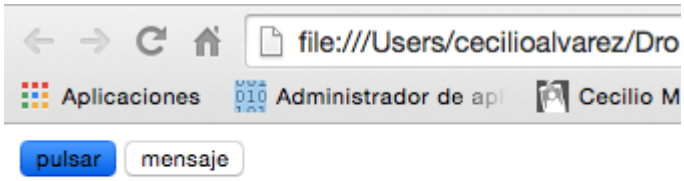
Una de las características que poco a poco comenzaremos a utilizar más con HTML5 es el manejo de HTML5 Web Workers. ¿Qué es un Web Worker? . Un Web Worker es una tarea que se ejecuta de forma paralela a la ejecución de nuestro código de JavaScript y su objetivo es liberar de carga al motor de Javascript principal que es el encargado de responder a los eventos y acciones del usuario. Vamos a ver un ejemplo práctico.

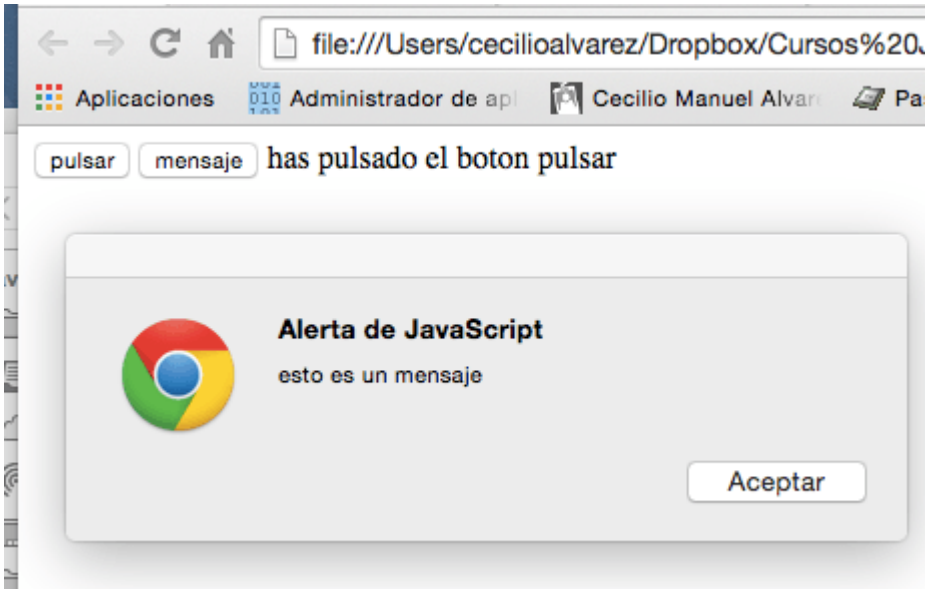
```
<html>
<head>
<script src="jquery-1.11.1.js" type="text/javascript"></script>
<script src="jquery-1.11.1.js" type="text/javascript"></script>
<script type="text/javascript" src="lowdash.js">
</script>

<script type="text/javascript">

$(document).ready(function() {
$("#pulsar").click(function() {

var mensaje="el boton pulsar";

var fecha= new Date();
while (new Date() - fecha <10000){

}

$("body").append("has pulsado "+ mensaje);

})
```

```
$("#mensaje").click(function() {
alert ("esto es un mensaje");
})
});
</script>
</head>
<body>
<input type="button" id="pulsar" value="pulsar"/>
<input type="button" id="mensaje" value="mensaje"/>
</body>
</html>
```

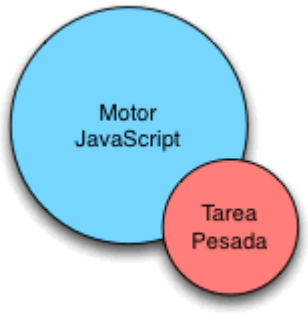
En este ejemplo disponemos de dos botones el botón pulsar entra en un bucle de 10 segundos antes de añadirnos el texto ” has pulsado el botón pulsar”. Por el otro lado el segundo botón simplemente nos muestra un alert por pantalla “esto es un mensaje”. Ahora bien si pulsamos primero el botón de “pulsar” la página quedará bloqueada durante 10 segundos.



Una vez finalizados estos 10 segundos podremos pulsar al botón de mensaje que nos imprimirá el mensaje de alerta.

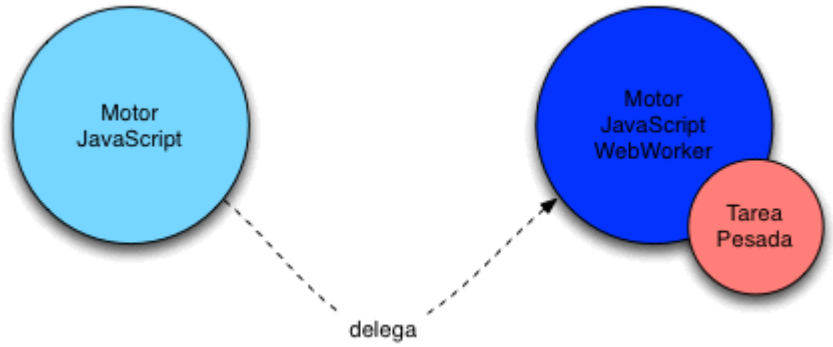


El problema que tenemos es que el primer botón obliga al Thread de JavaScript a realizar una tarea pesada. Concretamente un bucle que dura 10 segundos. El resultado es que el Thread de JavaScript queda completamente ocupado y la página deja de responder al estar realizando esta tarea pesada.



Para solventar este problema podemos hacer de los HTML 5 Web Workers. Estos se

encargan de cargar otro motor de JavaScript que se encargue de realizar la tarea pesada y libera al motor principal de Javascript que se encarga de responder a los eventos del usuario.



Para ello vamos a crearnos un fichero miworker,js que será el encargado de realizar esta tarea pesada.

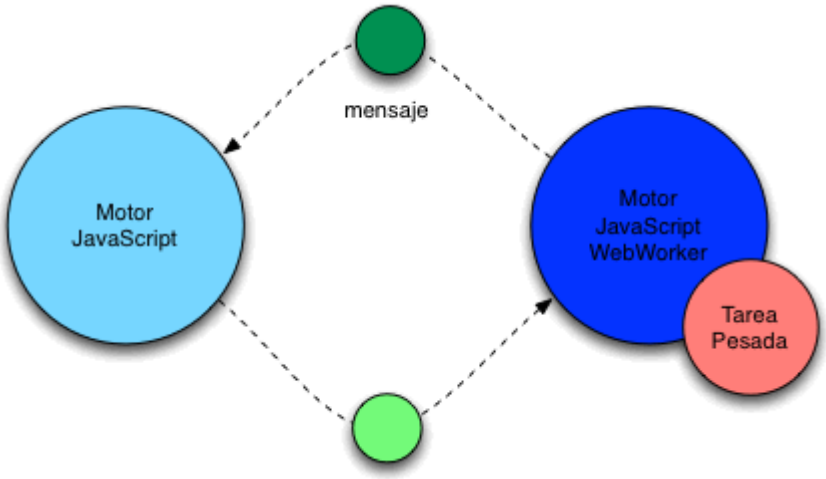
```
self.addEventListener('message', function (event) {

var fecha= new Date();
while (new Date() - fecha <5000){

}
self.postMessage("desde el worker :"+event.data)

}, false);
```

Una vez realizada esta operación modificaremos el JavaScript principal para que delegue la tarea en el Web Worker.

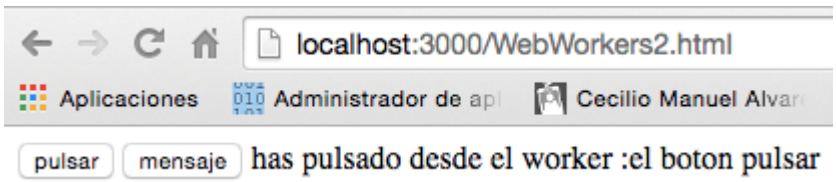
```
<html>
<head>
<script src="jquery-1.11.1.js" type="text/javascript"></script>
<script src="jquery-1.11.1.js" type="text/javascript"></script>
<script type="text/javascript">

$(document).ready(function() {
$("#pulsar").click(function() {

var mensaje="el boton pulsar";

var w1 = new Worker( "miworker.js");
w1.postMessage(mensaje);
w1.addEventListener("message",function(evento){

$("body").append("has pulsado "+ evento.data);
});
})

$("#mensaje").click(function() {
alert ("esto es un mensaje");
})
});
</script>
</head>
<body>
```

```
<input type="button" id="pulsar" value="pulsar"/>
<input type="button" id="mensaje" value="mensaje"/>
</body>
</html>
```

De esta forma JavaScript creará un WebWorker y le enviará un mensaje a través del método postMessage. El Web Worker recibirá el mensaje a través de un listener realizará la tarea pesasa y devolverá un nuevo mensaje al Thread principal de JavaScript.



El resultado será el siguiente :



Otros artículos relacionados: HTML5 Data Attributes , JavaScript y Threads , HTML

Performance