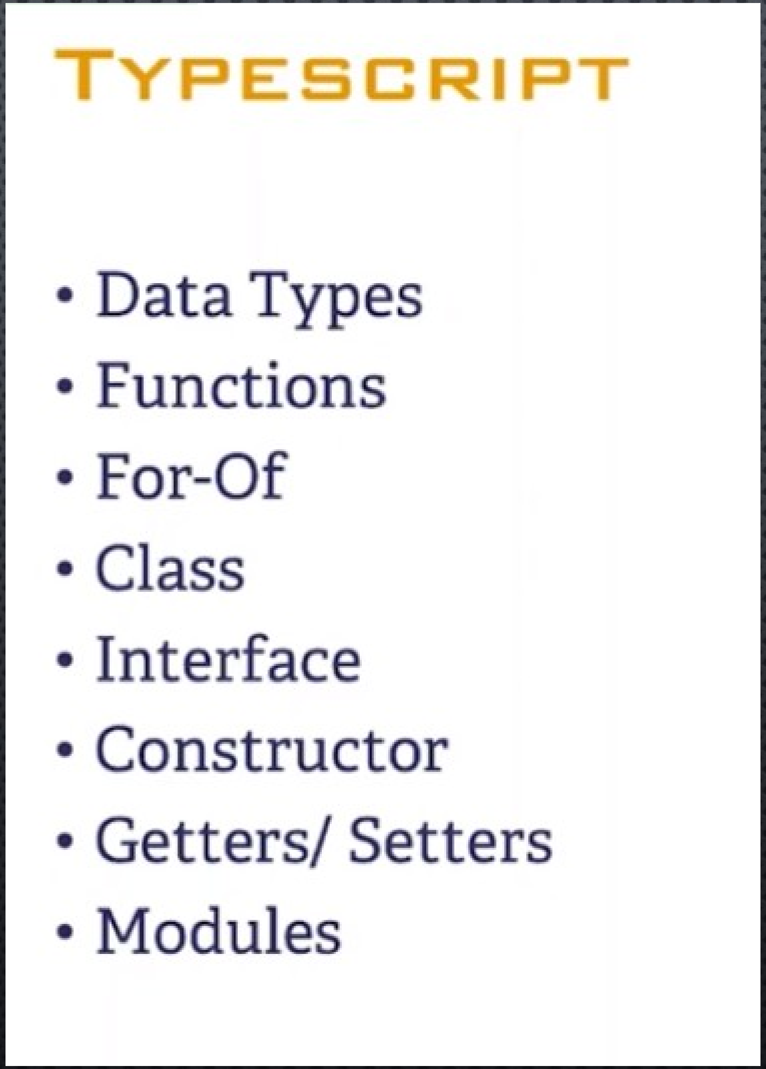
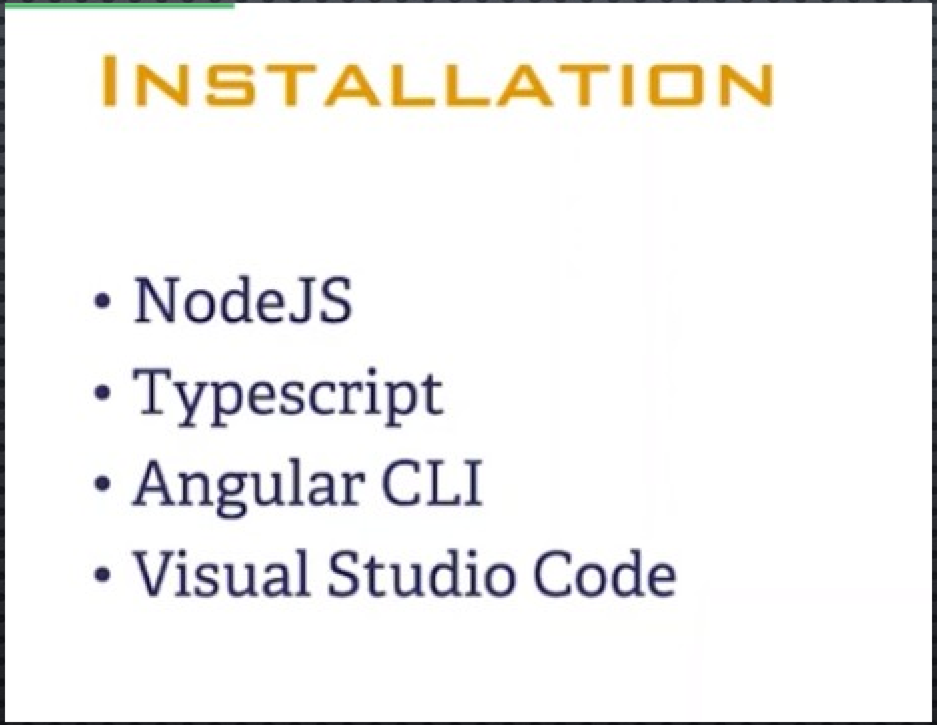
TYPESCRIPT

ChandraSekhar(CS) Baratam

TYPESCRIPT CONTENT


TYPESCRIPT

- Data Types
- Functions
- For-Of
- Class
- Interface
- Constructor
- Getters/ Setters
- Modules



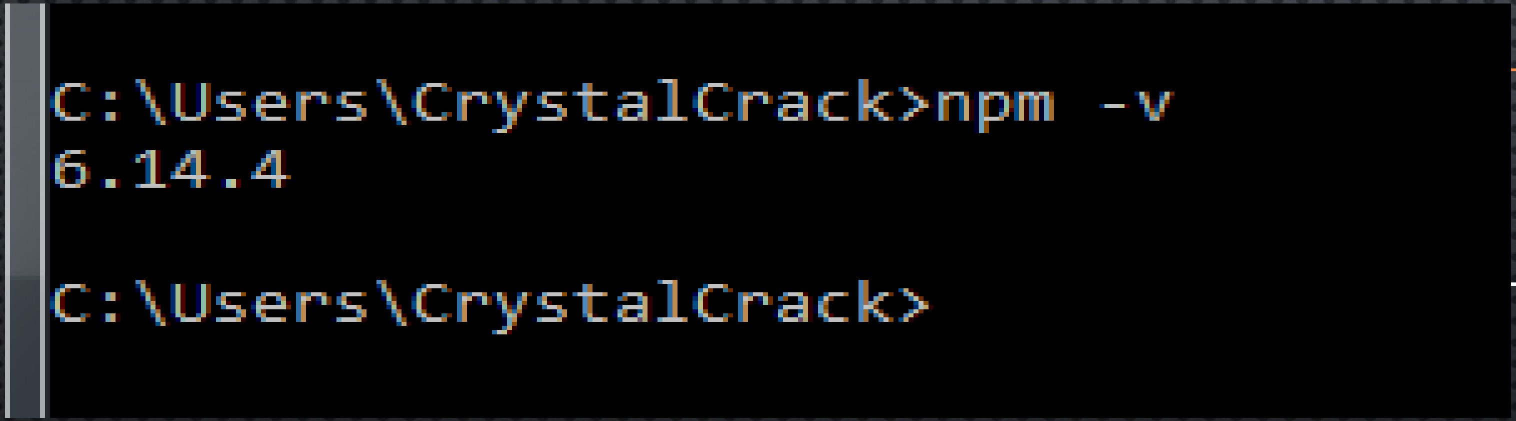
Installation

- NodeJS
- Typescript
- Angular CLI
- Visual Studio Code

Installation

- Install node.js from https://nodejs.org/en/download/
  - Check if node installed by typing node –v and npm -v on command prompt
- Installing TypeScript using npm
  - npm install –g typescript
- Installing the Angular CLI using npm
  - npm install -g @angular/cli
  - ng –v [to test if angular installed]
- Select editor of your choice to start creating angular apps
  - We will be using VSCode
  - Download from : https://code.visualstudio.com

INSTALL NODE JS FOR WINDOWS 7



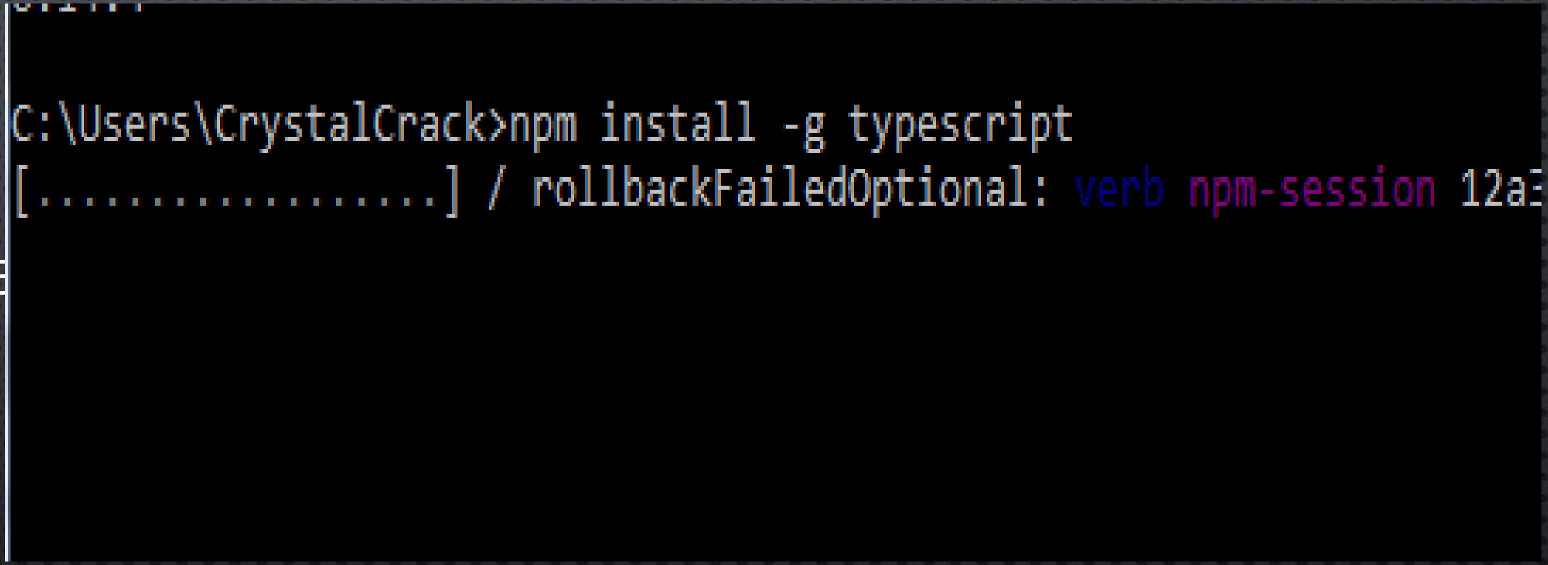
```
C:\Users\CrystalCrack>npm -v
6.14.4

C:\Users\CrystalCrack>
```

- 3.14.0/
-
- 29904896
- TER

ChandraSekhar(CS) Baratam

INSTALL TYPESCRIPT

- GLOBALLY INSTALLING TYPESCRIPT
- NPM INSTALL -G TYPESCRIPT >>>>>ENTE
-



```
C:\Users\CrystalCrack>npm install -g typescript
[..................] / rollbackFailedOptional: verb npm-session 12a3
```

ChandraSekhar(CS) Baratam

DOWNLOAD VISUALSTUDIO

- HTTPS://CODE.VISUALSTUDIO.COM/DOWNLOAD
- CLICK ON THE ICON WINDOWS 7,8,10 DOWNLOAD.
- INSTALL IT.THATS ALL!!!!!!!!

- THE NAME TYPESCRIPT IT INDICATES ITS BASED ON TYPES.
- JAVASCRIPT IS DYNAMIC BASED TYPE WHERE AS JAVA .NET C C++ WHICH ARE STATIC BASED TYPES.
- TYPESCRIPT SUPPORTS FULL OBJECT ORIENTED PROGRAMMING AND PRINCIPALS.

- TWO BENEFITS
- IT COMPILES SOURCE CODE INTO JAVASCRIPT.
- IT CAN RUN ON ANY OPERATING SYSTEM CAPABLE OF EXECUTING JAVASCRIPT.

WHY SHOULD WE LEARN TYPESCRIPT.

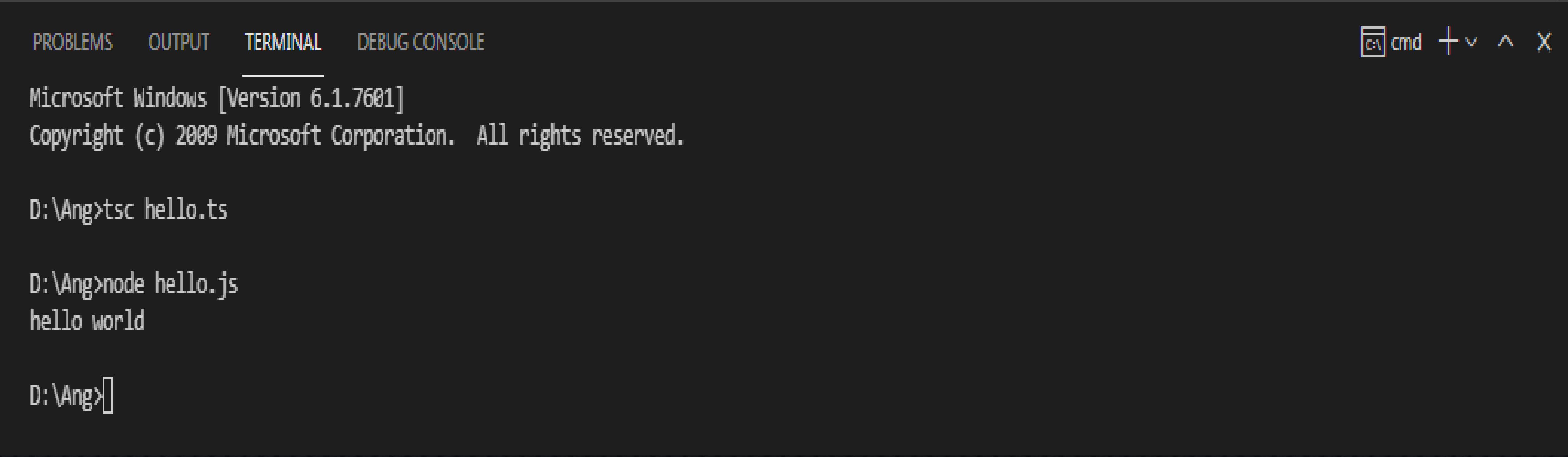
- TYPESCRIPT THE NAME INDICATES TYPE SAFETY,AND IT ENHANCE CODE QUALITY AND UNDERSTANDABILITY.
- JAVASCRIPT IS A TYPESCRIPT AND VICE VERSA.
- TYPES CAN BE IMPLICITY.>WHAT EVER YOU ASSIGNED THE TYPE,STRICTLY TYPE BASED.
- TYPES CAN BE EXPLICITY.>YOU WANT TO STORE IN A VARIABLE.
- TYPES ARE STRUCTURAL
- TYPE ERROR DO NOT PREVENT EMIT JAVASCRIPT CODE>MEANS ONE TYPESCRIPT PAGE HAVE ERRORS ONCE YOU COMPILE YOU WONT IDENTITY IN JAVASCRIPT,(BECAUSE IN JAVASCRIPT WONT IDENTITY THE ERRORS)
- SO MAKE SURE CLEAR ALL THE ERRORS IN TYPESCRIPT PAGE AND COMPILE IT OK.

IF ERROR WHILE EXECUTING TSC HELLO.TS

- THEN PROBLEM WITH POWERSHELL.
- KILL THE TERMINAL
- PRESS CTRL+SHIFT+P TO SHOW ALL COMMANDS.
- TYPE **PROFILE** IN THE DISPLAYED TEXT BOX TO FILTER THE LIST.
- SELECT **TERMINAL**: SELECT **DEFAULT** PROFILE.
- YOU WILL BE PROMPTED TO SELECT YOUR PREFERRED TERMINAL SHELL, YOU CAN CHANGE THIS LATER IN YOUR SETTINGS OR FOLLOW THE SAME PROCESS AS WE DO NOW
- SELECT COMMAND PROMPT (CMD.EXE)
- GO NEAR EXPLORER RIGHT CLICK CREATE NEW INTEGRATED TERMAL >>DEFAULT SHOULD BE CMD OTHERWISE DO ONCE AGAIN.

EXECUTION OF TYPESCRIPT.

- TSC HELLO.JS



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

D:\Ang>tsc hello.ts

D:\Ang>node hello.js
hello world

D:\Ang>
```

HELLO.TS

- SP DOCUMENT BASED CONCEPTS WONT WORK
- LIKE ALERT("HELLO WORLD");>>IT WONT WORK. IT IS NOT DEFINED.

- `CONSOLE.LOG("HELLO WORLD");`
- `CONSOLE.LOG("NOW ITS WORKING FINE");`
- `//ALERT("ALERT ME BABY");`
- `VAR X=10;`
- `CONSOLE.LOG("X WILL WORK OR NO BABA " + X);`

MAIN.TS

```
FUNCTION ADD(A,B)
{
    RETURN A+B;
}
CONSOLE.LOG("AREE BABA ADDING THE NUMBER "+ ADD(10,5));
```

ADD 2 FILES IN 1 FILE

- D:\ANG>TSC HELLO.TS MAIN.TS --OUT APP.JS
- WHEN YOU OPEN APP.JS >>HOW YOU WRITTEN EXACTLY SAME WILL IT WILL BE ADD.

- BUT USING WILD ITS NOT WORKING BECAUSE OF POWERSHELL
- D:\ANG>TSC *.TS --OUT HEY.JS>>IT WONT WORK: ERROR

STANDARDS OUTPUT

```
/**
 * STANDARD OUTPUTS
 * TSC HELLO.TS
 * TSC HELLO.TS MAIN.TS
 * TSC *.TS --OUT APP.JS >>NOT WORKING ALTERNATE
 * TSC HELLO.TS MAIN.TS --OUT APP.JS
 * TSC HELLO.TS --WATCH
 */
```

ALERT ADD IN TS FILE>> WATCH >>AUTOMATIC ADDED TO TS>>ADD IN INDEX.HTML>>OPEN INDEX IN BROWSER

- HELLO.TS
- `ALERT("ALERT ME BABY");`
- AUTOMICATILLY SAMEAS IN HELLO.JS BY WATCH APPLIED
- CREATE INDEX.HTML PAGE ADD IT

```
<!DOCTYPE HTML>
<HTML>
    <SCRIPT SRC="HELLO.JS"></SCRIPT>
<BODY>
    <H1>THIS REFER TO JS FILE</H1>
</BODY>
</HTML>
```

WANT TO INCREASE THE FONT SIZE

- OPEN VS CODE.
- TYPE COMMAND CTRL + SHFT + P.
- TYPE SETTINGS.>>INDICATES >>PREFERENCE:OPEN USER SETTINGS>>CLICK ON IT
- IN USER>>TEXT EDITOR>>FONT>>INCREASE THE SIZE AND CLOSE IT.

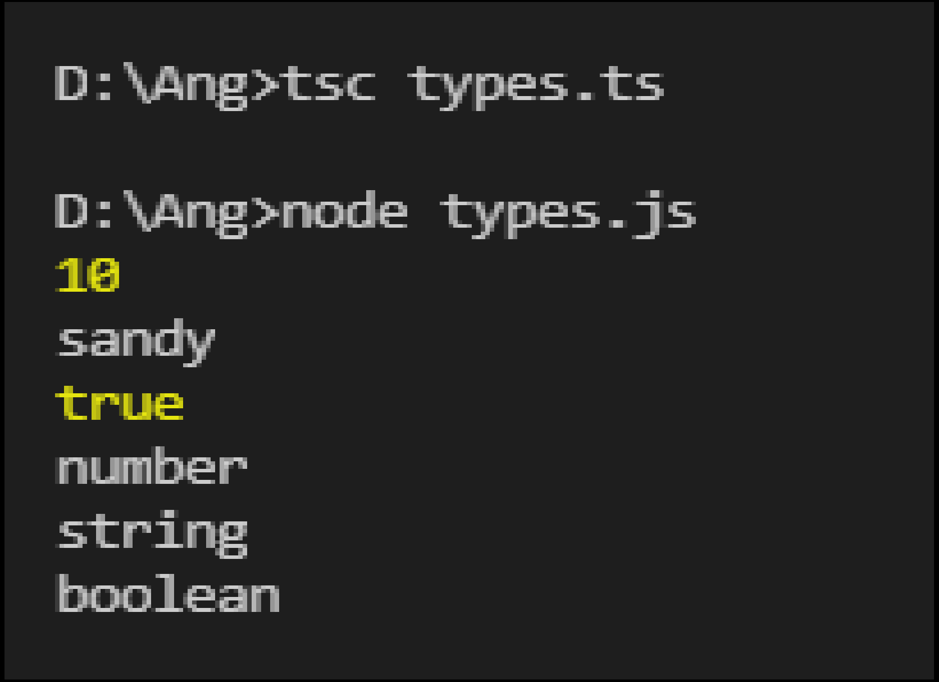
USING TYPE ANY,THEN APPLY NUMBER,THEN STRING

- `//WE CAN PASS ANYTHING USING ANY FOR VAIABLE`
- `VAR NO:ANY=10;`
- `CONSOLE.LOG(NO);`
-
  `NO=“CS;"`
- `CONSOLE.LOG(NO);`
- `-----------------------------------------------------`
  `VAR NO1=10;`
- `CONSOLE.LOG(NO1);`
-
  `//NO1=“CS";//ALREADY OCCUPIED WITH NUMBER`
- `------------------------------------------------------`
  `VAR NO2=“CS";`
- `CONSOLE.LOG(NO2);`
-
  `NO2=10;// ALREADY OCCUPIED WITH STRING`

NUMBER,STRING,BOOLEAN

- JAVASCRIPT DOESNOT SUPPORT TYPES.THEY ARE JUST NORMAL VARIABLES.
- `VAR NO:NUMBER=10;`
- `VAR UNAME:STRING=“CS";`
- `VAR CHOICE:BOOLEAN=TRUE;`
-

  `CONSOLE.LOG(NO);`
- `CONSOLE.LOG(UNAME);`
- `CONSOLE.LOG(CHOICE);`
- `CONSOLE.LOG(TYPEOF(NO));`
- `CONSOLE.LOG(TYPEOF(UNAME));`
- `CONSOLE.LOG(TYPEOF(CHOICE));`



```
D:\Ang>tsc types.ts

D:\Ang>node types.js
10
sandy
true
number
string
boolean
```

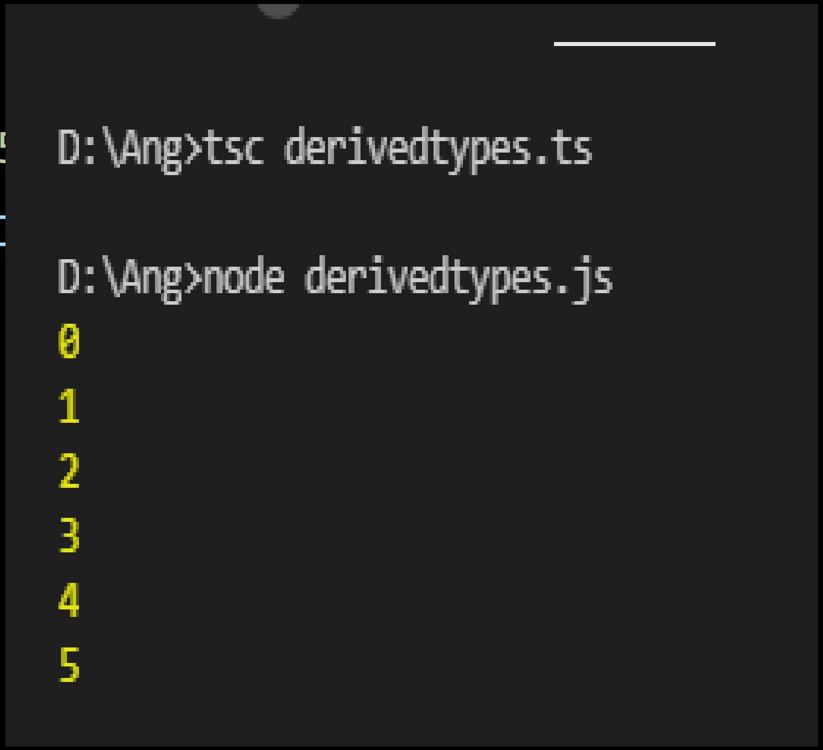
VAR AND LET

- VAR=GLOBALLY
- LET=LOCALLY
- CONSTANT VALUE CANNOT BE CHANGED.

```
• VAR X=10;
• LET Y=20;
• IF(X==10){
•     VAR I=Y+89;//IF YOU REPLACE VAR TO LET WILL HAVE ERROR
•
• }
• CONSOLE.LOG(I);
• CONST C=100;
• C=250;//AGAIN ASSIGN IT WONT WORK
```

DERIVED TYPES MEANS ARRAYS CONCEPT

- VAR NU:NUMBER[]=[1,2,3,4,5
- FOR(VAR I=0;I<=NU.LENGTH;I
- CONSOLE.LOG(I);



```
D:\Ang>tsc derivedtypes.ts

D:\Ang>node derivedtypes.js
0
1
2
3
4
5
```

TEMPLATESTRING ITS INTERPOLATION

- //ABOVE THE TAB KEY
- // ` BACKTICKS …WILL WORK AS IT IS LIKE SPACE.
- VAR CNAME = ‘SANDEEP';
- // ${} -> STRING INTERPOLATION
- LET DESCRIPTION = `HOW ARE YOU? ${CNAME}
-     HOPE HAVING FUN LEARNING ANGULAR
- Z`;
- CONSOLE.LOG(DESCRIPTION);

GENERICS ,BEFORE GENERICS ,THIS PAGE WILL GET ERROR

```
• FUNCTION REVERSE(ITEMS:NUMBER[]) //STRICT TYPE NUMBER
• {
•       VAR REVNOS = [];
•       FOR (VAR I = ITEMS.LENGTH-1 ; I>=0 ;I--)
•       {
•           REVNOS.PUSH(ITEMS[I]);
•       }
•       RETURN REVNOS;
• }
• VAR SAMPLE = [ 1,2,3,4,5];
• VAR REVERSENOS = REVERSE(SAMPLE);
• CONSOLE.LOG(REVERSENOS);
• VAR NAMES =["SHALINI","NAVIN","VIHAAN"];
• VAR REVNAMES = REVERSE(NAMES); //CANT PASS ,ITS ALREADY NUMBER TYPE
• CONSOLE.LOG(REVNAMES);
```

GENERICS,   THAT IS THE RESON WE CAME UP WITH CONCEPT GENERICS

```
• FUNCTION REVERSE<T>(ITEMS:T[])//NOW IT’S A TYPE>>EITHER NUM OR STRING
• {
•     VAR REVNOS = [];
•     FOR (VAR I = ITEMS.LENGTH-1 ; I>=0 ;I--)
•     {
•         REVNOS.PUSH(ITEMS[I]);
•     }
•     RETURN REVNOS;
• }
• VAR SAMPLE = [ 1,2,3,4,5];
• VAR REVERSENOS = REVERSE(SAMPLE);
• CONSOLE.LOG(REVERSENOS);
• VAR NAMES =["SHALINI","NAVIN","VIHAAN"];
• VAR REVNAMES = REVERSE(NAMES);//NOW NO PROBLEM WITH STRING CONCEPT
• CONSOLE.LOG(REVNAMES);
```

FUNCTIONS (RETURN AND VOID)

```
FUNCTION DISPLAY(NAME:STRING):STRING //RETURN A VALUE
{
    RETURN "WELCOME "+NAME;
}
CONSOLE.LOG(DISPLAY('SHALINI'));
FUNCTION SHOW():VOID{                    //WONT RETURN A VALUE
   // RETURN "HELLO";
}
```

FUNCTION>>OPTIONAL AGRUMENTS

```
//OPTIONAL ARGUMENTS
// REQUIRED
//N3 -> OPTIONAL
FUNCTION ADD(N1:NUMBER,N2:NUMBER,N3?:NUMBER)
{
    IF(N3 === UNDEFINED)
    {
        CONSOLE.LOG(N1+N2);
    }
    ELSE
        CONSOLE.LOG(N1+N2+N3);
}
ADD(1,2);
ADD(1,2,3);
```

FUNCTIONS DEFAULT ARGUMENTS

```
//DEFAULT ARGUMENTS
FUNCTION MESSAGE(FOOD:STRING, DRINKS:STRING = 'PEPSI')
{
    CONSOLE.LOG(`HAVE THIS TASTY ${FOOD} ALONG WITH ${DRINKS}`);
}
MESSAGE('PIZZA');  //DEFAULT IT WILL LOAD PEPSI
MESSAGE('NOODLES','LEMONADE');//EXPLICITY LOADED.
```

FUNCTIONS REST PARAMETERS

- //REST PARAMETERS
- FUNCTION GREET(COMPANY, ...NAMES)
- {
-     CONSOLE.LOG(NAMES.LENGTH);
-     CONSOLE.LOG(`${COMPANY} WELCOMES YOU ${NAMES[3]}`);
- }
-
  GREET(‘MYTRAINING');
- GREET(‘CS',‘RAM KRISHNA',‘BABU','RIYA','SANDY');

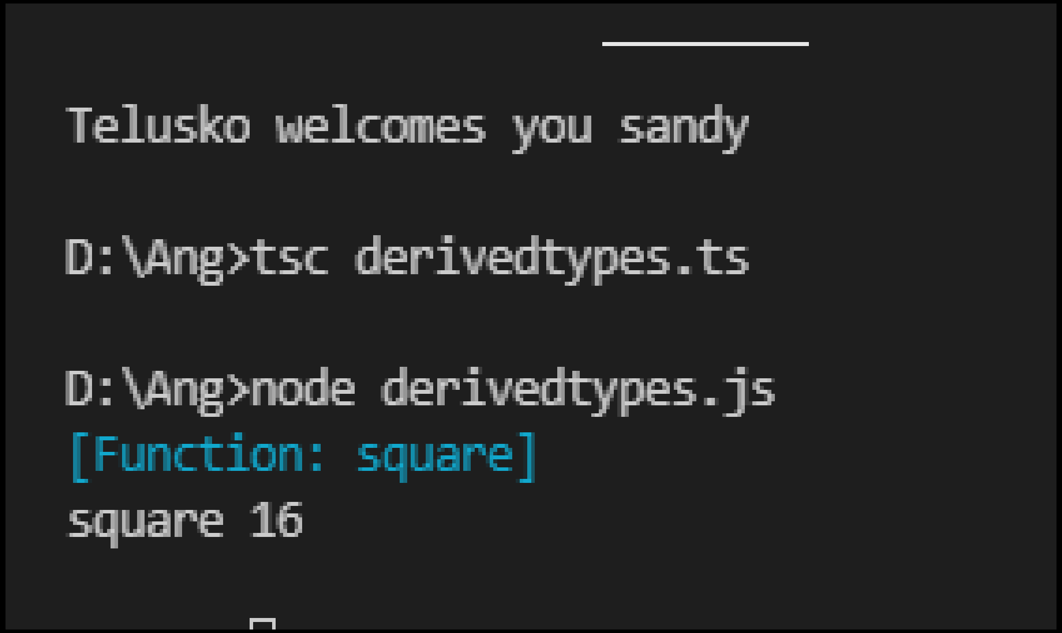
ARROW FUNCTIONS

- NORMAL FUNCTION

```
FUNCTION SQ(X)
{
    CONSOLE.LOG(X*X);
}
//ARROW
VAR SQUARE = (P) => {
    CONSOLE.LOG("SQUARE "+ P*P);
  //  RETURN P*P;
}
SQUARE(4);
```

ARROW FUNCTION RETURN AND VOID

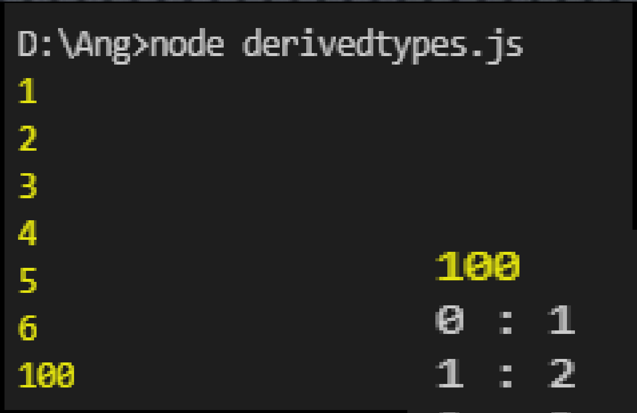
```
//ARROW
VAR SQUARE = (P:NUMBER):NUMBER => {
    CONSOLE.LOG("SQUARE "+ P*P);
    RETURN P*P;
}
CONSOLE.LOG(SQUARE);
SQUARE(4);
```



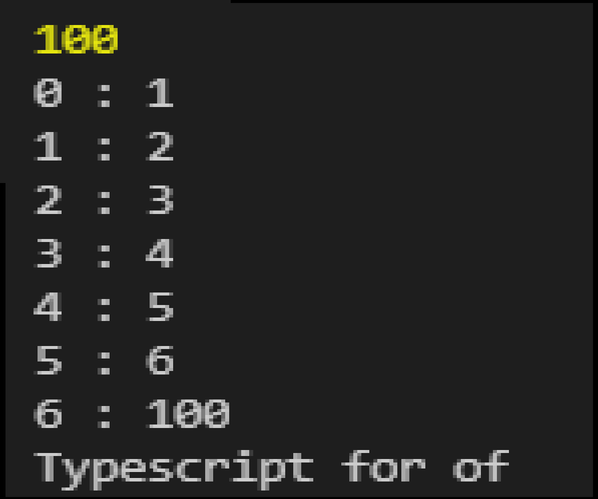
```
Telusko welcomes you sandy

D:\Ang>tsc derivedtypes.ts

D:\Ang>node derivedtypes.js
[Function: square]
square 16
```

FOR OF

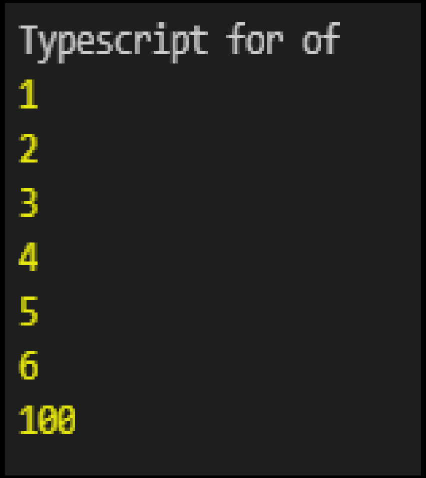
```
VAR NOS = [1,2,3,4,5,6,100];
FOR(VAR I =0; I< NOS.LENGTH;I++)
{
    CONSOLE.LOG(NOS[I]);
}
FOR(VAR J IN NOS)
{
    CONSOLE.LOG(J+" : "+NOS[J]);
}

//TYPESCRIPT -> FOR -OF
CONSOLE.LOG("TYPESCRIPT FOR OF");
FOR(VAR N OF NOS)
{
    CONSOLE.LOG(N);
}
```



```
D:\Ang>node derivedtypes.js
1
2
3
4
5
6
100
```



```
100
0 : 1
1 : 2
2 : 3
3 : 4
4 : 5
5 : 6
6 : 100
Typescript for of
```



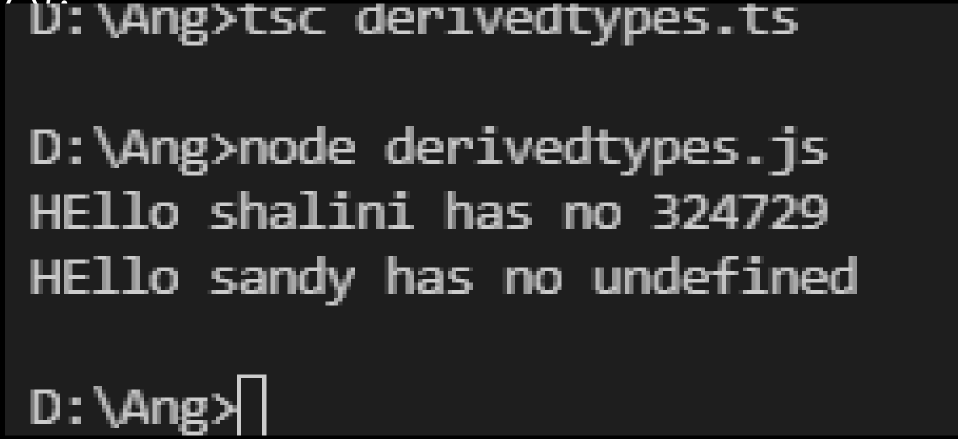
```
Typescript for of
1
2
3
4
5
6
100
```

- //ITS NEW TYPE NO NEED TO CALL INDEX DIRECTLY IT WILL BE LOAD THE DATA

INTERFACE

- INTERFACE JUST REFER TO DATA STORE(DON'T COMPARE WITH JAVA).
- TO MARK A STRUCTURE OF PARTICULAR DATA.

```
INTERFACE PERSON{
    NAME:STRING,
    PHONE?:NUMBER
}
FUNCTION DISPLAYDETAILS(PERSON:PERSON)//STRUCTURE
{

    CONSOLE.LOG("HELLO "+PERSON.NAME + " HAS NO "+ PERSON.PHONE);
}

VAR P1 = {NAME : 'SHALINI',PHONE :324729};//DATA STORE
DISPLAYDETAILS(P1);
DISPLAYDETAILS({NAME :'SANDY'});
```



```
D:\Ang>tsc derivedtypes.ts

D:\Ang>node derivedtypes.js
HEllo shalini has no 324729
HEllo sandy has no undefined

D:\Ang>
```

INTERFACE

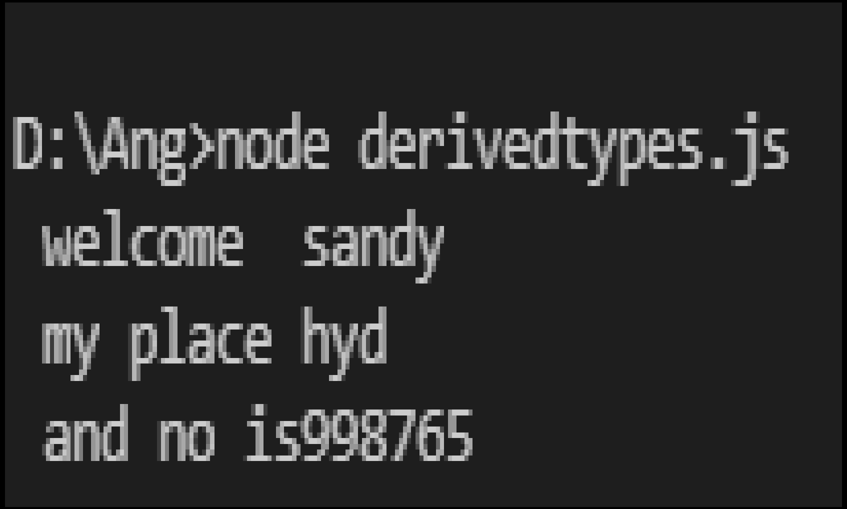
- AN INTERFACE IS A WAY TO DEFINE A CONTRACT ON A FUNCTION WITH RESPECT TO THE ARGUMENTS AND THEIR TYPE.
- JAVASCRIPT DOES NOT SUPPORT INTERFACE

CLASS

```
CLASS USER{
    //DATA MEMBERS OF THE CLASS USER
    NAME:STRING;
    CITY:STRING;
    PHONE:NUMBER;
}

//USER1 -> OBJECT
VAR USER1 = NEW USER();
USER1.NAME='SANDY';
USER1.CITY='HYDERABAD';
USER1.PHONE=1234512345;
CONSOLE.LOG(" WELCOME  "+USER1.NAME +"\N MY PLACE "+USER1.CITY+"\N AND NO IS" +USER1.PHONE)
```

CONSTRUCTOR

```
CLASS USER{
    //DATA MEMBERS OF THE CLASS USER
    NAME:STRING;
    CITY:STRING;
    PHONE:NUMBER;
    CONSTRUCTOR(UNAME:STRING,CITY:STRING,PHONE:NUMBER)
{
    THIS.NAME = UNAME;
    THIS.CITY = CITY;
    THIS.PHONE = PHONE;
}
}

//USER1 -> OBJECT
VAR USER1 = NEW USER('SANDY','HYD',998765);
CONSOLE.LOG(" WELCOME  "+USER1.NAME +"\N MY PLACE "+USER1.CITY+"\N AND NO IS" +USER1.PHONE)
```



```
D:\Ang>node derivedtypes.js
 welcome  sandy
 my place hyd
 and no is998765
```

GETTER AND SETTER (INSIDE USER CLASS CONSTRUCTOR AND GETTER METHOD)

```
CLASS USER{
    CONSTRUCTOR(PRIVATE UNAME:STRING,PRIVATE CITY:STRING,PRIVATE PHONE:STRING) //PRIVATE
{
    THIS.PHONE='+91-'+THIS.PHONE;
}
 //GETTERS OR ACCESSORS
 PUBLIC GET NAME()
 {
     RETURN THIS.UNAME; //CONSTRUCTOR NAME
 }
 PUBLIC GET CITY()
 {
     RETURN THIS.CITY; //CONSTRUCTOR NAME
 }
 PUBLIC GET PHONE()
 {
     RETURN THIS.PHONE; //CONSTRUCTOR NAME
 }
}
```

CALLING THE GETTER METHOD

```
/USER1 -> OBJECT
VAR USER1 = NEW USER('SANDY','HYD','998765');
CONSOLE.LOG(" WELCOME  "+USER1.NAME +"\N MY PLACE "+USER1.CITY+"\N AND NO IS" +USER1.PHONE)
```

//EXECUTION USING EC5 WITHOUT IT WILL SHOW ERROR.

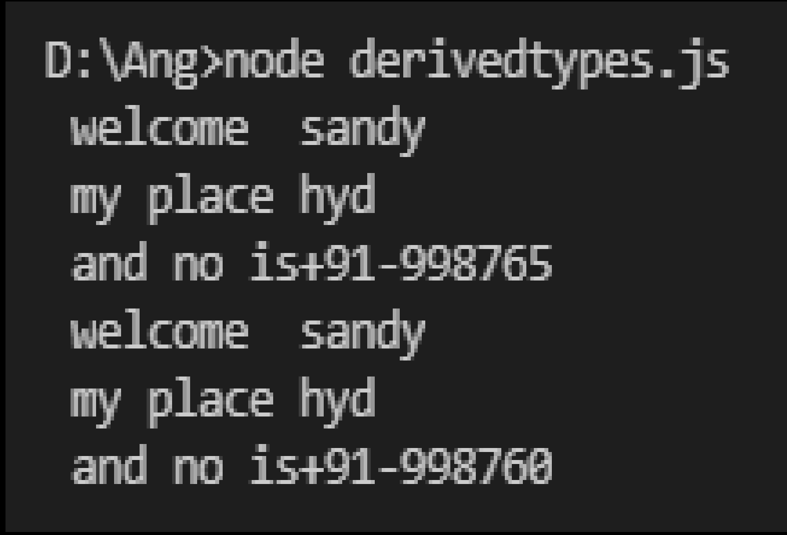
- TSC DERIVEDTYPES.TS --TARGET ES5

- NODE DERIVEDTYPES.JS

CALLING SETTER METHOD

```
CLASS USER{
    CONSTRUCTOR(PRIVATE UNAME:STRING,PRIVATE CITY:STRING,PRIVATE PHONE:STRING) //PRIVATE
{
    THIS.PHONE='+91-'+THIS.PHONE;
}
 //GETTERS OR ACCESSORS
 PUBLIC GET NAME()
 {
     RETURN THIS.UNAME; //CONSTRUCTOR NAME
 }
 PUBLIC GET CITY()
 {
     RETURN THIS.CITY; //CONSTRUCTOR NAME
 }
 PUBLIC GET PHONE()
 {
     RETURN THIS.PHONE; //CONSTRUCTOR NAME
 }
 PUBLIC SET PHONE(PH:STRING){
        THIS.PHONE='+91-'+PH;
 }
}

//USER1 -> OBJECT
VAR USER1 = NEW USER('SANDY','HYD','998765');
CONSOLE.LOG(" WELCOME  "+USER1.NAME +"\N MY PLACE "+USER1.CITY+"\N AND NO IS" +USER1.PHONE)
USER1.PHONE='998760'
CONSOLE.LOG(" WELCOME  "+USER1.NAME +"\N MY PLACE "+USER1.CITY+"\N AND NO IS" +USER1.PHONE)
```

- //OUTPUT   >TSC DERIVEDTYPES.TS
-            >NODE DERIVEDTYPES.JS



```
D:\Ang>node derivedtypes.js
 welcome  sandy
 my place hyd
 and no is+91-998765
 welcome  sandy
 my place hyd
 and no is+91-998760
```

FUNCTIONS IN CLASS

- EVERY TIME AM WRITING CONSOLE.LOG
- NOW I WANT TO USE FUNCTION,USING FUNCTION I WILL DISPLAY THE RECORDS.
- INSIDE THE CLASS WE ARE CALLING SO USING THIS. WILL USE.
- FUNCTIONS CAN ALSO APPIED FOR PASSING ARGUMENTS AND INCLUDE VOID.

FUNCTIONS….

```
CLASS USER{
    CONSTRUCTOR(PRIVATE UNAME:STRING,PRIVATE CITY:STRING,PRIVATE PHONE:STRING)
{
    THIS.PHONE='+91-'+THIS.PHONE;
}
 //GETTERS OR ACCESSORS
 PUBLIC GET NAME()
 {
     RETURN THIS.UNAME;
 }
 PUBLIC GET CITY()
 {
     RETURN THIS.CITY;
 }
 PUBLIC GET PHONE()
 {
     RETURN THIS.PHONE;
 }
PUBLIC SET PHONE(PH:STRING){
        THIS.PHONE='+91-'+PH;
 }
 PUBLIC DISPLAY():VOID
 {
   CONSOLE.LOG(THIS.NAME+" WELCOME HERE, DETAILS\N CITY : "+ THIS.CITY+" \NPHONE "+ THIS.PHONE);
 }
}

//USER1 -> OBJECT
VAR USER1 = NEW USER('SANDY','HYD','998765');
USER1.DISPLAY();
USER1.PHONE='0098765';
USER1.DISPLAY();
```

MODULES

- IF I WANT TO ACCESS THE USER CLASS IN OTHER MODULE THEN WE USE MODULE.
- WHICH EVER CLASS I WANT TO USE USER CLASS THEN WE NEED USE IMPORT
- WE NEED TO WRITE EXPORT IN USER CLASS.

MODULES USER CLASS EXPORT AND MODULE FILE IMPORT

- `EXPORT CLASS USER{`
- `    CONSTRUCTOR(PRIVATE UNAME:STRING,PRIVATE CITY:STRING,PRIVATE PHONE:STRING)`
- `{`
- `    THIS.PHONE='+91-'+THIS.PHONE;`
- `}`
- ….GO ON
- MODULE.TS FILE
- `IMPORT {USER} FROM './DERIVEDTYPES'`
- `VAR OBJ = NEW USER('JOSMINE','TRIVENDRAM','78956412');`
- `OBJ.DISPLAY();`

- //OUTPUT TSC MODULES.TS --TARGET ES5

ChandraSekhar(CS) Baratam

ChandraSekhar(CS) Baratam

ChandraSekhar(CS) Baratam