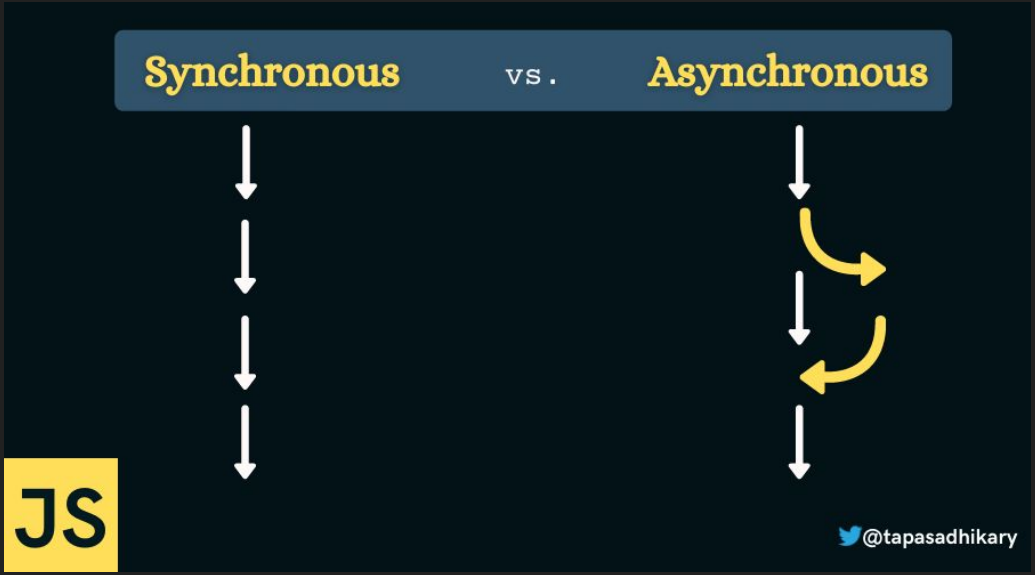
Welcome Geeks

Topics to be covered

- JS concepts continued
- DOM Manipulation and Events
- Small game development using JS

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?

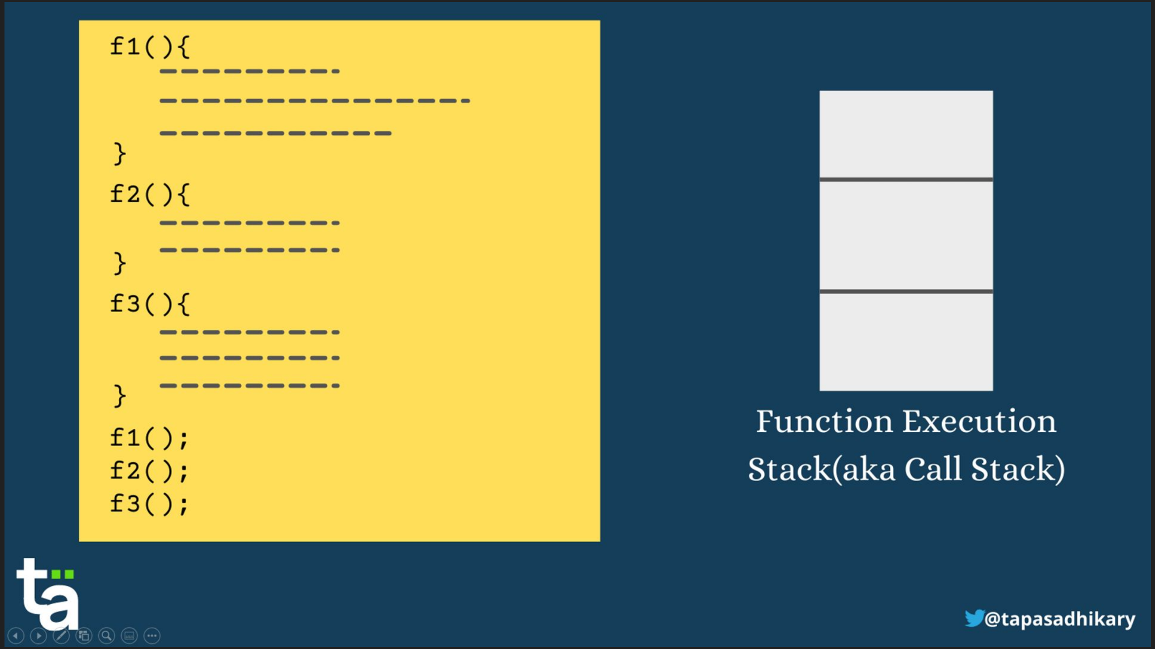
The JavaScript engine maintains a stack data structure called function execution stack. The purpose of the stack is to track the current function in execution.

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();
```



```
f1(){
    ----------
    -----------------
    -------------
}
f2(){
    ----------
}   ----------
f3(){
    ----------
    ----------
}   ----------
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);
```

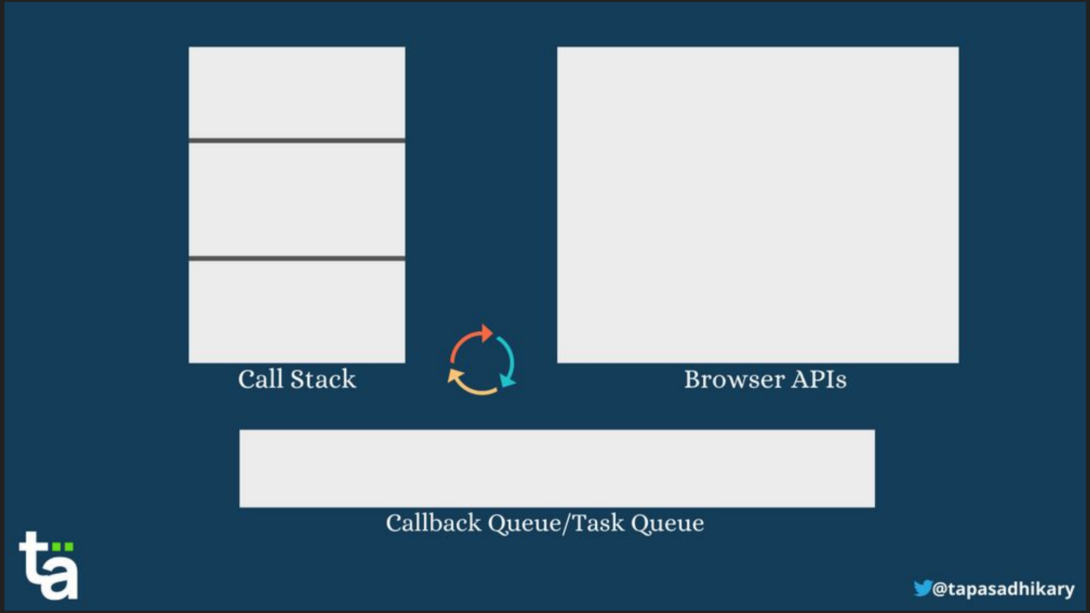
The setTimeout function executes a function after a certain amount of time has elapsed. In the code above, the text print me logs into the console after a delay of 5 seconds.

Asynchronous JS

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

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

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

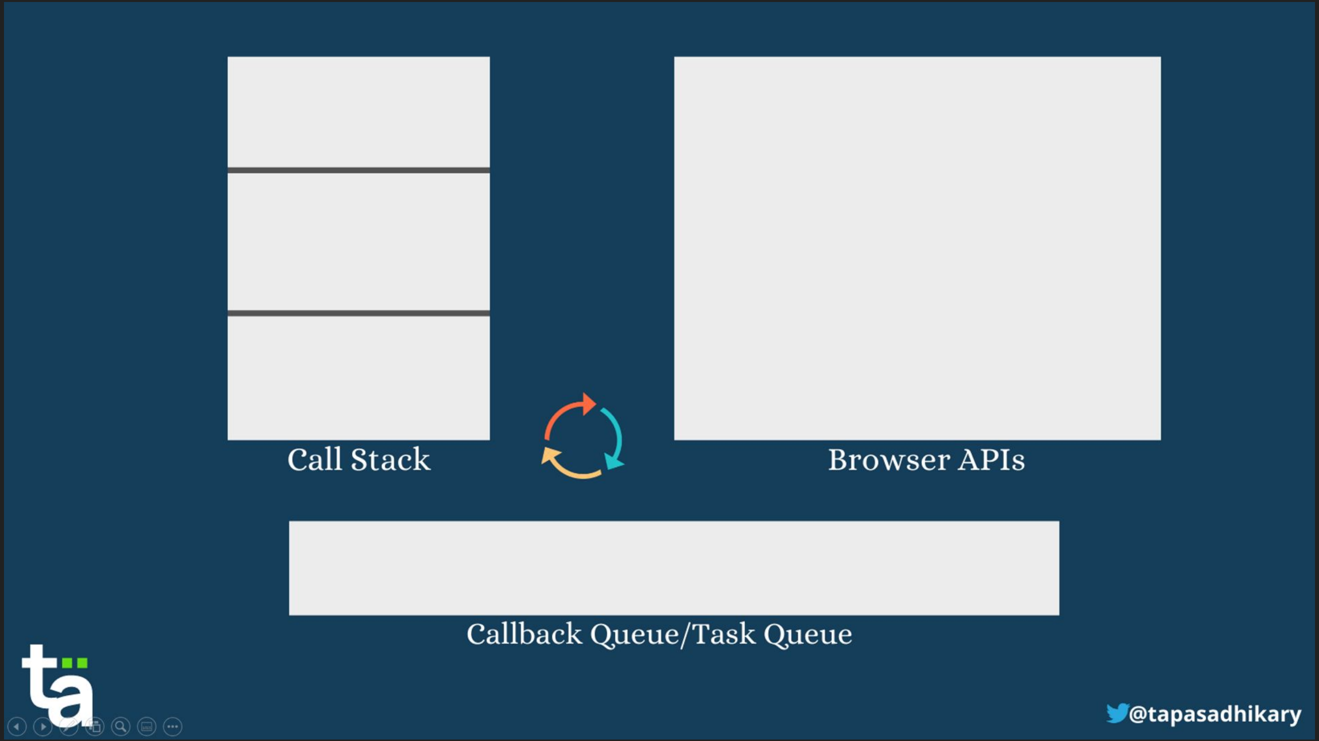
Asynchronous JS

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();
```

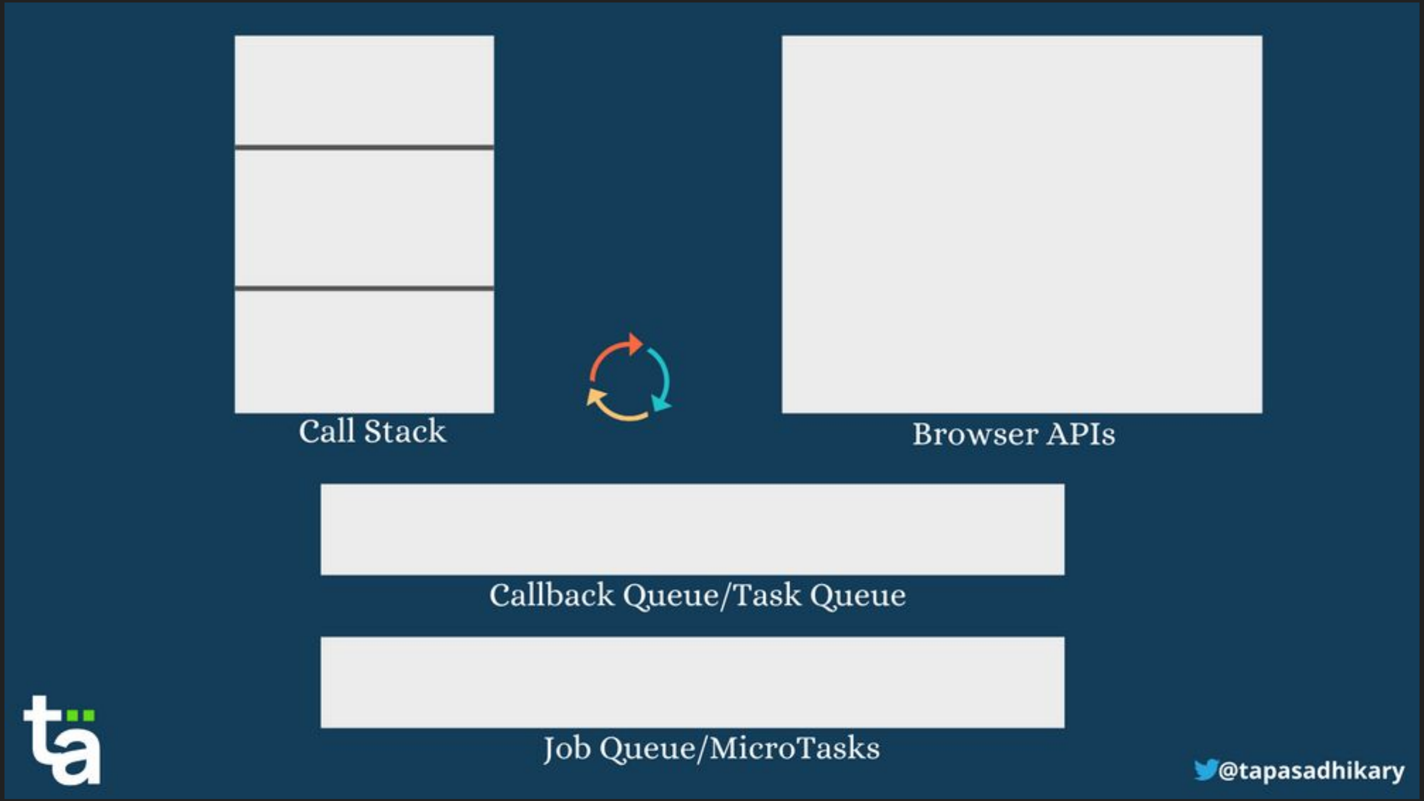


Asynchronous JS

JavaScript engine doesn't use the same callback queue we have seen earlier for browser APIs. It uses another special queue called the Job Queue.

The item in the callback queue is called a macro task, whereas the item in the job queue is called a micro task.

If both the queues got entries at the same point in time, the job queue gets preference over the callback queue.



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

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

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

  setTimeout(f1, 0);

  new Promise((resolve, reject) =>
    resolve('I am a promise')
  ).then(res => console.log(res))

  f2();
}
main();
```

Regular vs Arrow functions

- Syntactical difference
- Arguments binding
- No duplicate parameters

callback functions

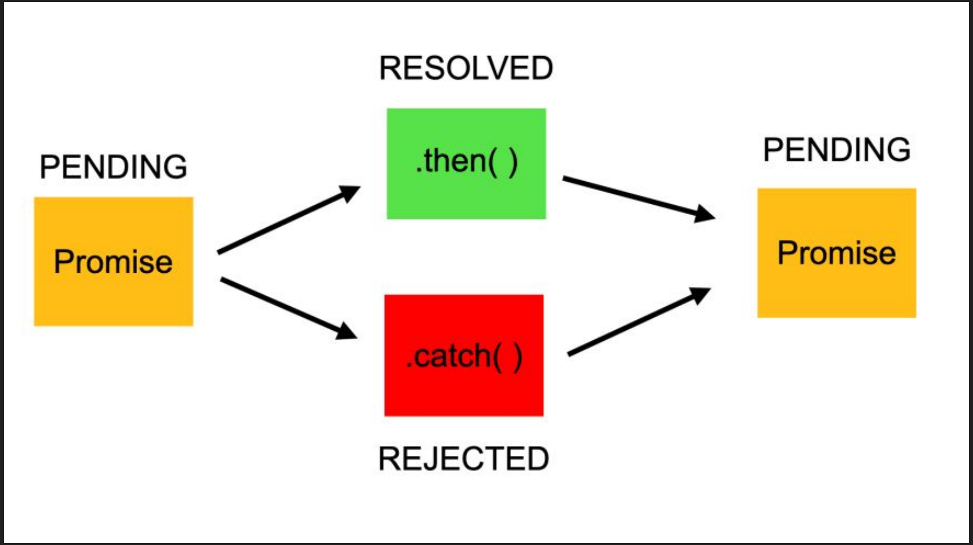
- A callback is a function passed as an argument to another function

```
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.

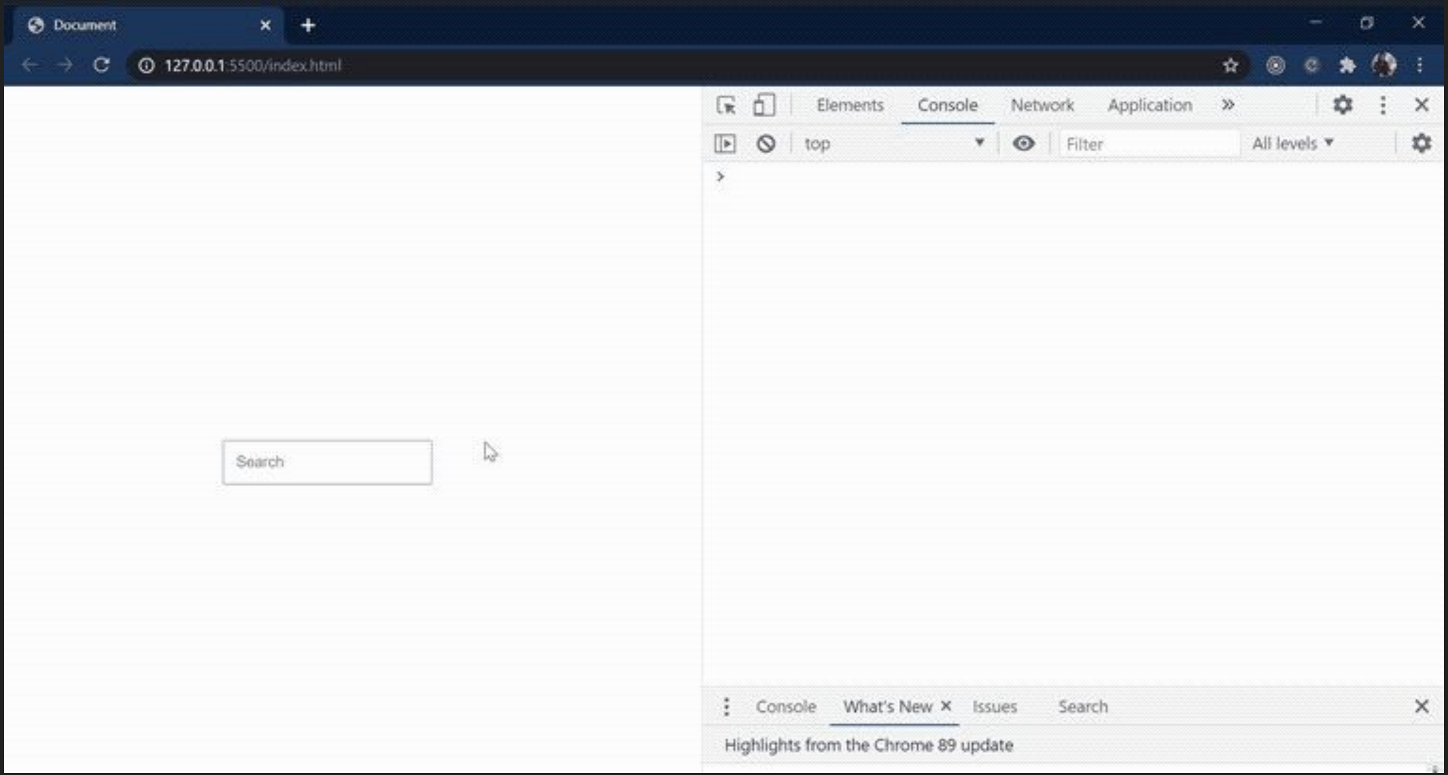
Example

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.

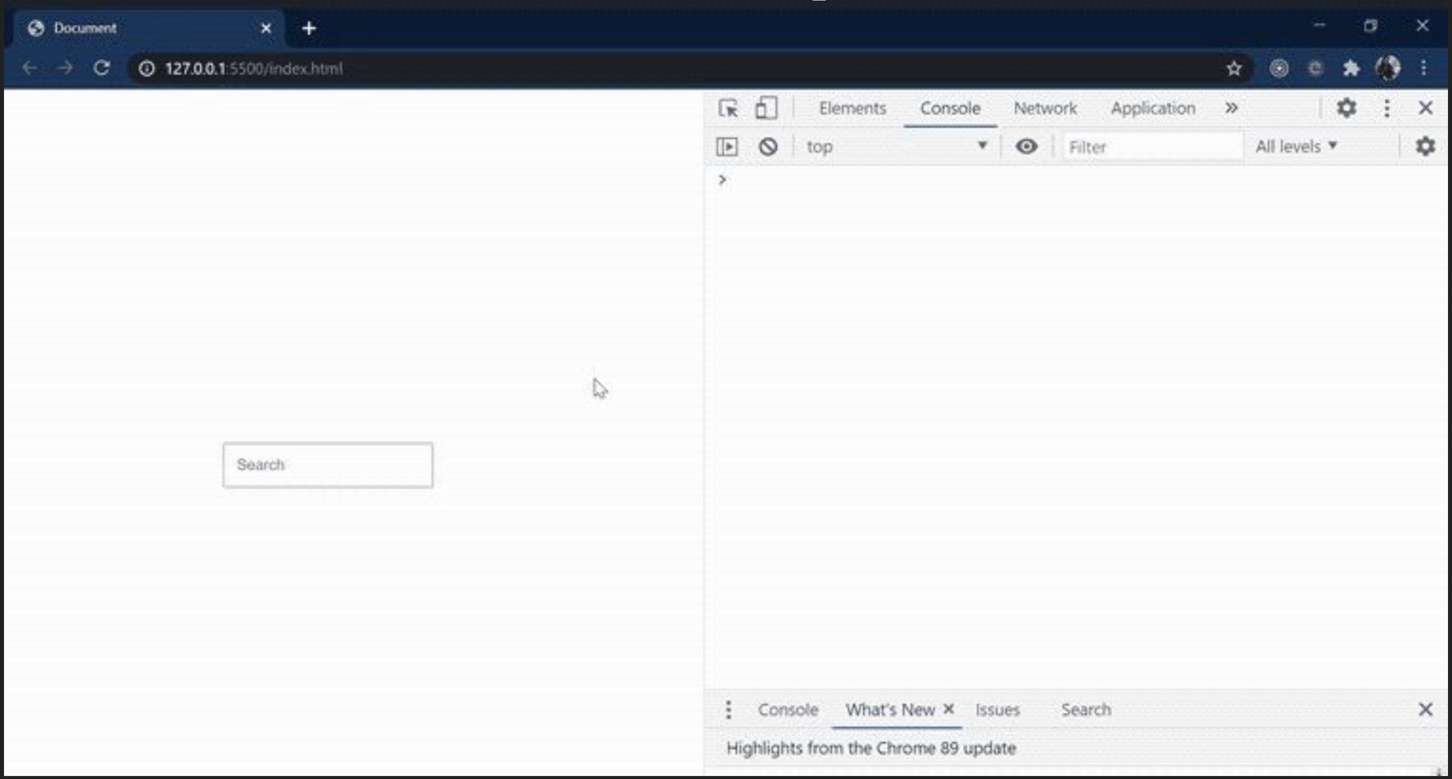
https://web.archive.org/web/20220117092326/http://demo.nimius.net/debounce_throttle/

Example

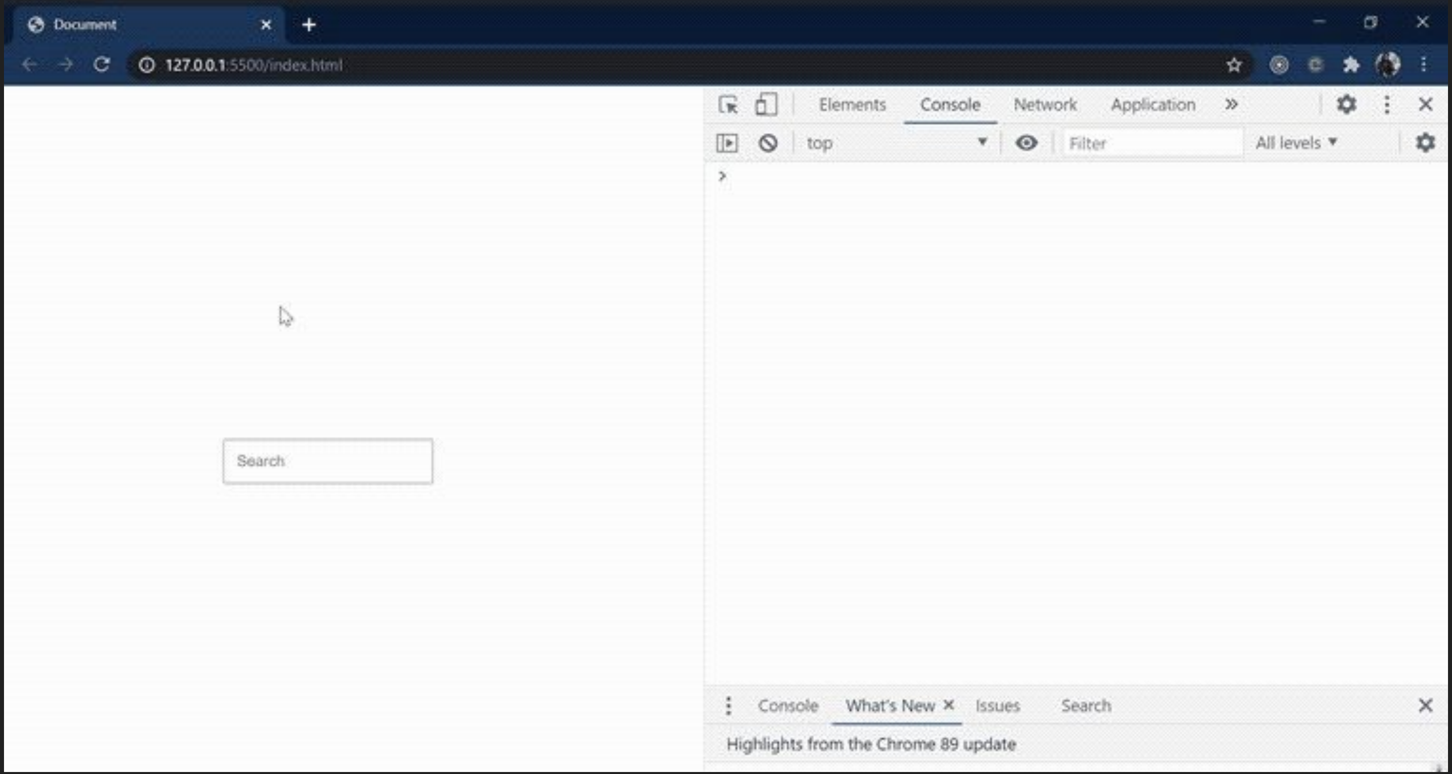


Highlights from the Chrome 89 update

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

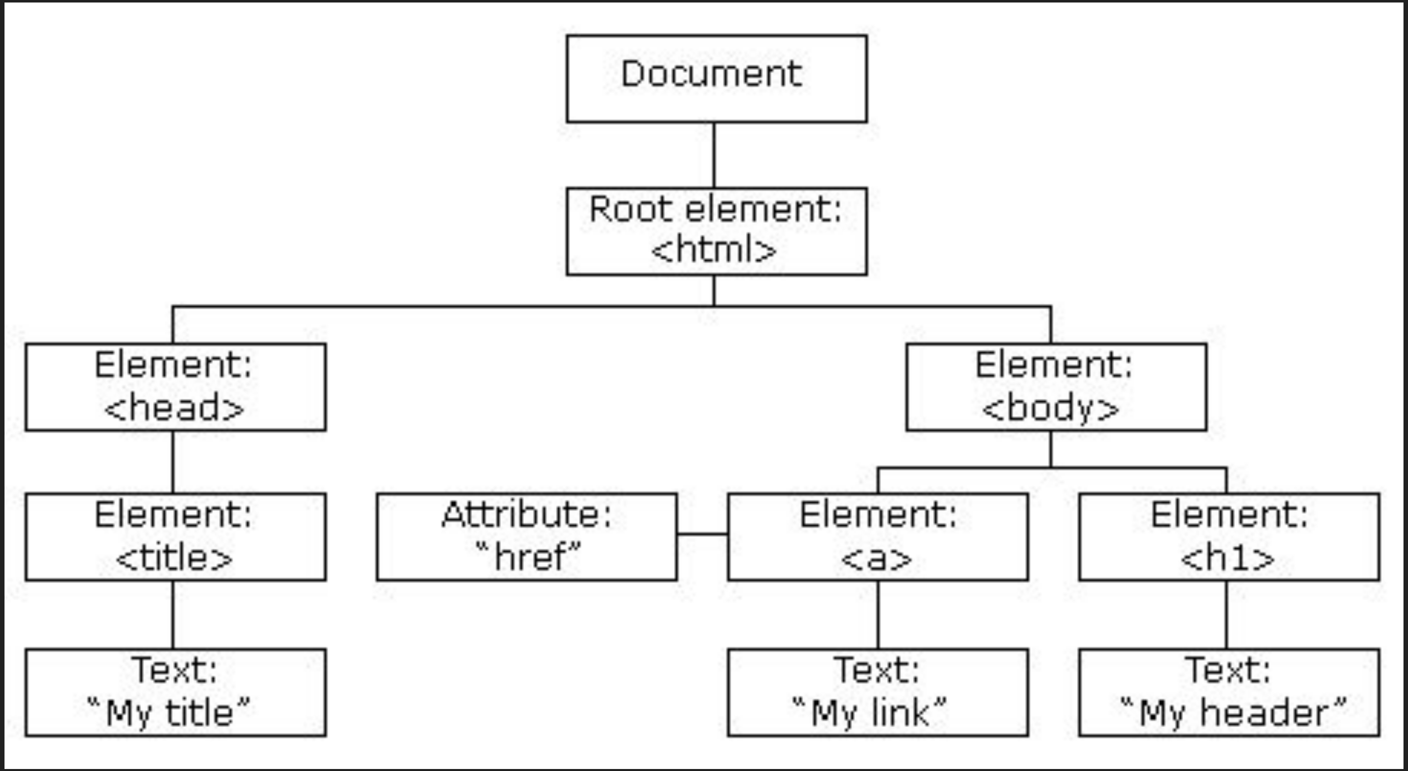
Example


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

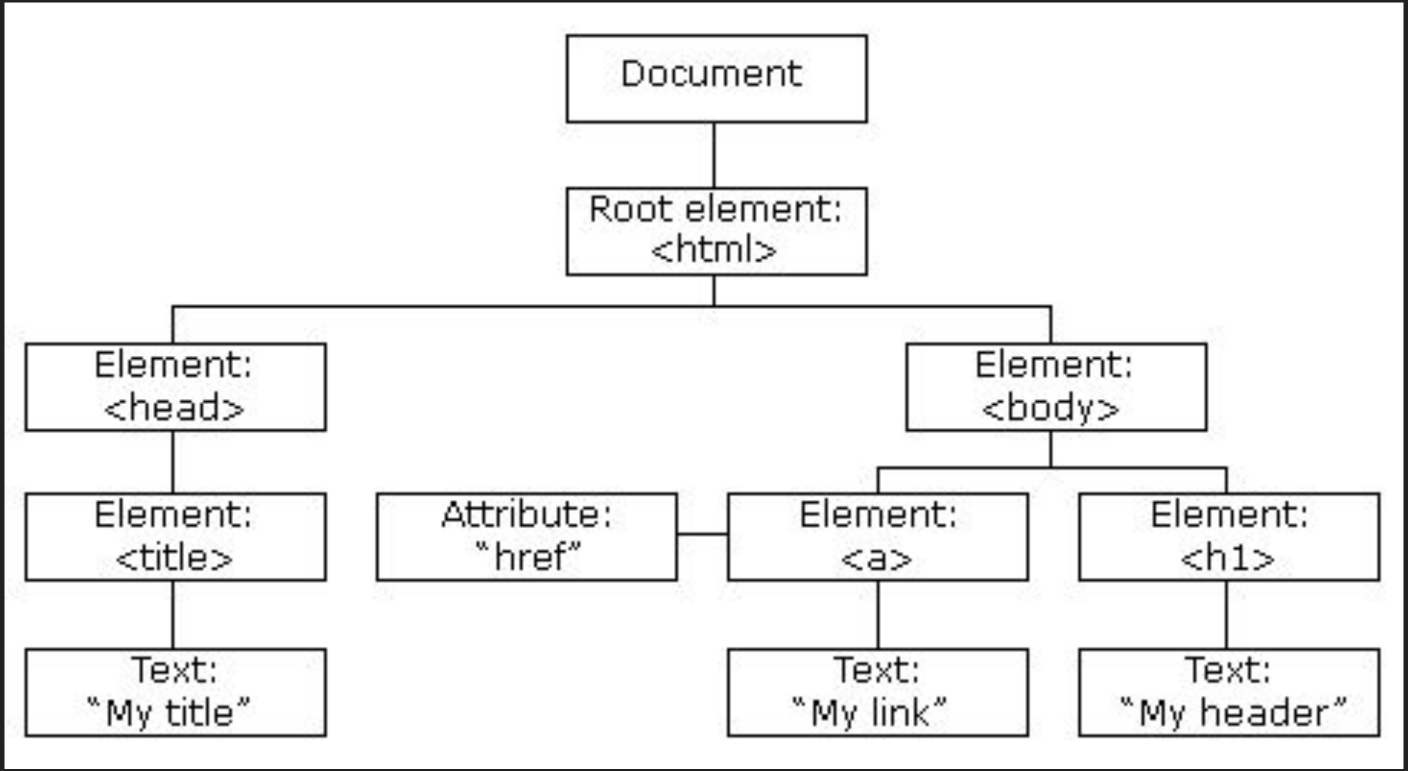
DOM

- DOCUMENT OBJECT MODEL
- Structural representation of the HTML Document
- JavaScript can be used to read/write/manipulate the DOM

DOM


Document
Root element: <html>
Element: <head>
Element: <body>
Element: <title>
Attribute: "href"
Element: <a>
Element: <h1>
Text: "My title"
Text: "My link"
Text: "My header"

DOM Manipulation????



Document Object

When an HTML document is loaded into a web browser, it becomes a document object.

The document object is the root node of the HTML document.

The document object is a property of the window object.

Single Element Selectors

```
document.getElementById();
document.querySelector();
```

Multi Element Selectors

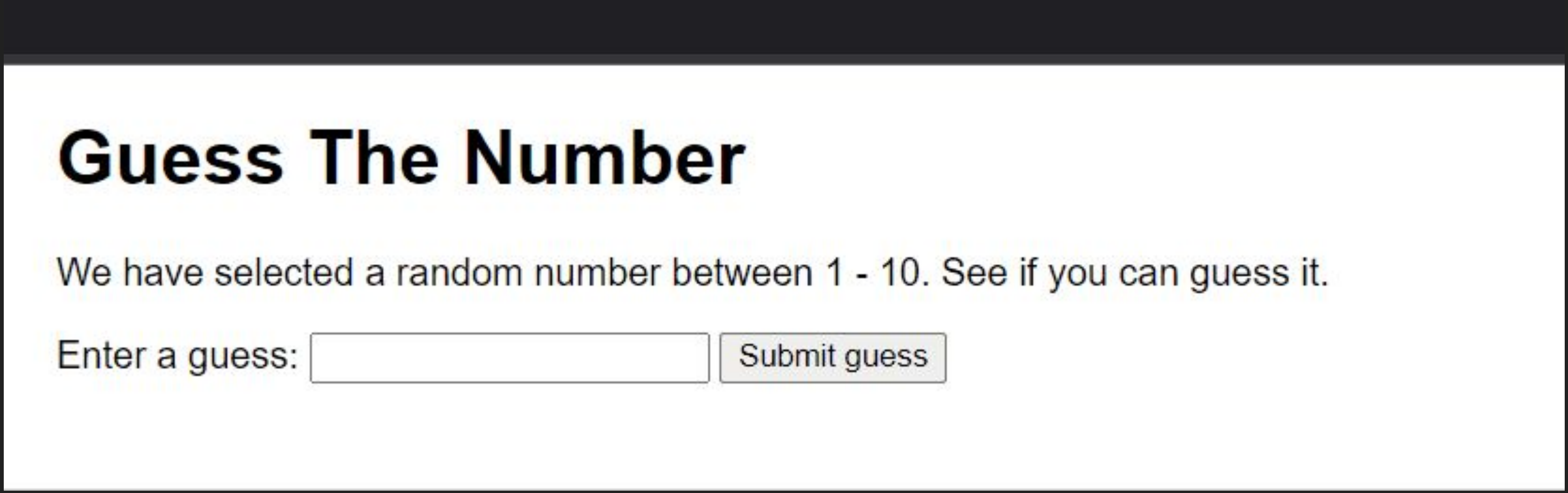
```
document.getElementsByClassName();
document.querySelectorAll();
```

Events

HTML DOM allows JavaScript to react to HTML events.

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

Number Guessing Game using JS



Guess The Number

We have selected a random number between 1 - 10. See if you can guess it.

Enter a guess: