שלום לכולם.  
  
הקדמה קצרה.

למי שלא מכיר אותי שמי ניר וכיום אני עובד בקונדואיט על פיתוח מוצר חדש שאמור לצאת לשוק בחודשים הקרובים.

כשדיברנו על ההרצאה אני ורוני (שירצה אחרי הפסקת הצהרים), רצינו לבנות אפליקציה לדוגמא שתכיל את הנושאים שתכנו לדבר עליהם ולהציג את האפליקציה. הרעיון היה לא להציג מצגת של רשימת מכולת אלא דוגמאות עובדות של קוד.

בגלל מגבלות בעיקר של חוסר זמן (כי עדיין המחויבות הראשית היא למקום עבודה) ומגבלות טכניות שאת חלקם אני אעביר בהרצאה לא הספקנו להרים אפליקציה אבל מרות זאת ההרצאה שלי מלווה בלא מעט קוד והמטרה שלי היא שמי שירצה יקח את השקפים לאחר ההרצאה ויוכל ישר להתחיל לשחק עם הקוד ולא רק להיות עם רקע תאורטי.  
  
למי שיש שאלות אני מעדיף שתרשמו אותם ואני אנסה לענות בסוף ההרצאה או בהפסקה. תרגישו חופשי לגשת לשאול ואם לא הספקתם קחו כרטיס ביקור ותשלחו לי שאלות במייל.

התכנים שאני עומד לדבר עליהם היום:

Web workers  
כאשר בWEBWORKERS נכסה את הסוגים הבאים

1. **Dedicated Workers**
2. **Inline Workers**
3. **Shared Workers**

אני מניח שרובכם נתקל בWEB WORKERS אבל יחד עם זאת אני מניח שרובכם לא שמע על INLINE וSHARED WORKERS.

fileAPI  
הנושא הזה מכיל בתוכו 2 חלקים: עבודה מול קבצים מקומיים במחשב וAPI מקביל שמאפשר עבודה מלאה מול קבצים בSANDBOX של הדפדפן ומכיל בתוכו API שלם לעבודה עם FILES וDIRECTORIES.   
מקוצר זמן אני לא אספיק לעבור על הנושא האחרון של ההרצאה אבל גם עבורו הכנתי מצגת נפרדת עם דוגמאות של קוד.  
  
2 הנושאים הללו חולקים מספר דברים משותפים ובמהלך ההרצאה נראה שיש לא מעט קוד משותף ל2 השנואים הללו.

1. הרעיון המרכזי של WEBWORKER הוא לפתור את הבעיה שאנחנו המפתחים עומדים מולה היום והיא שדפדפנים לא תומכים בMULTI THREADS.  
   כיום על לדמותTHREADS היינו משתמשים באחת מהצורות הבאות על מנת להריץ קוד בצורה אסינכרונית.  
   AJAX  
   2XHR – רכיב נוסף ב 5HTML שלא יכוסה בהרצאה הזו.  
   EVENTS  
   SETINTERVAL שאני באופן אישי לא ממליץ להשתמש בו בכלל היות ואף פעם לא נדע בוודאות כמה זמן לקח לקוד לרוץ. לדוגמא אם אנחנו מעדכנים חלק בדף וטוענים תוכן ממקור חיצוני כל 0.5 שניות ובפועל כמה בקשות לא נסתיימו בזמן, התשובה טרם היגיע חזרה לדפדן אבל בגלל שהשתמשנו בSETINTERVAL ולא SETIMEOUT הפונקציה נקראת שוב ונשלחת עוד בקשת עדכון.  
   הבעיה היא: איזה מידע יוצג אחרון.  
   הפתרון – שימוש רק בSETTIMEOUT במקום כאשר את הSETTIMEOUT נפעיל בסוף הפונקציה כאשר היא סיימה את פעולתה. ככה נוכל להבטיח שלא יווצר מצב שהפונקציה תקרא שוב לפני שהיא נסתיימה.  
     
   שימוש בWEB WORKERS מייתר את הצורך למצוא פתרונות יצירתיים שכאלו.
2. WORKERS
   1. הרעיון המרכזי הוא שהקוד שWORKER מאפשרים שימוש בTHREADS להרצת קוד JS והם רצים במקביל לTHREAD הראשי שלרוב יטפל בכל נושאי ה UI בדף.
   2. הקוד של הWORKER נמצא בקובץ נפרד (חוץ ממקרה יוצא מהכלל שנראה מאוחר יותר)
   3. יצירה של WORKER מורידה את הקוד בצורה אסינכרונית ואם הקובץ לא נמצא נקבל שגיאה 404 והקוד מן הסתם לא ירוץ.
   4. WORKERS יכולים לתקשר עם העולם החיצון בעזרת POST MESSAGE ו ONMESSAGE.  
      שזו גם הצורה היחידה שבה ניתן לבצע DEBUG על WEBWORKERS.
3. לWORKERS יש גישה לאובייקטים הבאים:  
   IMPORT SCRIPT זוהי הפונקציה שמאפשרת טעינה של קבצים חיצוניים לWORKER
4. היות שWORKERS רצים בסביבת הרצה נפרדת משלהם אין להם גישה לאובייקטים הבאים:
   1. DOM – not a thread safe is there was an aces
   2. WINDOW - כנגזרת מזה אין גישה לפונקציות ולמשתנים בדף הראשי.
   3. Document - לא ניתן לשנות או להוסיף EVENTS
   4. ושוב העברת מידע בין הWORKER לחלון הראשי או לחלונות- טאבים אחרים מתבצעת באמצעות MESSAGES
5. WORKER CODE MAIN WINDOW
   1. זוהי הצורה הבסיסית ליצירת WORKER. יוצרים NEW WORKER וכפרמטר מעבירים את שם הקובץ שיש לטעון
   2. העברת מידע לWORKER יכולה להתבצע ב1 מ2 דרכים:
      1. דרך ONMESSAGE
      2. דרף LISTENER
      3. 2 הדרכים הללו זהות לחלוטין
   3. על מנת להתחיל את הWORKER יש לשלוח לו הודעה בעזרת POST MESSAGE רוב הדפדפנים תומכים גם בהעברת הודעות בפורמט JSON.  
      כרום מגרסה 14 תומך גם העברת מידע המכיל ARRAYBUFFER (נגיע לזה יותר מאוחר)
6. WORKER FILE
   1. בWORKER CONTEXT אנחנו מאזינים לMESSAGE כמו הדף הראשי.
   2. כמו כן SELF THIS או כלום כולם מתייחסים לWORKER CONTEXT
7. DEDICATED WORKER DEMO  
   ניתן לראות ששימוש בWEBWORKERS לא חוסם את הדפדפן מלבצע פעולות אחרות ובמקביל פעולות שצורכות זמן וכח חישוב מתבצעות ברקע.
8. INLINE WORKER  
   בניגוד לDEDICATED WORKER שבו כאמור הקוד נמתא בקובץ נפרדINLINE WORKER הוא WORKER שנוצא בצורה דינמית כאשר. זה מחזיר אותנו לשקף השני שבו ציינו שהקוד של הWORKER מוגדר בקובץ נפרד ונטען בצורה אסינכרונית. כאן כאמור הקוד נכתב ON THE FLY ויכול אפילו לשבת בדף הראשי או בכל מקום אחר.  
     
   INLINE WORKER כפי שתכף נראה בדוגמת קוד נוצר באמצעות BLBOBUILDER. אובייקט זה יחזור אלינו מאוחר יותר כשנעבור לFILEAPI.  
   השימוש בBLOBLBUILDER יותר לנו אובייקט מסוגBLOBURL שמאוחסן בזכרון של הדפדפן. משך החיים שלו הוא כל עוד הדף חי.  
   רישמה של כל הBLOBS ניתן לראות בקישור הבא.
9. דוגמת קוד לINLINE WORKER
   1. ה"טריק" שאנו מציגים בדוגמא הנל הוא לרשום את הקוד של הWORKER בתוף תגית סקריפט מסוג לא מוכר על מנת שהדפדפן "יתעלם" ולא יעבד את התוכן כסקריפט. זו טכניקה נפוצה כיום ומי שמכיר JQ TEMPLATE מתבסס על אותו הרעיון.  
      בתוך תגיד הסקריפט נכתוב את הקוד כאילו הוא כתוב בקובץ חיצוני.