



APLICAÇÕES INTERATIVAS

APRESENTAÇÃO

Quem sou eu?

- Prof. **Fábio Abenza**
- **12 anos** atuando na área de TI
- Formado em **Sistemas de Informação** pela **Universidade de Mogi das Cruzes (UMC)** e **Desenvolvimento de Jogos no Senac**

Quem sou eu?

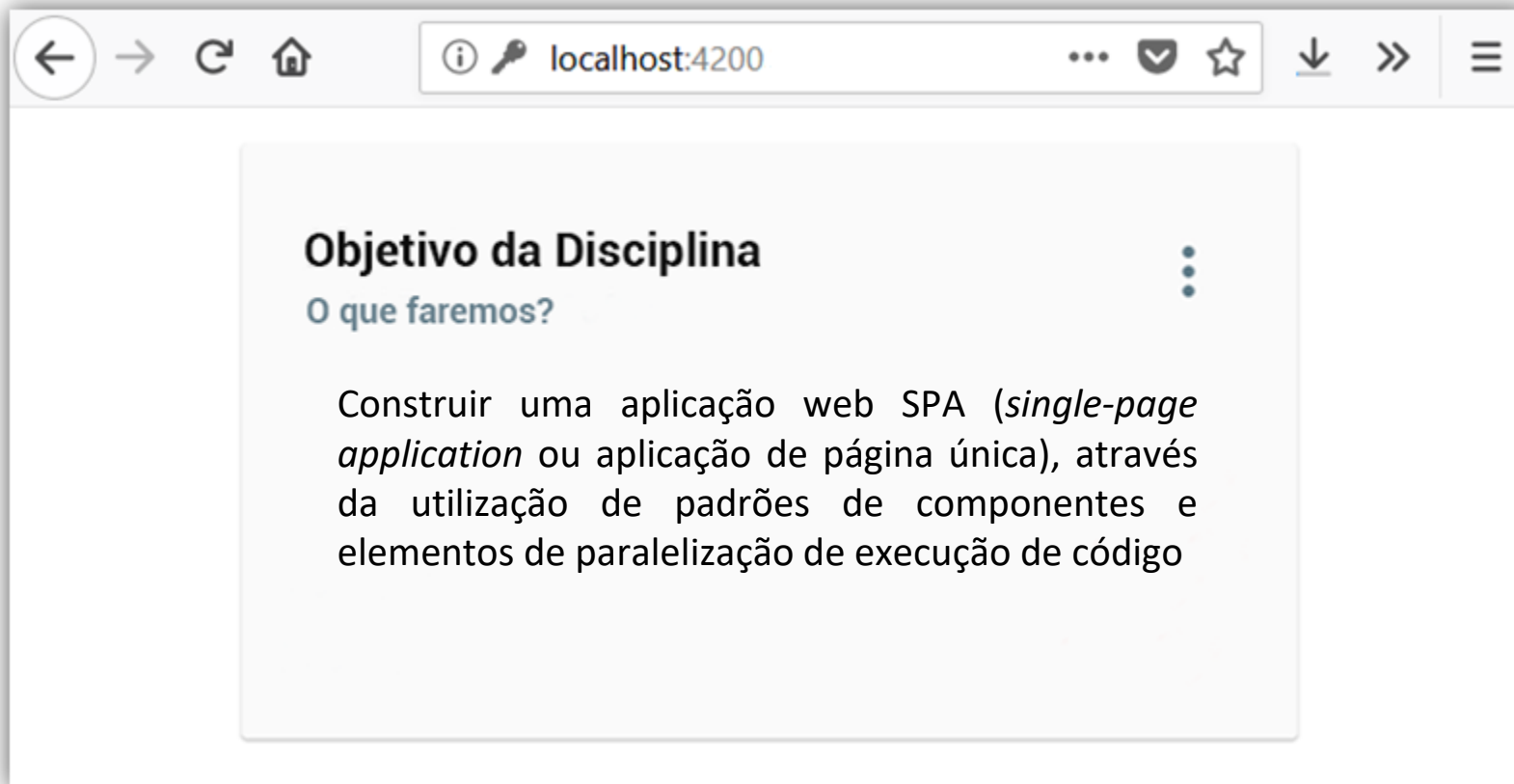
- 8 anos na **Capgemini** em consultoria para o **Bradesco**
- **Java EE, Spring, JavaScript, Node.js e Angular**
- Faço parte do **Centro de Inovação e Tecnologia (CIT)** do Senac

Como falar comigo?



✉ fabio.hafreire@sp.senac.br ✓

Objetivo da disciplina



Como?

- Entendendo o conceito de aplicações SPA
- Conhecendo o padrão de construção de código e páginas baseado em componentes
- Entendendo o uso de frameworks JS para criar front-ends com foco em uma melhor UX
- Criando aplicações responsivas

Por quê?

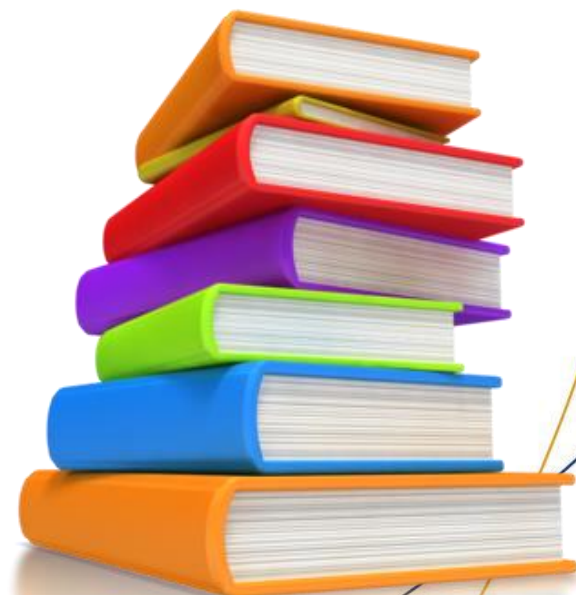
- Pois aplicações SPA estão se tornando o padrão
- Pois frameworks Javascript estão dominando a construção de aplicações web a passos largos
- Para entender os agentes atuantes em aplicações SPA (back e front end)
- Para criar aplicações prontas para mobile

Como será feita a avaliação?

- **Teste I (30%)**
- **Teste II (30%)**
- **Trabalho da Disciplina (40%)**

Bibliografia

- <http://www.typescriptlang.org/docs> (*documentação oficial do TypeScript, em inglês*)
- angular.io/docs (*documentação oficial do Angular, em inglês*)



APLICAÇÕES INTERATIVAS

O que são aplicações interativas?

- A world wide web nasceu com o conceito de interatividade pequeno ou inexistente (apenas para consulta de artigos e documentos)
- O uso de links permitia alguma interatividade, ainda que dependesse exclusivamente do servidor

Formulários

- Itens de formulário (caixas de entrada, botões, etc) aumentaram a capacidade de interação, permitindo o envio de mais dados ao servidor
- O servidor possuía limitações de tratamento de dados, já que não existia o conceito de páginas dinâmicas, apenas estáticas

Páginas Estáticas x Dinâmicas

- Páginas estáticas representam páginas que possuem conteúdo completamente imutável no servidor. Ou seja, o conteúdo enviado ao cliente é sempre o mesmo
- No início da web, eram o único tipo existente

Páginas Estáticas x Dinâmicas

- Com o tempo, surgiu a necessidade do servidor devolver conteúdo de acordo com as entradas do usuário
- Páginas dinâmicas foram criadas, sendo capazes de sofrer alterações no servidor e enviadas para o cliente de forma customizada

Páginas Estáticas

- Páginas estáticas representam páginas HTML imutáveis
- Normalmente, servidores HTTP simples (como o Apache HTTP server ou o Microsoft IIS) são utilizados para devolução de páginas deste tipo

Páginas Dinâmicas

- Páginas dinâmicas do servidor representam arquivos montados dinamicamente no servidor, dependendo de informações enviadas pelo cliente e/ou obtidas de bases de dados

Páginas Dinâmicas

- Servidores de aplicação especiais (como Tomcat em Java) são utilizados para gerenciamento de páginas dinâmicas
- É possível acoplar extensões em servidores de aplicação estáticos (como PHP no Apache) para permitir a devolução de conteúdos dinâmicos

Limitações do Conteúdo Dinâmico

- A capacidade de devolução de páginas dinâmicas permitiu conteúdo muito mais interativo em páginas
- No entanto, as limitações de usabilidade e velocidade de carregamento ainda limitavam a interatividade

Limitações do Conteúdo Dinâmico

- Com o uso de conteúdo dinâmico, é necessário recarregar toda a página
- Além disso, o carregamento normalmente não é instantâneo e não permite interações muito avançadas entre elementos

Interatividade no Cliente

- JavaScript surgiu diante de algumas destas limitações e necessidades, visando permitir interação e dinamismo no cliente
- JavaScript é uma linguagem de programação de alto nível, interpretada, de tipagem fraca, orientada a objetos e multiparadigma

O que é JavaScript?

- Paralelamente a HTML e CSS, JavaScript é uma das três principais tecnologias da produção de conteúdo para internet. É utilizada para tornar páginas dinâmicas e fornecer software interativos

Criação do JavaScript

- JavaScript foi criada em 1995 pela Netscape para o Netscape Communicator (navegador)
- Atualmente, a grande maioria dos sites utilizam JavaScript

Motores JavaScript

- Em navegadores web modernos (como Chrome, Firefox e Edge), não há a necessidade de instalação de plug-ins para uso de JavaScript, já que estes possuem mecanismos de interpretação de JavaScript incorporados (“motor” JavaScript)

Compatibilidade de JavaScript

- Cada motor JavaScript representa uma implementação diferente da linguagem, todas baseadas na especificação ECMAScript (ECMA - *European Computer Manufacturers Association*)

Compatibilidade de JavaScript

- Embora alguns mecanismos não suportem a especificação completamente, muitos utilizam recursos adicionais além do padrão ECMA





LAWFUL GOOD

*On a mission to make the Internet better.
For us, for you, for everyone.*



NEUTRAL GOOD

KHTML was always better.



CHAOTIC GOOD

Oh, snap!



LAWFUL NEUTRAL

*Cool add-on.
Watch as I do the exact same thing with my Settings menu.*



TRUE NEUTRAL

We're just here to have fun.



CHAOTIC NEUTRAL

*We didn't like your puny attempts at browsers.
So we built our own. End of story.*



LAWFUL EVIL

*You whippersnappers and your "CSS." Bah.
Most of you probably don't remember what a newsgroup is.*



NEUTRAL EVIL

Standards compliance?



CHAOTIC EVIL

*Do you want to install and run "YOU have an OUT OF DATE
browser which can cause you get infected with viruses, spam, & spyware.
To prevent this press YES now"*

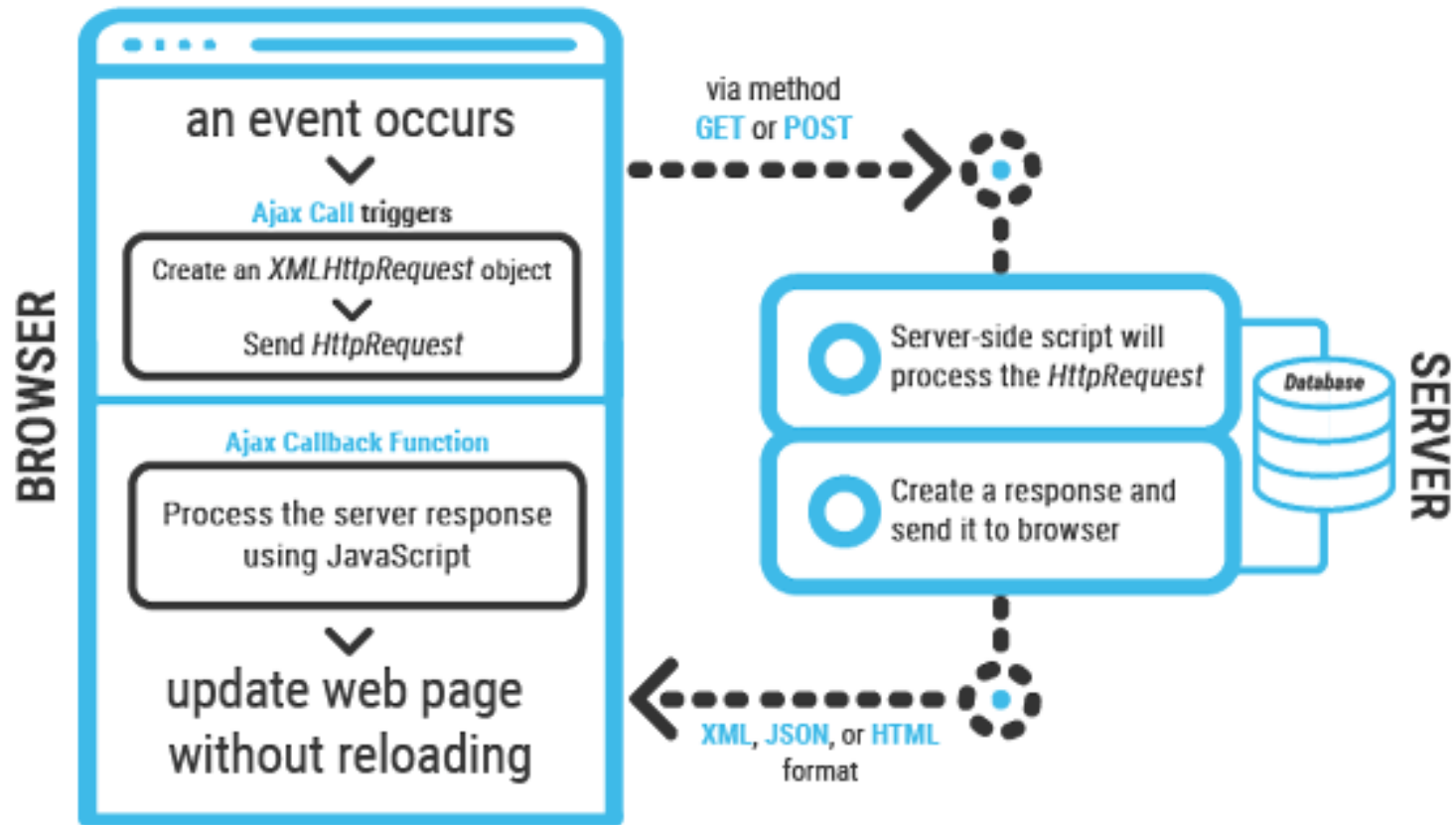
Evolução do JavaScript

- JavaScript evoluiu muito ao longo do tempo, passando de uma linguagem simples de cliente para uma complexa linguagem de manipulação de elementos de front-end e de servidor

Javascript e AJAX

- A evolução do JavaScript e das tecnologias de servidor permitiram atualizar trechos da página com o uso de AJAX (“Asynchronous JavaScript and XML” ou "JavaScript Assíncrono e XML"), aumentando ainda mais o dinamismo e a interatividade

Javascript e AJAX



Páginas Mais Interativas

- Gigantes da tecnologia, como o Google e o Facebook, desenvolveram novas tecnologias a fim de permitir ainda maior interatividade e diminuir a complexidade de uso em relação ao AJAX, utilizadas primariamente em seus principais produtos, como GMail e Facebook

Aplicações de Página Única

- Tais tecnologias deram origem ao conceito que hoje chamamos de Aplicação de Página Única ou *Single-Page Application* (SPA)

Aplicações de Página Única

- Aplicações de página única levam além o conceito de uso de JavaScript e AJAX e fazem com que toda a composição da aplicação seja feita em uma única página
- Para o usuário, há o efeito de que a página nunca é recarregada

Aplicações de Página Única

- Toda estrutura de SPAs é baseada em uso de JavaScript e AJAX. Assim, grande parte do código da aplicação estará no lado do cliente
- O servidor é acionado apenas quando estritamente necessário (carregamento ou salvamento de dados, por exemplo)

Aplicações de Página Única

- A intenção de SPAs é fornecer uma experiência do usuário similar à de um aplicativo desktop, com grande dinamismo, carregamento ágil de elementos, interatividade com componentes visuais avançados e responsividade

Frameworks SPA

- Muitos frameworks JavaScript foram criados a partir destas novas tecnologias
- Cada um deles possui suas peculiaridades e formas de tratamento de conteúdo e eventos, visando facilitar o trabalho do desenvolvedor em aumentar a interatividade da página

Frameworks SPA

- AngularJS
- Angular
- Meteor.js
- React
- Vue.js

Angular

- Na disciplina, estudaremos os principais conceitos de aplicações de página única através do Angular

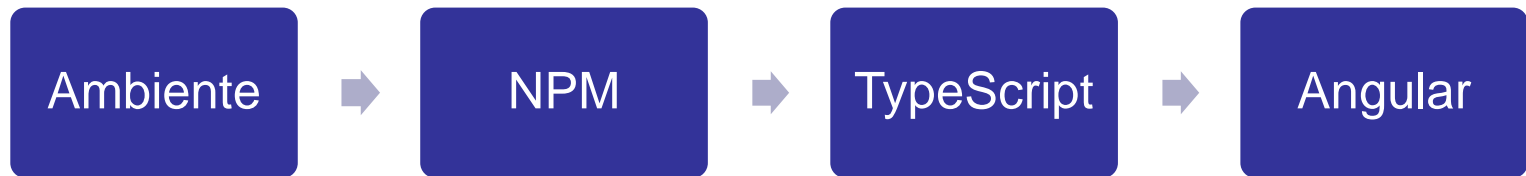
NPM e TypeScript

- Angular é feito com e utiliza TypeScript, uma linguagem de programação baseada em JavaScript, com tipagem forte
- Angular também utiliza fortemente o mecanismo de resolução de dependências do Node.js, o NPM (*Node Package Manager*)

NPM e TypeScript

- Portanto, estudaremos melhor estes conceitos como pré-requisitos a seguir, antes de nos aprofundarmos em Angular

NPM e TypeScript



NODE.JS E RESOLUÇÃO DE DEPENDÊNCIAS

Node.js

- O Node.js é um ambiente de execução de código JavaScript aberto e de multiplataforma, capaz de executar código JavaScript fora de um navegador
- “Servidor JavaScript” (mas muito mais que isso)

Node.js

- Permite que sejam escritas ferramentas de linha de comando e scripts no lado do servidor visando produzir conteúdo dinâmico da página da web (entre outras muitas tarefas)

Node.js

- Representa um paradigma de *JavaScript Everywhere*, permitindo que JavaScript seja utilizado tanto no lado do Cliente quando no lado do Servidor

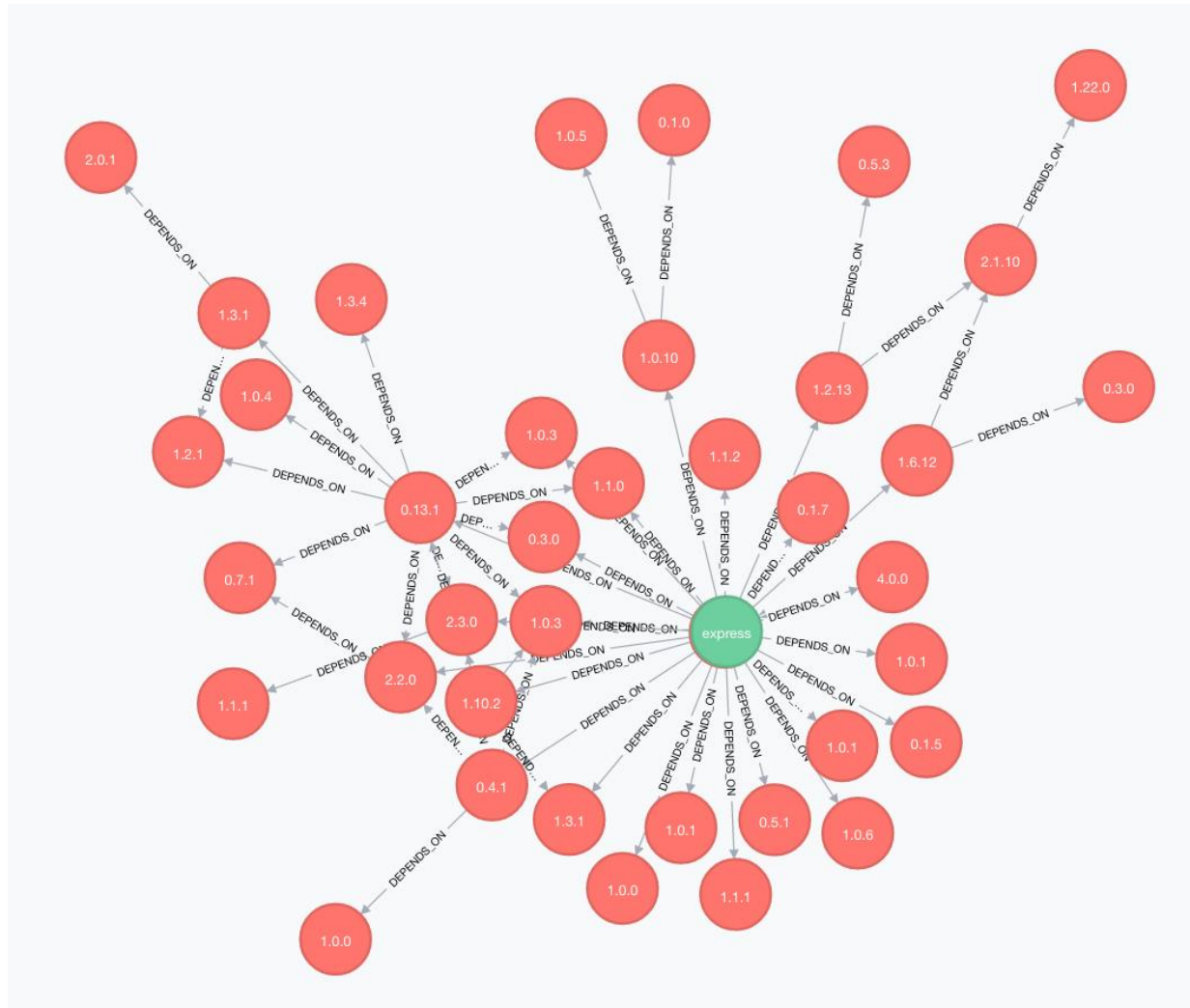
NPM

- Com o surgimento e a utilização de múltiplos frameworks JavaScript, surgiu a necessidade do controle de tais bibliotecas
- O Node.js possui um gerenciador próprio de bibliotecas, amplamente utilizado pela comunidade, o NPM (*Node Package Manager*)

NPM

- Com o NPM, podemos definir quais bibliotecas utilizamos, instalá-las, atualizá-las e, principalmente, ter suas dependências automaticamente resolvidas pela ferramenta

NPM



Node.js e NPM

- O Node.js será utilizado apenas para resolução de dependências e execução de testes pelas ferramentas que utilizaremos
- Não utilizaremos o Node.js diretamente, como servidor de elementos back-end (ainda!)

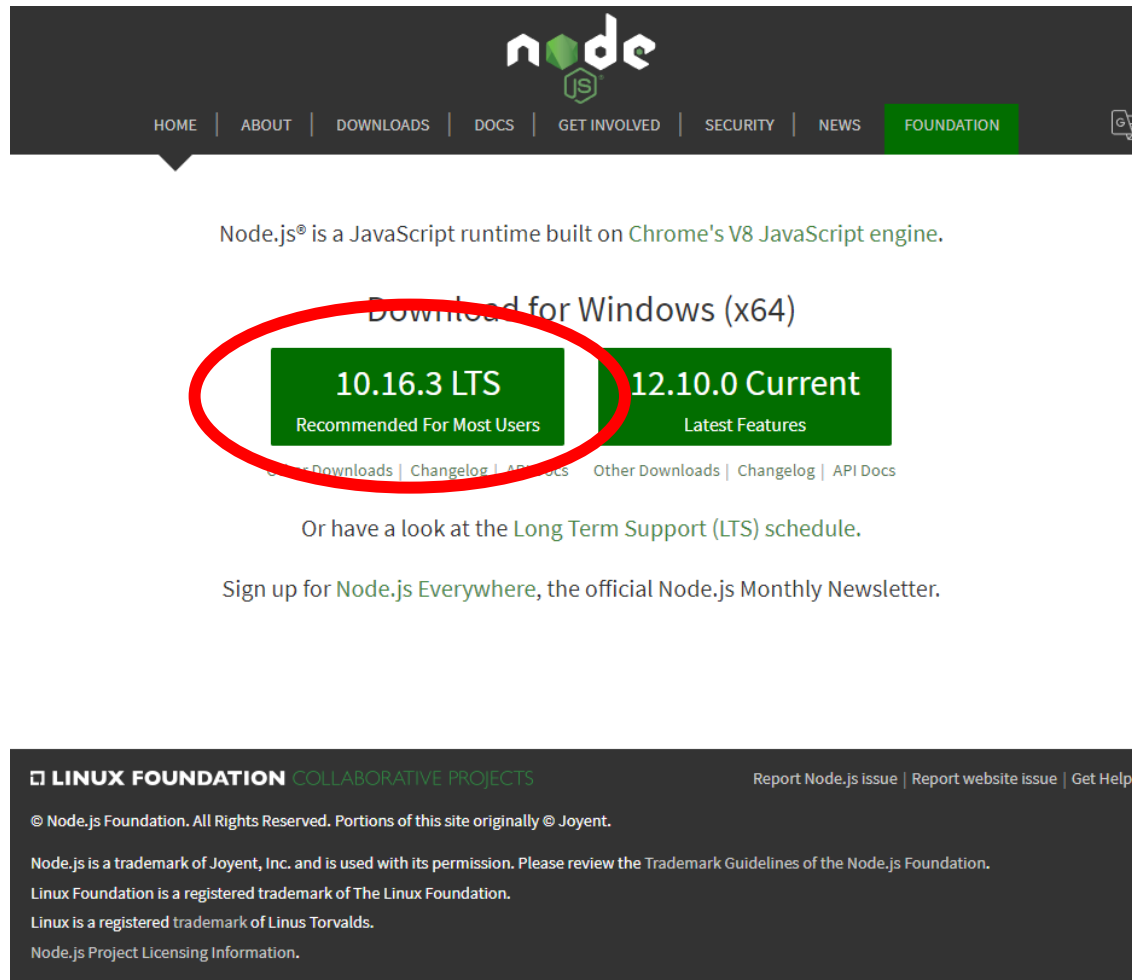
Obtendo as Ferramentas

- Para baixar o Node.js, basta acessar o endereço nodejs.org/en e efetuar o download da versão LTS (long-term support ou com suporte prolongado, a versão mais estável)

Obtendo as Ferramentas

- Em seguida, basta iniciar a instalação e prosseguir com o assistente

Obtendo o Node.js



The screenshot shows the Node.js website's download page. The navigation bar at the top includes links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. The main heading states "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." Below this, the "Download for Windows (x64)" section features two green buttons: "10.16.3 LTS Recommended For Most Users" and "12.10.0 Current Latest Features". The "10.16.3 LTS" button is circled in red. Below the buttons, there are links for "Downloads", "Changelog", "API Docs", "Other Downloads", and "API Docs". A text block suggests looking at the "Long Term Support (LTS) schedule" and signing up for the "Node.js Everywhere" newsletter. The footer contains the "Linux Foundation Collaborative Projects" logo, copyright information, and links to report issues or get help.

node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

10.16.3 LTS
Recommended For Most Users

12.10.0 Current
Latest Features

Downloads | Changelog | API Docs | Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

Sign up for Node.js Everywhere, the official Node.js Monthly Newsletter.

LINUX FOUNDATION COLLABORATIVE PROJECTS

Report Node.js issue | Report website issue | Get Help

© Node.js Foundation. All Rights Reserved. Portions of this site originally © Joyent.

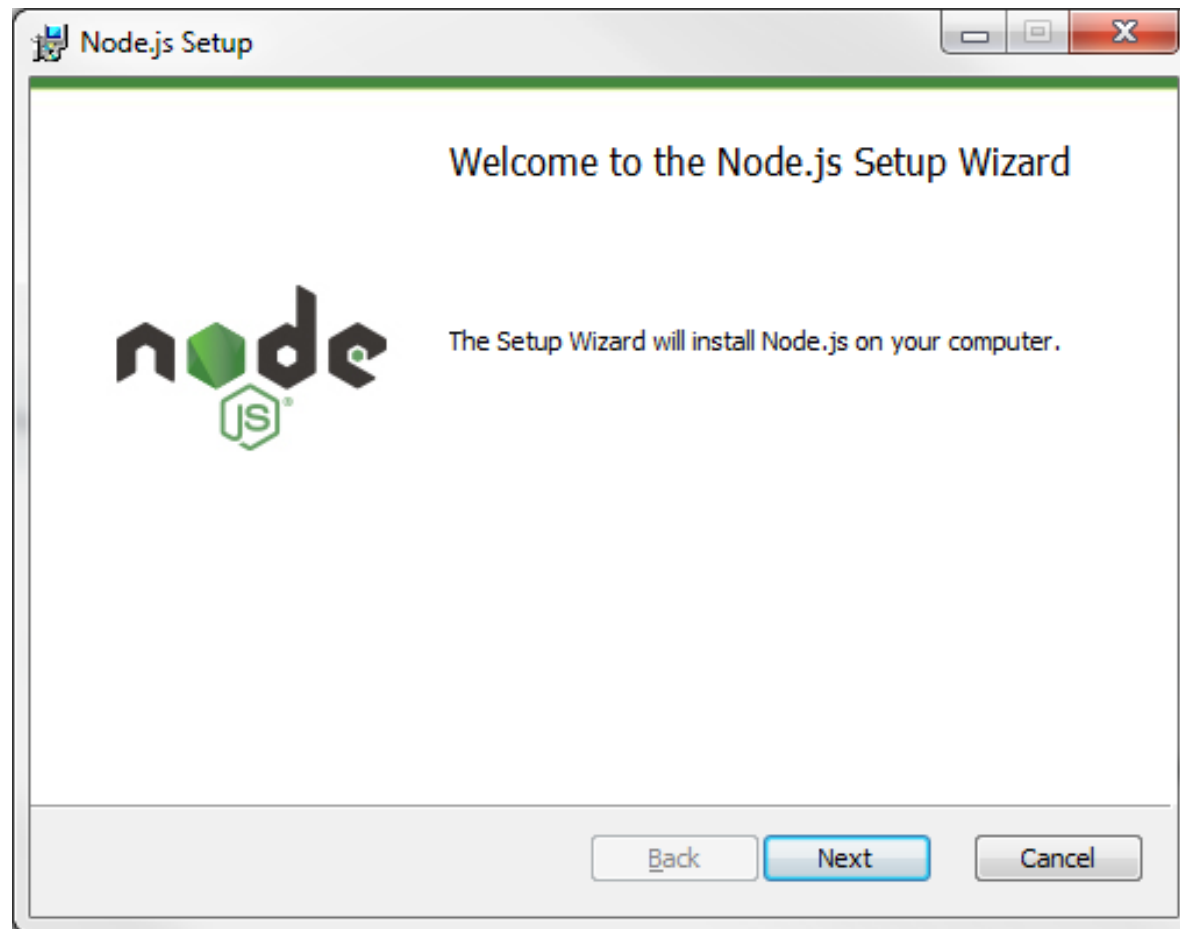
Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark Guidelines of the Node.js Foundation.

Linux Foundation is a registered trademark of The Linux Foundation.

Linux is a registered trademark of Linus Torvalds.

Node.js Project Licensing Information.

Instalação do Node.js



Ferramentas de Desenvolvimento

- Além do Node.js, é interessante utilizar um editor ou IDE que auxilie no gerenciamento de projetos e elementos de código

Ferramentas de Desenvolvimento

- Há muitas ferramentas interessantes, como os editores **Atom**, **Sublime** e **Visual Studio Code**, além de IDEs específicas, como o **JetBrains Webstorm**

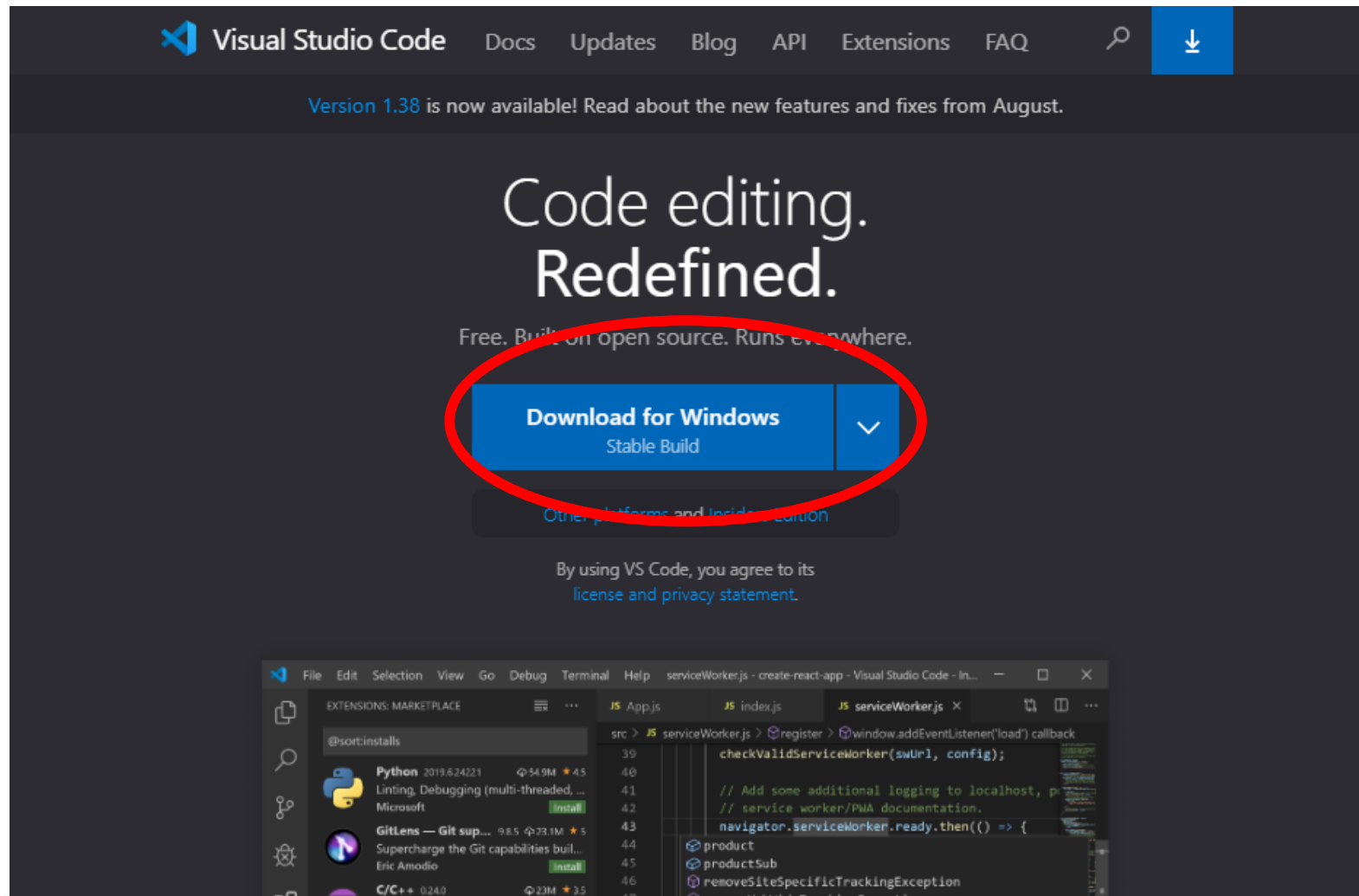
Visual Studio Code

- Na disciplina, utilizaremos o **Visual Studio Code** para gerenciamento de código e desenvolvimento

Visual Studio Code

- Para obter o Visual Studio Code, vá até o endereço code.visualstudio.com e efetue o download
- Em seguida, prossiga com a instalação

Visual Studio Code



TYPESCRIPT

Typescript

- Linguagem de programação open source desenvolvida e mantida pela Microsoft
- Adiciona validações de sintaxe e tipagem mais estritas ao JavaScript tradicional
- “Transmpila” (é convertida) para JavaScript

Typescript

- Surgiu devido as deficiências do JavaScript para o desenvolvimento de aplicativos em grande escala
- Desafios ao lidar com código JavaScript complexos levaram à demanda por ferramentas para facilitar o desenvolvimento

Typescript

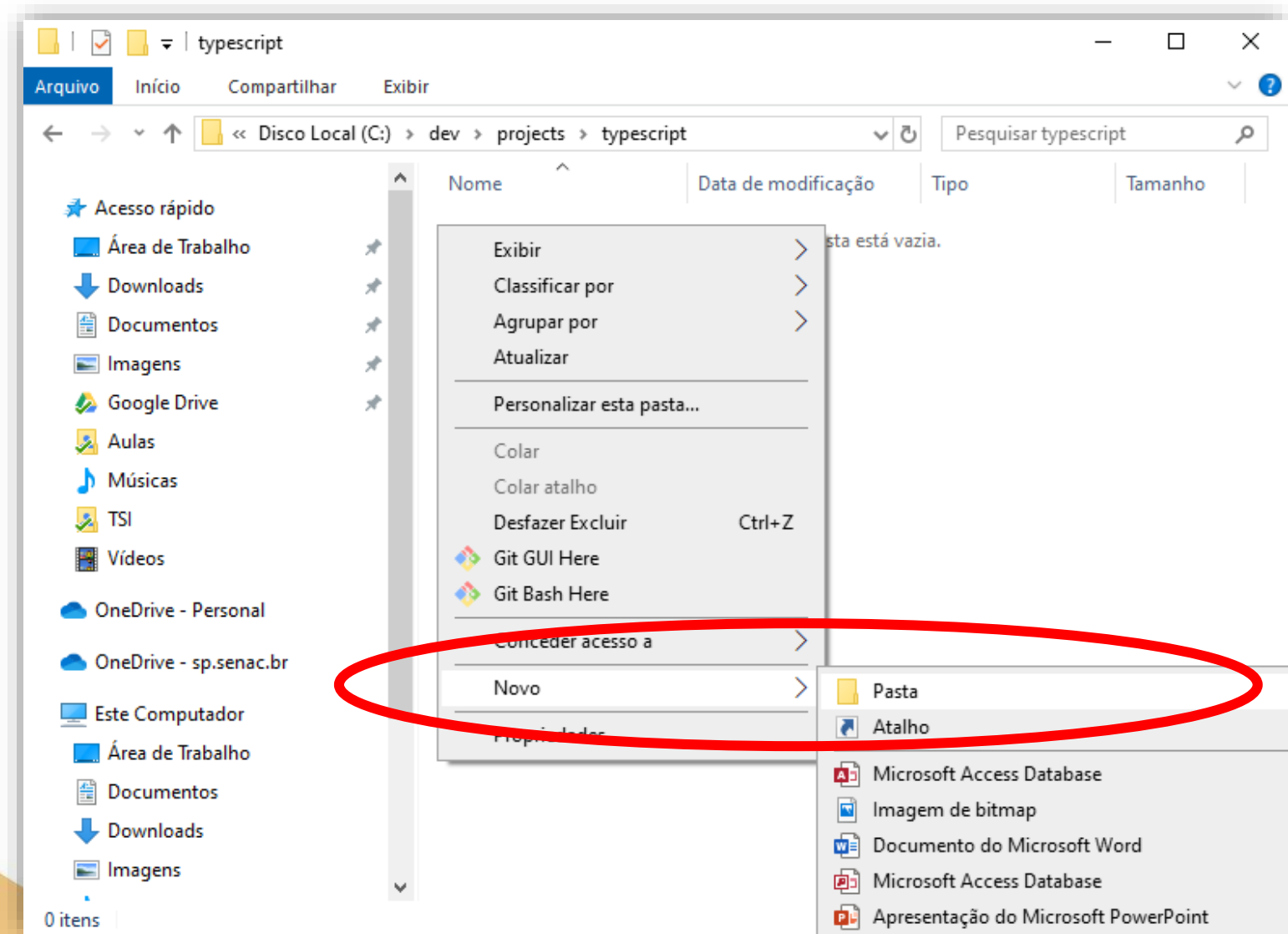
- Buscou-se uma solução que não quebrasse a compatibilidade com o padrão ECMA
- Sabendo que a proposta do ECMA prometia suporte para a programação utilizando classes, escolheu-se basear o TypeScript nessa proposta

TRABALHANDO COM TYPESCRIPT

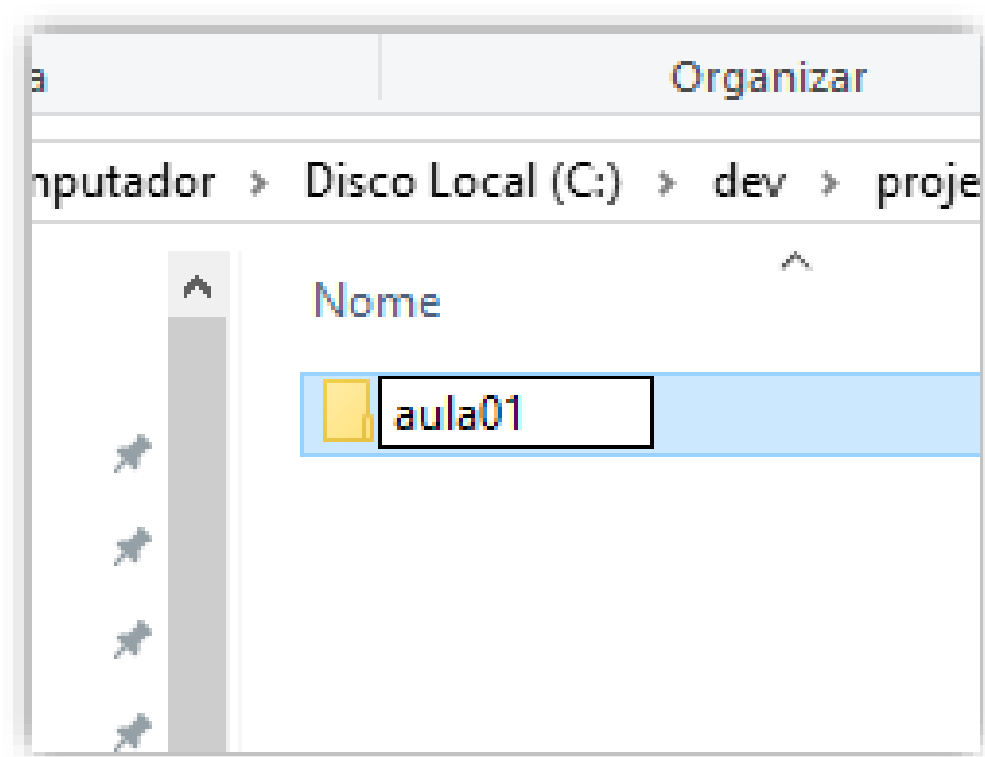
Criando um Projeto TypeScript

- Inicialmente, trabalharemos com TypeScript “puro”, sem uso do Angular
- Para criar um projeto TypeScript, crie uma pasta no explorador de arquivos, vá até o VisualStudio Code e abra-a

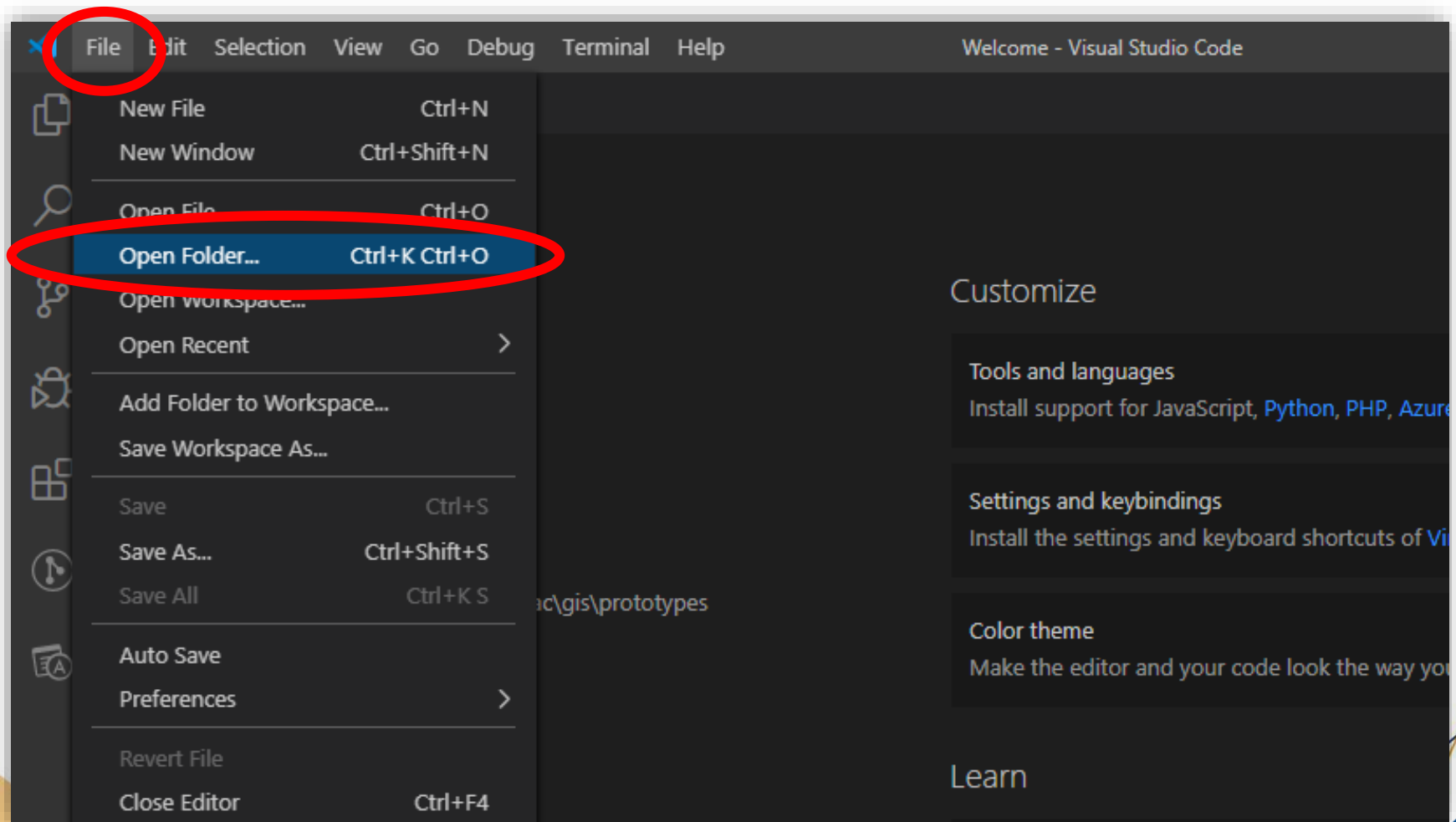
Criando um Projeto TypeScript



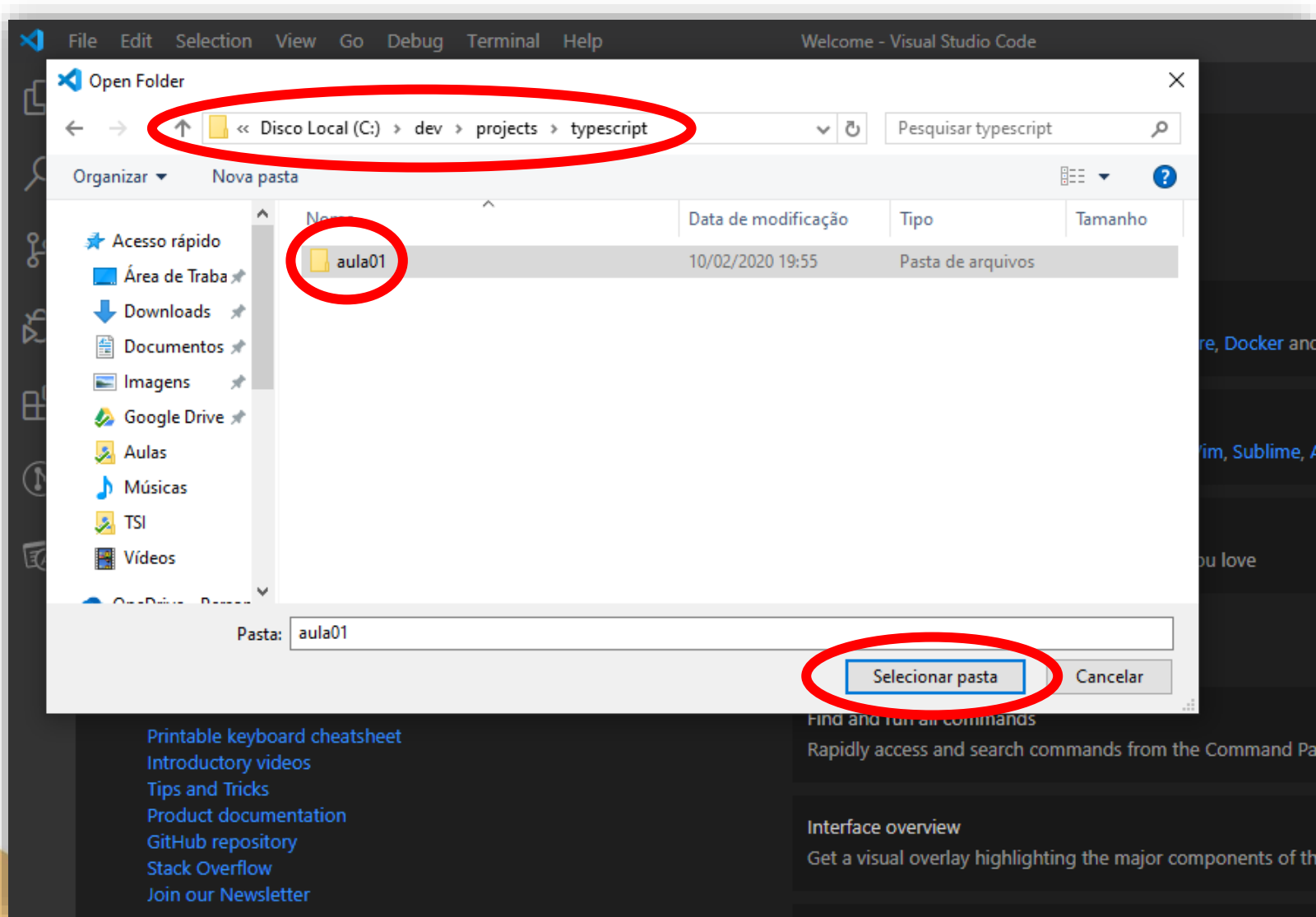
Criando um Projeto TypeScript



Criando um Projeto TypeScript



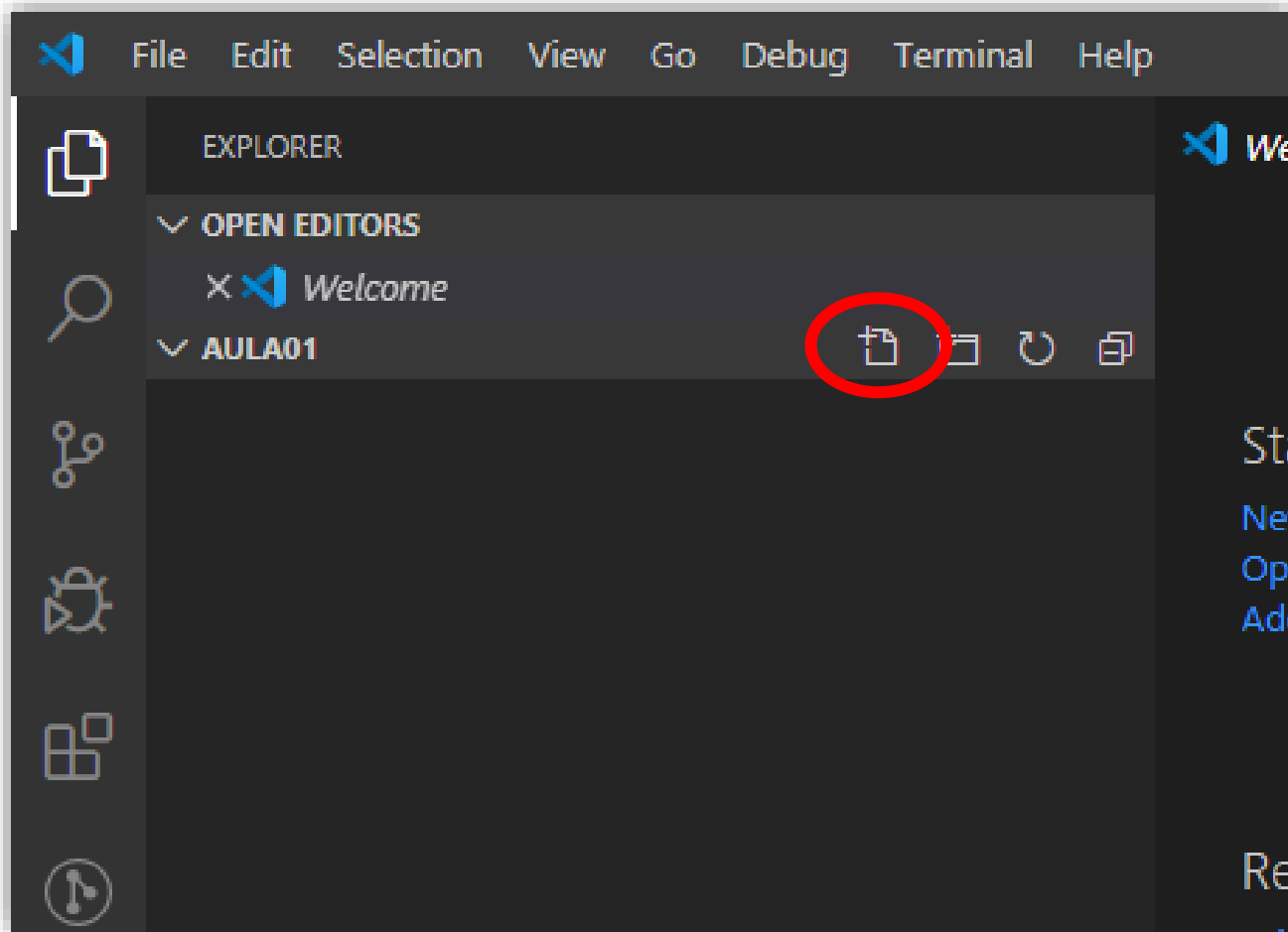
Criando um Projeto TypeScript



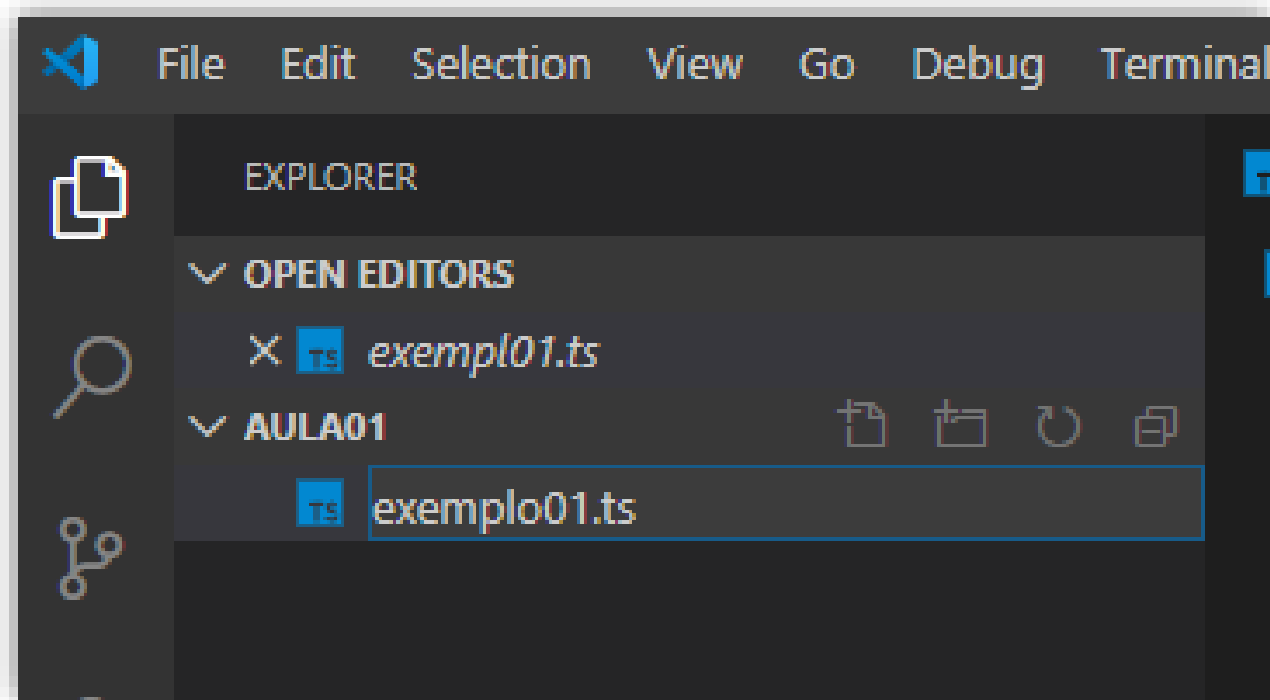
Criando um arquivo .ts

- Crie um novo arquivo TypeScript, clicando no ícone de “+” ao lado do nome do projeto
- Nomeie-o **exemplo01.ts**

Criando um arquivo .ts



Criando um arquivo .ts



Código TypeScript

- Digite o código a seguir na área de edição do arquivo aberta automaticamente pelo Visual Studio Code
- **Estudaremos o código detalhadamente logo a seguir**

Código TypeScript

A screenshot of the Visual Studio Code editor interface. The title bar at the top reads "exemplo01.ts - aula01 - Visual Studio Code". The editor window shows a file named "exemplo01.ts" with two lines of TypeScript code:

```
1 const message: string = 'Olá, mundo!';  
2 console.log(message);
```

The code is color-coded: "const" is blue, "message:" is green, "string" is green, "=" is blue, "'Olá, mundo!'" is orange, ";" is blue, "console.log" is green, and "(message)" is blue. The status bar at the bottom is blue and displays "0 errors, 0 warnings", "Ln 2, Col 22", "Spaces: 4", "UTF-8", "CRLF", "TypeScript", "3.7.5", "Prettier", and icons for a search and a bell. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Extensions, Testing, and Settings.

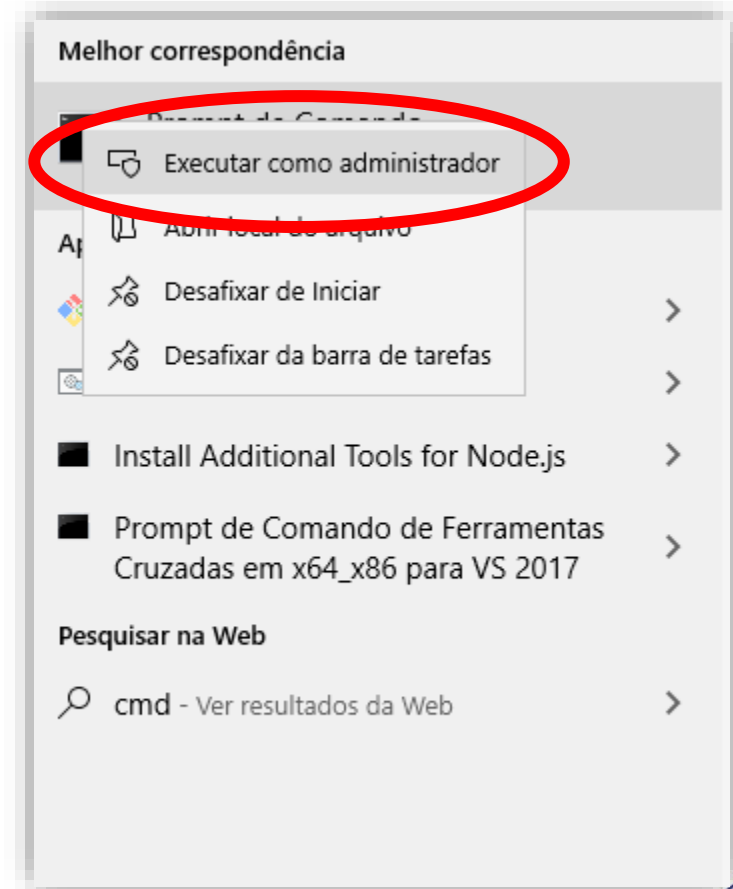
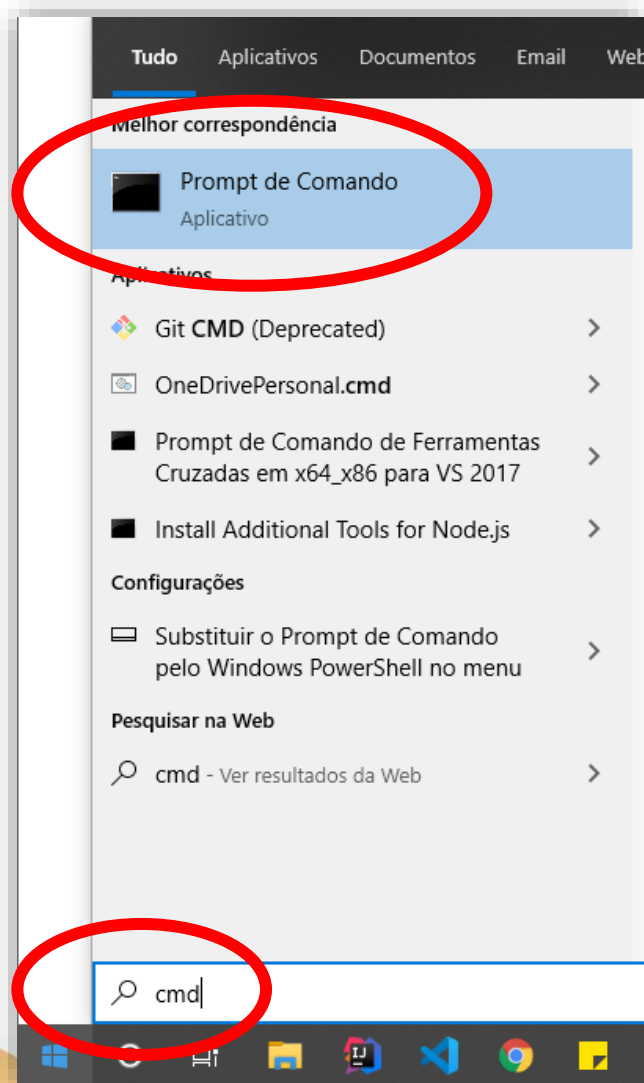
“Compilando” o código

- Os navegadores e o Node.js não são capazes de processar diretamente JavaScript por padrão
- É necessário compilar (ou “transpilar”, ou seja, traduzir) o código de TypeScript para JavaScript
- Para isso, é necessário instalar a biblioteca da linguagem, que contém o compilador

Instalando o TypeScript

- Abra uma janela de linha de comando como administrador (*Iniciar > Pesquisar > cmd > botão direito > Executar como Administrador*)

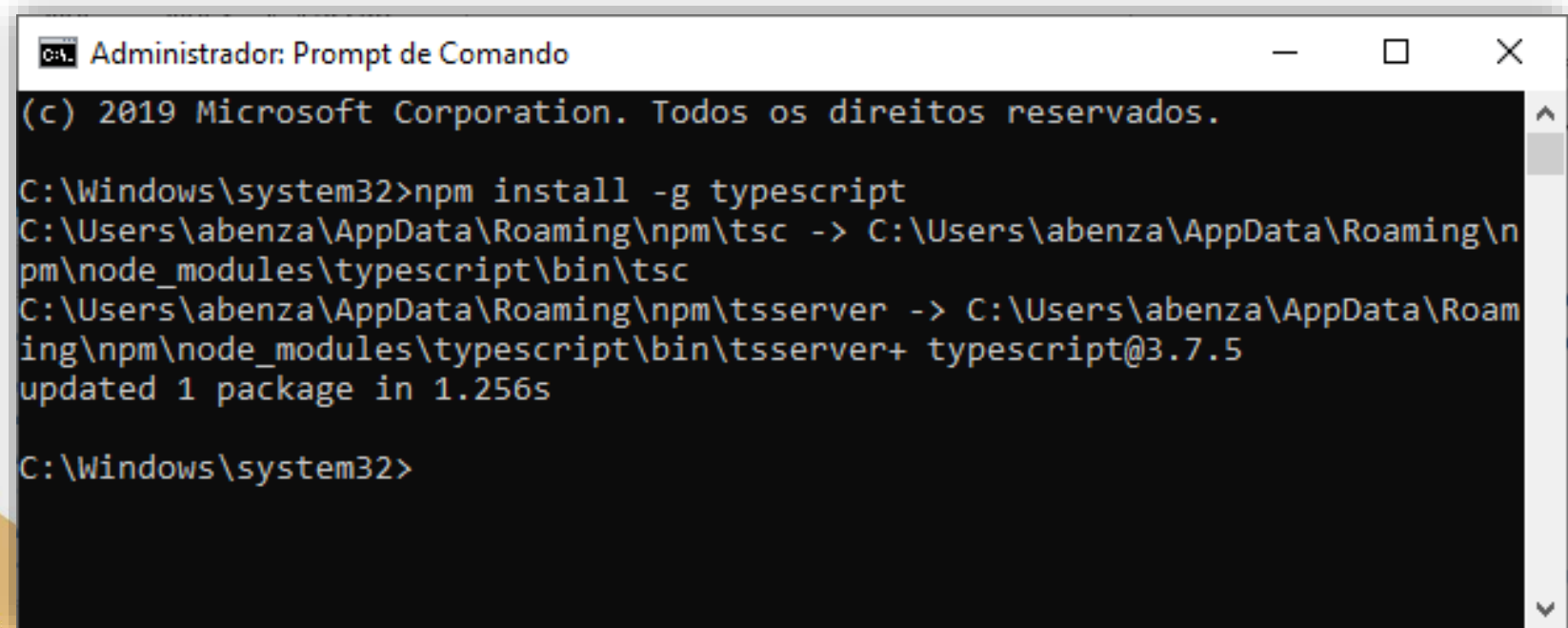
Instalando o Typescript



Instalando o Typescript

- Na janela que surgir, digite:

npm install -g typescript



```
Administrador: Prompt de Comando
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Windows\system32>npm install -g typescript
C:\Users\abenza\AppData\Roaming\npm\tsc -> C:\Users\abenza\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\abenza\AppData\Roaming\npm\tsserver -> C:\Users\abenza\AppData\Roaming\npm\node_modules\typescript\bin\tsserver+ typescript@3.7.5
updated 1 package in 1.256s

C:\Windows\system32>
```

Comandos de instalação

- O comando **npm** permite acionar o gerenciador de pacotes do Node.js, o NPM
- **install** é o subcomando utilizado para instalação de pacotes

Comandos de instalação


- Ao instalar um pacote, todos os elementos JavaScript do pacote, bem como suas dependências serão baixados da internet e ficarão disponíveis para uso no projeto ou globalmente (parâmetro **-g**), para todos os projetos executados na máquina

Comandos de instalação


- Por fim, é indicado o nome do pacote a instalar
- Em nosso caso, **typescript**, a biblioteca que contém os utilitários de compilação e execução de TypeScript

Comandos de instalação


`npm install -g typescript`




Comando para uso
do gerenciador de
pacotes do Node, o
NPM



Indica que será
solicitada a
instalação de um
pacote



Indica que o pacote
deverá ser instalado
para todos os
projetos e usuários
da máquina, de
forma global



Nome do pacote a
instalar

Compilando o programa

- Com a instalação da biblioteca, os comandos de compilação estarão disponíveis para uso
- Pode ser necessário fechar e abrir a janela do prompt de comando para que os mesmos estejam disponíveis (não é necessário abrir a nova janela como administrador)

Compilando o programa

- Na nova janela do prompt de comando, vá até a pasta do projeto (utilize o comando `cd` para tanto) e compile o mesmo com o seguinte:

`tsc exemplo01.ts`

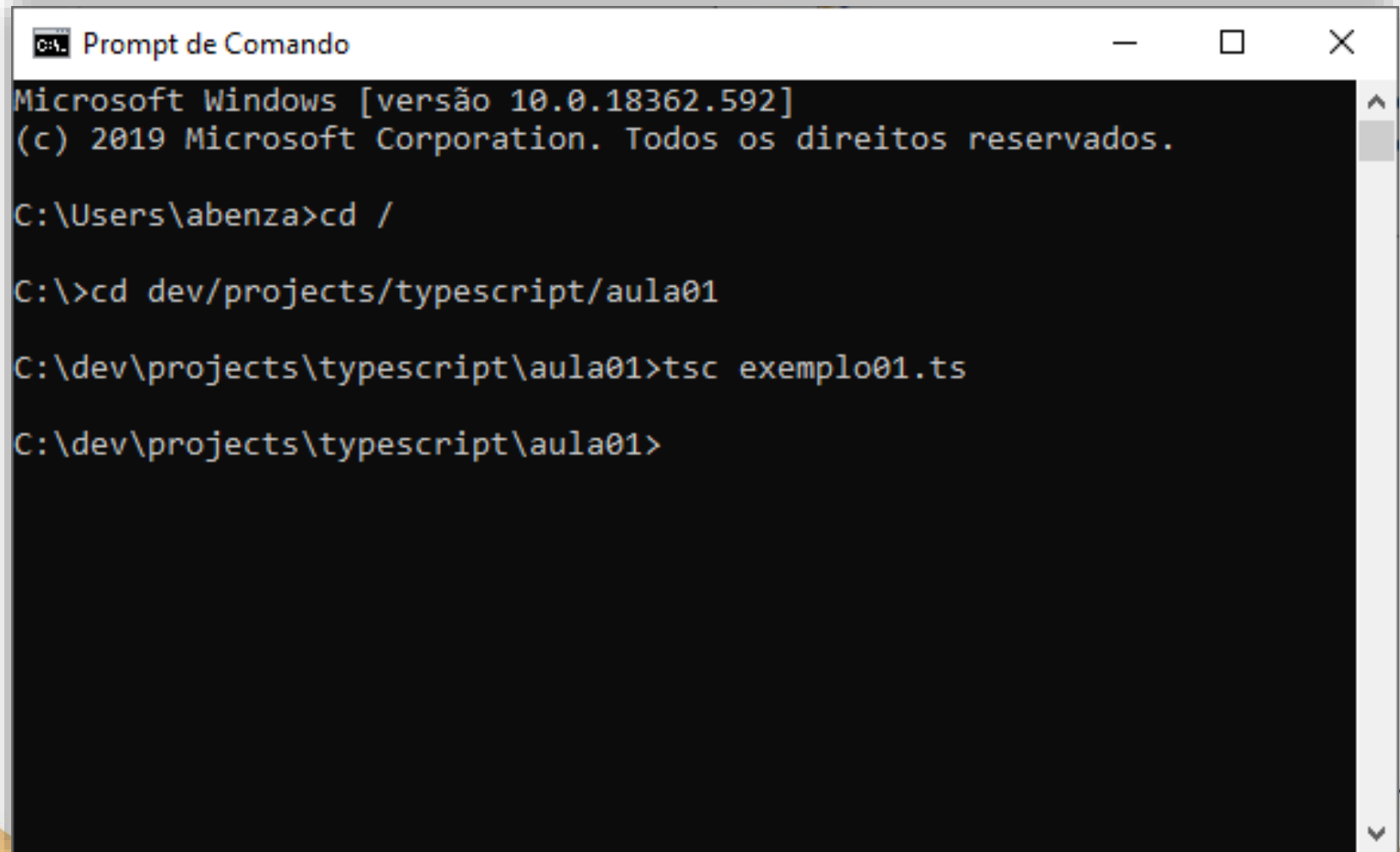
Em caso de problemas

- Caso ocorram problemas, abra um janela do PowerShell como administrador e execute:

Set-ExecutionPolicy RemoteSigned

Get-ExecutionPolicy

Compilando o programa



```

C:\> Prompt de Comando

Microsoft Windows [versão 10.0.18362.592]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\abenza>cd /

C:\>cd dev/projects/typescript/aula01

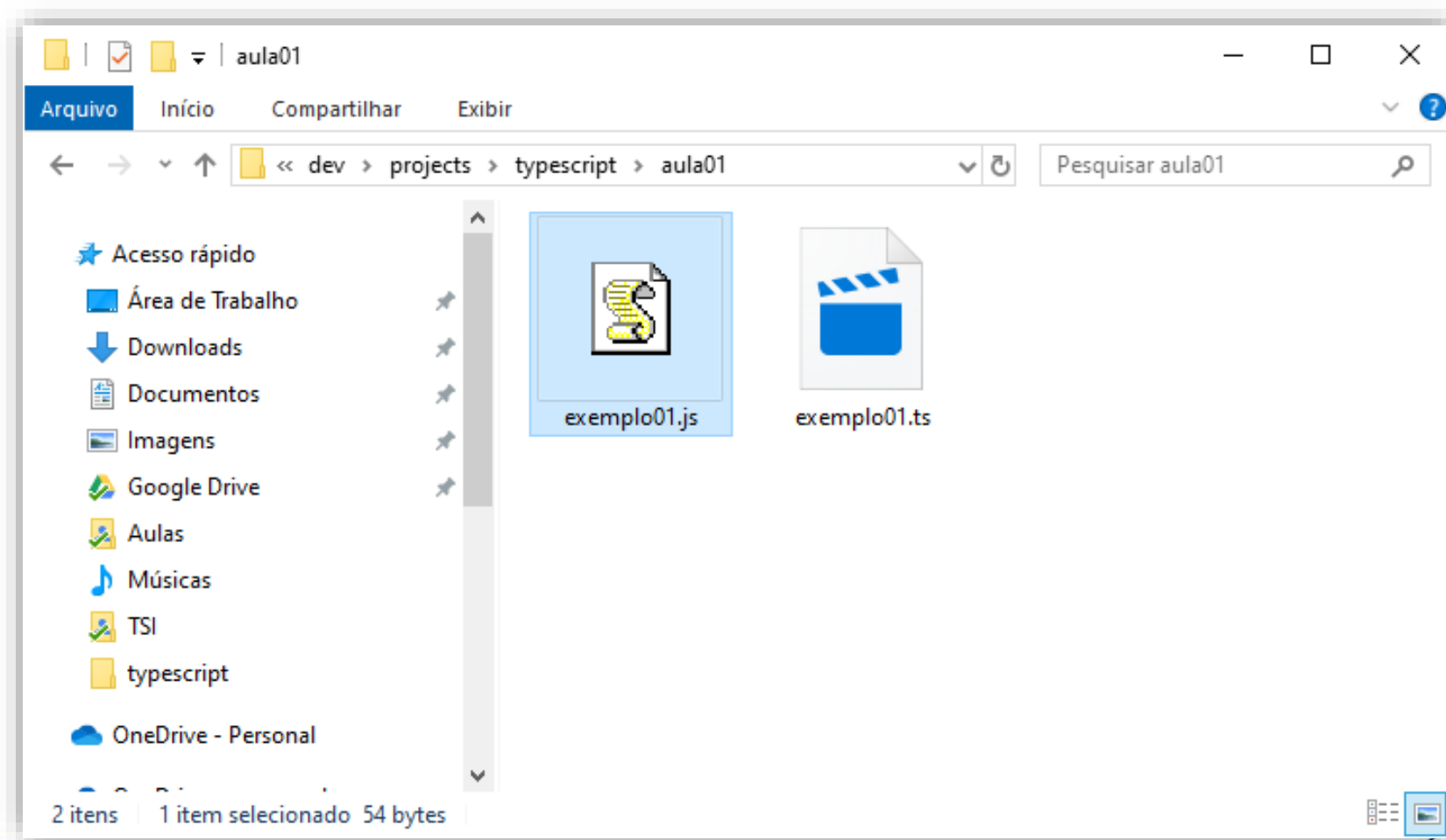
C:\dev\projects\typescript\aula01>tsc exemplo01.ts

C:\dev\projects\typescript\aula01>
```

Compilando o programa

- Observa a pasta do projeto. Um arquivo JavaScript contendo o programa “transpilado” para a linguagem terá sido criado

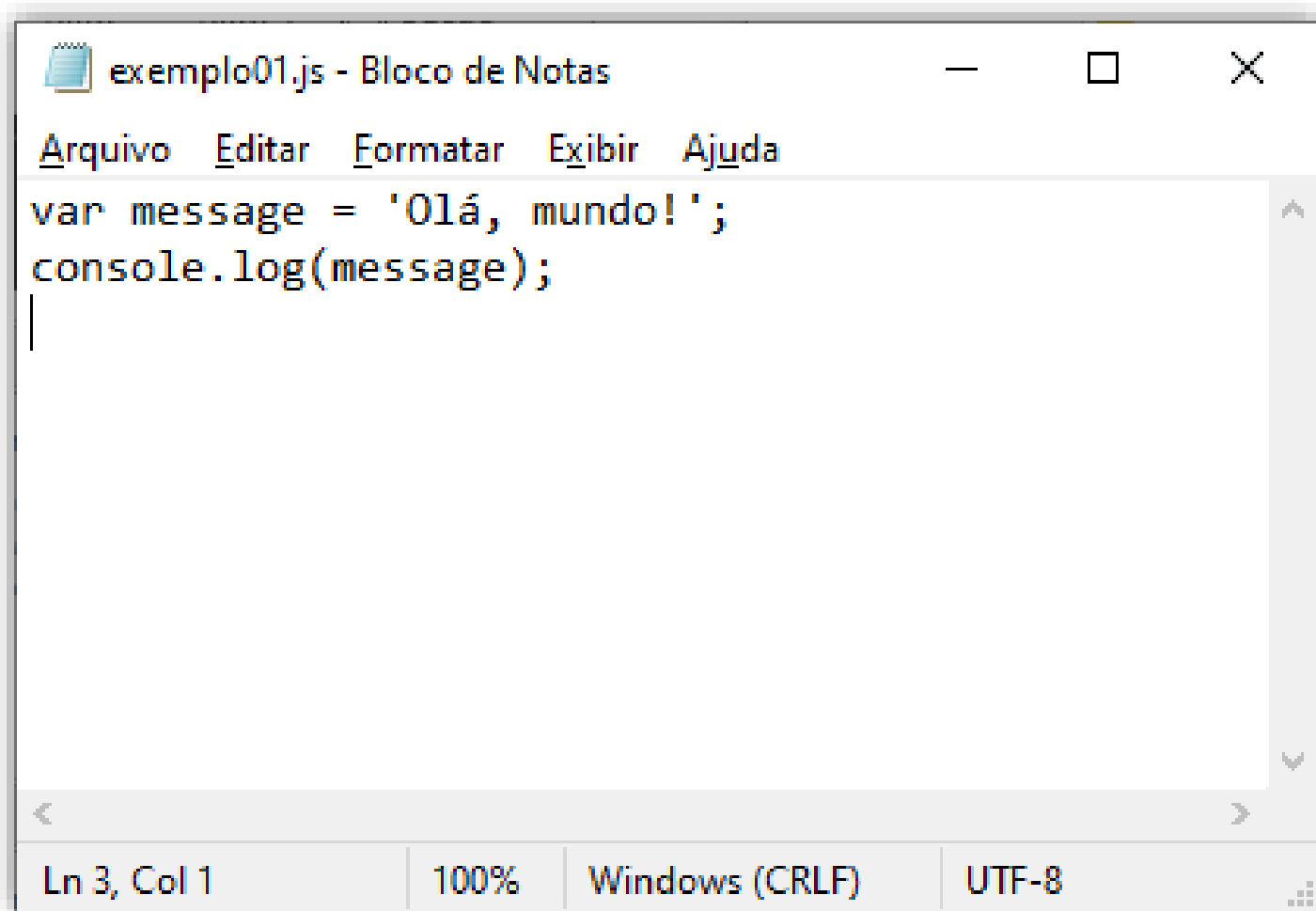
Compilando o programa



Compilando o programa

- Ao abrir o arquivo num editor de textos comum (ou mesmo no VisualStudio Code), será possível perceber que se trata de um arquivo JavaScript sem grandes diferenças com relação ao padrão da linguagem (isso pode variar dependendo da complexidade do programa)

Compilando o programa



A screenshot of a Windows Notepad application window. The title bar reads "exemplo01.js - Bloco de Notas". The menu bar includes "Arquivo", "Editar", "Formatar", "Exibir", and "Ajuda". The text area contains the following JavaScript code:

```
var message = 'Olá, mundo!';  
console.log(message);  
|
```

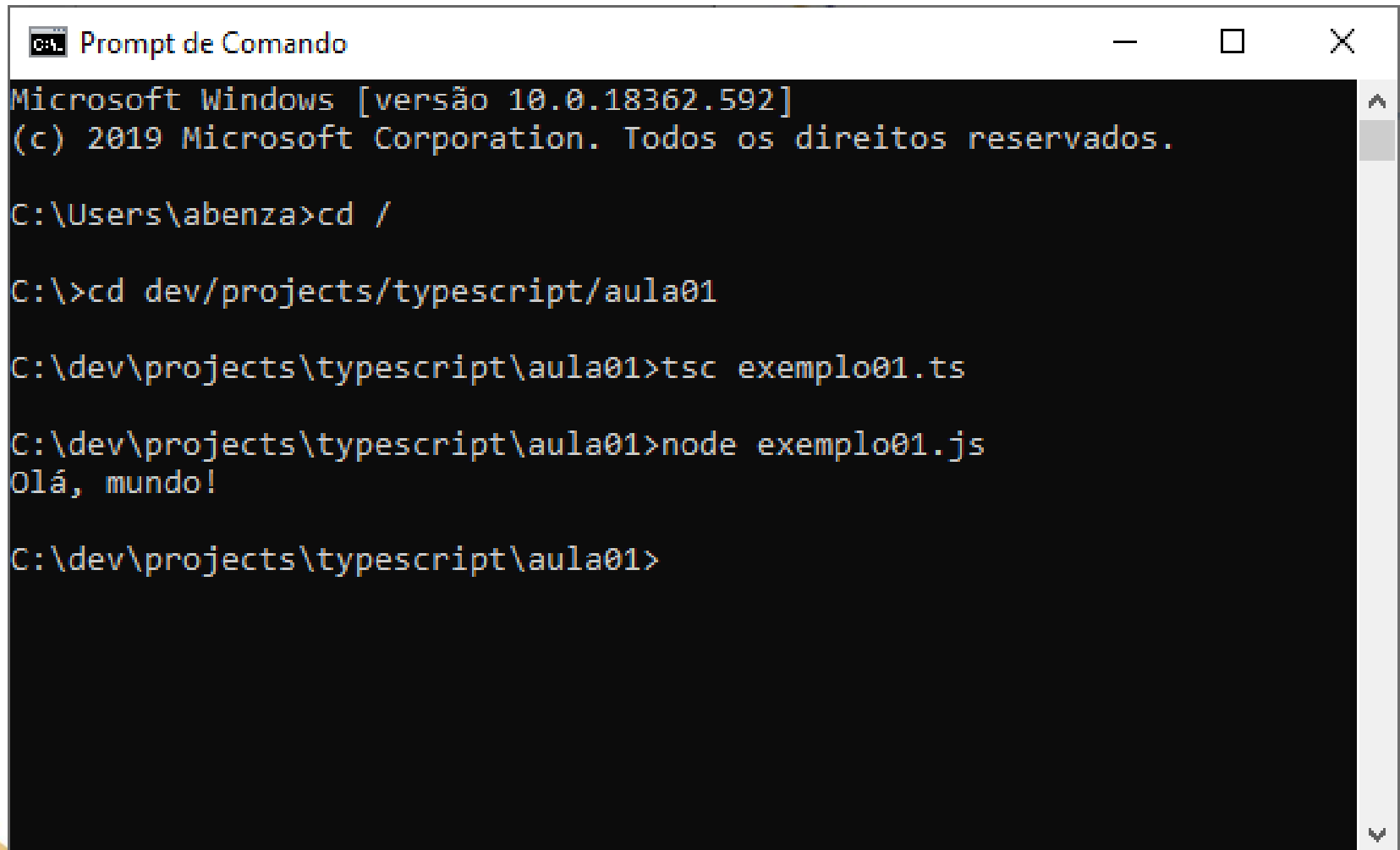
The status bar at the bottom shows "Ln 3, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Executando o programa

- Utilize o Node.js para executar o programa, através do comando:

node exemplo01.js

Executando o programa



```

C:\> Prompt de Comando

Microsoft Windows [versão 10.0.18362.592]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\abenza>cd /

C:\>cd dev/projects/typescript/aula01

C:\dev\projects\typescript\aula01>tsc exemplo01.ts

C:\dev\projects\typescript\aula01>node exemplo01.js
Olá, mundo!

C:\dev\projects\typescript\aula01>
```

ESTRUTURA BÁSICA DE CÓDIGO TYPESCRIPT

Estrutura Básica

- Assim como em JavaScript, código TypeScript pode ser digitado diretamente no arquivo e será executado na sequência em que aparecer (caso esteja fora de funções, classes, condições e etc)

Declaração de Variáveis

- Em TypeScript, variáveis são declaradas através de dois comandos, **const** e **let**
- **const** é utilizado quando a variável é imutável (pode receber valor apenas uma vez)
- **let** é utilizado para variáveis que podem receber valores mais de uma vez (mutáveis)

Declaração de Variáveis

- Além disso, TypeScript é uma linguagem **fortemente tipada**
- Isso quer dizer que sempre será necessário declarar os tipos das variáveis e tais tipos não podem ser modificados após a criação da variável

Declaração de Variáveis

- A declaração de tipos em TypeScript se dá através do símbolo de “:” logo após o nome da variável, seguido do tipo da mesma
- O tipo deve ser especificado tanto para variáveis mutáveis (let) como para imutáveis (const)

Declaração de Variáveis

- Variáveis let podem ser inicializadas quando necessário
- Variáveis const precisam ser inicializadas assim que declaradas

Tipos de Variáveis

Tipo	Palavra-chave	Descrição
Número	number	Representa números, sejam inteiros ou de ponto flutuante
Booleano	boolean	Representa valores de verdadeiro e falso (true e false)
String	string	Utilizado para texto
Void	void	Normalmente utilizado em funções para declarar que não retornam valores
Nulo	null	Representa um elemento que não possui valor
Indefinido	undefined	Representa um elemento não inicializado
Qualquer	any	Valor de variável sem tipo, aceita qualquer tipo de atribuição

Exemplos

```
const val: string = 'Teste';
```

```
let n: number;
```

```
const flag: boolean = true;
```

```
let qualquerCoisa: any;
```

Atribuição de Valores

- Assim como em JavaScript, a atribuição de valores em TypeScript é feita utilizando o =
- Variáveis let podem ser atribuídas a qualquer momento, elementos declarados com const só podem receber atribuição uma única vez, na declaração

Atribuição de Valores

- Também é possível atribuir valores em variáveis `let` assim que são declaradas, como é feito com JavaScript

Atribuição de Valores

```
const val: string = 'Teste';
```

```
let flag: boolean;  
flag = true;
```

```
const qualquerCoisa: any = 'AAA';
```

Entrada e Saída de Dados

- Como, por enquanto, estamos trabalhando com TypeScript puro, sem uso de páginas HTML ou do Angular, precisaremos de mecanismos para entrada e saída de dados no console

Entrada e Saída de Dados

- Faremos a entrada e a saída de dados como a seguir temporariamente, enquanto não estudamos o uso de Angular
- Quando passarmos a usar o Angular, esses mecanismos de entrada e saída não serão mais utilizados

Entrada e Saída de Dados

- Também seria possível utilizar TypeScript em páginas da web, como JavaScript tradicional
- Dessa forma, os mecanismos de entrada e saída seriam como sempre fazemos em páginas web (usando alert, prompt, document.getElementById e createElement e etc)

Saída de Elementos (Console)

- Para exibir valores no console, utilizamos o `console.log(texto)`, assim como em JavaScript
- O elemento console é um item pré-definido no Node.js (assim como nos navegadores) e sua função “log” permite exibir dados na saída principal do terminal

Saída de Elementos

```
console.log( 'Mensagem' );
```

```
let val: string = 'mensagem teste';  
console.log(val);
```

```
const num: number = 5;  
console.log(num);
```

Código de Exemplo

```
const message: string = 'Olá, mundo!';  
console.log(message);
```

Entrada de Dados (Console)

- Para entrada de dados no console, podemos utilizar o módulo JavaScript **readline-sync**
- Será necessário instalar o módulo
- Como o módulo será utilizado em runtime, será necessário remover o parâmetro “-g” e estar no diretório da aplicação ao rodar o comando

Entrada de Dados (Console)

- Vá até uma janela do console
- Utilize o comando `cd` para chegar até a pasta do projeto (caso não esteja)
- Digite o comando abaixo:

`npm install readline-sync`

Entrada de Dados (Console)

```
C:\dev\projects\typescript\aula01>npm install readline-sync
npm WARN saveError ENOENT: no such file or directory, open '
C:\dev\projects\typescript\aula01\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\
dev\projects\typescript\aula01\package.json'
npm WARN aula01 No description
npm WARN aula01 No repository field.
npm WARN aula01 No README data
npm WARN aula01 No license field.

+ readline-sync@1.4.10
updated 1 package and audited 1 package in 1.421s
found 0 vulnerabilities

C:\dev\projects\typescript\aula01>
```


Entrada de Dados (Console)

- Adicione a importação ao módulo no topo do arquivo
- Os comandos de importação em TypeScript seguem o formato **`import elemento as apelido from 'nome-modulo'`**

Entrada de Dados (Console)

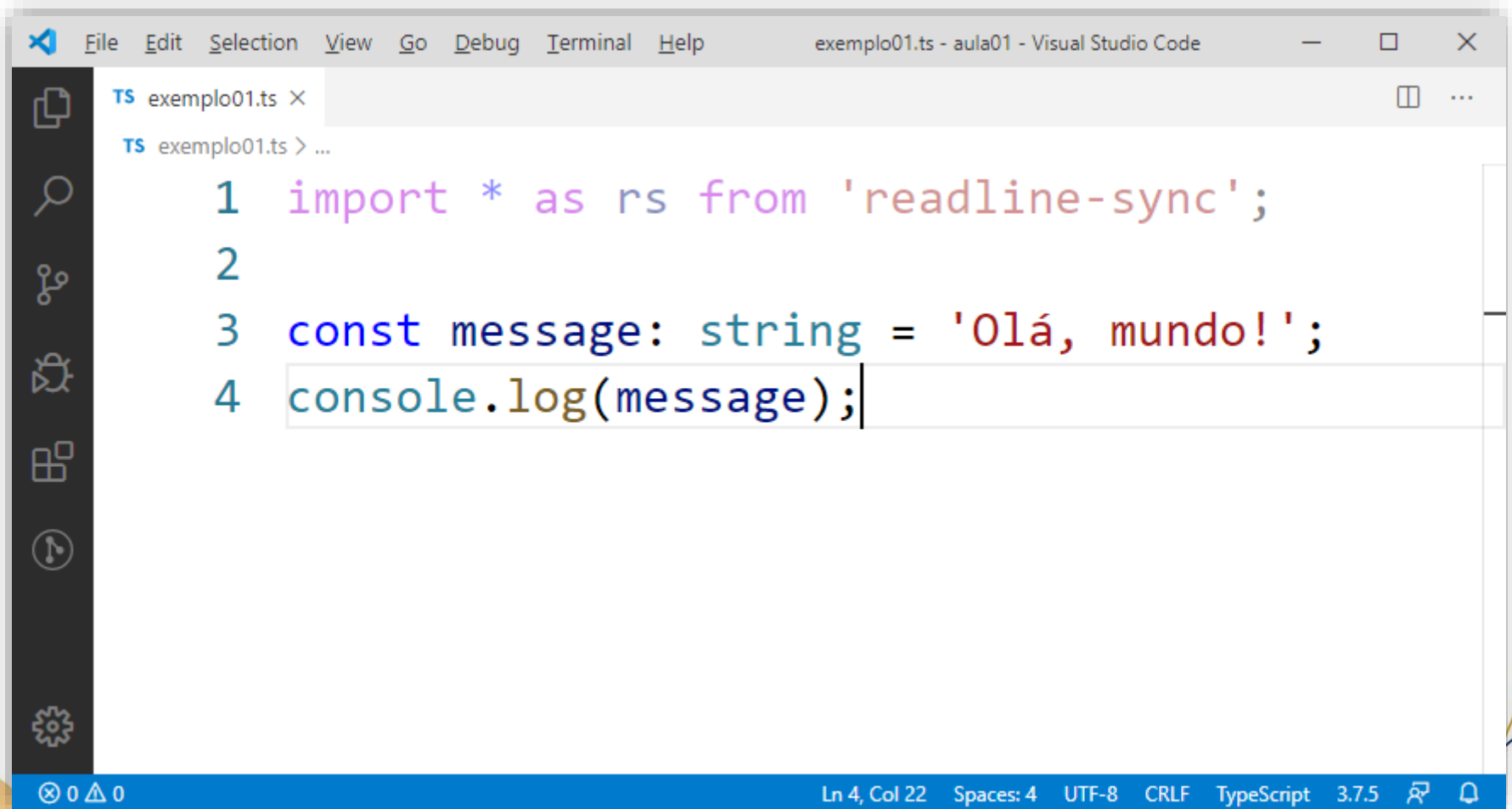
- O apelido nem sempre é necessário, dependendo de como o módulo é estruturado
- A importação de bibliotecas e elementos JavaScript (que estamos realizando) é diferente de elementos TypeScript (veremos a respeito em breve!)

Entrada de Dados (Console)

- Para importar o módulo readline-sync, adicione a linha abaixo ao topo do arquivo:

```
import * as rs from 'readline-sync';
```

Entrada de Dados (Console)



```
File Edit Selection View Go Debug Terminal Help exemplo01.ts - aula01 - Visual Studio Code
TS exemplo01.ts x
TS exemplo01.ts > ...
1 import * as rs from 'readline-sync';
2
3 const message: string = 'Olá, mundo!';
4 console.log(message);
```

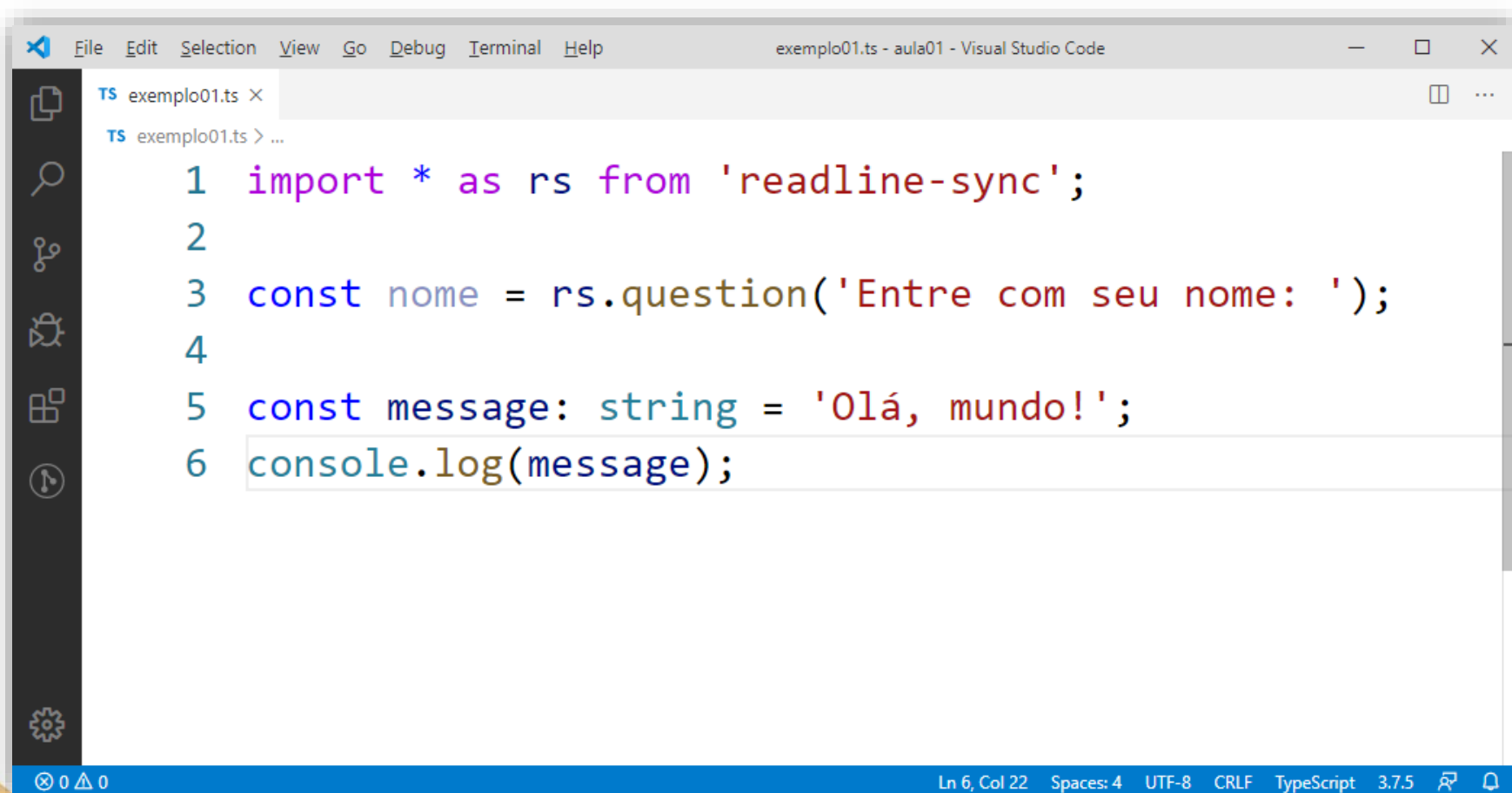
⊗ 0 △ 0 Ln 4, Col 22 Spaces: 4 UTF-8 CRLF TypeScript 3.7.5

Entrada de Dados (Console)

- Para ler um dado, utilize a função “question” do readline-sync e salve o valor em uma variável

```
const nome = rs  
  .question('Entre com seu nome: ');
```

Entrada de Dados (Console)



```
File Edit Selection View Go Debug Terminal Help exemplo01.ts - aula01 - Visual Studio Code
TS exemplo01.ts x
TS exemplo01.ts > ...
1 import * as rs from 'readline-sync';
2
3 const nome = rs.question('Entre com seu nome: ');
4
5 const message: string = 'Olá, mundo!';
6 console.log(message);
```

0 0 0 Ln 6, Col 22 Spaces: 4 UTF-8 CRLF TypeScript 3.7.5

Entrada de Dados (Console)

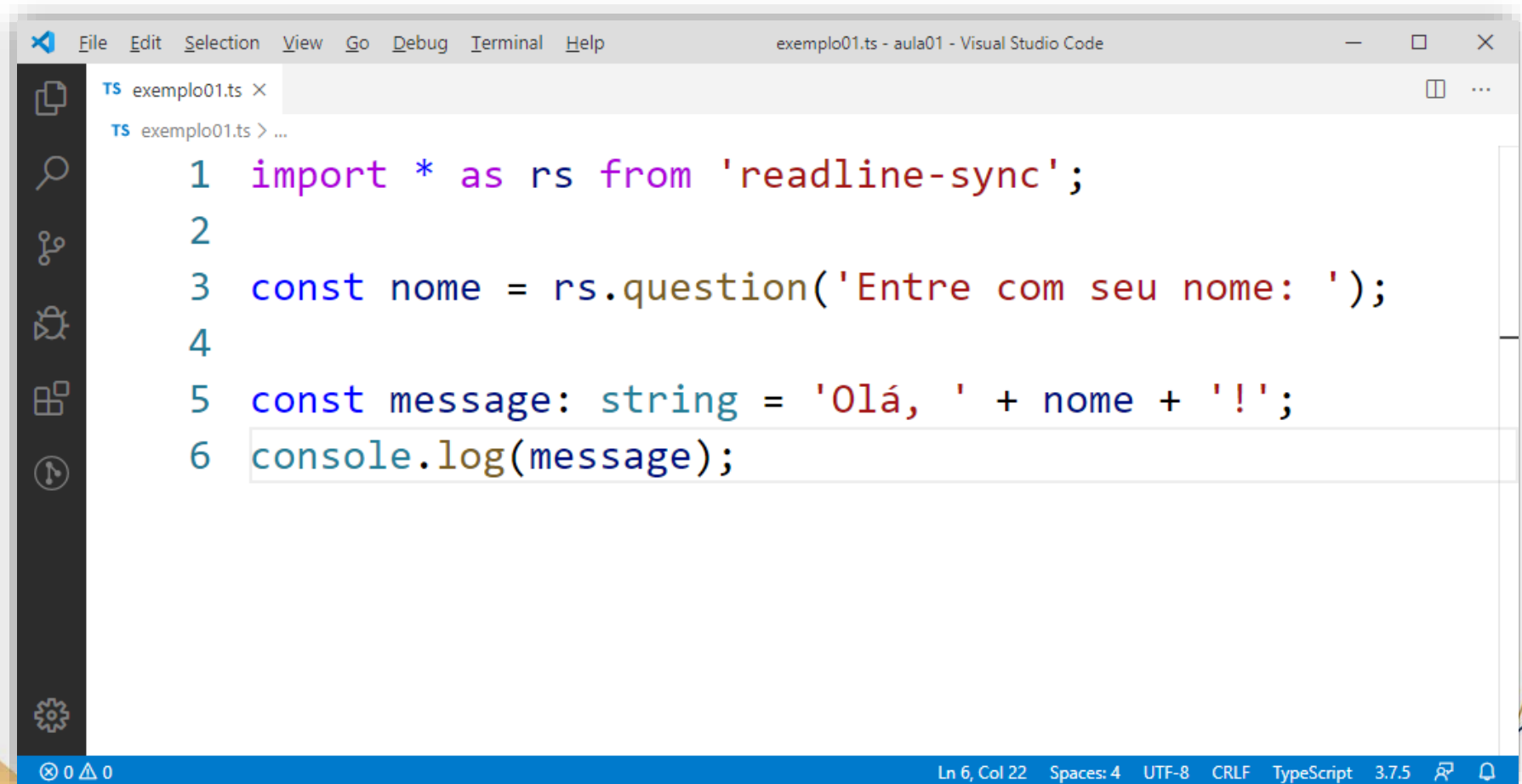
- Por fim, utilize a variável lida para exibir o nome lido na mensagem do console

```
const message: string = 'Olá, ' + nome + '!';
```

ou

```
const message: string = `Olá, ${nome}!`;
```

Resultado Final



```
File Edit Selection View Go Debug Terminal Help exemplo01.ts - aula01 - Visual Studio Code
TS exemplo01.ts x
TS exemplo01.ts > ...
1 import * as rs from 'readline-sync';
2
3 const nome = rs.question('Entre com seu nome: ');
4
5 const message: string = 'Olá, ' + nome + '!';
6 console.log(message);
```

Ln 6, Col 22 Spaces: 4 UTF-8 CRLF TypeScript 3.7.5

Exercícios

- Exercícios disponíveis no blackboard
- Dicas
 - Os operadores matemáticos em TypeScript são **idênticos** aos do JavaScript
 - É possível converter texto em números em TypeScript utilizando as funções **`parseInt(texto)`**, **`parseFloat(texto)`** ou **`Number(texto)`**

"That's all Folks!"