* - So far in this chapter, I showed you how to use the primitive and built in types that JavaScript offers, as well as how to define your own custom types using interfaces.
* One other way of referring to types and defining your own custom types is called **a type alias.**
* As its name implies, *a type alias is simply providing an alias or alternate name for an already existing type.*
* You define a type alias using the *type keyword followed by the name of the alias.*
* Let's call this one ContactName.
* Then follow that with an equal sign and the type you wish to alias, in this case string.
* Now that I've defined the type alias, I can use it in any place that I would use the original type.
* For example, I can refactor the contact interface example from the previous video, changing the type of the name field from string to ContactName.

Graphical user interface, text, application

Description automatically generated

* Note however, the type aliases are truly just that, an alias for another type.
* *They are not a new type themselves, which means that they can be used interchangeably with the type that they alias.*
* For example, even though I've now defined the name field of contact interface as the type ContactName, I can still assign it to a regular old string value.
* So what is the value of type aliases you may be wondering? Well, even when used to give another name to a primitive type, as in this example, I think they add a little more meaning to the field or variable that they're describing.
* For instance, in this case, it's not just a string value, but a contact name value.
* And if I wanted to change all references of contact names I could simply change the type that the ContactName alias refers to.
* If that justification doesn't satisfy you, however, I don't blame you, but hold out judgment for just a little while, because later in this course, I will show you some more advanced uses of type alias that will probably have you changing your mind.