```
//PROMISE:

//As promises são um tipo de função do Javascript que utilizamos para basear as nossas requisições de um conteúdo por através de URL na rede. Por que são chamadas de promises? Por que esse é justamente o conceito de uma requisição URL, quando fazemos uma requisição, temos a promessa de que o recurso estará realmente disponível ou não infelizmente ele não estará, gerando um erro na nossa aplicação.

//As promises foram criadas justamente para lidar com essas requisições, onde podemos adicionar um tratamento ao conteúdo informático caso a promessa seja resolvida, ou poderemos dar um tratamento de erro caso a promessa seja rejeitada. Trabalhando nesses 2 conceitos, a promisse possuí 2 métodos, um para resolução e outro para rejeição:
//      * .then() - usado quando uma promise recebeu o dado informático;
//      * .catch() - usado quando ocorreu algum tipo e a promise não conseguiu o conteúdo informático;

//Tanto ".then()" quanto ".catch()" recebem callbacks que irão realizar um tratamento sobre os dados de acordo com as suas respectivas funcionalidades, o que significa que, "then()" só vai receber dados quando o promise tiver dado certo, e "catch()" só vai receber dados quando o promise tiver dado errado. Como then() e catch() sabem que os dados deram errado ou deram certo? Isso acontece por que promise recebe 2 parâmetros:
/*      promise(resolve, reject)
            onde "resolve" será uma callback que será executada se o promise tiver dado certo e "reject" é outra callback
            que será executada quando promisse tiver dado errado...*/

//Um detalhe importante das promises é que elas sempre irão trabalhar com chamadas assícronas, ou seja, enquanto o código executa, a promessa é chamada e procura por uma resolução retornando ela quando o processo acaba.

//Promise, faz parte de uma função construtora, por isso devemos instanciar um objeto onde desejamos gerar um promise...

//GERANDO UM PROMISE FICTÍCIO:
//Abaixo estamos gerando um promise que recebe uma requisição e uma resposta positiva, note que esse promisse possuí um time para acontecer, mostrando que ele é assícrono e só é executado quando o time acaba, depois temos uma execução independente que será executada antes do promise...
function falarDepoisDe(segundos, frase) {
    return new Promise((resolve, reject) => { //Veja que o retorno da função irá retornar uma instância de um promise que recebe 2 parâmetros, o 1º é sempre uma função callback que será executada se o promisse tiver dado certo e o 2º sempre será uma função callback para ser executada se o promise tiver dado errado...
        setTimeout(() => { //Perceba que dentro do promisse temos uma função arrow que vai executar o resolve - caso os dados sejam pegos - ou o reject - caso ocorra algum erro.
```

```
            resolve(frase) //Note que tanto resolve quanto reject só podem receber um parâmetro, se quisermos que mais parâmetros sejam passados temos que usar um objeto...
            reject(frase)
        }, segundos * 1000)
    })
}

falarDepoisDe(2, 'Essa frase veio de um promise...') //Aqui temos a passagem de parâmetros para a função que irá gerar o promise, do jeito que está a execução irá entrar só no resolve, mas se desejar que ela gere um erro, apague um dos pâmetros...
    .then((frase) => frase.concat(' Sim ela veio')) //Veja que estamos usando then() e catch() encadeados, e assim como os callback de promisse, then() e catch() só podem receber 1 parâmetro...
    .then(fraseNova => console.log(fraseNova + ', mas só depois de 2 segundos'))
    .catch(frase => console.log(frase + '!!!\nHouve o erro descrito acima, queira se atentar para resolvê-lo!')) //catch() será executado só se o promisse der errado...

console.log('Esse execução é aleatória e fora do promise...') //Esse console.log() é independente do promise, sendo assim, ele será executado antes, de forma síncrona, enquando o promise será executado depois, respeitando o seu aspecto assícrono...
```

RESULTADO NO CONSOLE CASO O PROMISE DÊ CERTO...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\155-Promises.js"
Esse execução é aleatória e fora do promise...
Essa frase veio de um promise... Sim ela veio, mas só depois de 2 segundos

[Done] exited with code=0 in 2.11 seconds
```

RESULTADO SE O PROMISSE DER ERRADO...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\155-Promises.js"
Esse execução é aleatória e fora do promise...
TypeError: Cannot read property 'concat' of undefined!!!
Houve o erro descrito acima, queira se atentar para resolvê-lo!

[Done] exited with code=0 in 0.126 seconds
```