Typescript

Это расширение над языком javascript, которое вводит типизацию. Расширение \*.ts. Реактивная компонента будет иметь расширение \*.tsx. Реактовое приложение на typescript’е создается следующей командой – npx create-react-app --typescript name. Приложение написанное на ts поддерживает в том числе и js файлы. Не имеет смысла переписывать полифилы (написанная на старом стандарте новая функция языка). Главная и практически единственная особенность – переменным необходимо задавать тип. Тип точно также может быть определен значением.

Основные типы данных:

1. number
2. string
3. boolean
4. undefined
5. null
6. any (лучше не должно быть)
7. never (нафиг надо?)
8. void (функция, которая ничего не возвращает)
9. HTMLElement (HTMLDivElement и тд)
10. number[ ] – массив

Для констант и переменных, а также для аргумента функции тип объявляется через двоеточие.

var a=7;

var b:number;

var c:number=13;

Самописные типы данных:

В ts можно задавать свои типы и структуры данных, самый просто это структура данных в которой находится 2 значения.

let m:’distance’|’lumen’|undefined;

type TPoint = ‘name’|’surname’|’sex’|;

=number[][][];

={

x:number;

y:number;

z:number;

}

const p:TPoint;

Могут быть обязательные и не обязательные свойства в типе данных. Необязательность свойства задается таким образом z?:number; Мы указали, что в типе TPoint свойство z может быть не указано, а вот x и y должны быть в любом случае.

function f1(a:number, b:string=’’):void{

//void – ничего не возвращает, а так может быть, что угодно

}

const f2=():boolean=>{}

type TPoint = {

…

f1:(a:number, b:number)=>boolean;

}

Наследование типов

type TLight = TPoint & {lumen: number};

У типа данных можно удалить свойства

type TPoint2D = Omit<TPoint, ‘z’>

Перечисление

enum ESex{

male=’male’,

female=’female’,

}

enum EPages{

Calc,

Graph2D,

Graph3D,

}

Тип данных interface

interface IMorda{

a:number;

b:string;

c?:Boolean;

}

Интерфейсы могут наследоваться друг от друга с переопределением свойств.

Их можно имплементировать в классы, это делает обязательным прописывание свойств в классе указанных в интерфейсе.

interface ICalculator{

add(a:number, b:number)=>number;

…

}

class Calculator implements ICalculator{…}

Generic – передаваемый тип данных, который может быть использован в структуре данных либо в функции.

function sum<T>(a:T, b:T):T{

return a+b;

}

sum<number>(7,-8);

App.tsx

export enum Epages{

Calc,

Graph2D,

Graph3D,

}

const App:React.FC =() =>{

const[pageName,setPageName]=useState<EPages>(EPages.Calc);

{pageName===EPages.Calc && <Calc/>}

Header.tsx

type THeader={

setPageName:(name:EPages)=>void;

}

const Header:React.FC<THeader>=({setPageName})=>{…}

Домашка: Переписать на тайпскрипт Граф2Д и эссе и всякую хрень, но не Граф3Д и Кальк. К зачету лучше переписать на Тайпскрипт.