1. Installing + Setting up the editor.  
     
   npm install -g typescript  
   tsc .\usingts.ts  
   Extensions : ESLint : code quality check support.

Material icon theme.  
Path Intellisense : better support when we work with imports.  
Prettier - Code formatter

1. npm install --save-dev lite-server : server index.html file [something like live-server]
2. core types javascript knows and typescript(lowercase) also supports:  
   number : +ve , -ve, floats  
   string : “abc”, `${template literals}`

boolean: true / false  
…objects and so on.

const add = (n1: number,n2: number) => {  
 return n1+n2;  
}  
const number1 = 5;  
const number2 = 2.8;  
add(number1,number2);

1. object types.  
     
   const person : {name:string;age:number} = {  
    name:"steve",  
    age:33  
   }  
   console.log(person.name)
2. Nested objects and types.  
     
   const person : {name:string;age:number;address:{ line1:string;line2:string;pin:number} } = {  
    name:"steve",  
    age:33  
   }
3. Array – can have arrays with strings and numbers mixed.

Types of the array can be flexible or strict.  
  
const person:{name:string;age:number;hobbies:string[]}= {

name:"steve",

age:33,

hobbies : ['sports','cooking']

}

for(const hoby of person.hobbies){

console.log(hoby.toUpperCase())

//typescript automatically detects that this is a string

}

person.hobbies.//map/pop/push/reduce... etc all will be listed automatically

1. Working with Tuples – only with TS  
   [1,2] – fixed length array + fixed type array  
     
   const person: { name:string; age:number; hobbies:string[]; role:[number,string] }= {   
   name:"steve",  
   age:33,  
   hobbies : ['sports','cooking'],  
   role:[2,'dev-eng'] // WHY ? here we wanted to have an array with exactly to elements  
   }

person.role.push(22);

person.role.push('admin'); // we only need two elements - but still TS does allow this - size can not controlled by typescript

// person.role = [1,'simple-role','another-fake-entry'] - when assigning it this way Ts complains about the third entry

person.role[1] = "test-eng"; // can switch values

// person.role[1] = 20; //this will be complained by typescript - the order of the types is strict.

**// So if you have scenario that there should be exactly only two elements in an array**

**// + And you know the type of each element in advance : TUPLE is the perfect solution**

1. Working with Enums(custom type in typescript) : There might global constants – which are represented as numbers, but for which you want to assign a label.

**BEFORE ENUMS : with constants**  
const ADMIN = 0;

const READ\_ONLY\_USER = 1;

const AUTHOR = 2;

const person = { name:"steve", role: READ\_ONLY\_USER }

if(person.role === READ\_ONLY\_USER){

console.log("The role is READ\_ONLY\_USER")

}

// The advantage of this is instead of using the numbers we use strings which are more readable

// The downside is ANY number can be stored + v have lot of constants and we have to manage them

role: 100  
  
**WITH ENUMS**enum Roles {

ADMIN,READ\_ONLY\_USER,AUTHOR

};

const person = { name:"steve", role: Roles.READ\_ONLY\_USER }

if(person.role === Roles.READ\_ONLY\_USER)

console.log("The role is READ\_ONLY\_USER")