Web Sockets

היא דרך לתקשורת ברשתות תקשורת. עבור מפתחי ווב – הכוונה היא לתקשורת בין שרת לבין לקוח (client) כאשר הלקוח יכול להיות דפדפן או שרת אחר, אבל ברוב המקרים יש לנו שרת מרוחק של אתר כלשהו ו-JavaScript שמתקשר עם השרת הזה באמצעות web socket ו-API מיוחד שהוטמע בכל הדפדפנים המודרניים שתומכים ב-HTML 5.

מה ההבדל בין Web Sockets ל-AJAX? עם AJAX אנחנו יוצרים בקשה בצד הלקוח אל צד השרת (צד השרת לא יכול ליזום בקשה). בנוסף, כל בקשה כוללת את ה-headers ועטופה במידע. מה שהופך את ה-AJAX לבזבזני.

Web Sockets פותחים לנו קשר דו כיווני – כלומר גם השרת יכול לדחוף מסרים משלו אל הלקוח – דבר שהוא לא טריוויאלי. בנוסף כל מה שעובר דרך ה-Web Socket לא כולל headers ו-junk אלא פשוט עובר. לא מדובר בבקשה-תגובה כמו ב-AJAX אלא בקשר שנשאר פתוח.

Web Sockets נשמע אידיאלי, לא? גם קל יחסית להשתמש בו. למה אנחנו משתמשים עדיין ב-AJAX? כי יש דפדפנים שלא תומכים בו: אקספלורר 9 ומטה ואנדרואיד גרסה 4.3 ומטה. איזה כיף.

בנוסף, קשה יחסית להפעיל Web Sockets מבחינת צד השרת. כאשר אני אומר 'קשה', אני לא מתכוון למשהו שצריך מיליון הגדרות וקינפוגים. אבל הרבה יותר קל למישהו ללא הרבה ניסיון להקים את ה-AJAX מצד השרת מאשר Web Sockets.

בנוסף, עבור הגדרות רשת מסוימות ושרתי פרוקסי מסוימים, Web Sockets לא יעבדו.

אבל השרתים הולכים ומתעדכנים והדפדפנים העתיקות הולכים ונעלמים וכך גם הסיבות לא להשתמש ב-Web Sockets. וגם אין שום סיבה שלא להכיר את הפרוטוקול הזה.

במאמר הזה אני אדבר על אימפלמנטציה של Web Sockets בפרונט אנד. בעתיד אני אפרסם מאמרים על איך עושים את זה עם Node.js ועם PHP.

על מנת להשתמש בפרונט אנד, אני צריך שרת כלשהו. החבר'ה הטובים מ-Kazing בנו [שרת שתומך בבקשות של http ושל https](http://www.websocket.org/echo.html). השרת הוא שרת echo, כלומר הוא תמיד מחזיר את מה ששולחים לו. שימו לב לכתובת של השרת, היא קצת לא סטנדרטית: ws://echo.websocket.org  
הפרוטוקול הוא לא http אלא ws. ופרוטקול מאובטח הוא wss

איך אני מתחבר ל-Web Socket?

var wsUri = "ws://echo.websocket.org/";

websocket = new WebSocket(wsUri);

ברגע שעשינו את זה, אנו מחוברים ל-Web Socket שאותו פירטנו. עכשיו אפשר להשתמש במתודות שלו – המתודה העיקרית היא send שבאמצעותה אנחנו שולחים משהו לשרת.

websocket.send(message);

כאשר אנחנו רוצים לסגור את הקשר, מכל סיבה שהיא, אנו מעבירים את הפקודה:

websocket.close();

זה הכל! אבל רגע. אמרתי קודם שהשרת יכול לשלוח אלינו מידע מתי שמתחשק לו – כל עוד הקישור פתוח – כלומר מהרגע שעשינו instance ל- WebSocket ועד הרגע שעשינו websocket.close. זה נכון, השרת יכול לשלוח מידע, אבל אנחנו צריכים להגדיר מה לעשות עם המידע הזה. לשמור אותו? להציג אותו בפני המשתמש?

על מנת להגדיר מה בדיוק לעשות שמתקבל מידע, יש לנו אירוע 'קבלת מידע' שאפשר להצמד אליו באופן הבא:

websocket.onmessage = function (evt) {

//print evt.data to whatever you want to

};

זה הכל! בדיוק כמו AJAX, רק יותר פשוט לטעמי. יש עוד אירועים שיכולים לסייע לנו – למשל אירוע שפועל כאשר יש תקלה מסוג כלשהו (השרת שמנסים לעבוד מולו לא זמין) – כך זה נראה:

websocket.onerror = function (evt) {

// This is the error message: evt.data. print it or do something else

};

או אירוע שמתקיים כאשר אנו סוגרים את הקשר באופן יזום עם websocket.close.

websocket.onclose = function (evt) {

//do something

};

משהו חשוב שחייבים לשים אליו לב – את שליחת המסרים אל ה-WebSocket חייבים לעשות אחרי שיצרנו את התקשורת. על מנת לוודא את זה, עדיף לעטוף את שליחת המסרים באירוע שנורה כאשר ה-Web Socket מוכן:

websocket.onopen = function (evt) {

websocket.send(message);

};

זו דוגמה חיה, פשוט לחצו על הכפתור ותראו איך הבקשה נשלחת אל ה-websocket server והתוצאה יוצאת ב-alert. אפשר לראות את הבקשה באמצעות שימוש בכלי הפיתוח. ב-chrome dev tools אפשר להסתכל על network (רק אל תשכחו לפלטר לפי Web Socket וללחוץ פעמיים על שם ה-host, זה באג שעדיין לא תיקנו בכרום) וגם בפיירבאג אפשר לעשות את זה.

הקוד נראה כך:

<script>

function connectToWebSocket() {

var wsUri = "wss://echo.websocket.org";

websocket = new WebSocket(wsUri);

websocket.onopen = function (evt) {

websocket.send('helllooooo');

};

websocket.onmessage = function (evt) {

alert("Message from remote server : "+evt.data);

};

}

</script

<button onclick="connectToWebSocket()">Click me</button>

דוגמה נוספת, ויותר מפורטת (אך מעט יותר מסובכת) נמצאת [בתחתית של דף ההסבר לשרת הדוגמה](http://www.websocket.org/echo.html).

זהו, עד כאן בנוגע ל-Web Sockets – אלטרנטיבה ל-AJAX שמאוד חשובה במקרים מסוימים. אם אנו יוצרים אפליקציה שמקיימת תקשורת אינטנסיבית עם השרת (למשל צ'אט או משחק), ללא ספק Web Sockets הוא הבחירה הטובה ביותר. בכל מקרה, כדאי להכיר את הפרוטוקול הזה.