**Contents**

1. Scripting Language and Applications

2. Java Script History

3. Advantages & Disadvantages of JS.

4. Java-script Enabling.

5. Java-Script Placing.

6. Literals

7. Variable

8. DataType

9. Hoisting

10. Type Casting

11. Pop-Up Boxes

12. Control Statements

13. Function

14. Data Structures.

14.1.Array

14.2.JSON

14.3.String

15.Pre Defined Objects.

15.1.Date

15.2.Math

16.Event Handling

17.Validation

**1.Scripting Language and Applications:-**A scripting language is a programming language designed for integrating and communicating with other programming languages. The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted.

**Application of Scripting Languages:** Scripting languages are used in many areas:

* Scripting languages are used in web applications. It is used in server side as well as client side. Server side scripting languages are: JavaScript, PHP, Perl etc. and client side scripting languages are: JavaScript, AJAX, jQuery etc.
* Scripting languages are used in system administration. For example: Shell, Perl, Python scripts etc.
* It is used in Games application and Multimedia.
* It is used to create plugins and extensions for existing applications.

**2.Java Script History**:- Java script firstly appeared in netscape 2.0 in 1995 with name live script. Later nestcape changed its name to java script. The java script is dynamic programming language.

Q) What is dynamic programming language?

A) A **dynamic programming language** is a programming language in which operations otherwise done at compile-time can be done at run-time. For example, in JavaScript it is possible to change the type of a variable or add new properties or methods to an object ,delete existing properties from object while the program is running.

The java script is light weight programming language.

Q) what is light weight programming language?

A) The light weight programming languages uses small amount of main memory while it is running and it is easy to implement. These programming languages have simple syntax and semantics so they could be learnt easily and in little time.

The java script is used to write both server side script and client side script.

**3.Advantages & Disadvantages of JS:**

The Advantages are

* Live Server Interaction:- we can validate user input before sending page to webserver. This saves server traffic, which means less load on your server.
* We can add dynamic interactivity to web pages using java script.

The disadvantages are

* Client-side java script does not allow reading or writing of files.
* Java script does not has multi threading /multi process capabilities.
* Java script can not be used to developing networking applications.
* Java script can’t directly communicate with DB& file sysem in our computer system.

**4.Java –Script Enabling:-**All modern browsers come with built-in support for java –script . we need to enable or disable this support manually.

4.1) Tools-> Internet options from menu.

4.2) Select security tab from dialog box.

4.3) Click custom level button.

4.4) Scroll down till you find scripting option.

4.5) Select enable or deselect enable button under Active scripting.

4.6) Finally click ok and come out.

**5.JavaScript –Placement:-**The java script code is collection of statements. The java script code can always be in between <script> and </script> tags. We can put <script> tag in

* Head section of document.
* Body Section of document.
* Both head & body section.

We can put script code in another way. We write java script in separate file (external file). We have to save that file with .JS extension. Now Include That External file in Head section of Html document.

**6.Java Script Literals:-** In a programming language, a literal is constant value.

1.1) Numeric Literals:Number literal can be written with or with out decimals.

Ex: 20,70.47.

1.2) String Literals:String literal can be written in single or double quotes.

Ex: “sukumar”, ‘sukumar’.

1.3) Object Literals:It defines an object.

{ name:’suku’,age:37}

1.4) Function literals:It defines a function.

Function function-name(par1,par2,…parn){statements}

1.5) Expression Literals: It evaluates to value.

Ex: 3+8, 5+7.5

**7.Variable:-**Variable is named container. You can place data in container and refer to data simply by naming container.

Syntax to Declare Variable:

Var name1,name2,……name-n;

Note:-If you don’t specify an initial value for a variable with the var statement, the variable is declared, but its initial value is **undefined** until your code stores a value into it.

Syntax to Variable Initialization:

Variable-name=value;

Note:- Initializing a variable means **specifying an initial value to assign to it** (i.e., before it is used at all).

We can assign the initial value to variable at the time of declaration.

Syntax: var name1=value;

Syntax to Assign value to Variable:

name= value;

Syntax to Constant variable Declaration:

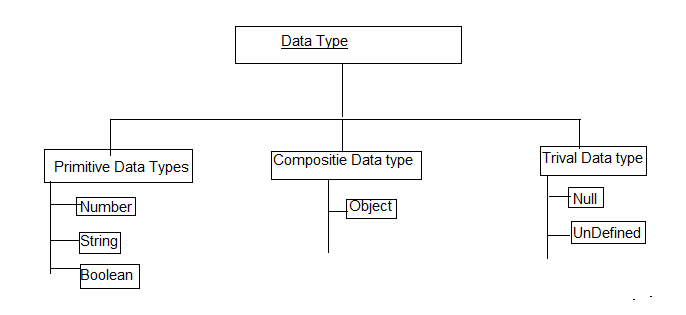
const name=value;

Note:-

* + - When we declare constant variable, we should not write keyword(var).
    - When we declare constant variable, we should assign value otherwise we get error.
    - We can’t change constant variable value in the program.

Java script is **un-typed language/Loosely typed Language**. It means it does not require a data type to be declared. You can assign any literal values to a variable, e.g., string, integer, float, boolean, etc.

**8. Data Types:-**Data types specifies the type of data that a variable can hold. In JavaScript while declaring a variable it is not required to specify the data type while declaring the variable. ‘var’ is a predefined keyword through which we could declare variable in JavaScript.

****

8.1. number:The variable stores integer , float value. The numbers are always stored as double precision floating-point numbers. The number stores in 64 bits.

This format stores numbers in 64 bits, where the number (the fraction) is stored in bits 0 to 51=52bits, the exponent in bits 52 to 62=11bits, and the sign in bit 63=1bit.

Therefore number type variable can’t store integer value which is largerthan

2(253-1) and can’t store value which is less than –(253-1). To avoid this problem, java script provided BigInt data type.

8.1.1)BigInt:- Big Int is a special numeric type that provides support for integers of arbitrary length. A bigint is created by appending n to end of a n integer literal or by calling the function BigInt that creates bigint from strings, numbers etc.

Example:

const a= 1234561235n;

const b=BigInt(“12334343434324953453”);

Note:- All operations on bigint returns BigInt type value.

Ex:- alert(1n+2n); // 3 is BigInt type value.

We can’t mix BigInt variable with other type variable in arithmetic expression.

8.2. Boolean:- Js supports only two Boolean values which is true and false.

8.3.Null:- Variable can be emptied by setting value to Null.

8.4.undefined:- The value of variable with no value is undefined.

Example: var car;

8.5.String:-String is group of characters. String should be written inside single or double quotes.

**9.Hoisting:**-Most of the programming languages forces to declare a variable only at the starting of the program so that it can allocate the memory and then works on variable. There are two types of Hoisting.

9.1. Variable Hosting.

9.2. Funciton Hosting.

9.1.1. Variable Hosting:-

* In JavaScript, it is not mandatory to declare the variables only at the starting of the program but can be declared any were we wanted.
* While executing the application JavaScript performs variable hosting process in which it identifies all the variable declarations with in the application and moves all the declarations to starting of the applicationsandinitialized them with ‘undefined’ or specified value .
* The process of moving all the variable declarations to the starting of the application is called variable hosting.

Note:- After moving all variable declaration to beginning, in them ‘undefined’ value is kept.

Examples:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

       console.log('Value of a is:'+ a);

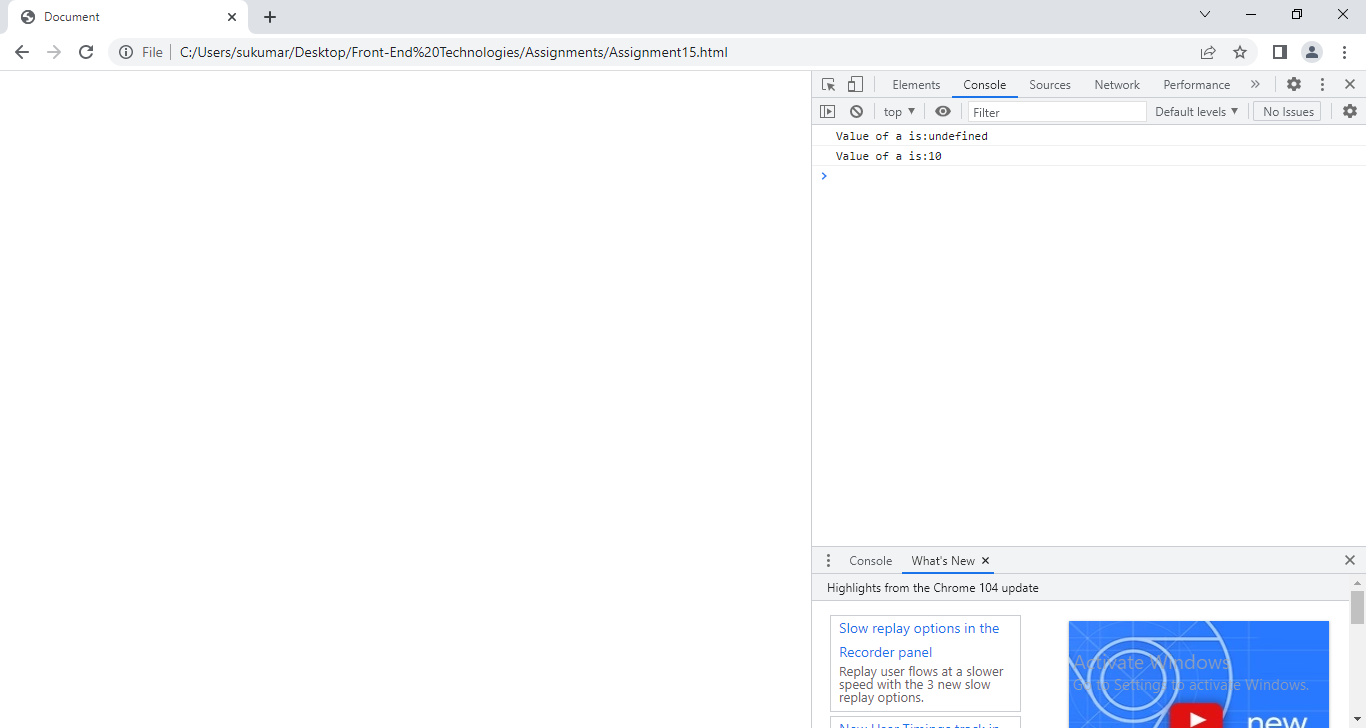
       vara=10;

       console.log('Value of a is:'+a);

   </script>

</body>

</html>



**10. Type Casting:-**Converting a data type into another is known as type casting.Sometimes there is a need to convert the data type of one value to another. There are two types of conversions.

10.1. Implicit Conversion.

10.2. Explicit conversion.

**10.1. Implicit Conversion**:- Java script compiler automatically converts one data type into another data type. This conversion is said to be **implicit conversion**. For implict converting , the programmer does not write any statement in java script program.

Examples:

a)When a number is added to string,Java script converts number to string before concatenation.

Ex:-1 10+”suku” -----------🡪10suku

2 .10+undefined-----------------🡪10undefined.

3.10+null--------------------------🡪10null

4.10+true----------------------------🡪10true.

b) Implicitly string is converted into number.

let result;

result = '4' - '2';

console.log(result); // 2

result = '4' - 2;

console.log(result); // 2

result = '4' \* 2;

console.log(result); // 8

result = '4' / 2;

console.log(result); // 2

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width='', initial-scale=1.0">

    <title>Document</title>

</head>

<body>

 <script>

    vara=10;

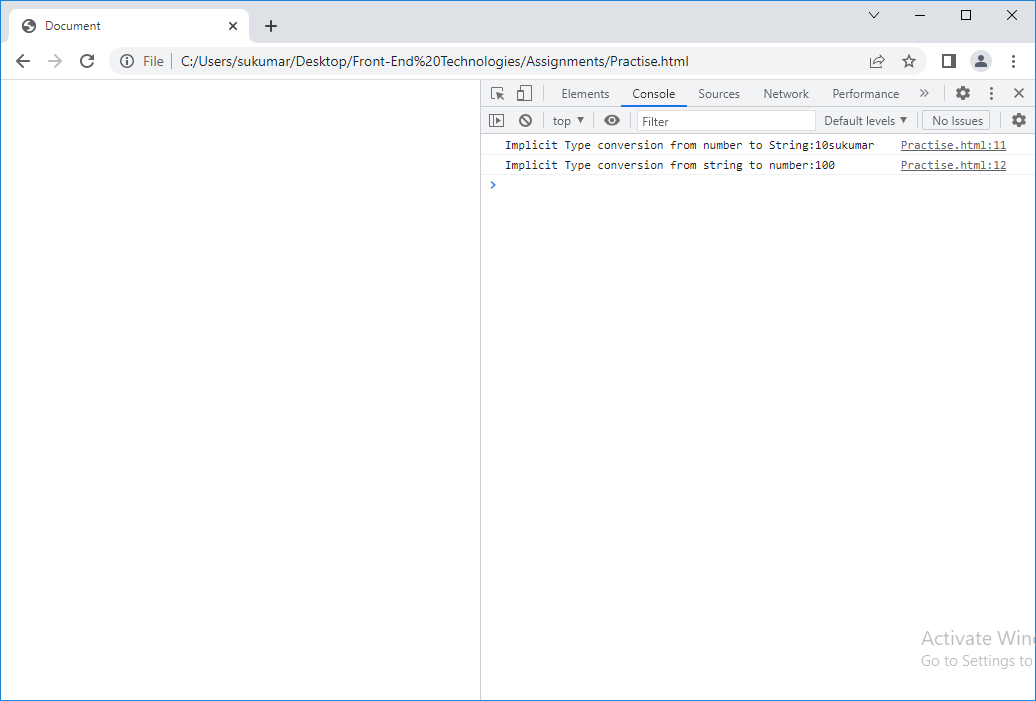
    varb='sukumar';

    console.log('Implicit Type conversion from number to String:'+a+b);

    console.log('Implicit Type conversion from string to number:'+a\*'10');

</script>

</html>



**10.1.1)Dynamic Type Casting**:- The data type of Variable is automatically converted based on assigned value dynamically. This type of conversion is called dynamic type casting.

Example:

<!DOCTYPEhtml>

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <title>Document</title>

</head>

<body>

 <script>

    vara=10;

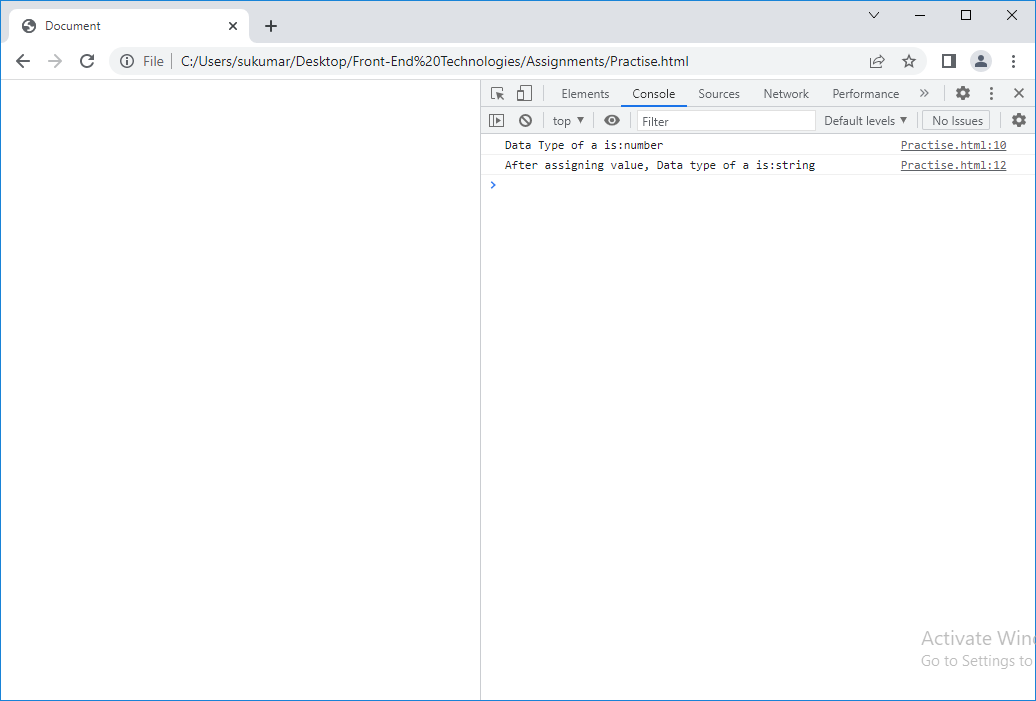
    console.log('Data Type of a is:'+typeof(a)) ;

    a='sukumar';

    console.log('After assigning value, Data type of a is:'+typeof(a));

</script>

</html>



**10.2. Explicit Conversion**:- The programmer writes a statement to convert the one data type into another type. In Js, explicit conversion is done by built-In methods.

10.2.1.convert to Number Explictly:-To convert numeric strings and boolean values to numbers, you can use Number().

let result;

// string to number

result = Number('324');

console.log(result); // 324

result = Number('324e-1')

console.log(result); // 32.4

// boolean to number

result = Number(true);

console.log(result); // 1

result = Number(false);

console.log(result); // 0

10.2.2. convert to String Explictly:-To convert other data types to strings, you can use either String() or toString().

//number to string

let result;

result = String(324);

console.log(result); // "324"

result = String(2 + 4);

console.log(result); // "6"

//other data types to string

result = String(null);

console.log(result); // "null"

result = String(undefined);

console.log(result); // "undefined"

result = String(NaN);

console.log(result); // "NaN"

result = String(true);

console.log(result); // "true"

result = String(false);

console.log(result); // "false"

// using toString()

result = (324).toString();

console.log(result); // "324"

result = true.toString();

console.log(result); // "true"

10.2.3. convert to Boolean explicitly:-To convert other data types to a boolean, you can use Boolean().

let result;

result = Boolean('');

console.log(result); // false

result = Boolean(0);

console.log(result); // false

result = Boolean(undefined);

console.log(result); // false

result = Boolean(null);

console.log(result); // false

result = Boolean(NaN);

console.log(result); // false

11. **Pop-Up Boxes**:-Java script has 3 kinds of dialogue boxes to request a response from user.

* AlertBox
* confirmbox
* promptbox.

11.1)Alertbox:- alertbox is used when a warning messages is needed to be produced. When alertbox is displayed, the user will have to click “ok” to proceed. Alert box returns true, when user clicks ok button .

Syntax:- Alert(“variable/string”);

Example:- alert() can be used to alert the user if a wrong input is given in HTML form element.

4.2)confirm box:- It is used to get authorization or permission from user. When confirm box is displayed, the user will have to click ‘ok’ or ‘cancel’ button. If user clicks “ok” box returns true; if user clicks “cancel” box returns false.

Syntax:-window. Confirm (variable/string);

4.3)prompt Box:- It is often use, if you want user to input value. When prompt box is displayed, the user clicks ‘ok’ or cancel button to proceed after enter value. If user click “ok” the box returns input value. The input value data type is ‘string’. If user click cancel, it returns null.

Syntax:-var varname=[window.] prompt(“sometext”, ”default value”);

Example:1

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

         vara,b,c;

         a=window.prompt('Enter the first Value:');

         b=window.prompt('Enter the Second Value');

         a=Number(a);

         b=Number(b);

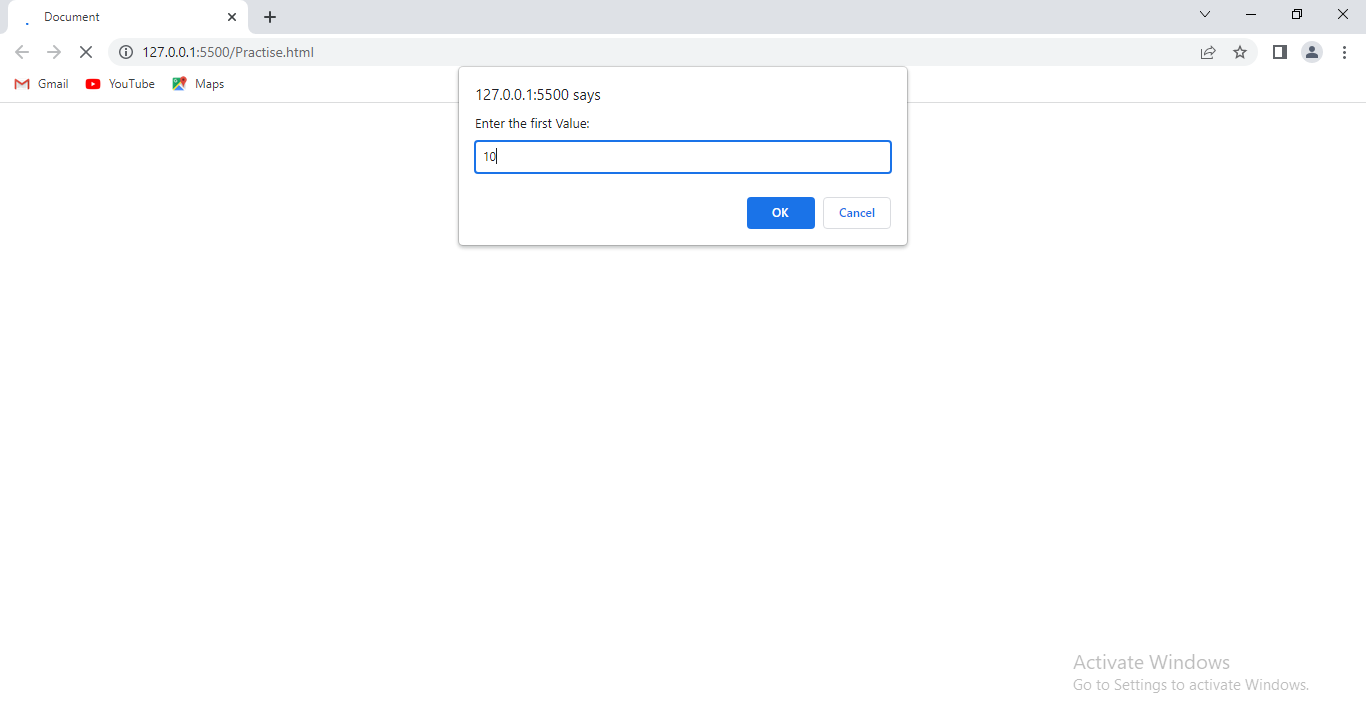
         c=a + b;

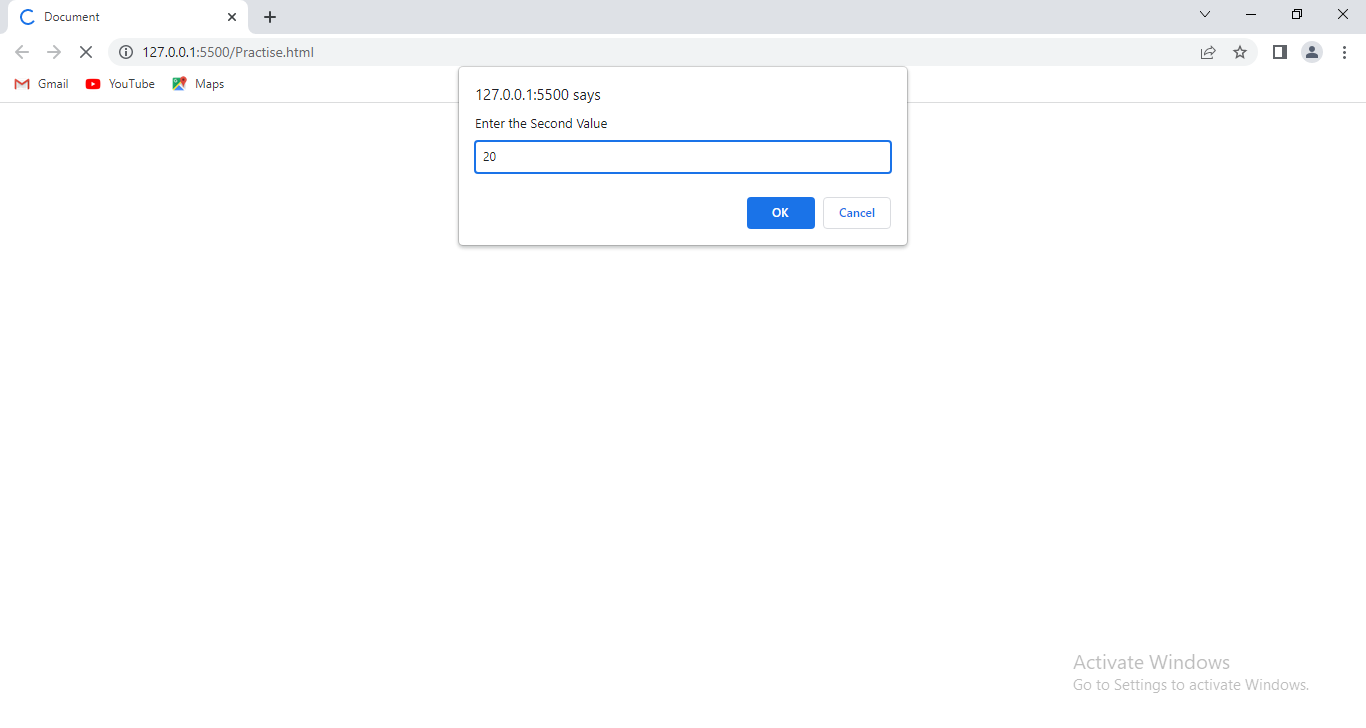
         console.log('Sum of Two numbers is:'+ c);

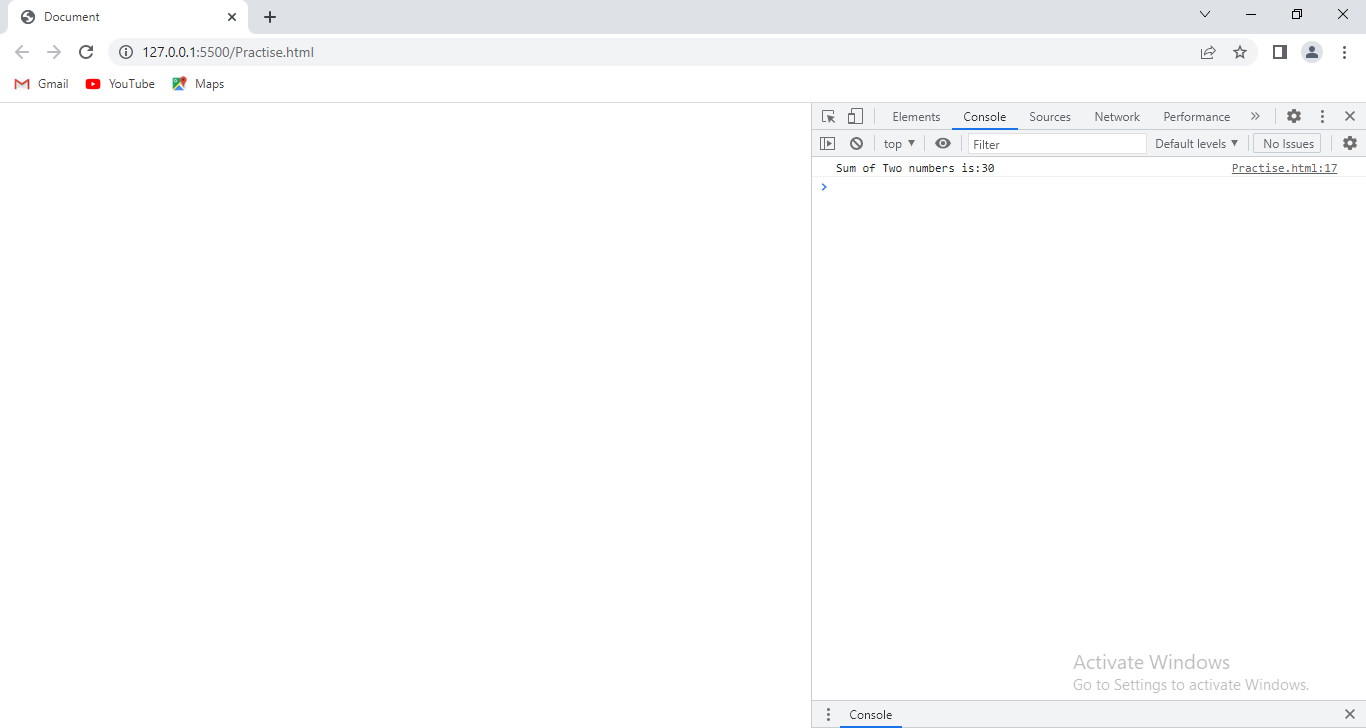
    </script>

</body>

</html>







Example:2 Write a Java script to calculate total salary of employee. The program should collects all emp details(emp name,department,gender,basic salary,pf and hra) at run time.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varename,dept,gender,bs,hra,pf,ts;

        ename=prompt('Enter Emp name:');

        dept=prompt('enter the department:');

        gender=prompt('Enter Gender:');

        bs=Number(prompt('Enter the Basic Salary:'));

        hra=Number(prompt('Enter the % of hra:'));

        pf=Number(prompt('Enter the % of pf:'));

        hra=bs\*hra/100;

        pf=bs\*pf/100;

        ts=bs+pf+hra;

        console.log('Emp Name:'+ename);

        console.log('Department:'+dept);

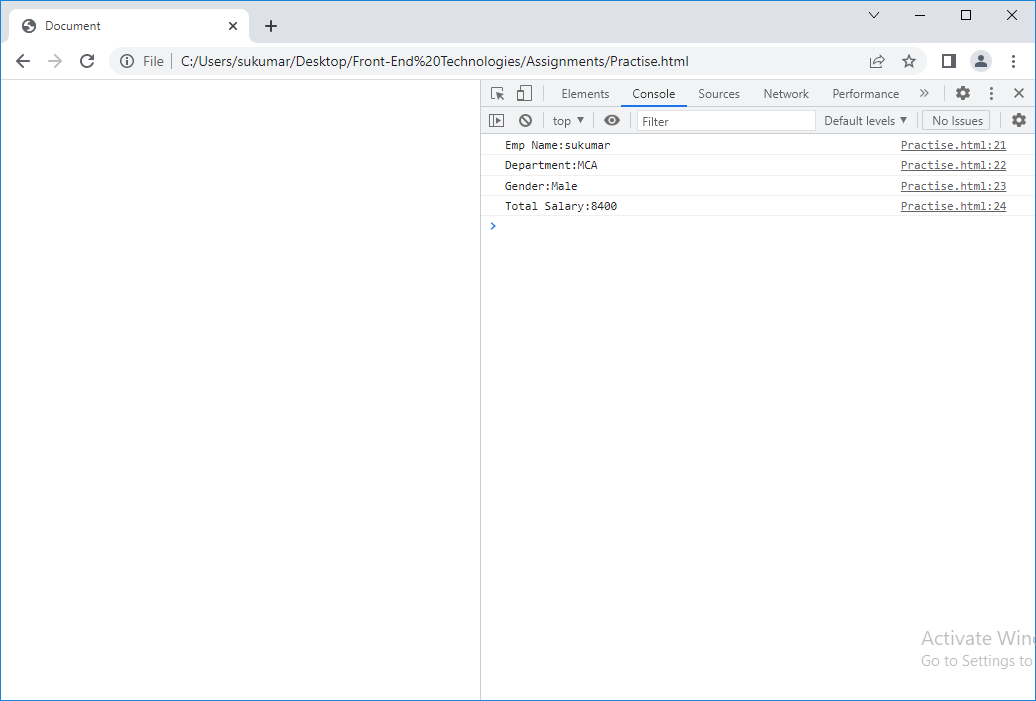
        console.log('Gender:'+gender);

        console.log('Total Salary:'+ts);

    </script>

</body>

</html>

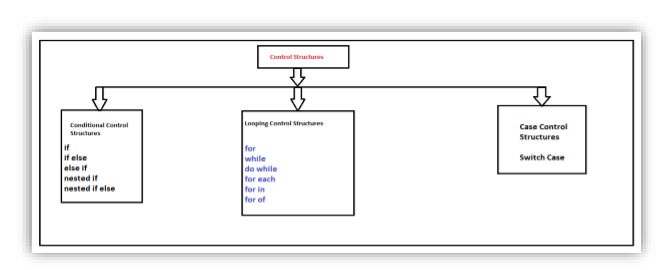


**12.Control Statements**:-Following are the set of Control Structures being supported through which we could able to control the sequence of execution flow in a application.

1. Conditional Control Structures.

2. Looping Control Structures/Iterative Control statements.

3. Case Control Structures.



**12.1) Conditional Control Structures:**

1.If:-

Syntax:

if (condition)

{

// block of code to be executed if the condition is true

}

In the preceding syntax, if the JavaScript keyword that signifies an if statement and contains a condition, that needs to be evaluated to true, then the script statement, represented by statement 1, enclosed within the curly braces, is executed. If the condition evaluates to false, then the statement enclosed within the curly braces is skipped and the statement immediately after closing curly brace (}) is executed.

2. If Else:-The if statement allows us to execute a set of statements only when a particular condition is true. However, if we want to execute another set of statements when the condition is false, then we can use the if....else statement.

Syntax:

if (condition)

{ // block of code to be executed if the condition is true

}

else

{ // block of code to be executed if the condition is false

}

3. Else If :

Syntax:

if (condition1)

{ // block of code to be executed if the condition1 is true. }

else if (condition2)

{ // block of code to be executed if the condition1 is false and condition 2 is true.

}

Else

{ // block of code to be executed if the condition1 is false and condition 2 is False.

}

4. Nested If : when an if statement is present inside the body of another if then this is called nested if.

Syntax:

If(condition 1)

{

Block of code to be executed if condition 1 is true.

If ( condition 2)

{

Block of code to be executed if condition 2 is true.

}

}

5. Nested If Else:- when an if else statement is present inside the body of another if , else or both then this is called nested if else.

Syntax:

If(condition-1)

{

Block of code to be executed if condition 1 is true.

}

Else

{

Block of code tobe executed if condition 1 is false.

If(condition-2)

{

Block of code to be executed if condition2 is true.

}

Else

{

Block of code to be executed if condition2 is false.

}

}

a.Write a java script program to take 2 numbers from user and print largest number.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter First Value:'));

        varb=Number(prompt('Enter Second Value:'));

        if(a!=b)

        {

        if (a>b)

        {

            console.log('Big Number is:'+a);

        }

        else

        {

            console.log('Big Number is:'+b);

        }

        }

        else

        {

            console.log('Given two numbers are equal');

        }

    </script>

</body>

</html>

b.Write a java script program to get a number from user and print whether it is positive or negative.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter a Value:'));

        if(a<0)

        {

            console.log('Given number is negative');

        }

        elseif(a>0)

        {

            console.log('Given number is positive');

        }

        else

        {

            console.log('Given number is zero');

        }

    </script>

</body>

</html>

**c**.Write a java script program to take 3 numbers from user and print largest number.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter First Value:'));

        varb=Number(prompt('Enter Second Value:'));

        varc=Number(prompt('Enter Third Value:'));

        if(a==b)

        {

            if(b==c)

            {

                console.log(' Three values are equal');

            }

        }

        else

        {

            if(a>b)

            {

                if(a>c)

                {

                    console.log('Largest Number is:'+a);

                }

           }

           else

           {

                if(b>c)

                {

                    console.log('Largest Number is:'+b);

                }

                else

                {

                    console.log('Largest Number is:'+c);

                }

           }

        }

    </script>

</body>

</html>

**d**. Write a Java script to find a number is present in given range.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter First Value:'));

        if(a>0&&a<100000)

        {

            console.log('Given Value is in a Range');

        }

        else

        {

            console.log('Given value is not in range');

        }

    </script>

</body>

</html>

e. Write a Java Script to check if a number is odd or even in java script.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter First Value:'));

        varb=a%2;

        if(b==0)

        {

            console.log('Given Number is Even');

        }

        else

        {

            console.log('Given Number is odd');

        }

    </script>

</body>

</html>

f) Write a Java Script to find check if a year is a leap year or not.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter Year:'));

        if(((a%4==0)&&(a%100!=0))||(a%400==0))

        {

          console.log('Given Year is a Leap Year');

        }

        else

        {

            console.log('Given year is not a Leap Year');

        }

    </script>

</body>

</html>

g)Find the no.of Days in a given month.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter Number of month:'));

        if(a==2)

        {

            varb=Number(prompt('Enter the Year:'));

        }

        if(a<0||a>12)

        {

            console.log('Invalid Month');

        }

        elseif(a==2)

        {

            if((a%4==0&&a%100!= 0)||(a%400==0))

            {

                console.log('No.of Days:28');

            }

            else

            {

                console.log('No.of Days:29');

            }

        }

        elseif(a==4 || a==6 || a==9 || a==11)

        {

            console.log('No.of Days:30');

        }

        else

        {

            console.log('No.of Days:31');

        }

    </script>

</body>

</html>

h) Write a java script to print grade of a student.

If marks is greater than 90, grade is A.

If marks is greater than 80 and less than or equal to 90, grade is B.

If marks is greater than 70 and less than or equal to 80 ,grade is C.

If marks is greater than 34 and less than or equal to 70, grade is D.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter the Marks of Student:'));

        if( a>90)

        {

            console.log('Grade-A');

        }

        elseif(a>80&&a<=90)

        {

            console.log('Grade-B');

        }

        elseif(a>70&&a<=80)

        {

            console.log('Grade-c');

        }

        elseif(a>34&&a<=70)

        {

            console.log('Grade-D')

        }

        else

        {

            console.log('Failed');

        }

    </script>

</body>

</html>

i)Write a JS which take values of length and breadth from user and check if it is square or not.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter the Length:'));

        varb=Number(prompt('Enter the Breadth:'));

        if(a==b)

        {

            console.log('Shape is Square');

        }

        else

        {

            console.log('Shape is Rectangle');

        }

    </script>

</body>

</html>

j) A shop will give discount of 10% if the cost of purchased cost is more than 10000. Ask user for purchased cost. Print the total cost for user.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter the Purchased Cost:'));

        if(a<10000)

        {

            console.log('Total Bill:'+a);

        }

        else

        {

            a=a-a\*10/100;

            console.log('Total Bill:'+ a);

        }

    </script>

</body>

</html>

k) A company decided to give bonus of 5% to employee if his/her service is more than 5 years. Ask user for their salary and service and print the net bonus amount.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vara=Number(prompt('Enter the Salary:'));

        varb=Number(prompt('Enter the Service:'));

        if(b<=5)

        {

            console.log('You Have less than 5 years Experience.So Bonus is :0');

        }

        else

        {

            b=a\*5/100;

            console.log('Bonus is :'+b);

        }

    </script>

</body>

</html>

**12.2. Iterative control structures/Iterative control statements**:-Looping Control Structures been supported in Java Script takes the set of instructions and repeat them for multiple times.

**1.For**:-The general format of the for statement is

**Syntax:**

for (initialization; loop continuation test; increment/decrement)

{

Set of statements to be executed;

}

In following order, the for statement is executed.

1. Control variable is initialized. It happens only once.
2. Loop continuation test is executed.
3. Set of statements are executed.
4. Control variable is either incremented or decremented.

**2. While:-** The genera format of the while statement is:

**Syntax:**

initialization;

While ( loop continuation test)

{

Statement;

Increment/Decrement;

}

**3.Do while :-** The do……while repetition statement is similar to the while statement. In the while statement, the loop continuation test occurs at the beginning of the loop, before the body of the loop executes. The do…while statement tests the loop-continuation condition after the loop body executes; therefore , the loop body always executes at least once.

**Syntax:**

initialization;

do

{

Statement;

Increment/Decrement;

} while ( condition);

When a do…while terminates, execution continues with the statement after the while clause. Note that it is not necessary to use braces in a do…while statement if there is only one statement in the body.

1. Write a java script to print the numbers 1 to 5
2. <!DOCTYPEhtml>
3. <htmllang="en">
4. <head>
5. <metacharset="UTF-8">
6. <metahttp-equiv="X-UA-Compatible" content="IE=edge">
7. <metaname="viewport" content="width=device-width, initial-scale=1.0">
8. <title>Document</title>
9. </head>
10. <body>
11. <script>
12. vari;
13. for(i=1;i<6;i++)
14. {
15. console.log(i+'');
16. }
17. </script>
18. </body>
19. </html>

**2. Write**  Java script to print even numbers between the 1 and 10.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vari;

        console.log('Even Numbers:');

        for(i=1;i<10;i++)

        {

            if(i%2==0)

            {

                console.log(i);

            }

        }

    </script>

</body>

</html>

3. Write a JS code to print a pattern using for loop.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vari,j='\*';

        console.log('Even Numbers:');

        for(i=1;i<10;i++)

        {

           console.log(j);

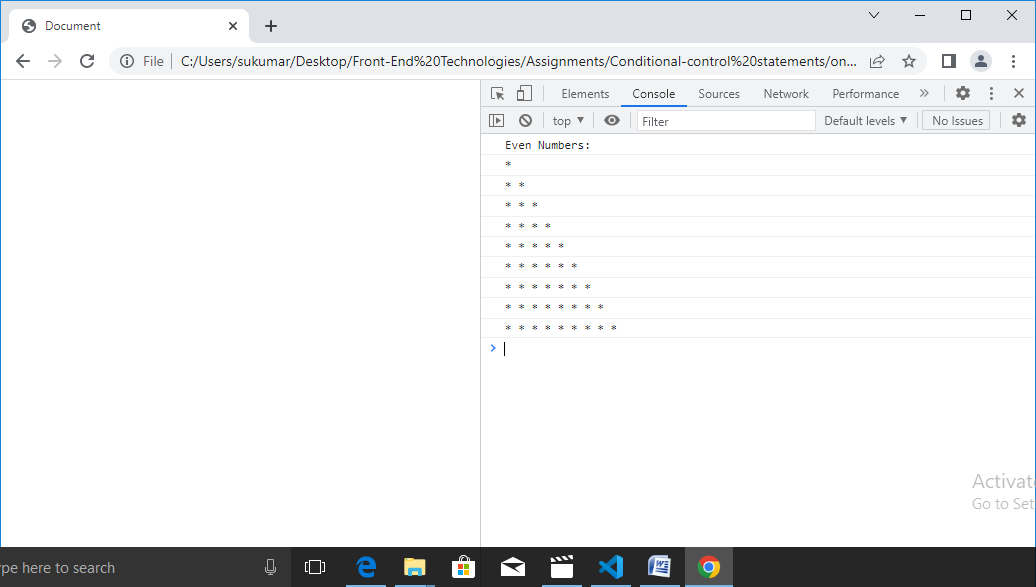
           j= j+''+'\*';

        }

    </script>

</body>

</html>



4. Write a js code to find sum of first five numbers.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        vari=1;

        varsum=0;

        while(i<6)

        {

            sum=sum+i;

            i=i+1;

        }

        console.log('SUM of First 5 Numbers is:'+sum);

    </script>

</body>

</html>

5. Write a JS to find given number is prime or not.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varn = Number(prompt('Enter a Number'));

        vari = 2;

        count = 0;

        while(i<n-1)

        {

            if( n % i == 0)

            {

                count = count + 1;

            }

            i = i + 1;

        }

        if (count == 0)

        {

            console.log('Prime');

        }

        else

        {

            console.log('Not Prime');

        }

    </script>

</body>

</html>

6.Write a JS to find given number is Armstrong or not.

Example:

a.153=13+53+33=153.//Armstrong.

b. 23=22+32=13 // not Armstrong.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varn = Number(prompt('Enter a Number:'));

        vardigits = 0,dup,r = 0,sum = 0 ,i = 0;

        dup = n;

        while(dup != 0)

        {

            digits = digits + 1;

            dup = dup / 10 ;

            dup = Math.floor(dup);

        }

        dup=n;

        while(i<digits)

        {

          r = dup % 10;

          sum = sum + Math.pow(r,digits);

          dup = Math.floor(dup / 10);

          i = i + 1;

        }

        if (sum == n)

        {

            console.log('Armstrong');

        }

        else

        {

            console.log('Not Armstrong');

        }

    </script>

</body>

</html>

7.Write a Js to reverse the Given Number.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varn = Number(prompt('Enter a Number:'));

        varsum = 0, r = 0;

        while(n != 0)

        {

            r = n % 10;

            sum = sum \* 10 +  r;

            n = Math.floor(n / 10);

        }

        console.log('Reverse Number is :'+ sum);

    </script>

</body>

</html>

8. Write Js to find given number is palindrome or not.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varn = Number(prompt('Enter a Number:'));

        varsum = 0, r = 0 , dup ;

        dup = n;

        while(n != 0)

        {

            r = n % 10;

            sum = sum \* 10 +  r;

            n = Math.floor(n / 10);

        }

        if(dup==sum)

        {

            console.log('Given Number is Palindrome');

        }

        else

        {

            console.log('Given number is not palindrome');

        }

    </script>

</body>

</html>

9. Write a Js to display the factorial of given number.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varn = Number(prompt('Enter a Number:'));

        varsum = 1;

        while(n != 0)

        {

          sum = sum \* n;

          n = n - 1;

        }

        console.log('Factorial of Given Number is :'+ sum);

    </script>

</body>

</html>

**12.3.Switch Case:**It is multi-way Branch.

Syntax:

switch (<value>)

{ Case <value>/exp : //stmts

Break;

Case <value>/exp : //stmts

Break;

---

---

Default: //stmts

}

The switch, case, and break are JavaScript keywords. The switch keyword indicates the switch statement. In a switch statement, the expression that is to be evaluated is specified within parentheses. If any of the case values match the value of the expression, the group of statements (statement 1, statement 2 or statement 3) specified in the respective case statement is executed.

If none of the case values matches the value of the expression, then the default statement, specified by the default keyword, is executed. The default statement is generally placed at the end of the switch statement; however, we can place it anywhere within the switch statement.

Example:1 Write a JS to print day of week name using switch case.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

    varn = Number(prompt('Enter A number'));

    switch (n)

    {

        case (0):

            console.log('Sunday');

            break;

        case (1):

            console.log('Monday');

            break;

        case (2):

            console.log('Tuesday');

            break;

        case (3):

            console.log('Wednesday');

            break;

        case (4):

            console.log('Thursday');

            break;

        case (5):

            console.log('Friday');

            break;

        case (6):

            console.log('Saturday');

            break;

        default:

            console.log('Rong Number');

    }

</script>

</body>

</html>

Example:2Write a JS to check whether number is positive,negative or zero using swith case.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

    varn = Number(prompt('Enter A number'));

    switch (true)

    {

        casen>0:

            console.log('Positive Number');

            break;

        casen<0:

            console.log('Negative Number');

            break;

        casen==0:

            console.log('Zero');

    }

</script>

</body>

</html>

Example:3 write a Js to print total number of days in a month using switch case.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

    varn = Number(prompt('Enter A number'));

    switch (n)

    {

        case1:

            console.log('Number of Days:31');

            break;

        case2:

            console.log('Number of Days:28 or 29');

            break;

        case3:

            console.log('Number of Days:31');

            break;

        case4:

            console.log('Number of Days:30');

            break;

        case5:

            console.log('Number of Days:31');

            break;

        case6:

            console.log('Number of Days:30');

            break;

        case7:

            console.log('Number of Days:31');

            break;

        case8:

            console.log('Number of Days:31');

            break;

        case9:

            console.log('Number of Days:30');

            break;

        case10:

            console.log('Number of Days:31');

            break;

        case11:

            console.log('Number of Days:30');

            break;

        case12:

            console.log('Number of Days:31');

            break;

        default :

            console.log('Wrong Input') ;

    }

</script>

</body>

</html>

**13. Function:**The function is a set of statements. The statements are binded as module with in “{}”. All statements together performs specific task.

* In a single page we can define any number of functions in JavaScript.
* Once the function been defined it can be invoked any number of times through the function name.
* The function can be invoked from anywhere in program.
* While defining the function we name it through function keyword, while calling we simply use function name.

**Function Definition/Declaration**:

Syntax:

function function-name(parameter[s])

{

Body

[return statement;]

}

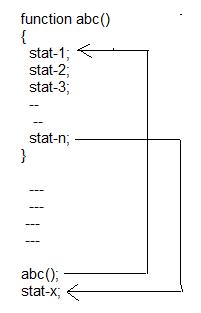
**Function Invoking**:

Syntax:

[Var-name=]function-name(parameter[s]);

The function defintion is not executed immediately. It is executed,when it is invoked.

Note:- when function is invoked, the execution control immediately jumps from calling statement to called statement. After executing function, execution control come back to statement which is next to calling statement.



**Advantages By Function**:

1. We can avoid the rewriting same logic again and again in program.
2. We can achieve the modularity.
3. Reusability of code.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     varename,dept,bs,hra,da,ta,ts;

     functionreadEmpBasicData()

     {

        ename=prompt('Enter the ename');

        dept=prompt('Enter dept:');

        readEmpSalData();

     }

     functionreadEmpSalData()

     {

        bs =Number(prompt('Enter the Basic Salary:'));

        hra =Number(prompt('Enter the HRA:'));

        da=Number(prompt('Enter the Da:'));

        ta=Number(prompt('Enter the travelling allowence:'));

        calEmpSalary();

     }

     functioncalEmpSalary()

     {

        ts=bs+hra+da+ta;

        console.clear();

        console.log('Total Salary:',ts);

     }

     readEmpBasicData();

    </script>

</body>

</html>

**13.1.Local Variable And Global Variable:**The scope is region of program in which variable is accessed. Java script variable have always two scopes.

1. Global scope

2. Local scope.

1. Global Scope: A global variable has global scope which means it can be defined or accessed anywhere in your java script code. The lifetime of global variable starts when it is declared. The global variable is deleted when browser is closed. The global variable always has **public data**.

2.Local Scope:- Local variables are always visible with in function/block where it is defined. The life time of local variable starts when it is declared. The local variable is deleted when function completed. The local variable always has **private data**.

Note:- If local variable & global variable with same name, with in function local variable hides global variable (i.e) local variable has high precedence than global variable.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     functionreadEmpBasicData()

     {

        varename,dept;

        ename=prompt('Enter the ename');

        dept=prompt('Enter dept:');

        readEmpSalData();

     }

     functionreadEmpSalData()

     {

        varbs,hra,da,ta;

        bs =Number(prompt('Enter the Basic Salary:'));

        hra =Number(prompt('Enter the HRA:'));

        da=Number(prompt('Enter the Da:'));

        ta=Number(prompt('Enter the travelling allowence:'));

        calEmpSalary();

     }

     functioncalEmpSalary()

     {

        varts;

        ts=bs+hra+da+ta;

        console.clear();

        console.log('Total Salary:',ts);

     }

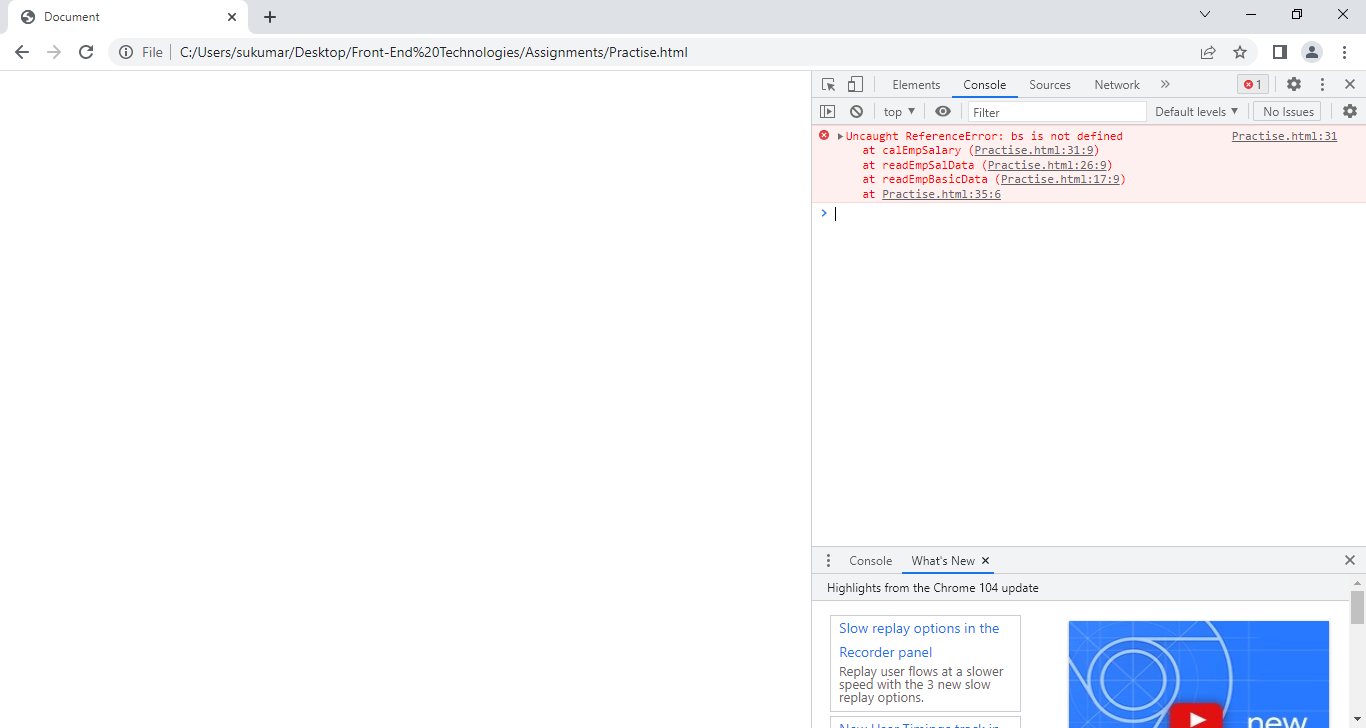
     readEmpBasicData();

    </script>

</body>

</html>

Output:-



**13.2.Accessing private data of a function outside of it:**

• Through passing parameters while calling a function.

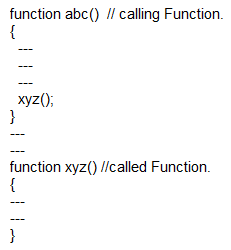
• Returning value from called function to calling function.

• Both passing parameters and returning values.

1.Calling Function: The function which calls another function is said to be calling function.

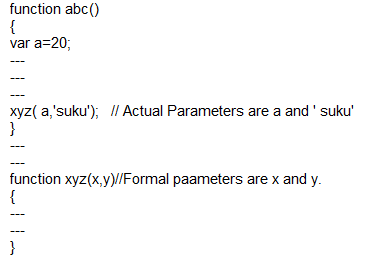
2.Called Function: The function which is called by another function is said to be called function.

Example:



3. Parameter: Arguments or parameters are the means to pass values from the calling function to the called function. The variables used in the function definition as parameters are known as formal parameters. The constant, variable or expression used in the calling function are known as actual parameter or actual argument.

Example:



13.2.1) Passing Parameter:- The calling function send its private data to calling function by passing parameter mechanism.

Parameter Rules:-

* JS function does not perform type checking on passed Arguments.
* JS function checks no.of arguments.
* If function is called with missing arguments(less than declared missing values are set to undefined).
* JSfunction is called with too many arguments, these arguments can ‘t be referenced because they don’t have name, They can only be in ‘Argument’ Object.

Example:1 Write a js to calculate total salary of employee.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     functionreadEmpBasicData()

     {

        varename,dept,bs,hra,da,ta;

        ename=prompt('Enter the ename');

        dept=prompt('Enter dept:');

        bs =Number(prompt('Enter the Basic Salary:'));

        hra =Number(prompt('Enter the HRA:'));

        da=Number(prompt('Enter the Da:'));

        ta=Number(prompt('Enter the travelling allowence:'));

        calEmpSalary(bs,hra,da,ta);

     }

     functioncalEmpSalary(bs,hra,da,ta)

     {

        varts;

        ts=bs+hra+da+ta;

        console.log('Total Salary:',ts);

     }

     readEmpBasicData();

    </script>

</body>

</html>

13.2.2. Returning Value from called function to calling function:- The calling function gets private data of called function by **return** statement.

(or)

The called function send its private data to calling function by **return** statement.

Syntax:

Return [v1,v2, v3];

Return statement carries only one value from called function to calling function. If more than one value is written after the return keyword, then last value is returned and remaining values are ignored.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     functionreadEmpBasicData()

     {

        varename,dept,bs,hra,da,ta,ts;

        ename=prompt('Enter the ename');

        dept=prompt('Enter dept:');

        bs =Number(prompt('Enter the Basic Salary:'));

        hra =Number(prompt('Enter the HRA:'));

        da=Number(prompt('Enter the Da:'));

        ta=Number(prompt('Enter the travelling allowence:'));

        ts=calEmpSalary(bs,hra,da,ta);

        console.log('Total Salary:',ts);

     }

     functioncalEmpSalary(bs,hra,da,ta)

     {

        varts;

        ts=bs+hra+da+ta;

        returnts;

     }

     readEmpBasicData();

    </script>

</body>

</html>

Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     functionabc()

     {

      vara;

      a=xyz();

      console.log('Value is:'+a);

     }

     functionxyz()

     {

        return10,20,30;

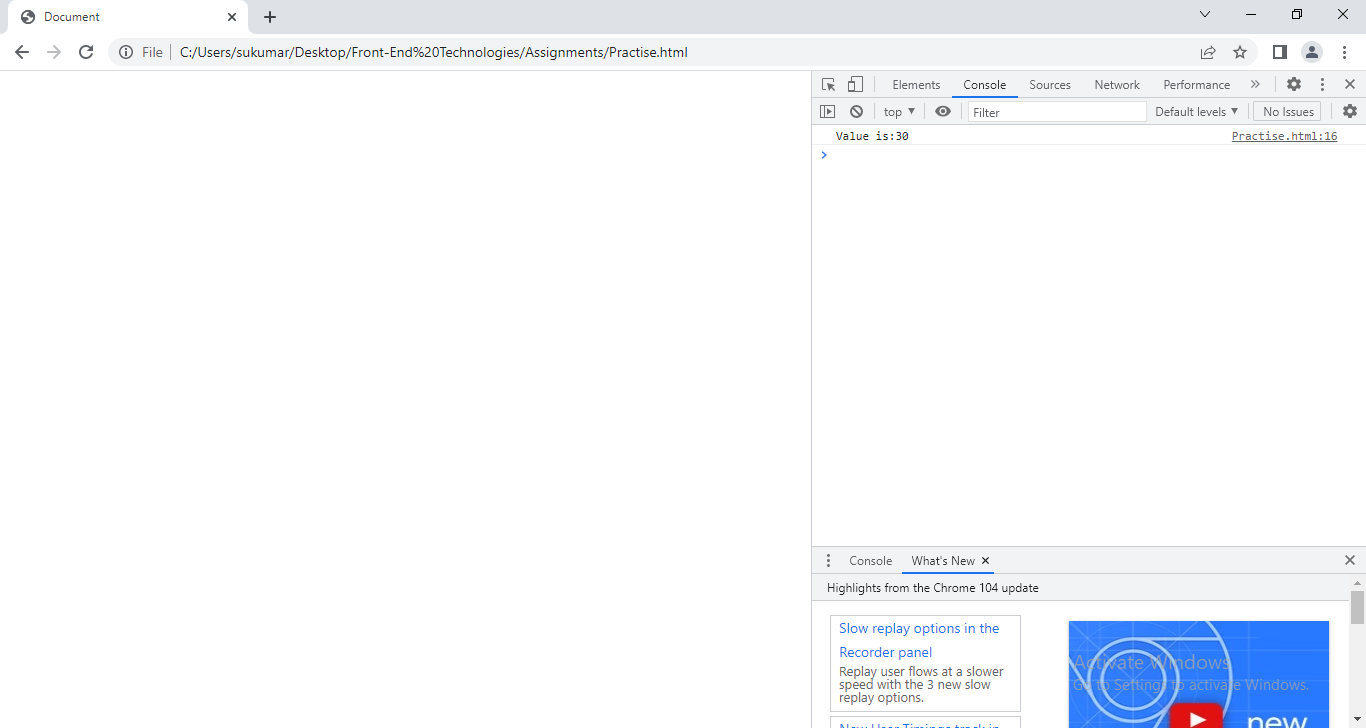
     }

     abc();

    </script>

</body>

</html>



13.3. Argument:- The java script function have built-in object called Argument object.Irrelevant of whether the parameters been catched or not, arguments keyword gets created in every function and holds all the parameters being passed to it in the form of an array.it’s life time is up to that function. It’s scope is also up to that function.

Note:-The Argument array is not created when we call **arrow** function.Instead of Argument Array, The rest parameter has to be used.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

     functionabc()

     {

      xyz(10,20,30,40);

     }

     functionxyz()

     {

        for(vari=0;i<arguments.length;i++)

        {

         console.log(arguments[i]);

        }

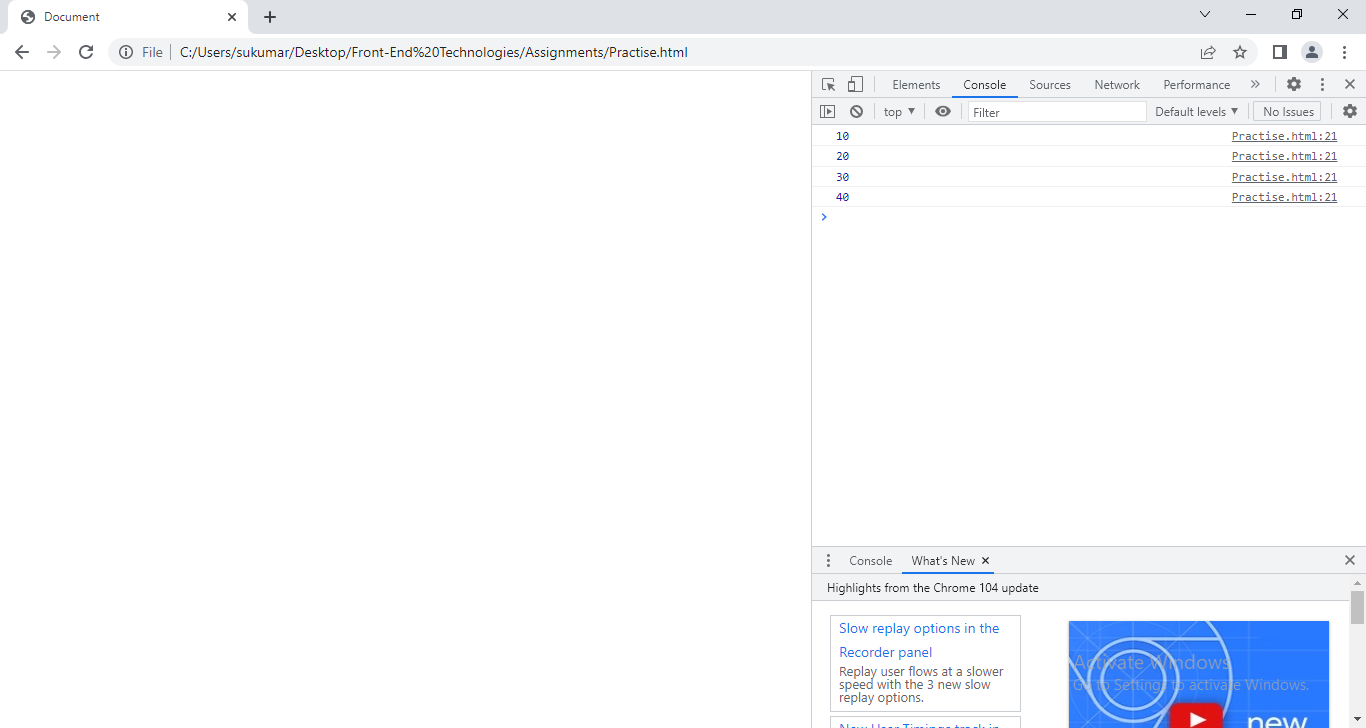
     }

     abc();

    </script>

</body>

</html>



**13.4 Default Value:**- Goto ECMa6.

**13.5. anonymous function:-** If the function does not have name ,then it is called anonymous function.

Declaration/definition:

Syntax:

var function-name=function([parameter[s]]){ body}

Function invocation:

Syntax:-

function-name([argument[s]);

Example: Write a JS to calculate and display the average marks of student using anonymous function.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      varsname,rollNo,m1,m2,m3,avg;

      sname=prompt('Enter the Name:');

      rollNo=prompt('Enter the RollNo:');

      m1=Number(prompt('Enter the m1:'));

      m2=Number(prompt('Enter the m2:'));

      m3=Number(prompt('Enter the m3:'));

      varcalAverage=function(m1,m2,m3)

      {

        returnm1+m2+m3/3;

      }

      ts=calAverage(m1,m2,m3);

      console.log('Average of Student:'+ts);

   </script>

</body>

</html>

**13.6.Arrow function:-** It is also called ‘fat arrow functions’. By using arrow functions, we avoid having to type function keyword,name of function, return keyword and curly braces.

Declaration/Defintion:

Syntax:

Varname=([parameter[s]])=> expression;

(or)

Varname=([parameter[s])=> {body};

Function Invoking:

name([parameter[s]]);

Note:- **The anonymous function and arrow function can be passed as argument from one function to another one.**

Example:Example: Write a JS to calculate and display the average marks of student using arrow function.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      varsname,rollNo,m1,m2,m3,avg;

      sname=prompt('Enter the Name:');

      rollNo=prompt('Enter the RollNo:');

      m1=Number(prompt('Enter the m1:'));

      m2=Number(prompt('Enter the m2:'));

      m3=Number(prompt('Enter the m3:'));

      varcalAverage=(m1,m2,m3)=>

      {

        returnm1+m2+m3/3;

      }

      ts=calAverage(m1,m2,m3);

      console.log('Average of Student:'+ts);

   </script>

</body>

</html>

**13.7.Function Overloading:**When a function name is overloaded with different jobs it is called Function Overloading. In Function Overloading “Function” name should be the same and the arguments should be different(no.of arguments or data type of arguments.

Example:

function add(){}

function add(int a){}

function add(int a, int y){}

function add(float x){}

when overloaded function is invoked, the compiler differentiates among the functions based on no.of arguments or type of arguments.

The Java script does not support the function overloading Because It is untyped language or loosely typed programming language.

Example: Demonstrate what happens when function overload in JS.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=function()

      {

        console.log('This is function');

      }

      vara=function(a)

      {

        console.log('This is second Function');

      }

      vara=function(a,b)

      {

        console.log('This is third function');

      }

      a();

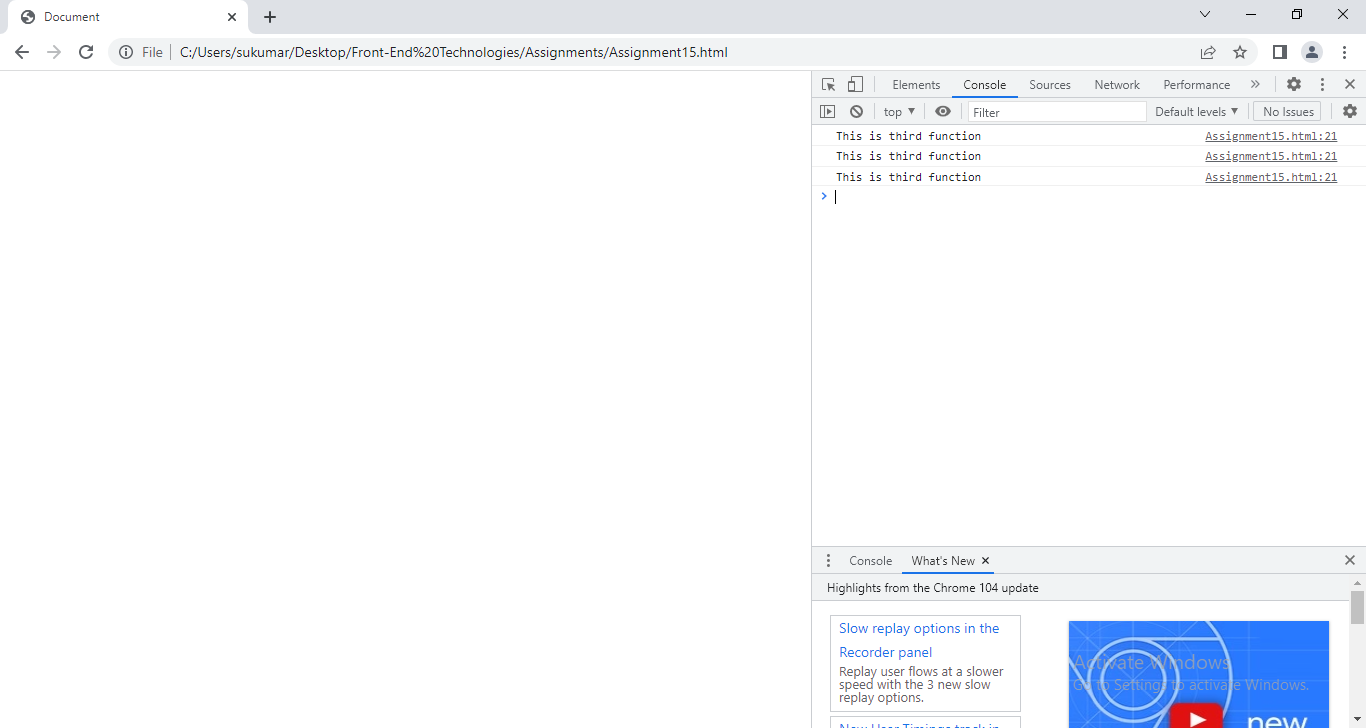
      a(10);

      a(20,30);

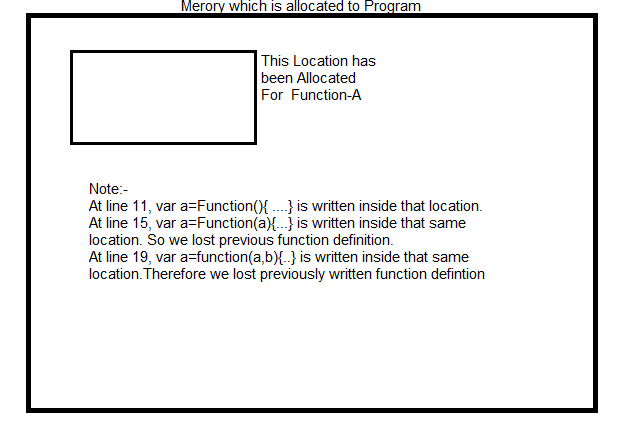
   </script>

</body>

</html>



Explanation: Why did we get same message 3 times?



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=function()

      {

        console.log('This is function');

      }

      a();

      vara=function(a)

      {

        console.log('This is second Function');

      }

      vara=function(a,b)

      {

        console.log('This is third function');

      }

      a();

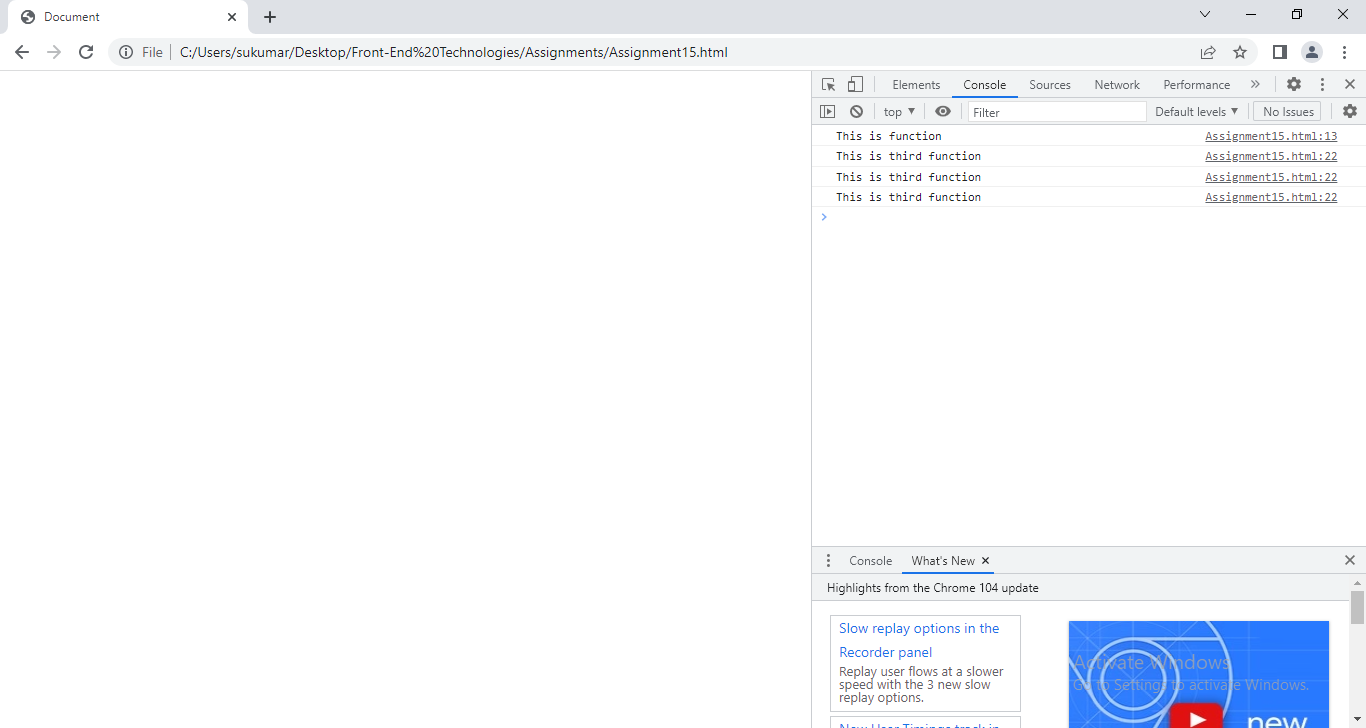
      a(10);

      a(20,30);

   </script>

</body>

</html>



**13.8.IIFE(Immediately Invoking Function Expression):go to Ecma6.**

**14.Data Structures:**Following are the different data structures being supported in the JavaScript through which different type of data can be stored and retrieved.

• Arrays

• Json

• Map (ECMA6)

• Set(ECMA6)

• Tables

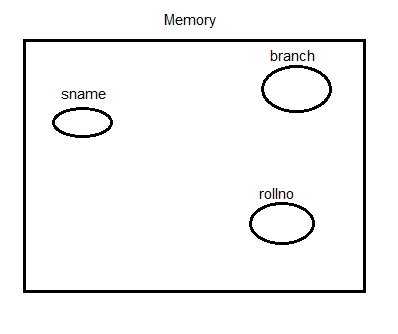
**14.1.Array:-**

Normal variable can only hold single value. It can’t hold the multiple values.

Consider following scenario:

I want to store student name, roll no, branch in memory. What problems do we have to store them?

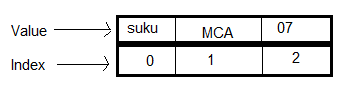
1. We must declare 3 variables in application.
2. We must remember all variable names.
3. There 3 variable values may not be stored in continuous memory locations. Due to in different locations, we need more time to access them one by one.



To overcome such problems, Java Script provided Array.

* The Array is special type variable.Array is capable of holding different and multiple types of data.
* All the values within the array will be automatically assigned with a numeric index value.
* The index value is always starts with “Zero”.
* Within the memory, arrays always occupy continuous memory allocation for all the values within it.

Example:



An array in JavaScript can be created/declared and initialized in two ways.

1. Array literal

2. Array constructor

**1.Array Literal**:

Array Declaration:

Syntax:

Var name=[]// empty array is created.

At the time of Array declaration, we can assign initial value to array by following syntax:

Var name=[v1,v2,….vn]; Array created with specified values.

**2.Array Constructor**:

Array Declaration:

Syntax:

Var name=new Array();// empty array is created.

At the time of Array declaration, we can assign initial value to array by following syntax:

Var name=new Array(v1,v2,….vn); Array created with specified values.

Note :- Array type is ‘object’.

Pre-Defined methods supports in array: Following are the some predefined methods supported by array in JavaScript.

•Length: It is a property which returns the total number of values in an array.

Syntax:

arrayName.length;

•Push(<values>): Used to insert single or multiple values to an array from right direction. It returns length of array.

Syntax:

intarrayName.push(<value[s]>);

• Pop(): Used to delete single value from an array from right side.It returns the last element of array.

Syntax:

arrayName.pop();

• Shift(): Used to delete single value from an array from left direction. It returns the first element of array.

Syntax:

arrayName.shift();

• Unshift(<values>): Used to insert single/multiple values to an array from left direction. It returns length of array.

Syntax:

numberarrayName.unshift(<value[s]>);

• Splice(starting position, <no.of.values to be deleted>,<optional values to be inserted>):

Used to insert or delete values.

Syntax:

arrayName.splice(starting position, <no.of.values to be deleted>,<optional values to be inserted>);

• Join(): Used to merge all the values of array to a single value with/without separator. It returns joined value as string.

Syntax:

String arrayName.join([argument]);

reverse(): Used to reverse the elements of array.

Syntax:

arrayName.reverse();

concat():- It joins two/more arrays as single array.

Syntax:

arrayName.concat(array1,array2,…);

sort():

lastIndexOf():- It returns the last index of occurrence of element.

Syntax:

Number lastIndexOf(element);

indexOf():- It returns the first index of occurrence of element.

Syntax:

Number indexOf(element);

isArray():- It returns the true, if given argument is array otherwise returns false.

Syntax:

Boolean isArray(arg1);

Includes():- it returns the true, if array has given argument otherwise it returns false.

Syntax:

Boolean includes(val);

toString():- It returns the String representation of array.

Syntax:

String toString();

Example:1 Write a program to take array with values and display elements of array.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[2,3,4,5,6,7,8,9,10],i;

      varb=new Array(10,20,30,40);

      console.log('First Array Elements:');

       for(i=0;i<a.length;i++)

      {

        console.log(a[i]);

      }

      console.log('Second Array Elements:');

      for(i=0;i<b.length;i++)

      {

        console.log(b[i]);

      }

 </script>

</body>

</html>

Example:2 Write a program to take array with values and return all the even numbers with in it.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[2,3,4,5,6,7,8,9,10],i=0;

      varevenNumbers=[];

      for(;i<a.length;i++)

      {

        if(a[i]%2==0)

        {

          evenNumbers.push(a[i]);

        }

      }

      console.log('Array:'+a);

      console.log(evenNumbers);

 </script>

</body>

</html>

Example:3 Write a program to find out sum of all the values with in the given array.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3,4,5,6,7,8,9,10],i,sum=0;

      for(i=0;i<a.length;i++)

      {

        sum=sum+a[i];

      }

      console.log(a);

      console.log('Sum of Array elements is:'+ sum);

 </script>

</body>

</html>

Example:4 write a program to find out the sum of all odd numbers with in the given array.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3,4,5,6,7,8,9,10],i,sum=0;

      for(i=0;i<a.length;i++)

      {

        if(a[i]%2 != 0)

        {

        sum=sum+a[i];

        }

      }

      console.log(a);

      console.log('Sum of odd Numbers is:'+ sum);

 </script>

</body>

</html>

Example:5 write program to find out the smallest and largest numbers from the given array.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3,4,5,6,7,8,9,10],i,small=a[0],large=a[0];

      for(i=0;i<a.length;i++)

      {

        if(a[i]<small)

        {

            small=a[i];

        }

        if(a[i]>0)

        {

            large=a[i];

        }

      }

      console.log(a);

      console.log('smallest Number:'+ small);

      console.log('Largest Number is:'+large);

 </script>

</body>

</html>

Example:6 Write a program to find out the total number of values within the array without using length property.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3,4,5],i=0,length=0;

      while(String(a[i])!='undefined')

      {

        length=length+1;

        i=i+1;

      }

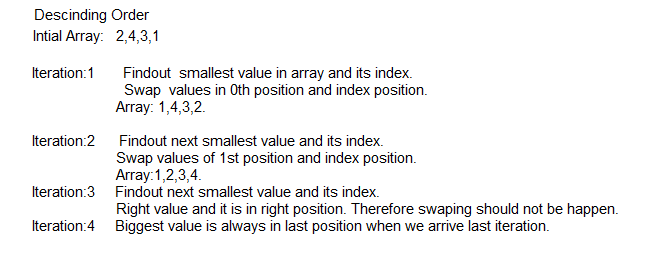
      console.log('Length of Array:'+length);

</script>

</body>

</html>;

Example:7 Write a program to read values dynamically into an array and sort the values to either ascending order.



<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[22,4,3,120];

      varch,i,j,small=0,big=0,index,temp1,count=0;

      ch=Number(prompt('1.Assending'+'\n'+'2.Descing'));

      switch (ch)

      {

        case1:

           for(i=0;i<a.length;i++)

           {

            small=a[i];

            for(j=i+1;j<a.length;j++)

            {

                if(a[j]<small)

                {

                    small=a[j];

                    index=j;

                }

            }

            if(small!=a[i])

            {

            temp1=a[i];

            a[i]=small;

            a[index]=temp1;

            }

           }

           break;

        case2:

        for(i=0;i<a.length-1;i++)

           {

            big=a[i];

            for(j=i+1;j<a.length;j++)

            {

                if(a[j]>big)

                {

                    big=a[j];

                    index=j;

                }

            }

            temp1=a[i];

            a[i]=big;

            a[index]=temp1;

        }

           break;

        default:

            console.log("Wrong Input has been given");

      }

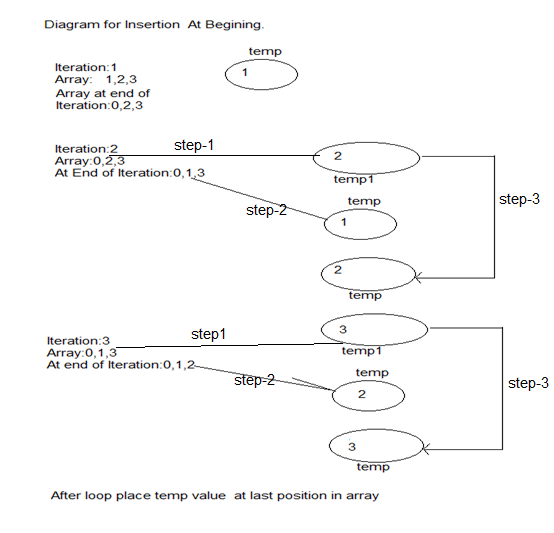
      console.log(a);

</script>

</body>

</html>;

Example:8 Write a program to insert value at given position in Array.



<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3,4,5];

      varposition,element,ch,i,j;

      ch=Number(prompt('1.Begining'+'\n'+'2.Ending'+'\n'+'3.middle'));

      element=Number(prompt('Enter a Number:'));

      switch (ch)

      {

        case1:

            vartemp=0;

            for(i=0;i<a.length;i++)

            {

                if(i==0)

                {

                    temp=a[i];

                    a[i]=element;

                }

                else

                {

                temp1=a[i];

                a[i]=temp;

                temp=temp1;

                }

            }

            a[i]=temp;

            break;

        case2:

            a[a.length]=element;

            break;

        case3:

            position=Number(prompt('Current Array Length is:'+a.length+'\n'+'Enter Elememt position:'));

            vartemp=0;

            if(position!=a.length)

            {

            for(i=position;i<a.length;i++)

            {

                if(i==position)

                {

                    temp=a[i];

                    a[i]=element;

                }

                else

                {

                temp1=a[i];

                a[i]=temp;

                temp=temp1;

                }

            }

            a[i]=temp;

        }

        else

        {

            a[position]=element;

        }

            break;

        default:

            console.log("Wrong Input has been given");

      }

      console.log(a);

</script>

</body>

</html>;

**Array Destructing:-**Assigning the values of array to individual variables is called array destructing.

Syntax :

Var [n1,n2,n3[=defaultvalue],…]=arrayName;

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=[1,2,3];

      var [b,c,d,e=20]=a;

      console.log('Array:'+a);

      console.log(b);

      console.log(c);

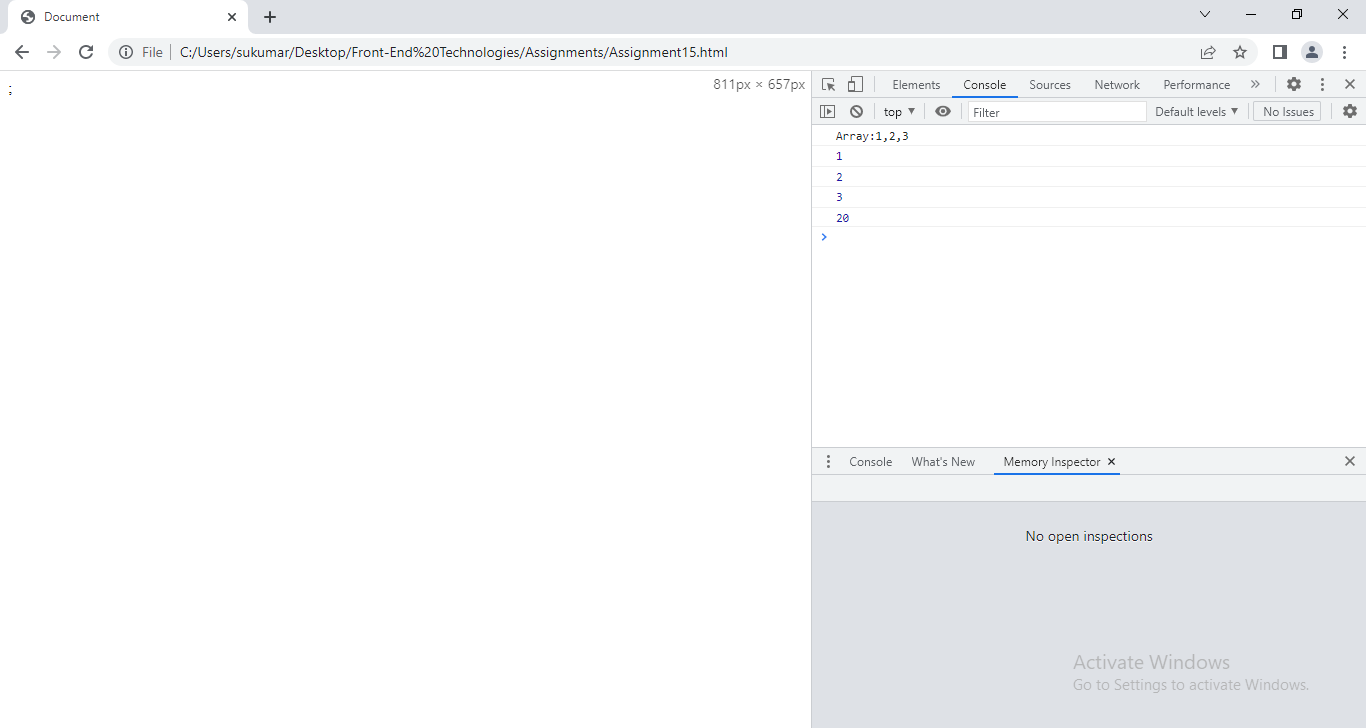
      console.log(d);

      console.log(e);

</script>

</body>

</html>;



**Multi Dimensional Array**:-Multidimensional arrays with two subscripts often are used to represent **tables** of values consisting of information arranged in rows and columns. Arrays that require two subscripts to identify a particular element called **two-dimensional arrays.Javascript does not support multidimensional arrays directly.**

**Example:**

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        varstudents=new Array(2);

        vari,j,stuRec='';

        students[0]=new Array('sukumar',41,'nlr');

        students[1]=new Array('veena',38,'chi');

        for(i=0;i<students.length;i++)

        {

            for(j=0;j<students[i].length;j++)

            {

             stuRec=stuRec+''+students[i][j];

            }

            console.log(stuRec);

            stuRec='';

        }

    </script>

</body>

</html>

**14.2.JSON:-**JavaScript Object Notation (JSON) is a text-based, human-readable interchange format used for representing simple data structures and objects in JS.

Syntax: To create object.

Var var-name={}//create Empty object.

Var var-name=new Object() // creates Empty object.

* The JSON object have multiple key/value pairs.
* The JSON object should be written in between {}.
* The key and value is separated by (:).
* The key/value pairs are separated by (,)
* All the keys within the JSON Object should be only string data type, whereas the corresponding value could be of any JavaScript supported Data Type.
* As per JSON standards, The key should be between “ and “.

Syntax: To create non-Empty object.

Var var-name={“key1”:v1,’key2’:v2,...}

Var var-name=new Object( key/value pairs);

This object contains data in JSON format.

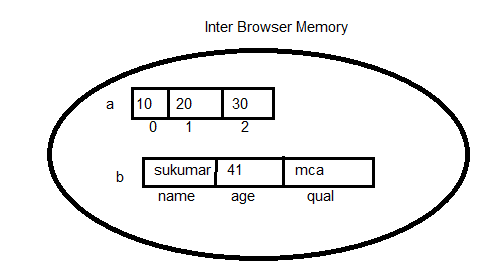
Note: object type is ‘object’.

7.2.1Memory Allocation to JSON object:

Example:

Var a=[10,20,30];

Var b={‘name’:’suku’,’age’:41,’qual’:’mca’};



7.2.2Accessing JSON Data:-

Syntax: 1

Varname.key;

Syntax: 2

Varname[‘key’];

7.2.3 Dynamically Adding key/value pair to Existing JSON object:

Syntax:1

Varname.key=value;

Syntax:2

Varname.‘key’=value;

7.2.4 Deleting key/value pair from JSON object:

Syntax:1

Delete Varname.key

Syntax:2

Delete varname[‘key’];

Example: Write program to demonstrate the JSON Object.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

      varstudent={'name':'sukumar','age':41,marks:[77,85,71],'branch':'mca'};

      vartotal=student['marks'][0]+student.marks[1]+student.marks[2];

      if(total>150)

      {

        pass=true;

      }

      else

      {

        pass=false;

      }

      student['total\_marks']=total;

      student.pass=pass;

      console.log(student);

      console.log('----------------Student Details---------------');

      console.log('Name:'+student['name']);

      console.log('Age:'+student['age']);

      console.log('Marks:'+student['marks']);

      console.log('total\_marks:'+student.total\_marks);

      console.log('Pass:'+student.pass);

      delete student.pass;

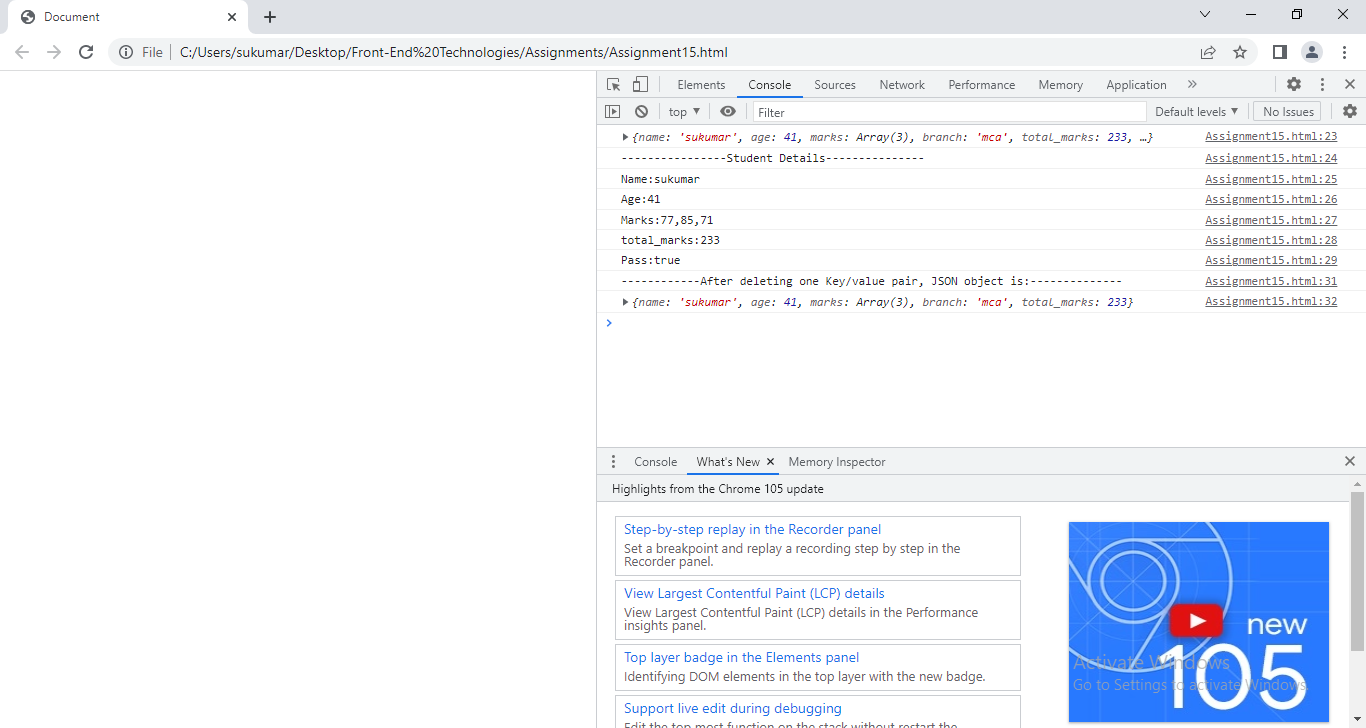
      console.log('------------After deleting one Key/value pair, JSON object is:--------------');

      console.log(student);

        </script>

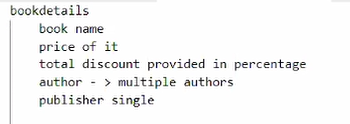
</body>

</html>



14.2.5. Complex JSON object:

1.Create JSON object for following Data.



Var bookDetails={

“bookName”:”DBMS”,

“price”:500,

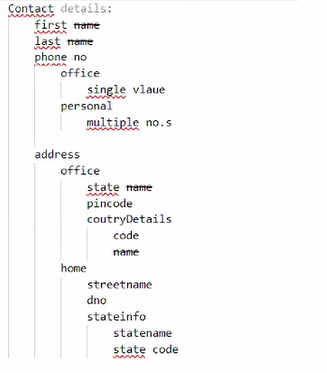
“totalDiscount”: 30,

“author”:[‘korth’,’raghu’,’krishna’],

“publisher”:”Tata MCgrahill”

}

2. Create JSON object for following Data.



Var contacDetails= {

“first”:”sukumar”,

“last”:”Atmakuru”,

“phoneNo”:

{

“office”:9703393965,

“personal”:[1,2,3]

}

“address”:

{“office”:{“state”:”ap”,

”pincode”:524001,

“countryDetails”: {

“code”:”IND”, “name”:”India”

}

}

“home”:{“streetName”:”kamitistreet”,

“dno”:”22-10”,

“stateinfo”:{

“statename”:”aP”,

“stateCode”:001

}

}

}

}

14.2.6. Loops:

1.For in:

Syntax:

For ([var] varname in JSON object name)

{

---

}

Example:1

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varbookDetails={

         "bookName":"DBMS",

         "price":500,

         "discount":30,

         "author":['korth','raghu'],

         "publisher":"Tata MCgrahil"

     }

     for(vari in bookDetails)

     {

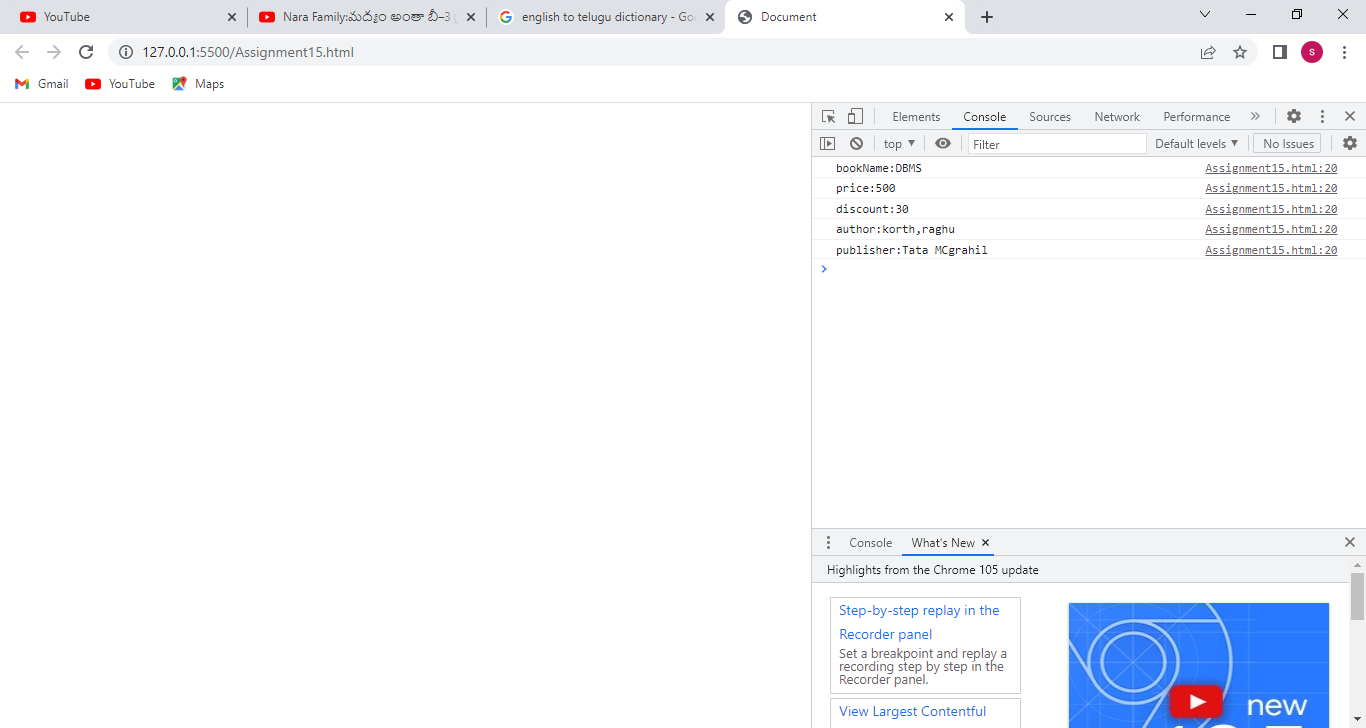
        console.log(i+":"+bookDetails[i]);

     }

    </script>

</body>

</html>



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varcontacDetails= {

                    "first" :'sukumar' ,

                    "last":"Atmakuru",

                    "phoneNo" :

                           {

                            "office":9703393965,

                            "personal":[1,2,3]

                            },

                    "address":

                            {"office":{"state":"ap",

                                                "pincode":524001,

                                                "countryDetails": {

                                                       "code":"IND", "name":"India"

                                                                  }

                                      },

                                        "home":{

                                                    "streetName":"kamitistreet",

                                                    "dno":"22-10",

                                                    "stateinfo":{

                                                              "statename":"aP",

                                                              "stateCode":001

                                                            }

                                                        }

                                        }

                                    }

     functiondisplay(j)

     {

            for(vark in j)

            {

                if(typeof(j[k])!='object')

                {

                    console.log(k+":"+j[k]);

                }

                else

                {

                    display(j[k]);

                }

            }

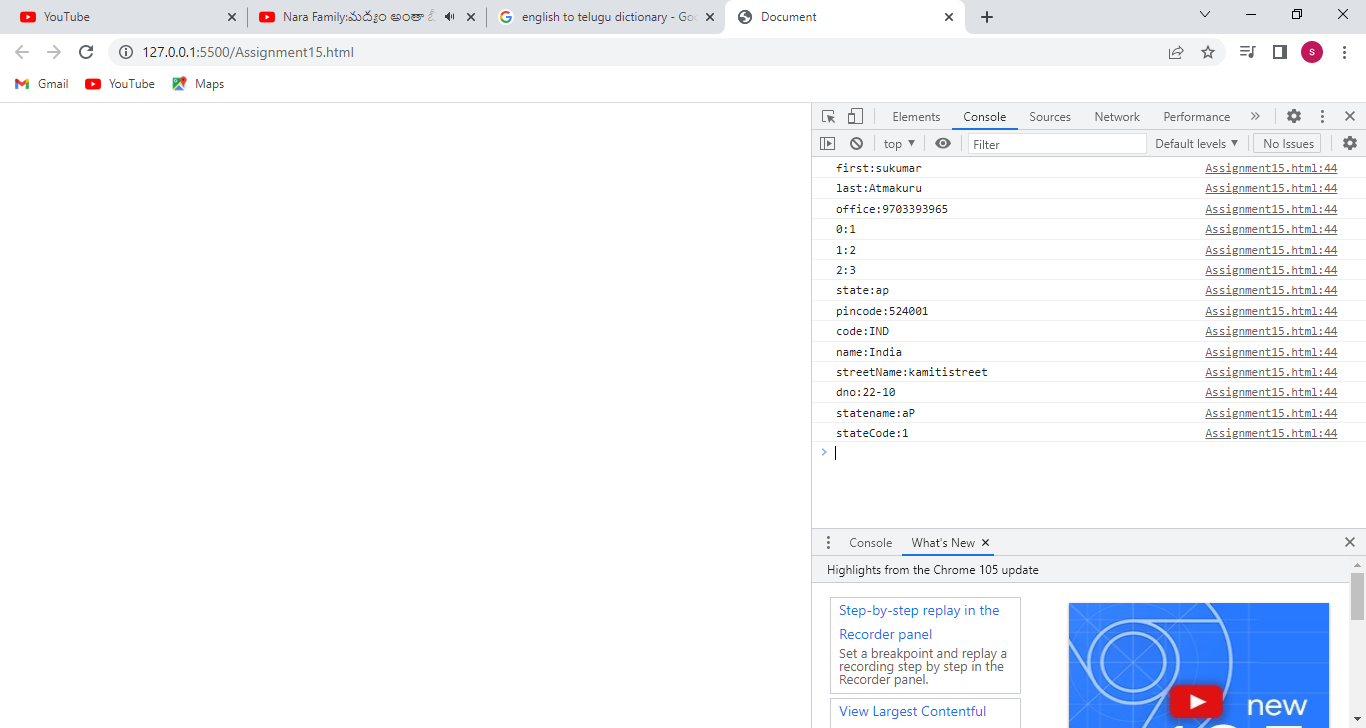
     }

     display(contacDetails);

    </script>

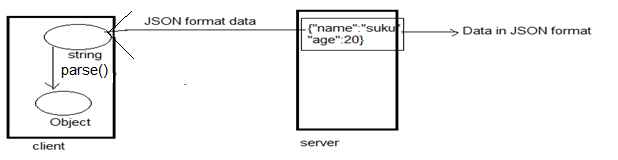
</body>

</html>



**14.2.7.stringify and parse** :

When a data came from server in JSON format, the data is always String object. Before using such data in client application, the client application must convert into JS object. To convert data from string into JS Object, we should use parse() method.



Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varstuData='{"name":"sukumar","age":20}';

     console.log(typeof(stuData));

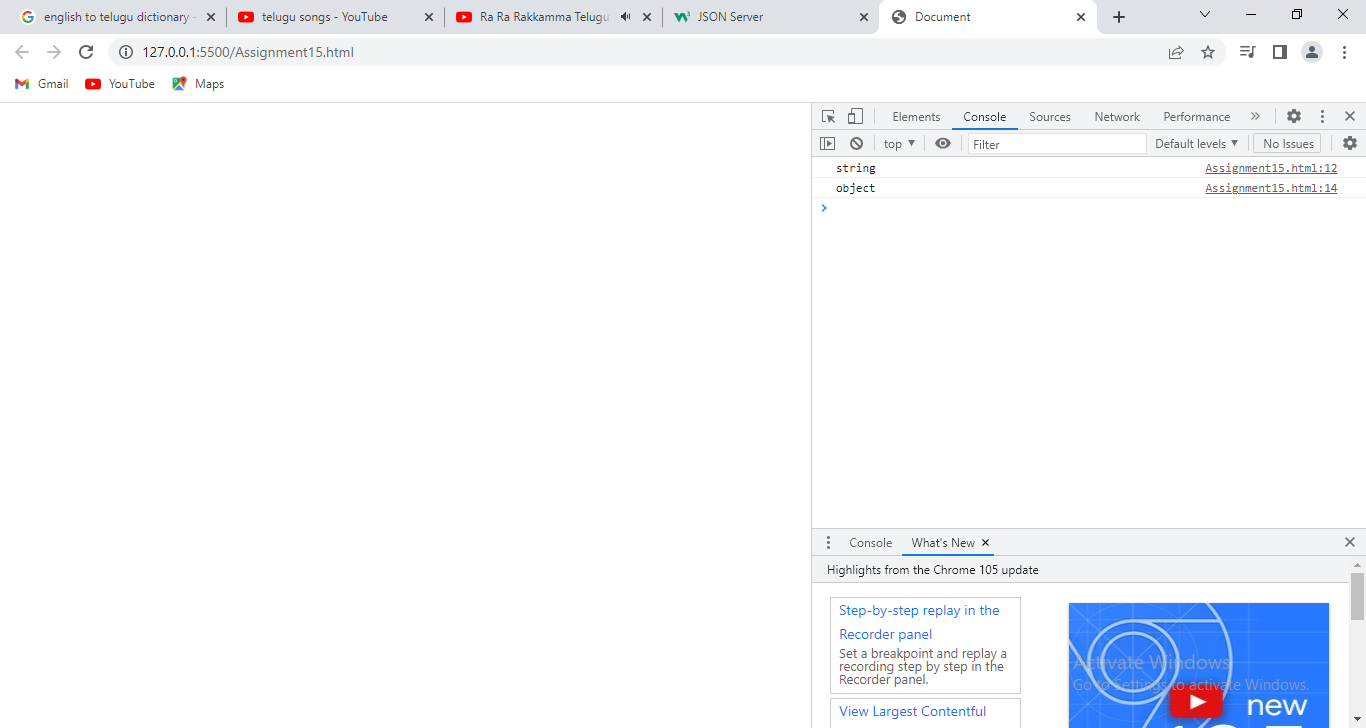
     varstuData=JSON.parse(stuData);

     console.log(typeof(stuData));

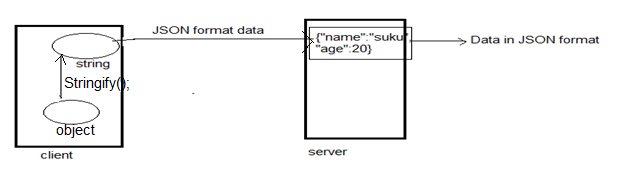
    </script>

</body>

</html>



When the JSON format data is sent from client application to server, the data is object and data must be translated from JS object into string . That conversion is done by stringify() method.



Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varstuData={"name":"sukumar","age":20};

     console.log(typeof(stuData));

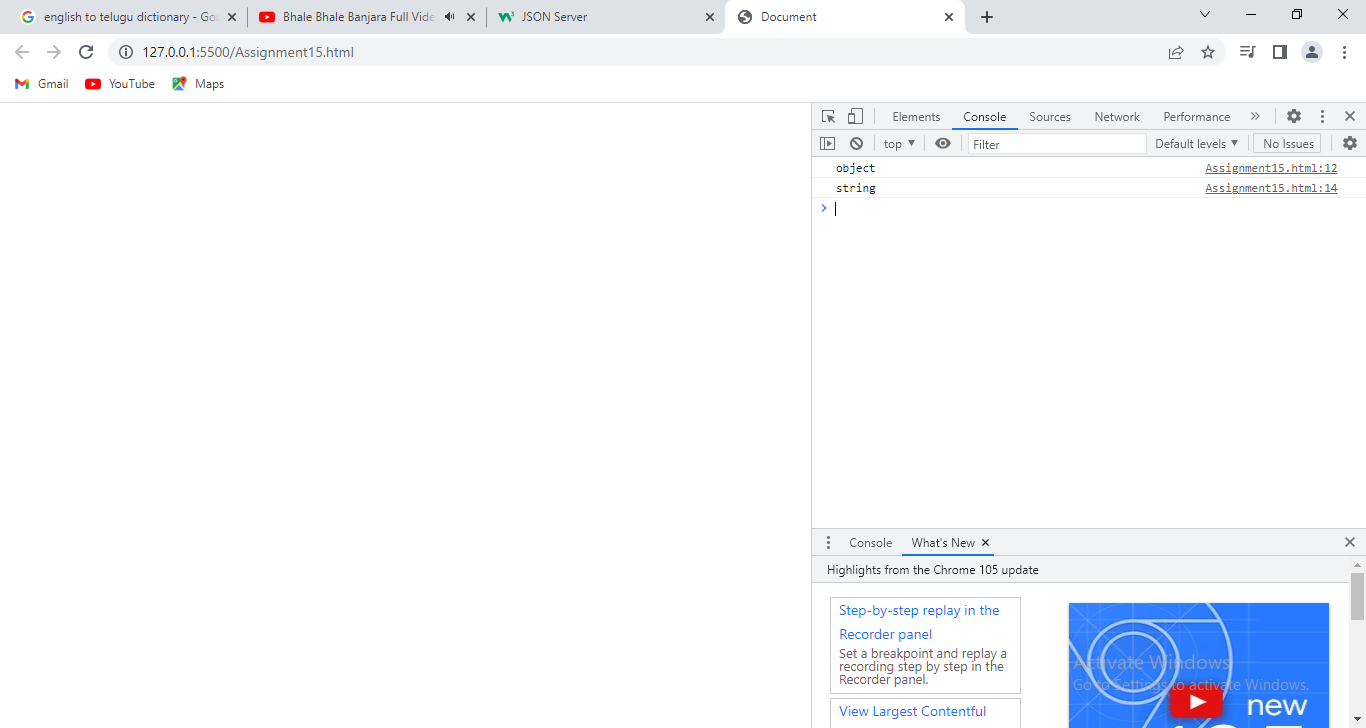
     varstuData=JSON.stringify(stuData);

     console.log(typeof(stuData));

    </script>

</body>

</html>



14.2.8. How to store Real world object properties and behaviours inJSON format:

Example:

Real world object: car

Properties:model,price,pur\_date;

Behaviours: run(),stop(),display().

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varcar={

        "model":'ambasador',

        "price":9000,

        "pur\_date":new Date(),

        "run":function(){console.log('Car can move'); car.stop();},

        "stop":()=>{console.log('Car stopped');car.display();},

        "display":()=>

        {

            console.log('Model:'+car.model);

            console.log('price:'+car.price);

            console.log('purdate:'+car.pur\_date);

        }

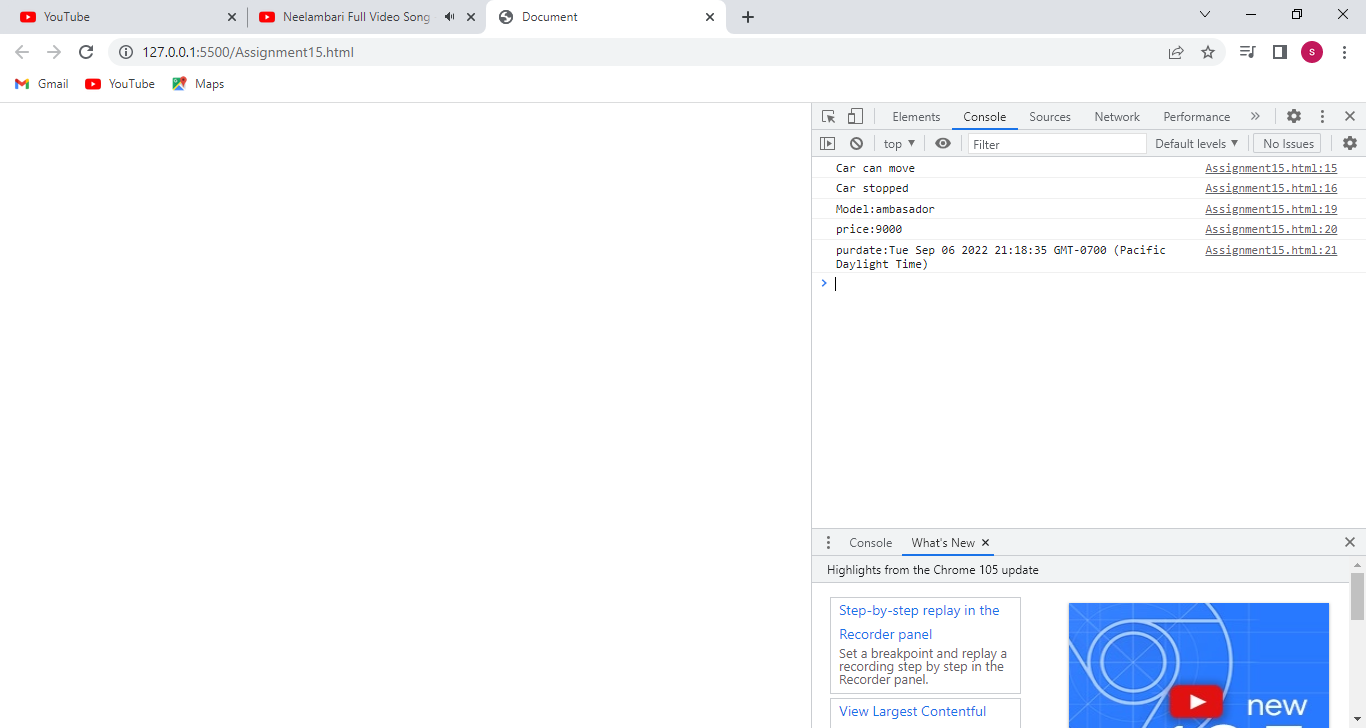
     }

     car.run();

    </script>

</body>

</html>

14.2.9. this:-A predefined Key word been supported in JavaScript which refers the current object data. In order to access objects data within its corresponding methods, we can access either by using object name or by using this keyword.

Example:1

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varcar={

        "model":'ambasador',

        "price":9000,

        "pur\_date":new Date(),

        "run":function(){console.log('Car can move'); },

        "stop":()=>{console.log('Car stopped'); },

        "display":function() {

            console.log('Model:'+this.model);

            console.log('price:'+this.price);

            console.log('purdate:'+this.pur\_date);

            this.run();

            this.stop();

        }

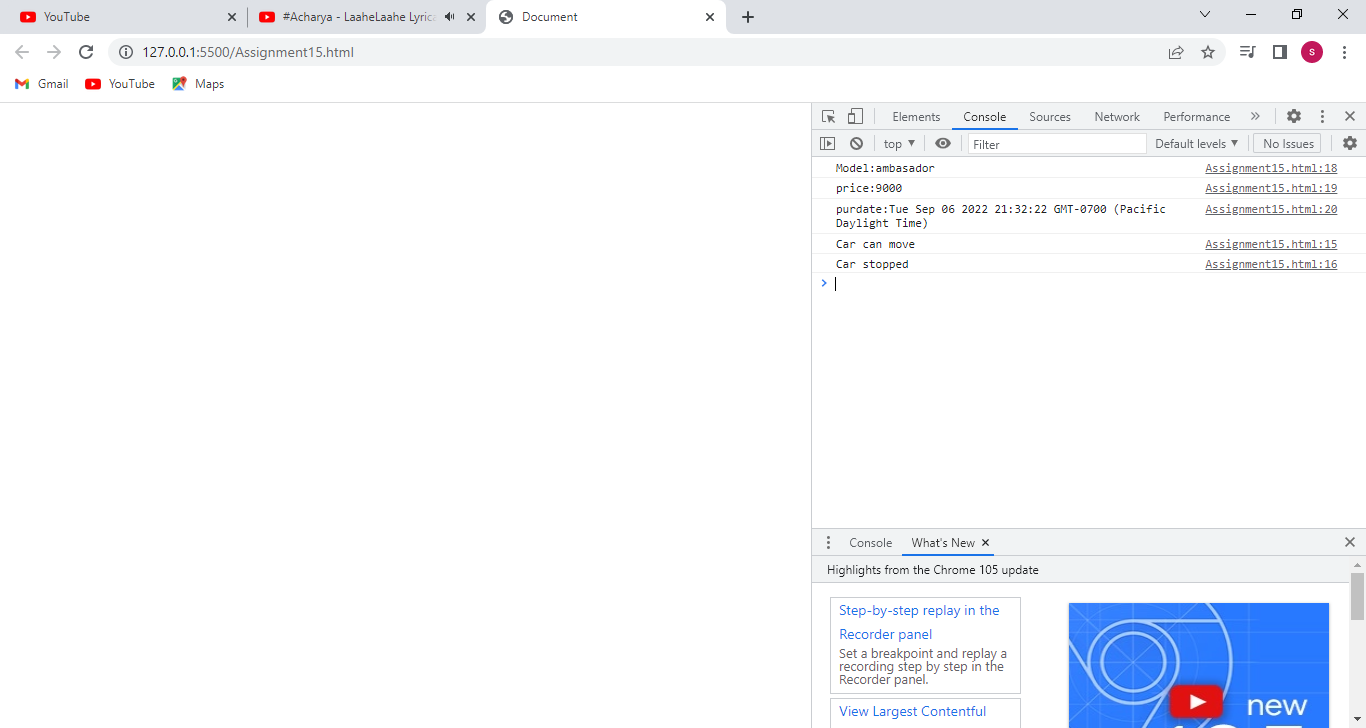
     };

     car.display();

    </script>

</body>

</html>



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varcar={

        "model":'ambasador',

        "price":9000,

        "pur\_date":new Date(),

        "run":function(){console.log('Car can move'); },

        "stop":()=>{console.log('Car stopped'); },

        "display":function() {

            console.log('Model:'+car.model);

            console.log('price:'+car.price);

            console.log('purdate:'+car.pur\_date);

            car.run();

            car.stop();

        }

     };

     car1= car;

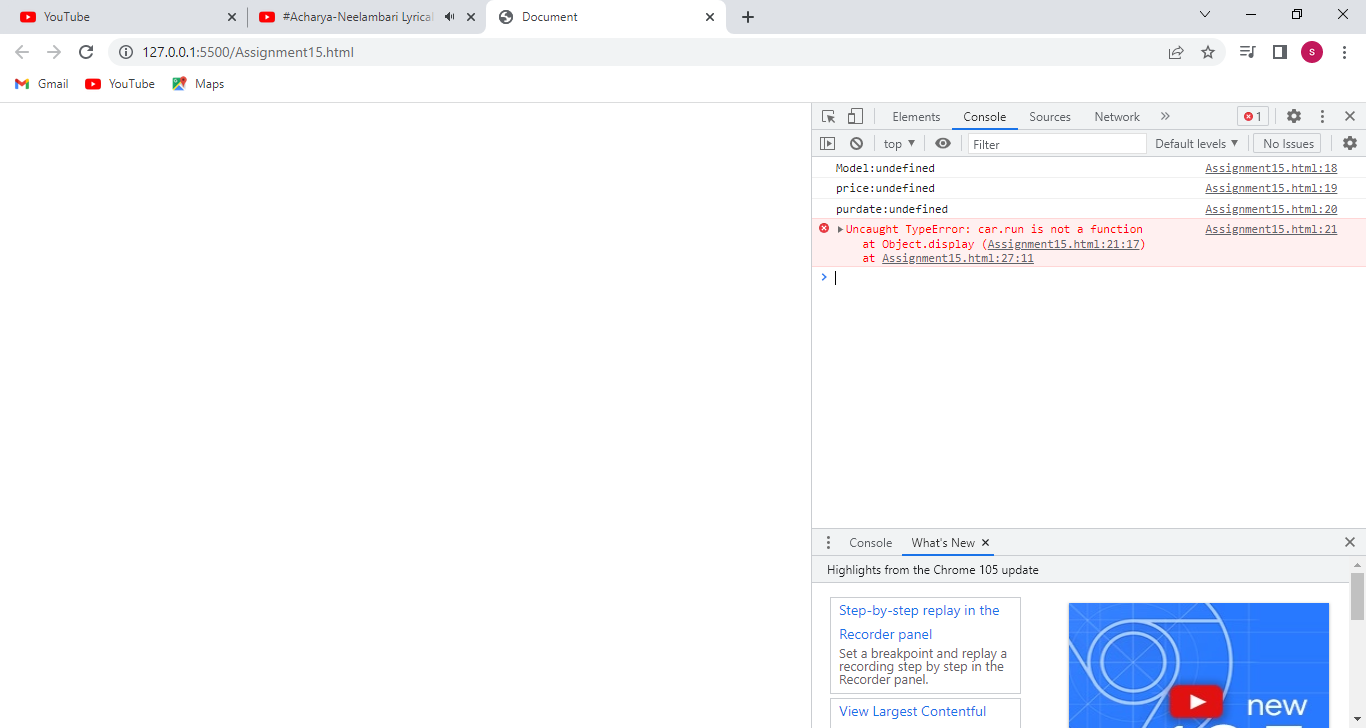
     car='';

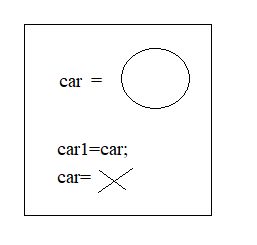
     car1.display();

    </script>

</body>

</html>





To solve above error, this keyword is useful.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <scripttype="text/javascript">

     varcar={

        "model":'ambasador',

        "price":9000,

        "pur\_date":new Date(),

        "run":function(){console.log('Car can move'); },

        "stop":()=>{console.log('Car stopped'); },

        "display":function() {

            console.log('Model:'+this.model);

            console.log('price:'+this.price);

            console.log('purdate:'+this.pur\_date);

            this.run();

            this.stop();

        }

     };

     car1= car;

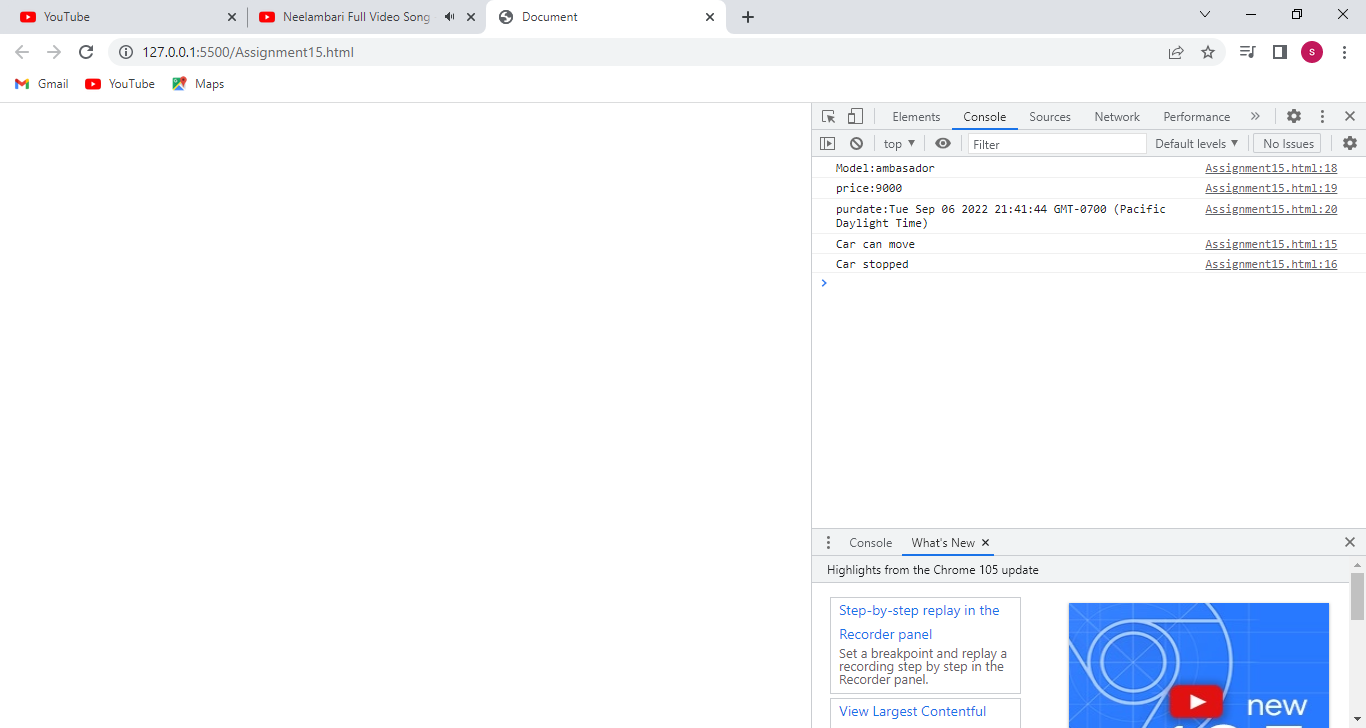
     car='';

     car1.display();

    </script>

</body>

</html>



**14.3. String**:- The string is sequence of characters.

An string in JavaScript can be created/declared and initialized in two ways.

1. string literal

2. string constructor

**1.String Literal**:

stirng Declaration:

Syntax:

Var name=’’// empty string is created.

Var name=’string’; String created with specified values.

Note:- The string type is ‘String’.

**2.String Constructor**:

String Declaration:

Syntax:

Var name=new String();// empty String is created.

At the time of String declaration, we can assign initial value to string variable by following syntax:

Var name=new String(‘-----‘); String created with specified values.

Note :- String type is ‘object’.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

       varx;

       vary=new String();

       varz='sukumar';

       varxy=new String('nellore');

       x='veena';

       y='Chirala';

       console.log('First String:'+x);

       console.log('Second String:'+y);

       console.log('Third String:'+z);

       console.log('Fourth String:'+xy);

</script>

</body>

</html>;

* Any string being defined in JavaScript it occupies the memory internally as like an array where each character occupies individual block with the corresponding index value.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

       varx='sukumar',i;

       console.log('String characters:');

       for(i=0;i<x.length;i++)

       {

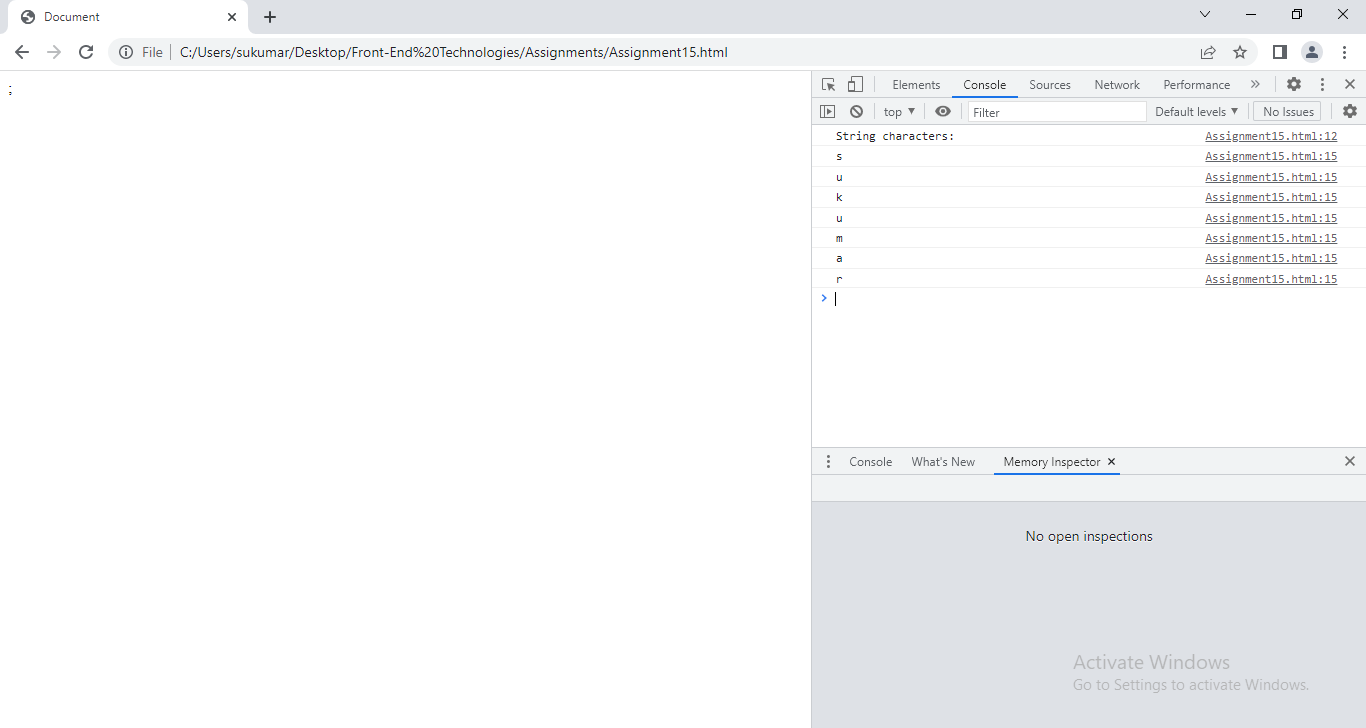
        console.log(x[i]);

       }

</script>

</body>

</html>



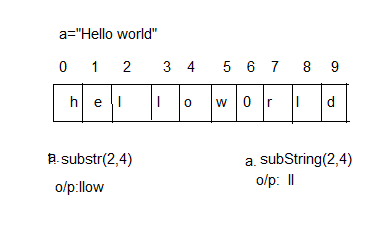
**Predefined methods can be applied on string:** Every string being created in JavaScript internally it gets created in the form of an array.

Following are the predefined methods can be applied on string in JavaScript,

* Length: It returns the total number of characters within a string, including spaces.
* charAt(<index>): Returns the character present at the provided index position.
* charCodeAt(<index>): Returns the ASCII value of a character present at provided index position.
* subStr(index,length of substring): Returns the substring from the given string.

SubString(starting index, ending index):Returns the substring from the given string to ending index-1.

Diff Between substr and Substring.



* replace(“what”,”with”): Used to replace the string parts with required strings.
* Split(“optional separator”): Used to split the main string in the form of an array.
* Includes(“<string>”): Checks weather the provided string is part of main string and returns true or false.
* Indexof(“string”): Returns at what index provided substring exist in main string. If string does not exist , then it returns -1.
* tolowercase(): Converts the provided string to lower case format.
* touppercase(): Converts the provided string to upper case format.
* Slice(start index, end index): Returns the substring from one index to other.
* Concat(str1,str2):Returns the concated string.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

       varx='Hello World';

       console.log('String Function Demonstrations:');

       console.log(x.charAt(2));

       console.log(x.charCodeAt(2));

       console.log(x.substr(1,4));

       console.log(x.substring(1,4));

       console.log(x.slice(1,4));

       console.log(x.indexOf('world'));

       console.log(x.indexOf('suku'));

       console.log(x.includes('hello'));

       console.log(x.includes('rock'));

       console.log(x.toUpperCase());

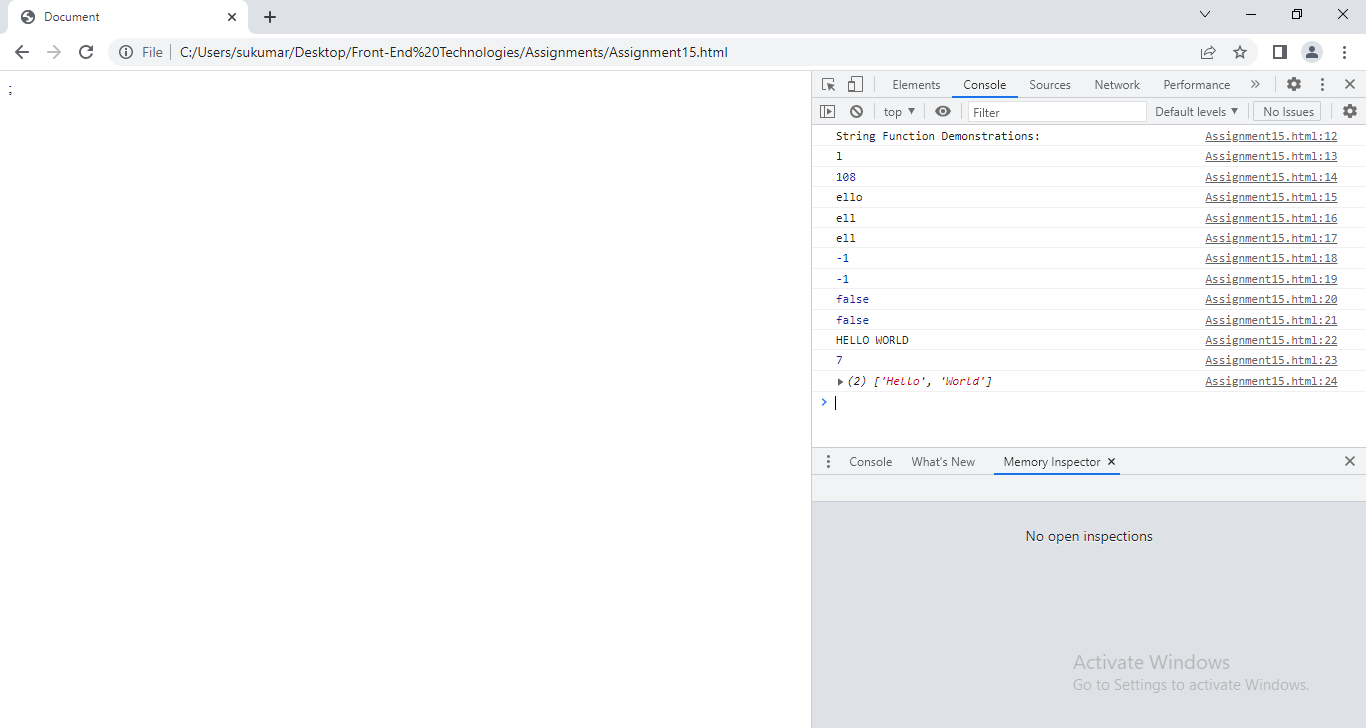
       console.log(x.lastIndexOf('o'));

       console.log(x.split(''));

</script>

</body>

</html>;



**15.PreDefined Objects:-**

**15.1.Date**:-“Date” is a predefined class supported in JavaScript using which we could able to work with the systems current date or user defined custom date.

Syntax:

Var varname=new Date();

//It creates Date object with current system date and time.

Var varname=new Date(‘Custom date’);

//It creates Date object with custom date and time.

Var varname=new Date(‘milliseconds’);

// It creates new date object with no.of millisecond since 1970/01/01.

Note:-When setting a date, without specifying the time zone, JavaScript will use the browser's time zone.

Default Date format is: MM Dayofmonth YYYY.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara = new Date();

      varb = new Date('nov 6 1981 07:07:45');

      varc = new Date(15759090150005);

      console.log('Current System Date:'+a);

      console.log('Custom Date:'+b);

      console.log('Date by Milliseconds:'+c);

</script>

</body>

</html>;

Following are the predefined methods which can be applied on date object,

get Date(): Returns the Day of the Month( 1st -31st ).

get Month(): Returns the current month value(0th-11th).

getFullYear(): Returns the full year

get Hours(): Returns the hours value.

get Minutes(): Returns 0-59

get Seconds(): Returns 0-59

get Milliseconds(): Returns 0-999

get Day(): Returns 0-6.

setDate(day of month): set day of month;

set Month(month number):Set the Month.

Set FullYear(year):set the Year.

Set Hours(hour):set the Hours.

set Minutes(minutes):set the Minutes.

Set Setconds(seconds):set the Seconds.

Set Milliseconds(milliseconds):set the milliseconds.

toLocalString(): It returns string which represents date object.

Parse():- It returns milliseconds from date to 1970 january 1.

Example:

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      varb = new Date('nov 6 1981 07:07:45:07');

      console.log("======================Getting Methods==========================");

      console.log('Custom Date:'+b);

      console.log('Day of Month:'+b.getDate());

      console.log('Month:'+b.getMonth());

      console.log('FullYear:'+b.getFullYear());

      console.log('Weak of Day:'+b.getDay());

      console.log('Hours:'+b.getHours());

      console.log('Minutes:'+b.getMinutes());

      console.log('Seconds:'+b.getSeconds());

      console.log('Milleseconds:'+b.getMilliseconds());

      console.log('=====================Setting Methods==========================');

      b.setDate(1);

      b.setMonth(1);

      b.setFullYear(2000);

      b.setHours(07);

      b.setMinutes(55);

      b.setSeconds(37);

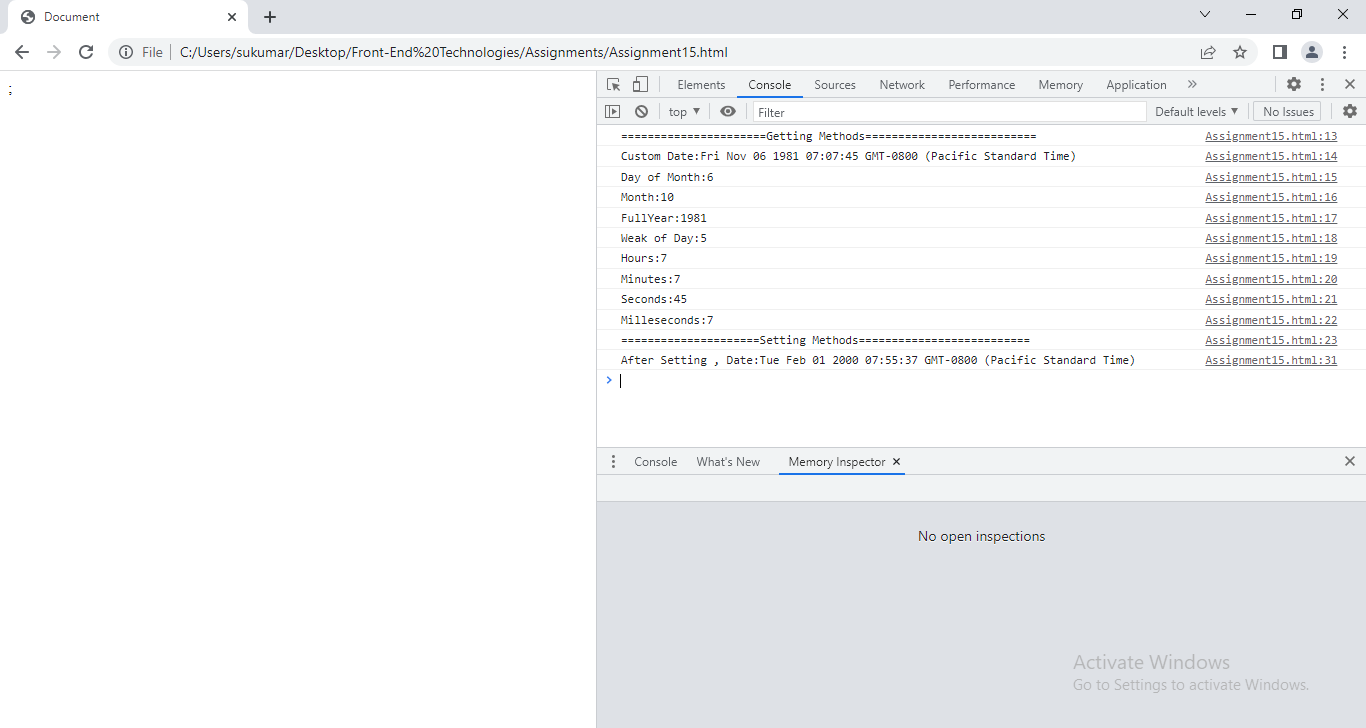
      b.setMilliseconds(45);

      console.log('After Setting , Date:'+b);

</script>

</body>

</html>;



Example: We display day Name and month name instead of Numbers.

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metahttp-equiv="X-UA-Compatible" content="IE=edge">

    <metaname="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

      vara=new Date();

      varday=['sun','mon','tue','wed','thu','fri','sat'];

      varmonth=['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'];

      console.log('Current System Date:'+a);

      console.log(day[a.getDay()]);

      console.log(month[a.getMonth()]);

</script>

</body>

</html>;

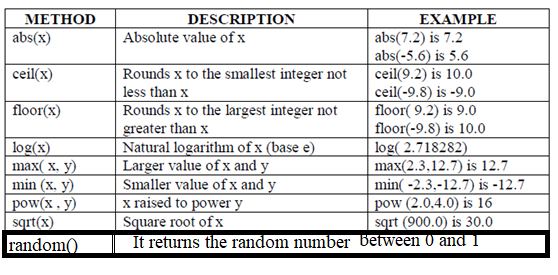
**15.2.Math:**The math object provides you properties and methods for mathematical constants and functions.

1) Properties:-

* + E \:Euler's constant and the base of natural logarithms, approximately 2.718.
  + LN2:Natural logarithm of 2, approximately 0.693.
  + LN10:Natural logarithm of 10, approximately 2.302.
  + LOG2E:Base 2 logarithm of E, approximately 1.442.
  + LOG10E:Base 10 logarithm of E, approximately 0.434.
  + PI:Ratio of the circumference of a circle to its diameter, approximately 3.14159.
* SQRT1\_2:Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.

SQRT2:Square root of 2, approximately 1.414.

2) Methods:-

****

Note: -

1. Random()\*10 return number between 0&9.
2. Random()\*100 return number between 0 and 99.
3. Random()\*1000 retrun number between 0 and 999.

Example:1 Write a JS that Generates 3Digit-OTP and 3Digit Captcha.

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<body>

  Generated 3Digit-OTP:<inputtype="text" id="otp">

  <button  onclick="generateOtp()">submit</button><br>

  Generated 3Digit-Capcha:<inputtype="text" id="capcha">

  <button  onclick="generateCapcha()">submit</button>

  <script>

      functiongenerateOtp(){

        vard1=Math.floor(Math.random()\*10);

        vard2=Math.floor(Math.random()\*10);

        vard3=Math.floor(Math.random()\*10);

        varx=document.querySelector('#otp');

        x.value=new String(d1)+new String(d2)+new String(d3);

        }

       functiongenerateCapcha()

       {

        varAlpha=['a','b','c','d','e','f','g','h','i','j'];

        varAlpha1=['A','B','C','D','E','F','G','H','i','j'];

        vard1=Math.floor(Math.random()\*10);

        vard2=Math.floor(Math.random()\*10);

        vard3=Math.floor(Math.random()\*10);

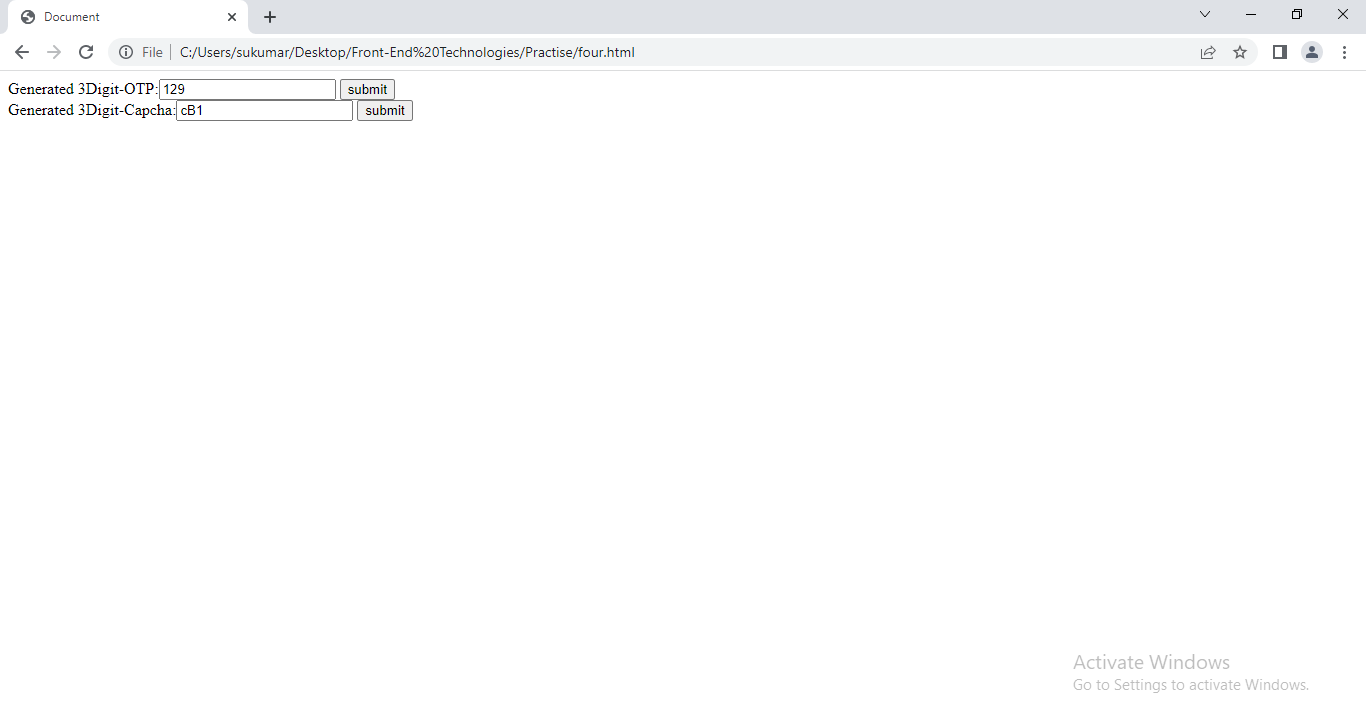
       document.querySelector('#capcha').value=Alpha[Math.floor(d1)]+Alpha1[Math.floor(d2)]+d3;

      }

  </script>

</body>

</html>



**16.Event Handling:**

16.1. Event:The Event Is notable action which is happened inside the browser .

16.2. Event Handler: Event handler is java script code which is associated with part of document and event. The event handler is executed when event occurs.

16.3. HTML Events:

Mouse Related attribute names:-

|  |  |
| --- | --- |
| Attribute Name | Description |
| Onclick | Fires on mouse click on element |
| Ondblclick | Fires on mouse dbl click on element. |
| Onmousemove | Fires when mouse pointer is moving over element |
| Onmouseout | Fires when mouse pointer is came out of element. |
| Onmouseup | Fires when mouse button is released over element. |
| Onmousedown | Fires when mouse button is pressed over element. |
| Onwheel | Fire when mouse wheelrollsup over element. |

Keyboard Related attribute names:-

|  |  |
| --- | --- |
| Attribute name | Description |
| Onkeydown | Fired when you pressdown key on keyboard |
| Onkeyup | Fired when you pressup key on keyboard |
| Onkeypress | Fires when you press the key on keyboard. |

Clipboard related attribute names:

|  |  |
| --- | --- |
| Attribute name | Description |
| Oncopy | Fires when user copies content of element |
| Oncut | Fires when user cut content of element. |
| Onpaste | Fires when user paste some content. |

Form Related attribute names:

|  |  |
| --- | --- |
| Attribute name | Description |
| Onblur | Fires when element has lost focus |
| Onfocus | Fires when element gets focus |
| Oninput | Fires when element get user input |
| Onchange | Fires when element value is changed |
| Onselect | Fires when element value is selected. |
| Onsubmit | Fires when form is submitted. |
| Onreset | Fires when reset button is pressed. |
| Onsearch | Fires when user writes in searchbox |
| oninvalid | Fires when element is invalid. |

Window related attribute names:

|  |  |
| --- | --- |
| Attribute name | Description |
| Onload | Indicates that the object (typically a document,frameset,body,script,or style.) has finished loading into the browser. |
| Onunload | Indicates that the object(document) has finished unloading from the browser. |
| Onbeforeunload | Fires before document is unloaded. |
| Onresize | Fires when window is resized. |
| Onafterprint | Fires after document has been printed. |
| Onbeforeprint | Fires before document is printed. |
| Onoffline |  |
| Ononline |  |

Drag related attribute names:

|  |  |
| --- | --- |
| Attribute names | description |
| Ondrag | Fires when element is being dragged |
| Ondragstart | Fires when user starts to drag the element |
| Ondragend | Fires when user has finished the dragging. |
| Ondrop | Fires when dragged element has dropped on drop target . |
| Ondragover | Fires when the dragged element is over the drop target |
| Ondragenter | fireswhen the dragged element enters the drop target |
| Ondragleave | Fires when the dragged element leaves the drop target |

DOMContentLoaded: It is event. When DOM content is loaded with out images, stylesheets, this event is fired.

16.4. Event models:- There are several ways to

i.Bind event and event handler.

ii. Triggering events.

Based on them, Event models are 3 types.

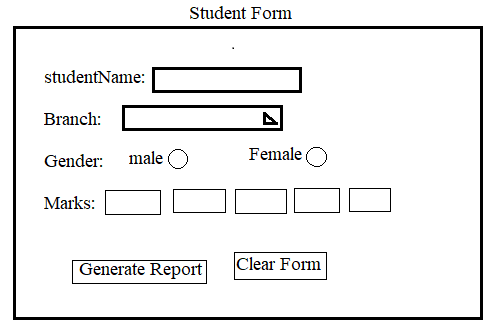
1. Basic Event Model.
2. Proprietary(own) Event model
3. DOM model.

9.4.1. Basic Event Model:The basic model is simple, widely supported, and easy to understand. Proprietary browser event models and the newer DOM2 model are compatible with this basic model. This means that you can stick to the basic model even in the most recent browsers.

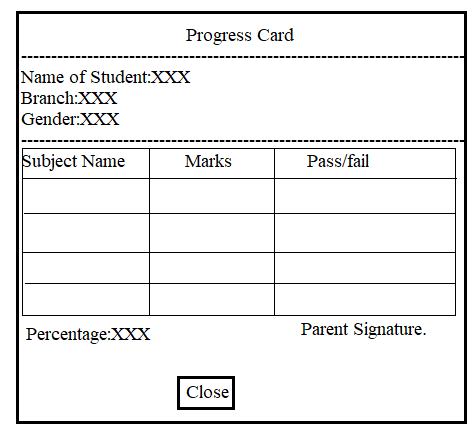
**Event binding**:- we assign event handler to event attribute in Html element. when the browser parses page and create document hierarchy, browser binds the event handler to element using attribute.

Ex: <p onclick=’function-name()/ js code’>…</p>

Example: Mock Up



After pressing the Generate report



<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

  <style>

   div{

    margin:5px0px;

    box-sizing: border-box;

    height:5vh;

    width:50vw;

    border:1px solid red;

    background-color: black;

    color:white;

    text-align: center;

    padding:1vh0px;

    position: relative;

    left:25vw;

    }

    ul{

      list-style-type: none;

      box-sizing: border-box;

      width:50vw;

      height:auto;

      position:relative;

      left:25vw;

      border:1px solid green;

      margin:10px0px;

      padding:0px;

      background-color: green;

      }

    ul>li{

      margin:10px;

      padding:5px;

      border:1px solid black;

    }

    ul>li:nth-child(4)>input[type='text']

    {

      width:2vw;

      margin:0px2px;

    }

    ul>li:last-child{

      text-align: center;

    }

    ul>li:last-child>button,input[type='reset']

    {

      font-family:'Times New Roman', Times, serif;

      background-color: red;

      color:blue;

      font-size: 1.3rem;

    }

  </style>

</head>

<body>

  <div>STUDENT-FORM</div>

  <ul>

    <li>

      Name Of the Student: <inputtype="text" name="sname">

    </li>

    <li>

      Branch: <selectname="branch" id="branch">

               <optionvalue="MCA" selected>MCA</option>

               <optionvalue="MBA">MBA</option>

              </select>

    </li>

    <li>

      Gender:   <inputtype="radio" value="Male" name="g"> Male

                <inputtype="radio" value="Female" name="g">Female

    </li>

    <li>

      Marks:

      <inputtype="text"><inputtype="text"><inputtype="text"><inputtype="text">

    </li>

    <li>

         <buttononclick="getData()">generateProgressCard</button><inputtype="reset">

    </li>

  </ul>

  <script>

  varstudentData={}

  functiongetData(){

  varallInputElements=document.getElementsByTagName('input');

  studentData.Name=allInputElements[0].value;

  if(allInputElements[1].checked==true)

  {

    studentData.Gender='male';

  }

  else

  {

    studentData.Gender='Female';

  }

  studentData.marks= new Array(allInputElements[3].value,allInputElements[4].value,allInputElements[5].value,allInputElements[6].value);

  studentData.branch=document.getElementsByTagName('select')[0].value;

  varchild1=document.getElementsByTagName('div');

  document.body.removeChild(child1[0]);

  varchild1=document.getElementsByTagName('ul');

  document.body.removeChild(child1[0]);

  progressCard();

  }

  functionprogressCard(){

    varcon1=document.createElement('div');

    con1.style='border:2px solid red; height:auto; width:50vw;postion:relative;left:25vw;background-color:orange;'

    varcomm=document.createElement('h2');

    comm.style='text-align:center;';

    comm.innerHTML='PROGRESS CARD';

    con1.appendChild(comm);

    con1.appendChild(document.createElement('hr'));

    place('p',studentData.Name,'Name',con1);

    place('p',studentData.branch,'Branch',con1);

    place('p',studentData.Gender,'Gender',con1);

    con1.appendChild(document.createElement('hr'));

    varmarkTable=document.createElement('table');

    markTable.setAttribute('border',1);

    markTable.style='width:100%;';

    tabRow(markTable,'th',['subject','marks','Pass/Fail']);

    tabRow(markTable,'td',['DBMS',studentData.marks[0]]);

    tabRow(markTable,'td',['DS',studentData.marks[1]]);

    tabRow(markTable,'td',['OS',studentData.marks[2]]);

    tabRow(markTable,'td',['JAVA',studentData.marks[3]]);

    con1.appendChild(markTable);

    varxy=document.createElement('h6');

    xy.style='text-align:right;';

    xy.innerHTML='ParentSignature';

    con1.appendChild(xy);

    xy=document.createElement('button');

    xy.innerHTML='close';

    xy.style='font-size:1.5rem;text-align:center';

    xy.setAttribute('onClick',"close1()");

    con1.appendChild(xy);

    document.body.appendChild(con1);

  }

  functionplace(a,b,c,d)

  {

    varx=document.createElement(a);

    x.innerHTML=c+':'+b;

    x.style='text-align:left;font-size:1.2rem;margin:2px 0px';

    d.appendChild(x);

  }

  functiontabRow(a,b,c){

    varx=document.createElement('tr');

    for(vari=0;i<3;i++)

    {

      vary=document.createElement(b);

      if(i==2&&b=='td')

      {

        if(c[1]>35)

        {

          y.innerHTML='pass';

        }

        else

        {

          y.innerHTML='fail';

        }

      }

      else{

        y.innerHTML=c[i];

      }

      x.appendChild(y);

    }

    a.appendChild(x);

  }

  functionclose1()

  {

    varx=document.getElementsByTagName('div');

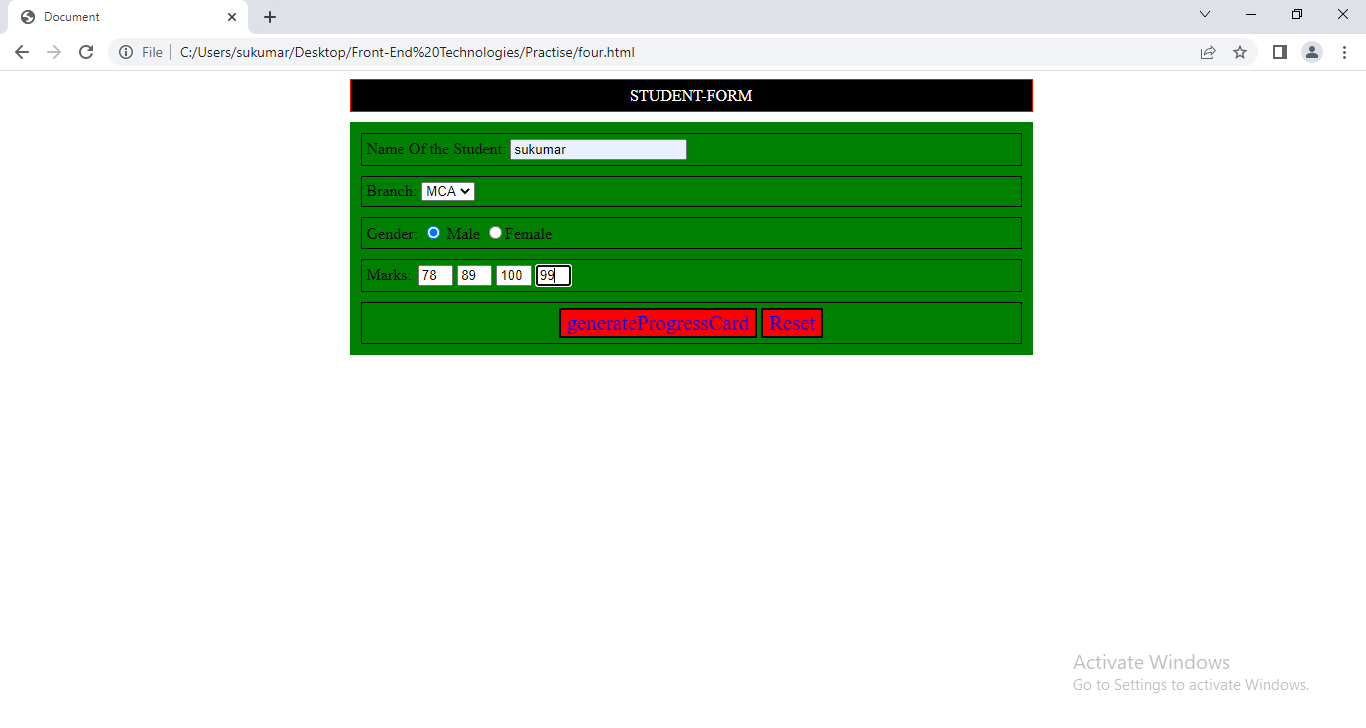
    document.body.removeChild(x[0]);

  }

</script>

</body>

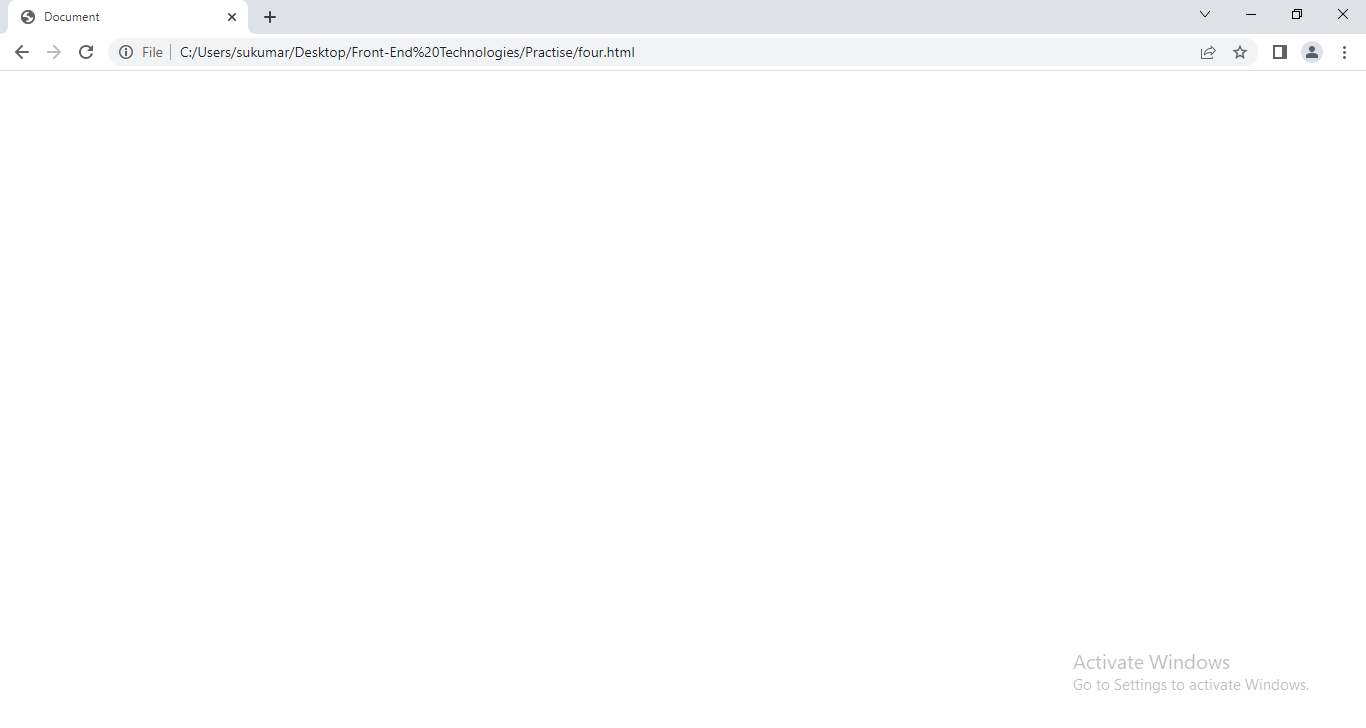
</html>



After pressing the Generating report Button.



After pressing close window button.



**Event handler return value**:- One of the most useful features of event handlers is that their return values can affect the default behaviour of the event. The default behaviour is what would normally happen when the event occurs if left unhandled. For example:

* The default behaviour of a click on a link is to load the link target in the browser.
* The default behaviour of activating a Submit button is the submission of the form.
* The default behaviour of a Reset button is to clear form fields, and so on.

To cancel the default behaviour of an event, simply return ‘false’ from its event handler.

Ex:

<p onclick=’return abc()’>click Me</p>

Function abc()

{

Return false;

}

**Firing Events Manually**:- You can also invoke events directly on certain objects with JavaScript. Doing so causes the default action for the event to occur.

Event MethodElements

Click() <input type=’button/checkbox/reset/submit/radio’ >,</a>

Blur() <select><input><text><area>

Focus() <select><input><text><area>

Select() <input type=’text/password/file><textarea>

Submit() <form>

Reset() <form>

Limitations of Basic Event model:-

* First off, in the basic model, no extra information about the event is passed to the handler.
* you are limited to firing events manually on those elements that provide event methods.

16.4.2. Event:- when event raises, Browser will creates ‘event’ object .

The properties are

1.pageX:- It is numeric value indicating horizontal co-ordinate where event occurred.

2.pageY:- It is numeric value indicating vertical co-ordinate where event occurred.

3.screenX:- It is numeric value indicating horizontal co-ordinate where event occurred relative to whole screen.

4.screenY:- It is numeric value indicating vertical co-ordinate where event occurred relative to whole screen.

5. target:- Reference to the object at which the event occurred.

6. type: String containing the event type (for example, "click").

7.cancellable:- The event is cancellable if it is possible to prevent events default action. This property returns Boolean value.

8.bubbles:- This property returns Boolean value that indicates whether or not event is bubbling event.

Methods:-

1.stopPropagation():- This method prevents propagation of event. Propagation means bubbling up to parent element. (or) Capturing down to child Element.

2. Prevent Default():- This method cancel default action of event.

9.4.3.Proprietary(own) Event model: Version 4 of Netscape and Internet Explorer added new proprietary event models.

1.Version4 of Netscape:- events begin at the top of the hierarchy and ―trickle‖ down to the object at which they occurred, affording enclosing objects the opportunity to modify, cancel, or handle the event. It is called ‘EventCapturing’.

Example:1

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<style>

    div:first-child

    {

      border:1px solid red;

      text-align: center;

      margin: 5px;

    }

    div:nth-child(2)

    {

      border: 1px solid green;

      text-align:center;

      margin:5px;

    }

   p{

      border:1px solid orange;

      margin:5px;

    }

</style>

<body>

  /\*Dynamically Add Event to html element.\*/

  <div>This is Grand Parent Element.

    <div>This is Parent Element.

      <p>This is Child Element.</p>

    </div>

  </div>

  <script>

  vareles=document.querySelectorAll('div');

  vareles1=document.querySelector('p');

  eles[0].addEventListener('click',abc,true);

  eles[1].addEventListener('click',xyz,true);

  eles1.addEventListener('click',xyz1,true);

  functionabc(a)

  {

    console.log('grand parent');

  }

  functionxyz(b)

  {

    console.log('Parent Element');

  }

  functionxyz1(c)

  {

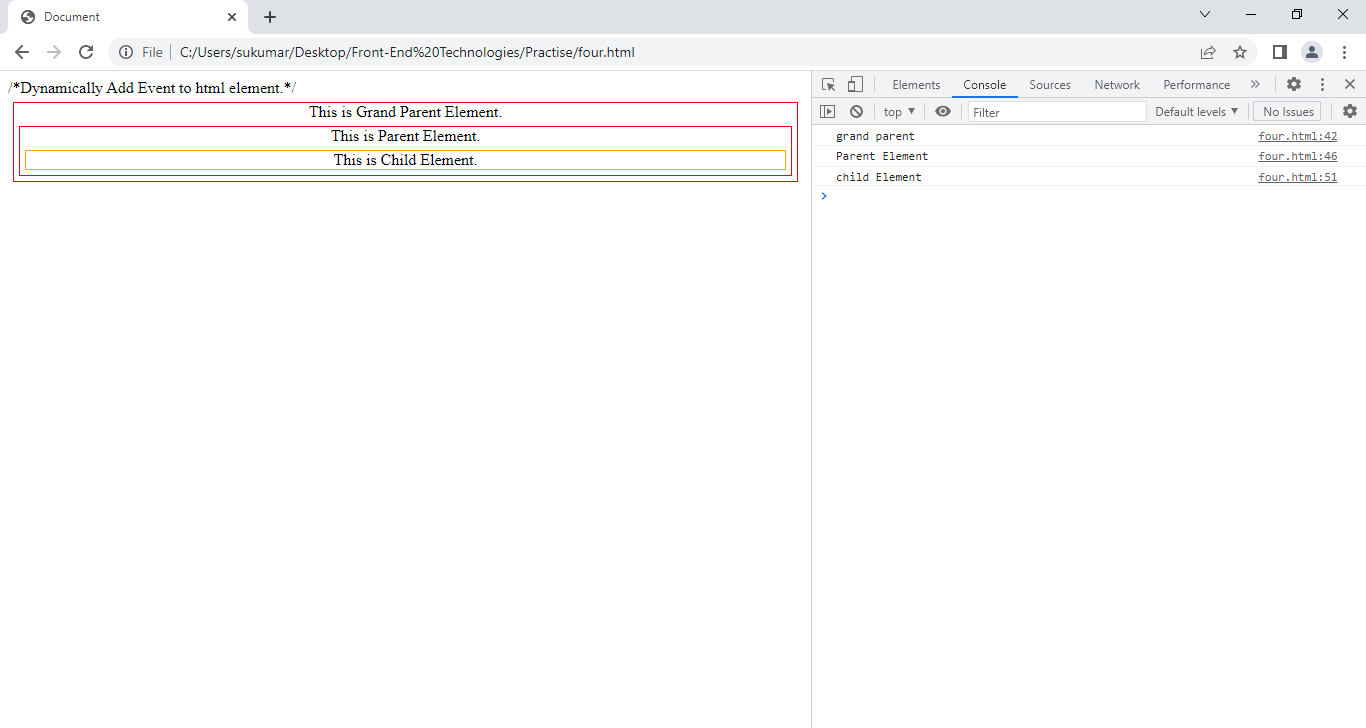
    console.log('child Element')

  }

  </script>

</body>

</html>



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<style>

    div:first-child

    {

      border:1px solid red;

      text-align: center;

      margin: 5px;

    }

    div:nth-child(2)

    {

      border: 1px solid green;

      text-align:center;

      margin:5px;

    }

   p{

      border:1px solid orange;

      margin:5px;

    }

</style>

<body>

  /\*Dynamically Add Event to html element.\*/

  <div>This is Grand Parent Element.

    <div>This is Parent Element.

      <p>This is Child Element.</p>

    </div>

  </div>

  <script>

  vareles=document.querySelectorAll('div');

  vareles1=document.querySelector('p');

  eles[0].addEventListener('click',abc,true);

  eles[1].addEventListener('click',xyz,true);

  eles1.addEventListener('click',xyz1,true);

  functionabc(a)

  {

    console.log('grand parent');

  }

  functionxyz(b)

  {

    b.stopPropagation();

    console.log('Parent Element');

  }

  functionxyz1(c)

  {

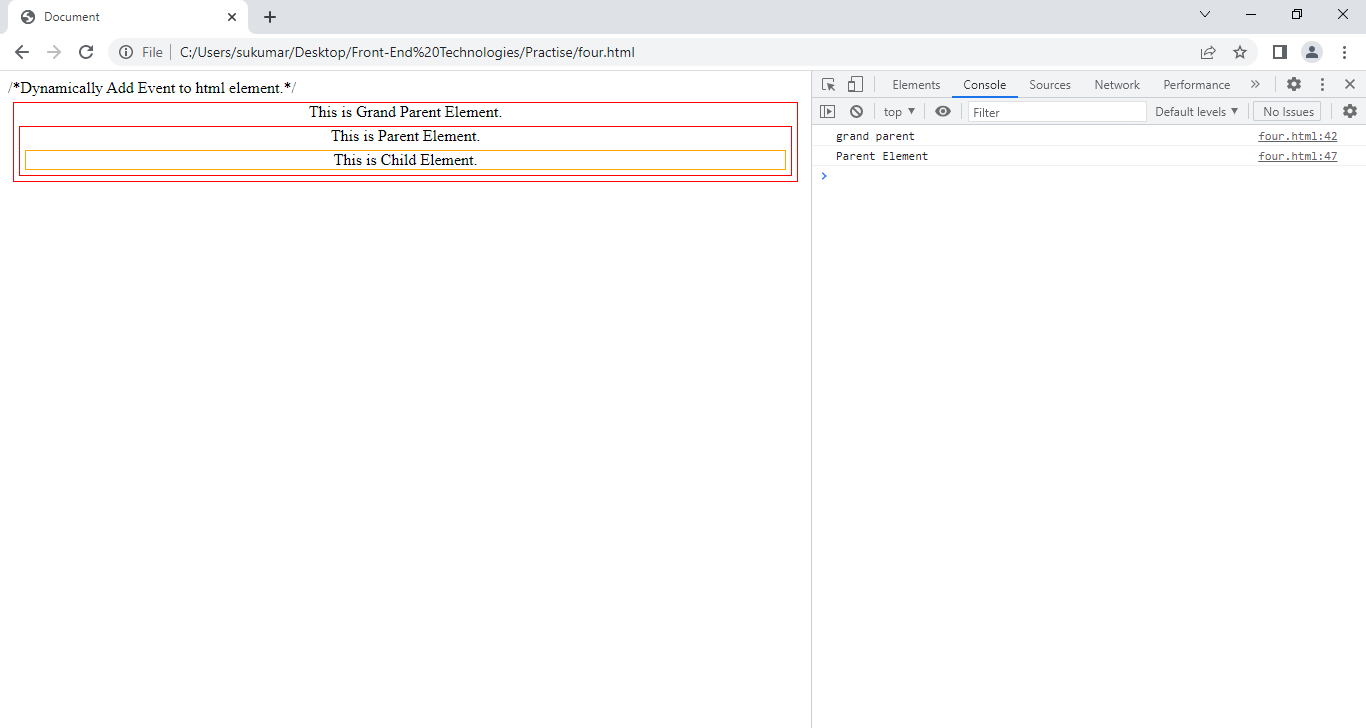
    console.log('child Element')

  }

  </script>

</body>

</html>



2.Version4 of InternetExplorer:- Under Internet Explorer, events begin at the object where they occur and ―bubble‖ up the hierarchy.It is called ‘Event Bubbling’.

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<style>

    div:first-child

    {

      border:1px solid red;

      text-align: center;

      margin: 5px;

    }

    div:nth-child(2)

    {

      border: 1px solid green;

      text-align:center;

      margin:5px;

    }

   p{

      border:1px solid orange;

      margin:5px;

    }

</style>

<body>

  /\*Dynamically Add Event to html element.\*/

  <div>This is Grand Parent Element.

    <div>This is Parent Element.

      <p>This is Child Element.</p>

    </div>

  </div>

  <script>

  vareles=document.querySelectorAll('div');

  vareles1=document.querySelector('p');

  eles[0].addEventListener('click',abc);

  eles[1].addEventListener('click',xyz);

  eles1.addEventListener('click',xyz1);

  functionabc(a)

  {

    console.log('grand parent');

  }

  functionxyz(b)

  {

    console.log('Parent Element');

  }

  functionxyz1(c)

  {

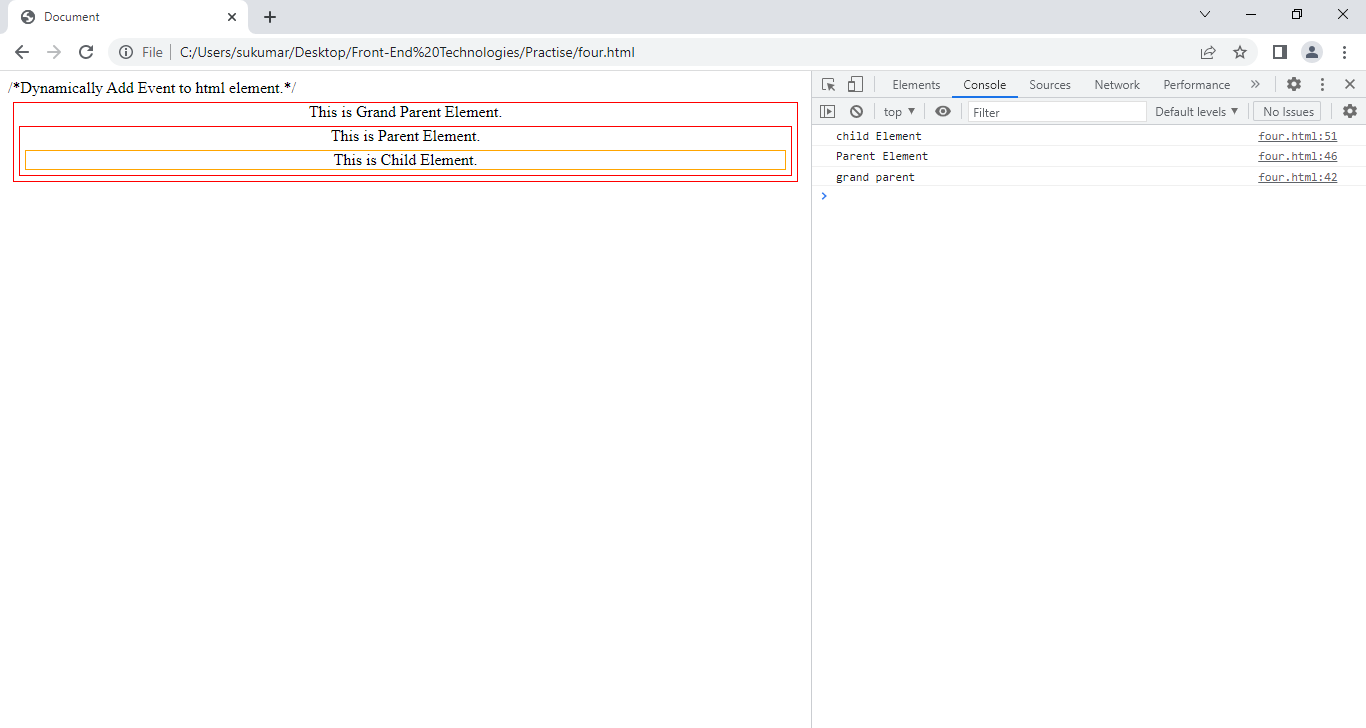
    console.log('child Element')

  }

  </script>

</body>

</html>



Example:2

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<style>

    div:first-child

    {

      border:1px solid red;

      text-align: center;

      margin: 5px;

    }

    div:nth-child(2)

    {

      border: 1px solid green;

      text-align:center;

      margin:5px;

    }

   p{

      border:1px solid orange;

      margin:5px;

    }

</style>

<body>

  /\*Dynamically Add Event to html element.\*/

  <div>This is Grand Parent Element.

    <div>This is Parent Element.

      <p>This is Child Element.</p>

    </div>

  </div>

  <script>

  vareles=document.querySelectorAll('div');

  vareles1=document.querySelector('p');

  eles[0].addEventListener('click',abc);

  eles[1].addEventListener('click',xyz);

  eles1.addEventListener('click',xyz1);

  functionabc(a)

  {

    console.log('grand parent');

  }

  functionxyz(b)

  {

    b.stopPropagation();

    console.log('Parent Element');

  }

  functionxyz1(c)

  {

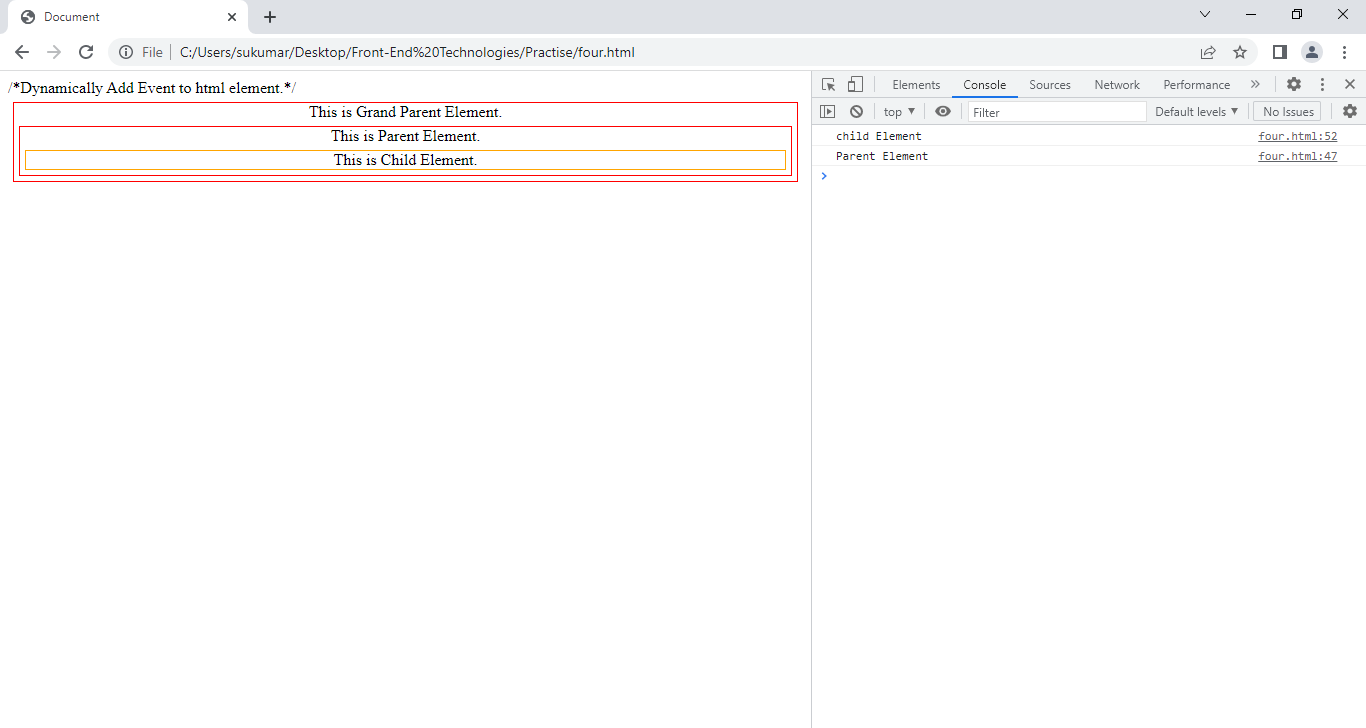
    console.log('child Element')

  }

  </script>

</body>

</html>



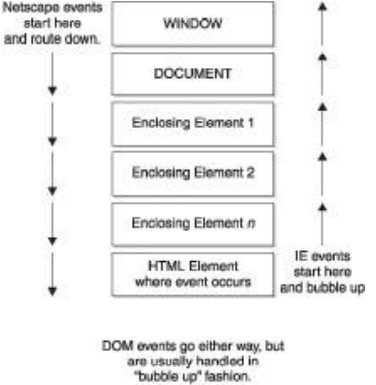
16.4.4**.DOM2 Event Model**:- The Level 2 DOM goes even further by merging the proprietary models into one standard and extending its capabilities considerably. Dom2 Event model is supported by all browsers.

This event model follows below steps:

1.Event capturing.

2.Target Element.

3. Event bubbeling.



**16.4.6.** Dynamically Binding Event to Html Element and Dynamically Removing Event from HTML element:

1. addEventListener:-

Syntax:-elementobject.addEventListener(“event“,handler, capturePhase);

* + - object is the node to which the listener is to be bound.
    - "event" is a string indicating the event it is to listen for.
    - handler is the function that should be invoked when the event occurs.
    - capturePhase is a Boolean indicating whether the handler should be invoked during the capture phase (true) or bubbling phase (false). False is default value.

1. removeEventListener:-

Syntax:

elementobject.removeEventListener(“eventype’,’Eventhandler’);

This method removes event handler which is attached to specific element by addeventlisterner() method.

Example:

**17.Validation:**when you enter the data in input element, web application/browser checks it to see that the data is correct. This checking is called validation.There are two types of validations.

1.client-side validation

2.server-side validation.

10.1 Server-side Validation:validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.JavaScript provides a way to validate data on the client's computer before sending it to the web server.

Some validations should be done at server side.

Example:

1. OTP and CAPCHA checking.
2. Username/Id and password validation.

...etc.

10.2. Client Side Validation:-This validation occurs in browser/client side, before data has been submitted to server. It can be divided into 2 types.

1.Java-script validation: The programmer writes a logic in java script to validate data.

Examples:

1. Data should start with capital letter.
2. Total length of string should be > 8 characters.

...etc.

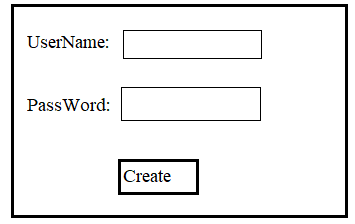
2. built-in form validation: Browser Defaultly validates the data.

Examples:

1. When you don’t enter the data in required input element, Browser know that immediately show that the “field is required.”

…etc.

Example:1 Java Script validations:



1. The User Name length should only be 3 characters. Otherwise Error message should be displayed just beneath of input element.
2. A. The password should start with ‘capital letter’.

B. The password length should be minimum 3 characters.

C. The password should end with number.

d. The password should has one ‘special character’.

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

  <metahttp-equiv="X-UA-Compatible" content="IE=edge">

  <metaname="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

  <style>

    body

    {

      background-color: blue;

    }

    div,ul{

      text-align: center;

      color:black;

      background-color: white;

      width: 50vw;

      position: relative;

      left:25vw;

      border:2px solid  red;

      box-sizing: border-box;

      margin:5px0px;

      font-size:25px;

    }

    ul{

      list-style-type: none;

      text-align: left;

      }

    ul>li{

      margin:5px;

    }

    ul>li>input{

      border:2px solid blue;

      background-color: black;

      color:white;

      font-size: 20px;

    }

    ul>li>p{

       color:red;

       font-size: .75rem;

        }

    button{

      position: relative;

      left:45vw;

      height:5vh;

      width:10vw;

      background-color:gray;

    }

  </style>

</head>

<body>

  <div>Create Credentials</div>

  <ul>

    <li>

      UserName:

      <inputtype="text" onkeyup="dataValidate(event)">

      <pstyle="display:none;color:red;">The UserName should only be 3 characters.</p>

    </li>

    <li>

      PassWord:

      <inputtype="text" onkeyup="check1(event)">

      <p>\*Pwd Starting letter should be Capital Letter</p>

      <p>\*Pwd length should only be 3 characters.</p>

      <p>\*pwd should end with digit.</p>

    </li>

  </ul>

  <buttononclick="clearData()">Create</button>

  <script>

    varcount=0;

   functiondataValidate(a)

    {

       varcount=document.querySelector('input[type=text]').value;

       if(String(count).length==3 || String(count).length==0)

      {

        document.querySelector('p').style.display='none';

      }

      else

      {

        document.querySelector('p').style.display='block';

      }

    }

    functioncheck1(a)

    {

      varpEle=document.querySelectorAll('p');

      varpwd=document.querySelectorAll('input[type]');

      if((a.keyCode>=48&&a.keyCode<=57) &&String(pwd[1].value).length==3  )

      {

        pEle[3].style.color='green';

      }

      else

      {

        pEle[3].style.color='red';

      }

      if(String(pwd[1].value).length==3 )

      {

        pEle[2].style.color='green';

      }

      else

      {

        pEle[2].style.color='red';

      }

      if(String(pwd[1].value).length==1&& (a.keyCode>=97&&a.keyCode<=126)  )

      {

        pEle[1].style.color='green';

      }

      else

      {

        pEle[1].style.color='red';

      }

      }

  functionclearData(){

    varpwd=document.querySelectorAll('input[type]');

    varpEle=document.querySelectorAll('p');

    pwd[0].value='';

    pwd[1].value='';

    pEle[0].style.color='red';

    pEle[1].style.color='red';

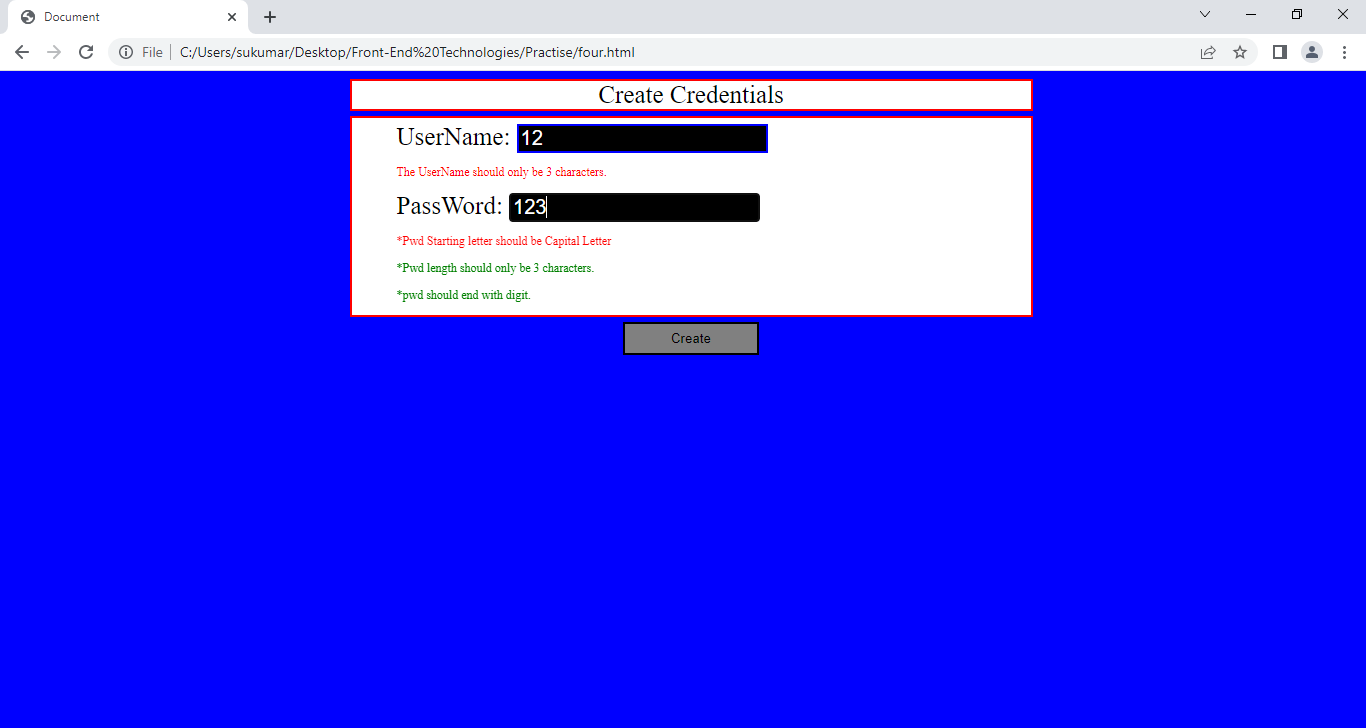
    pEle[2].style.color='red';

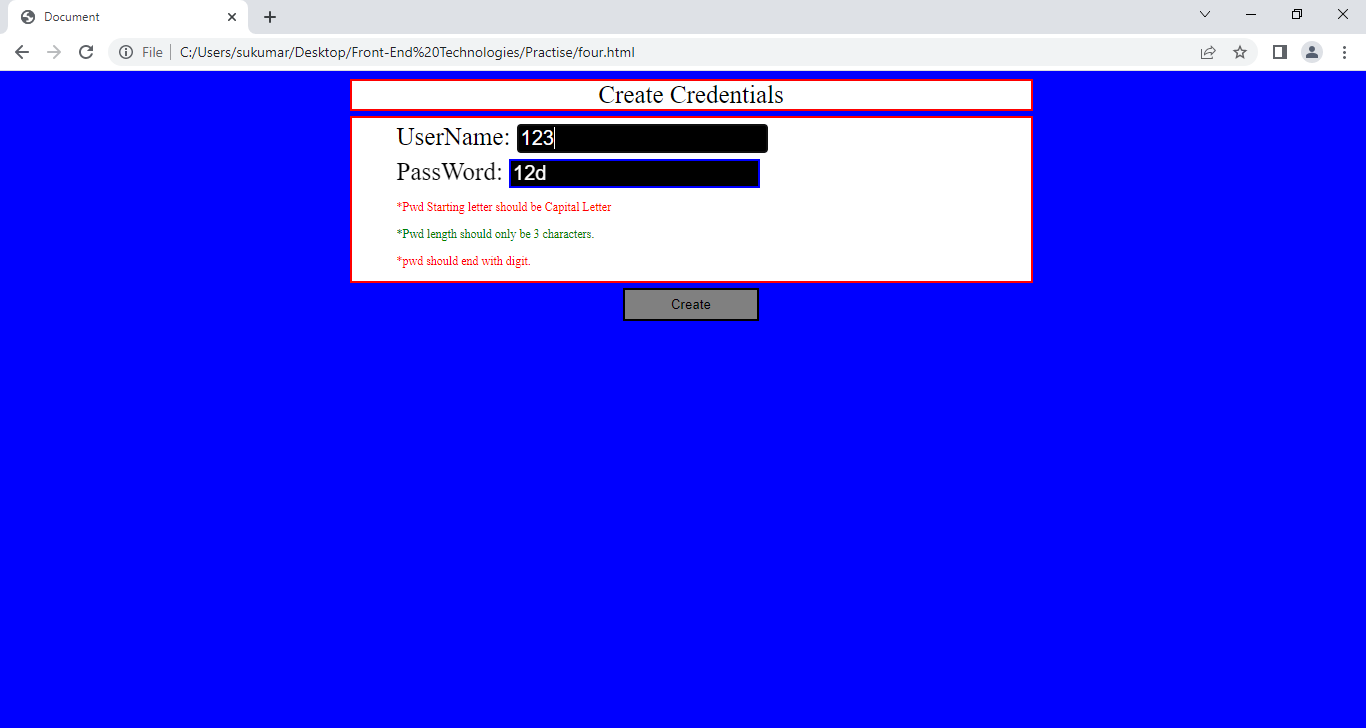
  }

  </script>

</body>

</html>





After pressing create button.

