Lab: New ES6 Features

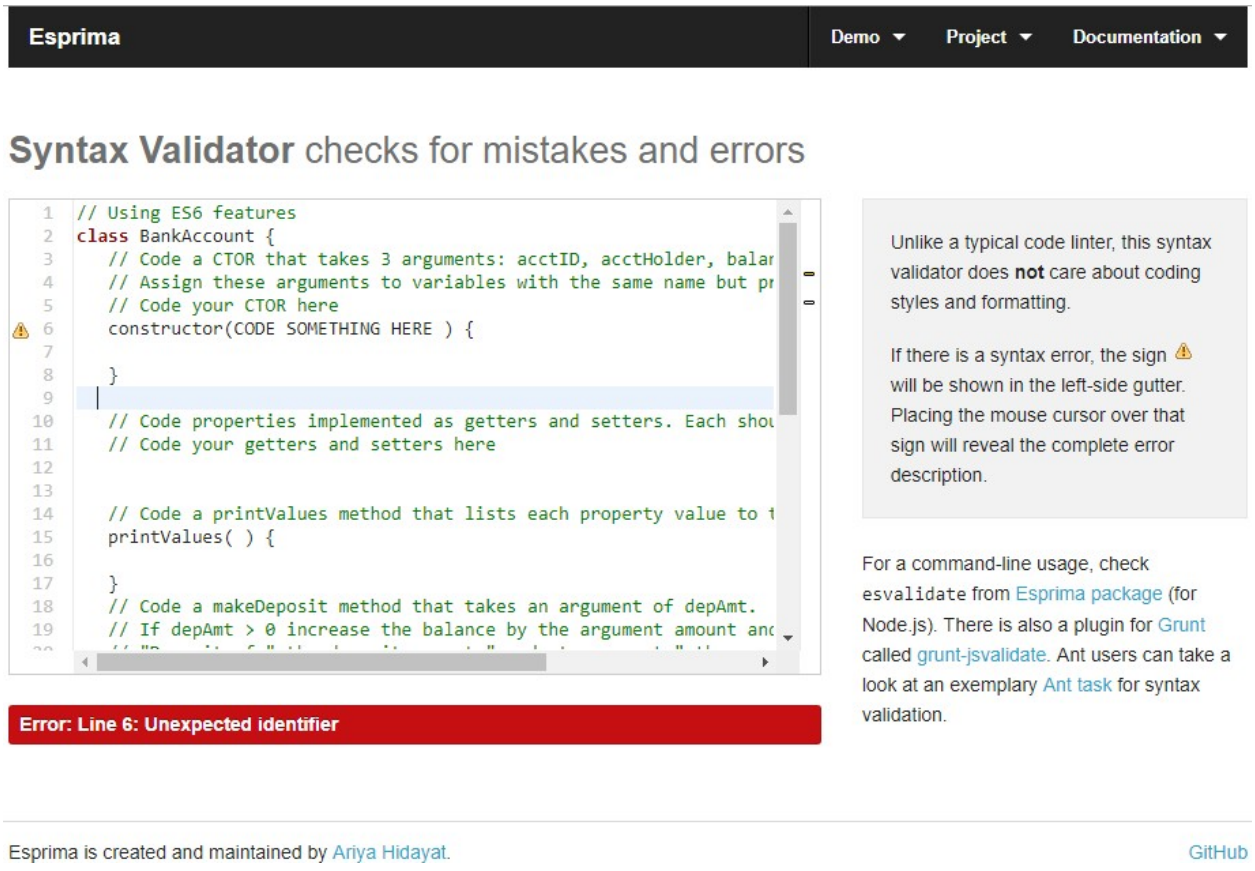
In this lab, you'll use some new ES6 features. You'll code classes the ES6 way and use *arrow functions/expressions* among other things.

Part 1: Using ES6 Classes

Here you'll create a set of classes using ES6 constructs. You'll create three classes: BankAccount (parent) and two children (Savings and Checking).

After you write code you can check its syntax by using *Esprima* located at *http://esprima.org/demo/validate.html* (Just one of countless JS syntax checkers out there!) Nothing fancy; just gets the job done. Just copy/paste code into its window.

Here's what it looks like:



The folder starterforclassesBackup is available in case you totally hose your starter files :(

Step 1: Create a BankAccount class

The file *starterforclasses/bankaccountES6.js* has copious instructions for creating your bankaccount class. Short story – code a CTOR, properties (getters/setters), *printValues*, *deposit* and *withdrawal* methods.

This class will serve as the *parent* class for the two classes (savings, checking) that you will code next. Put your code in the ***classes*** directory.

Step 2: Create the CheckingAccount class

Again, start with *starterforclasses/checkingES6.js* for instructions. This class is a *child* of the aforecoded (I made up a word!) *BankAccount* class. The checking account class has a property, *odProtection* (Overdraft Protection) in addition to the inherited properties from its parent. The odProtection property is a BOOLEAN that states whether or not the account has overdraft protection.

Include the additional property in your CTOR and assign AFTER calling the parent class CTOR; *override* the *printValues* method to include the value of the additional property odProtection and *overide* the *makeWithdrawal* method to check the *odProtection* flag.

Look in the starter file for more detailed instructions. Put your code in the ***classes*** directory

Step 3: Create the SavingsAccount class

Last step is to *create a child class* called Savings**Account** that will inherit ALL the properties of BankAccount and include an additional property called *interest*. This property is a double that we'll use in a totally nonsense way to adjust the balance of a savings account.

Pretty much the same spiel as given for the CheckingAccount class; code the CTOR to accept the parent properties and the additional *interest* property, *override* the *printValues* method to list the value of the parent and new property values to the log and *override* the *deposit* method to use our interest rate property. Put your code in the ***classes*** directory

Step 4: Doing stuff with these classes

Remember to ***use the developer tools*** since your output is going to the console!!!

As you likely figured, you have a starter file, *useES6FeaturesStarter.html*. Take a look while you read what follows.

The function *createBunchaAccounts* returns an *array of account objects*. It's argument is the number of accounts to create. Use a new ES6 feature to default tha arguments value to 50.

Don't code ***var*** for all your JS variables! Think of some more appropriate ES6 keyword that may describe the 'kind' of variable better.

List out the account IDs with a *forEach* function call. The *forEach* function takes **a function argument** that describes what to do with each element of the array. You can code the function externally or code the function body within the call to forEach. The function *listAcctID* is already coded for you; you can use this as an argument to *forEach*.

Next, use *filter* to count the number of savings accounts. The accountIDs start with the letter that describes the account. ('S' for savings, etc.)

Use *reduce* to compute the sum of ALL accounts, then divide by the number of accounts to compute the average. Print the average to the log.

The next two are ***OPTIONAL***

Filter out the savings accounts with balance > 5,000$. Change the balance by using the map function to apply the interest rate (newbalance = oldbalance * (1 + interestRate)). Finally, list the acctID and the (new) balance with a *forEach* function.

Lastly, use *reduce* to **count the number of each account type.**

The starter has lots of comments. Remember to use the syntax checker from time to time!

Step 5: Using ES6 *Arrow functions*

Take the program you just wrote and change the calls to reduce, forEach, map, filter to use **arrow functions**.

Refer to the Ohs for syntax examples.