JavaScript Promises

‹ Anterior Próximo ›

"Eu prometo um resultado!"

"Produzir código" é um código que pode levar algum tempo

"Código de consumo" é o código que deve esperar pelo resultado

Uma promessa é um objeto JavaScript que vincula a produção de código e o consumo de código

Objeto JavaScript Promise

Um objeto JavaScript Promise contém o código de produção e chamadas para o código de consumo:

Sintaxe de promessa

```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Quando o código em execução obtém o resultado, ele deve chamar um dos dois retornos de chamada:

Resultado	Ligar
Sucesso	myResolve(result value)
Error	myReject(error object)

Promise Object Properties

A JavaScript Promise object can be:

- Pending
- Fulfilled
- Rejected

The Promise object supports two properties: **state** and **result**.

While a Promise object is "pending" (working), the result is undefined.

When a Promise object is "fulfilled", the result is a value.

When a Promise object is "rejected", the result is an error object.

myPromise.state	myPromise.result
"pending"	undefined
"fulfilled"	a result value
"rejected"	an error object

You cannot access the Promise properties **state** and **result**.

You must use a Promise method to handle promises.

Promise How To

Here is how to use a Promise:

```
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Promise.then () leva dois argumentos, um retorno de chamada para sucesso e outro para falha.

Ambos são opcionais, portanto, você pode adicionar um retorno de chamada apenas para sucesso ou falha.

Exemplo

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

let myPromise = new Promise(function(myResolve, myReject) {
  let x = 0;

// The producing code (this may take some time)

  if (x == 0) {
    myResolve("OK");
  } else {
    myReject("Error");
  }
});

myPromise.then(
  function(value) {myDisplayer(value);},
  function(error) {myDisplayer(error);}
);
```

Tente você mesmo »

Exemplos de promessa de JavaScript

Para demonstrar o uso de promessas, usaremos os exemplos de retorno de chamada do *capítulo anterior*:

- Esperando por um tempo limite
- Esperando por um Arquivo

Esperando por um tempo limite

Exemplo de uso de retorno de chamada

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
```

Tente você mesmo »

Exemplo usando promessa

```
let myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
```

Tente você mesmo »

Esperando por um arquivo

Exemplo usando retorno de chamada

```
function getFile(myCallback) {
  let req = new XMLHttpRequest();
  req.open('GET', "mycar.html");
  req.onload = function() {
    if (req.status == 200) {
      myCallback(req.responseText);
    } else {
      myCallback("Error: " + req.status);
    }
  }
  req.send();
}

getFile(myDisplayer);
```

Tente você mesmo »

Exemplo usando Promise

```
let myPromise = new Promise(function(myResolve, myReject) {
  let req = new XMLHttpRequest();
  req.open('GET', "mycar.htm");
  req.onload = function() {
    if (req.status == 200) {
      myResolve(req.response);
    } else {
      myReject("File not Found");
    }
  };
  req.send();
});

myPromise.then(
  function(value) {myDisplayer(value);},
  function(error) {myDisplayer(error);}
);
```

Tente você mesmo »

Suporte de navegador

ECMAScript 2015, também conhecido como ES6, introduziu o objeto JavaScript Promise.

A tabela a seguir define a primeira versão do navegador com suporte total para objetos Promise:

Chrome 33	Edge 12	Firefox 29	Safari 7.1	Opera 20
Feb, 2014	Jul, 2015	Apr, 2014	Sep, 2014	Mar, 2014

‹ Anterior Próximo ›