* (upbeat music) - [Instructor] In this challenge, I showed you a JavaScript application that had just been converted to TypeScript but lacked any kind of additional type information.
* And I asked you to give it that type information.
* Here's how I solved this challenge.
* First, I asked you to create an interface to describe the structure of this variable on lines one through five.
* The way I would approach this is, first, come up with a name.
* What are we looking at here? It's an array of to-do items.
* So I'm going to call my interface something like to-do item or just to-do.
* Next, I'm going to take a look at what is similar and what is different in these items.
* They all have an ID that's a number, a title that's a string, and a status that's also a string.
* Also, one of them has a completed on property, which is a date value, however, this is not always present on all to-do items.
* So I'll make this one optional.
* Then, once I have my type, I can apply it to my variable.
* Next, I challenged you to strongly type the to-do status values using an enum to make sure they're consistent everywhere.
* First, I'll create a new enum type named to-do status.
* Then I look at the different statuses that are being used.
* It looks like there are three to-do in progress and done, so I add them to my enum.
* But, uh oh, if I do it this way, I'll break the in dash progress value since the original value has a hyphen in it but my new enum value does not.
* So I'll give it a value.
* In fact, I'll give all of my enums better looking names and explicit values.

Graphical user interface, text, application, email

Description automatically generated

* Now that I have my fancy new enum, I can use it as the type of my status property.

Graphical user interface, text, application

Description automatically generated

* And then I'll update all of my existing references.
* Next, I challenged you to apply types to the parameters and return values of the two functions in this code.
* This part is simple.
* Since we've already created the to-do interface, we can use basic type syntax for the parameters and the return type.

Graphical user interface, text, application

Description automatically generated

* From looking at the implementation, I can see that this function accepts a string parameter and returns the newly added to-do item.
* Finally, I challenged you to apply generic types to the get next ID function.
* I'll start simply by defining a generic type T for the items parameter.
* But since this function expects an array of items, the parameter type will actually be an array of T.
* And the function always returns a numeric value no matter what T may be.

Graphical user interface, text, application

Description automatically generated

* Now that I've done this, I can see these new errors are telling me that a simple, generic parameter like this one won't be enough.
* I have to tell TypeScript about the ID property that I'm using in this type, so I'll add a generic constraint that says that any one of these parameters must be an object with a property called ID of type number.
* And that's it.
* We have fully typed this application.
* Now there's no one single way to do any of this, so your solution may have varied a decent amount from what I have here, but as long as you were able to get all of these typings in place, then your solution is just fine.
* Also, I've included my solution in the exercise files, so if you have any problems or just want to see my completed code, feel free to check it out.