kiwi'];
var cars = ['BMW', 'Ford', 'Volkswagen', 'Audi'];
document.write(fruits.shift()); → Apple
document.write(cars.pop()); → Audi
document.write(fruits); → ['Mango', 'Orange', 'Kiwi']
document.write(cars); → ['BMW', 'Ford', 'Volkswagen']
```

⊛ Objects as Associative Array –

As we have seen the dot (.) operator is used to access the properties of an object. We can also used that the access the properties of an array.

object.propty

object["property"]

This is popular with associate array.

An associative array is array of key : value pairs where keys acts as index to get any value.

To create an associative array used object literal as follows:

```
var arr = {"One": 1, "Two": 2, "Three": 3};
var y = arr["One"];
```

we can not use for loop to display all elements but with the help of for in loop we do that.

```
for (var key in arr)
{
   document.write(arr[key] + "<br>");
}
```

Other method of Array objects are

1. concat() → Combine two arrays
2. indexOf() → Returns index of first occurance of value
3. join() → combine arrays with specified character
4. lastIndexOf() → Returns index of last occurance
5. pop() → Removes last element of array
6. push() → Appends new element
7. reverse() → Reverse all array elements
8. sort() → sorts an array elements
9. slice() → Return a section of an array
10. shift() → Removes first element
11. unshift() → Inserts element to start
12. toString() → Return string representation of an array.

(2.2) Function –

A function is a block of code designed to perform a particular task.

A function is a group of reusable code which can be called anywhere in program. This eliminates the need of writing the same code again and again. As we have used Javascript function like alert() and write().

Functions are of two types :–

① Built-in functions

② User-defined functions

(*) Defining a function –

A function must be defined before it can be called in JS. It is recommended that the function should be define in <head> tag.

A function definition consists of four parts: the name, parenthesis, a code block, and an optional return keyword.

① Function name – It is the name assigned to the function. It is placed at top of function definition & to the left of parentheses.

The name must contain :
- Letter(s), digit(s) or underscore character
- No two functions in JS have same names

The name cannot be :
- Begin with digit
- Be a keyword
- Be a reserved word

② Parenthesis –

Parenthesis placed to the right ~~side~~ of function name at top of function definition. They are used to pass values to the function called as arguments.

③ Code block –

It consists of a set of statements that are executed when the function is called. Open and close french braces (curley braces) define the boundaries of code block.

④ Return (Optional) –

The return keyword tells the browser to return a value to the statement that called the function.

★ Writing a Function Definition –

In JavaScript a function can be written with argument or without argument. The syntax to write a function definition is given as:

Syntax =
```
function function_name ()
{
    //statements;
}
```

Example –
```
<script>
    function message()
    {
        alert("Hello world"); //Predefined function
    }
</script>
```

(*) Adding an Argument -

A JavaScript function can also take parameters (arguments). The arguments must be unique and seperated by comma.

Syntax -

```
function function-name(parameter-list)
{
    //Statements;
}
```

Example-

①
```
function cube(x)
{
    return x*x*x;
}
```

② Function expressions can include function name, which is useful for recursion.

```
var f = function fact(x) {
            if (x<=1)
                return 1;
            else
                return x*fact(x-1);
        };
```

③ Function expressions can also be used as arguments to other functions:

```
data.sort(function(a,b) { return a-b; });
```

⭐ Scope of Variable and Arguments –

The scope of variable refers to the visibility of variables. In JavaScript there are two types of scopes.

1. Local scope
2. Global scope

① Local scope – A variable can be declared within a function is called a local variable. A variable declared inside a function block is only accessible to code within that same function block. This type of variable

② Global scope – A variable can be declared outside a function which is called as global variable. and is available to all part of JS. This variable is accessible globally within any function in that script.

① </script>

```
function abc()
{
  var x=12; //local variable
}
</script>
```

② </script>

```
var a=23; //Global var.
function x()
{
  document.write(a);
}
function y()
{
  document.write(a);
}
x();
y();
</script>
```

(2.3) Calling a function -

A function is called from anywhere in JS. A function is called by using name followed by parenthesis.

Syntax - functionname();

(*) Calling a function without Argument -

A function having no arguments can be called by using function name followed by empty parenthesis.

```
//Program to display hello world message
<html>
<head>
   <title> Function without arguments </title>
   <script>
      function sayHello()    //Function body
      {
         alert ("Hello World!");
      }
   </script>
</head>
<body>
   <script>
      sayHello();    // Function call
   </script>
</body>
</html>
```

⑧ Calling a function with Arguments –

A function can take arguments which are seperated by comma. To call this function you have to pass arguments in order they are defined in function definition.

Syntax – function function_name(param1, param2, ---, Param_n)
{
//statements;
}

```
//Program to add and multiply two numbers.
<html>
 <head>
   <script>
      function add(x,y)
      {
        result = x+y;
      } document.write("Addition is "+result);
      function mul(x,y)
      { result = x*y;
        document.write("Multiplication is"+result);
      }
   </script>
 </head>
 <body>
   <script>
      add(12,35);
      add(25,82);
      mul(12,4);
   </script>
 </body>
 </html>
```

⋆ Calling function from HTML -

A function can be called from HTML code on web page. Typically, this type of function called in response to an event, such as when the web page is loaded or unloaded by the browser. The function call is same as previous except that we have to assign the function call as a value of an HTML tag attribute.

For example- <body onload = "sayHello()">

<body onunload = "sayGoodbye()">

```
<html>
<head>
  <script>
    function sayHello()
    {
      alert("Hello");
    }
    function sayGoodbye()
    {
      alert("Goodbye");
    }
  </script>
</head>
<body onload="sayHello()" onunload="sayGoodbye()">
</body>
</html>
```

⊛ Function calling Another Function –

Developers may divide an application into many functions, each of which handles a portion of the application. We can call one function from another functions.

```
<html>
<head>
<script>
    function logon()
    { var user= prompt("Enter username");
      var pwd = prompt("Enter password");
      var isValid= validateUser(user,pwd);
      if (isValid==true)
      {
         alert('Valid login');
      }
      else
      { alert('Invalid login');
      }
    }
    function validateUser(u,p)
     {
        if (u=='admin' && p=='123')
            return true;
        else
            return false;
     }
</script>
</head>
<body onload="logon()">
</body>
</html>
```

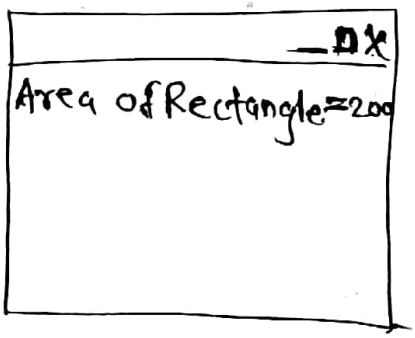
⭐ Returning a value from function -

Sometime we want the functions to return result back to statement from where they have called. So if we want to return a value to back then we can use return statement.

For example- area = calculate (radius);

```
//Program to calculate area of a rectangle
<html>
<head>
<script>
   function areaRect (l,b)
   {
      var a= l*b;
      return a;
   }
</script>
<body>
<script>
   area = areaRect(10,20);
   document.write ("Area of Rectangle =" +area);
</script>
</body>
</html>
```

Output



(2.4) String —

JavaScript strings are used to storing and manipulating text. It can contain zero or more characters within quotes.

JavaScript string is an object that represent a sequence of characters. A string can be any text inside single or double quotes.

There are two ways to create a string in JS:

(1) By string literal —

The string literal is created using single or double quote.

Syntax — `var stringname = "value";`

Example — `var fruit = "Apple";` OR `var fruit = 'Apple';`

(2) By string object (Using new operator) —

The new operator is used to create a string object.

Syntax — `var string-name = new String("value");`

Example — `var one = new String("ONE");`
`var car = new String("Audi");`

(*) Manipulate a string –

Strings are objects within Javascript language. They are not stored as character arrays, so built-in functions must be used to manipulate their values. These functions provide the ways to access the contents of a string variables. Javascript String object has various properties and methods

String Properties –

① constructor → Returns strings constructor function
② length → Returns the length of string
③ prototype → Allows to add properties and methods to an object.

String Methods –

All string methods return a new value. They do not change the original variable.

① charAt() → Returns the character at the specified Index (position)
② charCodeAt() → Returns the Unicode of character at the specified index
③ concat() → Joins two or more string & new joined strings
④ endsWith() → Checks whether a string ends with specified string/character

⑤ fromCharCode() → Convert Unicode values to characters

⑥ includes() → checks whether a string contains a specified string/chars

⑦ indexOf() → Returns position of first found occurance of specified value.

⑧ lastIndexOf() → Returns position of last found occurance of specified value.

⑨ match() → Searches a string for a match against a regular expression & returns a match

⑩ replace() → Searches a string and return a new string where the specified values are replaced

⑪ search() → Searches a string with RE & returns the position

⑫ slice() → Extracts a part of strings & returns a new string.

⑬ split() → splits a string into an array of substrings.

⑭ startsWith() → checks whether a string begins with specified characters

(15) substr() → Extracts the characters from string beginning at start position upto specified no. of characters

(16) substring() → Extracts the characters from a string between two specified indices

(17) toLocaleLowerCase() → Converts a string to lowercase letters according to host's locale

(18) toLocaleUpperCase() → Converts a string to uppercase letters according to host's locale.

(19) toLowerCase() → Converts a string to lowercase letters

(20) toUpperCase() → Converts a string to uppercase letters

(21) toString() → Returns the value of a String object

(22) trim() → Removes white spaces from both ends of a string

(23) valueOf() → Returns the primitive value of a String object.

// Program to find the length of string

```
<html>
<head>
    <title> Length of a string </title>
</head>
<body>
    <script>
        var str = "Java Script";
        var length = str.length;
        document.write("Length = " + length);
    </script>
</body>
</html>
```

Output
Length = 11

⭐ Joining a string -

When we want to concatenate two strings and form a single new string then that is called as joining a string. The new string contains the first and second string.

There are two ways to join (concatenate) two string

① Using concatenation operator (+)

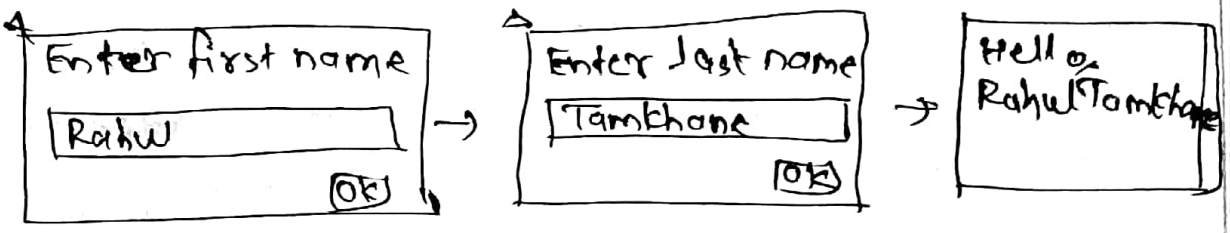
② Using concat() method of string object

```
var first = "Hello";
var second = "world!";
var final = first + second;    // Helloworld!
```

// Program to read first and last name & display fullname

```
<html>
<head>
   <title> Display Fullname </title>
</head>
<body>
<script>
    var fname = prompt('Enter firstname', ' ');
    var lname = prompt('Enter last name', ' ');
    var fullname = fname + lname;
    document.write('Hello' + fullname);
</script>
</body>
</html>
```

Output.



(*) Retrieving a character from given position –

A string is an array of characters. Each character in a string is an array element that can be identified by its index.

To retrieve a character from given position we have to use charAt() method of string object.

Syntax– `charAt(index)`

Example - var str = "Python";
var res = str.charAt(3); // 'h'

// Program to demonstrate use of charAt()

```
<html>
 <head>
    <title> charAt() demo </title>
 </head>
 <body>
    <script>
       var str = "Rahul";
       for (var i=0; i<str.length; i++)
       {
         document.write(str.charAt(i) + "<br>");
       }
    </script>
 </body>
</html>
```

Output
R
a
h
u
l

(*) Retrieving a position of a character in a string -

Two methods are used to find the character or characters in string

① indexOf() - Returns a position of first occurance of a specified character in string.

Syntax - string.indexOf(searchValue, [start])

start is optional, specifies start position of search. Returns -1 if not found.

(2) search() - Searches a string for a specified value & returns the match position. Return -1 if no match found. search value can be a string or regular expression

Syntax - string.search(searchValue)

```
var str = "Hello world!";
var n = str.indexOf("e");          // 1
    n = str.indexOf("o",5);        // 7

var str = "I want to learn JavaScript";
var n = str.search("learn");       // 10
```

(*) Dividing Text -

Suppose we want to divide a comma seperated string into seperate strings into array. The split() method create a new array and then copies portions of the string (substrings) into its array elements. This method requires one argument which is delimiter. We can divide a string into many substrings by using split() method.

Syntax - split(delimiter)

string.split(delimiter[, limit])

Example-

(1)
```
var str = "How are you doing today?";
var res = str.split(" ");
document.write(res);
```

output - How,are,you,doing,today?

② If omit the seperator parameter

res = str.split();

output - How are you doing today?

③ seperates each character

res = str.split("");

output - H,O,W, ,a,r,e, ,y,O,u, ,d,O,i,n,g, ,t,O,d,A,y,?

③ Use limit parameter.

res = str.split(" ", 3);

output - How, are, you

④ Use a letter as a seperator

res = str.split("O");

output - H,W are y, u d, ing t, day?

★ Copying a sub string -

We can also copy one substring into a new string. In order to extract a small portion from a string, JavaScript provides two methods of string object.

① substr() - This method extracts parts of a string, beginning at the character at the specified position & returns the specified no. of characters.

Tip - To extract characters from the end of string, use negative start number.

Note - This method does not change original string.

Syntax – string.substr(start,[length]);
① ② optional

Example –

① var str="Unicode";
var res= str.substr(2); // icode
Extract string from index 2 to the last (end)

② var res = str.substr(0,1); // U
Extract only first character

② substring – This method extracts the characters from a string, between two specified indices and returns the new sub string. (Not includes 'end' itself)
If start > end → will swap two arguments
means
str.substring(1,4) == str.substring(4,1)

Syntax – string.substring(startindex [,end index]);

Example – var str="Hello world!";

① res = str.substring(0,1); → Extract only first character
H

② str.substring(2); → Extract from 2 to rest of string
llo world!

③ str.substring(4,1); → swap two arguments
ell

④ str.substring(-3); → start < 0 so it will start extraction from index position 0
Hello world!

⋆ Converting String to Number -

In Javascript, 82 and '64' both are different things first is number and second is string containing a number. If we add these two we will get 8264 rather than addition 146. so to convert string datatype into number JS provides different methods for conversion.

① parseInt() - This method converts a string into an integer (a whole number)
It accepts two arguments. The first argument is the string to convert and the second argument is called the radix.

```
var size = '42 inch';
var integer = parseInt(size, 10);   // returns 42
```

② parseFloat() - This method converts a string into a point number (floating no.)

```
var size = '3.14 inch';
var floatnum = parseFloat(size);   // returns 3.14
```

③ Number() - This method converts a string to a number. If string contains other than number then it will return NaN (Not a Number)

```
Number('245')      // returns 245
Number('24.6')     // returns 24.6
Number('3.14 inch') // returns NaN
Number('16 px')    // returns NaN
```

// Program to display cube of given no.

```
<html>
<head>
  <title> cube of a number </title>
</head>
<body>
  <script>
    var n = prompt('Enter a value',' ');
    n= parseInt(n);
    var cube= n * n * n;
    document.write("Cube is " + cube);
  </script>
</body>
</html>
```

ⓐ Converting Numbers to strings –

In Javascript, you can convert any value to a string by using two ways.

① toString() method – This method converts both integer and decimal (float) values to string.

Syntax – number.toString();

Example – var num = 100;
var str= num.toString();

② Concatenation operator (+) –

The concatenation operator (+) is used to combine a string to a number. It automatically calls the toString() method on numeric values.

Example – var num= 340;
var str= "" + num;

⋆ Changing the case of string -

JavaScript provides two methods to change the cases of strings:

① **toLowerCase()** — Converts all characters to lowercase in a string. Has no arguments.

A - 65 → a - 97 (65+32)
B - 66 → b - 98 (66+32)
⋮ ⋮
Z - 90 → z - 122 (90+32)

② **toUpperCase()** — Converts all characters to uppercase in a string. Has no arguments.

a - 97 → A - 65 (97-32)
b - 98 → B - 66 (98-32)
⋮ ⋮
z - 122 → C - 90 (122-32)

```
<script>
    var str = "Rahul Tamkhane";
    var cap = str.toUpperCase();
    var small = str.toLowerCase();
    document.write(str + "<br>");
    document.write(cap + "<br>");
    document.write(small);
</script>
```

⋆ Finding a Unicode of a character -

Computers understands only numbers & not characters. When you enter a character it is automatically converted into a number called unicode number that computer understands.

Unicode is a standard that assigns a number to every character, number and symbol.

① charCodeAt() -

This method returns the Unicode of the character at the specified index in a string.

Syntax - String.charCodeAt([index]);

If index is not provided then method will use 0 as default.

Example - var str = "JavaScript";
var c = str.charCodeAt(); 74
var a = str.charCodeAt(1); 97

② fromCharCode() -

This method converts Unicode values into characters. This is a static method of String object.

Syntax - String.fromCharCode(n1, n2, ..., nx)

Example - var res = String.fromCharCode(72, 69, 76, 76, 79);

Output - HELLO