Web Workers

Web Workers מגדירים דרך סטנדרטית ליצירת background threads ב- JavaScript, כך שאפליקציות הכתובת ב- JavaScript יכולות לנצל בצורה טובה יותר את המשאבים של מחשבים בעלי מעבדים מרובי-ליבות.

ביצוע פעולות וחישובים ארוכים באמצעות Web Workers מאפשרים למשתמש להמשיך לעבוד עם הדף בזמן ביצוע הפעולה ועל-ידי כך מונעים "תקיעות" של הדף.

כמו-כן ניתן להשתמש ב- Web Workers על-מנת להאזין להודעות המגיעות משרת האינטרנט (כמו למשל ידיעות חדשותיות) ולעדכן את הדף ברגע שמגיעה הודעה חדשה מהשרת.

JavaScript עצמה היא שפה שרצה בסביבה שהיא Single-Threaded. טכניקות שונות בהן נעזרו מפתחים בעבר לעבודה אסינכרונית, כמו setInterval(), setTimeOut(), XMLHttpRequest וכו', אמנם איפשרו להריץ פעולות ברקע מבלי שהמשתמש ייאלץ להמתין לתוצאה שלהן, אולם לא באמת יצרו Thread-ים חדשים ולכן לא איפשרו להריץ מספר פעולות בו-זמנית (concurrent).

את ה- Web Worker יש לכתוב בקובץ JavaScript חיצוני.

אחת המגבלות המרכזיות של Web Workers היא שאין להם גישה ישירה לאובייקט ה- window, וגם לא לעץ האלמנטים של הדף (ה- DOM), מכאן שה- Web Worker צריך לקבל את המידע הנחוץ לעבודתו באמצעות הודעות המועברות אליו מהדף (יוסבר בפירוט בהמשך הסעיף).

ל- Web Workers יש גישה רק לאובייקטים ולפונקציות ה- JS הבאות:

- אובייקט ה- navigator
- אובייקט ה- location
- XMLHttpRequest
- setTimeOut(), setInterval()
- ייבוא של סקריפטים חיצוניים באמצעות המתודה importScripts()
- יצירת Web Workers בנים

לעומת זאת, ל- Web Workers אין גישה לאובייקטים הבאים:

- ה- DOM (הוא לא thread safe)
- אובייקט ה- window
- אובייקט ה- document
- אובייקט ה- parent

Web Workers נתמכים כיום על-ידי הדפדפנים הבאים:

- Chrome החל מגירסה 3.0
- Firefox החל מגירסה 5.0
- Safari החל מגירסה 4.0
- IE ייתמך בגירסה 10.0

לדוגמא, נכתוב Web Worker שיבצע בדיקה של כל המספרים הראשוניים בטווח מסוים הנקבע על-ידי המשתמש באמצעות תהליך שירוץ ברקע.

תחילה נכתוב את ה- Web Worker. את ה- Web Worker יש לכתוב בקובץ js נפרד.

<u>השלבים שיש לבצע בדף הראשי</u>

1. יצירת אובייקט Worker חדש שמקבל ב- C'tor שלו את שם קובץ ה- js של ה- Worker (במקרה שלנו primesWorker.js).
במידה והקובץ נמצא, הדפדפן יוריד אותו בצורה אסינכרונית מהשרת ויצור Thread נפרד.
2. הפעלת ה- worker מתבצעת באמצעות הפונקציה postMessage(). פונקציה זו יכולה לקבל פרמטר יחיד מטיפוס מחרוזת או אובייקט JSON. בדוגמא שלנו, עלינו להעביר ל- worker שני פרמטרים (ערך מינימלי וערך מקסימלי), ולכן ניעזר באובייקט JSON שיעטוף את שני הפרמטרים.
3. הדף הראשי יכול לקבל הודעות מה- worker על-ידי רישום לאירוע onmessage של ה- worker (בדוגמא שלנו, בכל פעם שה- worker ימצא מספר ראשוני בטווח הוא ישלח אותו לדף). הגישה למידע שהגיע מה- worker מתבצעת דרך מאפיין ה- data של האובייקט שנשלח לדף.

להלן הקוד המלא של הדף הראשי:

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML5 Web Workers</title>
    <script type="text/javascript" src="primesWorker.js"></script>
    <script>
        function checkWorkerSupport() {
            if (typeof(window.Worker) === "undefined")
                alert("Your browser doesn't support HTML5 Web Workers!");
        }

        function findPrimes() {
            var min = document.getElementById("txtMin").value;
            var max = document.getElementById("txtMax").value;

            // Create a new worker object
            var worker = new Worker("primesWorker.js");

            // Register to the onmessage event of the worker, that will
read the prime number
            // sent from the worker and add it to the listbox
            worker.onmessage = function (e) {
                var lstPrimes = document.getElementById("lstPrimes");
                var opt = document.createElement("option");
                opt.innerHTML = e.data;
                lstPrimes.appendChild(opt);
            }

            // Show errors from the worker
            worker.onerror = function (error) {
                alert(error.data);
            }

            // Activate the worker
            worker.postMessage({ 'min': min, 'max': max });
        }
    </script>
</head>
<body onload="checkWorkerSupport();">
    <h2>Find Primes</h2>
    Min: <input type="text" id="txtMin" /><br />
    Max: <input type="text" id="txtMax" /><br />

    <input type="button" id="btnFindPrimes" value="Find Primes"
onclick="findPrimes();"/><br />
    <select id="lstPrimes" size="10">

    </select>
</body>
</html>
```

<u>השלבים שיש לבצע ב- Worker</u>

1. כאשר נשלחת הודעה מהדף ל- Worker באמצעות הפונקציה postMessage(), מופעל אירוע ב- Worker שנקרא onmessage. על ה- Worker להירשם לאירוע הזה ולהגדיר מהי הפונקציה שתופעל בעת האירוע. פונקציה זו תתבצע במקביל להמשך פעולת הדף.
2. פונקציה זו מקבלת כפרמטר אובייקט המכיל את ההודעה שנשלחה מהדף במאפיין ה- data שלו.
3. ה- Worker יכול לשלוח הודעות חזרה לדף באמצעות קריאה לפונקציה postMessage(). הודעות אלו יתקבלו בדף על-ידי הפונקציה שנרשמה לאירוע onmessage של ה- worker.

להלן הקוד המלא של ה- worker:

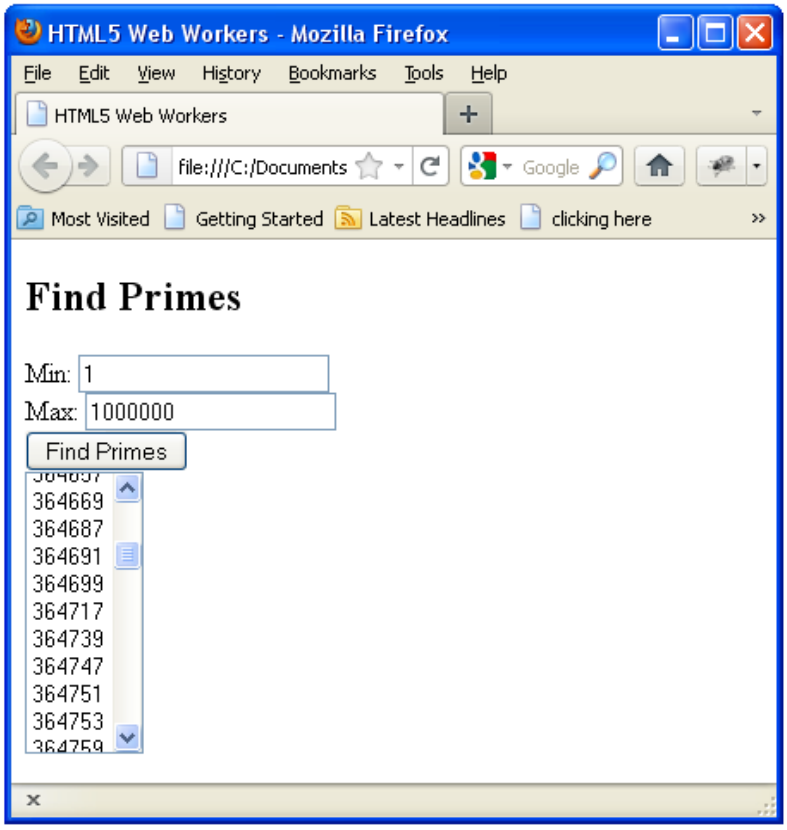
```
// primesWorker.js

// This function will be activated by the postMessage() sent from the page
onmessage = function checkPrimes(e) {
    var min = parseInt(e.data.min);
    var max = parseInt(e.data.max);

    for (i = min; i <= max; i++) {
        if (isPrime(i))
           postMessage(i); // send the newly found prime number to the page
    }
}

// An auxiliary function
function isPrime(num) {
    for (var i = 2; i <= Math.sqrt(num); i++) {
        if (num % i == 0)
            return false;
    }
    return true;
}
```

להלן התוצאה המתקבלת בדפדפן:



הסברים נוספים על Web Workers ניתן למצוא באתר:

http://www.html5rocks.com/en/tutorials/workers/basics/

תרגיל 5

Create an HTML page that launches a web worker on the initialization of the page. When the web worker is launched, it should in turn launch two other web workers.

On the client page the user is asked to enter two numbers in two different input boxes. The client page then sends these two numbers to the main web worker, which in turn will send them on to the two child web workers.

One child web worker will multiply the numbers together and return the result, while the other web worker will add the two numbers and return the result. When both results have been returned, the main web worker will create a string which contains the results and return this text string to the main HTML page for display.