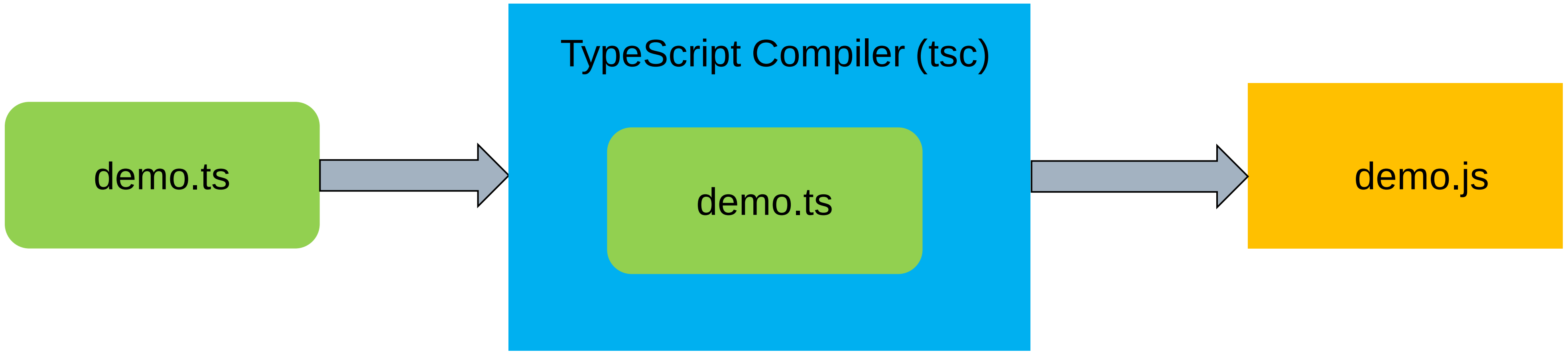
TypeScript

What is TypeScript

- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java.



Features

- TypeScript supports Static typing, Strongly type, Modules, Optional Parameters and more.
- TypeScript is object-oriented and supports features such as classes, interfaces, inheritance, generics.
- TypeScript provides compile time error-checking.
- TypeScript supports the latest JavaScript features, including ECMAScript 2015.
- TypeScript supports all the benefits of ES6.
- Developers can save a lot of time with TypeScript.

TypeScript Syntax

- Syntax defines a set of rules as how to define
  - Modules
  - Functions
  - Variables
  - Statements and Expressions
  - Comments

Your first TypeScript code

```
var message:string=“hi from TypeScript, Happy Learning”;
console.log(message);
```

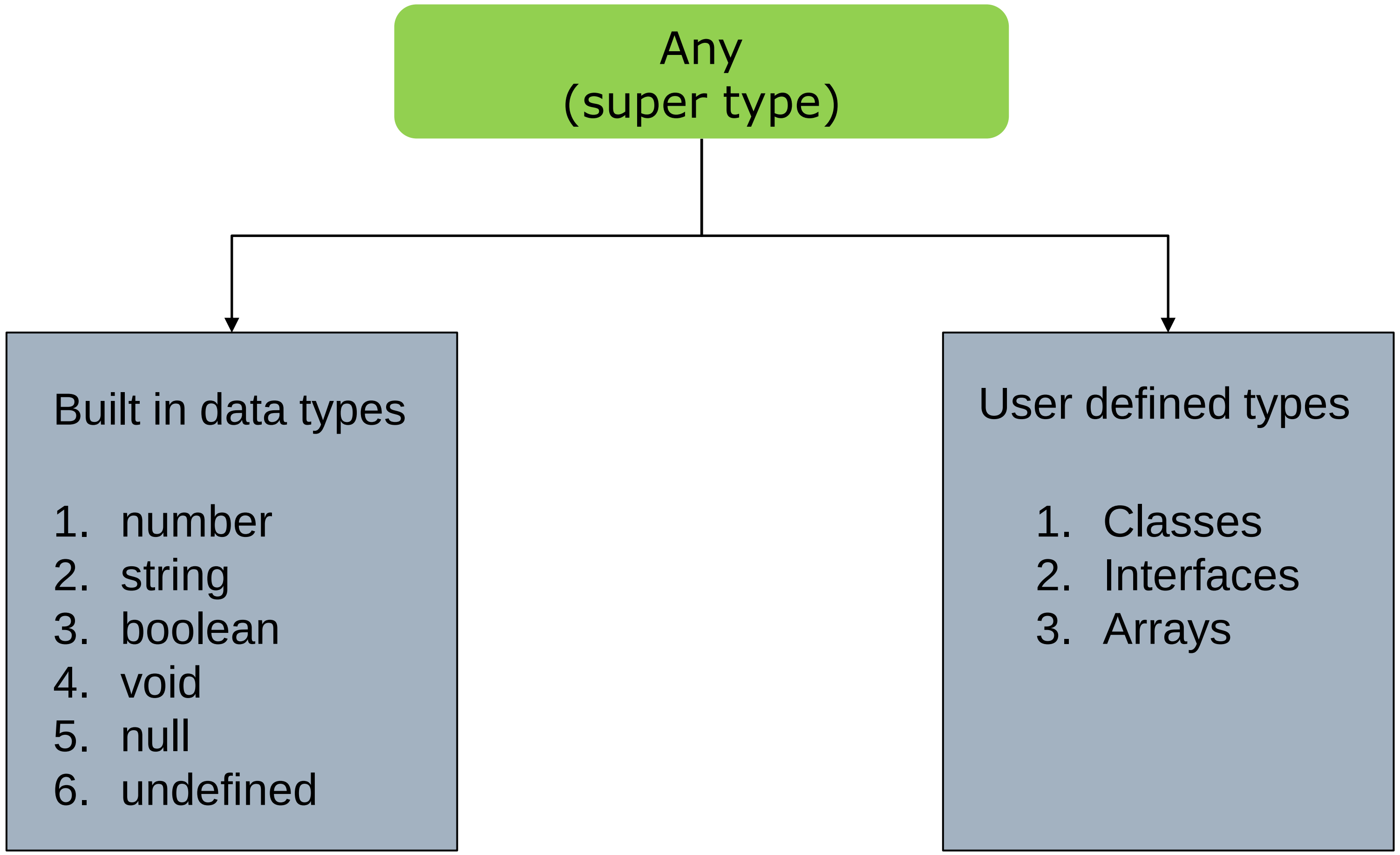
1. Create a folder “demots” in your filesystem
2. Open VisulaStudio Code IDE
3. Open the folder “demots” in Visual Studio Code
4. Create a file named **sample.ts** and paste the above code
5. Open a terminal from within Visual Studio Code and install typescript as

   ```
   npm install -g typescript
   ```
6. Compile “sample.ts” as

   ```
   tsc sample.ts
   ```
7. It will generate “sample.js”, you run it as

   ```
   node sample.js
   ```

Type System in TypeScript



How to declare variables

var [indentifier]:[type]=value; or var [indentifier]:[type]=value

```
var name:string = "Chethan";
var age:number = 25;
var marks:number = 67.9;
var details =name+" is "+age+" years old and got "+marks;
```

Basic Types

Boolean
true or false

```
let isDone: boolean = false;
```

Number
All numbers in TypeScript are either floating point values or BigIntegers.

```
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
let big: bigint = 100n;
```

String
we use the type string to refer to textual datatypes. TypeScript uses double quotes (") or single quotes (') to surround string data.

```
let color: string = "blue";
 color = "red";
```

Array
TypeScript allows you to work with arrays of values. Array types can be written in one of two ways.

```
let list: number[] = [1, 2, 3];
```

Alternatively, use a generic array type, Array<elementType>:

```
let list: Array<number> = [1, 2, 3];
```

Basic Types

Enum

An **enum** is a way of giving more friendly names to sets of numeric values.

```
enum Color { Red, Green, Blue, } ;
let c: Color = Color.Green;
```

By default, **enums** begin numbering their members starting at 0. You can change this by manually setting the value of one of its members. For example, we can start the previous example at 1 instead of 0:

```
enum Color { Red = 1, Green, Blue, }
let c: Color = Color.Green;
```

Or, even manually set all the values in the enum:

```
enum Color { Red = 1, Green = 2, Blue = 4, }
let c: Color = Color.Green;
```

How to define functions

function_name (param1[:type], param2[:type], param3[:type])

```
 getEmployeeDetails(empId:number,name:string,salary:number,email?:string):
<return_Type>{
    console.log(empId+' '+name+" "+salary+" "+email);
}


getEmployeeDetails(123,"Shantanu",45000);
getEmployeeDetails(123,"Shantanu",45000,"sbtalk@yahoo.com");
```

If any function parameter ends with ‘?’, then it becomes optional

Some examples of Control Structures

```
var marks = 80;
if (marks > 45) {
   console.log("You passed the exam");
}
```

```
var marks = 80;
if (marks > 45) {
   console.log("Yu passed the exam");
}else{
     console.log("You failed!!");
}
```

```
var num=2;
for(let i = num;i<=7;i++) {
   console.log("execution no "+i);
}
```

```
var num=0;
while(num <6) {
    console.log("in While loop"+num);
               num++;
}
```

```
var num=0;
do{
    console.log("in While loop"+num);
               num++;
} while(num <6) ;
```

Tuples in TypeScript

- In Arrays, we need to store values of the same type.
- Tuples allow us to store values of different types in single variable.
- Tuples are declared as :

  var tuple_name = [value1,value2,value3,…value n]

Example:
```
var emp=[123,"Shantanu","Hyderabad",45000,9.5];
console.log(emp[0]+" "+emp[1]+" "+emp[2]+" "+emp[3]+' '+emp[4]);
```

Operations in Tuple

push() appends an item to the tuple
pop() removes and returns the last value in the tuple

Interface

```
interface Person {
  name: string;
  age: number;
}

interface ReadonlyPerson {
  readonly name: string;
  readonly age: number;
}

let writablePerson: Person = {
  name: "Person McPersonface",
  age: 42,
};

// works
let readonlyPerson: ReadonlyPerson = writablePerson;

console.log(readonlyPerson.age); // prints '42'
writablePerson.age++;
console.log(readonlyPerson.age); // prints '43'
```

Abstract Class

```
abstract class Base {
  abstract getName(): string;

  printName() {
    console.log("Hello, " + this.getName());
  }
}

const b = new Base();

//Cannot create an instance of an abstract class.
```