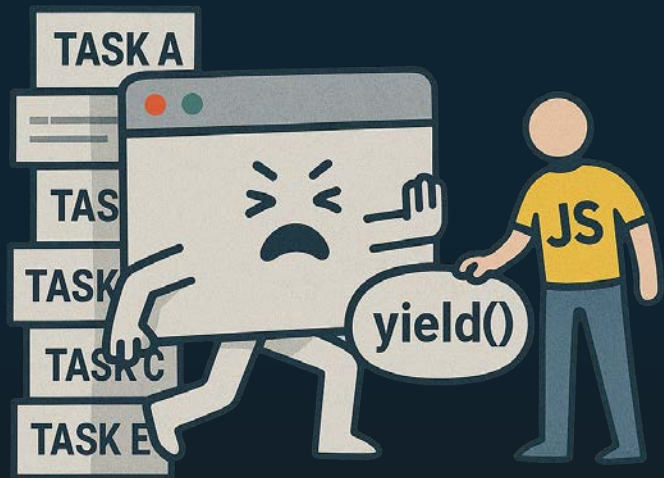


Let Your Browser Take a Breather



O L E K S A N D R T K A C H E N K O



ABOUT ME:

- Frontend developer with 6+ years of experience.
- Author of technical and scientific articles.
- Judge and mentor at international hackathons.
- Speaker at global conferences.
- Open-source contributor.
- Creator of the "Skeleton Mammoth" open-source CSS library.

What is
`scheduler.yield()`?



TERMINOLOGY

■ Main Thread

This is the central place where the browser does most of its work. It handles rendering, layout, and runs most of your JavaScript code.

■ Long task

This is any JavaScript task that keeps the **Main Thread** busy for too long – usually more than 50 milliseconds. When that happens, the page can freeze or feel unresponsive.

■ Blocking task

Is a synchronous operation on the **Main Thread** that prevents the browser from processing other important things, like responding to clicks or updating the UI. Usually, long tasks are blocking tasks.

TERMINOLOGY

■ Main Thread

This is the central place where the browser does most of its work. It handles rendering, layout, and runs most of your JavaScript code.

■ Long task

This is any JavaScript task that keeps the **Main Thread** busy for too long – usually more than 50 milliseconds. When that happens, the page can freeze or feel unresponsive.

■ Blocking task

Is a synchronous operation on the **Main Thread** that prevents the browser from processing other important things, like responding to clicks or updating the UI. Usually, long tasks are blocking tasks.

TERMINOLOGY

■ Main Thread

This is the central place where the browser does most of its work. It handles rendering, layout, and runs most of your JavaScript code.

■ Long task

This is any JavaScript task that keeps the **Main Thread** busy for too long – usually more than 50 milliseconds. When that happens, the page can freeze or feel unresponsive.

■ Blocking task

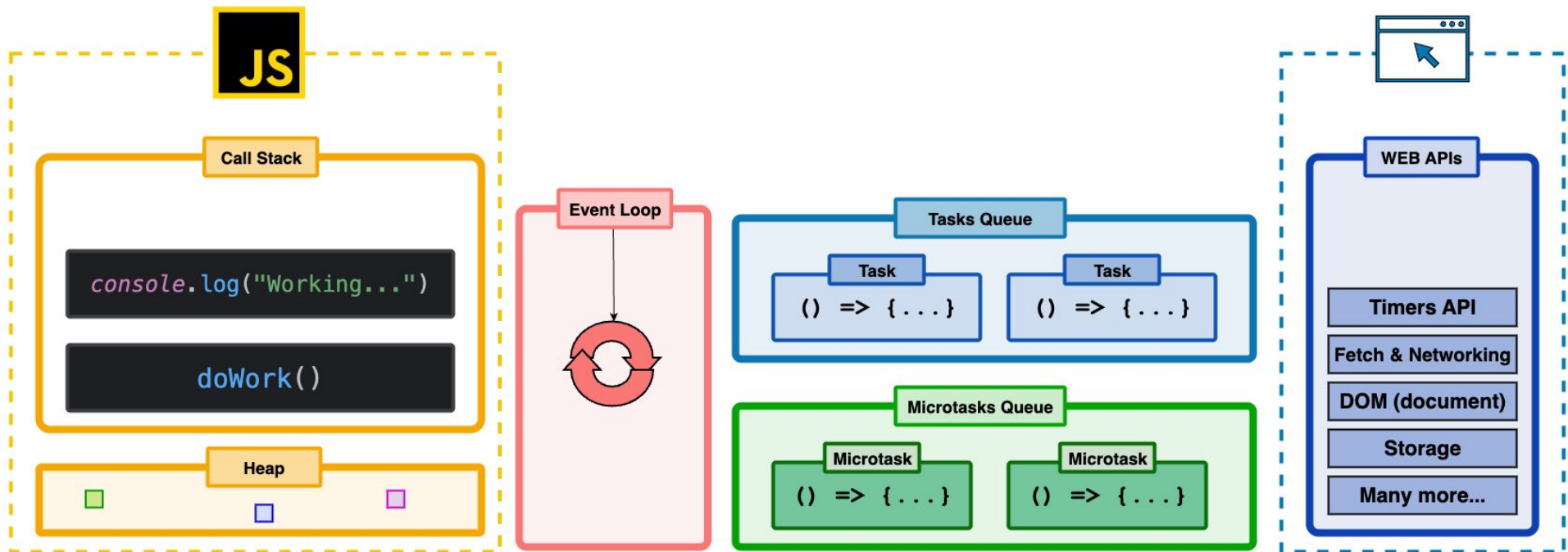
Is a synchronous operation on the **Main Thread** that prevents the browser from processing other important things, like responding to clicks or updating the UI. Usually, long tasks are blocking tasks.

The Problem

?



Task Processing in the Browser



The Problem

?



Description

blockingTask

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1  function blockingTask(ms = 50) {  
2      const arr = []  
3      const start = performance.now()  
4  
5      while ((performance.now() - start) < ms) {  
6          // Perform pointless computation to block the CPU.  
7          arr.unshift(Math.sqrt(Math.random()))  
8      }  
9  
10     return arr  
11 }
```

heavyWork

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1      function heavyWork () {
2          const data = Array.from({ length: 200 }, (_, i) => i)
3          const result = []
4
5          for (let i = 0; i < data.length; i++) {
6              result.push(blockingTask(10))
7          }
8
9          return result;
10     }
11
```

The Problem Demonstration

Configuration

Main Thread blocking:

☒ Disabled ☐ Enabled

Scheduler.yield():

☒ Disabled ☐ Enabled

Heavy task

Run a heavy task?

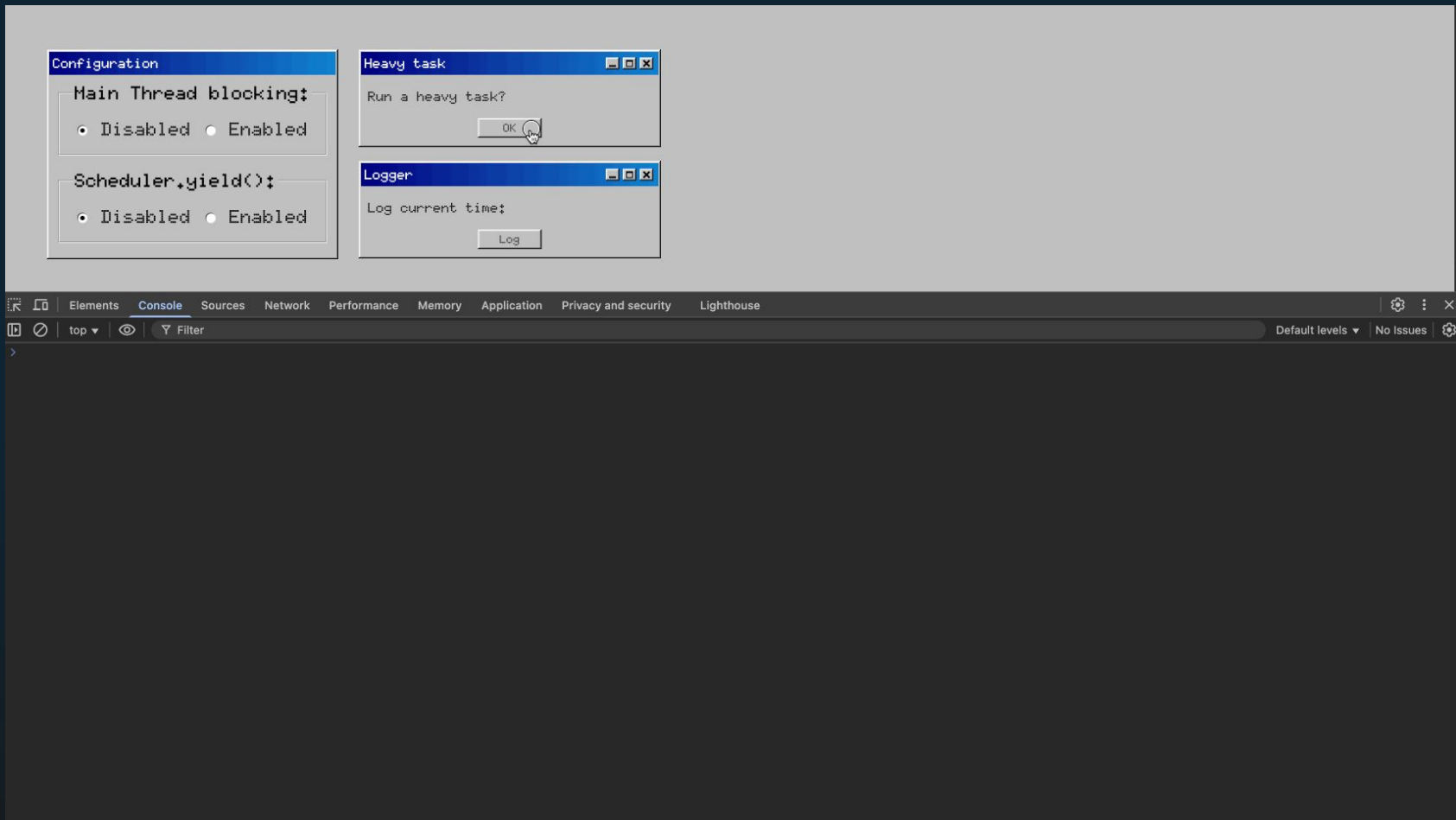
OK

Logger

Log current time:

Log

The Problem Demonstration



The Problem Demonstration

Configuration

Main Thread blocking:

☐ Disabled ☒ Enabled

Scheduler.yield():

☐ Disabled ☒ Enabled

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse

top Filter

Default levels No Issues

HEAVY_TASK_DONE [index.js:27](#)

[index.js:28](#)

(200) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...]

11/04/2025, 14:40:04.236 [index.js:41](#)

11/04/2025, 14:40:05.021 [index.js:41](#)

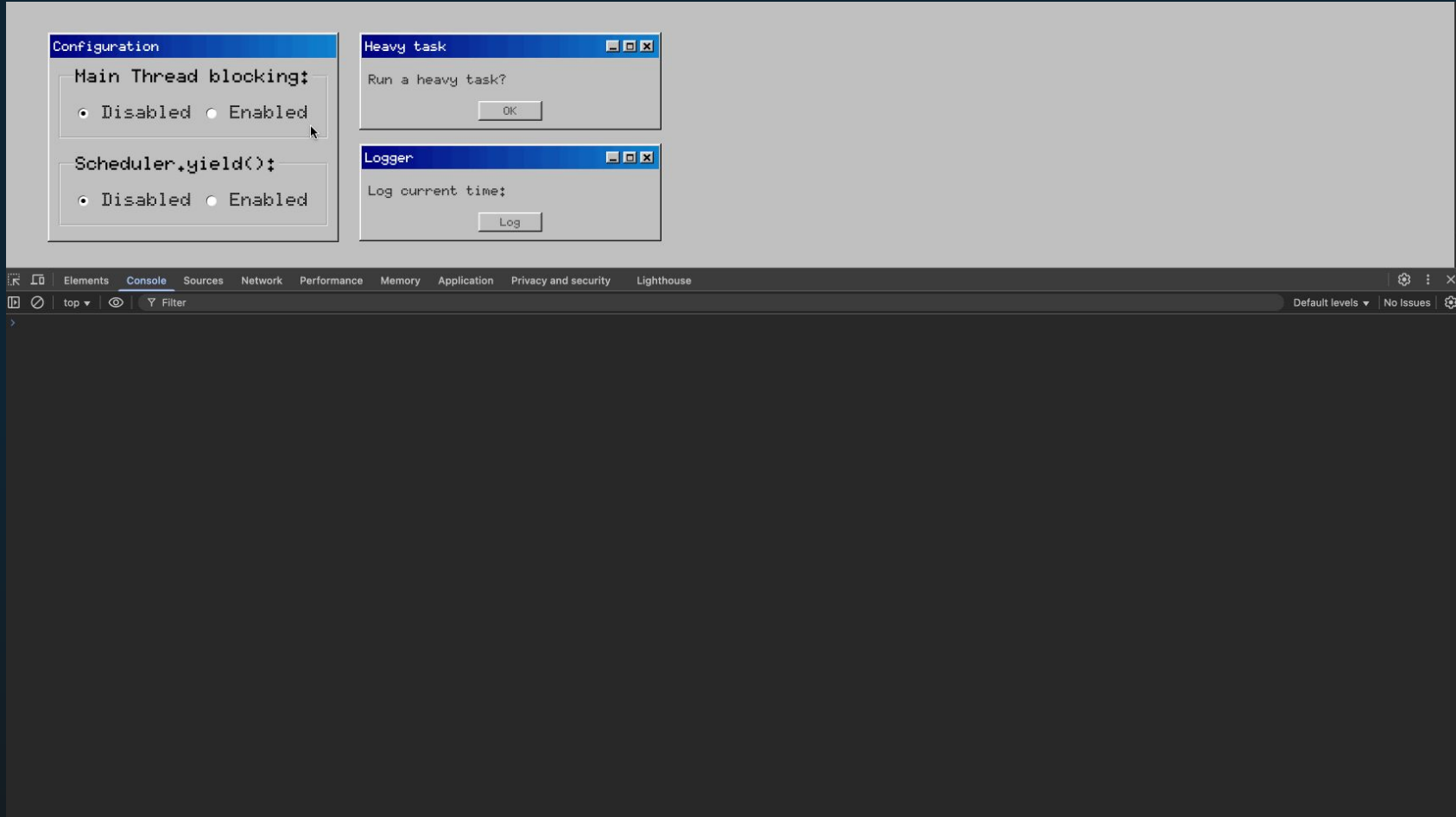
11/04/2025, 14:40:05.778 [index.js:41](#)

HEAVY_TASK_DONE [index.js:27](#)

[index.js:28](#)

(200) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...]

The Problem Demonstration



The Problem Solution



heavyWork - Splitted

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1  function heavyWork() {  
2    // Do heavy work...  
3  
4    /**  
5     * Take a breather!  
6     * Yield the execution to the Main Thread...  
7     * */  
8  
9    // Continue to do heavy work...  
10 }  
11
```

heavyWork - Splitted

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1      function heavyWork() {  
2          // Do heavy work...  
3  
4          /**  
5              * Take a breather!  
6              * Yield the execution to the Main Thread...  
7              * */  
8  
9          // Continue to do heavy work...  
10     }  
11
```

heavyWork - Splitted

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1      function heavyWork() {  
2          // Do heavy work...  
3  
4          /**  
5           * Take a breather!  
6           * Yield the execution to the Main Thread...  
7           * */  
8  
9          // Continue to do heavy work...  
10     }  
11
```

heavyWork - Splitted

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1      function heavyWork() {
2          // Do heavy work...
3
4          /**
5           * Take a breather!
6           * Yield the execution to the Main Thread...
7           * */
8
9          // Continue to do heavy work...
10     }
11
```

Old Problem-Solving Approaches



heavyWork - setTimeout()

Project ▾

> public

> src

> assets

> components

JS App.js

JS Header.js

JS Footer.js

```
1  async function heavyWork() {
2    // Yield to Main Thread to avoid UI blocking before heavy work
3    await new Promise(resolve => setTimeout(resolve, 0))
4
5    const data = Array.from({ length: 200 }, (_, i) => i)
6    const result = []
7
8    // Interval at which execution will be yielded to the main thread (approx. ~ 25%).
9    const yieldInterval = Math.ceil(data.length / 4)
10
11    for (let i = 0; i < data.length; i++) {
12      // Yield control to Main Thread to update UI and handle other tasks.
13      if (i % yieldInterval === 0) {
14        await new Promise(resolve => setTimeout(resolve, 0))
15      }
16
17      result.push(threadBlockingEnabled ? blockingTask(10) : data[i])
18    }
19
20    return result
21  }
```

heavyWork - setTimeout()

Project ▾

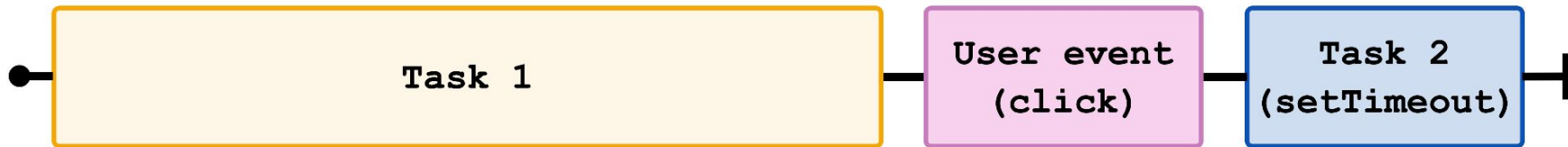
- > public
 - > src
 - > assets
 - > components
 - JS App.js
 - JS Header.js
 - JS Footer.js

```
1  setInterval(() => { /* Another heavy work... */ })
2
3  async function heavyWork() {
4    // Yield to Main Thread to avoid UI blocking before heavy work
5    await new Promise(resolve => setTimeout(resolve, 0))
6
7    const data = Array.from({ length: 200 }, (_, i) => i)
8    const result = []
9
10   // Interval at which execution will be yielded to the main thread (approx. ~ 25%).
11   const yieldInterval = Math.ceil(data.length / 4)
12
13   for (let i = 0; i < data.length; i++) {
14     // Yield control to Main Thread to update UI and handle other tasks.
15     if (i % yieldInterval === 0) {
16       await new Promise((resolve, reject) => setTimeout(resolve, 0))
17     }
18
19     result.push(threadBlockingEnabled ? blockingTask(10) : data[i])
20   }
21
22   return result
23 }
```

Scheduler.yield()



No yielding



`new Promise(r => setTimeout(r, 0))`



`await scheduler.yield()`



No yielding



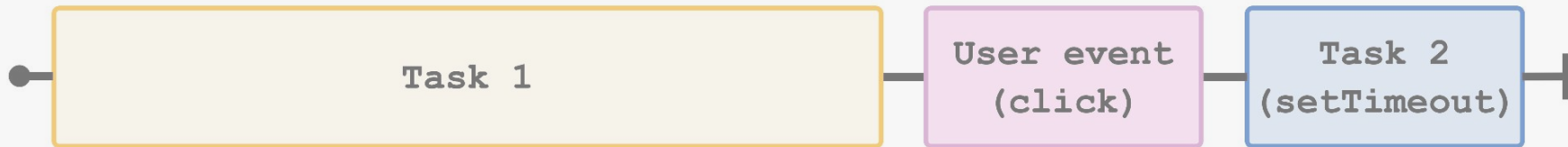
`new Promise(r => setTimeout(r, 0))`



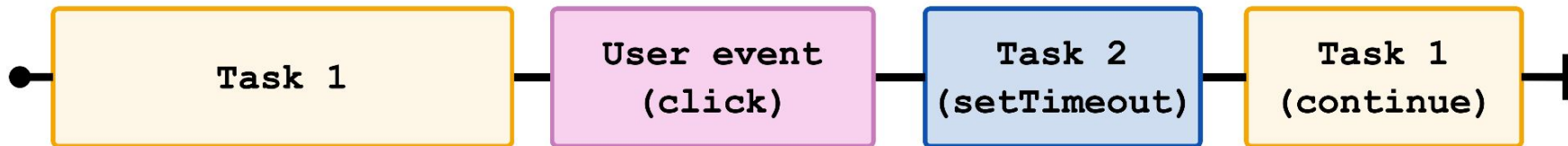
`await scheduler.yield()`



No yielding



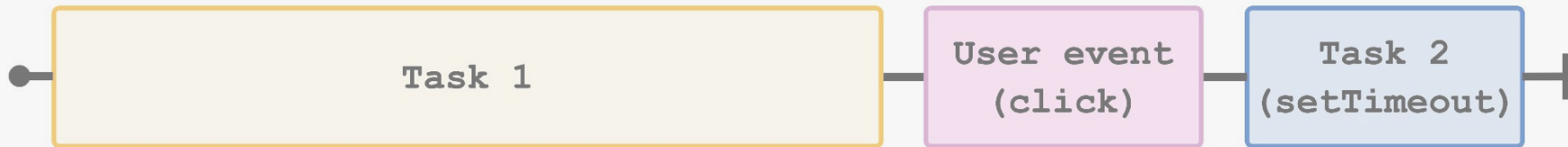
`new Promise(r => setTimeout(r, 0))`



`await scheduler.yield()`



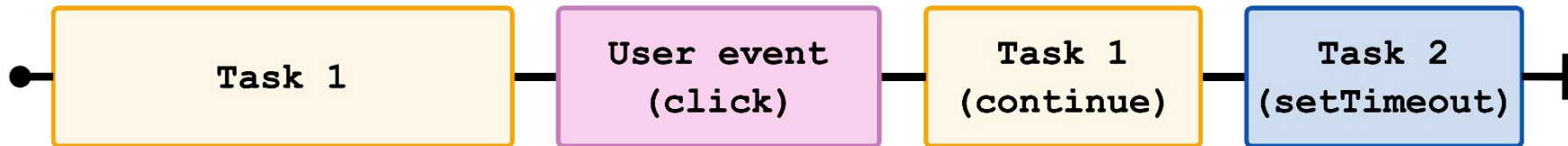
No yielding



`new Promise(r => setTimeout(r, 0))`



`await scheduler.yield()`



Priorities



Priorities

1

"user-blocking"

The highest priority tasks that directly affect user interaction, such as handling clicks, taps, and critical UI operations.

Priorities

1

"user-blocking"

The highest priority tasks that directly affect user interaction, such as handling clicks, taps, and critical UI operations.

2

"user-visible"

Tasks that affect UI visibility or content, but are not critical for input.

Priorities

1

"user-blocking"

The highest priority tasks that directly affect user interaction, such as handling clicks, taps, and critical UI operations.

2

"user-visible"

Tasks that affect UI visibility or content, but are not critical for input.

3

"background"

Tasks that can be safely postponed without affecting the current user experience, and are not visible to the user.

How to use `Scheduler.yield()` ?



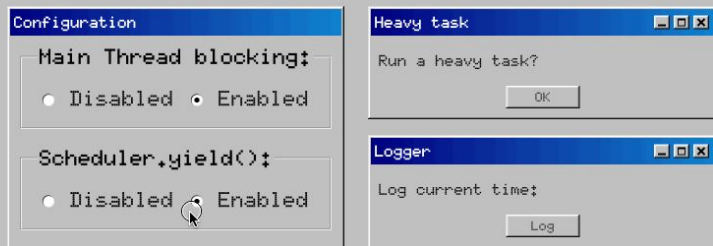
heavyWork - scheduler.yield()

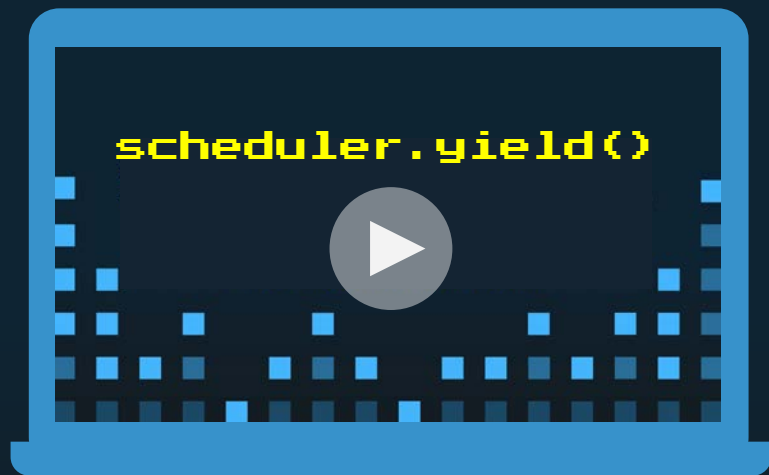
Project ▾

- > public
 - > src
 - > assets
 - > components
 - JS App.js
 - JS Header.js
 - JS Footer.js

```
1  async function heavyWork() {
2    // Yield to Main Thread to avoid UI blocking before heavy work
3    await scheduler.yield()
4
5    const data = Array.from({ length: 200 }, (_, i) => i)
6    const result = []
7
8    // Interval at which execution will be yielded to the main thread (approx. ~ 25%).
9    const yieldInterval = Math.ceil(data.length / 4)
10
11    for (let i = 0; i < data.length; i++) {
12      // Yield control to Main Thread to update UI and handle other tasks.
13      if (i % yieldInterval === 0) {
14        await scheduler.yield()
15      }
16
17      result.push(threadBlockingEnabled ? blockingTask(10) : data[i])
18    }
19
20    return result
21  }
22
23
```

How to use Scheduler.yield() ?





DEMO

Demo

Let Your Browser Take a Breather

Demo simple

Demo full

Configuration

Main Thread blocking:

☐ Disabled ☒ Enabled

Scheduler.yield():

☐ Disabled ☒ Enabled

Data array length:

200

Blocking time duration (ms):

10

Yield interval (%):

25

Balance:

.....

Show balance

Credit



.... - - - 4444

TKACHENKO OLEKSANDR

VISA

Show card details

Generate report

Date	Amount	Description

Demo

Let Your Browser Take a Breather

Demo simple

Demo full

Configuration

```
Main Thread blocking:
```

☒ Disabled ☐ Enabled

```
Scheduler.yield();
```

☒ Disabled ☐ Enabled

```
Data array length:
```

A horizontal number line with a tick mark at 200.

Blocking time duration (ms):

10

Yield interval (%):

25

Balance:

• • • • •

Show balance

Credit



***** - ***** - ***** - 4444

TKACHENKO OLEKSANDR

[Show card details](#)

Generate report

[illegible]

Demo

Let Your Browser Take a Breather

Demo simple

Demo full

Configuration

```
Main Thread blocking:
```

☒ Disabled ☐ Enabled

```
Scheduler.yield();
```

☒ Disabled ☐ Enabled

```
Data array length:
```

A horizontal number line with a tick mark at 200.

Blocking time duration (ms):

10

Yield interval (%):

25

Balance:

• • • • •

Show balance

Credit



***** - ***** - ***** - 4444

TKACHENKO OLEKSANDR

[Show card details](#)

Generate report

[illegible]

Demo

Let Your Browser Take a Breather

Demo simple

Demo full

Configuration

```
Main Thread blocking:
```

☒ Disabled ☐ Enabled

```
Scheduler.yield();
```

☒ Disabled ☐ Enabled

```
Data array length:
```

A horizontal number line with a tick mark at 200.

Blocking time duration (ms):

10

Yield interval (%):

25

Balance:

◆ ◆ ◆ ◆ ◆ ◆ ◆

Show balance

Credit



***** - ***** - ***** - 4444

TKACHENKO OLEKSANDR

[Show card details](#)

Generate report

[illegible]

Demo

Configuration

Main Thread blocking:

☐ Disabled ☒ Enabled

Scheduler.yield():

☐ Disabled ☒ Enabled

Data array length:

200

Blocking time duration (ms):

10

Yield interval (%):

25

Demo

Let Your Browser Take a Breathe

Demo simple

Demo full

Configuration

Main Thread blocking:

☐ Disabled ☒ Enabled

Scheduler.yield():

☐ Disabled ☒ Enabled

Data array length:

1000

Blocking time duration (ms):

10

Yield interval (%):

25

Balance:

.....

Show balance

Credit



.... - - - 4444

TKACHENKO OLEKSANDR

VISA

Show card details

Generate report

Date	Amount	Description

Demo

Configuration

Main Thread blocking:

☐ Disabled ☒ Enabled

Scheduler.yield():

☐ Disabled ☒ Enabled

Data array length:

1300

Blocking time duration (ms):

10

Yield interval (%):

5

Demo

Let Your Browser Take a Breather

Demo simple

Demo full

Configuration

```
Main Thread blocking:
```

☐ Disabled ☒ Enabled

```
Scheduler.yield();
```

☐ Disabled ☒ Enabled

Data array length:

—|— 1300

Blocking time duration (ms):

10

Yield interval (%):

— 5

Balance:

• • • • •

Show balance

Credit



••••• - ••••• - ••••• - 4444

TKACHENKO OLEKSANDR

[Show card details](#)

Generate report

[illegible]

THANKS

Scan for useful links

