

# Node.js com TypeScript



Douglas Matoso  
📅 Atualizado em 03/05/2020  
🕒 Leitura: 3 min.

🌐 [Read in English](#)

Fala, pessoal! Neste post vou mostrar como você pode desenvolver no [Node.js](#) usando [TypeScript](#) e ter os [benefícios desta linguagem](#) também no backend.

## ts-node-dev

Para fazer a transpilação do código **TypeScript** para **JavaScript**, vamos usar o [ts-node-dev](#).

Ele fornece um executável que vamos usar no lugar do **Node** para rodar o código TypeScript. Por baixo dos panos ele vai transformar o TypeScript em JavaScript e executar usando o próprio Node.

Outra coisa legal do ts-node-dev é que ele substitui o [nodemon](#) também, então sempre que alterarmos algum arquivo .ts ele vai pegar a alteração e recarregar a aplicação.

## Setup

Vamos precisar do **ts-node-dev** e do próprio **typescript** como dependências de desenvolvimento:

```
npm i -D ts-node-dev typescript
```

Também vamos precisar de um arquivo de configurações para o compilador do TypeScript. Podemos gerar uma configuração padrão com o comando:

```
npx tsc --init
```

Ele vai criar um `tsconfig.json` na raiz da aplicação. Você pode customizar as opções, mas o default já vai servir pra gente.

## Escrevendo o código

Com este setup já podemos escrever o código da aplicação usando arquivos .ts e todas as funcionalidades do TypeScript.

Como exemplo, vamos criar um "Hello World" com [Express.js](#).

Vamos precisar instalar o próprio express, e também as definições de tipos para ele. Note que o express é dependência da aplicação, enquanto as definições de tipos são dependências de desenvolvimento.

```
npm i express
```

```
npm i -D @types/express
```

Algumas bibliotecas já fornecem as próprias definições de tipo, então esta segunda instalação não é necessária. Você vai descobrir se uma lib precisa quando tentar importá-la no código e o editor reclamar que estão faltando as definições de tipo.

Para o nosso "Hello, World" vou escrever este código em um arquivo

`index.ts`:

```
import express, { json } from 'express'

const app = express()
app.use(json())

app.get('/', (request, response) => {
  return response.json({ message: 'Hello, TypeScript!' })
})

app.listen(3000, () => {
  console.log('🔥 Server started on http://localhost:3000')
})
```

Veja que podemos usar o padrão [ES Modules](#) (import ... from ...).

## Executando a aplicação

Para executar a aplicação, vamos adicionar no script de `start`, do

`package.json`:

```
"start": "ts-node-dev --transpile-only index.ts"
```

A opção `--transpile-only` vai dizer para ele apenas transpilar, sem fazer verificação de tipos. Esta verificação pode ser feita direto no editor, de forma muito mais ágil, apontando exatamente no seu código onde estão os erros.

## Em produção

Em produção, como queremos o máximo de performance, vamos usar diretamente o Node com o código transpilado em JavaScript. Para isso podemos adicionar um script de build:

```
"build": "tsc"
```

Ele vai usar o compilador do TypeScript para transpilar todo o código `.ts` e salvar os `.js` correspondentes no disco.

## Conclusão

O código final deste exemplo está aqui: <https://github.com/doug2k1/node-typescript>

O TypeScript tem muitas vantagens, e a cada dia a integração com outras ferramentas está cada vez madura. Está cada vez mais fácil adotar, seja no frontend ou backend, e é uma opção cada vez mais interessante para a maioria dos projetos.

Tags: [nodejs](#) | [typescript](#)


Compartilhe!



#### Comentários

Comentários desabilitados



 English Português

[Contato](#) [Política de Privacidade](#)



© 2022 - Feito com  e  por Douglas Matoso