1st Class.

0325-2968211
M. Qasim

Audacious=Bold/Brave.

GUI

CUI

Supervize Learning

un-supervize Learning.

Reforment Learning.

AGI=2nd Human/General Purpose/Multitasking.

A. Narrow Intelligence=Specfic PurPose.

Symbolic AI=Programming with code.make a product.

Machine learning=Machine learning with algorithms you give.

AWS.

Google Cloud.

2nd Class.

Binary Language=Low Level Language.

Developers=are the builders of digital world.

Program to Print Hello world:-

```
console.log("Hello World");
```

3rd Class.

console.log= To print any thing.

Variable= A container to store our data.

Datatype= string, numbers, Boolean.

-String= Any thing between "" is called string. } core types.

-Boolean= True or false is called boolean.

-Number= 9.001, 00,799 ; it is the type of it.

4th Class.

Operators:-(Program);

```
Let num=1;
Let num=3;
console.log("Addition is=", num1+num2);
```

code:-(Sub)

```
Letnum=5;
num--;
console.log(num);
```

code:-(Add)

```
Let num=5;
num++;
console.log(num);
```

code:-

```
//assignment operater
Let num=1;
Let num2=num+3;
console.log(num2);
```

code:-

```
Let n1=3;
n1+=10;
console.log(n1);
```

code-Comparision Operators:-(Equals to)

```
Let n1=3;
console.log(n1==3);
```

!=Not Eqeals to

code:-(Not equals to);

```
Let n1=3;
console.log(n1!==3);
```

code:-(greater);

```
Let n1=3;
console.log(n1<=3);
```

code:-(Lesser);

```
Let n1=3;
console.log(n1>3);
```

code:-(less than equals to);

```
Let n1=3;
console.log(n1<=2);
```

code:-(greater than equals to);

```
Let n1=3;
console.log(n1>=2);
```

code:-(&&OR);

```
Let n1=3;
console.log((n1===3)&&(n1<2));
```

Class 5th

Code: (If, Else):

```
Let user = "Ratan Lal";
Let pw = "123";
if(user=="Ratan lal" && pw=="123") {
    console.log("Hello");
}

else {
  console.log("wrong");
}
```

Code(If, Else, Else If):-

```
Let user = "lal"
if (user=="Ratan lal"){
   console.log("Hello", user)
} else if (user=="Chaman Lal"){
    console.log("Hello", user)
} else if(user=="Gulzar lal"){
  console.log("Hello", user)
} else {
      console.log("No")
}
```

-Code:-(Add)

```
function myfunc(){
  let total= 2+2;
  console.log(total)
}
myfunc()
```

-Code:-(Make a variable)

```
function myfunc(){
let total =2+2;
console.log(total)
}
Let 1=myfunc()
console.log("Total num", totalnum)
```

-Code:-(return)

```
function myFunc(){
 Let total= 2+2
 return total
}
Let totalnum= myfunc()
console.log("Total Num", totalnum)
```

-Code(Add):-

```
function add(){
    2+2
}
```

Code: (If, Else, Else If) nasted = One in one

```
Let user= " Ratan Lal";
Let OPT=" 123";

if (user == "Ratan Lal"){
   console.log ("Here is Ratan lal")
  if (OPT=="123") {
    console.log ("Welcome");
  } else {
    console.log (" Invalid OPT");
  }

} else{
  if (user== "Any body"){
    console.log ("Invalid OPT");
  } else {
    console.log ("Invalid OPT");
  }
}
```

Class 6th

Code:- (To print many time)

```
function my func () {
    console.log ("Hello world")
}

my fune()
my fune()
```

```
let num= add()
console.log("num")
```

-Code(string):-

```
function myfunc(){
  return "Hanten lal"
}
let num= myfunc()
console.log(num)
```

-Code:-(variable)

```
let user="lal"
let pw="132"
if(user=="lal"){
  console.log("sending code---")
}
else{console.log("Invalid user")}
if(otp=="132"){
  console.log("welcome",user)
}
else{console.log("Invaild otp")}
```

-Code(function/nested):-

```
function login(){
Let user="lal"
Let pw="2"
if(user=="lal"){
  console.log("Here is lal")
 if(pw=="2"){
  console.log("welcome")
 }
  else{
   console.log("No pw")
  }else{
  if(user=="Anybody"){
    console.log("No user")
  }else{
    Console.Log("Invalid pw")
  }
  }
```

-Code:-(Calculation):-

```
Function add(){
  letnums=578;
  return nums
}
Let sum=add()
console.log(sum)
```

Code:- Arrow function:-

```
1- let greeting = () => {
   console.log("Arrow function")
};
greeting()
```

```
2- Let mycal = (n1: number, n2: number, sign: string) => {
   if(sign == "+") {
   console.log(n1+n2)
};
   elseif(sign == "-") {
   console.log(n2-n1).
   }
    else {
   console.log("No Sign")
   }
};
mycal(10,20,"+")
my cal(20,10,"-")
```

Code: (None)

```
function greet(name:string){
    console.log("Hi", name);
}
greet("Ali")
```

~~Code: (sum)~~
~~function sum1(){}~~

- Code:- (Sum)

```
Function sum1(num1:number, num2:number){
    let add = num1 + num2
    return add
}
console.log(sum1(10,9))
```

Class 7th :-

```
function mycal(n1:number, n2:number, sign:string){
if(sign=="+"){
  console.log(n1+n2).
}else if(sign=="-"){
  console.log(n1-n2)
}else{
   console.log("No sign")
}
}
mycal(10,5,"+")
```

Class 8th

- One line code:

```
Let multiply = (num1, num2) => num1 * num2

console.log(multiply(3,6))
```

- Function (Nasted):-

```
Function yourname(){
  let area=98
  Function abc(){
  let x= 786
  console.log(x)
  }
  abc()
  console.log(area)
}
yourname()
```

- Code:-(name=variable with all datatype).

```
Let name1: string|number|boolean|undefined = "la"

name1=3

name1= true

name1=undefined
```

- Tsconfirogation file (open):.

tsc --init

on cmd

- for watching the file

tsc -W

Class 9th

~~- Modules~~
~~... files ...~~
~~import { mything } from "./file1"~~

tsconfig.json file setup

tsc --init |- to make file

To modify:-

14 = "target": "es 2023"

28 = "module": "nodenext"

30 = "moduleResolution": "node next"

package.json file setup

npm init -y | to make file

To modify:-

line 5 Ke baad / Enter

line 6 pe = "type": "module",

-code(modules):-

make a files package/tsconfig

First file = export let teacher name = "Boy"

Second file = import {teacher name} from "./1"

console.log(teacher name)

- code(array):-

Let arr=[ ] //syntax of arr.

Let arr : (number | string | Boolean) [ ]

- Code(array):-

```
let arr=["a","b",4,5,true,"Bilal","Ali"]
console.log(arr)
```

- code(arrayinarray)

```
let arr = ["a","b",["Ali"]]
console.log(arr[1])
```

- code(array in array)

```
let arr =["a","b",["Ali","Baig"]])
console.log(arr[2][0])
```

Code:-(To remove last array) Class 10th Tell that array

```
let students=["huma","Ali","bilal","sana"]
students.pop()
console.log(students);
```

Double
console.log(students.pop())

code:-(To add an arry in the arry at last variable) length also in double

```
let students = ["huma","ali","bilal","sana"]
students.push("ahmed")
console.log(students)
```

Double
console.log(students.push("ahmed")

Class 11th

- Tuple: type array
- To tell the data type of every variable in array / defined its length / Predefined length

- code (tuple array):-

```
let arr: [string, number, boolean] = ["lol", 5, true]
console.log(arr)
```

- Code (tuple array optional):- ?

```
let arr: [string, number, boolean, string?] = ["a", 5, true, "b"]
console.log(arr)
```

- Obj = A to give the properties to an key

Syntax Let obj = {}

- code (object)..

```
let obj = {
    name: "ratan lal",
    age: 25,
    cell: 158,
    cars: ["honda", "farari"],
    salary: () => {
        return
        console.log("berozgar")
    }
}
console.log(obj)
console.log(obj.salary())
```

•code: (To remove first array): Tell that array in double

```
let students=["huma","ali","bilal","sana"]
students.shift()
console.log(students)
```

Double
console.log(students.un
shift())

•code:-(To add variablie in first) length also in double

```
let students=["huma","ali","bilal","sana"]
students.unshift("ayaan")
console.log(students)
```

Double
console.log(students.unshift
("ayaan"))

•code:-(slice):-

```
let steidents=["huma","ali","bilal","sana"]
consolelog(stedents.slice(1,3))
```

•code:-(splice):-

```
let students=["huma","ali","bilal","sana"]
console.log(students.splice(1))
```

- Code (Object):-

```
Let obj = {
    name: "lal",
    age: 1,
    cell: 1235,
    cars: ["honoda", "Toyota"]
}
console.log("name", obj.name)
console.log("age", obj.age)
console.log("cell", obj.cell)
console.log("cars", obj.cars)
```

- Code (object = `) :- Back tag

```
Let obj = {
 name: "lal"
 age: 1
 cell: 1356
}
console.log(`my name is ${obj.name}, my age is ${obj.age}`)
```

---

inqererer = To take input from user.

If you want to have inquier so run command npm i inquirer\ on file cmd or delete npm i from website.

Bohot چیز ki ہیں hai

- Code (loop):-

for() = syntax (main important)

* for(let j=0; i<=10; i++)

variable condition increment/diacrement

- Code (loop):-

```
For (let i=0; i<=100; i++) {
  console.log (i)
}
```

- Code (loop to add):-

```
let sum = 0
For (Let i=0; i<=100; i++) {
  sum += i
}
console.log (sum)
```

-Code(type defined previsly in Objeet):-

```
type student = {
    name:string,
    rollno: number,
    Phoneno: number,
    city: string,
} class?: number

et student 1:student={
    name: "Ali;"
    rollno: 12,
    phoneno: 2,
    city: "Karachi;"
    class: 1,
}
console.log(student1)
```

-Code(interface):-

interface my{} = syntax

```
interface student {
    name: string,
    phonen: number
}

let student 1: student={
    name: "Ali;"
    phoneno: 1802
}
console.log (student 1)
```

Class 12th

while loop

+ syntax = while(){}
+ let i = 1 // variable

condition
while(i<=10){
    console.log(i) // To print
    i++ // Increment
}

-Code(while loop):-
let i = 1;
while(i<=10){
    console.log(i)
    i++
}

- do,while loop = Type of while loop
  make a variable = let i=1
- syntax do{
    console.log(i)
    i++
}while(i<=10)

-Code(while loop type do)

let i = 1
do{
  console.log(i)
  i++
}while(i<=10)

- code (interface extends):-

```
interface std{
  name: String,
  rollno: number,
  phone: number,
  age: number
}
interface tec extends std{
  gmail: String
}
let s1: std ={
  name: "a",
  rollno: 1,
  phone: 256,
  age: 12
}
let t1: tec ={
  name: "Ahmed",
  rollno: 3,
  phone: 8945,
  age: 28,
  gmail: "myad@gmail.
}
console.log(t1)
// stale obj
s1=t1
```

Class 13th

+ Callback = kisi bhi function kay argument me fuction pass kerne ko callback kehte hain

-Code(callback) :-

```
Function hello(cb:any) {
    cb()
}
function my() {
  console.log("Hello cb")
}
hello(my)
```

-Code (callback arrow):-

```
function hello(cb:any){
  cb()
}
hello(()=>{
   console.log("Hello cb")
})
```

intersection = combine to things

- Code (intersection):-

```
type Obj = {
    name: string,
    age: number
}

type obj1 = {
    rollno = number,
    class = string
}

type r1 = obj & obj1

let boy : r1 = {
    name: "Boy",
    age: 5,
    rollno: 8,
    class: "1"
}

console.log(boy)
```

Synconis = line by line [redacted] result/run

-Asyconis = line by line but if it will take time it will move forward.

-Code(Synconis):-

```
console.log(1)
console.log(2)
console.log(3)
console.log(4)
```

-code (Asyconis):-

```
console.log(1)
console.log(2)
[redacted] setTimeout(()=>{ console.log(3)},2000)
console.log(4)
```

-Code(Promise):-

```
Let myPromise = new Promise((resolve, reject) => {
    console.log("Pending")
    setTimeout(()=>{
      console.log("reslove")
      resolve("Bilal")
      },1000)
})

mypromise.then((res) => console.log(res))
```

Class 14th

-code(class):-

```
class Human {
    name:string
    age:number
    constructor(n:string,a:number){
      this.name=n
      this.age=a
    }
}
Let s1 = new Human('Ali',66)
Let s2= new Human('lal', 30)
console.log(s1)
console.log(s2)
```

-code(class rename):-

```
class Human {
  name:string
  constructor(n:string){
    this.name=n
  }
   rename(n:string){
     this.name=n
   }
Let s1 =new Human('chaman lal')
console.log("Before",s1)
s1.rename ("lal")
console.log("After",s1)
```

```
Let my = new Promise ((resolve, reject) => {
    console.log("pending")
    setTimeout (() => {
        console.log("reject")
        reject (new Error("nah!"))
    }, 1000)
})

myPromise.
.then ((res) => {console.log(res)})
.catch ((err) => console.log (err)
```

-Code (Promise smaller)

```
Let url = "Json place holder"
Let fetch = fetch (url)
.then ((res) => console.log (res))
.then ((res) => res.json)
.catch ((err) => console.log (err))
```

-Code (Promise in async function)

```
async function fetchDataFunc() {
    let url = "Json place holder"
    let fetchdata = await fetch (url)
    let res = await fetchdata.json()
    console.log (res)
}

fetchDataFunc()
```

- Code (class extends):

```
class student {
  name;
  rollnumber;
  constructor (n:any, r:any) {
    this.name = n
    this.rollnumber = r
  }
}

class Teacher extends student {
  id;
  constructor (n:any, r:any, i:any) {
    super (n, r)
    this.id = i
  }
}
let s1 = new student("Ali", 30)
let T1 = new teacher("Ali", 66, 7)
console.log (s1)
console.log (T1)
```

-Code(inheritence):-

```
// Base class or Parent Class
class Product {
    name;
    price;
    constructor(n:any, p:any) {
    this.name= n
    this.price=p
    }
    disply(){
    console.log(`led ${this.name}'s price is ${this.price}`)
    }
}
// child class or derived class
Class Electronics extends Product {
    warranty;
    constructor(name:string, price:number, warranty:number){
        super(name, price)
        this.warranty=warranty
    }
    showwarranty(){
        super.display()
        console.log(`the warranty is ${this.warranty} years`)
    }
}
let led = new Electronics("Sony", 20000, 3)
led.showwarranty()
```

-Code(Class Easy):-

```
class Person {
  name!: string
  age  : number
}

let person = new Person()
person.name="Ali"
person.age = 19
console.log(Person)
```

-Code(Product):-

```
class Product {
    name;
    price;
    constructor (n:string, p:number) {
       this.name=n
       this.price=p
    }
    disply(){
       console.log(`${this.name}'s price is ${this.price}`)
    }
}

let laptop= new Product("Dell", 10000)
console.log(laptop.price)
laptop.name="Hp"
laptop.disply()
let mobile= new Product("Apple", 500000)
mobile.disply()
```

```
let obj1 ={
    name: "Ali"
    age:30
}
let obj2 = {...obj1, name: "waqar"}
console.log(obj1)
console.log(obj2)
```

-Code (res):-

```
function abc (a:number, b:number, c:number, ...rest:number[]){
    console.log("Rest", rest)
    return a+b+c
}
console.log(abc(1,2,3,4,5))
```

-Code (class Private):-

```
class Person {
  private _name:string = "Ali"
  getname(){
      return this._name = "Ali"
  }
}
Let p1:Person = new Person()
console.log(p1.getname())
console.log(p1 instance Person)
```

Class 15th

- Code (If Else short)

Let user = "al"

user == "al"

? console.log ("Hi")

: console.log ("Bye")

Let user = "al" (short circuit)

user == "al" && console.log ("Hi")

OR

Let user = "al"

!user && console.log ("Empty")

- Code (spilt operator) -

Let arr1 = ["a", "b", "c"]

Let arr2 = ["d", "e", "f"]

Let newarr = [...arr1, ...arr2]

console.log (newarr)

-Code (Protected):-

```
class Person{
  protected _name:any = "lal"
  age=30
}
class Human extends Person{
    getname(){
        return this._name = "lal Bhai"
    }
}

let h1 = new Human()
console.log (h1.name)
console.log (h1.getname)
```

-Code (Overright):-

```
Class. Person {
name = "Ratan Lal"
 age=35
}
Class Human extends Person {
   name = "lal"
}
let h1 = new Human ()    console.log(h1.name)
```