* Instructor: Transforming data is of course something we could also do here inside of subscribe, and that would generally not be a problem .
* But it is a good practice to **use observable operators** because it simply allows us to write cleaner code, with different steps we funnel our data through, that can easily be swapped or adjusted, so that you have a lean subscribe function here, and have our steps that focus on our parts .
* Therefore, here before we subscribe we can now call pipe, because **pipe**, as you learned is a method **that allows you to funnel your, observable data through multiple operators before they reach the subscribe method** .
* Now the **operator I need here is the map operator**, and therefore I will import map from RX JS slash operators .
* The map operator allows us to get some data, and return new data, which is then automatically re-wrapped into an observable so that we can still subscribe to it, if it would not be wrapped into an observable again, we could not subscribe .
* So let's map here, let's add map as an argument to pipe, and map is a function which we call, and that function now again, takes another function as an input, a function, which will get our response data, and that should now return the converted response data .
* And here the idea is that we return an array of posts instead of an object with that cryptic key which then holds our actual post .
* Now to convert a JavaScript object, which we have here, to an array, we have to manually loop through all the keys and create a new array .
* So here I'll create a new post array constant, which is empty initially, and then I'll use a for-in loop, to go through all my keys in response data which I know will be an object .
* And then I want to push each piece of data into my post array .

Text

Description automatically generated

* So here I will then use post array and push response data key, so I'm accessing the key we're currently looking at in response data, and therefore, I'm accessing the nested JavaScript object and I'm pushing that nested JavaScript object into my post array .

Graphical user interface, text, application, email

Description automatically generated

* However, I don't just want to push that object like this in there .
* Instead I want to push a new object in there, so let me add curly braces and let's use the spread operator now, this will pull out all the key value pairs of that nested object we're accessing here .
* And let's close this with curly braces too, because this now allows me to also add one new key value pair to that object we're adding to post array .

Text

Description automatically generated

* And that should be an ID field, which actually stores the key because that key here, that cryptic string is a perfect ID and it is a unique ID generated by Firebase .

Graphical user interface, text, application

Description automatically generated

* Therefore, of course I want to keep that so that we have this, for example, if you wanted to delete a single post or anything like that .
* So now we're pushing this new object into our post right here .
* Now it is a good practice to wrap this with a if statement if you're using a four in loop where you check if response data has key as it's own properties so that you're not trying to access the property of some prototype .
* Text

  Description automatically generated
* And with that, posts here now really is an array of posts .
* And now we only need to make sure that we return posts array here in map so that this is now forwarded to our subscribe function with data that now reloads, indeed, here we see we have an array, we have square brackets, and in there we have an element which has a content .

Text

Description automatically generated

* The ID will be added and the title .

Graphical user interface, text, application

Description automatically generated

* And this is how we use observable operators to transform our data .
* Learning reminders