FULLSTACK

הדרכה על JS

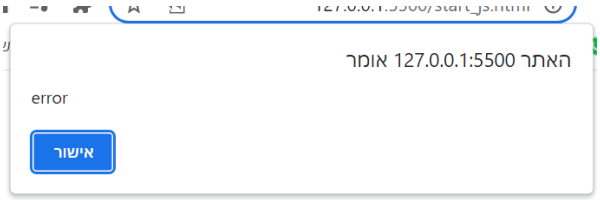
לקראת קורס FULLSTACK

תמר כהן

יוני 2022

- elert –
הפונקציה alert("xxx") תיצור חלון אזהרה קופץ כשננסה להריץ את האתר, על החלון יהיה רשום xxx

```
<script>
  alert("error");
</script>
```



- הערה – האתר ייטען רק אם נלחץ על "אישור"

- D.O.M –
1. היררכיה משפחתית – למשל <h1> בן של <body> כי ה <h1> יושב בתוך <body>
2. לכל תגית מוגדרת התנהגות – למשל זו תמונה
3. יש attributes ייחודיים לכל תגית ויש גם כלליים – למשל class או ID הם כלליים
4. לתגית שנפתחת ונסגרת קיים inner html – למשל ל אין אבל ל <h1>...</h1> יש
5. לכל תגית קיים עיצוב ברירת מחדל וניתן לשנות אותו

- קריאות בזמן אירועים –
  - Onload – קריאה בזמן טעינת העמוד
  - Onclick – קריאה בזמן לחיצה על כפתור
  - Oninput – קריאה בזמן הזמנת ערך קלט
  - Onchange – קריאה בזמן שינוי (למשל בשינוי של select)

- פונקציה בJS
  - הגדרת פונ'

```
function nameOfFunction() {

// statements

}
```

  - זימון פונ' יכול להתבצע ע"י לחיצה על כפתור, טעינת עמוד, גלילת עמוד, זימון בפונ' אחרת וכו'
  - גישה ל attributes של תגיות לפי ID של אותה תגית –
גישה לטקסט (innerHTML)

```
document.getElementById("ID_name").innerHTML
```

גישה לעיצוב , רוחב וכו'

```
document.getElementById("30").style.color = "red";
document.querySelector("#id_img").width = 200;
```

גישה למקור תמונה (במקרה זה שינוי התמונה המוצגת)

```
document.querySelector("#id_img").src = "images/s.jpg";
```

  - הערה – אין הבדל גדול בין querySelector לבין getElementById בכל הקשור למתן גישה לID מלבד הכתיבה עם #
  - קריאה לפונ' colorRed() באמצעות לחיצה על כפתור תיראה כך –

<button onclick="colorRed()"> click </button>

- קבלת גישה לערך טקסט שהוכנס לinput מסוג text עם ID ששמו "input_id" –
document.querySelector("#input_id").value;

- אפשר להגדיר פונ' ככה שתרוץ בטעינת העמוד ע"י
Window.onload = function nameOfFunction() {

}

- פונ' ללא שם –
Let x = function nameOfFunction() {

}

ואז ניתן לקרוא ל x(); ובעצם "לקרוא" לפונ'

- משתנים –
הכרזה על משתנה תתבצע ע"י let או var (אין צורך לעשות משתנה ספציפי של int או string למשל)
- גישה לאובייקט שנצמא בתוך מערך של json –

```
<script>
  let arr = [[{ num1: 10 }, { num1: 40 }], [{ num1: 20 }]];

  function check1() {
    document.write("," + arr[0].map((o) => o.num1));
    document.write("," + arr[1][0].num1);
    document.write("<br>");
    check2();
  }

  function check2() {
    document.write("," + arr[0][0].num1);
    document.write("," + arr[0][1].num1);
    document.write("," + arr[1][0].num1);
    document.write("<br>");
    check3();
  }

  function check3() {
    for (let i = 0; i < arr.length; i++) {
      let innerArr = arr[i];
      for (let j = 0; j < innerArr.length; j++)
        document.write("," + innerArr[j].num1);
    }
  }

</script>
```

פלט -

,10,40,20
,10,40,20
,10,40,20

- פונ' נחשבת א-סינכרונית אם יש בה לפחות פעולה א-סינכרונית אחת
  - פונ' א סינכרונית לא יכולה להחזיר ערך (שכן יכול להיות שהערך ימשיך להשתנות אחרי ההחזרה)
  - פונ' א-סינכרונית מחזירה טיפוס מסוג promise וכך נוכל להבטיח שההחזרה תתבצע אחרי ביצוע הפעולות
  - דוגמא –

הפונ' הנ"ל תחזיר את הערך 0 שכן הפעולה 2*num מתבצעת "אחרי" ההחזרה

```
function GetData(num)
{
    let total=0;
    setTimeout(() => total = num*2 ,2000)

    return total;
}


let result =  GetData(5)
console.log(result)
```

הפונ' הנ"ל תחזיר את הערך 10 בעזרת השימוש בpromise

```
function GetData(num)
{
    let prom = new Promise(resolve =>
        {
            setTimeout(() => resolve(num*2),2000)
        })

    return prom;
}

GetData(5).then(x => console.log(x))
```

- תהי funcA פונ' שמקבלת מערך ומחזירה את סכום האיברים אחרי 2 שניות בעזרת promise

כדי לקרוא לתוצאה של funcA אחרי המתנה –

אופציה 2 לא מצריכה את async בשם הפונ'

```
async function funcB(arr) {
  // option 1 - get data from async function funcA
  let output = await funcA(arr);
  console.log(output);
  // option 2 -
  funcA(arr).then((x) => console.log(x));
}
```

- פקודות על מערכים

```
//          0     1     2     3     4
let arr = ["a", "b", "c", "d", "k"];

arr[4] = "e"; //4 מחליף את הערך בתא
// ["a" , "b" , "c" , "d" , "e"]

arr.push("f"); // מוסיף תא בסוף המערך
// ["a" , "b" , "c" , "d" , "e" , "f"]

arr.unshift("z"); // מוסיף תא בתחילת המערך
// ["z" , "a" , "b" , "c" , "d" , "e" , "f"]

let last = arr.pop(); // מוחק את התא האחרון ומחזיר את הערך
// ["z" , "a" , "b" , "c" , "d" , "e"]
// last = "f"

let first = arr.shift(); // מוחק את התא הראשון ומחזיר את הערך
// ["a" , "b" , "c" , "d" , "e"]
// first = "z"

arr.splice(3, 2); // מוחק מהתא השלישי (כולל) כשני תאים
// ["a" , "b" , "c"]

let indexOfA = arr.indexOf("a");
let indexOfD = arr.indexOf("d");
// מחזיר את המיקום של האיבר, אם האיבר לא קיים יחזיר -1
// indexOfA = 0
// indexOfD = -1

let partial1 = arr.slice(2); // חותך את המערך מהתא השני (כולל) עד הסוף
// partial1 = ["c"]

let partial2 = arr.slice(1, 2); // חותך את המערך מהתא הראשון (כולל) עד התא השני
(לא כולל)
// partial2 = ["b"]
```

- לולאות –

```
let arr = ["a", "b", "c", "d", "e", "f"];

for (let i = 0; i < arr.length; i++) {}
// לולאת פור רגילה

for (let i in arr) {
}
// צורה נוספת ללולאת פור רגילה , רץ על אינדקסים
// 0 <= i < arr.length

for (let item of arr) {
}
// לולאת פור שרצה על הערכים בתוך התאים ולא על האינדקסים
// item = arr[i]

arr.forEach(function (item) {
  // what to do with item ?
});
// לולאת פור שרצה על הערכים בתוך המערך
// וקוראת לפונ' אנונימית שתבצע פעולה
// item = arr[i]
// הפונ' יכולה לקבל גם את האינדקס
// function (item , i){}
```

- פונקציית map , filter , reduce –

```
let arr = ["Yaron", "Dov", "Gilat", "Dana"];
let arrLonger = arr.filter((o) => o.length > 4);
let arrLength = arrLonger.map((o) => o.length);
let sum = arrLength.reduce(function (total, element) {
  return (total += element);
});
console.log(sum);
```

בשורה הראשונה נתון מערך מחרוזת

צריך להחזיר את סך התווים שיש במחרוזות אשר אורכן גדול מ4

בשורה 2 – באמצעות פונ' פילטר ניצור מערך חדש , עם המחרוזות אשר אורכן גדול מ4 בלבד

["Yaron" , "Gilat"]

בשורה 3 – באמצעות פונ' מפ ניצור מערך חדש שאבריו הם אורכי המחרוזות

[5,5]

בשורה 4 – באמצעות פונ' רדיוס נסכום אם איברי המערך

- לולאת foreach

```
let given = [
  [1, 6, 3, 9],
  [6, 12, 5, 21],
  [4, 11, 23, 1],
];
let total = 0;
given.forEach(function (innerArr) {
  innerArr.forEach(function (innerCell) {
    total += innerCell;
  });
});
console.log(total);
```

בפונ' זו נחשב את סכום האיברים בכל תתי המערכים שבמערך
הפונ' רצה בלולאה על המערך הראשי
ולאחר מכן רצה בלולאה על איברי המערך הפנימי וסוכמת

- פקודות Math

```
let rnd = Math.random();
// get a random num in range (0,1)
// example - 0.02365259582590573

let rndRange = Math.random() * 10;
// get a random num in range (0,10)
// example - 8.958526708577164

let rndRangeRound = Math.round(rndRange);
// if rndRange is closer to its upper val it returns its upper val
// if rndRange is closer to its lower val it returns its lower val
// example -
// rndRange = 8.958526708577164
// rndRangeRound = 9

let rndRangeUpper = Math.ceil(rndRange);
// return its upper val
// example -
// rndRange = 8.958526708577164
// rndRangeUpper = 9
// we will never get 0 in this example

let rndRangeLower = Math.floor(rndRange);
// return its lower val
// example -
// rndRange = 8.958526708577164
// rndRangeLower = 8
// we will never get 10 in this example

let pow = Math.pow(4, 2);
// return 4^2
// pow = 16

let max = Math.max(1, 2, 3, 4, 5, 6);
// max = 6

let min = Math.min(1, 2, 3, 4, 5, 6);
// min = 1

let maxArr = Math.max(...[0, 1, 2, 3, 4, 5]);
// maxArr = 5

let minArr = Math.min(...[0, 1, 2, 3, 4, 5]);
// minArr = 0
```

- דרך נוספת ליצור אלמנט (בדרך זו קל יותר להאזין למה שקורה לאלמנט מול המשתמש)

```
let btn = document.createElement("button");
btn.innerHTML = "btn";
let p = document.querySelector("#id_body").appendChild(btn);
```

יצירה של כפתור, שמופיע עליו "btn" בתוך גוף העמוד (id_body)

- תנאים –

```
let x = 0;

// if else

if (x == 0) {
  // condition is true
  // what to do
} else {
  // condition is false
  // what to do
}

// switch case

switch (x) {
// switch checks like === then , if we get data from user
// we will need to check the cases as strings
// example - case "0" :
// check if x === 0
  case 0:
    //what to do
    break;
// if x != 0 check if x === 1
  case 1:
    //what to do
    break;
// if x != 0 and x != 1 check if x === 2
  case 2:
    //what to do
    break;
// if x != 0 and x != 1 and x != 2 then do
  default:
    // what to do
    break;
}

// ternary if


(x > 0) ? /*what to do if true*/ : /*what to do if false*/ ;
```