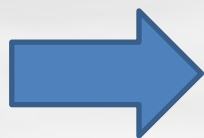


JAVA SCRIPT

JAVA SCRIPT

**הוצג לראשונה בשנת 1995 כדרך להוסיף תוכניות לדפי אינטרנט
בדפדפן Netscape Navigator.
מאז כל הדפדפנים המודרניים אימצו את השפה להרצת קטעי קוד
ללא "הידור" לצד הלקוח.
כיום שפה JS הינה השפה הפופולרית ביותר בעולם הפרונט-אנד.**



svcollege
ללמוד. לדעת. לעבוד.

Create SCRIPT

ישנם 2 דרכים ליצור סקריפט:

1. ניתן ליצור אלמנט script בכל חלק בעמוד שלנו (HEAD OR BODY).

```
<script type="text/javascript"> ...  
</script>
```

2. ניתן ליצור קובץ נפרד עם סיומת .js ולהוסיף אותו לעמוד שלנו.

```
<script type="text/javascript" src="Filename.js"> </script>
```

מיקום הסקריפט מאוד קריטי. רצוי למקם אותו בסוף העמוד לאחר טעינת כל האלמנטים בעמוד.**

OUTPUT

הצגת ערכים: ישנן דרכים רבות להצגת ערכים ב JS. אנחנו נתייחס ל 3 עיקריות:

1. הצגת תוכן ב Console.Log

2. הקפצת הודעה חיצונית למשתמש.

3. הכנסת ערכים לתוך אלמנט ספציפי.

```
console.log('...');
```

הדפסת ערכים ל console:

```
console.log('hello world');
```

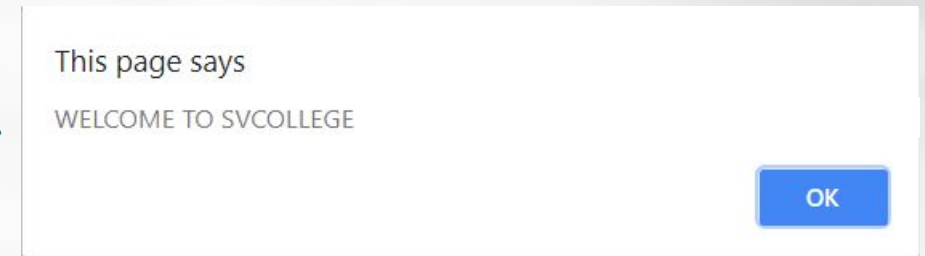
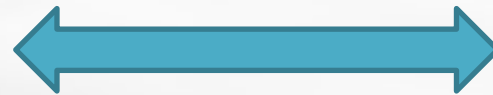
ניתן לראות את התוצאה ב devTools
F12 בדפדפן.

window.alert

הקפצת הודעה חיצונית למשתמש:

window.alert('תוכן')

```
<body>  
.....  
<script type="text/javascript">  
window.alert("WELCOME TO SVCOLLEGE");  
</script>  
</body>
```



****לא ניתן לשנות את עיצוב ההודעה. תלוי בדפדפן בלבד!**

innerHTML

הכנסת ערכים לתוך אלמנט ספציפי:

```
document.getElementById('idName').innerHTML = value;
```



ID ספציפי התביות על האלמנט לפי המזהה
הייחודי שלו
ID



איזה ערך
יוחלף בתוך האלמנט



ניתן לחפש לפי ID, CLASS, NAME**
**הערך יחליף את הערך שהיה בתוך האלמנט.

innerHTML

קבלת התוכן של אלמנט ספציפי:

```
alert(document.getElementById('idName').innerHTML);
```



התבייתות על האלמנט לפי המזהה
הייחודי שלו
ID



ID ספציפי



תוכנו של האלמנט

ניתן לחפש לפי ID, CLASS, NAME**
**הערך יחליף את הערך שהיה בתוך האלמנט.

Data Types

יחידות זיכרון: שימוש בשפת תוכנה מתבצעת בעזרת הקצאת זיכרון לטובת חישובים ושימושים של התוכנה.

לדוגמא:

נניח וניצור "שדה" של סיסמא ונרצה לוודא את תקינותו, נשמור את השדה בתוך יחידת זיכרון וכך נוכל לבצע על הסיסמא כל פעולה שנרצה.

דוגמא נוספת:

נניח ויצרנו משחק ובמשחק אנחנו יכולים לצבור נקודות, את הנקודות נשמור בתוך יחידת הזיכרון וכך נוכל במהלך המשחק להוסיף עוד נקודות למאזן ולהציג את מספר הנקודות בסוף המשחק.

Data Types

יצירת יחידת זיכרון

let **name**;

let – מתאר את הטיפוס של יחידת הזיכרון.
name – שם המשתנה שהגדרנו.

const **info**;

const – הגדרת טיפוס **קבוע**, משתנה שלא ניתן לשנות לאחר היצירה.
Info – שם המשתנה שהגדרנו.

****שם המשתנה יכול להיות כל שם, ללא מספרים בתחילתו וללא רווחים.**

****שם המשתנה צריך להיות שם משמעותי כדי שנוכל להבין מה הוא משמו.**

****כתיבה של משתנים תעשה ב camelCase.**

Operator Assignment

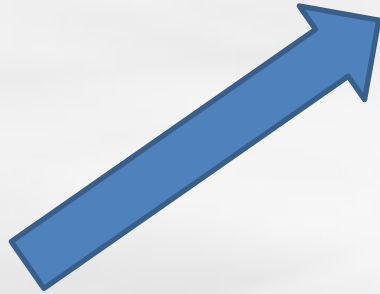
=

אופרטור השמה: בניגוד למתמטיקה ששם משתמשים בסימן ה"שווה" לבדיקת שוויון, אופרטור זה משמש להכנסת נתונים לתוך המשתנה שלנו. כל מה שמימין לשווה יישמר במשתנה משמאל.

Operator Assignment

=

name = value



שם המשתנה



ערך

Operator Assignment

=

דוגמאות להכנסת ערכים:

```
let name = 'svcollege';  
let number = 35;  
name = 'sv college';  
let newName = 'SBCOLLEGE';  
name = newName;
```

Arithmetic Operators

אופרטורים מתמטיים: ניתן לבצע פעולות מתמטיות לפני הכנסתם למשתנה.

לדוגמא:

```
let sum = 5+5;
```

1. המערכת תחבר $5+5$.

2. התוצאה 10 תישמר לתוך המשתנה `sum`.

Arithmetic

OPERATOR	Description
()	סוגריים
% / *	כפל חילוק שארית
- +	חיבור חיסור

```
let num = (5+5*12)/2;
```

מהי התוצאה לדעתכם?

** גם בתכנות יש דגש ל - סדר פעולות חשבון

Shortcuts

```
let num = 1;  
num = num + 1;
```

בקטע קוד זה אנו רואים הוספה של 1 למשתנה:

```
let num = 1;
```

ניתן לקצר זאת בדרך הבאה:

```
num +=1;
```

ואפילו כך:

Shortcuts

קיצור דרך	ללא קיצור
num+=value	num = num + value
num-=value	num = num - value
num*=value	num = num * value
num/=value	num = num / value
num%=value	num = num % value
num++	num = num + 1
num--	num = num - 1

Math

הינה ספריה אשר מכילה פונקציות מתמטיות רבות:

PI() - מחזיר את פאי

```
Math.PI(); // 3.1459...
```

round() – מעגלת את המספר

```
Math.round(92.5); // 92
```

```
Math.round(92.7); // 93
```

pow(number , exponent) - מבצעת חזקה של המספר

```
Math.pow(5,2); // 25
```

sqrt(number) - מבצעת שורש על מספר מסוים

```
Math.sqrt(81); // 9
```

String

מחרוזת הינה רצף של תווים המורכב מכל תו במקלדת – אותיות מספרים וסימנים מיוחדים.
כל ערך שנכתב ב' ' נחשב למחרוזת.
** כולל רווח!

name

```
let firstName = 'shem';  
let lastName = 'bar';  
let name = firstName + ' ' + lastName;
```

shem bar

String Actions

פעולות: ישנם פונקציות (פעולות שנלמד בהמשך) אשר מובנות עם השפה ו - ניתן להיעזר בהם.

length – מחזיר את אורך המילה.

```
let str = 'svcollege';  
let len = str.length;
```

0 1 2 3 4 5 6 7 8 = length 9
s v c o l l e g e

String Actions

indexOf(...) מחזיר את **מיקום** האות/מחרוזת -

```
let str = 'hello world';
```

```
let index = str.indexOf('world');
```

מיקום התחלה 6 n

******במידה והערך לא נמצא יוחזר -1

String Actions

substring(**start**,
end) - פעולה אשר מחזירה חלק מ - המחרוזת -

```
let str = 'hello world';
```

```
let subStr = str.substring(6 , 11);
```

```
subStr = world
```

שים לב ש - הפרמטר השני **end**, צריך להיות גדול מ - **start**.
אם נזין 7 בפרמטר השני נקבל את w, אם נזין 8 נקבל wo וכן הלאה.

String Actions

replace(**source** , **new**) פעולה אשר מחליפה ערך בערך -

```
let str = 'hello world';
```

```
let newStr = str.replace('world', 'shem');
```

```
newStr = hello shem
```


String Actions

charAt(index). פעולה אשר מחזירה אות במיקום ספציפי -

```
let str = 'hello world';
```

```
let letter =
```

```
str.charAt(0);
```

```
letter = h
```

Condition IF



svcollege
ללמוד. לדעת. לעבוד.

במידה ונרצה שתנאי יקרה רק כאשר ... נשתמש בתנאי
"אם".

הקוד שבבলוק יתבצע רק כאשר התנאי "**אמת**".

```
if(true) {
```

```
.....
```

```
}
```

Comparison Operators

פירוש	סימן
שווה ל..	==
שונה מ..	!=
גדול מ..	>
קטן מ..	<
גדול או שווה	>=
קטן או שווה	<=
NOT	!

פירוש	סימן
או	
וגם	&&

Comparison Operators

EXAMPLE

```
let age = 18;  
  
if(age <= 18) {  
    alert('access denied');  
}  
if(age > 18) {  
    alert('access complete');  
}
```

מה התוכנית עושה?

else

אחרת – במידה ותנאי "אם" לא מתקיים, נבנס ל `else`.
ל `else` אין תנאי!

```
if(true) {
```

```
.....
```

```
}
```

```
else {
```

```
}
```

if / else

EXAMPLE

```
let age = 18;
```

```
if(age <= 18) {  
    alert('access denied');  
}  
else {  
    alert('access complete');  
}
```

else if

אחרת אם – במידה ונצטרך לבדוק יותר מ 2 תנאים נוכל להוסיף

let bestCollege = 'svcollege'; "אחרת אם" לסקריפט שלנו.

```
if(bestCollege.length == 2) {  
  
    alert('To short');  
  
} else if (bestCollege.length <2 ) {  
  
    alert('?!?');  
  
} else {  
    if(bestCollege == 'svcollege') {  
        alert(' 😊 ');  
    } else {  
        alert(' 😞 ');  
    }  
}
```



Debugger

בעזרת devTool, ניתן לרוץ על הסקריפט "צעד אחר צעד".
בשביל לבצע זאת יש להגדיר breakpoint בקוד.

```
<script>  
.... code  
....  
debugger  
.... code  
....  
</script>
```



מרגע שהקטע קוד מגיע ל breakpoint
ניתן לרוץ על הקוד step by step
F10 - המשך צעד אחד קדימה
F11 - הכנס לתוך הפונקציה

function

פונקציה הינה קטע קוד שניתן לשלוח ולקבל ממנה ערכים.

```
function name() {
```

```
}
```

יצירת פונקציה:

אז למה זה טוב בדיוק?

function

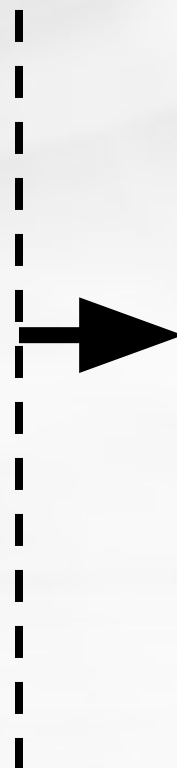


svcollege
ללמוד. לדעת. לעבוד.

דוגמא:

```
function alertHelloWorld(){  
    alert(' hello world ');  
}
```

```
alertHelloWorld();  
alertHelloWorld();  
alertHelloWorld();
```



בדוגמא זו יקפצו 3
הודעות בזו אחר זו עם
המשפט hello world

function

דוגמא לפונקציה המקבלת ערכים:

```
function alertSomething (printMe){  
    alert(printMe);  
}
```

```
alertSomething('hello');  
alertSomething('class');
```

return value

פונקציה יכולה להחזיר ערכים מכל סוג:

```
function func ( ) {  
  let result = 'hello world';  
  return result;  
}
```

```
let res = func();
```

**** שימו לב שהערך המוחזר מהפונקציה חייב להישמר במשתנה או להיות מודפס.**

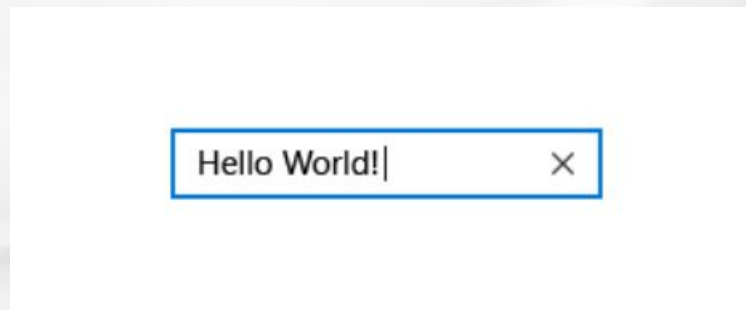
function

הפעלת פונקציה בעזרת כפתור

```
<button onclick="funcName()"> CLICK PLZ </button>
```

הגדרה זו מפעילה את הפונקציה בעת לחיצה על הכפתור

Value from input (type text)



```
let inputValue = document.getElementById('SomeID').value;
```

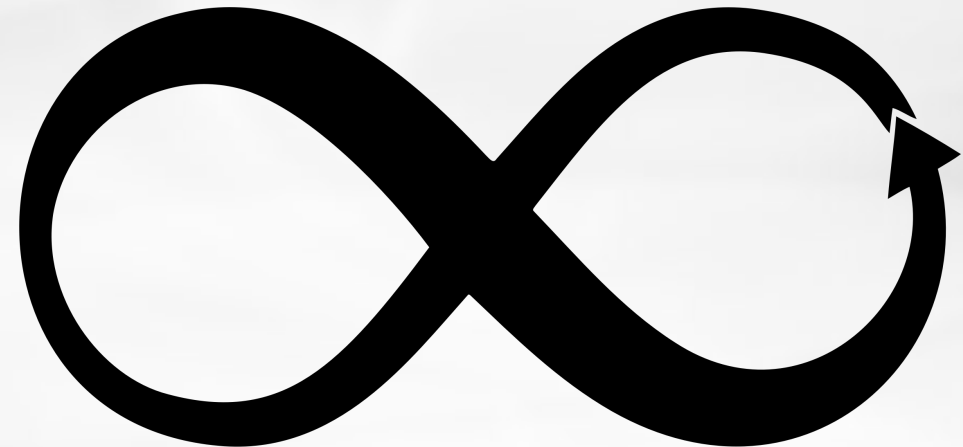
פקודה זו תאפשר את שמירת הנתונים מ - הערך שהתקבל ב INPUT למשתנה.

****שימו לב להפעיל אופציה זו בתוך פונקציה בלבד.**

While loop

לולאה – קטע קוד אשר ממשיך להתבצע כל עוד התנאי נכון.

```
while(true)
{
    .....
    .....
}
```



****יש לשים לב שבמידה ולא נחשוב על תנאי עצירה ה - לולאה תמשיך לעד.**

While example

```
let i = 0;  
while(i < 4) {  
  
    alert('hello');  
    i++;  
  
}
```

i	(While(i<4
0	True
1	True
2	True
3	True
4	False

for

לולאת for – נשתמש בה כאשר אנו יודעים מראש את מספר
הסיבובים שהלולאה אמורה לרוץ.

for(**אתחול ; תנאי ; קידום**)

for example

**דוגמא ל – לולאה שרצה על כל האותיות במחרוזת, במידה
והיא מוצאת ערך מספרי היא מקפיצה הודעה על כך.**

```
let fullName = 'Shem1Bar';  
let i;  
for(i=0 ; i<fullName.length ; i++){  
    if(fullName.charAt(i) >= '0' && fullName.charAt(i) <= '9'){  
        alert('number exists');  
    }  
}
```

for example

```
function checkPrime(num) {  
  let i;  
  if(num == 1 || num == 2)  
    return true;  
  
  for(i=2 ; i < num ; i++) {  
    if(num%i == 0){  
      return false;  
    }  
  }  
  return true;  
}
```

דוגמא לפונקציה המקבלת מספר ומחזירה אם הוא ראשוני או לא.

break & continue

break – פקודה אשר "שוברת" את הלולאה.

דוגמא לשימוש ב break:

```
for(i = 11 ; i != 0 ; i--) {  
    if(i%3 == 0) {  
        break;  
    }  
}
```

מתי הלולאה תעצור ?

break & continue

continue – פקודה אשר ממשיכה לסיבוב הבא של הלולאה.

דוגמא לשימוש ב continue:

```
while(num < 1500) {  
    if(num < 0) {  
        num*=-1;  
        continue;  
    }  
    num++;  
}
```

מה הלולאה עושה ?

Array

מערך הוא סוג של מבנה נתונים.

מערך הינו רצף של תאים צמודים בזיכרון. לכל התאים יש את אותו שם ומיקום יחסי שונה.

דוגמא למערך בעל 5 תאים:

	0	1	2	3	4	5
arr						

Array

יצירת מערך:

```
let arr = [value1, value2, value3, value4, value5];
```

כל איבר במערך מקבל אינדקס החל מ - 0

```
let x = arr[index];
```

גישה לתא ספציפי במערך:

```
arr[2] = 26;
```

דוגמא להכנסת ערך מספרי לתא השלישי במערך:

Array – methods

toString() – מדפיסה את הערכים במערך עם “,” בין כל ערך.

```
let arr = [value1, value2, value3, value4, value5];  
alert(arr.toString());
```

push() & pop () – הוצאת והכנסת ערכים לסוף המערך. תחשבו על המערך בתור מחסנית, הכדור האחרון שהכנסתם יצא ראשון LIFO – last in first out.

```
let x = arr.pop(); // מוציא את הערך האחרון במערך  
arr.push("newValue"); // מוסיף את הערך לסוף המערך
```

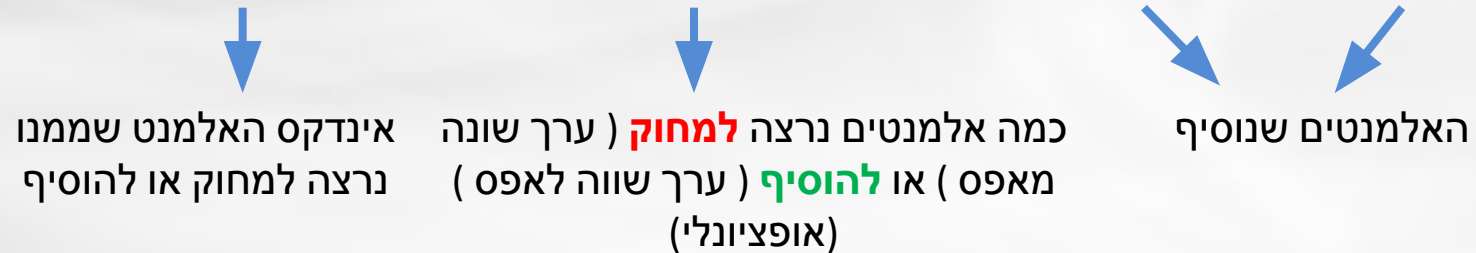
```
let arr = [1, 4, 77, 22, 200];  
let len = arr.push(8); // כעת במשתנה נשמר האורך 6
```

shift() & unshift() – הוצאת והכנסת ערכים לתחילת המערך.

Array – methods

splice()- מוסיפה/מסירה ערכים מהמערך (גם מהאמצע) ומחזירה את הערכים שהוסרו.

```
splice(index, howManyToRemove, "item1", "item2"...);
```

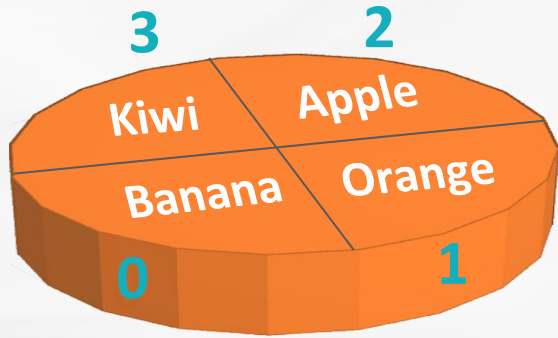


מוסיפה ערכים כאשר howManyToRemove = 0

מסירה ערכים כאשר howManyToRemove != 0

splice() - הוספת ערכים למערך.

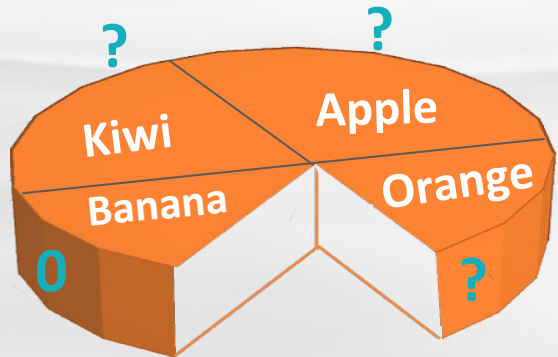
```
let fruits = ["Banana", "Orange", "Apple", "Kiwi"];
```



בואו נדמיין שהמערך שלנו היא עוגה, כך היא תראה:

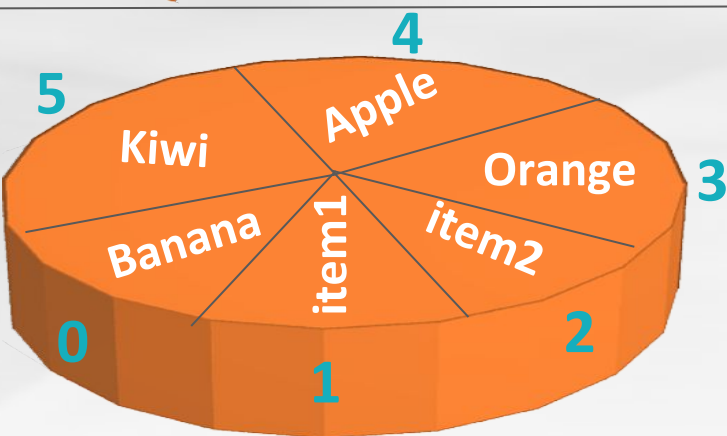
```
splice(index, howManyToRemove, "item1", "item2"...);  
fruits.splice(1, 0, "item1", "item2");
```

כאשר באינדקס שמנו את הערך אחד, נשים את סכין החיתוך על האיבר במקום 1 במערך -
Orange.



```
splice(index, howManyToRemove, "item1", "item2"...);  
fruits.splice(1, 0, "item1", "item2");
```

כאשר שמנו 0, זה אומר שאנו רוצים להוסיף למערך ולא להסיר ממנו, ולכן נפנה מקום ונשאיר את Orange, העוגה תראה כך:

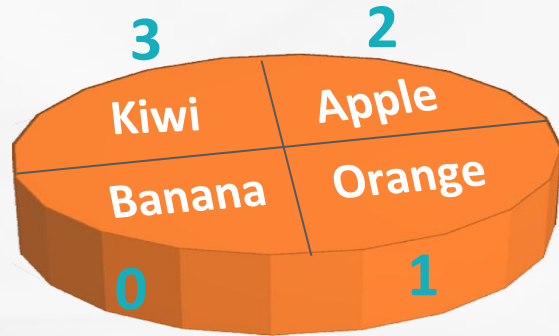


```
splice(index, howManyToRemove, "item1", "item2"...);  
fruits.splice(1, 0, "item1", "item2");
```

כאשר הוספנו שני ערכים לדוגמא (item1, item2), העוגה תראה כך:

splice() - הסרת ערכים למערך.

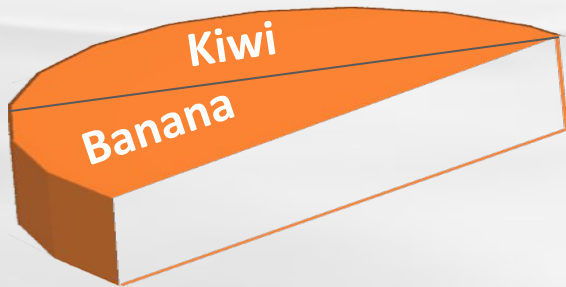
```
let fruits = ["Banana", "Orange", "Apple", "Kiwi"];
```



בואו נדמיין שהמערך שלנו היא עוגה , כך היא תראה:

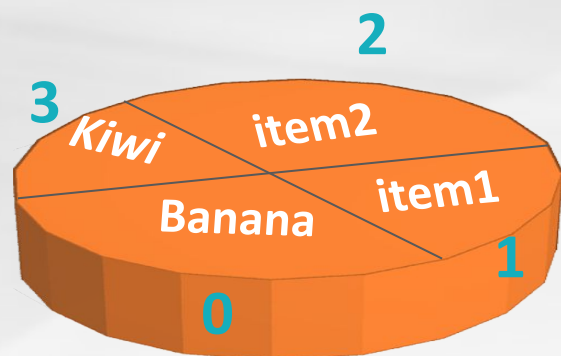
```
splice(index, howManyToRemove, "item1", "item2"...);  
fruits.splice(1, 0, "item1", "item2");
```

כאשר באינדקס שמנו את הערך אחד , נשים את סכין החיתוך על האיבר במקום 1 במערך -
Orange.



```
splice(index, howManyToRemove, "item1", "item2"...);  
Plate = fruits.splice(1, 2, "item1", "item2");
```

כאשר שמנו 2 , זה אומר שאנו רוצים ממיקום 1 לחתוך שתי חתיכות, ולכן המשתנה plate יכיל את Orange, Apple הם יחתכו מהעוגה (יוסרו מהמערך), ולכן העוגה תראה כך:



```
splice(index, howManyToRemove, "item1", "item2"...);  
fruits.splice(1, 2, "item1", "item2");
```

כאשר הוספנו שני ערכים לדוגמא (item1, item2), העוגה תראה כך:

```
fruits = ["Banana", "item1", "item2", "Kiwi"];
```

Array – methods

`sort ()` – פונקציה אשר ממיינת את המערך מהקטן לגדול:

```
let arr = ['b', 's', 'a', 'e', 'b'];  
arr.sort(); // a, b, b, e, s
```

שימוש במיון למספרים:


```
arr.sort(function(a, b){return a-b}); // מיון מערך בסדר עולה  
arr.sort(function(a, b){return b-a}); // מיון מערך בסדר יורד
```

Array - matrix


מערך של מערכים – כל תא במערך מחזיק מערך. משחק צוללות

נחלק שני דפים, בכל דף נצייר כמי לוח שחמט אשר מחולק לשורות ועמודות ובפנים נצייר צוללת כך:

	1	2	3	4	
A					ש
B					ח
C					ק
D					א



	1	2	3	4	
A					ש
B					ח
C					ק
D					ב



כאשר שחקן א' רוצה לפגוע בצוללת של שחקן ב, הוא אומר B3, וזו אכן פגיעה, משום שהצוללת של השורה B עם עמודה 3 אכן

פוגעת בצוללת.



Array- matrix

מערך של מערכים – כל תא במערך מחזיק מערך. משחק צוללות

פניה לאיבר במערך // `mat[row][col]`

```
let mat = [  
  mat[0] ← [1, 5, 9],  
  mat[1] ← [2, 4, 7],  
  mat[2] ← [6, 8, 3]  
];
```

```
mat[1][1] // 4
```

```
mat[2][0] // 6
```



Object

אובייקטים – סוג של מבנה נתונים אשר מחזיק כמה מאפיינים ובכך
מאפשר שמירה של נתונים תחת ארגומנט אחד.

```
let student = {  
  firstName: 'dor',  
  lastName:  
    'dekel',  
  ID: '123456789',  
  GPA: 82.7  
};
```

Object

אובייקטים – סוג של מבנה נתונים אשר מחזיק כמה מאפיינים ובכך
מאפשר שמירה של נתונים תחת ארגומנט אחד.

כאשר רוצים להגדיר אובייקט, מגדירים את שמו (student בדוגמה
הזאת)

סימן =, פותחים וסוגרים סוגריים מסולסלות (להבדיל ממערך שזה
סוגריים מרובעות), ואז ישנו מפתח שווה לערך, השווה כאן זה כמו ב
CSS (:)

```
let student = {  
  firstName: 'dor',  
  lastName:  
    'dekel',  
  ID: '123456789',  
  GPA: 82.7  
}
```


Object

גישה לכל מאפיין מתבצעת באופן הבא:

nameOfObject.**property**

(בהמשך לדוגמא הקודמת):

```
student.lastName = 'levi';  
document.getElementById('..').innerHTML = student.lastName;
```

Object

ובכן היכן היעילות של האובייקט לעומת משתנה?

אם תשים לב תבין שכל דבר בעולם שלנו הוא בעצם אובייקט, שזה אוסף של תכונות, לדוגמה אתם הסטודנטים בעצם אובייקטים, יש לכל אחד מכם שם פרטי, שם משפחה ת.ז וכיו"ב.

```
let student = {  
  firstName: 'dor',  
  lastName:  
    'dekel',  
  ID: '123456789',  
  GPA: 82.7  
}
```

Object

ובכן היכן היעילות של האובייקט לעומת משתנה?

בואו ונאמר שאני רוצה לרשום בקוד שלי, את כל המאפיינים של

הסטודנטים,

בתוך משתנים, בואו נבין עד כמה זה מייגע:

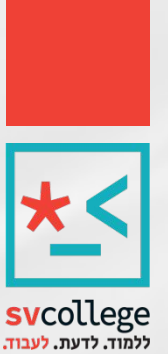
```
let student_firstName1 = 'dor';  
let student_lastName1 = 'Cohen';  
let student_ID1 = '123456789';
```

```
let student_firstName2 = 'dor';  
let student_lastName2 = 'Cohen';  
let student_ID2 = '123456789';
```

```
.  
.   
.
```

Object

מערך של אובייקטים



הפתרון הוא די פשוט, **מערך** של **אובייקטים**! אוקיי יש פה בעצם חיה מוזרה מערך שהאיברים שלו הם בצעם אובייקטים! בואו נדבר אל המחשב בצורה שהוא יבין אותנו: כאשר אמרנו מערך זה אומר סוגריים מרובעות:

```
let arr = []
```

כאשר איבר המערך הוא אובייקט (סוגריים מסולסלות) כך זה יראה (נגדיר 3 אובייקטים ריקים):

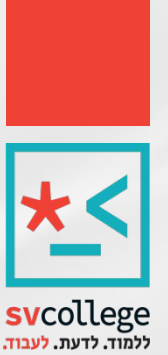
```
arr = [ {}, {}, {} ];
```

בואו ונגדיר את תכונות וערכי התכונות:

```
arr = [    {firstName: 'Dor', lastName: 'Cohen'},
           {firstName: 'Mos', lastName: 'Kline'},
           {firstName: 'Shem', lastName: 'Wise'}
];
```

Object

מערך של אובייקטים



ובכן איך מתייחסים לערך Dor? קודם כל האובייקט ש מכיל אותו הוא האיבר הראשון במערך זאת אומרת באינדקס 0:

```
arr[0]
```

אז arr[0] בעצם מחזיק את האובייקט:

```
{firstName: 'Dor', lastName: 'Cohen'}
```

כאשר רוצים להתייחס באובייקט למפתח מסויים זה עם נקודה (.) ושם המפתח:

```
.firstName
```

ביחד זה יוצא:

```
arr[0].firstName
```

```
// Dor
```

Object – function

ניתן להוסיף פונקציות בתוך אובייקט.

```
let objectName = {  
  variable1 : value,  
  variable2 : value,  
  sum : function(){  
    return this.variable1 + this.variable2;  
  }  
};  
document.getElementById('..').value = objectName.sum();
```

this = המתייחס לאלמנט הנוכחי.

Random number

יצירת מספר אקראי:

```
Math.floor(Math.random() * max - min) + min;
```

דוגמא למספר רנדומלי בין 2 – 10 (לא כולל 10!)

```
let rand = Math.floor(Math.random() * 10 - 2) + 2;
```

exception

try / catch / throw / finally

טיפול בשגיאות - ככל שנדע "לטפל" בשגיאות מראש, כך
נוכל להימנע מקריסות עתידיות.

```
try{  
    .... Code ....  
} catch(e) {  
    alert(e.name);  
}
```


exception

try / catch / throw / finally

דוגמא לאופציה בשימוש במשתנה שלא קיים / חלוקה ב 0

```
var num1 = document.getElementById("..1").value;  
var num2 = document.getElementById("..2").value;  
try {  
    alert(num1/num2);  
} catch(e) {  
    alert(e.name);  
}
```

exception

try / catch / throw / finally

finally - קטע קוד אשר פועל גם במידה ונתפסה שגיאה וגם אם לא

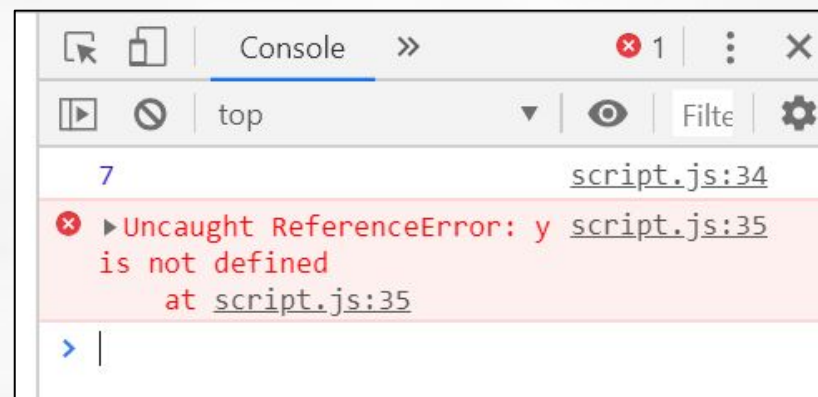
```
try {  
    .... Code ....  
} catch(e) {  
    alert(e.name);  
}
```

finally** הינה הפקודה היחידה שיכולה להופיע גם אחרי פעולת return.

הטיפוסים `let` / `var`

`let` הינו משתנה לוקאלי (מוגדר ו"מת" בתוך ה- scope).
`var` הינו משתנה גלובלי (נוכל "לקרוא" לו מכל מקום).

```
if(true){  
    var x = 7;  
    let y = 5;  
}  
console.log(x);  
console.log(y);
```



arrow function

דרך נוספת ליצור פונקציה

function

```
function func(name) {  
    Console.log('hello' + name);  
}  
func();
```

arrow function

```
let func = (name) => {  
    console.log(`hello ${name}`);  
};  
func();
```

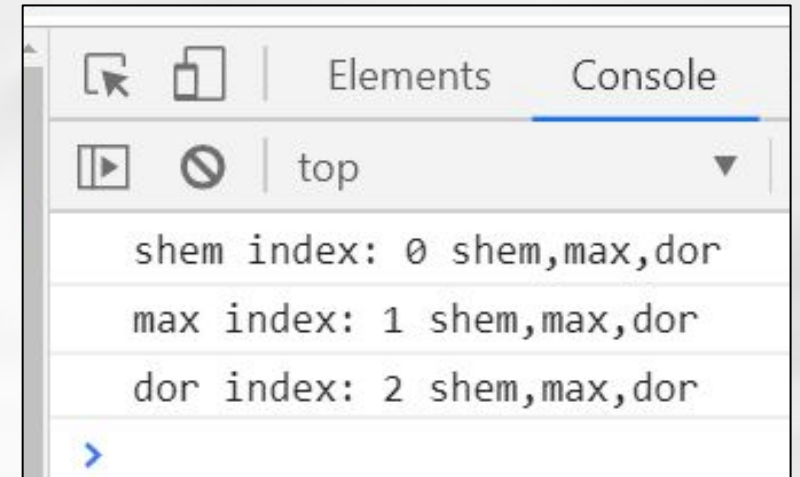
****הפונקציה נשמרת בתוך משתנה, לאחר מכן ניתן לקרוא/לזמן את המשתנה כמו פונקציה רגילה.**

forEach

לולאה ייעודית אשר רצה על כל האברים המערך.

```
const names = ['shem', 'max', 'dor'];
```

```
names.forEach((item, i, names)=> {  
  console.log(`${item} index: ${i} ${names}`)  
});
```



item – ערך האיבר במערך.

i – אינדקס האיבר במערך.

names – שם המערך.

forEach

דוגמא ללולאת forEach ששולחת רשימה לתוך אלמנט שיוצג באתר.

```
script.js  X
const names = ['shem', 'max', 'dor'];
names.forEach((item, i, names) => {
  document.getElementById("myDiv").innerHTML +=
    `<p>${item}</br> index: ${i}<br> ${names}</p>`;
});
```

```
index.html
shem
index: 0
shem,max,dor

max
index: 1
shem,max,dor

dor
index: 2
shem,max,dor
```

- item – ערך האיבר במערך.
- i – אינדקס האיבר במערך.
- names – שם המערך.

filter

יצירת מערך חדש עם סינון של ערכים ספציפיים.

```
const names = ['shem', 'max', 'dor',  
  'orgad'];  
  
let afterFilter = names.filter((item)=>  
  item !== 'dor');  
  
console.log(afterFilter);
```



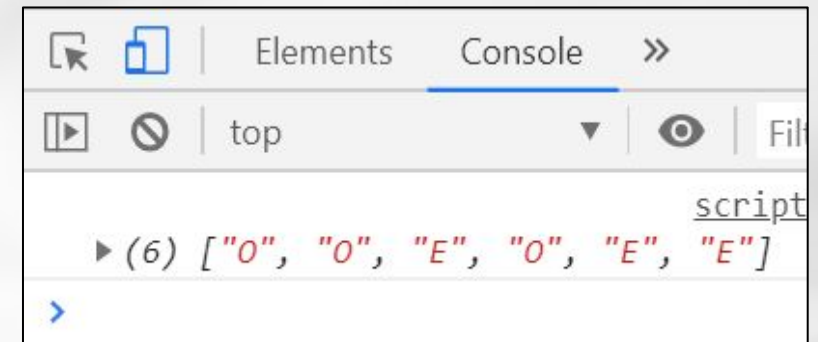
item – ערך האיבר במערך.
names – שם המערך.

map

יצירת מערך חדש עם ערכים שונים.

```
let arr = [1, 3, 4, 5, 6, 6];  
let arrNew = arr.map((item) => {  
  if (item % 2 == 0) {  
    return 'E'  
  }  
  return 'O';  
})
```

```
console.log(arrNew);
```



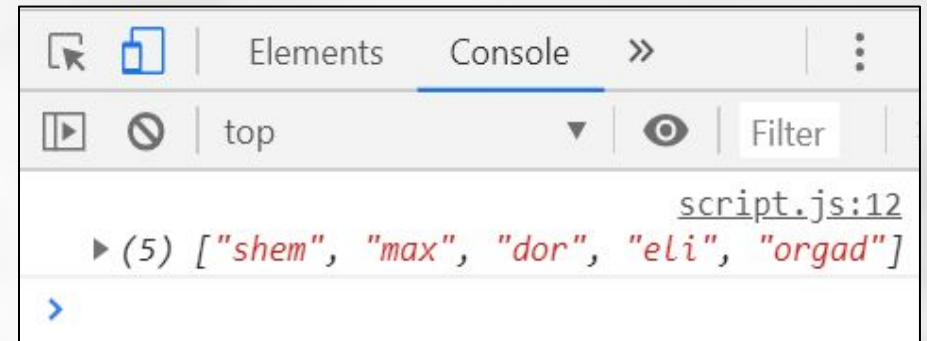
E = Even O = Odd

spread

אפשרות העתקת איברים ממערך אחד למערך אחר.

```
const names = ['shem', 'max', 'dor'];  
let moreNames = [...names, 'eli', 'orgad'];  
// '...' is spread syntax
```

```
console.log(moreNames);
```





Checked or not



svcollege
ללמוד. לדעת. לעבוד.

הערך המתקבל מ"תיבת סימון", ערך בוליאני checked - (True/False)

```
let checkBoxPosition = document.getElementById('..').checked;
```

```
if(checkBoxPosition == true) {  
    return 'continue';  
}  
else {  
    return 'stop';  
}
```

הצגת פרויקטים