# **Como uma "promessa" é feita**

# Aprendemos logo nos primeiros capítulos que as funções de $http (get, post, put, delete) devolvem uma promessa, no inglês *promise*. Uma promise contém o resultado futuro de uma ação. Quando a ação é bem sucedida, temos acesso ao valor retornado da ação, através da função then e erros através da funçãocatch. Apesar de $http suportar as funções success e error, por baixo dos panos uma promise é utilizada.

# Trabalhar com o *pattern* promise não é exclusividade do Angular, podemos implementá-lo em nosso código. Para isso, o Angular possui o serviço $q, especializado na criação de promises. Normalmente, retornamos uma promise a partir de uma chamada de função. Vejamos um exemplo:

function exibe(texto) {  
 return $q(function(resolve, reject) {  
  
 // simulando ação assíncrona com setTimeout  
  
 setTimeout(function() {  
 if(texto == 'Alura') {  
 resolve('resolvida');  
 } else {  
 reject('rejeitada')  
 }  
 }, 5000);  
 });   
}  
  
// executando nossa promise  
  
exibe('Alura').then(function(resultado) {  
 console.log(resultado);  
}).catch(function(erro) {  
 console.log(erro);  
});  
console.log('FIM');

O código acima é funcional, sendo assim, o que será exibido no console é:

* Primeiro será exibido "rejeitada"` e depois "FIM"
* Primeiro será exibido "resolvida" e depois "FIM"
* Primeiro será exibido "FIM" e depois "rejeitada"
* **Primeiro será exibido "FIM" e depois "resolvida"**

**Explicação**

**Relembrando do código para refletirmos sobre ele logo em seguida:**

**function exibe(texto) {  
 return $q(function(resolve, reject) {  
  
 // simulando ação assíncrona com setTimeout  
  
 setTimeout(function() {  
 if(texto == 'Alura') {  
 resolve('resolvida');  
 } else {  
 reject('rejeitada')  
 }  
 }, 5000);  
 });   
}  
  
// executando nossa promise  
  
exibe('Alura').then(function(resultado) {  
 console.log(resultado);  
}).catch(function(erro) {  
 console.log(erro);  
});  
console.log('FIM');**

**A função exibe recebe como parâmetro um texto. Faça uma analogia com $http.get, que recebe como parâmetro uma URL. Muito bem, se o texto passado for igual à Alura, a promise será resolvida com o texto "resolvida", caso contrário rejeitada com o texto "rejeitada" . A mesma coisa acontece com $http.get: quando resolvida, recebemos os dados do servidor, se for rejeitada, recebemos uma mensagem de erro. Por fim, $http.getrealiza uma requisição assíncrona, que não bloqueia a execução de outros scripts em nossa página e simulamos esse assincronismo em nossa função exibe, através de setTimeout. Isto é, quando a função for chamada, nossa promise só será resolvida depois de 5 segundos.**

**Como nossa promise executa assincronamente, o navegador passará e executará a próxima linha, que exibe "FIM" no console. Depois dos cinco segundos terem passados, será exibido no console "resolvida". Sendo assim, a saída será "FIM" seguida de "resolvida". Não faz muito sentido para o usuário, certo? Podemos resolver isso facilmente movendo a instrução que imprime "FIM" para dentro da função then, dessa maneira, temos certeza que ela será executada só depois que a promise for resolvida, e melhor, depois da impressão deresolvida:**

**// executando nossa promise  
  
exibe('Alura').then(function(resultado) {  
 console.log(resultado);  
 console.log('FIM');  
})  
.catch(function(erro) {  
 console.log(erro);  
});**