ES6

מיוחדות

מיוחדות ב- JS שמאפשר הגישה למשתני var - כולם
אליהם מגדולות ה scope שלהם.

הגישה של משתנים לגלובלי ה scope - hoisting

משתנים מגדירים ה scope - יש לי אזור - scope
מחוץ ל scope לא מגדירים
אי נמצאים כלי var מחוץ ל scope אבל כן אפשר
משתנה שמוגדר עם var קורה hoisting.

let -

ההבדל בין let ל var,
var - לא הסתרה אז לכל הסקריפט מבלבל (לא הרס אלוהים)
let - אלוקי scope יותר מתוך ה block שנוצרה. (לולאות, תנאים)

```
var x = "global";
let y = "global";
console.log(this.x); // global
console.log(this.y); // undefind;
```

פונקציה הגדרה let אותה רק באותו בלוק
יש גם hoisting לפונקציה, אבל אם עושים הגדרה של הפונקציה לתוך משתנה - הוא לא עולה

closures - סגירות - scope מתוך scope
פונקציה סגירות יכולה לגשת כל המשתנים שלה וגם המשתנים
פונקציה פנימית יכולה ל 3 סוגים של scope - יש, - scope גלובלי, את ה scope הלוקלי - שלה,
ואת ה scope של הפונקציה הגלובלית יותר
closer לא reference למשתנים שבפונקציה הגלובלית
אפשר לגשת ל IIFE - פונקציה שמופעלת את עצמה מיידית
(function(){}) () -
פונקציה

```
var callbacks = []
for(var i = 0; i < 10; i++) {
    callbacks.push(function() { console.log(i) })
}
callbacks[7]() ->
```

callback =

מחוץ לפונקציה - אבל scope מגדיר מה עדיין פונקציה שמגדיר את i

1. const מקומי / let - לפי סוג הנכסים.

... (שלב קודם) - spend Attribute - אפשר לקבל הרבה פרמטרים אחרי אלה לראות

בס"ד

Javascript - scope & closure

תרגילים

מה יהיה הפלט של התרגילים הבאים:

1.

```
{
    let callbacks = [];
    for (var i = 0; i <= 2; i++) {
        callbacks[i] = function () {
            return i * 2;
        };
    }
    console.log(callbacks[0]() === 0); false
    console.log(callbacks[1]() === 2); false
    console.log(callbacks[2]() === 4); false
}
```

תקני את התרגיל!
(יש 3 דרכים שונות לכך)

2.

```
function test() {
    console.log(a);    undefind
    console.log(foo());    2
    var a = 1;
    function foo() {
        return 2;
    }
}
test();
```

3.

```
{
    (function () {  — IIFE
        var a = b = 5;
        var c = d = 2;
    })();
    console.log(b);    — 5 — יצר
    console.log(c);    error
}
```

b יצבירים אין לו let/var
אותו scope

4.

```
var a = 1;
function someFunction(number) {
    function otherFunction(input) {
        return a;
```

```
        }
        a = 5;
        return otherFunction;
    }
    var firstResult = someFunction(9);
    console.log(firstResult(2));
```

5

5.

```
var a = 1;
function b() {
    a = 10;
    return;
    function a() {}
}
b();
console.log(a);
```

1

6.

```
var fullname = 'John Doe';
var obj = {
    fullname: 'Colin Ihrig',
    prop: {
        fullname: 'Aurelio De Rosa',
        getFullname: function () {
            return this.fullname;
        }
    }
};
console.log(obj.prop.getFullname());
var test = obj.prop.getFullname;
console.log(test());
```

Aurelio De Rosa

John Doe

7.

```
function func2() {
    if (true) {
        var a = 5;
    }
    console.log(a);
}
```

5

8.

```
var a = 6;
function test() {
    var a = 7;
    function again() {
        var a = 8;
```

```
        alert(a);
    }
    again();
    alert(a);
}
test();
alert(a);
```

9.

```
var b = 1;
function outer() {
    var b = 2
    function inner() {
        b++;
        var b = 5;
        console.log(b)
    }
    inner();
}
outer();
```

תרגילים

10. Write a function that would allow you to do this:

```
var addSix = createBase(6);
addSix(10); // returns 16
addSix(21); // returns 27
```

11. Use a closure to create a private counter

אם יש לי אובייקט לא, ונלכת לקבל ערך מאותו אובייקט אבל

```
var obj = {
  prop1: " ";
  prop2:  ;
}
```

var

var [a] = [3]; // fail-safe destructing

אפשר לשים גם מערכים

Symbol Type

זו אובייקט-מספר מיוחד, ל סימבול שונים גם שווה ל אותו ערך אחר

Iterators

מאפשר לעבור איך אנחנו שולטים על הלולאה. לולאה מאוחרת איטרטור

Generator

function*

מאפשר לעצור ולהמשיך פעולה עם הפקודה yield

ל מפה שקולה ל text הוא מקבל מחרוזות שעבור מאוחר הם ה status ו

Map — weakmap — שניהם לוקליים

key : value

Javascript - scope & closure
רשימות לתרגילים

① • בשביל - הבדל בין let ל- var שמתוך הלולאה ל- let נוצר בכל איטרציה scope נפרד
• ולכן לפונקציה שלה כל פעם יהיה את i אחר לפונקציה שלה לפונקציה

```
for(let i=0; i<=2; i++){
    callbacks[i] = func(i);
}
```

• IIFE - פונקציה שמפעלת עצמה עצמה -

```
for(let i=0; i<=2; i++){
    callbacks[i] = (function(num){
        return function(){
            return num*2;
        }
    })(i);
}
```

② כל משתנה שמצהירים עליו ב var עולה hoisting ולכן ההצהרה לא נוצרת

let timeout = setTimeout(function() { לאחר 1000 },
clearTimeout(timeout);

promise, מי נרצה callback-ים בה קיים, אחרת - מה בפנים

let promise = new Promise(function(resolve, reject) {
// קוד
});

- בדיקה ותקיעה שלמה שלמה חשוב -

resolve - מה מה שעשינו הצלחנו
reject - מה מה שעשינו נכשל

אומר לתוכנית לסיים את הקוד
לסיים אחרת

try catch (סמן !) מה כזה חשוב

בס"ד

שם: שני קולדין

קבוצה: ב

בוחן מס' 2 Javascript – async

1. מה הפלט של הקוד הבא?

```
function job(state) {
    return new Promise(function(resolve, reject) {
        if (state) {
            resolve('success');
        } else {
            reject('error');
        }
    });
}

let promise = job(true);
promise
        .then(function(data) {
            console.log(data);
            return job(false);
        })
        .catch(function(error) {
            console.log(error);
            return 'Error caught';
        })
        .then(function(data) {
            console.log(data);
            return job(true);
        })
        .catch(function(error) {
            console.log(error);
        });
```

הפלט:

2. מה הפלט של הקוד הבא?

```
var promise = new Promise(function(resolve, reject) {
    setTimeout(function() {
        resolve('hello world 1');
        resolve('hello world 2');
        resolve('hello world 3');
        resolve('hello world 4');
    }, 1000);
});
promise.then(function success(data) {
    console.log(data);
});
```

(א) hello world 1

(ב) hello world 2

(ג) hello world 3

(ד) hello world 4

3. מה ה-state של כל promise (לאחר הרצת הקוד)?

```
let data = "Do you know js well?";
let firstPromise = new Promise((resolve, reject) => resolve("ok"));

let secondPromise = new Promise((resolve, reject) => {
    if (data.includes("js"))
        resolve("yes");
    reject("no");
});

let thirdPromise = new Promise((resolve, reject) =>
    isNaN(data) ? reject("NaN") : resolve("valid")
);

let fourthPromise = secondPromise.then(x => thirdPromise)
        .then(res => console.log(res)).catch(res => console.log(res));
```

(א) firstPromise: fulfilled

(ב) secondPromise: _____

(ג) thirdPromise: _____

(ד) fourthPromise: _____

4. מה ההבדל בין שני קטעי הקוד הבאים (A ו- B):

```
let promise = new Promise((resolve, reject) => reject("Error"));

promise.catch(x=>console.log(x)); //(A)

promise.then(null,x=>console.log(x)); //(B)
```

(א) אין הבדל

(ב) A תקין ו-B לא תקין

(ג) A לא תקין ו-B תקין

5. מה הפלט של הקוד הבא?

```
console.log("4")
new Promise((resolve, reject) => resolve("2")).then(x => console.log(x));
console.log("3")
```

(א) 1 2 3

(ב) 2 3 4

(ג) 3 2 4

(ד) 3 4

AsyncJS

{asyncJs}

מבוא

callback - מאפשרים פונקציה לעשות דברים בלי לעצור אותה
פונקציה שמעבירים אותה כפרמטר לפונקציה אחרת

settlement - מקבלת 2 פרמטרים, callback לעשות ובמידה אין - זה סימן לזה אסינכרוני
אסינכרוני
הפונקציה האסינכרונית עדיין לוחצת הפונקציות הסינכרוניות !

Sanp

אוריגל שמפעל יכולות אסינכרוני לעדיין קבל מה בלעם אז לזה.
אסינכרוני
promise מחזיר שזה אמר מה זה בלוח.
• מצב המתנה - לא הבטחנו pending, אין תוצאה indifinud מצבה של בקשה restuck
• מצב שני - הצלחה fullfilled
• מצב שלישי - כשלון rejected

- פונקציה שיש לה אז resolve או reject - צה לחזר
- אחרי שפונקציה - צה יחזיר promise הבטחה שיש pending נניח שאחרי שזו
תוצאה יחזירה את הבטחה הבאה - לא הראשונה זה לא indifinud)

Promise API

- resolve - resolve מחזיר promise חדש
- reject - reject " " " "
- all - promise מחזיר promise-ים ומקבל מערך של
  (.then) - מחזיר אותו שרק אחרי - או מקבלים
- race -

זה גם promise - מבטיח לפני הפונקציה async
אבל לא מבטיח מתי יתבצע כולל שמדבר - await
מבטיח ל async

issues - איך

מנהל תלויות מקומי node package manager

כן מתקינים את הפרויקט / npm init -y

נוצר קובץ לוגים yes לכל השאלות

כמה משתמשים - רשת

אינטרנט - סקיל

כמה מתחזקים - אדם

readme / תאור	dependencies
	רשימה תלויות הפרויקט
	הגרסאות תלויות נדרשות

טיפים:

מאוד שמורגלים תלויות נוספות - package-lock

node_modules

- npm i להתקין

- npm uninstall שם תלות i להסיר

versions שם - view npm - לראות את כל הגרסאות

npm update / latest / מספר גרסה שם i npm להוסיף גרסה

אפשרי לגרסה הנוכחית מכל התלות

מעדכן את כל התלות - npm u

---

webpack לוקח את הפרויקט כ-entry point לקובץ אחד

כמה לוגו קובץ קונפיג

הפקודה (npx webpack) זו אפשרות להגדיר תוך כ package.json

build: webpack

npm run-script build