Using Web Workers With React

By Chen Feldman
Frontend Architect

Vamos

```
self.file = None
self.fingerprints =
self.logdupes = True
self.debug = debug
self.logger = logging.
if path:
    self.file = open(os.path.
    self.file.seek(0)
    self.fingerprints.update(

@classmethod
def from_settings(cls, settings):
    debug = settings.getbool('DUPEFILTER
    return cls(job_dir(settings),

def request_seen(self, request):
    fp = self.request_fingerprint(req
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp +

def request_fingerprint(
    return request
```

Vamos

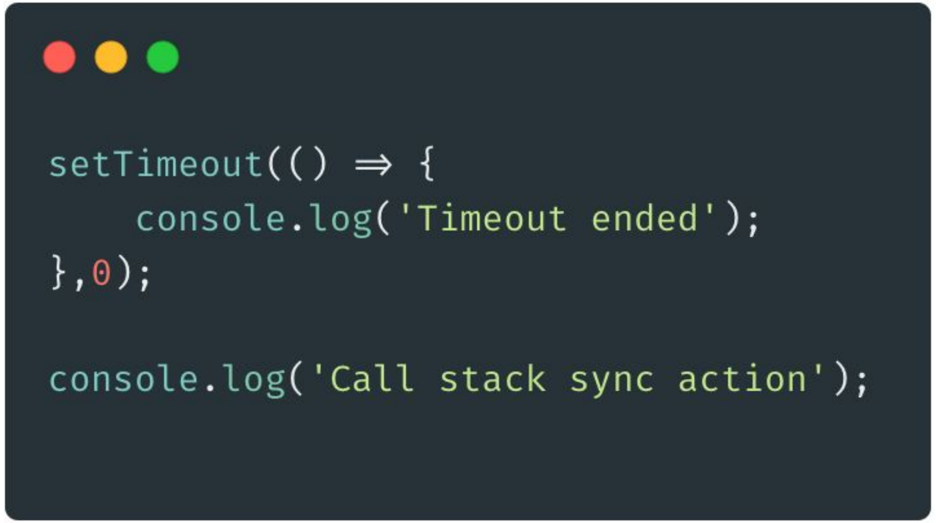
AHA!
MOMENT

{עושים תוכנה}

Lecture’s Timeline

- Asynchronous Programming != Parallel
- Service vs Web Workers
- Live Demo Web Workers

What will be the output?

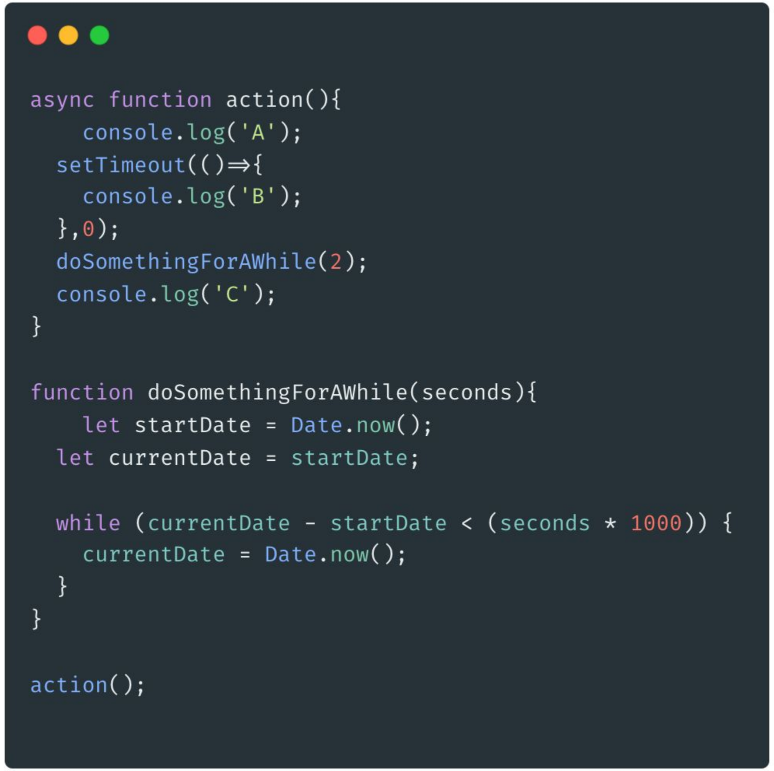


```
setTimeout(() => {
    console.log('Timeout ended');
},0);

console.log('Call stack sync action');
```

The Output

```
Call stack sync action
Timeout ended
```

So again, what will be the output?

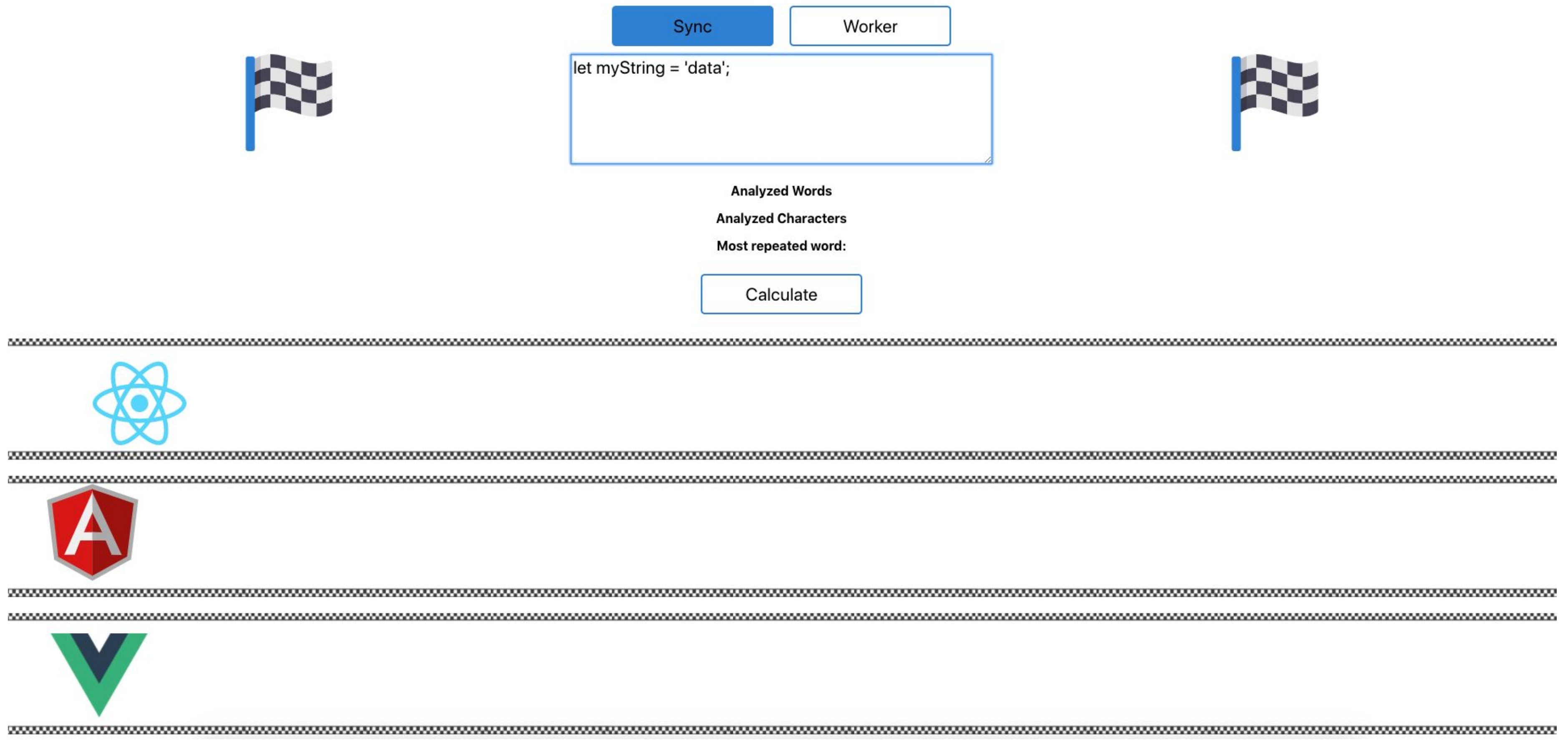


```
async function action(){
    console.log('A');
  setTimeout(()=>{
    console.log('B');
  },0);
  doSomethingForAWhile(2);
  console.log('C');
}

function doSomethingForAWhile(seconds){
    let startDate = Date.now();
  let currentDate = startDate;

  while (currentDate - startDate < (seconds * 1000)) {
    currentDate = Date.now();
  }
}

action();
```

The Output

```
A

C

B
```

Live Demo - Using Async Abilities



```
let myString = 'data';
```

Live Demo

Let’s solve it using promise!!!

Recommended Using Promise

- Network calls (server, API)
- I/O operations

Promises Will Not Perform Better

- Image processing
- Heavy UI calculations & logics
- Running algorithms inside the main thread

Asynchronous Does Not Mean Parallel!

Let’s Use Workers Instead!

Workers - What Are They?

Script that runs on a thread separate to the browser’s main thread

From :https://bitsofco.de/web-workers-vs-service-workers-vs-worklets/

Represents a background task that can be created via script

From :https://developer.mozilla.org/en-US/docs/Web/API/Worker

Workers - What Are They?

Script that runs on a thread separate to the browser’s main thread

From :https://bitsofco.de/web-workers-vs-service-workers-vs-worklets/

Represents a background task that can be created via script

From :https://developer.mozilla.org/en-US/docs/Web/API/Worker

Workers Types - Service Worker

- A proxy between the browser and the network/cache
- Are able to intercept any network request from the main document
- Use case :

working offline (PWA) returning from cache instead of network

Workers Types - Service Worker

- A proxy between the browser and the network/cache
- Are able to intercept any network request from the main document
- Use case :

  working offline (PWA) returning from cache instead of network

Workers Types - Service Worker

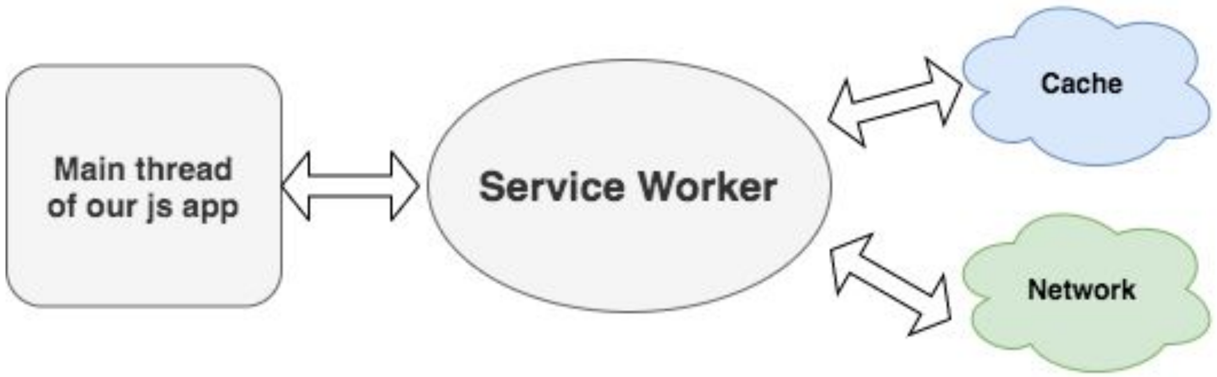
- A proxy between the browser and the network/cache
- Are able to intercept any network request from the main document
- Use case :

  working offline (PWA) returning from cache instead of network

Workers Types - Service Worker

- A proxy between the browser and the network/cache
- Are able to intercept any network request from the main document
- Use case :

working offline (PWA) returning from cache instead of network



Workers Types - Web Worker

- Dedicated (specific web page) vs Shared worker (shared between pages)
- Can run separately from the main thread in it own thread

Common use cases: spell checking, code analyzing, image processing...

Workers Types - Web Worker

- Dedicated (specific web page) vs Shared worker (shared between pages)
- Can run separately from the main thread in it own thread

Common use cases: spell checking, code analyzing, image processing...

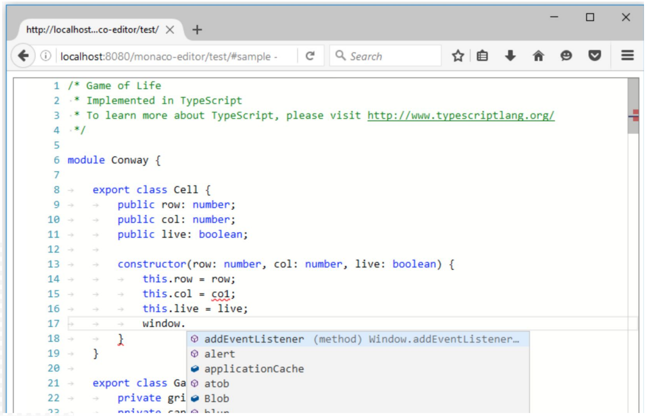
FYI - Products Who Uses Web Workers

- Monaco Editor (VS Code Online)

  https://dev.decoupled.com/docs-magic-webWorker-example-monaco

- Google Squoosh app

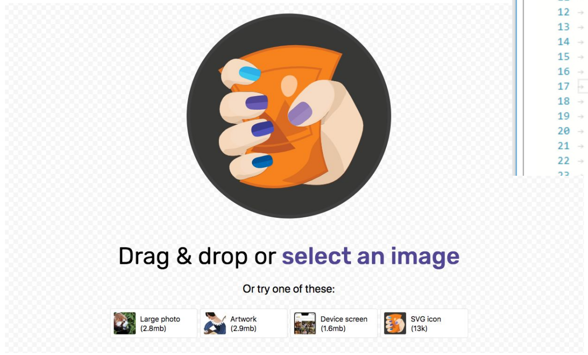
  https://squoosh.app/



http://localhost...co-editor/test/
localhost:8080/monaco-editor/test/#sample -
Search

```
/* Game of Life
 * Implemented in TypeScript
 * To learn more about TypeScript, please visit http://www.typescriptlang.org/
 */

module Conway {

    export class Cell {
        public row: number;
        public col: number;
        public live: boolean;

        constructor(row: number, col: number, live: boolean) {
            this.row = row;
            this.col = col;
            this.live = live;
            window.
        }
    }

    export class Ga
        private gri
```

addEventListener (method) Window.addEventListener…
alert
applicationCache
atob
Blob



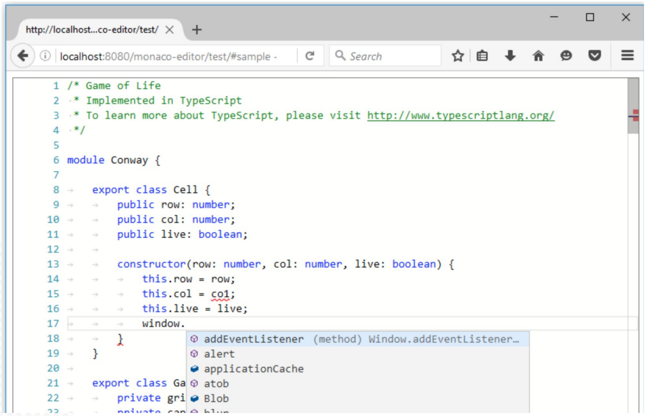
FYI - Products Who Uses Web Workers

- Monaco Editor (VS Code Online)

  https://dev.decoupled.com/docs-magic-webWorker-example-monaco

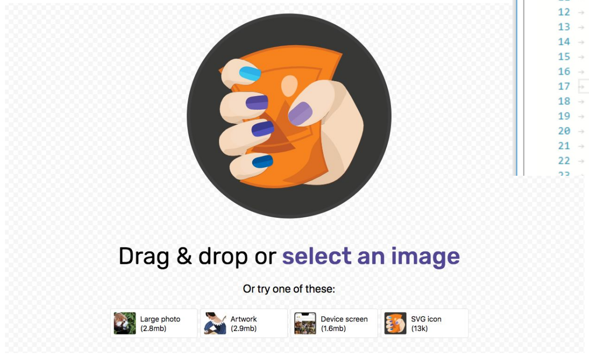
- Google Squoosh app

  https://squoosh.app/


http://localhost...co-editor/test/
localhost:8080/monaco-editor/test/#sample -
Search

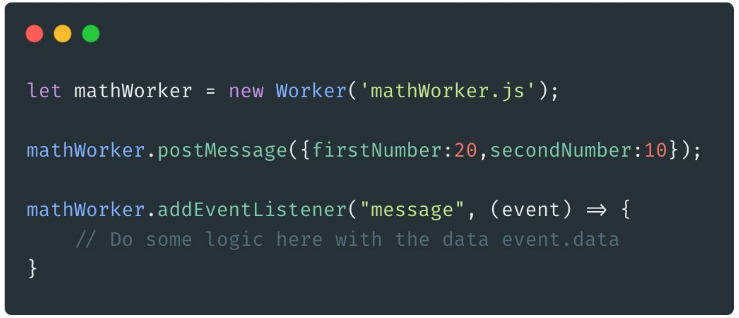
```
/* Game of Life
 * Implemented in TypeScript
 * To learn more about TypeScript, please visit http://www.typescriptlang.org/
 */

module Conway {

    export class Cell {
        public row: number;
        public col: number;
        public live: boolean;

        constructor(row: number, col: number, live: boolean) {
            this.row = row;
            this.col = col;
            this.live = live;
            window.
        }
    }

    export class Ga
        private gri
```

addEventListener (method) Window.addEventListener…
alert
applicationCache
atob
Blob



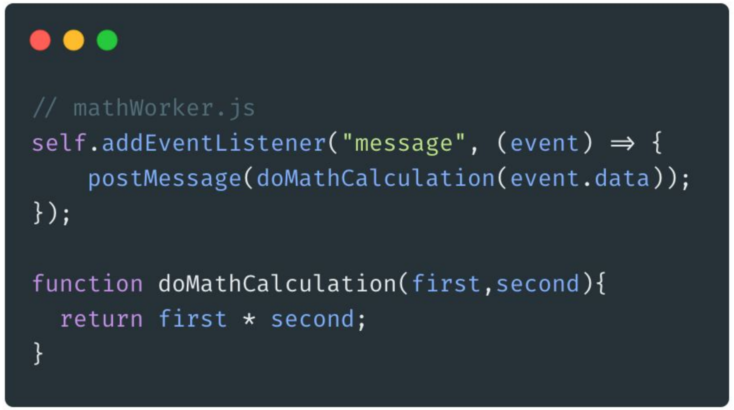
BUT I DON'T KNOW
HOW IT WORKS.

Main Thread (main.js)

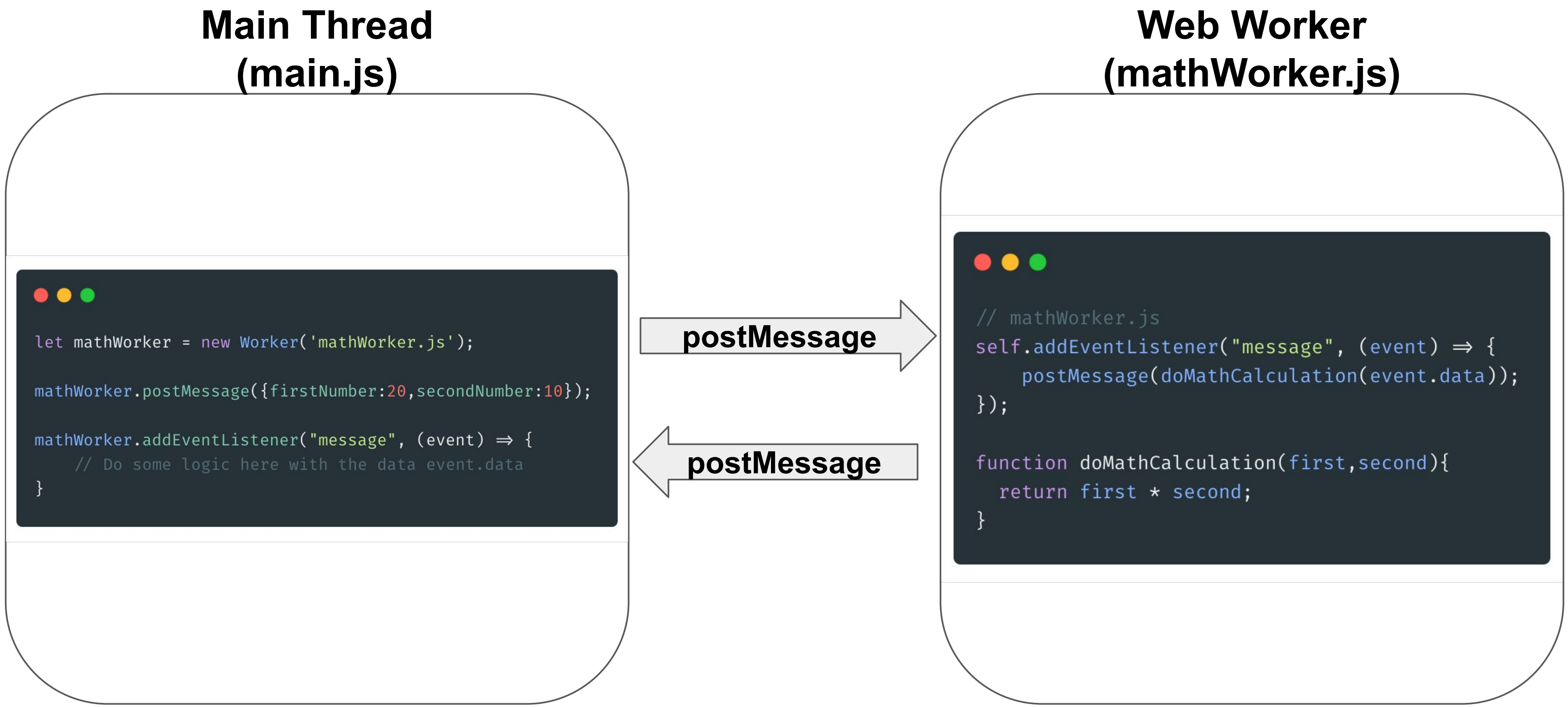


```
let mathWorker = new Worker('mathWorker.js');

mathWorker.postMessage({firstNumber:20,secondNumber:10});

mathWorker.addEventListener("message", (event) ⇒ {
    // Do some logic here with the data event.data
}
```

Web Worker (mathWorker.js)



```
// mathWorker.js
self.addEventListener("message", (event) => {
    postMessage(doMathCalculation(event.data));
});

function doMathCalculation(first,second){
  return first * second;
}
```



```
let mathWorker = new Worker('mathWorker.js');

mathWorker.postMessage({firstNumber:20,secondNumber:10});

mathWorker.addEventListener("message", (event) ⇒ {
    // Do some logic here with the data event.data
}
```

```
// mathWorker.js
self.addEventListener("message", (event) ⇒ {
    postMessage(doMathCalculation(event.data));
});

function doMathCalculation(first,second){
  return first * second;
}
```

How It Works - Communication

- Event-based communication
- postMessage - data is **cloned** and **not shared** between the two
- Main thread can also terminate the worker - mathWorker.terminate()

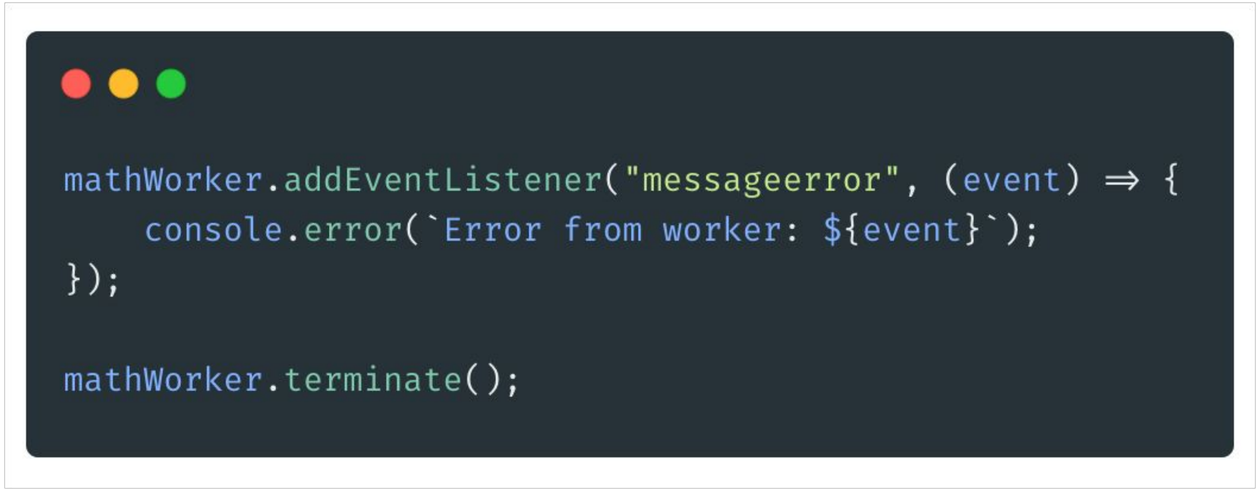
How It Works - Communication

- Event-based communication
- postMessage - data is **cloned** and **not shared** between the two
- Main thread can also terminate the worker - mathWorker.terminate()

How It Works - Communication

- Event-based communication
- postMessage - data is **cloned** and **not shared** between the two
- Main thread can also terminate the worker - mathWorker.terminate()

How It Works - Communication



```javascript
mathWorker.addEventListener("messageerror", (event) ⇒ {
    console.error(`Error from worker: ${event}`);
});

mathWorker.terminate();
```

@LateNightSeth
WAIT, WHAT?

Live Demo - Our Demo Using Workers

```
▼<body>
  ▼<div id="app">
    ▼<todo-application>
      ▼#shadow-root (open)
          <link rel="stylesheet" type="text/css" href="//
          maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/
          bootstrap.min.css">
        ▶<style>…</style>
        ▶<nav class="navbar navbar-expand-md navbar-dark bg-dark">…</nav>
        ▼<main class="container">
          ▼<todo-form>
            ▶<style>…</style>
            ▶<div class="card todo-form">…</div>
          </todo-form>
          <hr>
          ▼<todo-list ref="list">
            ▶<style>…</style>
             <h2>Tasks:</h2>
            ▼<ul ref="todos" class="list-group">
              ▶<todo-task ref="task-1517176192142" id="task-1517176192142">
              …</todo-task> == $0
              ▶<todo-task ref="task-1517176320397" id="task-1517176320397">
              …</todo-task>
              ▶<todo-task ref="task-1517176329096" id="task-1517176329096">
              …</todo-task>
              ▶<todo-task ref="task-1517176334849" id="task-
              …</todo-task>
            </ul>
```

```
Filter                :hov
element.style {
}
*,                    _reboot
::after,
::before {
  box-sizing: bord
    box;
}
Inherited from ul.list
ul,  user agent styl
menu
  display: block;
  list-style-type:
  -webkit-margin-be
    1em;
  -webkit-margin-af
    1em;
  -webkit-margin-sta
    0px;
  -webkit-margin-end
    0px;
  -webkit-padding-st
    40px;
```

```
▼<div id="app">
  ▼<todo-application>
    ▼#shadow-root (open)
      <link rel="stylesheet" type="text/css" href="//
      maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/
      bootstrap.min.css">
      ▶<style>…</style>
      ▶<nav class="navbar navbar-expand-md navbar-dark bg-dark">…</nav>
      ▼<main class="container">
        ▼<todo-form>
          ▶<style>…</style>
          ▶<div class="card todo-form">…</div>
        </todo-form>
        <hr>
      ▼<todo-list ref="list">
        ▶<style>…</style>
        <h2>Tasks:</h2>
        ▼<ul ref="todos" class="list-group">
          ▶<todo-task ref="task-1517176192142" id="task-1517176192142">
          …</todo-task> == $0
          ▶<todo-task ref="task-1517176320397" id="task-1517176320397">
          …</todo-task>
          ▶<todo-task ref="task-1517176329096" id="task-1517176329096">
          …</todo-task>
          ▶<todo-task ref="task-1517176334849" id="task-
          …</todo-task>
        </ul>
```

Live Demo - Debugging

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Advantages

- New working thread - working on background
- Not blocking the UI
- Simple communication (event-based)
- Split responsibility between workers for even better performance
- Can communicate with server (not every library)
- Can be terminated in any given moment by main thread

Disadvantages

- Cannot communicate with the DOM
  - Cannot load images or create canvas elements
- Limited access to functions and properties inside the window object
- Communication and passing data has some price when done too much
  - You cannot control the context switching behind the scenes

Disadvantages

- Cannot communicate with the DOM
  - Cannot load images or create canvas elements
- Limited access to functions and properties inside the window object
- Communication and passing data has some price when done too much
  - You cannot control the context switching behind the scenes

Disadvantages

- Cannot communicate with the DOM
  - Cannot load images or create canvas elements
- Limited access to functions and properties inside the window object
- Communication and passing data has some price when done too much
  - You cannot control the context switching behind the scenes

Browser compatibility

Support varies for different types of workers. See each worker type's page for specifics.

Update compatibility data on GitHub

	🖥️						📱					
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
`Worker`	4	12	3.5	10	10.6	4	4	18	4	11	5.1	1.0
`Worker()` constructor	4	12	3.5	10	10.6	4	4	18	4	11	5.1	1.0
`message` event	4	12	3.5	10	10.6	4	4	18	4	11.5	5.1	1.0
`messageerror` event	60	18	57	?	47	?	60	60	57	47	?	8.0
`onmessage`	4	12	3.5	10	10.6	4	4	18	4	11	5.1	1.0
`onmessageerror`	60	18	57	?	47	?	60	60	57	44	?	8.0
`postMessage`	Yes	12	Yes	10 *	47	Yes	Yes	Yes	Yes	44	Yes	Yes
`terminate`	4	12	3.5	10	10.6	4	4	18	4	11	5.1	1.0

Recommended Open Source - Ecosystem

- Comlink (by Google) - https://github.com/GoogleChromeLabs/comlink
- Workerize - https://github.com/developit/workeriz
- Parallel.js - https://github.com/parallel-js/parallel.js
- useWorker React hook - https://github.com/alewin/useworker

Takeaways

JS is not really working in parallel

Takeaways

Workers can be relevant for specific use case

Takeaways

Choose it
because of
your real needs

Not because
it's cool

Takeaways

Takeaways

JS is not really working in parallel

Workers can be relevant for specific use case

Choose it because of your real needs

Not because it's cool

Go Go Go!

Write Some Awesome Code!

Recommended Links

- Link to my frameworks race demo
- Open source libs to make your life easier using web workers
  - https://github.com/GoogleChromeLabs/comlink
  - https://github.com/developit/workerize
- Real-world examples that uses web workers
  - https://github.com/GoogleChromeLabs/squoosh/
  - https://microsoft.github.io/monaco-editor/
  - https://github.com/parallel-js/parallel.js - parallel js lib using web workers
- Recommended reading/video resources
  - https://www.youtube.com/watch?v=8aGhZQkoFbQ

Follow Me =>

www.chenfeldman.com
chen@vamos-tech.com

@chenfeldmn

@chenosfeldman

www.ranlevi.com/software

https://github.com/ChenFeldman

https://www.linkedin.com/in/chen-feldman-2404682a/

Credits

https://www.youtube.com/watch?v=EiPytIxrZtU

https://www.youtube.com/watch?v=pcYuOOe-kbw

https://www.newline.co/fullstack-react/articles/introduction-to-web-workers-with-react/

https://blog.logrocket.com/integrating-web-workers-in-a-react-app-with-comlink/

https://medium.com/prolanceer/optimizing-react-app-performance-using-web-workers-79266afd4a7

https://kentcdodds.com/blog/speed-up-your-app-with-web-workers

https://www.jstips.co/en/javascript/improving-your-async-functions-with-webworkers/

https://github.com/nodeca/pica?fbclid=IwAR3bvSvU4HPQyU-1VTHr24-4ZGTL-eGSLdG3tBs9TP0jxEdMXjOb-eZbHpE

https://www.freecodecamp.org/news/web-workers-in-action-2c9ff33be266/

https://blog.logrocket.com/real-time-processing-web-workers/, https://dev.to/trezy/loading-images-with-web-workers-49ap

kevinhoyt.com/2018/10/23/image-processing-in-a-web-worker/

https://developer.mozilla.org/en-US/docs/Web/API/Worker

https://www.youtube.com/watch?v=nLF0n9SACd4&feature=youtu.be

https://www.sitepoint.com/using-web-workers-to-improve-image-manipulation-performance/

https://medium.com/samsung-internet-dev/web-workers-in-the-real-world-d61387958a40

https://bitsofco.de/web-workers-vs-service-workers-vs-worklets/

https://github.com/Shopify/quilt/blob/master/packages/react-web-worker/README.md

https://medium.com/@azizhk/building-an-async-react-renderer-with-diffing-in-web-worker-f3be07f16d90