* - [Narrator] In the MDN web docs, VAR is defined as a globally scoped variable.



* That means if you first assign a value to have VAR, and then inside a function, assign it a new value, that new value inside the function takes effect in all corresponding code.
* Even outside the function.
* This can cause what's known as scope issues, which result in unintended behavior errors, and a lot of frustration, as we try to figure out, why things aren't working properly.
* Let me show you a practical example of how this works.
* In the exercise files, I've set up these two boxes, ah, the color and the text in here, is populated using JavaScript.
* Looking at the script, it's straightforward, We set up a VAR, with the name color, and then we populated with a color in this click case, purple.
* This is a CSS color, which is why it works.
* You can also put in a hex value, if you want it to here, and it'll work the same way.
* Then we use query selector, to find the left, and right boxes.



* And we populate the style property and the background property with the color, and also the inner HTML, with the same color value.
* So we can see that on the front end.

Graphical user interface, text, application

Description automatically generated

* All right, knowing this, I also see, if I reassign the color variable, down here, to something else, sky blue for instance, that new color will apply only to the right box.
* Because, JavaScript is read from the top down, right? So for the first two decorations here, we get purple, for the two second ones, we get sky blue.
* Save that, check it in the browser.

Graphical user interface, application

Description automatically generated

* Look, now we have two different colors.
* That's cool.
* Now we can continue, down here at the bottom.
* I've also set up a function and I'll uncommand it.
* This function, called heading color, sets the color variable to blue.
* Then it goes and finds the title, and sets the text color, to the value of color.
* So if we now save this, and go to the browser, we get a blue heading, a purple box, and a sky blue box.
* All right.
* So far everything is working properly.
* Now we get to the **scoping** issue.
* Watch what happens if I take this function, and place it directly after we declare color sky blue.
* Now, looking at this, you would think, that because we are declaring color blue, inside this function, it would only apply inside the function.
* But if I save this, and go to the front end, you see it actually doesn't.
* It also affects the right box.
* That's because we're using a VAR, and a VAR is globally scoped.
* So even though we are changing the value of the VAR, inside a function, that new assignment, applies everywhere in the document, after this instance.
* So, JavaScript reads from the top down, says, color is purple.
* Now color is sky blue.
* Now color is blue.
* And that therefore, at the bottom here we get blue.
* Now, interestingly, this only happens, if you have the call to the function, over this item.
* So if I cut this out and paste it in after, you'll see the function is only invoked, after we've already set the color of the right box.
* And now we get this light blue instead.
* This a scope issue.
* It becomes really hard to understand why, sometimes the color is what it's supposed to be, and other times it's not.
* And looking at the code here, it's not immediately obvious what's happening.
* This issue has been a pain in the side of JavaScript developers, since the very beginning.
* And the result of it is that we've had to do things like declare custom variables inside functions that only exist inside those functions, and then pass variables in and out to make everything work properly, and it's gotten really cluttered.
* For this reason, we now have **two new types of variables**, a Let, and a Scope.
* and they are **both locally scoped variables**, to avoid this problem altogether.