**Instalando o Node**

Entrar no site oficial do Node atualmente é o node.org

Ir no botão de instalação e baixar a versão LTS que é a mais recomendada.

Baixe e acordo com seu sistema operacional.

**Testando o Node e o NPM no terminal e verificando versões**

Comando: node -v

Comando: node --version

Comando:npn -version

**Criando o primeiro projeto**

No terminal na pasta do projeto roda o comando:

Comando: npm init

Próximo passo é só dar enter nas perguntas relacionadas ao projeto ou nomear se desejável.

Após todos os passos sucedidos criara um arquivo chamado package.json que é os dados básicos informados relacionados ao projeto.

Caso queira ignorar as perguntas relacionadas ao projeto rode o comando abaixo:

Comando: npm init -y

O arquivo package.json é onde estará todas as informações e bibliotecas importantes do projeto.

**Executando um código js no terminal.**

**Comando:** node nomeDoArquivo.js

**Instalando o Typescript no projeto**

Comando: npm install -g typescript

**Iniciando o typescript**

Comando: tsc --init

**Separando os arquivos por pastas**

Criar a pasta dist: Onde ficara os arquivos de distribuição (compilado, traduzido pelo js)

Criar a pasta src: Onde ficara o código fonte do projeto

**Instalando o arquivo de configuração Typescript:**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Todas a configurações do typescript estão armazenadas no arquivo **tsconfig.json**

**Dentro deste arquivo definidos algumas configurações importantes para o nosso projeto**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**target é onde dizemos a versão do ecmascript que queremos usar para traduzir o arquivo js para a versão desejada.**

**ModuleResolution é onde dizemos para o nosso projeto que vamos trabalhar com node, para não ter variações de comportamento é uma boa pratica informar esta configuração, caso contrário deixar no classic.**

**Após devemos definir as pastas (src, dist)que criamos para o nosso projeto:**

**Tela de celular

Descrição gerada automaticamente com confiança média**

**Descomentar o rootDir e informar que a pasta do projeto é a ‘src’**

**Descomentar o outDir e informar que a pasta final do projeto é a ‘dist’**

**Salve e o projeto já estará configurado.**

**Instalando uma dependência de inteligência para o typescript (auto complete e etc)**

O comando: npm install --save-dev @types/node

--save-dev informa que a biblioteca será apenas para desenvolvimento para criar o projeto.

Vai criar a pasta node\_modules onde contém os módulos Node e as bibliotecas que baixamos, esta pasta não pode ser alterada, é a pasta mais pesada do projeto, pode conter vários arquivos.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Não incluir a pasta node\_modules no repositório, pois ela pode ser criada novamente quando baixado o projeto, esta biblioteca esta nas dependências do package.json, com o comando **npm install** a pasta node\_modules será criada.

**Deixando o typescript monitorando a pasta do projeto Wath mode**

Comando: tsc -w

Deve deixar o typescript monitorando a pasta do projeto, para sempre que houver modificações ele cria na pasta dist o código onde contém o js puro.

Para isso vamos abrir outro terminal ao lado e digitar o comando > tsc -w , após isso ele ficará executando e monitorando as modificações que fizer no typescript. O comando tsc -w significa wath mode.

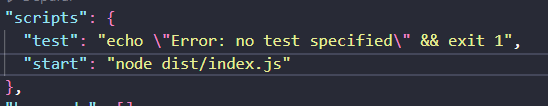
Deve rodar o node em outro terminal e deixar o wath mode monitorando ao lado.

Para rodar o node usa o comando: node dist/index.js

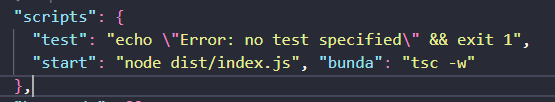
**Scripts do package.json**

Package.json e não pode ser alterada, dentro desta pasta no “scripts” podemos colocar atalhos para rodar(scripts de atalho) comandos personalizados. Exemplo abaixo colocamos o comando “"start": "node dist/index.js"

Executamos esse comando no terminal como> npm run start , o comando start já existe como no caso de >npm start é a mesma coisa. Só pra saber podemos personalizar outros comandos no scripts.



Pode criar outro atalho por exemplo para iniciar o wath mode do tyscript com o comando bunda.



Pode criar atalhos que executam vários códigos, que rodam ouros atalhos por exemplo vários comandos usando o && entre os comandos dentro da virgula.

**Entendendo o import/export em Commonjs (usando de forma padrão)**

Exportar o conceito é habilidade para uso em outros arquivos

Primeiro habilitamos para uso externo o arquivo e depois importando para o arquivo atual

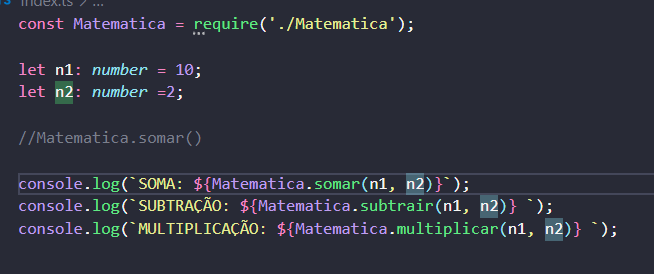
Imagem: arquivo Matemática.js

Texto

Descrição gerada automaticamente

Vou para o arquivo que eu quero importar uso o const para importar o arquivo

Imagem: arquivo index.js



Roda com o comando node dist/index.js

Resultado:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Técnicas mais modernas de fazer essas importação usando o ES6**

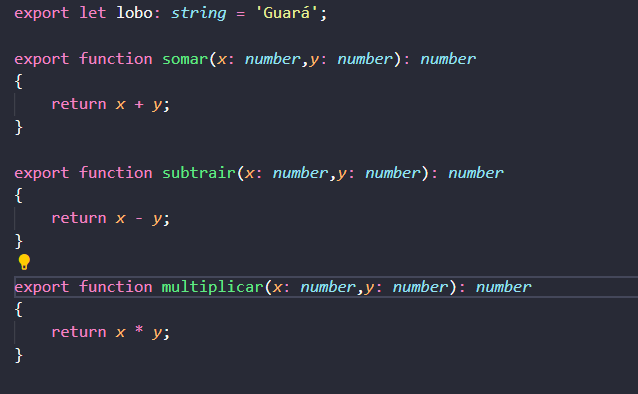
Apenas colocando o export antes da variável ou função o que quiser exportar

Exportando uma variável :



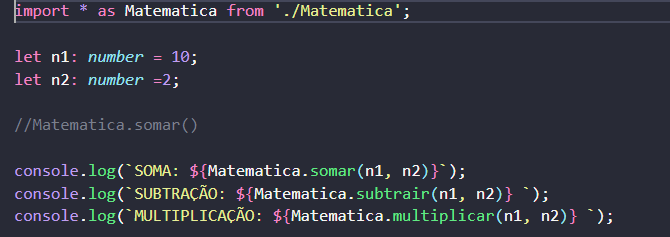
Habilitando para uso externo(exportando) um variável e três funções:

Imagem arquivo: Matematica.ts



Importando tudo:

Imagem: aquivo index.ts



Importando apenas algumas funções:

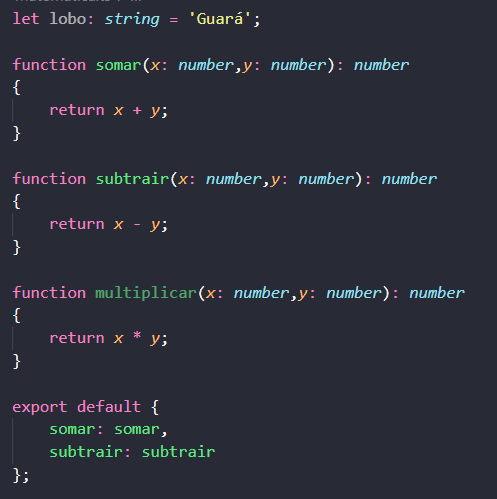
Imagem: arquivo index.ts

Texto

Descrição gerada automaticamente

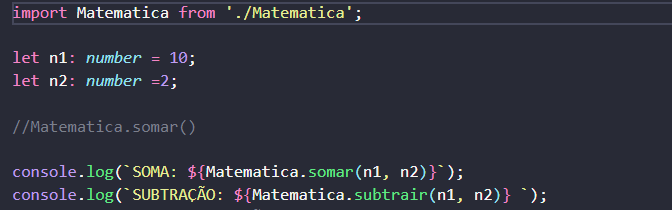
Exportando apenas uma única coisa:

Imagem: arquivo Matematica.ts



Importando padrão

Imagem: arquivo index.ts



**Instalando biblioteca na comunidade npm.js**

Instalando a biblioteca validator

<https://www.npmjs.com/package/validator>

para instalar digite o comando:

comando: npm install validator

usando a biblioteca

no arquivo index.js devemos importar a biblioteca

import validator from ‘validator’;

**instalando as declarações da biblioteca validator**

npm install --save-dev @types/validator