Web Workers

A brownbag workshop at

MX®

by Seth House

@whiteinge
seth@eseth.com

Why web workers?

The UI thread can be blocked.

Heavy computation can make the UI completely unresponsive -- animations, mouseovers, clicks, etc.

The UI thread can be blocked.

Heavy computation can make the UI completely unresponsive -- animations, mouseovers, clicks, etc.

```
// Block the UI thread (on purpose)
function sleep(time) {
    var now = Date.now();
    while (Date.now() < (now + time)) {}
}
```

The UI thread can be blocked.

Heavy computation can make the UI completely unresponsive -- animations, mouseovers, clicks, etc.

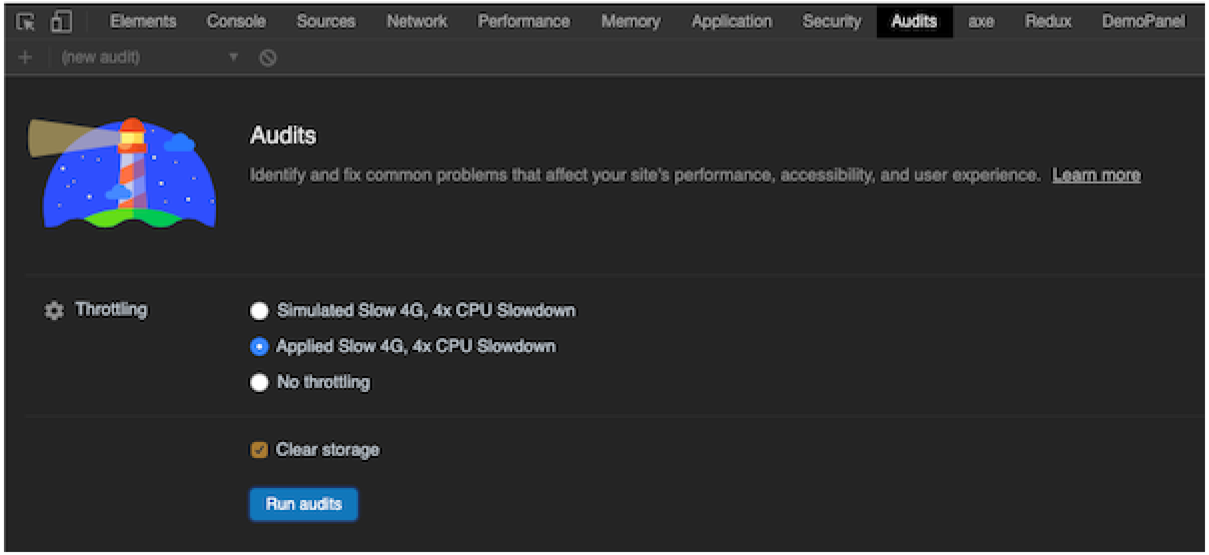
```
// Block the UI thread (on purpose)
function sleep(time) {
    var now = Date.now();
    while (Date.now() < (now + time)) {}
}
```

Another example

...especially on lower end devices.

...especially on lower end devices.

- No substitute for testing.
- Use `console.time('foo'); console.timeEnd('foo');`
- Use Lighthouse in Chrome dev tools:



Audits

Identify and fix common problems that affect your site's performance, accessibility, and user experience. Learn more

Throttling

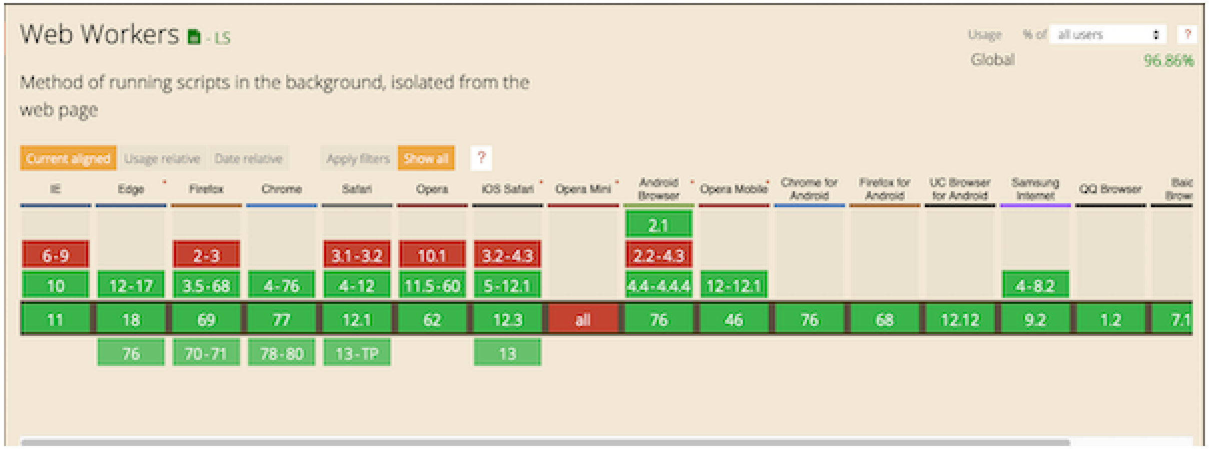
- ○ Simulated Slow 4G, 4x CPU Slowdown
- ● Applied Slow 4G, 4x CPU Slowdown
- ○ No throttling

✓ Clear storage

There have been hacks...

```javascript
function heavyComputation() {
    // ...do part of a thing.
    setTimeout(function() {
        // ...next part.
        setTimeout(function() {
            // ...next part.
            setTimeout(function() {
                // ...next part.
            }, 50);
        }, 50);
    }, 50);
}
```

But now we have web workers!



Web Workers - LS

Method of running scripts in the background, isolated from the web page

Usage % of all users ?

Global 96.86%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baic Brow
								2.1							
6-9		2-3		3.1-3.2	10.1	3.2-4.3		2.2-4.3							
10	12-17	3.5-68	4-76	4-12	11.5-60	5-12.1		4.4-4.4.4	12-12.1				4-8.2		
11	18	69	77	12.1	62	12.3	all	76	46	76	68	12.12	9.2	1.2	7.1
	76	70-71	78-80	13-TP		13									

How web workers

Start a dedicated thread.

```
var myWorker = new Worker('./path/to/script.js');
```

Start a dedicated thread.

```
var myWorker = new Worker('./path/to/script.js');
```

...or via a crazy workflow that involves stringifying functions.

Start a dedicated thread.

```
var myWorker = new Worker('./path/to/script.js');
```

...or via a crazy workflow that involves stringifying functions.

This part is async.

Worker features.

- The `navigator`object.
- The `location` object (read-only).
- `XMLHttpRequest`.
- `setTimeout`()/`clearTimeout`() and `setInterval`()/`clearInterval`().
- The Application Cache.
- Importing external scripts using `importScripts()`.
- Creating other web workers.

Messaging.

- Copy (for messages).
- Transfer (for binary data (like images)).

Import scripts.

```
importScripts('./lib/lodash.min.js');
```

Import scripts.

```
importScripts('./lib/lodash.min.js');
```

This part is synchronous!

Import scripts.

```
importScripts('./lib/lodash.min.js');
```

This part is synchronous!

Even works in IE10.

Worker tips

- Put lots of stuff in the worker.
- Dispatch function calls (e.g. the Redux pattern).
- Avoid bundling third-party libs shared between DOM & worker.