08 - Conhecendo as Promises

- Uma promise é uma forma mais fácil de trabalhar com códigos assíncronos no javascript. Ela é um objeto que é instanciado através da classe Promise e recebe uma função como parâmetro. Essa função é executada quando a promise é criada, porém sem bloquear a execução do código:

```
const p = new Promise(() => { console.log('A promise está sendo executada.') setTimeout(() => { console.log('Resolvendo a promise...') }, 3 * 1000) }) console.log(p)
```

- Como visto no código acima, uma promise possui diferentes estados, que são: pendente (estado padrão), rejeitada e resolvida (ou cumprida). A promise ficará em estado pendente até que seja resolvida utilizando o primeiro parâmetro da função usada na sua construção. Esse primeiro parâmetro é uma função que "resolve" a promise e passa adiante o valor do resultado:

```
const p = new Promise((resolve, reject) => { console.log('A promise está sendo executada.') setTimeout(() => { console.log('Resolvendo a promise...') resolve('Resultado') }, 3 * 1000) }) console.log(p) setTimeout(() => { console.log(p) }, 5 * 1000)
```

- Também podemos rejeitar a promise caso queiramos indicar que algo deu errado:

  Obs.: repare que nesse caso a rejeição da promise é como um erro no código, que se não for tratado adequadamente irá resultar no término da aplicação. Veremos como fazer isso na próxima aula.

```
const p = new Promise((resolve, reject) => { console.log('A promise está sendo executada.') setTimeout(() => { if (1 + 1 === 2) { reject("Algo deu errado!") } console.log('Resolvendo a promise...') resolve('Resultado') }, 3 * 1000) }) console.log(p) setTimeout(() => { console.log(p) }, 5 * 1000)
```

- Por fim, a prática comum ao usar as promises é retorná-las no final de uma função, assim elas serão executadas quando a função for chamada:

```
function execute() { return new Promise((resolve, reject) => {
console.log('A promise está sendo executada.') setTimeout(() => {
console.log('Resolvendo a promise...') resolve('Resultado') }, 3 *
1000) }) } const p = execute() console.log(p) setTimeout(() => {
console.log(p) }, 5 * 1000)
```