NAME-ARPNA

STUDENT ID : C0913838

Ans1: JavaScript received a significant boost with ES6 (ECMAScript 2015) that is a standardized version of javascript.

Common features of ES6:

1. Let and Const: New methods for defining variables i.e no more need to write "var" variable.
2. Arrow functions: It helps to write functions using a shorter syntax .
3. Template literals: Template literals simplify string creation by using backticks (`) and embedding variables with ${}. This approach replaces the need for traditional string concatenation, making the code more readable.
4. Destructuring : Makes it simpler to extract values from objects and arrays.
5. Modules: Provides the ability to import and export code that has been divided into separate files.
6. Default Parameters: Parameter default values are now possible for functions.

Ans2: let and const: They both are block-scoped, they can only be found in the block where you define them, such as an if statement or a for loop.

For example:

let message = "Hello, world!";

if (true) {

let message = "Hello, inside block!"; // different "message" inside this block

console.log(message); // "Hello, inside block!"

}

console.log(message); // "Hello, world!" (outside the block)

Constant : It is a keyword that means “Fixed” i.e the value cannot be change over time. We can use const to declare a variable but unlike let and var ,but it must be immediately initialized.

For eg :

**const birthYear = 1995;**

**// birthYear = 2000; // Error! You can't change it.**

**On the other hand**  var is function-scoped, meaning it is accessible throughout the entire function. This can sometimes lead to unexpected behavior. Additionally, var is less reliable because it allows for redeclaration within the same scope, making it less predictable.

For Example:

var name = "Alice";

var name = "Bob"; // No problem with 'var', but can cause issues.

Ans3 : An **arrow function** is a more concise way to write functions in JavaScript, introduced in ES6. It uses the => syntax called fat arrow functions and typically allows for shorter, cleaner function expressions.

|  |  |  |
| --- | --- | --- |
| Aspect | Arrow function | Normal function |
| 1.Syntax | shorter | Longer and more explicit. |
| 2. Handling multiple expressions | More difficult to handle (requires curly braces and return). | Handles multiple expressions easily. |
| 3. Constructor functionality | Cannot be used as a constructor. | Can be used as a constructor to create objects. |
| 4. Duplicate named parameters | Not allowed. | Allowed in non-strict mode. |
| 5. Arguments object | Does not have its own arguments object. | Has its own arguments object. |

Ans 4: Set : It is Similar to maps, sets only hold keys,but the map not key-value pairs.   
Although, they are a recent addition to JavaScript in ES6, they are widely used in other computer languages.

We create a Set using the new keyword,

like so let set = new Set();

We can then add entries by using the add method.

// Create a new Set

let set = new Set();

// Add entries to the Set

set.add(1);

set.add(2);

set.add(3);

set.add(2); // Duplicate values are ignored

console.log(set); // Output: Set { 1, 2, 3 }

Ans5: Destructuring allows us to easily extract values from objects and arrays into variables.

For eg :

const book = { title: 'The Great Gatsby', author: 'F. Scott Fitzgerald', year: 1925 };

Before ES6, we would extract values like this:

const t = book.title;

const a = book.author;

console.log(t); // The Great Gatsby

console.log(a); // F. Scott Fitzgerald

With destructuring, we can do it in one line.

const { title: t, author: a } = book;

console.log(t); // The Great Gatsby

console.log(a); // F. Scott Fitzgerald

In this example, { title: t, author: a } tells JavaScript to take the title and author properties from the book object and store them in new variables named t and a. This makes the code cleaner and more efficient!

Ans6: ES6+ and other current JavaScripts can be translated into previous browser-compatible versions using Babel, a JavaScript utility. It makes it easier for developers to utilize the newest features without being concerned about incompatibility.

Key words:

1. Transpilation: The conversion of current code into previous syntax.

2. Compatibility: Ensures code runs in browsers that don't support new features.

**3. Customization**: Allows the use of plugins and presets to adjust the transformation process to meet specific project needs.

For example:

const greet = (name) => `Hello, ${name}!`;

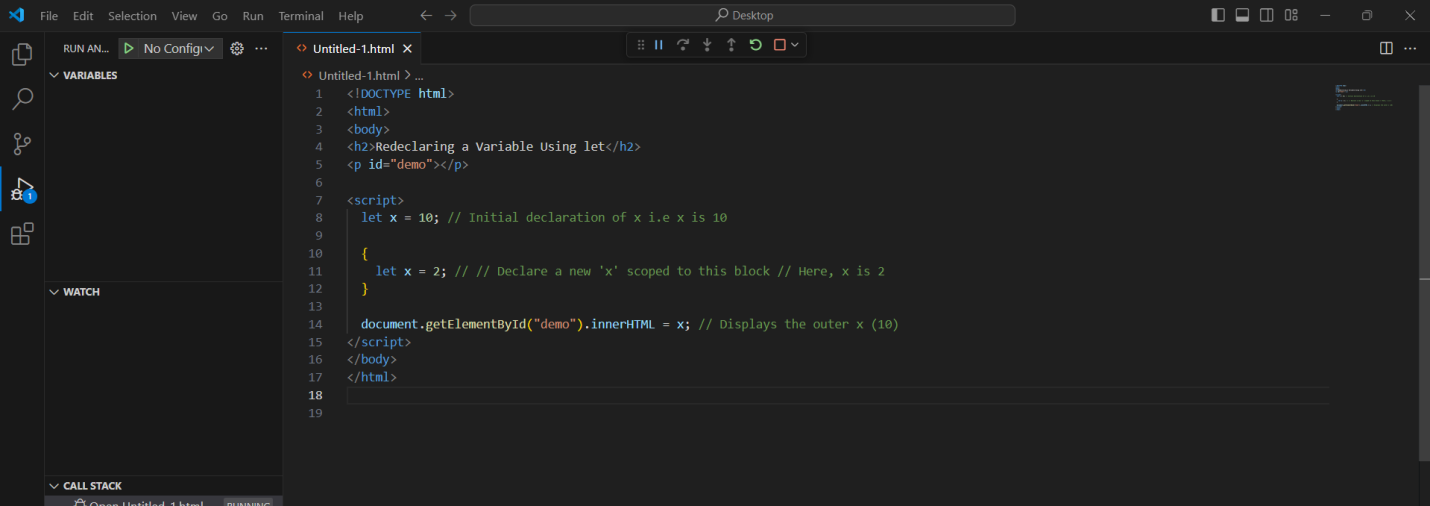
Babel converts to it :

var greet = function(name) {

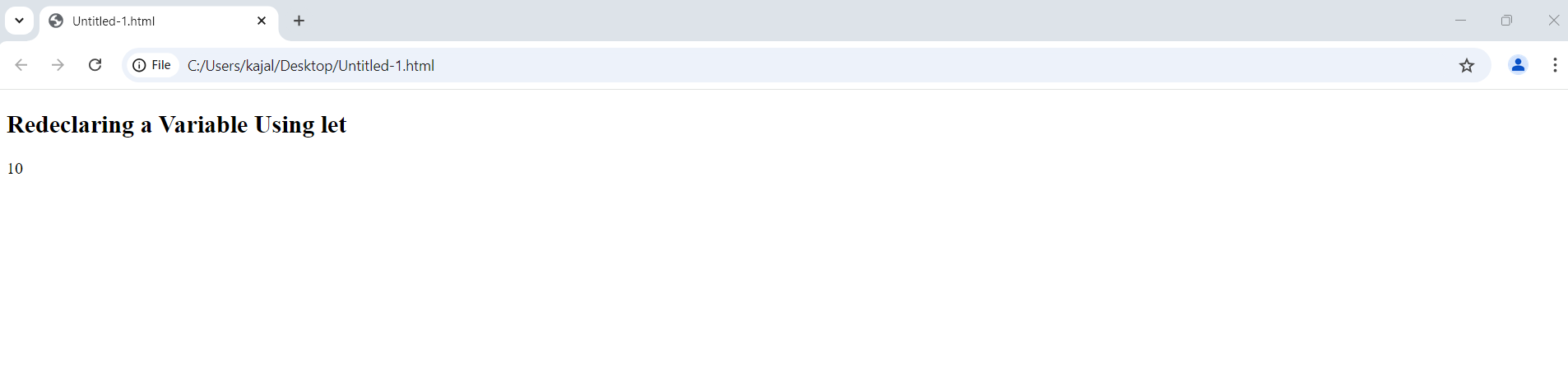
return "Hello, " + name + "!";

};

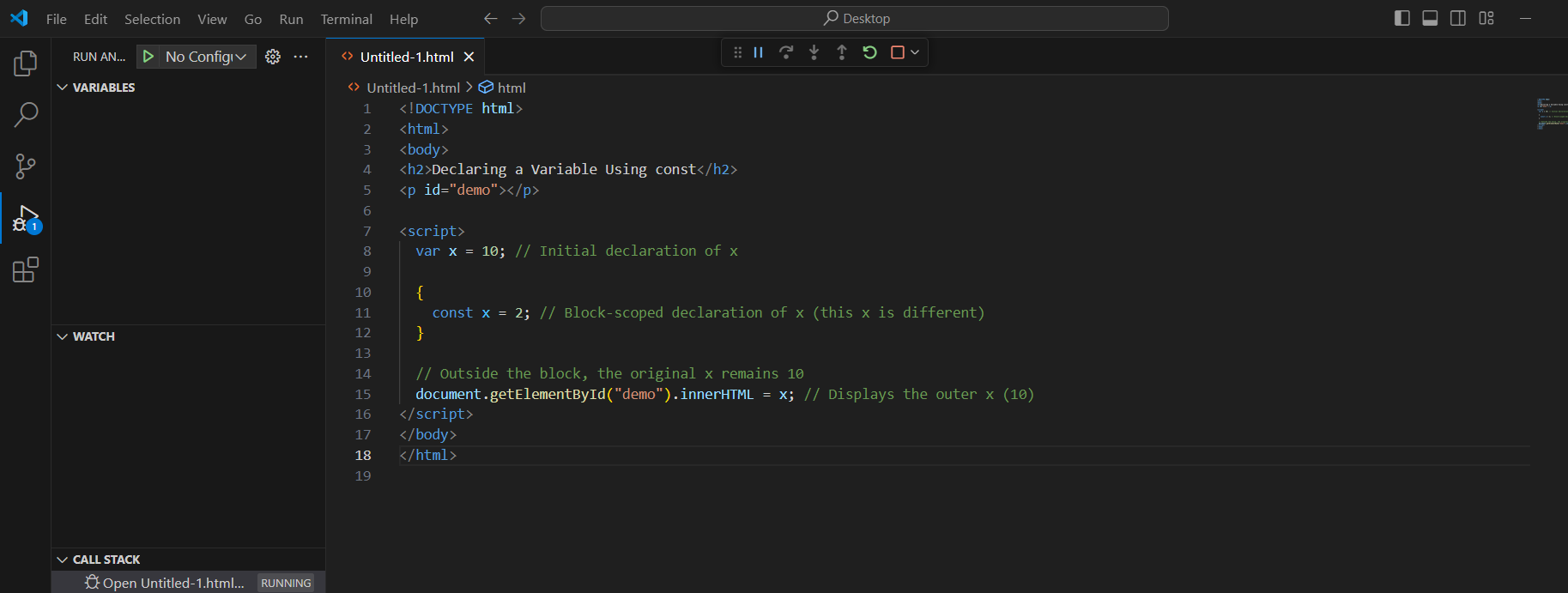
Part 2: JavaScript let

­­­

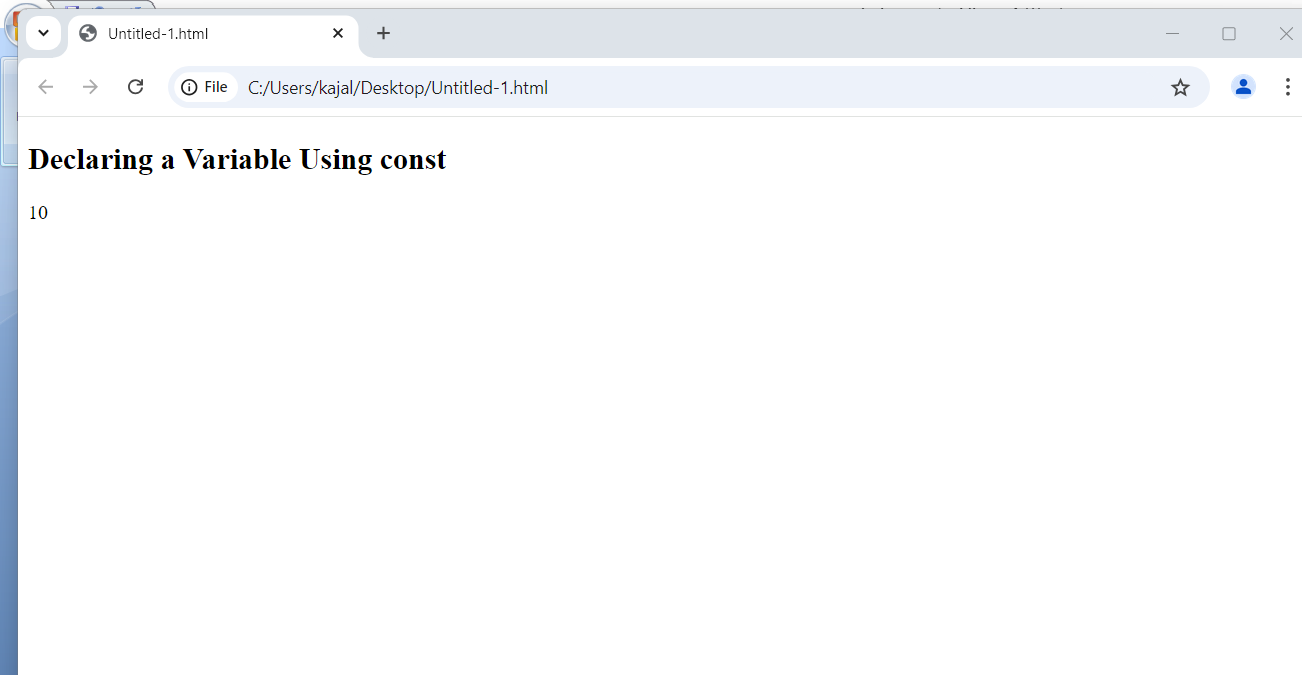
OUTPUT:



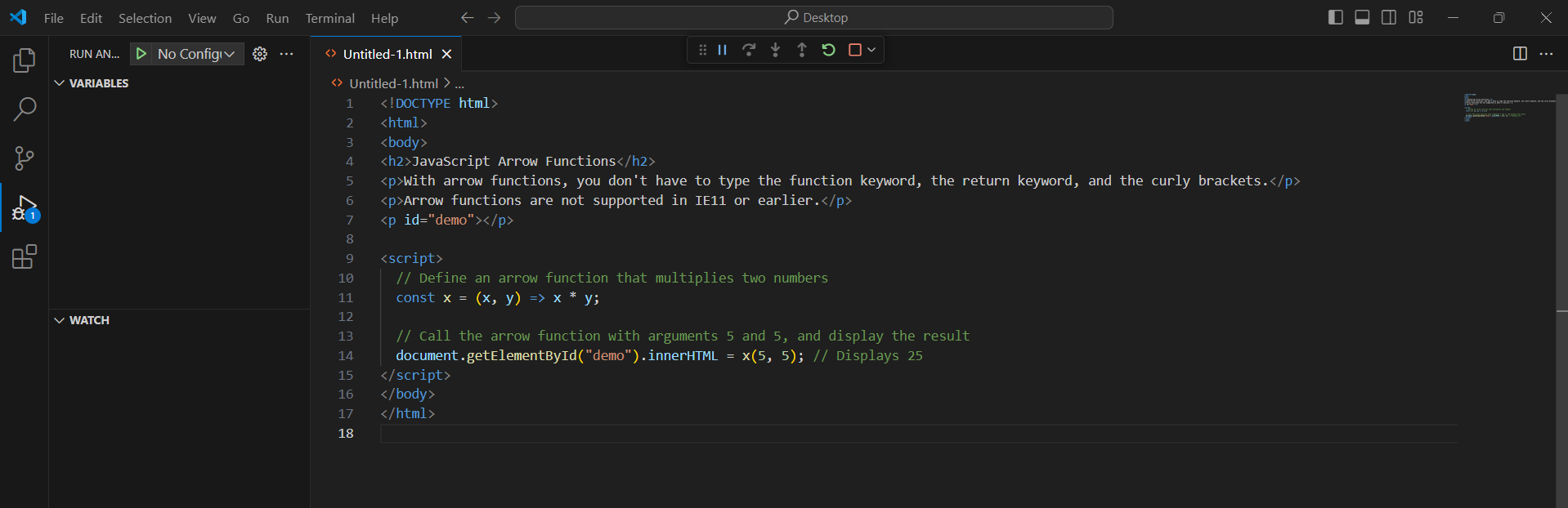
JavaScript const



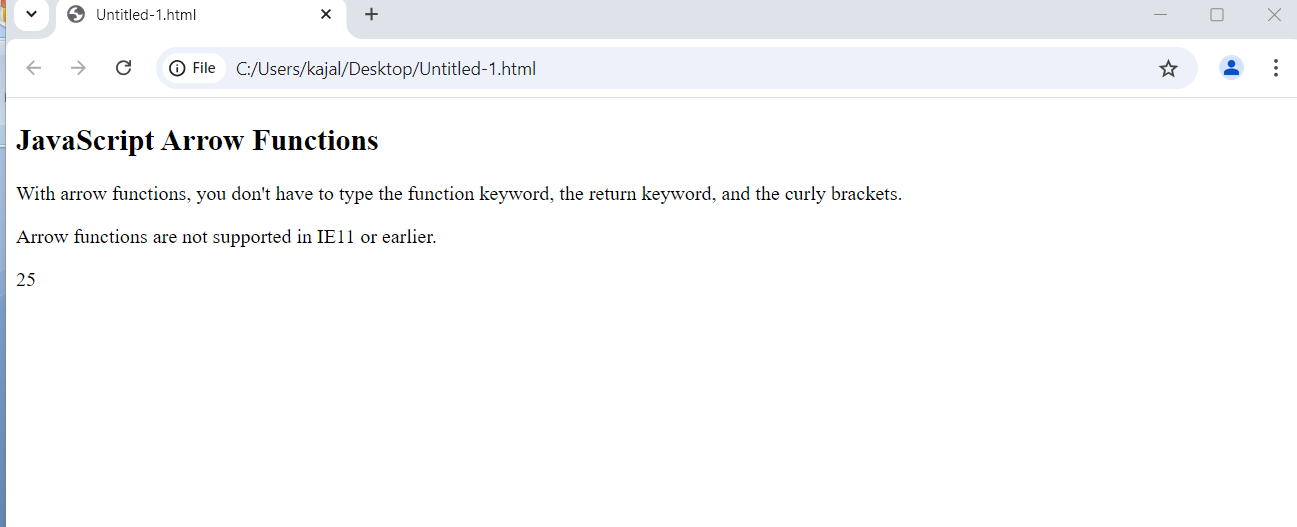
OUTPUT:



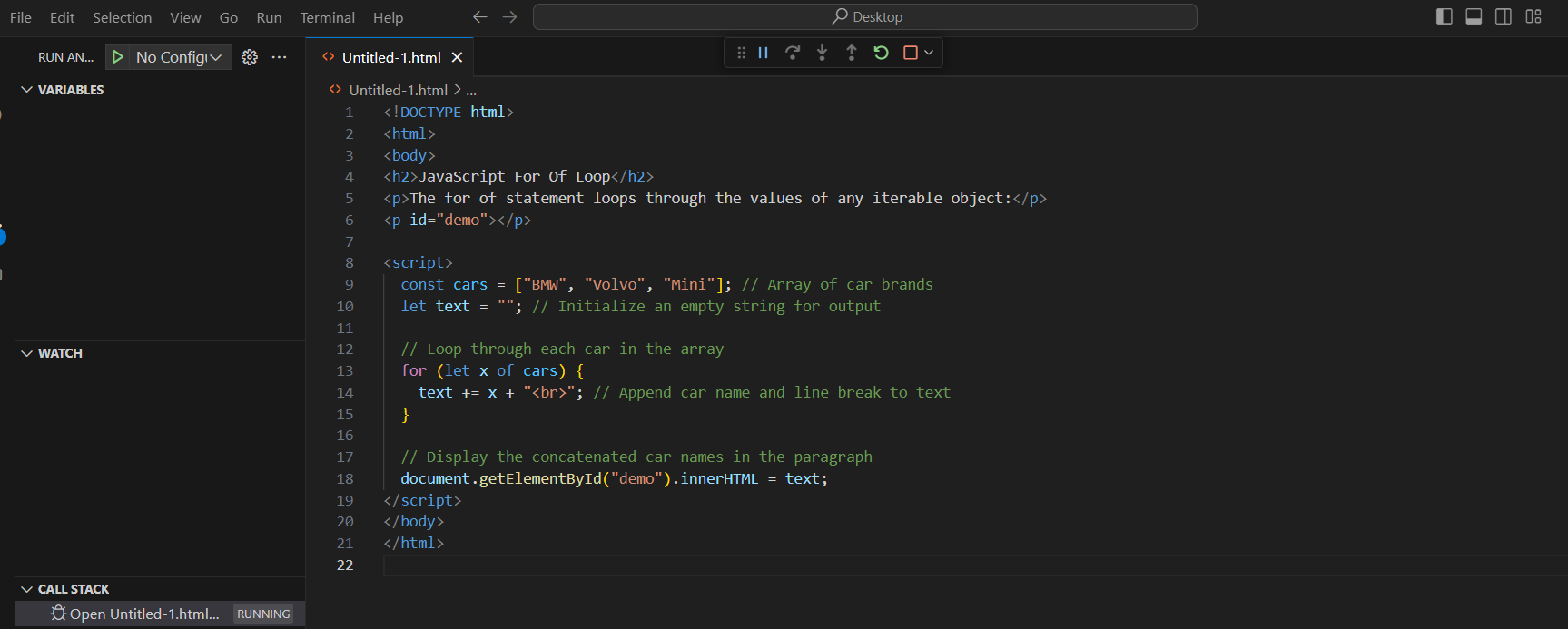
Arrow Functions:



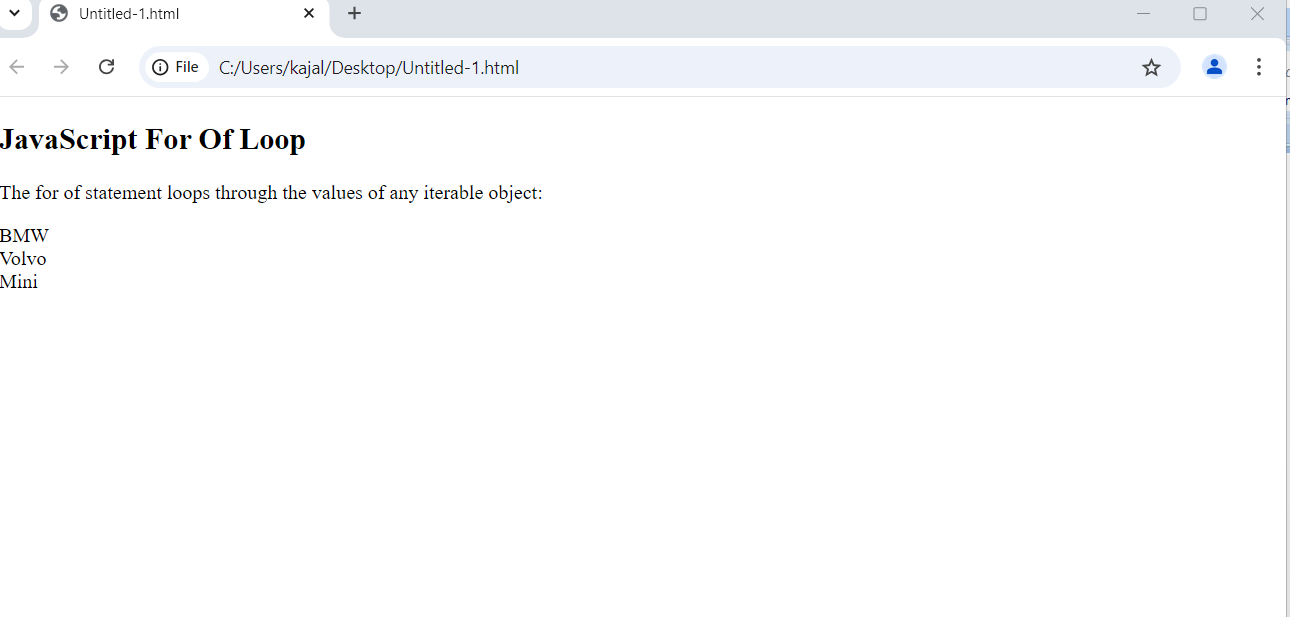
OUTPUT :



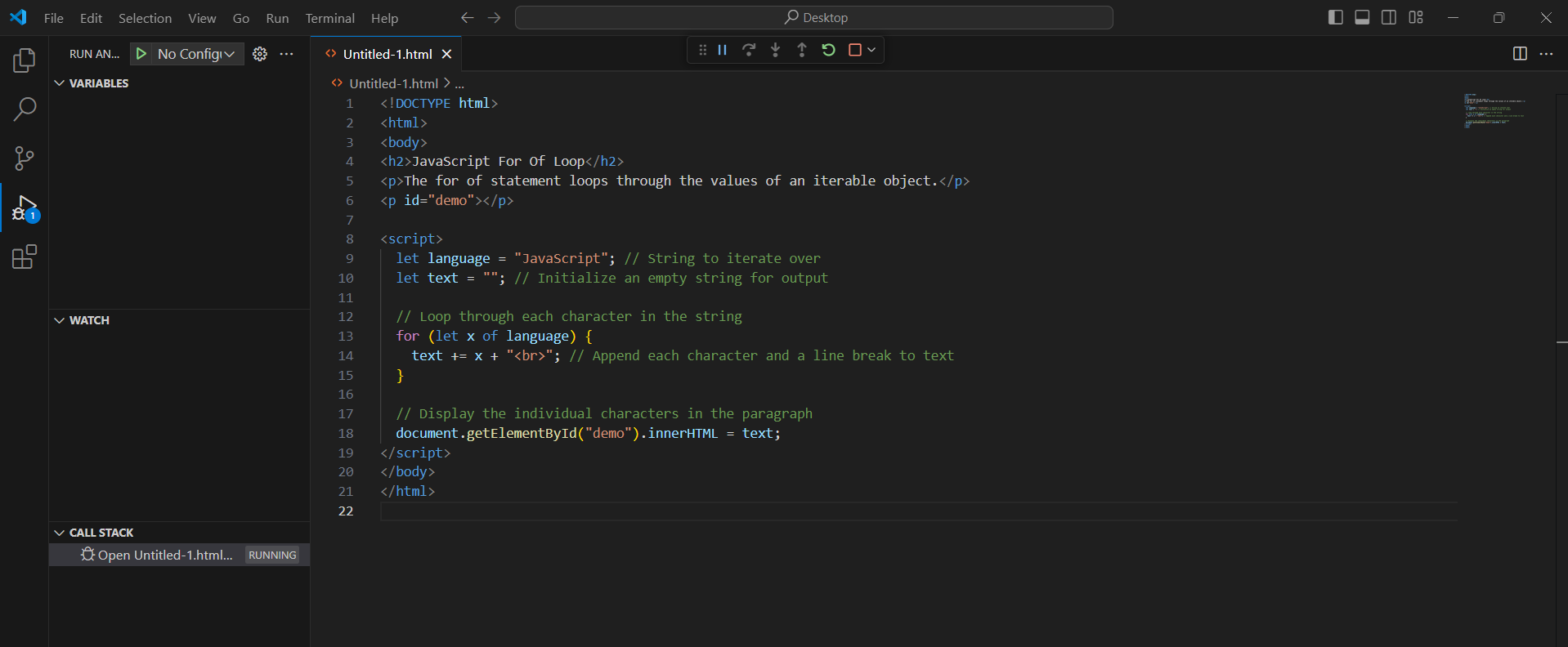
The For/Of Loop:



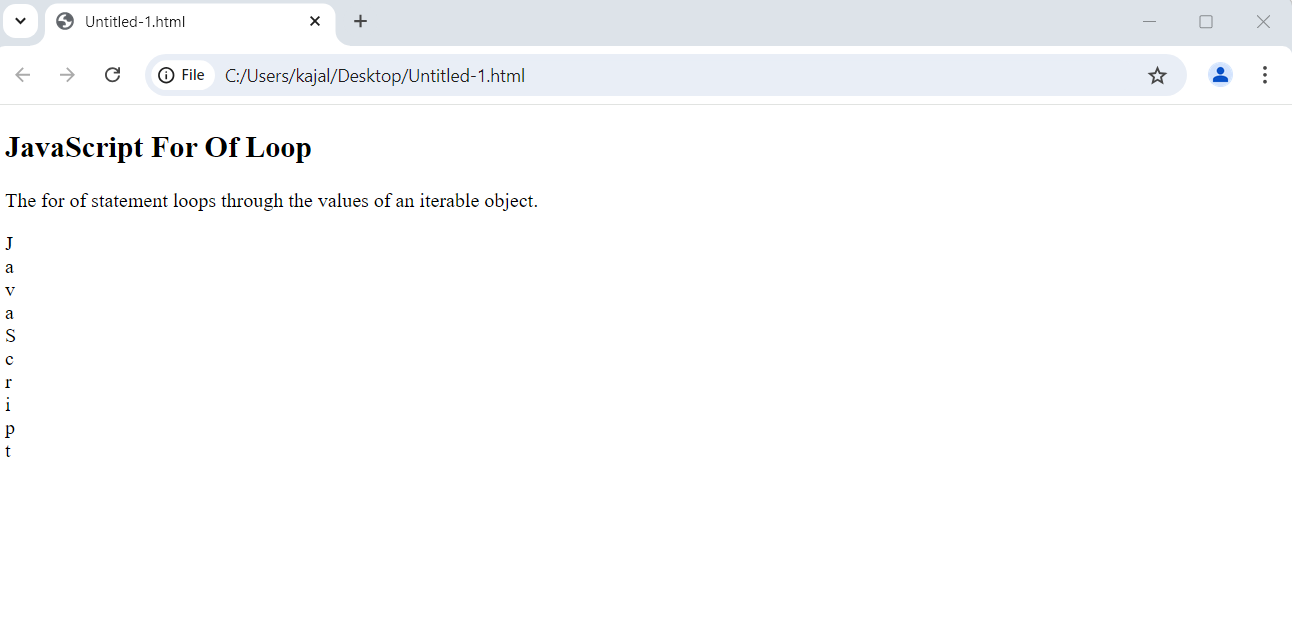
OUTPUT :



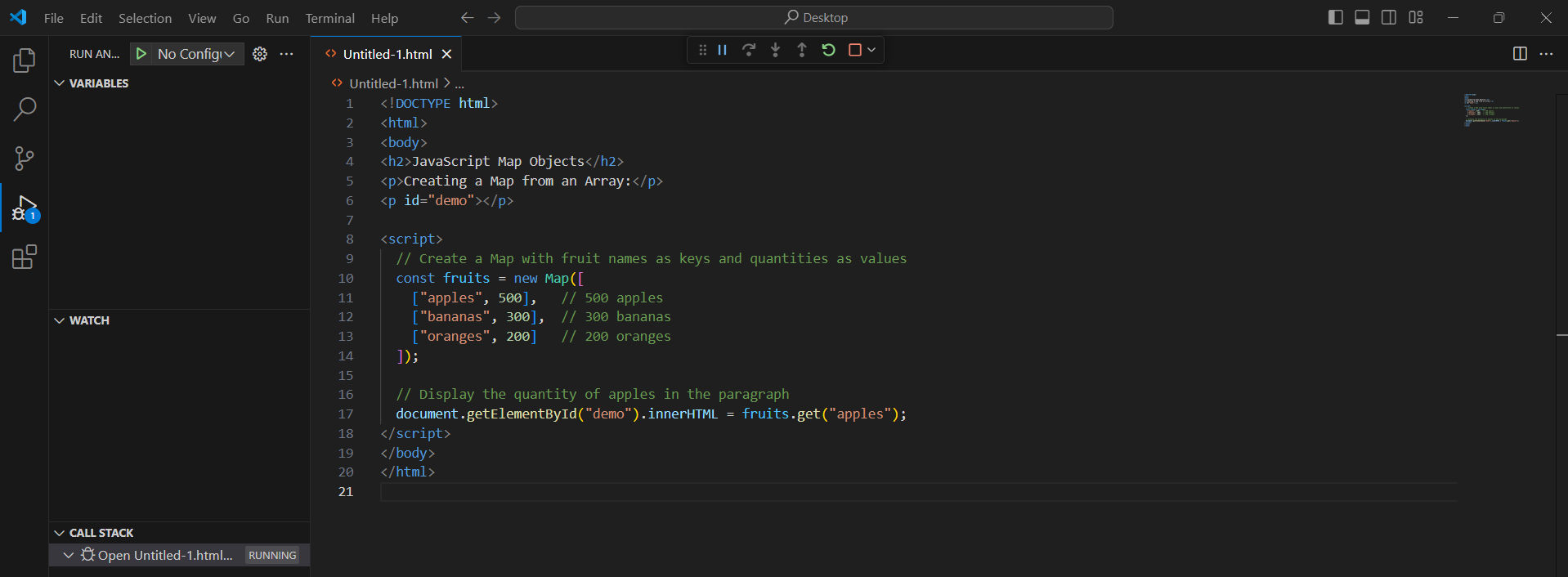
Looping over a String:



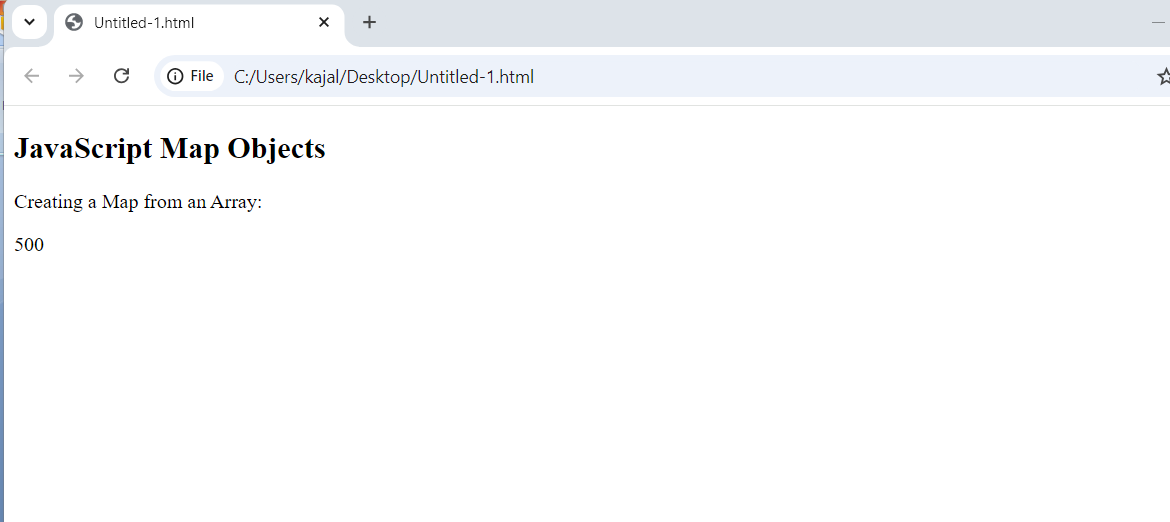
OUTPUT :



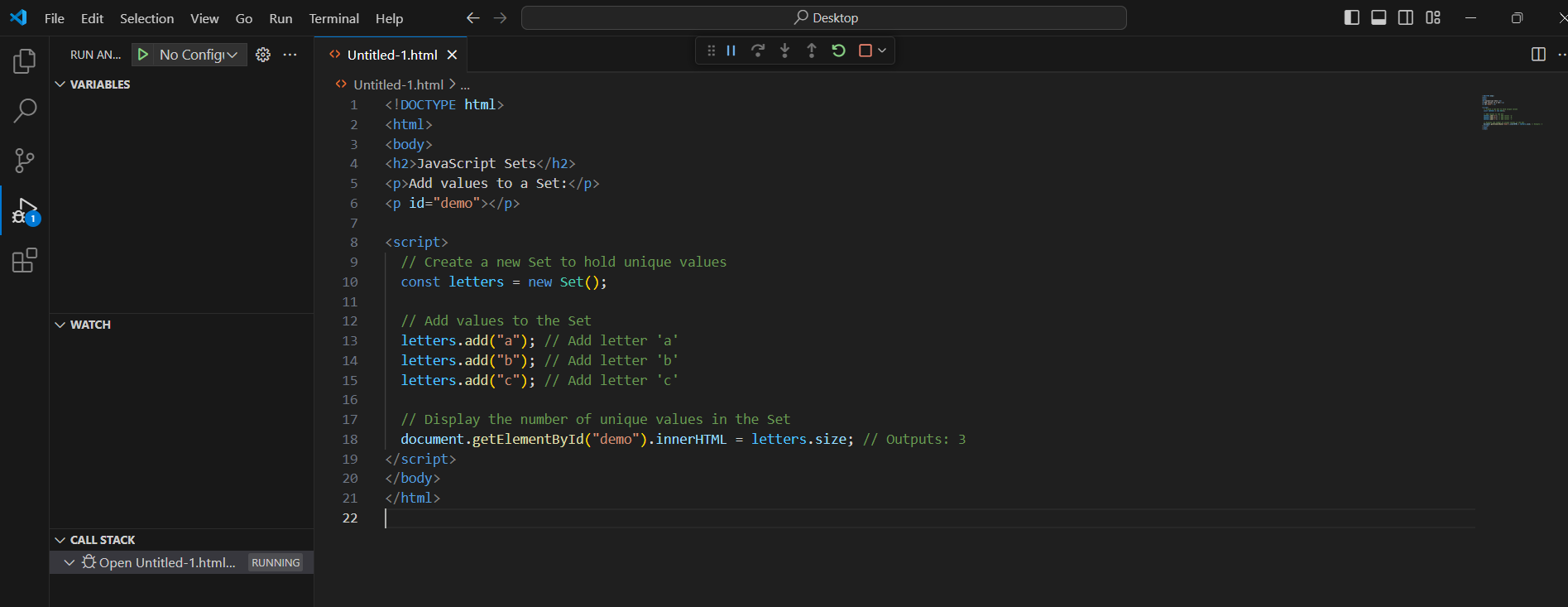
JavaScript Maps:



OUTPUT



SET :



OUTPUT :

