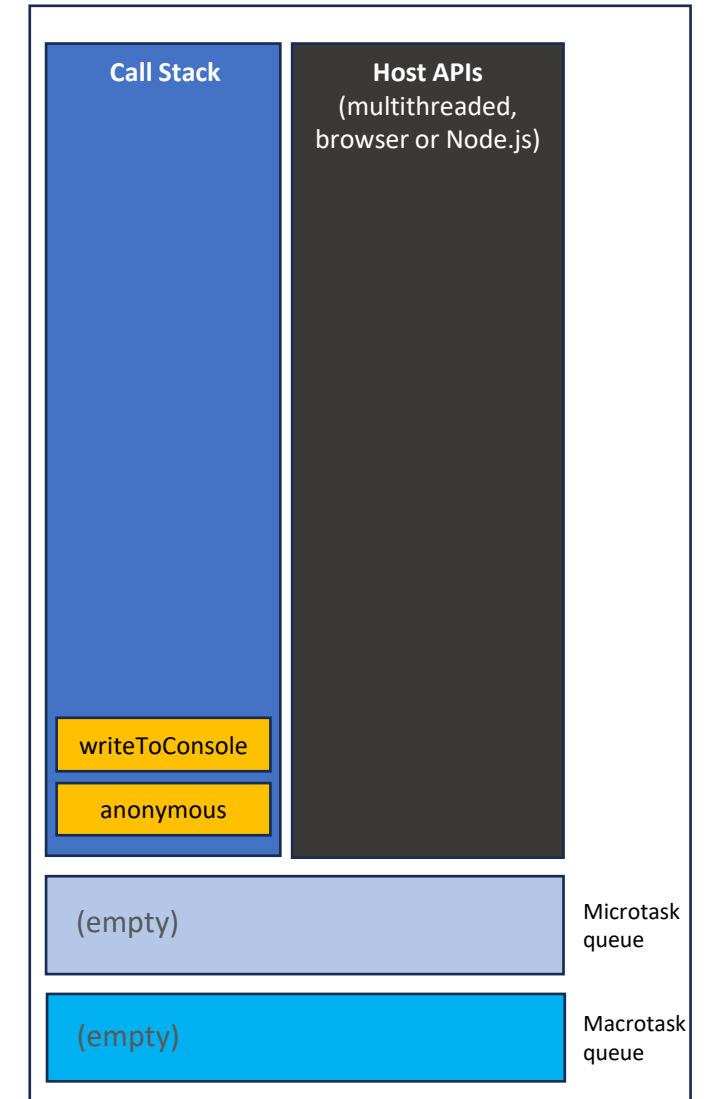


## Event 1: Macrotask

```
1  #!/usr/bin/env node
2  use strict;
3
4  function writeToConsole(message) { message = "starting"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

**Paused on breakpoint**

- Watch
- Breakpoints
- Scope
  - Local
    - this: undefined
    - message: "starting"
  - Script
    - p: <value unavailable>
  - Global
- Call Stack
  - writeToConsole index.js:5
  - (anonymous) index.js:8
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints

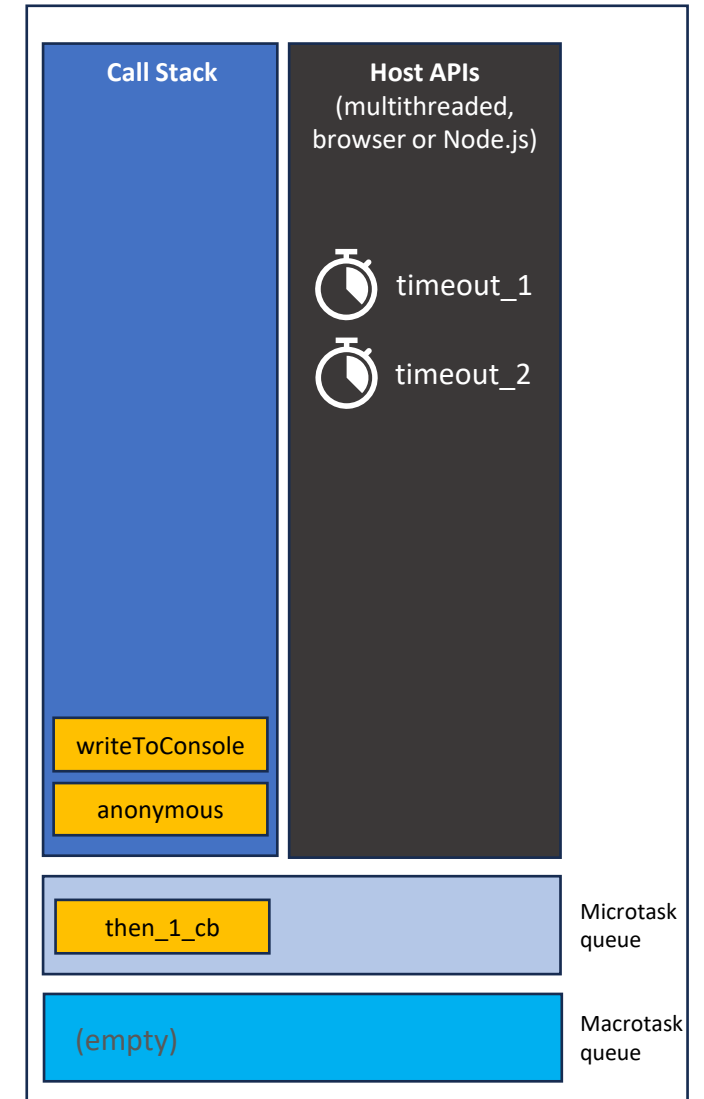


## Event 1: Macrotask (cont'd)

```
1  #!/usr/bin/env node
2  use strict;
3
4  function writeToConsole(message) { message = "ending"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

**Paused on breakpoint**

- Watch
- Breakpoints
- Scope
  - Local
    - this: undefined
    - message: "ending"
  - Script
    - p: Promise {<fulfilled>: undefined}
  - Global
    - Window
- Call Stack
  - writeToConsole index.js:5
  - (anonymous) index.js:28
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints

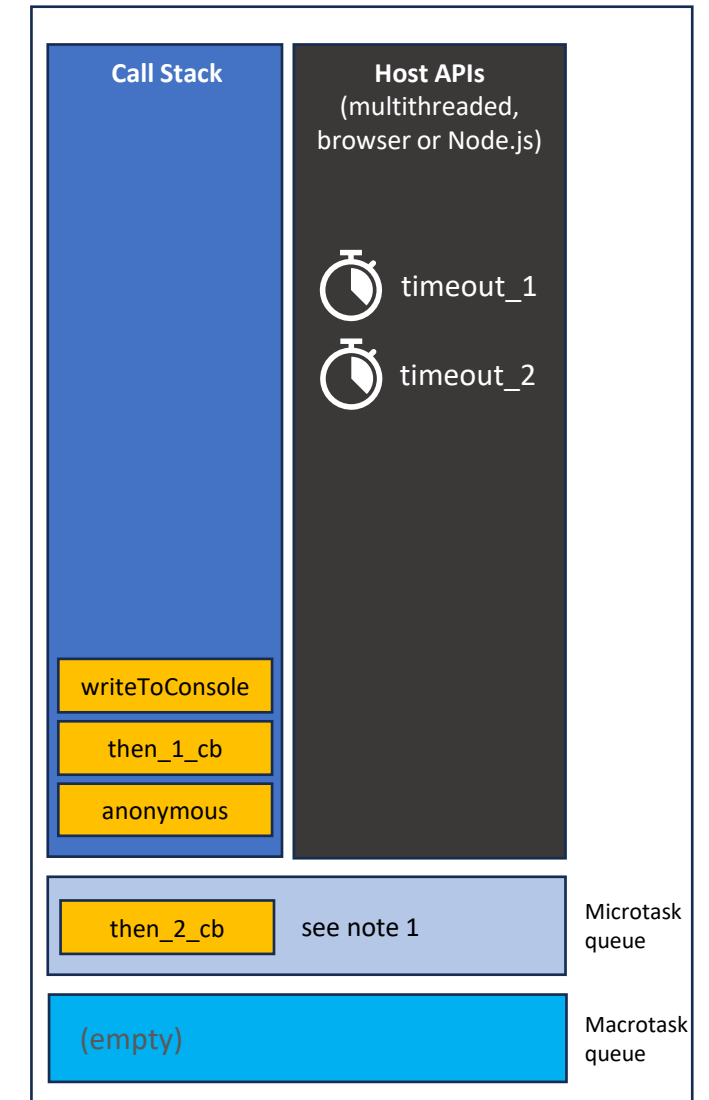


## Event 2: Microtask then\_1\_cb

```
1  #!/usr/bin/env node
2  'use strict';
3
4  function writeToConsole(message) { message = "then 1"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

**Paused on breakpoint**

- Watch
- Breakpoints
- Scope
  - Local
    - this: undefined
    - message: "then 1"
  - Script
    - p: Promise {<fulfilled>: undefined}
  - Global
    - Window
- Call Stack
  - writeToConsole index.js:5
  - then\_1\_cb index.js:23
  - Promise.then (async) —
  - (anonymous) index.js:22
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints



Note 1: placed in the microtask queue *after* the promise returned by `then_1_cb` is fulfilled.

## Event 3: Microtask then\_2\_cb

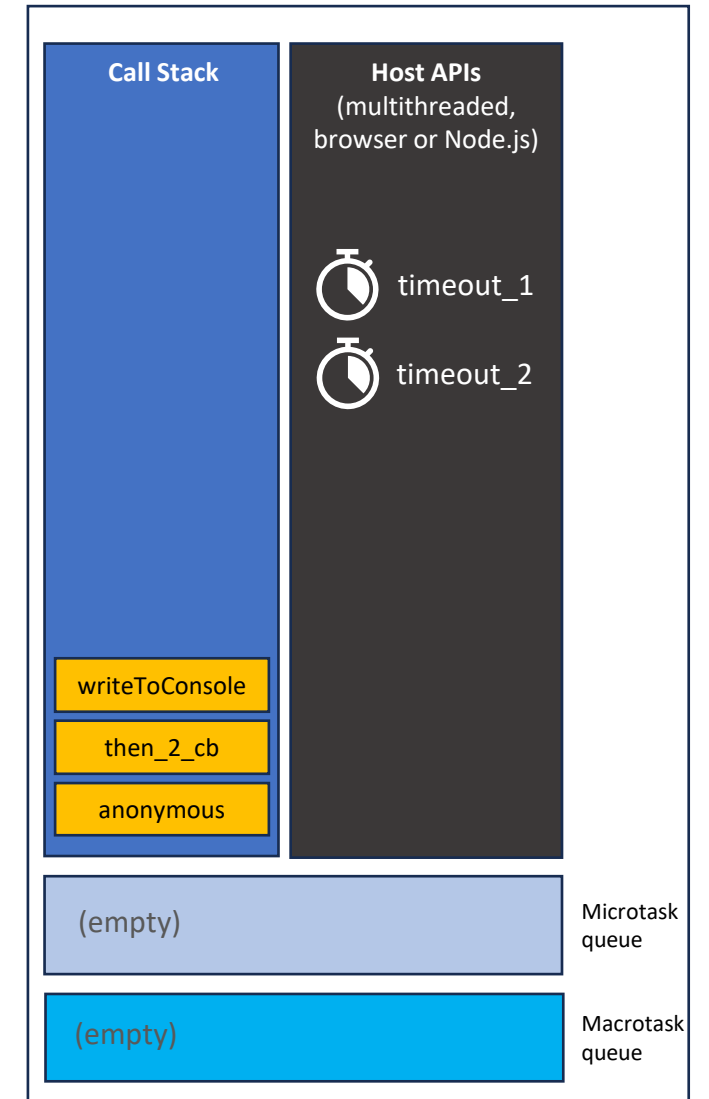
The screenshot shows a code editor with a file named `index.js`. The code is as follows:

```
1  #!/usr/bin/env node
2  'use strict';
3
4  function writeToConsole(message) { message = "then 2"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

A red arrow points to line 5, where a breakpoint is set. The Chrome DevTools interface is open, showing the "Paused on breakpoint" message. The "Scope" panel is expanded, showing the local variables `this: undefined` and `message: "then 2"`. The "Call Stack" panel is also expanded, showing the following frames:

- `writeToConsole` (index.js:5)
- `then_2_cb` (index.js:25)
- `Promise.then (async)`
- `(anonymous)` (index.js:24)

The "Host APIs" panel shows two timers: `timeout_1` and `timeout_2`.

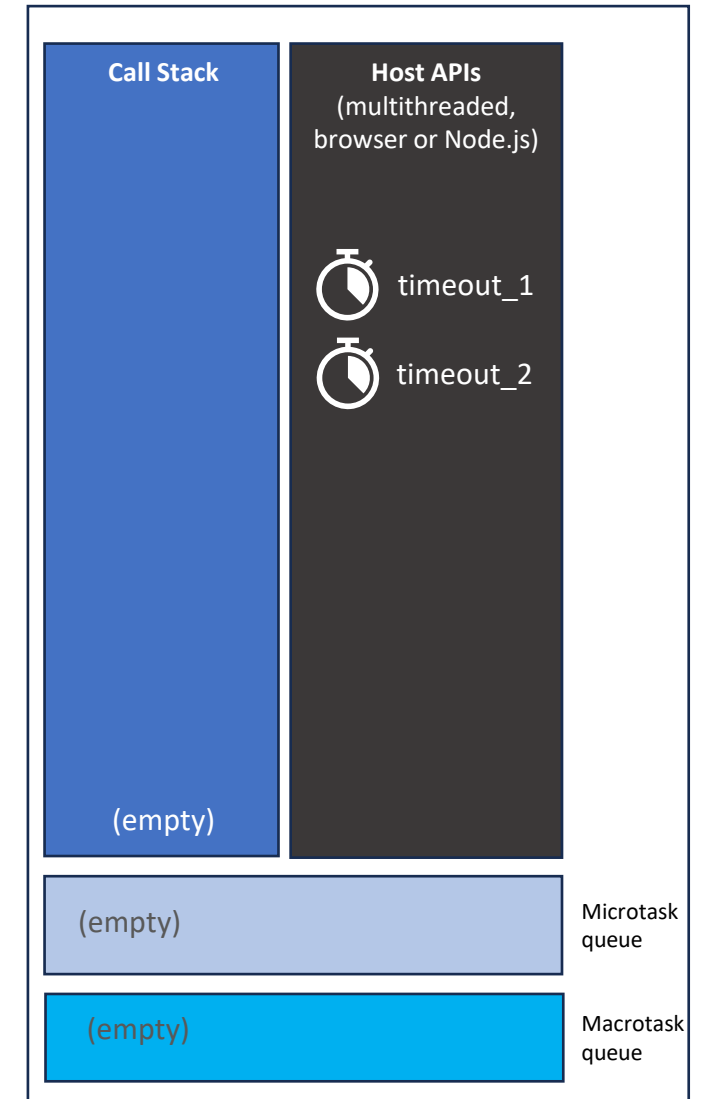


Idle until timeout\_1 fires

```
index.js X Grammarly-check.js
1  #!/ What will be printed to the console and in what order
2  'use strict';
3
4  function writeToConsole(message) {
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

Debugger sidebar:

- Watch
- Breakpoints
- Scope
- Call Stack (Not paused)
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints

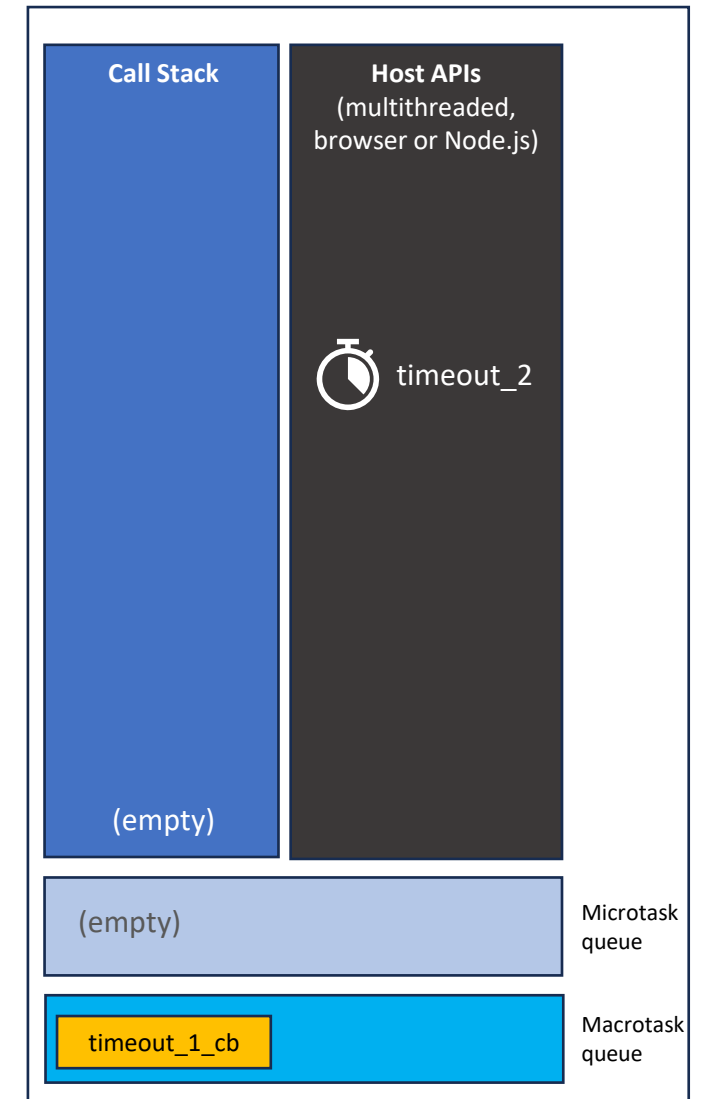


timeout\_1 fired

```
index.js X Grammarly-check.js
1  ///! What will be printed to the console and in what order
2  'use strict';
3
4  function writeToConsole(message) {
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

Debugger sidebar:

- Watch
- Breakpoints
- Scope
- Call Stack (Not paused)
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints



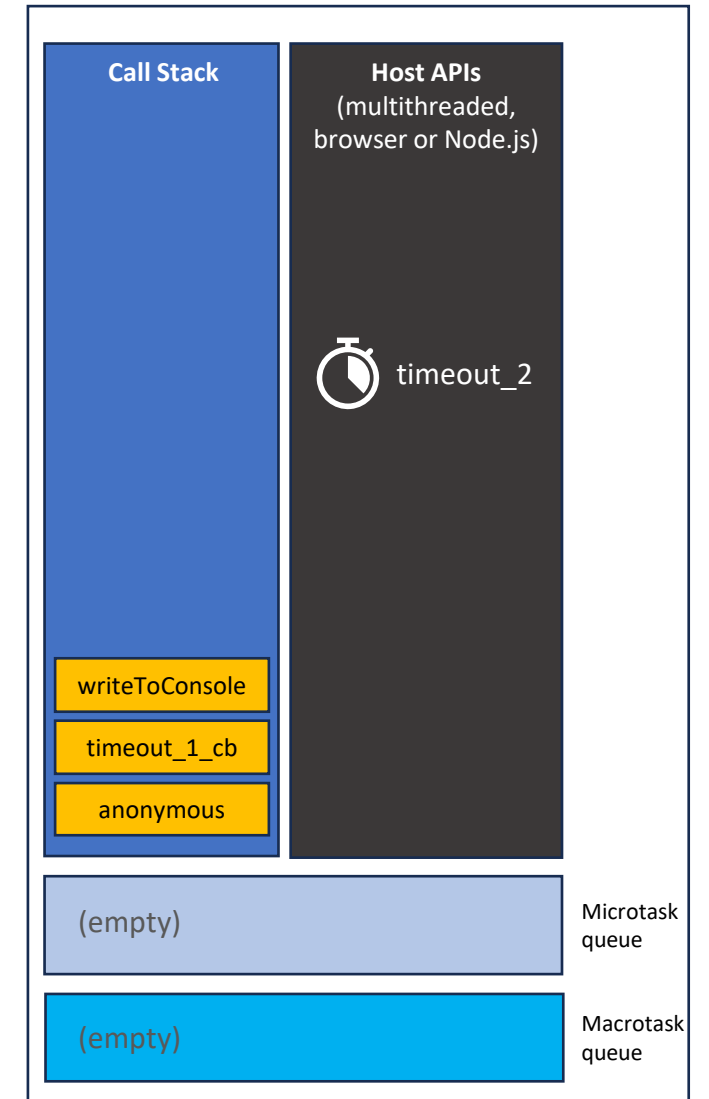
## Event 4: Macrotask timeout\_1\_cb

The screenshot shows a code editor with the following JavaScript code in `index.js`:

```
1  #!/usr/bin/env node
2  'use strict';
3
4  function writeToConsole(message) { message = "timeout 1"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

A red arrow points to line 5, where a breakpoint is set. The Chrome DevTools interface shows the following details:

- Paused on breakpoint**
- Scope**
  - Local**
    - `this`: undefined
    - `message`: "timeout 1"
  - Script**
    - `p`: Promise {<fulfilled>: undefined}
  - Global**: Window
- Call Stack**
  - `writeToConsole` (index.js:5)
  - `timeout_1_cb` (index.js:11)
  - `setTimeout (async)`
  - `(anonymous)` (index.js:10)

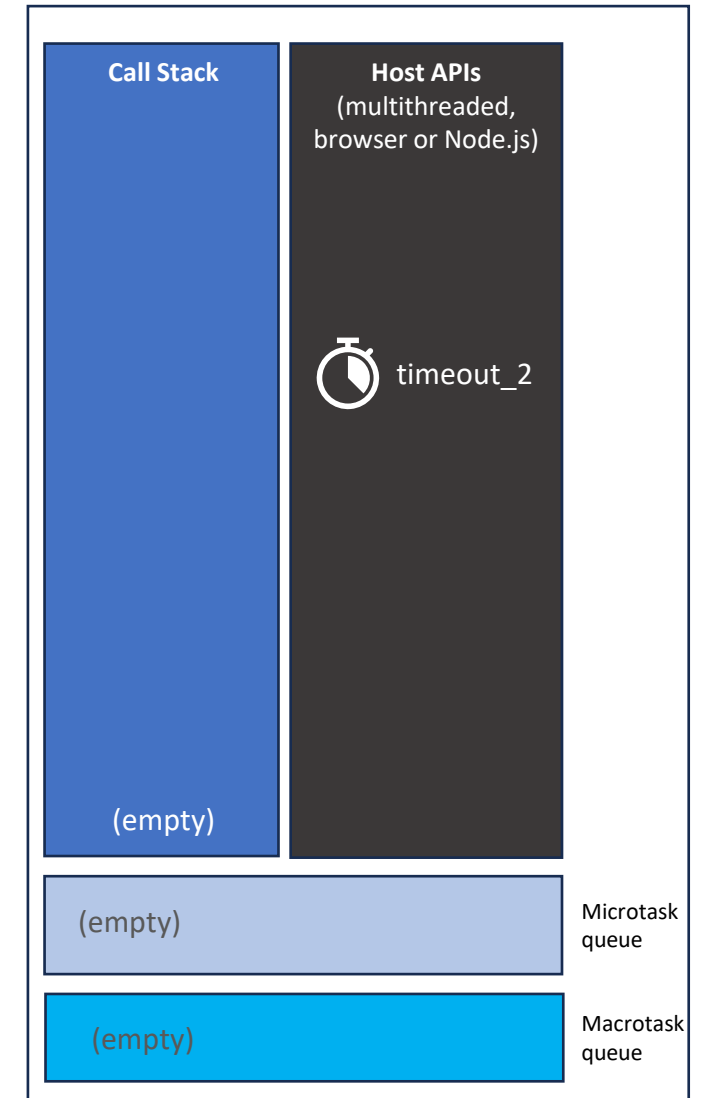


Idle until timeout\_2 fires

```
index.js X Grammarly-check.js
1  #!/usr/bin/env node
2  'use strict';
3
4  function writeToConsole(message) {
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

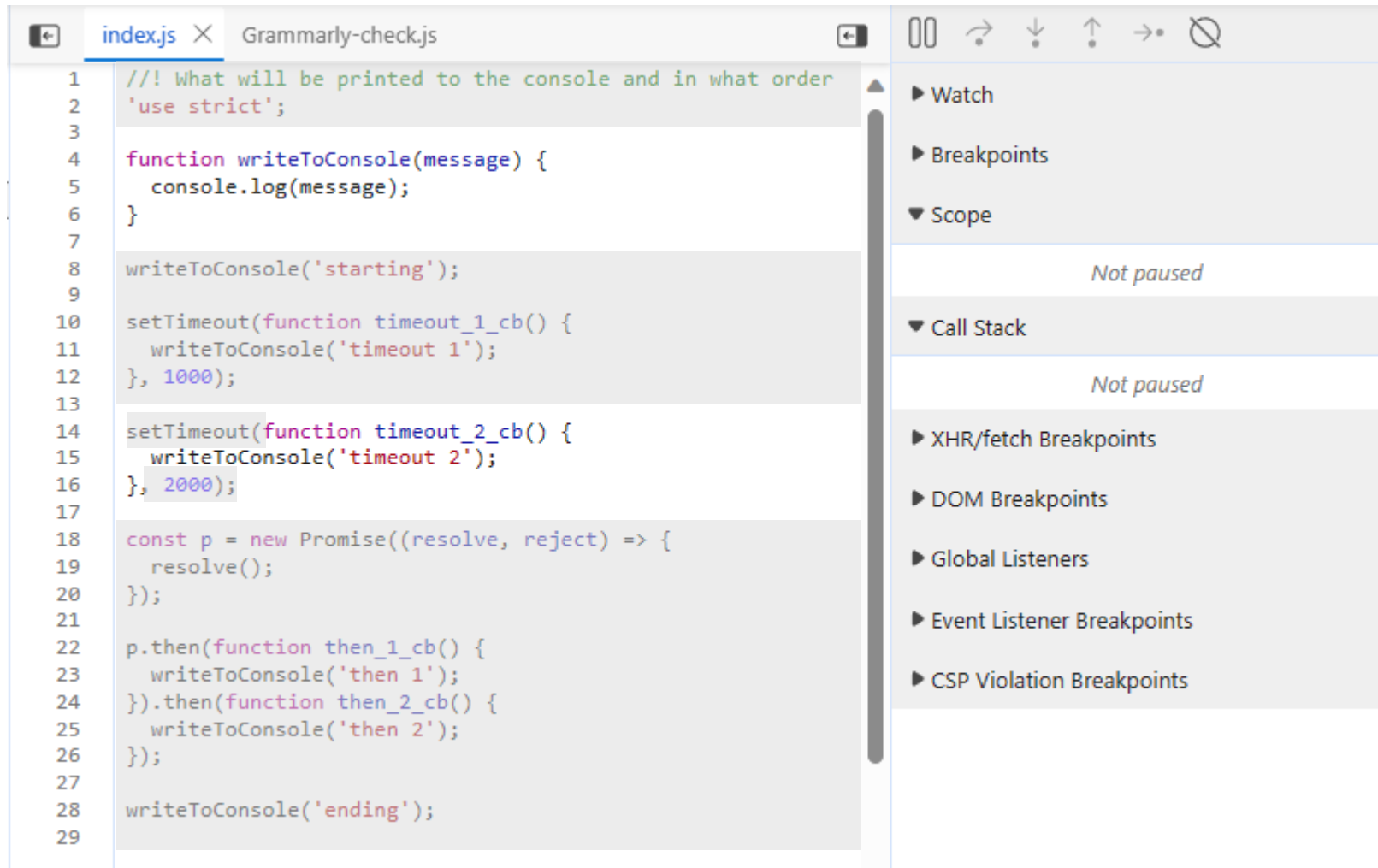
Debugger sidebar:

- Watch
- Breakpoints
- Scope
- Call Stack (Not paused)
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints
- CSP Violation Breakpoints



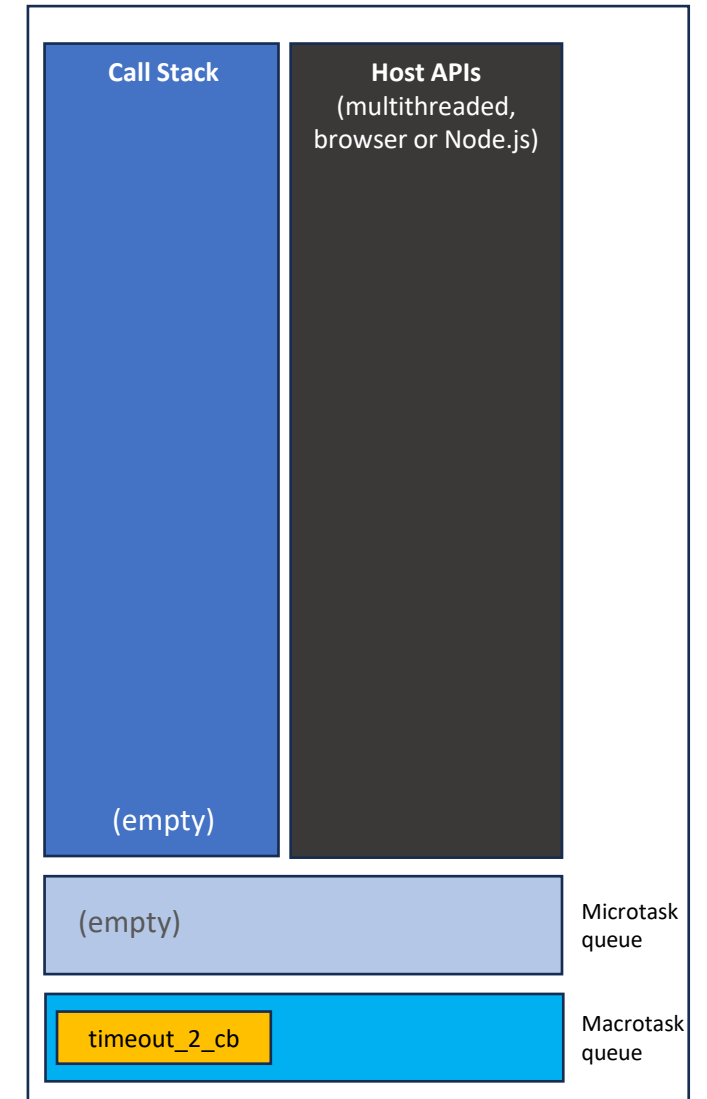


timeout\_2 fired



The screenshot shows a code editor with a file named `index.js`. The code defines a `writeToConsole` function and sets two timeouts. The first timeout, `timeout_1_cb`, is set for 1000ms and logs 'timeout 1'. The second timeout, `timeout_2_cb`, is set for 2000ms and logs 'timeout 2'. A Promise is also defined with two then callbacks, `then_1_cb` and `then_2_cb`, which log 'then 1' and 'then 2' respectively. The code ends with `writeToConsole('ending')`. The Chrome DevTools sidebar on the right shows the 'Call Stack' panel, which is currently empty, indicating that the script is not paused.

```
1  #!/ What will be printed to the console and in what order
2  'use strict';
3
4  function writeToConsole(message) {
5      console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11     writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15     writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19     resolve();
20 });
21
22 p.then(function then_1_cb() {
23     writeToConsole('then 1');
24 }).then(function then_2_cb() {
25     writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```



## Event 5: Macrotask timeout\_2\_callback

The code in `index.js` is as follows:

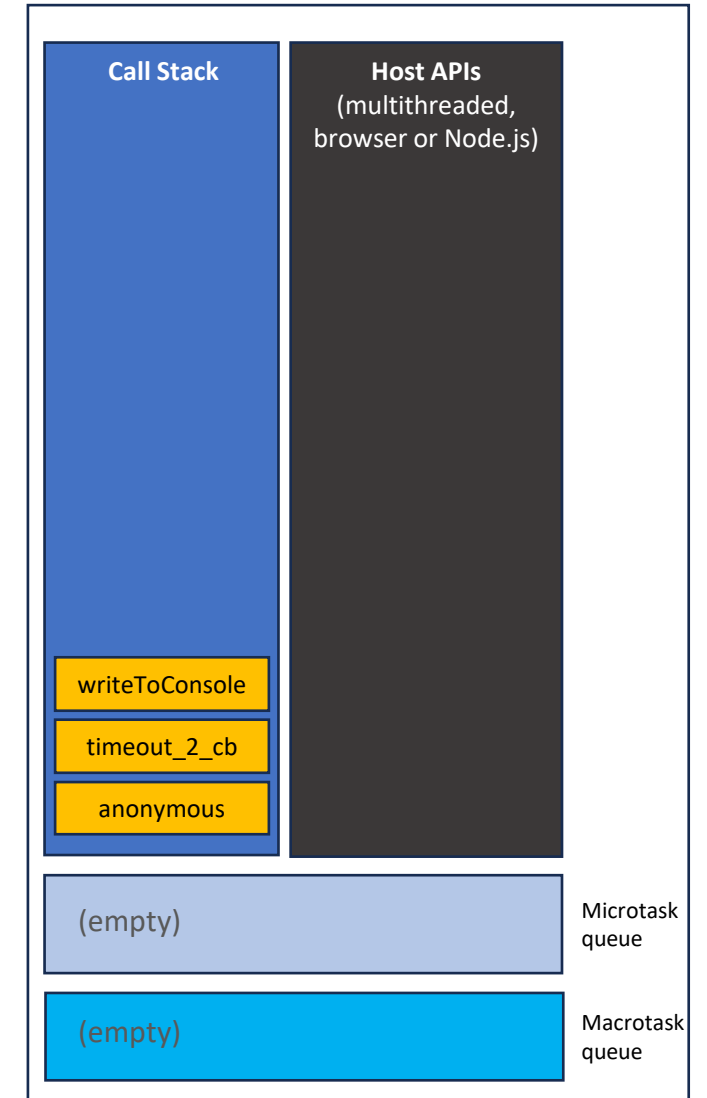
```
1  #!/usr/bin/env node
2  'use strict';
3
4  function writeToConsole(message) { message = "timeout 2"
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

The right sidebar shows the 'Paused on breakpoint' state. The scope window displays the following variables:

- Local
  - `this`: undefined
  - `message`: "timeout 2"
- Script
  - `p`: Promise {<fulfilled>: undefined}
- Global
  - Window

The call stack shows the following frames:

- `writeToConsole` (index.js:5)
- `timeout_2_cb` (index.js:15)
- `setTimeout (async)`
- `(anonymous)` (index.js:14)



Program finished

```
1  #!/ What will be printed to the console and in what order
2  'use strict';
3
4  function writeToConsole(message) {
5    console.log(message);
6  }
7
8  writeToConsole('starting');
9
10 setTimeout(function timeout_1_cb() {
11   writeToConsole('timeout 1');
12 }, 1000);
13
14 setTimeout(function timeout_2_cb() {
15   writeToConsole('timeout 2');
16 }, 2000);
17
18 const p = new Promise((resolve, reject) => {
19   resolve();
20 });
21
22 p.then(function then_1_cb() {
23   writeToConsole('then 1');
24 }).then(function then_2_cb() {
25   writeToConsole('then 2');
26 });
27
28 writeToConsole('ending');
29
```

Watch

Breakpoints

Scope

Not paused

Call Stack

Not paused

XHR/fetch Breakpoints

DOM Breakpoints

Global Listeners

Event Listener Breakpoints

CSP Violation Breakpoints

