Java script

* Java script is a Dynamic. It means interactive.
* Java is a both programming & scripting language
* By Java script we can develope various websites.

MVC structure.

Model veiw Controller

Front end libraries.

React Js, View Js, angular Js

* Front end developed on above libraries.
* Back end developes on Node Js
* who learn ~~x~~ Js (or) work on is known as MERN Developers

Mango DB, Express, React Js, Node Js

* Js Engine is used for convert user code to machine code in java.
* IDE's Integrated Development Editor
  some IDE's are vs code, Turbo c ... etc
* Javer and Java script are completely different
* Node Js is the Run time environment, it is a library (or) software
* Node Js software is used to convert Java code to binary code.

command prompt

* If we want to open ~~x~~ directly through file explorer. Then enter code cmd enter on file path section.

code ↓ space • enter

If we want open vs code through command prompt then follow code ↓ space • enter.

* if we want to run the code in vs code
They are to ways.
① Run option.
② Enter the following code in terminal section (node index.js)
* we can store the values in Js by using variables.
* Js has 3 key words.
Those are let, var, const
* By using these key word we declare variables (mandatory).
syntax key word some variable = "some data";

declaration
declaration means creating a variable with the help of key words. syntax key word variable;
Ex: let user;

Initialization
Initialization means assign values to created variable syntax created variab name = some data

* declaration & initialization can be done in two ways.
① declaration & initialization can be done in one by one
② declaration & initialization can be done in at a time
* Java script done only in 3 cases.
Snak_case - mean there is underscore blw two words
camelcase - mean 2nd word 1st letter should be capital
pascal case - mean 1st word 1st letter should be capital
* If we want print data on console with help of "console.log(" ")"
* let is a block-scoped variable, we can't recreate a variable
* var has drawback that can allows users to recreate a variable. (It has both redeclaration & re initialization)
* const keyword should be declared and initialized at time. It can't be recreated.

* Java script is debined from ATMA script core.
* Now we use ES 6 version it released on 2015.

Datatypes:

In Java script they are two types of datatypes.

1. primitive datatypes.
2. Non primitive / reference datatypes.

* primitive mean directly stored actual data in that variable.
  1. number, 2. string 3. boolean 4. undefined, 5. null
* Non primitive means variable can store reference of the data.
  1. object 2. array 3. function
* If we want to know the type of variable. we use type of operator.

  syntax: typeof variable name.

  ex: console.log (typeof variable);
* undefined means no value initialization to the variable.
* object is a combination of properties.
* properties are a key value pair.

  key means variable name but variable name should not be let, var and char.

  syntax: {some variable name (key) : "value" (value)}

  object should be debined in '{}'

  Type of an object is an object
* array is a collection of values, values should be any datatype. It can debine in "[]". Type of an array is an object
* Null means empty value. ex: let number = null.
* If we print datatypes of number in the above exp output is an object. It is loophole of JS

* compiler:
It can execute code at a time.

Interpreter:
It can execute code line by line.

* Java script is an Interpreter.

* Let and var keywords is allows user to create more than one variable at a time, but const didn't

conversions:

1. Implict - It means converts datatype automatically by Interpreter
2. Explict - It means converts datatype by users with the help of Number, string, boolean

Implict Ex: let number = 5 + "5"
console.log (number)

output: 55

Explict ex: let number = 5 + boolean ("4")
cosole.log (number)

output: 6

* Interpreter treat string as True.
* Interpreter ... empty string " " False

Function:

These are three types.

1. Function declaration
2. anonymus function (or) Function expression.
3. arrow function

1. Function declaration: Function declaration has follows below syn

syn: function someName() {
console.log ("Hello");
};
hello(); → function calling

2. In anonymus function it has no function name.

syn: let someFn = function ()
{ console.log ("Anonymus function!");
};
someFn(); → Function calling

3. Arrow Function :

It has no function name and also no function.

syn : let arrowFn = () => {
console.log ("Arrow Function");
};
arrowFn ();

return keyword : It is used two ways.

It gives two behaviours.

1. return values.
2. stop the execution.

* If we want to use the values of function outside the function then use return keywords.

This can be done by storing the values of a function in a variable.

Ex
```
function msg() {
let username = "student";
return user-name;
}
let somevalue = msg();
consle.log (somevalue, "somevalue");
```

parameters & arguments :

* parameters are variable passing at function declaration section.
* arguments are values passing at function calling section.

```
function sum (a,b);
{ return a+b;
};
sum (4,5);
sum (6,1);
sum (8,9);
```

* Any type of datatype should be passed as arguments & parameters

template literals :

Ex console.log (`hello ${name1} ${name2}`);

Backticks

storing values in variables and console both text and

Template literals are used for variables.

parameters & arguments in arrow functions.

Ex: Let someFn2 = (name1, name2) => {
console.log (`hello ${name1} ${name2}`);
};
someFn2 ("John", "smith");

output: hello John smith

* If we want to find the length of an array, we use length property.

Syntax: console.log (array.length); → is a property

* change elements in an array.

Syntax: array[1] = 10;
↓ array name, ↓ index position, ↳ New element

Ternary operator (or) conditional operator: Looping statements (if, if else ...)

Ternary operator is simple way of ~~if else loop~~ condition.

Syntax: condition ? statement1 : statement2

* If condition true statement1 displayed.
* If condition false statement2 displayed.

Else if in Ternary operator.

Ex: condition ? statement1 : condition ? statement2 : .... : condition n.

post increament & pre increament:

post increament:

In this first it assigns value and next increased.

Ex: let nu = 1;
let nu1 = nu++;
console.log (nu1)
output : 1

and vice-versa for pre increament

post-/pre decreament

This works same as post/pre increament.

~~for loop~~

array looping with for :

```
let sing = ["Hi", "Hello"]
for (let i=0 ; i< singers.length ; i++)
  {  console.log (singer[i]);
  }.
```

for in loop:

It is used for looping the object.

Object:

retriving values in an object done in two ways.

1. with .
2. with [ ]

Ex let obj = {
firstName : "Hi";
MiddleName : "Hello";
age : 10 0; } ;
console.log (obj.age)

output : 10

* In some times key name has spaces then use [ ] for retriving.

Ex In the above example FirstName like first Name it must be placed in double (or) single quotes
then console.log (obj ["first Name"])
In this double quotes represent as property.

object looping with for in loop:

Ex for (let key in obj) {
console.log (obj [key]);
}

scope:

scope mean lifetime, (or) limit.

Types of scopes:

1. global scope. we can use variables any block in the code.
2. local scope.
- block (or) scope. we can use variables with in the block only.

In this block scope we can declare variable with let only because let block scope properties.

3. function scope we can use variable with in the function only.

* let has block scope
* var has function scope } let, var & const has global scope.

Hosting:

It is mechanism.

* In this if we declare variable with var keyword, then it arranges first declaration, next consoles, and last initialization. (During compilation).

Ex: If we give code like below

```
console.log (num);
var num=10;
```

For above example. due to hosting it change like below

```
var num;
console.log (num);
num = 10;
```

* Varible declaration with var and functions declaration with var should be got to top due to hosting.

call back function:

call back function is a function passed as an argument is known as call back function.

Ex
```
function updated (name, fn)
{
 console.log (name);
  fn();
}
function wishes () {
   console.log (" some ");
}
updated ("students", wishes);
```

In above ex wishes is a call back function

Methods:

Method is a function placed in an object, It can be represents as value.

Ex
```
const car = {
    name : "Mahendra thar",
    start : funct() {
        console.log ("Engine started");
    },
};
car.start();
```

In the above ex start act as a method. because it value as function.

Built in methods

Js has some Built in methods.

* we use ~~build~~ in methods for arrays. Because js interpreter array treated as an array.

some built in methods are:

push method : It add value at last of an array.

Ex: const array = [1,2,3,4,5,6,7,8];
array.push("10");
console.log(array);

pop method : It works vice versa of push method.

shif method : It remove first values of an array.

unshift method : It add value at 1st position of an array.

Join method : It seperate values of an array as individual element.

syntax: array.join(",");

concat method : It is used for join two arrays. syn: name1.concat(name2)

to lowecase method : It converts uppercase letters to lower case letters. syn: name.tolowercase()

to uppercase method : vice versa of to lowercase method. syn: name.toUppercase()

Map method : It is a method. It is used for looping a object.

syntax: Arrayname.map(function(currentvalue, index){});
Any 3 function should be allow.

It is used for modification in an array.

Filter Method : It is used for filtering an array.

syntax: Arrayname.filter(function(value, index){});
Any 3 function should be allow

* Map method is always return values as an array.

print current data in Js.

console.log(new Date());
↓ It is a object.
It is a keyword

console.log(new Date().toString());

* Js is always link in body tag at last.
* Js should not be link in head tag, because it take more time to load. It is not a correct process.
* Js link with the help of <script> </script> tag.

script has src attribute, it is used to link external Js.

syn <script src="name"> </script>

* Java script done in two ways ① external ② Internal.
* Internal Javascript done blw opening & closing script tag.

Ex: <script>
Javascript code
</script>.

* Internal & External Js didn't work at a time.
* alert("Hello world") is used to alerting the websites it works only after linking with html file.
* prompt is used to take input at compilation stage.

Ex: let num = prompt ("Enter your text");
console.log (num);

DOM: Document object Model

* It is a HTML tree structure (or) Interface
* In this each tag will be treated as an object, so it is called as DOM
* It is used for stylishing HTML also.

Syntax: document.getElementById ("p-tg");

Id should be unique. In case id should be duplicate, it gives only first id values.

Inner HTML : It is a property.

* It is used to change text in that tag.

Ex: document.getElementById("p_tag").innerHTML = "some matter"

Ex: let paraTag = document.getElementById("p_tag")

paraTag.innerHTML = "some matter"

If we want to make styling for above ex with the help of style attribute.

paraTag.style.someProperties.

↓

All css properties are worked in JS

* If we get HTML by class name and tag name into JS

syn: document.getElementByClass("class_name")

document.getElementByTag("Tag_name")

* Any HTML tag in DOM can be represented as, datatype as an object.
* In DOM, Methods are used to access data.
* In DOM, property is used for change data (or) Make stylishing.
* document.getElementId is an object in DOM.
* document.getElementsByclass (or) getElementsByTags is a collection of objects in DOM, because in HTML tags can be duplication.
* we didn't use properties on collection of objects, but we use properties on individual object.

Ex:

let tags = document.getElementsByclassName("hi")

tags[0].innerHTML = "Hi"

↓

dot represents property. So we don't write like below (tag.[0]).

* If we want to loop DOM obj collection, first it converts into array.

syn:

let arr = Array.from(obj collection name (tags))

For looping converted array:

syn

```
array name.forEach(function(obj){
    console.log(obj, "obj");
    obj.innerHTML = "changed"
});
```

Create Element using DOM:

* create a tag using DOM

syn

```
let abc = document.createElement("tag_name(a)")
abc.innerHTML = "click Me"
abc.Href = "3#".
bodyTag.appendchild(abc)
```

(for creating an Element)

(abc: variable name; for assign; for display (or) assign position.)

The above syn works when we specify the position, and also access the position tag.

Events in JS:

Event: any action is known as event.

addEventListner: It is used for listen the action, then execute the function.

syn

```
let heading = document.getElementById("heading")
heading.addEventListner("click", function(){
    console.log("clicked")});
```

(click: event name; function: call back function.)

any executable statements nothing but creating (or) removing (or) any other.

Ex:

```
let ptag = document.getElementById("tag")
ptag.addEventListener("Mouseover", function(){
    let ab = document.createElement("p")
    ab.innerHTML = "Hello world"
    bodyTag.appendchild(ab)});
ptag.addEventListener("mouseleave", function(){
    bodyTag.removechild(ab)});
```

* we can perform many events at a time.

Prevent default method

Prevent default method is for remove load behaviour from submit button. This method can be done when we give event as parameter in callback function.

Syn: button.addEventListner ("click", function (event) {
    event.preventDefault();
})

value attribute in input tag:

value attribute is used for store data what ever you give.

* To access the attribute in j.s by "."

Ex: button.value

Local storage:

* Local storage is a permanent storage (or) web storage.
* Every browser has local storage.
* In j.s local storage is keyword.
* local storage has 3 methods.
    1. setItem - It is used to store data in localstorgge
    2. getItem - It is used to retrive data from localstorge
    3. removeItem - It is used to remove data " " " "

* setItem : It has two values.
    1. key - must be unique
    2. value - value.

Syntax: localstorage.setItem ("name", some value);

* Local storage allows user to give more than one value with the help of object. but created object didn't accept by the local storage. then convert

object in string with the help of JSON.stringify(objname).

Ex: Create obj.

```
let obj = {
  Name: "Bhavi",
  Age: 30; }
localstorage.setItem("signupdata", JSON.stringify(obj));
```

* getItem :- It has only one value.
  * key - created key at setItem session.

To change location of web :

It can be done with the help

Syn: window.location.href = "./#";

* removeItem - It has only one value.
  * key - created key at setItem session.

Closures :-

Closure is a Mechanism.

* By closure mechanism, we can access data from out side function to inside function after the execution.

Ex:
```
function name() {
  let message = "hello";
  function innerFn() {
    console.log(message);
  }
  return innerFn();
}
let fn = name();
fn();
```

Asynchronous

Set time out & set interval Methods

These methods are used to execute statements, whatever time you given.

Set time out :- These execute after the completion of time only once, It has two arguments.
1. call back function, 2. time in ms

Syn:
```
setTimeout(() => {
  console.log("Hello world");}, 7000);
```
Takes only ms.

setInterval :- These execute repeatedly what you time given. It has two arguments.
1. call back function & time in ms :

synr setInterval (() ⇒ {
console.log ("hi")}, 100);

clearInterval :

These helps to handle the setInterval method.
* It has only one argument.

ex Let Interval = setInterval (() ⇒ {
console.log ("hi"), 100);
setTimeOut (() ⇒ {
clearInterval (Interval)
}, 800);

synchronizer

It excute line by line, and also blocks the code.

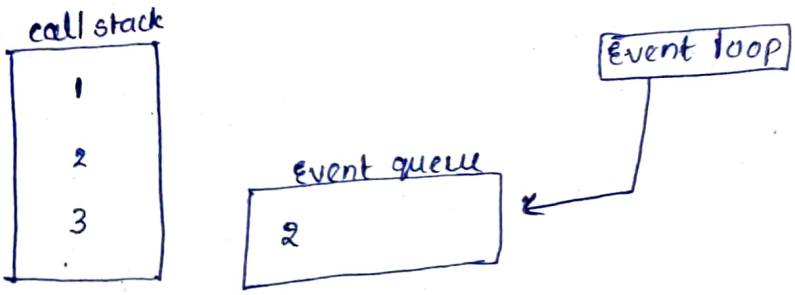
Asynchronizer

It doesn't execute line by line and also doesn't blocks the code.

ex setTimeout, setInterval, API

Javascript handling Asynchronize behaviour with Event loop :

ex
```
console.log ("1")
setTimeout (() ⇒ {
  console.log ("2"), 700);
console.log ("3")
```



* In the above ex firstly '1' is executed and next '2' is stored in event queue because it take time to execute and

next execute 3. After completion of execution event loop check event queue, If any element in Event queue then event loop throw the element to call stack, (after 700 based on above ex) after ~~execute~~ throwing execute 2.

API (Application program Interface):

"==" operator:

* == is used for comparing values only.
* === is used for comparing both values and datatypes.

Toggle:

It used for add and remove ~~tage~~ class.

* If the tage has already class, it remove lly
* If the tag ~~has~~ doesn't have class, it add.

Syn: variablename.classlist.toggle("classname").

Ex: para.classlist.toggle("show").

API:

* It acts as bridge blw Front end & Back end.
* It is used for toransform& retrive the data from back to front end & fronnt to back end.
* API develops at back end.