
SOFTEX
PERNAMBUCO

Softex

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

# Aula 27 | Módulo: Typescript e Orientação a Objetos (continuação)

- ➔ Polimorfismo: sobrescrita e comportamento dinâmico
- ➔ Tratamento de exceções: try/catch, criação de erros personalizados

**Caso queira estudar usando um editor na web**

- Acesse:
  **https://vite.new/react-ts**

# Typescript e Orientação a Objetos (continuação)

## Configurando o ambiente para rodar o React

- Node.js (LTS recomendado >= 20):
  - **Windows/macOS:**
    - baixe o instalador LTS no site do Node.
  - **Linux (Deb/Ubuntu):**
    - curl -fsSL https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
    - nvm install --lts
    - nvm use --lts
- Confirmar instalação rodando os comandos abaixo em algum terminal:
  - **node -v**
  - **npm -v**
- Se “npm/node não reconhecido”, feche o terminal e tente pelo Command Prompt.

# Typescript e Orientação a Objetos (continuação)

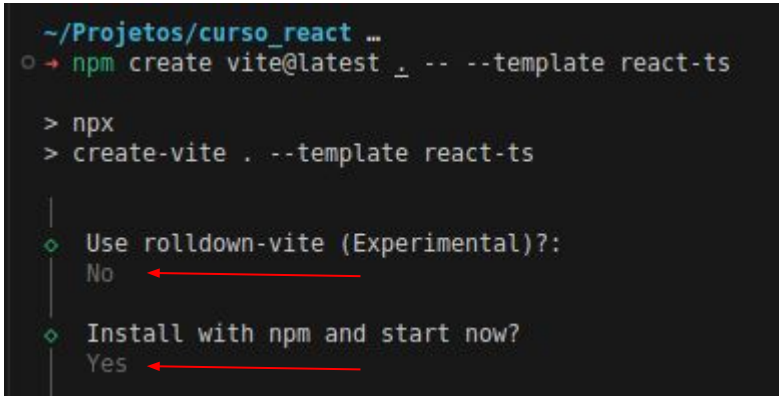
## Abrindo/Criando um projeto

- Para ABRIR um projeto existente:
  - Abra o VSCode, clique em File > Open Folder… (Arquivo > Abrir Pasta…) e procurar a pasta do projeto existente.
  - No terminal, rode o comando ao lado para iniciar: **npm run dev**
- Para CRIAR um projeto novo
  - Abra o VSCode, clique em File > Open Folder… (Arquivo > Abrir Pasta…).
  - Escolha um local para criar a sua pasta, crie uma nova pasta e dê o nome de **seunome_aula_react_27**. Depois dê dois clique nessa pasta criada e clique em Selecionar pasta. O VSCode reabrirá dentro dessa pasta que foi criada.

# Typescript e Orientação a Objetos (continuação)

## Criar o projeto com Vite

- Criando um projeto com TypeScript:
  - **npm create vite@latest . -- --template react-ts**
- Durante a instalação do React usando o Vite, algumas opções podem surgir:



# Typescript e Orientação a Objetos (continuação)

**Se algo travar: portas ou permissões**

- Porta ocupada (erro ao iniciar ou tela vazia):
  - Iniciar em outra porta: **npm run dev -- --port 5174**
- Windows: “npm não reconhecido”
  - Feche e reabra o PowerShell/CMD.
  - Verifique se o Node foi instalado para “All Users” (às vezes a variável de ambiente não atualiza).
- Linux/macOS: erro de permissão (EACCES)
  - mkdir -p ~/.npm-global
  - npm config set prefix ~/.npm-global
  - echo 'export PATH=$HOME/.npm-global/bin:$PATH' >> ~/.bashrc && source ~/.bashrc

# Typescript e Orientação a Objetos (continuação)

## O que é Polimorfismo?

- “Polimorfismo” significa muitas formas.
- Em POO, permite que métodos com o mesmo nome se comportem de forma diferente em classes diferentes.
- Normalmente aparece quando usamos herança.
- Exemplo: todos os animais têm o método **emitirSom()**, mas o som muda conforme o tipo.

# Typescript e Orientação a Objetos (continuação)

## Exercício prático sobre Polimorfismo

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/Polimorfismo.tsx**
- Ou faça o clone do repositório e acesse a branch “polimorfismo”:
  - Comando: **git checkout polimorfismo**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/polimorfismo/src**

# Typescript e Orientação a Objetos (continuação)

## Exercício 1

- Crie uma classe base **Instrumento** com o método **tocar()**.
- Crie classes **Violao**, **Piano** e **Bateria** que sobrescrevam **tocar()**.
- Mostre o comportamento polimórfico em um componente React.

# Typescript e Orientação a Objetos (continuação)

## Lidando com erros em TypeScript

- Às vezes, nosso código pode gerar erros inesperados (ex: uma conta errada ou dado inválido).
- O React não trava se usarmos try e catch para “capturar” o erro e reagir de forma controlada.
  - **try** = tentar executar
  - **catch** = o que fazer se der erro
- Podemos também personalizar a mensagem de erro:
  - **throw new Error("mensagem")** interrompe o código e envia um erro com texto personalizado.
- O catch recebe esse erro e permite exibir a mensagem na tela.
- O React continua funcionando normalmente, apenas mostra o erro ao usuário.

# Typescript e Orientação a Objetos (continuação)

## Lidando com erros em TypeScript

- Às vezes, nosso código pode gerar erros inesperados (ex: uma conta errada ou dado inválido).
- O React não trava se usarmos try e catch para “capturar” o erro e reagir de forma controlada.
  - **try** = tentar executar
  - **catch** = o que fazer se der erro
- Podemos também personalizar a mensagem de erro:
  - **throw new Error("mensagem")** interrompe o código e envia um erro com texto personalizado.
- O catch recebe esse erro e permite exibir a mensagem na tela.
- O React continua funcionando normalmente, apenas mostra o erro ao usuário.

# Typescript e Orientação a Objetos (continuação)

## Exercício prático de tratamento de erros em TypeScript

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/TratamentoErros.tsx**
- Ou faça o clone do repositório e acesse a branch “tratamentoerros”:
  - Comando: **git checkout tratamentoerros**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/tratamentoerros/src**

## Lidando com erros em TypeScript

- Agora faremos um exemplo mais útil:
  - O usuário tenta converter texto em número, e o sistema verifica se ele digitou algo válido.
  - Se não for um número, exibimos uma mensagem de erro amigável.

# Typescript e Orientação a Objetos (continuação)

## Exercício prático de tratamento de erros em TypeScript

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/ValidacaoSimples.tsx**
- Ou faça o clone do repositório e acesse a branch “validacaosimples”:
  - Comando: **git checkout validacaosimples**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/validacaosimples/src**

## Lidando com erro (SyntaxError)

- O SyntaxError aparece quando o código tem erro de estrutura (ex: parêntese faltando, JSON malformado, etc.).
- Em React, não conseguimos “criar” um erro de sintaxe diretamente (o compilador não deixa), mas podemos **simular** um erro real usando funções como **eval()** ou **JSON.parse()**.
- Esses casos geram SyntaxError automaticamente, e podemos capturá-los com try/catch.

# Typescript e Orientação a Objetos (continuação)

## Exercício prático de tratamento de erros em TypeScript

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/ErroDeSintaxe.tsx**
- Ou faça o clone do repositório e acesse a branch “errosintaxe”:
  - Comando: **git checkout errosintaxe**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/errosintaxe/src**

# Typescript e Orientação a Objetos (continuação)

## Lidando com erro (ReferenceError)

- O **ReferenceError** acontece quando o código tenta usar algo que não foi declarado, como uma variável inexistente.
- Em React, o compilador normalmente impede isso, mas podemos simular o erro dentro de uma função, tentando acessar uma variável que não existe.
- Esse tipo de erro é gerado automaticamente e pode ser capturado com try/catch, evitando que o aplicativo trave.

# Typescript e Orientação a Objetos (continuação)

## Exercício prático de tratamento de erros em TypeScript

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/ErroDeReferencia.tsx**
- Ou faça o clone do repositório e acesse a branch “erroreferencia”:
  - Comando: **git checkout erroreferencia**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/erroreferencia/src**

# Typescript e Orientação a Objetos (continuação)

## Outros Tipos de Erros

- O JavaScript (e o TypeScript) também possui outros tipos de erros padrão, que aparecem em situações específicas, e todos podem ser tratados com try/catch.

| Tipo de Erro | Quando acontece | Exemplo |
|---|---|---|
| TypeError | Quando tentamos usar um valor de tipo incorreto. | null.toString() ou "texto" * 5 |
| RangeError | Quando o valor está fora do intervalo permitido. | new Array(-1) |
| URIError | Quando funções de URL recebem valores inválidos. | decodeURI("%") |
| EvalError | Quando eval() é usado de forma incorreta. | eval("if )") |
| AggregateError | Quando várias promessas falham ao mesmo tempo. | Promise.any([]) |

➔ Todos herdam de Error.
➔ Cada tipo tem um significado específico.
➔ O try/catch pode tratar qualquer um deles.
➔ Use o tipo certo para **melhorar a clareza do seu código**.

# Typescript e Orientação a Objetos (continuação)

## Gabarito Exercício 1

- Vamos acessar o repositório abaixo e copiar o código dos arquivos:
  - **src/App.tsx**
  - **src/components/Instrumento.tsx**
- Ou faça o clone do repositório e acesse a branch “instrumento”:
  - Comando: **git checkout instrumento**
- Link do repositório:
  - **https://github.com/hygorrasec/curso_react/tree/instrumento/src**

# ATÉ A PRÓXIMA AULA!

*Front-end - Design. Integração. Experiência.*

**Professor:** Hygor Rasec
https://www.linkedin.com/in/hygorrasec
https://github.com/hygorrasec