**CS 628 Full-Stack Development II**

**HOS01E Understanding JavaScript – Variable Scopes and Formatting Practices**

July 16, 2020

12/29/2020 Updated by Min

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* This HOS does not explain all steps in detail**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Resources**

* Holmes, S. & Harber, C. (2019). [Getting MEAN with Mongo, Express, Angular, and Node, Second Edition](https://login.proxy.cityu.edu/sso/skillport?context=148121). Manning Publications. (ISBN 9781617294754)
  + Appendix D - Reintroducing JavaScript
    - Section 2 – Good Habits or Bad Habits
    - Section 7 – Formatting Practices
* VSCode ESLint Extension, <https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint>

**Learning Outcomes**

“JavaScript is easy to start with and **forgiving in the way it’s written**. Unfortunately, this looseness and low barrier to entry can **encourage bad habits, which can cause unexpected results**. (Appendix D)”

Students will be able to:

* Understand differences between “var” (before ES2015), “const,” and “let” in terms of different scopes – global, local – function (using the “var” keyword) and lexical/block (using “let” and “const” keywords).
  + If you declare a variable without using **var**, JavaScript creates the variable in the global scope
  + JavaScript “hoists” all **var** declarations to the top of their scope.
  + Instead of **var**, **const** and **let** are used since ES2015.
    - Lexical/block or Global scope (no function scope)
    - No variable hoisting in function
    - **const** – immutable, **let** - mutable
* Understand best formatting practices:
  + A best practice you should get into is placing the opening bracket of a code block at the end of the statement that starts the block.
  + Blocks of code b associated with **if**, **switch**, **for**, **while**, **try**, **catch**, and **function** should not end with a semicolon. However, when assigning a function or object to a variable, you *do* have a semicolon after the curly braces.
  + When you are defining a long list of variables at the top of a scope, the most common approach is to write one variable name per line.
  + Use whitespace as the delimiters for words,
  + ESLint checks the quality and consistency of your code.
* Analyze JavaScript codes.

**Activities**

* Preparing for a MEAN project environment
* **Installing a code-quality checker**
* **Experiencing variables, scope, and functions**
* **Experiencing formatting practices**
* Pushing your work to GitHub

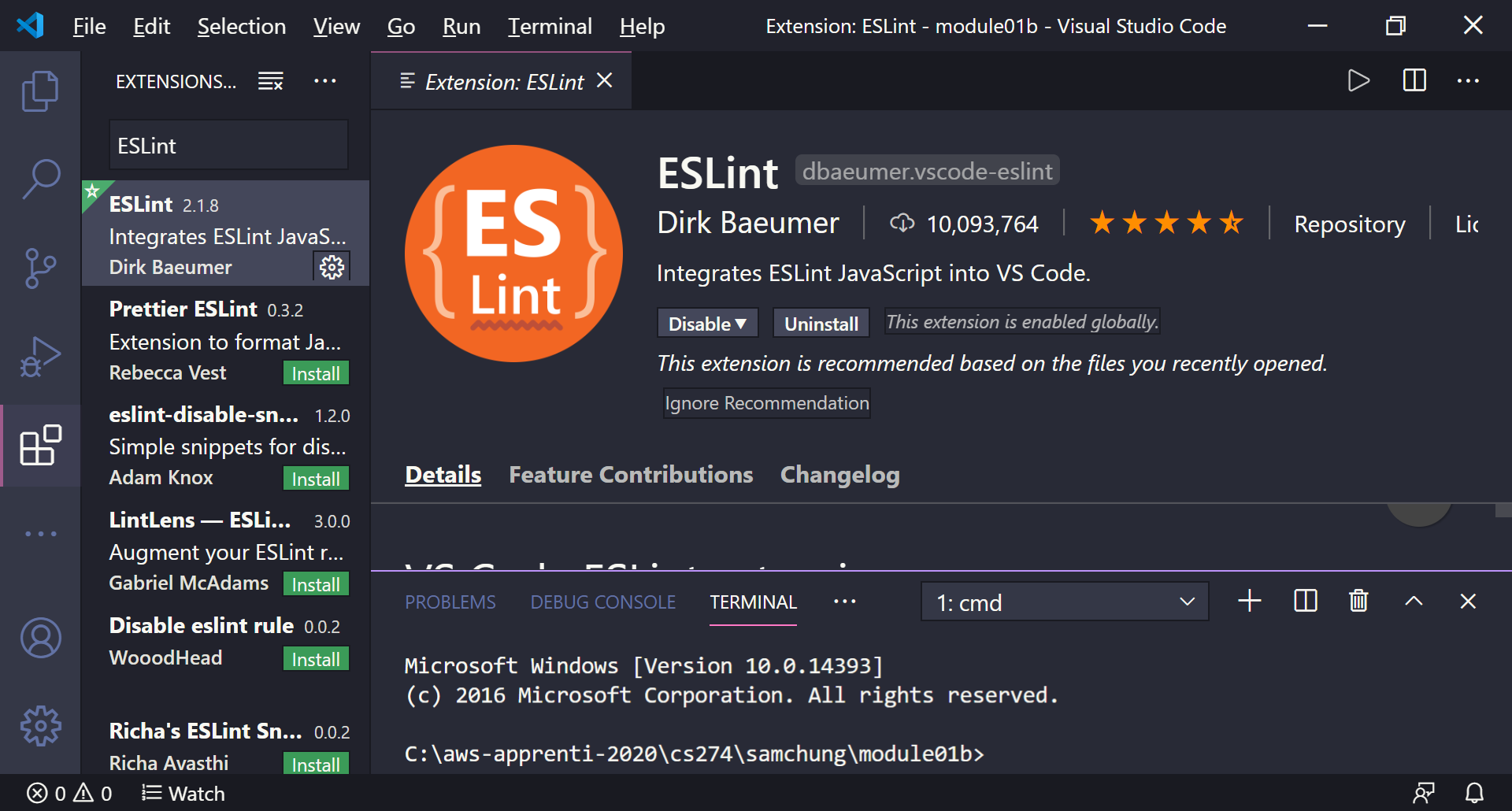
**Preparing for a MEAN project environment**

1. Create the “module01b” folder under your repo:  
   >>>mkdir module01b  
   you will see like that:

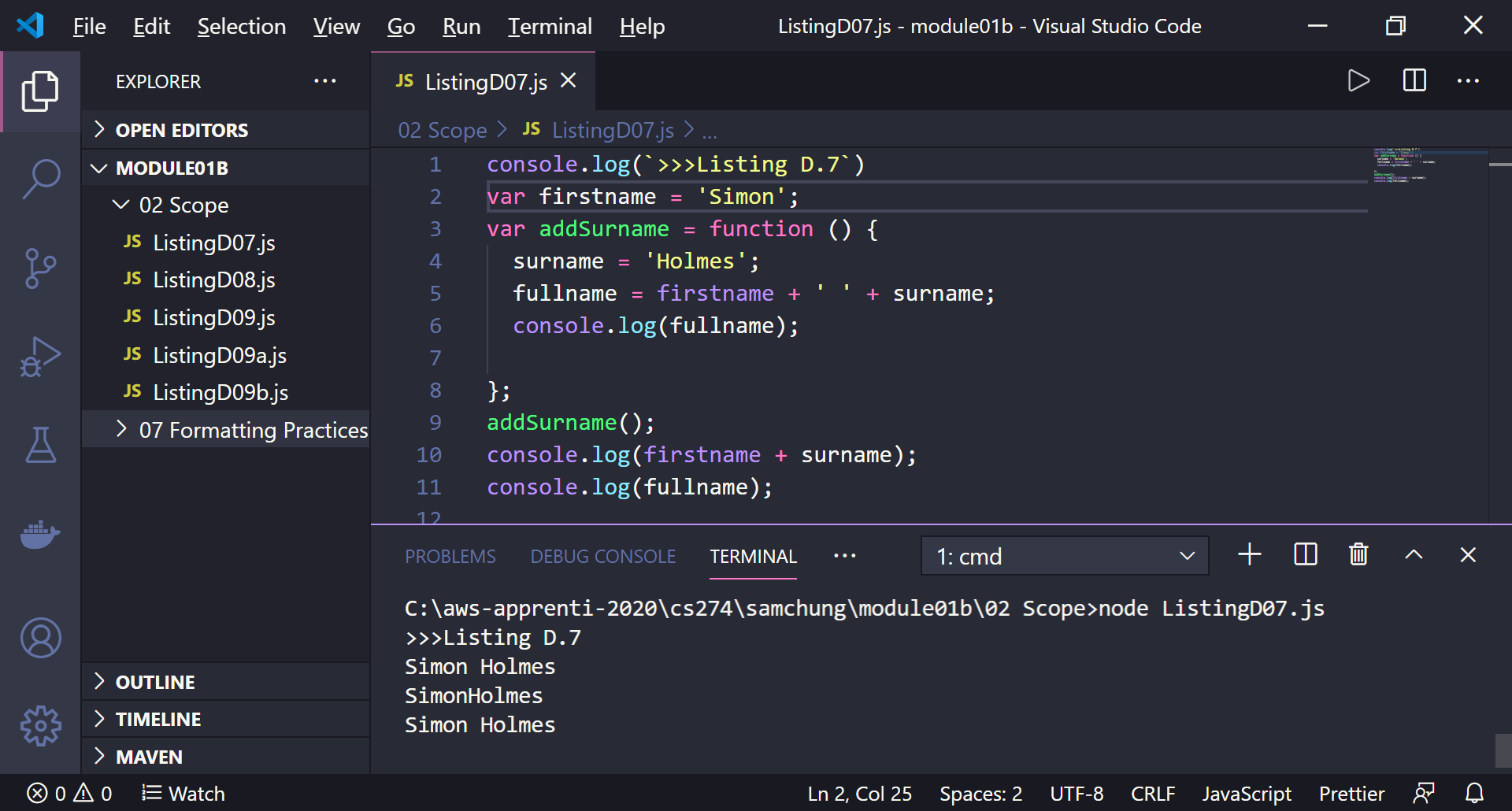


1. Open the “module01b” folder in the VSCode.  
   Close the “Welcome” window.  
   Open the terminal (Ctrl+`).

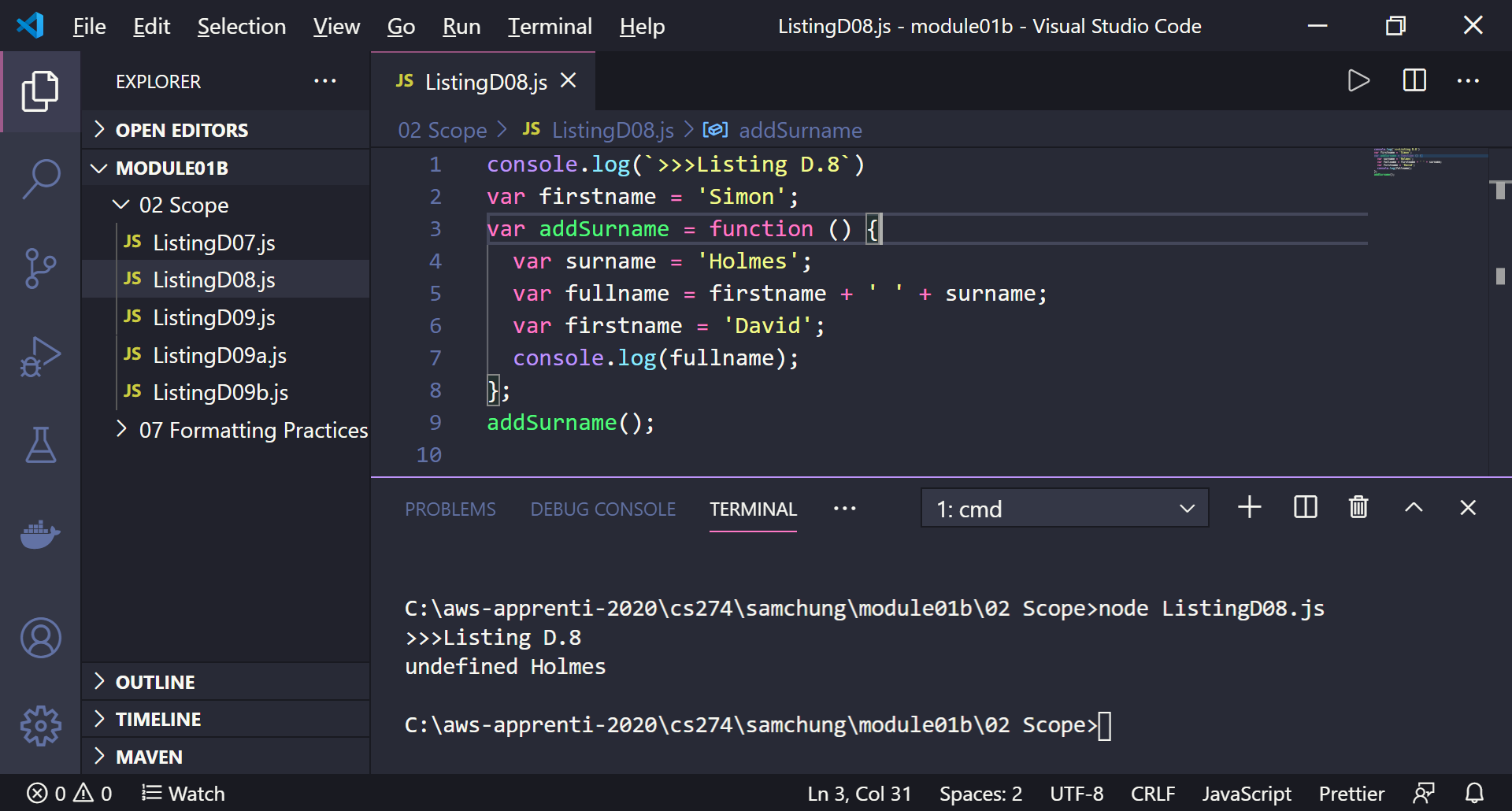
**Installing a code-quality checker**

1. Install a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs.
   1. ESLint is geared more toward linting ES2015 code. TypeScript has its own linter, TSLint, which Angular installs by default
   2. Type **ESLint** in the search bar.
   3. Find the **ESLint** extension in the search results and click the green Install button next to it.  
      

**Experiencing variables, scope, and functions**

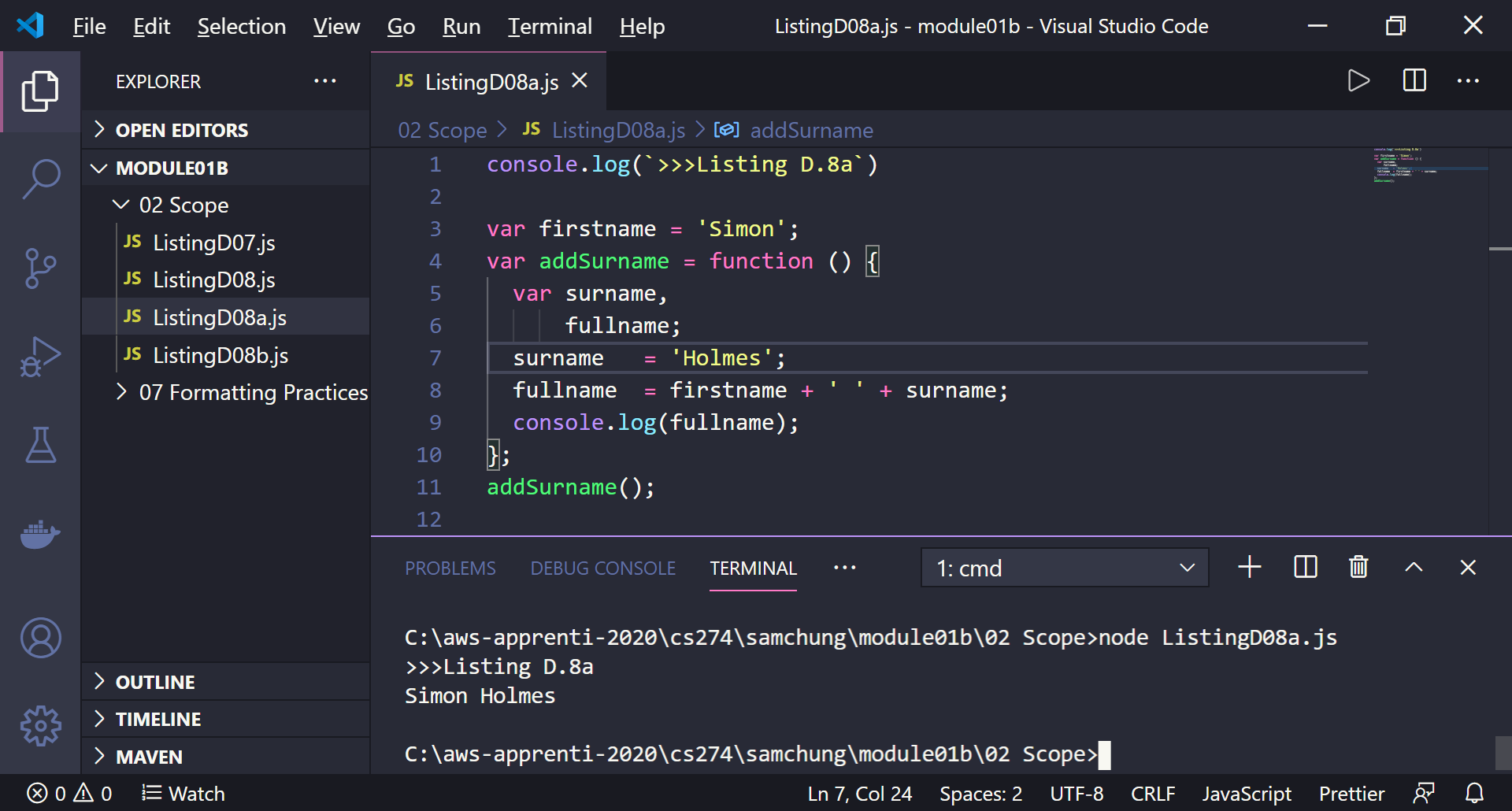
1. Create a “02 Scope” directory and move to the directory.
2. Create a “ListingD07.js” file with scaffold.
3. Execute the sever-side JavaScript file.
4. From Appendix D, find Listing D.7.
5. Enter the source code and execute it.  
   
6. Explain how it works in terms of each variable scope – global, function, & lexical.

|  |  |  |
| --- | --- | --- |
| Location | Variable | Scope |
| Listing07D.js | firstname | Global |
| addSurname( ) | surname | Global |
|  | firstname | Global |
|  | fullname | Global |

1. Create a “ListingD08.js” file by copying the “Listing07.js” file.
2. Update the source code with changes.  
   Execute it.  
    
3. Explain how it works in terms of each variable scope – global, function, & lexical.  
     
    JavaScript has declared the variable firstname at the top of the scope, but it doesn’t have a value to assign to it, so JavaScript leaves the variable undefined when you first try to use it.

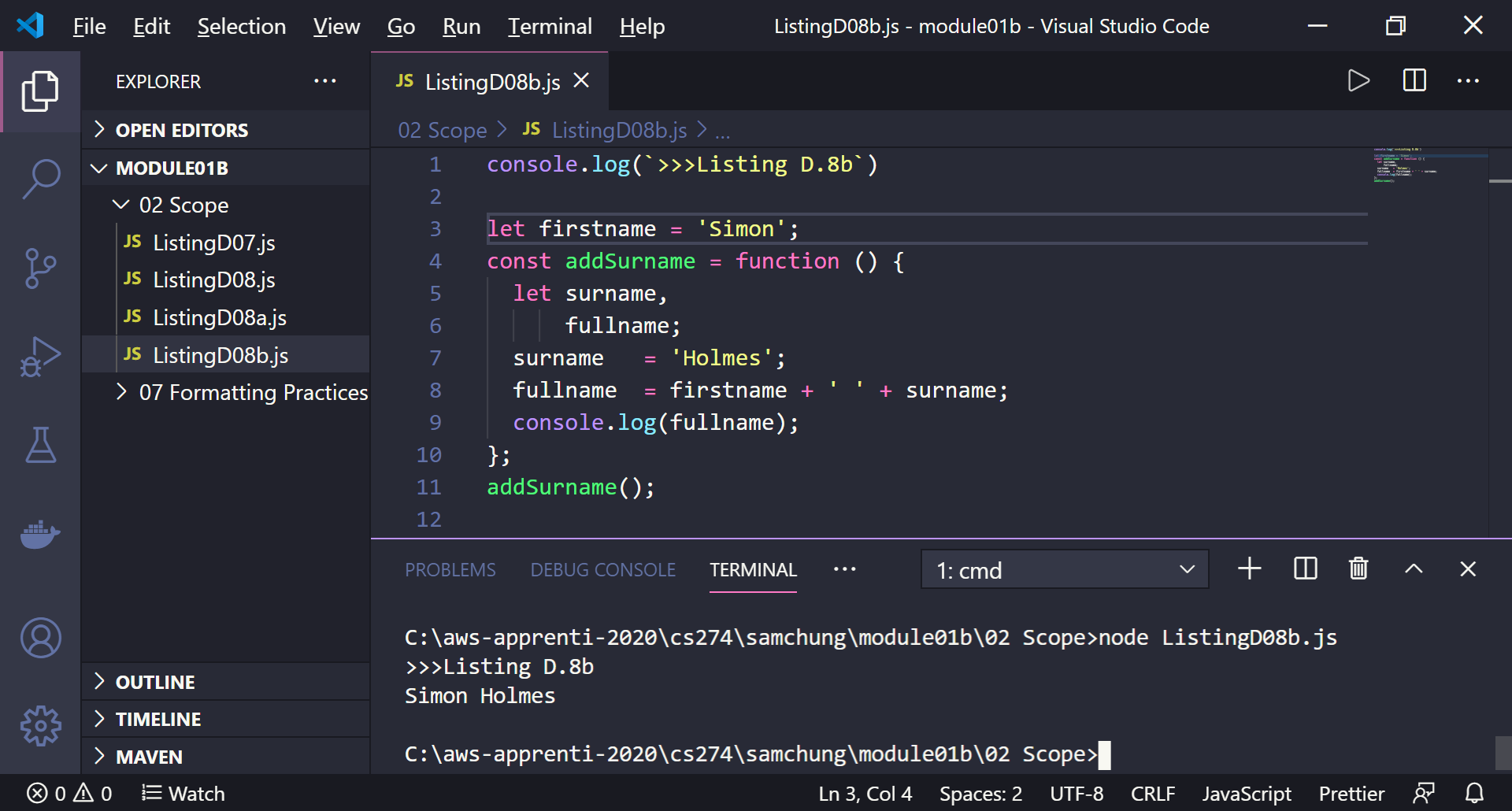
|  |  |  |
| --- | --- | --- |
| Location | Variable | Scope |
| Listing07D.js | firstname | Global |
| addSurname( ) | surname | Lexical |
|  | fullame | Lexical |
|  | firstname | Lexical |

1. Create a “ListingD08a.js” file by copying the”Listing08.js” file.  
   Update the source code with the change.  
   Then, execute it.



1. Explain how it works in terms of each variable scope – global, function, & lexical.

All the variables are global.

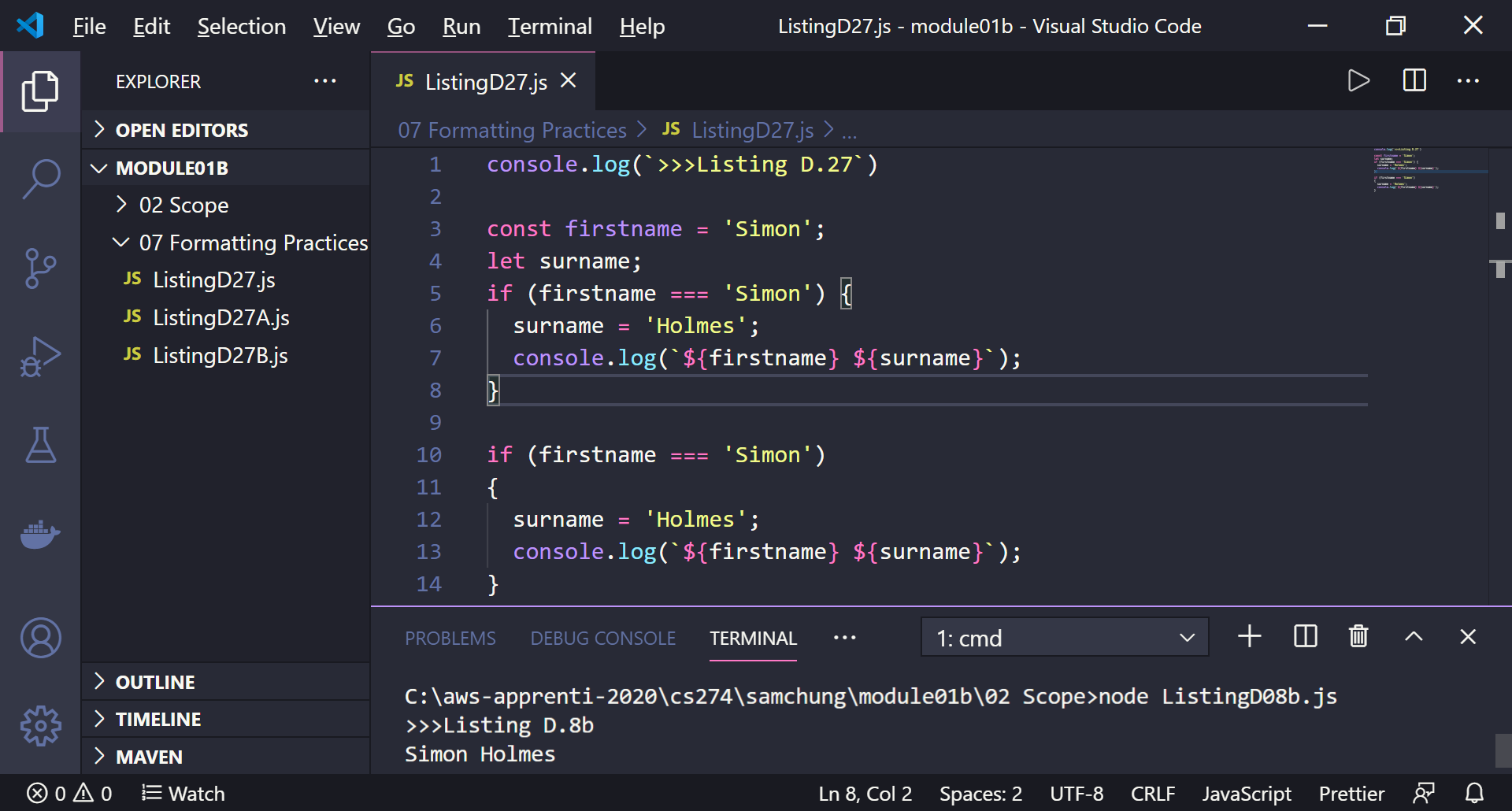
1. Create a “ListingD08b.js” file by copying the”Listing08a.js” file.  
   Update the source code with the change.  
   Then, execute it.  
   
2. Explain how it works in terms of each variable scope – global, function, & lexical.

This time we used local variables. Surname and fullname are limited to function only.

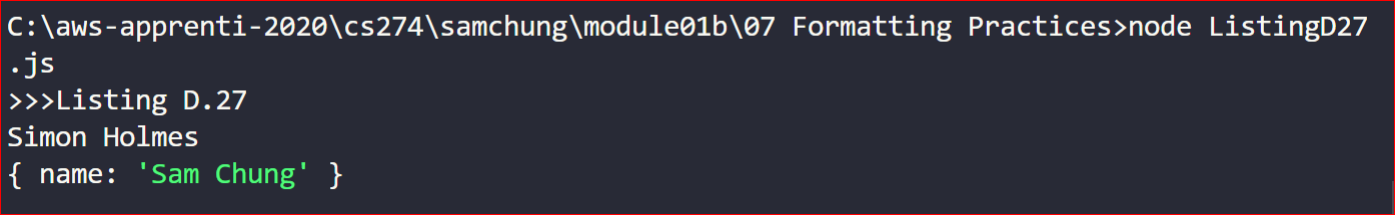
**Experiencing formatting practices**

1. Create a “07 Formatting Practice” directory and move to the directory.
2. Create a “ListingD27.js” file with scaffold

console.log(`>>>Listing D.27`)

1. Execute the sever-side JavaScript file.
2. From Appendix D, find Listing D.27.
3. Enter the source code and execute it.  
   
4. There are two if statement that are doing the same operation.  
   Delete one for better practice.
5. Add the following code to see why the practice is important.

* function returnFullname() {
* return
* {
* name : "Sam Chung"
* };
* }
* console.log (returnFullname());

1. Execute it and fix the problem in this code to show the following outcome:
2. For the name display, use your first and last name.  
   

**Pushing your work to GitHub**

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hitting the control + ~ key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 1--yourname”

>>> git push origin master

If you cannot remember your branch name, run the command “git status” to check.