JS Advanced

30/12/2024

תודה על ההקשבה

אני וצוות המכללה כאן עבורכם

Async and Await

Async and Await

JS_Advanced_Async_Await.html

בחלק זה נלמד מגוון נושאים.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה התחביר של async ו-await ?
- מה הם הכללים לשימוש?
- מדוע פונקציה חייבת להיות מוגדרת כאסינכרונית על מנת שנוכל לממש בה await?
- מה זה בא לפתור לנו?
- מה עלינו לעשות על מנת שנוכל לאפשר מימוש של awaitב- top level של המסמך?
- מתי עלינו להצהיר על מסמך כמודול?

```
async function myFunction1() {
    return "Hola Class";
}

myFunction1().then(
    function (value) { console.log(value); },
    function (error) { console.log(error); }
);


function myFunction2() {
    return Promise.resolve("Hola Class");
}

myFunction2().then(
    function (value) { console.log(value); },
    function (error) { console.log(error); }
);
```

Async and Await

בחלק זה נלמד מגוון נושאים.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה התחביר של async ו-await ?
- מה הם הכללים לשימוש?
- מדוע פונקציה חייבת להיות מוגדרת כאסינכרונית על מנת שנוכל לממש בה await?
- מה זה בא לפתור לנו?
- מה עלינו לעשות על מנת שנוכל לאפשר מימוש של awaitב- top level של המסמך?
- מתי עלינו להצהיר על מסמך כמודול?

JS_Advanced_Async_Await1.html

הגדרת פונקציה אסינכרונית

```
var x = 1;

async function connectToServerDemo1() {

    let myPromise = new Promise(function (funcResolve, funcReject) {

        console.log("Connecting to the server, reply in 3000 milliseconds");
        setTimeout(function () {
            if (x == 1) {
                funcResolve("Connecting success");
                x = 0
            } else {
                funcReject("Connecting error");
                x = 1
            }
        }, 3000)
    });
    return await myPromise;
}


function demo1Option1() {
    connectToServerDemo1().then(
        function (value) { console.log("OK", value); },
        function (error) { console.log("error", error); }
    );
}
```

הגדרת המתנה לתשובה

קריאה לפונקציה המצריכה המתנה

Async and Await

JS_Advanced_Async_Await2.html

בחלק זה נלמד מגוון נושאים.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה התחביר של async ו-await ?
- מה הם הכללים לשימוש?
- מדוע פונקציה חייבת להיות מוגדרת כאסינכרונית על מנת שנוכל לממש בה await?
- מה זה בא לפתור לנו?
- מה עלינו לעשות על מנת שנוכל לאפשר מימוש של awaitב- top level של המסמך?
- מתי עלינו להצהיר על מסמך כמודול?

הגדרת הסקריפט כמודול

הגדרת פונקציה אסינכרונית

```
<script type="module" >

    const URL = 'http://universities.hipolabs.com/search?country=Israel';

    async function getPromise(url) {
        let myPromise = new Promise((resolve, reject) => {
            let request = new XMLHttpRequest();
            request.open("GET", url);
            request.onload = () => {
                if (request.status == 200) {
                    return resolve(request.response);
                } else {
                    return reject(error.message);
                }
            };
            request.send();
        })
        return await myPromise;
    }

    let promise2 = await getPromise(URL);
    console.log({ ...JSON.parse(promise2) });

</script>
```

הגדרת המתנה לתשובה

קריאה לפונקציה המצריכה המתנה

תרגול Async & Await

צרו קובץ חדש בשם **JS_Async_Await** לטובת הנושא ופתרו את התרגילים לפי הסדר חשוב להדפיס הודעות הצלחה ושגיאה למשתמש וכמובן לבדוק כל תרגיל בסיום.

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקראו לה getData. פונקציה זו תדע לקבל פרמטר שהוא שם של פוקמון. פונקציה זו תדע לפנות ל-API של פוקמונים על מנת להביא מידע על הפוקמון שקיבלנו כפרמטר. כתובתו של ה-API היא: https://pokeapi.co/
Ex-2	באמצעות שימוש במחלקת Promise ו-XMLHttpRequest, פנו ל-API וייבאו את הנתונים המבוקשים. הפונקציה תחזיר את ה-Promise.
Ex-3	צרו פונקציה אסינכרונית חדשה בשם printData. פונקציה זו תבקש מהמשתמש שם של פוקמון, לאחר מכן הפונקציה תשתמש בפונקציה getData על מנת להעביר את שם הפוקמון ולקבל את התוצאה. בסוף פונקציה זו תמיר את התוצאה ל-JSON ותדפיס את התוצאה לקונסול באמצעות שימוש ב-await.
Ex-4	צרו פונקציה חדשה בשם setPokemons, ייבאו בה 5 רשימות של יכולות של פוקמונים על ידי שימוש בפונקציה getData. לבסוף הציגו יכולות אלו על גבי הדפדפן ללא יצירת אלמנטים מראש במסמך ה-HTML באמצעות שימוש ב-await.
Ex-5	שילחו שמות שגויים של פוקמונים ל-API, טפלו בשגיאות באמצעות שימוש ב-try & catch כך שבמידה והתקבלה שגיאה אז תוצג הודעה תואמת למשתמש.

getData

country	"Israel"

printData

pokemonName	Prompt
data	await getData(pokemonName)

setPokemons

abilities	[ability1, ability2]

Fetch

Fetch

JS_Advanced_Fetch1.html

JS_Advanced_Fetch2.html

בחלק זה נלמד מגוון נושאים.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה תפקידה של הפונקציה fetch?
- איזה פרמטרים היא מקבלת?
- במה זה שונה משימוש ב-XMLHttpRequest?
- מה הם היתרונות שבשימוש?
- כיצד נשתמש בפרמטר options?
- כיצד נממש באמצעות שימוש ב- async ו-await?
- איזה מתודות נוכל להפעיל על האובייקט המתקבל? (ה-response)

```
function demo2() {
    fetch('https://restcountries.com/v3.1/name/israel')
        .then(function (response) {
            console.log(response);
            return response.text();
        })
        .then(function (myText) {
            console.log(myText);
            let myJson = JSON.parse(myText)
            console.log(myJson);
        })
}

async function demo3() {

    let response = await fetch('https://restcountries.com/v3.1/name/israel');
    console.log(response);

    let myText = await response.text();
    console.log(myText);

    let myJson = JSON.parse(myText)
    console.log(myJson);

}
```

תרגול Fetch API

צרו קובץ חדש בשם **JS_Fetch** לטובת הנושא ופתרו את התרגילים לפי הסדר
חשוב להדפיס הודעות הצלחה ושגיאה למשתמש וכמובן לבדוק כל תרגיל בסיום.

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקראו לה fetchData. פונקציה זו תדע לקבל פרמטר שהוא כתובת של API אינטרנטי. צרו משתנה חדש וקראו לו res, כערך תנו לו fetch לכתובת שהתקבלה כפרמטר. הפונקציה תחזיר את המשתנה res.
Ex-2	פנו לכתובת API לבחירתכם באמצעות הפונקציה שיצרנו. הדפיסו את התשובה לקונסול (ללא המרה ל-JSON) באמצעות שימוש ב-then. במקרה של שגיאה הדפיסו הודעת אזהרה לקונסול באמצעות שימוש ב-catch.
Ex-3	המירו את התשובה המתקבלת ל-JSON באמצעות שימוש ב-then והדפיסו אותה לקונסול.
Ex-4	צרו פונקציה אסינכרונית חדשה וקראו לה fetchData2. פונקציה זו תדע לקבל פרמטר שהוא כתובת של API אינטרנטי. צרו משתנה חדש וקראו לו res, כערך תנו לו fetch לכתובת שהתקבלה כפרמטר באמצעות שימוש ב-await, הפונקציה תחזיר את המשתנה res.
Ex-5	נסו להשתמש בפונקציה fetchData2 על ידי שליחת ערכים נכונים ושגויים באמצעות שימוש ב-try ו-catch והפיכת הנתונים ל-JSON. באמצעות שימוש ב-finally, הדפיסו את התוצאה בהתאמה.

fetchData

fetch(URL).then()	**res**

fetchData2

await fetch(URL)	**res**