4/3/2020

Java Script

A predefined scripting language only one language through which we could able to implement logical ability, arithmetic operations, dom operations in a web page.

(1) It is a client side scripting language capable of executing within the client side it self.

(2) It is the only programming language can be understandle by only browser.

(3) By default every browser comes with individual java script engine through which java script instructions gets complied and executed within client side.

(4) It comes with pre-defined objects and methods through which dom operations can be perporformed on a page.

(5) It is a super heroic programming lang. capable of creating, updating or deleting any html element or its corresponding css properties

(6) It comes with pre-defind objects and methods through which we could implement AJAX calls within the page.

(7) It provides pre-defined methods and objects through which any action being performed on the page can be handled by invoking corresponding call back methods.

(8) Java Script follows top to bottom Execution process, it doesn't hold the concept of having pre-defined static points (like, init or main method)

5/3/20

Following are the different ways we can inject java script

(1) use inline java script under script tag.

```
<script type = "text/javascript">
...
// javascript code
</script>
```

(2) External java script using script tag.

```
<script type = "text/javascript" src = "<External
js file path>"></script>
```

Eg: `<script src = "http://sample.com/data/abc.js"
type = "text/javascript"></script>`

Note:

(1) In a single page any number of script tag can be placed.

(2) Inside a script tag, we can add any number of lines java script code.

(3) using single script tag we can include one external java script file.

(4) script tag can be placed anywhere within the page.

Datatypes in javascript

data type specifies the type of data a variable could hold.

In java script while declaring a variable it is not required to specify the datatype which declaring the variable

'var' is pre-defined keyword through which we could declare variable in javascript.

following are the different types / data types been supported in java script.

(i) Number
(ii) string
(iii) object
(iv) undefined
(v) null
(vi) boolean
(vii) function

Eg:
```
var a;
a = 90;
var b, name;
name = 'RAJ';
b = true;
etc.,
```

6/3/20

① write a program to work with employee details like name, age, gender, department and basic salary and calculate Pf, HRA, total salary as like below.

Pf = 12 percent of basic sal
hra = 25 percent of basic sal
Total sal = basic + Hra + Pf.

```
<script type = "text/javascript">
Var ename = "Tina";
Var eage = "25";
Var gender = "female";
Var edep = "IT";
Var esalary = "25000";
Var Pf = (12/100) * esalary;
Var hra = (25/100) * esalary;
Var Totalsalary = esalary + Pf + hra;
Console.log ("name is " + ename + "\n age is " +
        eage + "\n gender is " + gender + "\n dep
        is " + edep + "\n salary is " + esalary +
        "\n Pf is " + Pf + "\n hra is " + hra + "\n
        Total salary is " + Total salary )

</script>
```

Java Script Type of method

a pre-defined method being supported in java script using which we could able to get type of data a variable is holding.

* It takes a variable as a parameter and returns, the type of data the variable is holding.

Syntax:

```
typeof (<variable name>);
```

eg:

```
Var a = 10;
typeof(a); // Returns number
```

var isfeepaid = true;
typeof (isfeepaid); // returns boolean

* In java script, it treats decimal value, float as "Number".
* In javaScript if any variable is not initialized it will be defined as "un-defined".

Variable Hoisting

Most of the programming languages forces to declare a variable only at starting of the program so that it can allocate the memory and then works on variable.

* In java script, it is not mandatory to declare the variables only at starting of the program but can be declared any where we wanted.
* while executing the application java script peyforms variable hoisting proces in which it identifies all the variable declarations within the application and moves all the declarations to starting of the application.
* the process of moving all the variable declarations to the starting of the application is called variable hoisting.

Note:

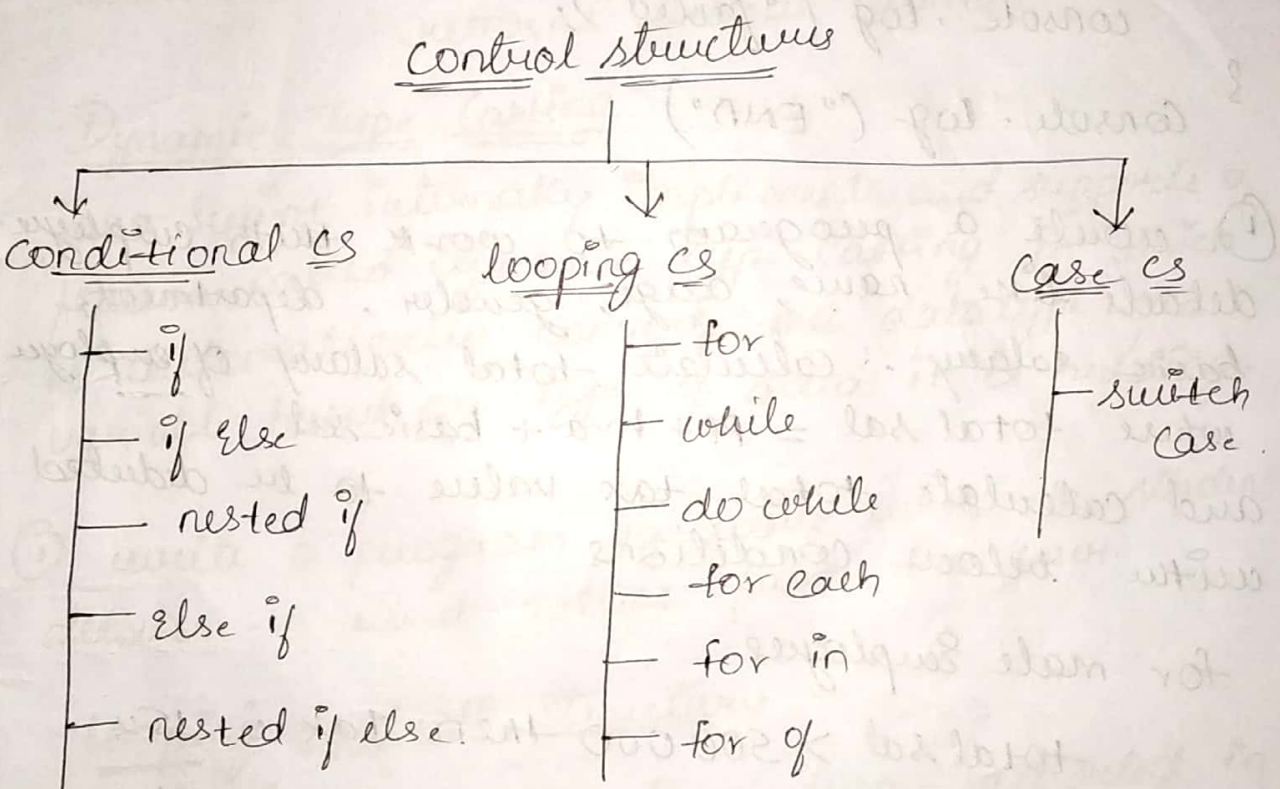
* While moving the declarations it only moves declarations to the starting but not the initialized values.

7/3/20

Control Structures

following are set of control structures being supported through which we could able to control the sequence of Execution flow in a application

1. conditional CS
2. looping CS
3. case CS



eg:

```
Var total = m1+m2+m3+m4;
Var avg = total/4;
Console.log("student name is" + sname);
console.log("age is" + sage);
console.log("total is" + total);
console.log("avg is" + avg);
if (avg >40) {
    console.log("passed");
    if (avg >=90) {
```

```
        console.log("you go distinction");
    } else if (avg >= 60) {
        console.log("got first class");
    } else {
        console.log("second class");
    }
} else {
    console.log("failed");
}
console.log("END")
```

(1) write a program to work with employee details like name, age, gender, department, basic salary .. calculate total salary of employee where total sal = pf + hra + basic sal and calculate total tax value to be deducted with below conditions

for male Employees

total sal > 500000 then tax is 15%.

total sal > 2500000 then tax is 10%.

if not above

tax is 0

for female Employees

total sal > 500000 then tax is 10%.

total sal > 2500000 then tax is 5%.

if not above.

tax is 0

10/3/20 Type Casting:

The process of converting data type of a variable to another is called type casting. following are the pre-defined methods supported in java script. through which we can forcefully convert any data type content to number.

parseInt() - Takes any datatype value and returns number.

ParseFloat() - Takes any datatype value and returns decimal number.

Dynamic Type Casting

java Script internally implements and supports a feature called dynamic type casting in which it automatically converts the datatype of a variable based on type of data it is holding.

① write a program to work with the student details to read values from the user.

Looping control Structures

Looping control structures were supported in java script takes the set of instructions and repeat them for multiple times.

1. for loop control structure

Syntax: for (<initialization>; <condition check>; <increment /decrement>)
{---
---// set of instructions get executed multiple times until the provided condition is satisfied
}

Eg: for(var i=10; i<=90; i++)
{

}

11/3/20

① write a program to display all the even nos from 10 to 20.

② write a program to find out all the odd number b/w 99 to 11

```
<Script type>
  for (var i = 99; i >= 11; i--)
    {
     if (i % 2 != 0)
     {
       console.log (" odd number" + i);
     }Else
      {
       console.log (" Even no." + i);
      }
```

③ write a program to find the sum of all even nos b/w 999 to 99.

```
Var Sum = 0.
for (var i = 999; i >= 99; i--)
  {
   if (i % 2 == 0)
   Sum = Sum + i
   {
   console.log (Even number is i)
   Sum = Sum + i;
  } console.log ("Sum is" + Sum);
```

difference b/w '=', '==' and '===' operators

* '=' → It is an assignment operator using which we could able to assign a value of right side to the variable of left side.

Ex: a = 10

* '==' → It is an comparison operator used to compare value of left side with value of right side and returns a boolean value (true/false) based on the equality of the values

```
a = 10
b = 20
a == b // returns false.
```

```
a = "10";
b = 10;
a == b // returns true
```

Note: == operator, while comparing values it only checks for value equality but not type equality.

* '===' → It is almost like '==' operator used to compare value equality where the only difference is while comparing values it not just checks for value equality but it also checks for type equality of data.

```
a = 10
b = 20
a === b // returns false.
```

```
a = "10";
b = 10
a === b // returns false.
```

④ write a program to display Even nos from user given min range to max range

```
Var min = prompt("Enter min value");
min = parseInt(min);
Var max = prompt("Enter max value");
max = parseInt(max);
for (Var i = min; i <= max; i++)
{
  if (i % 2 == 0)
  {
    console.log(i)
  }
}
```

⑤ write a program to display the table till 20 for user given no.

⑥ write a program to check whether given no. is prime or not

⑦ write a program to display the prime no.'s b/w 10 to 100

⑧ write a program to find the sum of all prime nos 999 to 99

12/3/20

While Loop Control Structures

It is almost like for loop control structure used to iterate instructions more than once where the only difference is while loop only takes the condition, initialization, increment or decrement can be placed anywhere within the code.

Syntax:

```
while (<condition>) {
    ...
    ...
    ... // set of instructions gets executed for multiple times, until the provided condition is unsatisfied.
```

① write a program to find the sum of individual values of a given no.

```
<Script>
    Var n = 272;
    Var rem;
    Var Sum = 0;
    while (n > 0) {
        rem = n % 10;
        Sum = Sum + rem;
        n = n / 10;
        n = parseInt(n);
    }
    console.log ("Sum is" + Sum);
</Script>
```

O/P: 11

② write a program to find reverse of a given no.

③ write a program to check whether given no is palindrome or not.

④ write a program to find the lucky no. of a given no.

⑤ write a program to check whether given no is armstrong no or not.

⑥ write a program to find out total no of digits in a given no.

Do while looping control structure

It is almost like a while loop control structure use to repeat a set of instructions for than once where the only difference is while loop will execute the instructions only when the provided condition is true, where as do while executes set of instructions at least for once irrelavant of whether provided condition is true or false.

Syntax:

```
do {
    ... // set of instructions for sure gets executed irrelavant of whether provided condition is true or false
} while (<condition>);
```

Case Control structure

It takes multiple cases within it and invokes corresponding / particular case. based on the value been passed.

Syntax:

```
Switch (<value>){
  case <value>':
    ...
      break;
  case <value 2:
    ...
      break;
  ...
  ...
  default:
    ... // gets executed automatically if
none of cases matches.
}
```

13/3/20

Array Data Structures

following are different data structures being supported in java script through which different type of data can be stored and retrieved.

1. Arrays
2. Json
3. Hash Map
4. Hash Tables

Arrays

a predefined data structure through which we could able to hold group of multiple values within it

① In java Script, arrays are heterogeneous data structures capable of holding different types of data within it.

② Array is capable of holding different and multiple types of data, every other value will be identified with single variable name.

③ all the values within the array will be automatically assigned a ~~unique~~ numeric index value

④ The index value always stard with "zero".

⑤ '[]' braces are always used to represent arrays in any language.

⑥ within the memory, arrays always occupy continous memory allocation for all the values within it.

⑦ we make use combination of array name and corresponding index ~~value~~ position while working with array values

⑧ following are the two different ways we can create a array in java Script.

```
Var a = new array ( ) // Dynamic allocation
```

(or)

```
Var a = [ ]; // Empty array.
Var a = [34, 34, 34, 5]; // declaring an array with value.
```

Eg:

```
<Script>
    Var Sname = "RAJ";
    Var Sage = 20;
    Var gender = 'Male';
```

```
Var marks = [90, 55, 77, 88];
Var total = 0
for (var i=0; i < marks.length; i++) {
    total = total + marks[i];
}
Var avg = total / marks.length;
console.log("student details");
console.log("name" + sname);
console.log("Gender" + gender);
console.log("Total" + total);
console.log("avg is" + avg);
</script>
```

(1) write a program to declare an array with values and find the sum of values in it

```
Var Values = [10, 20, 30, 40, 50];
Var Sum = 0
for (var i=0; i < values.length; i++) {
    Sum = Sum + values[i];
}
console.log("The sum is" + Sum);
```

(2) WAP to declare an array with values and display all the even nos within it.

```
Var values = [10, 20, 30, 40, 50];
for (var i=0; i < values.length; i++) {
    if (i%2==0) {
```

```
        print(i)
    }
}
```

O/P: 10, 20, 40.

(8) WAP to declare an array with values and list out all the prime nos in it.

```
var data = [2, 5, 41, 32, 64];
for(var i=0; i< data.length; i++){
    is prime = true;
    var n = data[i]
    for(var j=2; j<n; j++){
        if(n%.j==0){
            is Prime = false;
            break;
    } }
    if(isprime){
        console.log(n); }}
```

(14) write a program to read values in array dynamically and list out all palindrome nos

(5) WAP to read values into array dynamically and list out armstrong nos within it

(6) write a program to find the largest and smallest values in array

(4) WAP to find out second largest values within the array.

(8) WAP to sort an array either ascending or descending based on user request.

pre-defined methods being supported in arrays

following are pre-defined methods being supported in java script which can be applied on arrays

- length - It is a property which returns total number of values in a array.
- push (<values>) - used to insert single/ multiple values to an array from right direction
- pop() - used to delete single value from an array from right direction
- shift() - used to delete single values from an array from left direction
- unshift(<values>) - used to insert single / multiple values to an array from left direction
- Splice (start position, <no. of values to delete>, <optional values to be inserted>) - used to insert or delete values.
- join() - used to merge all values of array to a single value with/without seperator.

14/3/20

Pre-defined methods can be applied on strings

Every string being created in java script internally it gets created in the form of an array. following are pre-defined methods by default supported in java script which can be applied on strings

- length - property returns total no. of characters with in a string, including spaces.

- charAt (<index>) - Returns the character present at provided index position.
- charCodeAt (<index>) - returns the ASCII value of character present at provided index position
- subStr (index, length of sub string) - returns the substring from given string.
- replace ("what", "with") - used to replace string parts with required strings
- split (<"optional Sperator">) - ~~checks whether the provided string is part~~ used to split the main string in the form of an array.
- includes ("<string>") - checks whether the provided string is part of main string, and returns true/false.
- indexOf ("string") - returns at what index provided substring exists in main string.
- toLowerCase() - converts provided string to lower case format.
- toUpperCase() - converts provided string to upper case format
- slice (startIndex, endIndex) - returns a substring from one index to other.

Note: any string being defined in js it occupies the memory internally as like an array where each character occupies individual block with corresponding index value.

(1) write a program to find out no. of vowels present within the string.

(2) WAP to find out total no. of characters present within the string without using any pre-defined property.

(3) find out total no. of spaces within the string without any pre-defined methods.

(4) find out total no. of special characters present within provided string

Working with java Script "Date" object

"Date" is a pre-defined class supported in java script using which we could able to work with systems current date or user defined custom date

Syntax:

```
Var date = new Date(); // creates a reference
                          of systems current date.
Var date = new Date(<user specific date>); // creates
                          a reference of user specified date.
```

following are the pre-difined methods which can be applied on date object:

- getDate() - returns current date value (1-31)
- getMonth() - returns current month index value (0-11)
- getFullYear() - returns full year value.

- getHours() - return hours value (0-24).
- getMinutes() - returns 0-59.
- getSeconds() - returns 0-59.
- getMilliSeconds() - returns 0-999.

etc.

- getDay() - Returns value 0-6

① write a program to read custom date format from the user and return the date accordingly.

1. mm/dd/yy - 03/14/20
2. mmmm, dd yy - March, 14 20
3. mmmm, dd yyyy - March, 14, 2020
4. day, mmmm dd yyyy
5. mm-dd-yy
6. mm/dd/yy hh:mm
7. mmmm, dd yy hh:mm:ss

Java Script functions

The set of instructions binded as a module with { } assigned with user defined custom name which can be reused any other within the page is called a function.

① 'function' is a pre-defined keyword through which we can defined a function in java script (until ECMA5).

(2) In a single page, we can define any no. of java script functions

(3) Once the function been defined it can be invoked any no. of times through function name.

(4) while defining the function we name it through function keyword while calling we simply use function name.

21/3/20

local variables & Global variables

*Any variable which is declared outside of a module will become global variable can be accesble any where within the application

*Global variables can be accesible and modifiable any where within the page.

local variables

Any variable which is declared within the module will become private data or local data of that particular module.
The private data of any module cannot be accesible outside of the module.

Eg:

```
<script>
    Var a = 10; // global variable can be
                accessable anywhere within
                the app.
    ...
    function test () {
        ...
```

var b = 90; // Its local / private data can only be accessable within this function, cannot be accessable outside of this module.

}

</script>

Accessing private data of a function, outside of it:

① Through passing parameters while calling a function

② Returning a value from called function to calling function

③ Both passing parameters and returning values

① Passing parameters while calling a function

while calling a function within another function we could able to pass local variables of calling function to called function

(i) while calling a function we can pass any no of parameters.

(ii) while defining a function we could able to catch the values being passed while calling a function, using temporary variables.

(iii) The parameters being passed while calling a function are called actual parameters where as the temporary parameter declared

at function definition are called formal parameters.

Eg:
```
function test() {
    var a = 90;
    var b = 99;
    addValues(a,b); // a, b are actual parameters
    ...
}

function addValues(p1, p2) { // p1, p2 are formal parameters
// p1, p2 holds the values of a & b.
    ...
}
```

(iv) while passing formal parameters at function definition we don't need to specify or declare those parameters. we simply specify the variable names.

"Arguments" Keywords → 23/3/20

a pre-defined keyword supported in java script and by default available in every function definition which holds all the parameter values been passed to that function in the form of an array.

(i) In java script, it is not mandatory to catch the parameters in function definition while matching function call with

function definition it only matches function name but never looks for type of parameter or no. of parameter being passed.

(2) Irrelevant of whether the parameters been catched or not, arguments keywords gets created in every function and holds all the parameters being passed to it in the form of a array.

JSON Data Structure

JSON - Java Script Object Notation

(1) It is a pre-defined data structure being supported in java script by default.

(2) It is used to not just hold the data but, hold the data along with extra information of particular data.

(2) under JSON data gets stored in the form of key and value based.

(3) all the keys within the json object should be only of string data type where as the corresponding value could be of any java Script supported data type.

Ex: Number, string, boolean, string and Even an object

Eg:
```
Var data = {
    "name" : "RAT",
    "age" : '22',
```

"gender": 'Male'
};

(4) while accessing data from object we make use the combination of object name and corresponding key.

Eg: data.name
data.age
data.gender
etc.

24/3/20

Accessing data inside json object in diff ways

following are the 2 ways we could able to access data within the jason object

(1) using . delimeter
(2) using key as an index value of an array

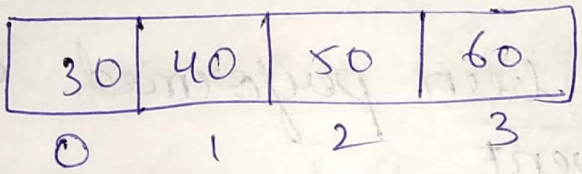
Memory Allocation of Ison objects

Internally json object gets memory allocated same as like array where the only differnc is, the key of data will be assigned as index value by default.

Eg: Var data = {
name: "Raj",
age: 20,
gender: 'male'
}

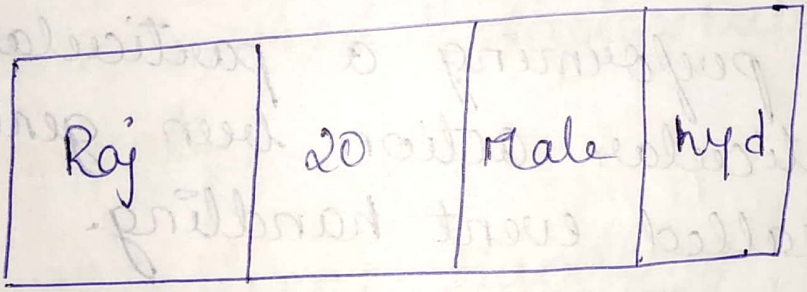
data.name
data['name']

marks



marks[1]

data



data.name

data['name']

name age gender loc.

Anonymous function 28/3/20

Any function which is defined without a name is called an anonymous function. In a case we want to define a function it will be invoked only once within the function do not waste the memory by giving a name to it, we can simply create anonymous function.

Syntax: function () {
....
}

For in looping control structure

through
a pre-defined looping control structure which we could able to iterate over a json object

Eg: for (temp in <object>) {
.....
....
}

Eg: var data = {.....};
for (var temp in data) {
....
// temp returns particular key on

Note:

* delete is a pre-defined keyword been supported in js using which we could able to delete a particular key along with corresponding value from an object.

Eg: var data = {
name : "raj",
age : 30,

gender : 'male'

};

delete data.age;

"This" Keyword:

a pre-defined keyword been supported in js which refers the current object data.

in order to access objects data within its corresponding methods, we can access either by using objects name or "this" keyword.

Js 'SetTimeout' and 'SetInterval' Methods

The two pre-defined methods been supported in js using which we could able to invoke set of instructions after not just when the controller reaches to it but, executes the set of instructions only after the provided interval time.

Syntax:

SetTimeout (<callbackmethod>, <delayvalue>);

Eg:

```
SetTimeout (function () {
    .....
    .... // set of instructions gets executed
only after 2 seconds of time.
}, 2000)
```

SetInterval(<callmethod>, <delayvalue>);

Eg: SetInterval(function(){

.....

.....// set of instructions keep gets Executed within every 2 sec of interval time.

}, 2000)

* SetInterval and SetTimeout always works the same where the only difference is setTimeout invokes the function only once after the provided interval time where as SetInterval method invokes the call back method repetively with a gap of provided interval time.

Note:

* clearInterval() is a pre-defined method supported through which we could able to stop a running interval job.

Eg: var ref = SetInterval(function(){

.....// set of instructions to repeat

}, 3000);

if (condition){

clearInterval(ref);

2/4/20

Event Bubbling and event Capturing

Any time we add event handling for the same event type for both parent and child, event bubbling or capturing will specify the order of event invoking either parent to child or child to parent.

Event Bubbling:

It is the default one which happens, Invokes the child event first following the parent.

Event Capturing:

It is vise versa of event bubling in which parent event fires first following with the child.

→ AddEventListner method takes following three parameters.

1. Type of Event
2. Call back method.
3. a boolean value which specifies whether to follow event bubbling or capturing