

API Web Workers

< Anterior

Próximo >

Um web worker é um JavaScript em execução em segundo plano, sem afetar o desempenho da página.

O que é um Web Worker?

Ao executar scripts em uma página HTML, a página deixa de responder até que o script seja concluído.

Um web worker é um JavaScript executado em segundo plano, independentemente de outros scripts, sem afetar o desempenho da página. Você pode continuar a fazer o que quiser: clicar, selecionar coisas, etc., enquanto o web worker é executado em segundo plano.

Suporte de navegador

Os números na tabela especificam as primeiras versões de navegador que oferecem suporte total para Web Workers:

				
Chrome 4	IE 10	Firefox 3.5	Safari 4	Opera 11.5
Jan 2010	Sep 2012	Jun 2009	Jun 2009	Jun 2011

Exemplo de Web Workers

O exemplo abaixo cria um web worker simples que conta números em segundo plano:

Exemplo

Count numbers:

Start Worker

Stop Worker

Tente você mesmo "

Verifique o suporte do Web Worker

Antes de criar um web worker, verifique se o navegador do usuário é compatível:

```
if (typeof(Worker) !== "undefined") {  
  // Yes! Web worker support!  
  // Some code.....  
} else {  
  // Sorry! No Web Worker support..  
}
```

Criar um arquivo de trabalhador da web

Agora, vamos criar nosso web worker em um JavaScript externo.

Aqui, criamos um script que conta. O script é armazenado no arquivo "demo_workers.js":

```
let i = 0;  
  
function timedCount() {  
  i ++;  
  postMessage(i);  
  setTimeout("timedCount()",500);  
}  
  
timedCount();
```

The important part of the code above is the `postMessage()` method - which is used to post a message back to the HTML page.

Note: Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

Create a Web Worker Object

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

The following lines checks if the worker already exists, if not - 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 an "onmessage" event listener to the web worker.

```
w.onmessage = function(event){  
  document.getElementById("result").innerHTML = event.data;  
};
```

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

Terminate a Web Worker

Quando um objeto de trabalho da web é criado, ele continua a escutar mensagens (mesmo depois que o script externo é concluído) até que seja encerrado.

Para encerrar um web worker e liberar recursos de navegador / computador, use o `terminate()` método:

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

Reutilizar o Web Worker

Se você definir a variável de trabalho como indefinida, depois de encerrada, você pode reutilizar o código:

```
w = undefined;
```

Código de exemplo de trabalhador da Web completo

Já vimos o código do Worker no arquivo .js. Abaixo está o código para a página HTML:

Exemplo

```
<!DOCTYPE html>  
<html>  
<body>  
  
<p>Count numbers: <output id="result"></output></p>  
<button onclick="startWorker()">Start Worker</button>  
<button onclick="stopWorker()">Stop Worker</button>  
  
<script>  
let w;  
  
function startWorker() {  
  if (typeof(w) == "undefined") {  
    w = new Worker("demo_workers.js");  
  }  
  w.onmessage = function(event) {  
    document.getElementById("result").innerHTML = event.data;  
  };  
}  
  
function stopWorker() {  
  w.terminate();  
  w = undefined;  
}  
</script>  
  
</body>  
</html>
```

Tente você mesmo "

Web Workers e o DOM

Como os web workers estão em arquivos externos, eles não têm acesso aos seguintes objetos JavaScript:

- O objeto da janela
- O objeto do documento
- O objeto pai

< Anterior

Próximo >

COLOR PICKER



COMO NÓS



Obtenha a
certificação
completando
um curso hoje!



iniciar

JOGO DE
CÓDIGOS



Jogar um jogo

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Web Courses

HTML Course
CSS Course
JavaScript Course
Front End Course
SQL Course
Python Course
PHP Course
jQuery Course
Java Course
C++ Course
C# Course
XML Course

Get Certified >

