4.4 Web Workersとマルチスレッド

元々 JavaScriptはシングルスレッドで動作するように設計されていた。しかし、今後HTML5の普及でリッチなアプリケーションが登場したときにブラウザ上で実行に時間のかかるJavaScriptプログラムが問題となり得る。そこで、JavaScriptでスレッドの生成とバックグラウンド実行ができるWeb Workersの仕様が策定された※1。

1 Web Workersのサポート状況

Web WorkersのAPIは先にFirefox、Chrome、Safariの3ブラウザで実装されたが、最新のInternet ExplorerやOperaでも動作するようになった（**表4.5**）。ちなみにブラウザのセキュリティ制約によってfile:プロトコルやlocalhost上ではWeb Workersが動作しないようになっているので、適当なドメインのWebサーバ上でサンプルプログラムを実行する必要がある。

表4.5 Web Workersをサポートしているブラウザ

対応ブラウザ	バージョン
Internet Explorer	10+
Firefox	3.5+
Chrome	3+
Safari	4+
Opera	10.60+

※1 http://www.w3.org/TR/workers/

2 Workerオブジェクトの生成

JavaScriptで時間のかかる処理を実行すると、すべての計算が終わるまでブラウザの画面が更新されず、計算途中はCPUの占有率が100%となり、フリーズしたように見えてしまう。

そこで、このように重たい処理であってもバックグラウンドで実行できるWeb Workersを利用して、10万までの素数をブラウザのHTMLに出力するプログラムを作成してみよう（**リスト4.6**）。

リスト4.6 Web Workersで10万までの素数を出力する（webworker.html）

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>Web Workers Demo</title>
<script>
// ↓ワーカスレッドの生成
var worker = new Worker("webworker.js");

// ↓ワーカからメッセージを受信したときの処理
worker.onmessage = function(e) {
  var prime = e.data.prime;
  setTimeout(function() {
    document.body.innerHTML += prime + ", ";
  }, 0);
}

// ↓10万までの素数を計算することをワーカに指示
worker.postMessage({"n": 100000});
</script>
```

3 Workerスレッドの定義

new Workerの第一引数でJavaScriptのファイル名を指定するとWeb Workersは別のスレッドを生成し、バックグランドで処理を実行する。ここでは、nまでの素数を小さい順に数え上げるJavaScriptプログラムを別スレッドとして実行する（**リスト4.7**）。

リスト4.7　別スレッドでnまでの素数を小さい順に数え上げる（webworker.js）

```
// ↓ワーカ側でメッセージを受信したときの処理
onmessage = function(e) {
  var n = e.data.n;
  loop:
  for (var i = 1; i <= n; i += 2) {
    for (var j = 2; j * j <= i; j++) {
      // ↓割り切れたら次の数字へ
      if (i % j == 0) continue loop;
    }
    postMessage({"prime": i}); // ←素数の出力
  }
}
```

4 Web Workersの処理の流れ

このときのWeb Workersの処理の流れを図4.9に示す。

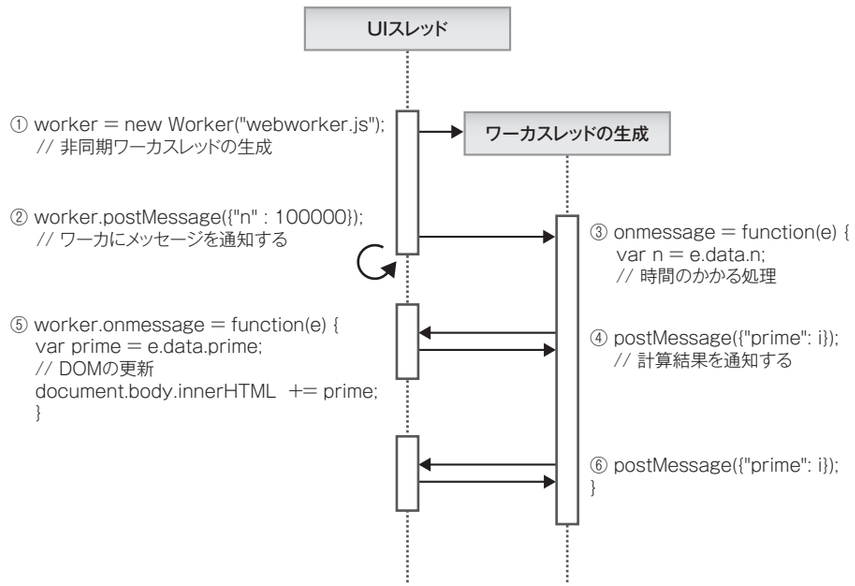

図4.9　Web Workersの処理の流れ

① ワーカスレッドを生成するには、new Workerの第一引数でJavaScriptのファイル名を指定する。
② 生成したワーカスレッドに対してメッセージを通知するにはworkerオブジェクトのpostMessageメソッドを実行する。データを送信するには第一引数にJSON形式で指定する。
③ ワーカ側のonmessageイベントハンドラで、メッセージを受信したときの動作を定義する。第一引数のdataオブジェクト経由でデータを受け取ることができる。
④ ワーカ側での処理が終わったらpostMessageメソッドで結果を返す。
⑤ ワーカオブジェクトのonmessageイベントハンドラで結果を受け取りDOMの更新を行う。
⑥ postMessageは何回でも呼び出すことができる。

実際にOpera11.52でこのWeb Workersのプログラムを実行した結果を示す（図4.10）。

ちなみに、ワーカスレッド内のグローバルオブジェクトはselfという名前で参照できる。しかし、Web Workersのセキュリティモデルの制約によって、ワーカー側ではDOM

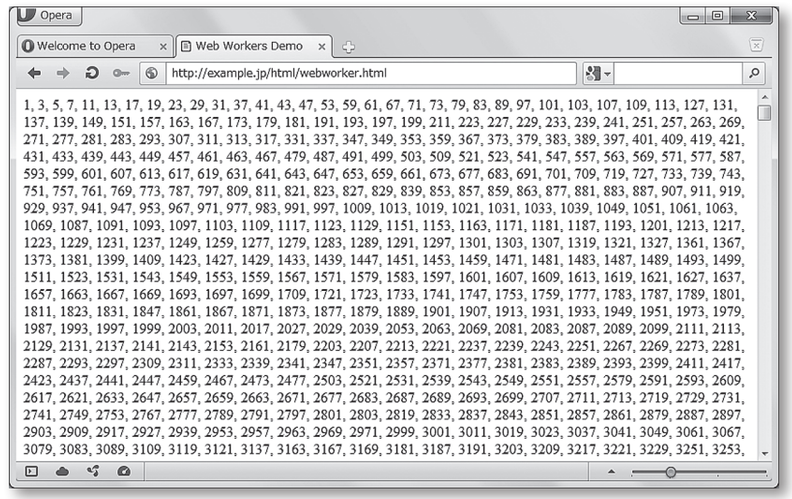

図4.10　Opera11.52でWeb Workersを実行した結果