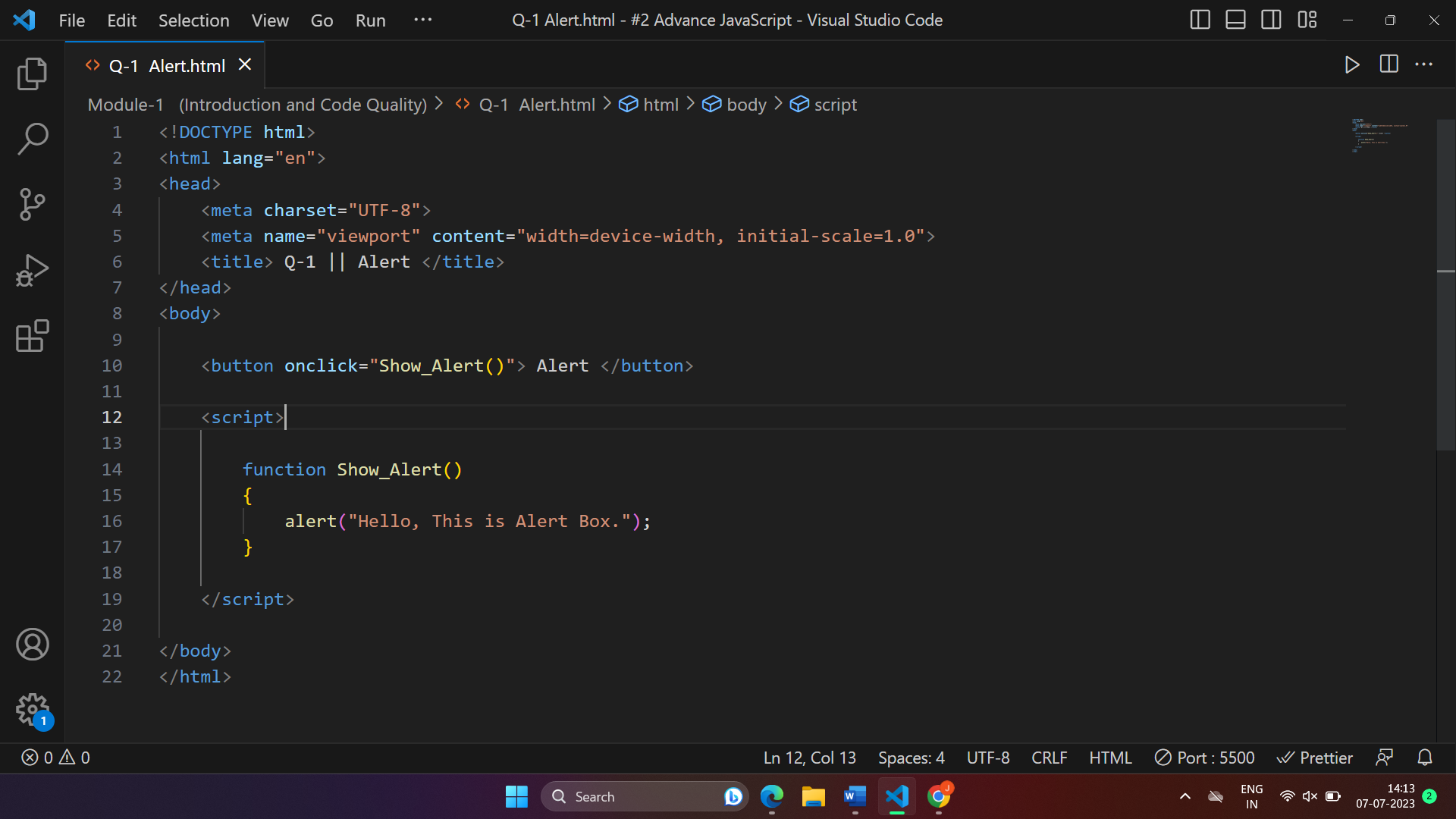
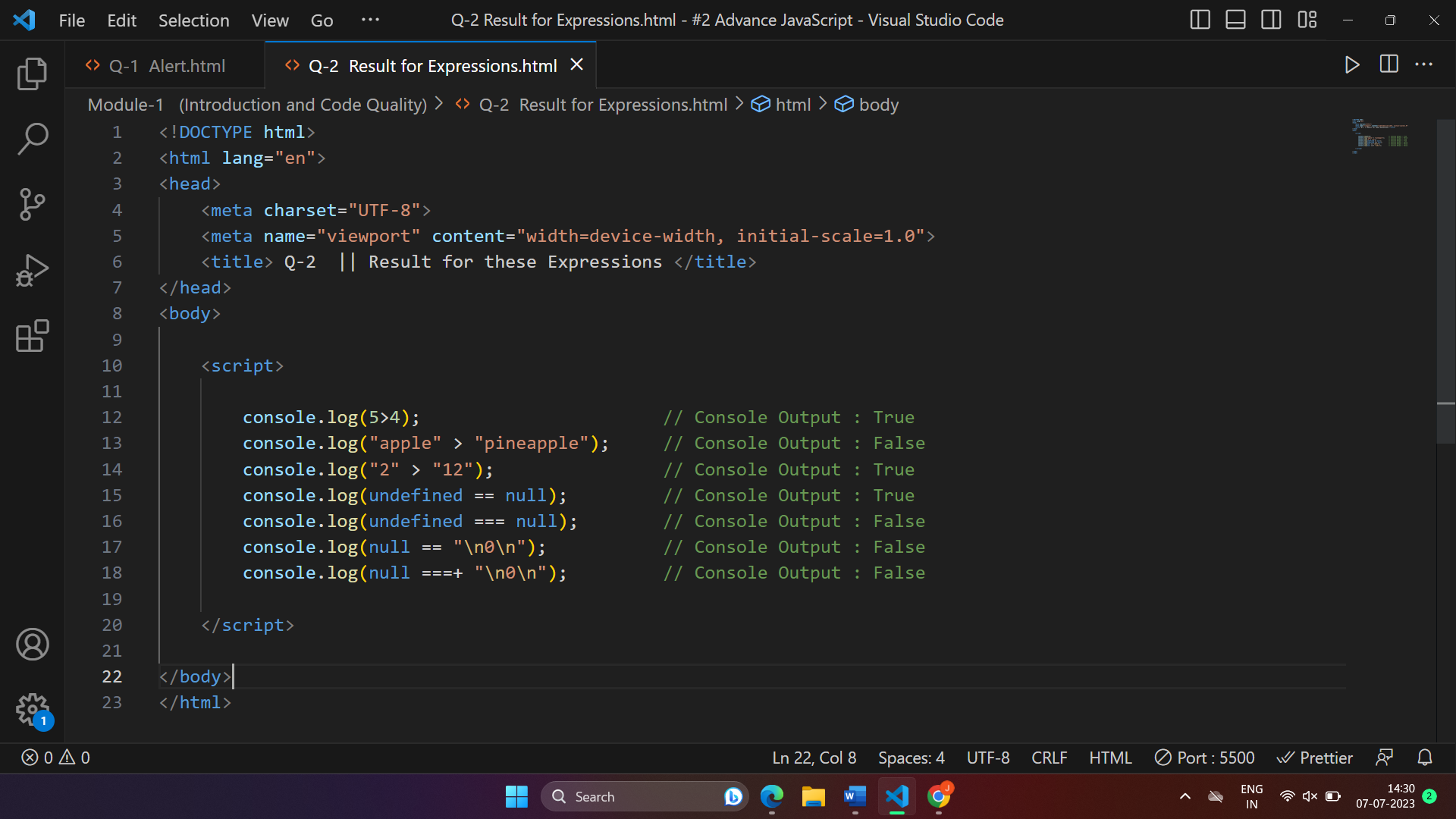
**Advance JavaScript**

**Module : 1 (Introduction and Code Quality)**

1. **Write a programme to Show an alert ?**

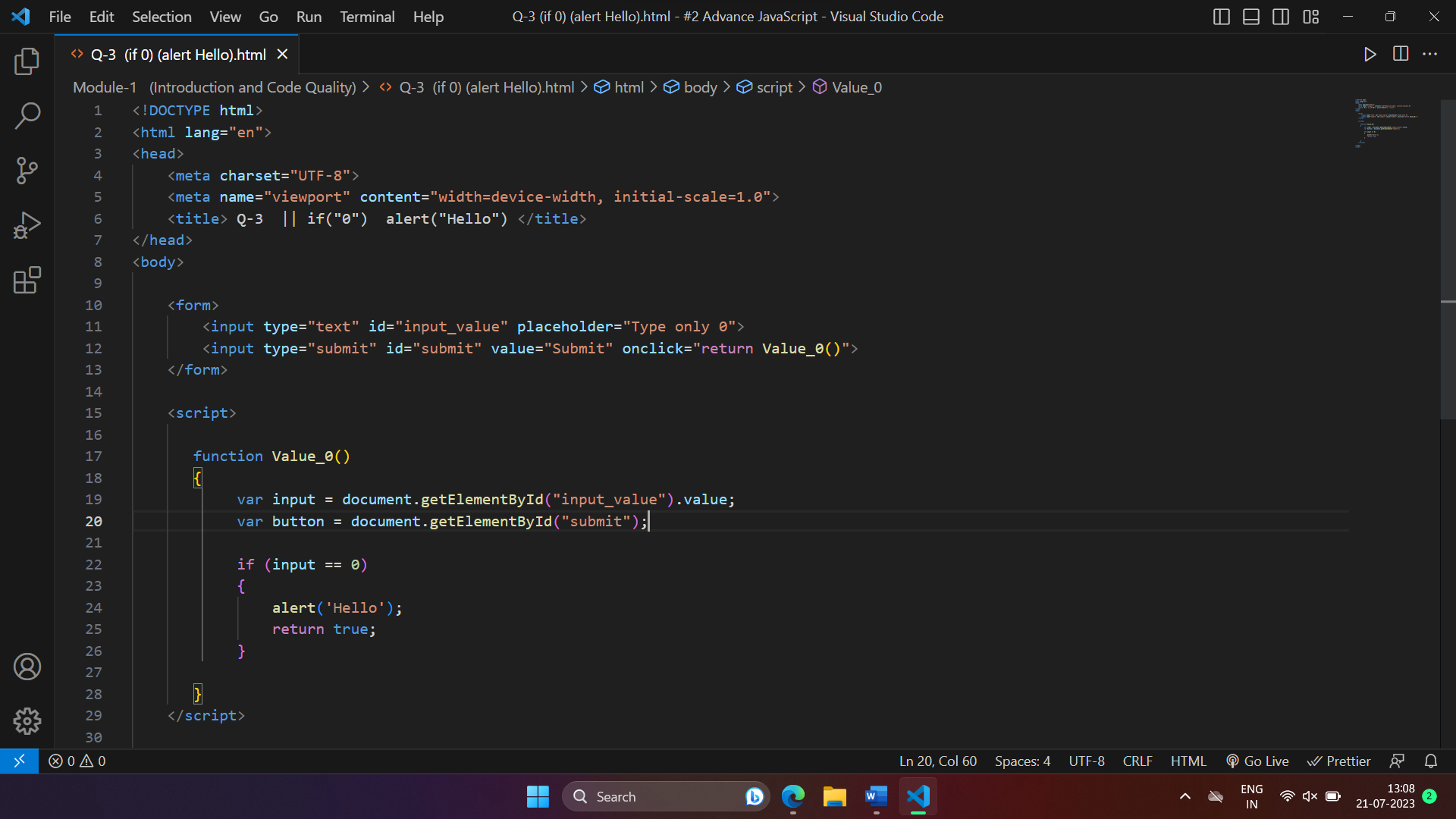


1. **What will be the result for these expressions ?**
2. **5>4**
3. **“apple” > “pineapple”**
4. **“2” > “12”**
5. **undefined == null**
6. **undefined === null**
7. **null == “\n0\n”**
8. **null ===+ “\n0\n”**



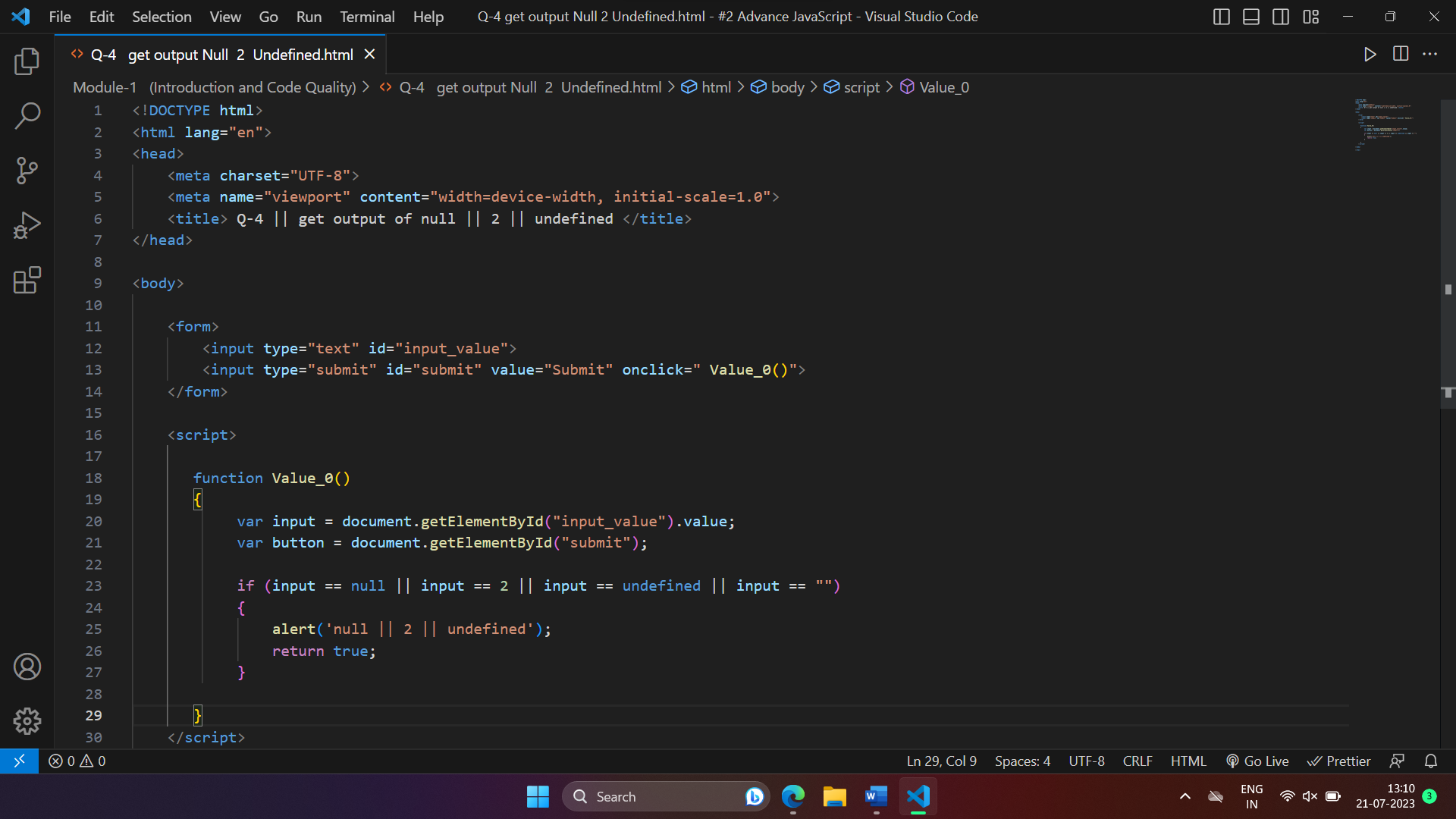
1. **Will alert be shown ?**

**If (“0”) { alert(“Hello”); }**

****

1. **What is the code below going to output ?**

**Alert( null || 2 || undefined );**

****

1. **The following function returns true if the parameter age is geater than 18. Otherwise it asks for a confirmation and returns its result :**

**function**

**checkage(age)**

**{**

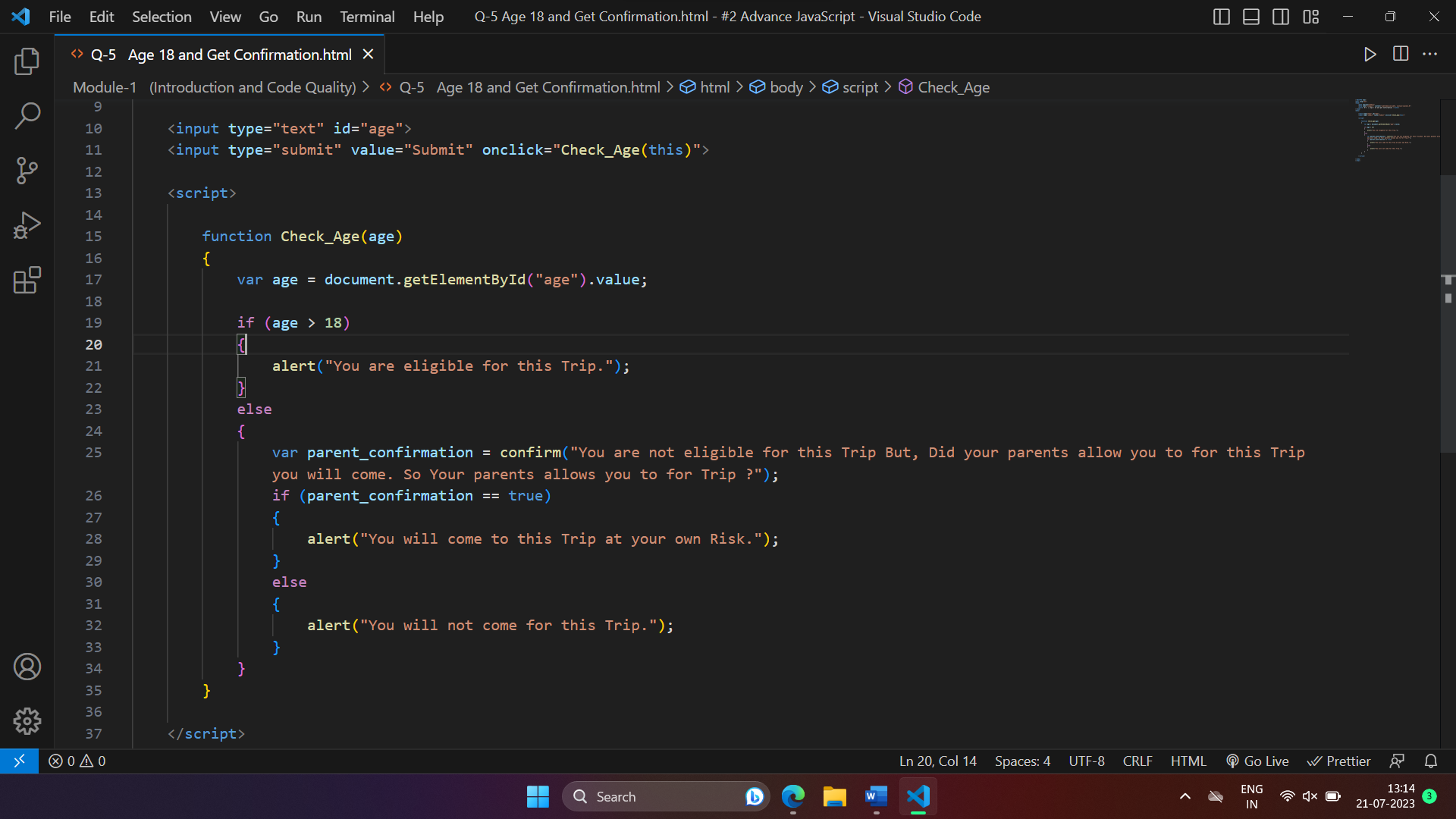
**If (age>18) { return true; }**

**else {**

**//…return confirm (‘did parents allow you?’);**

**}**

**}**

****

1. **Replace Function Expressions with arrow fucntions in the code below :**

**ask (question, yes, no)**

**{ if (confirm(quetion))yes();**

**else**

**no();**

**}**

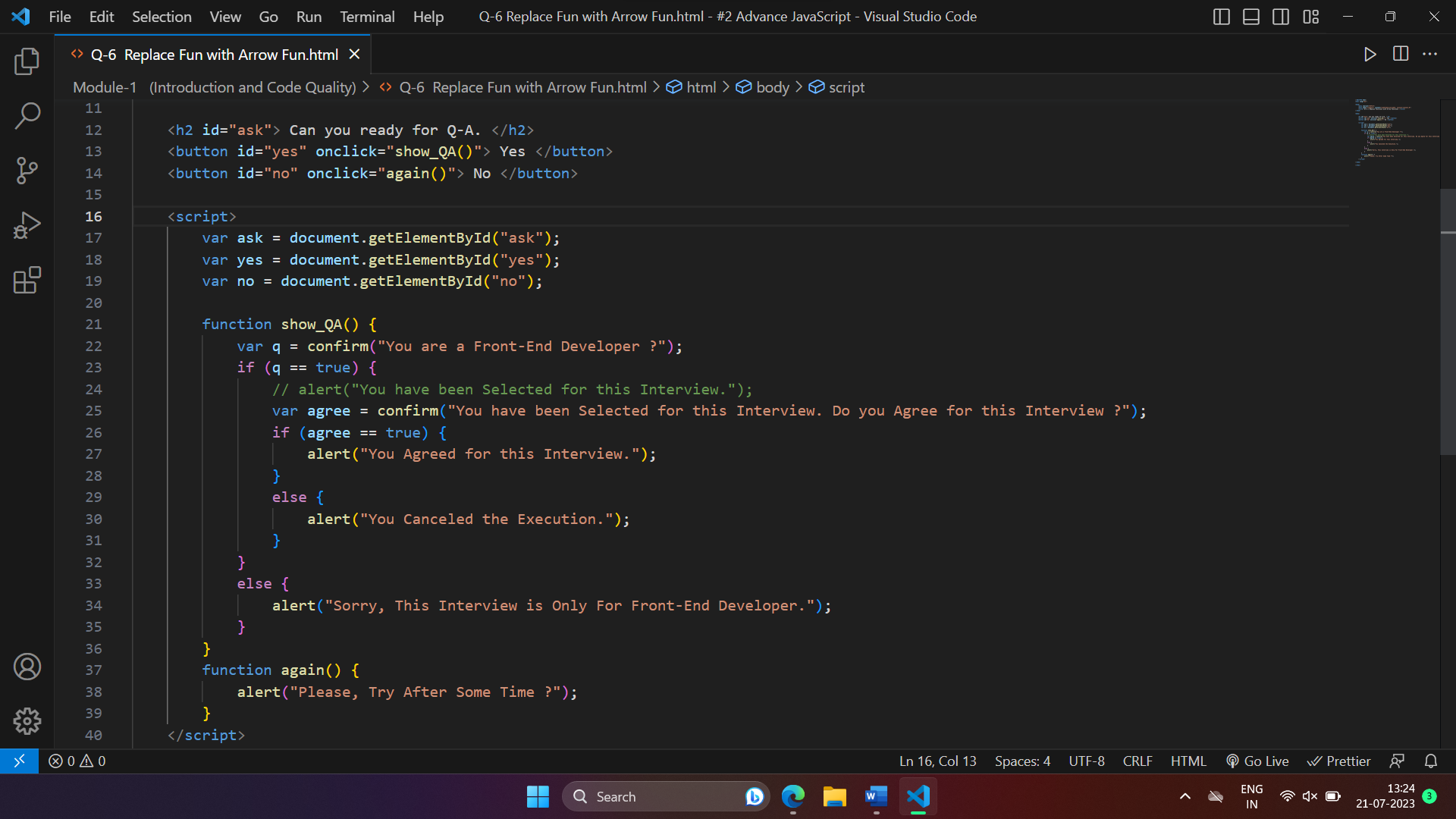
**ask (“Do you agree ?”, function)**

**{ alert(“You agreed.”); },**

**function() {**

**alert(“You canceled the execution.”); }**

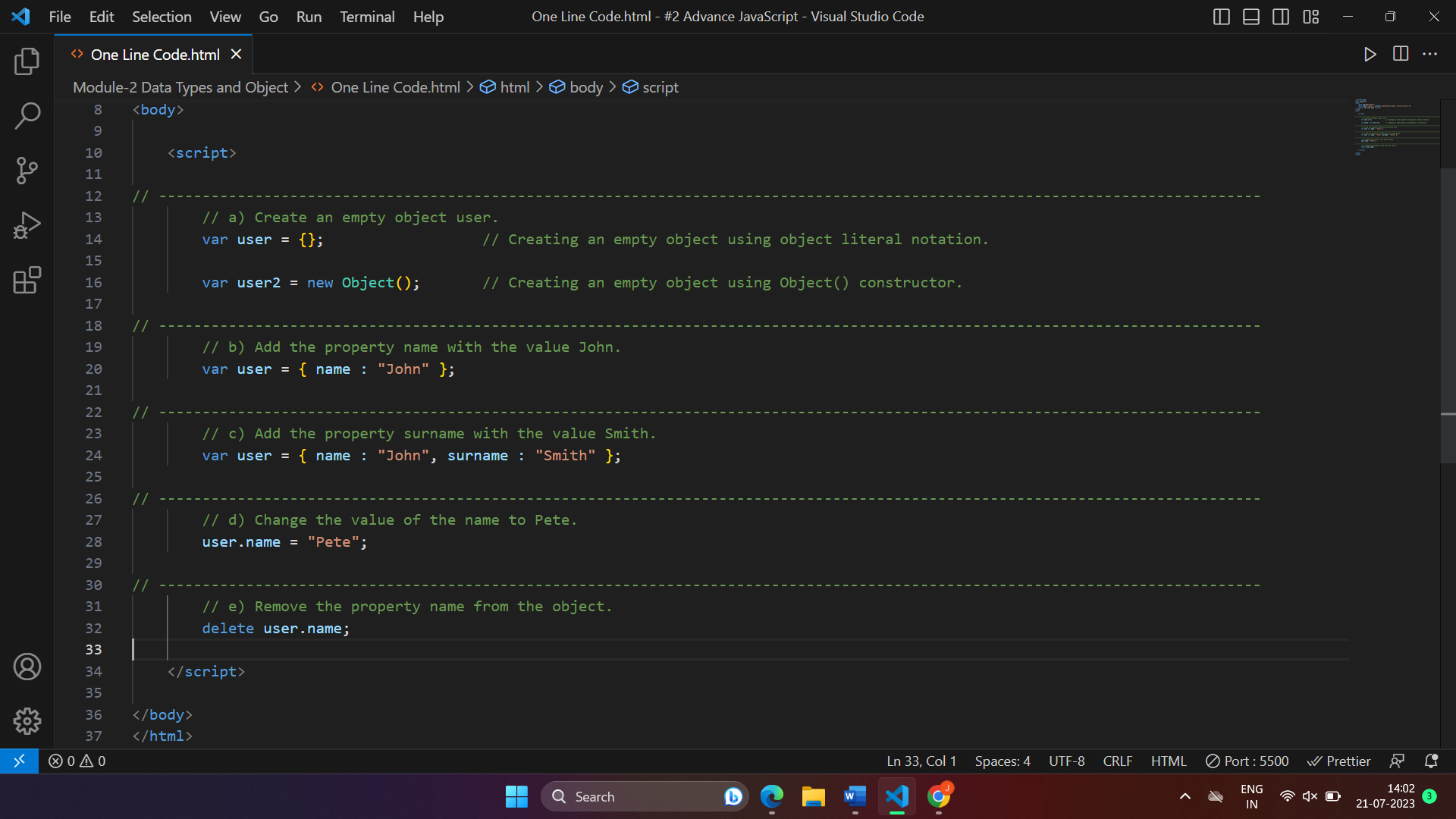
**}**

****

**Module : 2 (Data Types and Objects)**

* **Write the code, one line for each action :**

1. **Create an empty object user.**
2. **Add the property name with the value John.**
3. **Add the property surname with the value Smith.**
4. **Change the value of the name to Pete.**
5. **Remove the property name from the object.**

****

* **Is array copied ?**

**let fruits = [“Apples”, “Pear”, “Orange”];**

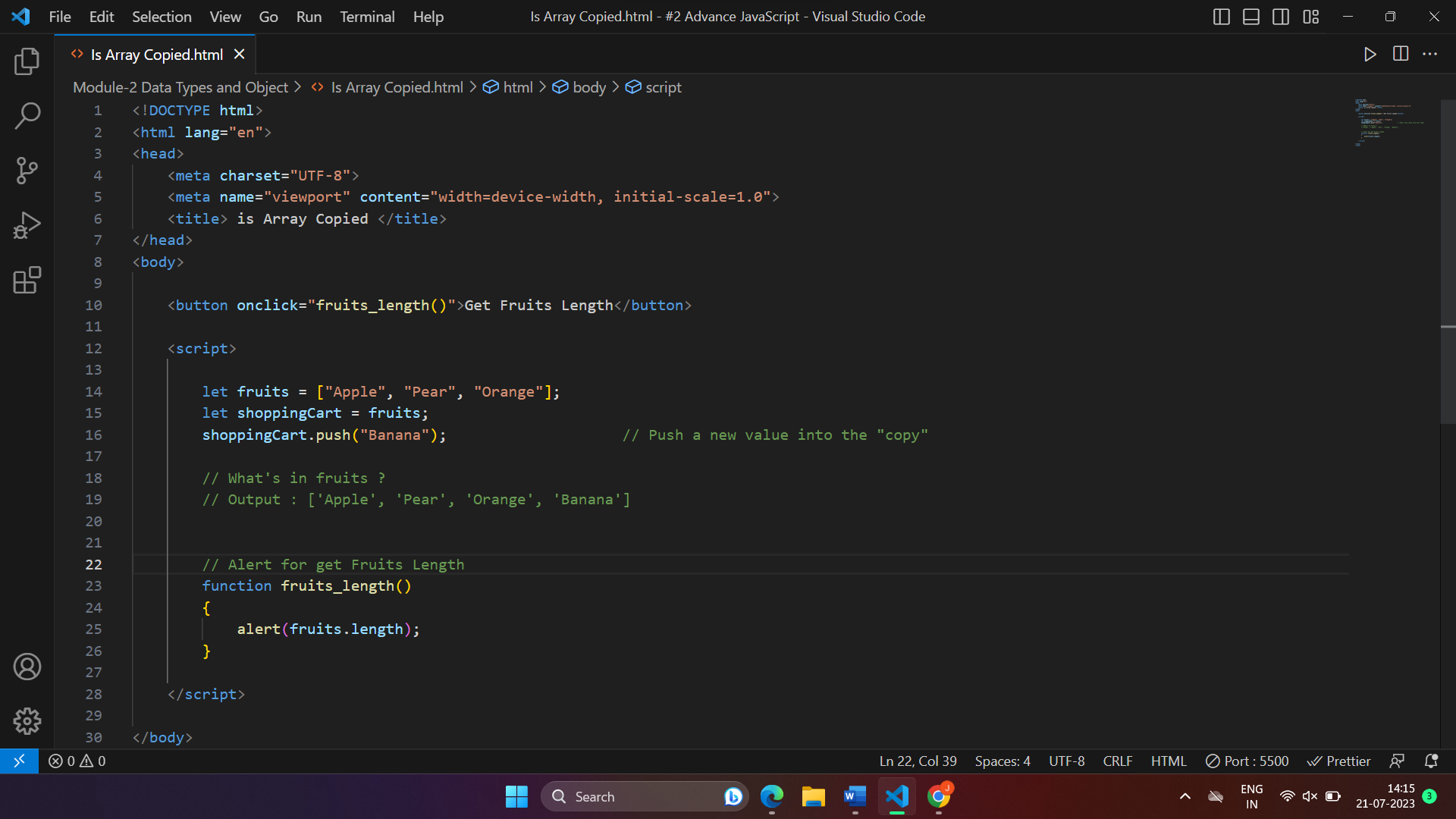
**// push a new value into the “copy”**

**let shoppingCart = fruits;**

**shoppingCart.push(“Banana”);**

**// what’s in fruits ?**

**alert( fruits.length );**

****

* **Map to names**

**let john = { name: “John”, age: 25 };**

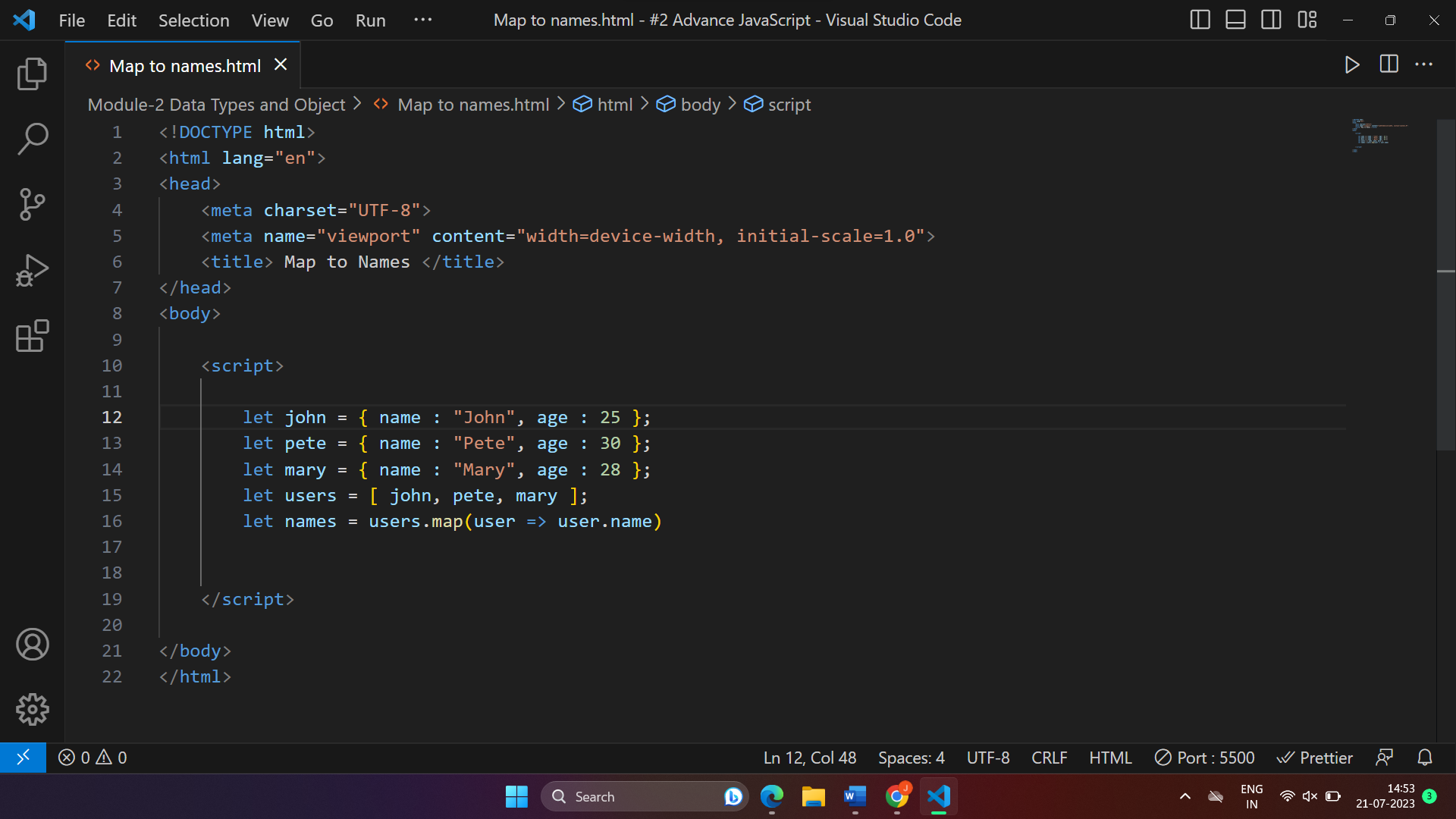
**let pete = { name:”Pete”, age: 30 };**

**let mary = { name: “Mary”, age: 28 };**

**let users = [ john, pete, mary ];**

**let names = /\* … your code \*/**

**alert(names); // John, Pete, Mary**

****

* **Map to objects**

**Let john = { name: “John”, surname: “Smith”, id: 1 };**

**Let pete = { name: “Pete”, surname: “Hunt”, id: 2 };**

**Let mary = { name: “Mary”, surname: “Key”, id: 3 };**

**Let users = [ john, pete, mary ];**

**let users Mapped = /\* …your code …\*/**

**/\***

**usersMapped = [**

**{ fullName: “John Smith”, id; 1 },**

**{ fullName: “Pete Hunt”, id; 2 },**

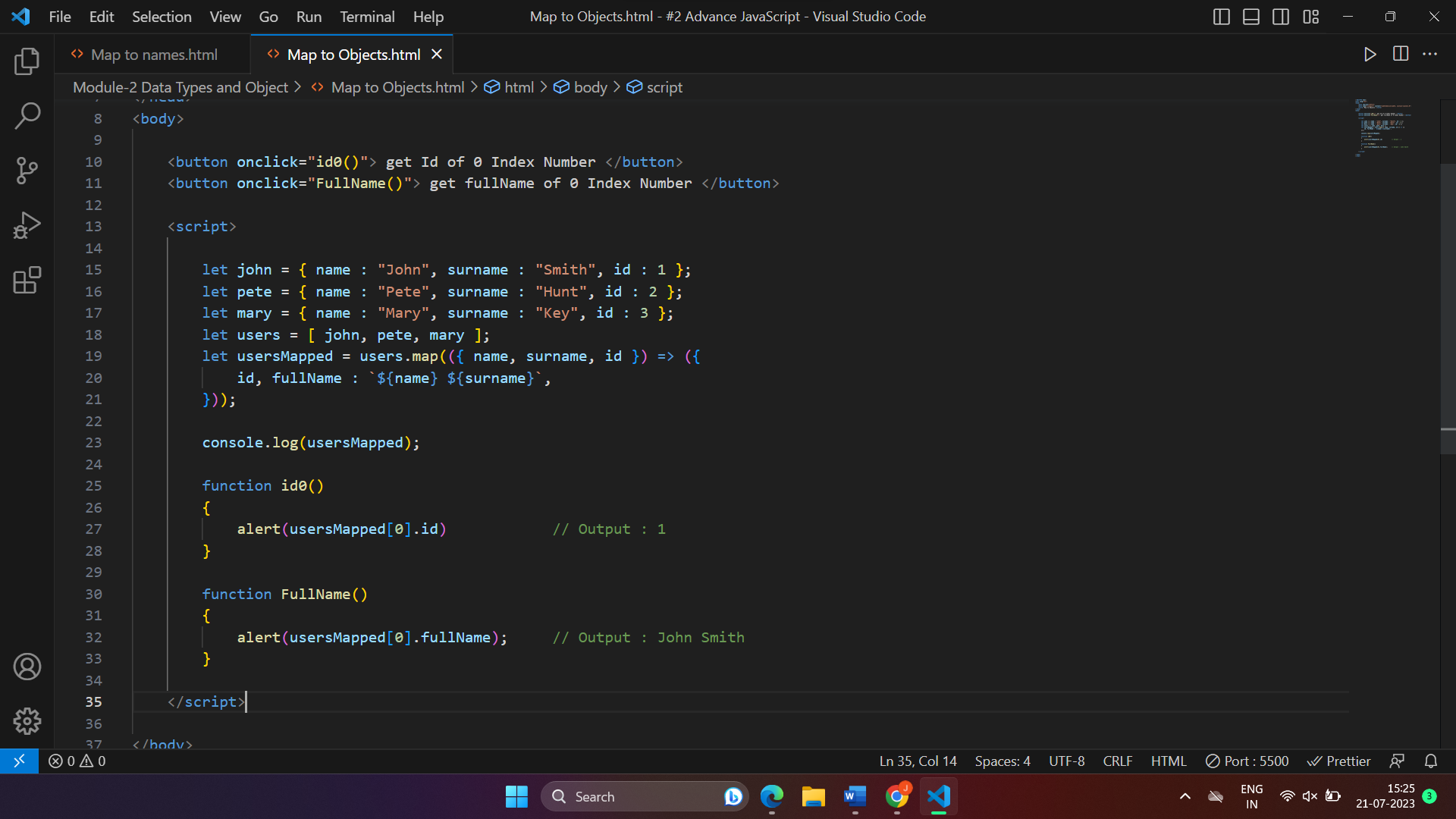
**{ fullName: “Mary Key”, id; 3 }**

**]**

**\*/**

**alert( usersMapped[0].id ) // 1**

**alert( usersMapped[0].fullName ) // John Smith**

****

* **Sum the properties There is a salaries object with arbitrary number of salaries. Write the function sumSalaries(salaries) that returns the sum of all salaries using Object. values and the for..of loop. If salaries is empty, then the result must be ().**

**let salaries = {**

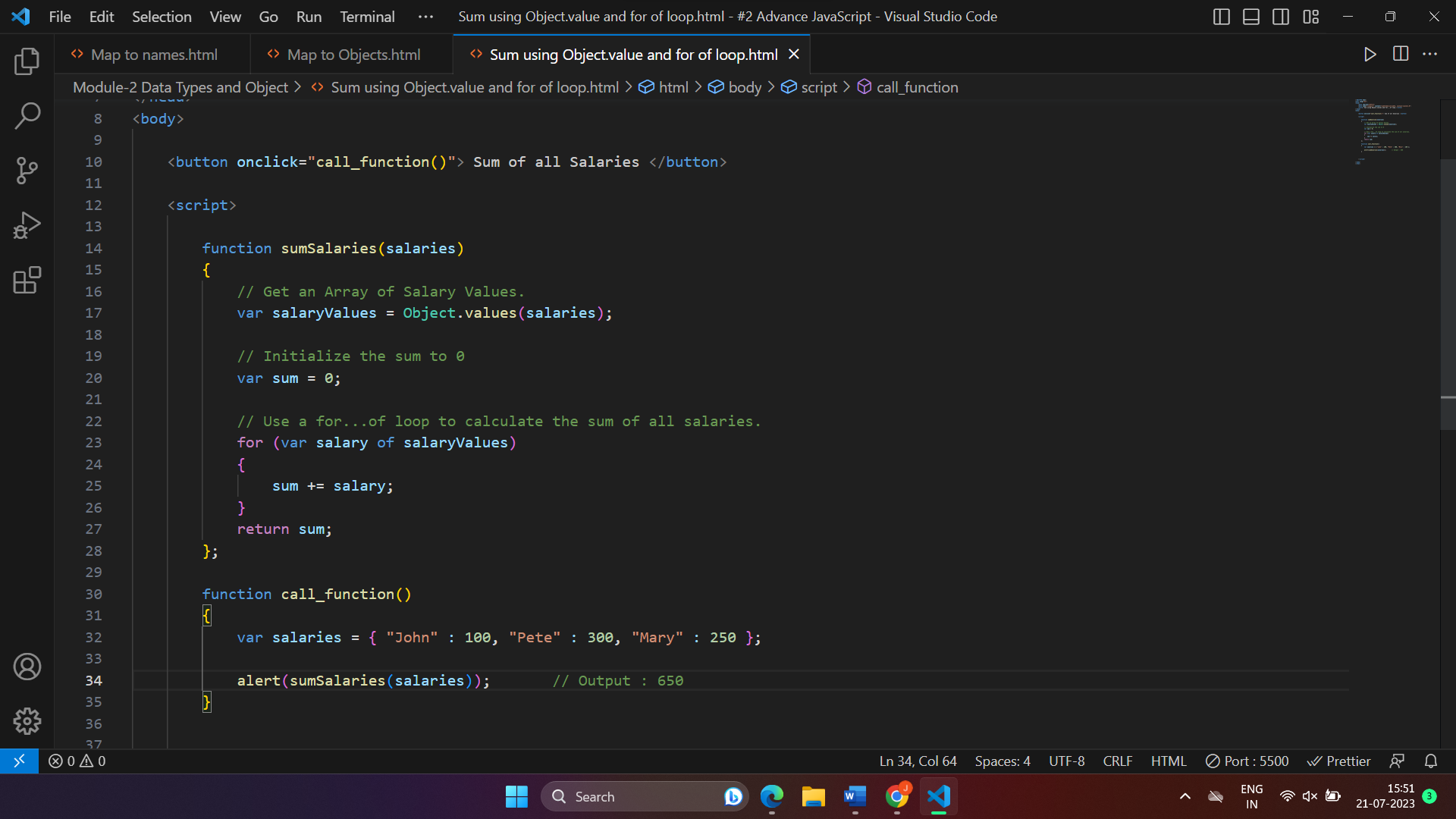
**“John”: 100,**

**“Pete”: 300,**

**“Mary”: 250**

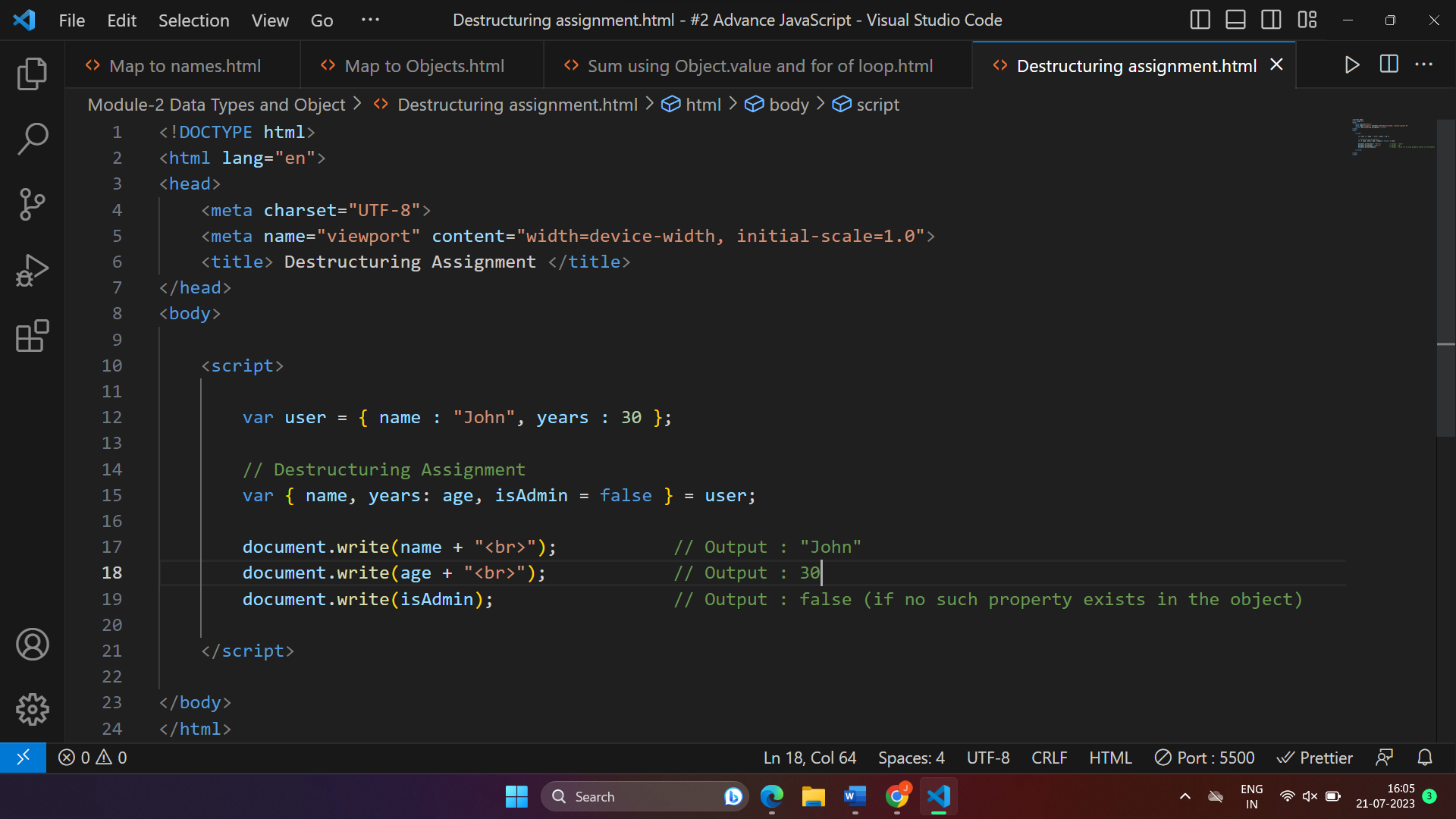
**};**

**alert( sumSalaries(salaries) ); // 650**

****

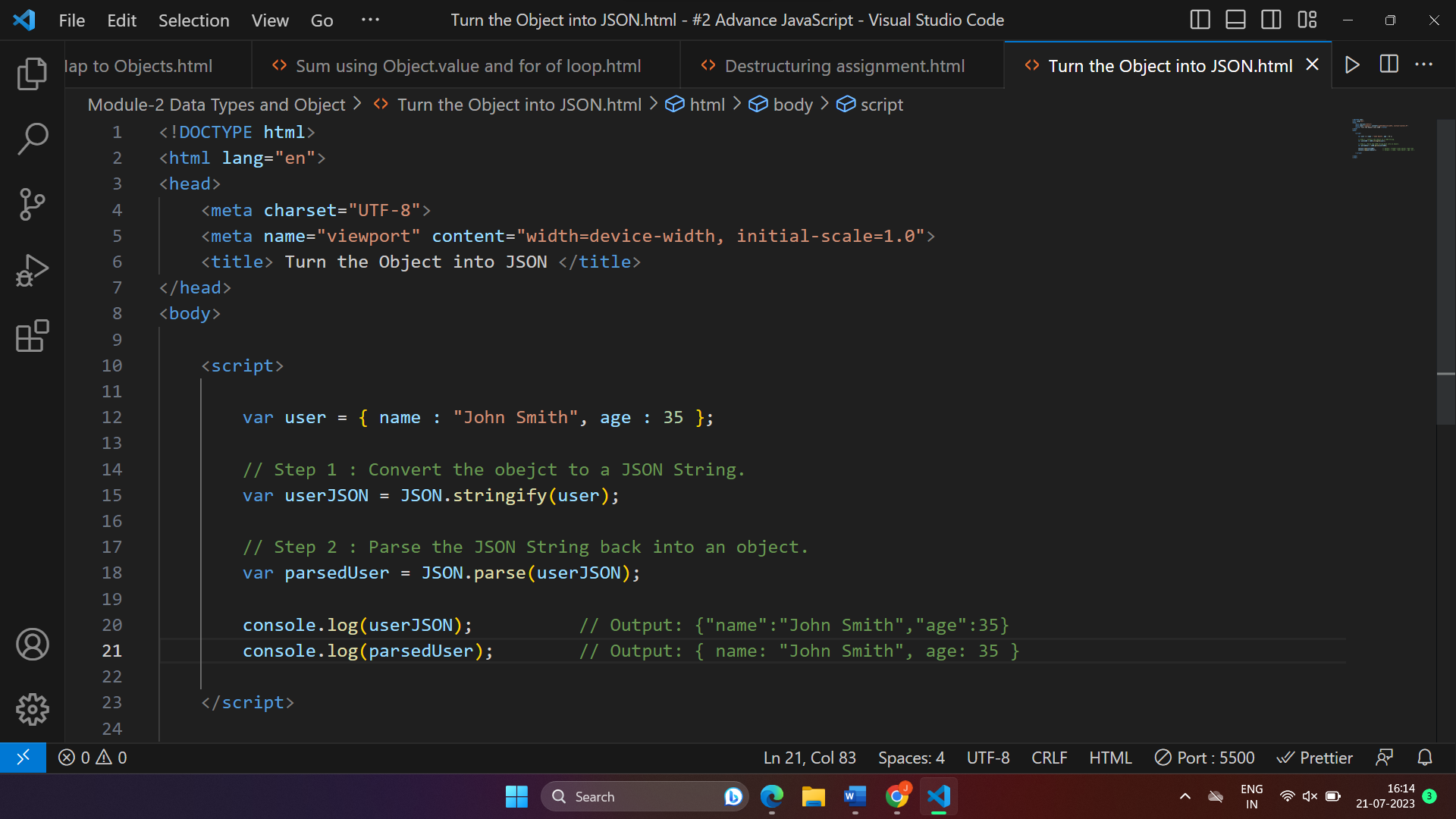
* **Destructing assignment We have an object : Write the Destructing assignment that reads :**

1. **Name property into the variable name.**
2. **Year’s property into the variable age.**
3. **isAdmin property into the variable isAdmin (false, if no such property)**
4. **let user = { name: “John”, age: 30 };**

****

* **Turn the object into JSON and back Turn the user into JSON and then read it back into another variable.**

**User = { name: “John Smith”, age: 35 };**

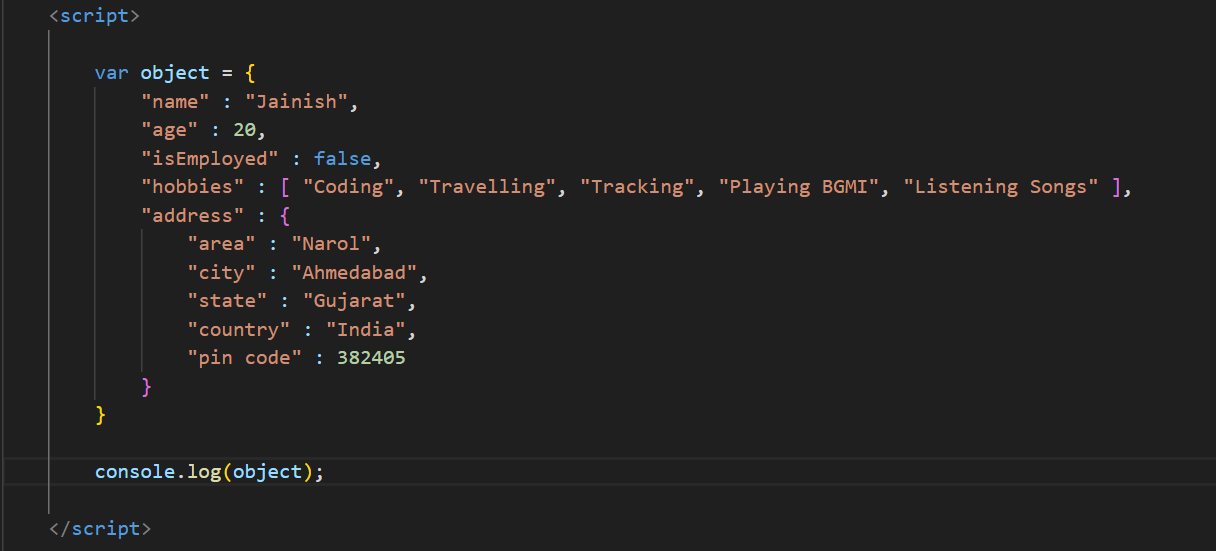
****

**Module : 4 (New Request)**

* **What is JSON**
* JSON stands for “JavaScript Object Notation”, and it is a lightweight data interchange format. JSON is commonly used for transmitting data between a server and a web application as an alternative to XML.

In simple terms, JSON provides a standardized way to represent data in a format that is both human-readable and machine-readable. It is based on key-value pairs, where each key is a string and each value can be a String, Number, Boolean, Array or Another JSON object.

**Example :**

****

In this above JSON example, we have a simple object with various data types :

* “name” : “Jainish” :- Key “name” with a String value “Jainish”
* “age” : 30 :- Key “age” with a Numeric value 30
* “isEmployed” : false :- Key “isEmployed” with a Boolean value False.
* “hobbies” : [ “Coding”, “Travelling”, “Tracking”, “Playing BGMI”, “Listening Songs” ] :- Key “hobbies" with an Array of Strings as its Value.
* “address” : { “area” : “Narol”, “city” : “Ahmedabad”, “state” : “Gujarat”, “country” : “India”, “pin code” : 382405 } :- Key “address” with an Object as its value containing keys “area”, “city”, “state”, “country” and “pin code”.

JSON is widely used in Web-Development for APIs, Configuration files and Data interchange due to its simplicity, readability and compatibility with different programming languages. It has become a standard format for data exchange and communication on the web.

* **What is promises**
* JavaScript Promise Object : A JavaScript Promise object contains both the producing code and calls to the consuming code.

Promise Object Properties

A JavaScript Promise Object can be :

1. Pending : The intial state when the task is still in progress.
2. Fulfilled (Resolved) : The task was successful and the promise has a result (data).
3. Rejected : The task encountered an error and the promise has an error message.

A Simple analogy for promises could be ordering food from a restaurant. You place an order (creating a promise) and the restaurant can do three things :

1. Preparing your order (Pending) : The restaurant is working on your order.
2. Delivering your order (Fulfilled) : The order is successfully delivered to you.
3. Apologizing for a mistake (Rejected) : The restaurant informs you that they cannot fulfil your order due to some issue.

**Module : 4 (JavaScript Essentials)**

* **What is JavaScript Output method ?**
* In JavaScript, There are several methods available to display output, which allow you to show information or results to the user.

JavaScript can “display” data in different ways :

1. Writing into an HTML element, using “innetHTML”.
2. Writing into the HTML output using “document.write()”.
3. Writing into the browser console, using “console.log()”.
4. Writing into an alert box, using “alert()”.
5. Writing into an confirm box, using “confirm()”.
6. Writing into an prompt box, using “prompt ()”.

* **How to used JavaScript Output method ?**
* Some JavaScript Output Methods :

1. **console.log()**

Use this method to display output in the browser console. It’ perfect for debugging and checking the values of variables or messages during development.

1. **document.write()**

Use this method to write content directly to the web page. It’s best for simple demonstrations or testing purposes, but avoid using it after the page has loaded, as it can overwrite the entire document.

1. **alert()**

Use this method to show a pop-up message to the user with an “OK” button. It’s useful for providing important notifications or messages.

1. **innerHTML()**

Use this property to update the HTML content of an element on the web page dynamically. It’s commonly used for displaying information based on user interactions or data from JavaScript.

* **How to used JavaScript Events to do all examples ?**
* JavaScript events are used to respond to user interactions or actions that occur on a web page. You can attach event handlers to specific HTML elements or document-wide events to execute JavaScript code when the events occur.

1. **console.log() using Event :**

We can use an event listener on a button to log a message to the console when the button is clicked.

1. **alert() using Event :**

We can show an alert when a link is clicked.

1. **document.write() using Event :**

We can use an event listener on a button to write content to the document when the button is clicked.

1. **innerHTML using Event :**

We can update content when a button clicked.

In each example, we select the HTML element with JavaScript (‘getElementById’, etc.) and use ‘addEvenListener’ to attach an event handler function to the element. When the specified event (Example : “click”) occurs, the associated function is executed, provided the desired functionality.

Events are fundamentals for creating interactive and responsive web pages. You can use various types of events, such as click, mouseover, mouseout, keydown, keyup, keypress, submit, blur, etc., to handle user interactions and execute JavaScript code accordingly.