Promises

מהו Promise?

Promise - הבטחה, הינו אובייקט המייצג פעולה אסינכרונית הנמצאת כרגע בתהליך ביצוע.

פעולה זו תסתיים בעתיד בהצלחה או בכישלון ובאמצעות אובייקט Promise ניתן להעביר את אותה פעולה בין מקומות שונים בתוכנית.

דבר זה מאפשר לנו להגדיר את קוד הטיפול בפעולה במקום שונה מהמקום בו התחלנו לבצע את הפעולה, ועדיין להשאיר את הקוד ברור וקריא.

דוגמא:

הפונקציה setTimeout הינה פעולה אסינכרונית (כיון שאינה מתבצעת כאשר התוכנית קוראת אותה, לאחר משך הזמן שהוגדר שתופעל).

הקוד הבא יציג את ההודעה start ולאחר 5 שניות את ההודעה hello:

```
alert('--- start');

setTimeout(function() {
     alert('hello');
}, 5000);
```

הפעולה האסינכרונית בדוגמא היא המתנה, וקוד הטיפול בפעולה מוגדר במקום בו התחלנו את הפעולה.

שכתוב הקוד כך שישתמש בפונקציה חיצונית לא ישנה מהותית את חלוקת התפקידים בקוד. ניתן לראות שגם במצב כזה החיבור בין הפונקציה החיצונית לאירוע מתבצע מאותו קטע הקוד שיוזם את הפעולה.

```
function after_timeout() {
     alert('hello');
}
alert('--- start');
setTimeout(after_timeout, 5000);
```

כיצד מממשים Promise?

הבטחה מאפשרת שינוי בחלוקת התפקידים באמצעות הפרדה בין ייזום פעולה אסינכרונית לבין הגדרת קוד הטיפול בפעולה זו. הבטחה הינה אובייקט העוטף פעולה אסינכרונית ומאפשר העברת הפעולה עצמה למקום אחר בתוכנית.

דוגמא לקוד המשתמש ב Promise:

```
var p = new Promise(function(resolve, reject) {
     setTimeout(function() {
          resolve();
     }, 5000);
});

p.then(successFunc, failureFunc);
```

```
function successFunc() {
     alert('hello');
}
function failureFunc() {
     alert('fails');
}
```

הקוד מחולק לשני חלקים:

תחילה מגדירים אובייקט Promise ושומרים אותו במשתנה p.

האובייקט Promise מגדיר פונקציה בעלת 2 פרמטרים:

resolve - הפעלת הקוד בעת הצלחת הפעולה האסינכרונית.

reject - הפעלת הקוד בעת כישלון הפעולה האסינכרונית.

שנית, מפעילים את הפונקציה then אשר שייכת למשתנה p מסוג Promise.

הפונקציה then

הפונקציה then גורמת להפעלת הקוד בעת הצלחה או כישלון מקבלת כפרמטר 2 **פונקציות**:

פונקציה שתתרחש בעת הצלחת הפעולה האסינכרונית - פונ' זו מופעלת בעת ש resolve מופעל.

פונקציה שתתרחש בעת כשלון הפעולה האסינכרונית - פונ' זו מופעלת בת ש reject מופעל.

(שם הפונקציה אינו שם שמור).

ניתן לכתוב את הפונקציה ישירות לתוך then.

דוגמא:

```
p.then(function () {
     alert('hello');
}, function () {
     alert('fails');
});
```

הבטחות אמורות להועיל במקומות בהם הקוד בחלק השני נמצא במרחק ניכר בתוכנית מהקוד בחלק הראשון.

אובייקט ה Promise שימושי מאוד בכל הספריות החדשות ובמסגרת זו לא נמחיש את הצורך בשימוש ב Promise.

דוגמאות נוספות לשימוש ב Promise

טעינת קבצים

פקד זה מאפשר לבחור קבצים ולטעון אותם, באמצעות JavaScript.

```
<input type="file" id="myFile" />
```

האירוע onchange של הפקד myFile מתרחש כאשר בוחרים קובץ.

קריאת הקובץ הינה פעולה אסינכרונית המתבצעת ע"י הפונקציה readAsText(...). האירועים onload ו onerror מתרחשים בעת הצלחה וכשלון הפונקציה readAsText. ניתן לכתוב את הקוד ישירות בתוך האירועים, וניתן להעביר את ביצוע הקוד למקום אחר ע"י שימוש ב Promise.

בדוגמא הבאה מועבר הביצוע למקום אחר ע"י שימוש ב Promise:

```
myFile.onchange = function () {
               var myPromise = new Promise(function (resolve, reject) {

                var file = myFile.files[0];
                var reader = new FileReader();
                reader.onload = function (e) {
                    resolve(e.target.result);
                };
                reader.onerror = function (e) {
                    reject(e);
                }

                reader.readAsText(file);

            });


            myPromise.then(success, failure);
}

function success(contentOfFile) {
     alert(contentOfFile);
}
function failure(err) {
     alert(err);
}
```

הפונקציה resolve - מפעילה את הקוד שיתרחש בעת הצלחה. הקוד נכתב בהמשך בפונקציה success.

הפונקציה reject - מפעילה את הקוד שיתרחש בעת כישלון. הקוד נכתב בהמשך בפונקציה failure.

```
reader.onload = function (e) {
                    resolve(e.target.result);
                };
                reader.onerror = function (e) {
                    reject(e);
                }
```

שימי לב כי הפונקציות success ו failure נכתבו במקום אחר במקום, אחרי האירוע.

ניתן להשתמש לחילופין בקוד למבדא לכתיבת תוכן הפונקציות.

קוד רגיל:

```
myPromise.then(success, failure);
```

ביטוי למבדא:

```
myPromise.then(contentOfFile ta => alert(contentOfFile), err=> alert(err));
```

פניה לשרת ב Ajax

פניה לשרת ע"י Ajax הינה פעולה אסינכרונית (Asynchrony javascript and xml).

בדוגמא הבאה נבצע פניית Ajax ע"י jQuery ונעביר את הפעולה שתתבצע בעת הצלחה או כישלון הפניה לשרת, למקום אחר בקוד, ע"י שימוש ב Promise.

כדי להימנע מהצורך להקים שרת, ניתן להוסיף לפרויקט קובץ טקסט שיכיל אובייקט JSON.

שימי לב: כאשר האובייקט נכתב בקובץ טקסט, יש להקיף בגרשיים גם שמות של תכונות, ולא רק ערכים טקסטואליים.

תוכן קובץ הטקסט:

```
{"id":1, "name":"sara", "hoby":"sleep"}
```

קוד הפניה לשרת:

```
btnLoad.onclick = function () {

            var prom = new Promise(function (resolve, reject) {

                $.ajax({
                    url: "TextFile1.txt",
                    method: "get",
                    success: function (dataFromServer) {
                        resolve(dataFromServer);
                    },
                    error: function (err) {
                        reject(err);
                    }               });

            });

prom.then(data => p1.innerHTML = JSON.parse(data).name, err => alert(err));

}
```

התוצאה על האלמנט p1:

sara