Library: React (Java Script) – create a beautiful website – type Script

(A tool that provides specific functionality)

Frameworks (technology) - Angular, Vue

(set of tools and guidelines for building apps)

Codesandbox.io

Component-based – REACT- use named function – at last export the same name function

Break UI into different component

Components folder

App.tsx

Index.tsx or main (beginning of your Project)

REACT - Change HTML when the user interact

JSX – a combination of HTML in a specific way

npm install (need node.js)

App.tsx

Inside index.html, it is important to have this id: main.tsx

*id*="root"

How React works:

ReactDOM(virtual DOM) will handle, create a copy and replace real DOM,

TypeScript –

Dynamically (JS, Python…) or statically typed language(JAVA, C, C++, Rust, Go)

Require compilation

Compile down to JS

Type inference

Type annotation

Production code

Let id: number = 5

Let company: string = ‘Coca’

Let isPublished: boolean = true

Let x: any = ‘Hello’

Let ids: number[] = [1, 2, 3, 4, 5]

//Tuple

Let person: [number, string, boolean] = [1, ‘Brad’, true]

//Tuple Array

Let employee: [number, string][]

Employee = [

[1, ‘Brad’],

[2, ‘Julie’],

[3,’ ‘Dom’]

]

//union (accept more than one type, one or other

Let pid: string | number

Pid = ‘22’

//enum = Set of named Constance can be string or numerical

Enum Direction1{

Up = ‘up’,

Down = ‘down’,

Left = ‘Left’,

Right = ‘Right’,

}

Enum Direction2{

Up,

Down,

Left,

Right,

}

console.log(Direction2.Up) will return 0 or you can change to Up =1

//Object

type User = {

id: number

name: string

}

const user: User = {

id: 1,

name: ‘John’,

}

// Type Assertion = telling the compiler

Let cid: any = 1

Let consumerId = <number>cid //one way

Let costumerId = cid as number //other way

//function

Function addNum(x: number, y: number): number {

Return x + y

}

Console.log(addNum(1,2))

// Void

Function log(message: string | number): void{

Console.log(message)

}

// Interface (use with object but cannot use with union

interface UserInterface = {

id: number

name: string

age?: number //? Means optional

}

const user1: UserInterface = {

id: 1,

name: ‘John’,

}

//readonly property

interface MathFunc {

(x: number, y: number): number}

const add: MathFunc = (x: number, y: number): number => x+y

const sub: MathFunc = (x: number, y: number): number => x-y

// create multiple object – use Class

class Person {

id: number

name: string

constructor(id: number, name: string){

this.id = id

this.name = name

}

Register(){

Return`${this.name} is now registerd`

}

Console.log(brad.register())

Const brad = new Person(1, ‘Brad’)

Const mike = new Person(2, ‘Mike’)

//by default is public

//protected extended class or in the same class

//private = can be accessed just from the same class

Interface PersonInterface {

Id: number

Name: string

Register(): string

}

Class Person implements PersonInterface

Id: number

Name: string

constructor(id: number, name: string) {

This.id = id,

This.name = name

}

//Subclasses

Class Employee extends Person {

Position: string

Constructor(id: number, name: string, position: string) {

Super(id, name)

This.position = position

}

}

CONST EMO = NEW Employrr(3, ‘jack’, ‘Developer’)

Console.log(emp.register())

// Generics – create reusable component, can be used as a place holder to define a data type after

Function getArray<T>(items: T[]): T[] {

Return new Array().concat(items)

}

Let numArray = getArray<number>([1,2,3,4])

Let strArray = getArray<string>([‘brad’, ‘John’, ‘Jack’])

strArray.push(‘hello’)

numArray.push(1)

concepts

useState – use to change value

useEffect

receive bundles as response from the server

lists of dependencies

session2 16/01/2024

convert HTML to JSX

1. All property names follow camelCase
2. Number attributes use curly braces
3. Boolean ‘true’ can be written just the property name. ‘False’ has to be written
4. The ‘class’ attribute is written ‘className’
5. In-line style need to provided as object {{ }}

Event System

State System

useState

import React from ‘react’;

function App() {

return (

<div className=”wrapper”>

<textArea

readOnly

maxLength={3}

spellcheck

style={{background-color: “gray;”}}

/>

</div>

);

}

export default App;

Relative path to walk from one file to other file ‘./filename’