Angular Framework

Angular is used to develop single page application, it uses components to create reusable interfaces which can be added in any part of the page, it uses mainly 2 technologies

1. HTML

2. Typescript - super set of Javascript

ECMAScript new features

1. New types of keywords let, const, class, constructor, extends, super
2. Template Strings
3. Arrow functions
4. Object Destructuring
5. Exponential Operator
6. String padding
7. Optional Chain
8. Rest & Spread operators

Arrow functions:

It is to simplifying writing the callbacks

Callback functions are the functions which are executed later

() => “Hello” // returns String

() => console.log(‘hello”) // no return value

(x, y) => x + y; returns x + y

(x, y) => {   
 //statements  
 return (x + y);  
}

forEach(callback)

map(callback)

Rest & Spread operators

Rest operator accepts 0 or more arguments, it must be the last formal argument in the function

function sum(x, y, …z) { }

Spread operator is used to distribute the values to the multiple arguments of the function or variables

String padStart & padEnd

These are used to add some special characters or even regular characters before or after the strings

Suppose we have 45678 to that we need to add \*\*\*\*45678 (or) 45678\*\*\*\*

padStart() & padEnd() functions part of String

Static keyword

Static members you can access directly by using class-names

class X {   
 a = 10;  
 static b = 20;  
  
 static demo() {   
 // print some message  
 }  
 test() {   
   
 }  
}

let x1 = new X(); x1.test();  
X.demo();  
 x1.demo(); // this says undefined/not a function  
 // x1.b >> gives undefined  
// X.b >> works  
// X.a >> undefined  
// x1.a >> works  
// Access a & b outside the class

Optional Chain:

It is used to access properties with a condition whether its present or not.

let users = [ { firstname: “Raj”, email:”raj@gmail.com”}, { firstname:”Vijay”}, { firstname:}];

Callback: The action is initiated now but will be completed later, these are useful to perform asynchronous task, in Javascript there are many methods that take call-backs to perform asynchronous tasks like

setTimeout(callback, time), XMLHttpRequest.onreadstatechange

Promises: It is used to perform asynchronous task in a simplified way without using nested callbacks, a Promise can be successful or unsuccessful which are called as resolved / rejected, based on this Promise objects takes a callback with 2 arguments

Promise(callback(resolve, reject) { } )

When promise is resolved you can use the callback of .then() if its rejected then you can use callback of .catch()

Async and Await: These are keywords used in asynchronous operations, it waits for the result from the Promise.

Note: await must be used inside async function



Output:



Typescript:

It is a super set of Javascript which is more reliable than Javascript, it can identify early errors because it is first compiled.

TS -> compiled to -> JS

Installing typescript compiler

>> npm install typescript

[OR]

>> npm install -g typescript

How to compile typescript

>> tsc file.ts

>> tsc --target es6 file.ts

Problems with Javascript

* Javascript data is not reliable, means results are sometimes unpredictable, i.e., a + b can either add numbers or concatenate strings
* Javascript functions can return any kind of value there’s not restriction
* Javascript is not compiled, so you can’t find any errors until you execute

Typescript

* It supports all the features of Javascript
* It has additional features which are not supported by Javascript but it needs to be converted to Javascript to make browser to understand the code.
* It provides following features
  + Types to Variables
  + Types to Return types
  + Tuples, Union, enum types
  + Access modifiers like private, protected & public
  + Short-cut constructor initialization
  + Readonly properties

In Javascript

function add(x, y) { return x + y; }

add(2, 3) [or] add(“hello”, 123)

In Typescript we have types like string, number, [], boolean, any

function add(x: number, y: number) : number {   
 return ( x + y )  
}

Now if you call

add(20, 30); // it works

add(“Hello”, 123); // it raises compilation error

List of datatypes

void

number

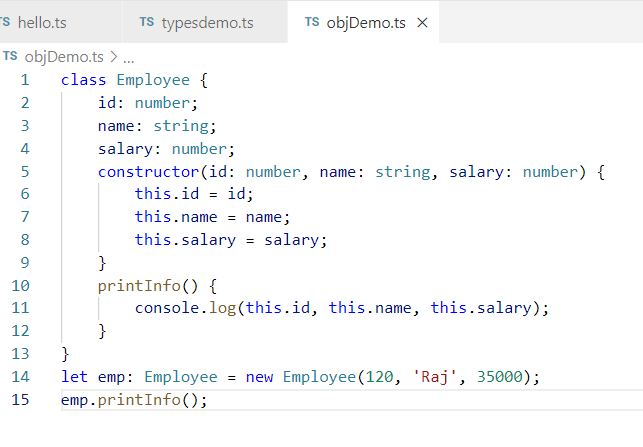
string

boolean

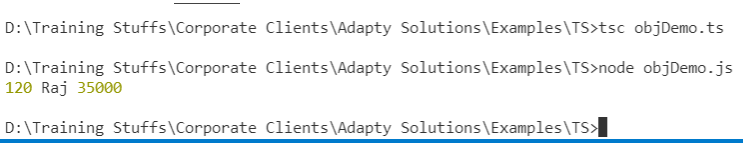
any

enum

objDemo.ts



Output:



Access Modifiers:

These are the keywords that determine the visibility of the members, there are 3 access modifiers

1. private
2. protected
3. public

private: Visible only within the class

public: Visible within the class & outside the class

protected: Visible within the class & only to the sub-class

class X {   
 private a: number;  
 protected b: number;  
 public c: number;  
   
 abc() {   
 can access this.a, this.b & this.c  
 }   
}   
  
class Y {  
 X x1 = new X();  
 can access only x1.c  
}  
  
class Z extends X {  
 // inherits b & c both

can access this.b & this.c  
}