HTML Web Workers API

A Web Worker is a JavaScript script that runs in the background, without affecting the page's performance.

A Web Worker is a JavaScript script that runs in the background, without affecting the page's performance.

What is a Web Worker?

When executing scripts in an HTML page, the page will not respond until the script finishes.

A Web Worker is a JavaScript script that runs in the background, independently of other scripts, without affecting page performance. You can continue doing whatever you want: clicking, selecting items, etc., while the Web Worker runs in the background.

Browser support

The numbers in the table indicate the browser version that first fully supported WebWorker.

API Web Workers	Google Chrome	MS Edge	Firefox	Safari	Opera
	4.0	10.0	3.5	4.0	11.5

Examples of HTML Web Workers

The example below creates a simple web worker that counts numbers in the background:

Count numbers:

Start Worker Stop Worker

Check Web Worker support

Before creating a web worker, check if the user's browser supports it:

```
if (typeof(Worker) !== "undefined") { // Yes! Web worker support! // Some code.
```

Create a Web Worker file

Now, let's create the web worker in an external JavaScript file.

Here, the article creates a valuable script. The script is stored in the file **"demo_workers.js"** :

```
var i = 0; function timedCount() { i = i + 1; postMessage(i); setTimeout("timedC
```

The crucial part of the code above is the method `postMessage()`– used to post the message back to the HTML page.

Note : Web workers are typically not used for simple scripts like this, but rather for tasks that require more CPU resources.

Create a Web Worker object

Now that we have the web worker file, the next step is to call it from an HTML page.

The following lines will check if the worker already exists - it creates a new web worker object and runs the code in **"demo_workers.js"** :

```
if (typeof(w) == "undefined") { w = new Worker("demo_workers.js"); }
```

Then, we can send and receive messages from the web worker.

Add the "onmessage" event listener to the web worker.

```
w.onmessage = function(event){ document.getElementById("result").innerHTML = ever
```

When a web worker posts a message, the code in the event listener is executed. Data from the web worker is stored in event.data.

Terminate a web worker.

When a web worker object is created, it will continue listening for messages (even after the external script terminates) until it finishes.

To terminate the web worker and free use of computer/browser resources, use the following method `terminate()`:

```
w.terminate();
```

Reuse Web Worker

If you set the worker variable to undefined, you can reuse the code after it's terminated:

```
w = undefined;
```

Complete sample code for Web Worker

The Code Worker has appeared in the .js file. Below is the code for the HTML page:

Count numbers:

Start Worker Stop Worker

Web Worker and DOM

Because web workers are located in external files, they do not have access to the following JavaScript objects:

1. Window object
2. Document object
3. Father

See more: