**Hoisting**

JavaScript only hoists declarations, not initializations.

**Hoisting** is a concept that enables us to extract values of variables and functions even before initializing/assigning value without getting errors and this happens during the 1st phase (memory creation phase) of the Execution Context.

let a; // Declaration

a = 100; // Assignment

console.log(a); // Usage

In Javascript assigning a value to an undeclared variable implicitly creates it as a global variable when the assignment is executed. This means that all undeclared variables are global variables.

<html>

    <body>

        <h2>Hoisting</h2>

        <script>

        function codeHoist()

        {

        a = 10;

        let b = 50;

    }

    codeHoist();

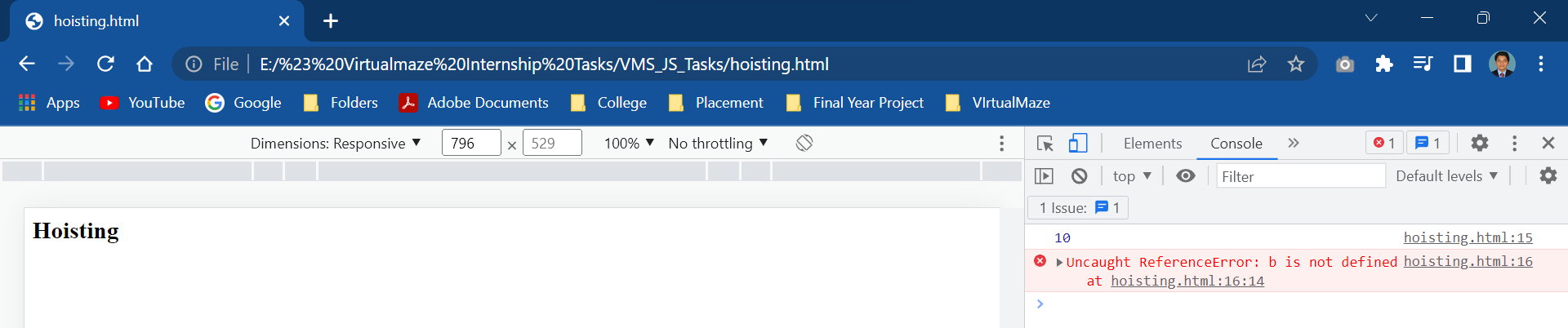
    console.log(a); // 10

    console.log(b); // ReferenceError : b is not defined

        </script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

    // var code (global)

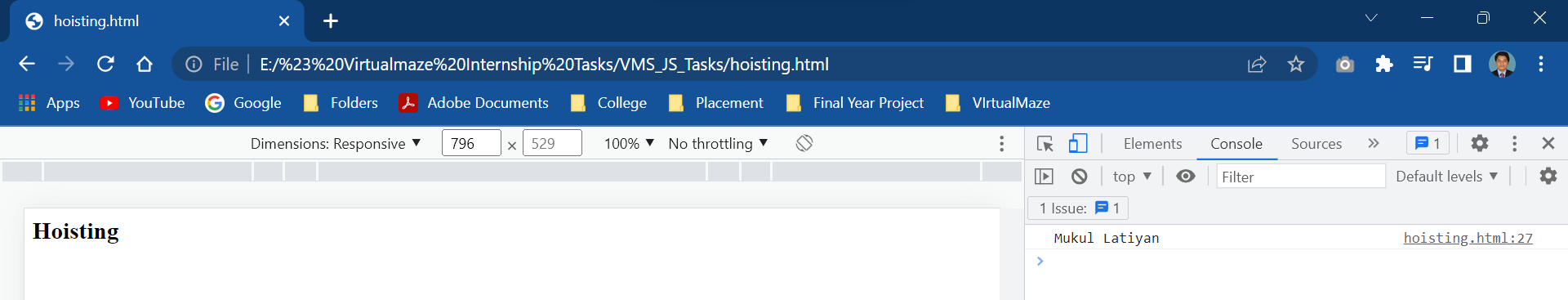
    console.log(name); // undefined

    var name = 'Mukul Latiyan';

</script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

            var name;

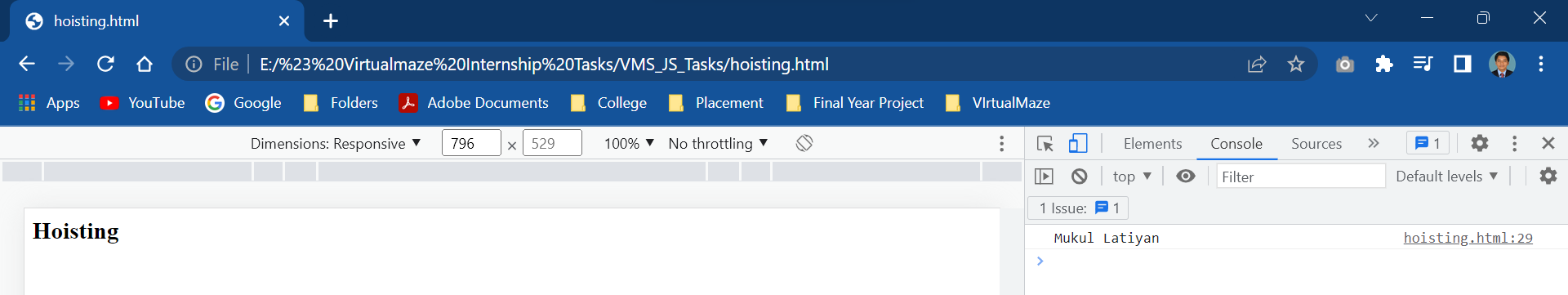
            console.log(name); // undefined

            name = 'Mukul Latiyan';

        </script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

            // Function scoped

            function fun(){

                console.log(name);

                var name = 'Mukul Latiyan';

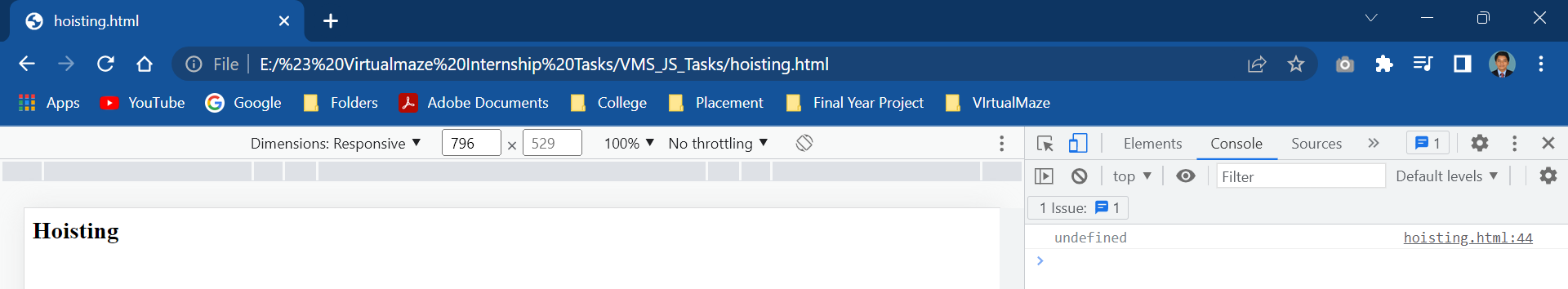
            }

            fun(); // Undefined

        </script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

            // in order to avoid it

            function fun(){

                var name = 'Mukul Latiyan';

                console.log(name); // Mukul Latiyan

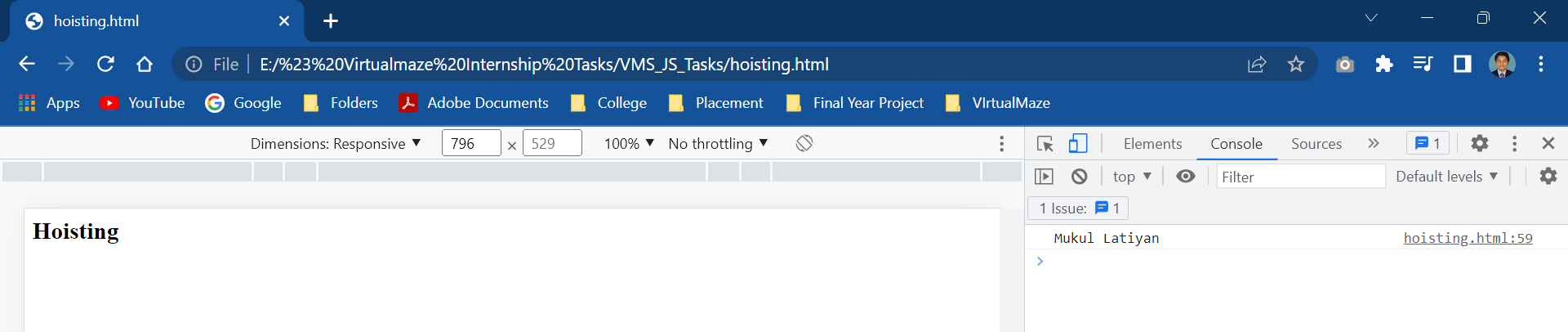
            }

            fun();

        </script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

            //let example(global)

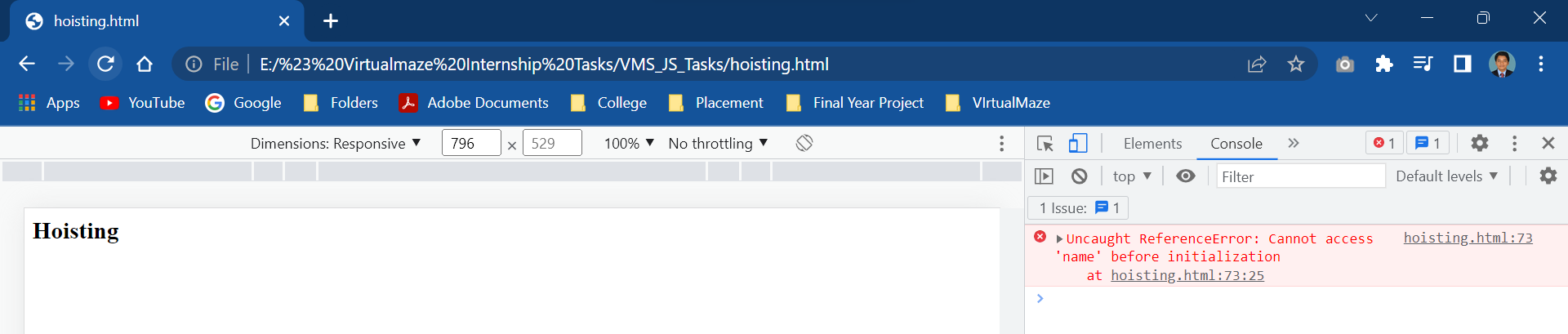
            console.log(name);

            let name='Mukul Latiyan';

        </script>

    </body>

</html>



<html>

    <body>

        <h2>Hoisting</h2>

        <script>

            fun(); // Calling before declaration

            function fun(){ // Declaring

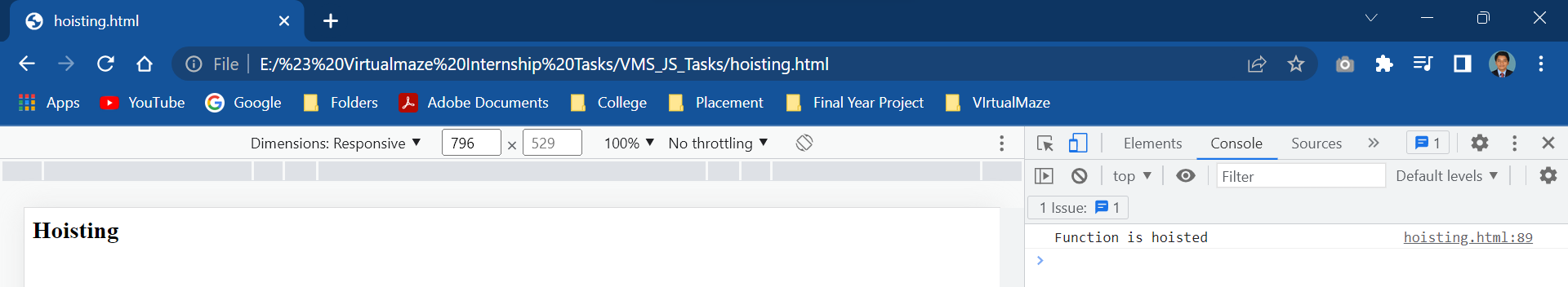
                console.log("Function is hoisted");

            }

        </script>

    </body>

</html>



**Scope**

**Global Scope**

A variable declared outside of a function is considered a global scope variable.

<html>

    <body>

        <h2>Global Scope</h2>

        <script>

        let a = "hello";

        function glo() {

            a = 3;

        }

        console.log(a); // before the function call

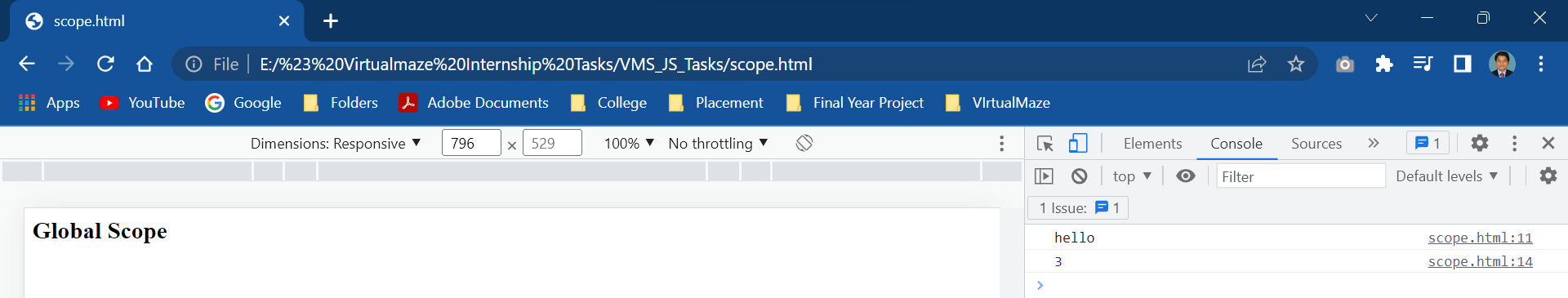
        glo();

        console.log(a); //after the function call

        </script>

    </body>

</html>



**Local Scope**

A variable can also have a local scope, i.e it can only be accessed within a function.

<html>

    <body>

        <h2>Local Scope</h2>

        <script>

        let a = "hello";

        function loc() {

            let b = "World"

            console.log(a + b);

        }

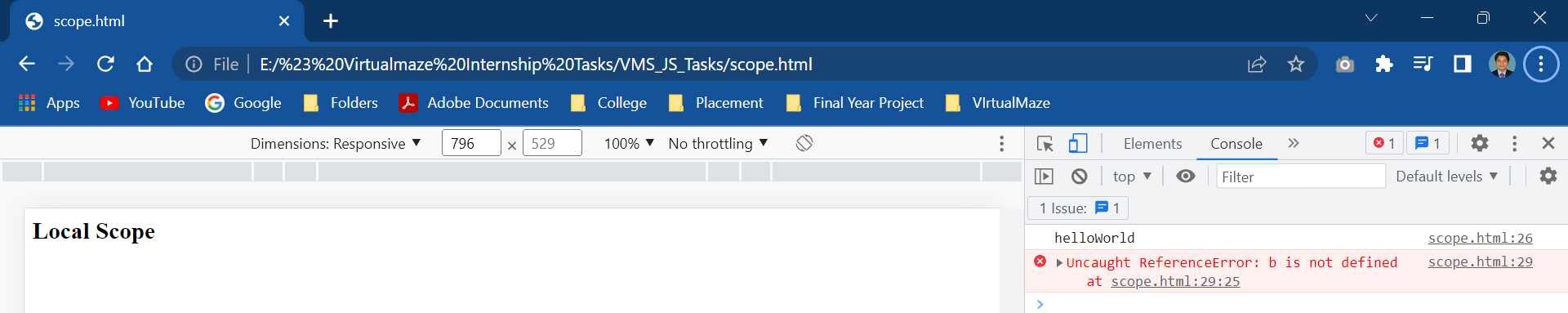
        loc();

        console.log(a + b);

        </script>

    </body>

</html>



**let is Block Scoped**

<html>

    <body>

        <h2>Block Scope</h2>

        <script>

        let a = 'Hello';

        function blo()

        {

            let b = 'World';

            console.log(a + ' ' + b);

            if (b == 'World')

            {

                let c = 'hello'; // block-scoped variable

                console.log(a + ' ' + b + ' ' + c);

            }

            console.log(a + ' ' + b + ' ' + c); // variable c cannot be access here

        }

        blo();

        </script>

    </body>

</html>

