[js Worker 线程](http://www.cnblogs.com/panmy/p/5764507.html)

    在平时的运行的javascript脚本都在主线程中执行，如果当前脚本包含复杂的、耗时的代码。那么JavaScript脚本的执行将会被阻塞，甚至整个页面看起来都失去响应。

例子：

   假设程序需要计算、收集1~9999的之间所有质数，不采用后台线程，而是直接使用JavaScript前台线程的计算、收集质数。代码如下。

[复制代码](javascript:void(0);)

<html>

<head>

<meta name="author" content="Yeeku.H.Lee(CrazyIt.org)" />

<meta http-equiv="Content-Type" content="text/html; charset=GBK" />

<title> 计算质数 </title>

</head>

<body>

<p>已经发现的所有质数：<div id="result"></div></p>

<script type="text/javascript">

var n = 1;

search:

while (n < 99999)

{

// 开始搜寻下一个质数

n += 1;

for (var i = 2; i <= Math.sqrt(n); i++)

{

// 如果除以n的余数为0，开始判断下一个数字。

if (n % i == 0)

{

continue search;

}

}

document.getElementById('result').innerHTML += (n + ", ");

}

</script>

</body>

</html>

[复制代码](javascript:void(0);)

浏览器久久未响应，一篇空白。最后过来七八秒全部崩出来。如果改为使用webWorker启用多线程呢？使用Worker创建线程非常简单，只要调用Worker的构造器就可以。

        Worker(scriptURL):scriptURL用于指定所使用JavaScript脚本的路径

[复制代码](javascript:void(0);)

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

<script>

var worker = new Worker('Worker12.js');

worker.onmessage = function (event) {

document.getElementById("result").innerHTML += event.data + ",";

}

</script>

</head>

<body>

<p>

已经发现的所有质数：<div id="result" style="width: 1500px; height:auto; border:1px solid red; word-wrap: break-word; ">

</div>

</p>

</body>

</html>

[复制代码](javascript:void(0);)

Worker12.js

[复制代码](javascript:void(0);)

var n = 1;

search:

while (n < 10000) {

// 开始搜寻下一个质数

n += 1;

for (var i = 2; i <= Math.sqrt(n); i++) {

// 如果除以n的余数为0，开始判断下一个数字。

if (n % i == 0) {

continue search;

}

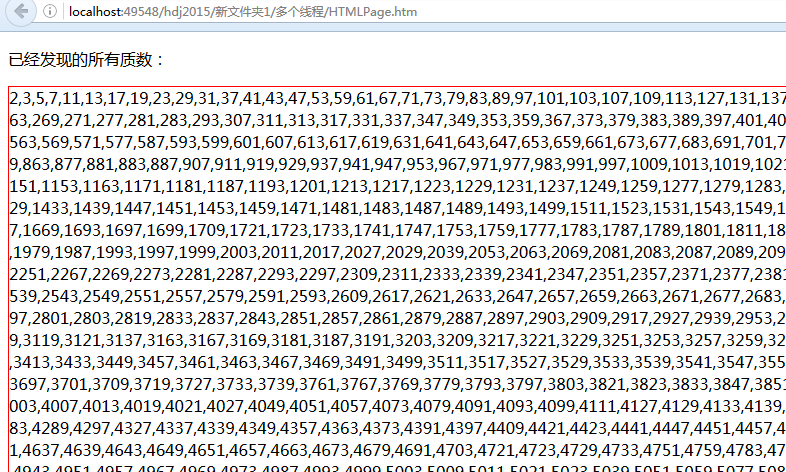
}

// 发现质数

postMessage(n);

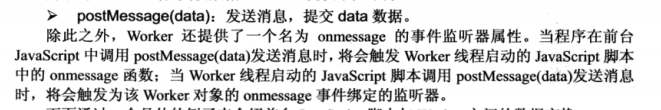
}

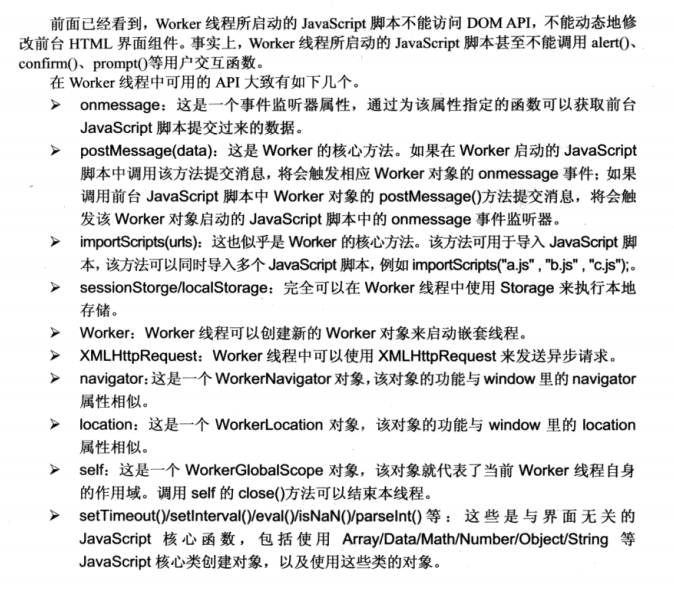
[复制代码](javascript:void(0);)



 注意：

   Worker启动的子线程找到质数之后，并不能直接把找到的质数更新在页面上显示，必须通过postMessage(n)发送消息给前台JavaScript通信。





 例子2：Worker线程交换数据：

[复制代码](javascript:void(0);)

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>多个线程交换数据</title>

<script>

var car = function () {

var start = document.getElementById("start1").value;

var end = document.getElementById("end1").value;

if (start >= end) {

return;

} else {

var cal = new Worker("Worker.js");

// 定义需要提交给Worker线程的数据

var data = {

start: start,

end: end

};

// 向Worker线程提交数据。

cal.postMessage(JSON.stringify(data));

cal.onmessage = function (event) {

document.getElementById("result").innerHTML += event.data + ",";

}

}

}

</script>

</head>

<body>

起始值：<input type="text" id="start1" /><br />

结束值：<input type="text" id="end1" /><br />

<input type="button" value="点击" onclick="car();" />

<div id="result" style="width: 1500px; height: auto; border: 1px solid red; word-wrap: break-word;">

</div>

</body>

</html>

[复制代码](javascript:void(0);)

Worker.js

[复制代码](javascript:void(0);)

onmessage = function (event) {

// 将数据提取出来。

var data = JSON.parse(event.data);

// 取出start参数

var start = data.start;

// 取出end参数

var end = data.end;

var result = "";

search:

for (var n = start; n <= end; n++) {

for (var i = 2; i <= Math.sqrt(n); i++) {

// 如果除以n的余数为0，开始判断下一个数字。

if (n % i == 0) {

continue search;

}

}

// 搜集找到的质数

result += (n + ",");

}

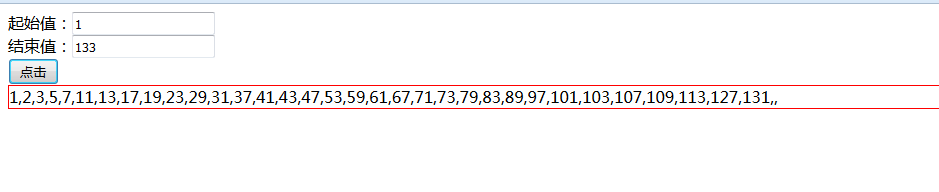
// 发送消息，将会触发前台JavaScript脚本中

// Worker对象的onmessage方法

postMessage(result);

}

[复制代码](javascript:void(0);)



例子3：多个线程嵌套

[复制代码](javascript:void(0);)

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>多个线程嵌套 </title>

<script>

var car = function () {

var start = document.getElementById("start1").value;

var end = document.getElementById("end1").value;

var count = document.getElementById("count1").value;

var worker = new Worker(‘03\_MainThread.js');

var data = {

start: start,

end: end,

count: count

};

worker.postMessage(JSON.stringify(data));

worker.onmessage = function (event) {

document.getElementById("result").innerHTML += event.data + ",";

}

}

</script>

</head>

<body>

起始值：<input type="text" id="start1" /><br />

结束值：<input type="text" id="end1" /><br />

个数：<input type="text" id="count1" /><br />

<input type="button" value="点击" onclick="car();" />

<div id="result" style="width: 1500px; height: auto; border: 1px solid red; word-wrap: break-word;">

</div>

</body>

</html>

[复制代码](javascript:void(0);)

MainThread.js

[复制代码](javascript:void(0);)

onmessage = function (event) {

// 将数据提取出来。

var data = JSON.parse(event.data);

// 取出start参数

var start = data.start;

// 取出end参数

var end = data.end;

// 取出count参数

var count = data.count;

var result = "";

search:

for (var n = start; n <= end; n++) {

for (var i = 2; i <= Math.sqrt(n); i++) {

// 如果除以n的余数为0，开始判断下一个数字。

if (n % i == 0) {

continue search;

}

}

// 搜集找到的质数

result += (n + ",");

}

// 再次启动Worker线程

var sub = new Worker("subworker.js");

// 把需要处理的数据传入启动的Worker线程中

sub.postMessage({ result: result, count: count });

sub.onmessage = function (event) {

// 发送消息，将会触发前台JavaScript脚本中

// Worker对象的onmessage方法

postMessage(event.data);

}

}

[复制代码](javascript:void(0);)

03\_SubThread.js

[复制代码](javascript:void(0);)

onmessage = function (event) {

// 将数据提取出来。

var data = event.data;

// 提取所有质数

var primeNums = data.result.split(",")

var randResult = "";

for (var i = 0; i < data.count; i++) {

// 计算一个随机索引值

var randIndex = Math.floor(Math.random()

\* (primeNums.length - 1));

// 随机地"收集"一个质数

randResult += (primeNums[randIndex] + ",");

}

// 发送消息，将会触发启动它的JavaScript脚本中

// 对应Worker对象的onmessage方法

postMessage(randResult);

}

[复制代码](javascript:void(0);)

