



nerdJfpb



nerdJfpb

What is Javascript?

JS



nerd_jfpb





JavaScript, often abbreviated as JS, is a high-level, just-in-time compiled, multi-paradigm programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. (Wiki)





**So mainly javascript is a
programming language which helps
us to make many things!
But mostly we knew it for web
development.**





nerdjfpb



nerdjfpb

Ever select anything from dropdown menu ? That one made by javascript. Javascript changed the whole things of web interactions.



nerd_jfpb





nerdjfpb



nerdjfpb

You can use javascript in everywhere now. Like - Web, mobile app, desktop app, robotics, machine learning etc sector.



nerd_jfpb





nerdjfpb



nerdjfpb

JavaScript has the most job in the world, so you can learn it for a better career!.



nerd_jfpb





nerdjfpb



nerdjfpb

Some learning resources -



nerd_jfpb





JavaScript

Web technology for developers › JavaScript

English ▾

On this Page

- [Tutorials](#)
- [Reference](#)
- [Tools & resources](#)

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with [first-class functions](#). While it is most well-known as the scripting language for Web pages, many [non-browser environments](#) also use it, such as [Node.js](#), [Apache CouchDB](#) and [Adobe Acrobat](#). JavaScript is a [prototype-based](#), multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. [Read more about JavaScript](#).

Related Topics

[JavaScript](#)

Tutorials:

- ▶ [Complete beginners](#)
- ▶ [JavaScript Guide](#)
- ▶ [Intermediate](#)
- ▶ [Advanced](#)

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about [APIs](#) specific to Web pages, please see [Web APIs](#) and [DOM](#).

The standard for JavaScript is [ECMAScript](#). As of 2012, all [modern browsers](#) fully support ECMAScript 5.1. Older browsers support at least ECMAScript 3. On June 17, 2015, ECMA International published the sixth major version of ECMAScript, which is officially called ECMAScript 2015, and was initially referred to as ECMAScript 6 or ES6. Since then, ECMAScript standards are on yearly release cycles. This documentation refers to the latest draft version, which is currently [ECMAScript 2020](#).

Do not confuse JavaScript with the Java programming language. Both "Java" and "JavaScript"





JS Tutorial

[JS HOME](#)[JS Introduction](#)[JS Where To](#)[JS Output](#)[JS Statements](#)[JS Syntax](#)[JS Comments](#)[JS Variables](#)[JS Operators](#)[JS Arithmetic](#)[JS Assignment](#)[JS Data Types](#)[JS Functions](#)[JS Objects](#)[JS Events](#)[JS Strings](#)[JS String Methods](#)[JS Numbers](#)[JS Number Methods](#)[JS Arrays](#)[JS Array Methods](#)[JS Array Sort](#)

JavaScript Tutorial

[◀ Home](#)[Next ▶](#)

JavaScript is the programming language of HTML and the Web.

JavaScript is easy to learn.

This tutorial will teach you JavaScript from basic to advanced.

Examples in Each Chapter

With our "Try it Yourself" editor, you can edit the source code and view the result.

[Example](#)

 Search 5,000+ tutorials

Introduction to JavaScript

JavaScript is a high-level programming language that all modern web browsers support. It is also one of the core technologies of the web, along with HTML and CSS that you may have learned previously. This section will cover basic JavaScript programming concepts, which range from variables and arithmetic to objects and loops.

[Go to the first lesson](#)[View the curriculum](#)

Upcoming Lessons

[Comment Your JavaScript Code](#)[Declare JavaScript Variables](#)[Storing Values with the Assignment Operator](#)[Initializing Variables with the Assignment Operator](#)[Understanding Uninitialized Variables](#)

8:14 PM





nerdJfpb



nerdJfpb

Javascript Types

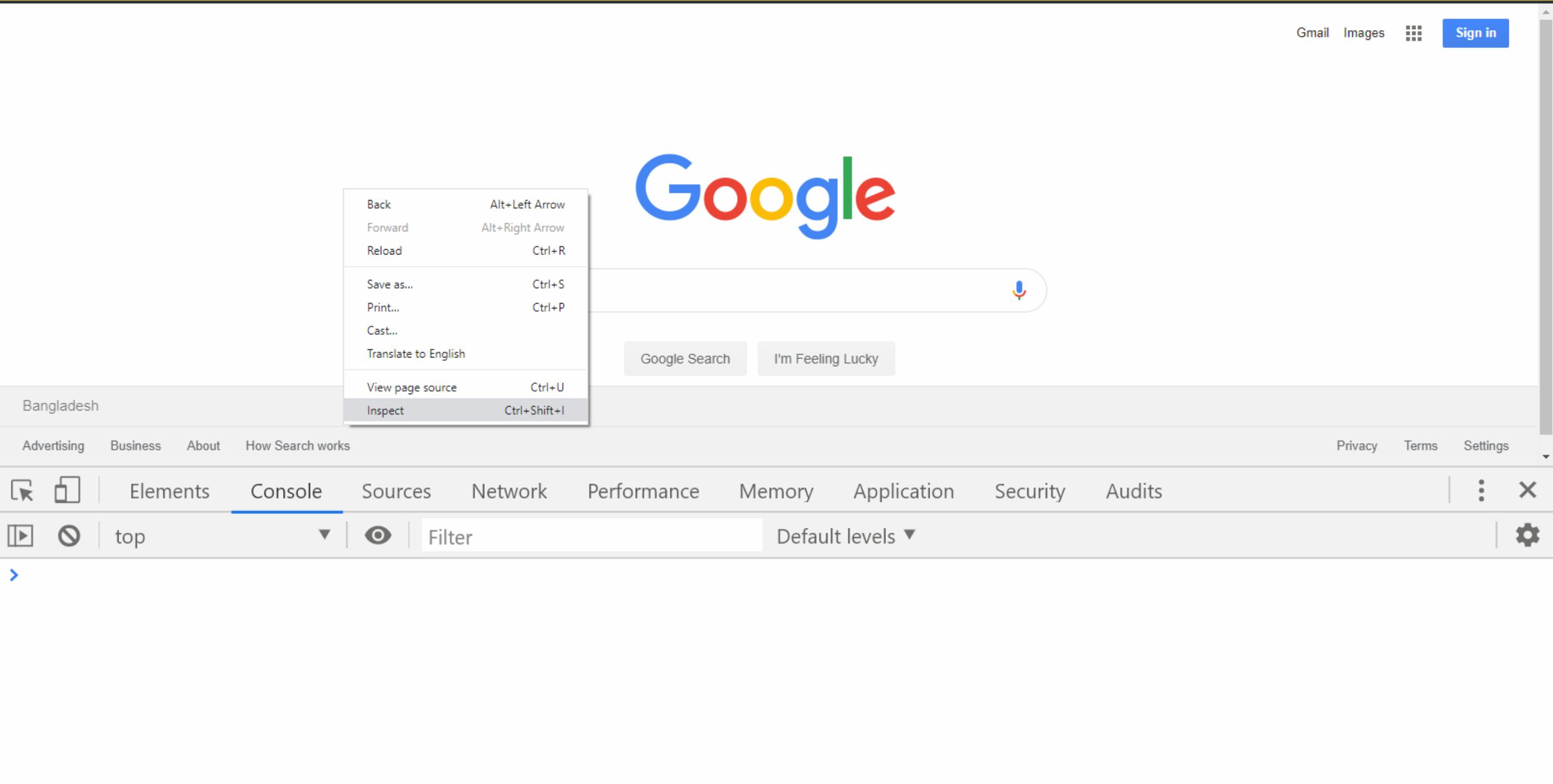
JS



nerd_jfpb



We'll writing js code on the console of google chrome





nerdjfpb



nerdjfpb

**Javascript have 7 types.
You can think types as values.
They are -**



nerd_jfpb





- a. Number**
- b. String**
- c. Boolean**
- d. Undefined**
- e. Null**
- f. Symbol (new in es6)**
- g. Object**





nerdjfpb



nerdjfpb

First we'll talk about Number.



nerd_jfpb





nerdjfpb



nerdjfpb

We can easily add number in javascript. Just adding the '+' operator in middle.

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output displays a series of additions where the '+' operator is placed between two numbers, demonstrating a common JavaScript mistake or trick. The output is as follows:

```
> 100+555
< 655
> 400+50
< 450
> 700+2
< 702
> 4852+75123
< 79975
> |
```



nerd_jfpb





We can do all the arithmetic things here using javascript.

The screenshot shows the 'Console' tab of a browser's developer tools. The console output is as follows:

```
> 15/5
< 3
> 3/4
< 0.75
> 3*51
< 153
> 47-8
< 39
> 42-100
< -58
> |
```





nerdjfpb



nerdjfpb

Working on some more arithmetic logics.

The screenshot shows the Chrome DevTools Console tab. The top navigation bar has tabs for Elements, Console (which is selected), Sources, Network, Performance, Memory, Application, and Settings. Below the tabs is a toolbar with icons for back, forward, and search, followed by dropdown menus for 'top' (with a dropdown arrow), 'Filter' (with an eye icon), and 'Default levels' (with a dropdown arrow). The main area contains the following command history:

```
> (5+4) * 3  
< 27  
> (2+10) / 4  
< 3  
> 2*2+4+9-4+56  
< 69  
> |
```



nerd_jfpb





nerdjfpb



nerdjfpb

What do you think about this '%' ?

The screenshot shows the Chrome DevTools Console tab active. The console output is as follows:

```
> 5 % 2
< 1
> 3 % 4
< 3
> 50 % 1
< 0
> 50 % 5
< 0
> |
```



nerd_jfpb





nerdjfpb



nerdjfpb

**This is actually 'modulo' this gives us
the reminder!**



nerd_jfpb



nerdJfpb



nerdJfpb

Javascript Types

Cont.

JS



nerd_jfpb





nerdjfpb



nerdjfpb

**Today we're going to learn
about String.**



nerd_jfpb





String is simple text type. We need a " for working with the string.

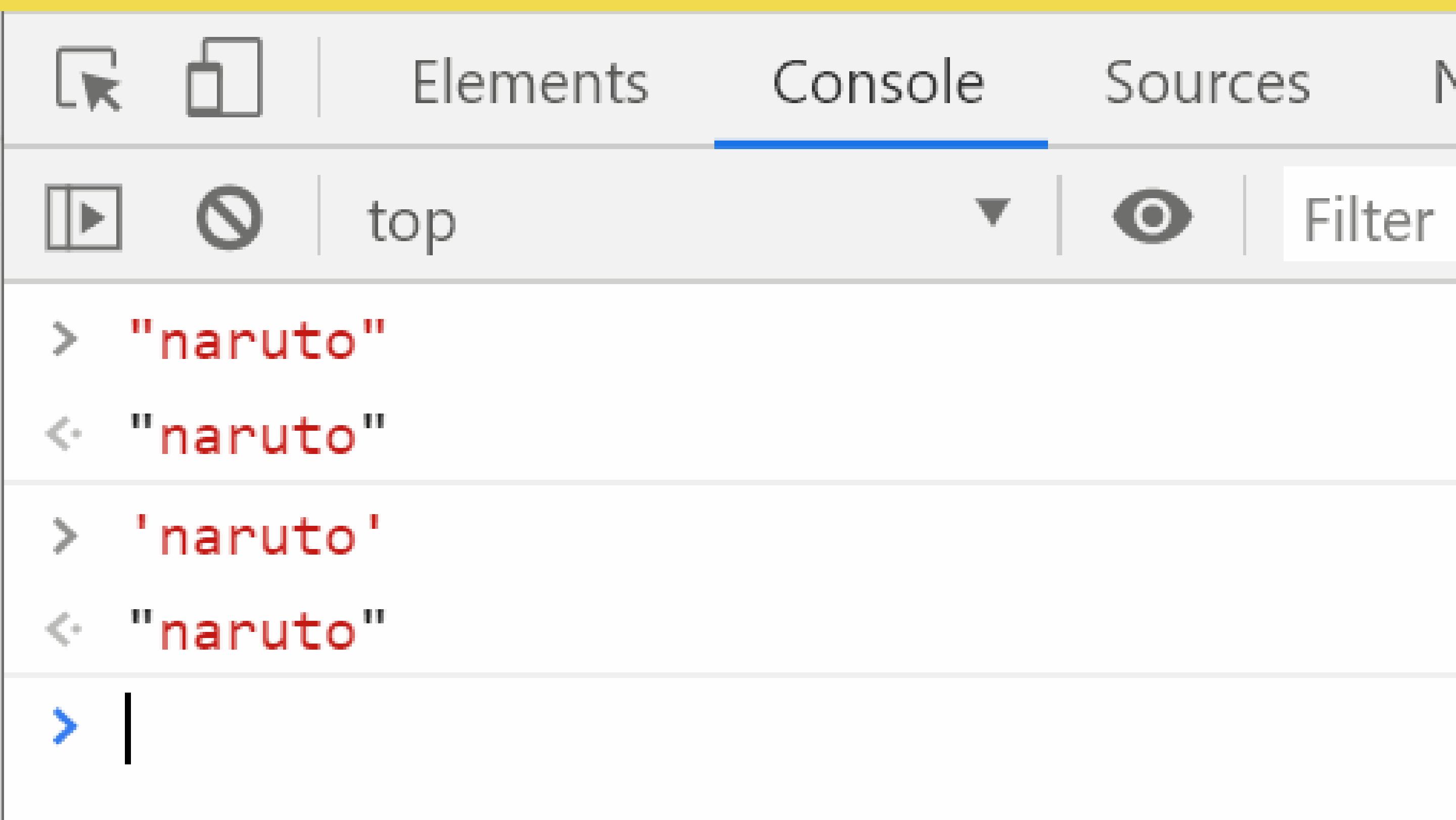
The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following:

```
> "naruto"
< "naruto"
>
```

The text "naruto" is displayed in red, indicating it was input by the user. The other text is standard black output from the console log statement.



We can use single ' too.

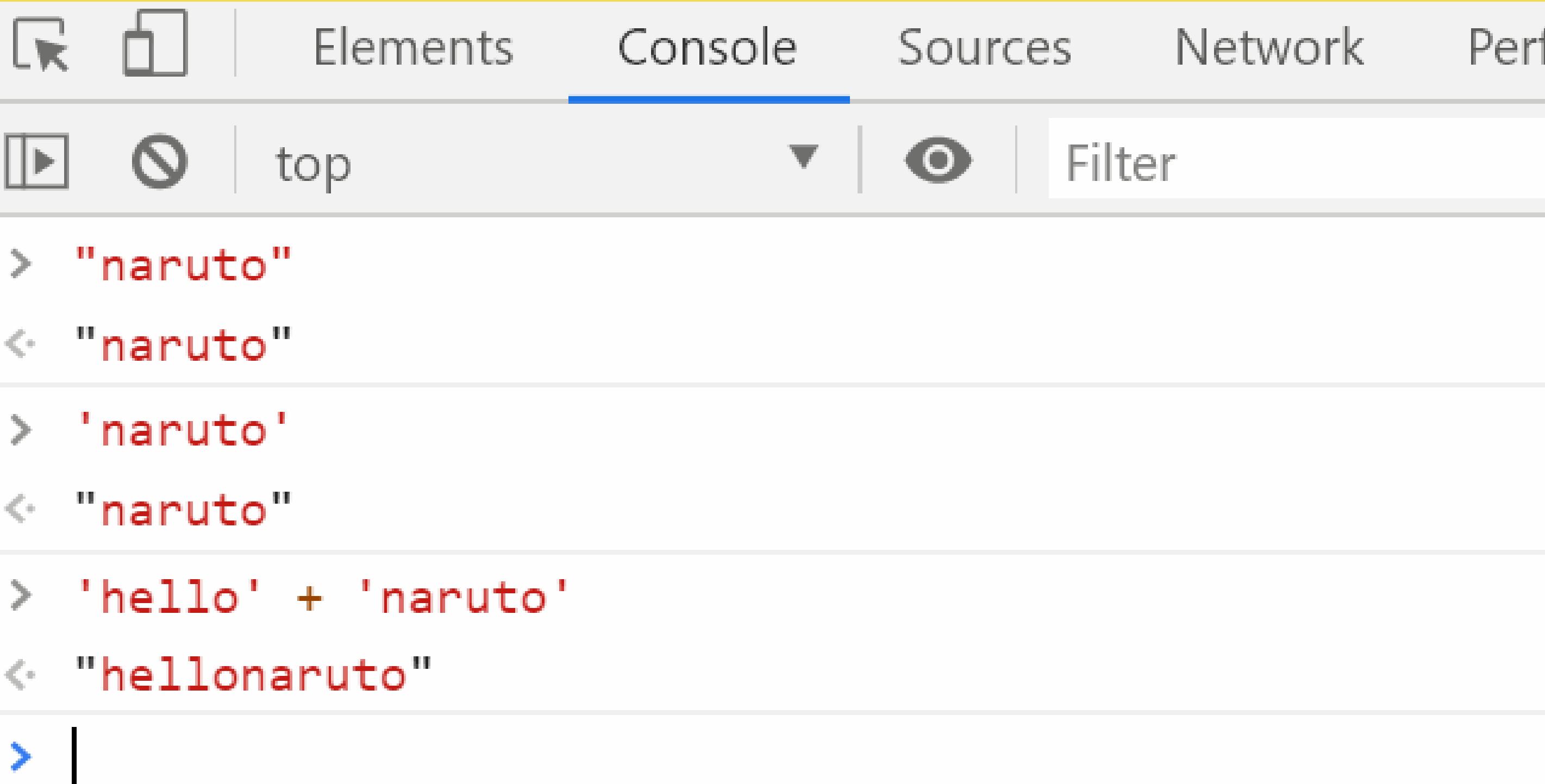


The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The console history displays the following entries:

- > "naruto"
- < "naruto"
- > 'naruto'
- < "naruto"
- >

The first four entries are in red, while the last one is in blue, indicating the current input state. The console also shows the current context as 'top'.

We can add strings using + between two strings.



The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The console output is displayed in red text, illustrating string concatenation:

```
> "naruto"
< "naruto"
> 'naruto'
< "naruto"
> 'hello' + 'naruto'
< "hellonaruto"
> |
```

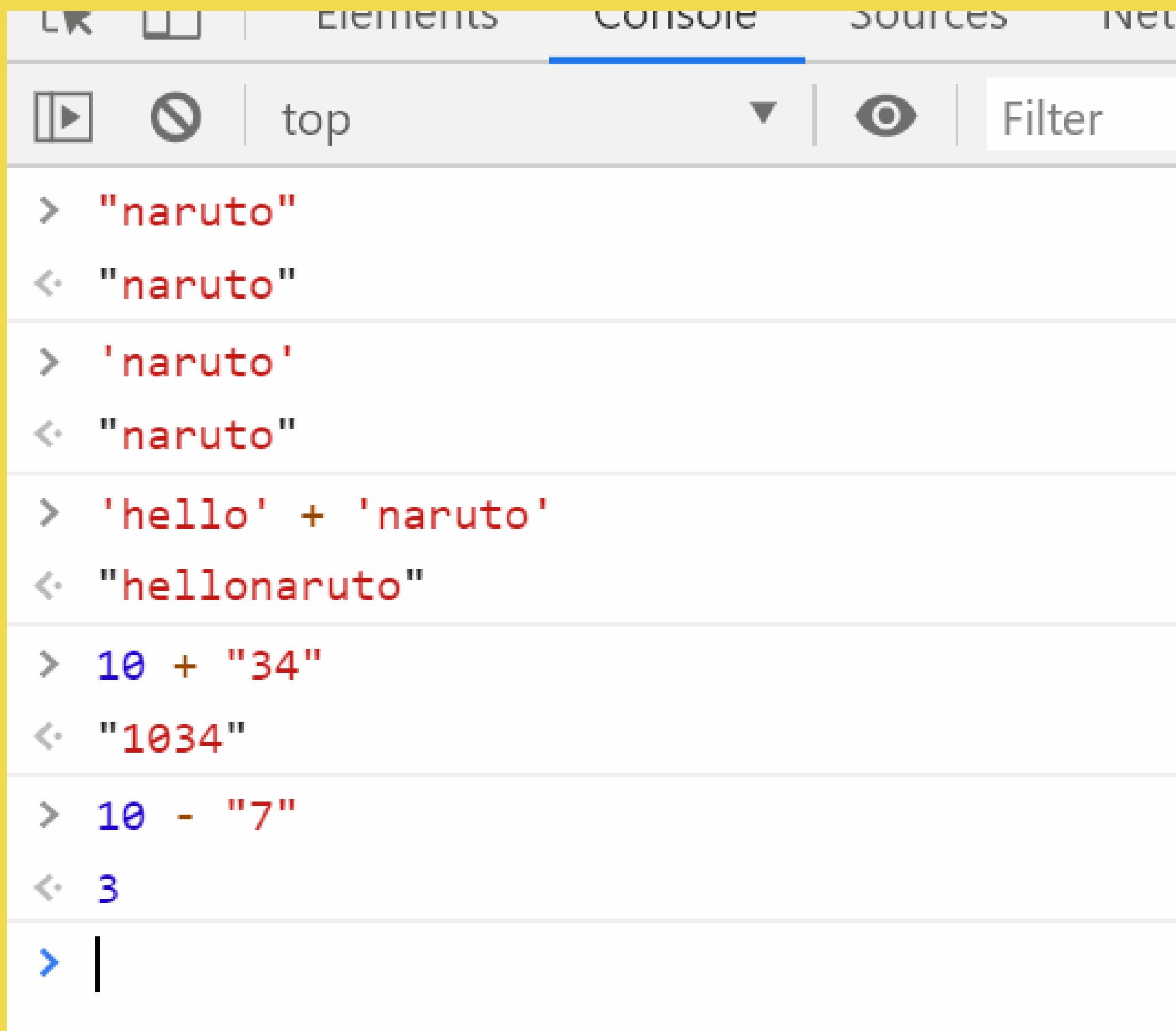


When we add number + string. Javascript made it to string.

```
> "naruto"  
< "naruto"  
> 'naruto'  
< "naruto"  
> 'hello' + 'naruto'  
< "hellonaruto"  
> 10 + "34"  
< "1034"  
> |
```



But when we delete string from a number. It assume that string is a number.



```
elements    console    sources    net
top
> "naruto"
< "naruto"
> 'naruto'
< "naruto"
> 'hello' + 'naruto'
< "hellonaruto"
> 10 + "34"
< "1034"
> 10 - "7"
< 3
> |
```

When we try to delete string from string it give us NaN.

```
| [ ] | top | Filter
> "naruto"
< "naruto"
> 'naruto'
< "naruto"
> 'hello' + 'naruto'
< "hellonaruto"
> 10 + "34"
< "1034"
> 10 - "7"
< 3
> "hello" - "naruto"
< NaN
> |
```



We normally don't do this, When we do any arithmetic operation, we don't contain any string in there. So that we're not facing any issue. .





nerdjfpb



nerdjfpb

**Do you learned something about
string today?**



nerd_jfpb



nerdJfpb



nerdJfpb

Javascript Types

Cont.

JS



nerd_jfpb





nerdjfpb



nerdjfpb

**Today we're going to learn
about Boolean.**



nerd_jfpb





**Boolean is a pretty easy stuff.
Boolean is just about two states.
True & False.**





nerdjfpb



nerdjfpb

See an example -

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
> true
< true
> false
< false
>
```

At the top of the console, there are several icons: a magnifying glass, a square, a play button, a stop button, and a filter input field containing 'top'. Below the tabs, there is a dropdown menu set to 'top'.



nerd_jfpb





nerdjfpb



nerdjfpb

Boolean is 1 and 0 actually!

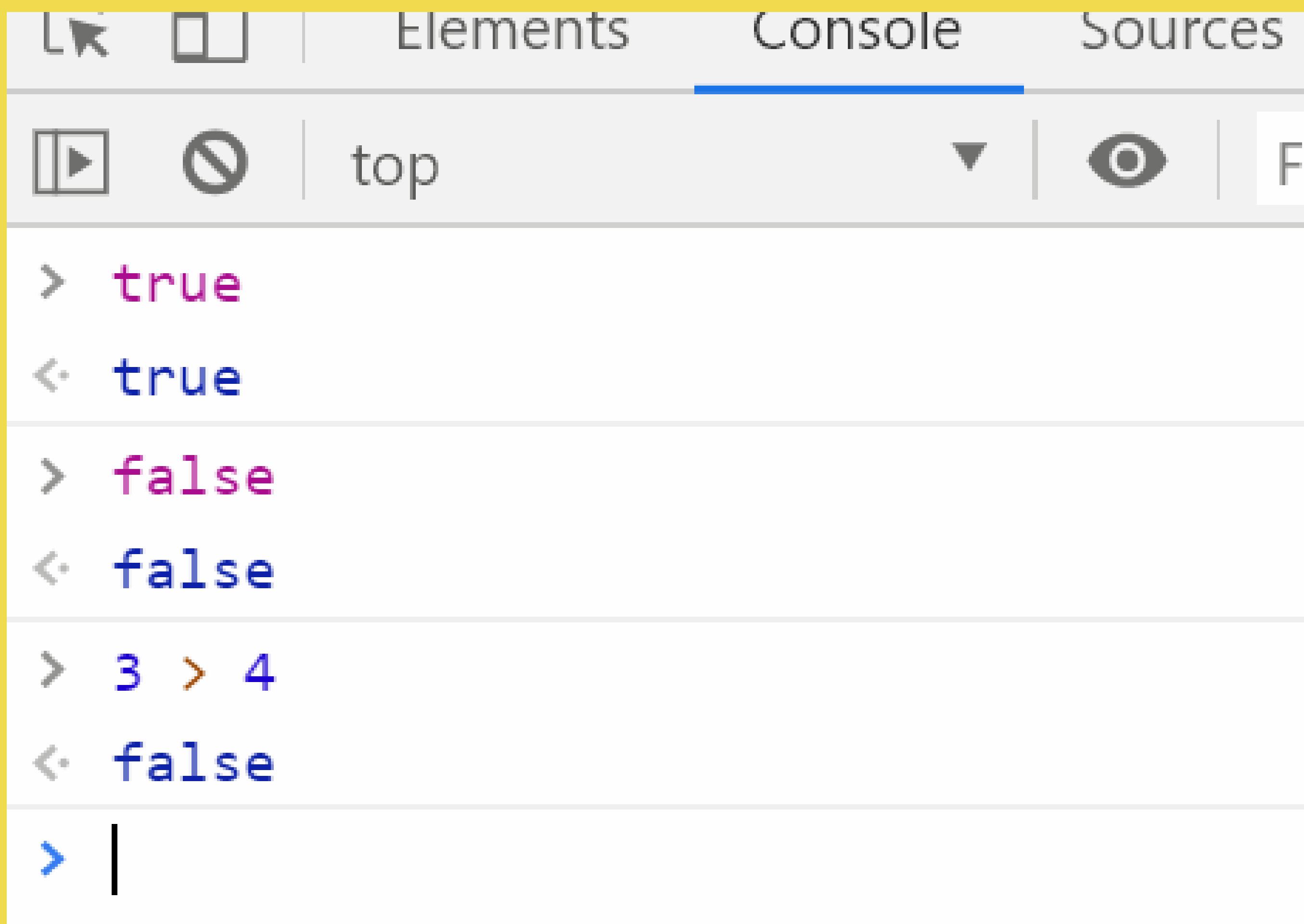


nerd_jfpb



Boolean is really useful in a code.

Example -



The screenshot shows a browser's developer tools with the "Console" tab selected. The console output is as follows:

```
> true
< true
> false
< false
> 3 > 4
< false
> |
```



**In the $3 > 4$ we get the false.
So this is how we can assume
some logic is right or not.**





nerdjfpb



nerdjfpb

Boolean data format is almost available in every programming language.



nerd_jfpb





**Do you want to use the 'true' and
'false' which is boolean or
want to use 1 and 0 which also gave
the same thing.**





nerdjfpb



nerdjfpb

**Let me know
if you have any questions.**



nerd_jfpb



nerdJfpb



nerdJfpb

Javascript Comparisons

JS



nerd_jfpb





**There are some JavaScript types
that we'll discuss right now.
We'll work on that later.
They are - Undefined, Null,
Symbol, Object.**





nerdjfpb



nerdjfpb

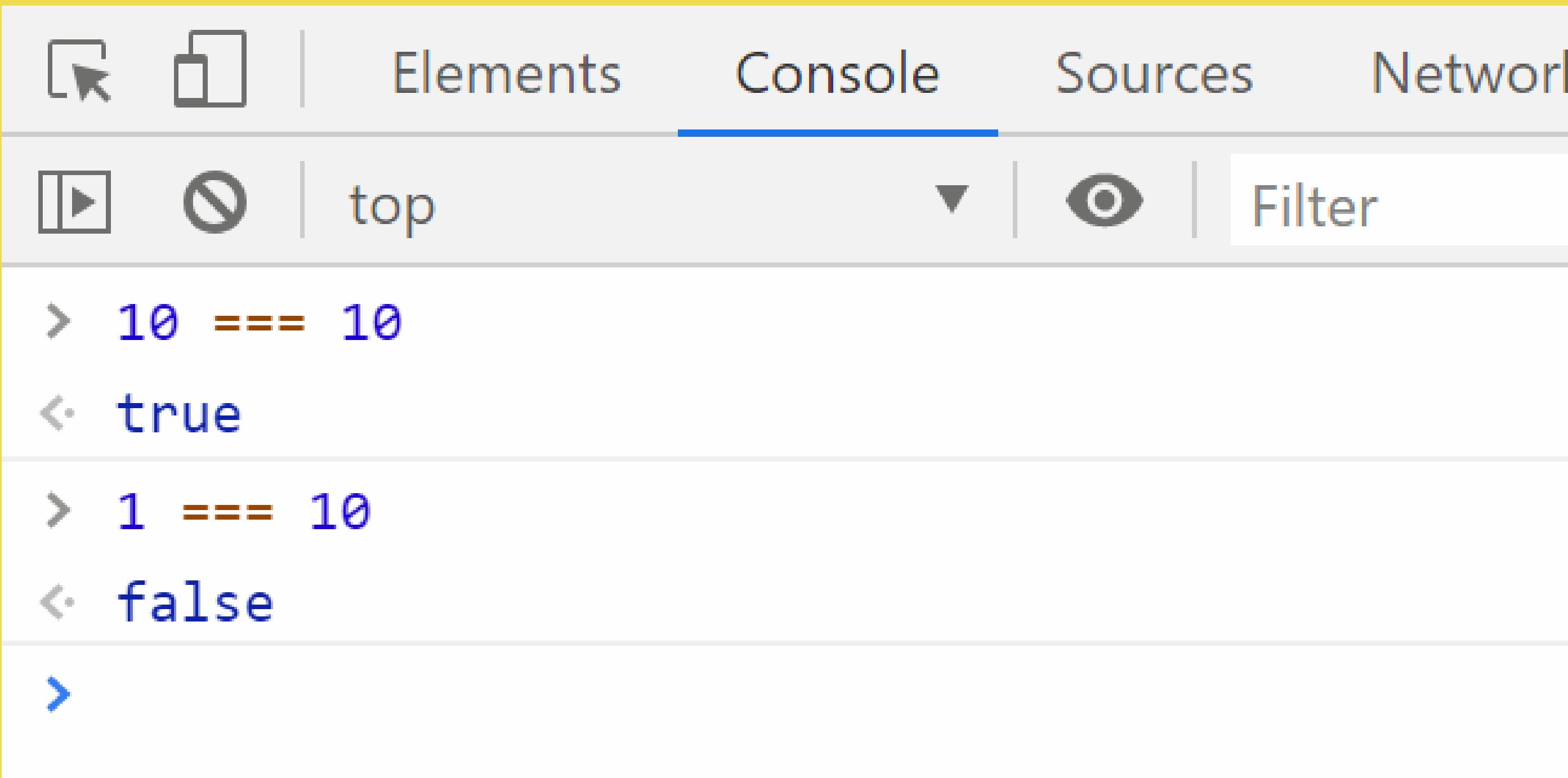
**Today we are going to learn
comparisons in JavaScript.**



nerd_jfpb



Lets start with checking equal.
In javascript we need to use === for
checking if they are same or not.

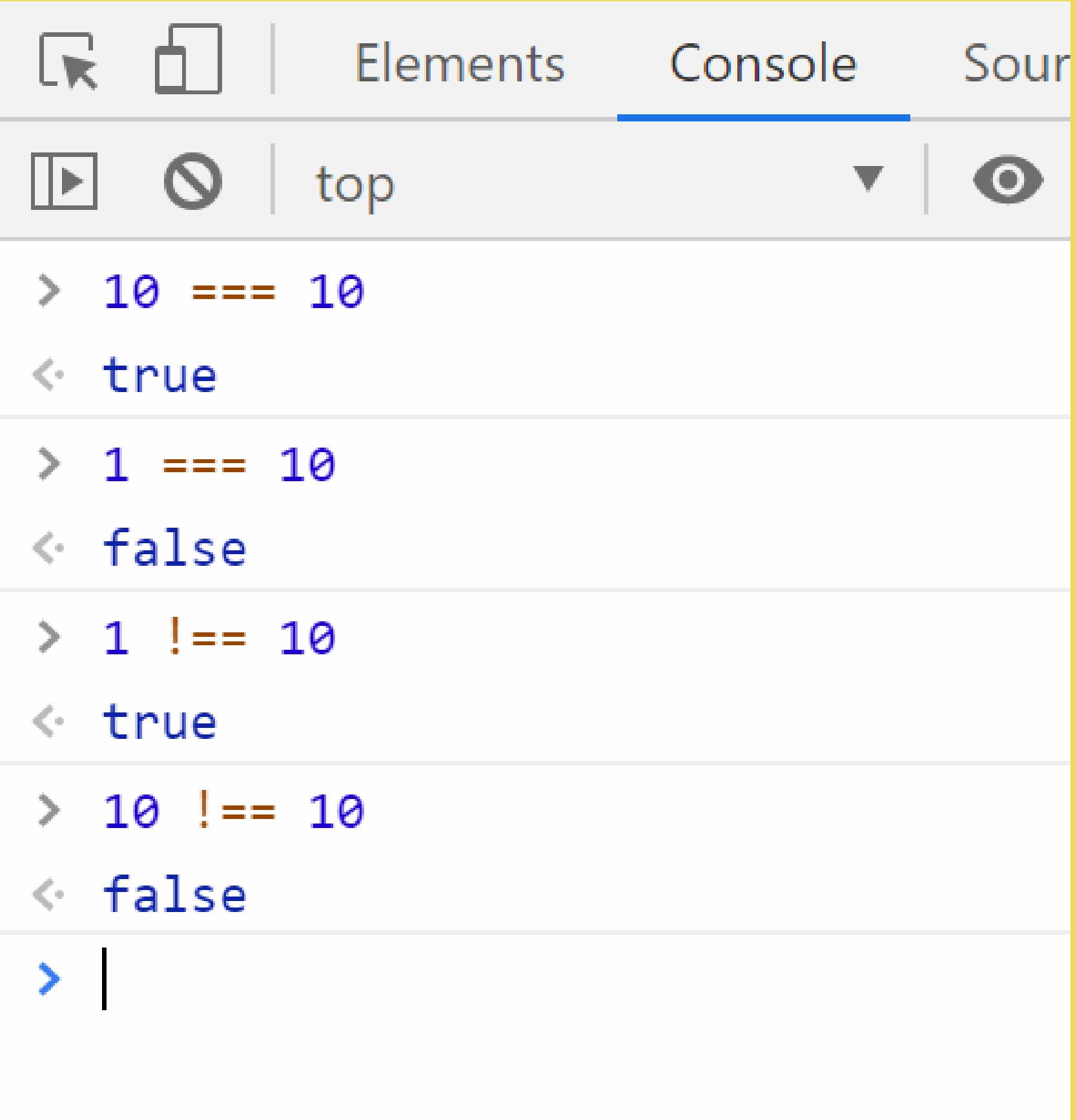


The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The console output is as follows:

```
> 10 === 10
< true
> 1 === 10
< false
>
```

To find out if they are not same
we use `!==`

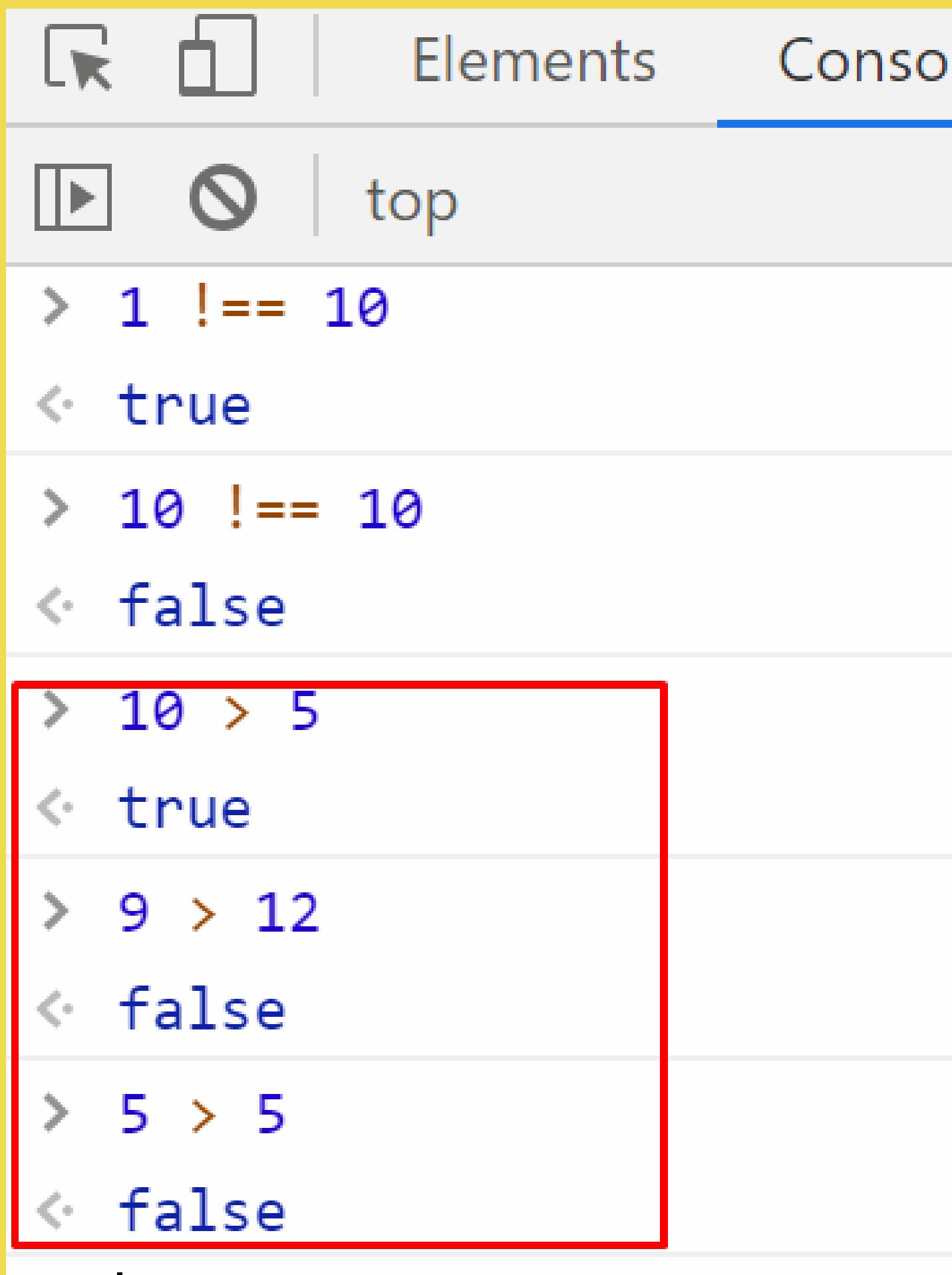
This gives us the boolean result



The screenshot shows a browser's developer tools console tab selected. The console output is as follows:

```
> 10 === 10
< true
> 1 === 10
< false
> 1 !== 10
< true
> 10 !== 10
< false
> |
```

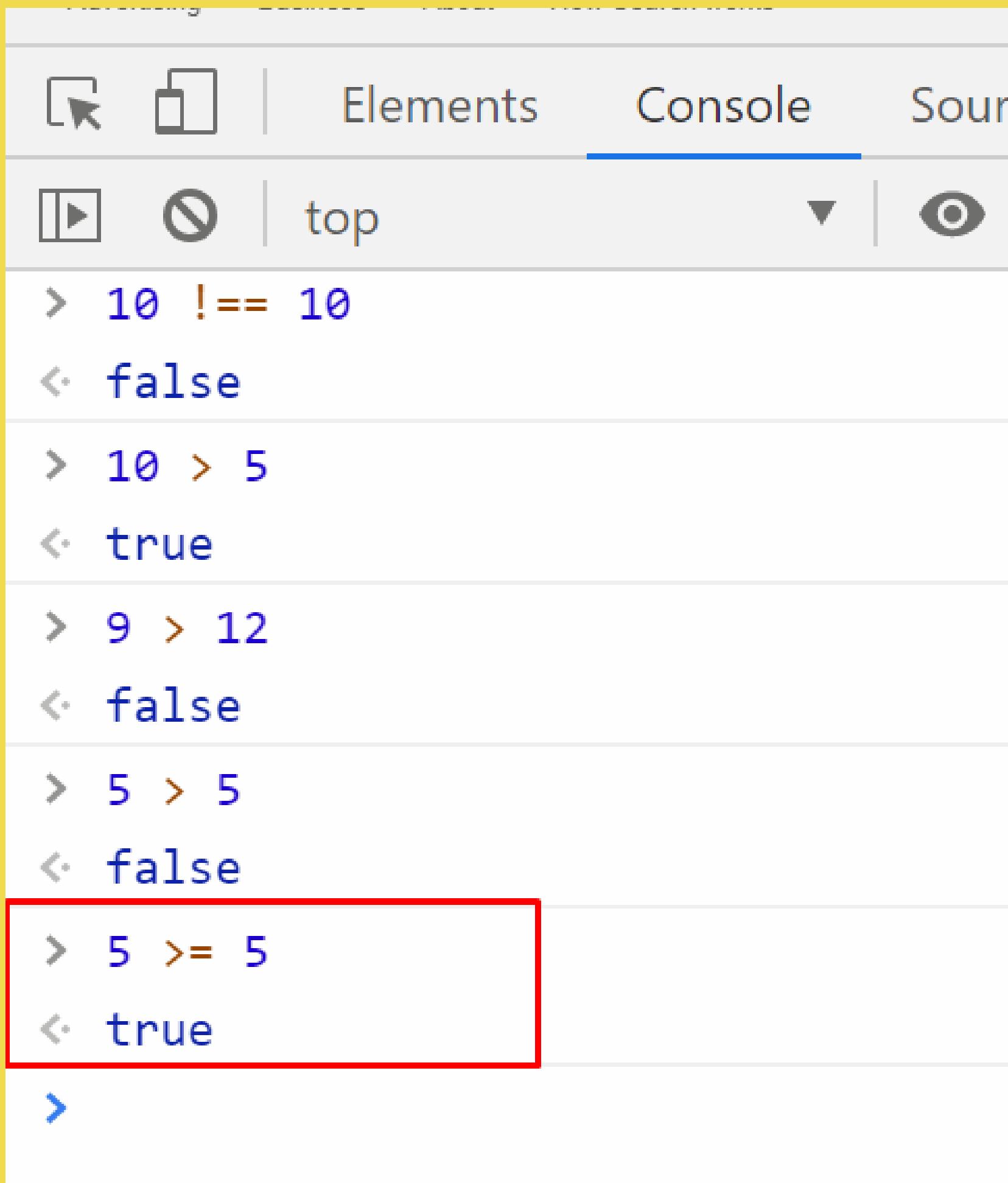
To find some value is greater than some value we use >



```
Elements    Console
top
> 1 !== 10
< true
> 10 !== 10
< false
> 10 > 5
< true
> 9 > 12
< false
> 5 > 5
< false
```

A screenshot of a browser's developer tools console tab. The console shows several JavaScript expressions being evaluated. The first two expressions, `1 !== 10` and `10 !== 10`, result in `true` and `false` respectively. The third expression, `10 > 5`, is highlighted with a red rectangle and results in `true`. The next two expressions, `9 > 12` and `5 > 5`, result in `false` and `false` respectively.

To find some value is greater than and equal to some value we use \geq

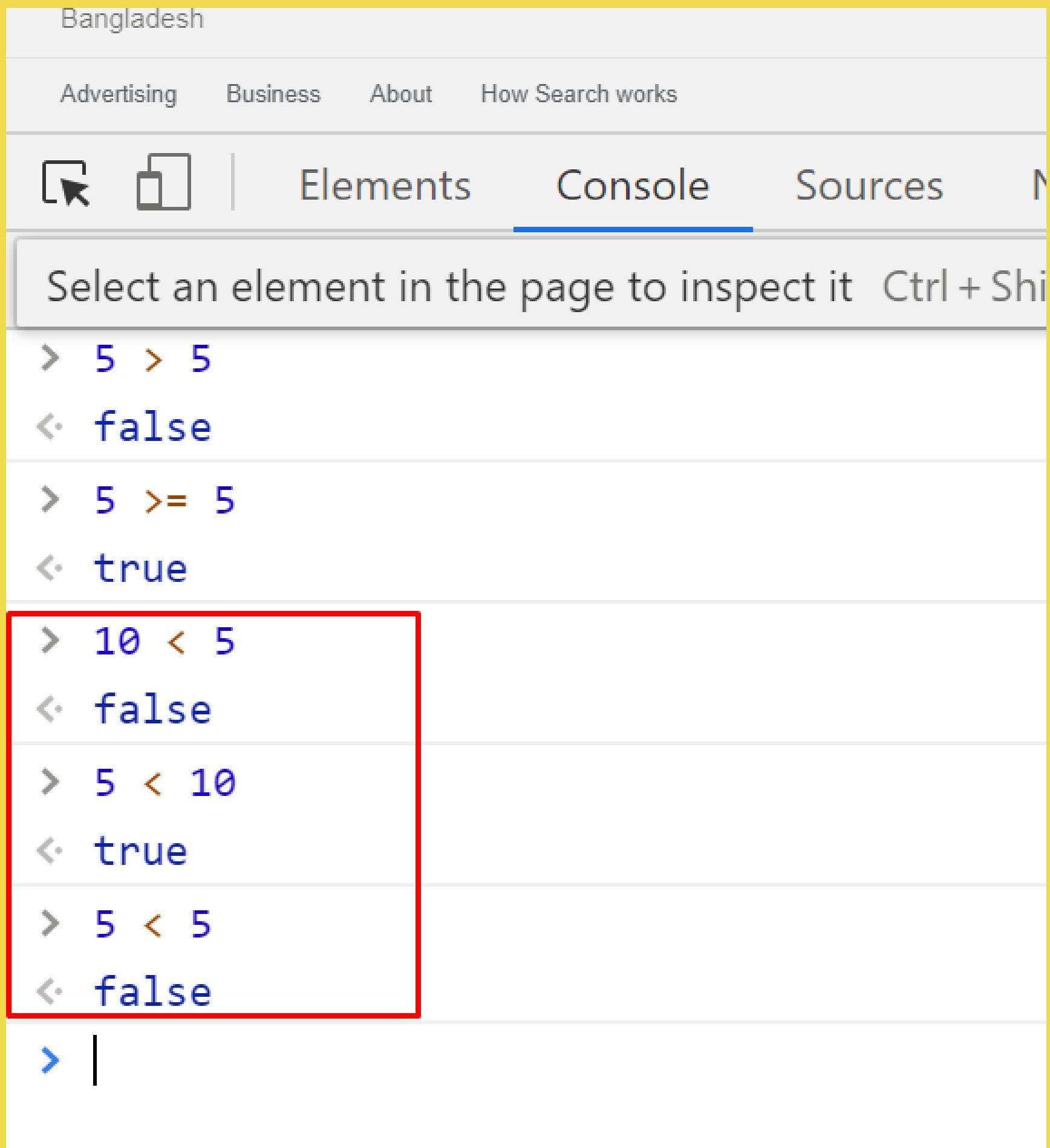


The screenshot shows a browser's developer tools with the "Console" tab selected. The console output is as follows:

```
> 10 !== 10
< false
> 10 > 5
< true
> 9 > 12
< false
> 5 > 5
< false
> 5 >= 5
< true
>
```

The line `> 5 >= 5` and its result `< true` are highlighted with a red rectangle.

To find some value is smaller than some value we use <



Bangladesh

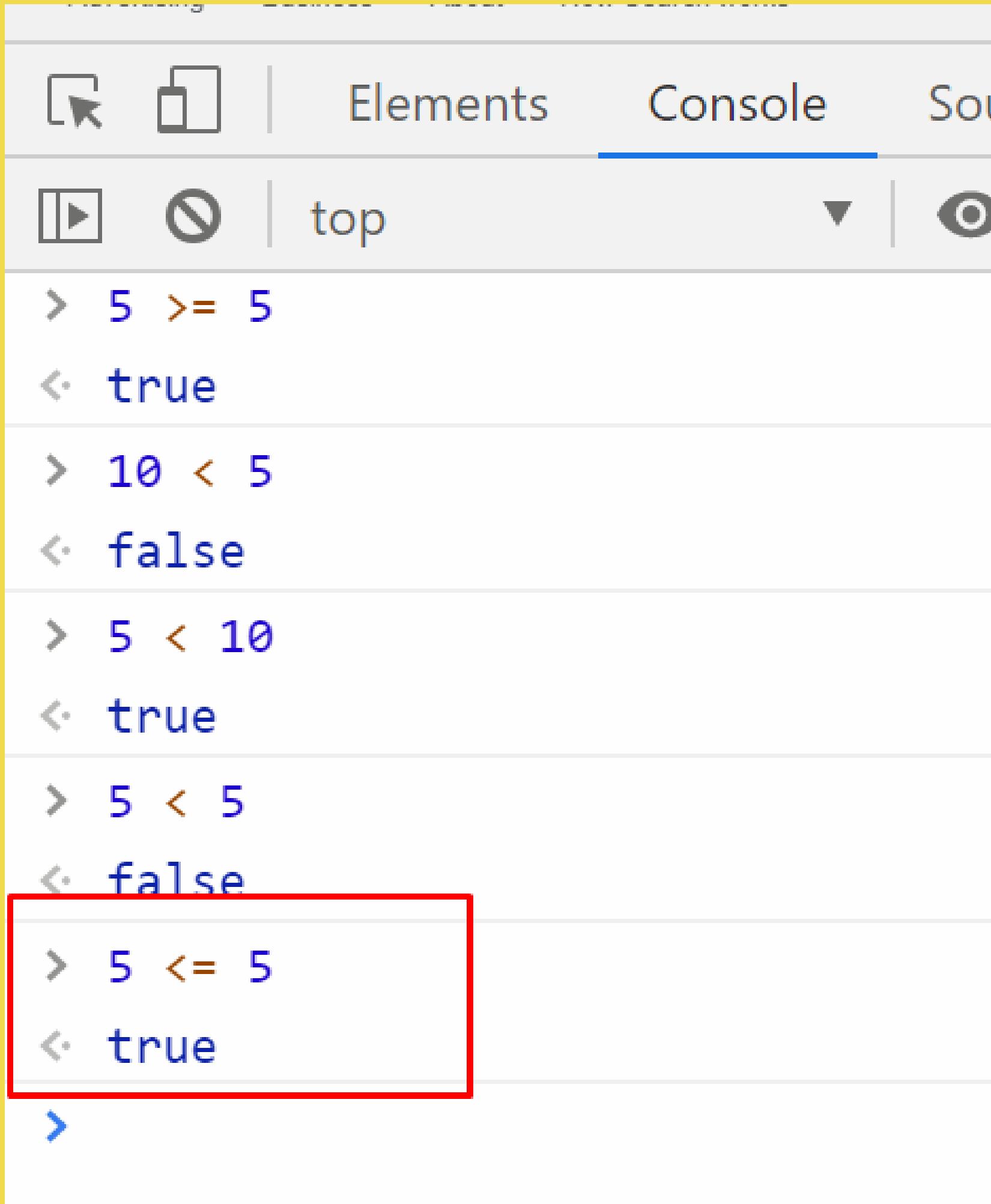
Advertising Business About How Search works

Elements Console Sources

Select an element in the page to inspect it Ctrl + Shift + I

```
> 5 > 5
< false
> 5 >= 5
< true
> 10 < 5
< false
> 5 < 10
< true
> 5 < 5
< false
> |
```

To find some value is smaller than and equal to some value we use \geq



The screenshot shows a browser's developer tools console tab selected. The console output is as follows:

```
> 5 >= 5
< true
> 10 < 5
< false
> 5 < 10
< true
> 5 < 5
< false
> 5 <= 5
< true
>
```

The last two lines, which demonstrate the use of the \leq operator, are highlighted with a red rectangular box.



nerdjfpb



nerdjfpb

**Do you understand
the comparisons signs now
in JavaScript ?**



nerd_jfpb



nerdJfpb



nerdJfpb

Javascript Variables

JS



nerd_jfpb





**Up until now,
we are not making to remember
something. Everything wasn't stored.
But from today we'll store some value
in some variables.
It's like $x = 5$ in math!**





**We can do different things with
the variables.
Like add 5 on the variable or
change the string stored there.**





**In JS we use `var` to initial
a variable.**

**There are some other methods also,
We'll learn those method later.
Because currently we are working
with basic.**





Now we can do `var name = "nerdjfpb"` and some other codes. For now don't think about the undefined.

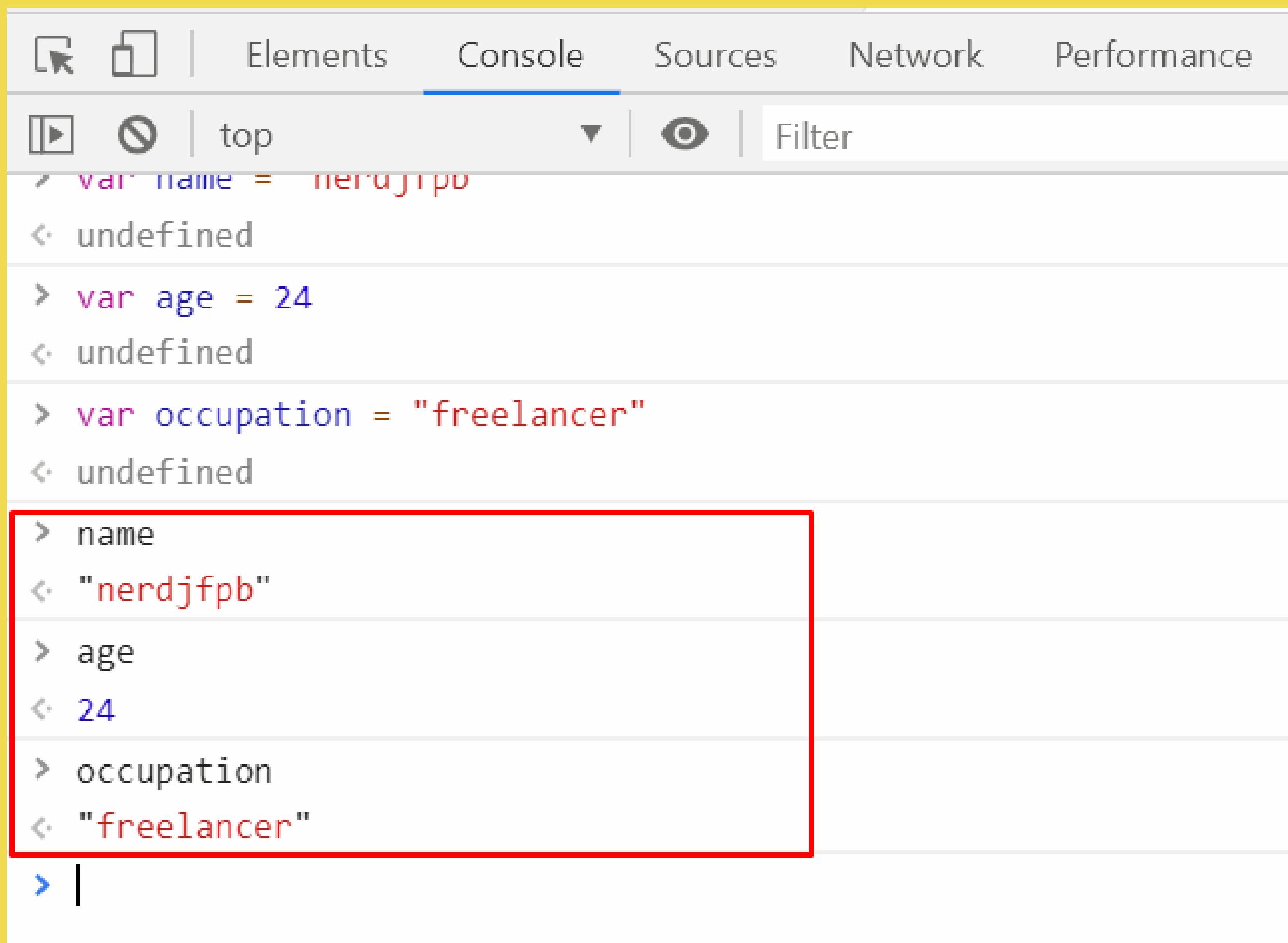
The screenshot shows the browser's developer tools with the 'Console' tab selected. The interface includes tabs for Elements, Console, Sources, Network, and Performance, along with icons for play, stop, and filter. The console output shows the execution of several JavaScript statements:

```
var name = "nerdjfpb"
< undefined
> var age = 24
< undefined
> var occupation = "freelancer"
< undefined
> |
```

The first statement, `var name = "nerdjfpb"`, is highlighted with a red box around the variable name and value. The output of this statement is `< undefined`. The subsequent statements define `age` and `occupation` variables, both resulting in `< undefined` outputs. A cursor is visible at the end of the final line.



Now we can get value by just calling the variable name.

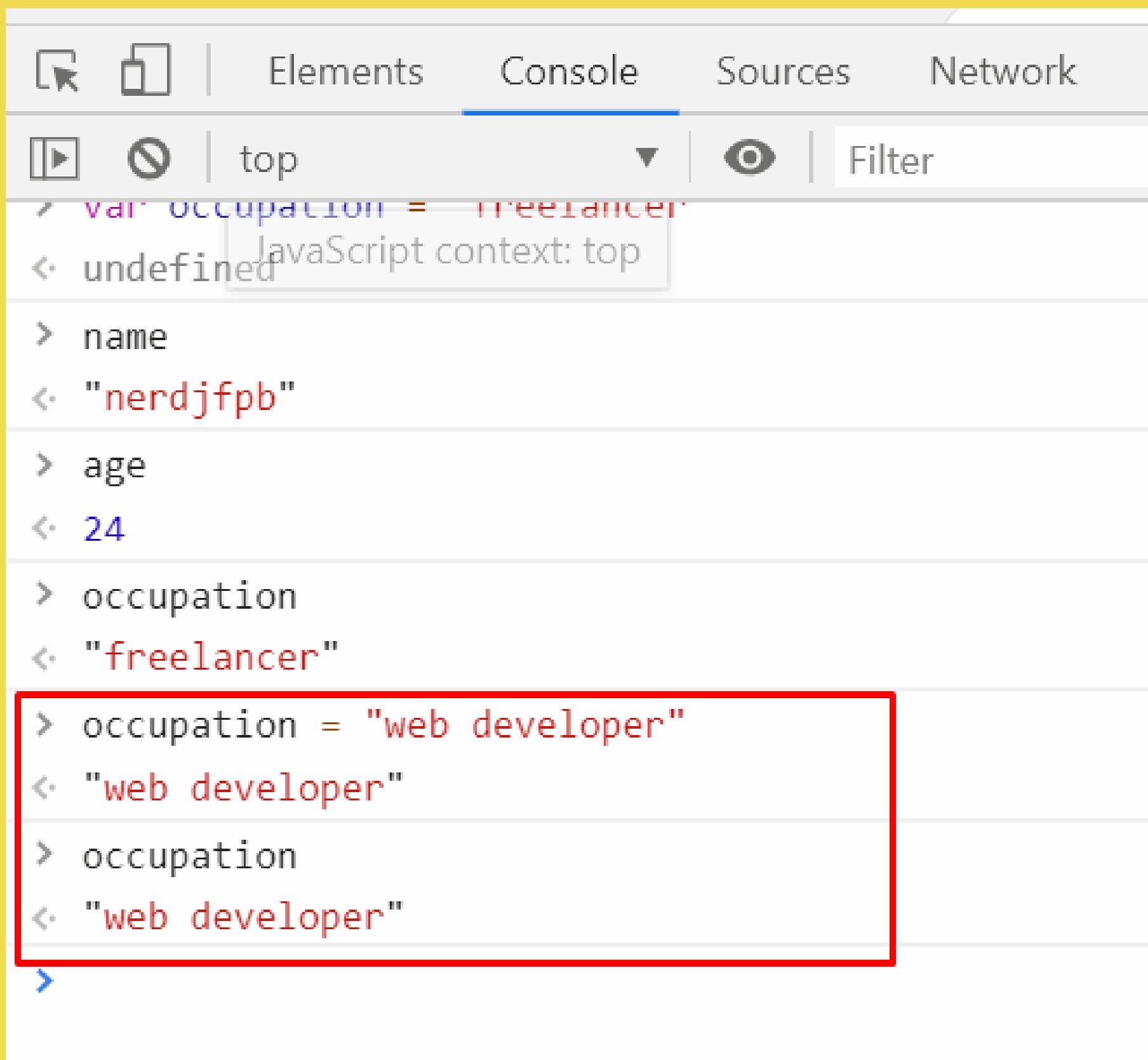


The screenshot shows the browser's developer tools with the 'Console' tab selected. The console output displays the following code and its execution results:

```
var name = nerdjfpb
< undefined
> var age = 24
< undefined
> var occupation = "freelancer"
< undefined
> name
< "nerdjfpb"
> age
< 24
> occupation
< "freelancer"
>
```

The lines starting with '>' represent the user's input, while the lines starting with '<' show the resulting values. The first three lines ('name', 'age', 'occupation') are not highlighted. However, the assignment of 'name' and its subsequent retrieval are highlighted with a red rectangle, demonstrating how a variable can be defined and then used later.

We can do different stuff with the variable, we can change a variable just to new value and it will work like -change the occupation to web developer.



The screenshot shows the Chrome DevTools console tab. At the top, there are tabs for Elements, Console, Sources, and Network. The Console tab is active, indicated by a blue underline. Below the tabs is a toolbar with icons for play, stop, and filter, followed by the text "top" and a dropdown menu set to "JavaScript context: top". The main area displays a list of variables and their values:

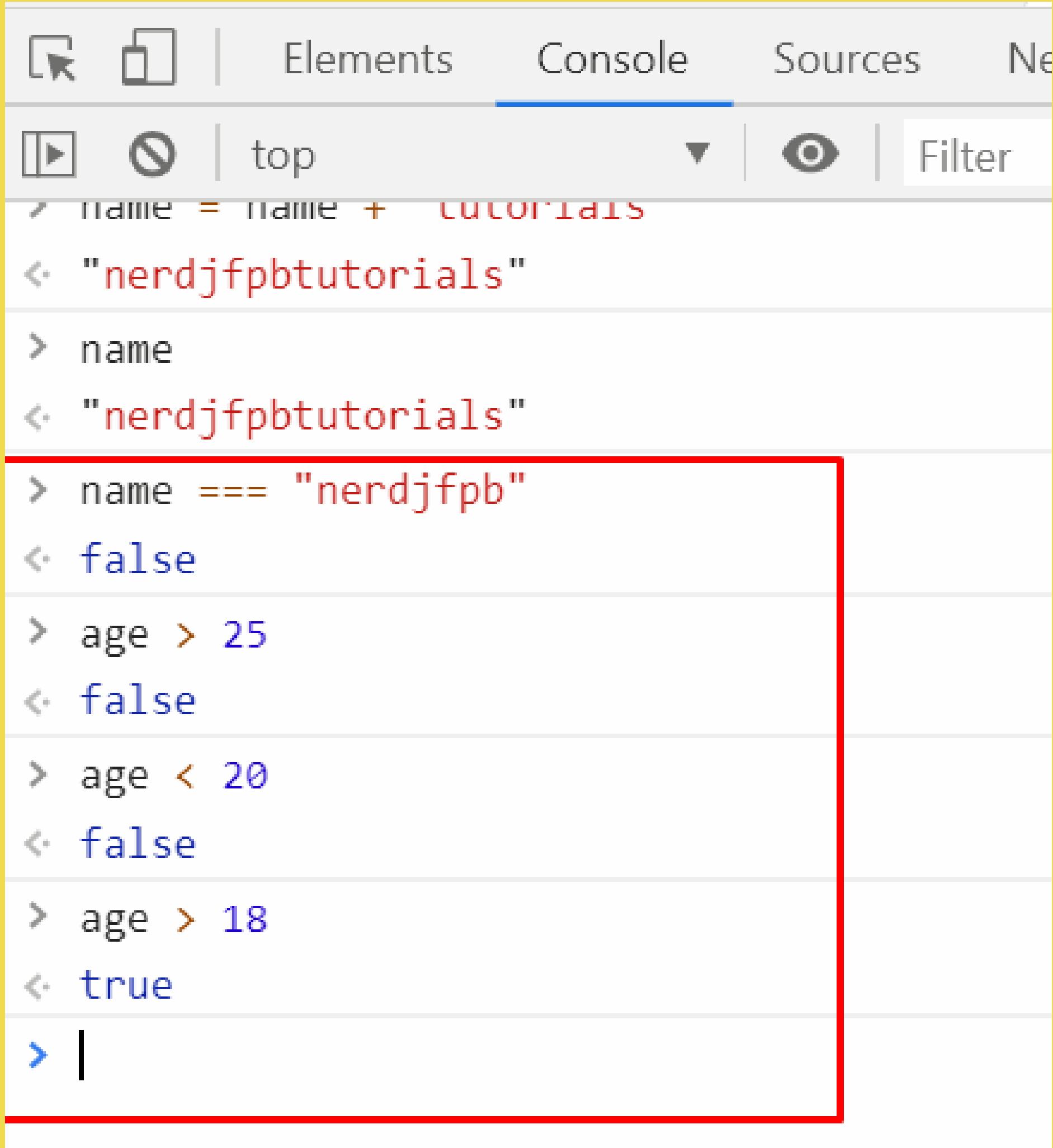
- > var occupation = "freelancer"
- < undefined JavaScript context: top
- > name
- < "nerdjfpb"
- > age
- < 24
- > occupation
- < "freelancer"
- > occupation = "web developer"
- < "web developer"
- > occupation
- < "web developer"

The last three lines of code (the assignment and the two retrievals) are highlighted with a red rectangular box.

We can add some value with occupation and save it inside of the same variable like-

```
▶ dge
◀ 24
> occupation
< "freelancer"
> occupation = "web developer"
< "web developer"
> occupation
< "web developer"
> name = name + "tutorials"
< "nerdJfpbtutorials"
> name
< "nerdJfpbtutorials"
>
```

Also we can do the old things we've done with values. example -



```
Elements Console Sources Network
top
✓ name = name + TUTORIALS
< "nerdJfpbtutorials"
> name
< "nerdJfpbtutorials"
> name === "nerdJfpb"
< false
> age > 25
< false
> age < 20
< false
> age > 18
< true
>
```



nerdjfpb



nerdjfpb

**Keep practicing the stuff
to learn more!**



nerd_jfpb



nerdJfpb



nerdJfpb

More About Variables - JavaScript Series -

Part 7

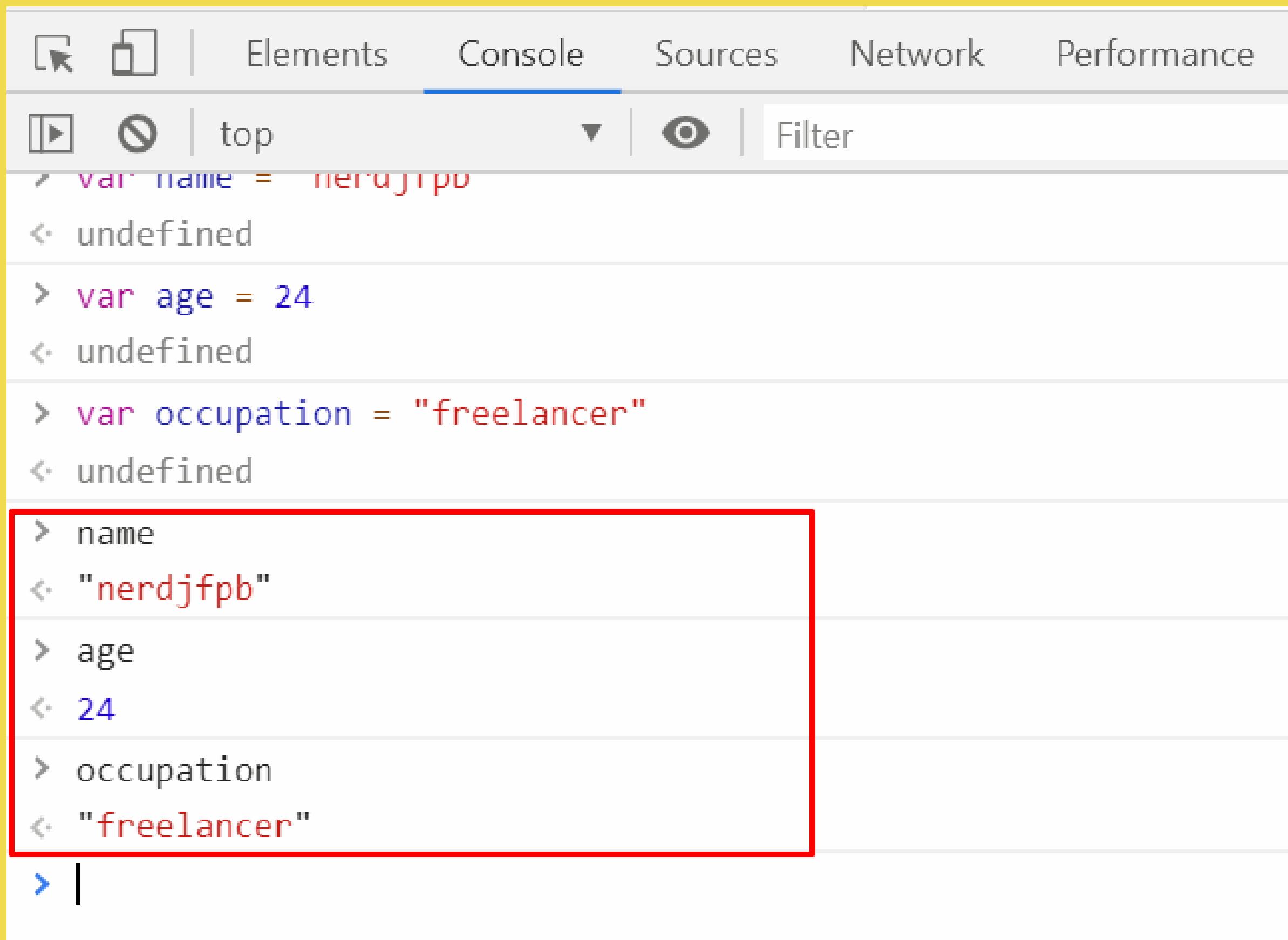
JS



nerd_jfpb



In last part we learned about variables right ? Now today we are going to learn a bit more about variables.

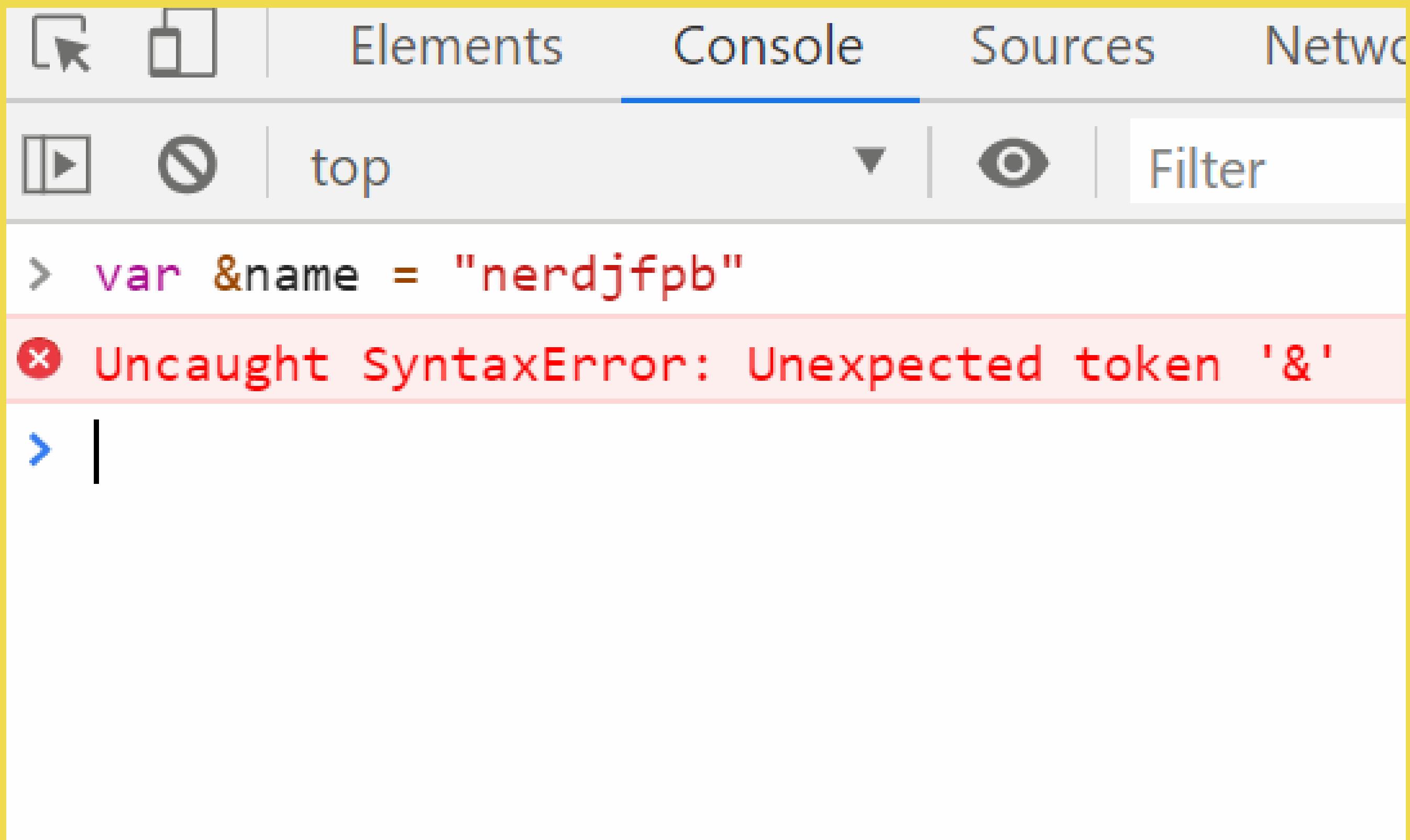


The screenshot shows the browser's developer tools open to the 'Console' tab. The console interface includes tabs for Elements, Console, Sources, Network, and Performance. Below the tabs, there are icons for play, stop, and filter, followed by the word 'top'. The main area displays a series of JavaScript variable assignments:

```
> var name = nerdJfpb
< undefined
> var age = 24
< undefined
> var occupation = "freelancer"
< undefined
> name
< "nerdJfpb"
> age
< 24
> occupation
< "freelancer"
>
```

A red rectangular box highlights the last four lines of code, specifically the variable assignments for 'name', 'age', and 'occupation'.

We can't start variable name with symbol. Like -



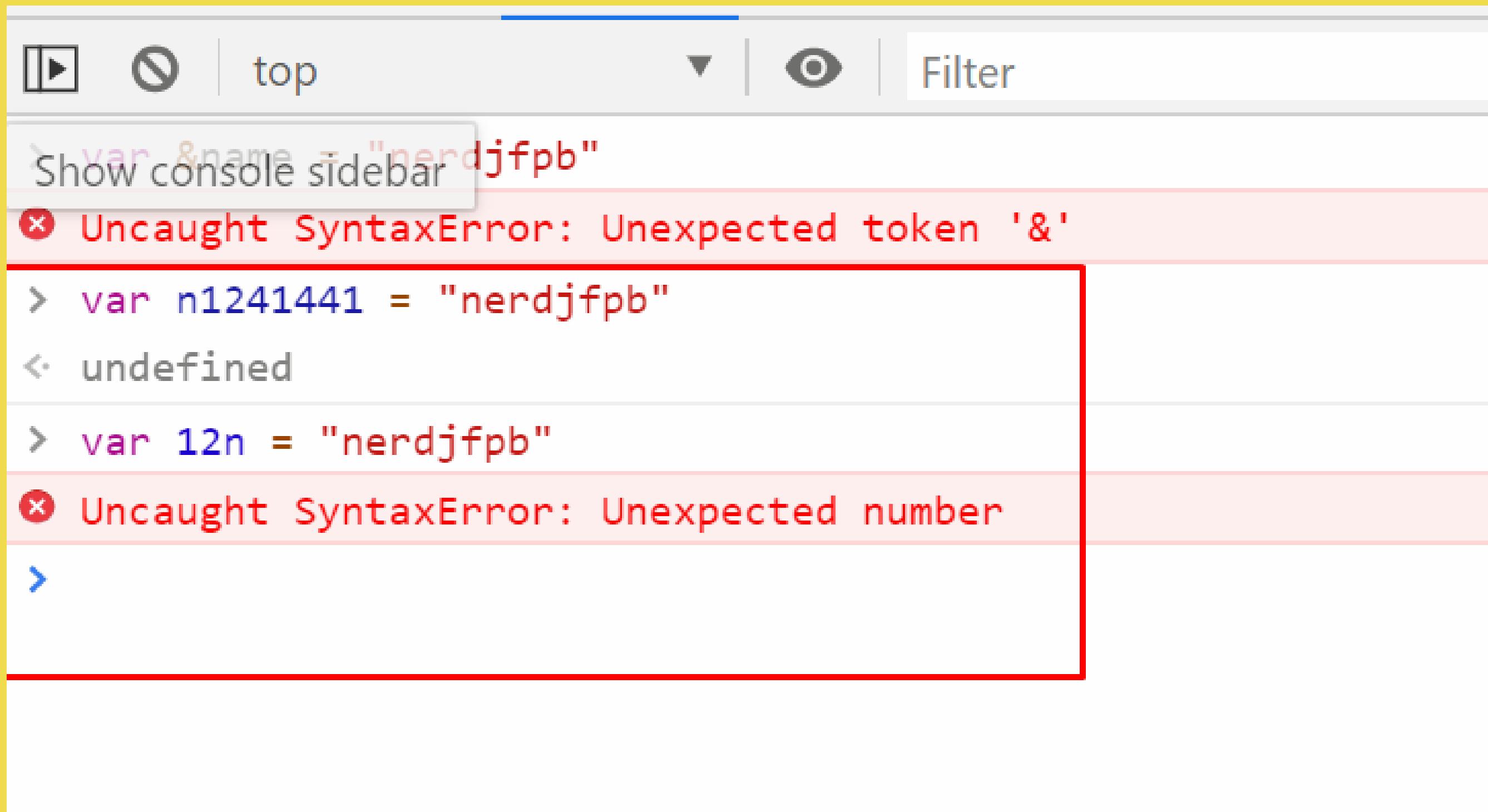
The screenshot shows the browser's developer tools with the 'Console' tab selected. The console output is as follows:

```
> var &name = "nerdjfpb"
✖ Uncaught SyntaxError: Unexpected token '&'
```

The first line shows a syntax error where the variable name starts with the ampersand symbol ('&'). The second line is a blank command prompt.

Variable can start with a letter than numbers, it's fine.

But no way to start numbers.



The screenshot shows a browser developer tools console with the following interactions:

- Copy icon, Paste icon, Stop icon | top ▾ | Eye icon | Filter
- Show console sidebar
- Uncaught SyntaxError: Unexpected token '&'**
- > var &name = "nerdjqfpb"
- < undefined
- > var n1241441 = "nerdjqfpb"
- Uncaught SyntaxError: Unexpected number**
- >

A red box highlights the error messages and the problematic code lines.



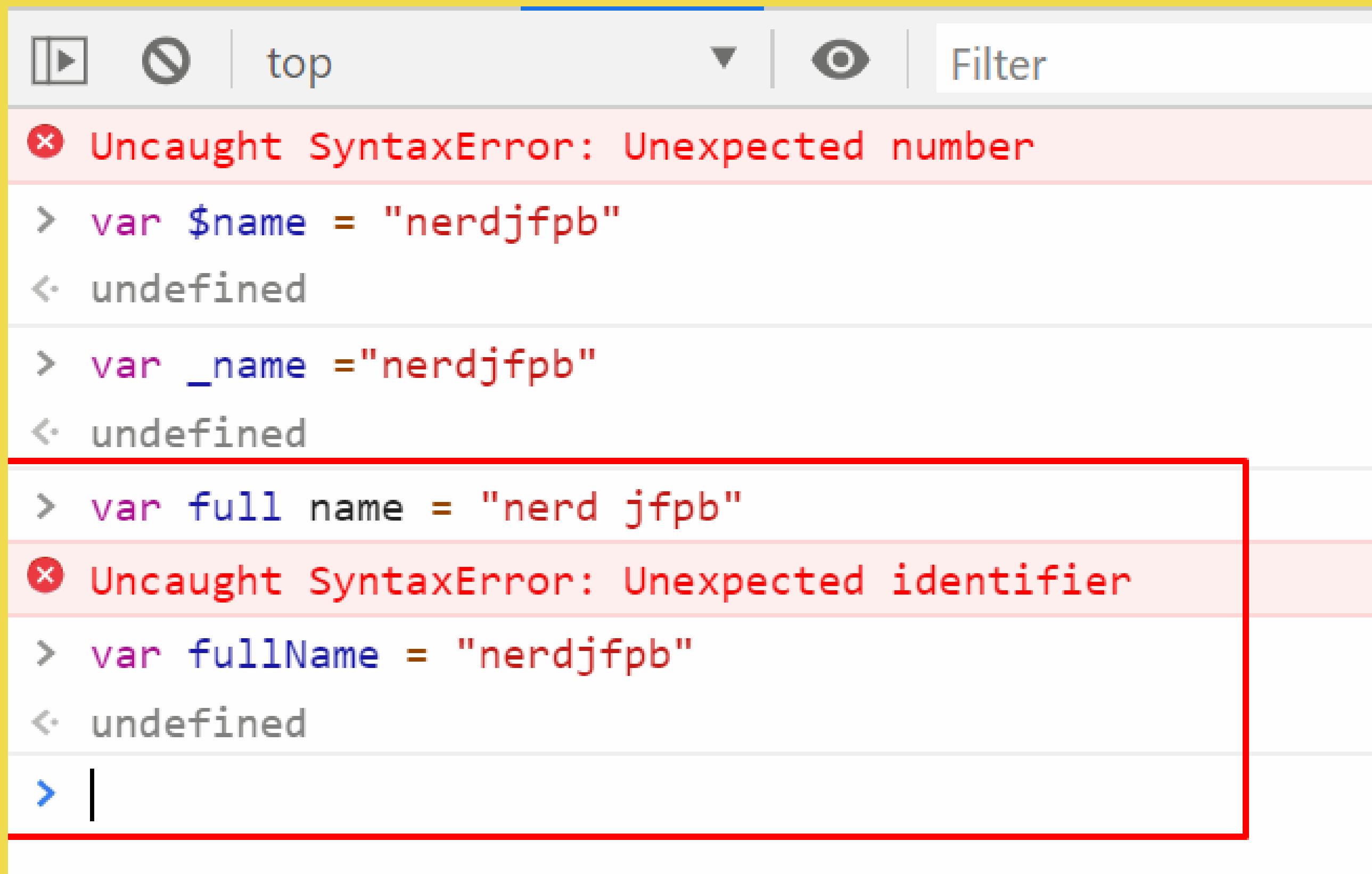
But you can start a variable with _ & \$

The screenshot shows the Chrome DevTools Console tab. The tabs at the top are Elements, Console (which is selected), Sources, Network, and Perf. Below the tabs are icons for Play, Stop, and a dropdown menu set to 'top'. To the right is a 'Filter' input field. The console output is as follows:

- Uncaught SyntaxError: Unexpected token '&'
▶ var n1241441 = "nerdjfpb"
◀ undefined
- Uncaught SyntaxError: Unexpected number
▶ var 12n = "nerdjfpb"
◀ undefined
- Uncaught SyntaxError: Unexpected token '\$'
▶ var \$name = "nerdjfpb"
◀ undefined
- Uncaught SyntaxError: Unexpected identifier
▶ var _name ="nerdjfpb"
◀ undefined
- ▶



We can't have space in variable name and if we need a long variable name. Then we use camelCase. Which means you'll write in big letter the second word. example -

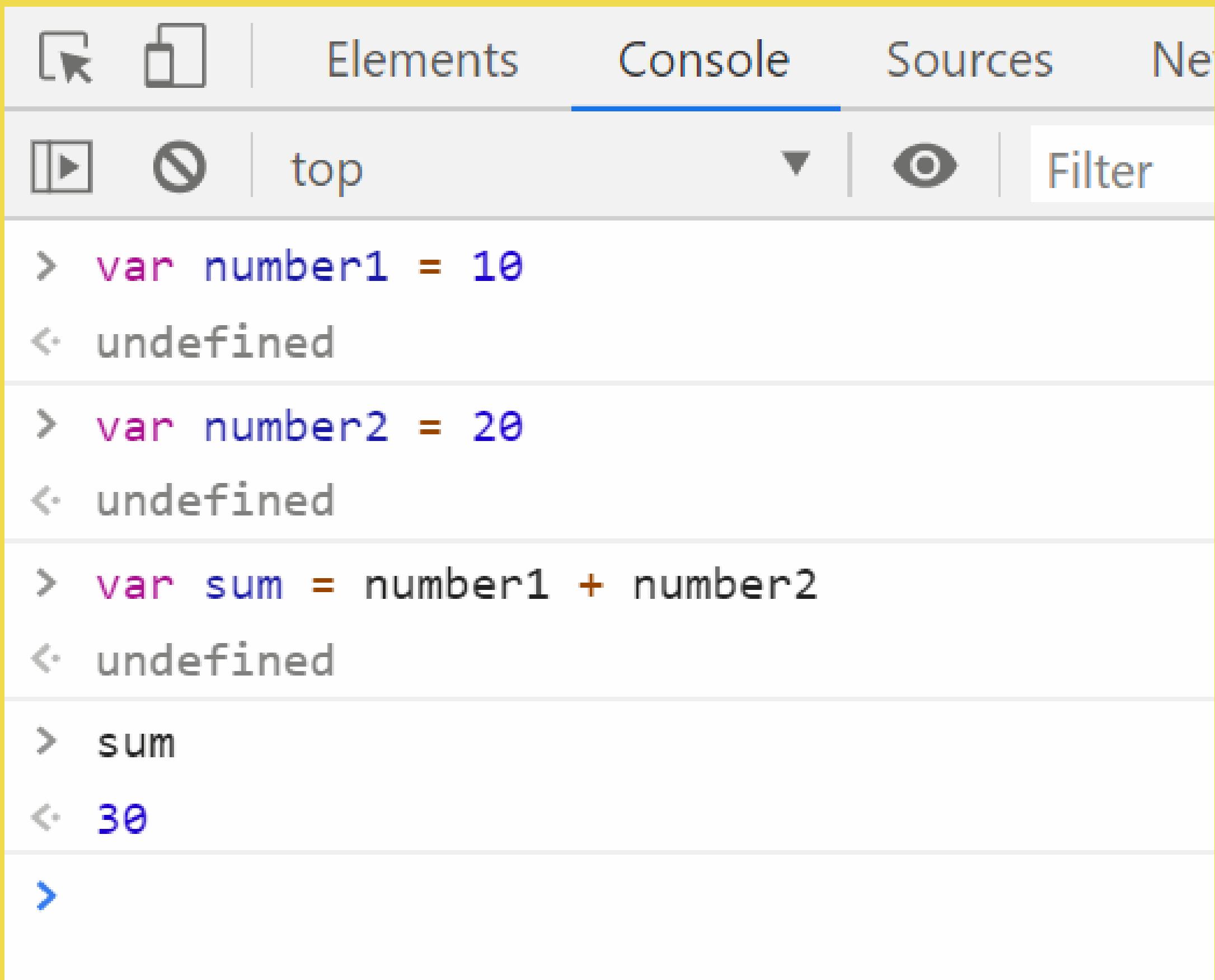


The screenshot shows a browser's developer tools console with the following content:

```
<-- top
x Uncaught SyntaxError: Unexpected number
> var $name = "nerdJfpb"
<- undefined
> var _name ="nerdJfpb"
<- undefined
> var full name = "nerd jfpb"
x Uncaught SyntaxError: Unexpected identifier
> var fullName = "nerdJfpb"
<- undefined
> |
```

A red rectangle highlights the last two lines of code and their resulting error message.

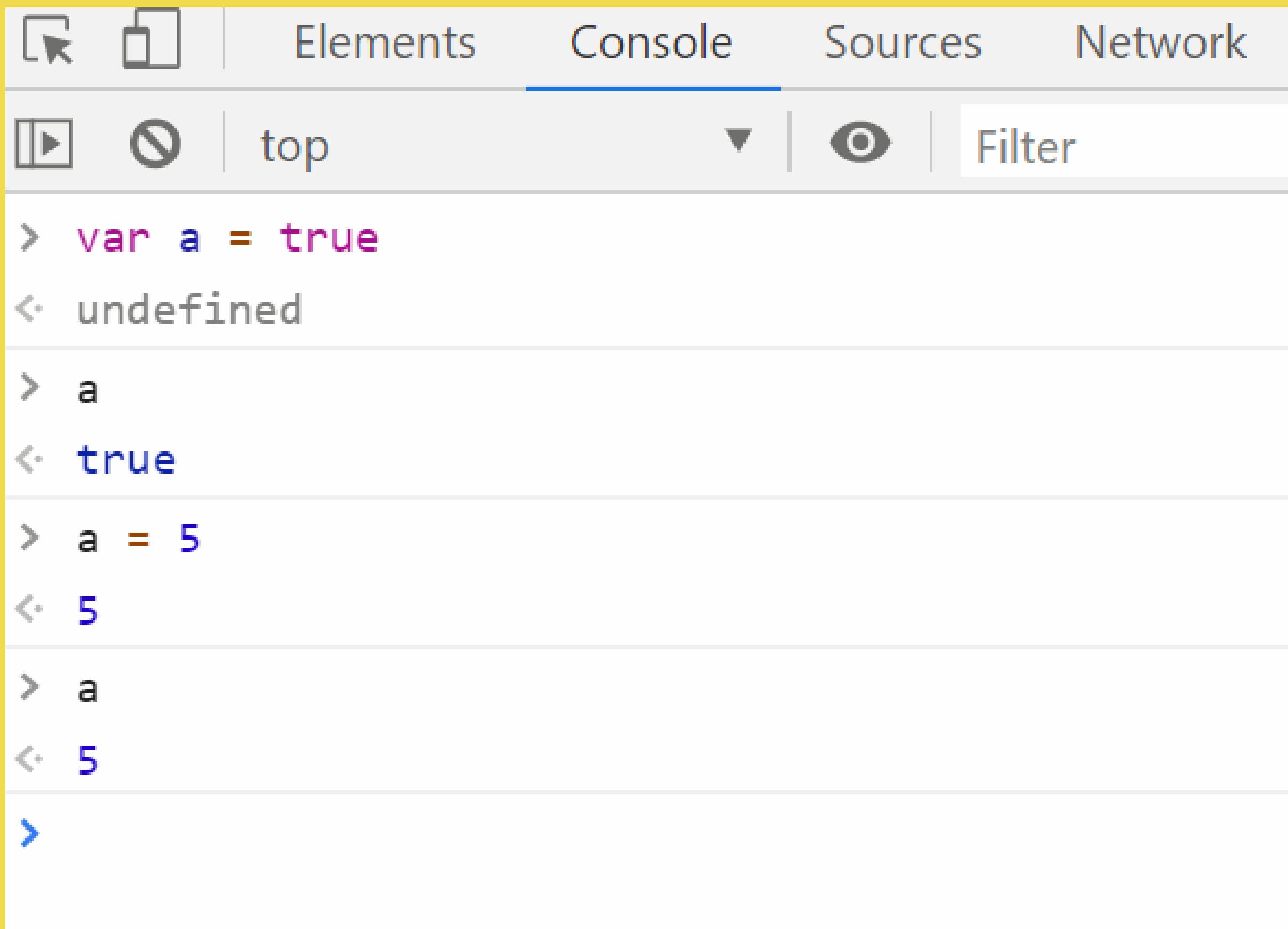
Let do a little easy trick with variable. First we'll get two variable and add them and store it on a new variable, then we'll see if we got the exact value.



The screenshot shows the browser's developer tools with the 'Console' tab selected. The console output is as follows:

```
> var number1 = 10
< undefined
> var number2 = 20
< undefined
> var sum = number1 + number2
< undefined
> sum
< 30
>
```

Another thing, we can change the types of variable just by storing that value on there, like -



The screenshot shows the Chrome DevTools Console tab. The top navigation bar includes Elements, Console (which is selected), Sources, and Network. Below the tabs is a toolbar with icons for play, stop, and filter, followed by the word "top". The main area displays the following JavaScript interactions:

```
> var a = true
< undefined

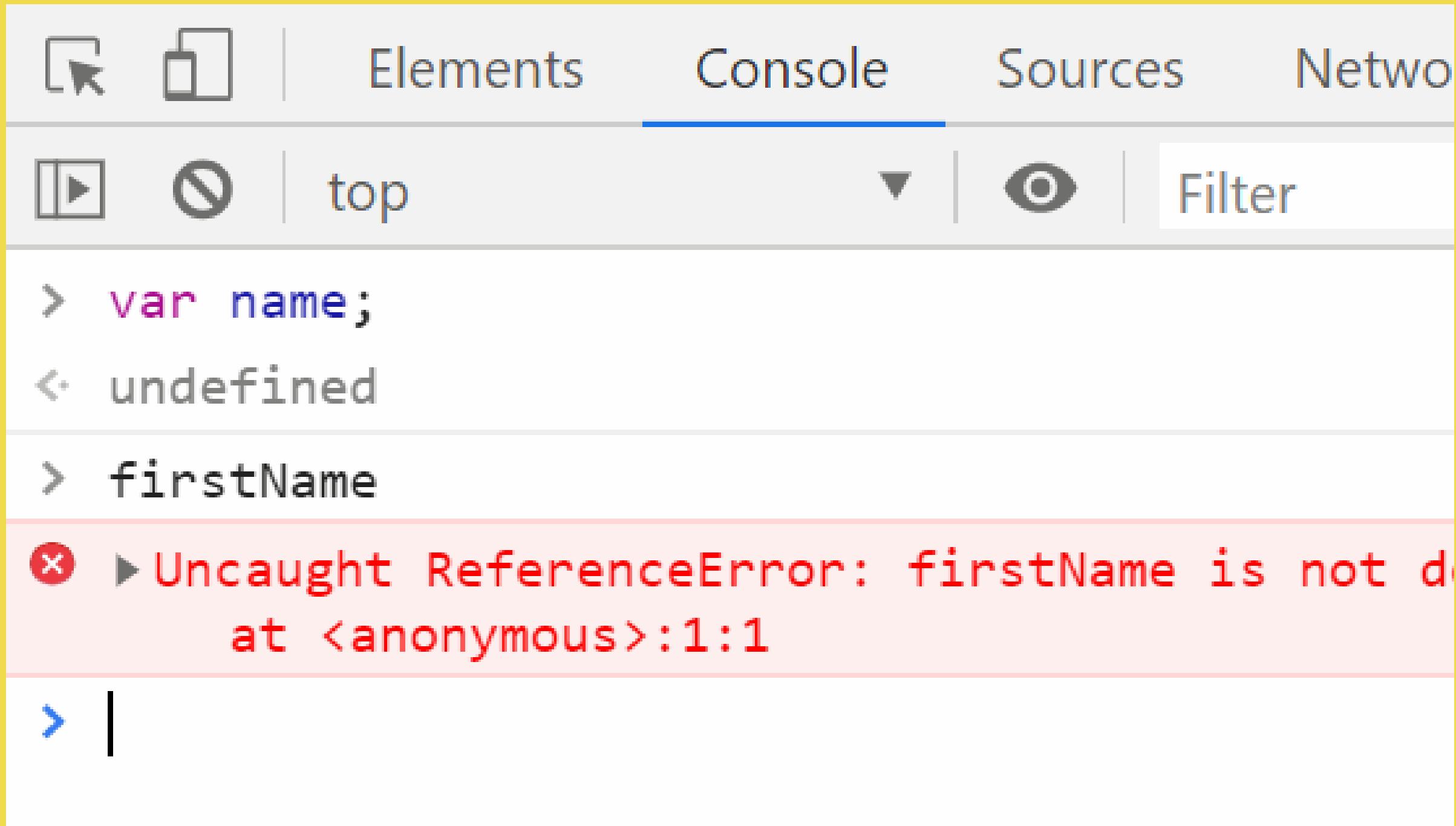
> a
< true

> a = 5
< 5

> a
< 5

>
```

What if we didn't mention any type while initializing a variable. It will be undefined. This is another type of JS, we didn't learn at first. If you never initialize a variable but trying to call it, then it will give you a reference error.



The screenshot shows the Chrome DevTools Console tab. The tabs at the top are Elements, Console (which is selected), Sources, and Network. Below the tabs are toolbar icons for play, stop, and filter. The main area displays the following JavaScript code and its execution results:

```
> var name;  
← undefined  
> firstName  
  
✖ > Uncaught ReferenceError: firstName is not defined  
    at <anonymous>:1:1  
> |
```

The last line, which contains a reference error, is highlighted with a red background.



nerdjfpb



nerdjfpb

Do you learn some new stuff today ?



nerd_jfpb



nerdJfpb



nerdJfpb

Conditional Statement - JavaScript Series -

Part 8

JS



nerd_jfpb





Up until now we never wrote any condition in our code, we just wrote one line code and executed those.

But now we are going to learn about conditional statements in Javascript.



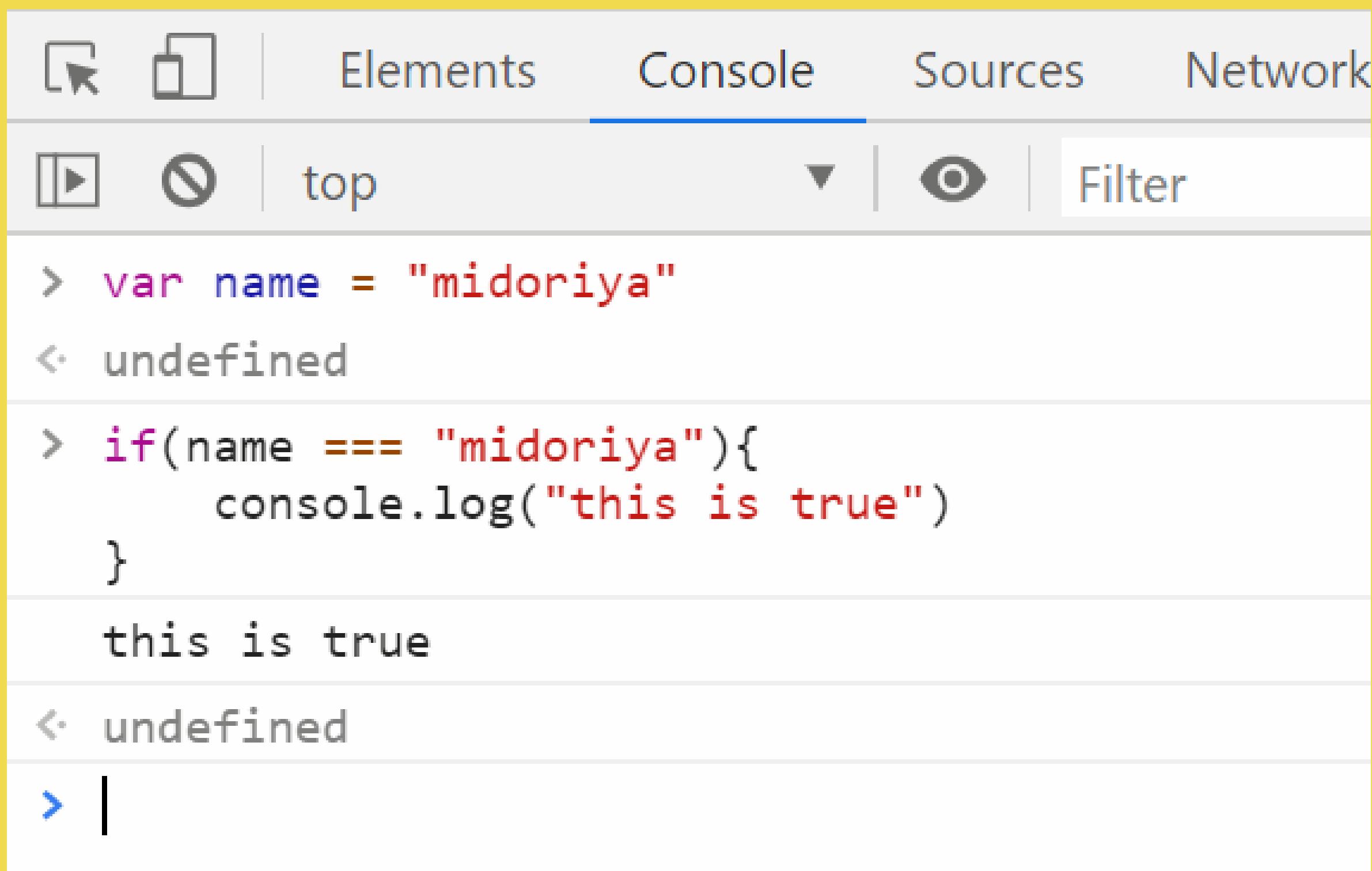


**There are
if
else
else if &
switch
these are available in javascript to
write conditions.**

**You can write ternary operators too.
But for now let's just focus on
if statement.**



This is like english. We'll write if first then first brackets and in the brackets we'll write the condition and finally there will be second brackets where we'll write what we want to do. Look at this code -



The screenshot shows the 'Console' tab of a browser's developer tools. The console output is as follows:

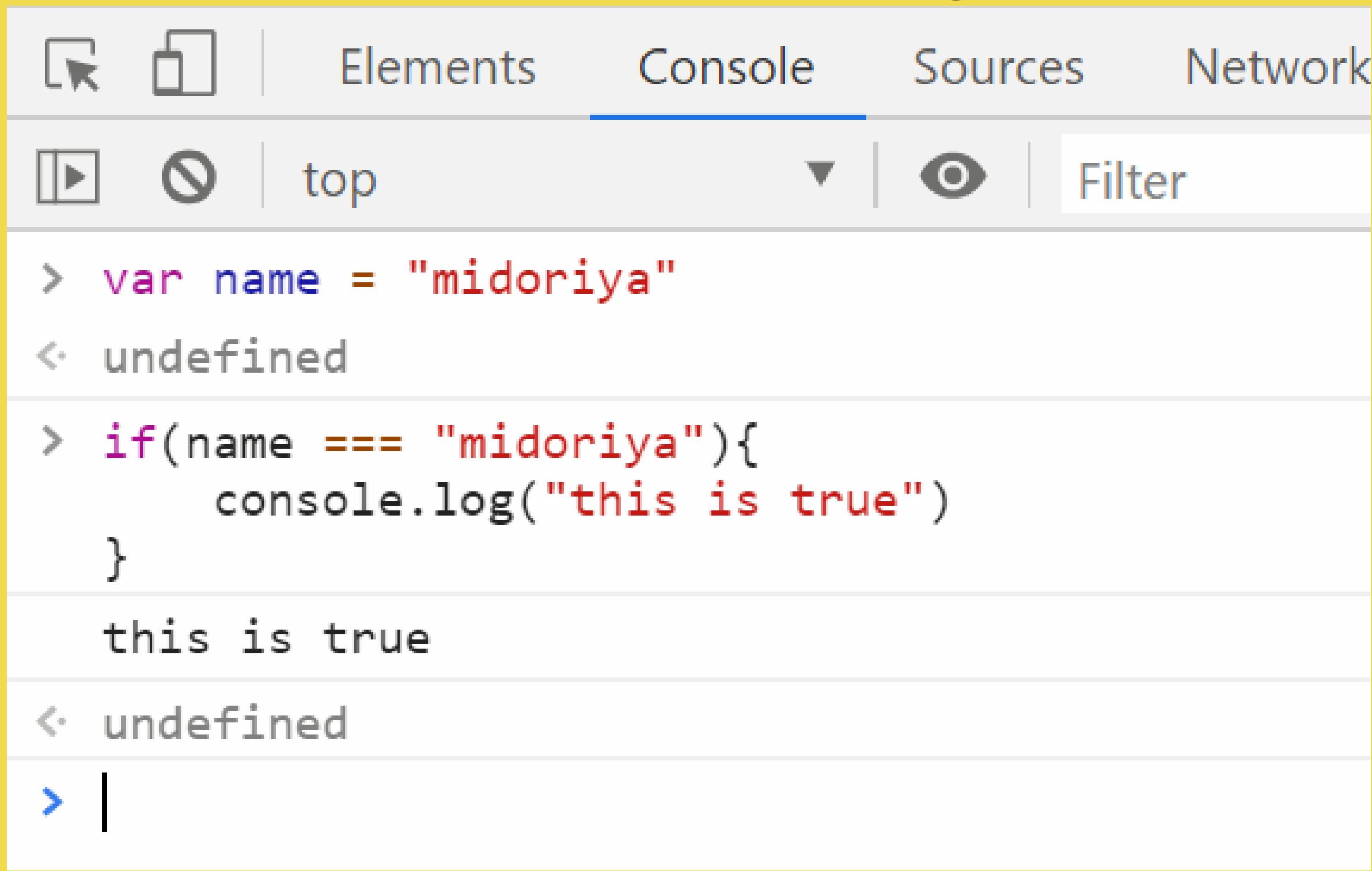
```
> var name = "midoriya"
< undefined

> if(name === "midoriya"){
    console.log("this is true")
}
this is true

< undefined

> |
```

Now let's break it down from the way. First we store midoriya string into the name, then we checked if the name is equal to midoriya then we just print in console a new string. This is easy right ? Just it's almost like plain english.



The screenshot shows the browser's developer tools with the 'Console' tab selected. The console output is as follows:

```
> var name = "midoriya"
< undefined

> if(name === "midoriya"){
    console.log("this is true")
}

this is true
< undefined

> |
```



Else part is easier also. If the condition doesn't match then else part will work.

Look here -

The screenshot shows a browser's developer tools with the 'Console' tab selected. The console output is as follows:

```
Elements Console Sources Network Performance
top
}
this is true
< undefined
> if(name === "todoroki") {
    console.log("string is midoriya")
} else {
    console.log("string is not midoriya")
}
string is not midoriya
< undefined
> |
```

The code demonstrates an if-else conditional. It checks if the variable `name` is equal to "todoroki". Since it is not, the `else` block is executed, printing "string is not midoriya" to the console.

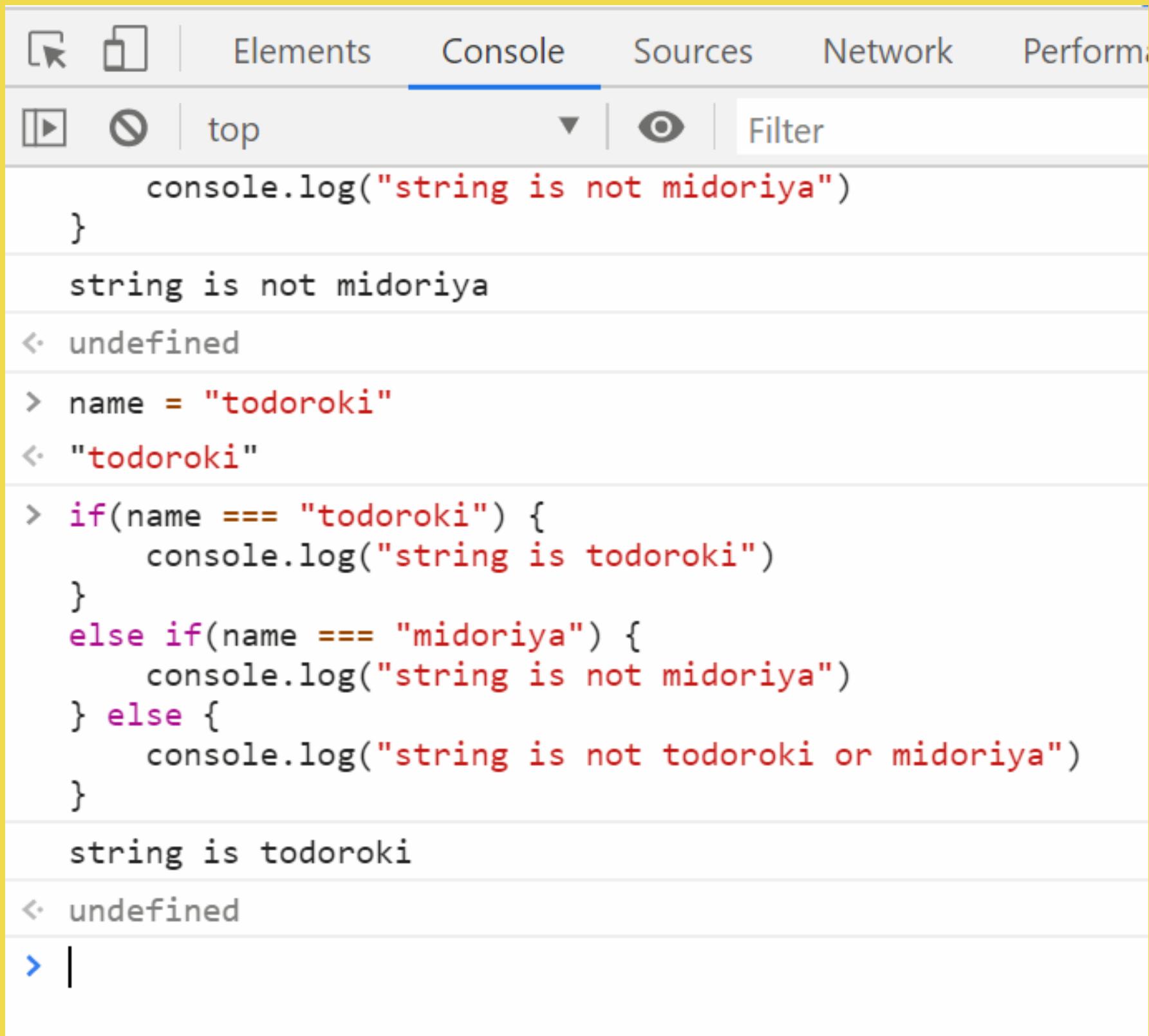




**Now we'll learn about the else if.
But let's change the name to
"todoroki" first.**



Like before if midoriya doesn't match then always the else part was printed, but now we want to check if the string is todoroki or midoriya or none of them. We can do it easily by just using a new else if(conditions) -



```
Elements    Console    Sources    Network    Performance  
top  
    console.log("string is not midoriya")  
}  
string is not midoriya  
< undefined  
> name = "todoroki"  
< "todoroki"  
> if(name === "todoroki") {  
    console.log("string is todoroki")  
}  
else if(name === "midoriya") {  
    console.log("string is not midoriya")  
} else {  
    console.log("string is not todoroki or midoriya")  
}  
string is todoroki  
< undefined  
> |
```



So finally what is our code ?

```
if(condition) {  
    block of code  
} else if (condition) {  
    block of code  
} else {  
    block of code  
}
```





nerdjfpb



nerdjfpb

**Let me know if you understand
this part clearly or not!**



nerd_jfpb



nerdJfpb



nerdJfpb

More Conditional Statement - JavaScript Series -

Part 9

JS



nerd_jfpb





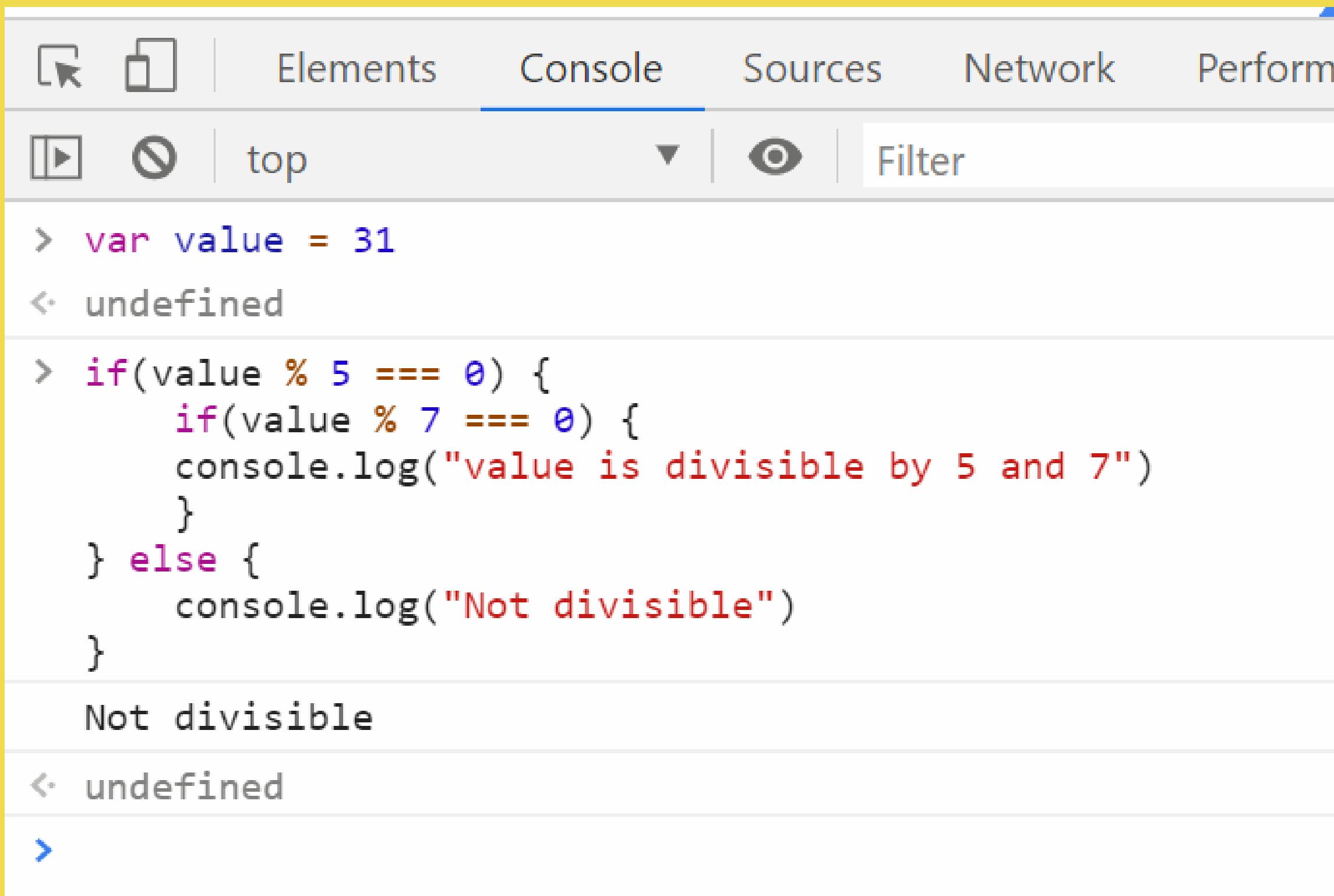
In last part we didn't talk much about conditions.

In this part we are going to talk about the conditions.



Suppose we need to found a value which is divisible by 5 and 7 both. How can we do this? The straight forward way will be just write one if inside of another one.

Example -



The screenshot shows a browser's developer tools with the "Console" tab selected. The console output is as follows:

```
> var value = 31
< undefined

> if(value % 5 === 0) {
    if(value % 7 === 0) {
        console.log("value is divisible by 5 and 7")
    }
} else {
    console.log("Not divisible")
}

Not divisible
< undefined
>
```



**Now we can do two conditions in one if.
By adding && we can do it.
Adding && means we need to satisfy
both conditions to go that block.
Example -**

```
> if(value % 5 === 0 && value % 7 === 0) {  
    console.log("value is divisible by 5 and 7")  
} else {  
    console.log("Not divisible")  
}
```

```
Not divisible
```

```
< undefined
```

```
>
```

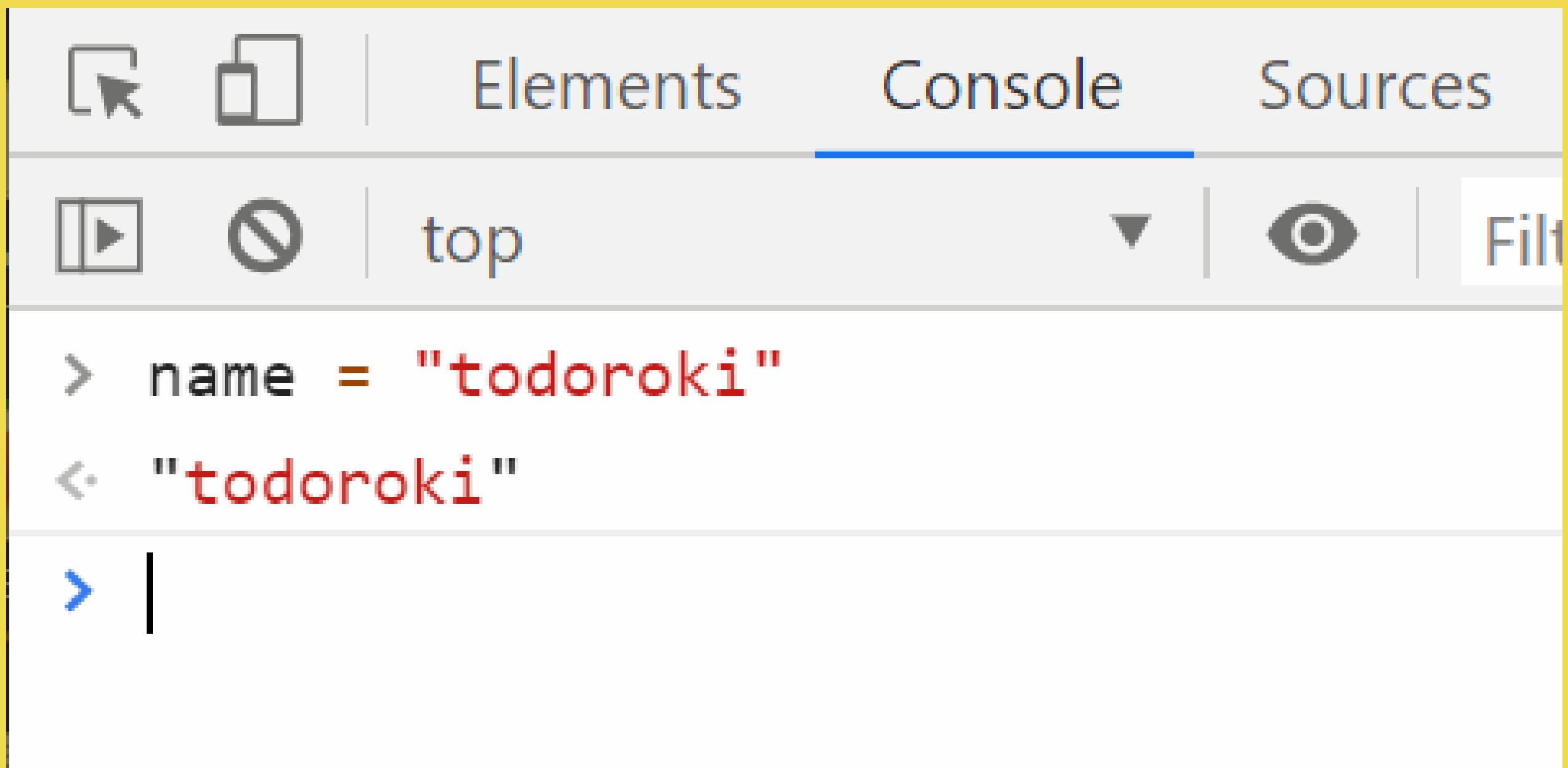




**Suppose we have string which can contain midoriya or todoroki.
If we get any of these, then we'll tell this is coming from BHNA anime or else we'll print - we don't know the source. Let's do it now!**



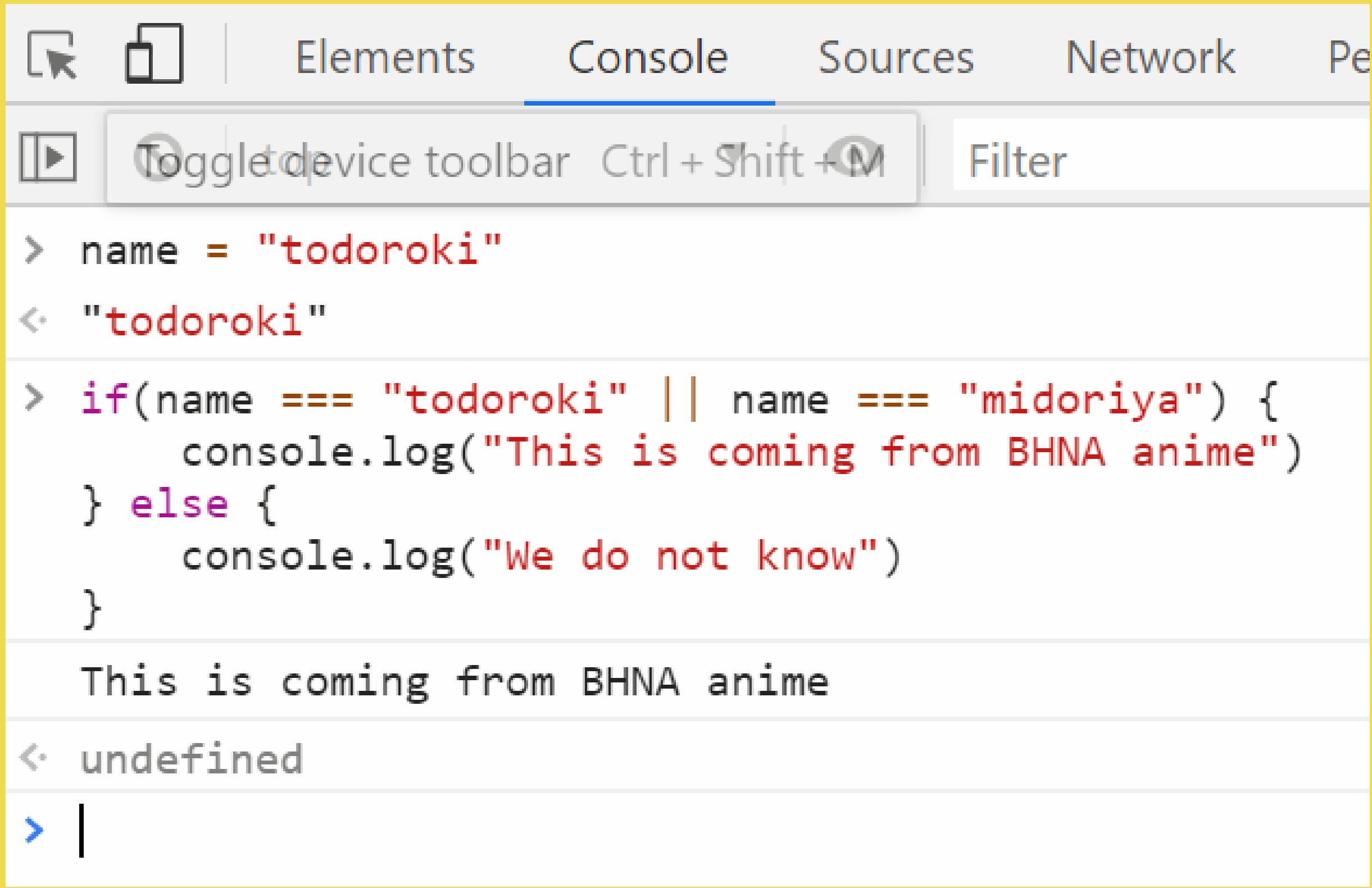
First we need a name variable and store the value in it -



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
> name = "todoroki"
< "todoroki"
> |
```

Now lets check the both condition and put `||` between them. This means OR. If any of these condition approved, then if block will run.



```
Elements      Console      Sources      Network      Pe
Toggle device toolbar  Ctrl + Shift + M  Filter

> name = "todoroki"
< "todoroki"

> if(name === "todoroki" || name === "midoriya") {
    console.log("This is coming from BHNA anime")
} else {
    console.log("We do not know")
}
This is coming from BHNA anime
< undefined

> |
```



nerdjfpb



nerdjfpb

So this was easy right ?



nerd_jfpb





nerdjfpb



nerdjfpb

We learned about && which means and.
Also we learn about || which means or.



nerd_jfpb





nerdjfpb



nerdjfpb

**Let me know if you have
any questions.**



nerd_jfpb



nerdJfpb



nerdJfpb

Switch - JavaScript Series -

Part 10

JS



nerd_jfpb





**Switch can be used instead of if else.
It's almost like if else too.**

```
switch(expression) {  
    case x:  
        code block  
        break;  
    case y:  
        code block  
        break;  
    default:  
        code block  
}
```





nerdjfpb



nerdjfpb

First we need to write the switch then expression, this is mean on which variable you are trying to apply.



nerd_jfpb





**Cases are the specific if.
Like case "todoroki" will be same
of variable==="todoroki"**





nerdjfpb



nerdjfpb

Let's try the last code in switch.



nerd_jfpb

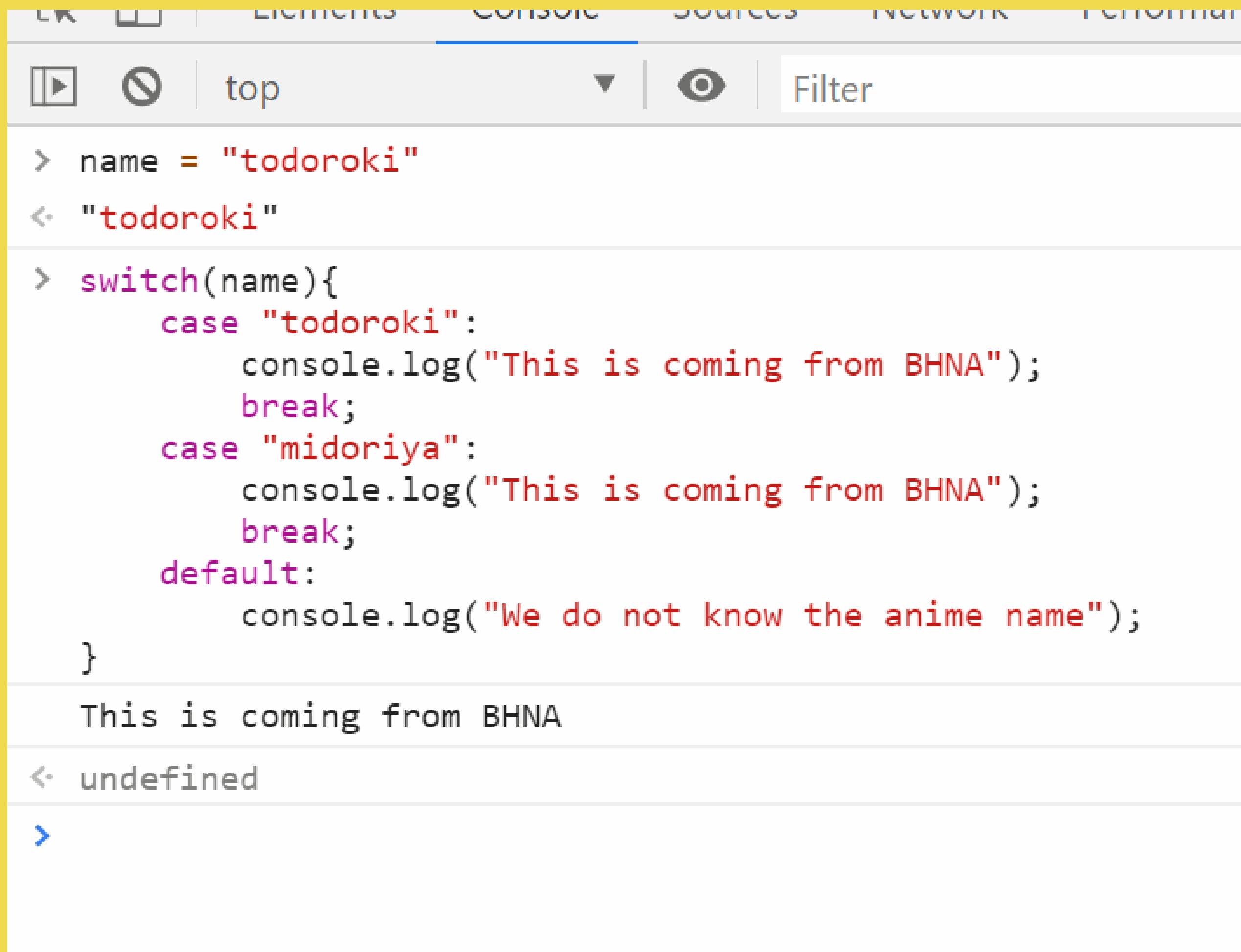




**First we need the variable name.
Then we'll write the switch for the
name - `switch(name) {}`
everything will go inside the second
brackets.**



Finally -



The screenshot shows a browser's developer tools with the 'Console' tab selected. The code being run is a switch statement:

```
> name = "todoroki"
< "todoroki"
> switch(name){
    case "todoroki":
        console.log("This is coming from BHNA");
        break;
    case "midoriya":
        console.log("This is coming from BHNA");
        break;
    default:
        console.log("We do not know the anime name");
}
This is coming from BHNA
< undefined
>
```

The output of the code is visible below the console input, showing the message "This is coming from BHNA".



nerdjfpb



nerdjfpb

**This is almost same right ?
So which one we should use ?**



nerd_jfpb





"As it turns out, the switch statement is faster in most cases when compared to if-else, but significantly faster only when the number of conditions is large. The primary difference in performance between the two is that the incremental cost of an additional condition is larger for if-else than it is for switch. Therefore, our natural inclination to use if-else for a small number of conditions and a switch statement for a larger number of conditions is exactly the right advice when considering performance. Generally speaking, if-else is best used when there are two discrete values or a few different ranges of values for which to test. When there are more than two discrete values for which to test, the switch statement is the most optimal choice."

[oreilly.com]





nerdjfpb



nerdjfpb

So which one you are goint to use ?



nerd_jfpb

JavaScript Series

Setup Code Editor





nerdjfpb



nerdjfpb

Up until now we just worked with
the console.
But now we'll work on a code editor.



nerd_jfpb





nerdjfpb



nerdjfpb

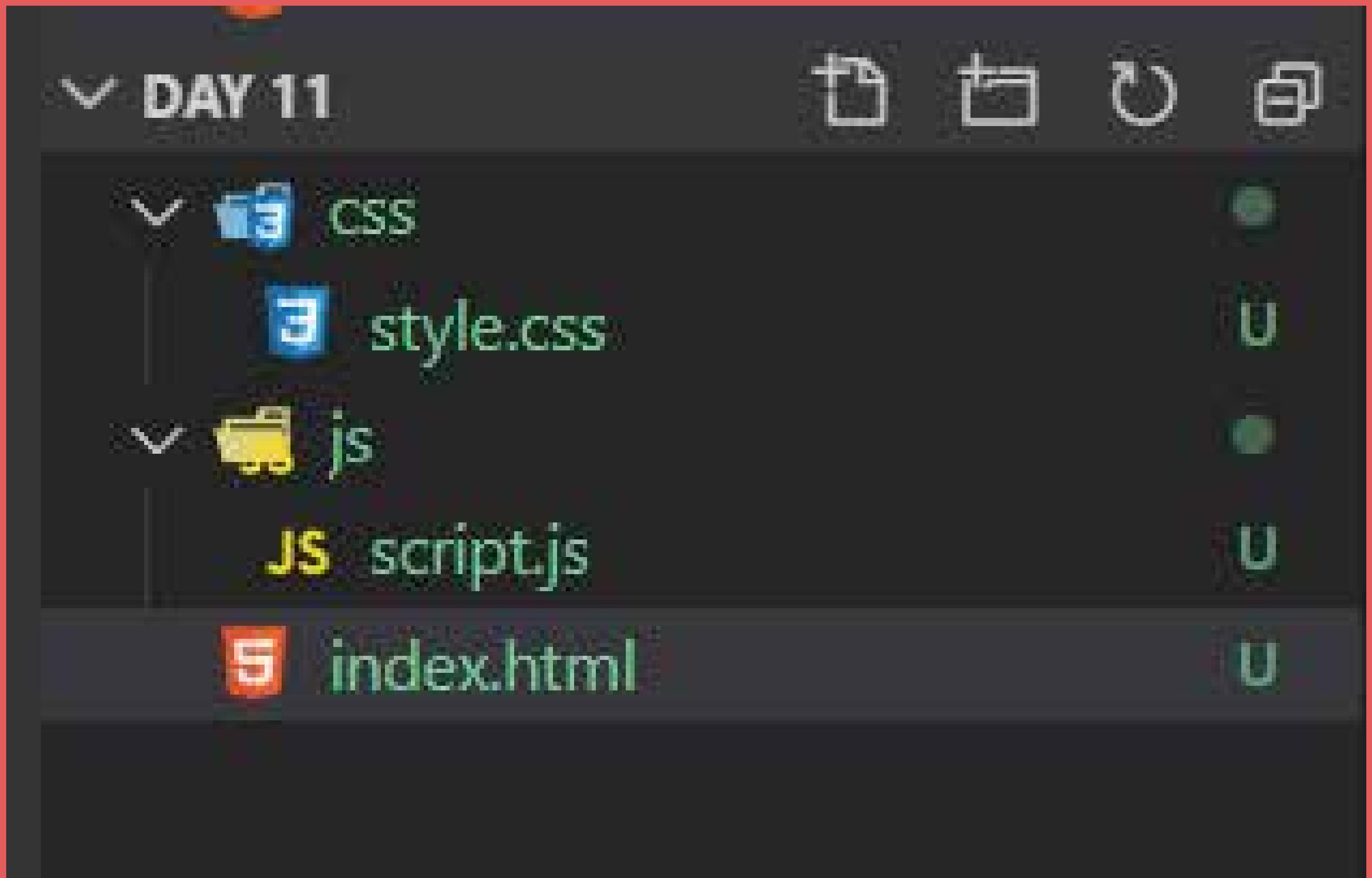
I'm mostly a vscode fan so I'm
going to use vscode.
You can use whatever you want.



nerd_jfpb



For folder I'll create like -





There is nothing in css or js file. Now we'll include this with the index.html

```
EXPLORER                                index.html X
OPEN EDITORS                               index.html ...
DAY 11                                     index.html > ...
index.html                                     u
css                                         u
style.css                                     u
js                                           u
script.js                                     u
index.html                                     u

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8" />
5  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6  <title>JavaScript Tutorial</title>
7  <link rel="stylesheet" href=".css/style.css" />
8  </head>
9  <body>
10 <h1>Welcome to JavaScript Tutorials!
11 <script src=".js/script.js"></script>
12 </body>
13 </html>
14 |
```



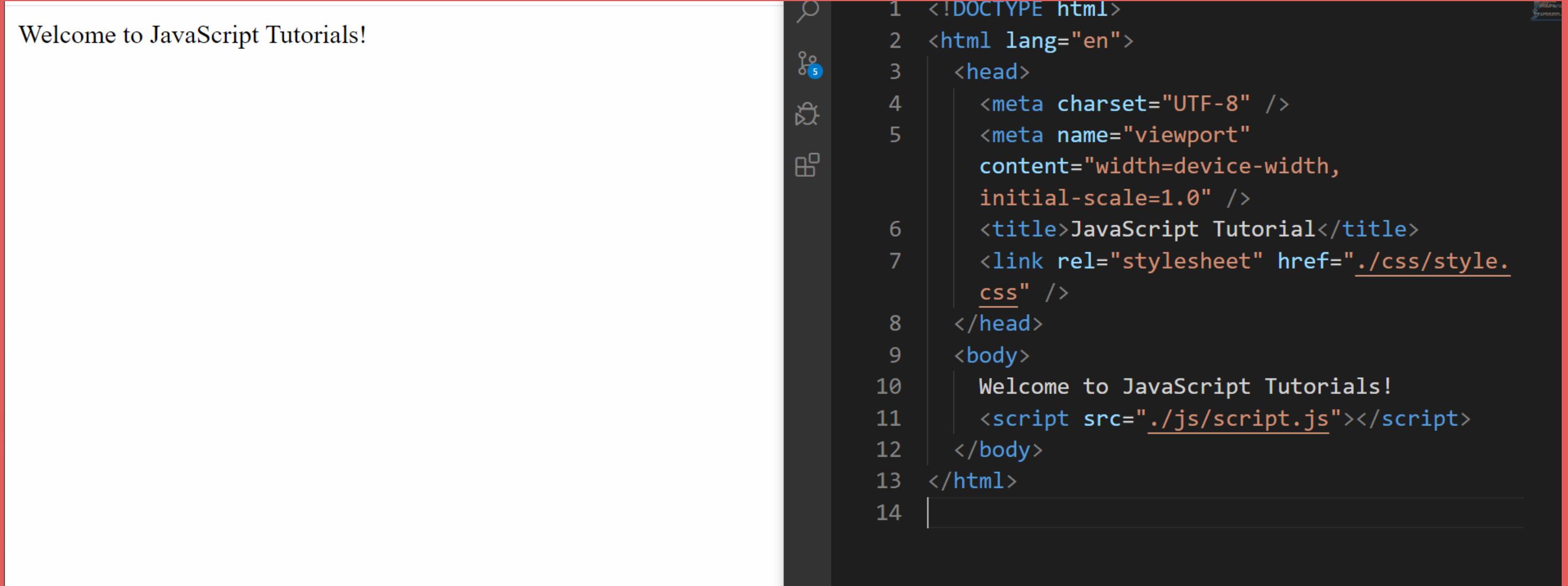


nerdjfpb



nerdjfpb

I'm using the live server extensions to run the files.



Welcome to JavaScript Tutorials!

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport"
6       content="width=device-width,
7         initial-scale=1.0" />
8     <title>JavaScript Tutorial</title>
9     <link rel="stylesheet" href=".css/style.
10       css" />
11   </head>
12   <body>
13     Welcome to JavaScript Tutorials!
14     <script src=".js/script.js"></script>
15   </body>
16 </html>
```



nerd_jfpb





nerdjfpb



nerdjfpb

Now we'll just console in our script.js file and we can see the result on our console of the website.

The screenshot shows a browser's developer tools open to the 'Console' tab. The main pane displays the message 'Welcome to new tutorial'. To the right, the code editor shows a single line of JavaScript: 'console.log('Welcome to new tutorial')'. The browser window above the tools shows a simple page with the heading 'Welcome to JavaScript Tutorials!'. The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, and Memory, along with various filter and settings options.

```
1
2 console.log('Welcome to new tutorial')
3
```



nerd_jfpb





nerdjqpb



nerdjqpb

I am using the live server extension which give me the way to run live the code

The screenshot shows the 'Live Server' extension page on the VS Code Marketplace. The extension is created by Ritwick Dey, has 3,851,221 downloads, and a 5-star rating. It is described as launching a development local Server with live reload feature for static & dynamic pages. A note indicates it is enabled globally. Below the main description, there is a message from the maintainer: '[I'm sorry but I'm super busy now. If you want to be a maintainer of the project, please feel free to contact me! You've to be passionate about programming]'. The 'Live Server' section highlights its support for multi-root workspaces and server-side PHP. It also mentions a fix for 'command not found error' #78. At the bottom, it reiterates the functionality: 'Launch a local development server with live reload feature for static & dynamic pages.'

Live Server ritwickdey.liveserver

Ritwick Dey | 3,851,221 | ★★★★★ | Repository | License

Launch a development local Server with live reload feature for static & dynamic pages

[Disable ▾](#) [Uninstall](#) This extension is enabled globally.

[Details](#) [Contributions](#) [Changelog](#)

[I'm sorry but I'm super busy now. If you want to be a maintainer of the project, please feel free to contact me! You've to be passionate about programming]

Live Server

Live Server loves ❤️ your multi-root workspace

Live Server for server side pages like PHP. [Check Here](#)

[For 'command not found error' #78]

vscode marketplace v5.6.1 downloads 7M rating 4.5/5 (222)

travis branch passing appveyor branch failing license MIT

Launch a local development server with live reload feature for static & dynamic pages.



nerd_jfpb





nerdjfpb



nerdjfpb

I use the prettier which helps me to easily arrange the code, so I don't have any messy codes.

The screenshot shows the Visual Studio Code extension marketplace. The top tab bar has three items: 'index.html' (highlighted in blue), 'script.js', and 'Extension: Prettier - Code formatter'. The 'Prettier - Code formatter' tab is active, displaying its details. The extension is titled 'Prettier - Code formatter' by 'esbenp.prettier-vscode'. It has 4,894,653 installs and a 5-star rating. The description says it's a 'Code formatter using prettier'. There are 'Disable' and 'Uninstall' buttons, and a note that it's enabled globally. Below this, there are links for 'Details', 'Contributions', and 'Changelog'. The main content area is titled 'Prettier Formatter for Visual Studio Code'. It describes Prettier as an opinionated code formatter that enforces a consistent style by parsing your code and re-printing it with its own rules. It supports multiple languages: JavaScript, TypeScript, Flow, JSX, JSON, CSS, SCSS, Less, HTML, Vue, Angular, GraphQL, Markdown, YAML, and 'Your favorite language?'. Below this, there's a section for 'Installation' with instructions to search for 'Prettier - Code formatter' in the VS Code extension marketplace or to run the command 'ext install esbenp.prettier-vscode' in VS Code's Quick Open feature.

Prettier - Code formatter esbenp.prettier-vscode

Esben Petersen | 4,894,653 | ★★★★☆ | Repository | License

Code formatter using prettier

Disable **Uninstall** *This extension is enabled globally.*

[Details](#) [Contributions](#) [Changelog](#)

Prettier Formatter for Visual Studio Code

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular
GraphQL · Markdown · YAML
Your favorite language?

Installation

Install through VS Code extensions. Search for **Prettier - Code formatter**

[Visual Studio Code Market Place: Prettier - Code formatter](#)

Can also be installed in VS Code: Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install esbenp.prettier-vscode
```



nerd_jfpb





nerdjfpb



nerdjfpb

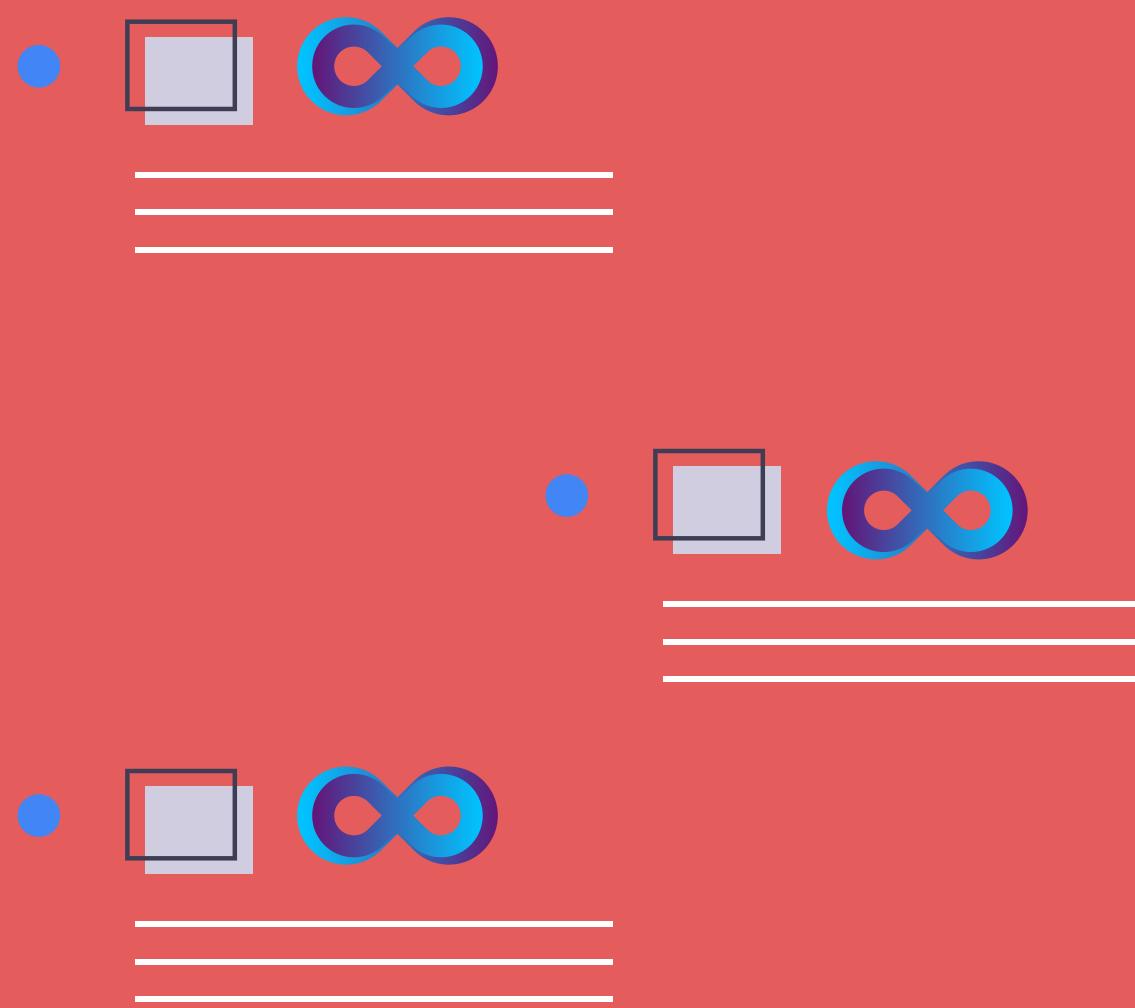
Is this lot to take or this is easy ?



nerd_jfpb

JavaScript Series

Loop





We are going to learn a really important topic today. I know this one is overwhelming at first, but if you practice then you'll get this properly!

Don't just see the post of today, work on it too!





There are some different kind of loop in js but all they do the same work.

for - loops through a block of code a number of times

for/in - loops through the properties of an object

for/of - loops through the values of an iterable object

while - loops through a block of code while a specified condition is true

do/while - also loops through a block of code while a specified condition is true (w3schools)





nerdJfpb



nerdJfpb

Most popular is for and while.
We'll learn both in your practice.



nerd_jfpb





Sometimes we need to do the same task again and again. Like for adding sum of 10, 12 and 30 we need to add all these. First we need to work $10 + 12$ then $22 + 30$ again right ? Or we can add all three at a time, but suppose there is huge numbers like 500 numbers.

What to do ?





We need to go through each and every number and add that to our total value.
Which is going to **start from zero** right?

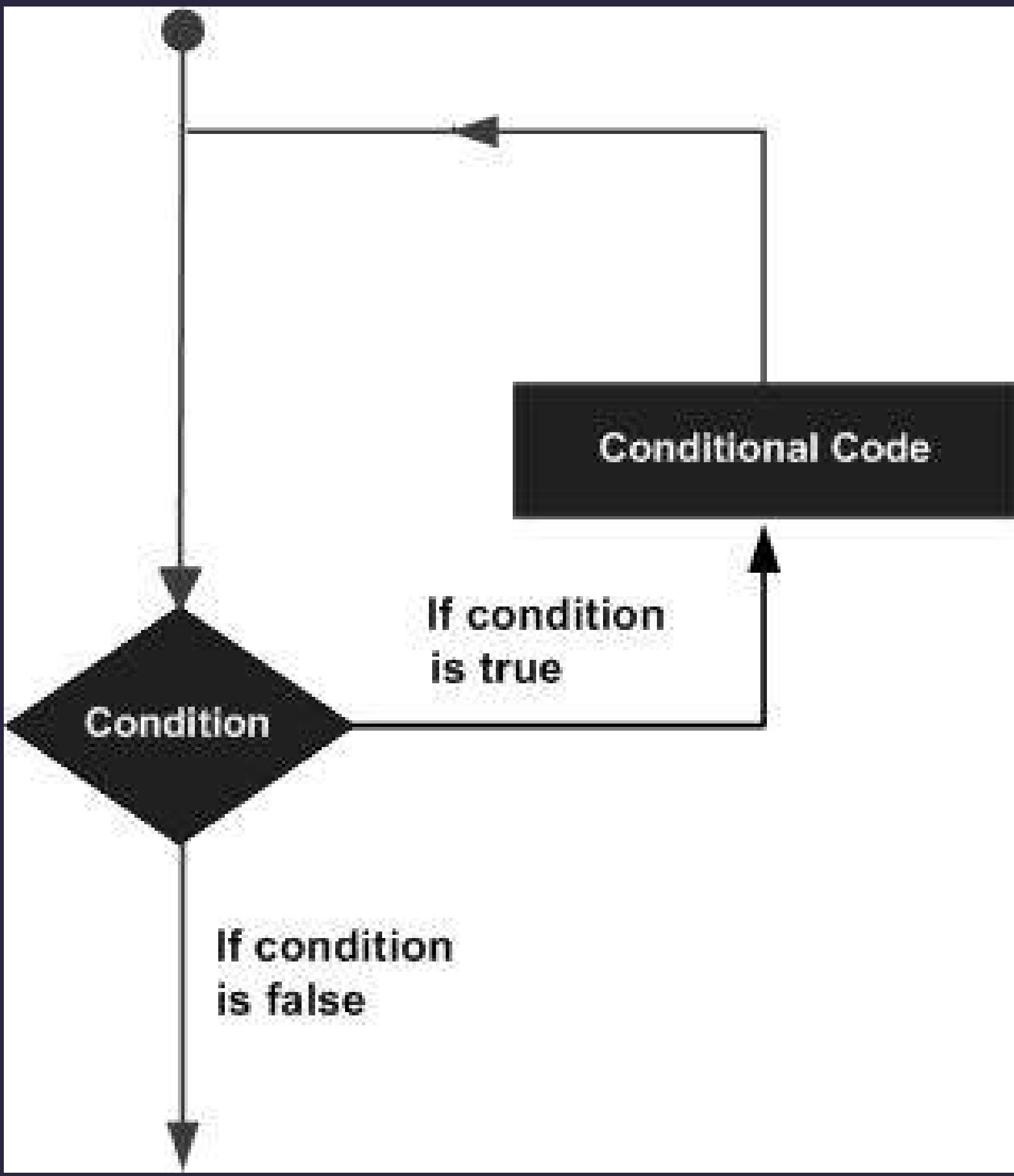




We can do this work by loop.
We can terminate doing the
same task again and again by
using loop.



Lets see this





nerdjfpb



nerdjfpb

Understand something ?



nerd_jfpb





nerdjfpb



nerdjfpb

We'll work on loop code
next tutorial.



nerd_jfpb

JavaScript Series

While Loop





Suppose we are going to print
1 to 10 on the console.

But how can we do it ?

We can do it easily

```
`console.log(1)  
console.log(2)
```

...

```
console.log(9)
```

```
console.log(10)
```

```
`
```





nerdjfpb



nerdjfpb

But this is not a good way to do it.
Currently we can do it because
it's only ten times.

But suppose we need to print
1 - 100.

How to do it ?



nerd_jfpb





nerdjfpb



nerdjfpb

This is where we use loops.
We're going to use
``while`` loop today!



nerd_jfpb





While is easy.

Just remember what we have learned in our last tutorial.

While loop syntax is -

```

```
while (condition) {
 // code block to be executed
}
```
```





Let's write some real codes now.
If we want to print 1 - 100 then
we are going to store the values
in variable and we are going to
start from 1.
So `var number = 1`

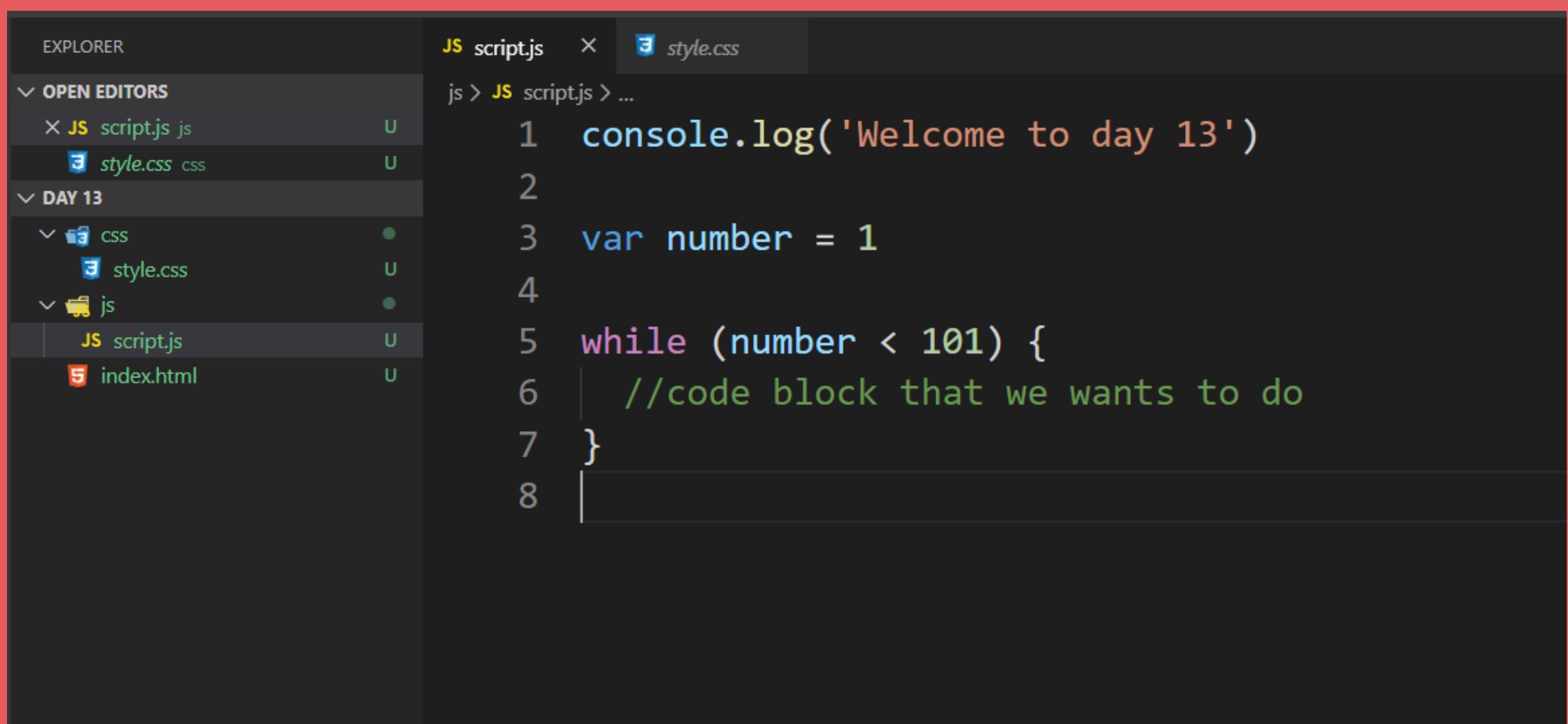


Now we are going to start our while.
While and the condition will be.

'''

```
while (number < 101) {  
    //code block that we wants to do  
}  
'''
```

'''

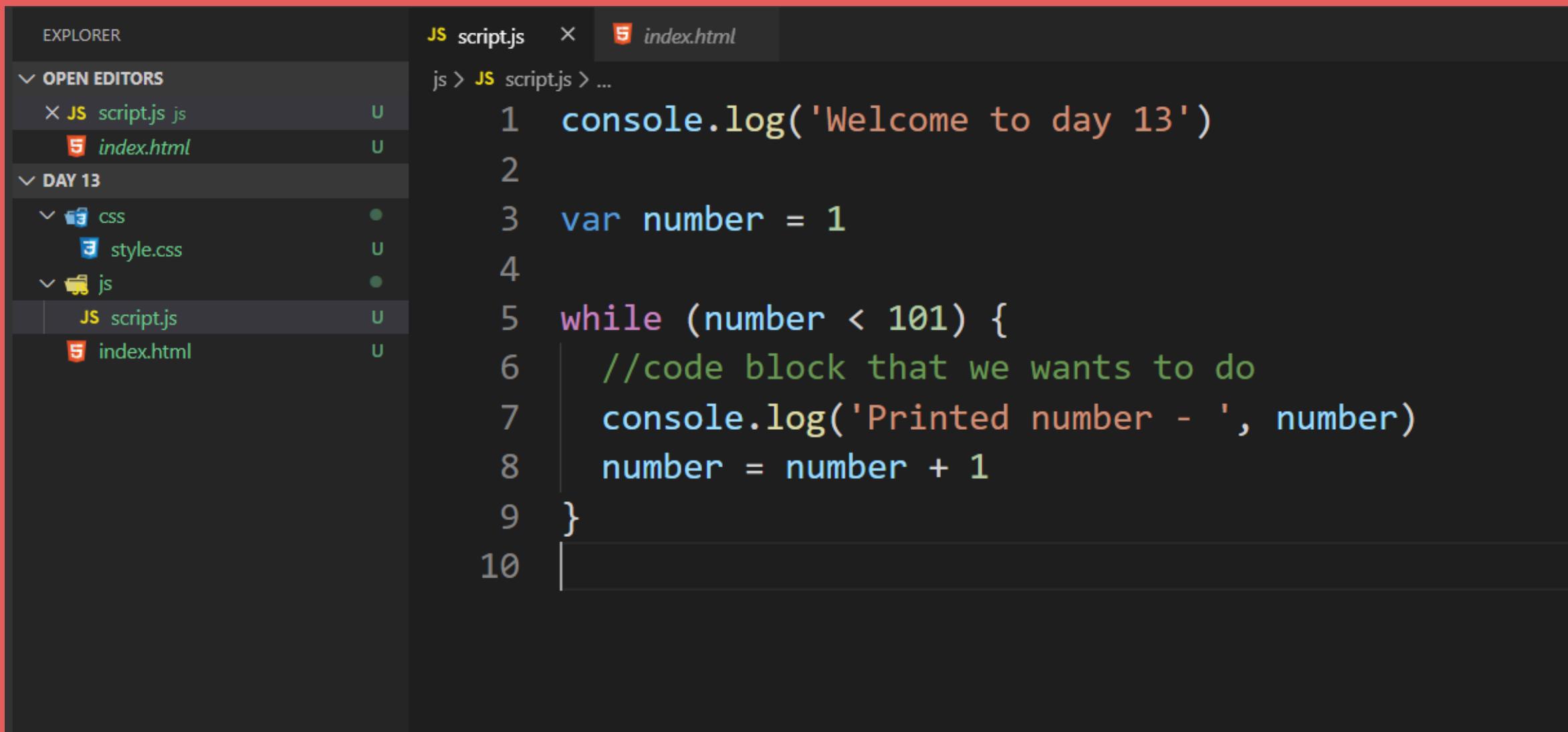


The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows a file tree with the following structure:
 - OPEN EDITORS: script.js (JS), style.css (CSS)
 - DAY 13:
 - css: style.css
 - js: script.js, index.html
- EDITOR:** The script.js file is open, showing the following code:

```
1 console.log('Welcome to day 13')  
2  
3 var number = 1  
4  
5 while (number < 101) {  
6     //code block that we wants to do  
7 }  
8
```

Now we are going to print the values from the first one. We need to increase the value of number to break the condition, if the condition doesn't break then it will stuck forever. So we always break the loop condition.



The screenshot shows a dark-themed instance of Visual Studio Code. On the left, the Explorer sidebar displays a file tree with 'OPEN EDITORS' containing 'script.js' and 'index.html', and a folder 'DAY 13' containing 'css/style.css' and 'js/script.js'. The main editor area shows the following code:

```
1 console.log('Welcome to day 13')
2
3 var number = 1
4
5 while (number < 101) {
6     //code block that we wants to do
7     console.log('Printed number - ', number)
8     number = number + 1
9 }
10
```



nerdjfpb



nerdjfpb

See the result in browser.

Welcome to JavaScript Tutorials!

The screenshot shows the browser's developer tools open to the 'Console' tab. The console output displays a series of printed numbers from 1 to 18, each followed by the file name 'script.js:7'. The console interface includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. A filter bar is visible at the top of the console area.

```
Welcome to day 13
Printed number - 1
Printed number - 2
Printed number - 3
Printed number - 4
Printed number - 5
Printed number - 6
Printed number - 7
Printed number - 8
Printed number - 9
Printed number - 10
Printed number - 11
Printed number - 12
Printed number - 13
Printed number - 14
Printed number - 15
Printed number - 16
Printed number - 17
Printed number - 18
```



nerd_jfpb





nerdjfpb



nerdjfpb

Do you understand the while loop ?



nerd_jfpb

JavaScript Series

For Loop





In last lesson we learn about the while loop,
but there is another loop which is really popular and people use this.
So we're going to learn it too.





Suppose we are going to print 1 to 10
on the console. But how can we do it ?
We can do it easily

```
`console.log(1)  
console.log(2)
```

...

```
console.log(9)  
console.log(10)
```

`

But this is not a good way to do it.
Currently we can do it because it's
only ten times.

But suppose we need to print 1 - 100.
How to do it ?





nerdJfpb



nerdJfpb

This is where we use loops.
We're going to use `For` loop today!



nerd_jfpb





For is easy.

Just remember what we learned
in our last tutorial.

For loop syntax is -

``

```
for (initial, condition, increment/  
decrement/rulesBreaking)
```

```
{
```

```
// code block to be executed
```

```
}
```

``





Let's write some real codes now.
If we want to print 1 - 10 then we
are going to **store the values** in
variable and we are going to
start from 1.

We can directly do this part in
`for` first part!





Now we are going to start our for. For then first part will be initial of the number, then condition for breaking the loop and final part will be how we can break the loop.

```
```for  
(var number = 1; number < 11; number++)
{
 // code block to be executed
 console.log('Printed number - ', number)
}
```
```

The screenshot shows a dark-themed interface of the Visual Studio Code code editor. On the left, the Explorer sidebar displays a file structure under 'OPEN EDITORS' and 'DAY 14'. The 'script.js' file is open in the main editor area. The code in 'script.js' is as follows:

```
1 console.log('Welcome to day 14')  
2 console.log('For Loop')  
3  
4 for (var number = 1; number < 11; number++) {  
5     // code block to be executed  
6     console.log('Printed number - ', number)  
7 }  
8
```





By the way did you notice that I used number++ ?

What that's mean ?

`number++ = (number = number + 1)`

same thing.

number++ is just a shortcut way to write it.





nerdjfpb



nerdjfpb

See the result in browser.

Welcome to JavaScript Tutorials!

The screenshot shows the browser's developer tools open to the 'Console' tab. The output window displays the following text:

```
Welcome to day 14
For Loop
Printed number - 1
Printed number - 2
Printed number - 3
Printed number - 4
Printed number - 5
Printed number - 6
Printed number - 7
Printed number - 8
Printed number - 9
Printed number - 10
```

Each line of output is preceded by a timestamp and the file path 'script.js:6'. The 'Console' tab is highlighted in blue at the top of the developer tools interface.



nerd_jfpb





nerdjfpb



nerdjfpb

Do you understand the For loop ?



nerd_jfpb

JavaScript Series

For Vs While





nerdjfpb



nerdjfpb

We have learned about both
for and while.

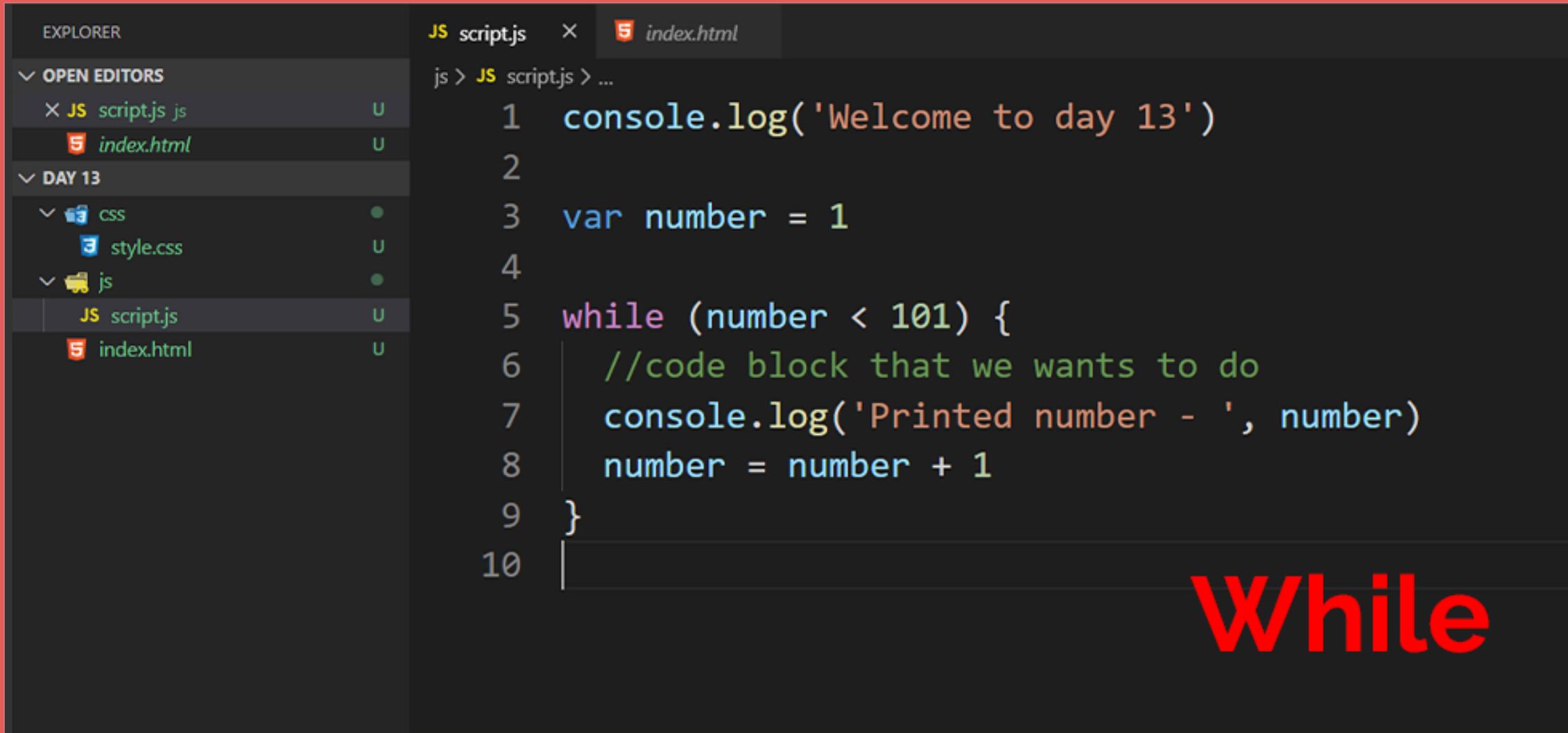
So let's get a comprehension
today!



nerd_jfpb



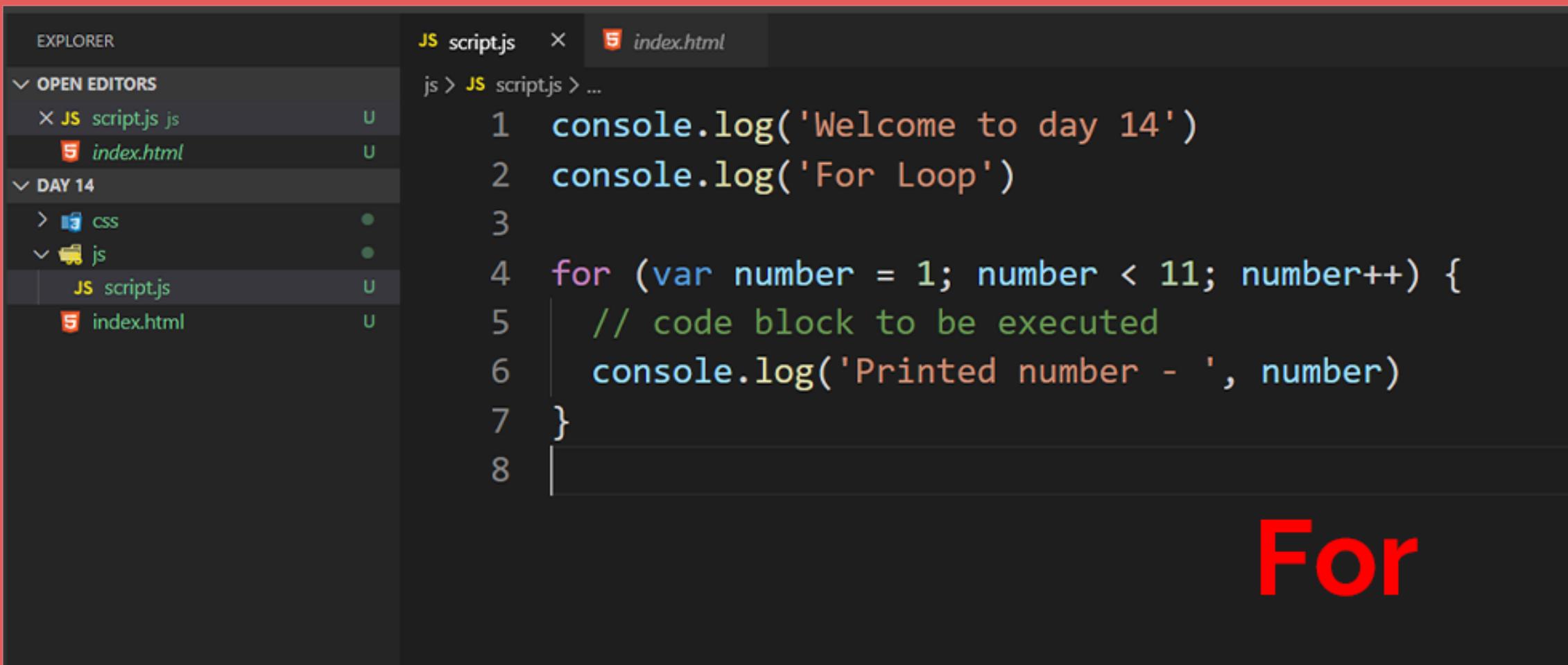
See both in one go.



The screenshot shows the VS Code interface with two open files: `script.js` and `index.html`. The `script.js` file contains the following code:

```
1 console.log('Welcome to day 13')
2
3 var number = 1
4
5 while (number < 101) {
6     //code block that we wants to do
7     console.log('Printed number - ', number)
8     number = number + 1
9 }
10
```

A large red **While** word is overlaid on the bottom right of the code editor.



The screenshot shows the VS Code interface with two open files: `script.js` and `index.html`. The `script.js` file contains the following code:

```
1 console.log('Welcome to day 14')
2 console.log('For Loop')
3
4 for (var number = 1; number < 11; number++) {
5     // code block to be executed
6     console.log('Printed number - ', number)
7 }
8
```

A large red **For** word is overlaid on the bottom right of the code editor.



In while Normally we have

```

```
while (condition) {
 // code block to be executed
}
```
```





In For Normally we have

```

```
for (initial, condition, increment
 decrement/rulesBreaking)
```

```
{
```

```
 // code block to be executed
```

```
}
```

```





nerdjfpb



nerdjfpb

So basically both do the same
work in different style right ?



nerd_jfpb





nerdjfpb



nerdjfpb

Yaah that's true.
Now it's your choice which one
you want to use!



nerd_jfpb





nerdjfpb



nerdjfpb

For loop is more readable in my sense, so I use for loop more.



nerd_jfpb





nerdjfpb



nerdjfpb

Which loop you like most and why ?



nerd_jfpb





nerdjfpb



nerdjfpb

Are you enjoying this series ?



nerd_jfpb

JavaScript Series

Functions





Function is a **block of code** where we write some tasks or a single task to do.

Function run only when we call it.
Sound confusing ?





nerdJfpb



nerdJfpb

If no then just skip next two slides/lines.



nerd_jfpb





Do you remember using
`console.log?`

What do you think this is ?

When we `pass something on the
"" in console.log` then we can see it
is printing on the console right ?
How this happens?





Actually there is **some native parts** of JavaScript.

Where there is function and we **call it using it and pass the value to show in the console**.

So we are already using the function right ?

Do you understand now ?





Now writing function in JavaScript
is really easy.
Just start with **function keyword**.
This is **reserve keyword** for
initializing a function like we initial
variable using var.





It is like giving name of variable.
We'll give a name after function keyword.

Then function need some **parameter** to pass.

We can leave it blank for now and our code will be -

``

```
function nameOfFunction() {  
    //Block of code  
}  
``
```





Now lets write a function -

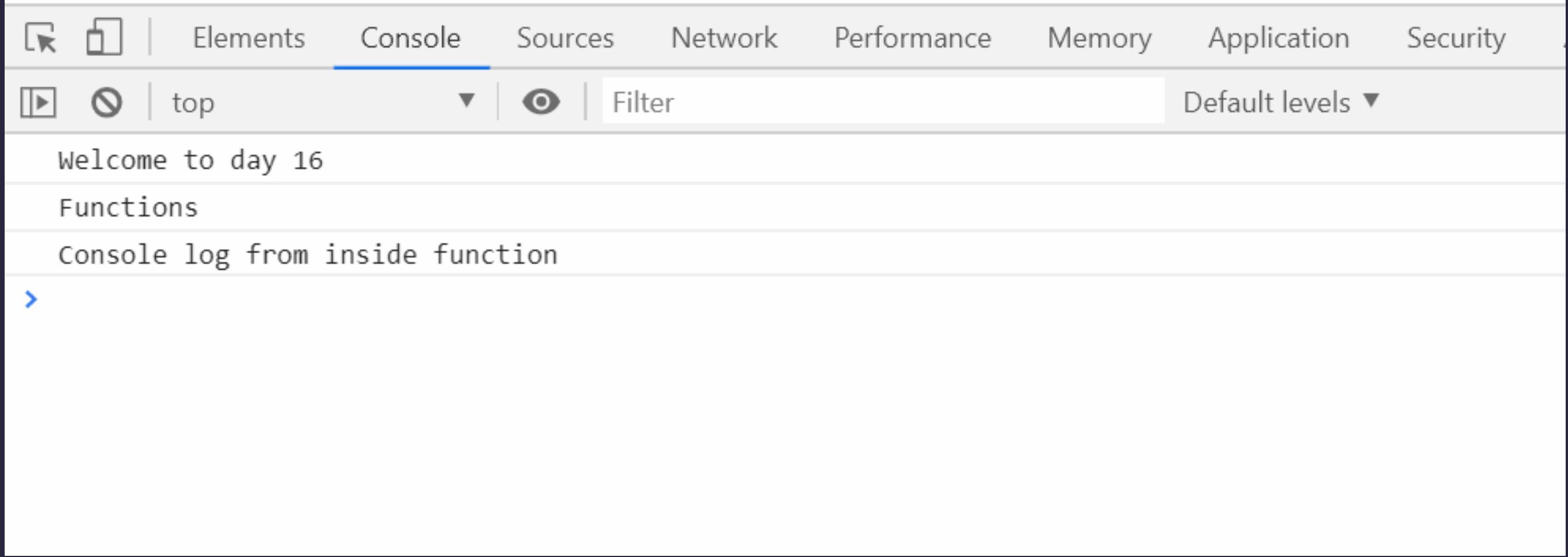
The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure with files: script.js, style.css, and index.html under a folder named DAY 16. The script.js file is currently open in the editor. The code in the editor is as follows:

```
1 console.log('Welcome to day 16')
2 console.log('Functions')
3
4 function simple() {
5     console.log('Console log from inside function')
6 }
7
8 simple()
9 
```



See the result in browser console

Welcome to JavaScript Tutorials!



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following text:

```
Welcome to day 16
Functions
Console log from inside function
```



nerdjfpb



nerdjfpb

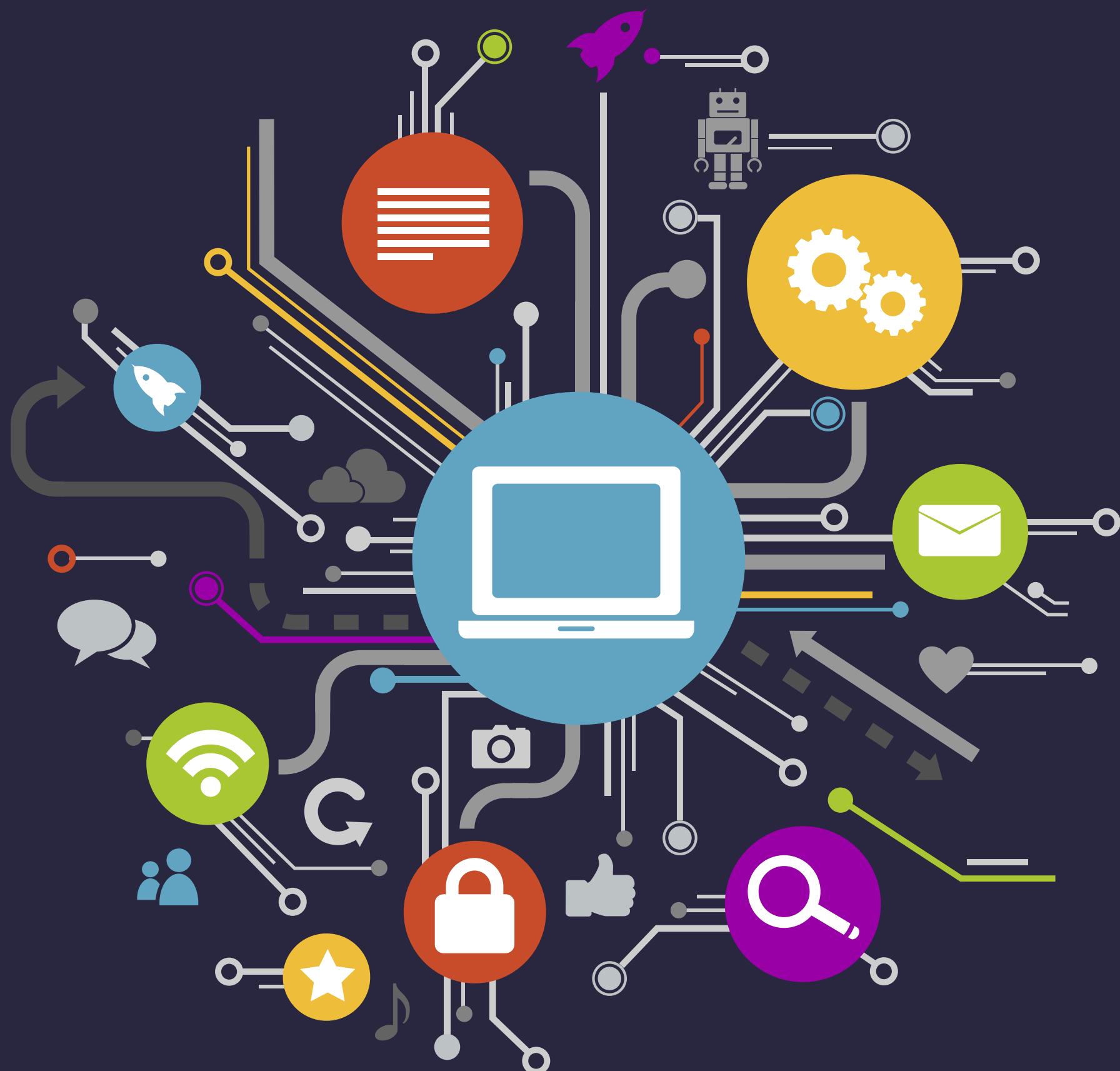
Can you write some basic functions now ?



nerd_jfpb

JavaScript Series

More About Functions





nerdjfpb



nerdjfpb

In last part of function we learn about a basic function and we just console.log inside the function but today we are going to learn more about it. We'll learn how we can add two numbers using a function!



nerd_jfpb





nerdjfpb



nerdjfpb

In last part we didn't use any
parameters right?
Now we'll use some parameters.



nerd_jfpb





We'll pass the parameters inside of "()".

We can pass variables here,
even function too.

But for now just think of variable.





nerdjfpb



nerdjfpb

Today we're going to create a plus function which will give us the result of two number.



nerd_jfpb





First we're going to write function keyword and the name of function and we'll pass two numbers.

``

```
function plus(number1,number2){  
  console.log(number1)  
  console.log(number2)  
}  
plus(10,12)  
``
```



We can easily access the values from the parameters.
Now we can store both in a variable like -

```

```
function plus(number1,number2){
 var total = number1 + number2
}
```
```



Now we can console log it on our console.

The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure with files: script.js, style.css, and index.html under DAY 17. The script.js file is open in the main editor area. The code contains a function named plus that adds two numbers and logs the result to the console:

```
1  console.log('Welcome to day 17')
2  console.log('More About Functions')
3
4  function plus(number1, number2) {
5      var total = number1 + number2
6      console.log('Total - ', total)
7  }
8
9  plus(10, 12)
10
```





nerdJfpb



nerdJfpb

See the result

Welcome to JavaScript Tutorials!

Elements **Console** Sources Network Performance

top Filter

```
Welcome to day 17
More About Functions
Total - 22
>
```



nerd_jfpb





nerdJfpb



nerdJfpb

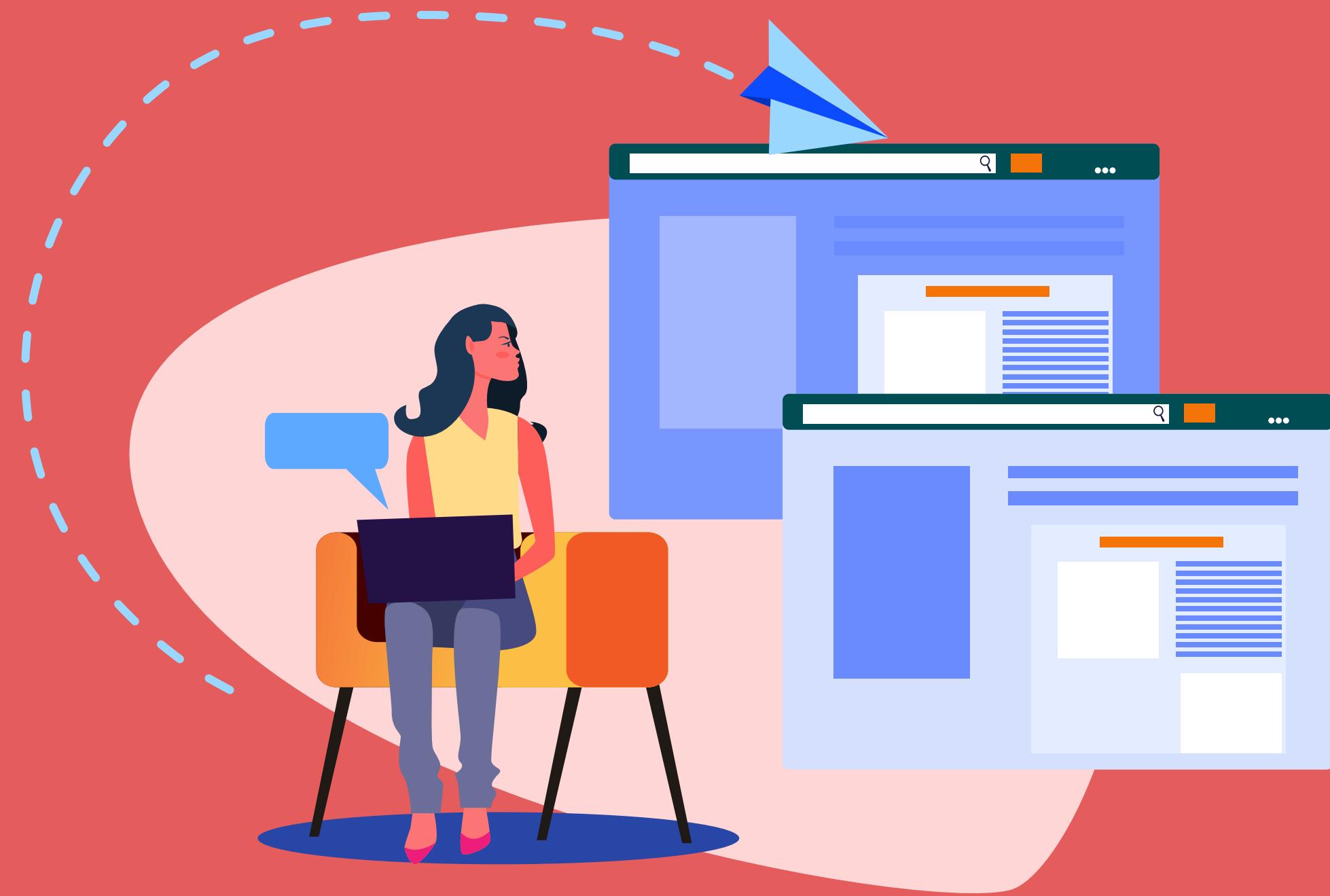
Can you write the minus function now ?



nerd_jfpb

JavaScript Series

Different types of function





In last two part we declare the function in a same style, but there is **another way** declare a variable. We are going to learn it today.





Up until now we wrote the functions declaration.

Now we are going to learn about **function expression**.





Here we don't have a name for the function.
So it also called
anonymous function.





nerdjfpb



nerdjfpb

Let's write one to understand more about it.



nerd_jfpb





```
'''
```

```
function() {  
  console.log('How you doin ?')  
}  
'''
```

this will be function and we'll store it on a variable and then we'll call the function using variable()





Example-

```
10
11 //function expression
12 const joeyTalking = function() {
13   console.log('How you doin ?')
14 }
15
16 joeyTalking()
17
```





nerdJfpb



nerdJfpb

This exactly does the same thing as our old function.

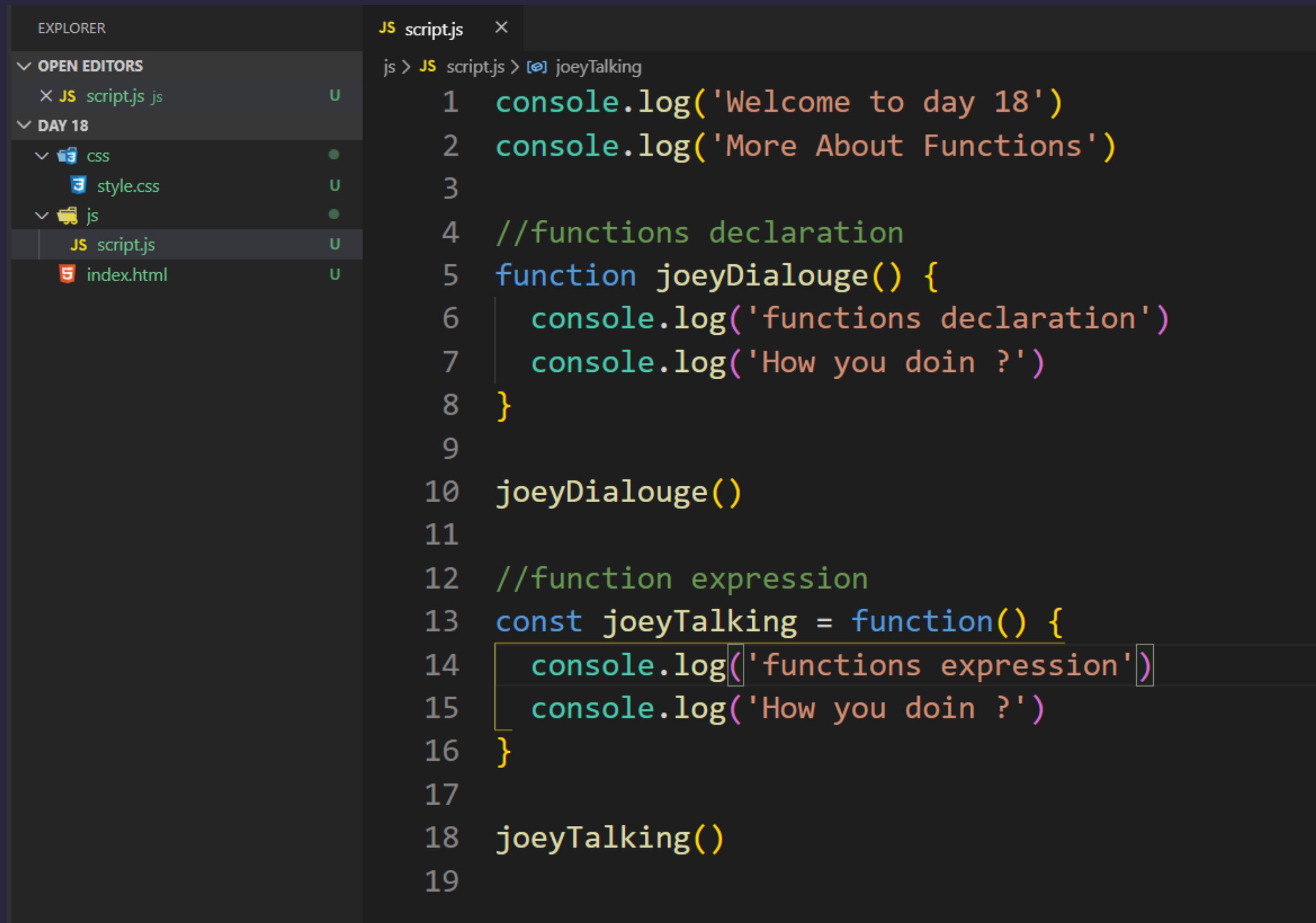
This pattern is widely use in js, so it is better to know both.



nerd_jfpb



Let's have a look on the both type



The screenshot shows a dark-themed code editor with the following file structure in the Explorer sidebar:

- OPEN EDITORS: script.js (marked with a green 'U')
- DAY 18
 - css (marked with a green 'U')
 - style.css (marked with a green 'U')
 - js (marked with a green 'U')
 - script.js (marked with a green 'U')
 - index.html (marked with a green 'U')

The script.js file contains the following code:

```
JS script.js  x
js > JS script.js > [o] joeyTalking
1 console.log('Welcome to day 18')
2 console.log('More About Functions')
3
4 //functions declaration
5 function joeyDialouge() {
6   console.log('functions declaration')
7   console.log('How you doin ?')
8 }
9
10 joeyDialouge()
11
12 //function expression
13 const joeyTalking = function() {
14   console.log('functions expression')
15   console.log('How you doin ?')
16 }
17
18 joeyTalking()
19
```

The code illustrates two ways to define functions: a standard function declaration and a function expression assigned to a constant.



nerdjfpb



nerdjfpb

Result -

Welcome to JavaScript Tutorials!

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following text multiple times:

```
Welcome to day 18
More About Functions
functions declaration
How you doin ?
functions expression
How you doin ?
```

At the bottom of the console, there is a blue arrow pointing right and a vertical line, indicating that more content is available below the visible area.



nerd_jfpb

JavaScript Series

Return in functions





Until now we didn't worked with
the return in functions.
But there is a return in function.





nerdjfpb



nerdjfpb

If we don't write anything in return
then it just return undefined.
But we can return anything.
Like - integer, string etc.



nerd_jfpb





nerdjfpb



nerdjfpb

Let's make our hand dirty by
coding.



nerd_jfpb





Suppose in function we are going to two parameters.

One is name and another is dialouge.

Now we are going to print it on the console using return within a function.





Now get the values from params first and put it inside a new variable. Now we are going to return this variable.

```
function dialouge(name, dialouge) {  
  const nameWithDialouge = name + ' ' + dialouge  
  return nameWithDialouge  
}
```





Now we need to change our function call.

Because this function is not doing anything.

It is returning something, we are going to catch the value and print it.



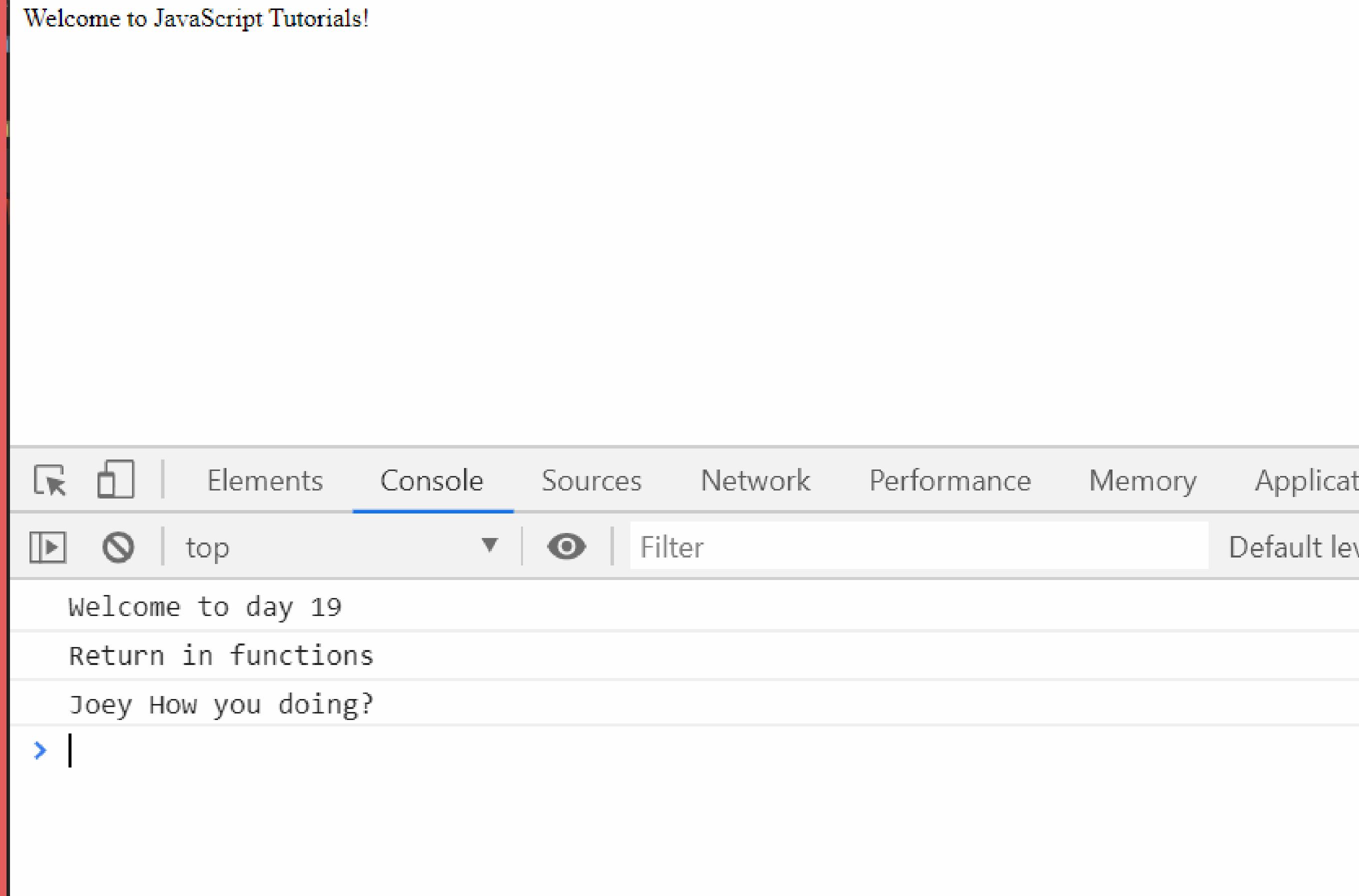


This step is a pretty easy to.
Just initial a variable by calling the
function.

```
1 console.log('Welcome to day 19')
2 console.log('Return in functions')
3
4 function dialouge(name, dialouge) {
5   const nameWithDialouge = name + ' ' + dialouge
6   return nameWithDialouge
7 }
8
9 const dialougeForPrint = dialouge('Joey', 'How you doing?')
10 console.log(dialougeForPrint)
11
```



See the result in browser -



Welcome to JavaScript Tutorials!

Elements Console Sources Network Performance Memory Application

top Filter Default level

```
Welcome to day 19
Return in functions
Joey How you doing?
```



nerdjfpb



nerdjfpb

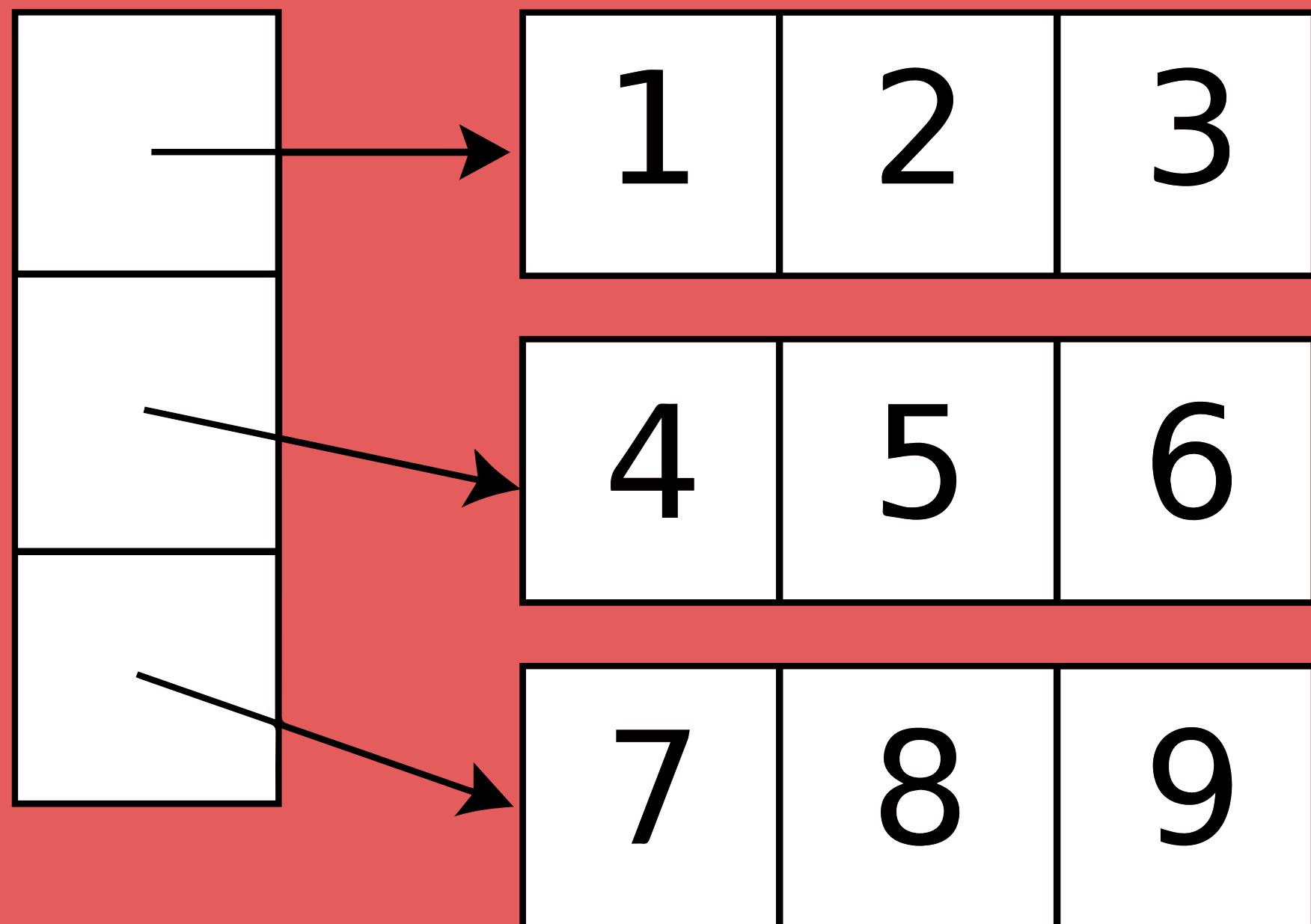
Can you write some functions now ?



nerd_jfpb

JavaScript Series

Array





nerdjfpb



nerdjfpb

We know about variable right?
Now we are going to learn about
data structures.



nerd_jfpb





In computer science, a data structure is a **data organization, management and storage format** that **enables efficient access and modification**.

More precisely, a data structure is a **collection of data values, the relationships among them, and the functions or operations that can be applied to the data**.

[wikipedia]





I think it's easy enough to understand.

Up until now we were just using the variable to store a single data, now we are going to learn about **array** where we can **store multiple** data.





nerdjfpb



nerdjfpb

It's easy.

we need to initial a variable and
put some values in there.



nerd_jfpb





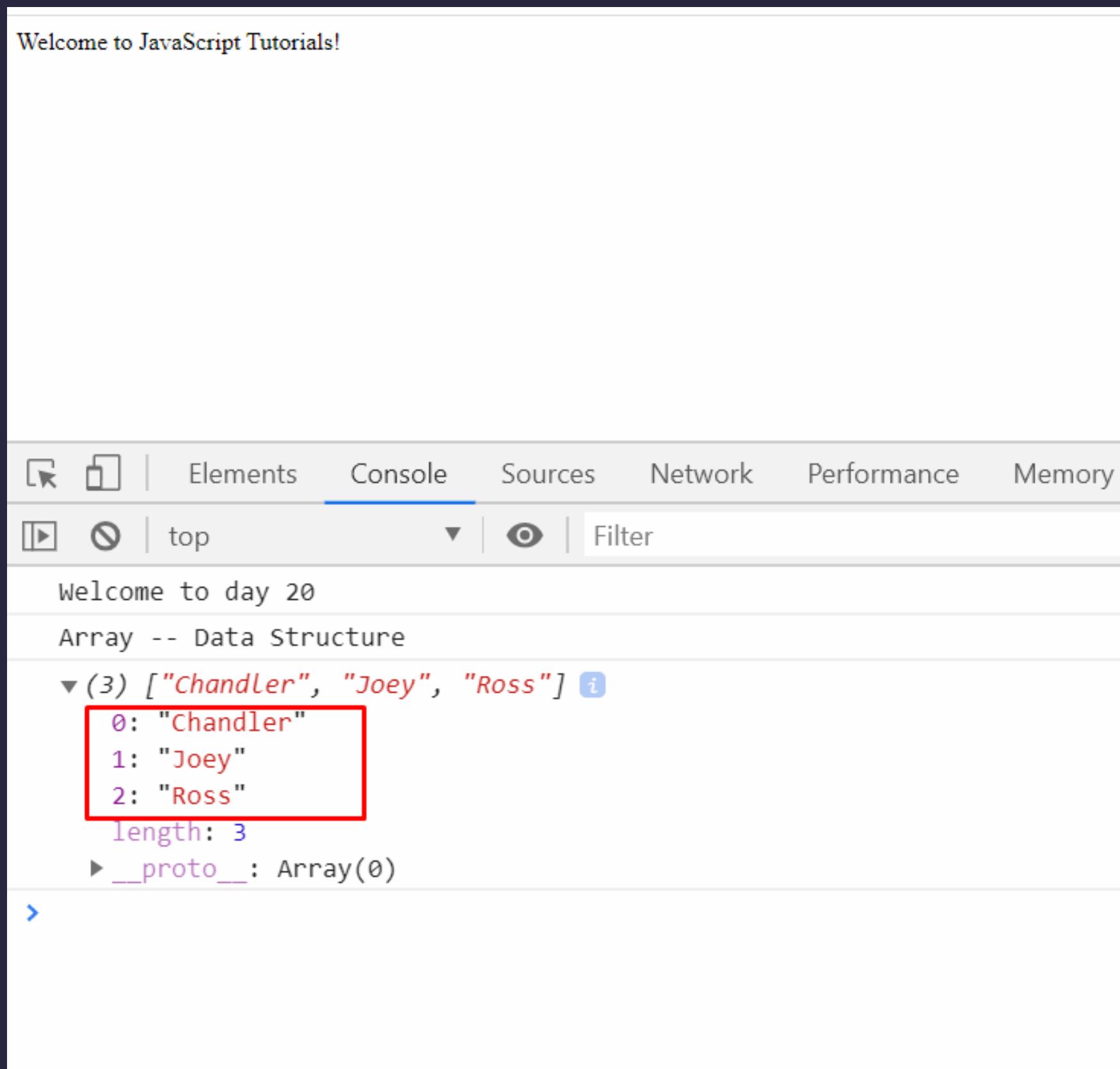
See this is so easy-

js > JS script.js > ...

```
1 console.log('Welcome to day 20')
2 console.log('Array -- Data Structure')
3
4 var friendsCharacters = ['Chandler', 'Joey', 'Ross']
5 console.log(friendsCharacters)
6 |
```



In the browser we can see some interesting things about the array. There is some indexing here of the each elements.



Array indexing count from one.
If we want, we can just console log
'Chandler` using array -

```
js > JS script.js > ...
1  console.log('Welcome to day 20')
2  console.log('Array -- Data Structure')
3
4  var friendsCharacters = ['Chandler', 'Joey', 'Ross']
5  console.log(friendsCharacters[0])
6
```

If you give a value which is more than index value then it will give you undefined -

```
› JS script.js > ...
1 console.log('Welcome to day 20')
2 console.log('Array -- Data Structure')
3
4 var friendsCharacters = ['Chandler', 'Joey', 'Ross']
5 console.log(friendsCharacters[4])
6
```



nerdjfpb



nerdjfpb

Do you understand the Array ?



nerd_jfpb



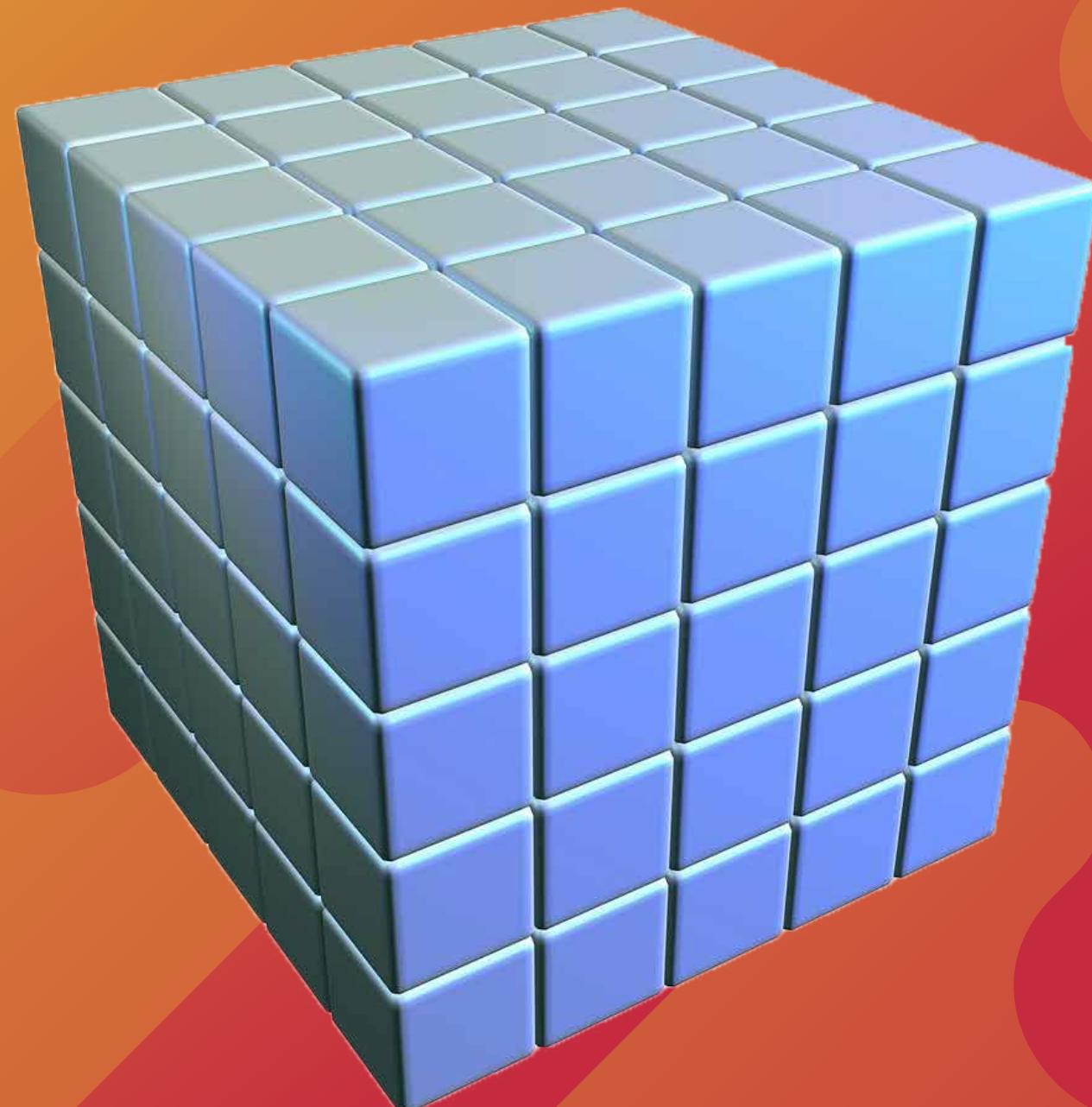
nerdJfpb



nerdJfpb

JavaScript Series

More About Array



nerd_jfpb

Part 21





nerdjfpb



nerdjfpb

In last part we learned about array,
in this part we are going to go a
little deep around arary.



nerd_jfpb





nerdjfpb



nerdjfpb

Suppose we've an array of
**favAnimeList = ['One piece',
'Dr Stone', 'Haikyuu', 'Attack on
Titan']**



nerd_jfpb





nerdjfpb



nerdjfpb

Now we want to **delete** the
last one.
How we can do this ?



nerd_jfpb



Using the pop.

```
js > JS script.js > ...
1 console.log('Welcome to day 21')
2 console.log('More Array')
3
4 var favAnimeList = ['One piece', 'Dr Stone', 'Haikyuu', 'Attack on Titan']
5 favAnimeList.pop()
6 console.log(favAnimeList)
7 |
```





nerdjfpb



nerdjfpb

Now we want to add "naruto"
in the array.
How we can do this ?



nerd_jfpb



Just using **push**.

This will add the value in end.

```
js script.js  x
js > js script.js > ...
1  console.log('Welcome to day 21')
2  console.log('More Array')
3
4  var favAnimeList = ['One piece', 'Dr Stone', 'Haikyuu', 'Attack on Titan']
5  //delete the last one from array
6  favAnimeList.pop()
7  console.log(favAnimeList)
8  //add a new item in array in last position
9  favAnimeList.push('Naruto')
10 console.log(favAnimeList)
11
```

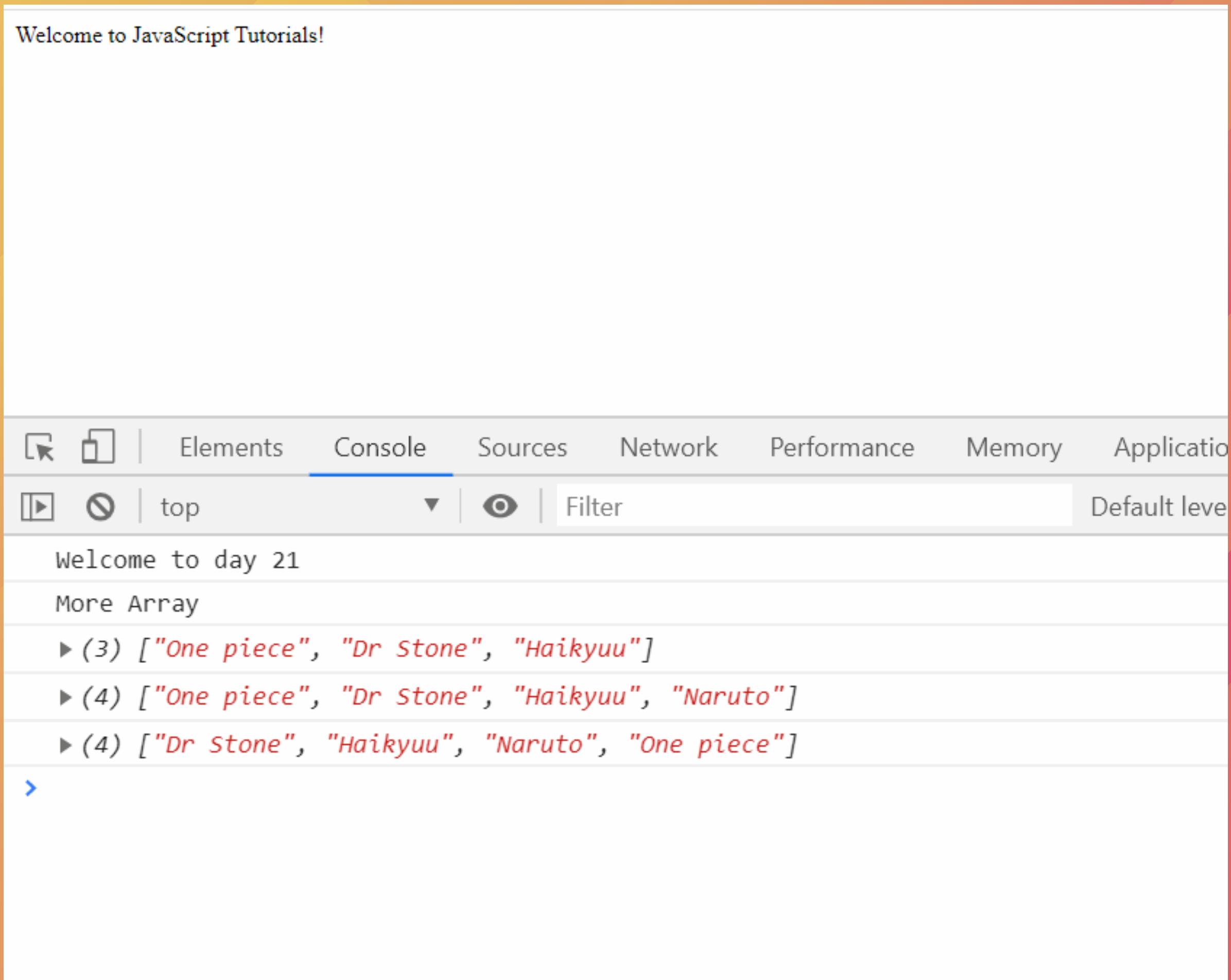


Now we can easily sort the array using **sort**.

```
JS script.js  ×
js > JS script.js > ...
1  console.log('Welcome to day 21')
2  console.log('More Array')
3
4  var favAnimeList = ['One piece', 'Dr Stone', 'Haikyuu', 'Attack on Titan']
5  //delete the last one from array
6  favAnimeList.pop()
7  console.log(favAnimeList)
8  //add a new item in array in last position
9  favAnimeList.push('Naruto')
10 console.log(favAnimeList)
11 //sort the array
12 favAnimeList.sort()
13 console.log(favAnimeList)
14
```



See all results on the browser



Welcome to JavaScript Tutorials!

Elements **Console** Sources Network Performance Memory Application

top Filter Default leve

```
Welcome to day 21
More Array
▶ (3) ["One piece", "Dr Stone", "Haikyuu"]
▶ (4) ["One piece", "Dr Stone", "Haikyuu", "Naruto"]
▶ (4) ["Dr Stone", "Haikyuu", "Naruto", "One piece"]
```





nerdjfpb



nerdjfpb

Can you use array now ?



nerd_jfpb



nerdJfpb



nerdJfpb

JavaScript Series

Object



nerd_jfpb

Part 22





nerdjfpb



nerdjfpb

Today we are going to learn about
a new javascript type.
This is called **‘Object’**



nerd_jfpb

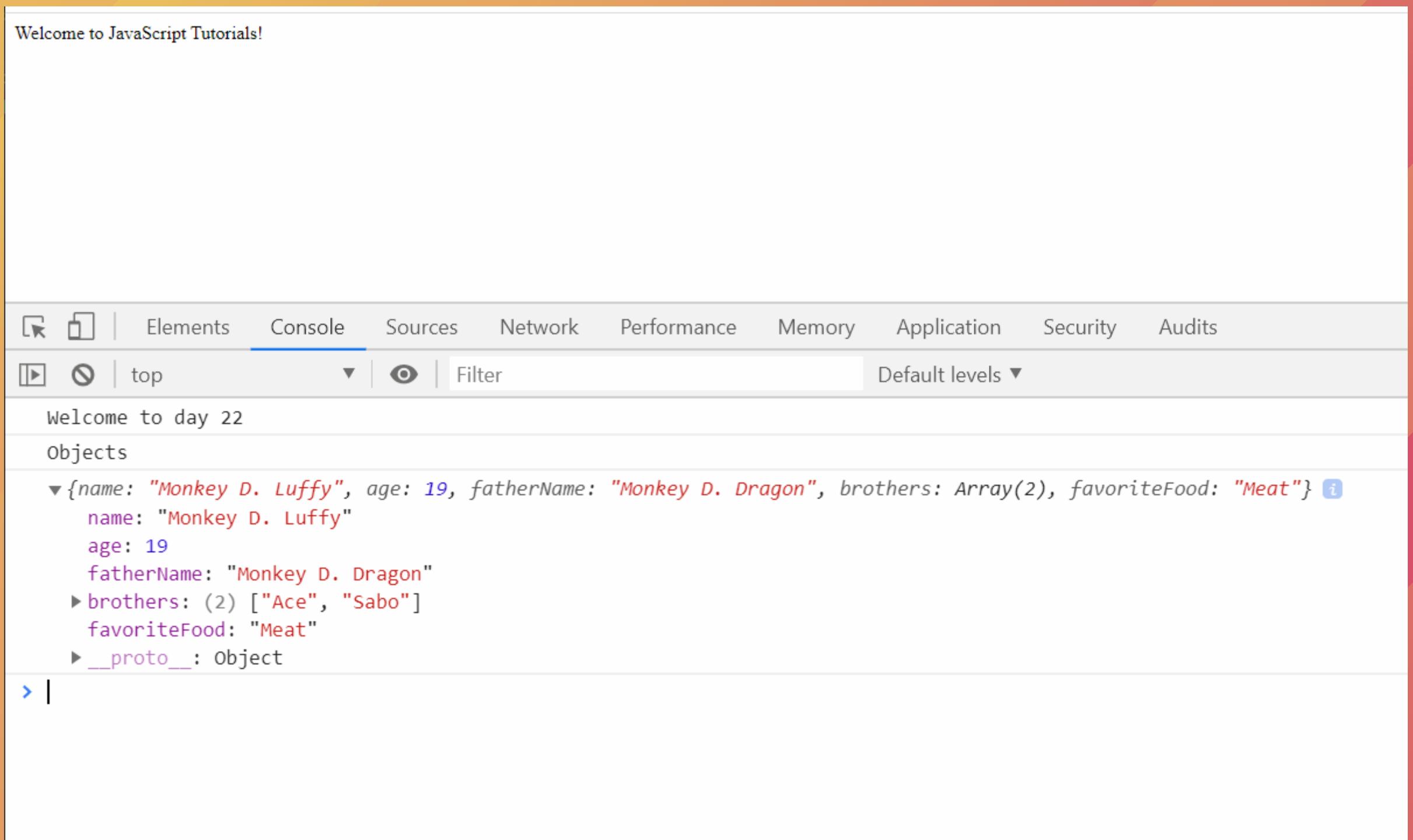


In object type we can store multiple types of data.

Like - numbers, string, arrays etc.

```
JS script.js  ×  
js > JS script.js > ...  
1  console.log('Welcome to day 22')  
2  console.log('Objects')  
3  
4  var characterDetails = {  
5      name: 'Monkey D. Luffy',  
6      age: 19,  
7      fatherName: 'Monkey D. Dragon',  
8      brothers: ['Ace', 'Sabo'],  
9      favoriteFood: 'Meat'  
10 }  
11
```

See what we got in the `characterDetails`



Welcome to JavaScript Tutorials!

Welcome to day 22

Objects

```
▼ {name: "Monkey D. Luffy", age: 19, fatherName: "Monkey D. Dragon", brothers: Array(2), favoriteFood: "Meat"} ⓘ
  name: "Monkey D. Luffy"
  age: 19
  fatherName: "Monkey D. Dragon"
  ▶ brothers: (2) ["Ace", "Sabo"]
    favoriteFood: "Meat"
  ▶ __proto__: Object
```



We can easily change the values of the object.

```
1 console.log('Welcome to day 22')
2 console.log('Objects')
3
4 var characterDetails = {
5     name: 'Monkey D. Luffy',
6     age: 19,
7     fatherName: 'Monkey D. Dragon',
8     brothers: ['Ace', 'Sabo'],
9     favoriteFood: 'Meat'
10 }
11
12 characterDetails.age = 20
13
14 console.log(characterDetails)
15
```





nerdjfpb



nerdjfpb

Do you know we can write function
inside of object ?



nerd_jfpb

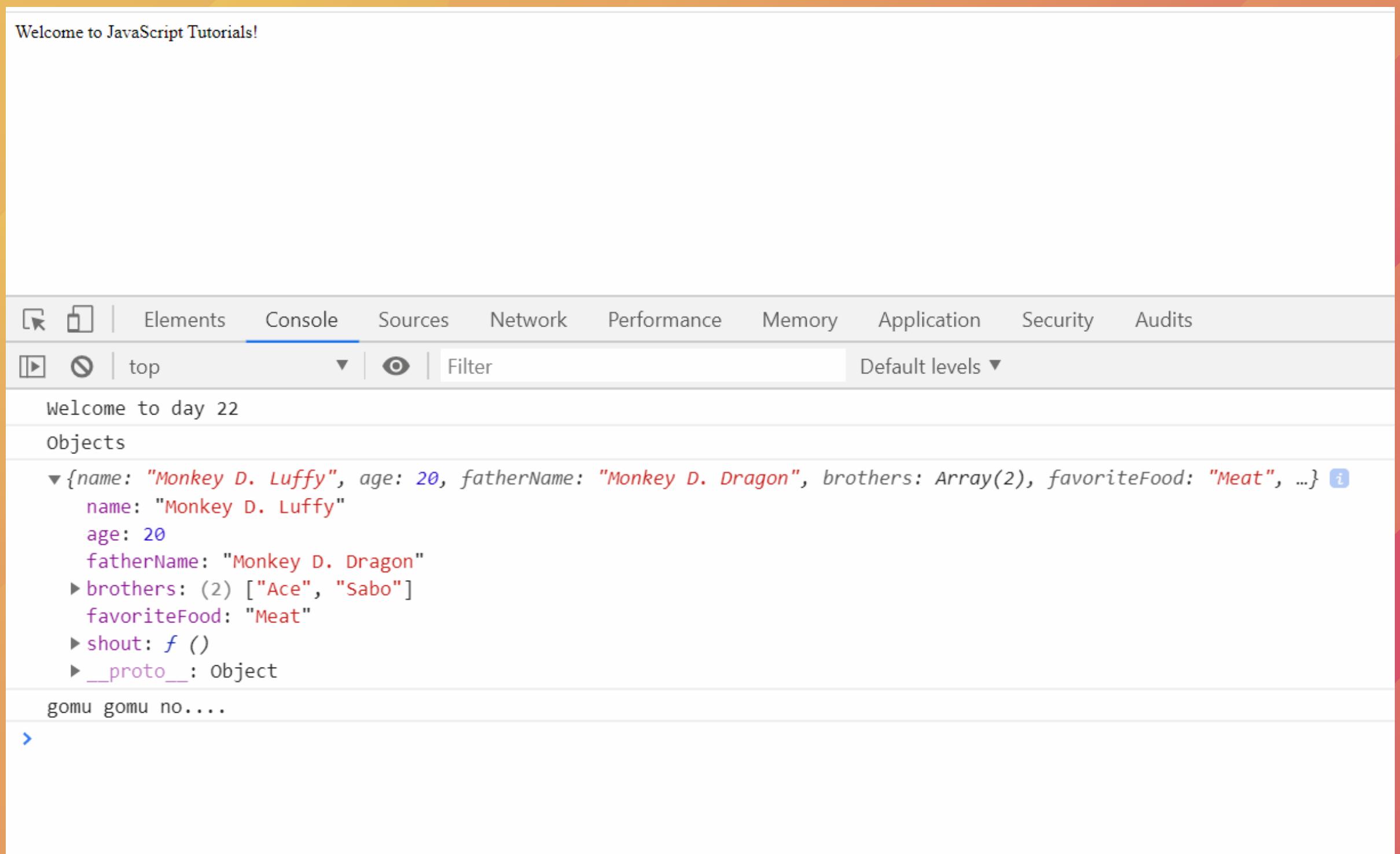


like -

```
JS script.js  x
js > JS script.js > ...
1  console.log('Welcome to day 22')
2  console.log('Objects')
3
4  var characterDetails = {
5      name: 'Monkey D. Luffy',
6      age: 19,
7      fatherName: 'Monkey D. Dragon',
8      brothers: ['Ace', 'Sabo'],
9      favoriteFood: 'Meat',
10     shout: function() {
11         console.log('gomu gomu no....')
12     }
13 }
14
15 characterDetails.age = 20
16
17 console.log(characterDetails)
18
19 characterDetails.shout()
20
```



See the result



Welcome to JavaScript Tutorials!

Elements **Console** Sources Network Performance Memory Application Security Audits

top Filter Default levels ▾

```
Welcome to day 22
Objects
▼ {name: "Monkey D. Luffy", age: 20, fatherName: "Monkey D. Dragon", brothers: Array(2), favoriteFood: "Meat", ...} ⓘ
  name: "Monkey D. Luffy"
  age: 20
  fatherName: "Monkey D. Dragon"
▶ brothers: (2) ["Ace", "Sabo"]
  favoriteFood: "Meat"
▶ shout: f ()
▶ __proto__: Object
gomu gomu no....
```





nerdjfpb



nerdjfpb

Can you initial a object now ?



nerd_jfpb





nerdjfpb



nerdjfpb

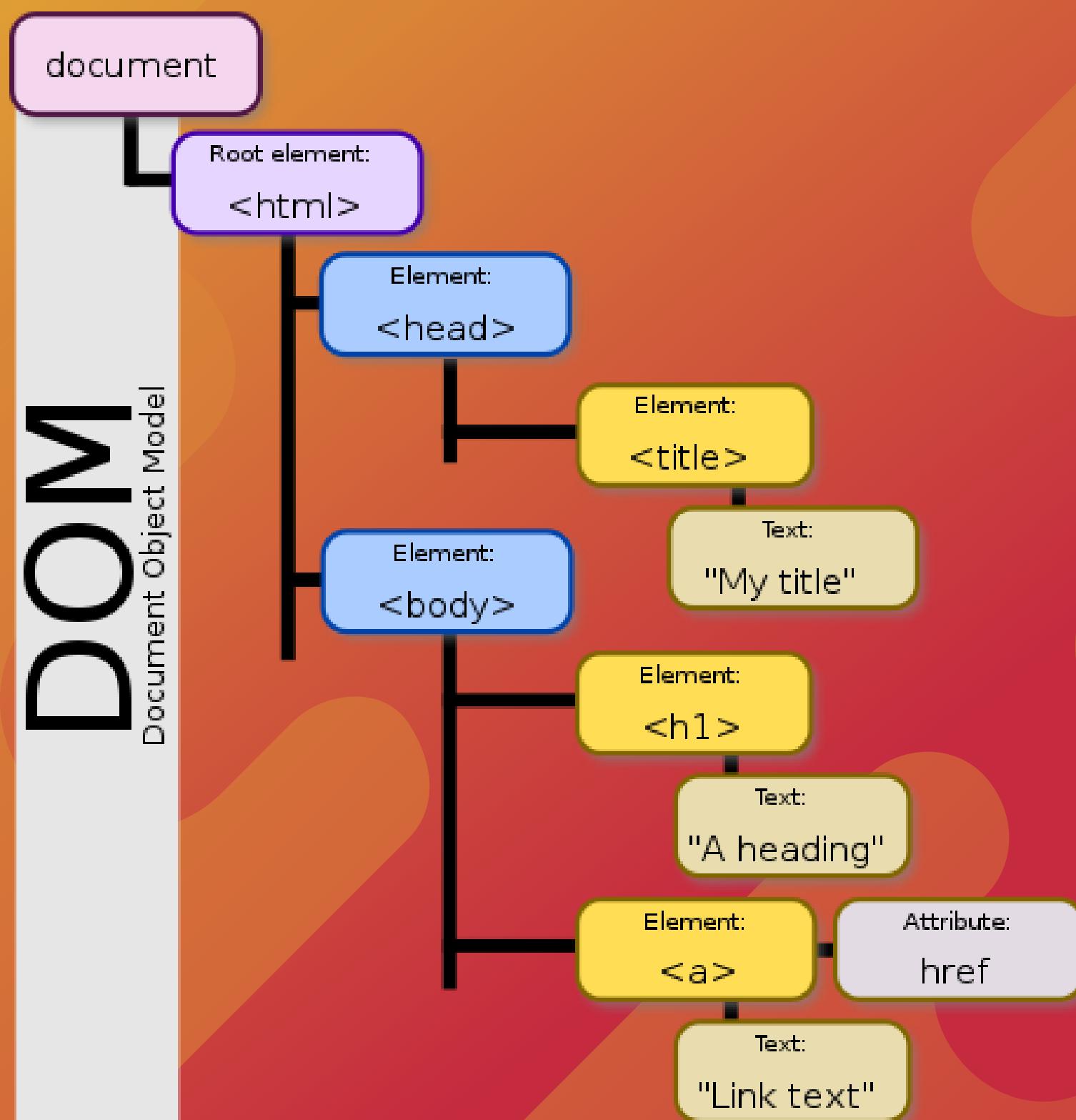
Are you enjoying this tutorial ?



nerd_jfpb

JavaScript Series

Dom Manipulation





nerdjfpb



nerdjfpb

We've ended the javascript basics,
what next ?

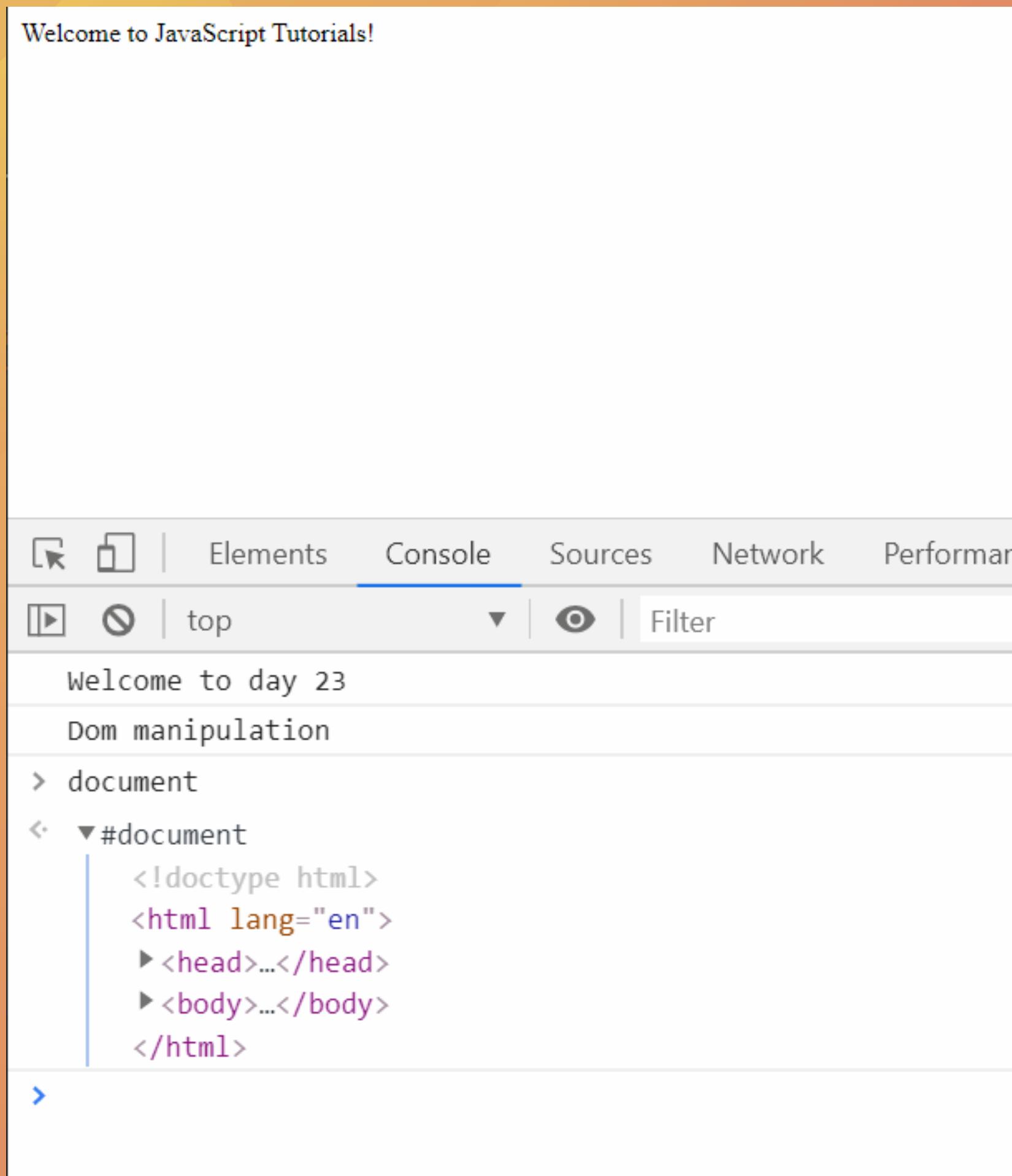
We are going to learn about the
javascript dom manipulation.



nerd_jfpb



Let's go to console in `write` document.





nerdjfpb



nerdjfpb

We can see the whole **html** we wrote right ?



nerd_jfpb





nerdjfpb



nerdjfpb

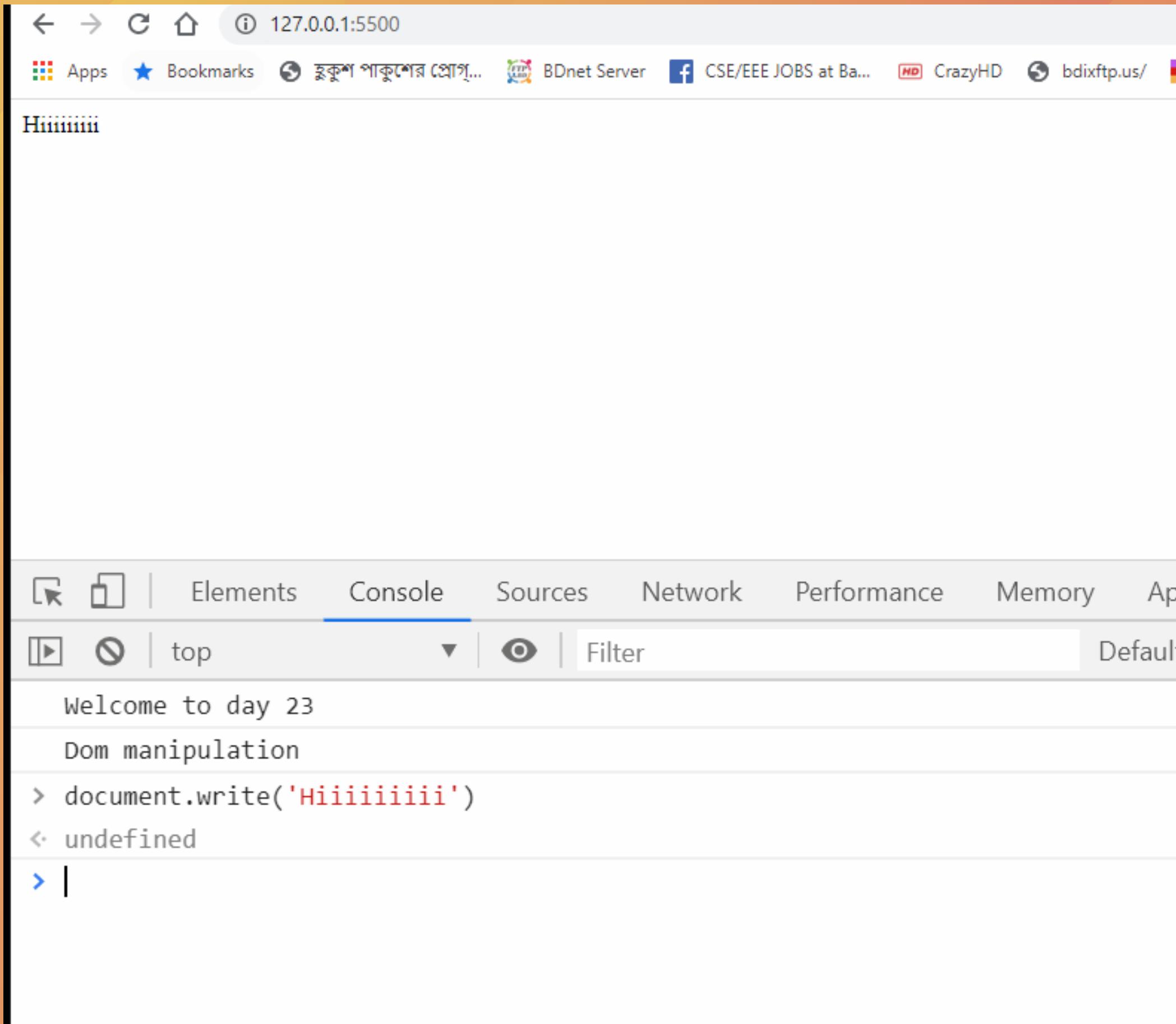
Now we can **manipulate** this with
our **javascript code**.
Sounds fun ?



nerd_jfpb



Let's try to write `document.write('Hiiiiiiii')` in console and see the magic!



The screenshot shows a browser window with the URL 127.0.0.1:5500. The page content displays "Hiiiiiiii". Below the page, the browser's developer tools are open, specifically the Console tab. The console output shows:

```
Welcome to day 23
Dom manipulation
> document.write('Hiiiiiiii')
< undefined
> |
```





nerdjfpb



nerdjfpb

So where the **document** is
coming from ?



nerd_jfpb





nerdjfpb



nerdjfpb

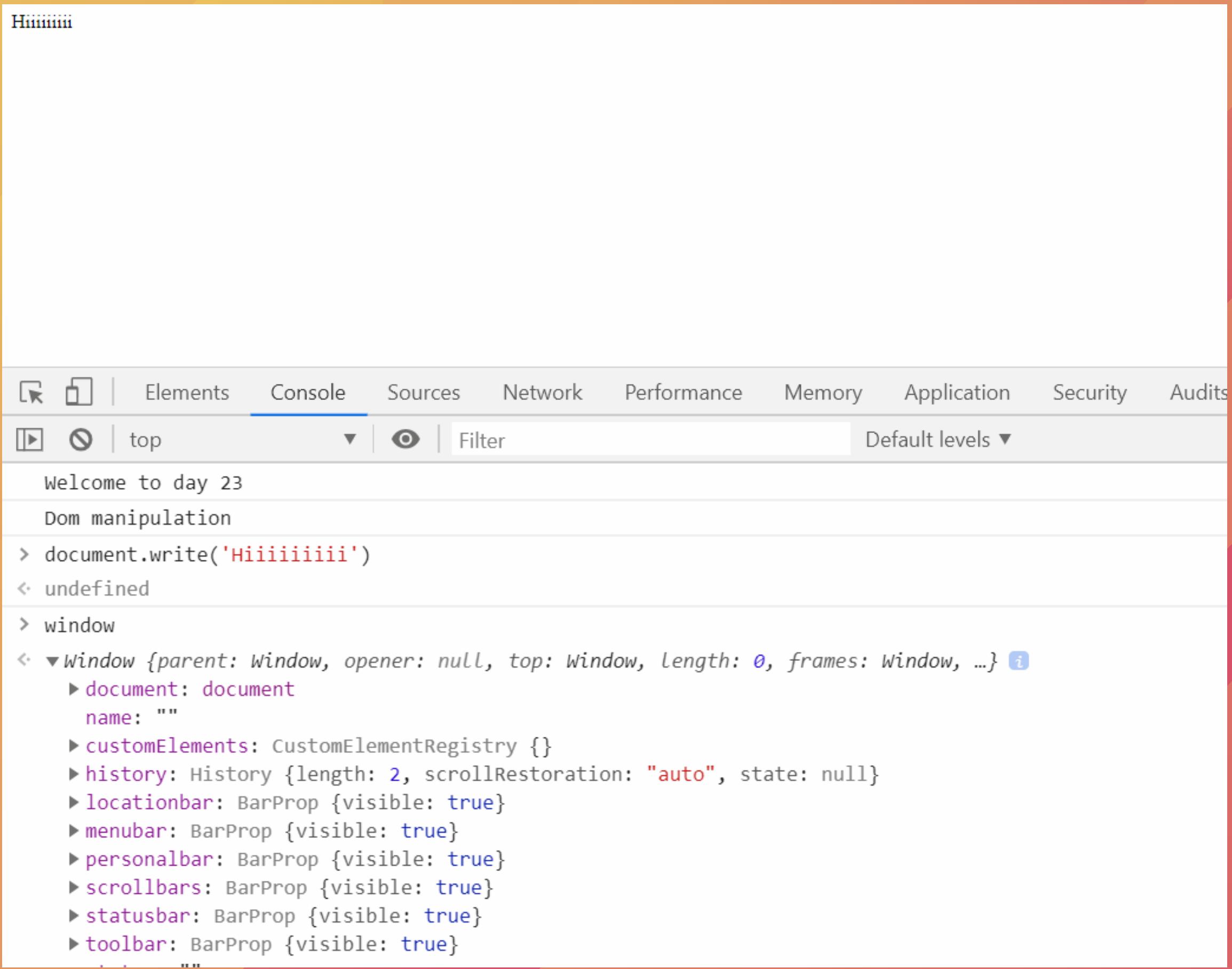
Document is a **global object** which
is available on the browser.
But it has a **parent**,
can you guess ?



nerd_jfpb



write `window` in the **console** and see what happens -



Hiiiiiii

Elements **Console** Sources Network Performance Memory Application Security Audits

top Default levels ▾

```
Welcome to day 23
Dom manipulation
> document.write('Hiiiiiii')
< undefined
> window
< ▼Window {parent: Window, opener: null, top: Window, length: 0, frames: Window, ...} ⓘ
  ► document: document
  name: ""
  ► customElements: CustomElementRegistry {}
  ► history: History {length: 2, scrollRestoration: "auto", state: null}
  ► locationbar: BarProp {visible: true}
  ► menubar: BarProp {visible: true}
  ► personalbar: BarProp {visible: true}
  ► scrollbars: BarProp {visible: true}
  ► statusbar: BarProp {visible: true}
  ► toolbar: BarProp {visible: true}
```





nerdjfpb



nerdjfpb

**Interested to make yourself into
a magician who can change the
dom ?**



nerd_jfpb



nerdjfpb



nerdjfpb

JAVASCRIPT SERIES-24

DOM SELECTOR



nerd_jfpb





In last tutorial we started
the **magic dom manipulation**.
Today we are going to learn
about the **dom selector** so that
we can select a dom part and
change it as we want.

Let's start.

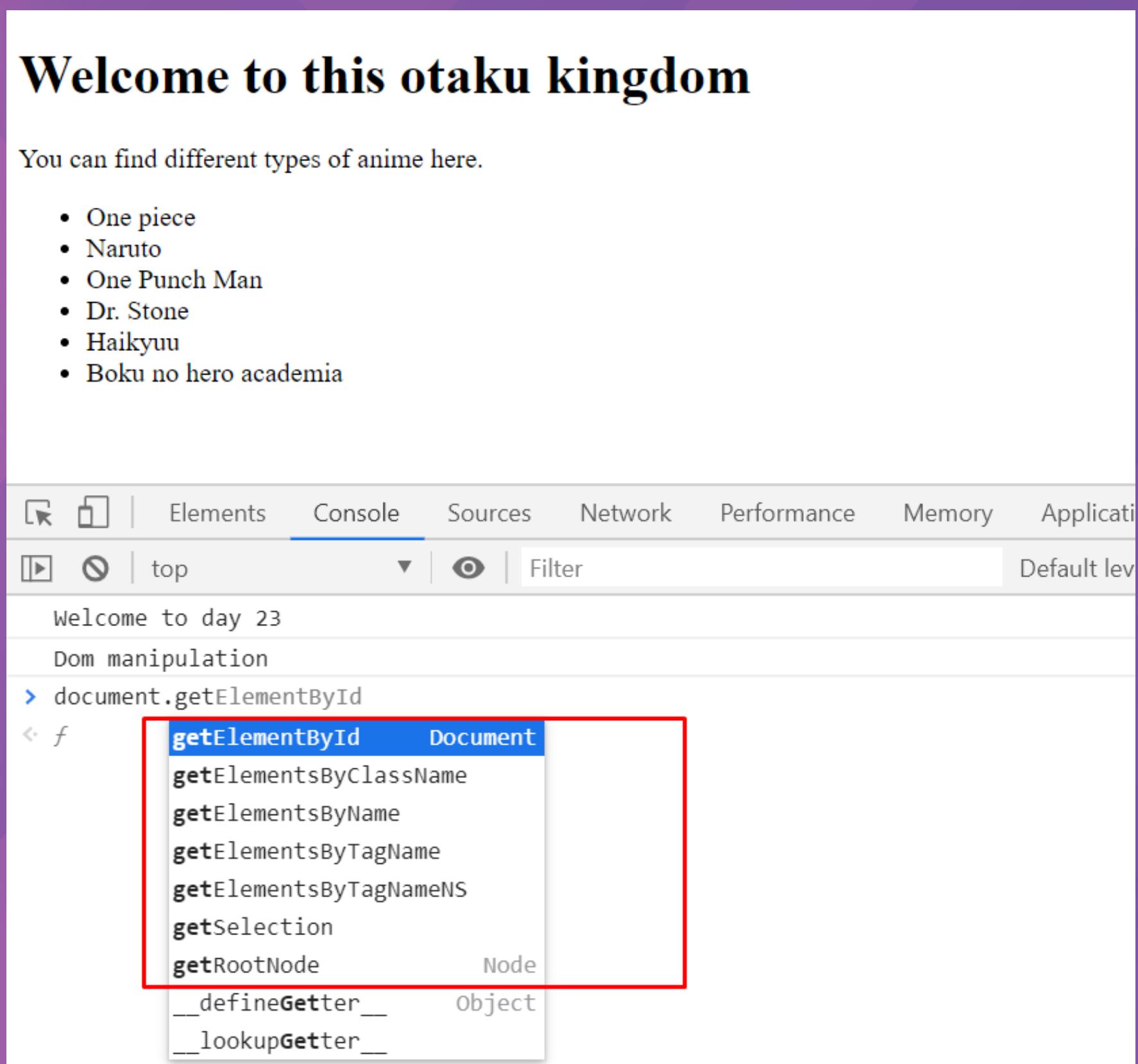


I made a new **html** for us to edit.
Have a look, **code available in
github.**

```
5 index.html ×
5 index.html > ⏺ html > ⏺ body > ⏺ h1.main-heading
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8" />
5    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6    <title>JavaScript Tutorial</title>
7    <link rel="stylesheet" href=".css/style.css" />
8  </head>
9  <body>
10  <h1 class="main-heading">Welcome to this otaku kingdom</h1>
11  <p class="sub-heading">You can find different types of anime here.</p>
12  <ul class="anime-list">
13    <li>One piece</li>
14    <li>Naruto</li>
15    <li>One Punch Man</li>
16    <li>Dr. Stone</li>
17    <li>Haikyuu</li>
18    <li>Boku no hero academia</li>
19  </ul>
20  <script src=".js/script.js"></script>
21  </body>
22 </html>
23
```



Now go to **console** and type `document`. Get it will bring the **autocomplete** other options.



So there is lot of **different selector** write?

We can use them to select something and edit that.

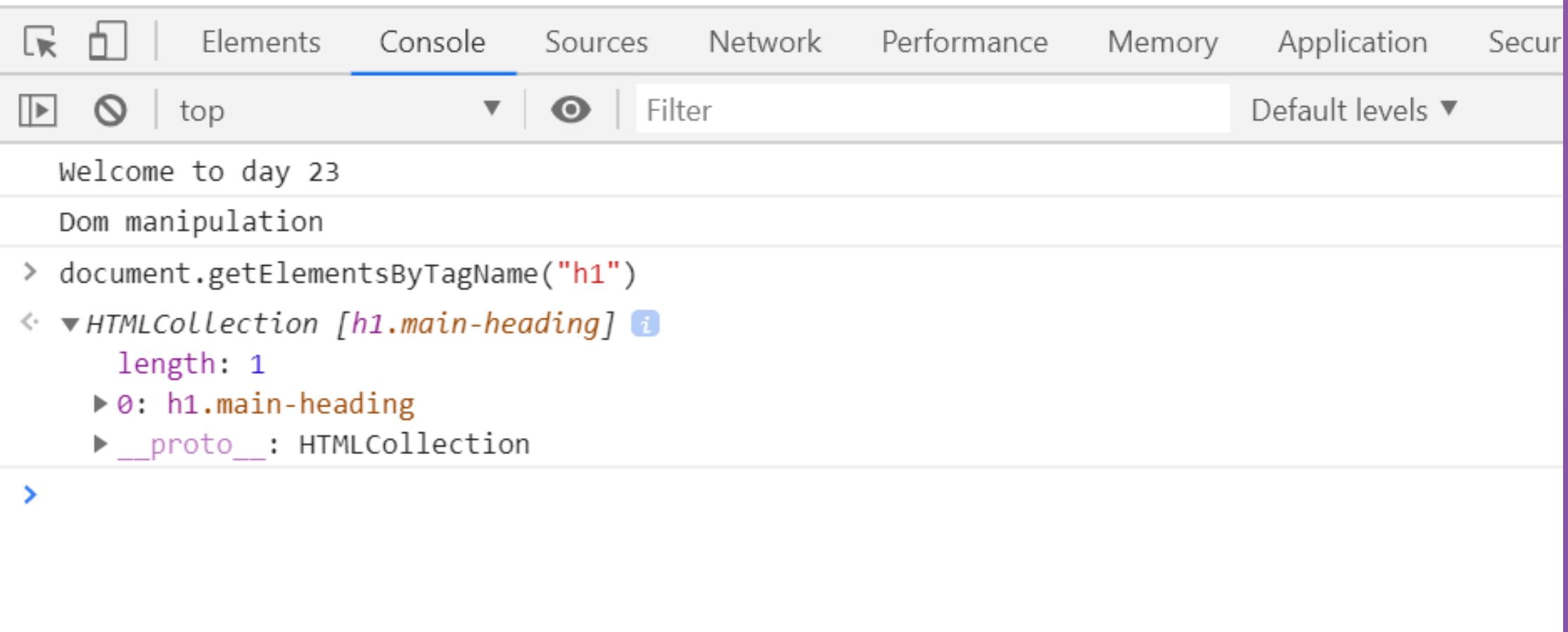
We're going to start from the **tag select**.

Now by just writing
`'document.getElementsByTagName("h1")'`
we can grab the h1. see

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows the browser's developer tools open to the 'Console' tab. The page content includes the heading 'Welcome to this otaku kingdom' and a list of anime titles. In the developer tools, the command `> document.getElementsByTagName("h1")` is entered, and the output is displayed as an `HTMLCollection` object with one item, `0: h1.main-heading`. The `length` property is shown as 1.

```
>Welcome to day 23
Dom manipulation
> document.getElementsByTagName("h1")
< HTMLCollection [h1.main-heading] ⓘ
  length: 1
  ▶ 0: h1.main-heading
  ▶ __proto__: HTMLCollection
>
```



We got and **HTMLCollection**
and in **0** we can see the
classname also.

So how can get the value ?

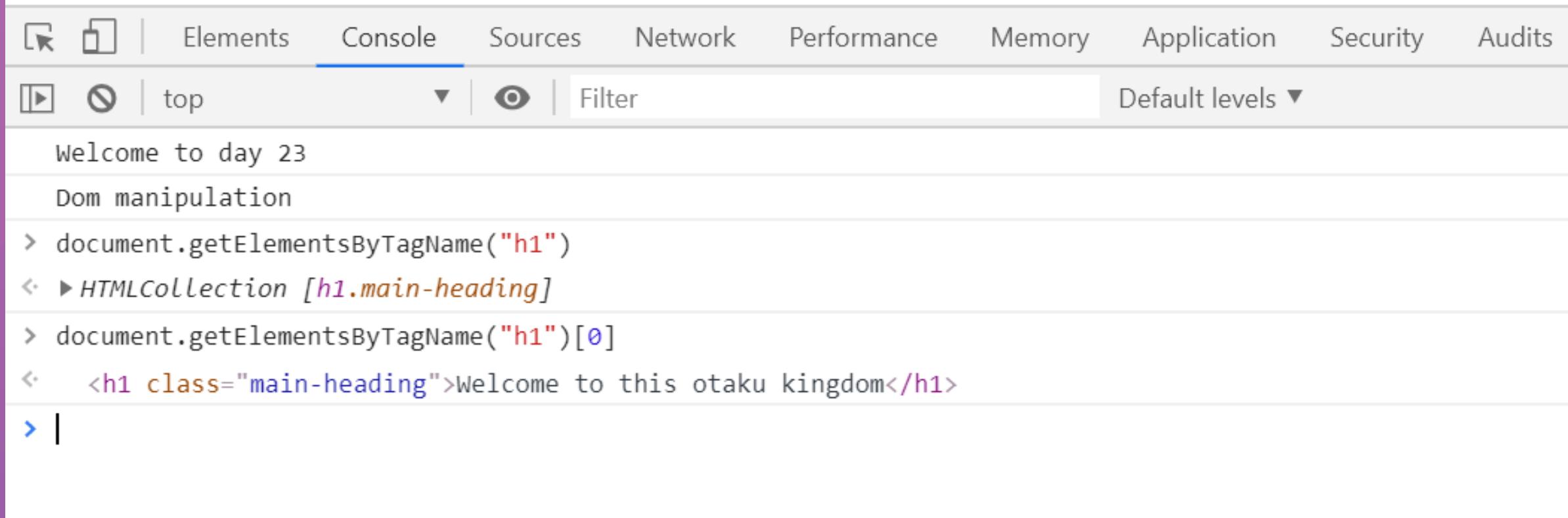


Just using `document.getElementsByTagName("h1")[0]`

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows the Chrome DevTools Elements tab selected. The page content is displayed above the tools. The DevTools interface includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. The Console tab is active, showing the following output:

```
Welcome to day 23
Dom manipulation
> document.getElementsByTagName("h1")
< ▶ HTMLCollection [h1.main-heading]
> document.getElementsByTagName("h1")[0]
<  <h1 class="main-heading">Welcome to this otaku kingdom</h1>
> |
```

To avoid writing this big thing we can just write
`'document.querySelector("h1")'` which is going to give the same thing for use.

The screenshot shows a browser's developer tools console tab selected. The page content above the console is a heading "Welcome to this otaku kingdom" and a list of anime titles. The console output shows two queries:

```
Welcome to day 23
Dom manipulation
> document.getElementsByTagName("h1")
<▶ HTMLCollection [h1.main-heading]
> document.getElementsByTagName("h1")[0]
<  <h1 class="main-heading">Welcome to this otaku kingdom</h1>
> document.querySelector("h1")
<  <h1 class="main-heading">Welcome to this otaku kingdom</h1>
>
```



nerdjfpb



nerdjfpb

So can you select the
elements
you have in your html ?



nerd_jfpb



nerdJfpb



nerdJfpb

JAVASCRIPT SERIES-25

STYLE CHANGE IN DOM



nerd_jfpb





Today we are going to change
the **style** of different elements
in the dom using some **new**
technique that we didn't learned
yet.





nerdjfpb



nerdjfpb

We are going you use the **last day codes.**

Welcome to this otaku kingdom

You can find different types of anime here.

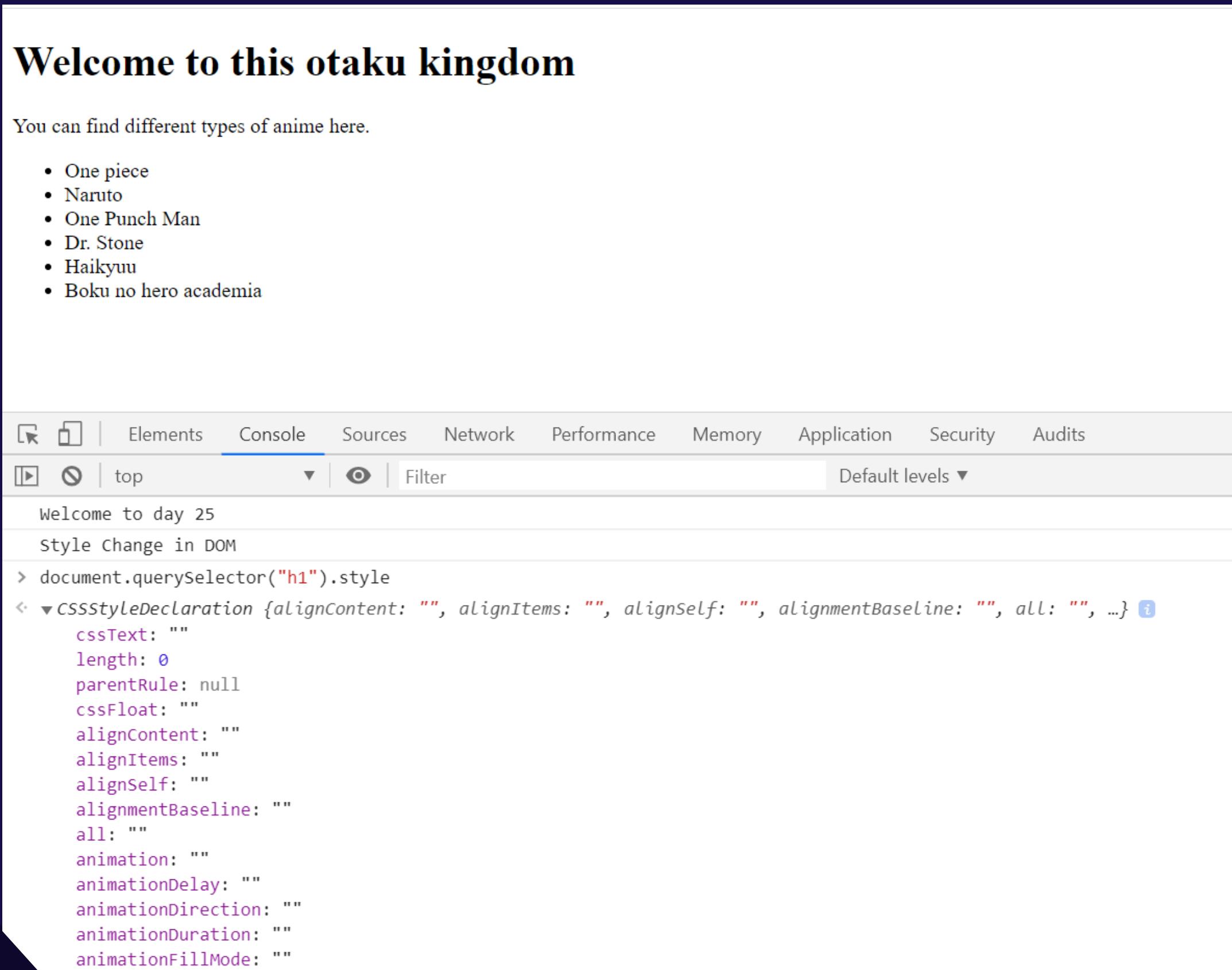
- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



nerd_jfpb



We'll start by **selecting** the **h1** and then we'll apply some style with it. See when we use **document.querySelector("h1").style**



Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia

Elements Console Sources Network Performance Memory Application Security Audits

top Default levels ▾

```
Welcome to day 25
Style Change in DOM
> document.querySelector("h1").style
< -> CSSStyleDeclaration {alignContent: "", alignItems: "", alignSelf: "", alignmentBaseline: "", all: "", ...} ⓘ
  cssText: ""
  length: 0
  parentRule: null
  cssFloat: ""
  alignContent: ""
  alignItems: ""
  alignSelf: ""
  alignmentBaseline: ""
  all: ""
  animation: ""
  animationDelay: ""
  animationDirection: ""
  animationDuration: ""
  animationFillMode: ""
```



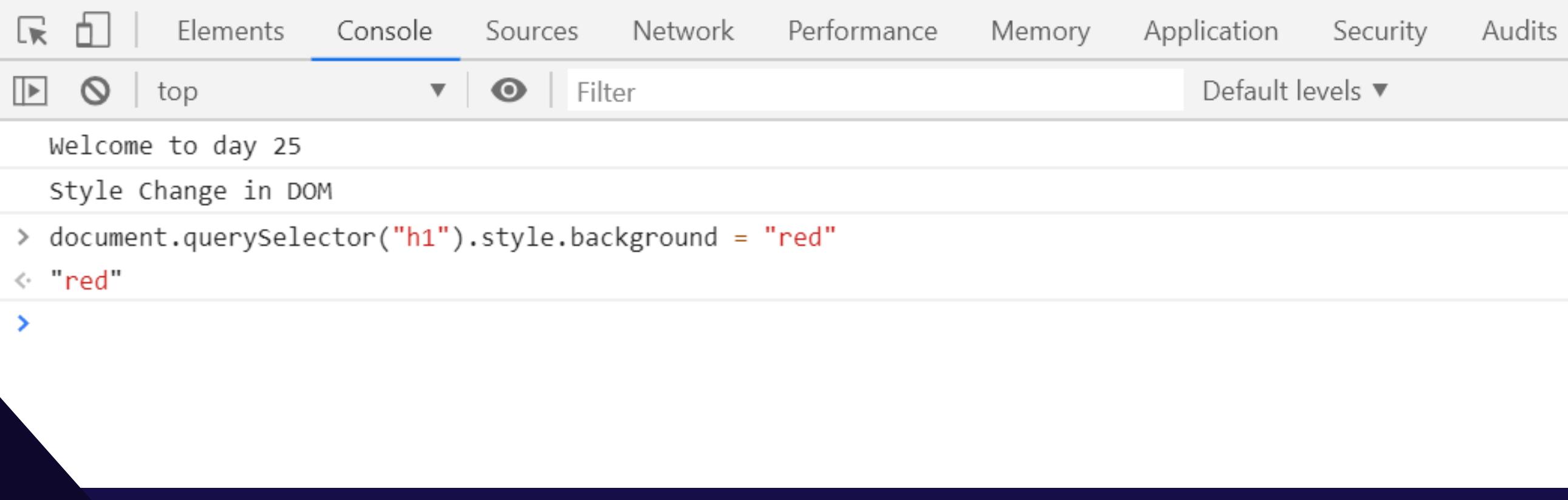
Now change the **background** of **h1**.
Using
``document.querySelector("h1").style.background = "red"'`

See result -

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The page content is displayed above the tools, featuring a large red header with white text. The DevTools console shows the following:

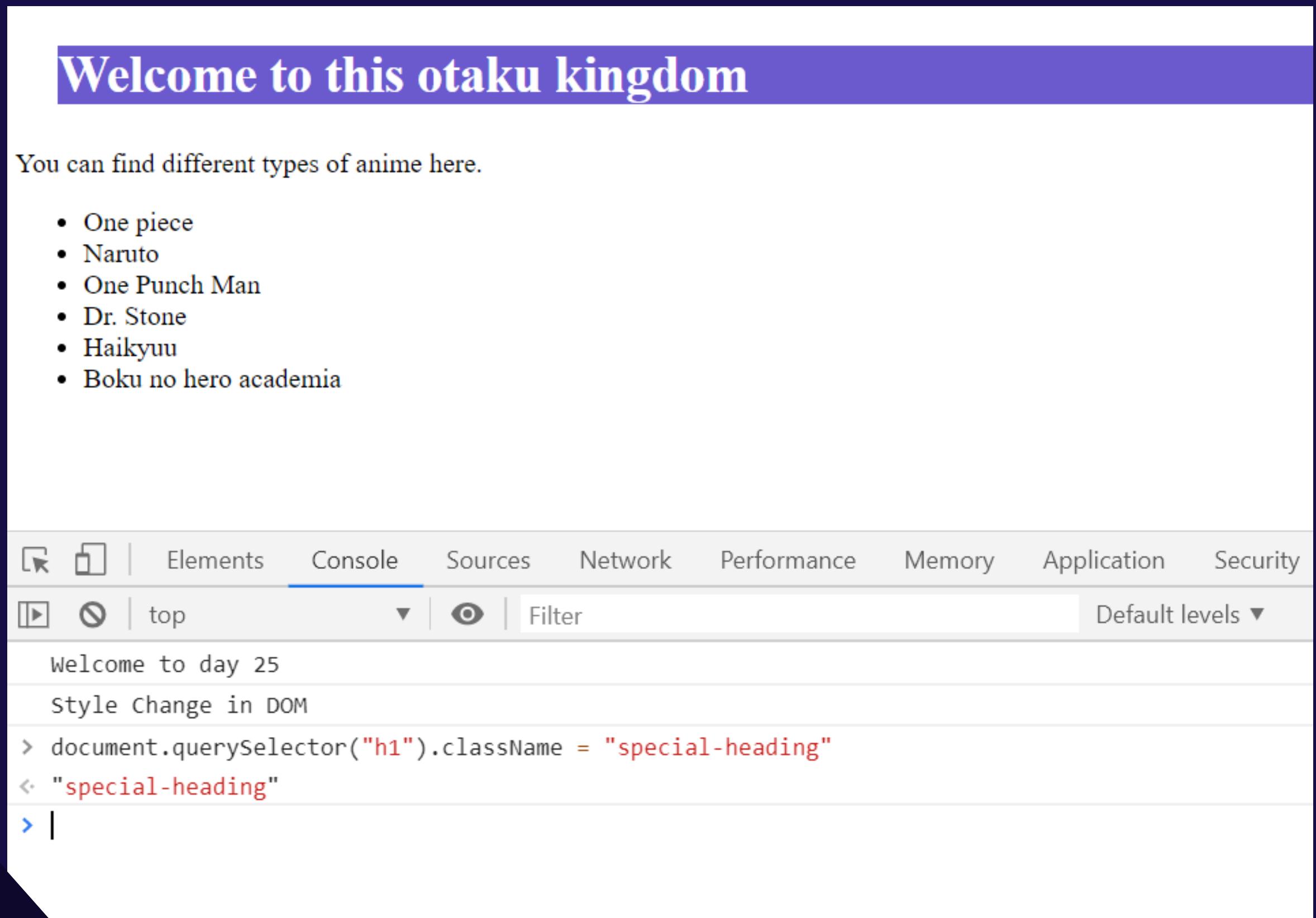
```
>Welcome to day 25
style Change in DOM
> document.querySelector("h1").style.background = "red"
< "red"
```



Now lets try to add a class in
h1 tag.
Let's write some styles in **style.css**



Using `'document.querySelector("h1"). className = "special-heading"'` we can change right?



The screenshot shows a browser's developer tools open to the Console tab. The page content above the console is a purple header with white text that reads "Welcome to this otaku kingdom". Below the header, there is a list of anime titles: One piece, Naruto, One Punch Man, Dr. Stone, Haikyuu, and Boku no hero academia. The browser's address bar shows the URL "http://localhost:3001". The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, and Security. The Console tab is active, displaying the command: `> document.querySelector("h1").className = "special-heading"`. The output of this command is shown below the prompt: `< "special-heading"`. A large white arrow points from the bottom right corner of the slide towards the bottom right corner of the screenshot.



nerdjfpb



nerdjfpb

What are you going to do
after this?

You have now superpower in your
hand to **change anything in Dom!**



nerd_jfpb





nerdjfpb



nerdjfpb

Let's learn **something amazing**
in next one!



nerd_jfpb

JAVASCRIPT SERIES-26

PLAY WITH DOM (QUERYSELECTOR)





nerdjfpb



nerdjfpb

In last tutorial we worked with the dom for **change the styles**, now we are going to work with the dom for **changing some html**.



nerd_jfpb



To change html before we use write.

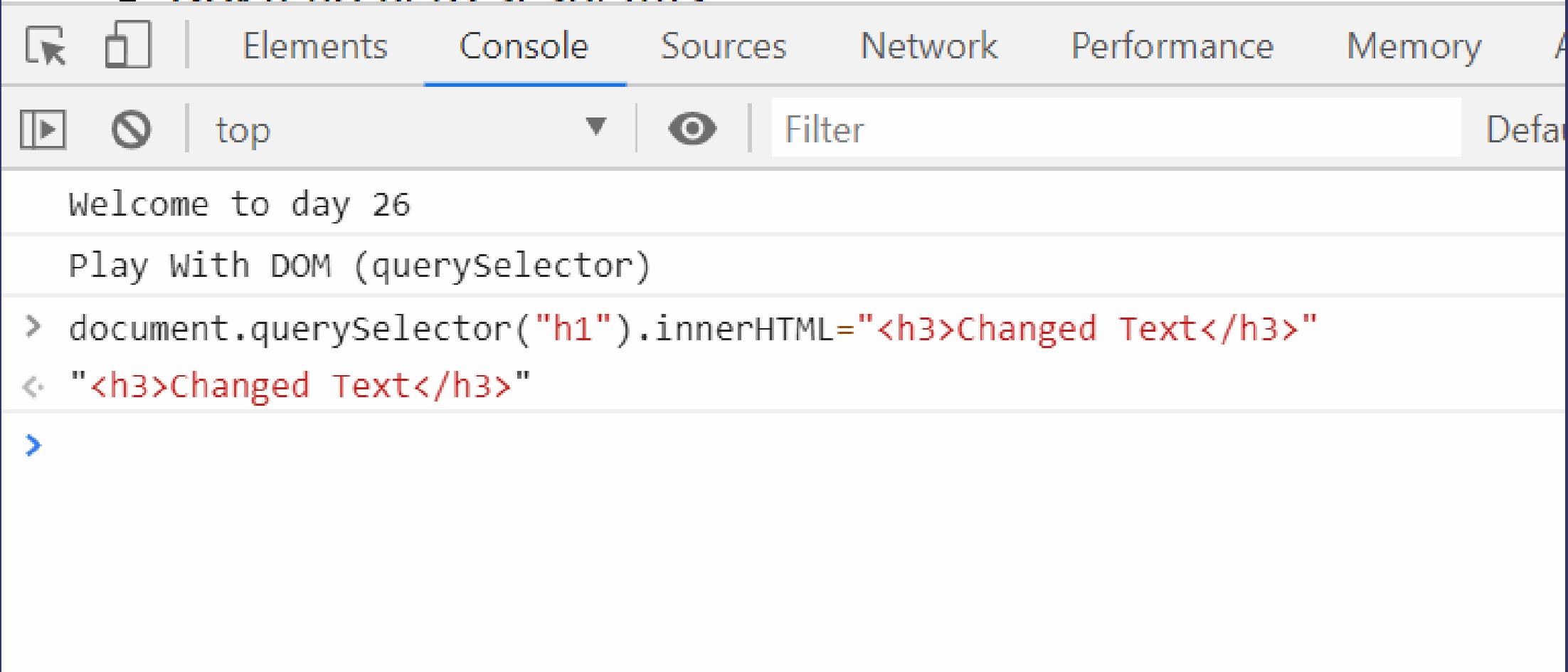
Now we are going to use
`innerHTML` which is going to
change the whole text with tag.

Example -

Changed Text

You can find different types of anime here.

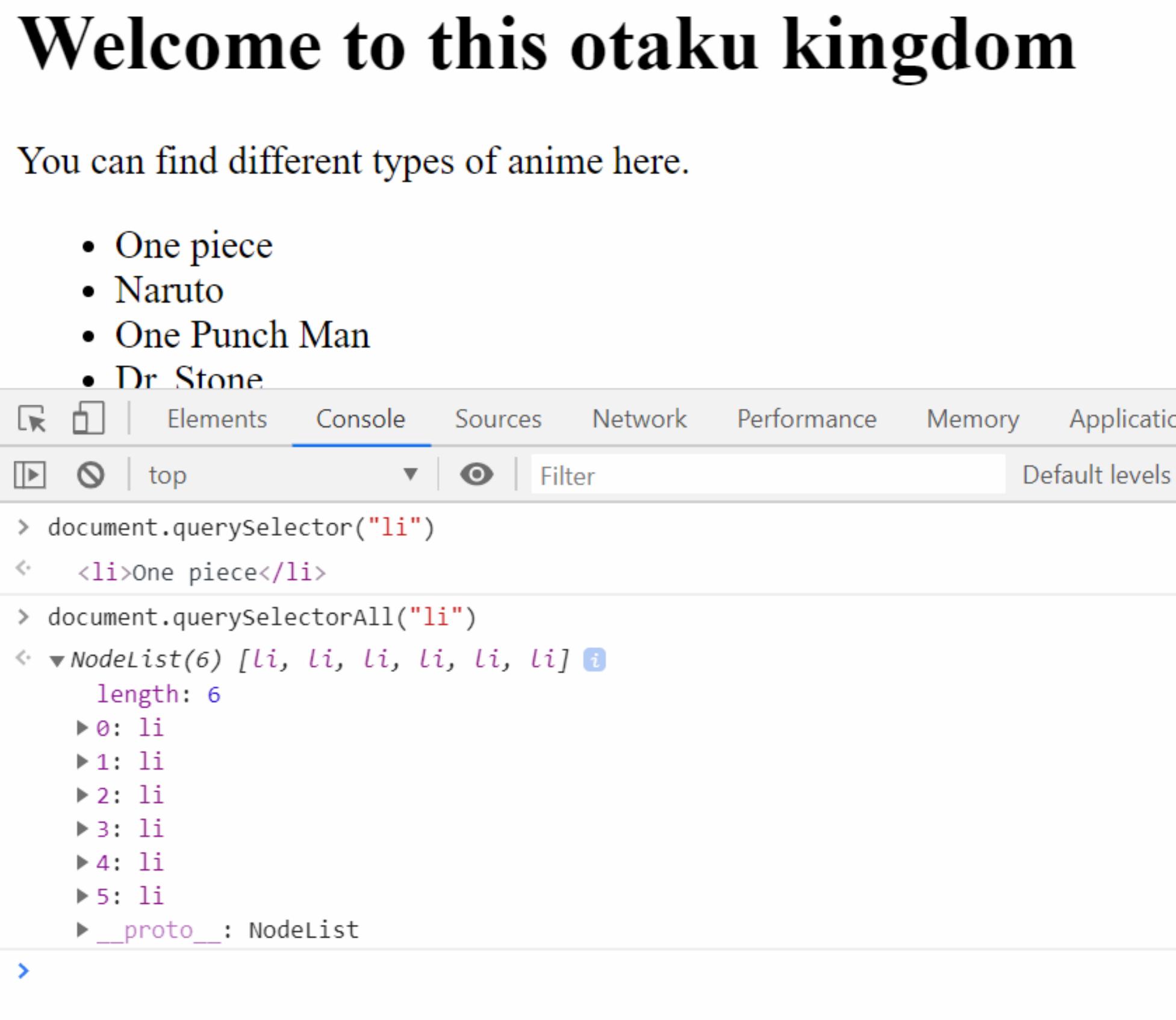
- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- ~~Roku no hero academia~~



The screenshot shows a browser's developer tools open to the 'Console' tab. The console output displays the following code and its execution results:

```
Welcome to day 26
Play With DOM (querySelector)
> document.querySelector("h1").innerHTML = "<h3>Changed Text</h3>"
< " <h3>Changed Text</h3> "
>
```

Up until now we've just used `querySelector`. If we use this on `li` then it will just give us first list of the item, so we need to call All of it by `querySelectorAll`



The screenshot shows a web browser's developer tools with the "Console" tab selected. The code in the console is as follows:

```
> document.querySelector("li")
<  <li>One piece</li>
> document.querySelectorAll("li")
<  ▾ NodeList(6) [li, li, li, li, li, li] ⓘ
  length: 6
  ▷ 0: li
  ▷ 1: li
  ▷ 2: li
  ▷ 3: li
  ▷ 4: li
  ▷ 5: li
  ▷ __proto__: NodeList
>
```

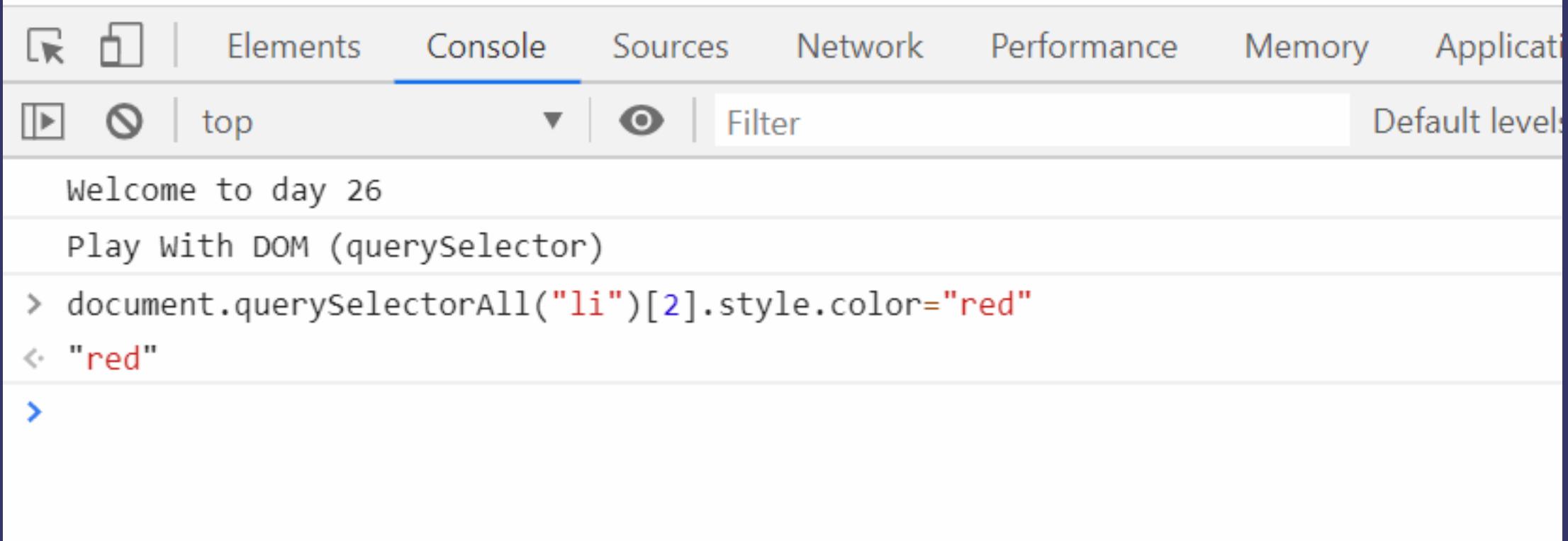
The output shows that `querySelector` returns the first `li` element, while `querySelectorAll` returns a `NodeList` containing all six `li` elements.

Can you remember the array? Using **index** now we can select an element from '*i*' and can change it.

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- **One Punch Man**
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows the Chrome DevTools console tab. The URL bar shows 'top'. The console output is as follows:

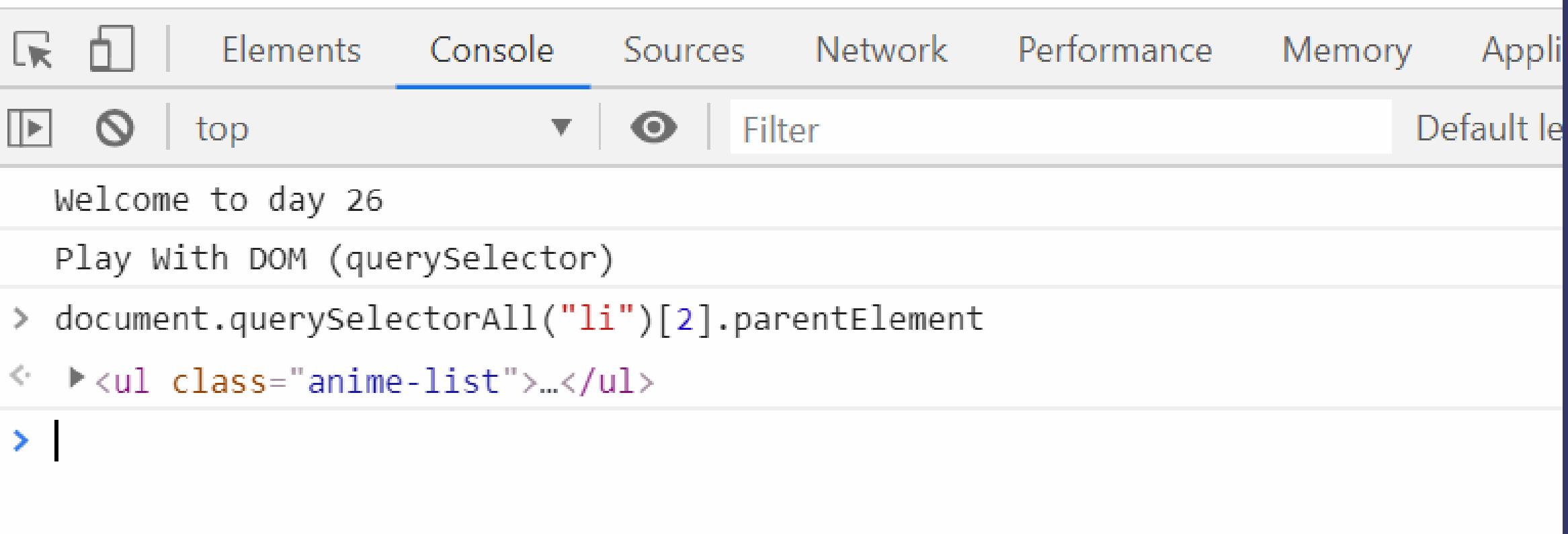
```
Welcome to day 26
Play With DOM (querySelector)
> document.querySelectorAll("li")[2].style.color="red"
< "red"
>
```

Now we can get the parent from a `li` item by just using `parentElement`

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



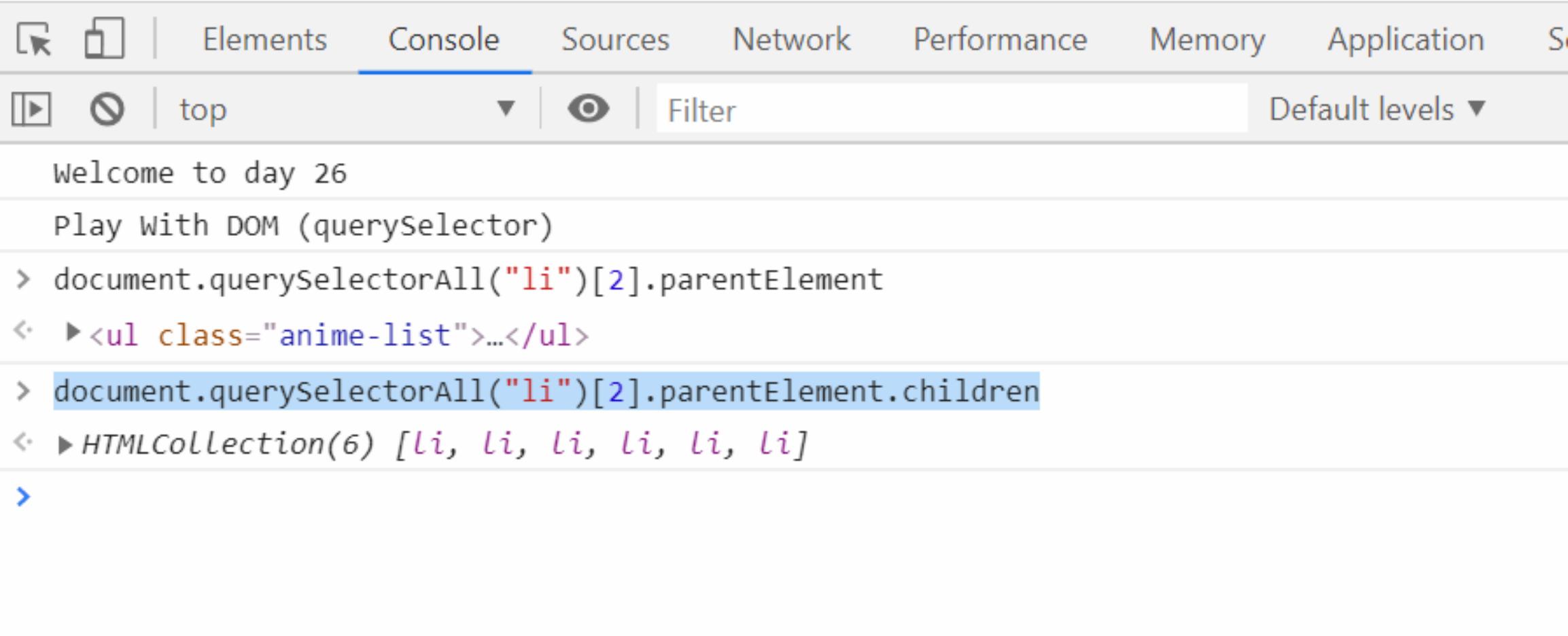
```
Elements Console Sources Network Performance Memory Application  
▶ top ▾ Filter Default le  
Welcome to day 26  
Play With DOM (querySelector)  
> document.querySelectorAll("li")[2].parentElement  
< ▶ <ul class="anime-list">...</ul>  
> |
```

Like in this way we can use 'children' too

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows a browser's developer tools open to the 'Console' tab. The interface includes navigation icons, tabs for Elements, Console, Sources, Network, Performance, Memory, Application, and Storage, and a filter bar. Below the tabs, there are buttons for play/pause, stop, and a dropdown set to 'top'. The main area displays a command history:

```
Welcome to day 26
Play With DOM (querySelector)
> document.querySelectorAll("li")[2].parentElement
< ▶<ul class="anime-list">...</ul>
> document.querySelectorAll("li")[2].parentElement.children
< ▶HTMLCollection(6) [li, li, li, li, li, li]
>
```

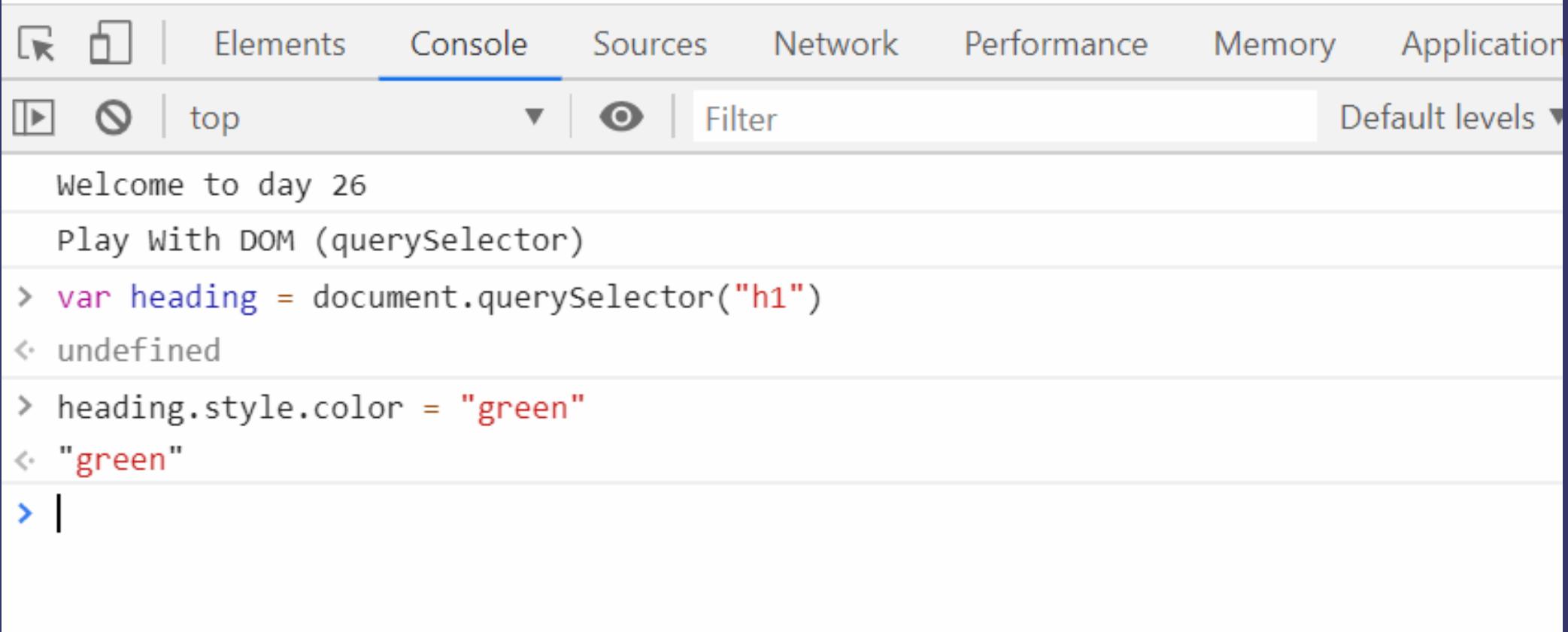


Finally we can use **variable** to store the **selector**, so that it will be easy to write something.

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows a browser's developer tools open to the 'Console' tab. The tabs at the top are Elements, Console, Sources, Network, Performance, Memory, and Application. The 'Console' tab is active, indicated by a blue underline. Below the tabs is a toolbar with icons for play/pause, stop, and filter, followed by 'Default levels'. The main area displays the following text and code:

```
Welcome to day 26
Play With DOM (querySelector)
> var heading = document.querySelector("h1")
< undefined
> heading.style.color = "green"
< "green"
> |
```





nerdjfpb



nerdjfpb

**Can you play with the dom now?
If not follow & DM me part you
didn't understand about it!**



nerd_jfpb



nerdJfpb



nerdJfpb

JAVASCRIPT SERIES-27

DOM EVENT INTRODUCTION



nerd_jfpb



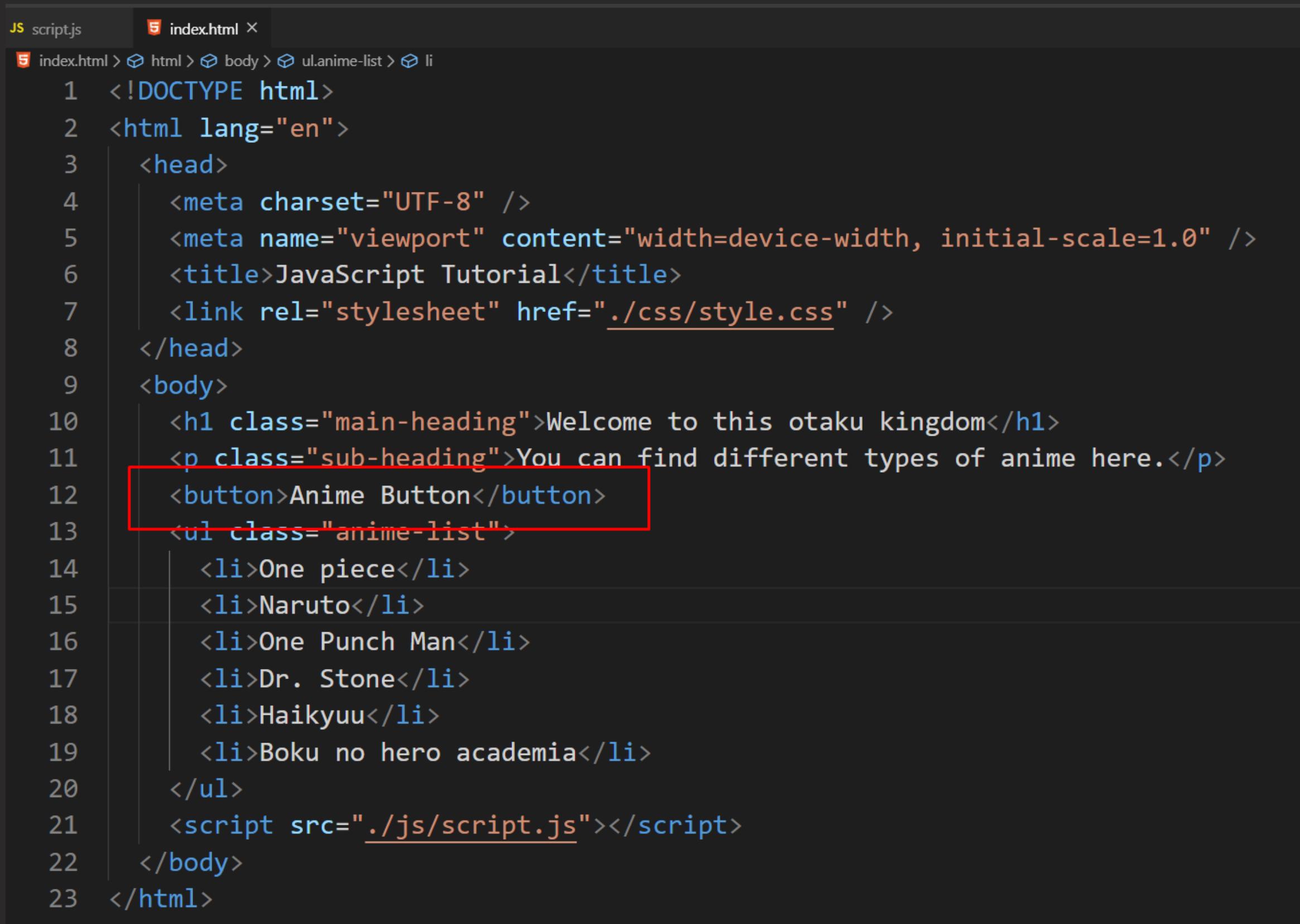


Events are things like -
clicking, hovering on
something or typing
anything on search.

There can be many others
event in a website.
We can do something
based on this event using
javascript.



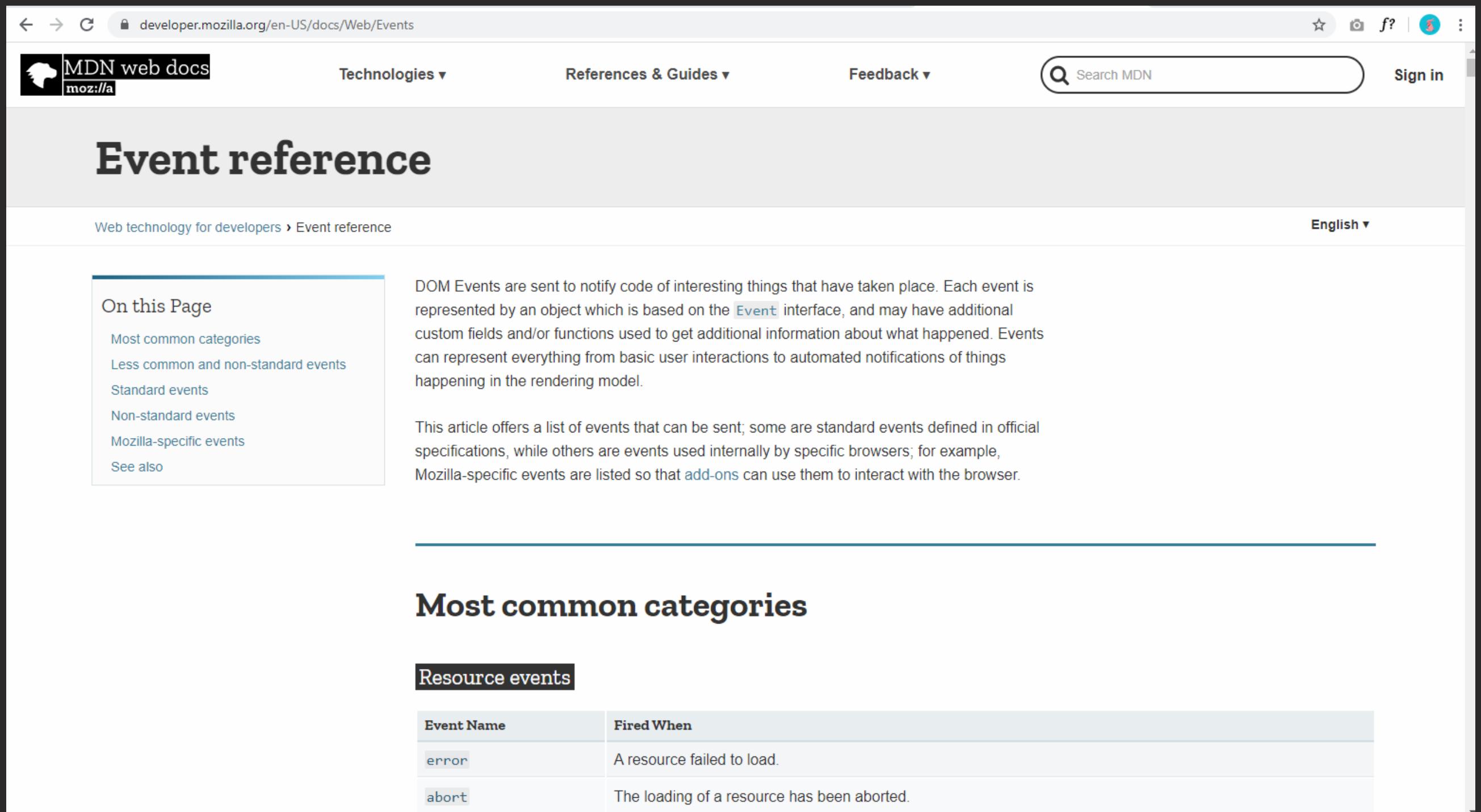
We'll start playing just by **using a button**. Let's **add a button** to the **html**. You can find the **code** in **github**.



```
JS script.js      S index.html ×
S index.html > ⏺ html > ⏺ body > ⏺ ul.anime-list > ⏺ li
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8" />
5    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6    <title>JavaScript Tutorial</title>
7    <link rel="stylesheet" href="./css/style.css" />
8  </head>
9  <body>
10   <h1 class="main-heading">Welcome to this otaku kingdom</h1>
11   <p class="sub-heading">You can find different types of anime here.</p>
12   <button>Anime Button</button>
13   <ul class="anime-list">
14     <li>One piece</li>
15     <li>Naruto</li>
16     <li>One Punch Man</li>
17     <li>Dr. Stone</li>
18     <li>Haikyuu</li>
19     <li>Boku no hero academia</li>
20   </ul>
21   <script src="./js/script.js"></script>
22 </body>
23 </html>
```



You can **read** more about events in mdn web docs.



The screenshot shows the MDN Web Docs page for 'Event reference'. The page has a dark header with the MDN logo and navigation links for Technologies, References & Guides, Feedback, and a search bar. The main content area has a light background. On the left, there's a sidebar titled 'On this Page' with links to 'Most common categories', 'Less common and non-standard events', 'Standard events', 'Non-standard events', 'Mozilla-specific events', and 'See also'. The main content area starts with a paragraph about DOM Events and then lists 'Most common categories' under 'Resource events'. A table provides details for two events: 'error' (fired when a resource fails to load) and 'abort' (fired when loading is aborted).

DOM Events are sent to notify code of interesting things that have taken place. Each event is represented by an object which is based on the [Event](#) interface, and may have additional custom fields and/or functions used to get additional information about what happened. Events can represent everything from basic user interactions to automated notifications of things happening in the rendering model.

This article offers a list of events that can be sent; some are standard events defined in official specifications, while others are events used internally by specific browsers; for example, Mozilla-specific events are listed so that add-ons can use them to interact with the browser.

Most common categories

Resource events

| Event Name | Fired When |
|-----------------------|---------------------------------------------|
| error | A resource failed to load. |
| abort | The loading of a resource has been aborted. |

We are going to use a **mouse event** for today's example.

Mouse events

| Event Name | Fired When |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>auxclick</code> | A pointing device button (ANY non-primary button) has been pressed and released on an element. |
| <code>click</code> | A pointing device button (ANY button; soon to be primary button only) has been pressed and released on an element. |
| <code>contextmenu</code> | The right button of the mouse is clicked (before the context menu is displayed). |
| <code>dblclick</code> | A pointing device button is clicked twice on an element. |
| <code>mousedown</code> | A pointing device button is pressed on an element. |
| <code>mouseenter</code> | A pointing device is moved onto the element that has the listener attached. |
| <code>mouseleave</code> | A pointing device is moved off the element that has the listener attached. |
| <code>mousemove</code> | A pointing device is moved over an element. (Fired continuously as the mouse moves.) |
| <code>mouseover</code> | A pointing device is moved onto the element that has the listener attached or onto one of its children. |
| <code>mouseout</code> | A pointing device is moved off the element that has the listener attached or off one of its children. |
| <code>mouseup</code> | A pointing device button is released over an element. |
| <code>pointerlockchange</code> | The pointer was locked or released. |
| <code>pointerlockerror</code> | It was impossible to lock the pointer for technical reasons or because the permission was denied. |
| <code>select</code> | Some text is being selected. |
| <code>wheel</code> | A wheel button of a pointing device is rotated in any direction. |





nerdjfpb



nerdjfpb

First we'll get the button
using
`'document.querySelector('button')'`



nerd_jfpb





Second step will be add a
EventListener.

Which is going to run the event.
addEventListener has
2 parameters.

One is the **event name**,
another one is **function** where
we are going to write
what happens if the event occurs.



Let's write some code now

```
JS script.js    X  5 index.html
js > JS script.js > ...
1 console.log('Welcome to day 27')
2 console.log('DOM EVENT INTRODUCTION')
3
4 var button = document.querySelector('button')
5
6 button.addEventListener('click', function() {
7   console.log('Anime Button is Clicked')
8 })
9
```

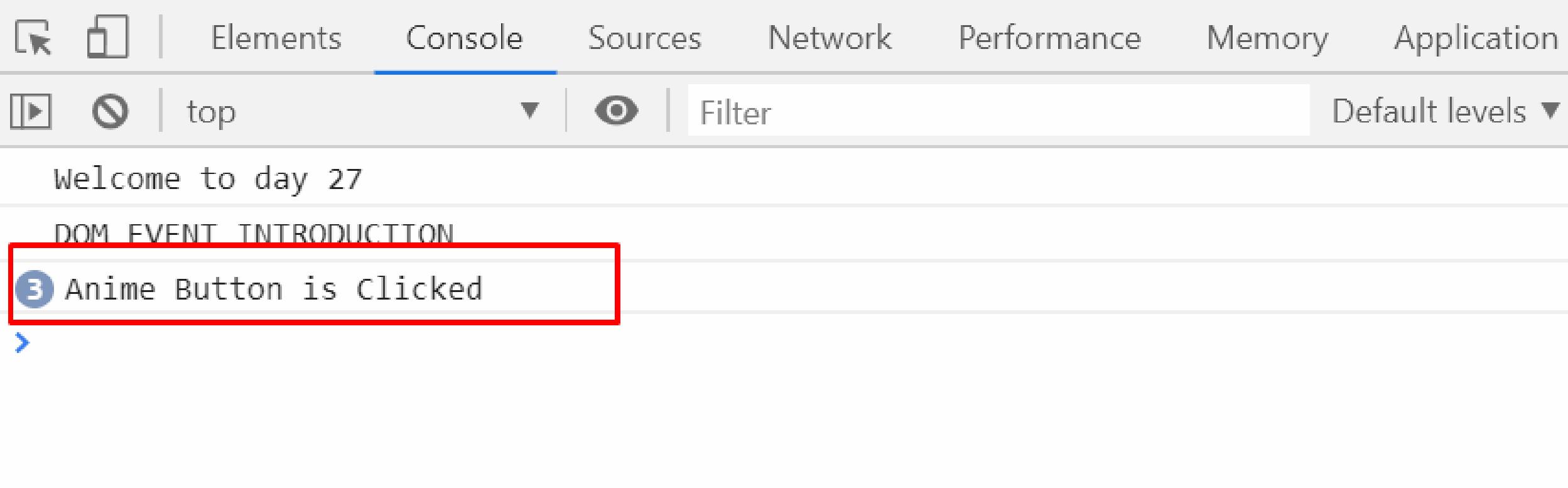
See the result

Welcome to this otaku kingdom

You can find different types of anime here.

Anime Button

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
>Welcome to day 27
DOM EVENT INTRODUCTION
3 Anime Button is Clicked
```

A red box highlights the third line of the log, which is the event message. The rest of the interface includes the Elements, Sources, Network, Performance, Memory, and Application tabs, along with a filter bar and a dropdown menu.



nerdjfpb



nerdjfpb

**Turn on the notification
so that you can stay with me
when I publish
the next part of this!**



nerd_jfpb



nerdJfpb



nerdJfpb

JAVASCRIPT SERIES-28

**ADD
NEW ANIME
ON THE LIST
USING EVENT**



nerd_jfpb



For this we need to have
a **html** like -

Welcome to this otaku kingdom

You can find different types of anime here.

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia



Writing html codes You can find the codes in github.

```
/   <link rel="stylesheet" href="./css/style.css" />
8   </head>
9   <body>
10  <h1 class="main-heading">Welcome to this otaku kingdom</h1>
11  <p class="sub-heading">You can find different types of anime here.</p>
12  <input id="entered-anime" type="text" placeholder="Enter anime name" />
13  <button id="add-anime">Anime Button</button>
14  <ul id="anime-list">
15    <li>...
16  </ul>
```



Our first step will be selecting the ul, input & button and store it into variable.

```
var ul = document.getElementById('anime-list')
var animeName = document.getElementById('entered-anime')
var addAnime = document.getElementById('add-anime')
```



Now we are going to add a event on the button.

```
7
8 ↴addAnime.addEventListener('click', function() {
9
10  })
11
```



We need to create an
`li item` for adding the new
anime.

```
addAnime.addEventListener('click', function() {  
    var li = document.createElement('li')  
})
```



We need to add the
`animeName` value inside
of that `li`

```
addAnime.addEventListener('click', function() {  
    var li = document.createElement('li')  
    li.appendChild(document.createTextNode(animeName.value))  
})
```



Now we just need to add this in our list `ul`

```
addAnime.addEventListener('click', function() {  
  var li = document.createElement('li')  
  li.appendChild(document.createTextNode(animeName.value))  
  ul.appendChild(li)  
})
```



nerdjfpb



nerdjfpb

See the result in browser

Welcome to this otaku kingdom

You can find different types of anime here.

[Shokugeki no soma](#) [Anime Button](#)

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia
- Shokugeki no soma

The screenshot shows the Chrome DevTools Console tab. The output in the console is:

```
Welcome to day 28
Add New Anime On The List Using Event
```



nerd_jfpb





nerdjfpb



nerdjfpb

**FOLLOW
FOR
SEE
POSTS
LIKE
THIS!**



nerd_jfpb



nerdJfpb



nerdJfpb

JAVASCRIPT SERIES-29

REFACTOR OUR LAST TUTORIAL CODE



nerd_jfpb



In our last code there is a easy bug, if you just add an empty anime name then it inserted in the list.

The screenshot shows a web application interface. At the top, a large dark blue header contains the text "Welcome to this otaku kingdom". Below this, a sub-header says "You can find different types of anime here." There is an input field labeled "Enter anime name" and a button labeled "Anime Button". A list of anime names follows:

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia
-

A red rectangular box highlights the last item in the list, "Boku no hero academia". Below the list is a screenshot of a browser's developer tools. The "Console" tab is selected, showing the following log entries:

```
Welcome to day 29
Refactor our last tutorial code
```

The browser's address bar shows "Default levels ▾".

To solve this we just need to add a logic that value of string length should be greater than 1 or as much character you want.

```
addAnime.addEventListener('click', function() {
  if (animeName.value.length > 0) {
    var li = document.createElement('li')
    li.appendChild(document.createTextNode(animeName.value))
    ul.appendChild(li)
  }
})
```



nerdjfpb



nerdjfpb

This code is kinda messy right ?
Let's make it better together.



nerd_jfpb



We'll start take the anonymous out for the addEventListener.

```
7
8 function addAnimeInList() {
9     if (animeName.value.length > 0) {
10         var li = document.createElement('li')
11         li.appendChild(document.createTextNode(animeName.value))
12         ul.appendChild(li)
13     }
14 }
15
16 addAnime.addEventListener('click', addAnimeInList)
17
```

Now let's add a new function to get the input value.

```
8  function inputLengthCheck(input) {
9    return input.value.length
10 }
11
12 function addAnimeInList() {
13   if (inputLengthCheck(animeName) > 0) {
14     var li = document.createElement('li')
15     li.appendChild(document.createTextNode(animeName.value))
16     ul.appendChild(li)
17   }
18 }
19
```

We can break our code into a new function for just creating a list item.

```
11
12  function createListElement(item) {
13    var li = document.createElement('li')
14    li.appendChild(document.createTextNode(item.value))
15    ul.appendChild(li)
16  }
17
18  function addAnimeInList() {
19    if (inputLengthCheck(animeName) > 0) {
20      createListElement(animeName)
21    }
22  }
```

Now we have another bug,
did you noticed yet? When we
add a new anime list, the input
field doesn't clear after that.

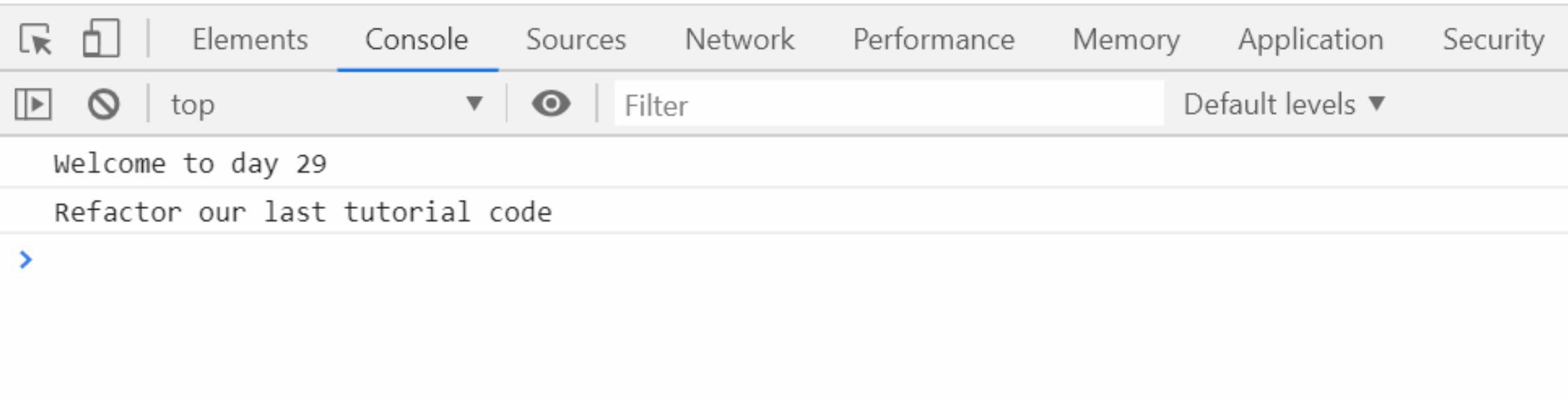
Welcome to this otaku kingdom

You can find different types of anime here.

Shokugeki No Soma

Anime Button

- One piece
- Naruto
- One Punch Man
- Dr. Stone
- Haikyuu
- Boku no hero academia
- Shokugeki No Soma



To solve this we just need to add a single line to clear the value.

```
11
12 function createListElement(item) {
13     var li = document.createElement('li')
14     li.appendChild(document.createTextNode(item.value))
15     ul.appendChild(li)
16     item.value = ''
17 }
18
```



nerdjfpb



nerdjfpb

**DO YOU LIKED THE TUTORIAL ?
TURN ON THE POST
NOTIFICATION TO FIND
WHAT IS COMING NEXT!**



nerd_jfpb



nerdJfpb



nerdJfpb

JAVASCRIPT SERIES-30

JQUERY?



nerd_jfpb





nerdJfpb



nerdJfpb

jQuery is a fast, small, and feature-rich JavaScript library. This helps to write js codes easily. This saves a ton of time.

The screenshot shows the official jQuery website at jquery.com. The header features the jQuery logo and navigation links for Download, API Documentation, Blog, Plugins, and Browser Support. A search bar is also present. The main content area highlights three key features: "Lightweight Footprint" (only 30kB minified and gzipped), "CSS3 Compliant" (supports CSS3 selectors), and "Cross-Browser" (works across Chrome, Edge, Firefox, IE, Safari, Android, iOS, and more). A prominent orange button allows users to "Download jQuery v3.4.1". Below these features, a section titled "What is jQuery?" provides a brief description of the library's purpose and benefits. Another section, "Other Related Projects", lists several jQuery spin-offs: jQuery UI, jQuery Mobile, QUnit, and Sizzle. The footer contains a "Resources" section with links to the Core API Documentation, Learning Center, Blog, Contribute page, About the Foundation, and Bug tracking.

Your donations help fund the continued development and growth of **jQuery**.

SUPPORT THE PROJECT

Download jQuery v3.4.1

The 1.x and 2.x branches no longer receive patches.

View Source on GitHub →

How jQuery Works →

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Other Related Projects

jQuery user interface **jQuery** mobile

QUnit JS unit testing **Sizzle** CSS selector engine

A Brief Look

Resources

- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)
- [About the jQuery Foundation](#)
- [Browse or Submit jQuery Bugs](#)



nerd_jfpb





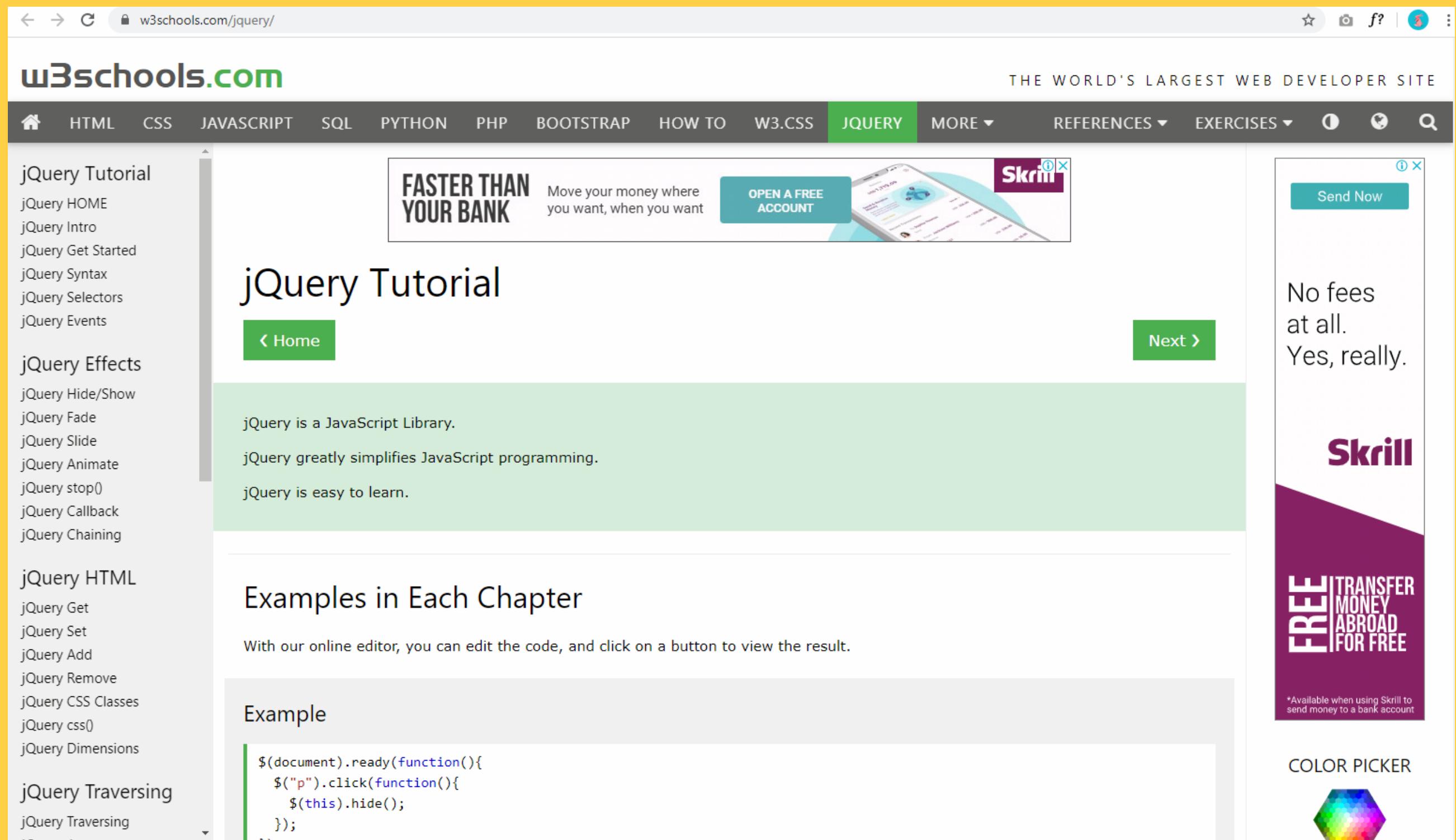
But people don't use it on new projects because there are more better library & you can do the same thing using **vanila js**. Check this out - <http://youmightnotneedjquery.com/>

The screenshot shows three sections of the You Might Not Need jQuery website:

- Add Class**:
 - JQUERY: `$(el).addClass(className);`
 - IE10+: `el.classList.add(className);`
- After**:
 - JQUERY: `$(target).after(element);`
 - IE8+: `target.insertAdjacentElement('afterend', element);`
- Append**:
 - JQUERY: `$(parent).append(el);`
 - IE8+: `parent.appendChild(el);`



But jQuery saves a ton of time before and helped to write better javascript. So there are many projects which use jQuery. To understand those project you should learn it.

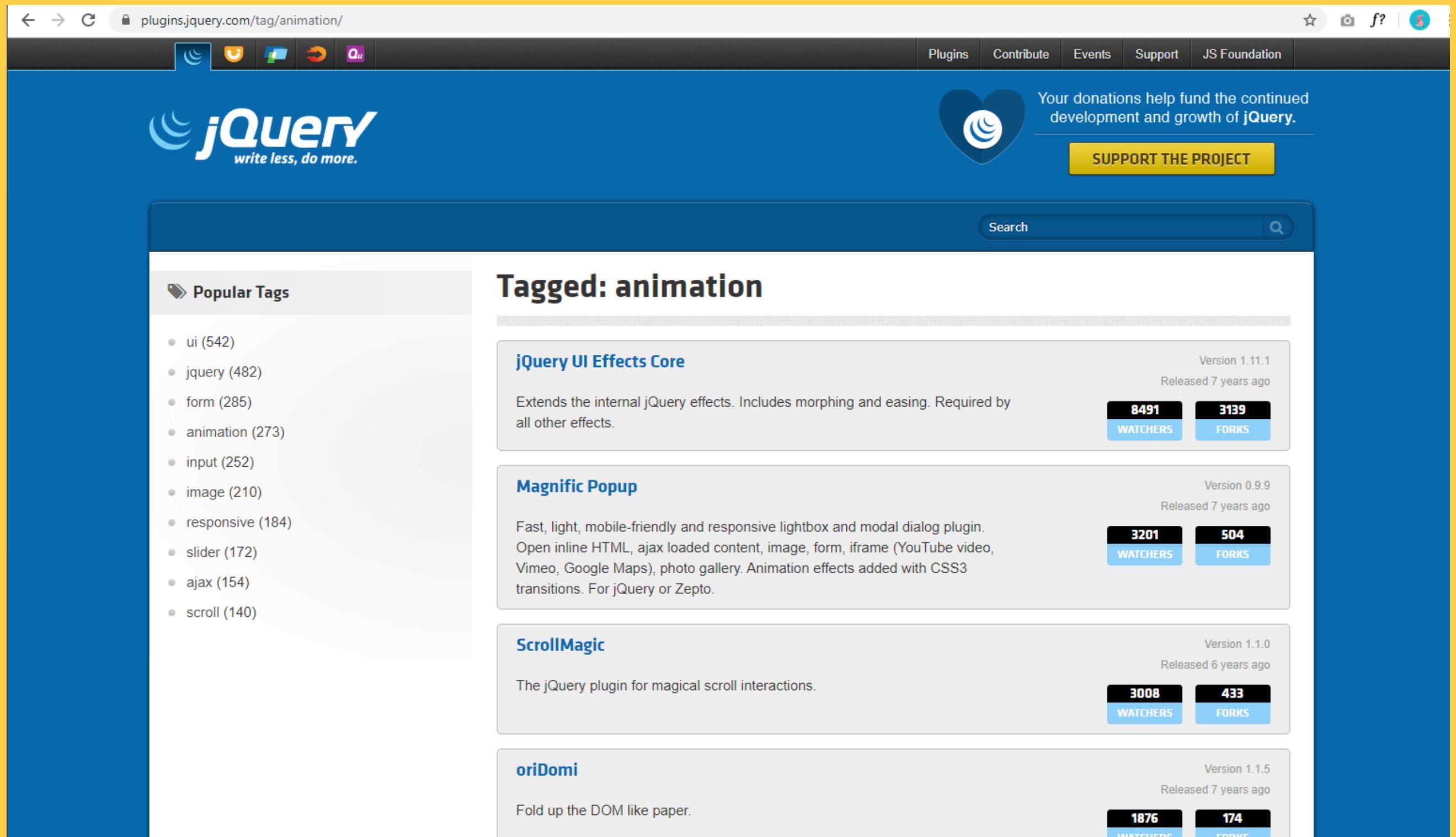


The screenshot shows the w3schools.com website with the URL "w3schools.com/jquery/" in the address bar. The page title is "jQuery Tutorial". The main content area starts with a green box containing three bullet points: "jQuery is a JavaScript Library.", "jQuery greatly simplifies JavaScript programming.", and "jQuery is easy to learn.". Below this is a section titled "Examples in Each Chapter" with a note: "With our online editor, you can edit the code, and click on a button to view the result." A code example is shown in a box:

```
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
```

On the left sidebar, there's a navigation menu for the "jQuery Tutorial" section, listing items like "jQuery HOME", "jQuery Intro", "jQuery Get Started", etc. On the right side, there's a Skrill advertisement with the text "FASTER THAN YOUR BANK", "Move your money where you want, when you want", "OPEN A FREE ACCOUNT", "Send Now", "No fees at all.", "Yes, really.", and "Skrill". There's also a "FREE TRANSFER MONEY ABROAD FOR FREE" offer and a "COLOR PICKER" icon.

jQuery has lots of plugins which can help you to do things easier.



The screenshot shows the jQuery plugin repository at plugins.jquery.com/tag/animation/. The page displays a list of plugins tagged with 'animation'. The top navigation bar includes links for Plugins, Contribute, Events, Support, and JS Foundation. A sidebar on the left lists popular tags such as ui, jquery, form, animation, input, image, responsive, slider, ajax, and scroll. The main content area features four plugin cards:

- jQuery UI Effects Core**: Version 1.11.1, Released 7 years ago. Description: Extends the internal jQuery effects. Includes morphing and easing. Required by all other effects. Watchers: 8491, Forks: 3139.
- Magnific Popup**: Version 0.9.9, Released 7 years ago. Description: Fast, light, mobile-friendly and responsive lightbox and modal dialog plugin. Open inline HTML, ajax loaded content, image, form, iframe (YouTube video, Vimeo, Google Maps), photo gallery. Animation effects added with CSS3 transitions. For jQuery or Zepto. Watchers: 3201, Forks: 504.
- ScrollMagic**: Version 1.1.0, Released 6 years ago. Description: The jQuery plugin for magical scroll interactions. Watchers: 3008, Forks: 433.
- oriDomi**: Version 1.1.5, Released 7 years ago. Description: Fold up the DOM like paper. Watchers: 1876, Forks: 174.





Bootstrap still using the jQuery.

B Home Documentation Examples Icons Themes Expo Blog v4.4 ▾ Q Twitter ⌂ C Download

Installation

Include Bootstrap's source Sass and JavaScript files via npm, Composer or Meteor. Package managed installs don't include documentation, but do include our build system and readme.

```
$ npm install bootstrap
```

```
$ gem install bootstrap -v 4.4.1
```

[Read installation docs](#)

BootstrapCDN

When you only need to include Bootstrap's compiled CSS or JS, you can use [BootstrapCDN](#).

CSS only

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8XAK4nJyVs8qWZlIrcqGZgVZGQXwGzg8zHnGZD0mzWcDZPf8eXqz+giC" crossorigin="anonymous">
```

JS, Popper.js, and jQuery

```
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6c3GU80PcYVc9v4/YV2oC4EaLmB6F8CqGZqCvD8jxGzGp0sHsQWVQmM" crossorigin="anonymous">
```

```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q2HkClZ0/krq+QKbmAuShiPc5k17VVja5H/qqTn8gkqgXyNz" crossorigin="anonymous">
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd9fdSphA" crossorigin="anonymous">
```

[Explore the docs](#)

Official Themes

Take Bootstrap 4 to the next level with official premium themes—toolkits built on Bootstrap with new components and plugins, docs, and build tools.



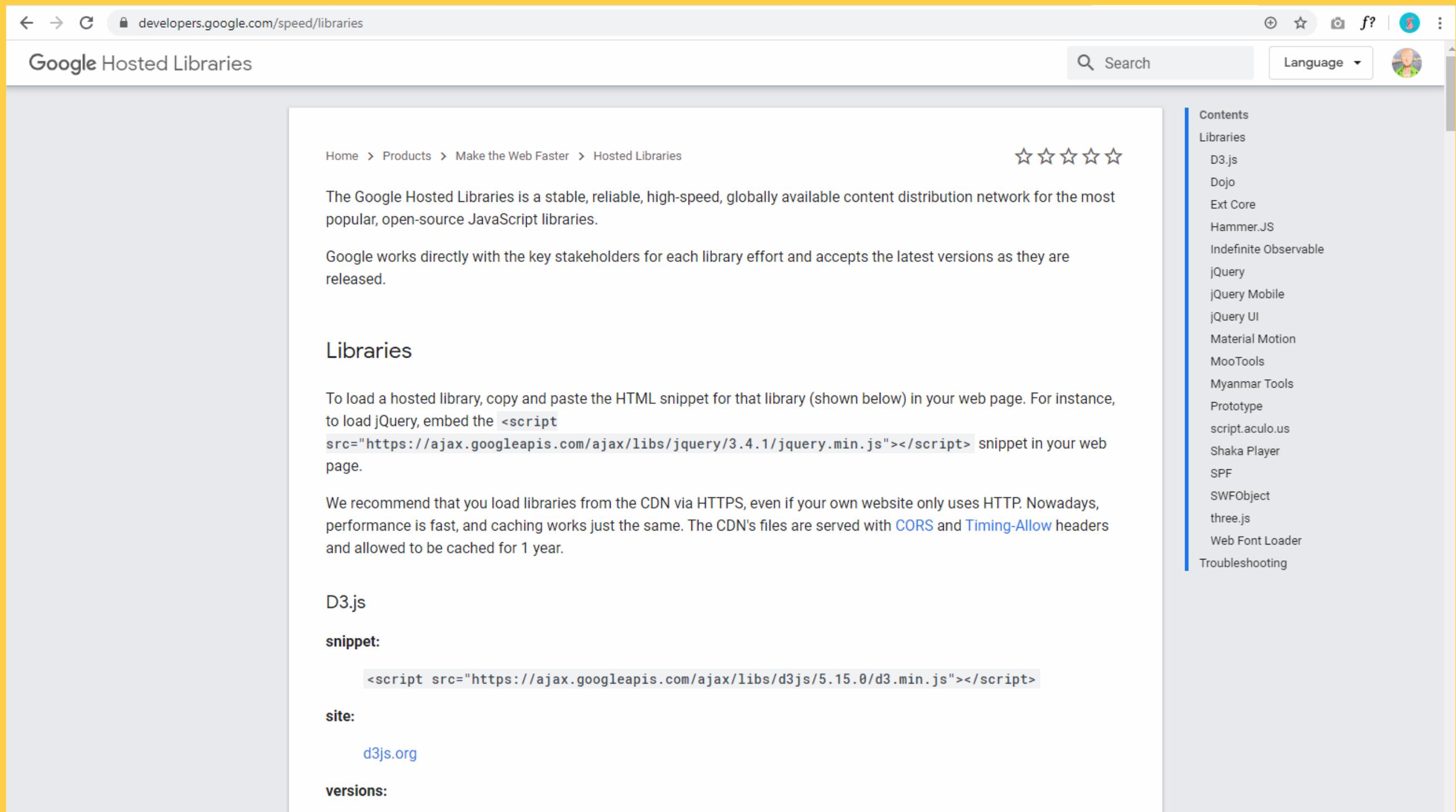
[Browse themes](#)

[GitHub](#) [Twitter](#) [Examples](#) [About](#)

Designed and built with all the love in the world by the Bootstrap team with the help of [our contributors](#).
Copyright © 2019, Bootstrap, Inc. All rights reserved. Code licensed MIT. [View license](#)



jQuery is really lightweight and it's also hosted in google so you can use cdn in your project very easily.



The screenshot shows a web browser displaying the Google Hosted Libraries page at developers.google.com/speed/libraries. The page has a header with the title "Google Hosted Libraries". On the right side, there is a sidebar titled "Contents" listing various JavaScript libraries. The "jQuery" entry is highlighted. The main content area shows information about the Google Hosted Libraries, including its purpose as a stable, reliable, high-speed, globally available content distribution network for popular open-source JavaScript libraries. It mentions that Google works directly with stakeholders and accepts latest versions. Below this, there is a section for the "jQuery" library, which includes a snippet of code to embed it into a web page:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

The sidebar on the right lists the following libraries:

- Contents
- Libraries
 - D3.js
 - Dojo
 - Ext Core
 - Hammer.JS
 - Indefinite Observable
 - jQuery
 - jQuery Mobile
 - jQuery UI
 - Material Motion
 - MooTools
 - Myanmar Tools
 - Prototype
 - script.aculo.us
 - Shaka Player
 - SPF
 - SWFObject
 - three.js
 - Web Font Loader
- Troubleshooting



nerdJfpb



nerdJfpb

Easy JavaScript for Designers.

The screenshot shows a web browser displaying an article from sitepoint.com. The header features a purple banner with the text "Get 3 months access to 400+ books and courses for \$3/m!" and a timer "23:56:37". Below the banner is a black navigation bar with the sitepoint logo, links for Blog, Community, Jobs, Library, Login, and a pink "Join Premium" button. The main content area has a yellow background. At the top left of the content area, there are social sharing icons for Facebook, Twitter, and Email. The main title "jQuery: Easy JavaScript for Designers" is centered in large white text. Below the title, a text block reads: "If the rebirth of JavaScript has been the biggest theme of the past two years, you could probably divide most of the talk surrounding this topic into two main areas." Further down, another text block says: "At the geekier end of town, we've seen smarties harnessing JavaScript to do all sorts of amazing – and occasionally ridiculous – things with Ajax." To the right of the text, there is a small image of a person in a blue jacket rowing a boat on a lake with mountains in the background. Next to the image is a blue box containing the text "Ready to build your website?". A pink circular icon with a white bell symbol is located at the bottom right of the content area.

Report Advertisement

JavaScript > February 08, 2008 > By Alex Walker

jQuery: Easy JavaScript for Designers

If the rebirth of JavaScript has been the biggest theme of the past two years, you could probably divide most of the talk surrounding this topic into two main areas.

At the geekier end of town, we've seen smarties harnessing JavaScript to do all sorts of amazing – and occasionally ridiculous – things with Ajax.

However for front-end guys like myself, much of the scripting fizz and bubble has been focussed around refitting your markup – that is, using JavaScript to make your markup work better after it gets to the browser. Long-time readers of the [Design View](#) newsletter will probably remember a few of my own experiments along these lines over the past few years:

Ready to build your website?



nerd_jfpb





nerdjfpb



nerdjfpb

It will be better if you start
with other js
library/frameworks instead
of jQuery
**(use react,angular & vue
logo image here)**



nerd_jfpb





nerdjfpb



nerdjfpb

**DO YOU WANT TO USE JQUERY
IN YOUR NEW PROJECT ?**



nerd_jfpb