```
//ASYNC / AWAIT:

//O "async" (assícrono) e o "await" (aguardar), são funções do Promise que podem ser utilizadas para quando desejamos que determinado trecho do nosso código funcione de forma assícrona, assim como a promise, e só seja executada depois que um trecho também assícrono seja executado antes dele.

//Por padrão, já sabemos que as nossas promises são assícronas, elas vão demorar um pouco para responder enquanto o nosso código vai executando. Mas podemos ter determinadas execuções que precisam de receber os dados de uma promise para que possam ser executadas. Para evitar que elas sejam executadas antes da promise terminar sua execução, foram criadas as funções "async" e "await":
/*
        * async: torna uma determina função assícrona, a palavra reservada "async" deve ser usada antes da função;
        * await: só pode ser usada dentro de uma função que tiver recebido "async", o await faz com que determinado trecho de
                código dentro da função espere até a chamada de uma promise termine para que ele seja executado...
 */

//Abaixo temos o exemplo de "async" e "await", onde os códigos de execução de alguns dados que se encontram numa URL terão de esperar até que uma promisse termine a sua execução para que eles possam ser executados...

//Aqui temos a biblioteca "http" ativada...
const http = require('http')

//Abaixo temos uma função que retorna uma promisse com os dados de alunos de uma turma, essa turma é escolhida de acordo com a letra, que é passada como parâmetro dessa função...
const getTurmaPromise = letra => {
    const url = `http://files.cod3r.com.br/curso-js/turma${letra}.json`
    return new Promise((resolve, reject) => {
        http.get(url, res => {
            let resultado = ''

            res.on('data', dados => {
                resultado += dados
            })

            res.on('end', () => {
                try {
                    resolve(JSON.parse(resultado))
```

```
                } catch(e){
                    reject(e)
                }
            })
        })
    })

}

//Aqui temos a função sícrona, que desejamos tornar assícrona...
let obterAlunos = async () => { //Note que estamos usando usando o async antes de chamar a função, nesse caso, como trata-se de uma arow function, estamos usando o async antes do campo de parâmetros, se fosse uma função normal o "async" seria colocado antes do nome "function"...
    console.log('fui executado, mas to esperando as promises acabarem para que eu possa continuar...') //Perceba que só colocar o async, não tornará uma função assícrona automáticamente, é preciso trabalhar em conjunto com o "await", é o await que vai dizer quando a execução deve esperar até que uma promisse seja cumprida...
    const turmaA = await getTurmaPromise('A') //Agora perceba que o await é usado antes de chamar a função que contém promise, isso sigfica que só depois que a execução da promisse terminar é que o restante da execução pós-await terá continuidade.
    const turmaB = await getTurmaPromise('B')
    const turmaC = await getTurmaPromise('C')
    return [].concat(turmaA, turmaB, turmaC)
}

obterAlunos()
    .then(alunos => alunos.map(a => a.nome))
    .then(nomes => console.log(nomes))
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\157-async_e_await__Trabalhando_Funcoes_de_Forma_Sincrona_com_a_Promise.js"
fui executado, mas to esperando as promises acabarem para que eu possa continuar...
[
  'Kellia',   'Hi',     'Inge',
  'Myrle',    'Doreen', 'Pennie',
  'Faye',     'Leena',  'Taylor',
```

```
  'Juieta',   'Rossie', 'Mary',
  'Dionysus', 'Myca',   'Sharlene',
  'Meghan',   'Perice', 'Micheil',
  'Nat',      'Bone',   'Kellina',
  'Barrie',   'Darda',  'Rainer',
  'Joan',     'Kasper', 'Sammie',
  'Scott',    'Kiel',   'Dell'
]

[Done] exited with code=0 in 1.818 seconds
```