Google
Traduzido para: Português Mostrar original Opções ▼ **Tutorials References** ▼ **Exercises ▼ Paid Courses** Log in G Português CSS **JAVASCRIPT** SQL HTML **PYTHON** PHP **BOOTSTRAP** Parâmetros de Função **PROPAGANDA** Invocação de Função **SIEMENS** Reyrolle 7SR17 - Rho Motor Protection Ingenuity for life Chamada de Função Learn More> Função Aplicar Fechamentos de funções JavaScript Promises Classes JS Introdução à aula Próximo > Anterior Herança de classe JS Async "Eu prometo um resultado!" JS Callbacks JS Assíncrono "Produzir código" é um código que pode levar algum tempo **JS Promises** JS Async / Await "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 Versões JS IC 2000 (ECE) Objeto JavaScript Promise Um objeto JavaScript Promise contém o código de produção e chamadas para o código de consumo: Sintaxe de promessa **COLOR PICKER** let myPromise = new Promise(function(myResolve, myReject) { // "Producing Code" (May take some time) myResolve(); // when successful myReject(); // when error COMO NÓS f o in // "Consuming Code" (Must wait for a fulfilled Promise) myPromise.then(function(value) { /* code if successful */ }, function(error) { /* code if some error */ } Obtenha a certificação completando Quando o código em execução obtém o resultado, ele deve chamar um dos dois retornos de chamada: um curso hoje! Resultado Ligar myResolve(result value) Sucesso myReject(error object) Error iniciar Promise Object Properties A JavaScript Promise object can be: JOGO DE Pending CÓDIGOS 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 Jogar um jogo "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) { // 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; });

Class Static

Versões JS

ANY SANY

```
Tente você mesmo "
Esperando por um arquivo
Exemplo usando retorno de chamada
```

if (req.status == 200) { myCallback(req.responseText); } else { myCallback("Error: " + req.status);

function getFile(myCallback) {

req.onload = function() {

let req = new XMLHttpRequest();

req.open('GET', "mycar.html");

```
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.

Feb, 2014	Jul, 2015	Apr, 2014	Sep, 2014	Mar, 2014
Anterior				Próximo >

Firefox 29

Safari 7.1

Opera 20

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

Edge 12

Chrome 33

Fórum Reportar erro Cerca de Comprar **Top References Top Examples** Web Courses Principais tutoriais HTML Tutorial HTML Reference HTML Examples HTML Course **CSS Tutorial** CSS Reference CSS Examples CSS Course JavaScript Tutorial JavaScript Reference JavaScript Examples JavaScript Course How To Tutorial SQL Reference How To Examples Front End Course SQL Tutorial Python Reference SQL Examples SQL Course Python Tutorial W3.CSS Reference Python Examples Python Course W3.CSS Tutorial Bootstrap Reference W3.CSS Examples PHP Course **Bootstrap Tutorial** PHP Reference Bootstrap Examples jQuery Course PHP Tutorial HTML Colors PHP Examples Java Course Java Tutorial Java Reference Java Examples C++ Course C++ Tutorial Angular Reference XML Examples C# Course jQuery Tutorial jQuery Reference jQuery Examples XML Course Get Certified » W3Schools é otimizado para aprendizagem e treinamento. Os exemplos podem ser simplificados para melhorar a leitura e o aprendizado. Tutoriais, referências e exemplos são constantemente revisados para evitar erros, mas não podemos garantir a correção total de todo o conteúdo. Ao usar o W3Schools, você concorda em ter lido e aceito nossos termos de uso, cookies e política de privacidade.