1. Write program using this keyword when we click on button, it should be disappear.

<html>

<body>

<h1>Q1 Using 'this' keyword and also making text disappear</h1>

<p id="demo"></p>

<button onclick="fun()" id="btn">Click me!</button>

<script>

function fun() {

document.getElementById("demo").style.display = "none";

}

const person = {

firstName: "MANSI",

lastName: "KAPIL",

id: 5566,

fullName: function () {

return this.firstName + " " + this.lastName;

};

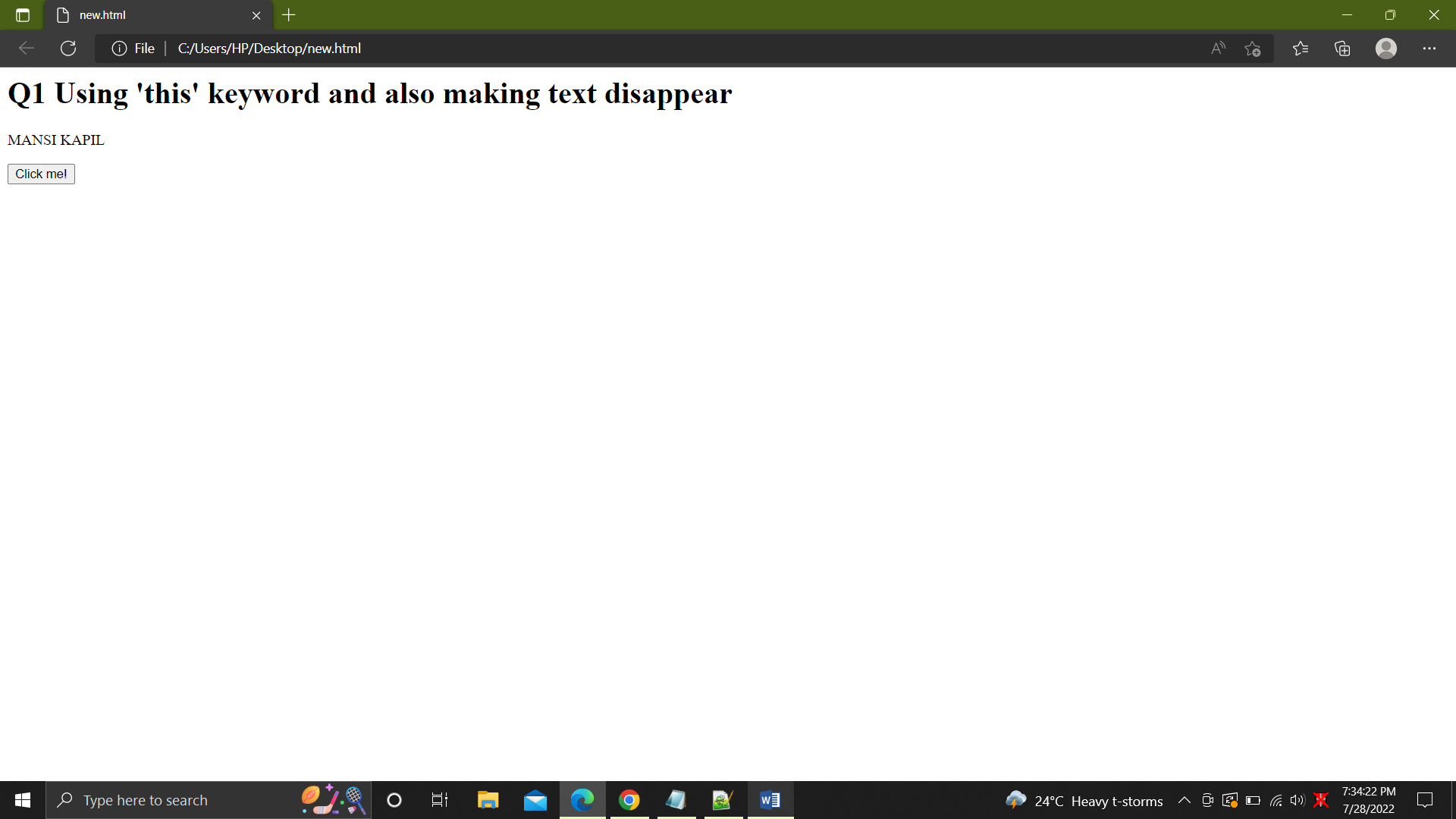
};

document.getElementById("demo").innerHTML = person.fullName();

</script>

</body>

</html>

Invoke a function using call() and apply(). Also Spot the difference between them.

2. Invoke a function using call() and apply(). Also Spot the difference between them.

<html>

<body>

<h2>Q2 call() and apply()</h2>

<p id="demo"></p>

<script>

const person = {

fullName: function (city, country) {

return (

this.firstName + " " + this.lastName + ", " + city + ", " + country

);

},

};

const person1 = {

firstName: "mansi",

lastName: "Kapil",

};

document.getElementById("demo").innerHTML = person.fullName.apply(

person1,

["orissa", "India"]

);

document.getElementById("demo").innerHTML = person.fullName.call(

person1,

"orissa",

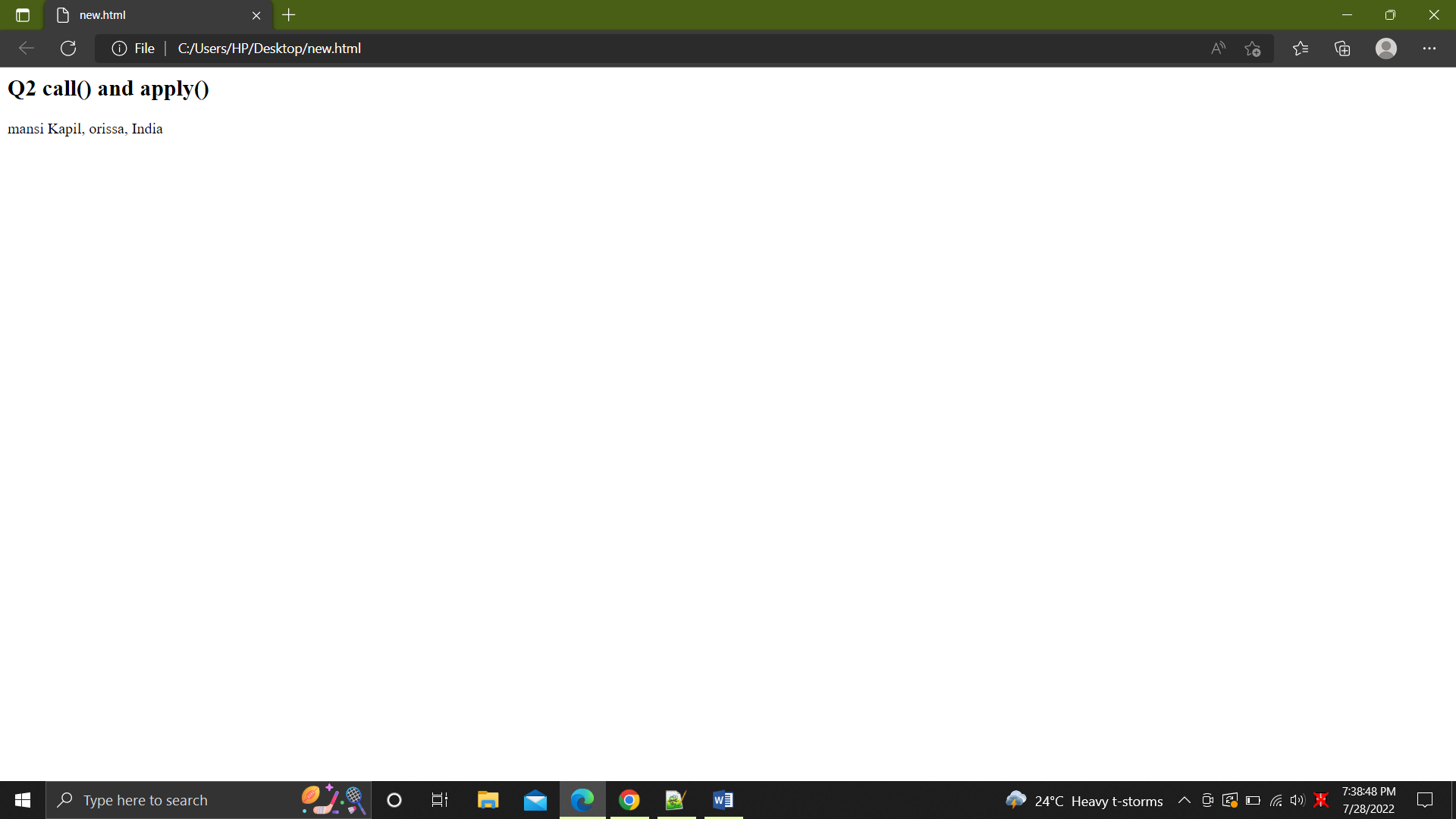
"India"

);

</script>

</body>

</html>



3. Write a program for COUNTER. Whenever we click on a "Count" button, value

should increment.

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>COUNTER</title>

</head>

<body>

<h4>Q3 Qounter button</h4>

<button id="btn">Click Here!</button>

<p>Button Clicked <span id="display">0</span> Times</p>

<script type="text/javascript">

var count = 0;

var btn = document.getElementById("btn");

var disp = document.getElementById("display");

btn.onclick = function () {

count++;

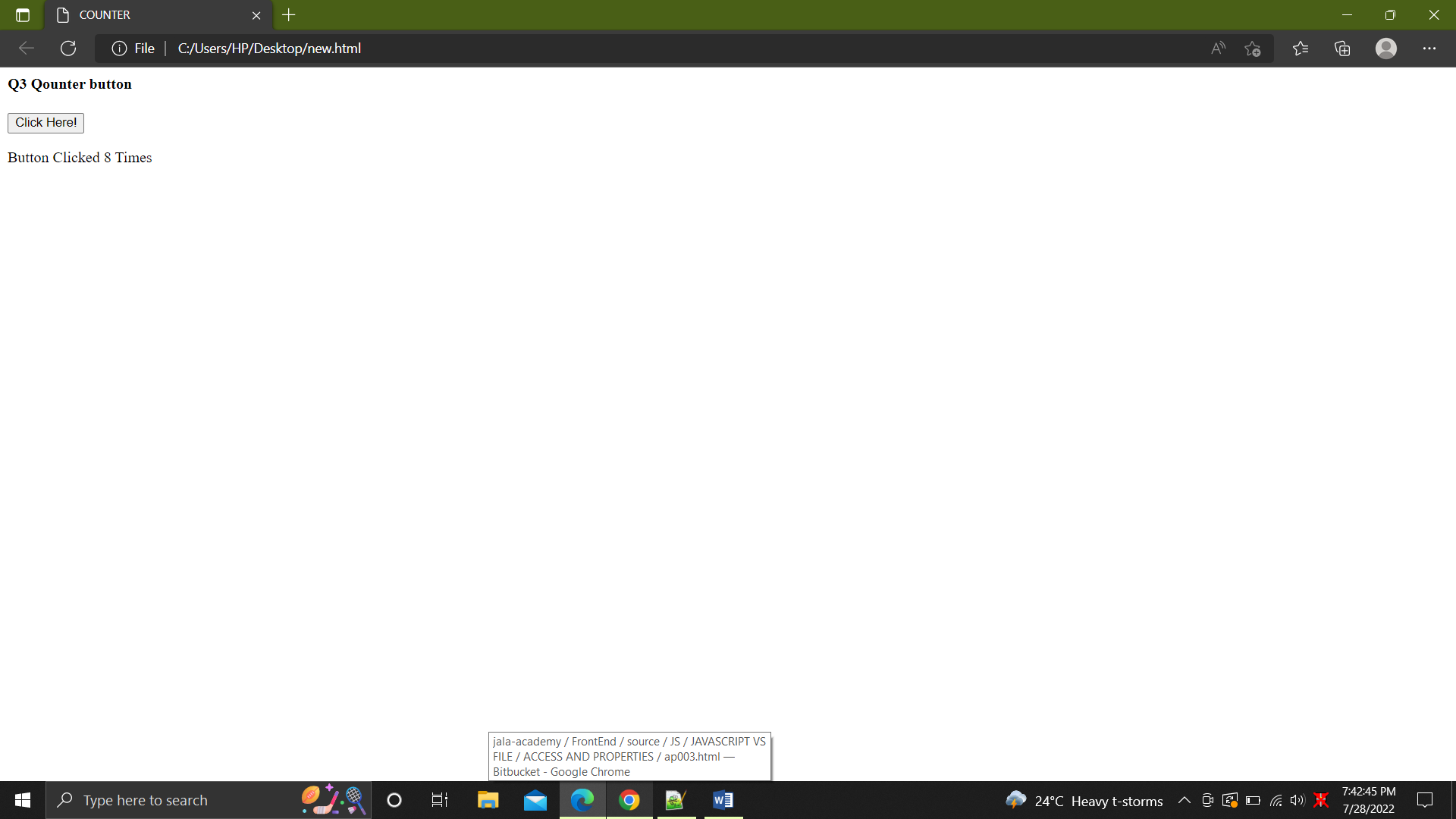
disp.innerHTML = count;

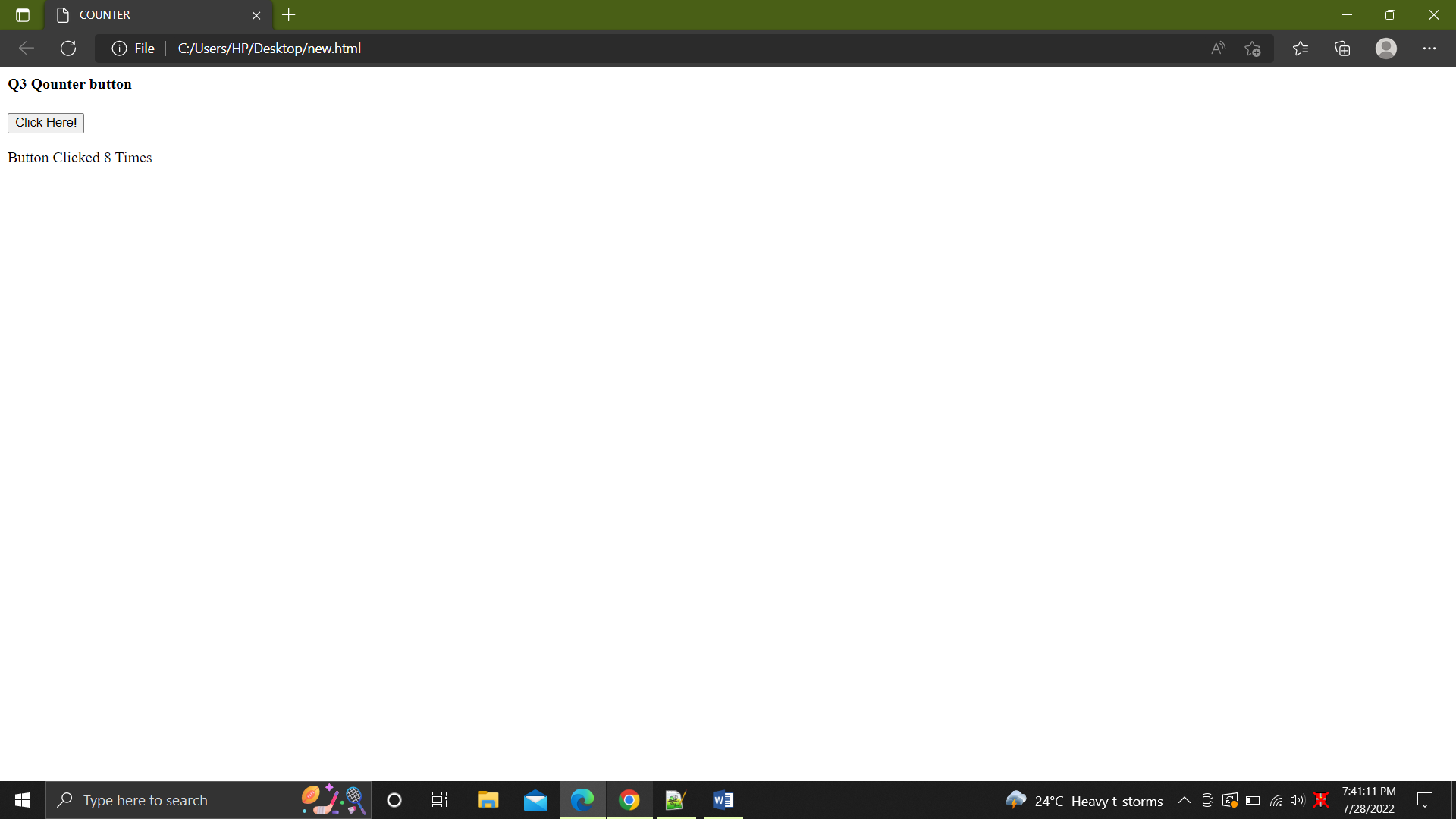
};

</script>

</body>

</html>





4. Create a Counter with the help of getter and setter accessors.

<html>

<body>

<h2>Q4 Getter and Setter Counter</h2>

<p id="demo"></p>

<script>

// Define an object

const obj = { counter: 0 };

// Define Setters and Getters

Object.defineProperty(obj, "reset", {

get: function () {

this.counter = 0;

},

});

Object.defineProperty(obj, "increment", {

get: function () {

this.counter++;

},

});

Object.defineProperty(obj, "decrement", {

get: function () {

this.counter--;

},

});

Object.defineProperty(obj, "add", {

set: function (value) {

this.counter += value;

},

});

Object.defineProperty(obj, "subtract", {

set: function (value) {

this.counter -= value;

},

});

obj.reset;

obj.add = 70;

obj.subtract = 20;

obj.increment;

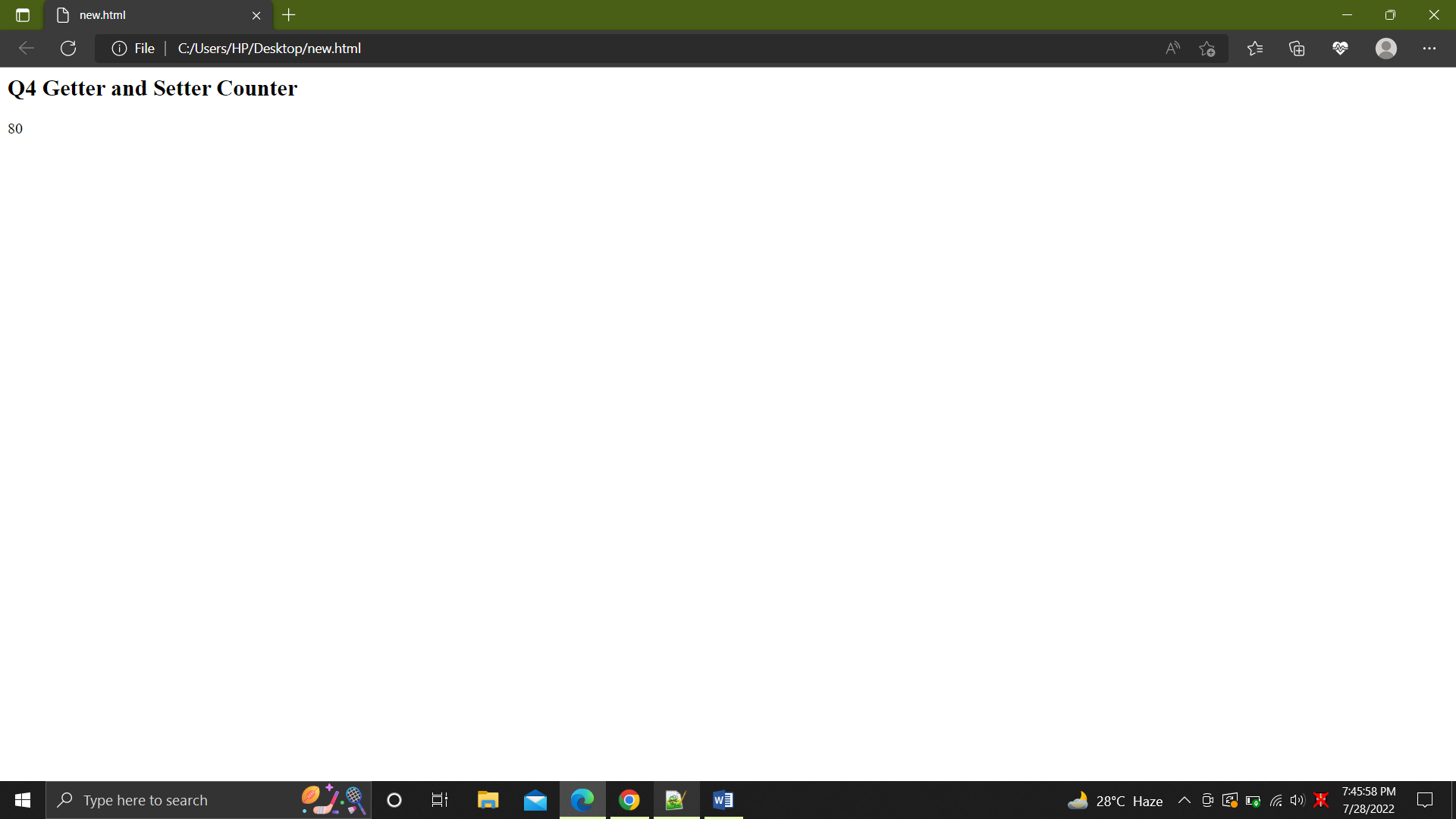
obj.decrement;

document.getElementById("demo").innerHTML = obj.counter;

</script>

</body>

</html>



5. Add some more properties to an existing function using object prototypes. <html>

<body>

<h2>Q5 Adding properties with object prototype</h2>

<p id="demo"></p>

<script>

function Person(first, last, age, eye) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eye;

}

Person.prototype.nationality = "Indian";

const me = new Person("Mansi", "Kapil", 20, "blue");

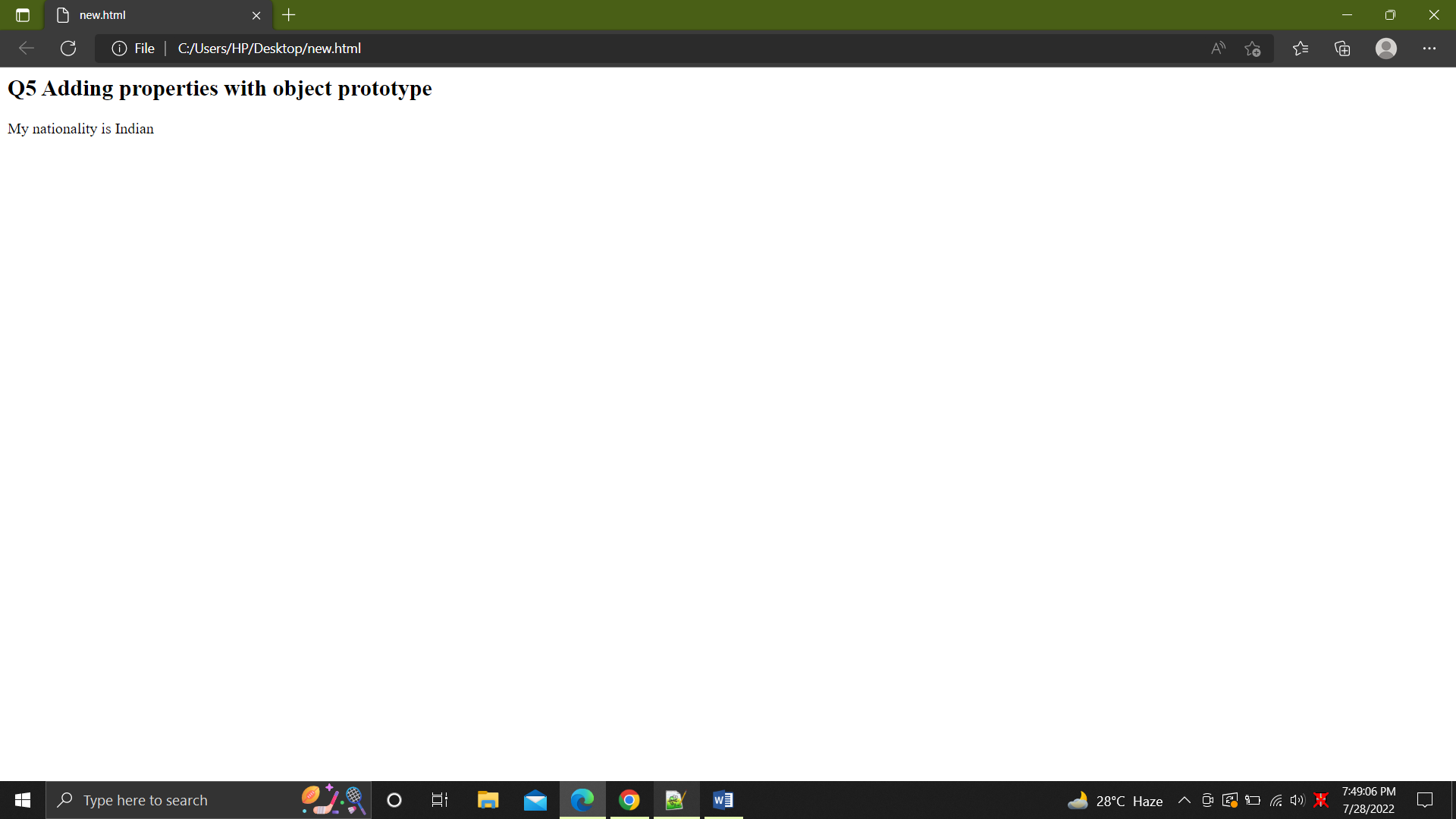
document.getElementById("demo").innerHTML =

"My nationality is " + me.nationality;

</script>

</body>

</html>



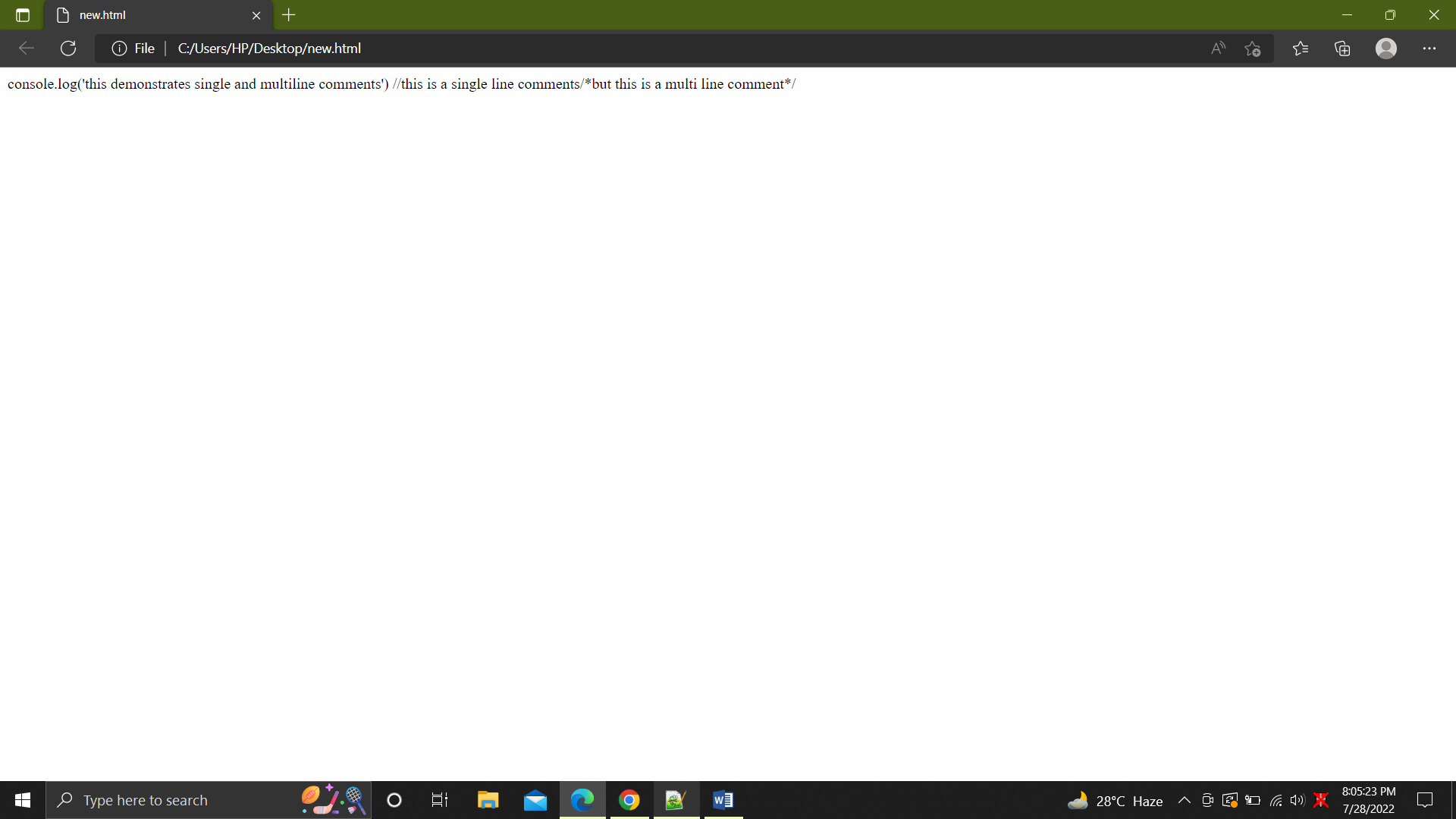
1. Write a program which consist of single line and multiline comments.

console.log('this demonstrates single and multiline comments')

//this is a single line comments/\*but this

is a multi line

comment\*/



2.

<html>

<body>

<h2>JavaScript Array Sort Reverse</h2>

<p>The reverse() method reverses the elements in an array.</p>

<p>

By combining sort() and reverse() you can sort an array in descending

order:

</p>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

const fruits = [

"Banana",

"Orange",

"Apple",

"Mango",

"Watermelon",

"Grapes",

"Papaya",

];

document.getElementById("demo1").innerHTML = fruits;

fruits.sort();

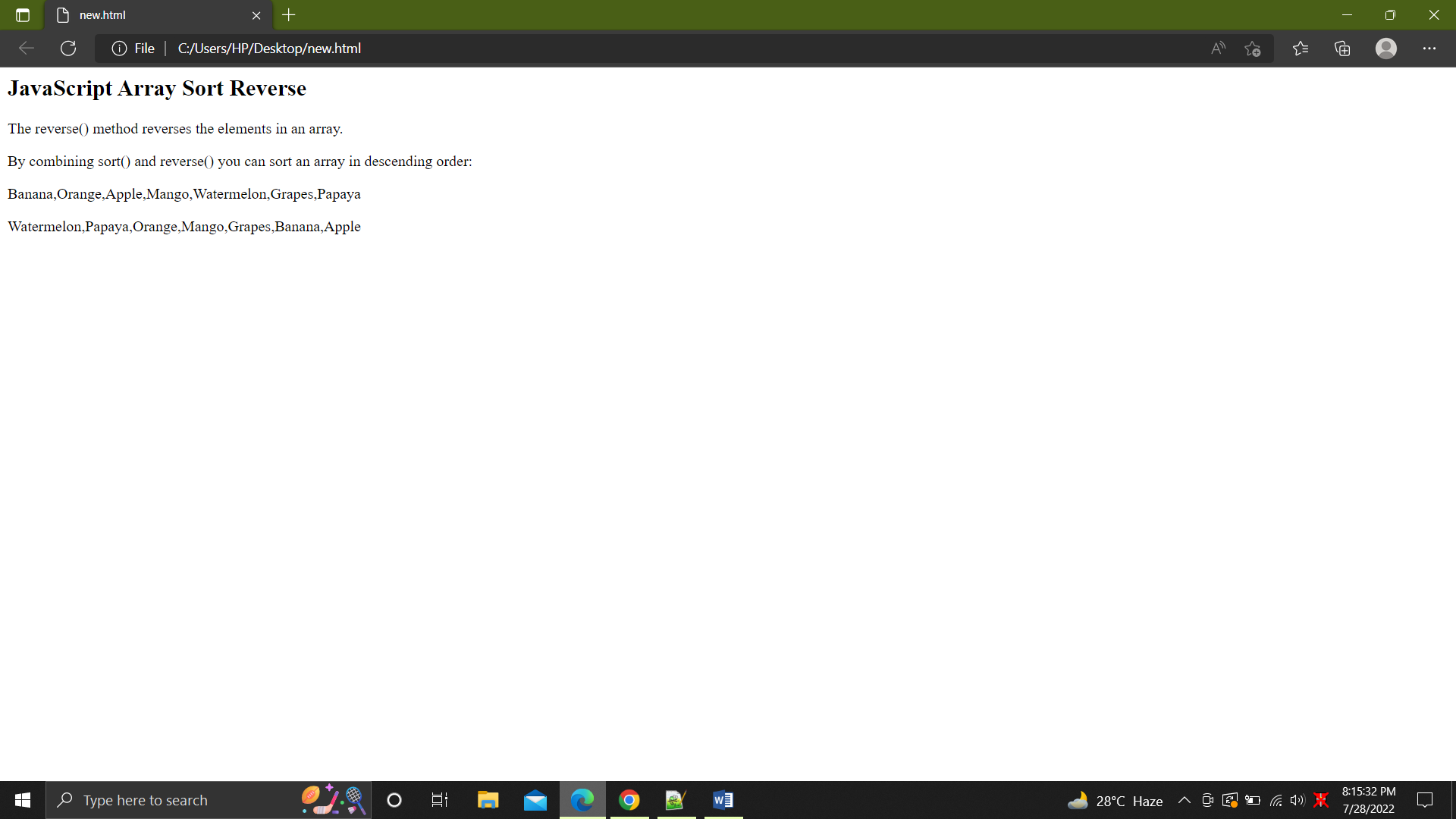
fruits.reverse();

document.getElementById("demo2").innerHTML = fruits;

</script>

</body>

</html>



3. Write a program with FOR/IN loop.

<html>

<body>

<h2>Q3 For/In Loop</h2>

<p>The for in statement loops through the properties of an object:</p>

<p id="demo"></p>

<script>

const person = { fname: "Mansi", lname: "Kapil", age: 20 };

let txt = "";

for (let x in person) {

txt += person[x] + " ";

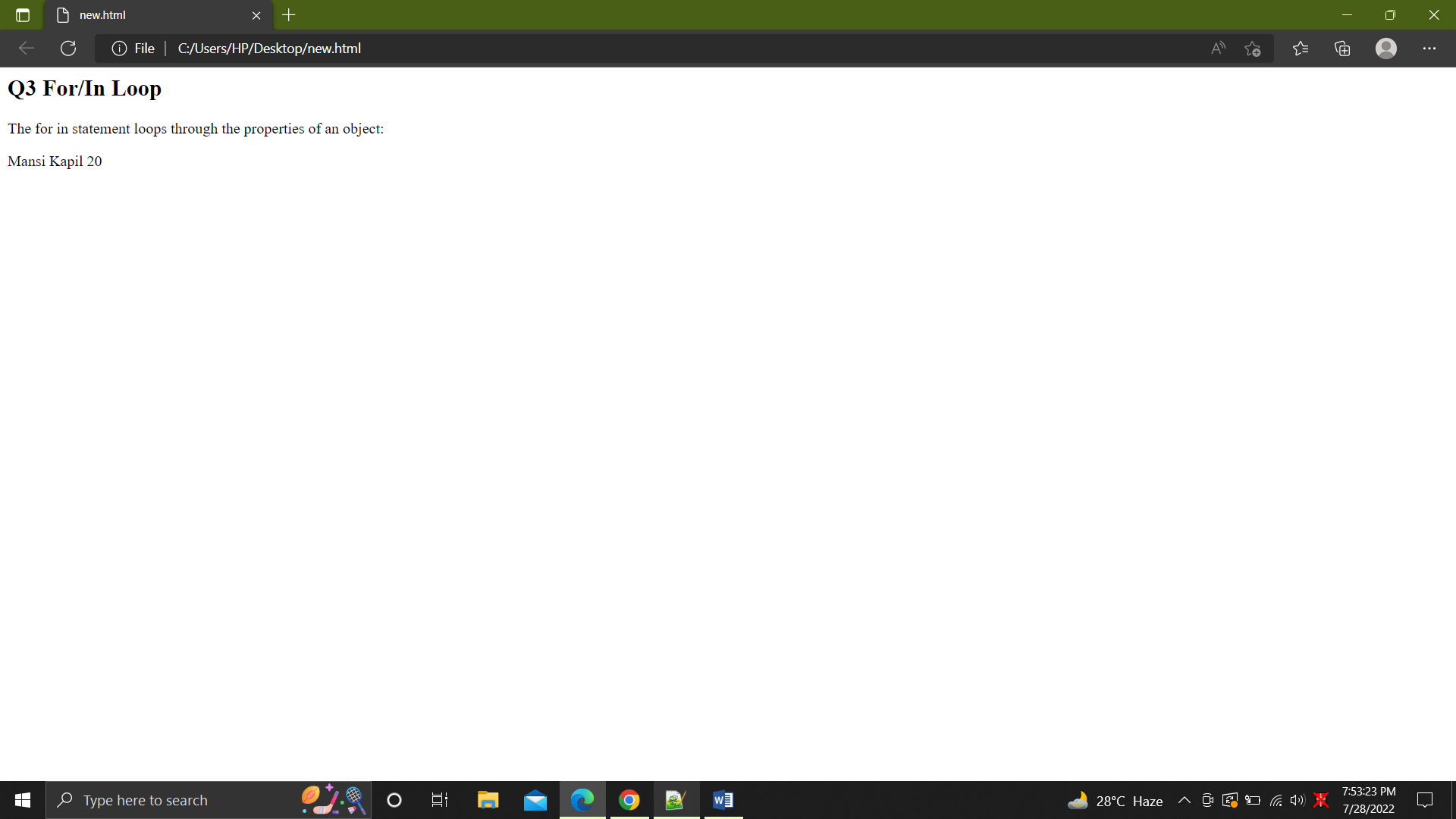
}

document.getElementById("demo").innerHTML = txt;

</script>

</body>

</html>



4. Create an object "person" with properties firstname and lastname. Display these

properties using 2 different ways of accessing.

Ans. const person = {

firstName: "Mansi",

lastName: "Kapil",

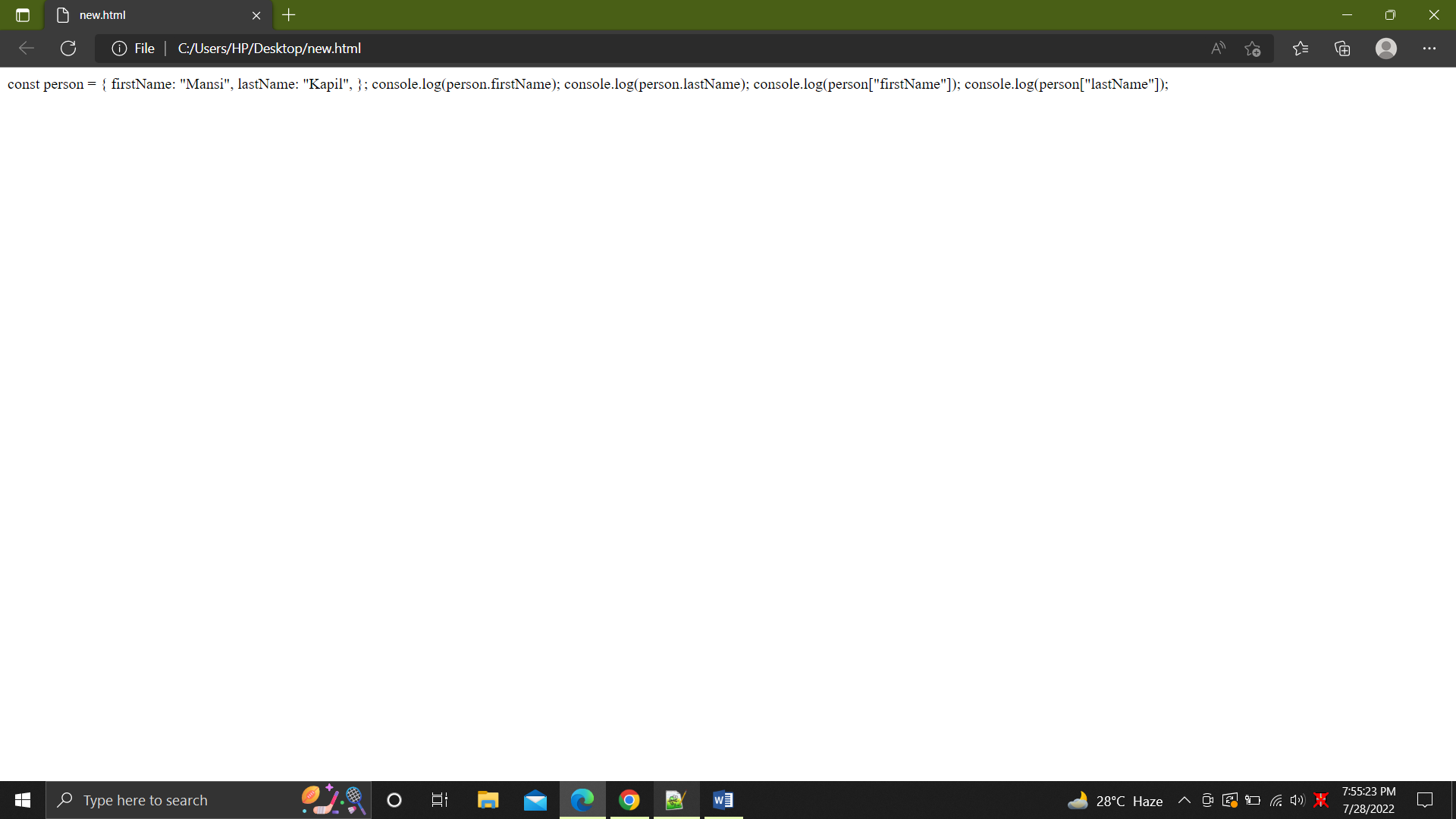
};

console.log(person.firstName);

console.log(person.lastName);

console.log(person["firstName"]);

console.log(person["lastName"]);



5. Write a program with variable hoisting(initialization first then declaration).

<html>

<body>

<h2>Q5 Hoisting</h2>

<p id="demo"></p>

<script>

var x =10;

var y;

elem = document.getElementById("demo");

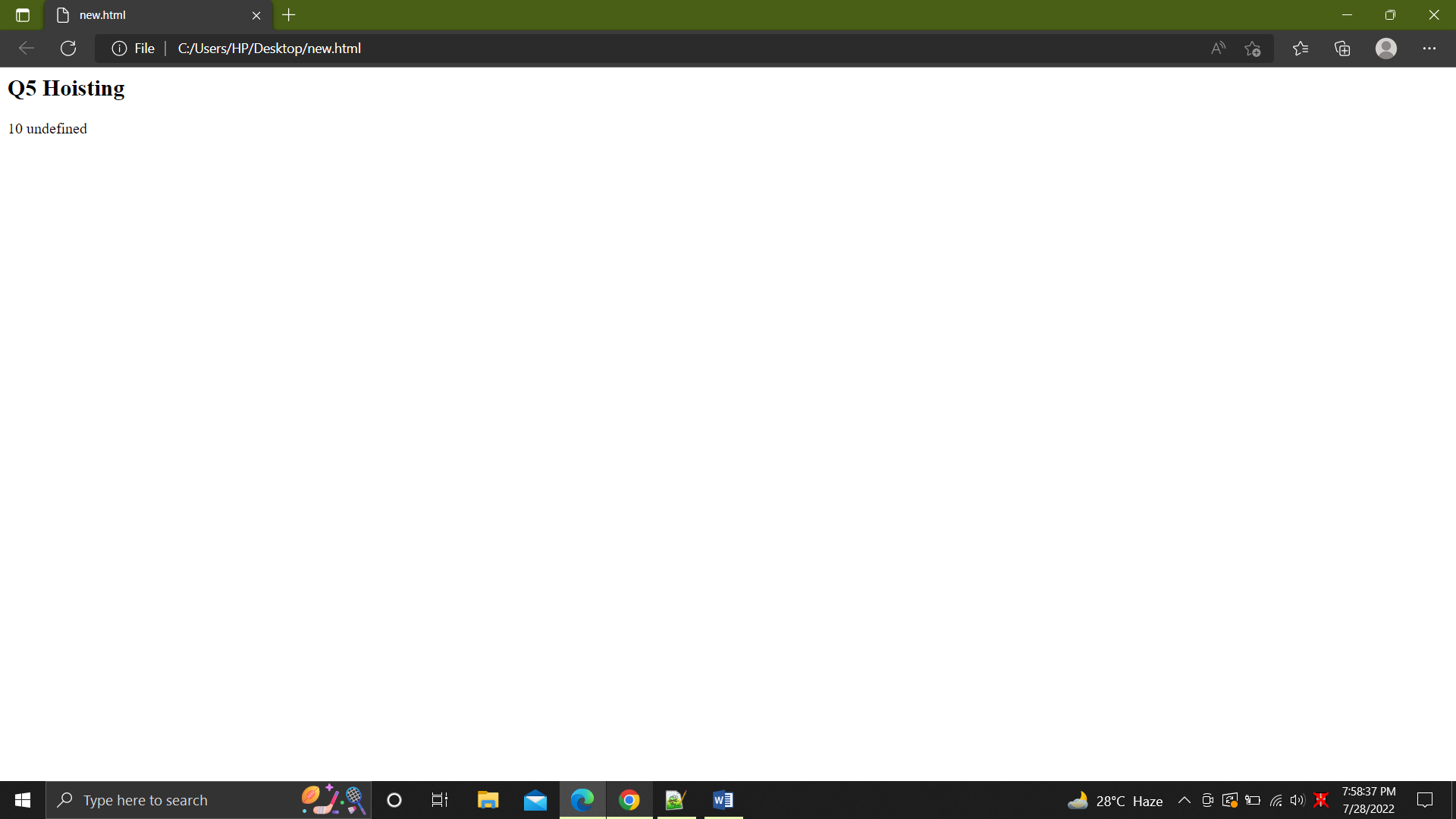
elem.innerHTML = x + " " + y;

y = 2;

</script>

</body>

</html>



6. Use strict mode in your program and understand why variable not declared causes

error.

<html>

<body>

<p>

Q6 Demonstarting that use strict will throw an error when variable is not

declared

</p>

<script>

x = 5;

myFunction();

function myFunction() {

"use strict";

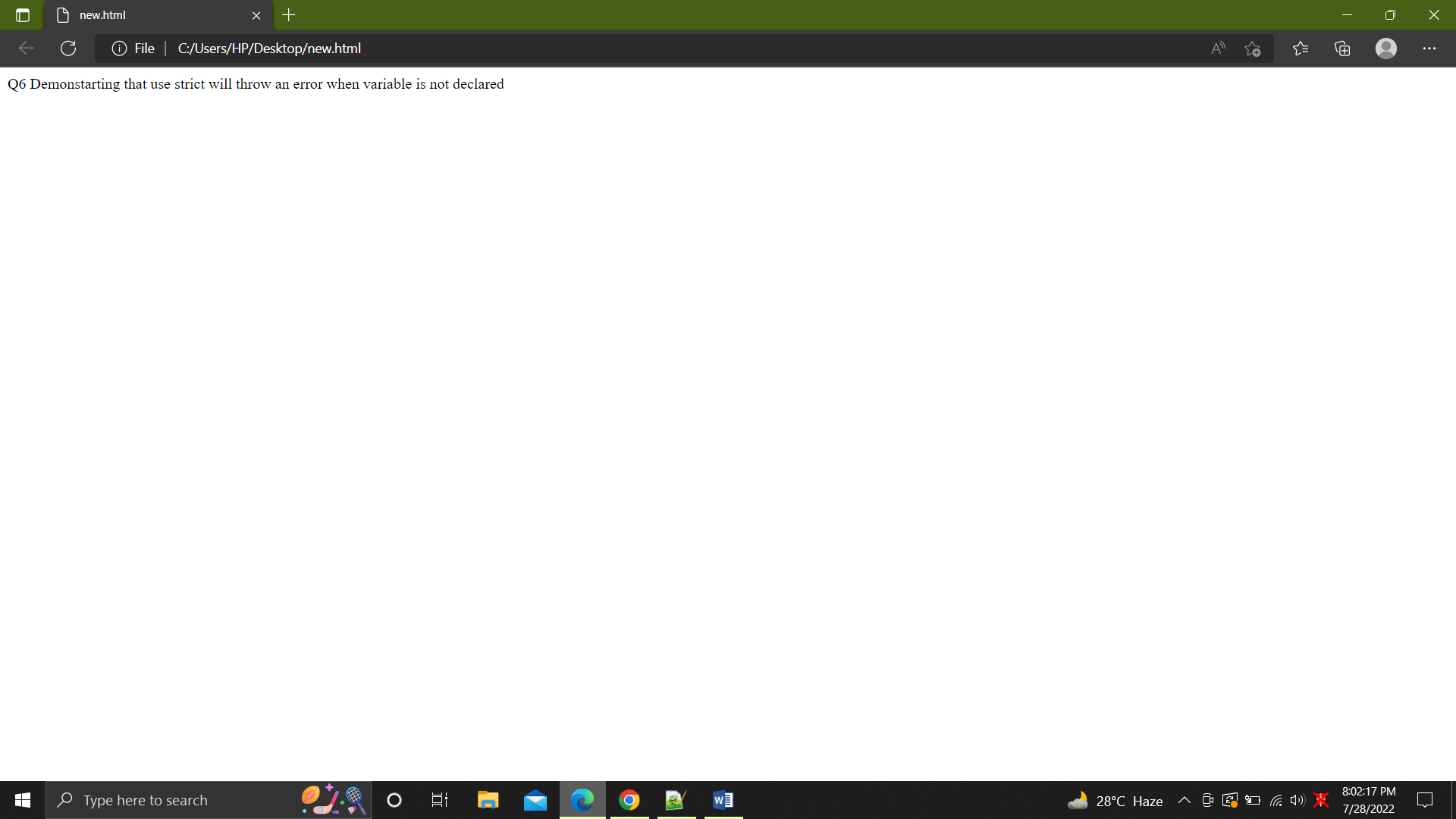
y = 9;

}

</script>

</body>

</html>



Events

1. Add an event listener to a button. When we click on it should display current date

and time.

<html>

<body>

<h2>Q1 Event listener to a button to show date</h2>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>

document.getElementById("myBtn").addEventListener("click", displayDate);

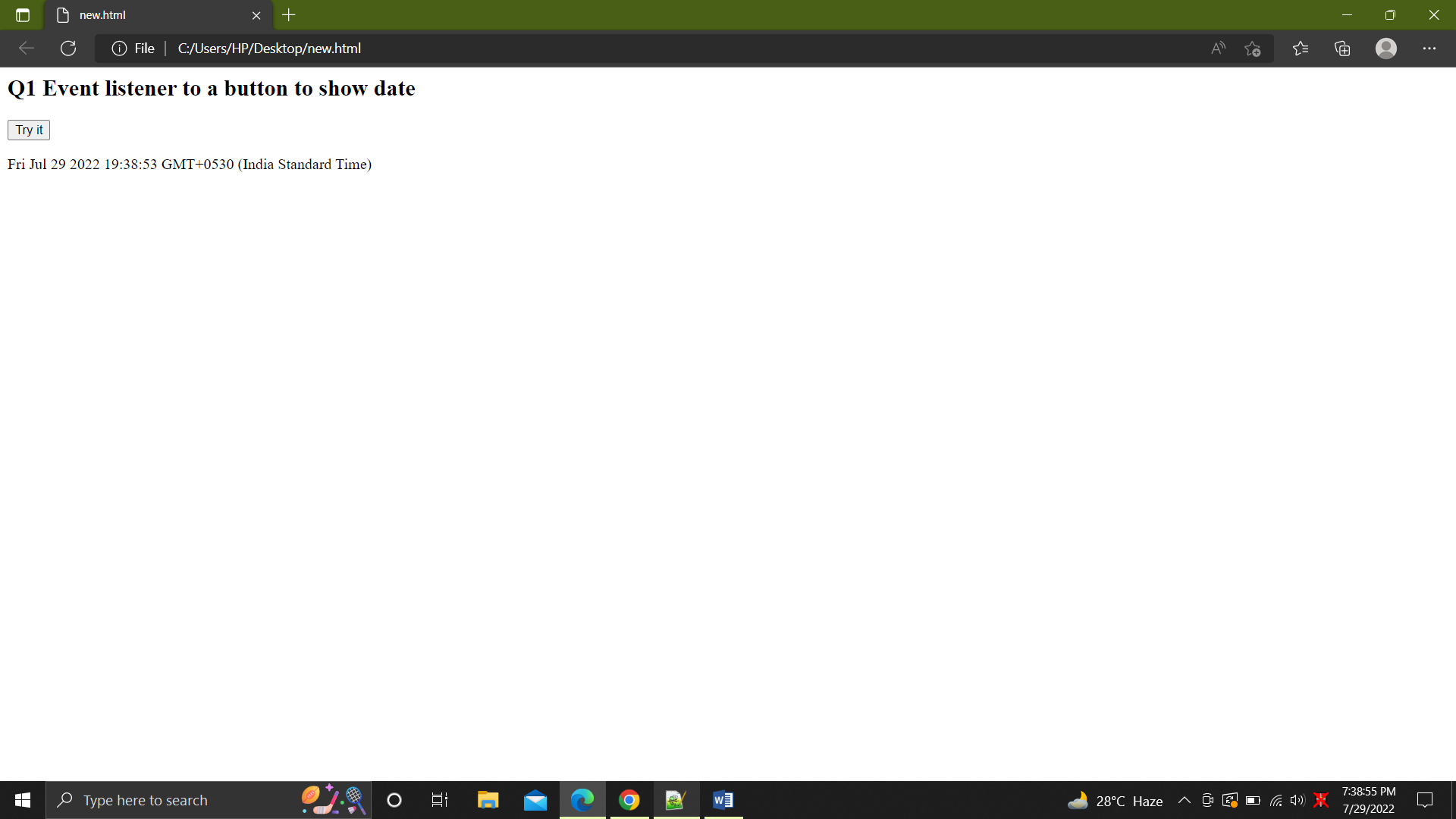
function displayDate() {

document.getElementById("demo").innerHTML = Date();

}

</script>

</body>

</html>

2. Show the javascript validation. When you click submit, the text box doesn't need to

be empty.

<html>

<head>

<script>

function validateForm() {

var x = document.forms["myForm"]["fname"].value;

if (x == "" || x == null) {

alert("Name must be filled out");

return false;

}

}

</script>

</head>

<body>

<h2>Q2 JS validation</h2>

<form

name="myForm"

action="/action\_page.php"

onsubmit="returnvalidateForm()"

method="post"

required >

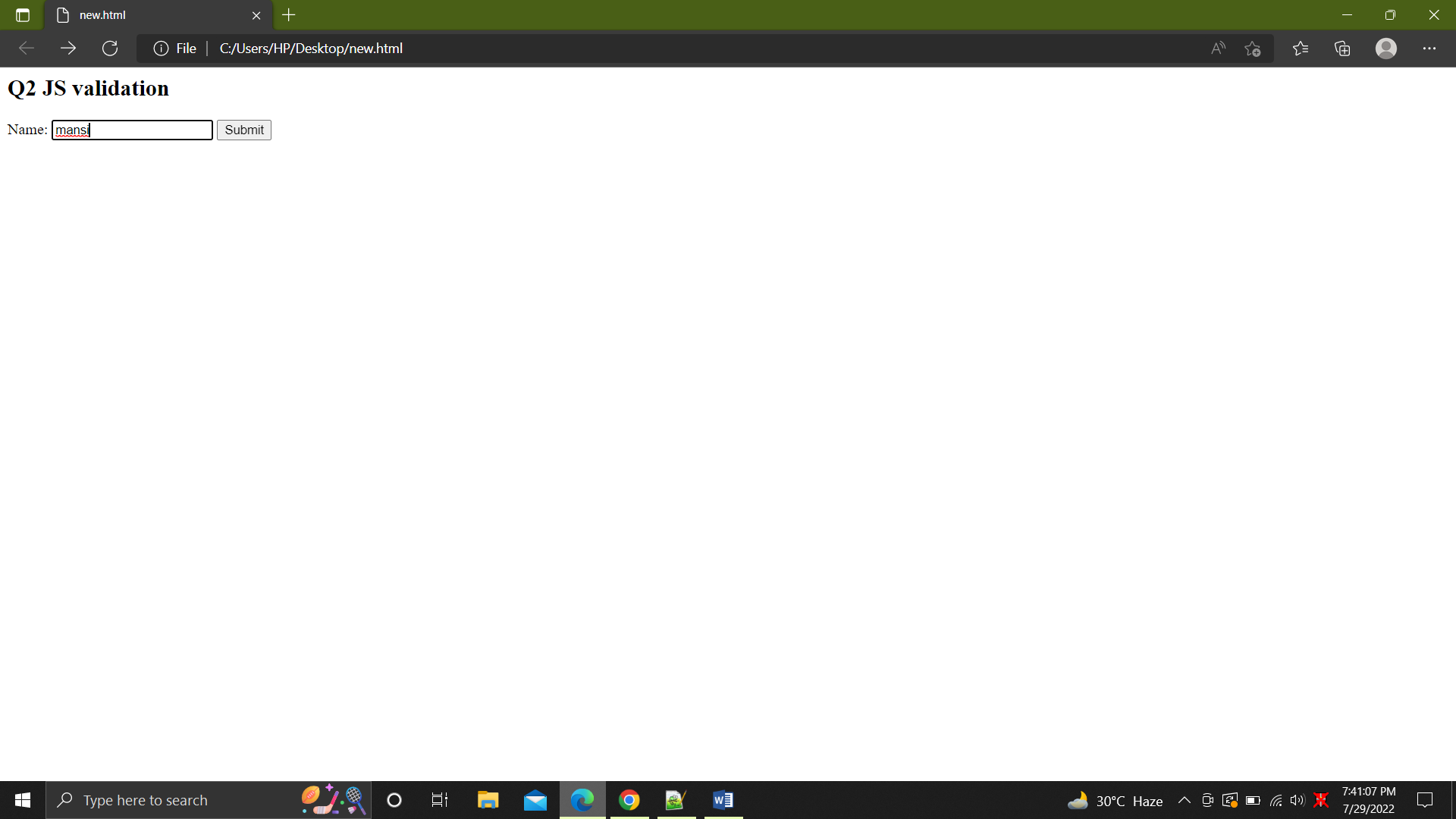
Name: <input type="text" name="fname" />

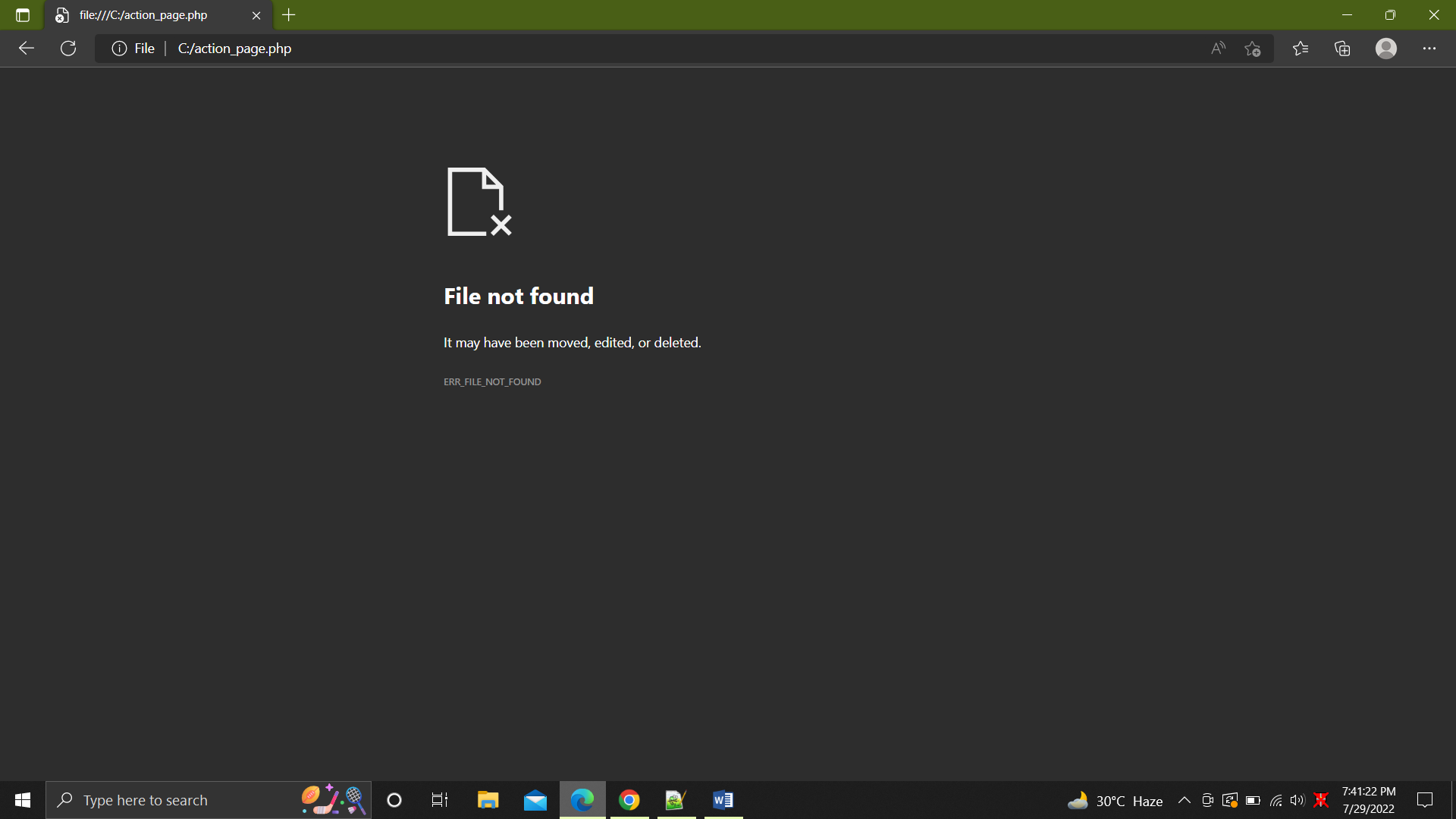
<input type="submit" value="Submit" />

</form>

</body>

</html>





3. How console.log() can be used for Debugging.

<html>

<body>

<h2>Q3 console.log() debugging</h2>

<p>F12 or crtl+shift+i > sources</p>

<script>

a = 8;

b = 4;

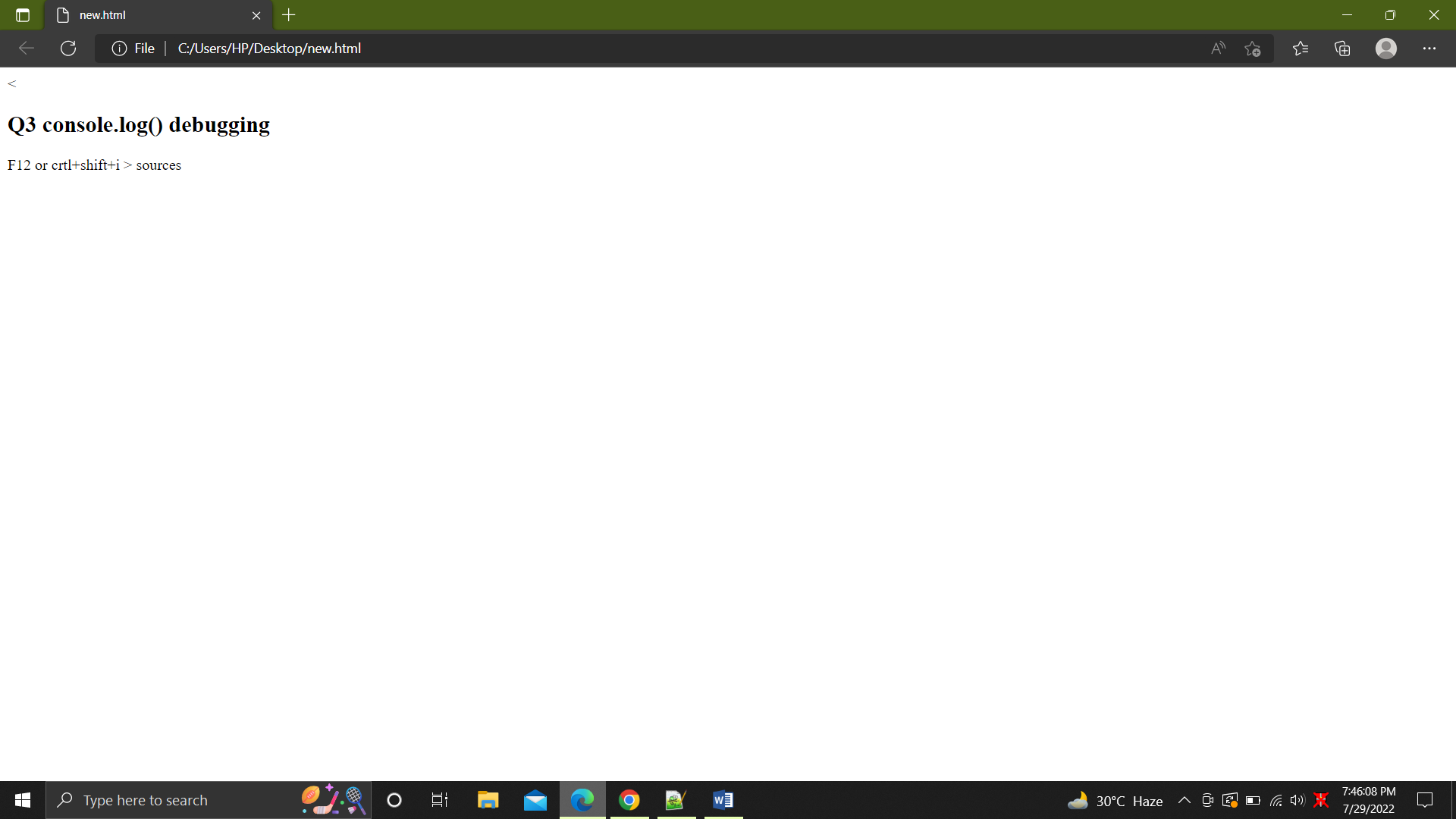
c = a + b;

console.log(c);

</script>

</body>

</html>



4. Write functions to set a cookie, get a cookie and check a cookie in a single program.

<html>

<head>

<script>

function setCookie(cname, cvalue, exdays) {

const d = new Date();

d.setTime(d.getTime() + exdays \* 24 \* 60 \* 60 \* 1000);

let expires = "expires=" + d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

function getCookie(cname) {

let name = cname + "=";

let decodedCookie = decodeURIComponent(document.cookie);

let ca = decodedCookie.split(";");

for (let i = 0; i < ca.length; i++) {

let c = ca[i];

while (c.charAt(0) == " ") {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

function checkCookie() {

let user = getCookie("username");

if (user != "") {

alert("Welcome again " + user);

} else {

user = prompt("Please enter your name:", "");

if (user != "" && user != null) {

setCookie("username", user, 30);

}

}

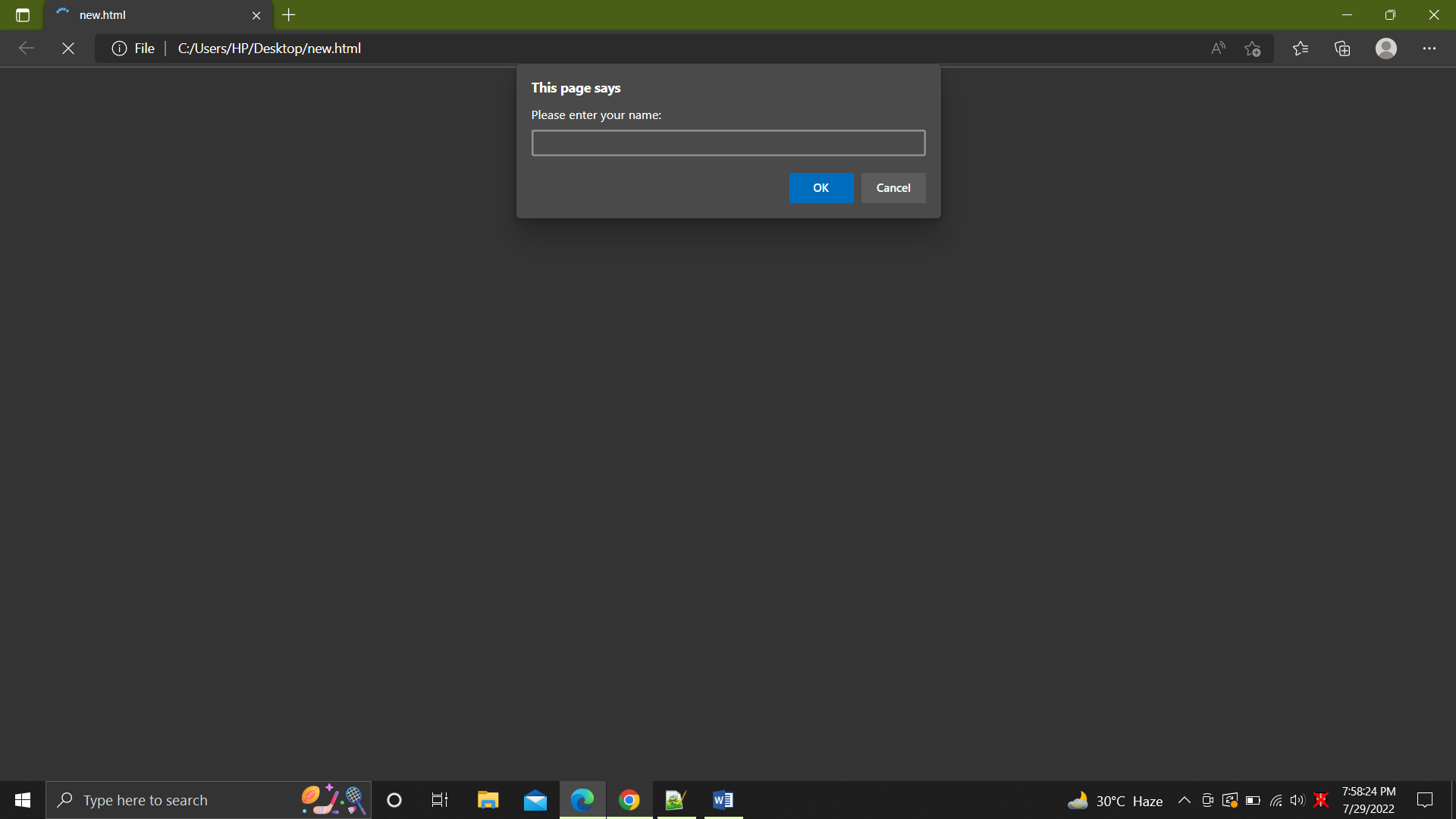
}

</script>

</head>

<body onload="checkCookie()"></body>

</html>



5. Create a JSON object and access it using dot notation.

<html>

<body>

<h2>Q5 Accessing JSON obj with . notation</h2>

<p id="demo"></p>

<script>

const myJSON = '{"name":"John", "age":30, "car":null}';

const myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.name;

</script>

</body>

</html>

