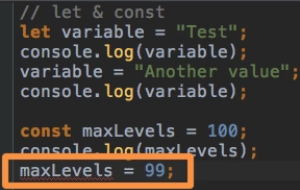
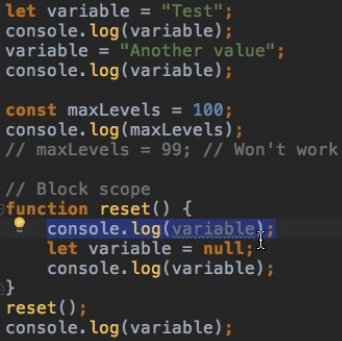
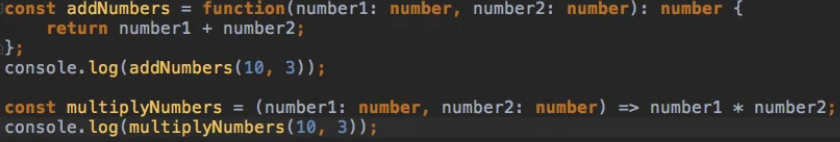
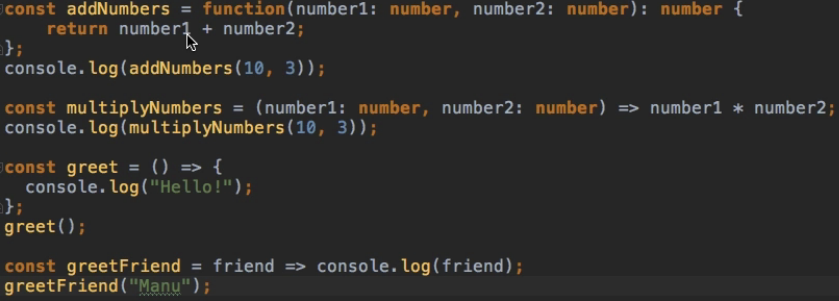
**Introduction**  
\* **TypeScript gives you access to a lot of the ES6 features**.  
\* **Your code, since it will compile to ES5, will run in all browsers even though you’re using ES6**.  
\* **Keep in mind that TypeScript doesn’t support all features of ES6 yet, to find out which features are supported, have a look at this compatibility chart**.  
<http://kangax.github.io/compat-table/es6/>  
\* Here you get this TypeScript row and you can learn which features of ES6 you may safely use with TypeScript.

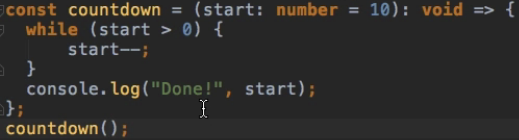
**ES6 & TypeScript Compatibility**  
\* Not necessarily all ES6 Features are supported by TypeScript, but quite a lot of them are.   
\* The following Compatibly Chart helps you find out if you're favorite Feature is supported:  
<http://kangax.github.io/compat-table/es6/>

**“Let” and “Const”**  
\* **The difference between var and let is the scope of the variable**.  
**let** => **creates a block scope variable**.  
=> It’s contained in the block in which you define this variable.  
**var** => **creates a global scoped variable which then sits on the global scope and doesn’t have a block scope**.  
  
  
**const** => **allows us to create immutable variables**.

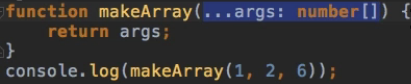
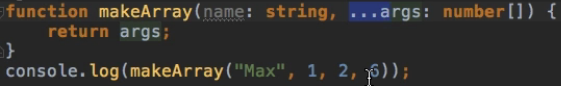
**Block Scope**  
\* They have the same name but technically they’re 2 different variables.  
  
\* **The reset() funciton logs null, the console.log outside the function logs “Another value”**.  
\* **Trying to access the variable inside of the function before declaring it gives undefined because we don’t have access to the variable outside of the function, we would have to pass it as a paremeter**.  
\* **That’s the behavior you probably know from a lot of other languages and now we finally have this with JavaScript and TypeScript as well**.

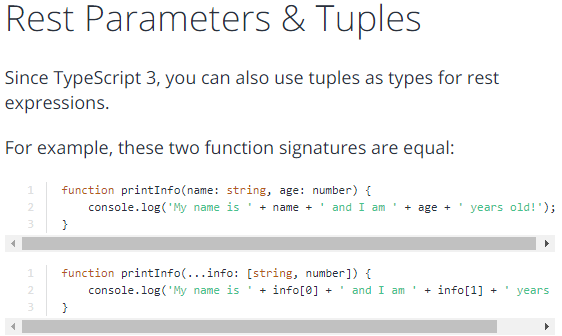
**Arrow Functions**  


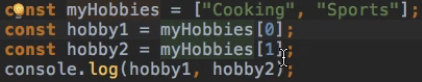
**Arrow Functions - Variations**  


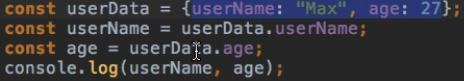
**Functions and Default Parameters**  
\* **Default Parameters**  
  
  
\* **You can do something like this**:  
  
\* But you can’t do:  
  
=> **So the order matters**.

**The Spread Operator**  
\* **Rest and Spread operators allow you to work with arrays and lists of data**.  
\* **An array is kind of an object where you store multiple values**.  
\* **A list of values is just a list of values**.  
**Array**  
  
**List of values**  
  
\* What if we wanted to use Math.max() for the array?  
**Spread Operator**  
  
=> **It spreads out the array into single values**.

**The Rest Operator**  
\* Let’s expect a list of numbers.  
\* If we use … in a function call, then it will use the Spread Operator.  
\* If we use … in a function creation, then it turns whatever values we pass into an array.  
  
  
  
  
\* **If you want to use more parameters, you have to use them before it because everything else will become the array**.  


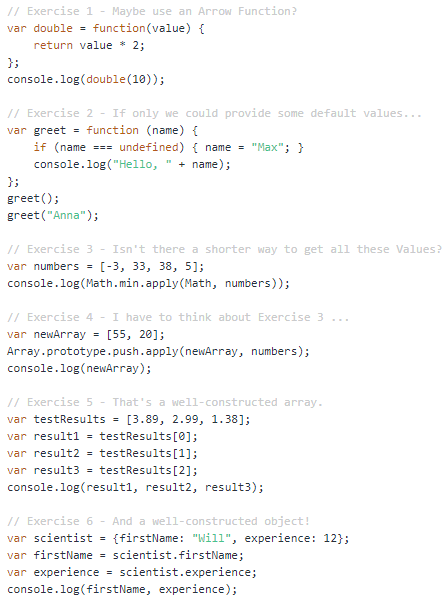
**Rest Parameters & Tuples**  


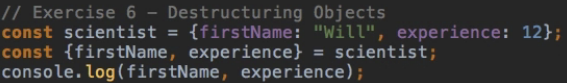
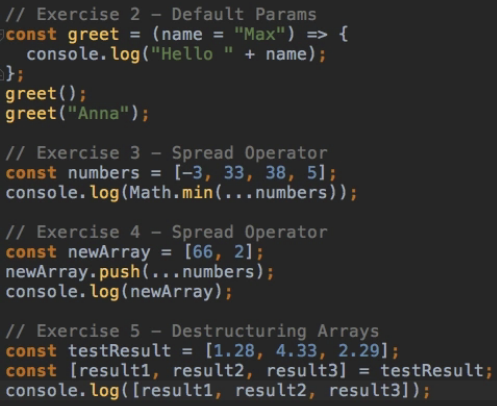
**Destructuring Arrays**  
\* After Rest and Spread, there are 2 other important changes I want to talk about in more detail.  
  
\* **We have a shorter syntax available for this**.  
**Destructuring**  
  
\* **It does the same thing as before**.

**Destructuring Objects**  
**Destructuring**  
  
\* **You can use different names**:  


**Template Literals**  
\* **You can break it over multiple lines if you want**.  
\* **${} to output a variable inside the Template Literal ``.**  
  


**Other ES6 Features**  
\* **Symbols**  
\* **Iterators**  
\* **Generators**  
\* These are more advanced concepts and they have nothing to do with TypeScript specifically.  
\* They’re probably something you don’t need to use every day.

**Module Exercise: Problem**  
\* Re-write the below Code using the ES6 Features you learned throughout this Module.  


**Module Exercise: Solution**  
  
  
**Module Summary**

**Resources**  
How to configure tsconfig.json  
<http://www.typescriptlang.org/docs/handbook/tsconfig-json.html>  
Compiler options  
<http://www.typescriptlang.org/docs/handbook/compiler-options.html>