**- - WebWorker alkalmazása - -**

A webmunkás a World Wide Web Consortium (W3C) és a Web Hypertext Application Technology Working Group (WHATWG) meghatározása szerint egy JavaScript- szkript , amelyet egy HTML- oldalról futtatnak le, amely a háttérben fut, függetlenül azoktól a szkriptektől, amelyek szintén előfordulhattak.

A webmunkások nem a JavaScript részei, hanem egy böngészőfunkció, amely a JavaScript segítségével érhető el.

A nem kívánják őket nagy számban alkalmazni, hiszen várhatóan hosszú élettartamúak, magas indítási teljesítményköltséggel és magas példányonkénti memóriaköltséggel rendelkeznek.

A HTML-dokumentum szkriptjein kívül futnak. Következésképpen megkönnyíthetik a JavaScript programok egyidejű végrehajtását. A dolgozók szálszerű üzenettovábbítást alkalmaznak a párhuzamosság eléréséhez. Tökéletesek ahhoz, hogy a felhasználói felületet frissen, hatékonyan és érzékenyen tartsák a felhasználók számára.

Böngészők tekintetében jelenleg a Chrome, a Firefox és a Safari támogatja teljes mértékben a webmunkást. A szabvány szerint szöveges üzeneteket küldhetünk, Firefox esetén bármi mást is, amit JSON-ná tud alakítani. Az Opera és az Internet Explorer nem támogatja, ez utóbbi esetben lehetséges a WorkerPool használata, azonban a két API nem kompatibilis egymással.

Munkásokat, és a munkásokon belül újabb munkásokat elméletileg korlátlan számban indíthatunk.

**Kétféle webmunkás létezik**

Dedikált webmunkás - A dedikált dolgozó csak az általa hívott szkript segítségével érhető el.

Megosztott webmunkás - A megosztott munkatárs több szkript által is elérhető - még akkor is, ha különböző ablakok, iframe-ek vagy akár dolgozók érik el őket.

Web Worker alkalmazások:

* Helyi tárolási adatbázisok olvasása/írása
* Valós idejű szövegformázás, helyesírás-ellenőrzés és szintaktikai kiemelés
* Képek szűrése vásznon
* Nagyméretű JSON-adatkészletek feldolgozása
* Multimédia feldolgozása (hang és videó)
* Bonyolult matematikai képletek, például a legnagyobb prímszám megtalálása

A webdolgozó üzenettovábbítással lépnek kapcsolatba a fő dokumentummal .

A következő kód létrehoz egy dolgozót, amely végrehajtja a JavaScriptet az adott fájlban.

**var** worker = **new** Worker("worker\_script.js");

Üzenet küldéséhez a munkásnak *postMessage* metódust kell használni az alábbiak szerint.

worker.postMessage("Hello World!");

Az *onmessage* tulajdonság eseménykezelőt használ az információk lekéréséhez egy munkástól.

worker.onmessage = **function**(event) {

alert("Received message " + event.data);

doSomething();

}

**function** doSomething() {

*//do work*

worker.postMessage("Work done!");

}

worker.terminate();

Amint egy dolgozó megszűnik, kikerül a hatókörből, és a rá hivatkozó változó definiálatlanná válik, ezen a ponton szükség esetén új dolgozót kell létrehozni.

A webmunkások legegyszerűbb használata egy számításigényes feladat elvégzése a felhasználói felület megszakítása nélkül.

Ebben a példában a fődokumentum egy webworkert indít prímszámok kiszámítására , és fokozatosan megjeleníti a legutóbb talált prímszámot.

Példa:

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Worker example: One-core computation</**title**>

</**head**>

<**body**>

<**p**>The highest prime number discovered so far is: <**output** id="result"></**output**>

</**p**>

<**script**>

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

worker.onmessage = **function** (event) {

document.getElementById('result').textContent = event.data;

};

</**script**>

</**body**>

</**html**>

A Worker()konstruktor hívás létrehoz egy webworkert, és visszaad egy worker objektumot, amely az adott webworkert reprezentálja, és amely a webworkerrel való kommunikációra szolgál.

Maga a WebWorker a következő:

**var** n = 1;

**var** end\_value = 10\*\*7;

search: **while** (n <= end\_value) {

n++;

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

**if** (n % i == 0)

**continue** search;

*// talált egy prímet!*

postMessage(n);

}

**Hozzunk fel még egy nagyon egyszerű példát!**

Feladatunk az, hogy hozzunk létre egy olyan webmunkást, ami segítségével egy számoló ciklust tudunk elindítani egy gomb megnyomásával. Továbbá tartalmazzon egy olyan funkciót is gomb formájában, mellyel ki tudunk lépni a ciklusból.

A feladatunk megoldásához szükségünk van egy Java Script-re és egy HTML oldalra.

Nos akkor vágjunk is bele!

1. Létrehozunk egy szkriptet workers.js néven

var i = 0;

function timedCount()

{

i = i + 1;

postMessage(i);

setTimeout("timedCount()",500);

}

timedCount();

1. Létrehozunk egy html fájlt webworker.html néven

<!DOCTYPE html>

<html>

<body>

<p>Számoló: <output id="result"></output></p>

<button onclick="startWorker()">Indítás</button>

<button onclick="stopWorker()">Leállítás</button>

<script>

var w;

function startWorker() {

if (typeof(Worker) !== "undefined") {

if (typeof(w) == "undefined") {

w = new Worker("workers.js");

}

w.onmessage = function(event) {

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

};

} else {

document.getElementById("result").innerHTML = "Sajnalom, de a webworker nem tamogatott!";

}

}

function stopWorker() {

w.terminate();

w = undefined;

}

</script>

</body>

</html>

Végeredmény:

