Ajax

Asynchronous javascript and xml

**במקום לטעון דף שלם מעבירים מידע ספציפי מהשרת לקליינט, מאחורי הקלעים, מבלי להשפיע על פעילות המשתמש בדף. זאת בכדי לשפר ביצועים וחווית משתמש.**

**יוצרים אוביקט שלxmlhttprequest שהוא משמש להעברת המידע בין הקליינט והסרבר.**

**דוגמה בסיסית:**

var req = new XMLHttpRequest(); //יצירת אוביקט הבקשה

req.open("GET","url"); // הגדרת סוג הבקשה:method,url,async

req.send(); //שליחה

req.onreadystatechange = function() {//רישום לארוע שינוי סטטוס הבקשה

    if (req.readyState == 4 && req.status == 200) {

        console.log(req.responseText);

    }

}

**Promise**

**השימוש בתחביר ה-promise  הוא מועיל במיוחד במקרים שבהם אנחנו צריכים לחכות שפונקציה אחת תסיים כדי שנוכל להריץ את הפונקציה הבאה בתור**

**באמצעות תחביר ה-Promise נוכל לכתוב קוד מודולרי כי הוא קריא יותר וקל יותר לתחזוקה.**

**Promise  הוא אובייקט של JavaScript  שבתוכו אנחנו שמים קוד, שמחזיר אחת משתי תוצאות, הצלחה או כישלון.**

**resolve – מעין "דגל" שמורם, ערך שמודלק ומודיע שהיתה הצלחה לפונקציה ואפשר להפעיל את הפונקציה הבאה. זה יכול להיות בסיום ביצוע הפונקציה וזה גם יכול להיות באמצע הפונקציה, בזמן שנניח כבר אין השפעה אם פונקציה אחרת תתחיל לפעול במקביל.**

**reject – מעין "דגל" שמורם, ערך שמודלק ומודיע שקיימת שגיאה. לא בזמן שבאמת יש שגיאת error, שאז התוכנה לבד כבר מזהה את השגיאה ואותה יש לתפוס בעזרת try-catch. אלא, נניח היה ניסיון לשלוף נתונים, והנתונים הגיעו ריקים, יהיה חסר טעם להמשיך לקרוא לפונקציות הבאות, ולכן יש להודיע על "שגיאה". כלומר, קיימת תקלה. אין להמשיך לקרוא ליתר הפונקציות המקושרות בזו אחר זו.**

**FETCH**

**Fetchהיא הדרך המודרנית לביצוע Ajax. אחרי שבמשך יותר מעשור נאלצנו לכתוב קוד Ajax מסורבל או להיעזר בספריות כדוגמת jQuery ו-Angular, התקן החדש של JavaScript מציע תחביר קומפקטי, מודרני וגמיש יותר.**

**fetch הוא promise. לפיכך, הוא יכול להחזיר אחת מ-2 תגובות אפשריות. הצלחה או כישלון. כך זה נראה:**

* **fetch('https://example/')**

**.then((res) => { *// do something with the response* })**

**.catch((error) => console.log(error))**

**במקרה של כישלון, נקבל תגובה שתטופל על ידי ה-catch, ובמקרה של הצלחה נטפל באמצעות בלוק then**

**את חילוץ המידע מה- body נבצע באמצעות מתודות שמספק לנו fetch. לדוגמה כשאנחנו מצפים ל-json  נשתמש במתודה שתפקידה לחלץ את ה- json**

**המתודה json  של fetch  מחזירה גם היא promise  שגם אותו צריך לתפוס, אז נשרשר בלוק then  נוסף:**

**fetch('https://example/’)   
.then((res)=>res.json())  
.then((data)=>console.log(data))   
.catch((error) => console.log(error))**

**דוגמה לPOST בFETCH:**

**fetch('https://reshetech.co.il/dummy-api/’,   
{ method : 'post’,   
mode: 'cors’,  
 headers: { 'Content-Type': 'application/json', *// sent request*   
'Accept' : 'application/json' *// expected data sent back* },   
body: JSON.stringify({min: 1, max: 100}) })   
.then((res) => res.json())   
.then((data) => console.log(data))   
.catch((error) => console.log(error))**