PROPAGANDA NIVEA Propriedades do objeto JavaScript Próximo > Anterior Propriedades são a parte mais importante de qualquer objeto JavaScript. **ESTÁ DE VOLTA!** SABE MAIS Propriedades JavaScript Propriedades do objeto Propriedades são os valores associados a um objeto JavaScript. Um objeto JavaScript é uma coleção de propriedades não ordenadas. Construtores de objetos Geralmente, as propriedades podem ser alteradas, adicionadas e excluídas, mas algumas são somente leitura. Sente-te lindo Acessando propriedades de JavaScript A sintaxe para acessar a propriedade de um objeto é: **COLOR PICKER** objectName.property // person.age ou objectName["property"] // person["age"] COMO NÓS ou f o in objectName[expression] // x = "age"; person[x] Obtenha a A expressão deve ser avaliada como um nome de propriedade. certificação completando um curso hoje! Exemplo 1 person.firstname + " is " + person.age + " years old."; Try it Yourself » iniciar Example 2 JOGO DE person["firstname"] + " is " + person["age"] + " years old."; CÓDIGOS Try it Yourself » JavaScript for...in Loop The JavaScript for...in statement loops through the properties of an object. Syntax Jogar um jogo for (let variable in object) { // code to be executed The block of code inside of the for...in loop will be executed once for each property. Looping through the properties of an object: Example const person = { fname:" John", lname:" Doe", age: 25 **}**; for (let x in person) { txt += person[x]; Try it Yourself » Adding New Properties You can add new properties to an existing object by simply giving it a value. Assume that the person object already exists - you can then give it new properties: Example person.nationality = "English"; Try it Yourself » **Deleting Properties** The delete keyword deletes a property from an object: Example const person = { firstName: "John", lastName: "Doe", age: 50, eyeColor: "blue" **}**; delete person.age; Try it Yourself » or delete person["age"]; Example const person = { firstName: "John", lastName: "Doe", age: 50, eyeColor: "blue" **}**; delete person["age"]; Try it Yourself » The delete keyword deletes both the value of the property and the property itself. After deletion, the property cannot be used before it is added back again. The delete operator is designed to be used on object properties. It has no effect on variables or functions. The delete operator should not be used on predefined JavaScript object properties. It can crash your application. **Nested Objects** Values in an object can be another object: Example $myObj = {$ name:"John", age:30, cars: { car1: "Ford", car2:"BMW", car3:"Fiat" You can access nested objects using the dot notation or the bracket notation: Example myObj.cars.car2; Try it Yourself » or: Example myObj.cars["car2"]; Try it Yourself » or: Example myObj["cars"]["car2"]; Try it Yourself » or: Example let p1 = "cars"; let p2 = "car2"; myObj[p1][p2]; Try it Yourself » Nested Arrays and Objects Values in objects can be arrays, and values in arrays can be objects: Example const myObj = { name: "John", age: 30, cars: [{name:"Ford", "models":["Fiesta", "Focus", "Mustang"]}, {name:"BMW", "models":["320", "X3", "X5"]}, {name:"Fiat", "models":["500", "Panda"]} To access arrays inside arrays, use a for-in loop for each array: Example for (let i in myObj.cars) { x += "<h1>" + myObj.cars[i].name + "</h1>"; for (let j in myObj.cars[i].models) { x += myObj.cars[i].models[j]; Try it Yourself » Property Attributes All properties have a name. In addition they also have a value. The value is one of the property's attributes. Other attributes are: enumerable, configurable, and writable. Esses atributos definem como a propriedade pode ser acessada (é legível ?, é gravável?) Em JavaScript, todos os atributos podem ser lidos, mas apenas o atributo de valor pode ser alterado (e apenas se a propriedade for gravável). (ECMAScript 5 tem métodos para obter e definir todos os atributos de propriedade) Propriedades do protótipo Os objetos JavaScript herdam as propriedades de seu protótipo. A delete palavra-chave não exclui propriedades herdadas, mas se você excluir uma propriedade de protótipo, ela afetará todos os objetos herdados do protótipo. Próximo > Anterior Fórum Cerca de Reportar erro Comprar Principais tutoriais **Top References** Top Examples Web Courses **HTML Tutorial** HTML Reference HTML Examples **HTML Course CSS Tutorial CSS Reference** CSS Examples CSS Course JavaScript Examples JavaScript Course JavaScript Tutorial JavaScript Reference How To Tutorial SQL Reference How To Examples Front End Course SQL Course SQL Examples SQL Tutorial Python Reference Python Tutorial W3.CSS Reference Python Examples Python Course W3.CSS Examples PHP Course W3.CSS Tutorial Bootstrap Reference PHP Reference **Bootstrap Examples Bootstrap Tutorial** jQuery Course PHP Tutorial HTML Colors PHP Examples Java Course Java Tutorial Java Reference C++ Course Java Examples C++ Tutorial XML Examples Angular Reference C# Course jQuery Examples XML Course jQuery Tutorial jQuery Reference 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 . Copyright 1999-2021 de Refsnes Data. Todos os direitos reservados. W3Schools é desenvolvido por W3.CSS. schools

Google
Traduzido para: Português

Tutorials ▼

HTML

JS esta palavra-chave

Função de seta JS

Classes JS

Depuração JS

Guia de estilo JS

JS Best Practices

JS Performance

Objetos JS

Palavras Reservadas JS

Definições de Objeto

Métodos de Objeto

Exibição de objeto

Acessores de objeto

Protótinos de objeto

JS JSON

JS Erros

Modo JS Strict

CSS

Mostrar original

SQL

Exercises ▼

PYTHON

PHP

BOOTSTRAP

References ▼

JAVASCRIPT

Opções ▼

Log in

Paid Courses

G Português