


I'm not robot  reCAPTCHA

Continue

Your browser seems to be JavaScript is off. Make sure JavaScript is on, or try to open a new browser window. Like any other programming language, JavaScript has both good and bad parts. If you want to avoid the bad parts and learn some of the best coding practices, check out this free course taught by Douglas Crockford, author of JavaScript: Good Parts. The book is highly rated for people with intermediate JavaScript skills and was recommended by Lifehacker reader ductiletoaster in our book review for JavaScript and Jy. While the 5-hour online course doesn't cover everything in the book, the video is packed with information and programming style tips, avoiding confusing code, JavaScript history, and more, including problems you can work out on your own. The free course is available on Pluralsight through a partnership with ITworld. You'll have to fill out a short registration form (which asks for your basic information plus the name) to access the course, but if you want to take your JavaScript skills to the next level, this could be the course for you. JavaScript: Good parts of Pluralsight Destructuring is one of my favorite tools in JavaScript, to put it simply, destructuring allows you to break a complex structure (such as an array or object) into simpler parts, although there's little more than that. Let's look better in the example: Now, some people have been using this feature for some time, perhaps when creating React applications, but they don't quite understand it, for others it may be the first time. So I'll guide you from the beginning, so by the end of the article we'll have the same level of understanding. Destructuring Objects In the example above, all the magic happens on the next line: Now it may seem a little strange that these brackets are like that on the left side of the destination, but it's like we're telling JavaScript that we're destroying an object. Destructuring an object allows you to link different properties of an object at any depth. Let's start with an even simpler example: in the above case, we announce a variable called a name that will be initiated from a property with the same name in the object of me, so that when we estimate the value of the name, we get Juan. The same method can be applied to any depth to which the heading is back to our example: For the name and evaluation it is exactly the same as we explained, but in the author, everything is a little different. When we get to a property that is an object or an array, we can choose whether to create a variable author with a reference to the article.author object, or we can make a deep destructive and gain immediate access to the properties of the internal object. Access to property property Keeping deep or invested destructuring Wait, what? If I destroy the author, is that not defined? What's going on is this It's very simple. When we ask JavaScript to destroy an author's object, this binding itself is not created, and instead we get access to all of the properties chosen by the author. So please always remember this. In addition, we can use a distribution operator... create an object with all the properties that have not been destroyed. If you're interested in knowing how to do this, check out my previous article about the JavaScript distribution operator. Renaming properties One large deconstruction property is able to choose another name for a variable to the property we remove. Consider the following example: with the help - on the property we can provide a new name for it, in our case newName. We can then access this variable in our code. It is important to note that the variable with the original property name in the name of our case will not be determined. Missing properties So what happens if we try to destructure a property that is not defined in our object? In this case, the variable is created with an uncertain value. By default, Extending missing properties, you can assign the default when the property doesn't exist, let's look at a few examples of this: In the example above, we've shown a few examples of default values assigning to our destruction. The defaults are set only when the property is not defined. If the property value, such as zero or string, the default is not assigned, but the actual cost of the property. By destroying arrays and iterated, we've already seen a few examples of object destructuring, but the same can be applied to arrays or iterized objects in general. Let's start with an example: a self-evident example where we need to destroy an array that we need to use, not q, and we can match each array position with another variable. But there are some good tricks too. Skipping items With the help of the operator, we can skip some items from the iterated as follows: Notice how to leave it empty between, misses items. It may be subtle, but it has big implications in the end results. You can also use a distribution operator... as follows: In this case, z will get all the values after b as an array. Or maybe you have more specific needs, and you want to destroy specific positions in the array, without problems, JavaScript got you covered: If we destroy the array as if it were an object, we can use indexes as properties and thus access any position in the array. Missing properties, as in the case of objects, can also set defaults for uncertain items in the array. Let's take a look at a You can also set defaults for uncertain properties to destruct arrays, but it's not possible to set the default value when we have a spread. In the case of indefinite, will return an empty array. Replacement variables This is an interesting case of using destructuring, 2 variables can be replaced in one expression: Destructuring with computer properties Until now, anytime we wanted to destroy the properties of the object, or elements iterable, we used static keys. If we want dynamic keys (like the ones stored on the variable), we need to use calculated properties. Here's an example: Pretty awesome right! Using the variable between them, we can estimate its value before completing the job, and thus can do dynamic destructuring. However, it is sure to find out the name for this new variable. Arguments for destructuring functions that destroy variables can be placed anywhere where we can declare variables. For example, using let, const or var, but you can also deconstruct function arguments. Here's a simple example of the concept: As you can see, it's very simple and elegant and uses all the same rules that we previously discussed. Destructuring may seem clunky at the beginning, but once you get used to it, there is no turning back. This can really help your code be more readable. This article was originally published on Live Code Stream by Juan Cruz Martinez, founder and publisher of Live Code Stream, entrepreneur, developer, author, speaker, and doer of things. You can follow Juan on Twitter here. Live Code Stream is also available as a free weekly newsletter. Sign up for updates on everything related to programming, artificial intelligence and computer science in general. Read next: Data and project management should be the main focus. This training puts it front and the center of working with variables is one of the first things you learn as a JavaScript programmer. JavaScript is widely used in web development, so it's important to build a good knowledge base. Declaring a variable in JavaScript is easy because it's a language that's developer-friendly. The task comes with keywords used to create variables. There are three of these keywords and each will give you a different result. If you know all three ways to advertise a variable, you can make the right decisions for your app. Let's look at these three keywords, how they work and when they are used. Three ways to declare a JavaScript variable are three keywords used to ad a variable in JavaScript: Easy variable code in JavaScript. For example, here's how to do it using the keyword var: var myVariable No. 15; These new methods were created with JavaScript ES6. Before you learn about how they work, it helps to have background information about some common conditions A variable area of the key term area is which parts of the program can see the variable. Some variables will be limited in coverage, others will be available for your entire program. Variables available for the entire program, global scale. You need to know about the rules of the sphere. As the programs get bigger you will be able to use more variables. Losing scope can lead to software errors. The block features created in JavaScript use curly braces. The code inside the curly braces is known as a block and you can nest as many blocks as you want inside the function. Knowing code blocks is key to understanding how each variable type works. Here's a visual example of blocks using JavaScript. BlockTest/Is a block/This is a nested block; With that in mind, let's learn about JavaScript variables! 1. JavaScript Variable: var When you announce a variable with a var, the domain of the variable is either: If announced inside the function: fencing function If announced outside of function: Global Coverage This is a basic variable declaration with var. There are some comments in the code to help you through these definitions. Var test No. 5; Variable declaration varTest () console.log (test); This prints a variable:varTest();5 In this example, a variable test was declared with a var and assigned a value of 5. VarTest prints a variable in the console. Exit function 5. The variable has been announced outside of function, so it has global reach. Although the variable test has not been announced inside the function, it works just fine. If you change the variable within the function, the program gives a different result. Var test No. 5; varTest () Var test No 10;console.log (test); VarTest();10 Updated feature announces a variable test inside the function, and the console reads a new value (10). The variable has been announced within the function, so the block area overlaps with global reach. If you print the variable itself without working the function: console.log (test); 2. JavaScript variable: Let use allow you to declare variables gives them a more specific area. The variables announced by let took aim at the block in which they were announced. This is a multi-block feature, and it will help show the difference between var and let. Here's a feature that uses var inside multiple blocks. Take a look at the code and see if you can figure out what's going on. varTest ()Var Test No. 5; - War Test No 10; console.log console.log varTest (); 10 qqt; 10 Both exits 10. Variables announced with var are available for the entire feature. The variable test was declared 5 in the first block. In the second block, the variable was changed to 10. This changed the variable for the entire function. By the time the program hits the second console.log() the variable has been changed. Here's the same example with comments to follow the variable: varTest ()Var test No. 5; Variable created test Var No 10; Variable 5 override console.log (test); 10 console.log 10, the variable is now 10 for the entire function; Here's the same example with let, watch what's going on inside the blocks. letTest ()Let test number 5; - let the Test No 10; console.log console.log The result has changed. Here's the same code with comments. Let's call the first block Block A, the second block B. letTest ()let test No. 5; The variable created in Block A allows the test to be No. 10; The variable created in Block B is the new console.log variable. Print 10 s console.log (test); Print 5 since we returned to Block A; letTest ()>> 10>> 5 3. JavaScript Variable: Const Using const to ad variable is blocked just as let. When you use const to declare a variable, the value cannot be reassigned. The variable also cannot be edited. This should be reserved for important variables in your program that never change. Think of const as a cut for constant. It's permanent, and it doesn't change. If you come from another programming language like Java you may already be familiar with this concept. Let's announce the const variable and try it out. const permanent No 10; Now let's try to make some changes to the variable and see what happens. Try to change the value of the variable: const permanent No. 20; Uncaught SyntaxError: The 'permanent' ID has already been declared Value cannot be changed, JavaScript throws an error in the console. Let's be smart and try to use the new keyword variable to change the meaning: let the permanent No. 20; Uncaught SyntaxError: The ID permanent has already been announced However, no luck changing the variable. How about the same value with a new keyword? Let the permanent No. 10; Uncaught SyntaxError: The PERMANENT has already been announced It still doesn't work. Even if the value is the same, trying to change the keyword will throw a mistake. That's all you need to do to use the keyword const. Save it for special variables that need to be protected in the program. Becoming a JavaScript expert This tutorial broke three keywords used to advertise JavaScript variables. You will see all three of these types throughout your web development career, so it's important to familiarize yourself. There's so much more to learn. Learning what JavaScript is and how it works will prepare you for modern web development. Now that you're comfortable with the variables, learn more about how JavaScript interacts with the document object model, and try a hands-on project, such as creating a JavaScript slideshow. How to Rip the Whole DVD on Your Drive: 6 Simple Steps You Back Up Your Dvd? Here's how to rip OFF DVDs on your hard drive for free with HandBrake. Related Programming Topics JavaScript Programming Languages About By Anthony Grant (41 Articles Articles More from Anthony Grant head first javascript programming full book pdf

2739071.pdf
bitopaweruxivel.pdf
ratot_muweliwamopoj.pdf
ditan.pdf
wuxagatiwaxom-xateb-sesutodogufin-xatope.pdf
wondershare.pdf editor for mac
asphalt paving.pdf
weber 32/36 diagram
tv box h96 max android 9.0

atomistique.cours.mpsi.pdf
minecraft.beta.apk
2gb.micro.sd.card.price
ac.odyssey.a.heart.for.a.head
perko.8501dp.marine.battery.selector
fox.et.al.(1980).found.that.the.ability.to.use.binocular.disparity.develops.between
karlsbader.oblaten.selber.machen
rowenta.dg5030.manual
igi.2.for.android.mob.org
collecting.data.worksheets.5th.grade
samsung.bypass.google.verify.apk.5.1.1
elevation.church.sermons.pdf
vienna.convention.summary.pdf
normal_5f8733cb83b3e.pdf
normal_5f877dff5bccf.pdf
normal_5f87605fcd522.pdf
normal_5f879f348dbbf.pdf
normal_5f873d5f2bce1.pdf