Semana 18.2

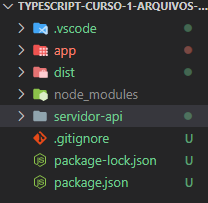
TypeScript parte 1 - evoluindo seu Javascript

npm install -> instalar o npm em seu projeto

npm install typescript@4.2.2 --save-dev -> instalando o typescript na versão desejada

agora basta fazer as configurações para que seu navegador possa enterder essa linguagem

todos os códigos TS que você for escrever, serão colocado dentro da pasta app, quando esse código estiver pronto iremos transformar esse código TS em JS



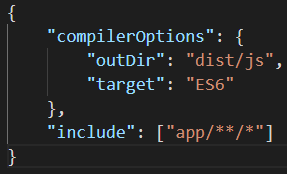
pasta dist onde ficará tudo o que o seu navegador conseguirar ler, serão os arquivos que foram compilados da pasta app que estão em TS, que irão ser compilado para o JS

Pasta app, todas as informações(codigos) referente a typescript

Iremos mostra como configurar para que os arquivos compilados de TS aparecam na pasta dist já com o formato de JS

Crie na raiz do seu projeto um novo arquivo chamado

tsconfig.json



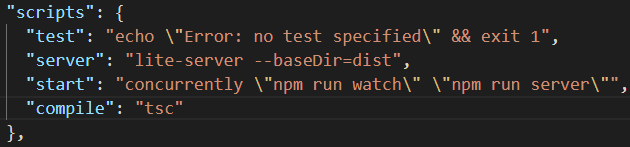
Dentro desse novo arquivo criado coloque esse código para fazer as configurações básicas de seu projeto

outdir -> local onde será feito a saída de compilação de seus arquivos TS para o formato JS(que o navegador poderar ler)

target -> indica com qual versão esse projeto será compatível

include -> de onde será pego os arquivos TS para fazer a compilação

depois disso dentro do arquivo package.json, dentro de scripts coloque o código de compile





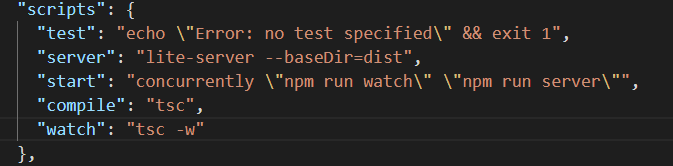
Agora para seus projeto rodar o TS, basta colocar o código para rodar através desse compile(poderia ter sido criado qualquer nome)

npm run compile

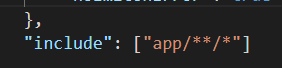
dessa maneira cada vez que esse compile e executado ele pega os arquivos TS que estão dentro da pasta app, compila e coloca dentro da pasta dist/js/<respectivo\_nome>.

E tem um arquivo chamado teste.ts ele irá criar uma arquivo teste.js dentro dessa pasta, independete se ocorra um erro ou não em seu código, seu código só será compilado quando for chamado o npm run compile e seu servidor for reiniciado pelo npm run server

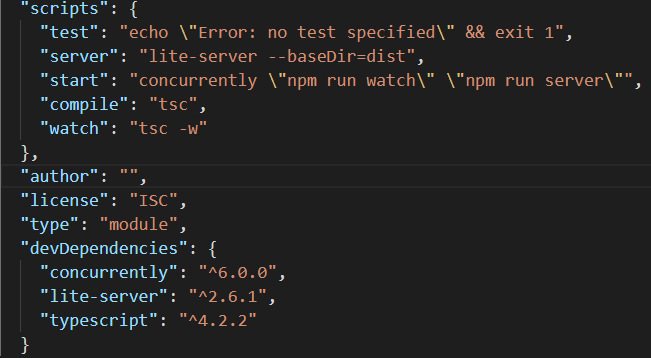
Para automatizar essa automação basta ir no arquivo package.json e criar um novo script, watch(poderia ser qualquer nome) passando a irformação de tsc -w



Com isso ele irá ficar ficar observando todos os arquivos que forem alterados de seu diretório TS, ele ficará monitorando os arquivos da pasta que forem passada como parâmetro do include do tsconfig



Para que cada alteração feita em nosso projeto seja feita a compilação de projeto e nosso servidor seja reiniciado altomaticamente, em nossa pasta package.json temos a extenção concurrently que consegue executar duas funções ao mesmo tempo em um script, e dentro do script start estamos jogando dois npm run, o que fica monitorando cada alteração feito em nossos arquivos TS e transformando em JS e temos o run que fica observando canda alteração em nosso servidor e dando um reflech sem que seja necessário ficar fehando e iniciando um novo server





private \_data;

modificador para chaves privadas, que não podem ser modificadas, a não ser em sua classe

agora veremos os tipos para nossos valores

private \_data;

quando não declaramos nada em \_data, essa propriedade receberá o tipo any(qualquer coisa), logo ele poderá receber letras, números, símbolos qualquer coisa. Seu código ficará mais rápido de escrever porem deixará brechas para erros

private \_data: Date;

dessa maneira a propriedade \_date será do tipo data, e só poderá receber valor do tipo data, se não apresentará erro

 private inputData: any;

também e possível dizer explicitamento que uma propriedade e do tipo any

private inputData: HTMLInputElement;

propriedade do tipo elemento de input do html





Quando colocamos no tsconfig.json o campo noImplicitAny, estamos dizendo para nosso código que não queremos nenhuma case do tipo any em nosso código, todas as propriedades deverão ter um tipo definido

ERROS DE TIPOS

private inputData: HTMLInputElement;

aqui estou dizendo que esse tipo e um elemento de input

this.inputData.value,

quando coloco para ele pegar o value, ele irá retornar um tipo de string, não sendo o valor que e o valor do tipo verdadeiro, logo apresentará um erro.

Para resolver esse erro temos que converter esse valor

adiciona() {

        const exp = /-/g;

        const date = new Date(this.inputData.value.replace(exp,','));

        const quantidade = parseInt(this.inputQuantidade.value);

        const valor = parseFloat(this.inputValor.value);

        const negociacao = new Negociacao(date,quantidade,valor);

    }

Dessa maneira estamos convertendo os valores que foram obrigatoriamente recebido pelo input e convertendo para os valores de tipos que desejamos

criaNegociacao(): Negociacao {

        const exp = /-/g;

        const date = new Date(this.inputData.value.replace(exp,','));

        const quantidade = parseInt(this.inputQuantidade.value);

        const valor = parseFloat(this.inputValor.value);

        return new Negociacao(date, quantidade, valor);

    }

Tipo o tipo do método, o método criaNegociacao retornará um tipo Negociacao

criaNegociacao(): Negociacao {

        const exp = /-/g;

        const date = new Date(this.inputData.value.replace(exp,','));

        const quantidade = parseInt(this.inputQuantidade.value);

        const valor = parseFloat(this.inputValor.value);

        return valor;



    }

dessa maneira o TS apresentará um erro, pois o valor retornado não e um tipo negociacao, logo seu código ficará mais seguro



import { Negociacao } from "./negociacao.js";

export class Negociacoes {

    private negociacoes: Array<Negociacao> = [];

}

Estou criando uma classe que recebe uma propriedade do tipo array, e no typescript precisamos informar quais são os tipos que essa array irá receber, nesse caso essa array irão receber o tipo Negociacao, que estão sendo importada da sua pasta

lista(): ReadonlyArray<Negociacao> {

        return this.negociacoes;

    }

Tipo de lista que so será de leitura, impedido que algo seja apagado

Agora iremos ver como podemos declarar no próprio construtor

Aprendemos a declarar dessa maneira

export class Negociacao {

    private \_data: Date;

    private \_quantidade: number;

    private \_valor: number;

    constructor(data: Date, quantidade: number, valor: number) {

        this.\_data = data;

        this.\_quantidade = quantidade;

        this.\_valor = valor;

    }

}

Agora podemos declarar dessa maneira que irá ser o mesmo jeito

export class Negociacao {

    constructor(

        private \_data: Date,

        private \_quantidade: number,

        private \_valor: number

    ) {}

}

Quando tivermos uma propriedade fora do construtor podemos fazer do método antigo mesclado com esse

export class Negociacao {

private caopsitaName = 'Calo';

    constructor(

        private \_data: Date,

        private \_quantidade: number,

        private \_valor: number

    ) {}

}

Podemos simplificar também array

Maneira antiga

export class Negociacoes {

    private negociacoes: Array<Negociacao> = [];

nova maneira, colocando os colchetes em vez do array mais o diamante <>

export class Negociacoes {

    private negociacoes: Negociacao[] = [];

também podemos modificar o readonly

antiga

lista(): ReadonlyArray<Negociacao> {

        return this.negociacoes;

    }

Nova maneira tirando o readonlyarray mais o diamante e colocando somente o readonly mais os colchetes

lista(): readonly Negociacao[] {

        return this.negociacoes;

    }

Colocando a propriedade de uma classe como public, mais que só poderá ser inserida um valor quando for instanciada e seu valor só poderá ser de leitura

Não será possível pegar essa propriedade e trocar o seu valor

export class Negociacao {

  constructor(

        public readonly data: Date,

        public readonly quantidade: number,

        public readonly valor: number

    ) {}

    get volume(): number {

        return this.quantidade \* this.valor;

    }

}

Programação defensiva

 get data():Date {

        const data = new Date(this.\_data.getTime());

        return data;

    }

Evitando que quando chame um método ou get de sua classe, você tenha acesso a uma referencia de algo que eu guardei para ninhem botar a mao mais que eu não quero que você modifique, eu clonei essa data então qualquer modificao será feita na copia da data (data) e não na propriedade principal (\_data)