Contents

Scope, If-else Conditionals & Switch Case in JavaScript

In this section, we will learn about scoping in JavaScript and then we will move on to If-Else and Switch Case conditions. Make a new file /and add the boilerplate to get the basic HTML template. Then give a title as **Scope and Conditionals** under the <title> tag.

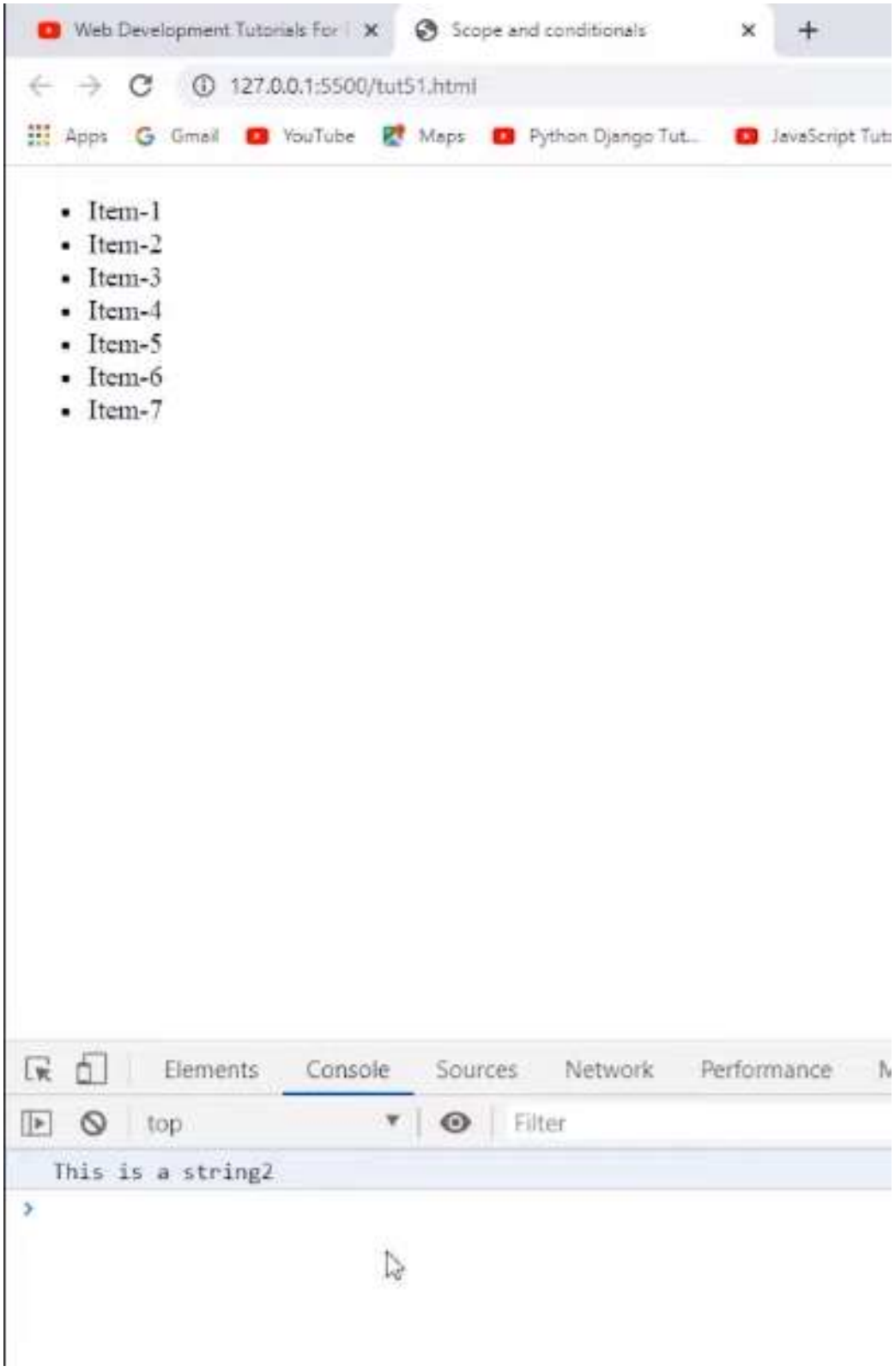
Let us now write the JavaScript under the <script> tag and understand the scope. If we write the following code-

```
var string1 = "This is a string";
    console.log(string1);
```

We will expect the output as “This is a string”. But if we update it as-

```
var string1 = "This is a string2";
    console.log(string1);
```

Then the output will be as follows-



So here, we have noticed that the value of the variable gets easily changed and it can create a problem if we do not remember the initial value given the variable. It is because the scope of **var** is global here. But if we declare the *var* inside any function, then its scope will remain under that function.

- If we write as follows-

```
let a = "u";
    {
    let a = "u6";
    console.log(a)
    }
    console.log(a)
```

In this example, the value of a is ‘u6’ under a particular scope but the value of a declared outside the scope is ‘u’. You will get the output as follows-

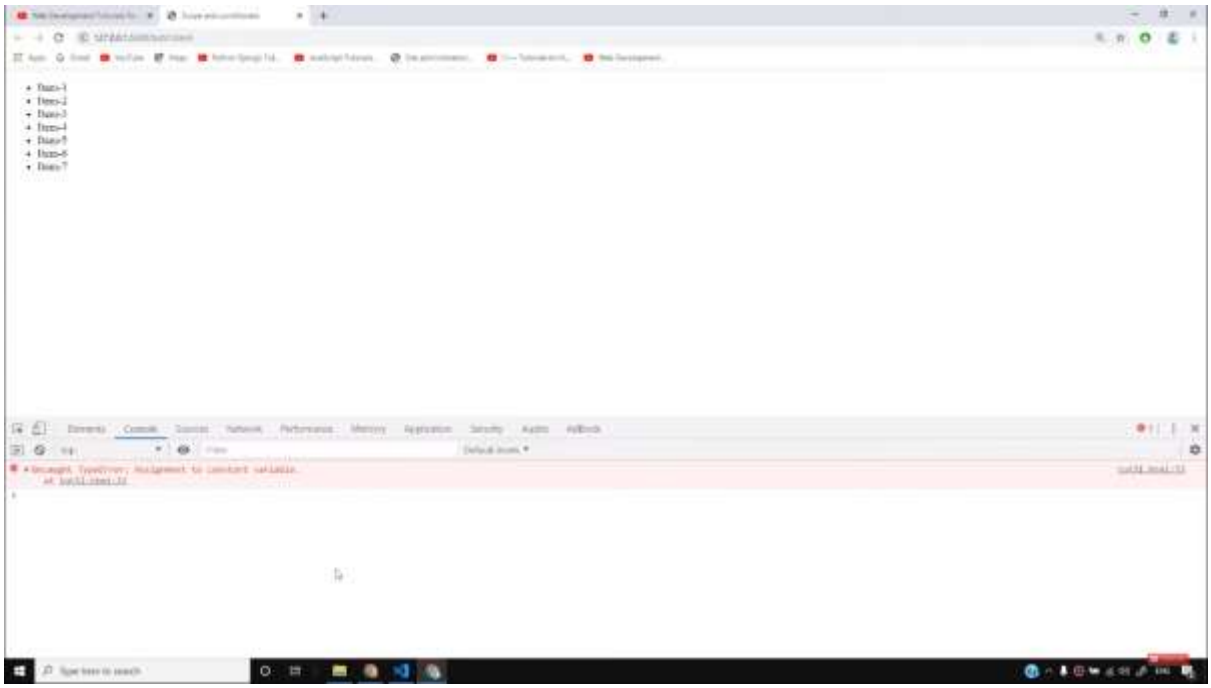

Web Development Tutorials For
Scope and conditionals
127.0.0.1:5500/tut51.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
Item-1
Item-2
Item-3
Item-4
Item-5
Item-6
Item-7
Elements
Console
Sources
Network
top
Filter
This is a string2
u6

However, it is highly recommended to use *let* instead of *var* to avoid confusion of the scope of the variable.

- If we write as follows-

```
const a = "This cannot be changed";

a = "I want to change this. This cannot be changed";

console.log(a);
```

We will see the output as-



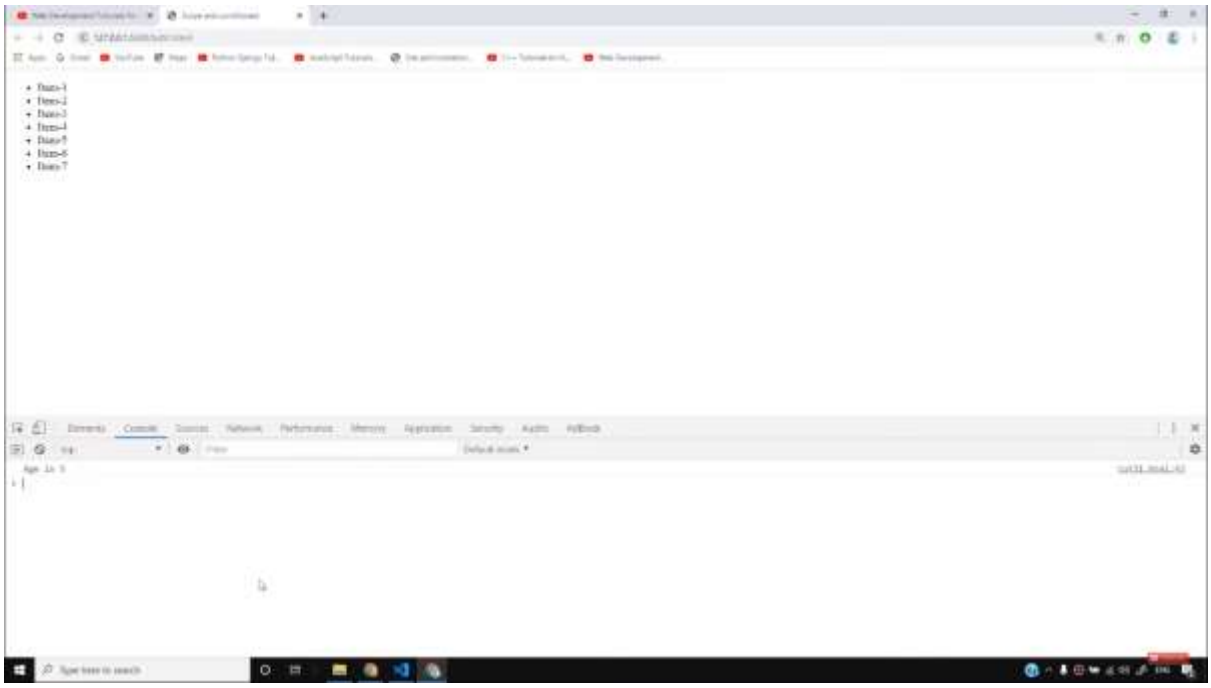
In the above example, we have declared **a** as a **constant**, therefore, its value cannot be changed further by any means. Hence, it shows an error.

- Now let us move towards conditional statements. If we write the code as follows-

```
let age = 5;
    if(age>18){
       console.log("You can drink water");
    }
    else if(age==2){
       console.log("Age is 2")
    }
    else if(age==5){
       console.log("Age is 5")
```

```
}
else{
    console.log("You can drink Cold Drink");
}
```

The output of the above code will be 5 as shown below-



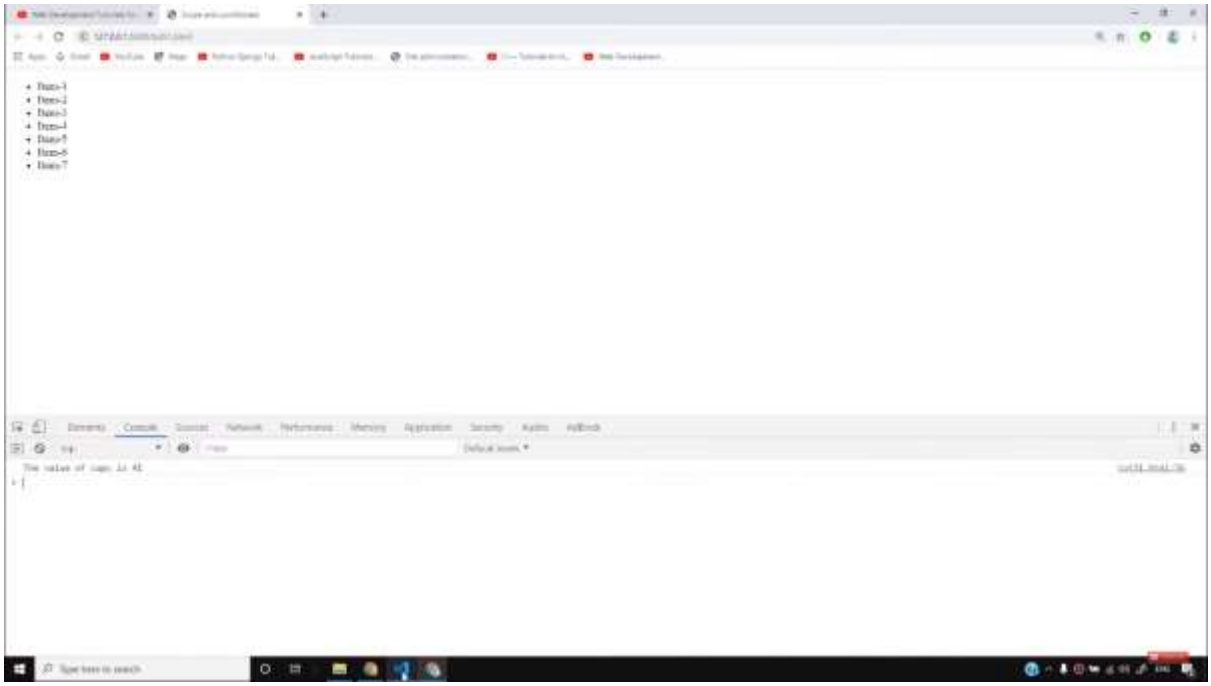
It is because we set the age initially as 5 and it prints the condition which is true for that particular statement. It is known as an **if-else ladder.**

- Let us now understand switch-case statements. If we write as follows-

```
const cups = 41;
  switch (cups) {
    case 4:
      console.log("The value of cups is 4")
      break;

    case 41:
      console.log("The value of cups is 41")
      break;

    case 42:
```

```
        console.log("The value of cups is 42")
        break;

    case 43:
        console.log("The value of cups is 43")
        break;

    default:
        console.log("The value of cups is none of 4, 41, 42, 43")
        break;
}
```

Here you will see the output as-



It is because the case has matched exactly with the value of *cups.* However, if we remove the break statement then all the values will be printed. And in case, no value matches with any case, then the **default** statement will be printed.

Arrays & Objects in JavaScript

In the previous tutorial, we have seen the scope of the variable and different conditional statements like if-else and switch-case. Moving further, we are going to learn about Arrays and Objects used in JavaScript. Make a new file and add an instant boilerplate to get the HTML template. Give the title as **Arrays and Objects** under the <title> tag.

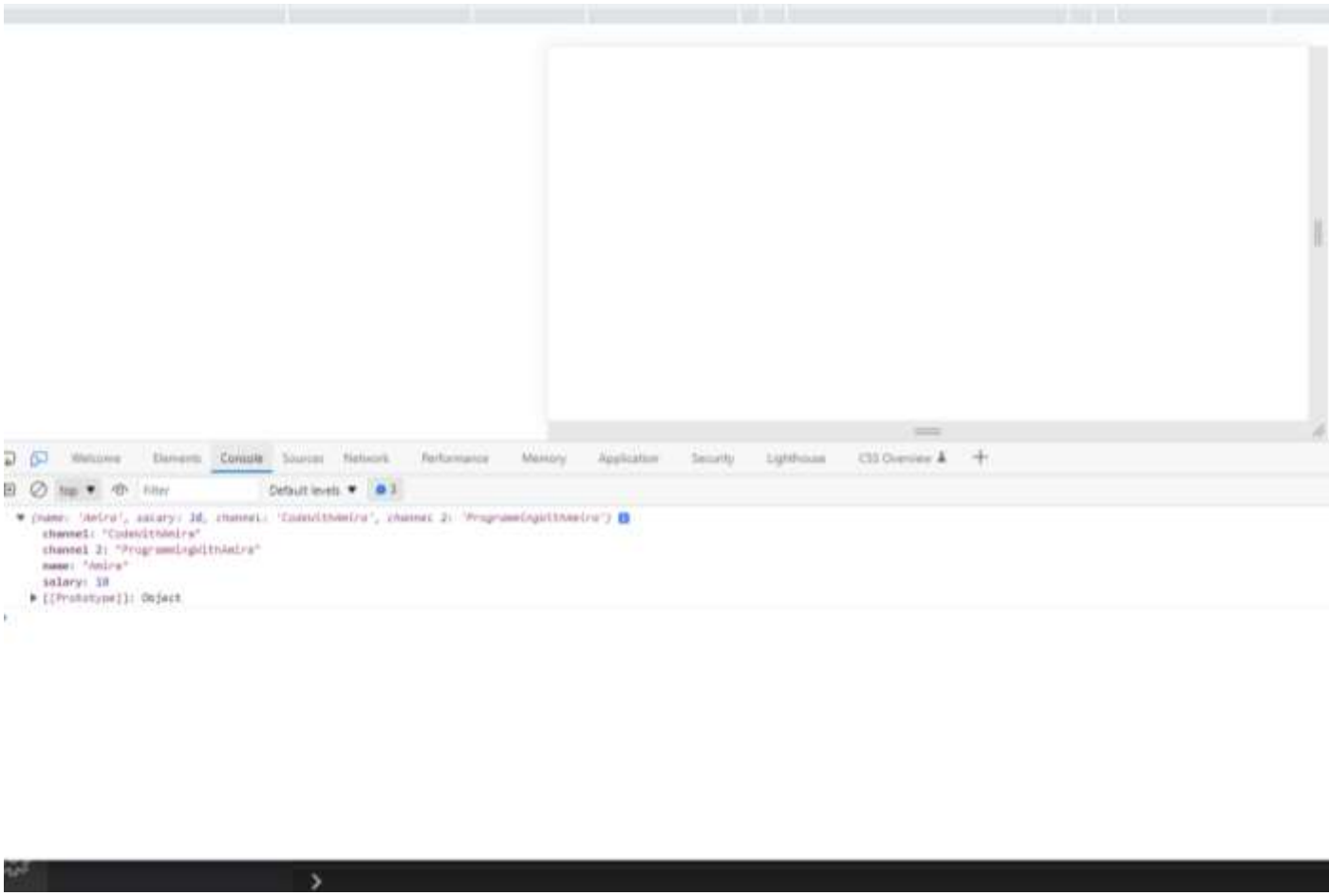
The object class represents one of JavaScript's data types. It is used to store various keyed collections and more complex entities. Objects can be created using the **Object()** constructor. we have two types of values used in JavaScript- primitive and reference. To be more simple, either we create *objects* or *primitive data types.* The primitive data types are as follows-

```
let myVar = 34;
    let myVar2 = "string";
    let myVar3 = true;
    let myVar4 = null;
    let myVar5 = undefined;
```

Apart from primitive data types, all the others are objects. Let us now see how to define objects. If we write as follows-

```
let employee = {
        name: "Amira",
        salary: 10,
        channel: "CodeWithAmira",
        "channel 2": "ProgrammingWithAmira",
    }
    console.log(employee);
```

From the above code, the output will be generated as follows-



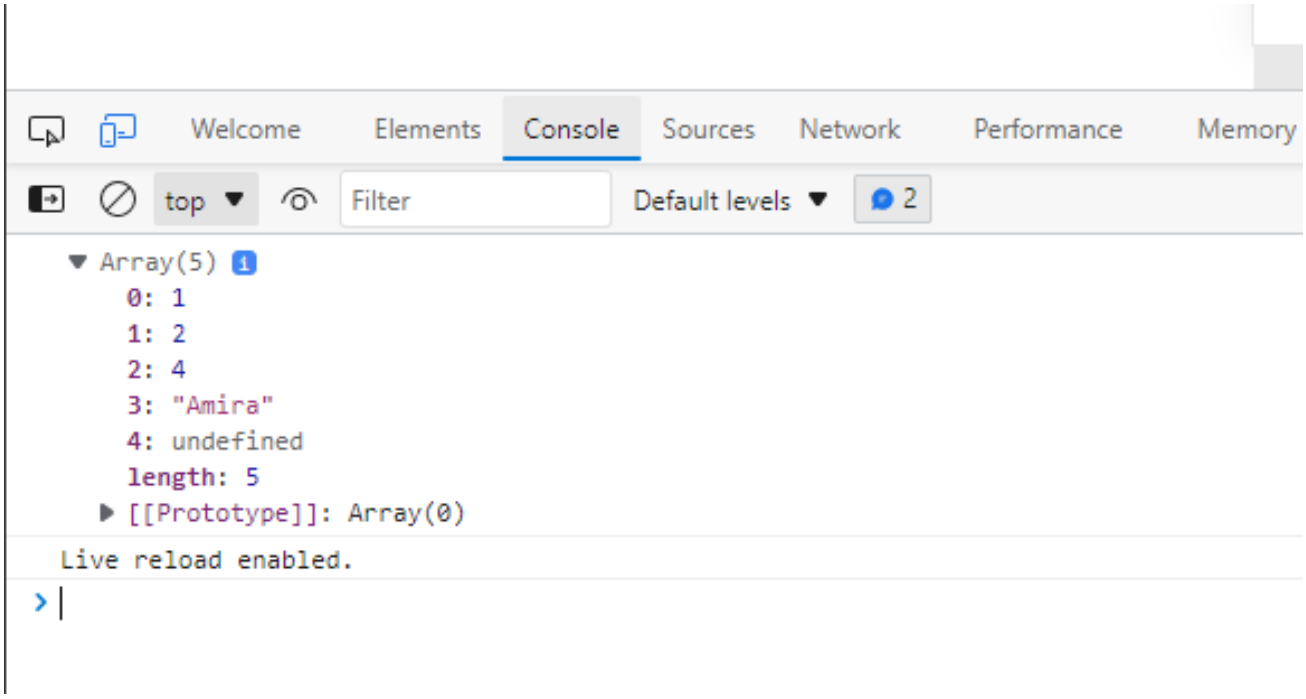
In this example, an **employee** is an object that contains the ***name, salary, channel, and channel 2*** inside it. This is an example, where we create objects manually by us.

There is a special kind of object known as Arrays. The JavaScript Array class is a global object that is used in the construction of arrays; which are high level, list-like objects. An array is a special variable, which can hold more than one value at a time.

We can declare an array as follows-

```
let names = [1, 2, 4, "Amira", undefined];
console.log(names);
```

An array can contain any type of value in it whether it is a string, integer, or boolean. After writing the above code, you will get the output as follows-



To print the value present at any index number, we can write as follows-

console.log(names[1])

It will print the value present at index number 1. Arrays are important because they make it easier to iterate through each and every element present in DOM. We can also create arrays with the help of a new keyword as follows-

let names = new Array(23);

In this way, an array will be treated as an object. The new keyword is used to create a new object. To know the length of an array, we can use the **length** function as follows-

console.log(names.length);

In this example, the output will be 5 because the array *names* contain 5 elements in it.

If we declare a new array as-

let names = new Array(41, 2, 4, "Amira", undefined);

However, to sort this array, we can use sort() function as follows-

names = names.sort();This will sort all the elements in the array.

JavaScript Tutorial: Navigation The DOM

In this section, we are going to learn how we can do DOM manipulation in JavaScript. Let us start by making a new file and then add the boilerplate to get the HTML code. Then give the title as **Manipulating DOM** under the <title> tag.

The HTML DOM (Document Object Model) is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to **get, change, add**, or **delete** HTML elements.

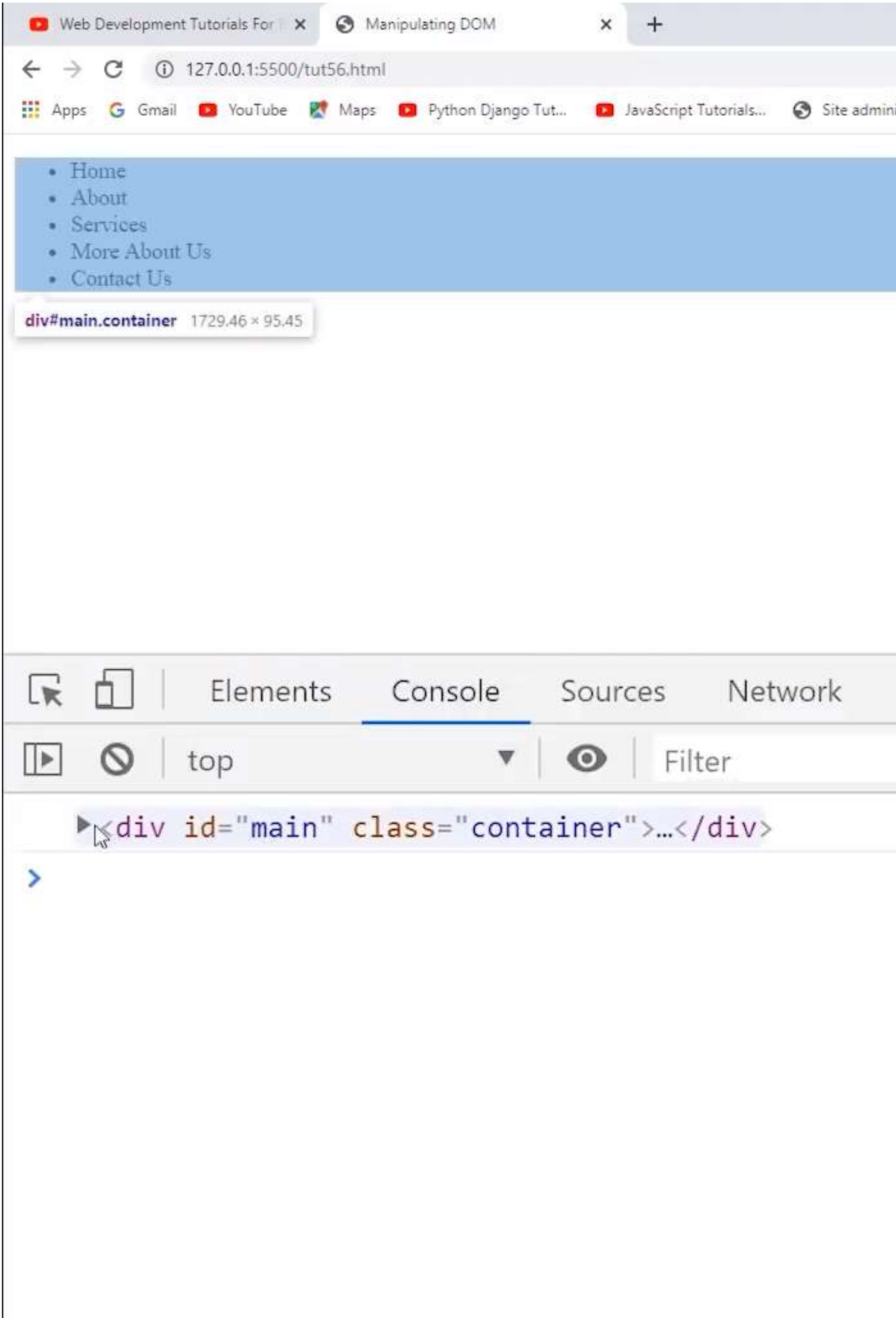
Let us begin with our HTML code as follows-

```
<div id="main" class="container">
    <ul id="nav">
        <li>Home</li>
        <li>About</li>
        <li>Services</li>
        <li>More About Us</li>
        <li>Contact Us</li>
    </ul>
</div>
```

- The most common way to access an HTML element is to use the *id* of the element. For example, if we write as follows-

```
let main = document.getElementById('main');
console.log(main);
```

By writing **getElementById,** we can target the HTML through its id. Therefore, the result of the above code will look like-

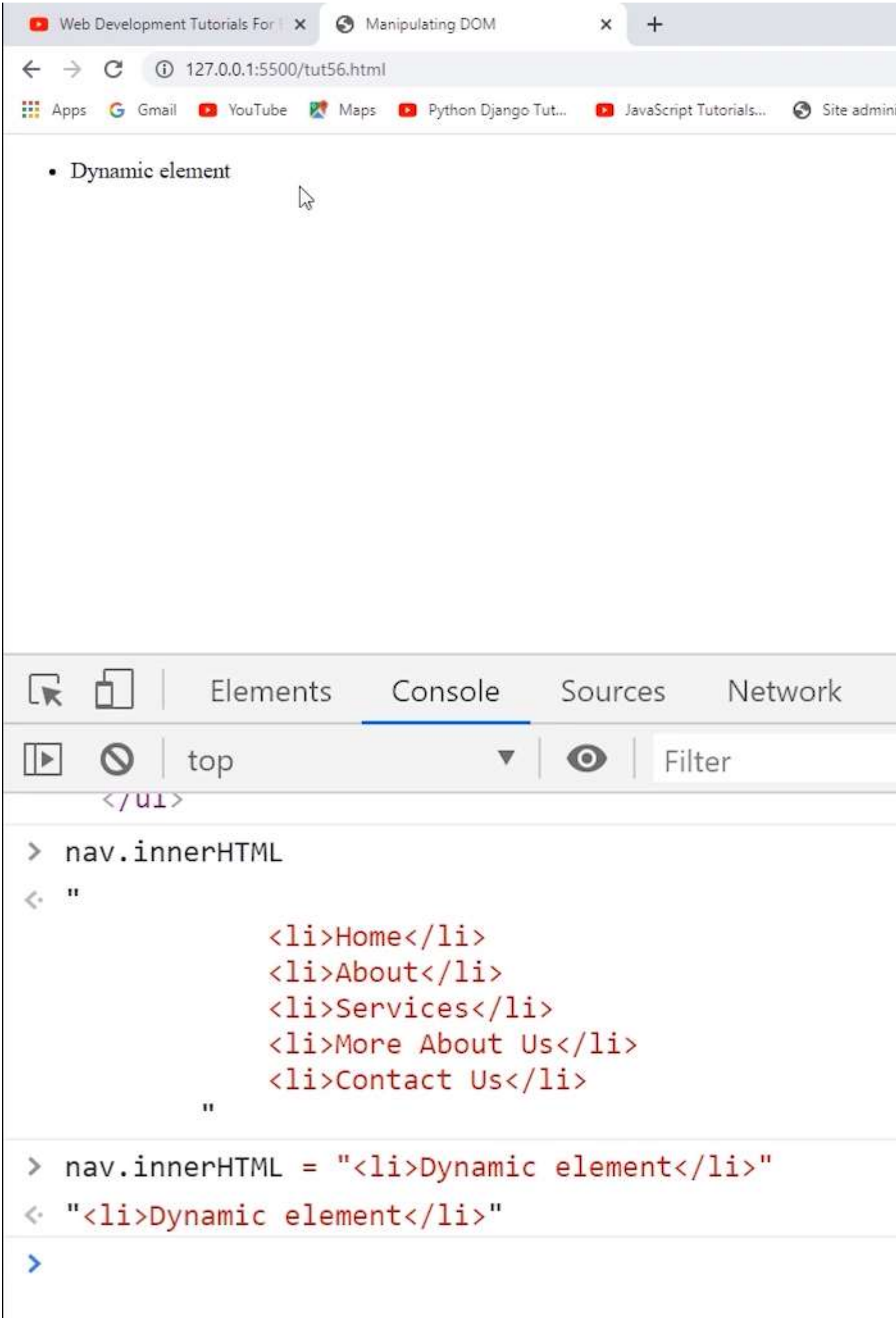

Web Development Tutorials For
Manipulating DOM
127.0.0.1:5500/tut56.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admini
Home
About
Services
More About Us
Contact Us
div#main.container 1729.46 × 95.45
Elements
Console
Sources
Network
top
Filter
…

Let us see another example. I want to target the elements under the id *nav,* we can write this as follows-

```
let nav = document.getElementById('nav');
console.log(nav);
```

For DOM to work, keep in mind that while selecting the elements with its id, there should be a unique id for every element in the HTML code. You cannot work with two same ids for different elements otherwise it will create a problem.

The easiest way to get the content of an element is by using the **innerHTML property.** It is useful for getting or replacing the content of HTML elements. The example is shown below-

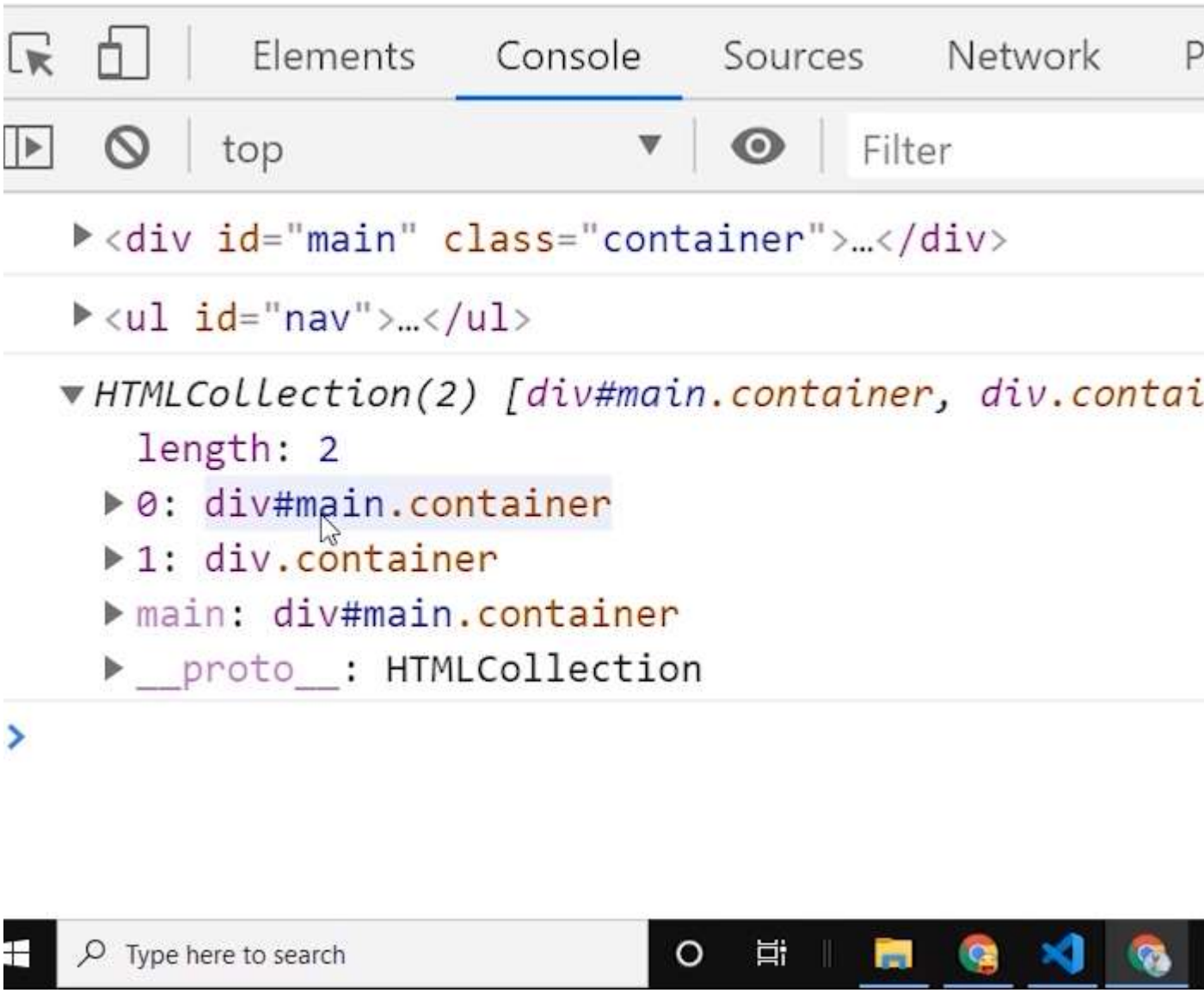

Web Development Tutorials For
Manipulating DOM
127.0.0.1:5500/tut56.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
• Dynamic element
Elements
Console
Sources
Network
top
Filter

> nav.innerHTML
"
Home
About
Services
More About Us
Contact Us
"
> nav.innerHTML = "
Dynamic element
"
"
Dynamic element
"
>

- However, we can also extract the elements from HTML with the help of **class** name. If we want to find all the elements with same class name, use **getElementsByClassName().** Let us understand this with an example. If we write as follows-

```
let containers = document.getElementsByClassName('container');
console.log(containers);
```

The output of the above code will be as follows-



Here, we can see two container classes as have made two container classes. But if we write as a **container(0)** or **container(1)** then we will get the first and second containers respectively.

JavaScript Tutorial: Events & Listening to Events

In this section, we are going to learn how we can make browser events come into action and how should we use them. An HTML event can be something the browser does or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

Here is a list of some common HTML events-

- **Onchange-** An HTML element has been changed
- **Onclick-** The user clicks an HTML element
- **Onmouseover-** The user moves the mouse over an HTML element
- **Onmouseout-**The user moves the mouse away from an HTML element
- **Onkeydown-** The user pushes a keyboard key
- **Onload-** The browser has finished loading the page

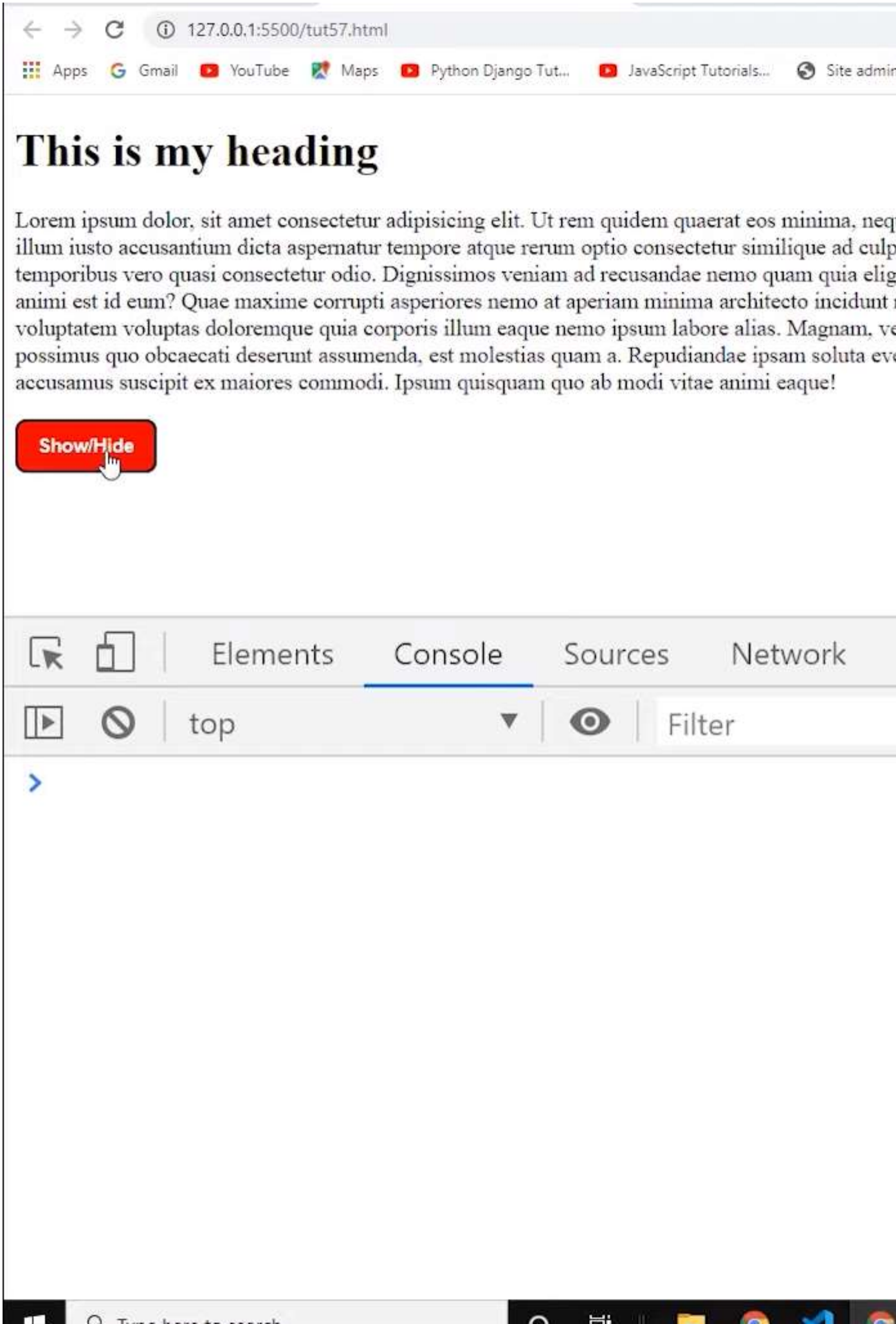
Make a new file and add the boilerplate to get the HTML template. Then give the title as **JS Events** under the <title> tag. Let us now write some HTML to begin the work. We will simply add the paragraph and heading and the result will be as follows-

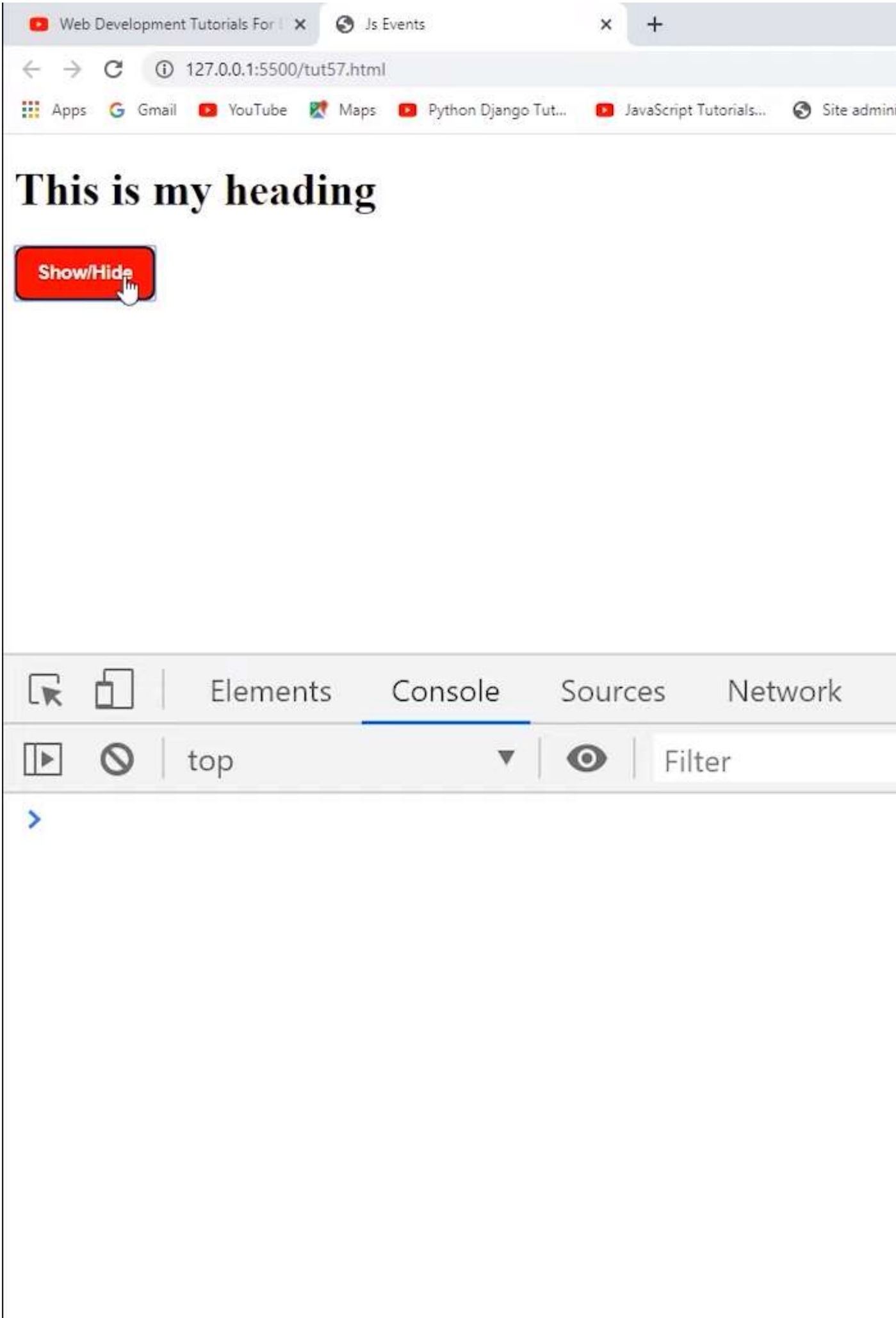
Now will make a *button* and we want to hide that button if someone clicks on it. But before that we can add some CSS to it to make it look attractive.

If we want that while clicking the button the paragraph should hide and by again clicking, it should come back, then we can write as follows-

```
<button id="btn" onclick="toggleHide()">Show/Hide</button>
function toggleHide(){
      let btn = document.getElementById('btn');
      let para = document.getElementById('para');
      if(para.style.display != 'none'){
      para.style.display = 'none';
      }
      else{
      para.style.display = 'block';
      }
  }
```

The result will be as follows-


127.0.0.1:5500/tut57.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is my heading
Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ut rem quidem quaerat eos minima, neq
illum iusto accusantium dicta aspernatur tempore atque rerum optio consectetur similique ad culp
temporibus vero quasi consectetur odio. Dignissimos veniam ad recusandae nemo quam quia elig
animi est id eum? Quae maxime corrupti asperiores nemo at aperiam minima architecto incidunt
voluptatem voluptas doloremque quia corporis illum eaque nemo ipsum labore alias. Magnam, ve
possimus quo obcaecati deserunt assumenda, est molestias quam a. Repudiandae ipsam soluta eve
accusamus suscipit ex maiores commodi. Ipsum quisquam quo ab modi vitae animi eaque!
Show/Hide
Elements
Console
Sources
Network
top
Filter


Web Development Tutorials For
Js Events
127.0.0.1:5500/tut57.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is my heading
Show/Hide
Elements
Console
Sources
Network
top
Filter

In the same way, there is another event listener known as a **mouseover.** For example, if we want to alert when the mouse pointer is on the paragraph, then we can write the code as follows-

```
let para = document.getElementById('para');
      para.addEventListener('mouseover', function run(){
         alert('Mouse Inside')
      });
```

The result of the above code will be that whenever the mouse pointer will be over the paragraph, it will make an alert as shown below-


Web Development Tutorials For
Js Events
127.0.0.1:5500/tut57.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is my heading
Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ut rem quidem quaerat eos minima, nequ
illum iusto accusantium dicta aspernatur tempore atque rerum optio consectetur similique ad culpa
temporibus vero quasi consectetur odio. Dignissimos veniam ad recusandae nemo quam quia eligo
animi est id eum? Quae maxime corrupti asperiores nemo at aperiam minima architecto incidunt n
voluptatem voluptas doloremque quia corporis illum eaque nemo ipsum labore alias. Magnam, ve
possimus quo obcaecati deserunt assumenda, est molestias quam a. Repudiandae ipsam soluta eve
accusamus suscipit ex maiores commodi. Ipsum quisquam quo ab modi vitae animi eaque!
Show/Hide

In the same way, if we want to alert when the pointer is outside the paragraph, we can write the code as follows-

```
para.addEventListener('mouseout', function run(){
        alert('Mouse now went outside')
    });
```

In the same way, you can try out with all the other events and practice more

Full code of Events Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Js Events</title>
</head>
<style>
#btn{
    padding:10px 14px;
    background-color: red;
    border: 2px solid black;
    color: white;
    font-weight: bold;
    border-radius: 8px;
    cursor: pointer;
}
</style>
<body>
    <!-- Browser events:
    click
    contextmenu
    mouseover/mouseout
    mousedown/mouseup
    mousemove

    submit
    focus

    DOMContentLoaded

    transitionend -->

    <div class="container">
        <h1>This is my heading  </h1>
        <p id="para">Lorem ipsum dolor, sit amet consectetur adipisicing
elit. Ut rem quidem quaerat eos minima, neque sapiente ipsam debitis
tempore quae sequi autem sunt ea cupiditate? Saepe corporis, laboriosam
facere distinctio sint qui unde ipsam molestias officiis totam ullam id
illum iusto accusantium dicta aspernatur tempore atque rerum optio
consectetur similique ad culpa veritatis. Dolores atque fuga, dignissimos
vel velit minus, necessitatibus ipsum culpa recusandae architecto
repellendus harum voluptatem placeat fugiat blanditiis minima temporibus
vero quasi consectetur odio. Dignissimos veniam ad recusandae nemo quam
quia eligendi. Cumque similique ut enim pariatur hic blanditiis
reprehenderit maiores quibusdam ratione quisquam beatae laborum aperiam
magnam iure debitis voluptas, molestiae animi est id eum? Quae maxime
```

```
corrupti asperiores nemo at aperiam minima architecto incidunt
necessitatibus. Minus explicabo similique quaerat! Tenetur quae amet sint
quaerat at ad veniam, pariatur similique qui totam beatae ut eos, maiores
minus assumenda voluptatem voluptas doloremque quia corporis illum eaque
nemo ipsum labore alias. Magnam, vel quaerat qui excepturi est tenetur ab,
esse asperiores porro beatae quasi nobis placeat. Natus in perferendis nam
vitae, enim odio eveniet animi sunt ut nobis rem velit dicta possimus quo
obcaecati deserunt assumenda, est molestias quam a. Repudiandae ipsam
soluta eveniet quia assumenda error autem cum repellat eius nisi. Dolores,
perferendis, velit nisi omnis dolore voluptatum nostrum quasi, sit odio aut
quas quia minus rerum accusamus suscipit ex maiores commodi. Ipsum quisquam
quo ab modi vitae animi eaque!</p>
    </div>
    <button id="btn" onclick="toggleHide()">Show/Hide</button>
    <script>
        let para = document.getElementById('para');
        para.addEventListener('mouseover', function run(){
            console.log('Mouse Inside')
        });

        para.addEventListener('mouseout', function run(){
            console.log('Mouse now went outside')
        });


    function toggleHide(){
        // let btn = document.getElementById('btn');
        let para = document.getElementById('para');
        if(para.style.display != 'none'){
        para.style.display = 'none';
        }
        else{
        para.style.display = 'block';
        }
    }
    </script>


</body>
</html>
```

JavaScript Tutorial: Math Object In JavaScript

In this section we are going to see different *Math* objects used in JavaScript. These are important for them who want to use different mathematical functions in their browser. This function allows us to perform mathematical tasks on numbers.

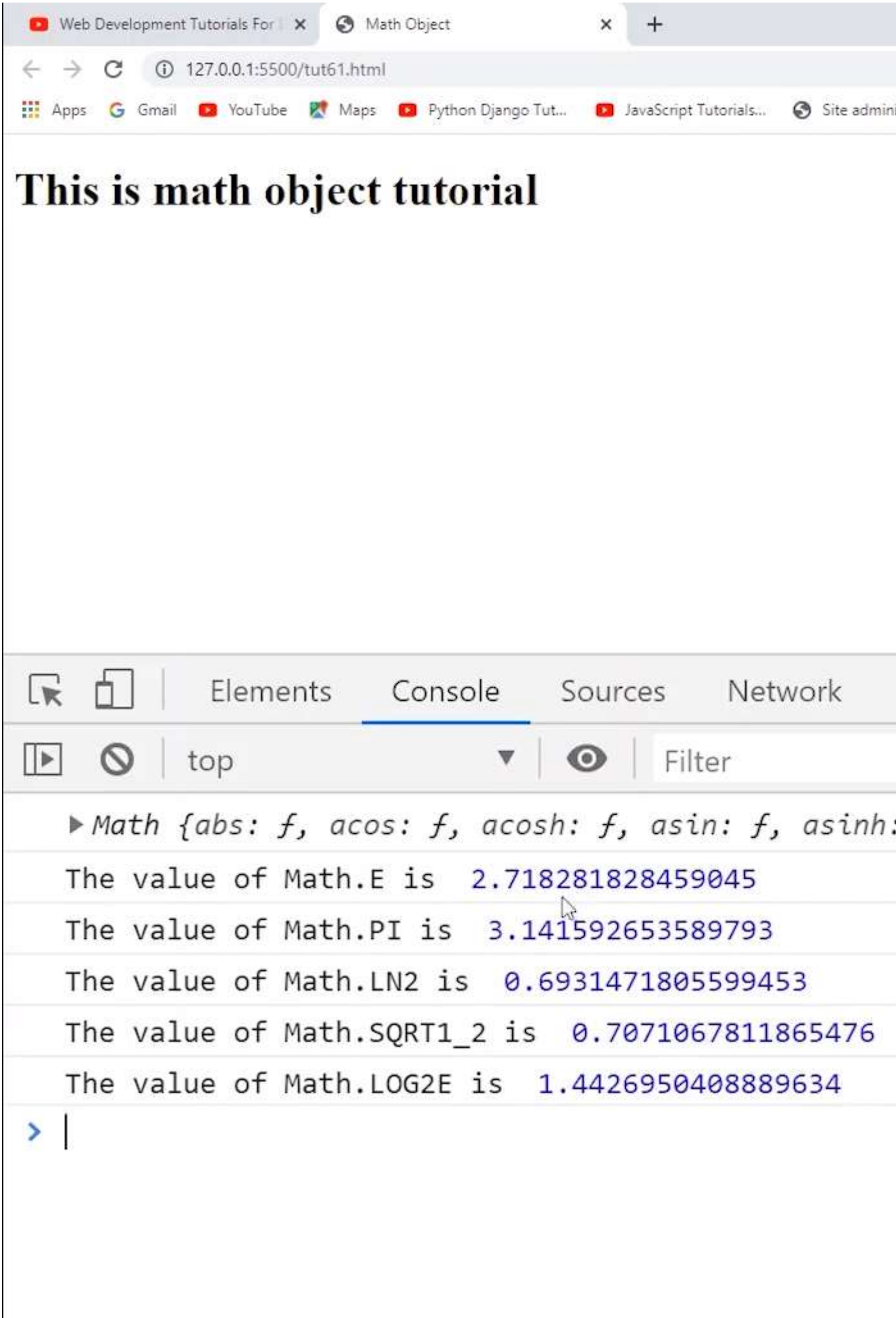
Make a new file and add the boilerplate to get the HTML template. Then give the title as **Math Object** under the <title> tag. If we write the following code under the <script> tag, then we will get the list of all those *Math* functions that are used in JavaScript.

```
let m = Math;
   console.log(m)
```

Let us now print the values of some constants using the Math functions.

```
// Printing the constants from Math Object
   console.log("The value of Math.E is ", Math.E)
   console.log("The value of Math.PI is ", Math.PI)
   console.log("The value of Math.LN2 is ", Math.LN2)
   console.log("The value of Math.SQRT1_2 is ", Math.SQRT1_2)
   console.log("The value of Math.LOG2E is ", Math.LOG2E)
```

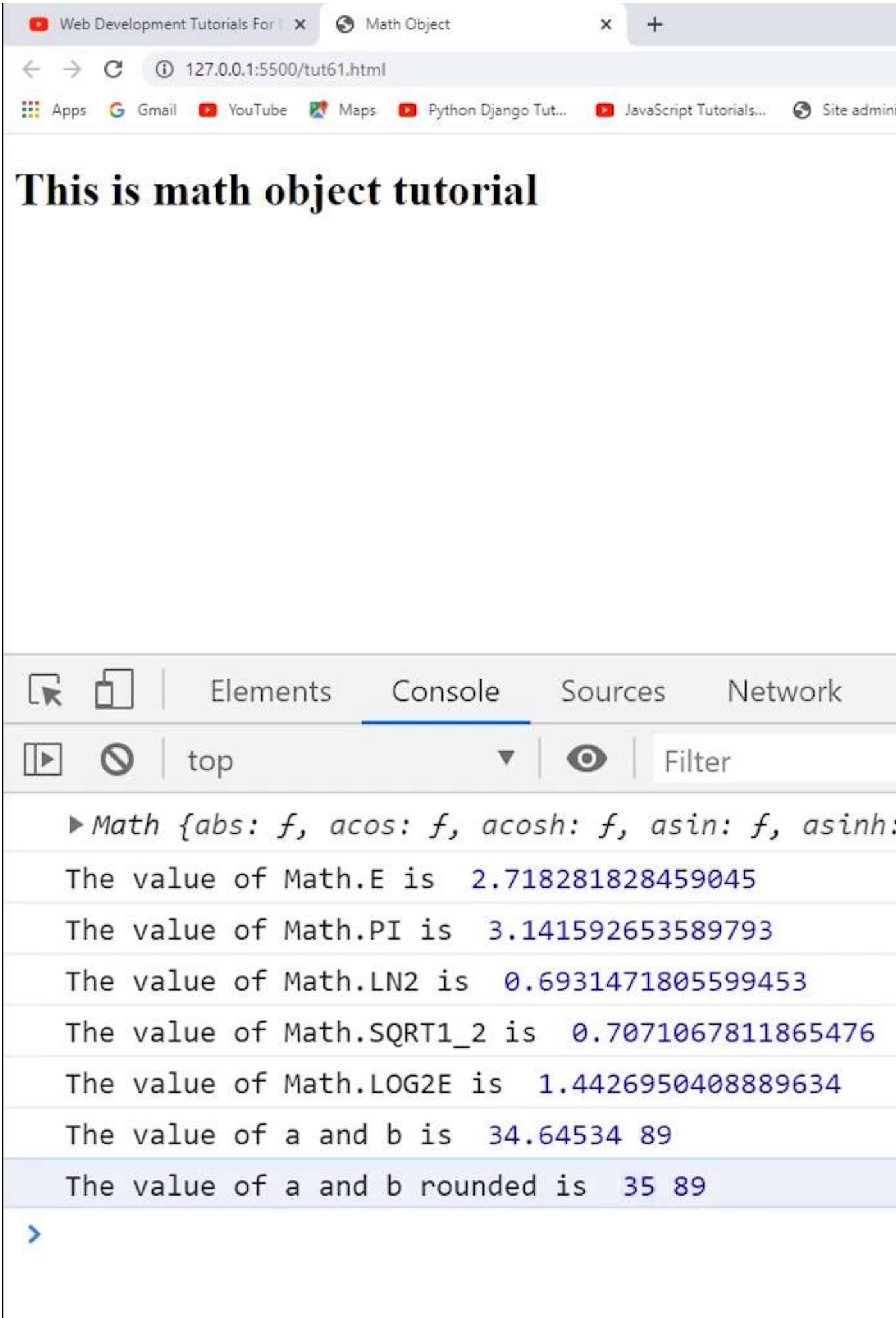
After executing the above JavaScript code, we will see the values of these constants in the output as follows-


Web Development Tutorials For
Math Object
127.0.0.1:5500/tut61.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admini
This is math object tutorial
Elements
Console
Sources
Network
top
Filter
Math {abs: ƒ, acos: ƒ, acosh: ƒ, asin: ƒ, asinh:
The value of Math.E is 2.718281828459045
The value of Math.PI is 3.141592653589793
The value of Math.LN2 is 0.6931471805599453
The value of Math.SQRT1_2 is 0.7071067811865476
The value of Math.LOG2E is 1.4426950408889634

Now let us see how to print the functions with the help of Math Object. Here we will initialize two variables and then call them under different functions as follows-

```
let a = 34.64534;
let b = 89;

console.log("The value of a and b is ", a, b);
console.log("The value of a and b rounded is ", Math.round(a), Math.round(b));
```

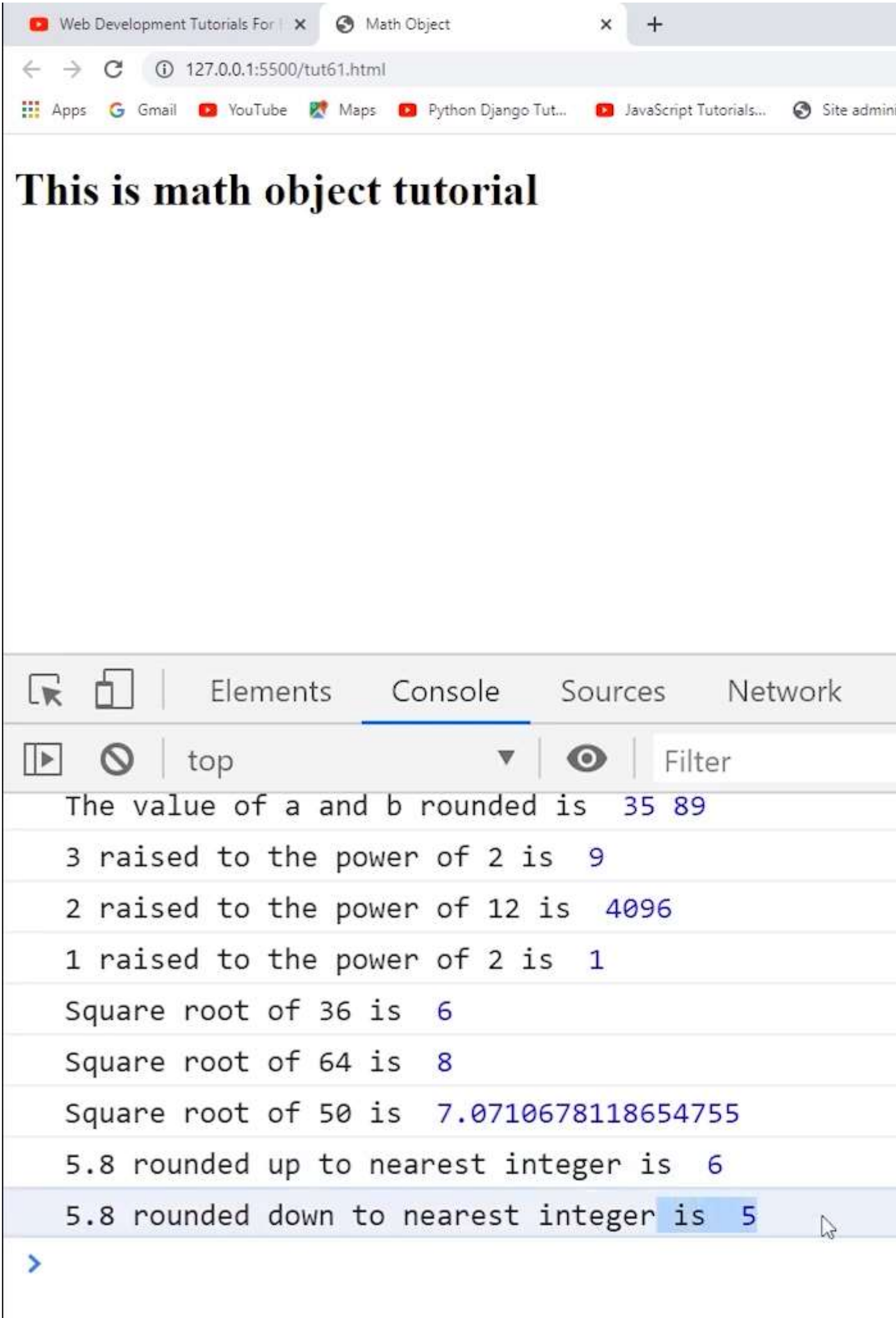
The output we get, is as follows-


Web Development Tutorials For
Math Object
127.0.0.1:5500/tut61.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is math object tutorial
Elements
Console
Sources
Network
top
Filter
Math {abs: ƒ, acos: ƒ, acosh: ƒ, asin: ƒ, asinh:
The value of Math.E is 2.718281828459045
The value of Math.PI is 3.141592653589793
The value of Math.LN2 is 0.6931471805599453
The value of Math.SQRT1_2 is 0.7071067811865476
The value of Math.LOG2E is 1.4426950408889634
The value of a and b is 34.64534 89
The value of a and b rounded is 35 89

Math.round is used to round the values to the nearest integer. Then there are Math.pow() and Math.sqrt() functions that are used to return the power and square root of any number respectively. There are other functions like **ceil** and **floor.** Ceil is used to round up the number to the nearest integer whereas floor is used to round down to the nearest integer.

```
console.log("5.8 rounded up to nearest integer is ", Math.ceil(5.8))
console.log("5.8 rounded down to nearest integer is ", Math.floor(5.8))
```

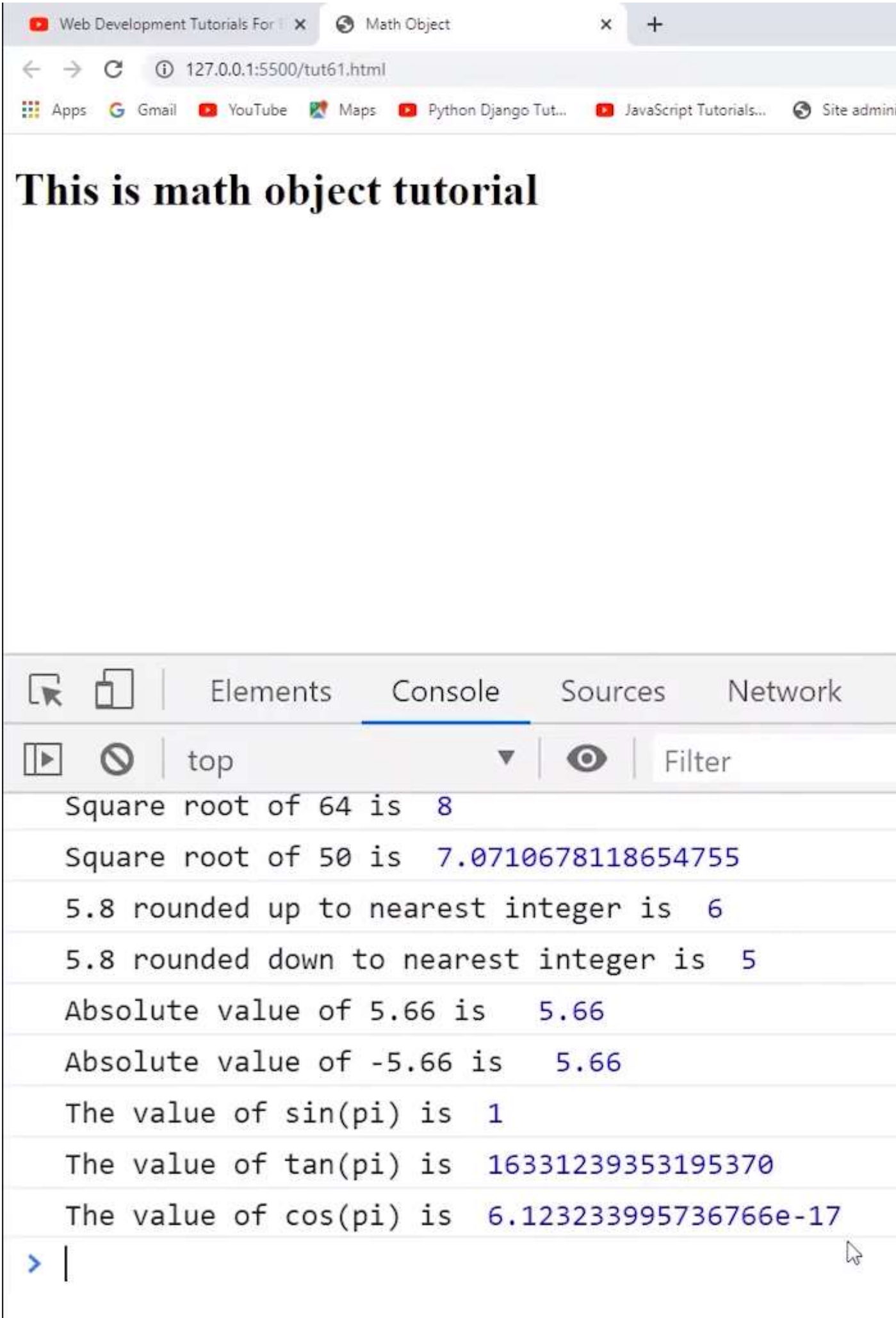
In the above example, 5.8 is rounded up and rounded down as follows-


Web Development Tutorials For
Math Object
127.0.0.1:5500/tut61.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is math object tutorial
Elements
Console
Sources
Network
top
Filter
The value of a and b rounded is 35 89
3 raised to the power of 2 is 9
2 raised to the power of 12 is 4096
1 raised to the power of 2 is 1
Square root of 36 is 6
Square root of 64 is 8
Square root of 50 is 7.071067811865475
5.8 rounded up to nearest integer is 6
5.8 rounded down to nearest integer is 5

Now let us see some trignometric functions. The values given here will be in *radians* as follows-

```
// Trinonometric Functions
   console.log("The value of sin(pi) is ", Math.sin(Math.PI/2))
   console.log("The value of tan(pi) is ", Math.tan(Math.PI/2))
   console.log("The value of cos(pi) is ", Math.cos(Math.PI/2))
```

The output of the following code will be as follows-


Web Development Tutorials For
Math Object
127.0.0.1:5500/tut61.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admin
This is math object tutorial
Elements
Console
Sources
Network
top
Filter
Square root of 64 is 8
Square root of 50 is 7.0710678118654755
5.8 rounded up to nearest integer is 6
5.8 rounded down to nearest integer is 5
Absolute value of 5.66 is 5.66
Absolute value of -5.66 is 5.66
The value of sin(pi) is 1
The value of tan(pi) is 16331239353195370
The value of cos(pi) is 6.123233995736766e-17

We can take the help of min and max functions to find the minimum and maximum numbers respectively.

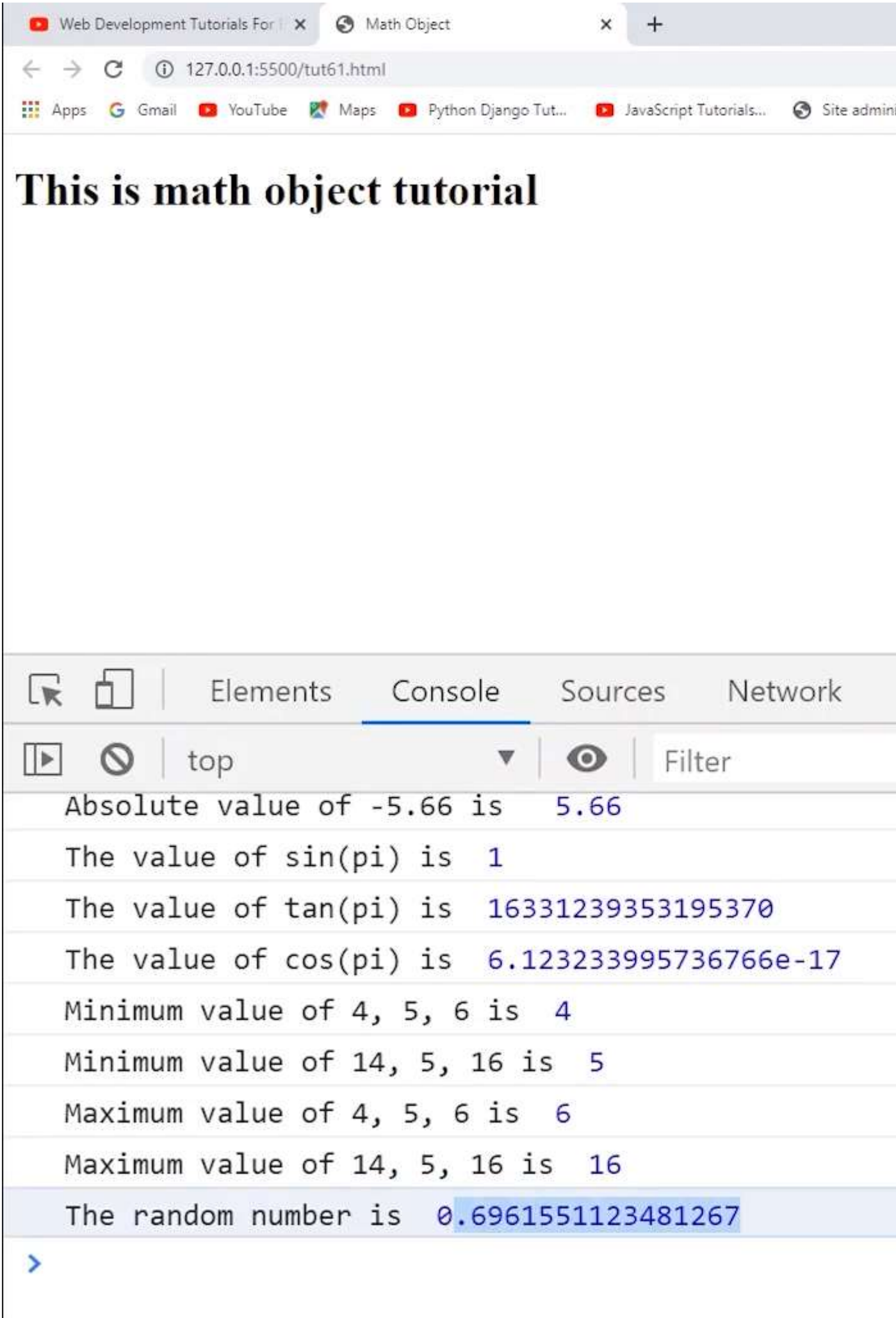
```
console.log("Minimum value of 4, 5, 6 is ", Math.min(4,5, 6));
console.log("Maximum value of 4, 5, 6 is ", Math.max(4,5, 6));
```

This code will return the minimum and maximum number between 4, 5, and 6.

Let us now see how to generate random numbers in JavaScript. We use Math.random() to generate any random number as follows-

```
let r = Math.random();
console.log("The random number is ", r)
```

It will generate any random number between 0-1 as follows-


Web Development Tutorials For
Math Object
127.0.0.1:5500/tut61.html
Apps
Gmail
YouTube
Maps
Python Django Tut...
JavaScript Tutorials...
Site admini
This is math object tutorial
Elements
Console
Sources
Network
top
Filter
Absolute value of -5.66 is 5.66
The value of sin(pi) is 1
The value of tan(pi) is 16331239353195370
The value of cos(pi) is 6.123233995736766e-17
Minimum value of 4, 5, 6 is 4
Minimum value of 14, 5, 16 is 5
Maximum value of 4, 5, 6 is 6
Maximum value of 14, 5, 16 is 16
The random number is 0.6961551123481267

But to generate a random number between **a** and **b**, we can use this particular formula and write as follows-

```
let a1 = 50;
let b1 = 60;
let r50_60 = a1 + (b1-a1)*Math.random();
console.log("The random number is ", r50_60)
```

It will generate a random number between 50 and 60.

This was all about different Math functions and you can use these functions to account for any mathematical related issues on your website. You can learn more by practising.

Full Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Math Object</title>
</head>
<body>
    <div class="container">
        <h1>This is math object tutorial</h1>
    </div>
    <script>
    // Printing the Math Object
    let m = Math;
    console.log(m)

    // Printing the constants from Math Object
    console.log("The value of Math.E is ", Math.E)
    console.log("The value of Math.PI is ", Math.PI)
    console.log("The value of Math.LN2 is ", Math.LN2)
    console.log("The value of Math.SQRT1_2 is ", Math.SQRT1_2)
    console.log("The value of Math.LOG2E is ", Math.LOG2E)

    // Printing the Functions from Math Object
    let a = 34.64534;
    let b = 89;

    console.log("The value of a and b is ", a, b);
    console.log("The value of a and b rounded is ", Math.round(a),
Math.round(b));

    console.log("3 raised to the power of 2 is ", Math.pow(3, 2))
    console.log("2 raised to the power of 12 is ", Math.pow(2, 12))
    console.log("1 raised to the power of 2 is ", Math.pow(1, 2))

    console.log("Square root of 36 is ", Math.sqrt(36))
    console.log("Square root of 64 is ", Math.sqrt(64))
    console.log("Square root of 50 is ", Math.sqrt(50))

    // Ceil and floor
    console.log("5.8 rounded up to nearest integer is ", Math.ceil(5.8))
    console.log("5.8 rounded down to nearest integer is ", Math.floor(5.8))
```

```
    // Abs function
    console.log("Absolute value of 5.66 is  ", Math.abs(5.66))
    console.log("Absolute value of -5.66 is  ", Math.abs(-5.66))

    // Trinonometric Functions
    console.log("The value of sin(pi) is ", Math.sin(Math.PI/2))
    console.log("The value of tan(pi) is ", Math.tan(Math.PI/2))
    console.log("The value of cos(pi) is ", Math.cos(Math.PI/2))

    // Min and max functions
    console.log("Minimum value of 4, 5, 6 is ", Math.min(4,5, 6));
    console.log("Minimum value of 14, 5, 16 is ", Math.min(14,5, 16));
    console.log("Maximum value of 4, 5, 6 is ", Math.max(4,5, 6));
    console.log("Maximum value of 14, 5, 16 is ", Math.max(14,5, 16));

    // Generating a random number
    let r = Math.random();
    // Random number b/w (a, b) = a + (b-a)*Math.random()
    let a1 = 50;
    let b1 = 60;
    let r50_60 = a1 + (b1-a1)*Math.random();
    console.log("The random number is ", r)
    console.log("The random number is ", r50_60)

    </script>
</body>
</html>
```

JavaScript Tutorial: Working with JSON in JavaScript

In this section, we are going to learn about the concept of JSON in JavaScript. JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for **JavaScript Object Notation**
- JSON is a lightweight data-interchange format
- JSON is language independent
- JSON is ***"self-describing"*** and easy to understand

The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. The code for reading and generating JSON data can be written in any programming language.

Make a new file and add an instant boilerplate to get the basic HTML template as usual. Then give the title as **JSON Tutorial** under the <title> tag. Now let us initialize the JSON object and see what the output comes in the console?

```
let jsonObj = {
        name: Amira",
        channel: "CWH",
```

```
    friend: "Jane Folkes",
    food: "Bhindi" //#bhindiLoverSquad
}
console.log(jsonObj)
```

The output we get will be as follows-

Now if we want to transport these properties using JSON, the question arises, we can do it normally one by one; why there is need for JSON? In some cases, we will have nested objects. Then, in that case, it is difficult to transport each property of the object and convert it into a string to parse them. In such cases, we take the help of JSON.

Let us now see, how to *stringify* the above object. If we write as follows-

```
let myJsonStr = JSON.stringify(jsonObj);
console.log(myJsonStr);
```

We will now see an output as follows which is now converted into a string-

The *stringify* function is used to convert a valid JavaScript object into a string.

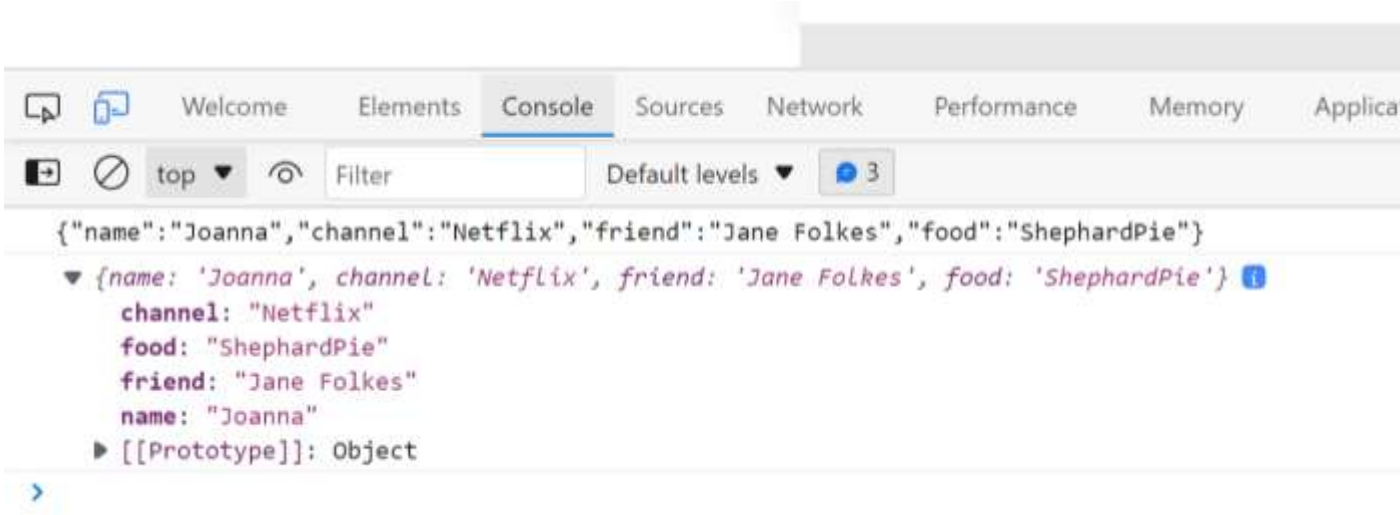
Once, this object is converted into a string, we can apply all the string functions to it. For example, if we want to replace *Amira* with *Joanna,* then we can write as follows-

```
myJsonStr = myJsonStr.replace('Amira', 'Joanna');
console.log(myJsonStr)
```

Now to convert this string again into an object, we can write as follows-

```
newJsonObj = JSON.parse(myJsonStr);
console.log(newJsonObj
```

The final output which you will see will be as follows-



So I believe, you must have understood the concept of JSON. It is very important and is used almost in all programming languages to transport objects.