

# Como instalar typescript no vscode

---

## 1. Introdução

1. [Objetivo.](#)
2. [Pre-requisitos.](#)
3. [benefícios.](#)

## 2. Instalar

1. [Instalar o compilador TypeScript.](#)
2. [Instalação do pacote npx](#)
3. [Instalação do pacote webpack](#)
4. [Instalar a ferramenta tsc-init](#)
5. [Instalar a ferramenta ESLint](#)
6. [Instalar a ferramenta TypeDoc](#)
7. [Instalar extensões vscode.](#)
8. [Instalar extensão better-comments](#)
9. [Instalar extensão codeMap = Visualizar classes](#)
10. [Auto Import](#)

## 3. Configurar o compilador TypeScript

1. [Arquivo tsconfig.json](#)
2. [Compilador versus serviço de linguagem](#)
3. [Como configurar compilador tsc \(typescript\) para executar e depurar com vscode?](#)
  1. [Configuração do arquivo tsconfig.json;](#)
  2. [Configuração do arquivo tasks.json](#)
  3. [Teclas de atalho do vscode.](#)

## 4. Exemplos.

## 5. Referências.


## 6. Histórico.

---

## 2. CONTEÚDO


### 1. Introdução

#### 1. Objetivo:


1. Este documento descreve como instalar e configurar typescript no vscode.
2.  [\[BACK\]](#)

#### 2. Pre-requisitos:

1. VsCode precisa estar instalado e configurado o básico.

2. Familiaridade com ide vscode.
3. Nodejs precisa estar instalado e configurado.
4. Conhecimento da linguagem typescript.
5. Conhecimento da linguagem javascript.
6. 

### 3. Benefícios:

1. Typescript permite criar código javascript isento de erros de sintaxe.
2. 

## 2. Instalar

### 1. Instalar o compilador TypeScript

1. O Visual Studio Code inclui suporte à linguagem TypeScript, mas não inclui o compilador TypeScript **tsc**.. Você precisará instalar o compilador TypeScript globalmente ou em seu espaço de trabalho para transpilar o código-fonte do TypeScript para JavaScript ( **tsc HelloWorld.ts**).

1. A maneira mais fácil de instalar typescript é através **npm**(Node.js Package Manager).
2. Se você tiver o **npm** instalado, poderá instalar o TypeScript globalmente (**opção -g**) em seu computador:

#### 1. ShellScript

```
#!/bin/sh
// instalação global
sudo npm install -g typescript
```

3. 

### 2. Instalação do pacote **npx**

1. O programa **npx** executa programas da pasta local **node\_modules/.bin** ou central, instalando todos os pacotes necessários para que o programa funcione.
2. O npx é um package runner do NPM. Ele executa as bibliotecas que podem ser baixadas do site npmjs. Essas bibliotecas ficam em um banco de dados chamado NPM Registry, que também podem ser baixadas via CLI com npm ou yarn e com npx. [Veja mais....](#)
3. Código ShellScript

```
#!/bin/sh
sudo npm install -g npx
```

4. 

### 3. Instalação do pacote **webpack**.

1. webpack é usado para compilar módulos JavaScript. Depois de instalado , você pode interagir com o webpack a partir de sua [CLI](#) ou [API](#).
2. Código ShellScript para instalação global:

```
sudo npm install -g webpack
```

3. Código ShellScript para instalação local por projeto:

```
#!/bin/sh
mkdir webpack-demo // cria pasta do novo projeto
cd webpack-demo    // move-se para a pasta no novo
projeto
npm init            // Cria o arquivo package.json do
projeto.
npm install webpack webpack-cli --save-dev
```

4. 

### 4. Instalar a ferramenta **tsc-init**.

1. O programa **tsc-init** é uma ferramenta para inicializar TypeScript e [Webpack](#) em seu projeto. Veja [webpack getting-started](#).
2. Código ShellScript para instalação global:

```
npm install tsc-init -g
```

3. Código ShellScript para instalação local por projeto:

```
npm install tsc-init
```

4. O programa **tsc-init** executa os seguintes comandos na pasta raiz do projeto:

1. Execute [npm init](#) para criar o arquivo package.json;
2. Execute [tsc --init](#) para criar um arquivo tsconfig.json;
3. Adicionar [webpack](#), [ts-loader](#) e TypeScript como dependências de desenvolvimento;

1. [Webpack - Curso rápido para iniciante](#)

4. Adicione [Karma](#), [karma-jasmine](#), [Karma-webpack](#) como dependências de desenvolvimento;
  5. Crie um arquivo [webpack.config.js](#) para incluir [ts-loader](#);
  6. Crie um arquivo [karma.conf.js](#);
  7. Crie um arquivo [.gitignore](#);
  8. Execute [git init](#);
  9. Adicione scripts [npm](#) para construir e agrupar;

5. [\[← BACK\]](#)

## 5. Instalar a ferramenta [ESLint](#):

1. O ESLint é uma ferramenta para identificar e relatar os padrões encontrados no código ECMAScript/JavaScript, com o objetivo de tornar o código mais consistente e evitar bugs.
2. Código ShellScript para instalação local:

```
npm install --save-dev eslint
npm install --save-dev @typescript-eslint/parser
npm install --save-dev @typescript-eslint/eslint-plugin
```

3. Veja como usar ESLint com TypeScript...

1. [Configurando ESLint](#)
  2. [Familiarize-se com as opções de linha de comando.](#)
  3. [Integração do ESLint com editores de textos.](#)
  4. [Como usar ESLint com TypeScript](#)

4. [\[← BACK\]](#)

## 6. Instalar a ferramenta [TypeDoc](#)

1. O programa [TypeDoc](#) converte comentários do código-fonte TypeScript em documentação HTML renderizada ou um modelo JSON.

## 2. Código ShellScript:

```
# Install
npm install typedoc

# Execute typedoc on your project
npx typedoc src/index.ts

# Use os argumentos --out ou --json para definir o
formato e o destino de sua documentação.
typedoc --out docs src/index.ts
```

3. [← BACK]

7. .

8. [[← BACK ← BACK]]

## 2. Extensões do vscode.

1. A extensão [ESLint](#) usa a biblioteca **ESLint** instalada na pasta da área de trabalho aberta. Se a pasta não fornecer um, a extensão procurará uma versão de instalação global. Se você não instalou o ESLint local ou globalmente, faça-o executando **npm install eslint** na pasta do espaço de trabalho para uma instalação local ou **npm install -g eslint** global.
2. A extensão [Visual Studio IntelliCode](#) fornece recursos de desenvolvimento assistido por AI para desenvolvedores Python, TypeScript / JavaScript e Java no Visual Studio Code, com insights baseados na compreensão do contexto do código combinado com o aprendizado de máquina.
3. A extensão [JavaScript \(ES6\) code snippets](#) contém trechos de código para JavaScript na sintaxe ES6 para o editor de código Vs (suporta JavaScript e TypeScript).
4. A extensão [TypeScript Importer](#) procura automaticamente por definições de TypeScript em arquivos de espaço de trabalho e fornece todos os símbolos conhecidos como itens de preenchimento para permitir o preenchimento de código.
5. A extensão [Latest TypeScript and Javascript Grammar](#) é o ramo de desenvolvimento da colorização VSCode JS/TS. Projeto gitHub: <https://github.com/Microsoft/TypeScript-TmLanguage/issues>.
6. A extensão [Paste JSON as Code](#) Copy JSON, paste as Go, TypeScript, C#, C++ and more.
7. A extensão [TSDoc Comment](#) é uma extensão para converter comentários simples no estilo C/C++ em comentários no estilo TSDoc
8. [← BACK]

## 3. Configurar o compilador TypeScript:

1. Arquivo **tsconfig.json** na raiz do projeto é usado para customizar o comportamento padrão do compilador **tsc**:

1. Ter um arquivo **TSConfig** em uma pasta indica que esta pasta é a raiz de um projeto TypeScript ou JavaScript. O arquivo **TSConfig** pode ser um *tsconfig.json* ou um *jsconfig.json*, os dois possuem o mesmo comportamento e as mesmas variáveis de configuração.

1. Arquivo **tsconfig.json** criado pelo comando ShellScript **tsc --init**:

```
{
  "compilerOptions" :
  {
    "target": "es6",
    "module": "commonjs",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "outDir": "out",
    "sourceMap": true //Permite compilar e depurar
com vscode
  }
}
```

## 2. Opções:

1. A propriedade **"target": "es6"** informa ao **tsc** a sintaxe javascript a ser gerada. Obs: Se sua aplicação depende de browser mais antigo informe a sintaxe apropriada.
2. A propriedade **"module": "commonjs"** define o **sistema de módulo** para o programa. Consulte a página de referência dos módulos para obter mais informações. Muito provavelmente você deseja "CommonJS" para projetos de nó.
  1. Os módulos podem ser:
    1. [ "CommonJS", "AMD", "System", "UMD", "ES6", "ES2015", "ES2020", "ESNext", "None" ]
3. A propriedade **"strict": true** ative todas as opções de verificação de tipo restrita.
4. A propriedade **"esModuleInterop": true** permite a interoperabilidade de emissão entre o módulo **CommonJS** e o módulos **ES** por meio da criação de objetos de **namespace** para todas as importações. Implica **'allowSyntheticDefaultImports'**.
5. A propriedade **"skipLibCheck": true**, ignora a verificação de tipo de arquivos de declaração.

6. A propriedade **"forceConsistentCasingInFileNames": true** proíbi referências com capitalização inconsistente para o mesmo arquivo.
7. A propriedade **"outDir": "out"** informa a pasta de saída dos arquivos .js.
8. A propriedade **"sourceMap": true** permite a geração de arquivos **sourcemap**. Esses arquivos permitem que os depuradores e outras ferramentas exibam o **código-fonte TypeScript original** ao trabalhar realmente com os arquivos JavaScript emitidos.
  1. Arquivos de mapa de origem são emitidos como .js.map(ou .jsx.map) arquivos próximos ao .js arquivo de saída correspondente.
  2. Agora, ao criar um .ts arquivo como parte do projeto, ofereceremos experiências de edição e validação de sintaxe ricas
  3. **ATENÇÃO: Após incluir essa propriedade no arquivo tsconfig.json o Vscode passa a compilar e depurar um arquivo com extensão .ts.**
3. Por padrão, o TypeScript inclui todos os **\*.ts** arquivos na pasta e subpastas atuais se as propriedades **outDir**, **outfile**, **files**, **extends**, **include**, **references** e **exclude**, não forem incluídas no arquivo **tsconfig.json**.
4. A forma mais prática para criar o arquivo **tsconfig.json** é executar o código abaixo:

1. Código ShellScript

```
tsc --init
```

5. Agora, para construir a partir do terminal, você pode simplesmente digitar **tsc** e o compilador TypeScript saberá olhar **tsconfig.json** para as configurações do seu projeto e opções do compilador e irá compilar o programa informado na propriedade **main** do arquivo **package.js**.

## 2. Compilador versus serviço de linguagem:

1. É importante ter em mente que o *serviço de linguagem TypeScript* do VS Code é separado do *compilador TypeScript instalado*. Você pode ver a versão TypeScript do VS Code na barra de status ao abrir um arquivo TypeScript.
2. Caso a versão do serviço de linguagem do vscode seja diferente da versão do comando "tsc --version" é necessário corrigir a diferença.
  1. O link **"Usando versões mais recentes do TypeScript"** encina como igualar as versões.

## 3. Como configurar compilador tsc (typescript) para executar e depurar com vscode?.

1. Configuração do arquivo tsconfig.json:

1. Para permitir que o vscode (Visual Studio Code) compile e depure um arquivo **.ts** é necessário adicionar as propriedades **"outDir"** e **"sourceMap"** em **"compilerOptions"** do arquivo **tsconfig.json** conforme o código abaixo:

1. Código json

```
{
  "compilerOptions" :
  {
    "outDir": "out", //Código compilado fica
    nesta pasta.
    "sourceMap": true //Permite compilar e
    depurar com vscode
  }
}
```

2. **Configuração do arquivo tasks.json**

1. O vscode pode ser configurado para que ferramentas externas possam ser executadas dentro do vscode e seus resultados são analisados pelo vscode.
2. As tarefas específicas da área de trabalho ou pasta são configuradas a partir do arquivo **tasks.json** na pasta **.vscode** para uma área de trabalho.
3. As extensões também podem contribuir com tarefas usando um [Provedor de Tarefas](#), e essas tarefas contribuídas podem adicionar configurações específicas do espaço de trabalho definidas no arquivo **tasks.json**.
4. Enquanto que o **tasks.json** permite que o usuário defina manualmente uma tarefa para uma pasta ou área de trabalho específica o [Provedor de Tarefas](#) pode detectar detalhes sobre uma área de trabalho e então criar automaticamente uma Tarefa de Código VS correspondente.

5. **Observação:**

1. o suporte a tarefas está disponível apenas ao trabalhar em uma pasta do espaço de trabalho. Não está disponível ao editar arquivos individuais.
6. As tarefas podem ser executadas a partir da opção **/terminal/Run Task** ou **/terminal/Run Build Task**. Obs: Tecla de atalho **Ctrl + shift + B**.
  1. Exemplo de como compilar um projeto typescript:
    1. Selecione o arquivo principal do projeto (ex: index.ts);
    2. Selecione a opção **/terminal/Run Task**
    3. Selecione a opção **build tsc -pasta/config.json** se o arquivo **tasks.json** não tiver configurado para executar um programa em particular.
  2. Exemplo de como executar o programa gerado pela opção anterior:
    1. Selecione a opção **/Run/Start Debugging** ou pressione na tecla **F9**.

3. **Transpile TypeScript para JavaScript** usando vscode.

1. O VS Code se integra com o tsc nosso executor de tarefas integrado. Podemos usar isso para transpilar **.ts** arquivos em **.js** arquivos. Outro benefício de usar as tarefas do VS Code é que você obtém a detecção integrada de erros e avisos



exibida no painel Problemas . Vamos transpilar um programa simples do TypeScript Hello World.

### 1. Etapa 1: Criar um arquivo TS simples

1. Abra o VS Code em uma pasta vazia e crie um helloworld.ts arquivo, coloque o seguinte código nesse arquivo:

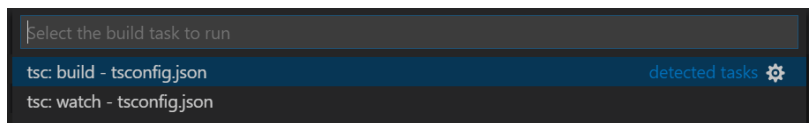
```
let message: string = 'Hello World';  
console.log(message);
```

2. Para testar se você tem o compilador TypeScript tsc instalado corretamente e um programa Hello World em funcionamento, abra um terminal e digite tsc helloworld.ts. Você pode usar o Terminal Integrado ( Ctrl + ` ) diretamente no VS Code.
3. Agora você deve ver o helloworld.js arquivo JavaScript transpilado , que pode ser executado se o Node.js estiver instalado, digitando:

```
node helloworld.js
```

### 2. Etapa 2: Execute o build # do TypeScript

1. Execute Executar Tarefa de Compilação ( **Ctrl + Shift + B** ) no menu Terminal global . Se você criou um tsconfig.json arquivo na seção anterior, ele deve apresentar o seguinte seletor:



2. Selecione a entrada **tsc: build** . Isso produzirá um arquivo HelloWorld.js HelloWorld.js.map na área de trabalho.
3. Se você selecionou **tsc: watch**, o compilador TypeScript observará as alterações em seus arquivos TypeScript e executará o transpilador em cada alteração.
4. Nos bastidores, executamos o compilador TypeScript como uma tarefa. O comando que usamos é:tsc -p .

### 3. Etapa 3: Tornar o TypeScript Build o padrão:

1. Você também pode definir a tarefa de construção do TypeScript como a tarefa de construção padrão para que seja executada diretamente ao acionar Executar Tarefa de Construção ( Ctrl + Shift + B ).
2. Para fazer isso, selecione **Configure Default Build Task (Ctrl + Shift + P)** no menu global Terminal . Isso mostra um seletor com as tarefas de construção disponíveis. Selecione TypeScript tsc : build , que gera o seguinte tasks.json arquivo em uma .vscode pasta:

```
{
  // See https://go.microsoft.com/fwlink/?
  LinkId=733558
  // for the documentation about the
  tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "type": "typescript",
      "tsconfig": "tsconfig.json",
      "problemMatcher": [
        "$tsc"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ]
}
```

3. Observe que a tarefa tem um **objeto JSON "group"** que define a tarefa **"kind": "build"** e torna o padrão na propriedade **"isDefault": true**. Agora, ao selecionar o comando Executar Tarefa de Compilação ou pressionar ( Ctrl + Shift + B ), você não será solicitado a selecionar uma tarefa e sua compilação será iniciada.

#### 4. ATENÇÃO:

1. Toda vez que alterar o arquivo .ts é necessário pressionar as teclas ( Ctrl + Shift + B ) para compilar o arquivo .ts e transformar e .js.
2. Após compilar o arquivo as opções de execução e debug do vscode podem ser usadas sem problema.
3. O [Node.js tutorial in Visual Studio Code](#) ensina como debugar os programas.

5. Mais detalhes pode ser visto em:

<https://code.visualstudio.com/docs/typescript/typescript-compiling>.

#### 4. Teclas de atalho do vscode

1. ...

2. ...

4. 

#### 4. Exemplos.

1. Exemplo 01 - Usa o terminal de comandos para compilar.

1. Crie um arquivo de nome helloworld.ts e adicione o seguinte código TypeScript:

1. Código typescript:

```
let message: string = 'Hello World';  
console.log(message);
```

2. Olhe o código TypeScript acima e observe a palavra **let** e **string** são da linguagem TypeScript e não pertence ao javascript.
3. Para compilar o arquivo **helloworld.ts** selecione o **terminal integrado (Ctrl + `)** do vscode e execute o código abaixo:

1. Código ShellScript:

```
//Obs: tsc é o compilador typescript  
tsc helloworld.ts
```

1. O comando acima irá compilar e gerar o arquivo **helloworld.js**.
2. Para executar o arquivo JavaScript **helloworld.js**, selecione o **terminal integrado (Ctrl + `)** do vscode e execute o código abaixo:

1. Código ShellScript:

```
// node é o interpretador javascript  
node helloworld.js
```

2. Se você abrir **helloworld.js**, verá que não é muito diferente de **helloworld.ts**. A informação de tipo foi removida e **let** agora é **var**..

1. Código javascript:

```
var message = 'Hello World';  
console.log(message);
```

4. .

2. ...

3. 


## 5. REFERÊNCIAS

1. [Tutorial do TypeScript no Visual Studio Code](#)
2. [Compilando TypeScript](#)
3. [ESLint](#)
4. [Configurando ESLint](#)
5. [Familiarize-se com as opções de linha de comando.](#)
6. [Integração do ESLint com editores de textos.](#)
7. [Como usar ESLint com TypeScript](#)
8. A Microsoft criou o padrão de documentação [TSDoc](#) - para a linguagem typescript com objetivo de que vários aplicativos possam gerar documentos do código em que a tag de um atrapalhe a tag de outro. É parecido com jsDoc.
9. O projeto [api-extractor.com](#) é um gerador de site typescript que espera a sintaxe **TSDoc** nos comentários do projeto. Reconhece também a linguagem markdown.
10. O projeto [jsdoc.app](#) é um gerador de documentos baseados nos comentários do código javascript e usa sintaxe **jsdoc** na qual o tsdoc foi estendido. O vídeo [jsdoc](#) descreve com muita clareza o uso do jsdoc.
11. O projeto [docstrap.github.io](#) gera site baseado nos comentários dos arquivos javascript e espera a sintaxe do jsdoc. Para instalar siga os passos descritos [aqui](#).
12. [Como usar os módulos do Node.js com o npm e o package.json](#)
13. [Use TypeScript para construir uma API de nó com Express](#)
14. [Node.js and TypeScript Tutorial: Build a CRUD API](#)

15. 

## 6. HISTÓRICO

### 1. 19/02/2021

- ☒ Criar este documento baseado no modelo02.md ;
- ☒ Escrever tópico Objetivos;
- ☒ Escrever tópico Pre-requisitos
- ☒ Escrever tópico Benefícios
- ☒ Escrever tópico Instalar o compilador TypeScript
- ☒ Configurar o compilador TypeScript
  - ☒ Arquivo tsconfig.json
  - ☒ Compilador versus serviço de linguagem
  - ☒ Como configurar compilador tsc (typescript) para executar e depurar com vscode?
- 

### 2. 22/02/2021

- ☒ Instalar extensões vscode.
- ☒ Escrever tópico Referências
- ☒ Atualizar o histórico deste documento.
- ☒ Adiciona no index as tags:
  - ▶
- ☒ Adicionar a extensões Gramática TypeScript e Javascript mais recente.
- ☒ TypeScript Importer

### 3. 04/03/2021

- ☒ No tópico instalar com npm:
  - ☒ Documentar instalação do pacote npx.

### 4. 27/04/2021

- ☒ Atualizar o index adicionando todos os programas instalados por este documento:

### 5. 28/04/2021

- ☐ Ler no dia seguinte este documento para checar os erros de português.
- 