threads no node.js com require(‘worker_threads’)

leonardo brito bittencourt

2021

agenda

- event loop
- qual problema ele resolve?
- quando usar?
- quando não usar?
- live code

leonardo brito bittencourt

2021

Threads no Node.js com

require(‘worker_threads’)

event loop

stack

background threads (libuv)

```
3  function main() {
4    // ...
5
6    const path = __dirname + '/users.json';
7    const options = { encoding: 'utf-8' };
8
9    fs.readFile(path, options, (err, data) => {
10     // ...
11   });
12
13   math()
14
15   // ...
16 }
```

readFile()

task queue

leonardo brito bittencourt

2021

Threads no Node.js com

require(‘worker_threads’)

event loop

stack

readFile()

```
3  function main() {
4    // ...
5
6    const path = __dirname + '/users.json';
7    const options = { encoding: 'utf-8' };
8
9    fs.readFile(path, options, (err, data) => {
10     // ...
11   });
12
13   math()
14
15   // ...
16 }
```

background threads (libuv)

task queue

leonardo brito bittencourt

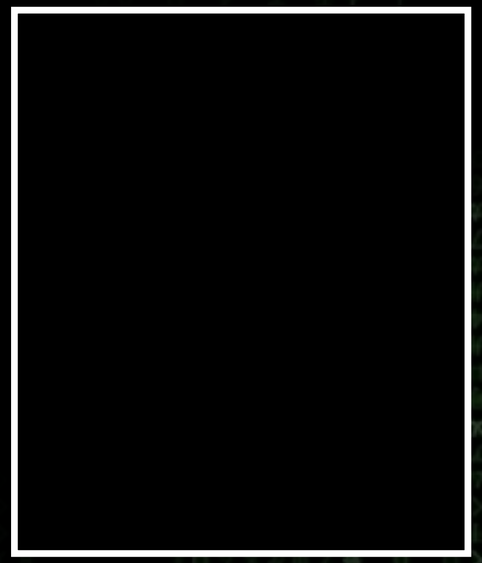
2021

Threads no Node.js com

require('worker_threads')

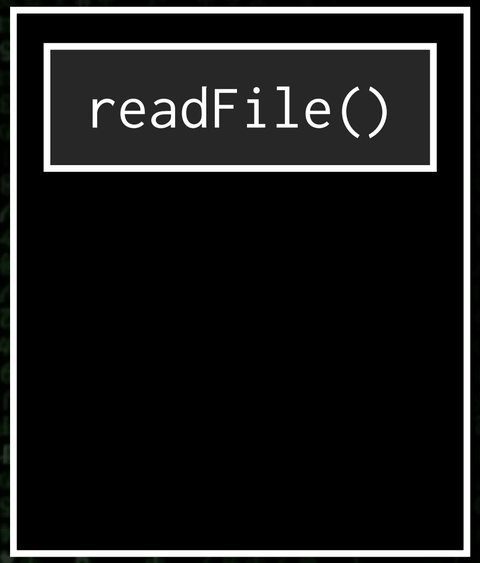
event loop

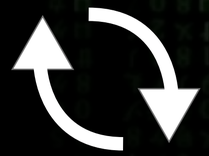
stack



```
3  function main() {
4    // ...
5
6    const path = __dirname + '/users.json';
7    const options = { encoding: 'utf-8' };
8
9    fs.readFile(path, options, (err, data) => {
10     // ...
11   });
12
13   math()
14
15   // ...
16 }
```

background threads (libuv)





task queue

leonardo brito bittencourt

2021

Threads no Node.js com

require(‘worker_threads’)

event loop

stack

background threads (libuv)

```
3  function main() {
4    // ...
5
6    const path = __dirname + '/users.json';
7    const options = { encoding: 'utf-8' };
8
9    fs.readFile(path, options, (err, data) => {
10     // ...
11   });
12
13   math()
14
15   // ...
16 }
```

readFile()

math()

task queue

leonardo brito bittencourt

2021

Threads no Node.js com
require(‘worker_threads’)

event loop

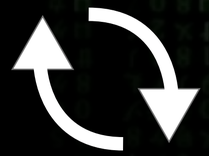
stack

math()

```
3   function main() {
4     // ...
5
6     const path = __dirname + '/users.json';
7     const options = { encoding: 'utf-8' };
8
9     fs.readFile(path, options, (err, data) => {
10      // ...
11    });
12
13    math()
14
15    // ...
16  }
```

background threads (libuv)

readFile()



task queue

leonardo brito bittencourt

2021

Threads no Node.js com
require(‘worker_threads’)

event loop

stack

Função síncrona

math()

```
3   function main() {
4     // ...
5
6     const path = __dirname + '/users.json';
7     const options = { encoding: 'utf-8' };
8
9     fs.readFile(path, options, (err, data) => {
10      // ...
11    });
12
13    math()
14
15    // ...
16  }
```

background threads (libuv)

readFile()

readFileCallback() readFileCallback() ...

task queue

leonardo brito bittencourt

2021

qual problema ele resolve?

funções síncronas que demanda muita CPU.

o worker_thread irá rodar de maneira paralela
funções síncronas sem “travar” a Stack

quando usar?

- cálculos matemáticos muito pesados
- funções de analytics
- funções de machine learning
- parse de json gigantescos
- parses e compilers (typescript)
- test runners (jest)

leonardo brito bittencourt

2021

quando não usar?

o node.js trabalha muito bem com I/O através da libuv (background threads), ou seja, é muito custoso e não há necessidade de worker_threads para executar funções como:

- ler arquivos de disco
- requests HTTP
- …

Threads no Node.js com

require(‘worker_threads’)

live code

time

leonardo brito bittencourt

2021

dúvidas?

github: leobritob
leonardobritobittencourt@gmail.com