async e await

async e await

Funções Assíncronas ⏱️

- Na última aula vimos que para fazer requisições precisamos usar funções assíncronas

- Para facilitar um pouco a nossa vida, existem as Promises, com o `.then()` e o `.catch()`

Funções Assíncronas

```
1 import axios from 'axios'
2
3 const getUsers = () => {
4   axios.get('https://users-api.com/users', {
5     headers: {
6       Authorization: 'nome-sobrenome-turma'
7     }
8   }).then((response) => {
9       console.log(response.data)
10  }).catch((error) => {
11      console.log(error.response)
12  })
13 }
```

Jeito novo - async/await ⏱️

- Existe uma maneira de pedir para que o código **ESPERE** a execução da requisição antes de progredir
- Podemos usar a palavra **await** antes de uma Promise
- Isso só é possível dentro de funções assíncronas, que devem ser marcadas com a palavra **async**
- Resultado da Promise é passado diretamente para uma **variável**

Jeito novo - async/await ⏱️

- Resultado da promise é passado diretamente para uma variável

```
1  import axios from 'axios'
2
3  const getUsers = async () => {
4    const response = await axios.get('https://users-api.com/users', {
5      headers: {
6        Authorization: 'nome-sobrenome-turma'
7      }
8    })
9
10   console.log(response.data)
11 }
```

Mas e os erros? ⏱️

- Para tratar erros, usamos a sintaxe **try/catch**, como mostrado abaixo:

```
1  import axios from 'axios'
2
3  const getUsers = async () => {
4    try {
5      const response = await axios.get('https://users-api.com/users', {
6        headers: {
7          Authorization: 'nome-sobrenome-turma'
8        }
9      })
10
11     console.log(response.data)
12   } catch(error) {
13     console.log(error.response)
14   }
15 }
```

Mas e os erros? ⏱️

- Para tratar erros, usamos a sintaxe **try/catch**, como mostrado abaixo:

```
1  import axios from 'axios'
2
3  const getUsers = async () => {
4    try {
5      const response = await axios.get('https://users-api.com/users', {
6        headers: {
7          Authorization: 'nome-sobrenome-turma'
8        }
9      })
10
11     console.log(response.data)
12   } catch(error) {
13     console.log(error.response)
14   }
15 }
```

Mas e os erros? ⏱️

- Bloco dentro do **try** é executado

```
1  import axios from 'axios'
2
3  const getUsers = async () => {
4    try {
5      const response = await axios.get('https://users-api.com/users', {
6        headers: {
7          Authorization: 'nome-sobrenome-turma'
8        }
9      })
10
11     console.log(response.data)
12   } catch(error) {
13     console.log(error.response)
14   }
15 }
```

Mas e os erros? ⏱️

- Se der erro, execução é interrompida e o bloco do **catch** é executado

```
1  import axios from 'axios'
2
3  const getUsers = async () => {
4    try {
5      const response = await axios.get('https://users-api.com/users', {
6        headers: {
7          Authorization: 'nome-sobrenome-turma'
8        }
9      })
10
11     console.log(response.data)
12   } catch(error) {
13     console.log(error.response)
14   }
15 }
```