* - [Instructor] At this point, I've now shown you how to add TypeScript to your application and compile TypeScript files.
* However, one of my favorite things about TypeScript is that it is very easy to add to an existing application with existing JavaScript code.
* This is because TypeScript is a super set of JavaScript, which means ***that all JavaScript*** is valid TypeScript.
* If you open the exercise folder for this video, you'll see that I've added an example JavaScript file, app.
* js.
* To enable checking of existing JavaScript files, I'll simply add two additional settings to my tsconfig.
* json file.
* allowJs.
* And checkJs.

Graphical user interface, text

Description automatically generated

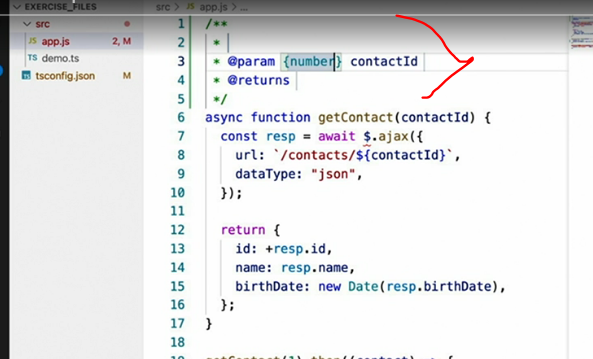
* And I'll set both of these options to true.
* Now, I'll run the TypeScript compiler and see
* Graphical user interface, text, application

  Description automatically generatedwhat happens.

Graphical user interface, text

Description automatically generated

* TypeScript found three errors in this new JavaScript file, errors that I planted in order to show how TypeScript can help you, even with JavaScript.
* I'm actually going to skip this first error: ***Cannot find dollar sign*** because it means that *TypeScript doesn't know anything about the jQuery library* I'm using and I'll show you how to fix that in the next video.
* The next two errors though show how TypeScript is able to look at my JavaScript code and see that I'm trying to reassign two properties to values that are a different type than they were previously assigned to.
* To see how it came to this conclusion, let's follow the code.
* It starts on line 14 where I call the getContact method, which is defined on line one.
* And TypeScript can tell that this method returns an object with three properties: id, name and birthDate.
* And even more, it can tell that the id property is a number and the birthDate property is a date value.
* So when I go to reassign those properties to other types on lines 15 and 16, TypeScript lets me know that this is probably a mistake.
* Now, when I change them to the correct types, and run the compiler again, those errors go away.
* Now that I fixed some of the errors that were obvious to TypeScript, just based on its ability to infer type information, let's add some more information to allow it to detect even more issues and no, I don't mean converting this file to TypeScript by renaming it to .
* ts, I mean adding types to the JavaScript code using the industry standard JSDoc syntax.
* The basic JSDoc syntax looks like this.



* True to its name, it's just a comment block but with some special syntax identified by the @ symbol.
* In this case, Visual Studio Code's auto complete has actually looked at my code and determined that this function that I'm documenting has a parameter named contactId and has defined its type in the braces as star, meaning it could be anything.
* However, if I change the star to number, which is what type I expect this parameter to be, I can now hover over the parameter and see that TypeScript has determined that this is indeed a number value.
* What's more, if we look down on line 24, I can also see that TypeScript has found another error in my code.
* I'm trying to locate a contact using a string value rather than a number value for the contactId.
* Well, that's not correct.
* So I'll fix it.
* And I'll see that the error goes away.
* And that's just one of the dozen or so JSDoc keywords that you can use to tell TypeScript about your JavaScript code without actually converting your JavaScript files to TypeScript.
* In fact, you can mark up just about all of your JavaScript code like this.
* However, if you're going to go that route, I would just recommend taking the time to fully switch to TypeScript instead.