

## macrotask#1

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) { message = "<<< main starting >>>"
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

Watch

Breakpoints

Scope

Local

this: undefined

message: "<<< main starting >>>"

Module

Global Window

Call Stack

logToConsole index.js:5

main index.js:9

(anonymous) index.js:30

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

Call Stack

logToConsole

main

(anonymous)

Host APIs  
(multithreaded,  
browser or Node.js)

(empty) microtask queue

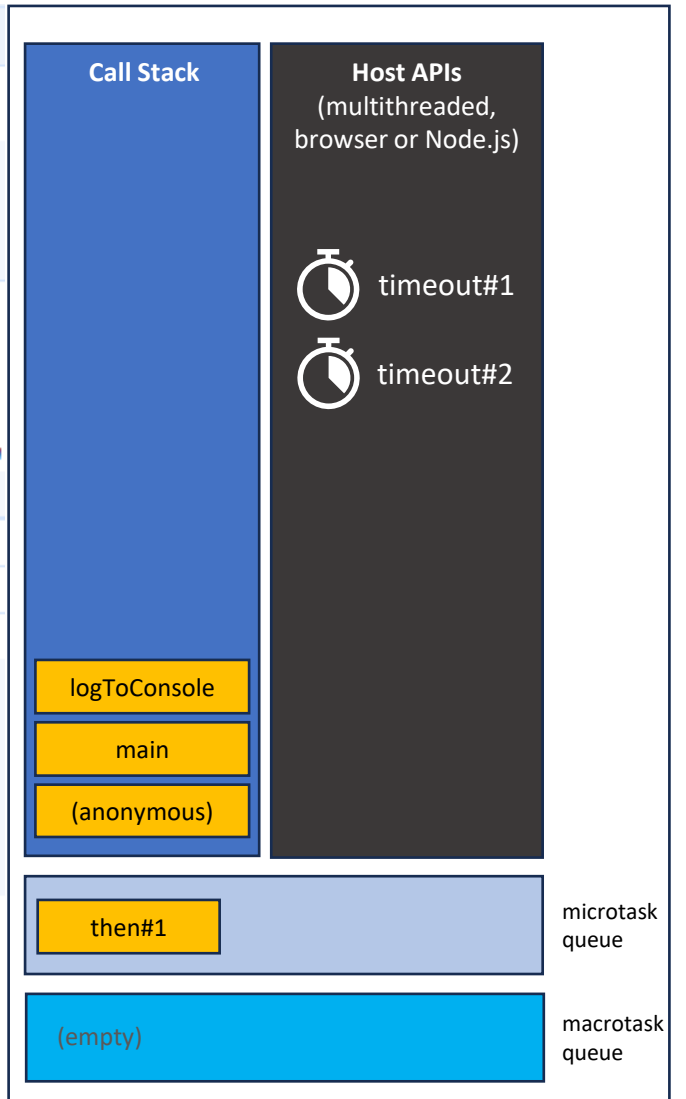
(empty) macrotask queue

## macrotask#1 (cont'd)

```
index.js x
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) { message = "<<< main ending >>>"
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

- Watch
- Breakpoints
- Scope
  - Local
    - this: undefined
    - message: "<<< main ending >>>"
  - Module
  - Global Window
- Call Stack
  - logToConsole index.js:5
  - main index.js:27
  - (anonymous) index.js:30
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints



## microtask#1

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) { message = ">>> then#1"
5 console.log(message);
6 }
7
8 function main() {
9 logToConsole('<<< main starting >>>');
10
11 setTimeout(function timeout1() {
12 |   logToConsole('>>> timeout#1');
13 | }, 1000);
14
15 setTimeout(function timeout2() {
16 |   logToConsole('>>> timeout#2');
17 | }, 2000);
18
19 Promise.resolve() /* promise#1 */
20 | .then(function onFulfilled1() /* then#1 */ {
21 |   logToConsole('>>> then#1');
22 | }) /* promise#2 */
23 | .then(function onFulfilled2() /* then#2 */ {
24 |   logToConsole('>>> then#2');
25 | }) /* promise#3 */;
26
27 logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

Watch

Breakpoints

Scope

Local

this: undefined

message: ">>> then#1"

Module

Global Window

Call Stack

logToConsole index.js:5

onFulfilled1 index.js:21

(anonymous) async-promise.js:87

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

Call Stack

logToConsole

onFulfilled1

(anonymous)

Host APIs  
(multithreaded,  
browser or Node.js)

timeout#1

timeout#2

then#2 see note

(empty)

Microtask queue

Macrotask queue

Note: enqueued *after* the `promise#2` is fulfilled with the return value (`undefined`) of `onFulfilled1()`.

## microtask#2

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) { message = ">>> then#2"
5 console.log(message);
6 }
7
8 function main() {
9 logToConsole('<<< main starting >>>');
10
11 setTimeout(function timeout1() {
12   logToConsole('>>> timeout#1');
13 }, 1000);
14
15 setTimeout(function timeout2() {
16   logToConsole('>>> timeout#2');
17 }, 2000);
18
19 Promise.resolve() /* promise#1 */
20 .then(function onFulfilled1() /* then#1 */ {
21   logToConsole('>>> then#1');
22 }) /* promise#2 */
23 .then(function onFulfilled2() /* then#2 */ {
24   logToConsole('>>> then#2');
25 }) /* promise#3 */;
26
27 logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

Watch

Breakpoints

Scope

Local

Module

Global

Call Stack

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

this: undefined

message: ">>> then#2"

Window

logToConsole

onFulfilled2

(anonymous)

index.js:5

index.js:24

async-promise.js:87

Call Stack

Host APIs  
(multithreaded,  
browser or Node.js)

timeout#1

timeout#2#2

(empty)

microtask queue

(empty)

macrotask queue

Idle until timeout#1 fires

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) {
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13     }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17     }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

▶ Watch

▶ Breakpoints

▼ Scope

Not paused

▼ Call Stack

Not paused

▶ XHR/fetch Breakpoints

▶ DOM Breakpoints

▶ Global Listeners

▶ Event Listener Breakpoints

▶ CSP Violation Breakpoints

Call Stack

(empty)

Host APIs  
(multithreaded,  
browser or Node.js)

timeout#1

timeout#2

(empty)

microtask queue

(empty)

macrotask queue



timeout#1

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) {
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

▶ Watch

▶ Breakpoints

▼ Scope

Not paused

▼ Call Stack

Not paused

▶ XHR/fetch Breakpoints

▶ DOM Breakpoints

▶ Global Listeners


▶ Event Listener Breakpoints

▶ CSP Violation Breakpoints

Call Stack

(empty)

Host APIs  
(multithreaded,  
browser or Node.js)

 timeout#2

(empty)

microtask queue

timeout1

macrotask queue

## timeout1

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper
3
4 function logToConsole(message) { message = ">>> timeout#1"
5 console.log(message);
6 }
7
8 function main() {
9 logToConsole('<<< main starting >>>');
10
11 setTimeout(function timeout1() {
12   logToConsole('>>> timeout#1');
13 }, 1000);
14
15 setTimeout(function timeout2() {
16   logToConsole('>>> timeout#2');
17 }, 2000);
18
19 Promise.resolve() /* promise#1 */
20 .then(function onFulfilled1() /* then#1 */ {
21   logToConsole('>>> then#1');
22 }) /* promise#2 */
23 .then(function onFulfilled2() /* then#2 */ {
24   logToConsole('>>> then#2');
25 }) /* promise#3 */;
26
27 logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

Watch

Breakpoints

Scope

Local

- this: undefined
- message: ">>> timeout#1"

Module

Global

Window

Call Stack

- logToConsole index.js:5
- timeout1 index.js:12
- (anonymous) setTimeoutWrapper.js:8
- setTimeout
- setTimeoutWrapper setTimeoutWrapper.js:6
- main index.js:11
- (anonymous) index.js:30

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

Call Stack

Host APIs  
(multithreaded,  
browser or Node.js)

logToConsole

timeout1

...

(empty)

Microtask queue

(empty)

Macrotask queue

timeout2

Idle until timeout#2 fires

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) {
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

▶ Watch

▶ Breakpoints

▼ Scope

Not paused

▼ Call Stack

Not paused

▶ XHR/fetch Breakpoints

▶ DOM Breakpoints

▶ Global Listeners


▶ Event Listener Breakpoints

▶ CSP Violation Breakpoints

Call Stack

(empty)

Host APIs  
(multithreaded,  
browser or Node.js)

 timeout\_2

(empty)

Microtask queue

(empty)

Macrotask queue





## timeout#2

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) {
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

▶ Watch

▶ Breakpoints

▼ Scope

Not paused

▼ Call Stack

Not paused

▶ XHR/fetch Breakpoints

▶ DOM Breakpoints

▶ Global Listeners

▶ Event Listener Breakpoints

▶ CSP Violation Breakpoints

Call Stack

(empty)

Host APIs  
(multithreaded,  
browser or Node.js)

(empty)

microtask queue

timeout2

macrotask queue

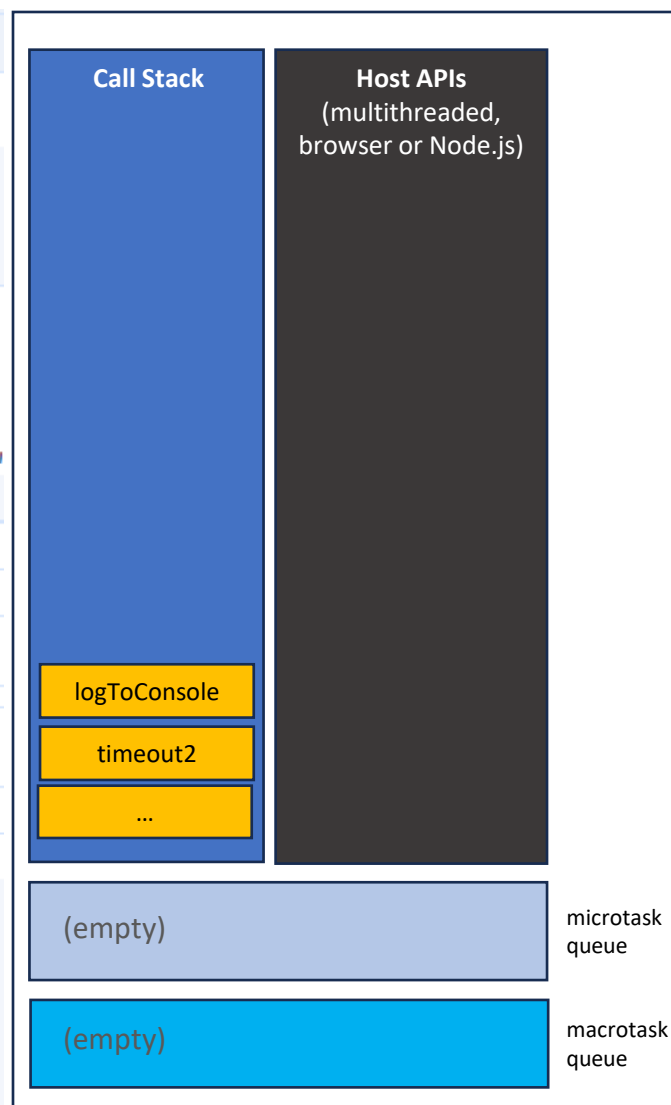
## timeout2

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper';
3
4 function logToConsole(message) { message = ">>> timeout#2"
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Paused on breakpoint

- Watch
- Breakpoints
- Scope
  - Local
    - this: undefined
    - message: ">>> timeout#2"
  - Module
  - Global
- Call Stack
  - logToConsole index.js:5
  - timeout2 index.js:16
  - (anonymous) setTimeoutWrapper.js:8
  - setTimeout
  - setTimeoutWrapper setTimeoutWrapper.js:6
  - main index.js:15
  - (anonymous) index.js:30
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints



execution finished

index.js x

```
1 import { AsyncPromise as Promise } from '../lib/async-promise.js';
2 import { setTimeoutWrapper as setTimeout } from './setTimeoutWrapper
3
4 function logToConsole(message) {
5   console.log(message);
6 }
7
8 function main() {
9   logToConsole('<<< main starting >>>');
10
11   setTimeout(function timeout1() {
12     logToConsole('>>> timeout#1');
13   }, 1000);
14
15   setTimeout(function timeout2() {
16     logToConsole('>>> timeout#2');
17   }, 2000);
18
19   Promise.resolve() /* promise#1 */
20     .then(function onFulfilled1() /* then#1 */ {
21       logToConsole('>>> then#1');
22     }) /* promise#2 */
23     .then(function onFulfilled2() /* then#2 */ {
24       logToConsole('>>> then#2');
25     }) /* promise#3 */;
26
27   logToConsole('<<< main ending >>>');
28 }
29
30 main();
31
```

Watch

Breakpoints

Scope

Not paused

Call Stack

Not paused

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

Call Stack

(empty)

Host APIs  
(multithreaded,  
browser or Node.js)

(empty)

microtask queue

(empty)

macrotask queue