

## Week 3 Outline

Introduction to JavaScript for Frontend & Backend .....	2
What is JavaScript?.....	2
What JavaScript can do?.....	2
What Can In-Browser JavaScript Do?.....	2
What Can't In-Browser JavaScript Do? .....	3
What Makes JavaScript a Unique language? .....	3
Versions of JavaScript .....	3
Writing in-browser JavaScript and Developer Console.....	3
Internal and External JS .....	5
1.Variables, Data Types and Operators in JavaScript.....	6
2. Strings in JavaScript.....	9
3.String Functions In JavaScript .....	11

### Complementary:

**Fork from this github:**

<https://github.com/mbeaudru/modern-js-cheatsheet>

**Free YouTube for JS beginners by Microsoft:**

<https://youtu.be/PXvI0ln6Nws>

**Project Odin for Self Learner's:**

**20 JS Challenges**

<https://skillcrush.com/blog/projects-you-can-do-with-javascript/>

# Introduction to JavaScript for Frontend & Backend

Week 3, we will learn about the JavaScript language. It is a **client-side** scripting language. Let us now understand more about JavaScript.

## What is JavaScript?

- JavaScript is a client-side scripting language.
- It is used to make web pages alive.
- It is used to programmatically perform actions within the page.
- When JavaScript was created, it was initially called "**Live Script**".
- But Java was a very popular language at that time, so it was decided that positioning a language as a "younger brother" of Java would help.

Now let us understand what the use of JavaScript is.

## What JavaScript can do?

1. JavaScript can execute not only in the browser but also on the server.
2. We will use JavaScript as a client as well as a server-side language.
3. JavaScript has evolved greatly as a language and is now used to perform a wide variety of tasks.

## What Can In-Browser JavaScript Do?

If JavaScript's are used in any website, then it should not be given any low-level CPU permissions like switching off the CPU etc. That is why JavaScript is made with extremely safe permissions that does not have any permission to access low-level CPU usage.

1. JavaScript can add new HTML and change existing HTML from DOM(Document Object Model).
2. It can even react to any events (actions).
3. It can also manage the AJAX requests (GET or POST requests)
4. JavaScript can **get** and **set** cookies and use local storage.

## What Can't In-Browser JavaScript Do?

1. JavaScript cannot read or write to and from a computer hard disk without user permissions.
2. The browser does not allow the JavaScript of any website to collect the AJAX information of the other website because it generates the error of the same-origin **policy**.
3. To summarize, JavaScript can only access the permitted resources but cannot access your documents on personal computers.

These strict policies are developed to make sure that your computer is safe.

## What Makes JavaScript a Unique language?

- The most important thing that makes it a unique language is, it has complete integration of HTML and CSS. They provide it with a lot of extra support.
- Also, it provides the use of simple APIs (Application Programming Interface).
- It also supports the major modern browsers which are enabled by default. If you turn off the feature of JavaScript in the browser, you cannot access any website.

## Versions of JavaScript

JavaScript is such an important language that it requires substantial updates to maintain its different versions. The ECMA is a standard maintained for any of the scripting languages that push for new updates. The ECMA was first launched in 1997.

# Writing in-browser JavaScript and Developer Console

In this section, we will see How to write JavaScript and what are the methods to write it. We will make a new file and add an instant boilerplate to get the basic HTML code to get started. Then give the title as [JavaScript Tutorial](#) under the <title> tag.

The next step is to add the HTML code in the body. We will add the container and a simple paragraph as shown below, just to understand the concept of JavaScript.

```
<body>

  <div class="container">

    <div class="row">

      <p>

        This is a row in this container

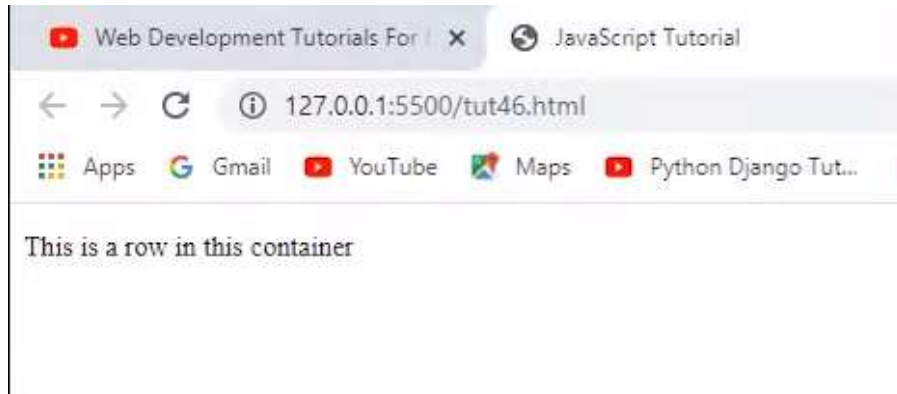
      </p>
```

```
</div>
```

```
</div>
```

```
</body>
```

The result of the above code will look as follows-



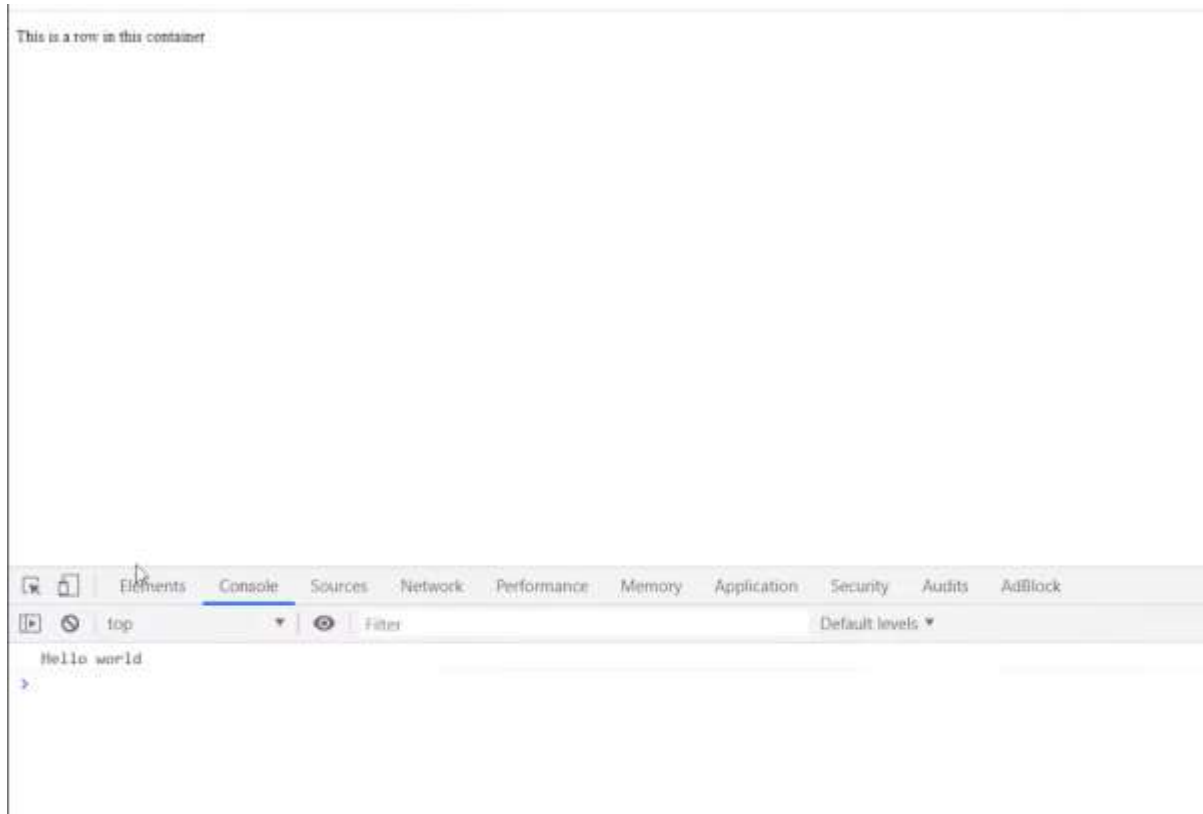
However, our main focus this week is to understand JavaScript. Therefore, we will not do any styling or add CSS here.

We have two options to place the JavaScript in the code. One is in the `<head>` section and the other is after the `<body>` tag. But if we view other professional websites, they place their JavaScript at the end of the `<body>` tag so that the DOM does not get affected. The basic code of JavaScript is as follows-

```
<script>
  //Write your js here
  console.log('Hello world');

</script>
```

If we open the [console](#) section of the browser, we can see the output of the above code as shown below-



Console is the only place, where you will find all the errors made in the code. If you have locked any request or have applied AJAX in the code, the errors regarding all these things will be shown in the console tab itself. Also, we can see the values of different variables if written in the code. Although there may be a chance that if you are using any other browser apart from Chrome like Firefox or Safari, you may find a different console tab, the high-level ideas of all the browsers will be the same. Therefore, it is recommended to use the Chrome browser due to its high developer tools.

### Internal and External JS

As you will be used to form CSS, it is common for JavaScript to be created in its own separate file. You then import it into any HTML documents that you wish to make use of the code:

For example, let's say you have a JavaScript file called **main.js**, You would import it into a HTML document using the following tag:

```
<script src="scripts/main.js"> </script>
```

## 1.Variables, Data Types and Operators in JavaScript

In the last section, we are discussing JavaScript in detail. Moving on, here we will see what are variables in JavaScript and how to create them. The JavaScript variables are the containers for storing data values. Make a new file and add an instant boilerplate to get the basic HTML code.

To understand it, initially, we have to add some HTML code to get started as follows-

```
<body>
  <div class="container">
    <h1>This is a heading</h1>

    <div class="content">
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Laborum atque laudantium vero
      tenetur repellendus consequuntur incidunt consectetur deleniti, reprehenderit dolores reiciendis
      aperiam ducimus aliquid, fugiat ipsum, corporis praesentium quibusdam exercitationem ex unde quae
      libero odio tempora. Voluptatibus odio molestiae esse unde quibusdam fuga accusantium facere
      quae, eum error asperiores itaque hic tempora temporibus illo vitae provident ad debitis libero
      dolorum dignissimos corporis dolore. Quia maxime quidem velit?</p>
    </div>
  </div>
</body>
```

As discussed earlier, **variables are the data that stores value**. Therefore, in the below example **a** and **b** are the variables that store integer and String values respectively.

```
var a = 78;
var b = "Amira";
```

To see the output, we have to write-

```
console.log(a);
console.log(b);
```

The result of both the variables will be displayed in the console tab of the browser. These types of variables are known as **dynamic typing** as you do not require to identify the data type.

If you are a web developer do not deep dive into the knowledge of core programming concepts because it can divert you from becoming a successful web developer. Therefore, it is recommended initially, do not to get in the concepts of Data Structures or OOPS. Once you get the basic command over JavaScript then you can move further.

Let us now understand the [operators in JavaScript](#). There are two types of operators present in JavaScript-

### **Binary Operators and Unary Operators.**

Unary Operators work only on 1 operand. For example, **3+4**. On the other hand, Binary operators work only on 2 operands. For example, **x= x+6**. Here '=' and '+' are two operands.

Let us now understand different types of operators with the help of examples-

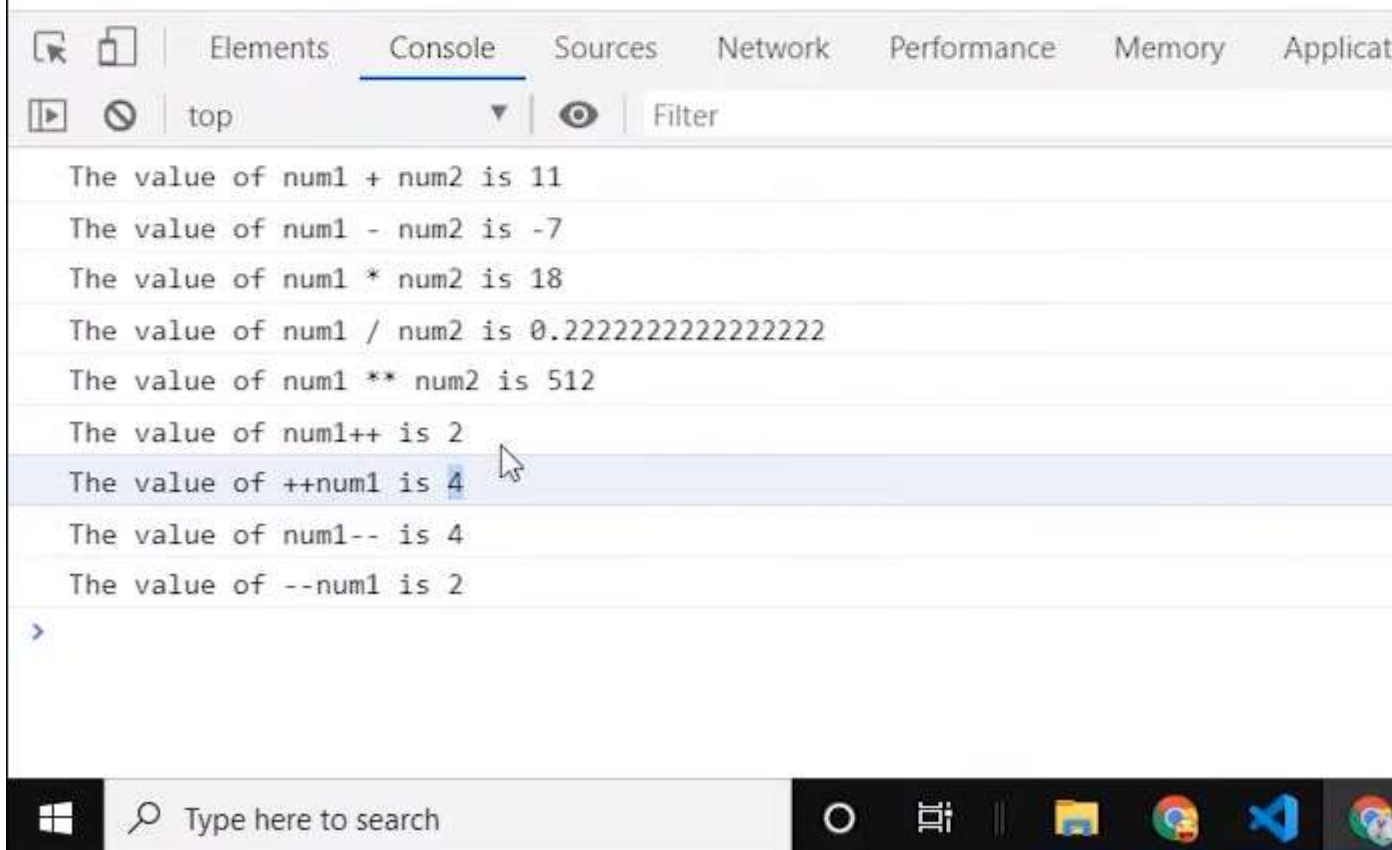
```
var num1 = 2;
var num2 = 9;

// Arithmetic operators in action in JavaScript
console.log("The value of num1 + num2 is " + (num1 + num2));
console.log("The value of num1 - num2 is " + (num1 - num2));
console.log("The value of num1 * num2 is " + (num1 * num2));
console.log("The value of num1 / num2 is " + (num1 / num2));
console.log("The value of num1 ** num2 is " + (num1 ** num2));
console.log("The value of num1++ is " + (num1++));
console.log("The value of ++num1 is " + (++num1));
console.log("The value of num1-- is " + (num1--));
console.log("The value of --num1 is " + (--num1));
```

The output of all the above arithmetic expressions is shown here-

# This is a heading

Lorem ipsum dolor sit amet consectetur adipisicing elit. Laborum atque laudantium vero tenetur exercitationem ex unde quae libero odio tempora. Voluptatibus odio molestiae esse unde quibusd maxime quidem velit?



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a series of log messages showing the results of arithmetic operations on variables `num1` and `num2`. The messages are as follows:

- The value of `num1 + num2` is 11
- The value of `num1 - num2` is -7
- The value of `num1 * num2` is 18
- The value of `num1 / num2` is 0.2222222222222222
- The value of `num1 ** num2` is 512
- The value of `num1++` is 2
- The value of `++num1` is 4
- The value of `num1--` is 4
- The value of `--num1` is 2

A mouse cursor is hovering over the log message 'The value of `++num1` is 4', which is highlighted in blue. Below the console, the Windows taskbar is visible, showing the search bar and several application icons.



## 2. Strings in JavaScript

In this section, we are going to learn about Strings in JavaScript and how to make them, what are the methods to manipulate those Strings, and how to display them in the browser using JavaScript.

**A JavaScript string is zero or more characters written inside the quotes.** The String object is used to represent and manipulate a sequence of characters.

Start by making a new file and add the boilerplate to get the basic HTML code. Name the file as JavaScript | String and String Methods under the <title> tag.

To start, we will add the basic HTML code

To write the JavaScript we will include a <script> tag. For example, if we write-

```
var string = "this";
```

We will see that the "this" word is reflected back on the console tab in the browser. However, apart from the double quotes, we can also write a String in a single quote as 'this'. It is recommended to use double quotes when we are using single quotes between the strings. For example-

```
var string = 'thi"s';
```

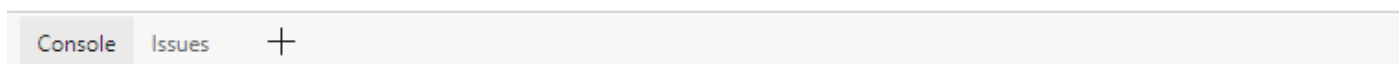
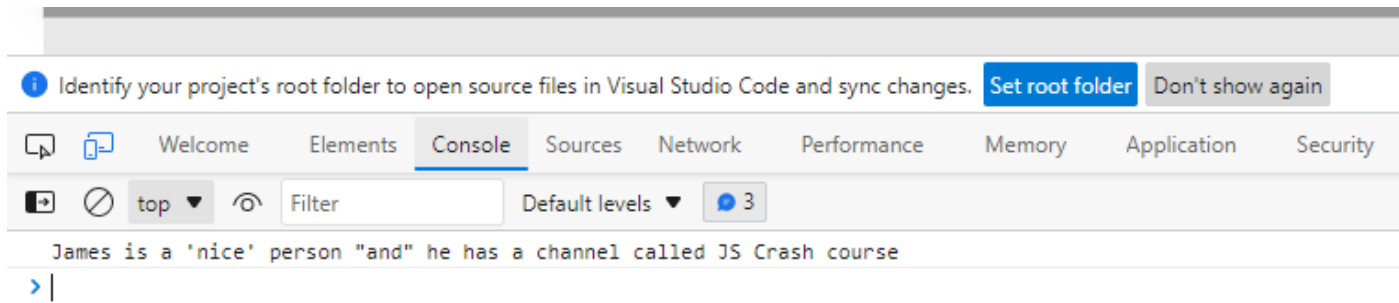
if we declare new variables with strings and concatenate them with a '+' sign as shown here-

```
var string = 'thi"s';  
var name = 'James';  
var message = 'James is a good boy';
```

Let us now understand what are **template literals**. These are written under ` sign. If we write the code as below-

```
var name = 'James';  
var channel = 'JS Crash course';  
var message = 'James is a good boy';  
var temp = `${name} is a 'nice' person "and" he has a channel called ${channel}`;  
$ symbol is used here to pick that particular string from the variable. And if we do,  
console.log(temp);
```

Then the result will be as follows-



The main benefit of using this method is, we can use both double quotes or single quotes to identify the strings.

To extract the number of characters in a particular string we can take the help of the `length` function as shown below-

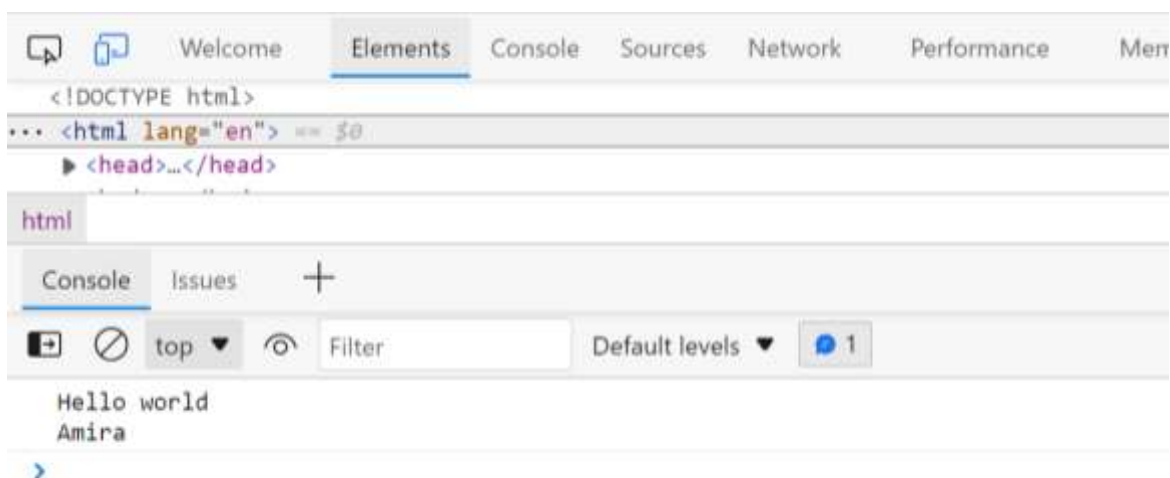
```
var len = name.length;
console.log(`Length of name is ${len}`)
```

It will return the number of characters in the variable name. We also have `escape sequence characters` in the Strings.

For example, if we write-

```
console.log("Hello world\nAmira");
```

The result will be as follows-



`"\n"` is an escape sequence that is used to take the string after it in the next line. It's a new line character.

### 3.String Functions In JavaScript

In this section, we are going to learn about different string functions that are mostly used in JavaScript. These functions are really helpful in extracting the strings either from DOM manipulation or from APIs or AJAX sources. Make a new file and add an instant boilerplate to get the HTML template. Then title it as **JavaScript String Functions** under the <title> tag.

We will begin by writing a very simple JavaScript code under the <script> tag as follows-

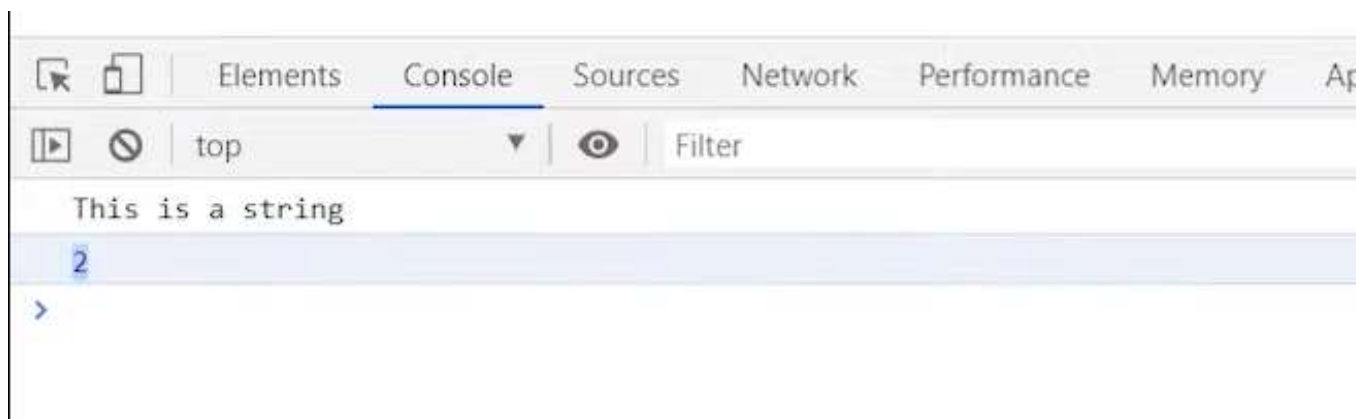
```
var str = "This is a string";  
console.log(str);
```

Now let us see those functions of strings by which we can either modify, alter, slice, break, etc. the strings.

- Suppose we want to locate the position of any word in the string, then we can do as follows-

```
/ First occurrence of a substring  
var position = str.indexOf('is');  
console.log(position)
```

**indexOf()** function is used here to locate the position of any string. It gives the first occurrence of the substring. Here the position of "is" is 2, therefore the output will be as follows-



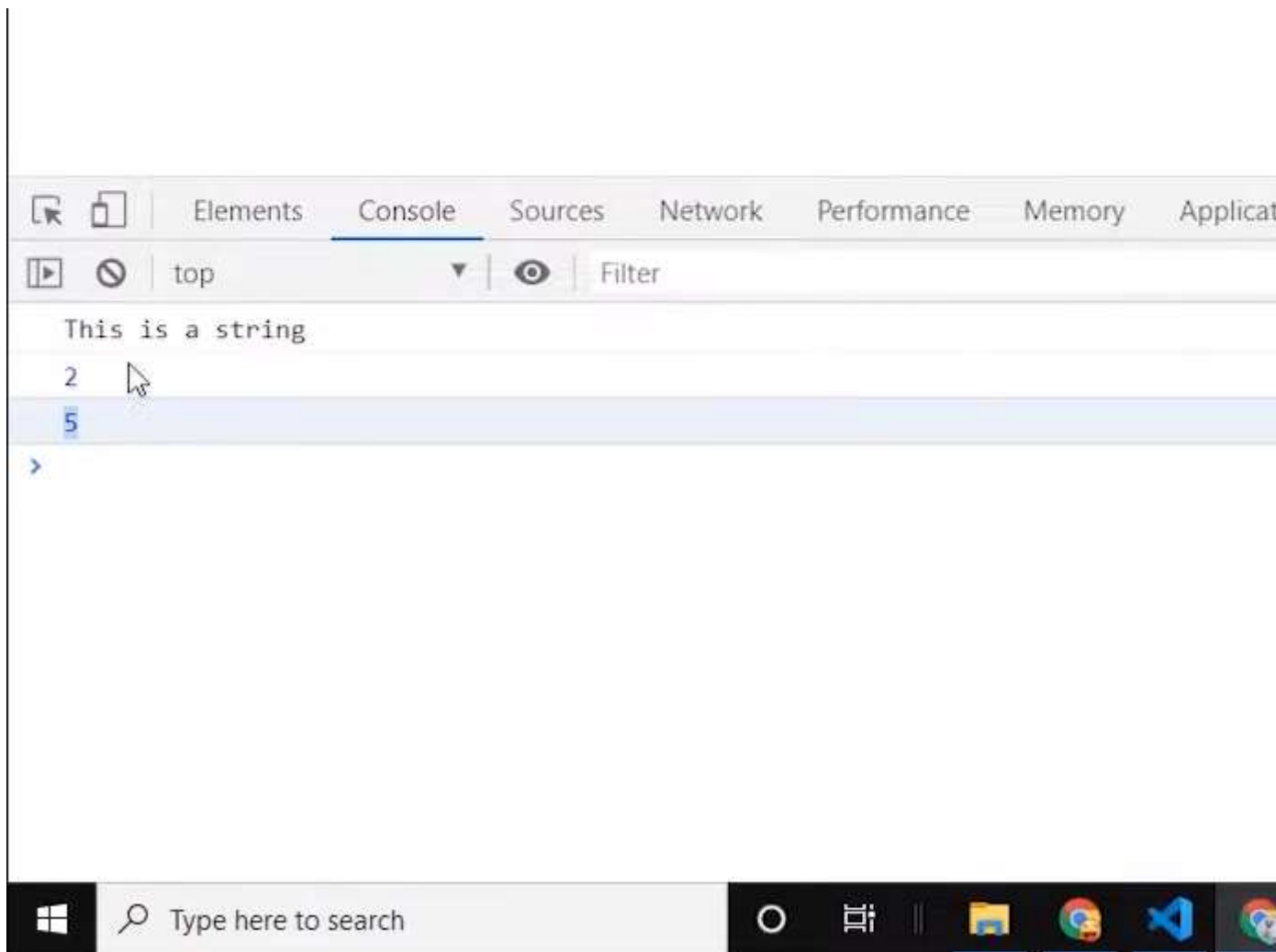
- To get the position of the **last substring**, we can use the function **lastIndexOf()** as follows-

```
// Last occurrence of a substring
```

Wk3\_D1

```
position = str.lastIndexOf('is');  
console.log(position)
```

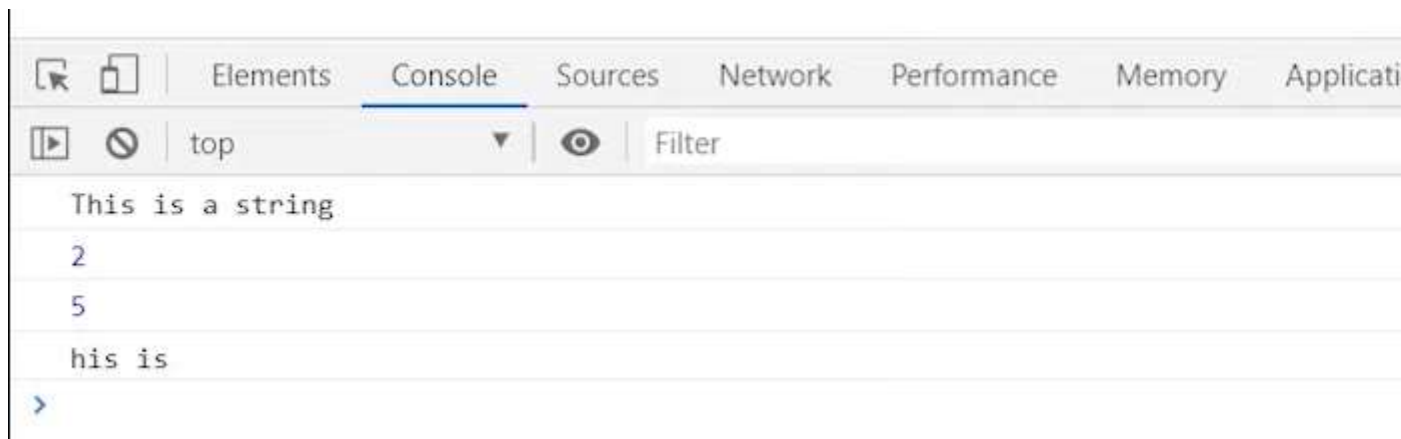
Here the output will be 5 as follows-



- To get a **substring from a string** we can use the **slice()** function as follows-

```
var substr = str.slice(1,7);  
console.log(substr)
```

Here you will get the output of the substring from index numbers 1 to 7 as follows-



However, there are other functions that help to get the substrings like-

```
var substr = str.substring(1,7);
var substr1 = str.substr(1,3);
```

- We can also [replace a particular string with another string](#) with the help of **replace()** function as follows-

```
var replaced = str.replace('string', 'Amira');
console.log(str)
console.log(replaced)
```

In this example, the string "This is a string" has been now been changed to "This is a Amira" and is saved to a variable **replaced**. The original string is still saved in the variable **str**.

We can also convert the whole strong to an [uppercase or lowercase letters](#) as shown below-

```
console.log(str.toUpperCase());
console.log(str.toLowerCase());
```

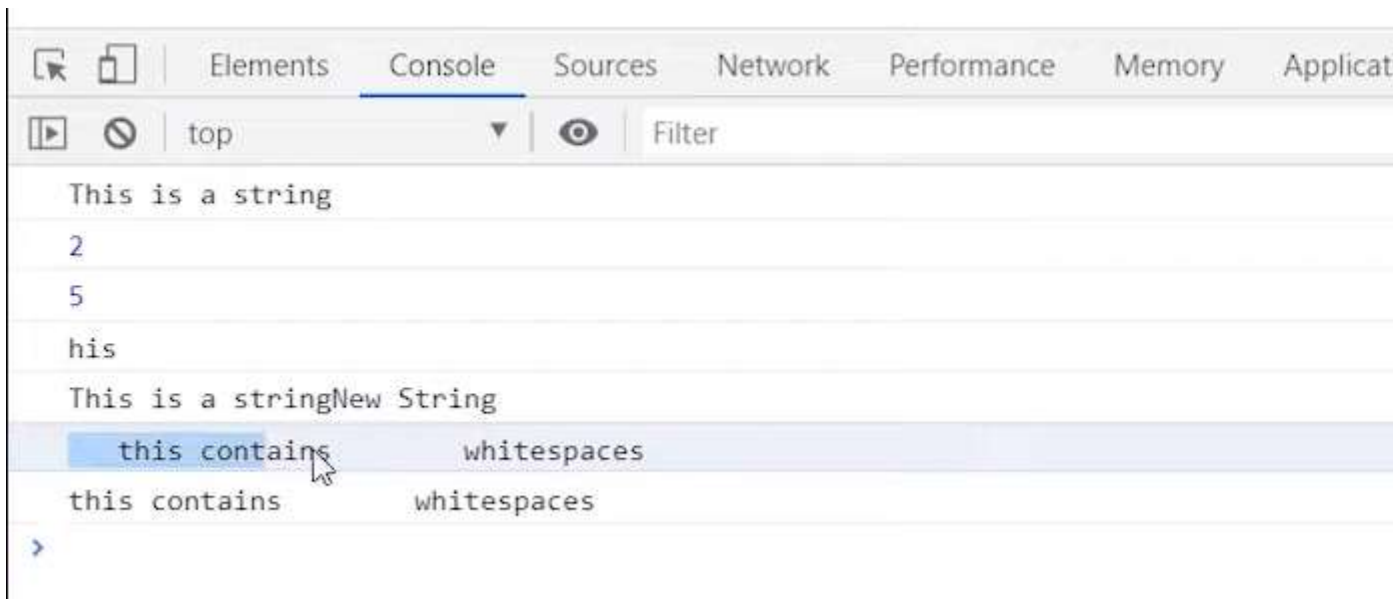
- To concat any two strings, we can take the help of [concat\(\)](#) function as follows-

```
var newString = str.concat('New String')
console.log(newString)
```

- To remove all the white spaces in the string, we can use [trim\(\)](#) function as shown below-

```
var strWithWhitespaces = " this contains    whitespaces ";
console.log(strWithWhitespaces)
console.log(strWithWhitespaces.trim())
```

The result will be as follows-



- To extract any character from a string, we can use `charAt()` function as follows-

```
var char2 = str.charAt(2);  
console.log(char2)
```

It will return 'i' as the output.