Web Workers: Handling heavy processing on the Client Side

Kevin Uehara

Staff Frontend Engineer at ifood

- I'm from Brazil, São Paulo
- Speaker and Tech Content Creator at Youtube
- Community partner at NodeBR
- Organizer of Campinas Frontend
- Trying to play CS (Counter strike) (I didn't played CS2 yet)
- Developer for about 9 years and focused on the front-end for about 5 years
- Typescript Sommelier TS ❤️

wavy Sympla CI&T CVC corp cpqd

Kevin Uehara

Staff Frontend Engineer at ifood

- IFood is the most big tech food delivery on Brazil
- I work in the IFood Logistics tribe
- I work cross-functionally in two Maps Platform teams: Location Areas and Location Geo
- Geoprocessing, geolocation, partner delivery areas, delivery routes and much more...

上原

What I will talk?

- What is Web Worker?
- Show me the code
- Results
- Hands on the code
- Contacts

上
原

Application Overview

- Let's create an application without using web workers, and then we will implement it and visualize the application's performance

上原

What is Web Worker?

上
原

Web Worker

- The web worker is an API provided by the browser.

- We know that Javascript itself, natively, is Single Thread (without going into the merits of NodeJS with libuv). However, with the Event Loop, Callback Queue, Stack architecture… we can say that JS can handle asynchronous processes on demand

上
原

Web Workers

- The Browser offers a series of APIs that can be used by the development team (storage, workers, PWA, performance monitoring, lighthouse)

- One of the topics I want to address in this talk is Web Workers, how do they live? How do they survive? how do they reproduce? Why use? Benefits?

上
原

Web Workers

- Web Workers are mechanisms that allow operations (usually large calculations) to be executed from the main thread.

- Making sure the main thread is not blocked.

- **What I want is for a function to perform certain heavy processing that will block integration with the page (UI)**

上
原

Kevin Toshihiro Uehara
Posted on Oct 14

Edit Manage Stats

1

Introducing Javascript Web Workers

#javascript #webdev #programming #browser

Hi, people!! How you doing?
It's nice to have you again!

In this artile I want to introduce you about Javascript Web Workers!

The idea to create this article came out after the creation of an application that I created through collaboration and "dispute" here in Brazil, called "Rinha Front end".

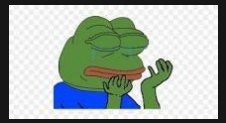
The main ideia of this app is read some json's files on client side. But you may ask yoursef: ok... so what is the problem?
And I answer you: Performance.

上原

Show me the code

上
原

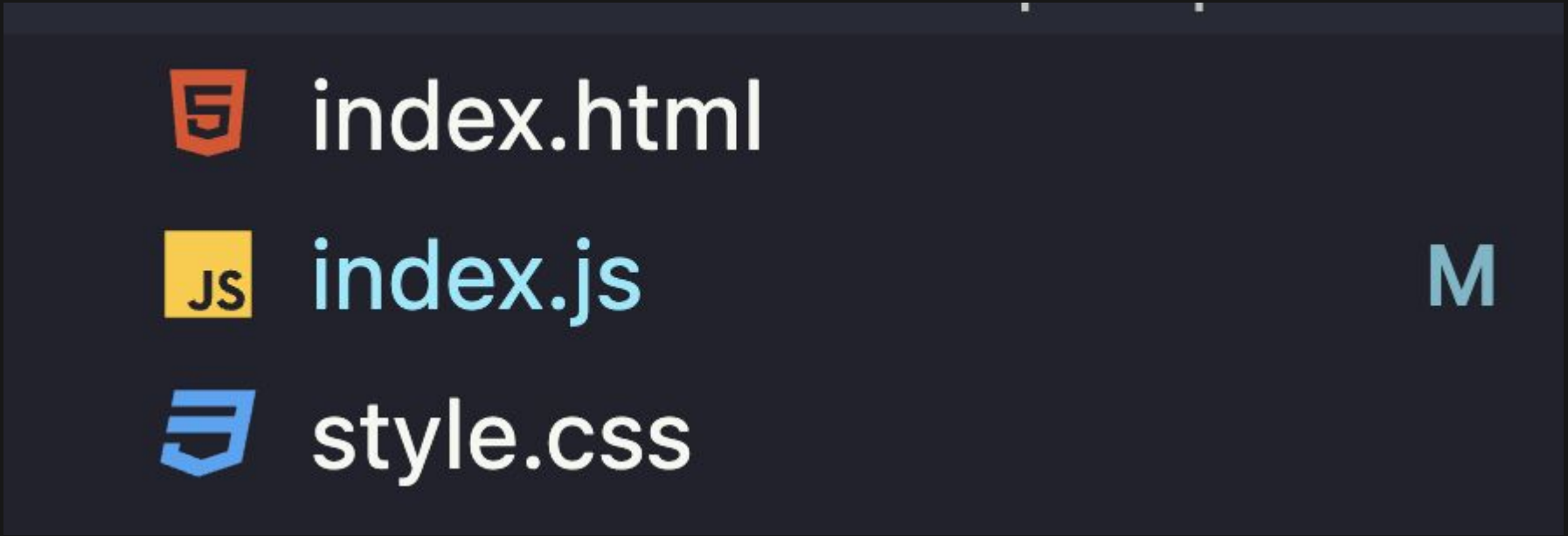
Show me the code

- For our demo project We will not use any framework or lib, just Vanilla JS 

- To do this, we will create our workspace with the following files:
  - index.html
  - index.js
  - style.css

上
原

Show me the code



S=Simple, isn't it?

上原

Show me the code

```
You, 1 second ago | 1 author (You)
1  const btnLargeOperation = document.getElementById("btn-large-operation");
2  const btnChangeColor = document.getElementById("btn-change-color");
3  const output = document.getElementById("output");
4  You, 2 weeks ago • feat: example of web workers
5  btnLargeOperation.addEventListener("click", () => {
6    output.textContent = "Started Large Process...";
7    const value = handleLargeOperation();
8    console.log(value);
9    // const worker = new Worker("worker.js");
10   // const before = Date.now();
11   // worker.postMessage("operation");
12   // worker.onmessage = (event) => {
13   //   const after = Date.now();
14   //   console.log("Executed in: ", (after - before) / 1000, " s");
15   //   console.log("Data processed Received: ", event.data);
16   //   output.textContent = event.data;
17   // };
18 });
19
20 // This event will be blocked by LargeOperation without web worker
21 btnChangeColor.addEventListener("click", () => {
22   const body = document.querySelector("body");
```

PORTS GITLENS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
➜ web-worker-exampole git:(main) ✗
```

In principle, don't pay attention to the code at this point.

上原

Show me the code: HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web Worker Example</title>
    <link rel="stylesheet" href="style.css">
    <script src="index.js" async></script>
</head>
<body>
    <button class="btn" id="btn-large-operation">Start Large Operation</button>
    <button class="btn" id="btn-change-color">Change Background Color</button>
    <pre id="output"></pre>
</body>
</html>      You, 2 weeks ago • feat: example of web workers
```

上原

Show me the code: CSS

```
body {
    margin: 0;
    padding: 0;
    font-family: Arial, Helvetica, sans-serif;
    box-sizing: border-box;
    background-color: #fff;
}

.btn {
    padding: 12px;
    background-color: #3730a3;
    cursor: pointer;
    color: #fff;
    border-radius: 4px;
    border: none
}

.btn:hover {
    background-color: #6366f1;
}      You, 2 weeks ago • feat: example of web w
```

上原

Show me the code: JS

- In the first function it will be the heavy operation where I will loop and give a huge value
- In the second eventListener function, I just call the first function
- In the third function I just change the application background from dark to white

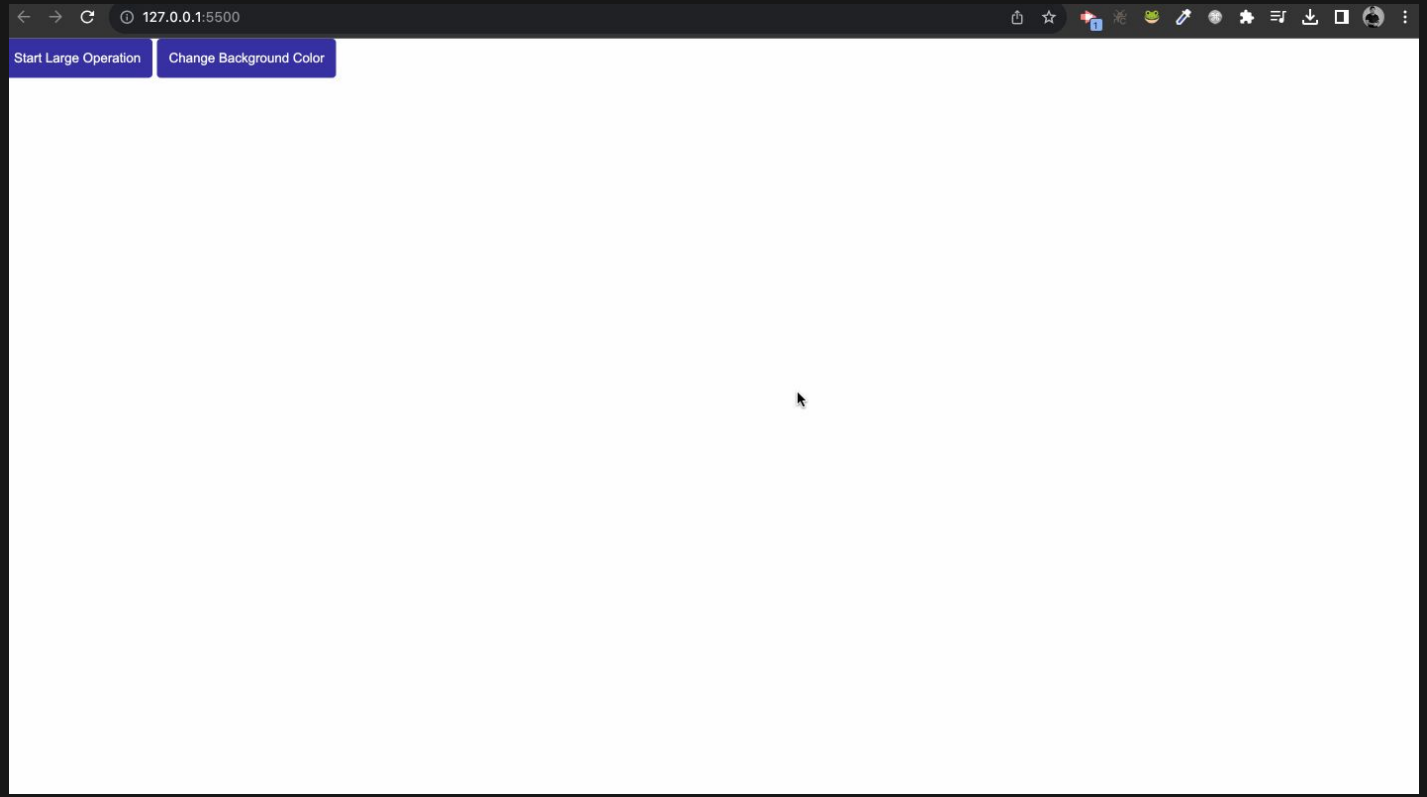
```
You, 3 minutes ago | 1 author (You)
const btnLargeOperation = document.getElementById("btn-large-operation");
const btnChangeColor = document.getElementById("btn-change-color");
const output = document.getElementById("output");

const handleLargeOperation = () => {
  let value = 0;
  for (let i = 0; i <= 1e8 * 30; i++) {
    value += i;
  }
  return value;
};

btnLargeOperation.addEventListener("click", () => {
  output.textContent = "Started Large Process...";      You, 2 weeks ago •
  const value = handleLargeOperation();
  console.log(value);
});

// This event will be blocked by LargeOperation without web worker
btnChangeColor.addEventListener("click", () => {
  const body = document.querySelector("body");
  if (
    body.style.backgroundColor &&
    body.style.backgroundColor !== "rgb(255, 255, 255)"
  ) {
    body.style.backgroundColor = "#fff";
  } else {
    body.style.backgroundColor = "#000";
  }
});
```

Result without Web Worker



Without Web Worker and starting the heavy process, the UI is blocked due to the main thread

上原

Show me the code

S=Let's create another file, called worker.js

上原

Show me the code - Web Worker

```
const handleLargeOperation = () => {
  let value = 0;
  for (let i = 0; i <= 1e8 * 30; i++) {
    value += i;
  }
  return value;
};

onmessage = (event) => {
  if (event.data === "operation") {
    const value = handleLargeOperation();
    postMessage(value);        You, 2 weeks ago
  }
};
```

- Note that I changed the function that was previously in `index.js, now it is in the web worker file

- the **onmessage** function represents all messages that were received in the Web Worker and based on them, perform some operation.

上
原

Show me the code - Changing the index.js

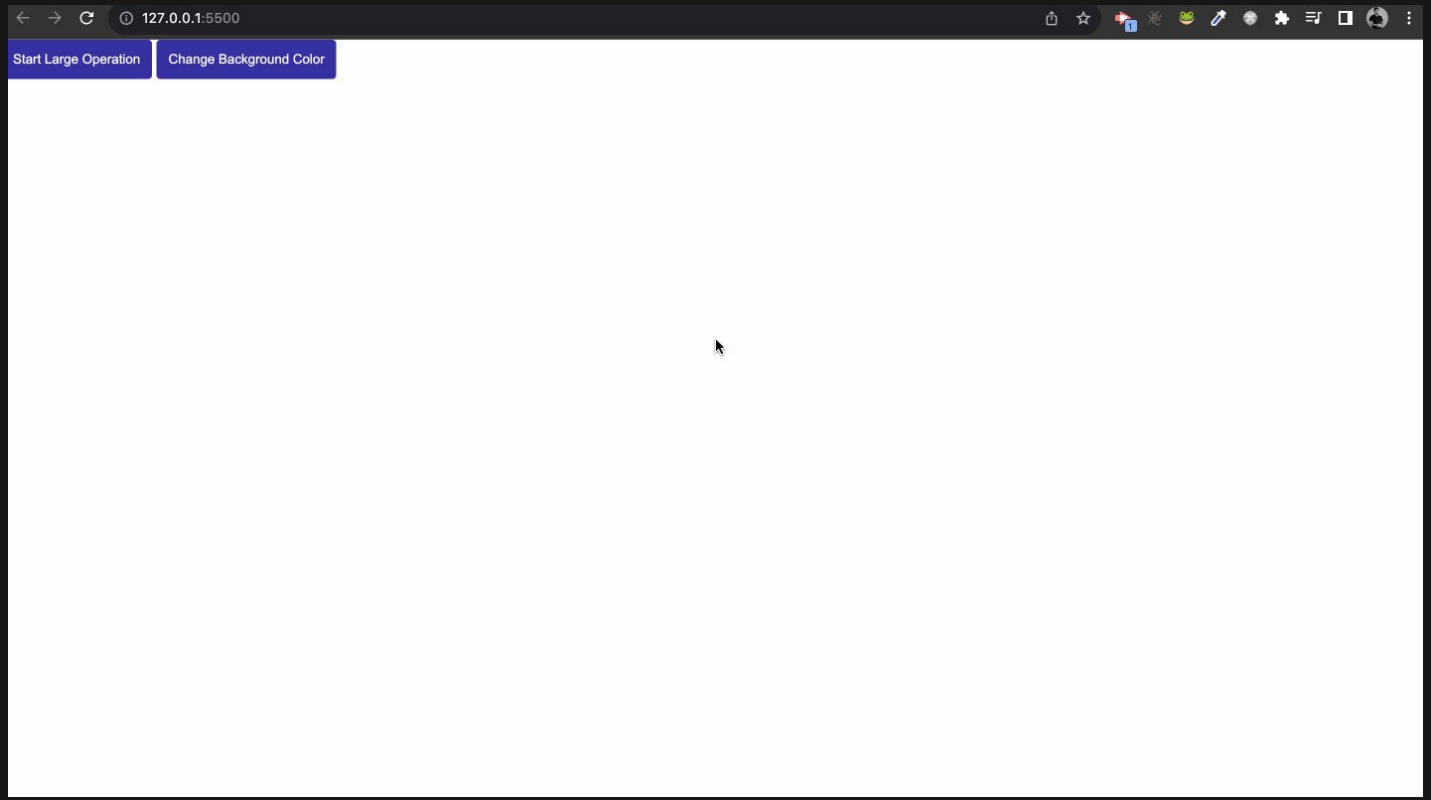
- Note that I removed the processing-heavy function for workers.js
- Now, we need to instantiate the worker API with new Worker("worker.js");
- Post a message like "operation" and a timestamp
- Other than that nothing changes. Let's see the end result:

```
const btnLargeOperation = document.getElementById("btn-large-operation");
const btnChangeColor = document.getElementById("btn-change-color");
const output = document.getElementById("output");

btnLargeOperation.addEventListener("click", () => {
  output.textContent = "Started Large Process...";
  //   const value = handleLargeOperation();
  //   console.log(value);
  const worker = new Worker("worker.js");
  const before = Date.now();
  worker.postMessage("operation");
  worker.onmessage = (event) => {
    const after = Date.now();
    console.log("Executed in: ", (after - before) / 1000, " s");
    console.log("Data processed Received: ", event.data);
    output.textContent = event.data;
  };
});

// This event will be blocked by LargeOperation without web worker
btnChangeColor.addEventListener("click", () => {
  const body = document.querySelector("body");
  if (
    body.style.backgroundColor &&
    body.style.backgroundColor !== "rgb(255, 255, 255)"
  ) {
    body.style.backgroundColor = "#fff";      You, 2 weeks ago • feat: exa
  } else {
    body.style.backgroundColor = "#000";
  }
```

Result with Web Worker

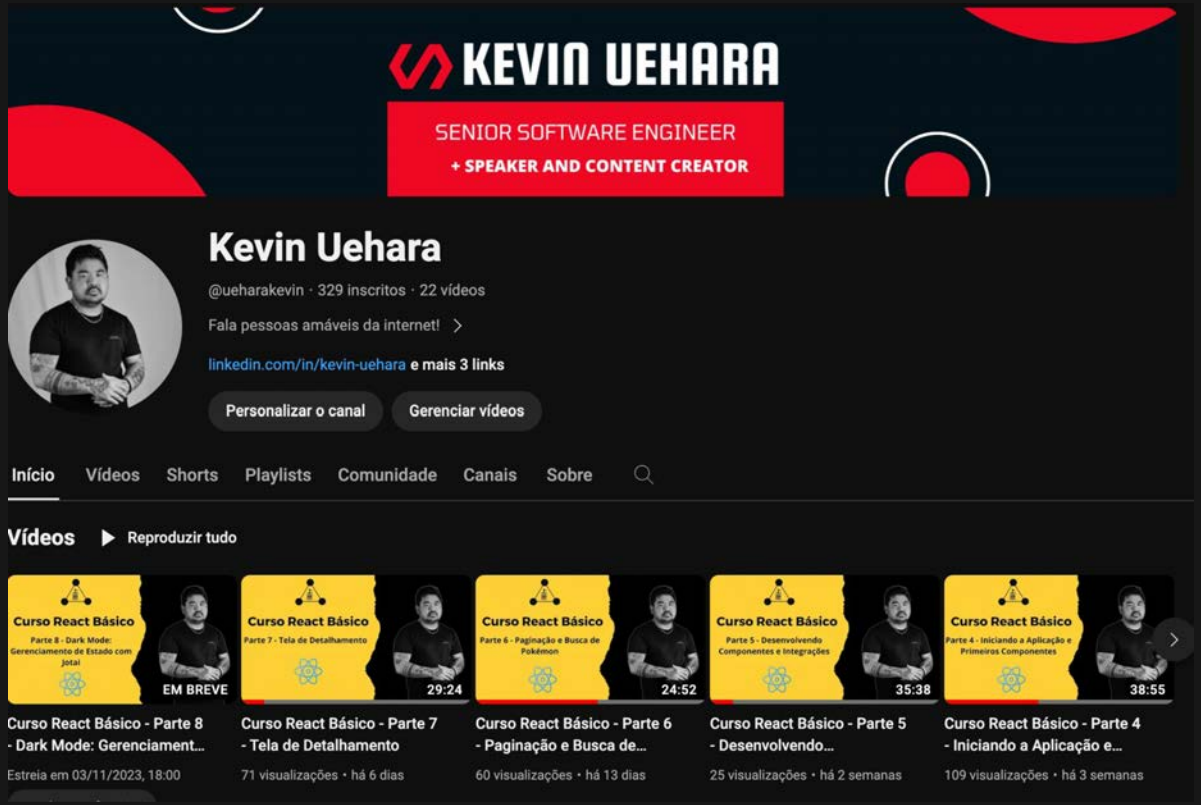


With Web Worker and starting the heavy process, the UI is not blocked due to the main thread

上原

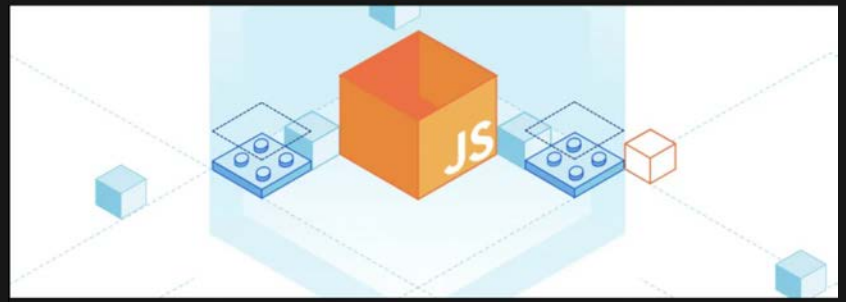
That's it in short

Youtube ADS



Skipp >>

Artigle on dev.to



Kevin Toshihiro Uehara
Posted on Oct 14

Edit Manage Stats

1

Introducing Javascript Web Workers

#javascript #webdev #programming #browser

Hi, people!! How you doing?
It's nice to have you again!

In this artile I want to introduce you about Javascript Web Workers!

Contacts

Linkedin:
https://www.linkedin.com/in/kevin-uehara

Instagram:
https://www.instagram.com/uehara_kevin

Twitter:
https://twitter.com/ueharaDev

Github:
https://github.com/kevinuehara

dev.to:
https://dev.to/kevin-uehara

Youtube:
https://www.youtube.com/@ueharakevin/

Hands on the code

上
原

Thank you so muck!

stay wel allways

上
原