javascript:

-----------

<https://www.youtube.com/watch?v=JVTZb4YvKjM>

**How to run javascript in visual studio code (VS)**

1st download visual studio code and install it.

2nd download Nodejs and install it.

Node installed or not checking process Goto CMD write command >**node - -version**  and check npm version >**npm - - version**

3rd create one folder take that folder location in visual studio code and there install sum tooles in visual code 1st is **Code Runner** and 2nd **JavaScript (ES6) code snippets**  right side you will get **Run button** (playbutton) writ the code and hit Run button code will be executed.

Ex:

Console.log(“hello”) // press run button you will get output.

**How to check out on Browser by using visual studio code**

If you want check output from browser using visual studio code in html component write **<script>** tag and write code in bitwin write **document.write();**

**EX:**

<script>

document.write(“hello”);

</script>

**datayptes:**

----------

**syntax:**

------

var a = 10

datatype variable value

var this is datatype it can store "string" value,"int" values and "Boolean" values for Exampul

Exp:

----

var a=50; // this holding (int)number data, we can able declare folat values also ex: var f=19.09;

var b="javascript"; // this is holding string data we can give single cots or double cots like Ex:(this is double cots) var a="javascript";

(this is single cots) var a='javascript';

there are 2 types of data types

---------------------------------

1.Primitive data types

2.No-Primitive data types

1.Primitive data types

\*Sting

\*number

\*Boolean

2.No-Primitive data types

\*Object

\*Array

\*RegExp

Identifier

----------

do's: donot //donot are not recomended first number and symbals are not recomended

----- ------

Score 8thplayer

$player1 Team\*

Player\_2

Ex:

var @variables // not recomened because symbals

var Score // this is valid

variables

----------

there is 4 types of variables

1.var

2.let

3.const

4.nothing

always declare javascript variables with var,let and const. is good

if you want change value in fucher use "let" variable, if you use "const" variable it will not work

for Exp:

---------

var car\_Name="tata";

const price=1000; /\* use this \*/ let price=1000; /\* it will work\*/

price++;

console.log(price);

output:

--------

You will get error why becase you using "const" you will not increase or change the value

Operators :

------------

there are diffrent types of javascript operators

Arithmetic Operators

Assignment Operators

Comparison Operators

Logical Operators

Conditional Operators

Arithmetic Operators

---------------------

+ addition

- Substraction

\* Multiplication

\*\* Exponentiation (ES2016)

/ Division

% Modulus(Division Remainder)

++ Increment

-- Decrement

Assignment Operators

----------------------

Operator Example Same As

= x = y x = y

+= x += y x = x + y

-= x -= y x = x - y

\*= x \*= y x = x \* y

/= x /= y x = x / y

%= x %= y x = x % y

\*\*= x \*\*= y x = x \*\* y

Comparison Operators

---------------------

Operator Description Comparing Returns

== equal to x == 8 false

x == 5 true

x == "5" true

=== equal value and equal type x === 5 true

x === "5" false

!= not equal x != 8 true

!== not equal value or notequaltype x !== 5 false

x !== "5" true

x !== 8 true

> greater than x > 8 false

< less than x < 8 true

>= greater than or equal to x >= 8 false

<= less than or equal to x <= 8 true

Logical Operators

------------------

Operator Description Example

&& and (x < 10 && y > 1) is true

|| or (x == 5 || y == 5) is false

! not !(x == y) is true

Example:-

Logical Operator Boolean Values

//logical AND Operator

Note:- two values must be true (in ower if condition both values must be ture then only your value print Other wise else Block will print)

true && true; //true

true && false; //fase

false && true; //false

false && false; //false

// logical OR operator

Note:- atlist one value must be true

true && true; //true

true && false; //true

false && true; //true

false && false; //false

//logical NOT operator

Note:- this NOT operator is Changing value "TREUE" TO "FALSE",(or) "FALSE" TO "TRUE" lets check Example

ex:

let x=5;

let y=9;

!(x===9)

x=5,y=9,boolean value is FALSE if you use NOT operatior it will become a TRUE.

this is changing value true to false and false to true

**Conditional Operators**

----------------------

----------------------------------------------------------

**Type Conversion:**

----------------

Type conversion is the process of convarting data of one type to another

For Example: convarting String data to Number

Thre are two types of conversion in javaScript

1.Implicit Convarsion - automatic type conversion

2.Explicit Convarsion - manual type conversion

**String:**

--------

length:To find the length of a string, use the built-in length properth.

charAt():Return the character at a specified index(position).

indexOf():Return the index(position) of the first occurrence of a value in a string.

toUpperCase():Return a string converted to uppercase laters

toLowerCase():Return a string converted to lowercase laters.

toString():Return a string or a string object as string.

\* Array of String:

program:

let a=["ram","laxman","bharth"];

console.log(a[0]);

console.log(a[2]);

o/p:-ram

bharth

-----------------------------------------------

**Control Statement:**

------------------

Control Statement are used to control the flow of execution of a program based on certain conditions

types of control statements:

-Decision-making/conditional statements

-Looping/iteration Statements

-jump statements

**Decision-making/conditional statements**

---------------------------------------

Conditional statements are used for a certain block of code needs to be executed when some condition is fulfilled.

-if

-if-else

-if-else-if ladder

-switch

the IF condition is true it will go to the IF body, incase IF condition is false it's going to the out of the IF condition.

< this is lessthen means SMALL\_VALUE Exp: 10<100 (true)

> this is greaterthen means BIG\_VALUE Exp: 1000>50 (true)

**if Program:**

-----------

if(10<20){

console.log("20 is big value");

}

-----------------------

**if-else:**

--------

The if statement evaluates the code if the condition is true, otherwise else statement tells the code what to do

**sysntax:**

if(condition){

// if condition is true code will execute, Other wise code move to the next section

}

else{

// code if condition is false this loop will executed

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

let a=100 , b=20;

if(a>b)

{

console.log("a is big");

}

else{

console.log("b is big");

}

**if-else-if laddr**

----------------

The if-else-if laddr statement same as if-else block but the main difference is in this else block will contain another if-else block.

syntax:

-------

if(condition1)

{

//code to be executed if condition1 is true

}

else if(condition2){

//code to be executed if condition2 is true

}

else{

//code to be executed if all the conditions are false

}

---------------------------------------------------------------

let a=10, b=200, c=30;

if(a>b){

console.log("a is big");

}

else if(c>b){

console.log("b is big");

}

else{

console.log("c is big");

}

----------------------------------------------

**Switch**

**----------**

switch statement is a multiway branch statement.

the expression is checked for different cases and the one match is executed.

\* Important Points to remember:

---------------------------

1) in switch duplicate case values are not allowed.

2) The data type of the variables in the switch and value of a case must be of the same type.

3) The value of a case must be a constant or a character. Variables are not allowed.

4) The break in switch statement is used to terminate the current sequence.

5) The default statement is optional and it can be used any where inside the switch statement.

6) Multiple default statements are not allowed

**sysntax:**

--------

switch(expression)

{

case value1:// statement sequence

break;

case value2://statement sequence

break;

.

.

.

case valueN:// statement sequence

break;

default:// default statement sequence

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Looping/Iteration Statements:**

------------------------------

Looping/Iteration Statements to execute a statement or a set of statements multiple times

The result condition should be true to execute statements within loops.

Here we use increment operator for keeping loop alive until condition gets false.

Iteration

|

|-while

|-do-while

|-for

**While loop:**

-----------

The test condition is given in the beginning of the loop and all statement are executed until the given Boolean condition becomes false.

Syntax:

-------

while(boolean condition)

{

loop statements...

}

**program**

----

let a=1;

while(a>5)

{

console.log("a="+a);

a++;

}

**do-while\_Loop:**

-------------

do while loop is similar to while loop with the only different that it check the condition after executing the

statements, i.e it will execute the loop body one time for sure bacause it checks the condition after executin the statements.

**syntax:**

**-------**

do

{

statements...

}while(condition);

pro

---

let a=5;

do

{

console.log("a="+a);

a++;

}while(a<15);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

**For Loop:**

----------------

for loop has similar functionality as while loop but with different syntax. for loops

are preferred when the number of times loop statements are to be executed is know beforedhand.

**systax:**

-------

for(loop variable initialization; testing condition;increment/decrement)

{

// statement to be executed

}

**programm**

----

for(a=1;a<=50;a++)

{

console.log("page no is" + a);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Jump Statements**

**----------------**

Jump Statements are used to trasfaer control form one point to another point

Jump

|

|-Break

|-Continue

**Break**

------

The break statement is used to terminate the loop or statement in which it present

**Continue**

---------

The statement is used to skip over the execution part of the loop on a certain condition.

**program: (Continue and Break) :**

--------------------------------

for(a=1;a<=50;a++)

{

if(a==5)

{

console.log("continue statement occured");

continue;

}

if(a==10)

}

console.log("break statement Occured");

break;

}

console.log("page no is" + a);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Local and Global and Block scopes:**

----------------------------------

Scope determines the accessibility(visibility) of varibles.

In computer programming, the scope of a name binding is the part of a program where the

name bindingis valid; thatis, where the name can be used to refer to the entity.

**scopes**:

--------

JavaScript has 3 types of scope:

1.Block scope

2.Global scope

3.Function/Local scope

**1.Block scope**

---------------

\* Before ES6(2015),javaScript had only GLobal Scope and function Scope.

\* ES6 introduced two important new javaScript keywords: let and const.

\* Thest two keywords provide Block Scope in JavaScript.

\* Variables declared inside a {} block connot be accessed from outside the block.

\* Variables declared with the var keyword can NOT have block scope.

\* Variables declared inside a {} block can be accessed from outside the block.

**Block Scope Example:**

--------------------

**Program:**

----

{

let x=2;

}

// x can NOT be used here

a=9;

if(a>3){

console.log("in");

let b=20;

}

console.log(b);

**Ex:-2**

{

var x=2;

}

// x CAN be used here

**Local/Function Scope:**

---------------------

\* variables declared within a javaScript function, become LOCAL to the function.

\* When you use javascript, local vaiables are variables that are defined within functions.

\* They have local scope, which means that they can only be used within the functions that define them.

\* Since local variables are only recognized inside their functions, variables with the same name can be used in diffrent functions.

\* Local variables are created when a function starts, and deleted when the function is completed.

\* Accessing them outside the function will throw an eror.

**Example:**

--------

// code here can NOT use carName

function myFunction(){

let carName="Volvo";

// code here CAN use carName

}

// code here can NOT use carName

Pro:

---

function fn()

{

var a=10;

console.log(a);

}

fn();

console.log("out side a value"+a);

**Global Scope:**

--------------

\* A variable declared outside a function, becomes GLOBAL.

\* Variables declared Globally (outside any function) have Global Scope.

\* Global variables can be accessed from anywhere in javascript program.

\* Variables declared with var, let and const are quite similar when delcared outside a block.

\* A Variable declared without a keyword is also considered global even though it is declared in function.

**Example:**

--------

let carName ="Volvo";

// code here can use carName

function myFunction(){

// code here can also use carName

}

**Program:**

----

a= 30

function fn(){

console.log("in side the function"+a);

}

fn();

console.log("out side a value"+a);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**FUNCTIONS:**

----------

**Topics:**

\*Functions

\*Function call

\*Return

**Functin:**

--------

A Function is a set of statements that take inputs,

do some specific computation, and produce output.

The idea is to put some commonly or

repeatdly done tasks together and make a function so that

instead of writing the same code again and again for diffrent inputs,

we can call that function.

**Function Rules:**

----------------

The rules for creating a function in javaScript:

=> Every function should begin with the keyword function.

=> Followed by, a user-defined function name that should be unique.

=> A list of parameters enclosed within parentheses and separated by commas.

=> A list of statements composing the body of the function enclosed within curly braces{}.

**FUNCTION CREATION:**

------------------

To create a function in javascript, we have to first use the keyword function,separated by name of the function and parameters within parenthesis.

The part of the function inside the curly braces{} is the body of the function.

**Syntax:**

-------

function name(parametr1,parameter2,parameter3)

{

//code to be executed

}

**FUNCTION CALL:**

--------------

After defining a function, the next step is to call them to make use of the function.

We can call a function by using the function name separated by the value of parameters

Enclosed between the parenthesis and a semicolon at the end.

**Syntax:**

-------

functionName(Value1,Value2,....);

**FUNCTION RETURN:**

----------------

There are some situations when we want to return some values from a function after

performing some operations.

In such cases,we can make use of the return statement in javascript.

when javascript reaches a return statement, the function will stop executing.

This is an optional statement and most of the time the last statement in javaScript function.

Function often compute a return value. The return value is "returned" back to the "caller".

**Syntax:**

-------

return value;

**Program:**

---

function first(c,d)

{

conlose.log("sum ="+(c+d))

return c+d;

}

const a=10,b=20;

const sum= first(a,b);

console.log(sum);

**ALERT, PROMPT, CONFIRM :**

-------------------------

JavaScript offers 3 types of popup boxes, they are:

1.Alert

2.Prompt

3.Confirm

**alert Box:**

-----------

An alert box is used when you want to ensure that information gets through to the user.

when an alert box pops up, the user must click OK to proceed.

The alert function takes a single parameter is the text display in the popup box.

**Syntax:**

-------

alert("message to show/parameters");

Be careful when using alert boxes, as the user can contine using the page only after clicking OK.

**Program** :(use this code in html page)

--------------------------------

<html>

<head>

</head>

<body>

<script>

var a="welcome to JS session";

alert(a);

document.write("Hello this is JS in HTML");

</script>

</body>

</html>

**PROMPT BOX:**

-----------

A prompt box is often used to have the user input a value before entering a page.

When a prompt box pops up, the user will have to click either OK of Cancel to proceed after

entering the input value.

If the user clicks OK, the box returns the input value. If the user clicks Cancel, the box retrns null.

The prompt() method takes two parameters.

-The first is the label, which you want to display in the text box.

-The second is a default string to display to display in the text box (optional).

**Syntax**:

-------

prompt("message to show");

when a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed

after entering an input value.

Do not overuse this method, because it prevents the user from accessing other parts of the

page until the box is closed.

Pro:

----

<html>

<head>

</head>

<body>

<script>

var a="welcome to JS session please Enter your Name";

var name=prompt(a);

document.write("Hello" +name+" this is JS in HTML");

</script>

</body>

</html>

**confirm box:**

------------

A confirm box is often used to have the user verify or accept something.

When a confirm box pops up, the user must click either OK or Cancel to proceeed.

If the user clicks OK, the box return true.If the user clicks Cancel,the box rerurn false.

**Syntax:**

-------

confirm("message to show");

Do not overuse this method, because it also prevents the user from accessing other parts of

the page until the box is closed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**OBJECTS**: time:3:33 start

-------

\*Objects

\*Object construction

**Objects:**

--------

JavaScript variables are containers for data values. Objects are variables too, but they cancontain many values.

Think of an object as a list of values that are written as name:value pairs, with the names

and the values separated by colons.

javaScript objects are containers for named values.

**Example:**

--------

var person = {name:"john",age:31,favColor:"green",height:150};

**Object Properties:**

------------------

**you can access object properties in two ways.**

objectName.propertyName (or) objectName['propertyName']

JavaScript's built-in length property is used to count the number of characters in a property or string.

objectName.propertyName.length

Object are one of the core concepts in JavaScript.

**program:**

----

var persion={name:"TEJS",age:15,class:10};

console.log(typeof(persion));// it tells what type of program, this line output id :- object

console.log(persion.name);

console.log(persion.age.length);

console.log(persion.age);

console.log(persion.class);

**Object Methods:**

---------------

An object method is a property that contains a function definition.

Use the following syntax to access an object method.

objectName.methodName()

document.write() output data. The write() function is actually a method of the document object.

Up to now we dissussed object using the object literal (or initializer) syntax.

var person={name:"Jhon",age:43,favColor:"green"};

This allows you to create only a single object.

**The Object Constructor:**

-----------------------

Sometimes, we need to set an "object type" that can be used to create a number of objects of

a single type.

The standard way to crate an "object type" is to use an object constructor function.

function person(name,age,color){

this.name=name; this.age=age; this.favColor=color;

}

The above function (person) is an object constructor, which takes parametrs and assigns

them to the object properties.

the this keyword refers to the current object. Note that this is not a variable.It is a

keyword, and its value cannot be changed.

**The Object Constructor:**

-----------------------

\* Once you have an object constructor, you can use the new keyword to create new objects of the same type.

\* var p1=new person("John",42,"green");

var p2=new person("Amy",21,"red");

\* p1 and p2 are now object of the person type. Their

properties are assigned to the corresponding values.

\* You can access value like

console.log(p1.name);

**Program:**

-----

function presion(name,age){

this.name=name;

this.age=age;

}

var p1=new persion("Ram",30);

var p2=new persion("Beem",24);

console.log(p1.name);

console.log(p1.age);

console.log(p2.name);

console.log(p2.age);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**METHODS:** time-3:53

---------

Methods are functions that are stord as object properties.

Syntax:

create an object method:

methodName = function(){

code lines

}

Access an Object method:

ObjectName.methodName()

A method is a function, belonging to an object. it can be

refered using the this keyword.

The this keyword is used as a reference to the current object,meaning that you can access

the objects properties and methods using it.

Note:

-----

changename => this is method

function => this is keyword

Pro:

----

var person={name:"ram",age:23,changename: function cn(a){this.name=a; console.log("fn call")}

console.log("before method call");

console.log(person.name);

console.log("after method call");

person.changename("laxman");

console.log(person.name);

Method Define:

--------------

Defining methods is done inside the constructor function.

Example:

--------

function person(name,age){

this.name=name;

this.age=age;

this.changeName=function(name){

this.name=name;

}

}

var p=new person("David",21);

p.changeName("john");

document.write(p.name);

--------------------------------------------------------

Pro:

----

function person(name,age)

{

this.name=name;

this.age=age;

this.changename=function cn(name){

this.name=name;

}

}

var p1=new person("ram",10);

console.log(p1.name+ p1.age);

console.log("before method call");

console.log(p1.name);

console.log("after method call");

p1.changename("laxman");

console.log(p1.name);

---------------------------------------------------------

Methods Define:

---------------

You can also define the function outside of the constructor and associate it with the object.

Example:

--------

function person(name,age){

this.name=name;

this.age=age;

this.yearOfBirth = bornYear;

}

function bronYear(){

return 2016.this.age;

}

-----------------------------------------------------------

The this keyword is used to access the age property of the object, which is going to call the method.

Note: it's not necessary to write the function's parentheses when assigning it to an object.

Pro:

----

function person(name,age)

{

this.name=name;

this.age=age;

this.changename= cn;

}

function cn(name){

this.name=name;

}

var p1=new person("ram",10);

console.log(p1.name+ p1.age);

console.log("before method call");

console.log(p1.name);

console.log("after method call");

p1.changename("laxman");

console.log(p1.name);

----------------------------------------------------------

Method Call:

------------

Call the method by the property name you specified in the constructor function,

rather then the function name.

Example:

var p=new person("A",33);

document.writer(p.yearOfBirth());

Pro:

----

function person(name,age)

{

this.name=name;

this.age=age;

this.yob= yob;

}

function yob(age){

return 2022-this.age;

}

var p1=new person("ram",10);

console.log(p1.name+ p1.age);

console.log("before method call");

console.log(p1.name);

console.log("after method call");

p1.changename(p1.yob());

console.log(p1.name);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ARRAYS time-4:12

-------

An array is a special variable, which can hold more then one value.

Arrays store multiple values in a single variables.

Syntax:

-------

var variable\_name = naw Array("e1","e2",....."en");

Example:

--------

var courses = new Array("HTML","CSS","JS");

This syntax declares an array named courses, which stores three values or elements

Pro:

----

var name= new Array("ram","Beem","alluri");

console.log(name);

Accessing an Array:

-------------------

you refer to an array element by referring to the index number written in square break.

This statement accesses the value of the first element in courses and changes the value of

the second element.

Example:

--------

var cources = new Array("HTML","CSS","JS");

var cources = courses[0];//HTML

courses[1] = "C++";// Changes the second element

[0] is the first element in an array.[1] is the second. Array indexes start with 0.

Pro:

----

var name= new Array("ram","Beem","alluri");

console.log(name);

console.log("array is 2nd element " +name[1]);

name[2]="Laxman";

console.log(name);

Arrays:

-------

Attempting to access an index outside of the array, return the value undefined

You can also declare an array, tell it the number of elements it will store, and add the

Element later Like:

--------------------

var courses = new Array(3);

courses[0] = "HTML";

courses[1] = "CSS";

pro:

----

var a =new Array(10);

a[0]=4;

a[1]=30

a[2]=80;

console.log(a);

console.("using loop");

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

console.log(a[i]);

}

An array is a special type of object.

An arrays uses numbers to access its elements, and an object uses names to access its members.

you can access and modify the elements of the array using the index number.

Arrays Methods:

---------------

\* The length Property

----------------------

-> JavaScript arrays have useful built-in properties and methods.

-> An arrays lenth property return the number of it's elements.

Example:

--------

array\_name.length

The length property is always one more then highest arrays index.

if the array is empty,the length property returns 0.

Pro:

----

var a =new Array(4);

a[0]=4;

a[1]=30

a[2]=80;

a[3]=40;

console.log(a);

console.("using loop");

/\*

for(i=0;i<3;i++){ // onley 3 elements will print why because using 3 in loop

console.log(a[i]);

}

console.log("array lenth is " +a.length);

\*/

for(i=0;i<a.length;i++){ //insted of using number is loop use array\_value and length(EX:- a.length)

console.log(a[i]);

}

console.log("array lenth is " +a.length);

Combaining Arrays:

------------------

JavaScript's concate() method allows you to join arrays and create an entirely new array.

Example:

--------

var c1=["HTML","CSS"];

var c2=["JS","C++"];

var cources = c1.concat(c2);

console.log(courses[2]);

Pro:

----

var a=new Array("1","2");

var b=new Array("3","4");

//console.log(a,b);

console.log("a value " +a);

console.log("b value " +b);

var c=a.concat(b);

console.log("c value " +c);

Poping and Pushing:

-------------------

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pusing is:

Popping items out of an array, or pushing itemss into an array,

The pop() method removes the last element from an array.

The push() method adds a new element to an array(at the end).

push()\_program: (push is adding element, it is added last of array)

--------------

var movie=["varadu","vasu","vana"]

movie.push("varsham")

console.log(movie)

pop()\_program: (pop revome last element of array)

------------

var movie=["varudu","vasu","vana"]

movie.push("varsham")

console.log(movie)

movie.pop(); or use this line // console.log(movie.pop())

console.log(movie)

unshift : (unshift element is add in 1st of an array)

---------

var movie=["varudu","vasu","vana"]

movie.push("varsham")

console.log(movie)

movie.unshift("vernaCar")

console.log(movie)

shift : (shift it will remove 1st of an array)

-------

var movie=["varudu","vasu","vana"]

console.log(movie)

//movie.shift() // it will work but if you want removed item name write below line of code

var remn=movie.shift()

console.log("removed name is:" +" " + remn)

console.log(movie)

indexOf()

----------

var movie=["varudu","vasu","vana"]

console.log(movie)

console.log("indexOf movie " +movie.indexOf("vana"))

clice() :

---------

var movie=["varudu","vasu","vana","vedam"]

console.log(movie)

console.log(movie.slice(0,2))

//console.log("0 to 3-1" + movie.slice(0,3))

Pro:

----

var a=new Array("1","2");

var b=new Array("3","4");

console.log("a value " +a);

a.pop();

console.log("a value after pop()\n" +a);

a.push("helloe");

console.log("a value after push\t " +a);

console.log("b value " +b);

Array Delete:

-------------

Array elements can be deletd using the JavaScript operator Delete.

Example:

--------

delete array\_name[index];

Using delete leaves undefined holes in the array.

Using pop() or shift() insted.

Pro:

----

var a=new Arra("1","2");

console.log("a value " +a);

a.pop()

console.log("a value after pop()\n" +a);

a.push(3)

a.push(4)

console.log("a value after push()\t" +a);

delete a[0];

console.log("a value after delete\t" +a);

console.log(a);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATH AND DATA OBJECS: time:4:42

----------------------

Math Object:

------------

The Math Objectt allows you to perform mathematical tasks,

and includes several properties.

Unlike other objects, the Math object has no constructor.

The Math object is static,

All method and properties can be used without creating a Math object first.

The syntax for Math any method is: Math.method(number)

Number to Integer

There are 4 common methods to round a number to a integer:

Math.round(x) Returns x rounded to its neaest integer

Math.ceil(x) Returns x rounded up to its nearest integer

Math.floor(x) Returns x rounded down to its nearest integer

Math.trunc(x) Returns the integer part of x (new in ES6)

Pro:

----

var a=20.5;

console.log(Math.round(a));

console.log(Math.cell(a));

console.log(Math.floor(a));

console.log(math.trunc(a));

Math Methods:

-------------

--------------------------------------------------------

Method | Description

--------------------------------------------------------

abs(x) Returns the absolute value of x

acos(x) Returns the arccosine of x, in radians

asin(x) Returns the arcsineof x,in radians

atan(x) Returns the arctanget of x as a numeric value between -pI/2 and PI/2 radians

atan2(y,x) Returns the arctangent of the quotient of its arguments

ceil(x) Returns x, rounded upwards to the nearest integer

cos(x) Returns the cosine of x(x is in radians)

exp(x) Returns the value of EX

floor(x) Returns x, rounded downwards to the nearest integer

log(x) Returns the natural logarithm (base E) of x

max(x,y,z...n) Returns the number with the highest value

min(x,y,z,...n) Returns the number with the lowest value

pow(x,y) Returns the value of x to the power of y

random() Returns a random number between o and 1

sin(x) Returns the sine of x(x is in radians)

sqrt(x) Returns the square root of x

tan(x) Returns the tangent of an angle

Math Properties(Constants)

--------------------------

Syntax: Math.property

JavaScript provides 8 mathematical constants that can be accessed as Math properties:

Math.E // return Euler's number

Math.PI // return PI

Math.SQRT2 //return the square root of 2

Math.SQRT1\_2 // return the square root of 1/2

Math.LN2 // return the natural logarithm of 2

Math.LN10 // return the natural logarithm of 10

Math.LOG2E // return base 2 logarithm of E

Math.LOG10E // return base 10 logarithm of E

Data Object:

------------

The Data Object enables us to work with dates.

A date consists of a year, a month, a day an hour, a minute, a second, and milliseconds.

Using new Data(), create a new data object with the current date and time

Example:

--------

var d=new Data();

//d stores the current data and time

Pro:

----

var d=new Date();

console.log(d);

Data Object are created with the new Date() constructor.

There are 9 ways to create a new date object:

new Date(date String)

new Date(year,month)

new Date(year,month,day)

new Date(year,month,day,hours)

new Date(year,month,day,hours,minutes)

new Date(year,month,day,hours,minutes,seconds)

new Date(year,month,day,hours,minutes,seconds,ms)

new Date(milliseconds)

The Other ways to initialize dates allow for the creation of new date objects from the specified date and time

Example:

--------

new Date(milliseconds)

new Date(dataString)

new Date(year,month,day,hours,minutes,seconds,milliseconds)

JavaScript dates are calculated in milliseconds from 01 january,1970 00:00:00 Universal Time (UTC).

One day contains 86,400,000 millisecond.

JavaScript counts months from o to 11.january is 0, and December is 11.

Data Objects are satic, rather then dynamic. The computer time is ticking, but date objects

don't change once created.

Data Methods:

-------------

When a Date object is created, a number of methods make it possible to perform operations on it.

Syntax:

--------

object.method();

Method | Description

----------------------------------------------

getFullYear() gets the year

getMonth() gets the month

getDate() gets the day of the month

getDay() gets the day of the week

getHours() gets the hour

getMinutes() gets the minutes

getSeconds() gets the seconds

getMilliseconds() gets the milliseconds

Pro:

----

var dt=new Date();

console.log(dt.getDate());

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

JAVASCRIPT IN BROWSER (HTML) time:5:1

-----------------------------

=> JavaScript in Brower (HTML) :

=> Html tag?

=> Script tag?

=> How to import JS file into html?

Browser:

--------

Brower is a computer program with a graphical user interface for displaying and navigating

between web pages.

it need to be connected to intenet to load web pages from www.

it used for assessing web pages build on various languages like:

HTML,CSS,JavaScript, and Java,PHP,Python...etc.

Popularly known browser are:

Google chrome,Microsoft Edge,Safari,Mozilla Firefox,Opera,Internet Explorer...etc.

HTML tag:

----------

HTML(Hyper text Markup Language) is the standard language for documents designed to be

displayed in a web browser.

HTML is at the core of every web page. Behind every web page you've ever visited there is HTML

code that your web browser uses to display elements like text,images and tables.

An HTML document can include different elements such as text headings, paragraphs, images or tables.

These elements are defined using different tags.

The basic Strucher of any HTML document is as follows:

------------------------------------------------------

<!DOCTYPE html>

<html>

<head>

<title></title>

</head>

<body>

<h2> your code goes here for web structure</h2>

</body>

</html>

There is no things in html to perform operations like add, sub...etc.

so we need to add JavaScript to our html page to perform these action

Script Tag:

-----------

The <script> tag is used to embed a client-side script(JavaScript).

The <script> element either contains scripting statements, or it points to an external script

file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

We use document object to print/write text to browser's page(this the only difference between console and web).

JavaScript statements using script tag:

----------------------------------------

syntax:

-------

<body>

<script>

<h1>your code goes here for Js</h1>

document.write("hello");

</script>

</body>

Pro:

-----

<html>

<head>

<title>

</title>

</head>

<body>

<h1>Hello</h1>

<script>

var a=parseInt(prompt("enter 1st number"));

var b=parseInt(prompt("enter 2st number"));

document.write("addition of 2 numbers " +(a+b));

</script>

</body>

</html>

Script Tag:

------------

Importing JavaScript file using script tag:

--------------------------------------------

Syntax:

<script type="text/javascript" src="script.js"></script>

----------------------------------------------------------------------------------------

|Attribute | Value | Description

-----------------------------------------------------------------------------------------

src URL Specifies the URL of an external script file

type scripttype Specifies the media type of the script

------------------------------------------------------------------------------------------

DOM (data object Model) time:5:20

------------------------

When you open any webpage in a browser, the HTML of the page is loaded and rendered

visually on the screen.

To accomplish that, the browser builds the Document Object Model of that page,

which is an object oriented model of its logical structure.

JavaScript can used to manipulate the DOM of a page dynamically to add, delete and modify elements.

DOM Tree:

---------

The DOM represents a document as a tree structure.

HTML elements become interrelated nodes in the tree.

All those nodes in the tree have some kind of relations among each other.

Nodes can have child nodes. Nodes on the same tree level are called siblings.

It is important to understand the relationsships between elements in an HTML document

in order to be able to manipulate them with javaScript.

document Object:

----------------

There is a predefined document object in JavaScript,

which can be used to access all elements on the DOM.

For example:

------------

document.body.innerHTML="some text";

As body is an element of the DOM, we can access it using the document object

and change the content of the innerHTML property.

The innerHTML property can be used on almost all HTML elements to change its content.

Pro:

----

<html>

<head>

<title>

Demo program

</title>

</head>

<body>

<div id="demo">

<h4>heading 4</h4>

</div>

<div id="display">

<p class="paragraph"> paragraph1</p>

<p class="paragraph"> paragraph2</p>

</div>

<script>

document.body.innerHTML = "some text";

</script>

</body>

</html>

Selectin elements:

------------------

All HTML elements are objects. And as we know every object has properties and methods.

The document object has methods that allow you to select the desired HTML element.

These three methods are the most commonly used for selecting HTML elements:

//finds element by id

document.getElementById(id)

//finds elements by class name

document.getElementByClassName(name)

//finds elements by tag name

document.getElementByTagName(name)

The getElementById method is used to select the element with id="demo" and change its content:

//by using ID

var elem=document.getElementById("demo");

elem.innerHTML = "Hello World";

The getElementsByClassName() method return a collection of all elements(in an aray format) in the

document with the specified class name.

//by using ClassName

var arr= document.getElementsByClassName("demo");

arr[1].innerHTML = "Hi";// accessing the second element

Similarly, the getElementByTagName method returns all of the elements of the specified tag name as an array

//by using TagName

var arr=document.getElementsByTagName("p);

arr[0].innerHTML ="Hi there";

Pro:

----

<html>

<head>

<title>

Demo program

</title>

</head>

<body>

<div id="demo">

<h4>heading 4</h4>

</div>

<div id="display">

<p class="paragraph"> paragraph1</p>

<p class="paragraph"> paragraph2</p>

<p class="paragraph"> paragraph3</p>

</div>

<script>

// document.body.innerHTML = "some text";

// by using ID

// var elem=document.getElementById("demo");

// elem.innerHTML ="this is call by id";

// by using ClassName

//var arr=document.getElementByClassName("paragraph");

//arr[1].innerHTML = "this is call by class name";

// by using TagName

var arr = document.getElementsByTagName("p");

for (var x=0; x < arr.length; x++){

arr[x].innerHTML = "Hi there is call by tag name";

}

</script>

</body>

</html>

Working with DOM:

------------------

Each element in the DOM has a set of properties and methods that provide information about their relationships in the DOM:

element.childNodes : rerurns an array of an element's chaild nodes.

element.firstChild : returns the first child node of an element.

element.lastChild : returns the last child node of an element.

element.hasChildNodes : rerurns true if an element has any child nodes, otherwise false.

element.nextSibling : returns the next node at the same tree level.

element.previousSibling : return the previous node at the same tree level.

element.parentNode : returns the parent node of an element.

Pro:

-----

<html>

<head>

<title>

Demo program

</title>

</head>

<body>

<div id="demo">

<h4>heading 4</h4>

</div>

<div id="display">

<p class="paragraph"> paragraph1</p>

<p class="paragraph"> paragraph2</p>

<p class="paragraph"> paragraph3</p>

</div>

<script>

var a=document.getElementById("display");

var ary= a.childNodes;

for(var x=0; x < ary.length;x++){

arry[x].innerHTML = "new text this is call by child nodes";

}

</script>

</body>

</html>

DOM Events:

------------

HTML DOM allows JavaScript to react to HTML events.

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute.

Common HTML Events:

--------------------

Here is a list of some common HTML events:

Event | Description

-------------------------------------------------

onchange An HTML element has been changed

onclick The user clicks an HTML element

onmouseover The user moves the mouse over an HTML element

onmouseout The user moves the mouse away from an HTML element

onkeydown The user pushes a keyboard key

onload The browser has finished loading the page

Pro:

---

<html>

<head>

<title>

event

</title>

</head>

<body>

<h2>JavaScript HTML Event</h2>

Enter your Name: <input type="text" id="fname" onchange="upperCase()">

<p>When you leave the input field, a function is triggered which transforms the input text to upper case</p>

<script>

function upperCase(){

const x = document.getElementById("fname");

x.value = x.value.toUpperCase();

}

</script>

<hr style="padding: 10px;border: 10px;">

<div onmouseover="mOver(this)" onmouseout="mOut(this)"

style="background-color:red;width:120px;height:20px;padding:40px;">

Mouse Over Me</div>

<script>

function mOver(obj){

obj.innerHTML = "Thank You"

obj.style.backgroundColor ="green";

}

function mOut(obj){

obj.innerHTML = "Mouse Over Me"

obj.style.backgroundColor = "red";

}

</script>

<hr style="padding:10px;border:0px;">

<div onmousedown="mDown(this)" onmouseup="mUp(this)"

style="background-color:black;color:white; width:90px;height:20px;padding:40px;">

Click Me</div>

<script>

function mUp(obj){

obj.style.backgroundColor = "blue";

obj.innerHTML = "Thank you";

}

</script>

</body>

</html>

js\_short cuts of vs

--------------------

ctrl+k+c // all commests

ctrl+alt+n // run the code

lesstheb & greaterthen:

-----------------------

< this is lessthen means SMALL\_VALUE Exp: 10<100 (true)

> this is greaterthen means BIG\_VALUE Exp: 1000>50 (true)

NOTE:

-----

<div> what is div tag

Table tad:=> border=2, callpadding=3, cellspacing=3, rowspan=2,

colspan=2, <caption></caption>,<dt></dt>,<dd></dd>

Audio and vidoe tags in HTML: =>

Exp:1

<audio controls>

<soruce src="C:\user\.........location.mp3"/>

</audio>

Exp:2

<audio controls autoplay>

<soruce src="C:\user\.........location.mp3"/>

</audio>

Exp:1 video: (same as audio but give .mp4):

-------------------------------------------

<video controls>

<soruce src="C:\user\.........location.mp4"/>

</video>

Exp:2

<video controls width =200px hight=200px>

<soruce src="C:\user\.........location.mp4"/>

</video>

brower object model is pending time= 6:00:32

javascript\_Consepts

---------------------

Variables

Data types

Operators

Arrays

Loops

Conditional Statements

Functions and Function Calls

Event and Event Handling

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

(**javascript**) from youtub changel name CSworld telugu

(videos-12)

**Arrays**

Why arrays

How to declare arrays

How to access arrays elements

How to update/add arrays elements

Methods on arrays

**Why arrays**

**How to declare arrays**

There are 2 type to declare array

Var movies = [‘rrr’,’bahubali’,’arya’] ; // this is **1st way** Ex: var movies = [];

Var movies = new array (‘rrr’,’bagubali’,’arya’); //this is **2nd way** Ex: var movies = **new arrar();**

**How to access arrays elements**

Var movies = [‘rrr’,’bahubali’,’arya’] ;

Conlose.log(movies[1]); //output is bahubali why bacase index start with 0 // this is accessing particular elements in array

**Note**: if you give outofthe index number it will not show error it will show **undefined** lets check example

Console.log(movies[6]); //outoput is undefined.

// arrays store in number elements also those are called **Heterogeneous** values

Ex: var movies = [‘rrr’,’bahubali’,’arya’, 200 ]; // number also added

Conlose.log(movies);

// we want check array length

Ex: var movies = [‘rrr’,’bahubali’,’arya’, 200 ];

Conlose.log(movies.length);

**Arrays Methods**

Methods on arrays

**Push()**

**Pop()**

**Unshift()**

**Shift()**

**indexOf()**

**slice()**

**Push()**

If you want add last element of the array use **push()** method lets check example.

EX: Var movies = var movies = [‘rrr’,’bahubali’,’arya’];

Movies.**push**(‘pushpa’); //end of the array element pushpa is added by using **push(**) method.

Console.log(movies);

**Pop()**

If you want to remove last element form array use **pop()** method. lets check example.

EX: var movies = [‘rrr’,’bahubali’,’arya’]; // arya is will remove by using pop() method that is last of the array element.

Movies.**pop**(‘arya’) //by using pop() method **last** of the **array** **element** is **removed**, that last element is **arya**

Console.log(movies)

**Unshift() //** (It will add 1st element in array line)

This Unshift() method is in over array list it will add first element. //It is exactly apposite as push() method.

EX:

Var movies = [‘rrr’,’bahubali’,’arya’];

Movies.unshift(“Pushpa”); //adding 1st element in array.

Console.log(movies)

**Shift()**

//it will remove 1st element of array

EX:

Var movies = [‘rrr’,’bahubali’,’arya’];

Moies.shift();

Console.log(movies);

**indexOf()**

// if you want index values in array list use this method

Var movies = [‘rrr’,’bahubali’,’arya’];

Console.log(“movies index : ” + Moies.indexOf(“**arya**”));

// if you give unkone value it will return **–1** lets check below

Console.log(“movies index : ” + Moies.indexOf(“**ya**”)); // it will return -1 value.

**Slice()**

// if you want to use **particular** element in array use **slice() method** but you need to give **FROM** index value to **TO** index value

Var movies = [‘rrr’,’bahubali’,’arya’,’pushpa’,’ram’,’beem’];

Console.log(movies.slice(2,4)); //it will print 2nd to before 4th value

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Functions**

* What is Function
* Why functions
* How to create functions
* How to call function
* What is function expression

**What is Function ? //**group of statement oka work I class those are called functions.

A block of code written to do a particular task

**What is Block**

* Anything that enclosed between {}

**EX:**

{

Statement1;

Statement2;

Statement3;

}

Q) different between loops and functions?

A) loops will repeatle flow continues but function you can use any ware in application

EX: if we are printing 23 time loop that is … Hello in console. In middle of loop we can print function or we can print any ware in application.

asume this Foorloop Console.log(“Hello”)

Console.log(“Hello”)

**//here we can use functions**

Console.log(“Hello”)

**//here we can use functions and any were we can use function.**

Console.log(“Hello”)…etc

// in function we can print middle of hello also we can use function.

**Why functions**

Why functions means Code reusability purpose we use functions .

**How to create functions/ Function definition:**

**Systax:**

function functionName(parameter1,parameter2,…..parameter n){

body

return value; //after performing body operation want to send other use **return** it may be variable or value what ever it.

**NOTE**: **return** statement and **parameter** are optional.

}

**ProgramEX: //** if we write a html page write <script> tag

<script>

function **sum**(num1,num2){

Var **result**=num1+num2

return **result**;

}

Console.log(**sum**(20,30)) //save it copy path location and past it in browser check inspect console.

document.write(**sum** (200,300)) // cheek in browser.

</script>

**How to call function**

functionName(parameters)

Ex: sum(7,8)

**What is function expression**

Assigning function to a **variable**

EX:

**Add** = function sum(num1,num2){

var result=num1+num2

return result;

}

**programEx:**

var **Add** = function sum(num1,num2){

var result=num1+num2

return result;

}

var res = add(20,45)

document.write(res)

------------------------------------------------------------------------------------------------------------------

**Type of functions** (video-14)

Types:

* Named functions
* Anonymous function
* Immediately Invoked function expression(iife)
* Arrow function

**Named Functions**

Named function:

**Function functionName**(parameter1,parameter2…..parameter n){

Body

Return value;

}

**EX:**

function sum(num1,num2){

Var result = num1+num2

Return result;

}

Console.log(sum(30,50));

**Anonymous function:**

**//**named function we will give name but Anonymous functions we are **not providing name**.

**Syntax**:

function(parameter1,parameter2…parameter n){ //we are not providing Name **only we gave function.**

Body

Return value;

}

**Note:** we cannot write anonymous function directly we have to assign to a **variable**

EX:

<script> //if you use html component use script tag.

Var **sum** = function(num1,num2){

Var result=num1+num2

Retrun result;

}

Document.write(sum(40,50));

</script>

**immediately Invoked function expression(iife)**

When we want to execute a function immediately where they Created, IIFE used.

**Syntax**:

**(**function definition**)** **(** **)**;

**Program:**

Var sum = **(**function add (num1, num2){ //iife assigned one variable

Var result = num1+num2

Return result

}**) (**40,50**)**; //function closed and declared values.

Document.write(sum)

**Arrow function:**

Without using function key word we can create function that is arrow function.

**Syntax:**

(parameter1,parameter2…parameter n) **=>**{

Body

Retrun value;

}

Program:

Var product = (num1, num2) => { //arrow function assigning to **variable** that is **product**

Var result = num1 \* num2

Return result;

}

Document.write(product(40,50));

-------------------------------------------------------------------- Program closed.

**In function:**

**1. If we have only one statement in function body.**

**Syntax:**

Var product = (num1, num2) => num1 \* num2

**Program:**

Product = (num1, num2) => num1 \* num2

Document.write(product(2,40))

**2. If only one parameter**

**Syntax:**

Var cube = num1 => num1 \* num1 \* num1

**// if you have one parameter write another ways also**

Var cube = (num1) => num1 \* num1 \* num1

**(Or)**

Var cube = (num1) => { return num1 \* num1 \* num1}

**Program**:

Cube = num1 => num1 \* num1\* num1

Document.write(cube(2));

**3. If no argument**

**Syntax:**

Var greet =( ) => console.log(“Good Morning”);

(Or)

Var greet = \_ => console.log(“Good Morning”);

**Program:**

// greet = ()=>document.write(“Good morning”)

// greet = **\_**=>document.write(“Good morning”)// use **underscore** also no issu.

// greet = **\_**=>document.write(“Good morning”) document.write(“<br> welcome”)//more then one statement it will not work. So give Curly brackets. Lets try below example.

greet = **\_**=>**{**

document.write(“Good morning”)

document.write(“<br> welcome”)

**}**

greet()

--------------------------------------------- Program closed.

greet = **\_**=>**{**

document.write(“Good morning”)

document.write(“<br> welcome”)

//return 6 //we can return any thing.

Return 20+40

**}**

Var res = greet()

Document.write(“<br>” + res)

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Objects in Javascript:**

What is object : lets assume We can consider every item(thing) one object so if we want to call that Object that object have the Properties like lets take car, car has a color and wheel like that. And that object have action the we can consider Method.

**Object:**

\* Object is collection of elements in the form properties and methods. //in array we have **collection values** so like that **Object is collection elements** we have

//those elements either propertyes or method.

In that property it will be key value pairs

What is key value pairs let take car as example: **color is key** and **red is value**

Lets take one **Example** as USER:

Var user = {

Name:’smith’,

Email:’smith@gmail.com’,

Phone:77889966

//Login: function(){

//Console.log(“log sucrssfully”)

//}

};

**Ways to create Objects:**

**1. Object Literal**

**2. Using a new keyword with Object Constructor**

**3. Using a new keyword with a constructor function**

**4. Object.create() method**

**5. Classes //this will diccuss in class section.**

**1. Object Literal**

**Syntax:**

Let move = { } // create an empty object.

**Program:**

Let movie = { //**Object with properties (Key: value pairs)**

Name:”rrr”,

Director: ‘rajamouli’,

Relesase:2022

}

Console.log(movie); //save it check in web\_browser console.

**How to access Object values: // How to access Particular value fallow below step.**

we can access values by using keys in 2 ways //1st options **square barackets notation** and 2nd (**dat**) **.notation**

**1.** **Obj[“key”]** //**quotes are mandatory**

EX: movie[“name”] 🡪 valid

Movie[name] 🡪 Invalid

// in array we will give index but Object we need to give **key**

**Program:**

Let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022}

console.log(movie[“name”]); // name is Object **key**

**2. Obj.key**

Ex: movie.name

**Program:**

Let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022}

Console.log(movie.Director);

**How to add new properties to Object:**

//create empty Objet and later we can add elements or sum elements already existing also we can able to add element.

here also we can add properties by using keys in 2 ways

**1.** Obj[“key”] = value

EX: movie[“budget”] = “400 cr”

**Program:**

let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022}

movie[“budget”] = “400cr”

console.log(movie) //comment this line and above ling check below console line.

console.log(movie[“budget”] = “400cr”)

**2.** Obj.key = value

EX: movie.budget= “400cr”

**Program:**

let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022}

movie.budget = “400cr” //check this worked or not => console.log(movie.budget=”400cr”);

console.log(movie)

**How to update values of Object: //** in Object updating particular value

//take object **key** already given and write **value** what you want.

We can update values by using keys in 2 ways

**1.** Obj[“key”] = value

EX: movie[“budget”] = “500cr” //previous we have **300cr** now iam changing to **500cr**

**Program:**

let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022,budget:’300cr’}

movie[“budget”] = “500cr”

console.log(movie)

**2.** Obj.key = value

Ex: movie.budget = “500cr”

**Program:**

let movie = { Name:”rrr”, Director: ‘rajamouli’,Relesase:2022,budget:’300cr’}

movie. budget = “500cr”

console.log(movie)

**2. Using a new keyword with Object Constructor / Using new Operator Object Constructer:**

**let movie = new Object();**

movie.name = “rrr”;

movie.director=”rajamouli”;

**Program:**

<script>

let movie = **new Object();**

console.log(movie) //save it and check.

// Using **.notation** => in that movie we will assign budget key

movie.budget = “400cr” //adding budget key values.

console.log(movie) //save it and check.

//Using **array notation**

movie[“release”] = 2023

console.log(movie)

</script>

**3. Using a new keyword with a constructor function / Using new Operator with Constructor function:**

**Step1**: create constructor function :

Function user(name,age,place){

This.name = name;

This.age= age;

This.place = place;

}

**Step2**: create Object with constructor function call:

let user1 = new user(“abc”,44,”hyd”);

**Advantages is** => in this Object we can use number of times. Lets check if you want one more object Ex: let user2 = new user(“xyz”,55,”bang”); like that we can create

**Program**:

**Function user(n,a,p){**

This.name = n;

This.age = a;

This.place = p;

}

let User1 = **new user**(‘abc’,40,’hyd’);

console.log(user1);

// lets print one more Object and with out creating function method means same function method we create number of objects.

let User2 = new user(‘xyz’,50,’bang’);

console.log(user2);

//if you want to add key in that object we can use two ways 1st .notation and array notation lets check example.

User2.mobile = 99887766 //using .notation

Another way array notation

User2[“gender”] = “m”

Console.log(user2);

// If you want to access **particular value**

Console.(user.name); //particular name accessing

Console.(user.age); //accesssing age

**4. Object.create() method**

let movie2 = Object.create(movie);

another way

let movie3 = Object.create(movie, {

name: {

value:”RRR

},

Music:{

Value: “Keeravani”

}

})

**Program:**

Let user1 = new user(“abc”,25,”hyd”)

Let user3 = Object.create(user1)

Console.log(“user1:”)

Console.log(user1)

Console.log(“user3:”)

User3.name = “pqr”;

User3.age = 50;

Console.log(user3)

In Object we an create Method.

// in our Object we can add key value pairs and we can add functions also.

Program:

Function user(n,a,p){

This.name = n;

This.name = a;

This.name = p;

This.login = function() {

Console.log(“hello” + “ “+ this.name +” “+”logged in successfully”)

}

}

Let user1 = new user(“abc”,34,”hyd”)

Let user2 = new user(“xyz”,44,”bang”)

User1.login()

User2.login()

----------------------------Program close

What is different between function and Object inside function

Function we can use any ware in application but inside of functions take function referenc and use it

EX: user.login();

If we want etarate key and value In over Object using For-in loop (maximum)

Function user(n,a,p){

This.name = n;

This.name = a;

This.name = p;

This.login = function() {

Console.log(“hello” + “ “+ this.name +” “+”logged in successfully”)

}

}

Let user1 = new user(“abc”,25,”hyd”)

Let user2 = new user(“xyz”,33,”bang”)

//For(key in user1){

// Console.log(key,”:”,user1[key]

//}

Console.log(object.keys(user1) //this will show keys

Console.log(object.values(user1) //this will show values

Console.log(object.entries(user1) //this will show both key and values.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

DOM (Document Object Model) (video-21)

**What is DOM**

By using HTML DOM Javascript can access, change or remove any elements of HTML document and also can create new elements at any position.

When a webpage loaded, browser create DOM of webpage

**Document Object has**

\*Properties

\*Methods

**Using this document object properties & methods we can**

\*select HTML elements

\*Modify HTML elements

\*Remove/delete HTML elements

\*Create HTML elements

\*Add/Remove/Change style to HTML elements

**Method to select HTML Elements:**

**1. document.getElementById(“idname”)**

**Return element with the specified id.**

**2. document.getElementByClassName(“classname”)**

**Return list of all elements belongs to the specified class**

**3. document.getElementsByTagName(“tagname”)**

**Return list of all elements with the specified tag**

**4. document.querySelector(“.class/#id/tagname”)**

**Return the first object matching CSS Style selector.**

**5. document.querySelectorAll(“.class/#id/tagname”)**

**Returns all objects Matches the CSS Style Selector**

-------------------------------------------------------------------------------------------------------------

**1. document.getElementById(“idname”)**

**Return element with the specified id.**

**Program.**

**<body>**

<h1 **id=”first”**>This is h1 tag</h1>

<script>

Let x = document.getElementById(“first”)

Console.log(x)

</script>

**</body>**

**Program Ex\_2**

**<body>**

<div **id=”first”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<script>

Let x = document.getElementById(“first”) //comment this line lets check rong id given line.

Let x = document.getElementById(“fir”) // this is rong id so you will get null as output.

Console.log(x)

</script>

**</body>**

**2. document.getElementByClassName(“classname”)**

**Return list of all elements belongs to the specified class**

// if you give getElementsByClassName it will display all elements lets check.

**Program Ex\_1**

**<body>**

<div **class =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **class =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.getElementsByClassName(“special”)

Console.log(x)

</script>

**</body>**

// if you give **id** is multiple times it will take first div tag it will not go second tag.

**Program Ex\_2**

**<body>**

<div **id =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **id =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.getElementByClassName(“special”)

Console.log(x)

</script>

**</body>**

**3. document.getElementsByTagName(“tagname”)**

**Return list of all elements with the specified tag**

//this is also retrun list of elements.

**Program:**

**<body>**

<div **id =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **id =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.getElementBysTagName(“h2”)

Console.log(x)

</script>

**</body>**

**4. document.querySelector(“.class/#id/tagname”)**

**Return the first object matching CSS Style selector.**

**Program:**

**<body>**

<div **id =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **id =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.**querySelector(“#special”)** **//it will select first id**

Console.log(x)

</script>

**</body>**

**// by using .class also it will shows only first values.**

**Program \_ 2:**

**<body>**

<div **class =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **class =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.**querySelector(“.special”) //it will select first class**

// if you give h2 also it will show 1st element

Let x = document.**querySelector(“h2”)** **//this is also return first tag**

Console.log(x)

</script>

**</body>**

**5. document.querySelectorAll(“.class/#id/tagname”)**

**Returns all objects Matches the CSS Style Selector**

//it will return all things like it mabe .class , id , and tagnames

**Program \_ 1:**

**<body>**

<div **class =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **class =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.**querySelectorAll(“h2”) //it will return all h2 files.**

//lets check class file

Let x = document.**querySelectorAll(“.special”)** //it will return all div tags means all class files.

Console.log(x)

</script>

**</body>**

**Program \_ 2:**

**//** we are selecting id so lets give # and give id in div tags

**<body>**

<div **id =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **id =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

<script>

Let x = document.**querySelectorAll(“#special”) //it will return all div elements.**

Console.log(x)

</script>

**</body>**

**DOM properties to select HTML Elements:**

document.body // only body will give

document.head // only heads will give

document.title // only title will give

document.anchors // in our html document all anchors tags will select and give

document.forms // in our html document all forms will select and give

document.images // in our document all images will select and give

document.scripts // in our script all forms will select and give

**Program:**

**<body>**

<div **id =”special”**>

<h2>This is heading tag<h2>

<p> this is paragraph tag</p>

</div>

<div **id =”special”**>

<h2>This is heading tag2<h2>

<p> this is paragraph tag2</p>

</div>

//adding ancors tag

<a href=” “>link1</a>

<a href=” “>link2</a>

<script>

Console.log(document) //we are treated as document as object so lets print only object

Console.log(document.body) //it will return only body.

Console.log(document.anchors)// or ues this Console.log(document.links)

// lets print title

Console.log(document.title)

// lets print head

Console.log(document.head)

</script>

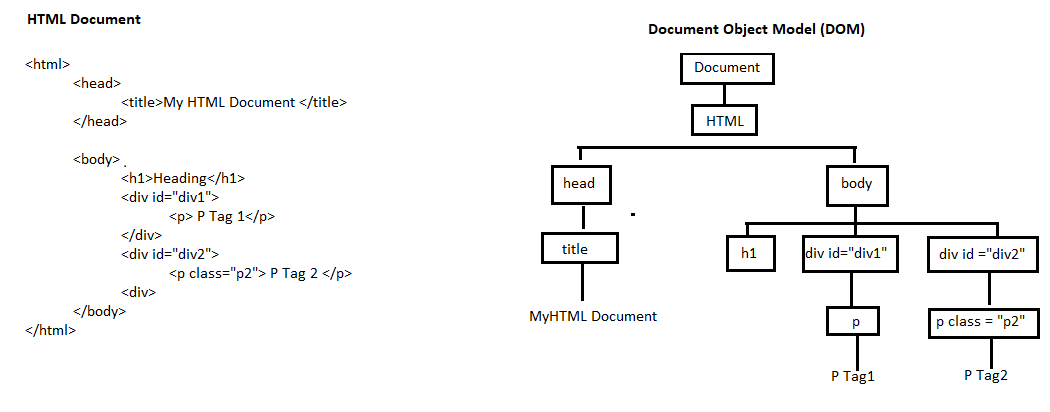
**</body>**

Q) What is different b/w method and property ?

1. Method you need to give parentheses and property you need to give only property name.

----------------------------------------------------------------------------------------------------------------------

**DOM Manipulation in javascript**  (video-22)



**Topics:**

**How to create HTML elements**

**How to Set content to element**

**How to append a new element**

**How to insert an element before another element**

**How to remove an element**

**How to remove child element**

**How to replace an element**

**How to create HTML elements**

<html>

<head>

</head>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p> this is first paragraph in division2 </p>

<p> this is second paragraph in division2 </p>

</div>

<script>

Var n = document.createElement(‘h1’) // Just h1 empty tag will show

Var n = document.createElement(‘p’) //here also we are printing empty paragraph element.

// Console.log(n)//save and check console. Just h1 empty tag will show.

//so if we want to add content to the element in javascript use **innerText** keyword and add text lets check.

Var n = document.createElement(‘p’)

n.**innerText** =”This is third Paragraph in divesion2”

console.log(n)

//taking second div element

Var parent = document.getElementById(“second”)

Console.log(parent) //printing second div element.

// Parent div element ki we created element adding as child by using appendChild() let’s check below.

Parent.appendChild(n)//save and execute it.

</script>

</body>

</html>

Q) What is difference b/w innerHTML and innerText lets see blow

a)inner HTML means it will print html tags and innerText means it will print only text lines.

<html>

<head>

</head>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p> this is first paragraph in division2 </p>

<p> this is second paragraph in division2 </p>

</div>

<script>

Var n = document.createElement(‘p’)

n.**innerText** =”This is third Paragraph in divesion2”

Var parent = document.getElementById(“second”)

Parent.appendChild(n)

//lets check innerText and innerHTML

// Console.log(parent.innerText) //only text line will print

Console.log(parent.innerHTML)//complete childe elements will printed along with tags.

</script>

</body>

</html>

**How to insert an element last before another element**

**Program**

<html>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p> this is first paragraph in division2 </p>

<p **id=”last”**> this is second paragraph in division2 </p>

</div>

<script>

Var n = document.createElement(‘p’)

n.innerText =”This is special Paragraph in divesion2”

Var parent = document.getElementById(“second”)

Var last = document.getElementById(“last”)

// by using insertBefore method it will print last before

parent.insertBefore(n,last) // **n** is our created variable and **last** is last element

</script>

</body>

</html>

**How to remove element**

So here we are removing second element.

**Program:**

<html>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p> this is first paragraph in division2 </p>

<p **id=”last”**> this is second paragraph in division2 </p>

</div>

<script>

**//removing second div tag inside second paragraph line**

Var last = document.getElementById(“last”)

Last.remove()

</script>

</body>

</html>

**How to remove child element / how to remove element child**

**Program:**

<html>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p **id=”pfirst”**> this is first paragraph in division2 </p>

<p **id=”last”**> this is second paragraph in division2 </p>

</div>

<script>

Var n = document.createElement(‘p’)

n.innerText = “This is special paragraph in division2”

var parent = document.getElementById(“second”)

var last = document.getElementById(“**pfirst**”)

console.log(last)

parent.removeChild(last)

</script>

</body>

</html>

**How to replace an element**

//I want to replace paragraph tag to hiding tag in second dive tag. By using **replaceChildren()** method.

**Program:**

<html>

<body>

<div id=”first”>

<p> this is first paragraph in division1 </p>

<p> this is second paragraph in division1 </p>

</div>

<div id=”second”>

<p **id=”pfirst”**> this is first paragraph in division2 </p>

<p **id=”last”**> this is second paragraph in division2 </p>

</div>

<script>

Var n = document.createElement(‘h1’)

n.innerText = “This is special heading in division2”

var last = document.getElementById(“**last**”)

last.replaceChild(n)

</script>

</body>

</html>

------------------------------------------------------------------------------------------------------------------------

**Asynchronous in javaScript** (video-46)

Before going to asynchronous we must know what is synchronous

Synchronous is line by line executing from top to bottom. Lets check example.

Program;

<html>

<head></head>

<body>

<script>

Console.log(“this is first line”)

Alert(“Hello”)

Console.log(“this is second line”)

//first first line will execute and second line will not print in middle alert box will come then second will print so this is line by line execution this is synchronous.

</script>

</body>

</html>

**asynchronous Program Example:**

<html>

<head></head>

<body>

<script>

setTimeout(function(){ //because of setTimeout it will wight 2000 mile seconds and in that time **next console will executed.**

console.log(“this is first line”)

},2000) //2000 is time mile seconds

Console.log(“this is second line”)

</script>

</body>

</html>

Program:2

<html>

<head></head>

<body>

<script>

test() // this is below test() function

console.log(“this is second line”) //this console line will not execute why because above test function is there so first test() function will executed. Then this line will executed.

function test(){ //first this line will executed why because above test() function is there.

Console.log(“this is first line”)

}

</script>

</body>

</html>

--------------------------------------------------------------------------------------------------------------------------------

**Call back function in javascript**  (video-34)

Call back function is a function that is passing as a argument to another function.

Oka function ni inko function ki argument kinda pass cheste a argument kinda pass chesina function ne call back function antaru.

Program:

<html>

<head></head>

<body>

<script>

Function test(a){

a() //must gibe paranthais and we called in call back function

}

test(function()){ //write argument as function it may be anonymous or arrow function that is call back function.

console.log(“call back function”) //this total function stored in **a** variable.

}

</script>

</html>

**Program:**

<html>

<head></head>

<body>

<script>

Function first(test){

test()

console.log(“this is first function”)

}

Function second(){

Console.log(“this is second function”)

}

**first(second)**

</script>

</html>

**Program:**

<html>

<head></head>

<body>

<script>

Function add(a,b){

Console.log(a+b)

}

Function product(a,b)

Console.log(a\*b)

}

Function calculator(val1,val2,operation){

Operation(val1, val2)

}

Calculator (4,5, product)

Calculator (4,5,add)

</script>

</html>

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Call back hell** (videos 49)

**What is call back hell**

**Program:**

<html>

<head></head>

<body>

<script>

</script>

</html>