Welcome Geeks

Introduction to JavaScript

- JavaScript is the world's most popular programming language.

- JavaScript can be used for Client-side developments as well as Server-side developments

https://survey.stackoverflow.co/2023/

Including JavaScript in an HTML Page

```
<script type="text/javascript">

//JS code goes here

</script>
```

Call an External JavaScript File

```
<script src="myscript.js"></script>
```

Including Comments

Single-line comments — To include a comment that is limited to a single line, precede it with //

Multi-line comments — In case you want to write longer comments between several lines, wrap it in /* and */ to avoid it from being executed

Console

DataTypes in JavaScript

Data Types: Every Variable has a data type that tells what kind of data is being stored in a variable. There are two types of data types in JavaScript namely Primitive data types and Non-primitive data types.

Primitive data types: The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

Non-primitive data types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

Primitive DataTypes

1. Number: Number data type in javascript can be used to hold decimal values as well as values without decimals.

```
<script>
    let x = 250;
    let y = 40.5;
    console.log("Value of x=" + x);
    console.log("Value of y=" + y);
</script>
```

Primitive DataTypes

2. String: The string data type in javascript represents a sequence of characters that are surrounded by single or double quotes.

```
<script>
    let str = 'Hello All';
    let str1 = "Welcome to my new house";
    console.log("Value of str=" + str);
    console.log("Value of str1=" + str1);
</script>
```

Primitive DataTypes

3. Undefined: The meaning of undefined is ‘value is not assigned’.

```
<script>
    console.log("Value of x=" + x);
</script>
```

Primitive DataTypes

4. Boolean: The boolean data type can accept only two values i.e. true and false.

```
<script>
    console.log("value of bool=" + bool);
</script>
```

Primitive DataTypes

5. Null: This data type can hold only one possible value that is null.

```
<script>
    let x = null;
    console.log("Value of x=" + x);
</script>
```

Non Primitive DataTypes

1. Object: Object in Javascript is an entity having properties and methods. Everything is an object in javascript.

How to create an object in javascript:

Using Constructor Function to define an object:

```
// Create an empty generic object
var obj = new Object();

// Create a user defined object
var mycar = new Car();
```

Using Literal notations to define an object:

```
// An empty object
var square = {};

// Here a and b are keys and
// 20 and 30 are values
var circle = {a: 20, b: 30};
```

Non Primitive DataTypes

2. Array: With the help of an array, we can store more than one element under a single name.

Ways to declare a single dimensional array:

```
// Call it with no arguments
var a = new Array();

// Call it with single numeric argument
var b = new Array(10);

// Explicitly specify two or
// more array elements
var d = new Array(1, 2, 3, "Hello");
```

Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

```
function functionName(Parameter1, Parameter2, ..)
{
    // Function body
}
```

JavaScript Strings

JavaScript strings are for storing and manipulating text.

.length
.slice
.substring
.substr

JavaScript Strings Methods

JavaScript Numbers

Numbers can be written with or without decimals.

JavaScript Number Methods

- toString() (numbers to strings)
- parseInt() (variables to numbers)

JavaScript Loops

Loops can execute a block of code a number of times.

```
for (let i = 0; i < 5; i++) {
  console.log("The number is " + i);
}
```

JavaScript Arrays

An array is a special variable, which can hold more than one value:

JavaScript Objects

Objects are variables too. But objects can contain many values.

JavaScript

https://www.learn-js.org/

JS Next Level

- synchronous and asynchronous JS
- call stack, event loop

Synchronous JS

- Synchronous JS - Function Execution and Call Stack

So what happens when you define a function and then invoke it?

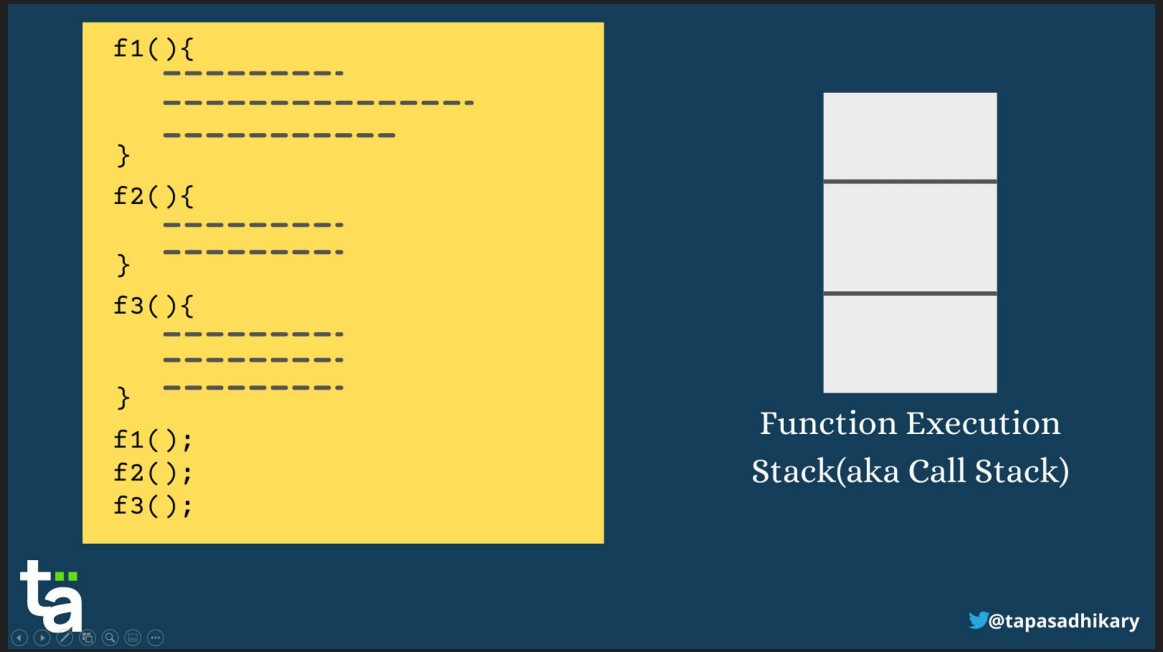
The JavaScript engine maintains a stack data structure called function execution stack. The purpose of the stack is to track the current function in execution.

Synchronous JS

- When the JavaScript engine invokes a function, it adds it to the stack, and the execution starts.
- If the currently executed function calls another function, the engine adds the second function to the stack and starts executing it.
- Once it finishes executing the second function, the engine takes it out from the stack.
- The control goes back to resume the execution of the first function from the point it left it last time.
- Once the execution of the first function is over, the engine takes it out of the stack.
- Continue the same way until there is nothing to put into the stack.

Synchronous JS

```
function f1() {
  // some code
}
function f2() {
  // some code
}
function f3() {
  // some code
}

// Invoke the functions
 one by one
f1();
f2();
f3();
```



Asynchronous JS

- Browser API/Web API events or functions. These include methods like setTimeout, or event handlers like click, mouse over, scroll, and many more.

- Promises. A unique JavaScript object that allows us to perform asynchronous operations.

Asynchronous JS

```
function printHello() {
  console.log('print hello');
}

setTimeout(printHello, 5000);
```

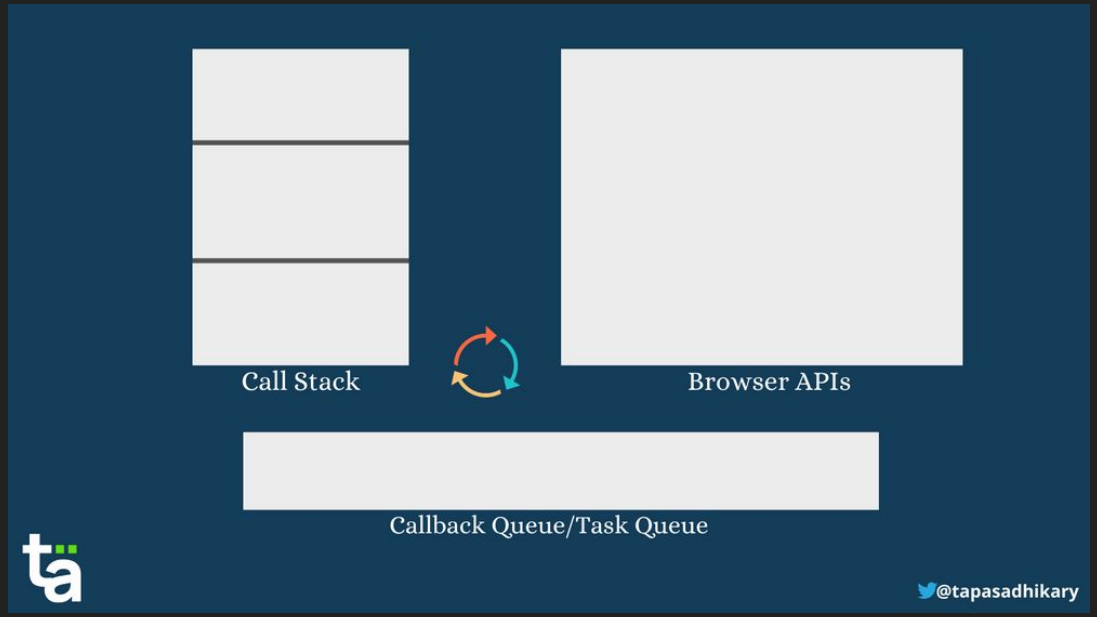
The setTimeout function executes a function after a certain amount of time has elapsed. In the code above, the text print me logs into the console after a delay of 5 seconds.

Asynchronous JS

```
function printHello() {
  console.log('print hello');
}

function test() {
  console.log('test');
}

setTimeout(printHello, 5000);
test();
```

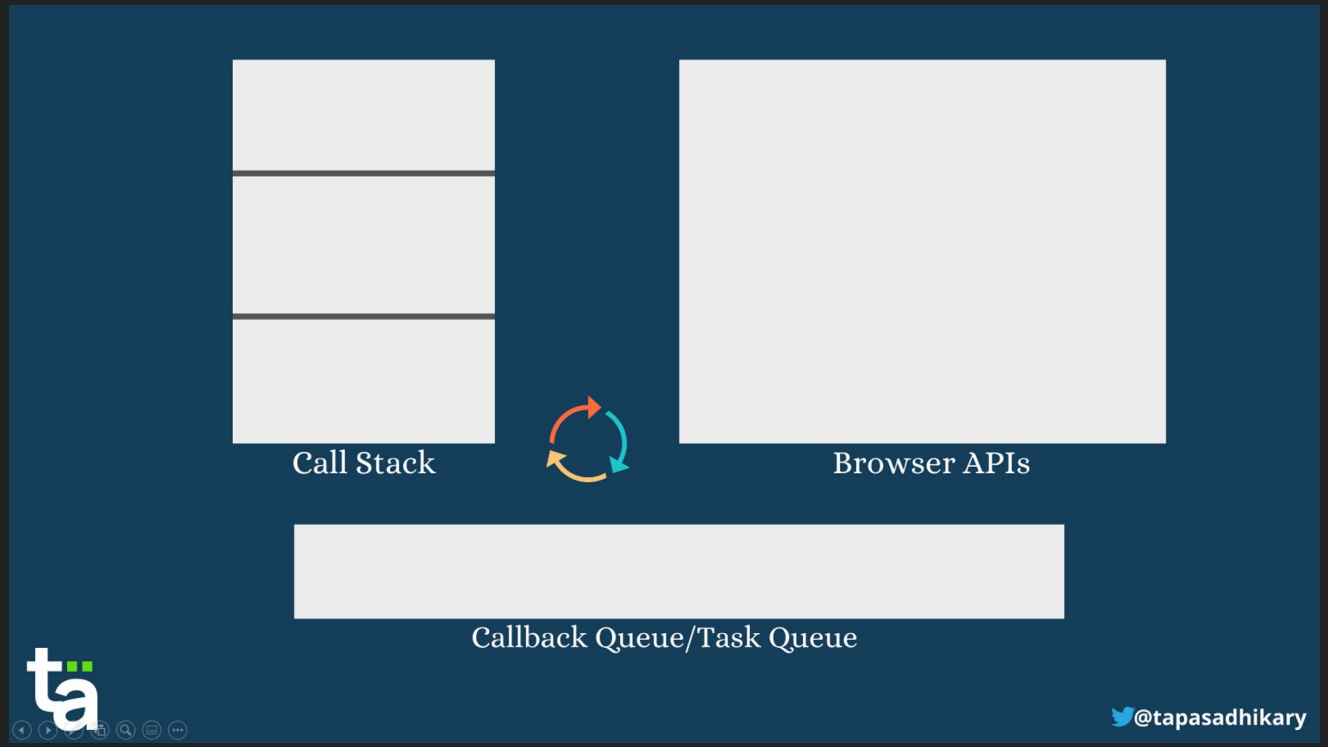
Asynchronous JS

Callback Queue comes into play here!



Asynchronous JS

```
function f1() {
    console.log('f1');
}

function f2() {
    console.log('f2');
}

function main() {
    console.log('main');

    setTimeout(f1, 0);

    f2();
}

main();
```



ES6

- let & const
- arrow functions
- promises

Arrow functions

Regular vs Arrow functions

- Syntactical difference
- No duplicate parameters
- Arguments binding

callback functions

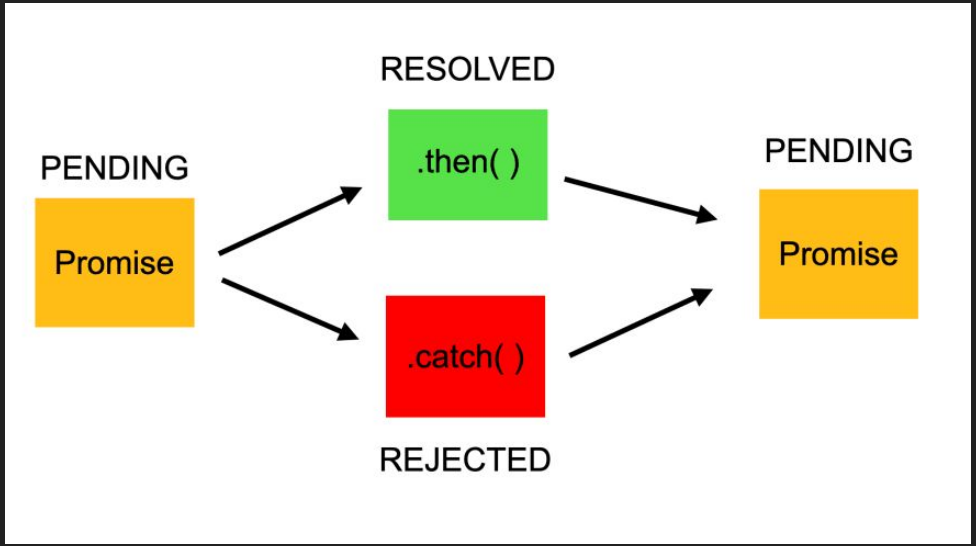
- A callback is a function passed as an argument to another function

```
const message = function() {
    console.log("This message is shown after 3 seconds");
}

setTimeout(message, 3000);
```

callback hell

promises

- Promises are the alternative to callbacks for delivering the results of asynchronous computation.



promises

```
const promise = new Promise((resolve, reject) => {
  // Async operation logic here....
  if (asyncOperationSuccess) {
    resolve(value); // async operation successful
  } else {
    reject(error);  // async operation error
  }
});
```

async await

Throttling & Debouncing

- Throttling will delay executing a function. It will reduce the notifications of an event that fires multiple times.

- Debouncing will bunch a series of sequential calls to a function into a single call to that function. It ensures that one notification is made for an event that fires multiple times.

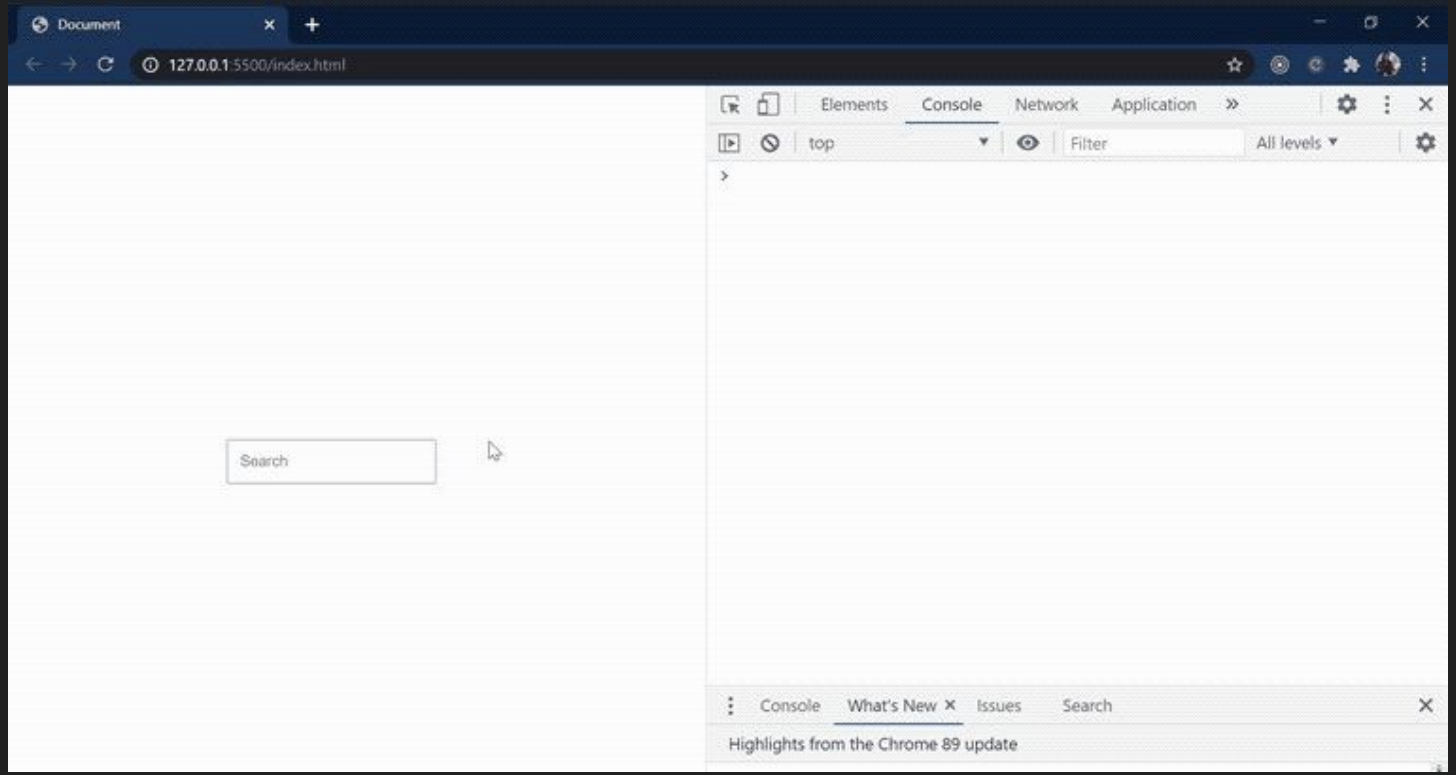
Example

Search bar - Don't want to search every time the user presses the key? Want to search when a user stopped typing for 1 sec. Use debounce 1 sec on key press.

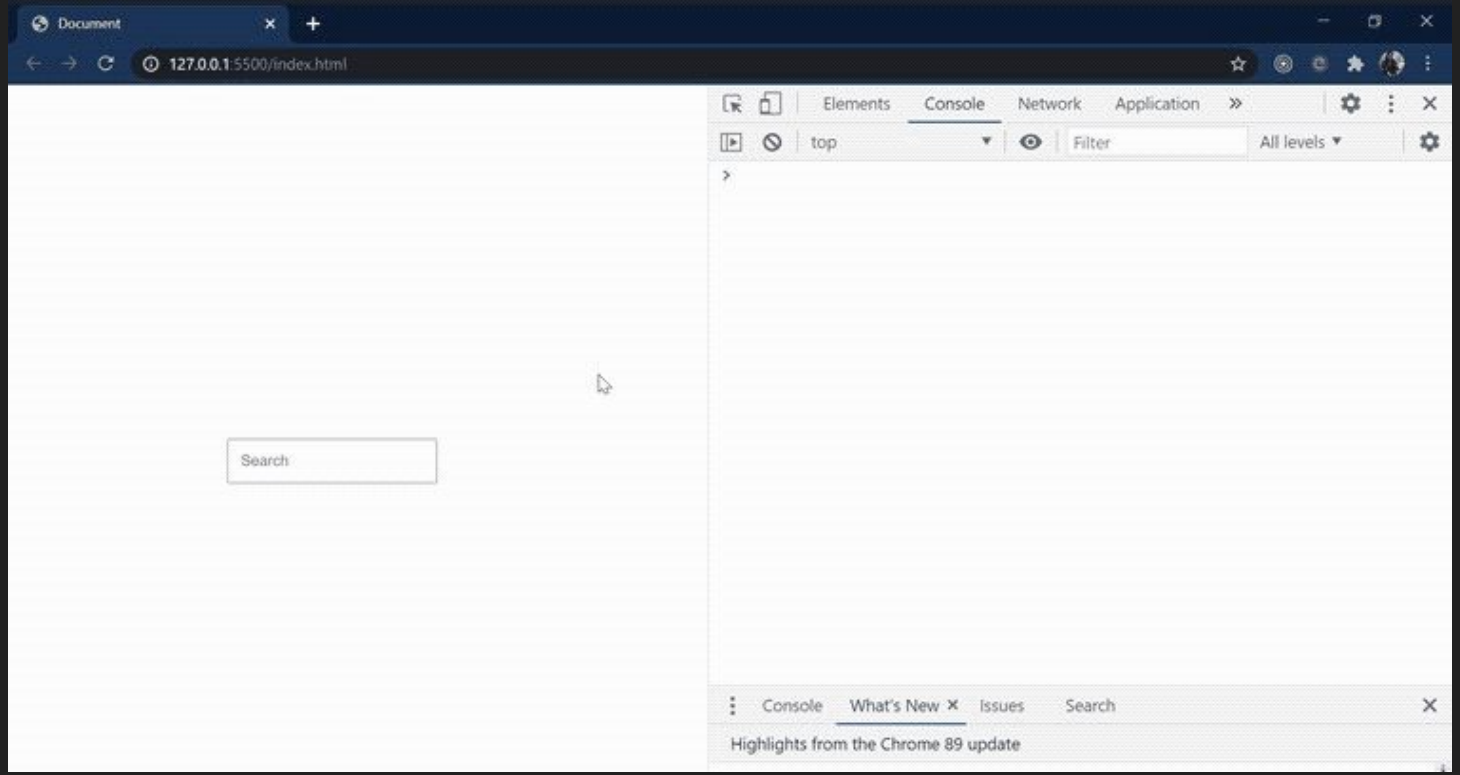
Shooting game - Pistol takes 1 sec time between each shot but the user clicks the mouse multiple times. Use throttle on mouse click.

https://web.archive.org/web/20220117092326/http://demo.nimius.net/debounce_throttle/

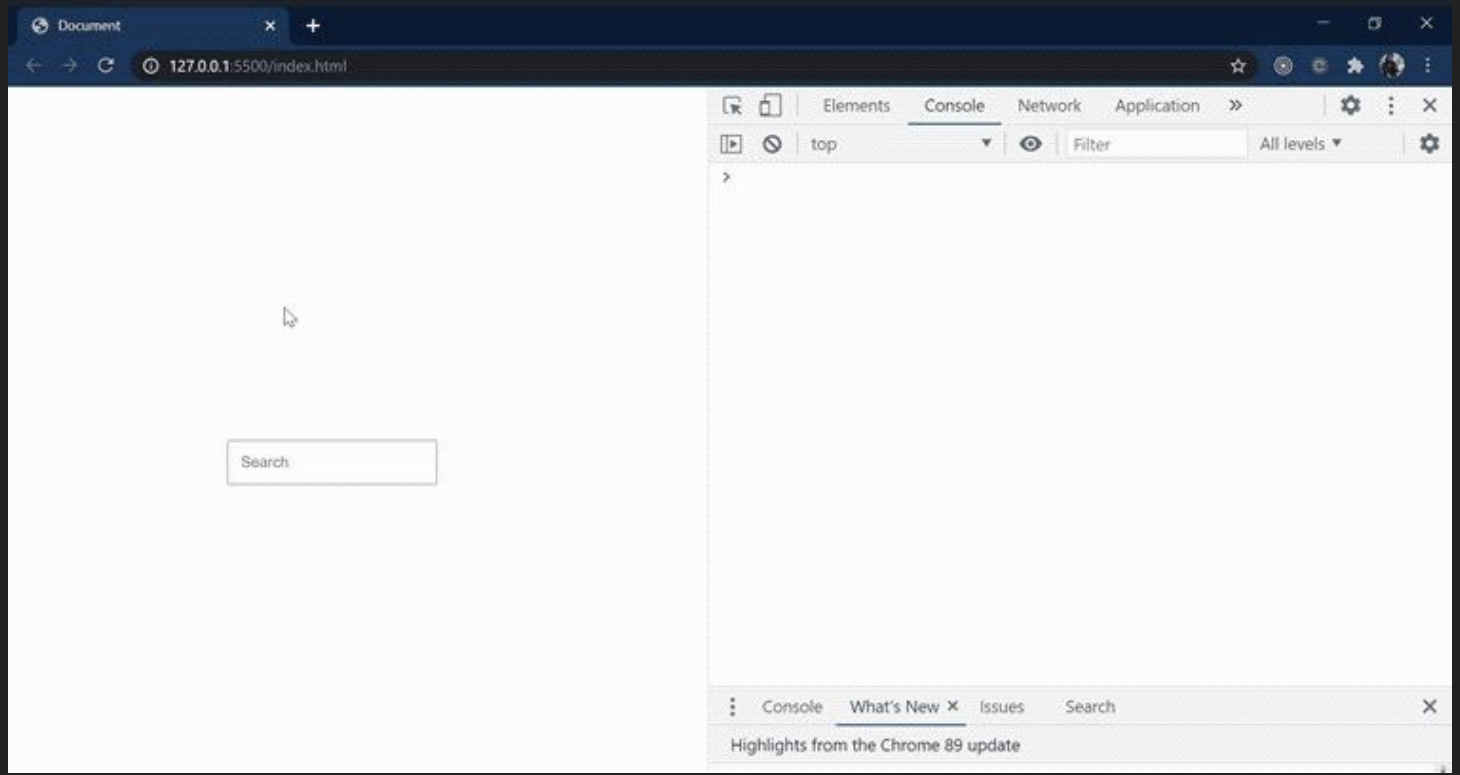
Example


Document
127.0.0.1:5500/index.html
Search
Elements
Console
Network
Application
top
Filter
All levels
Console
What's New
Issues
Search
Highlights from the Chrome 89 update

Example


Document
127.0.0.1:5500/index.html
Elements
Console
Network
Application
top
Filter
All levels
Search
Console
What's New
Issues
Search
Highlights from the Chrome 89 update

Example


Document
127.0.0.1:5500/index.html
Search
Elements
Console
Network
Application
top
Filter
All levels
Console
What's New
Issues
Search
Highlights from the Chrome 89 update