



new Promise()

Além do `.then()` e do `async/await`, o JavaScript também tem um método construtor para resolver promessas, o `Promise()`.

Enquanto `.then()` e `async/await` são utilizados quando temos que lidar com promessas já existentes - por exemplo, ao executarmos o método `fetch()` que, por definição, **sempre retorna uma promessa**, usamos o construtor `Promise()` para escrever do zero nossas próprias promessas.

Vamos ver um exemplo de função que recebe um valor booleano (`true` ou `false`) e com base nesse valor retorna uma `new Promise()` rejeitada ou completa.

```
function promessa(bool) {
  const x = bool;
  return new Promise((resolve, reject) => {
    if (!x) {
      reject(new Error("falha na promessa"));
    }
    resolve("sucesso na promessa");
  });
}

function exibeResposta(textoResult) {
  console.log(textoResult);
}

promessa(true)
```

```
.then((texto) => exibeResposta(texto))  
// sucesso na promessa
```

Veja que a função `promessa()` cria uma nova promessa a partir do construtor `new Promise()` e com dois parâmetros: `resolve` e `reject`. O objeto `Promise()` precisa trabalhar sempre com estes dois parâmetros, que devem ser invocados após a resolução (seja com ou sem sucesso).

Neste caso, passamos um texto como parâmetro de cada um deles. Quando executamos a função `promise(true)` este valor é carregado através das promessas até ser passado para a função `exibeResposta(texto)`, que por fim vai exibir a mensagem correta. No caso de `promessa(false)`, além da mensagem “falha na promessa” o NodeJS também vai lançar no terminal a stacktrace do objeto `Error`.

1. Promessas podem ser concluídas de duas formas: *fulfilled* (realizada, completa) ou *rejected* (rejeitada), o que equivale a duas situações possíveis, ou a promessa se concretizou (retornou os dados ou executou o código que deveria) ou não.
2. Promessas que não estão *fulfilled* nem *rejected* estão *pending* (pendentes). Ou seja, ainda não é possível saber o resultado final porque o processamento ainda não foi concluído.
3. Após a finalização do processamento, a promessa passa para o estado de *settled* (concluída), independente do resultado.
4. Uma vez que a promessa está *settled* seu resultado não se altera mais. Ou seja, uma promessa que se concluiu como *rejected* não muda mais para o estado de *fulfilled* e vice-versa.