



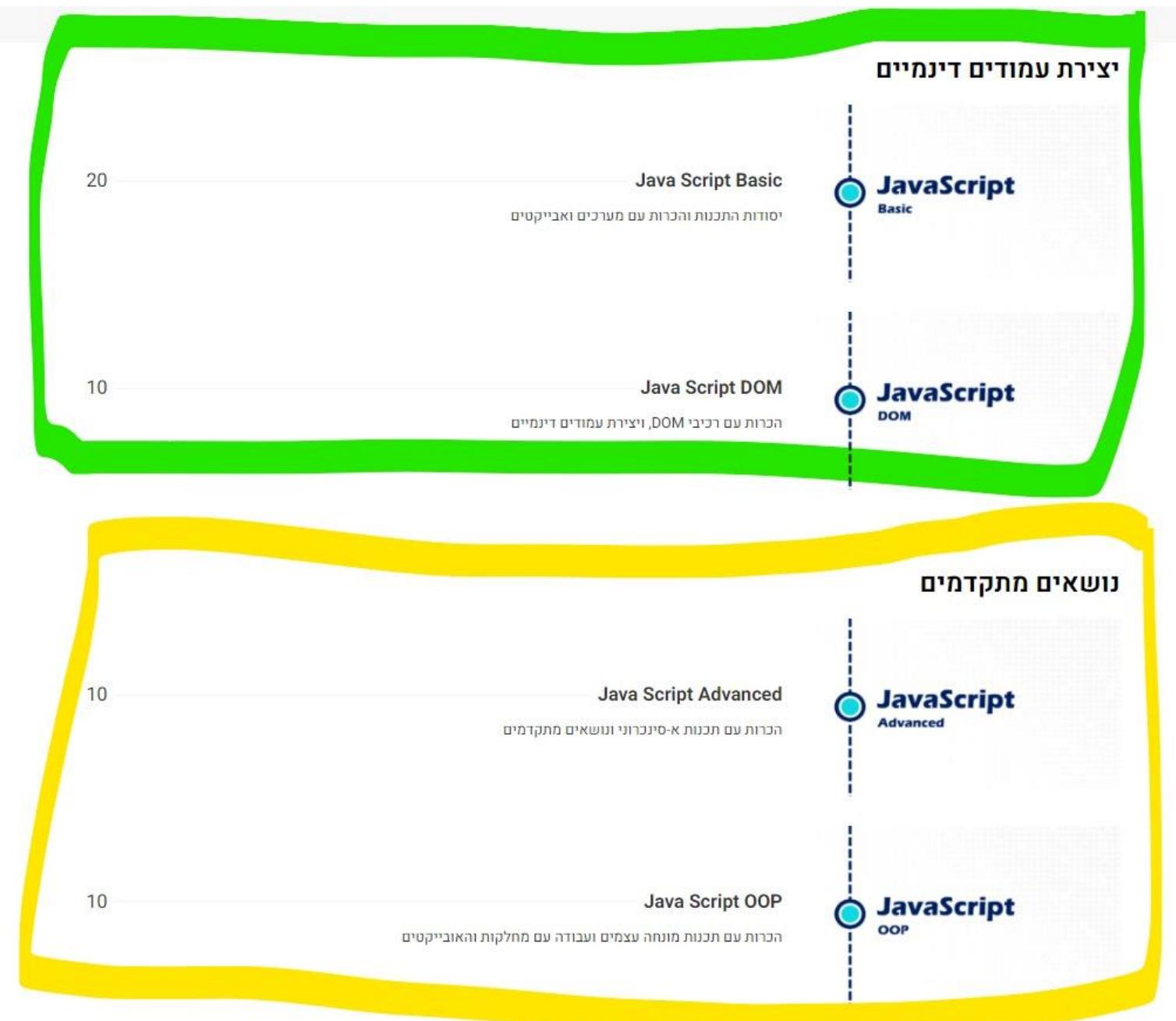
# Advanced Javascript

מרצה: תומר שגיא

שיעור 7

19/11/2023

- דניאל
- יהונתן (תגובה)



# פרויקטים

י הונtan הסביר לכם את פרויקט בונה מסכינים בשיעור האחרון  
י הונtan יסביר את ניהול המשתמשים מחר

## בונה מסכינים

## ניהול משתמשים

# JavaScript Advanced

הכרות עם תכונות אסינכרוני ונושאים מתקדמים

במסגרת הנושא נכיר ונעמיק ידע ב JavaScript ונלמד על אפשרויות מתקדמות ותכונות אסינכרוני הכולן קריאות לשרת.



- Spread ✓
- Shallow copy & Deep copy ✓
- פונקציות חץ - ✓
- שימוש ב try & catch ✓
- Callback ✓
- מבוא לתוכנות א-סינכרוני ✓
- XMLHttpRequest ✓
- Promise ✓
- שימוש ב then - catch ✓
- Async and Await ✓
- Fetch ✓
- Modules ✓

# House Keeping

- נשתמש בגייט שלי (כולל את כל החומרים של דניאל)
- <https://github.com/tomer79sagi/W310523ER>
- אפשר לעשות Clone מהאפליקציה Github Desktop לתיקייה קיימת של דניאל או חדשה
- להסיר את התיקייה של דניאל הקיימת מ-VSC ולהוסיף את התיקייה שלי

# אופרטור Spread



# מה זה ?Spread

It ‘spreads’ out all of the values into a new object or array  
פרישה של ערכים לאובייקט או מערך חדשים

# Spread operator in javascript



```
const arr4 = [4, 5, 6]
const arr3 = [1, 2, 3, ...arr4]
console.log(arr3); // [1, 2, 3, 4, 5, 6]
```



# אופרטור Spread

JS\_Advanced\_Spread\_Operator.html

בחלק זה נכיר אופרטור אשר מאפשר לבצע שימוש נוח ויעיל באובייקטים. בסיסו הנושא תוכלו לענות על השאלות הבאות:

```
let myArray = [11, 22, 33, 44, 55];
```

```
let newArray;
```

```
newArray = myArray;
```

מערך מועבר  
by Reference

```
myArray[0] = 100;
```

```
console.log(newArray);
```

```
console.log(myArray);
```

```
newArray = [...myArray];
```

מערך מועבר  
by Value

```
myArray[0] = 200;
```

```
console.log(newArray);
```

```
console.log(myArray);
```

- מה תפקידו של אופרטור `?spread`
- איך מגדירים אופרטור `(...)` `?spread`
- אילו שימושים נפוצים אלו מכיריהם?
- מה זה `rest parameter`
- האם משנה מיקומו של ה-`rest parameter` בסדר מיקומי הארגומנטים הנשלחים לפונקציה?
- מה הם היתרוןות בשימוש?
- כיצד משתמשים ה `spread` על אובייקט?

# אופרטור Spread

JS\_Advanced\_Spread\_Operator.html

בחלק זה נכיר אופרטור אשר מאפשר לבצע שימוש נוח ויעיל באובייקטים.  
בסיום הנושא תוכלו לענות על השאלות הבאות:

```
let myArray1 = [11, 22, 33, 44, 55];
function print(num1, ...myArr){
    console.log(num1);
    console.log(myArr)
}

print(100, 10, 20, 10, 30);
print(100, myArray1);
```

פונקציה מקבלת rest parameter

קריאה לפונקציה rest parameter

פונקציה מקבלת rest parameter

- מה תפקידו של אופרטור `?spread`
- איך מגדירים אופרטור `(...)` `?spread`
- אילו שימושים נפוצים אלו מכיריהם?
- מה זה `rest parameter`
- האם משנה מיקומו של ה-`rest parameter` בסדר מיקומי הארגומנטים הנשלחים לפונקציה?
- מה הם היתרוןות בשימוש?
- כיצד משתמשים ה `spread` על אובייקט?

# תרגול אופרטור **Spread**

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

תרגיל	תיאור המשימה
Ex-1	<p>צרו מערך בשם <code>cart1</code> וערך תנו לו רשימת מחירי מוצרים.</p> <p>הדפישו את המערך לקונסול, פעם אחת עם שימוש ב-<code>spread</code>, ופעם אחרת ללא שימוש, הבחינו בהבדל של הפלט בקונסול בין שתי הדפסות.</p>
Ex-2	<p>צרו מערך נוסף בשם <code>cart2</code>, כערך תנו לו את <code>cart1</code>.</p> <p>הדפישו את שני המארכים לקונסול.</p>
Ex-3	<p>שנו אחד מהמערכים בערך <code>cart1</code>.</p> <p>הדפישו לקונסול את שני המארכים, מי מהם השתנה?</p>
Ex-4	<p>Appending את הערך של <code>cart2</code>cart וערך תנו לו את רשימת המחירים שב-<code>cart1</code>,</p> <p> רק שהפעם השתמשו ב-<code>spread</code>, לאחר מכן הדפישו את שני המארכים לקונסול.</p>
Ex-5	<p>שנו אחד מהמערכים בערך <code>cart1</code>.</p> <p>הדפישו לקונסול את שני המארכים, מי מהם השתנה?</p>

# תרגול אופרטור **Spread**

צרו קובץ חדש בשם **JS\_spread\_object** לטובת הנושא ופתרו את התרגילים לפי הסדר חשוב להדפיס הודעות הצלחה ושגיאה למשתמש וממן לבדוק כל תרגיל בסיום.

## תרגיל טיאור המשימה

### Person { }

<code>firstName</code>	“Gal”
<code>lastName</code>	“Lavi”
<code>email</code>	“gal@email.com”

צרו אובייקט ליטרלי בשם **person** וכערכו תנו לו שם פרטי, שם משפחה וכתובת אימייל.

Ex-1

צרו אובייקט נוסף בשם **personCopy** כערך תנו לו את **person**. הדפיסו את שני המרכיבים לקונסול.

Ex-2

שנו את האימייל של **personCopy**. הדפיסו לקונסול את שני האובייקטים, מי מהם השתנה?

Ex-3

צרו אובייקט ליטרלי חדש בשם **newPerson** וכערכו תנו לו את **person**, רק שהפעם עשו זאת באמצעות שימוש ב-**spread**.

Ex-4

שנו את השם הפרטיאן של **newPerson**. הדפיסו לקונסול את שני המרכיבים, מי מהם השתנה?

Ex-5

# Shallow copy & Deep copy



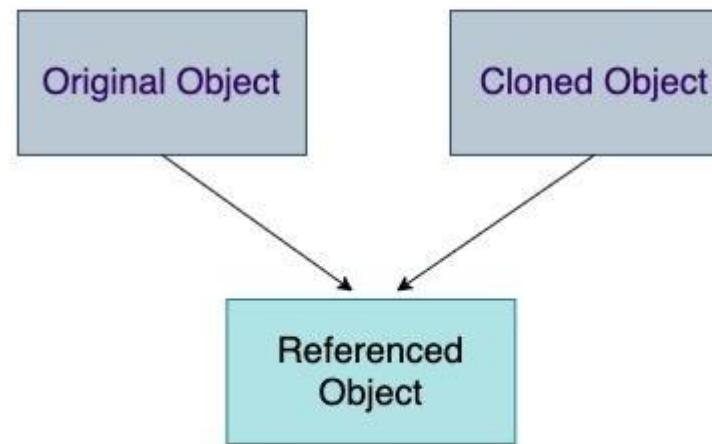
# deep copying vs. shallow copying

A deep copy means that all of the values of the new variable are copied and disconnected from the original variable.

A shallow copy means that certain (sub-)values are still connected to the original variable.

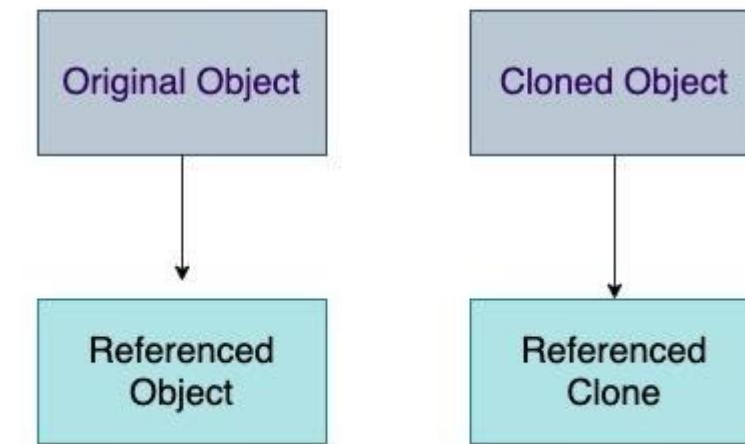
By Reference /  
מצבייע לזיכרון

Shallow Clone



By Value /  
ערךים בזיכרון

Deep Clone



# העתיקת פרימיטיביים – Deep Copy

```
const a = 5  
let b = a // this is the copy  
b = 6  
console.log(b) // 6  
console.log(a) // 5
```

כשמעתיקים משתנים פרימיטיביים, הם תמיד העתקים  
אמיתיים, **כלומר לא מחוברים למקור, Deep Copy**

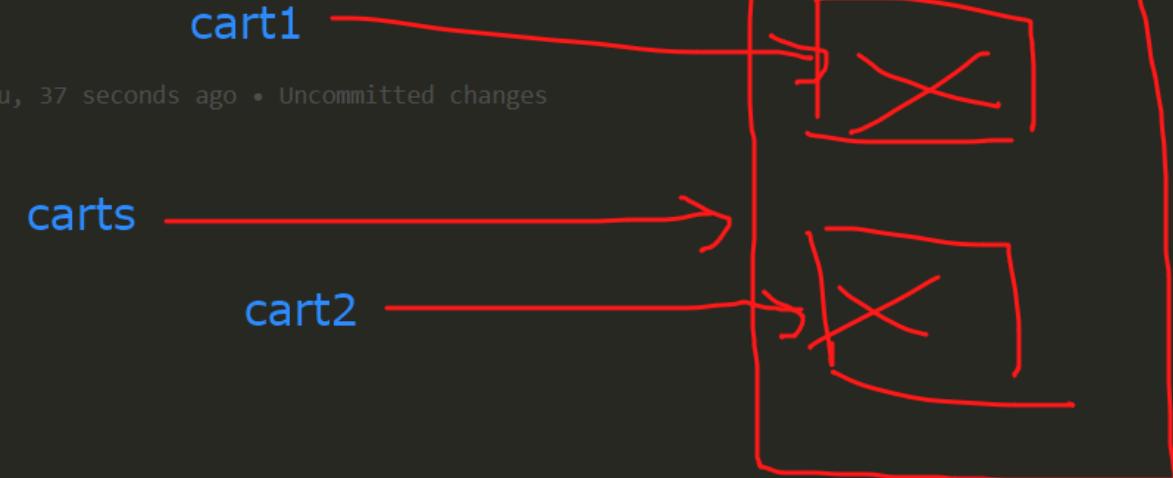
# העתיקת אובייקטים – Shallow Copy

```
const a = {  
    en: 'Hello',  
    de: 'Hallo',  
    es: 'Hola',  
    pt: 'Olá'  
}  
  
let b = a  
  
b.pt = 'Oi'  
  
console.log(b.pt) // Oi  
console.log(a.pt) // Oi
```

כאן זה שונה, כשהמעטיקים אובייקטים (וגם מערכיים הם סוג של אובייקט), מתבצעת כאן העתקת **Shallow Copy**, כי במאנה למקור

js-advanced > 19.11.23 > shallow\_and\_deep\_copy > **JS** script.js > [e] carts

```
You, 38 seconds ago | 1 author (You)
1 // -----
2 // 1. Exercise
3 const cart1 = [4, 30, 76, 100];
4 const cart2 = [1, 600, 56, 60];
5
6 const carts = [cart1, cart2];      You, 37 seconds ago • Uncommitted changes
7 // console.log(carts);
8
9 carts[1][2] = 99;
10 console.log(carts);
11 console.log(cart2);
12
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Filter (a)

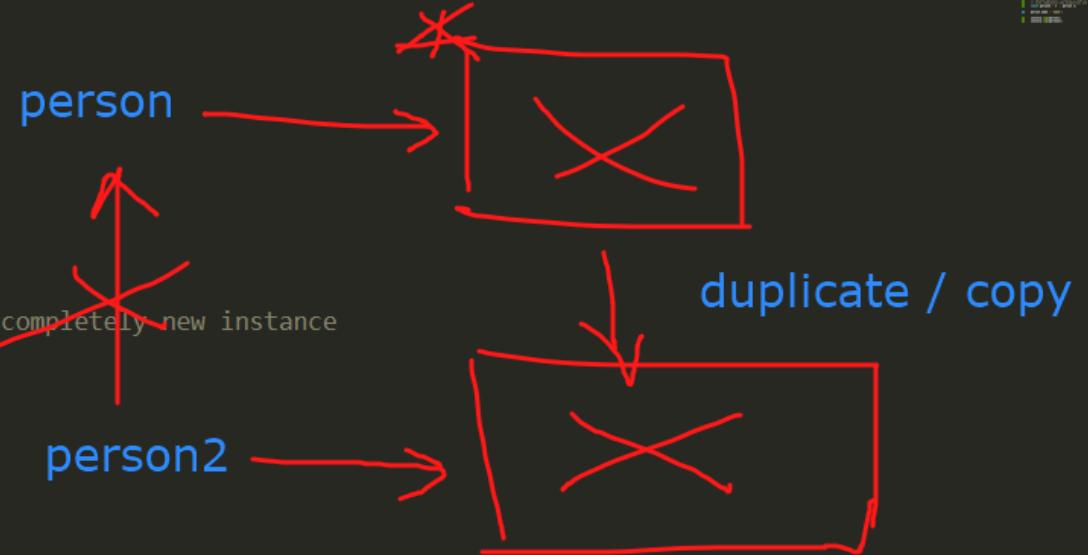
```
length: 2
> [[Prototype]]: Array(0)
> [[Prototype]]: Object
< (4) [1, 600, 99, 60]
  0: 1
  1: 600
  2: 99
  3: 60
  length: 4
> [[Prototype]]: Array(0)
> [[Prototype]]: Object
```

# העתקת אובייקטים – Deep Copy

```
const a = {  
    en: 'Bye',  
    de: 'Tschüss'  
}  
  
let b = {...a}  
b.de = 'Ciao'  
  
console.log(b.de) // Ciao  
console.log(a.de) // Tschüss
```

ע"י שימוש ב-Spread, נוכל לבצע העתקה عمוקה (Deep Copy) ובכך לנתק את החדש מהמקור

```
51 // -----
52 // 4. Spread operator with objects
53 const person = {
54   name: 'John',
55   age: 25,
56   city: 'New York'
57 };
58
59 // Spread operator, creates a copy of the original, by creating a completely new instance
60 // with completely new properties and values
61 const person2 = { ...person };
62
63 person.name = 'Jane';
64
65 console.log(person);
66 console.log(person2);
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Filter (e.g. text, !exclude)

```
{name: 'Jane', age: 25, city: 'New York'}
  age: 25
  city: 'New York'
  name: 'Jane'
> [[Prototype]]: Object
{name: 'John', age: 25, city: 'New York'}
  age: 25
  city: 'New York'
  name: 'John'
> [[Prototype]]: Object
```

script

script

```
js-advanced > 19.11.23 > shallow_and_deep_copy > JS script.js > ...
1  // ...
2  // 1. Exercise
3  const cart1 = [4, 30, 76, 100];
4  const cart2 = [1, 600, 56, 60];
5
6  let carts = [cart1, cart2];
7  console.log(carts);
8
9  carts[1][2] = 99;
10 console.log(carts);
11 console.log(cart2);
12
13
14 // -----
15 // 2. Exercise using spread operator
16 carts = [...cart1, ...cart2];
17 console.log(carts);
18
19 carts[7] = 99
20 console.log(carts);
21 console.log(cart2);      You, 45 seconds ago • Uncommitted changes
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
> (2) [Array(4), Array(4)]
> (2) [Array(4), Array(4)]
> (4) [1, 600, 99, 60]
> (8) [4, 30, 76, 100, 1, 600, 99, 60]
> (8) [4, 30, 76, 100, 1, 600, 99, 99]
> (4) [1, 600, 99, 60]
```

# הכרות עם Shallow Copy & Deep Copy

בחלק זה נכיר אפשרויות להעתיק אובייקטים וונமוד על ההבדלים בין האפשרויות.  
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה הבדל בין עותק "חלול" לבין עותק "מדויק"?
- במה אופרטור `spread` יכול לעזור לנו בהקשר זה?
- מה היא כתובת זיכרון ומהו פוינטර?
- מה הבדל בין `Reference Type` ל-`Value Type` ?
- מה היתרונות של כל שיטה?

# תרגול Shallow copy & Deep copy

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

תרגיל	תיאור המשימה
<b>cart1</b>	[4,30,76,100]
<b>cart2</b>	[1,600,56,60]
<b>carts</b>	[[cart1],[cart2]]
	<p>צוו 2 מערכים: cart1 וcart2, תנו להם ערכים - 4 איברים המסמלים מחירי מוצרים.</p> <p>צוו מערך חדש בשם carts וכערך תנו לו את cart1 וcart2.</p> <p>הדפיסו את carts לkonsole.</p> <p>Ex-1</p>
	<p>שנו אחד מהאיברים של המערך השני שנמצא בתחום carts.</p> <p>אם אותו האיבר השתנה גם במערך ?cart2</p> <p>Ex-2</p>
	<p>אפסו את המערך carts וכערך תנו לו שני מערכים.</p> <p>פזרו בכל אחד מהמפעדים את האיברים של cart1 וcart2 בהתאם באמצעות שימוש בspread.</p> <p>הדפיסו את carts לkonsole.</p> <p>Ex-3</p>
	<p>שנו אחד מהאיברים של המערך השני שנמצא בתחום carts.</p> <p>הדפיסו לkonsole את carts ואת ?cart2.</p> <p>אם אותו האיבר השתנה גם במערך ?cart2</p> <p>Ex-4</p>
	<p>שנו את אחד המחירים במערך cart1.</p> <p>הדפיסו לkonsole את carts ואת cart1.</p> <p>אם אותו האיבר השתנה השתנה גם במערך הראשון של carts</p> <p>Ex-5</p>

# פונקציות חץ - arrow



# Arrow Function

create more concise, cleaner, and more readable functions by using the => operator

# Function

```
// This is a function declaration  
function myName(name) {  
    console.log(`Hello ${name}`);  
}
```

JavaScript

# Function declaration vs function expression

## Function declaration

```
// This is a function declaration  
  
function myName(name) {  
    console.log(`Hello ${name}`);  
}
```

JavaScript

Can refer to the function from anywhere in the code  
= **Hoisting**

## Function expression

```
// This is a function expression  
  
const myName = function(name) {  
    console.log(`Hello ${name}`);  
}
```

Function is only known within the relevant scope

```
// This is a simple function  
function myCar(car) {  
  console.log(`My car is a ${car}`);  
}
```



```
// This is an arrow function  
const myCar = (car) => {  
  console.log(`My car is a ${car}`);  
}
```

# בשורה אחת Arrow Function

- אין צורך לכתוב `return`, זה קורה אוטומטית

```
const myCar = (car) => `My car is a ${car}`;
```

\* אם יש רק Property אחד, לא צריך סוגרים ()

```
const myCar = car => `My car is a ${car}`;
```

# פונקציות חץ - Arrow

JS\_Advanced\_Arrow\_Function.html

ב חלק זה נלמד ליצור פונקציות באמצעות שיטה נוספת לשיטות שהכרנו קודם. בסיסים הנושא תוכלו לענות על השאלות הבאות:

```
function plus1(num1, num2) {  
    return num1 + num2;  
}  
  
let plus2 = (num1, num2) => num1 + num2;  
  
console.log( plus1(10,20) ); // 30  
console.log( plus2(10,20) ); // 30  
  
let sayHola = (name) => "Hola " + name;  
  
console.log( sayHola("Gal") ); // "Hola Gal"
```

- מה היא פונקציה חץ? (arrow function)
- איך ניתן להגדיר פונקציה חץ?
- מה היתרונות בשימוש בפונקציות חץ?
- מה ההבדל בין פונקציה חץ לפונקציה רגילה?

# תרגול פונקציית חץ – Arrow Function

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

## handleUserName

<b>firstName</b>	“gal”
<b>lastName</b>	“lavi”

## getUser

<b>firstName</b>	Prompt
<b>lastName</b>	Prompt
<b>quickHandleName</b>	(first, last) => {}

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקרוו לה <code>handleUserName</code> . פונקציה זו תדע לקבל 2 פרמטרים – <code>lastName</code> ו- <code>firstName</code> – ותחזיר סטרינג אחד הכלול את השם + שם המשפחה עם רווח ביניהם. נסו לשולח לפונקציה ערכים שונים והדפיסו את התוצאות לקונסול.
Ex-2	צרו פונקציה חדשה וקרוו לה <code>getUser</code> . הfonקציה תבקש מהמשתמש להזין שם פרטי, ולאחר מכן תבקש שם משפחה. הציגו <code>alert</code> למשתמש אשר מבורך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>handleUserName</code> .
Ex-3	בtron הfonקציה <code>getUser</code> , צרו משתנה חדש בשם <code>quickHandleName</code> וכעורך תנו לו פונקציית חץ אשר "תחקה" את אופן הפעולה של הפונקציה <code>handleUserName</code> .
Ex-4	הציגו <code>alert</code> למשתמש אשר מבורך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>quickHandleName</code> .



## סינום שיעור

## שיעור בית

- לעبور על כל התרגילים של הכיתה ולסיים אותם לפני שיעור הבא



# תודה על ההקשבה

אני וצוות המכללה כאן עבורכם



# Advanced Javascript

מרצה: תומר שגיא

שיעור 8

22/11/2023

# JavaScript Advanced

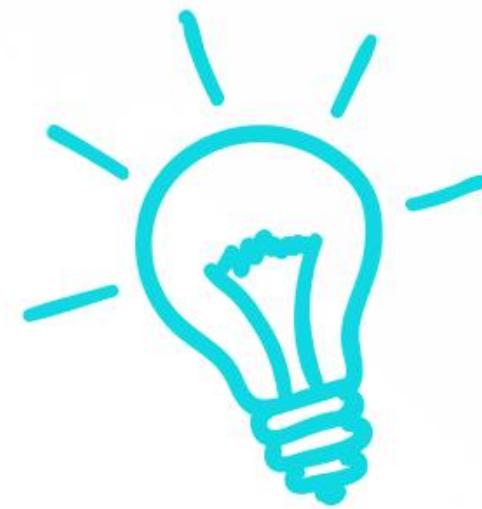
הכרות עם תכונות אסינכרוני ונוסחים מתקדמים

במסגרת הנושא נכיר ונעמיק ידע ב JavaScript ונלמד על אפשרויות מתקדמות ותכונות אסינכרוני הכולן קריאות לשרת.



- השלמות:
- Github Copilot
  - Template literals
  - By value vs by reference

- ✓ אופרטור Spread
- ✓ Shallow copy & Deep copy
- ✓ פונקציות חץ - arrow
- ✓ שימוש ב try & catch
- ✓ פונקציות Callback
- ✓ מבוא לתוכנות א-סינכרוני XMLHttpRequest
- ✓Promise
- ✓ מחלקה then - catch
- ✓ Async and Await
- ✓ Fetch
- ✓ Modules



**חזרה קצרה על מה שלמדנו  
ומענה לשאלות על משימת הבית**



# זמן שאלות

# השלמות

Github Copilot



# GitHub Copilot

v1.138.0

GitHub [github.com](#)

10,725,768

★★★★★ (958)

Your AI pair programmer

Disable

Uninstall

Switch to Pre-Release Version



This extension is enabled globally.

DETAILS

FEATURE CONTRIBUTIONS

CHANGELOG

EXTENSION PACK

RUNTIME STATUS



32k



Follow

@github



Views

14M

Try Copilot

Free trial

## Your AI pair programmer

Get Code Suggestions in real-time, right in your IDE

```
1
2  def common_prefix(a, b) :
3      """Return the common prefix of two lists."""
4      if len(a) < len(b) :
5          return common_prefix(b,a)
6      for i in range(len(a)) :
7          if a[i] != b[i] :
8              return a[:i]
9      return a
10
11 def test_common_prefix() :
12     assert common_prefix(['a','b','c'], ['a','b','c','d']) == ['a','b','c']
13     assert common_prefix(['a','b','c'], ['b','c','d']) == ['b','c']
14     assert common_prefix(['a','b','c'], ['b','c']) == ['b','c']
15     assert common_prefix(['a','b','c'], ['a','b']) == ['a','b']
16     assert common_prefix([], []) == []
```

### Categories

Programming Languages

Machine Learning

Education

Snippets

### Extension Resources

Marketplace

GitHub

### More Info

Published 2021-06-29, 17:26:17

Last released 2023-11-22, 03:39:43

Last updated 2023-11-22, 15:44:00

Identifier github.copilot

תוספַּת ל-VSC

- 

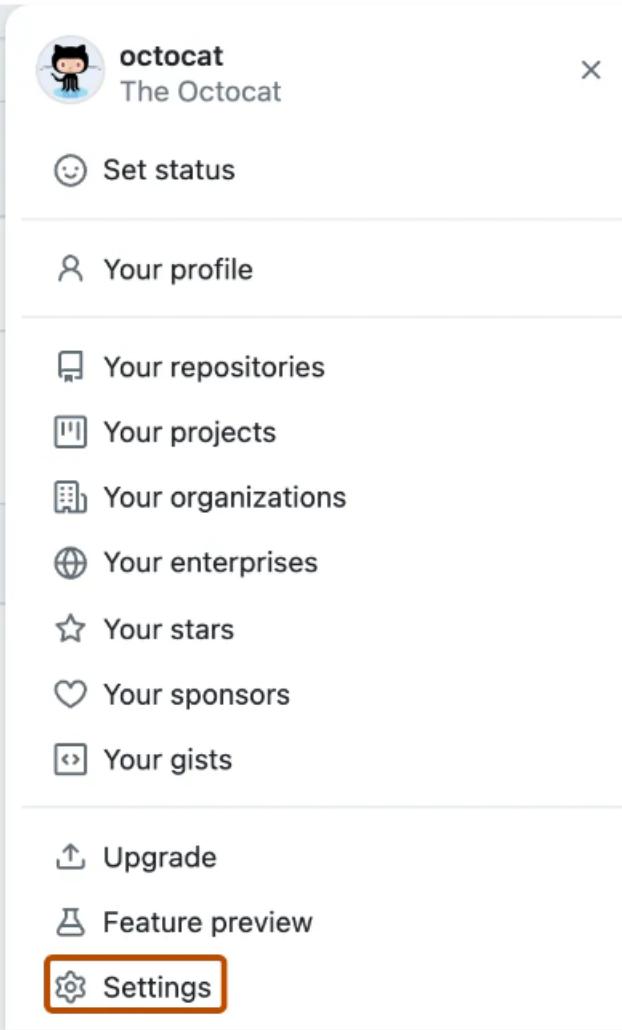
צריך להיות מנוי של Github  
להשתמש

- 

אפשר לנסות 30 יום  
חינם

-

- 1 In the upper-right corner of any page, click your profile photo, then click **Settings**.



- 2 In the "Code, planning, and automation" section of the sidebar, click **GitHub Copilot**.
- 3 On the GitHub Copilot settings page, click **Enable GitHub Copilot**.
- 4 Choose whether you want to pay monthly or yearly, and click **Continue to get access to Copilot**.
  - o If your personal account meets the criteria for a free GitHub Copilot subscription instead of a trial or paid subscription, you will automatically be taken to step 6.
- 5 Follow the steps to confirm your payment details, then click **Submit**.
- 6 Select your preferences, then click **Save and get started**.

You can change these preferences at a later time by returning to your GitHub Copilot settings.  
For more information, see "[Configuring GitHub Copilot in your environment](#)."

# Installing the GitHub Copilot extension for Visual Studio Code

To use GitHub Copilot, you must first install the Visual Studio Code extension.

- ➊ In the Visual Studio Code Marketplace, go to the [GitHub Copilot extension](#) page and click **Install**.
- ➋ A popup will appear, asking to open Visual Studio Code. Click **Open Visual Studio Code**.
- ➌ In the "Extension: GitHub Copilot" tab in Visual Studio Code, click **Install**.
- ➍ If you have not previously authorized Visual Studio Code in your GitHub account, you will be prompted to sign in to GitHub in Visual Studio Code.
  - If you have previously authorized Visual Studio Code in your GitHub account, GitHub Copilot will be automatically authorized.
- ➎ In your browser, GitHub will request the necessary permissions for GitHub Copilot. To approve these permissions, click **Authorize Visual Studio Code**.
- ➏ In Visual Studio Code, in the "Visual Studio Code" dialogue box, to confirm the authentication, click **Open**.

# השלמות

Template Literals

# Back-Tics

## Back-Tics Syntax

**Template Literals** use back-ticks (``) rather than the quotes (") to define a string:

```
let text = `Hello World!`;
```

# Quotes inside strings

## Quotes Inside Strings

With **template literals**, you can use both single and double quotes inside a string:

```
let text = `He's often called "Johnny";
```

# Multiline

## Multiline Strings

**Template literals** allows multiline strings:

```
let text =  
`The quick  
brown fox  
jumps over  
the lazy dog`;
```

# String interpolation

## Interpolation

**Template literals** provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

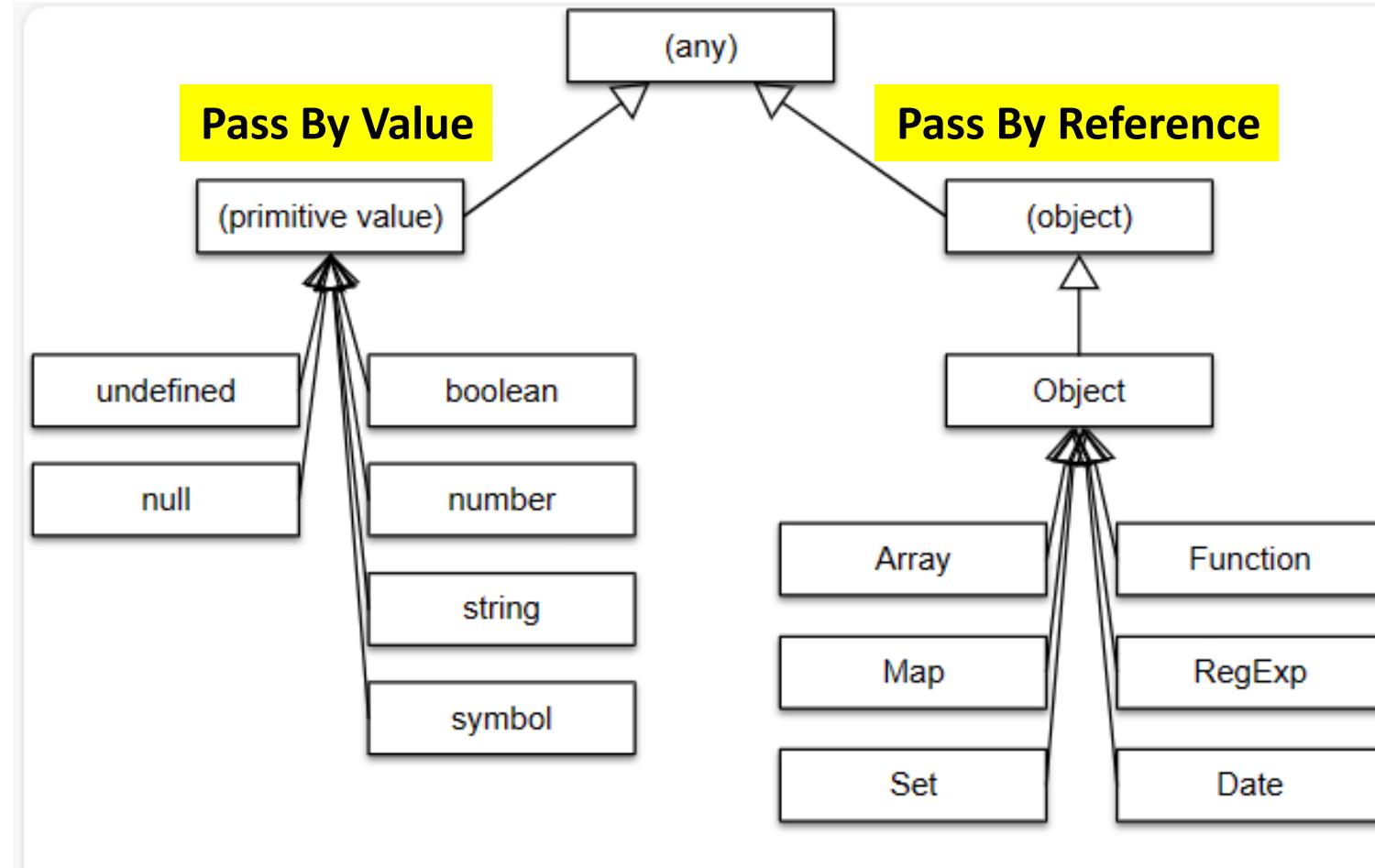
```
| ${...}
```

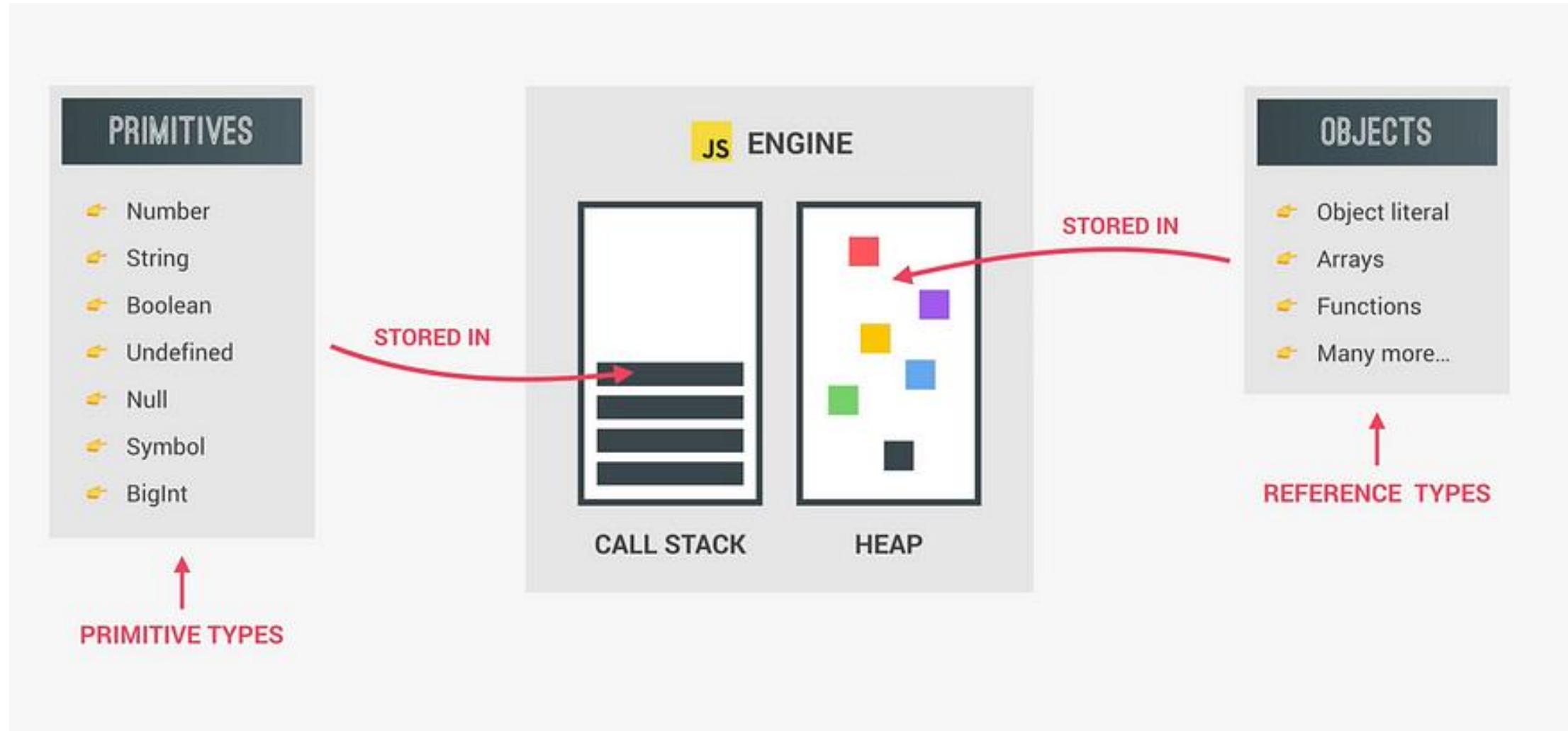
```
let firstName = "John";
let lastName = "Doe";

let text = `Welcome ${firstName}, ${lastName}!`;
```

# השלמות

Pass by value vs pass by reference





## 👉 Primitive values example:

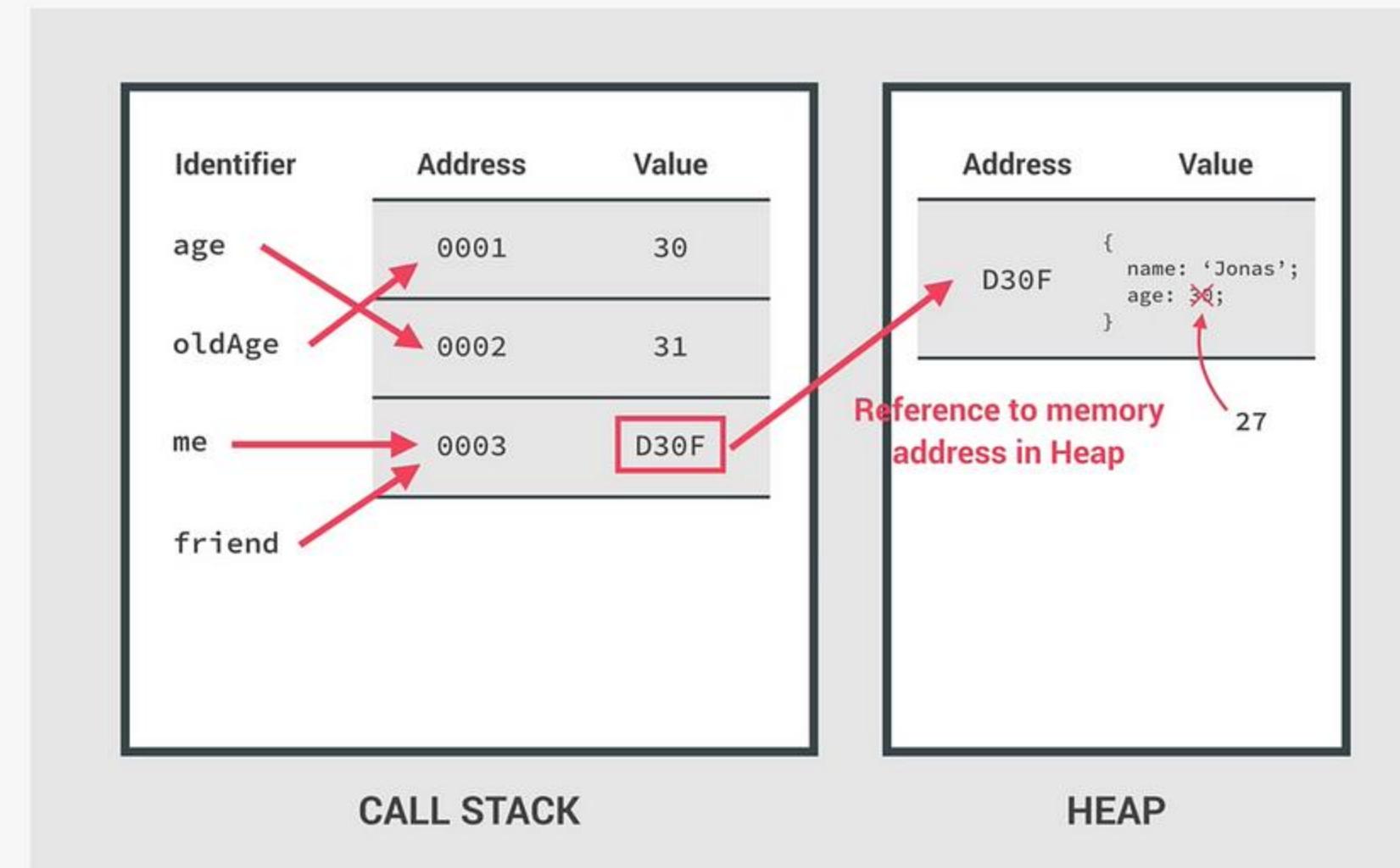
```
let age = 30;
let oldAge = age;
age = 31;
console.log(age); // 31
console.log(oldAge); // 30
```

## 👉 Reference values example:

```
const me = {
  name: 'Jonas'           No problem, because
  age: 30                 we're NOT changing the
};                         value at address 0003!
const friend = me;
friend.age = 27;

console.log('Friend:', friend);
// { name: 'Jonas', age: 27 }

console.log('Me:', me);
// { name: 'Jonas', age: 27 }
```



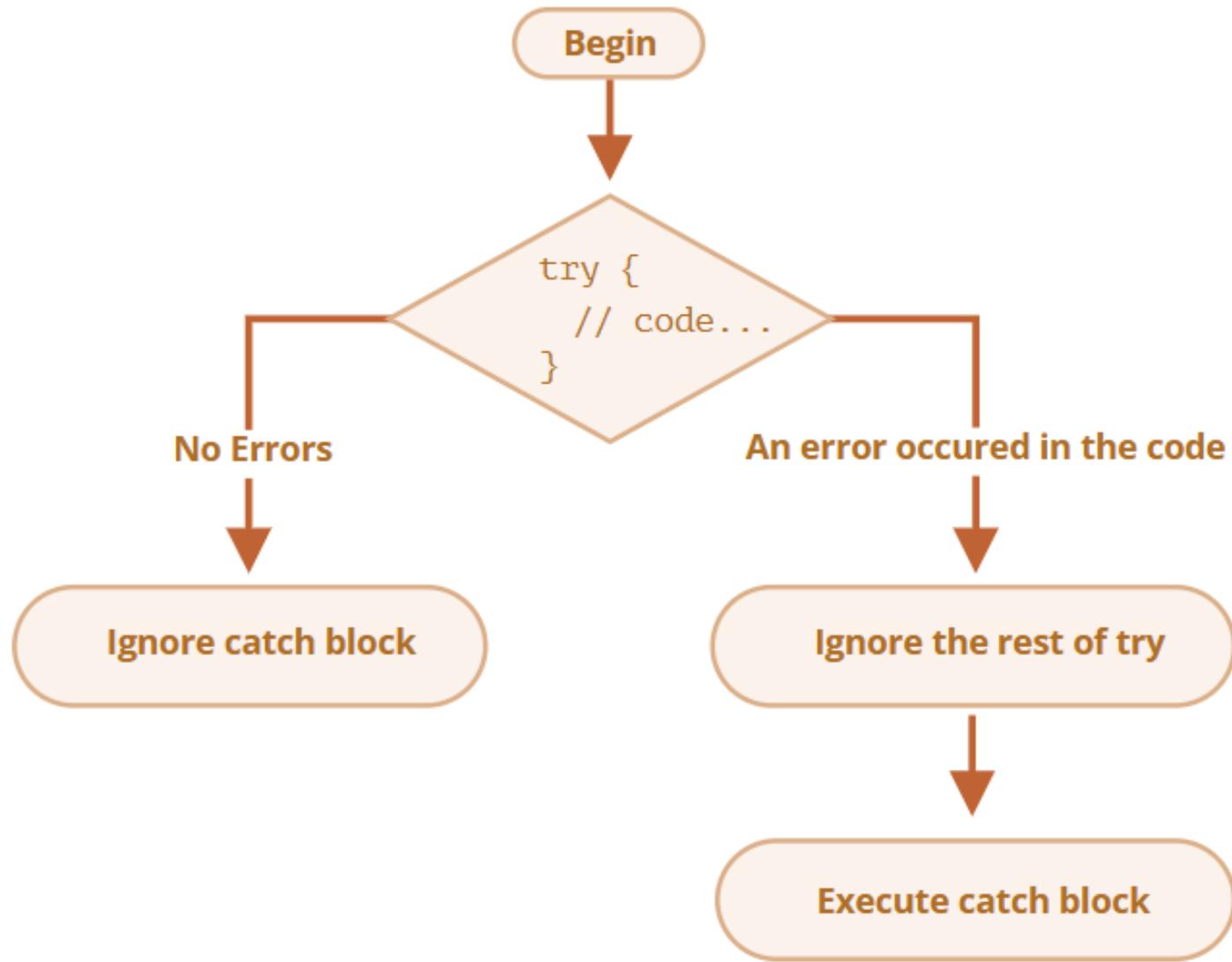
# שימוש ב **try & catch**



# try..catch

Allows us to “catch” errors so the script can, instead of dying, do something more reasonable.

אפשר לנו “لتפוס” שגיאות כך שהסקריפט יוכל, במקום למות, לעשות משהו הגיוני יותר.



## The “try...catch” syntax

The `try...catch` construct has two main blocks: `try`, and then `catch`:

```
1 try {  
2   // code...  
3  
4 } catch (err) {  
5   // error handling  
6  
7 }  
8  
9 }
```



```
1 try {
2
3   alert('Start of try runs'); // (1) <-
4
5   // ...no errors here
6
7   alert('End of try runs'); // (2) <-
8
9 } catch (err) {
10
11   alert('Catch is ignored, because there are no errors'); // (3)
12
13 }
```

```
1 try {  
2  
3   alert('Start of try runs'); // (1) <--  
4  
5   lalala; // error, variable is not defined!  
6  
7   alert('End of try (never reached)'); // (2)  
8  
9 } catch (err) {  
10  
11   alert(`Error has occurred!`); // (3) <--  
12  
13 }
```



## Error object

When an error occurs, JavaScript generates an object containing the details about it. The object is then passed as an argument to `catch`:

```
1 try {
2   // ...
3 } catch (err) { // <-- the "error object", could use another word instead of err
4   // ...
5 }
```



```
1 try {
2     lalala; // error, variable is not defined!
3 } catch (err) {
4     alert(err.name); // ReferenceError
5     alert(err.message); // lalala is not defined
6     alert(err.stack); // ReferenceError: lalala is not defined at (...call stack)
7
8     // Can also show an error as a whole
9     // The error is converted to string as "name: message"
10    alert(err); // ReferenceError: lalala is not defined
11 }
```



## try...catch...finally

Wait, that's not all.

The `try...catch` construct may have one more code clause: `finally`.

If it exists, it runs in all cases:

- after `try`, if there were no errors,
- after `catch`, if there were errors.

The extended syntax looks like this:

```
1 try {  
2     ... try to execute the code ...  
3 } catch (err) {  
4     ... handle errors ...  
5 } finally {  
6     ... execute always ...  
7 }
```

# שימוש ב try & catch

JS\_Advanced\_Try\_Catch.html

בחלק זה נלמד כיצד לטפל ולתפות בעיות בקוד ולטפל בהן, מה שיאפשר לך למשיך לroz.

בסיום הנושא תוכל לענות על השאלות הבאות:

```
const text = "Hola Class";

// demo 1
try{
  console.log(someText); ————— מנסה לבצע את הפעולה
}catch( error ){
  console.log(error); ————— קוד יירוץ במידה ותהייה שגיאה
  console.log(error.message);
}finally{
  console.log("Finish 1"); ————— קוד יירוץ בכל מקרה בסיום
}
```

```
// demo 2
console.log(someText); ————— דוגמה לקוד שיוציא שגיאה
console.log("Finish 2");
```

- מה יכולות להיגרם שגיאות ב-JS?
- אילו שגיאות יכולות להתרכש בזמן ריצת התוכנית שלא ניתן לצפות מראש?
- מה נוכל לעשות על מנת לטפל בשגיאות בזמן ריצה?
- מה הוא בлок ה-**try catch** ?
- מה נוכל לעשות במקרה שתפקידו שגיאה באמצעות בлок **?try-catch**
- מה תפקידו של **finally** ?

 ► Uncaught ReferenceError: someText is not defined  
at JS\_Advanced\_Try\_Catch.html:21:21

# תרגול שימוש ב-**try & catch**

צרו קובץ חדש בשם **Try\_Catch.js** לטובת הנושא ופתרו את התרגילים לפי הסדר  
חשובי להדפס הודעות הצלחה ושגיאה למשתמש וכמובן לבדוק כל תרגיל בסיום.

## divide

<b>num1</b>	50
<b>num2</b>	5

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקרוו לה <b>divide</b> . פונקציה זו תדע לקבל 2 פרמטרים – <b>num1</b> - <b>num2</b> .
Ex-2	הפונקציה <b>divide</b> תיקח את המספרים ותבצע פעלות חילוק מתמטית ביניהם ותחזיר את התוצאה. נסו את הפונקציה באמצעות שליחת ערכים שונים בכל פעם והדפסת התוצאה החזרה.
Ex-3	נסו לשלוח אל הפונקציה משתנים אשר טרם נוצרו.
Ex-4	על מנת למנוע את השגיאה המתעוררת משליחת הערכים האחרונות שביצעתם, השתמשו ב- <b>try</b> ו- <b>catch</b> כדי לתפוס את השגיאה ולהדפיס לკונסול הודעה תואמת.
Ex-5	צרו משתנה חדש בשם <b>finalMessage</b> . השתמשו בו על מנת להדפיס הודעה לკונסול שהטיפול הסתיים, השתמשו במשתנה <b>finalMessage</b> על מנת להציג הודעה תואמת, בסוף ה- <b>finally</b> אפסו את הערך של המשתנה <b>finalMessage</b> .



## סינום שיעור

## שיעור בית

- לעبور על כל התרגילים של הכיתה ולסיים אותם לפני שיעור הבא



# תודה על ההקשבה

אני וצוות המכללה כאן עבוריכם



# Advanced Javascript

מרצה: תומר שגיא

שיעור 9

26/11/2023

Callback  
מבוא לתוכנות אסינכרוני

# JavaScript Advanced

הכרות עם תכונות אסינכרוני ונוסחים מתקדמים

במסגרת הנושא נכיר ונעמיק ידע ב JavaScript ונלמד על אפשרויות מתקדמות ותכונות אסינכרוני הכולן קריאות לשרת.



שיעור 9

אופרטור Spread ✓

Shallow copy & Deep copy ✓

פונקציות חצ - ✓

try & catch ✓

פונקציות Callback ✓

מבוא לתוכנות א-סינכרוני ✓

קריאות XMLHttpRequest ✓

מחלקה Promise ✓

שימוש ב then - catch ✓

Async and Await ✓

שיעור 10

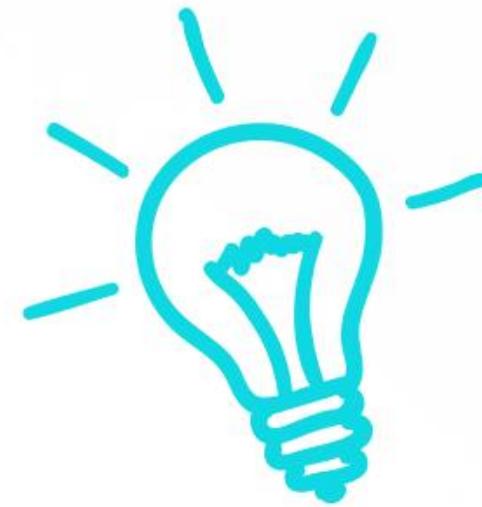
שיעור 11

שיעור 12

Fetch ✓

Modules ✓

2 שיעורים ספייר



**חזרה קצרה על מה שלמדנו  
ומענה לשאלות על משימת הבית**

# תרגילים משיעור קודם

Arrow Functions

# תרגול פונקציית חץ – Arrow Function

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

## handleUserName

<b>firstName</b>	“gal”
<b>lastName</b>	“lavi”

## getUser

<b>firstName</b>	Prompt
<b>lastName</b>	Prompt
<b>quickHandleName</b>	(first, last) => {}

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקרוו לה <code>handleUserName</code> . פונקציה זו תדע לקבל 2 פרמטרים – <code>lastName</code> ו- <code>firstName</code> – ותחזיר סטרינג אחד הכלול את השם + שם המשפחה עם רווח ביניהם. נסו לשולח לפונקציה ערכים שונים והדפיסו את התוצאות לקונסול.
Ex-2	צרו פונקציה חדשה וקרוו לה <code>getUser</code> . הfonקציה תבקש מהמשתמש להזין שם פרטי, ולאחר מכן תבקש שם משפחה. הציגו <code>alert</code> למשתמש אשר מבורך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>handleUserName</code> .
Ex-3	בtron הfonקציה <code>getUser</code> , צרו משתנה חדש בשם <code>quickHandleName</code> וכעורך תנו לו פונקציית חץ אשר "תחקה" את אופן הפעולה של הפונקציה <code>handleUserName</code> .
Ex-4	הציגו <code>alert</code> למשתמש אשר מבורך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>quickHandleName</code> .

# תרגול שימוש ב-**try & catch**

צרו קובץ חדש בשם **Try\_Catch.js** לטובת הנושא ופתרו את התרגילים לפי הסדר  
חשובי להדפס הודעות הצלחה ושגיאה למשתמש וכמובן לבדוק כל תרגיל בסיום.

## divide

<b>num1</b>	50
<b>num2</b>	5

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקרוו לה <b>divide</b> . פונקציה זו תדע לקבל 2 פרמטרים – <b>num1</b> - <b>num2</b> .
Ex-2	הפונקציה <b>divide</b> תיקח את המספרים ותבצע פעלות חילוק מתמטית ביניהם ותחזיר את התוצאה. נסו את הפונקציה באמצעות שליחת ערכים שונים בכל פעם והדפסת התוצאה החזרה.
Ex-3	נסו לשלוח אל הפונקציה משתנים אשר טרם נוצרו.
Ex-4	על מנת למנוע את השגיאה המתעוררת משליחת הערכים האחרונות שביצעתם, השתמשו ב- <b>try</b> ו- <b>catch</b> כדי לתפוס את השגיאה ולהדפיס לკונסול הודעה תואמת.
Ex-5	צרו משתנה חדש בשם <b>finalMessage</b> . השתמשו בו על מנת להדפיס הודעה לკונסול שהטיפול הסתיים, השתמשו במשתנה <b>finalMessage</b> על מנת להציג הודעה תואמת, בסוף ה- <b>finally</b> אפסו את הערך של המשתנה <b>finalMessage</b> .

# תרגילים נוספים

Arrow Functions

שאלה:

מה היתרון העיקרי שבסימוש בפונקציות-חץ ב JavaScript-בשיטת לביטוי פונקציה רגילים?

שאלה:

המיר את הפונקציה הרגילה הבאה לפונקציה-חץ:

```
function multiply(a, b) {  
    return a * b;  
}
```

שאלה:

כתב מחדש את הפונקציה הבאה באמצעות פונקציה-חץ:

```
const greet = function(name) {  
    return `שלום, ${name}!`;  
};
```

שאלה:

כתב מחדש את הפונקציה הבאה באמצעות פונקציה-חץ:

javascript

```
const greet = function(name) {  
    return `שלום, ${name}!`;  
};
```

שאלה:

המיר את הקוד הבא כך שיישמש בפונקציה-חץ וגוף קצר:

```
const add = (a, b) => {  
    return a + b;  
};
```

שאלה:

מה התחביר של פונקציה-חץ שאין לה פרמטרים?

## שאלות 2 – פונקציות חץ

- שאלה 1:  
כתב פונקציה בעזרת פונקציית חץ שמקבלת שני פרמטרים ומחזירה את סכום השניים.
- שאלה 2:  
צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של מספרים ומחזירה מערך חדש עם הריבוע של כל מספר.
- שאלה 3:  
כתב פונקציה באמצעות פונקציית חץ שבודקת אם מספר נתון הוא זוגי ומהזירה `true` אם כן ו-`false` אם הוא אי-זוגי.
- שאלה 4:  
צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של מחרוזות ומחזירה מערך חדש עם אורךי המחרוזות.
- שאלה 5:  
כתב פונקציה באמצעות פונקציית חץ שמחשבת את הגורמים של מספר תמיד חיובי נתון.
- שאלה 6:  
צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של מספרים ומחזירה את סכום כל המספרים החוביים במערך.
- שאלה 7:  
כתבו פונקציה באמצעות פונקציית חץ שמקבלת מחרוזת כקלט ומחזירה את המחרוזת הפונה.
- שאלה 8:  
צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של שמות ומחזירה מערך חדש עם השמות שמתחילה באות "א".
- שאלה 9:  
כתבו פונקציה באמצעות פונקציית חץ שמחשבת את שטח המלבן על פי אורכו ורוחבו שני נתונים כפרמטרים.
- שאלה 10:  
צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של מספרים ומחזירה את המספר הגדול ביותר במערך.

# שאלות השלם את החסר 1

תרגול 1:

המירו את הפונקציה הבאה לפונקציית חץ:

```
function add(a, b) {  
    return a + b;  
}  
  
const sum = _____ => a + b;
```

תרגול 2:

כתבו פונקציית חץ שמקבלת מערך של מספרים ומחזירה את הריבוע של כל מספר כמערך חדש:

```
const numbers = [1, 2, 3, 4, 5];  
const squares = _____ => {  
    return _____ .map(_____) => _____ * _____);  
};
```

תרגול 3:

המירו את הפונקציה הבאה לפונקציית חץ:

```
function greet(name) {  
    return `שלום, ${name}!`;  
}  
  
const sayHello = _____ => `!_____ שלוּם,
```

# שאלות השלם את החסר 2

תרגול 4:

כתבו פונקציה חץ שבודקת אם מספר נתון הוא זוגי ומחזירה `true` אם כן ו-`false` אם הוא אי-זוגני:

```
const isEven = _____ => {
  if (_____ % 2 === 0) {
    return _____;
  } else {
    return _____;
  }
};
```

תרגול 5:

המירו את הפונקציה הבאה לפונקציה חץ:

```
function multiply(a, b, c) {
  return a * b * c;
}

const product = _____ => a * b * c;
```

תרגול 6:

כתבו פונקציה חץ שמקבלת מערך של מחרוזות ומחזירה מערך חדש עם אורכי המחרוזות:

```
const words = ["בננה", "דובדבן"]
const lengths = _____ => {
  return _____ .map(_____.length);
};
```



# זמן שאלות

# פונקציות Callback



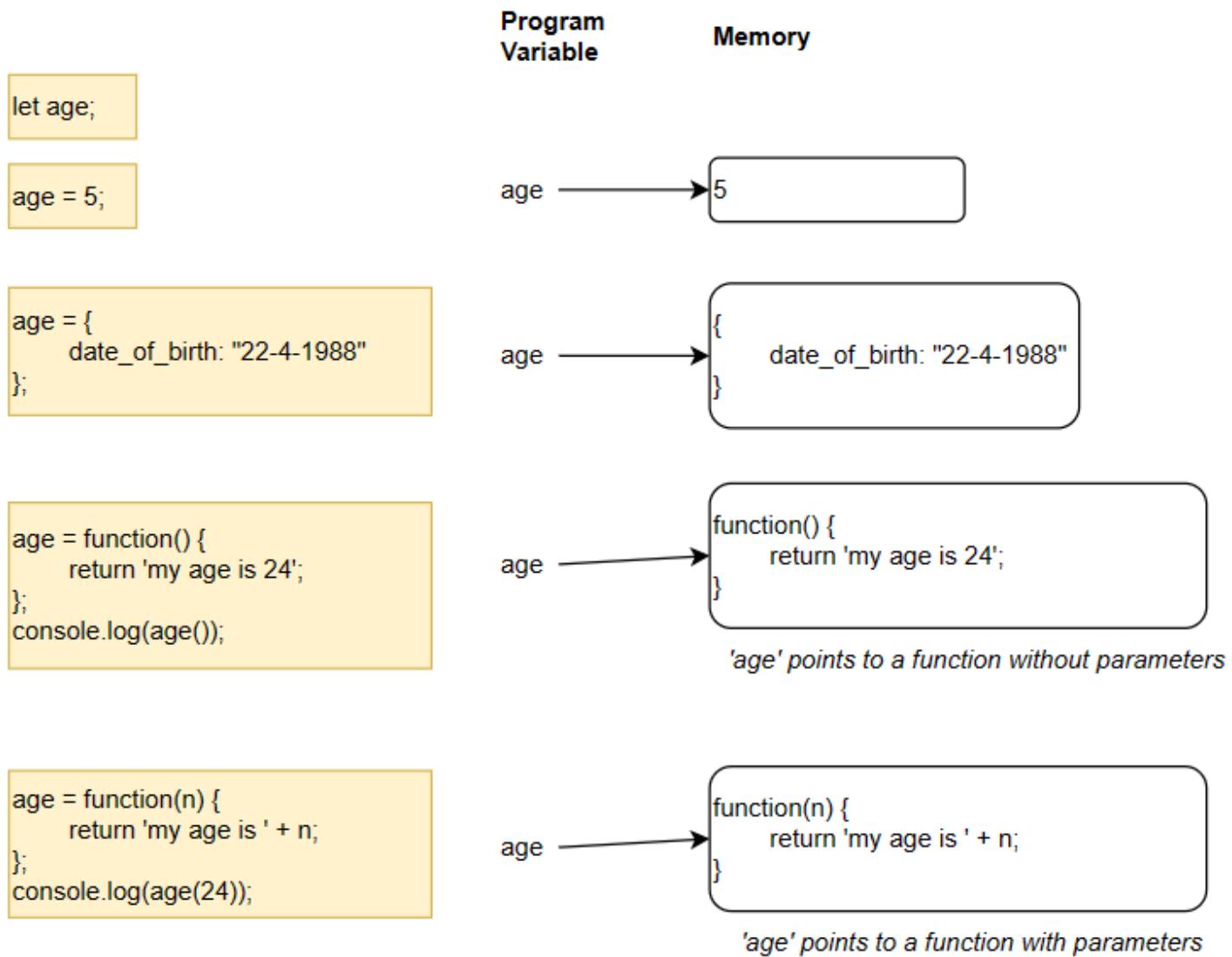
## 2.a. Callback functions - without parameters

*"I will call back later!"*

A callback is a function passed as an argument to another function

This technique allows a function to call another function

A callback function can run after another function has finished



```
let age;
```

```
age = 5;
```

```
age = {  
    date_of_birth: "22-4-1988"  
};
```

### Program Variable

### Memory



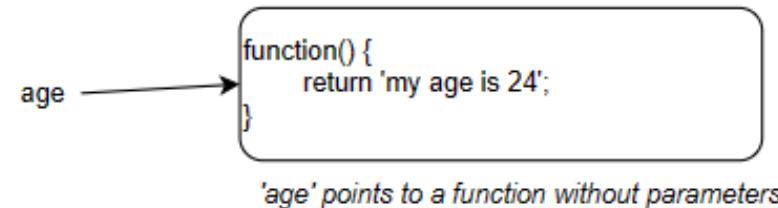
```
function print(a_variable) {  
    console.log(a_variable);  
}
```

```
print(age);
```

```
print(age);
```

5{date\_of\_birth: "22-4-1998"}

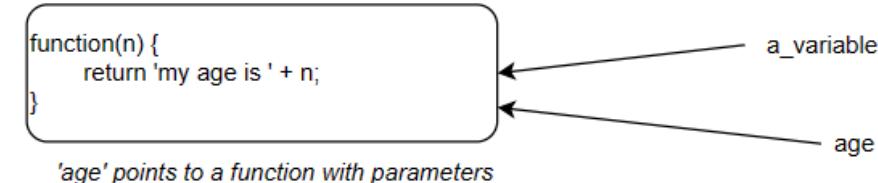
```
age = function() {
    return 'my age is 24';
};
console.log(age());
```



```
function print(a_variable) {
    console.log(a_variable());
}

print(age);
```

my age is 24



```
function print(a_variable) {
    console.log(a_variable());
}

print(age);
```

1. `print(age);` // → `age` is a pointer to a function call. Not called yet
2. `print(a_variable) {` // → `'a_variable'` is a local variable pointing to the `'age'` function passed to it
3. `console.log(a_variable());` // → invokes the `'a_variable()'` function (which is in fact the original `'age'` pointer).
4. `a_variable();` // → `a_variable();` calls the function block in memory
5. `return 'my age is 24';`
6. `console.log('my age is 24');`

`console.log(a_variable());`

We invoke the function as a regular function!

## 2.b. Callback functions - with parameters

2 common ways to call a function

```
function myDisplayer(some) {  
    document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2) {  
    let sum = num1 + num2;  
    return sum;  
}  
  
let result = myCalculator(5, 5);  
myDisplayer(result);
```



```
function myDisplayer(some) {  
    document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2) {  
    let sum = num1 + num2;  
    myDisplayer(sum);  
}  
  
myCalculator(5, 5);
```

# Callback function allows flexibility

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  myDisplayer(sum);
}

myCalculator(5, 5);
```

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

**2)** Calling the ‘**myCallback()**’ parameter will actually call the ‘**myDisplayer()**’ function

**3)** A parameter called ‘**myCallback**’ (could be any name!), and it points to the ‘**myDisplayer()**’ function in memory. We can now use ‘**myCallback()**’ as a function.

**1)** Send the function’s reference / pointer in memory

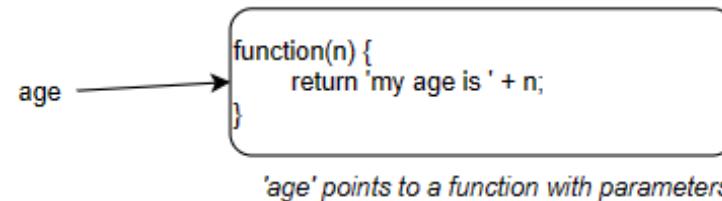
# Parameters in source function and callback function

```
function myDisplayer(some) {  
    document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2, myCallback) {  
    let sum = num1 + num2;  
    myCallback(sum);  
}  
  
myCalculator(5, 5, myDisplayer);
```

Note that the number of parameters in the function **must be used exactly** the same in the referenced parameter in other places, but the variable names are regular variables and can be **any name**.

myCallback(sum) →  
myDisplayer(some)

```
age = function(n) {
    return 'my age is ' + n;
};
console.log(age(24));
```



```
function print_with_value(a_variable, value) {
    console.log(a_variable(value));
}
```

```
print_with_value(age, 24);
```

my age is 24

1. `print_with_value(age, 24);` // → age is a pointer to a function call. Not called yet
2. `print_with_value(a_variable, value) {` // → 'a\_variable' is a local variable pointing to the 'age' function passed to it, 'value' is a local variable which will hold a regular value.
3. `console.log(a_variable(value));` // → invokes the 'a\_variable()' function (which is in fact the original 'age' pointer), and sends the value received as the 2<sup>nd</sup> parameter in the 'print\_with\_value' function
4. `a_variable(value);` // → `a_variable(value);` calls the function block in memory and sends the received value, i.e. `age(24)`;
5. `return 'my age is 24';`
6. `console.log('my age is 24');`

# Parameters in source function and callback function

```
function myDisplayer(some) {  
    document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2, myCallback) {  
    let sum = num1 + num2;  
    myCallback(sum);  
}  
  
myCalculator(5, 5, myDisplayer);
```

Remember, instead of sending the function like we learnt, we can also:

1. Create an anonymous function
2. Create an arrow function

# Implementation using anonymous and arrow functions

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Functions</h1>
<h2>Callback Functions</h2>

<p>The result of the calculation is:</p>
<p id="demo"></p>

<script>
function myDisplayer(something) {
  document.getElementById("demo").innerHTML = something;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

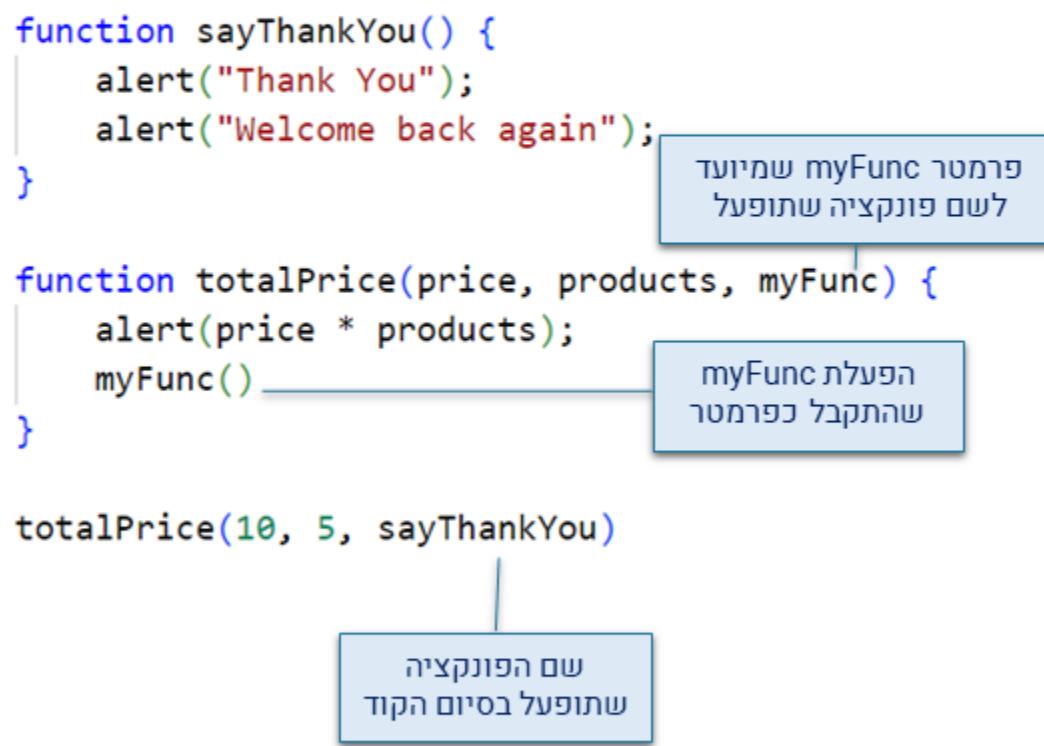
// Arrow function
myCalculator(5, 5, (something) => {
  document.getElementById("demo").innerHTML = something;
});
</script>

</body>
</html>
```

# פונקציות Callback

JS\_Advanced\_Callback\_Functions.html

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



- מה היא פונקציית `callback`?
- לאייה שימושים נועדה?
- באיזה מקרים נעזר בפונקציית `callback`?

# פונקציות Callback

ב חלק זה נלמד מגוון נושאים.  
בסיסם הנושא תוכלו לענות על השאלות הבאות:

```
// Demo 1
document.getElementById('myButton1').addEventListener("click", function () {
| alert('Hola Class From myButton')
});
document.getElementById('myButton2').addEventListener("click", function () {
| alert('Hola Class From myButton')
});
```

פונקציית callback אונומית  
שתוFunל בעת הפעלת האירוע

```
// Demo 2
function sayHola(){
| alert('Hola Class From myButton')
}
document.getElementById('myButton3').addEventListener("click", sayHola );
document.getElementById('myButton4').addEventListener("click", sayHola );
```

פונקציית callback שתוFunל  
בעת הפעלת האירוע

```
<button id="myButton1">click me 1</button>
<button id="myButton2">click me 2</button>
<button id="myButton3">click me 3</button>
<button id="myButton4">click me 4</button>
```

# תרגילים

Callback Functions

# Exercise

- Create 2 callback functions (functions to call them later)
  - One prints your name using the 'alert()' function
  - One prints your name using 'document.write()' function
- Create a 3rd function which will:
  - Have as an input parameter, a callback function (just a variable that points to a function)
  - Display the message “Displaying your name” using the 'alert()' function
  - Call the callback function
- Call the 3rd function and pass the name of the first function (the first callback function)
- Call the 3rd function and pass the name of the second function (the second callback function)
- You should see your name displayed first in an 'alert()' box, followed by your name appearing on the page

- צרו 2 פונקציות קולבק (פונקציה שניית לקרוא להן מאוחר יותר):
  - אחת מדפיסה את השם שלו באמצעות הפונקציה "alert"
  - ואחת מדפיסה את השם שלו באמצעות הפונקציה "document.write"
- צרו פונקציה 3 שתכלול:
  - כפרמטר קלט, פונקציה קולבק (פשוט מעתנה שמצויב לפונקציה)
  - תציג את הודעה "מציג את השם שלך" באמצעות הפונקציה "alert"
- **תפעיל את פונקציית הקולבך**
- קראו לפונקציה השלישית וMOVE her אליה שם הפונקציה הראשונה (הפונקציה הראשונה של הקולבך)
- קראו לפונקציה השלישית וMOVE her אליה שם הפונקציה השנייה (הפונקציה השנייה של הקולבך)
- אתם אמורים לראות את שמותם מוצג תחילה בתיבת ',(' alert() ואז שם השם יופיע על העמוד.

# השלם את החסור 1

תרגול 1:  
צרו פונקציית קולבק שמכפילה שני מספרים ומחזירה את התוצאה. לאחר מכן, צרו פונקציה בשם 'calculate' שמקבלת את הפונקציה הזו כารוגמן ומשתמשת בה כדי ליחס את התוצאה של כפל 5\*7.

```
const multiply = (a, b) => {
  return _____;
};

const calculate = (callback) => {
  const result = callback(5, 7);
  console.log(`ההתוצאה היא: ${result}`);
};

calculate(_____);
```

תרגול 2:  
צרו פונקציית קולבק שבודקת אם מספר נתון הוא זוגי ומחזירה `true` אם כן ו`false` אם הוא אי-זוגי. לאחר מכן, צרו פונקציה בשם 'checkNumber' שמקבלת את הפונקציה הזו כארוגמן ומשתמשת בה כדי לבדוק אם המספר 8 הוא מספר זוגי.

```
const isEven = (number) => {
  return _____;
};

const checkNumber = (callback) => {
  if (callback(8)) {
    console.log("המספר הוא מספר זוגי.");
  } else {
    console.log("המספר הוא מספר אי-זוגי.");
  }
};

checkNumber(_____);
```

# השלם את החסור 2

תרגול 3:

צרו פונקציית קולבק שמקבלת מערך של מספרים ומחזירה את סכום כל המספרים במערך. לאחר מכן, צרנו פונקציה בשם 'calculateSum' שמקבלת את הפונקציה הזו כารגומנט ומשתמשת בה כדי לחשב את סכום המספרים במערך [3, 12, 7, 5].

```
const sumArray = (numbers) => {
  return _____;
};

const calculateSum = (callback) => {
  const numbers = [3, 7, 12, 5];
  const result = callback(numbers);
  console.log(`סכום הוא: ${result}`);
};

calculateSum(_____);
```

תרגול 4:

צרו פונקציית קולבק שמקבלת מחרוזת ומחזירה את אורך המחרוזת. לאחר מכן, צרנו פונקציה בשם 'getStringLength' שמקבלת את הפונקציה הזו כארגומנט ומשתמשת בה כדי למצוא את אורך המחרוזת "שלום, עולם!".

```
const getLength = (string) => {
  return _____;
};

const getStringLength = (callback) => {
  const text = "שלום, עולם!";
  const length = callback(text);
  console.log(`האורך הוא: ${length}`);
};

getStringLength(_____);
```

# תרגול פונקציות Callback

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

## Divide, Multi, Add, Subtract

<b>num1</b>	50
<b>num2</b>	5

## calc

<b>num1</b>	50
<b>num2</b>	5
<b>calcFunc</b>	calcFunc(num1,num2)

## go

<b>num1</b>	Prompt
<b>num2</b>	Prompt
<b>operator</b>	Prompt

תרגיל	תיאור המשימה
Ex-1	<p>צור פונקציה חדשה וקרוואו לה <code>divide</code>. פונקציה זו תדע לקבל 2 פרמטרים – <code>num1</code>-<code>num2</code> ותדפיס בקונסולה את תוצאה החילוק של המספרים.</p>
Ex-2	<p>צור פונקציה נוספת עבור כל פעולה מתמטית בהתאם לסעיף הקודם.</p>
Ex-3	<p>צור פונקציה חדשה בשם <code>calc</code>. פונקציה זו תקבל כפרמטרים 2 מספרים ופונקציית <code>callback</code> בשם <code>calcFunc</code>. הפונקציה תשמש בפונקציית <code>the-back</code> <code>callback</code> שנשלחה אליה כפרמטר על מנת לבצע את החישוב על המספרים.</p>
Ex-4	<p>צור פונקציה חדשה בשם <code>go</code>. הfonקציה תבקש מהמשתמש 2 מספרים ואופרטור ותשמר את הערכים ב-3 משתנים שונים. באמצעות שימוש ב-<code>if</code> נוכל לדעת מה הפונקציה הנכונה לביצוע הפעולה המתמטית הרצויה. נשתמש בפונקציה <code>calc</code> על מנת לחשב, נשלח אליה את המספרים ואת פונקציית <code>callback</code> המתאימה.</p>
Ex-5	<p>אתגר: השתמשו בפונקציות <code>chz</code> אונומיות במקום להשתמש בפונקציות החישוב שכתבו בהתחלה.</p>



# Advanced Javascript

מרצה: תומר שגיא

שיעור 10  
26/11/2023

מבוא לתוכנות אסינכורוני  
XMLHttpRequest



# JavaScript Advanced

הכרות עם תכונות אסינכרוני וновאיים מתקדמים

במסגרת הנושא נכיר וنعمיק ידע ב JavaScript ונלמד על אפשרויות מתקדמות ותכונות אסינכרוני הכולן קריאות לשרת.

שיעור 7

אופרטור Spread ✓  
Shallow copy & Deep copy ✓

שיעור 8

פונקציות חצ - ✓  
try & catch ✓

שיעור 9

פונקציות Callback ✓

שיעור 10

מבוא לתוכנות א-סינכרוני ✓  
XMLHttpRequest ✓

שיעור 11

מחלקה Promise ✓  
שימוש ב then - catch ✓

שיעור 12

Async and Await ✓  
Fetch ✓

שיעור 13

Modules ✓

1 שיעורים ספייר



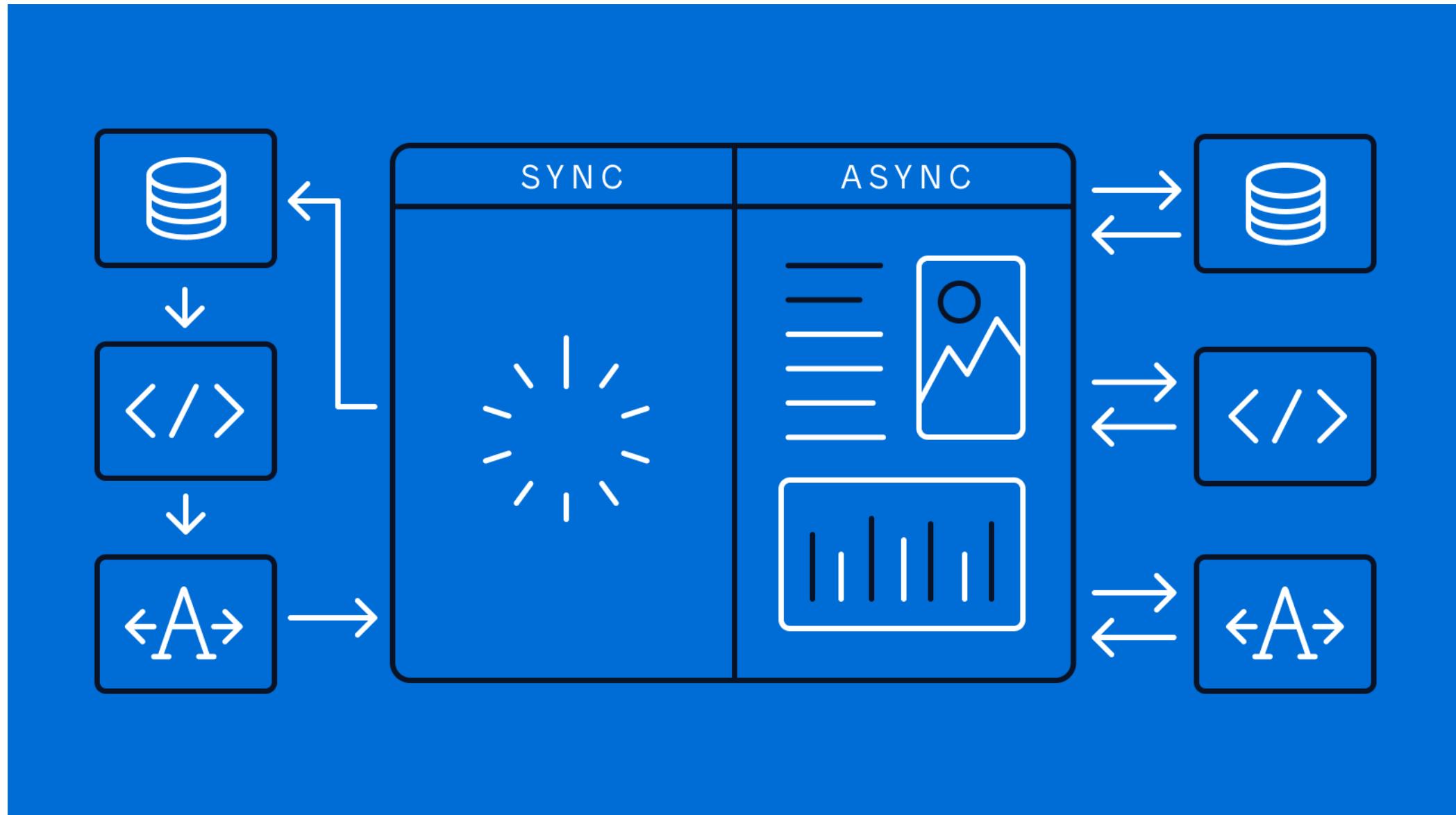
**חזרה קצרה על מה שלמדנו  
ומענה לשאלות על משימת הבית**

# תרגילים משיעור קודם

Arrow Functions

# מבוא לתוכנות א-סנכרוני





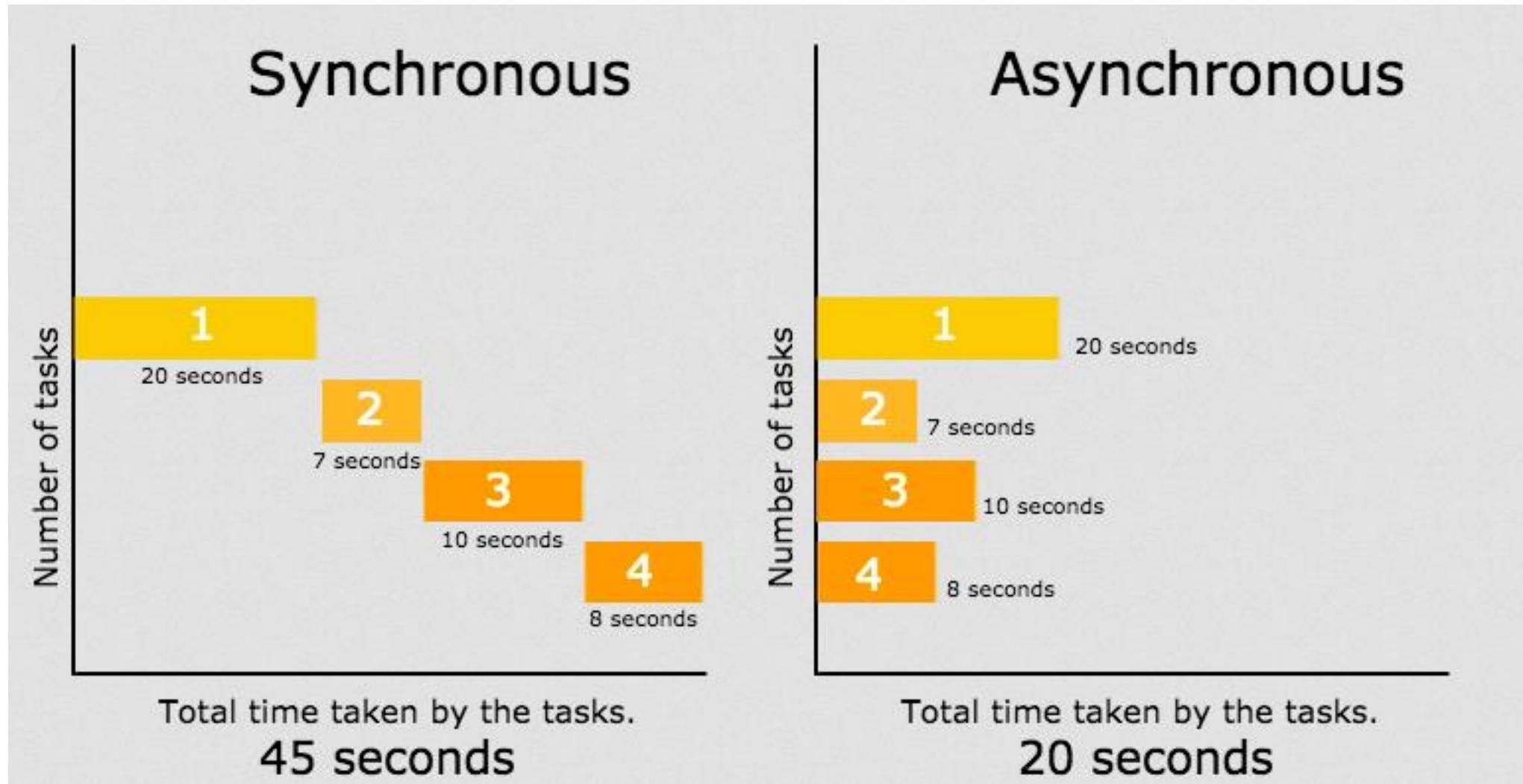
# Synchronous

vs.

# Asynchronous



@tapasadhikary



# setTimeout & setInterval

[https://www.w3schools.com/js/js\\_asynchronous.asp](https://www.w3schools.com/js/js_asynchronous.asp)

- ‘**setTimeout(function, time in milliseconds)**’
- ‘**setInterval(function, time in milliseconds)**’
- Using a ‘callback’ function after certain amounts of time
- E.g. when needed

```
setTimeout(myFunction, 3000);

function myFunction() {
    document.getElementById("demo").innerHTML = "I love You !!";
}
```

# setTimeout & setInterval - Exercise

- Create a function that:
  - Asks for your name (using prompt)
  - Creates a new 'div' element
  - Sets its innerHTML to the name provided from the input
  - Adds the 'div' to the body of the document
- Create a timer that calls the above function after 2 seconds
- After completing the above:
  - Replace the function with an anonymous function
  - And an arrow function

# Stopping a timer

- Used normally when using ‘setInterval()’
- clearInterval()

```
const myInterval = setInterval(myGreeting, 3000);

function myGreeting() {

    document.getElementById("demo").innerHTML = "Happy Birthday to You !!"}

function myStopFunction() {

    clearInterval(myInterval);

}
```

# Exercise

- Create a clock in HTML
- Use ‘setInterval()’ to update a <div> using ‘innerHTML’ every second
- Hints:
  - ‘let d = new Date();’ creates a ‘date’ object
  - You can then use the following methods to query different parts of the time
  - d.getHours() - Gives you hours
  - d.getMinutes() - Gives you minutes
  - d.getSeconds() - Gives you seconds

# מבוא לתוכנות א-סנכרוני

JS\_Advanced\_Asyncronous1.html

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

```
function connectToServer() {  
    setTimeout(function () { console.log("response from server"); }, 3000);  
}  
  
console.log("do something 1..");  
console.log("do something 2..");  
connectToServer(); ——————  
console.log("do something 3..");  
console.log("do something 4..");
```

- מהו תוכנות סנכרוני?
- מהו תוכנות א-סנכרוני?
- איזה בעיות יכולות להיווצר בקוד?

קריאה לשרת עשויה לקחת  
זמן, אך הקוד ממשיך לroz

do something 1..
do something 2..
do something 3..
do something 4..
response from server

# מבוא לתוכנות א-סנכרוני

JS\_Advanced\_Asyncronous2.html

ב חלק זה נלמד מגוון נושאים.  
בסיום הנושא תוכלן לענות על השאלות הבאות:

```
function fn1() {  
    console.log(1);  
}
```

```
function fn2() {  
    console.log(2);  
}
```

```
function fn3() {  
    console.log(3);  
}
```

```
function fn4() {  
    console.log(4);  
}
```

```
setTimeout(fn1, 500);  
fn2();  
fn3();  
fn4();
```

הפעולה עשויה לדרוש זמן  
אך הקוד ממשין להרוץ

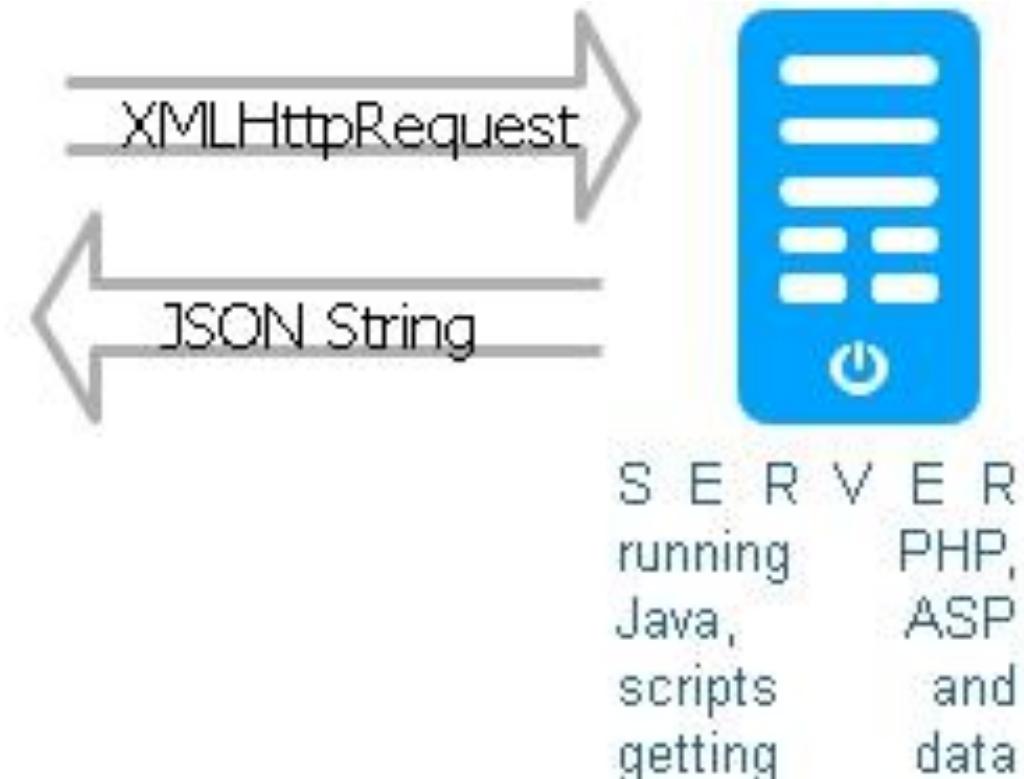
2
3
4
1

# קריאה XMLHttpRequest



# תקשורת לקוח ושרת

Client & server communications



# Basics

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

# Basics

Open connection method

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

Object

REST API Method

URL

Send request method

# REST API

אתר עם דатаה לדוגמא בcheinm



# REST API Methods



## GET

Receive information  
about an API resource



## POST

Create an  
API resource



## PUT

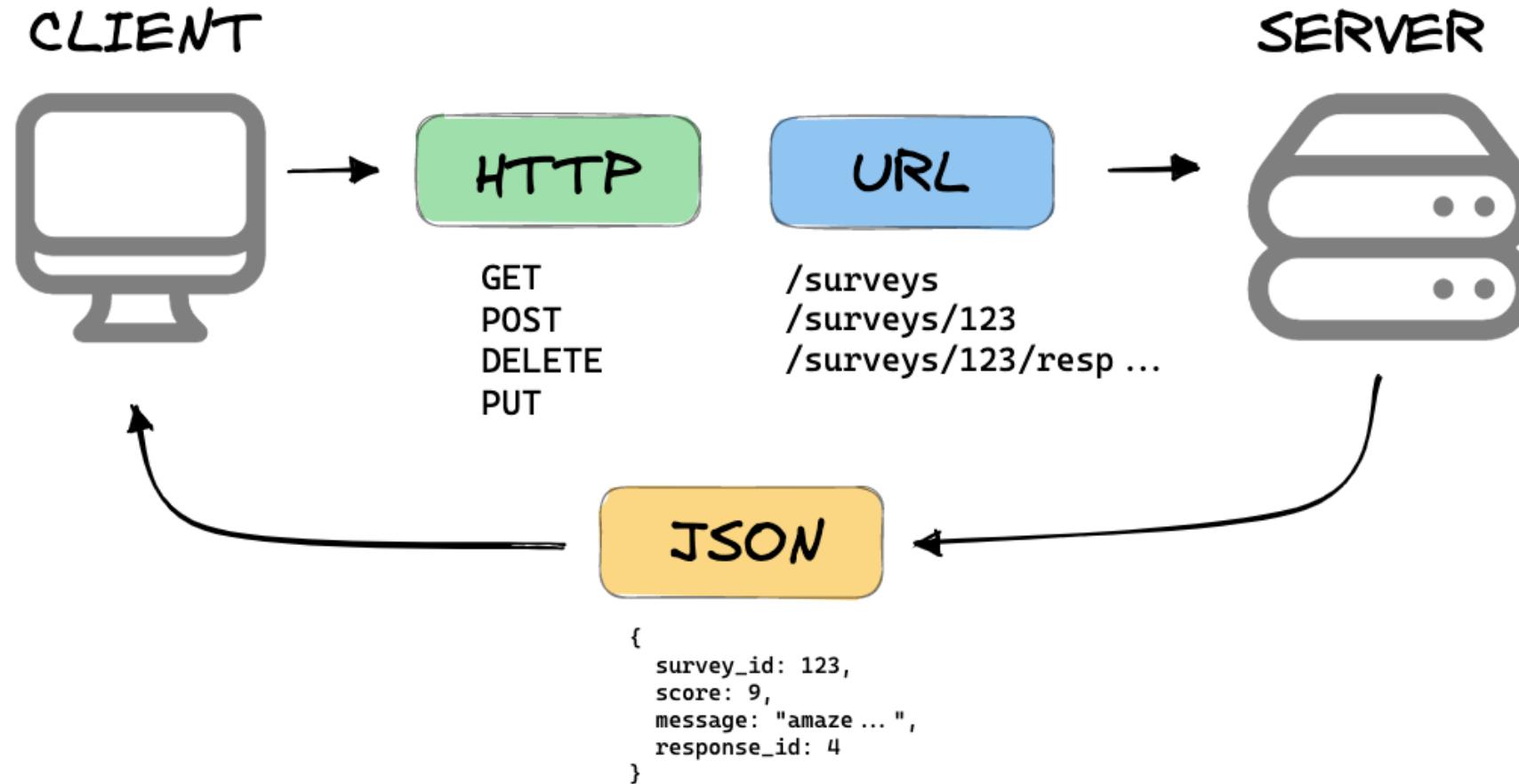
Update an  
API resource



## DELETE

Delete an  
API resource

# WHAT IS A REST API?



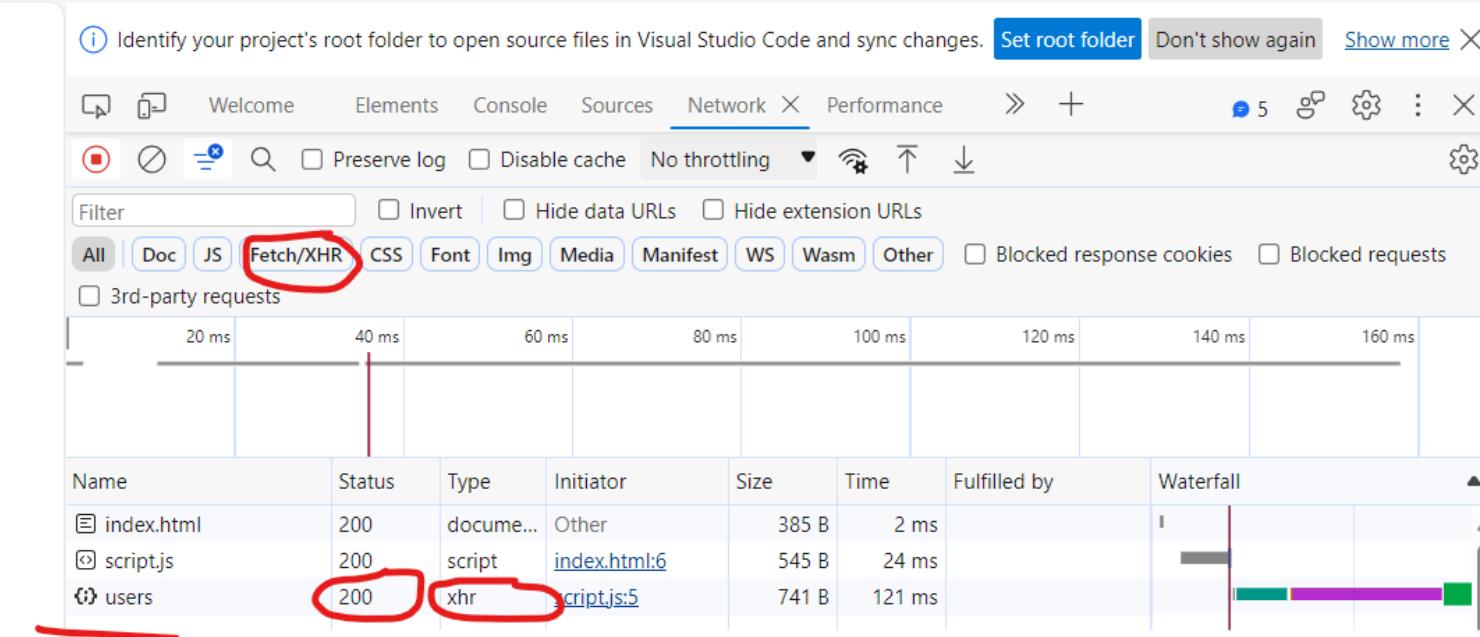
[mannhowie.com](http://mannhowie.com)

# Responses

תשובה של שרת לבקשת

19.11.23

## XMLHttpRequest



Filter  Invert  Hide data URLs  Hide extension URLs

All Doc JS Fetch/XHR CSS Font Img Media Manifest WS Wasm Other  Blocked response cookies  Blocked requests  
 3rd-party requests

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms

Name	Headers	Preview	Response	Initiator	Timing
index.html					
script.js					
users	<b>Headers</b>				

**General**

Request URL: <https://jsonplaceholder.typicode.com/users>  
Request Method: GET  
Status Code: 200 OK  
Remote Address: 104.21.59.19:443  
Referrer Policy: strict-origin-when-cross-origin

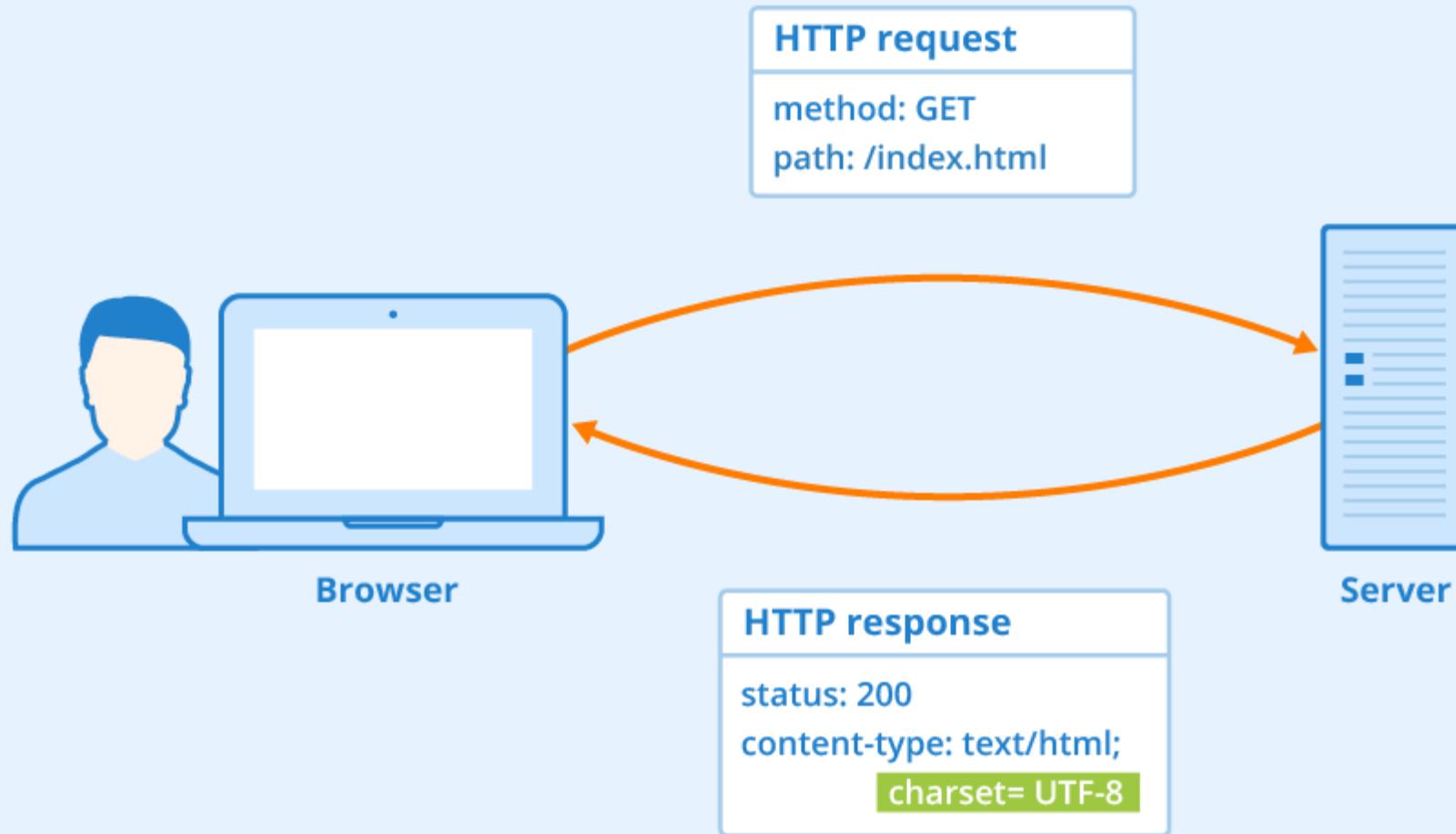
**Response Headers**

Access-Control-Allow- true  
Credentials:  
Access-Control-Allow-Origin: null  
Age: 2827  
Alt-Svc: h3=":443"; ma=86400  
Cache-Control: max-age=43200  
Cf-Cache-Status: HIT  
Cf-Ray: 82dabef698109cb-HFA  
Content-Encoding: br  
Content-Type: application/json; charset=utf-8  
Date: Wed, 29 Nov 2023 12:10:11 GMT  
Etag: W/"160d-1eMSsxeJFnVLRBmYJSbCiJZ1qQ"  
Expires: -1  
Nel: {"report\_to": "heroku-}

3 requests 1.7 kB transferred 6.6 kB

# Headers

אתר עם דатаה לדוגמא בחינוך



# Request & Response headers in 'Developer tools'

Name	Headers	Preview	Response	Initiator	Timing
index.html	X-Powered-By: express				
script.js					
users					
<b>Request Headers</b> <pre>:authority: jsonplaceholder.typicode.com :method: GET :path: /users :scheme: https :accept: /* :accept-encoding: gzip, deflate, br :accept-language: en-US,en;q=0.9 :if-none-match: W/"160d-1eMSsseJRfnVLRBmYJSbCijZ1qQ" :origin: null :sec-ch-ua: "Microsoft Edge";v="119", "Chromium";v="119", "Not?A_Brand";v="24" :sec-ch-ua-mobile: ?0 :sec-ch-ua-platform: "Windows" :sec-fetch-dest: empty :sec-fetch-mode: cors :sec-fetch-site: cross-site :user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36 Edg/119.0.0.0</pre>					
3 requests 1.7 kB transferred 6.6 kB					

Name	Headers	Preview	Response	Initiator	Timing
index.html					
script.js					
users					
<b>General</b> <p>Request URL: https://jsonplaceholder.typicode.com/users      Request Method: GET      Status Code: 200 OK      Remote Address: 104.21.59.19:443      Referrer Policy: strict-origin-when-cross-origin</p> <b>Response Headers</b> <pre>Access-Control-Allow-Origin: true Access-Control-Allow-Credentials: null Access-Control-Allow-Origin: 2827 Age: 2827 Alt-Svc: h3=":443"; ma=86400 Cache-Control: max-age=43200 Cf-Cache-Status: HIT Cf-Ray: 82dabef698109cb-HFA Content-Encoding: br Content-Type: application/json; charset=utf-8 Date: Wed, 29 Nov 2023 12:10:11 GMT Etag: W/"160d-1eMSsseJRfnVLRBmYJSbCijZ1qQ" Expires: -1 Nel: {"report_to": "heroku-00000000000000000000000000000000"}</pre>					
3 requests 1.7 kB transferred 6.6 kB					

Filter  Invert  Hide data URLs  Hide extension URLs

All Doc JS Fetch/XHR CSS Font Img Media Manifest WS Wasm Other  Blocked response cookies  Blocked requests  
 3rd-party requests

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms

Name	Headers	Preview	Response	Initiator	Timing
index.html					
script.js					
users	<b>Headers</b>				
	<b>General</b>				
	Request URL: <a href="https://jsonplaceholder.typicode.com/users">https://jsonplaceholder.typicode.com/users</a>				
	Request Method: GET				
	Status Code: 200 OK				
	Remote Address: 104.21.59.19:443				
	Referrer Policy: strict-origin-when-cross-origin				
	<b>Response Headers</b>				
	Access-Control-Allow-true				
	Credentials:				
	Access-Control-Allow-Origin: null				
	Age: 2827				
	Alt-Svc: h3=":443"; ma=86400				
	Cache-Control: max-age=43200				
	Cf-Cache-Status: HIT				
	Cf-Ray: 82dabef698109cb-HFA				
	Content-Encoding: br				
	Content-Type: application/json; charset=utf-8				
	Date: Wed, 29 Nov 2023 12:10:11 GMT				
	Etag: W/"160d-1eMSsxeJFnVLRBmYJSbCiJZ1qQ"				
	Expires: -1				
	Nel: {"report_to": "heroku-"} 3 requests 1.7 kB transferred 6.6 kB				

# HTTP Response Statuses

# HTTP STATUS CODES



# HTTP STATUS CODES

## 2xx Success

200

Success / OK

## 3xx Redirection

301

Permanent Redirect

302

Temporary Redirect

304

Not Modified

## 4xx Client Error

401

Unauthorized Error

403

Forbidden

404

Not Found

405

Method Not Allowed

## 5xx Server Error

501

Not Implemented

502

Bad Gateway

503

Service Unavailable

504

Gateway Timeout

# Content types

Application

- application/EDI-X12
- application/EDIFACT
- application/javascript
- application/octet-stream
- application/ogg
- application/pdf
- application/xhtml+xml
- application/x-shockwave-flash

**application/json**

- application/ld+json
- application/xml
- application/zip

**application/x-www-form-urlencoded**

Image

- image/gif
- image/jpeg
- image/png
- image/tiff
- image/vnd.microsoft.icon
- image/x-icon
- image/vnd.djvu
- image/svg+xml

Text

- text/css
- text/csv
- text/html**
- text/javascript (obsolete)
- text/plain
- text/xml

# דата לדוגמה

אתר עם דата לדוגמה בcheinm

[JSONPlaceholder](#)[Guide](#) [Sponsor this project](#) [Blog](#) [My JSON Server](#)

# {JSON} Placeholder

Free fake API for testing and prototyping.

Powered by [JSON Server](#) + [LowDB](#). Tested with [XV](#).

**Serving ~2 billion requests each month.**

# Resources

JSONPlaceholder comes with a set of 6 common resources:

[/posts](#) 100 posts

[/comments](#) 500 comments

[/albums](#) 100 albums

[/photos](#) 5000 photos

[/todos](#) 200 todos

[/users](#) 10 users

`https://jsonplaceholder.typicode.com/posts`

`https://jsonplaceholder.typicode.com/posts/1`

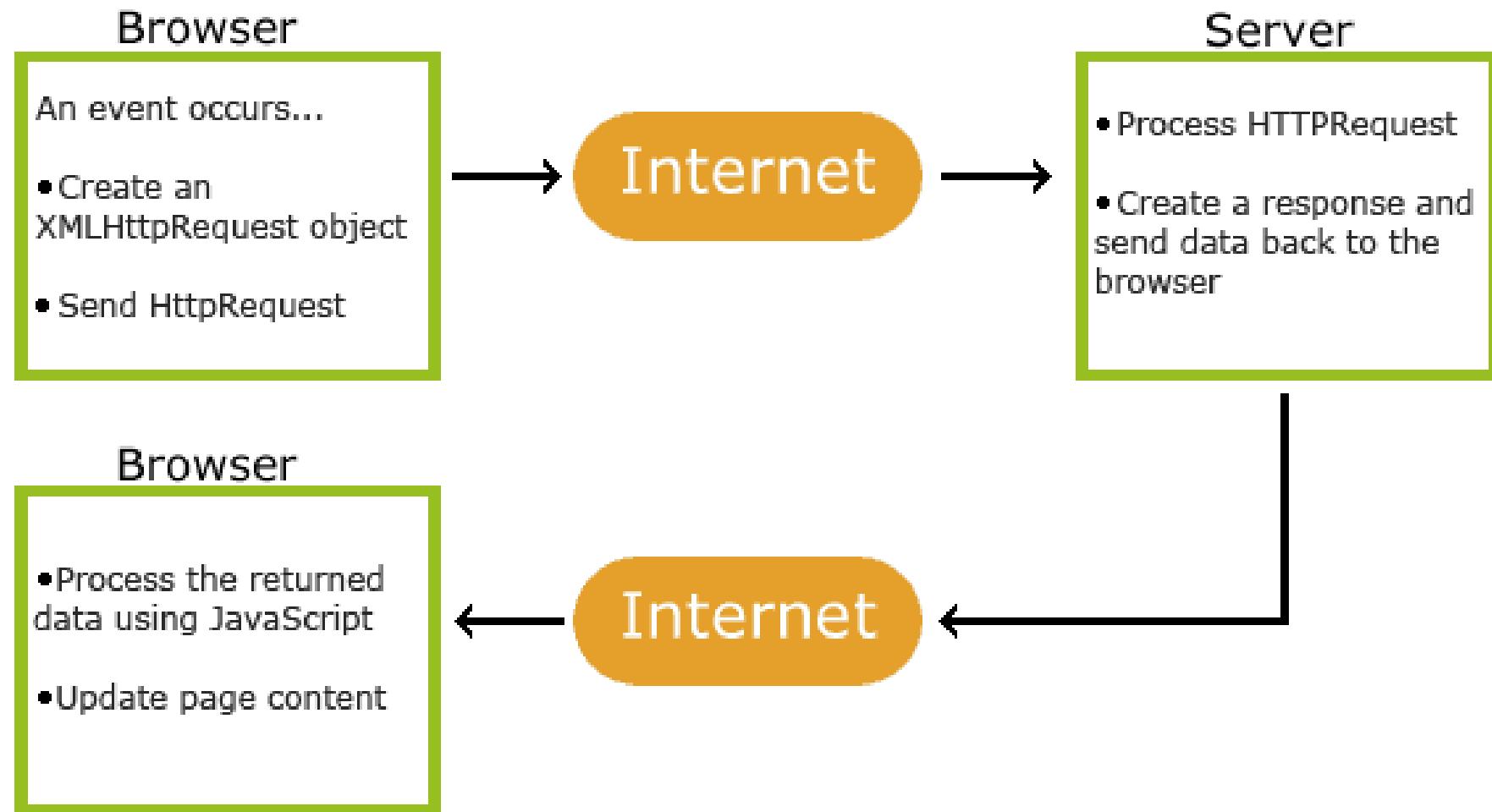
The available nested routes are:

- [/posts/1/comments](#)
- [/albums/1/photos](#)
- [/users/1/albums](#)
- [/users/1/todos](#)
- [/users/1/posts](#)

- תפעילו את הקוד על אפשרויות שונות
  - /comments
  - /todos
  - /todos/1
  - /todos/2
  - /posts
  - /posts/2/comments
  - /photos
- לנסות להציג תמונות באמצעות תגית `<img>` ב-HTML, וראות אם רואים תמונה (להעתיק את הכתובת ולשים בתאgit)

# איך קולטים את דатаה? **Callback**

שלבים בתקשרות וסטטוס התשובה



- Listen to `xhr` events for response.

These three events are the most widely used:

- `load` – when the request is complete (even if HTTP status is like 400 or 500), and the response is fully downloaded.
- `error` – when the request couldn't be made, e.g. network down or invalid URL.
- `progress` – triggers periodically while the response is being downloaded, reports how much has been downloaded.

# Handling successful response

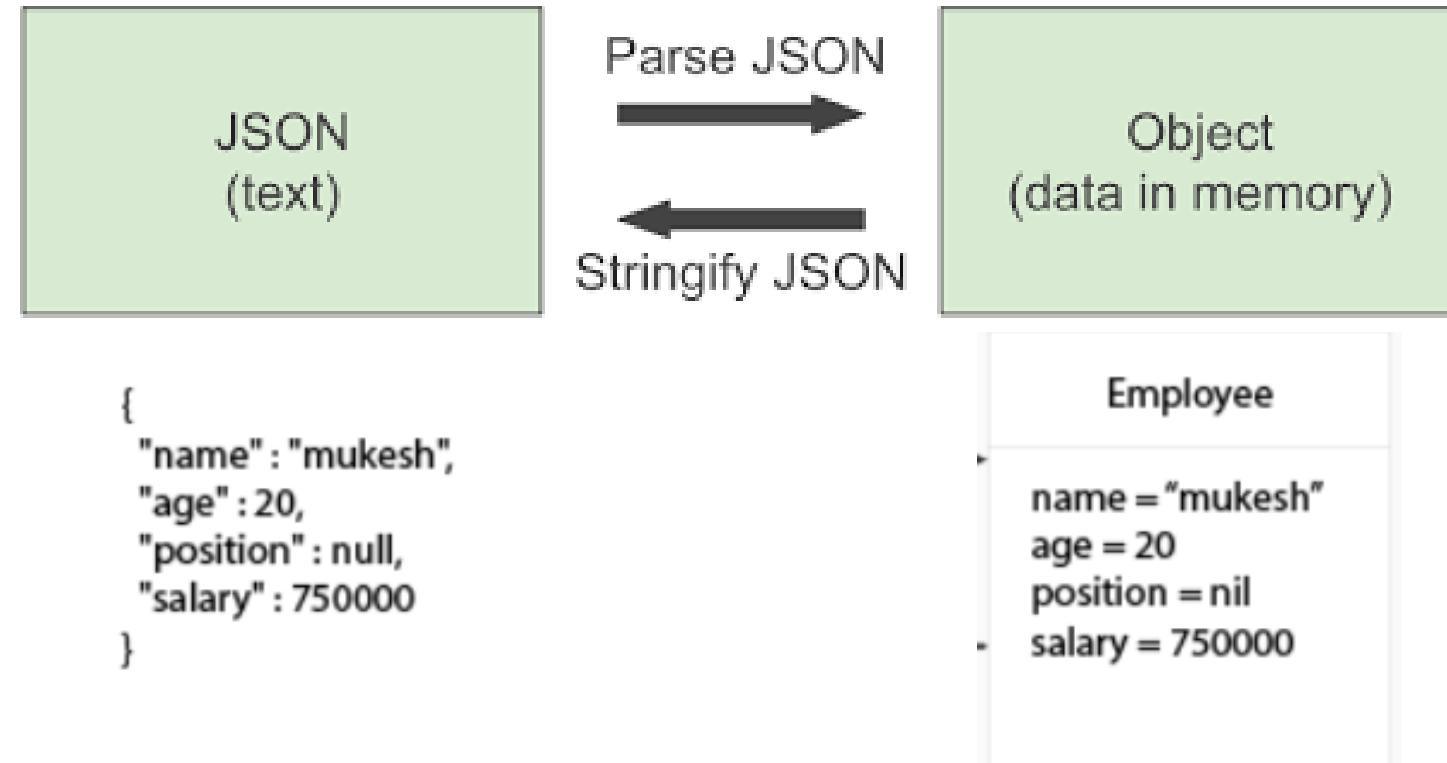
# הציג התשובה ב-HTML

```
const xhr = new XMLHttpRequest();
xhr.onload = function() {
    document.getElementById("output").innerHTML = xhr.responseText;
};

xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

# המרת תשובה JSON לאובייקט ושינוי בדף HTML

```
const xhr = new XMLHttpRequest();
xhr.onload = function() {
  const objResponse = JSON.parse(xhr.responseText);
  console.log(objResponse);
}
xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```



- **בשיעור:**

- להציג מידע על סטודנט אחד ב-HTML בצורה יפה
  - להשתמש בlienק <https://jsonplaceholder.typicode.com/users/1>
  - אפשר להשתמש בטבלה או שורה אחורי שורה, או אוסף של תגיות <div>
- לשנות את הקוד של onload כך שנשתמש ב-Arrow Function
- לשנות ל-function Callback function (לפצל לפונקציה נפרדת ולקרא לה)

- **שיעור בית:**

- להציג את כל התלמידים בטבלה כולל כל המידע
  - להשתמש בlienק <https://jsonplaceholder.typicode.com/users>
- לשנות את הקוד של onLoad כך שימוש בקוד הבא
  - xhr.addEventListener("load", transferComplete);
- שימושו לב שימושים במתודה אחרת בדומה למתודה שמחברת אינטראקטיה בדף HTML ומשתמשים ב- callback function

# Handling errors

# 1 – טיפול בבעית חיבור

```
const xhr = new XMLHttpRequest();
xhr.onload = function() {
    document.getElementById("output").innerHTML = xhr.responseText;
};

xhr.onerror = function() {
    console.log(`Error: ${xhr.status} : ${xhr.statusText}`);
};

xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

4XX Client Error		5XX Server Error	
<b>400</b>	Bad Request	<b>500</b>	Internal Server Error
<b>401</b>	Unauthorized	<b>501</b>	Not Implemented
<b>402</b>	Payment Required	<b>502</b>	Bad Gateway
<b>403</b>	Forbidden	<b>503</b>	Service Unavailable
<b>404</b>	Not Found	<b>504</b>	Gateway Timeout
<b>405</b>	Method Not Allowed	<b>505</b>	HTTP Version Not Supported
<b>406</b>	Not Acceptable	<b>506</b>	Variant Also Negotiates
<b>407</b>	Proxy Authentication Required	<b>507</b>	Insufficient Storage
<b>408</b>	Request Timeout	<b>508</b>	Loop Detected
<b>510</b>		<b>510</b>	Not Extended
<b>511</b>		<b>511</b>	Network Authentication Required
<b>599</b>		<b>599</b>	Network Connect Timeout Error

# HTTP STATUS CODES



# HTTP STATUS CODES

## 2xx Success

200

Success / OK

## 3xx Redirection

301

Permanent Redirect

302

Temporary Redirect

304

Not Modified

## 4xx Client Error

401

Unauthorized Error

403

Forbidden

404

Not Found

405

Method Not Allowed

## 5xx Server Error

501

Not Implemented

502

Bad Gateway

503

Service Unavailable

504

Gateway Timeout

## 2 – טיפול בשגיאות תשובה שרת

```
const xhr = new XMLHttpRequest();
xhr.onload = function() {
    if (xhr.status === 200) {
        console.log(`Loaded content`);
        document.getElementById("output").innerHTML = xhr.responseText;
    } else if (xhr.status === 404) {
        console.log(`Error: ${xhr.status} ${xhr.statusText}`);
    }
};
xhr.onerror = function() {
    console.log(`Error: ${xhr.status} : ${xhr.statusText}`);
};

xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

# תרגיל

- **בשיעור:**

- לשנות את הקוד כך שנשתמש ב-case במקום if-then-else

# Handling progress

....

```
xhr.onerror = function() { // only triggers if the request couldn't be made at all
    console.log(`Error: ${xhr.status} : ${xhr.statusText}`);
};
```

```
xhr.onprogress = function(event) { // triggers periodically
    // event.loaded - how many bytes downloaded
    // event.lengthComputable = true if the server sent Content-Length header
    // event.total - total number of bytes (if lengthComputable)
    console.log(`Received ${event.loaded} of ${event.total}`);
};
```

```
xhr.open("GET", "https://jsonplaceholder.typicode.com/users");
xhr.send();
```

# POST

# POST

```
var data = JSON.stringify({ title: 'foo', body: 'bar', userId: 1 });
xhr.open("POST", "https://jsonplaceholder.typicode.com/posts");
xhr.setRequestHeader("Content-type", "application/json;charset=UTF-8");
xhr.send(data);
```

# POST

```
if (xhr.status === 200) {  
    console.log(`Loaded content`);  
    document.getElementById("output").innerHTML = xhr.responseText;  
} else if (xhr.status === 201) {  
    console.log(`Created content`);  
    document.getElementById("output").innerHTML = xhr.responseText;  
} else if (xhr.status === 404) {  
    console.log(`Error: ${xhr.status} ${xhr.statusText}`);  
}
```

# תרגיל

- **בשיעור או שיעורי בית:**
- להוסיף שימוש ב-POST לעוד 3 URLs

# קריאות XMLHttpRequest

JS\_Advanced\_XMLHttpRequest.html

ב חלק זה נלמד מגוון נושאים.  
בסיסם הנושא תוכלו לענות על השאלות הבאות:

```
function demo1() {
    // Create New XMLHttpRequest object
    const request = new XMLHttpRequest();

    // callback function for onload
    request.onload = function () {
        console.log(request);
        console.log(request.status); // Checking the status
        console.log(request.response); // Checking the response (data)

        const myObj = JSON.parse(request.response)
        console.log(myObj)
    }

    // Open and Send request
    request.open("GET", "https://pokeapi.co/api/v2/pokemon/squirtle");
    request.send();
}
```

- מהי קריאה לשרת?
- איך מבצעים קריאה לשרת?
- מה מכיל אובייקט מסווג XMLHttpRequest?
- אילו פעולות צריך לבצע לביצוע הקריאה לשרת?
- על מה אחראית המתודה שב-send? onload?
- על מה אחראית המתודה open?
- על מה אחראית המתודה send?
- באיזה סדר צריך לבצע את הפעולות שיש לבצע?
- איך ניתן לגשת למידע שהתקבל?
- איך ניתן לבדוק את סטוס הפניה?
- מהם אפשרויות ה http הנפוצים בשימוש? (GET, POST, PUT, DELETE)
- איך ניתן לגשת להודעות שגיאה (במידה וקיימות)

# תרגול קריאות XMLHttpRequest

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

## getData

country	“Israel”
---------	----------

## printData

country	Prompt
countryData	<code>getData(country)</code>

תרגיל	תיאור המשימה
Ex-1	<p>צחו פונקציה חדשה וקראו לה <code>getData</code>.      פונקציה זו תדע לקבל פרמטר שהוא שם של מדינה.      פונקציה זו תדע לפנות ל-API של מדינות העולם על מנת להביא מידע על המדינה שקיבלו כפרמטר.      כתובתו של ה-API היא: <a href="https://restcountries.com">https://restcountries.com</a></p>
Ex-2	<p>באמצעות שימוש ב-<code>XMLHttpRequest</code>, פנו ל-API וייבאו את הנתונים המבוקשים.      היפכו את התוצאה המתתקבלת ל-JSON, הפונקציה תדפיס לkonsole ותחזיר תוצאה זו.</p>
Ex-3	<p>צחו פונקציה חדשה בשם <code>printData</code>.      פונקציה זו תבקש מהמשתמש שם של מדינה,      לאחר מכן הפונקציה תשתמש בפונקציה <code>getData</code> על מנת להעביר את שם המדינה ולקבל את התוצאה.      בסוף פונקציה זו תדפיס את התוצאה לkonsole.</p>
Ex-4	<p>השתמשו ב-<code>printData</code> באמצעות שלייחת ערכים שונים ובירדו האם ההדפסות שבפונקציה זו יוצאות לפועל לפני או אחרי שחרורה התשובה מהשרת.</p>

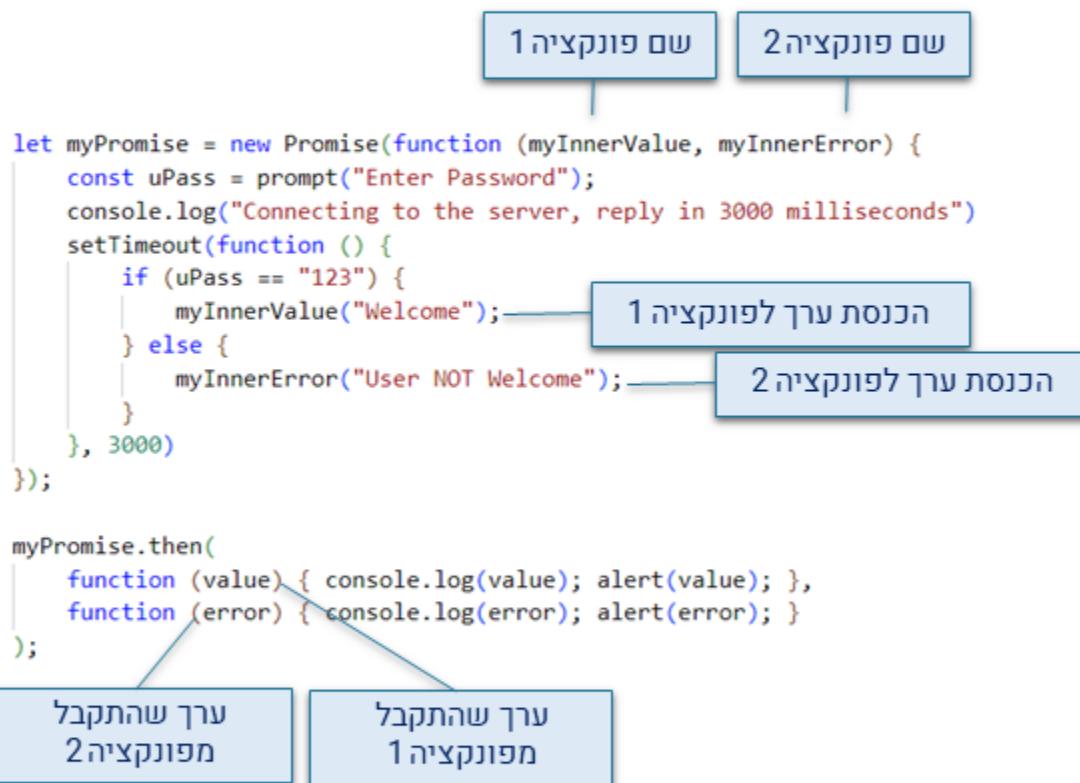
# מחיקת Promise



# מחלקה Promise

JS\_Advanced\_Promise.html

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



- מה זה **Promise** ?
- מה הבדל בין **reject** ל-**resolve** ?
- מה נדרש לשולח לבניאי של **Promise** ?
- באיזה מצבים נדרש להשתמש בטכניקה זו?
- מה הם המצבים האפשריים של **Promise** ?
- מה הם המאפיינים והמתודות המיוחדים למחלקה **Promise** ?

# מחלקה Promise

JS\_Advanced\_Promise.html

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

```
let myPromise = new Promise(function (funcValue, funcError) {
  console.log("Connecting to the server, reply in ??? milliseconds");
  const url = 'https://pokeapi.co/api/v2/pokemon/squirtle';
  const request = new XMLHttpRequest();
  request.open("GET", url);
  request.onload = function () {
    if (request.readyState == 4 && request.status == 200) {
      funcValue(request.response);
    } else {
      funcError("Error !!!!");
    }
  };
  request.send();
  console.log("end");
});

myPromise.then(
  function (value1) { console.log(value1); },
  function (value2) { console.log(value2); }
);
```

- מה זה **Promise** ?
- מה הבדל בין **? reject** ל **? resolve** ?
- מה נדרש לשולח לבנייה של **Promise** ?
- באיזה מצבים נדרש להשתמש בטכניקה זו?
- מה הם המצבים האפשריים של **Promise** ?
- מה הם המאפיינים והמתודות המיוחדים למחלקה **Promise** ?

# מחלקה Promise

JS\_Advanced\_Promise.html

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

```
let myPromise = new Promise(function (funcValue, funcError) {
    console.log("Connecting to the server.. ");

    const request = new XMLHttpRequest();
    request.onload = function () {
        if (request.readyState == 4 && request.status == 200) {
            funcValue(request.response);
        } else {
            funcError("Error !!!!");
        }
    };

    request.open("GET", 'https://pokeapi.co/api/v2/pokemon/squirtle');
    request.send();
    console.log("end");
});

myPromise.then(
    function (value1) { console.log(value1); },
    function (value2) { console.log(value2); }
);
```

- מה זה **Promise** ?
- מה הבדל בין **reject**-ל-**resolve** ?
- מה נדרש לשולח לבנייה של **Promise** ?
- באיזה מצבים נדרש להשתמש בטכנית זו?
- מה הם המצבים האפשריים של **Promise** ?
- מה הם המאפיינים והمتודות המיוחדים למחלקה **Promise** ?

# מחלקה - Promise

[JS\\_Advanced\\_Promise\\_Then\\_Catch.html](#)

[JS\\_Advanced\\_PromiseAll.html](#)

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

const pokemonsUrl = 'https://pokeapi.co/api/v2/pokemon?limit=20';
let promise = getPromise(pokemonsUrl);

const promiseHandler = () => {
  promise.then((result) => {
    let pokemonsArray = [...JSON.parse(result)['results']]
    pokemonsArray.forEach(item => console.log(item['name']));
  }).catch((error) => {
    console.log(error.message); // The code will run only if there is an error
  }).then(() => {
    console.log('\n this will be printed at the end of program, bye')
  });
}

promiseHandler();
```

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

- **למה משמשת המתודה `?then`?**
- **למה משמשת המתודה `?catch`?**
- **איזה פרמטרים ניתן לשלוח למתודה `?then`?**
- **אין ניתן להשתמש ב-`then` למטרות סינכרון?**
- **איזה פרמטרים ניתן לשלוח למתודה `?catch`?**
- **מה היתרונות בשימוש ב-`catch`?**
- **אין נוכל למשוך נתונים מ-API באמצעות שימוש ב-`XMLHttpRequest` ו-`Promise`.**
- **מה נוכל לעשות במקרה של שגיאה שמקורה בשרת?**

# תרגול מחלוקת Promise

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

## getData

country	“Israel”
---------	----------

תרגיל	תיאור המשימה
Ex-1	<p>צרו פונקציה חדשה וקרוו לה <code>getData</code>.</p> <p>פונקציה זו תדע לקבל פרמטר שהוא שם של מדינה.</p> <p>פונקציה זו תדע לפנות ל-<code>API</code> של מדינות העולם על מנת להביא מידע על המדינה שקיבלו כפרמטר.</p> <p>התובתו של ה-<code>API</code> היא: <a href="https://restcountries.com/">https://restcountries.com/</a></p>

## printData

country	Prompt
countryData	<code>getData(country)</code>

Ex-2	<p>באמצעות שימוש במחלקות <code>Promise</code> ו-<code>XMLHttpRequest</code>, פנו ל-<code>API</code> וייבאו את הנתונים המבוקשים.</p> <p>הfonקציה תחזיר את ה-<code>Promise</code>.</p>
Ex-3	<p>צרו פונקציה חדשה בשם <code>printData</code>.</p> <p>פונקציה זו תבקש מהמשתמש שם של מדינה,</p> <p>לאחר מכן הפונקציה תשתחש בפונקציה <code>getData</code> על מנת להעביר את שם המדינה ולקבל את התוצאה.</p> <p>בסוף פונקציה זו תמיר את התוצאה ל-<code>JSON</code> ותדפיס אותה לkonsole באמצעות שימוש ב-<code>then</code>.</p>

## setImages

countries	[country1, country2, country3, country4, country5]
-----------	--

Ex-4	<p>צרו פונקציה חדשה בשם <code>setImages</code>, ייבאו בה 5 דגליים של מדינות על ידי שימוש בפונקציה <code>getData</code>.</p> <p>לבסוף הציגו תמונות אלו על גבי הדףון ללא יצירת אלמנטים מראש במסך ה-<code>HTML</code> באמצעות שימוש ב-<code>then</code>.</p>
Ex-5	<p>שילחו שמות שגויים של מדינות ל-<code>API</code>, טפו בשגיאות באמצעות שימוש במתודה <code>catch</code> כך שבמקרה והתקבל שגיאה אז תציג תמונה קבועה מראש במקום התמונה שלא התקבלה מהשרת.</p>



# הטמעה ב-Github Pages

מרצה: תומר שגיא

19/11/2023



Code

Pull requests

Actions

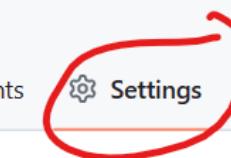
Projects

Wiki

Security

Insights

Settings



General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

## GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

### Build and deployment

#### Source

Deploy from a branch ▾

#### Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

None ▾

Save

Select branch

Select branch

master

✓ None

Try GitHub Enterprise risk-free for 30 days

Learn more about the visibility of your GitHub Pages site

 [General](#) [Access](#) [Collaborators](#) [Moderation options](#) [Code and automation](#) [Branches](#) [Tags](#) [Rules](#) [Actions](#) [Webhooks](#) [Environments](#) [Codespaces](#) [Pages](#) [Security](#)

## GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

### Build and deployment

#### Source

[Deploy from a branch](#)

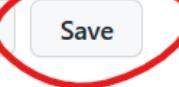
#### Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

master

/ (root)

[Save](#)



#### Visibility

[GITHUB ENTERPRISE](#)

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. A privately published site can only be accessed by people with read access to the repository the site is published from. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise.



tomer79sagi / W310523ER

Type  to search[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)[General](#)[Access](#)[Collaborators](#)[Moderation options](#)[Code and automation](#)[Branches](#)[Tags](#)[Rules](#)[Actions](#)[Webhooks](#)[Environments](#)[Codespaces](#)[Pages](#)[Security](#)[Code security and analysis](#)[Deploy keys](#)

## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://tomer79sagi.github.io/W310523ER/>

Last deployed by tomer79sagi 2 minutes ago

[Visit site](#)

### Build and deployment

#### Source

[Deploy from a branch](#)

#### Branch

Your GitHub Pages site is currently being built from the `master` branch. [Learn more about configuring the publishing source for your site.](#)

[master](#)[/ \(root\)](#)[Save](#)

Learn how to [add a Jekyll theme](#) to your site.

Your site was last deployed to the [github-pages](#) environment by the [pages build and deployment](#) workflow.

[Learn more about deploying to GitHub Pages using custom workflows](#)

# איך פותחים קבצים HTML מתיקיות ב-GITHUB?

[tomer79sagi.github.io/W310523ER/W310523ER-2023.11.19/](https://tomer79sagi.github.io/W310523ER/W310523ER-2023.11.19/) •

The screenshot shows a GitHub repository interface. On the left, there's a sidebar with navigation links like 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below that is a 'Files' section with a dropdown menu set to 'master'. A search bar and a 'Go to file' input field are also present. The main area displays a commit history for a branch named 'W310523ER / W310523ER-2023.11.19'. The first commit, by 'tomer79sagi', adds a '.gitignore' file. A message below the commit says 'This branch is 3 commits ahead of zadokdaniel/W310523ER:master'. The commit list includes an entry for 'index.html' which was added and committed 2 minutes ago. Two red circles highlight the branch name 'W310523ER / W310523ER-2023.11.19' and the 'index.html' file entry.

Name	Last commit message	Last commit date
index.html	Added gitignore file	2 minutes ago



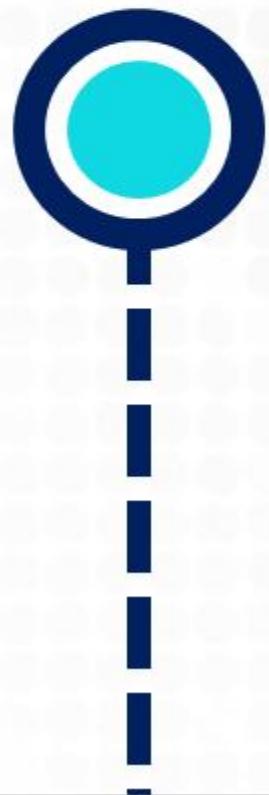
# פרויקט סיום

מרצה: תומר שגיא

19/11/2023

# Page Builder

משימת בונה מסכימים





## בונה מסכים

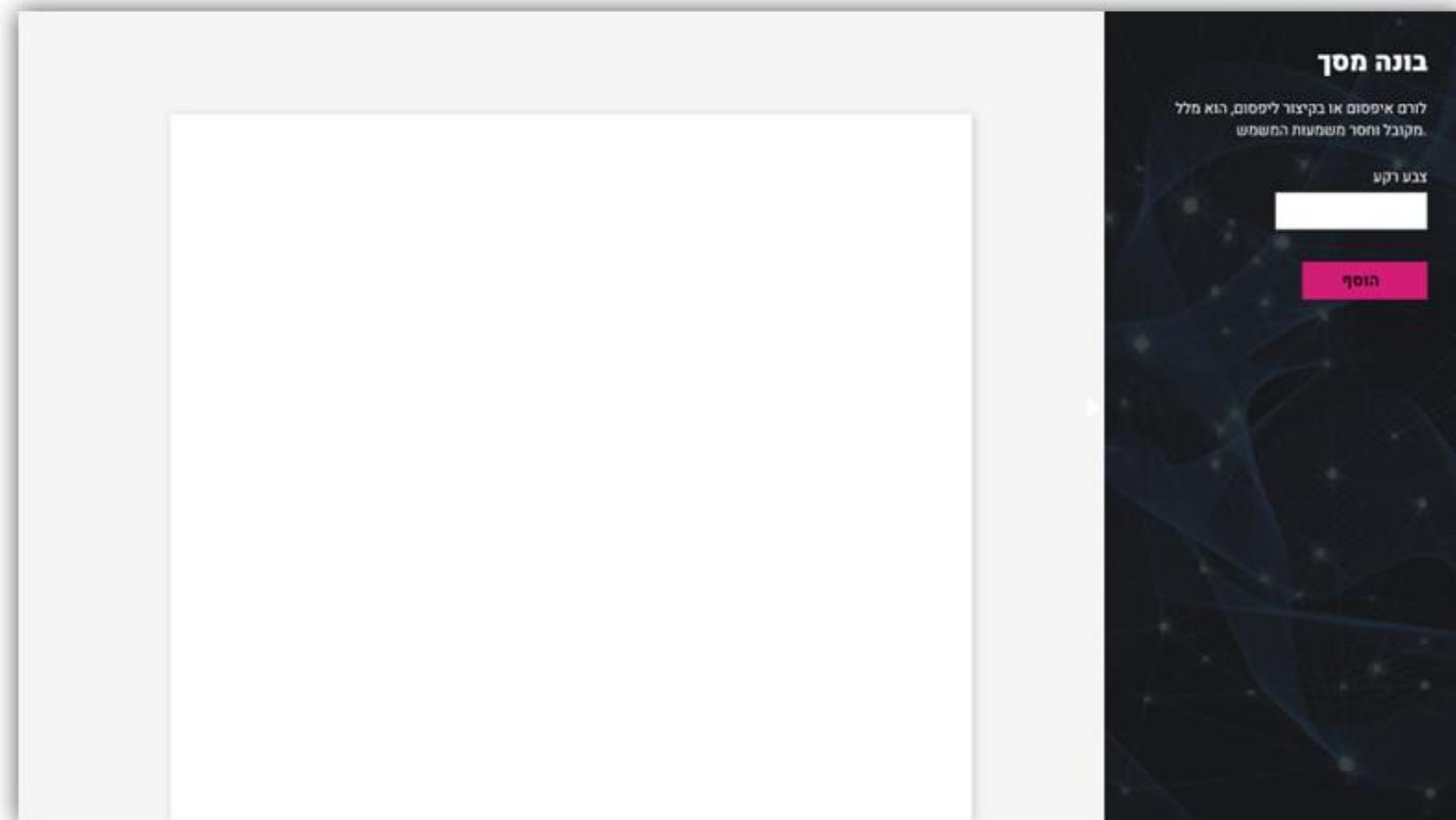
במסגרת המשימה נבנה מערכת חכמה המאפשרת למשתמש ליצור עמודי אינטרנט ללא צורך בקוד, המערכת תאפשר למשתמש להגדיר ולעצב את האלמנט ולהוסיף למסך בלחיצה כפтор.

[DOM](#)[תנאים](#)[משתנים](#)

# בונה מסכים

JS\_DOM\_Mission\_PageBuilder\_1.html  
JS\_DOM\_Mission\_PageBuilder\_2.html

משימה כוללת ייצירת ממשק ליצירת עמודי אינטרנט.  
המשימה מתחילה עם הממשק הבא:

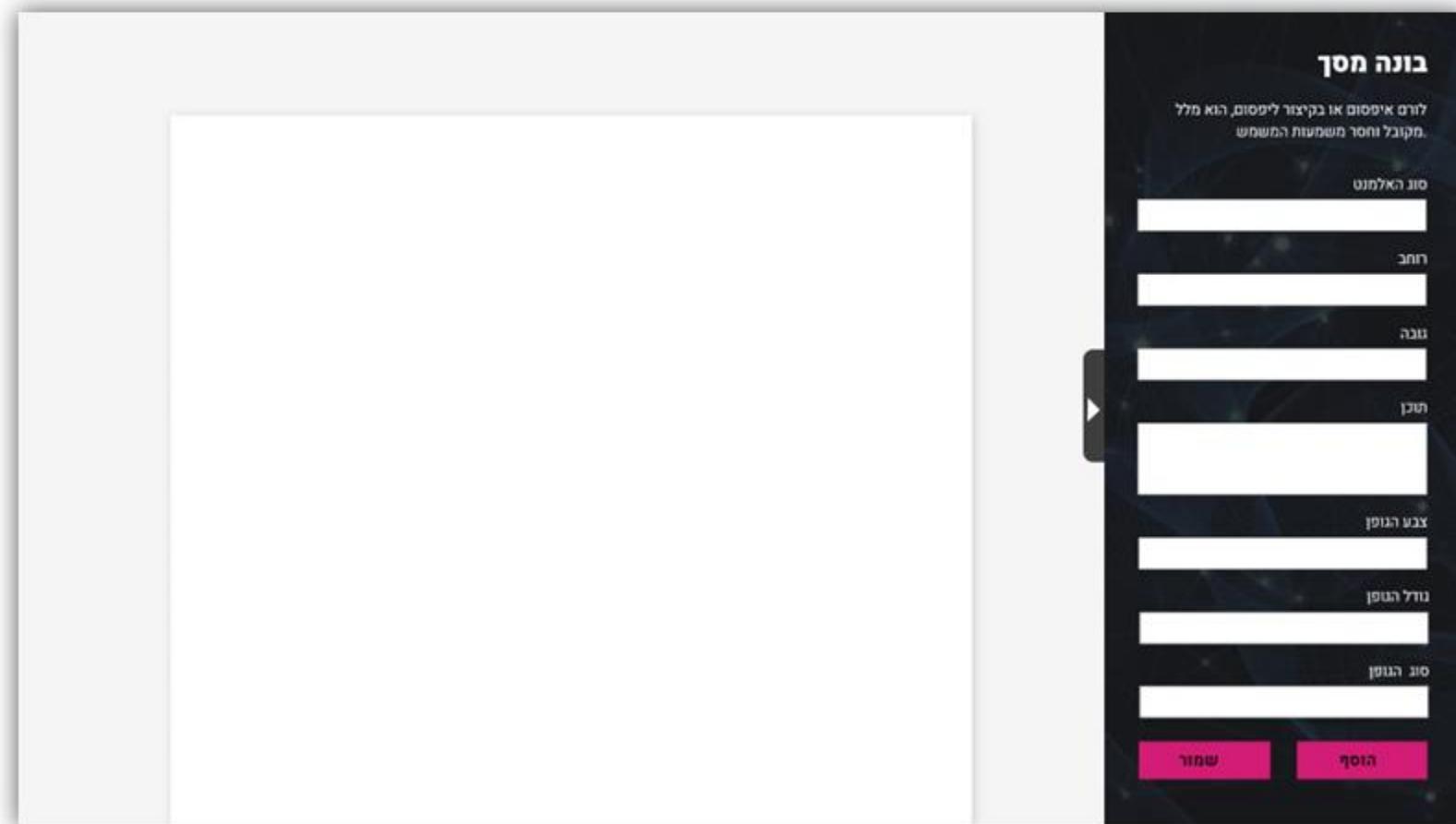


# בונה מסכים

JS\_DOM\_Mission\_PageBuilder\_3.html

JS\_DOM\_Mission\_PageBuilder\_4.html

משימה כוללת ייצרת ממשק ליצירת עמודי אינטרנט.  
המשימה מתפתחת וככה נראה הממשק בהמשך:



2

JS\_DOM\_Mission\_PageBuilder\_1.html  
 JS\_DOM\_Mission\_PageBuilder\_2.html  
 JS\_DOM\_Mission\_PageBuilder\_3.html  
 JS\_DOM\_Mission\_PageBuilder\_4.html

צورو קובץ לטובת הפרויקט, התחילה מיצירת ממתק המשמש.  
 עברו כל משימה צרו מסמן חדש.

<b>תרגיל</b>	<b>תיאור המשימה</b>
demo1	<p><b>הוספת אלמנט למסן</b>          במסנן ה- JS צרו פונקציה חדשה.          הפונקציה תיצור אלמנט חדש ותוסיף אותו למסן.          האלמנט מסוג <code>div</code> ברוחב וגובה של 100 פיקסל ומסגרת רציפה שחורה.</p>
demo2	<p><b>הוספת אפשרות להגדיר צבע ורקע לאלמנט:</b>          הוסיפו למסנן ה- HTML שדה קלט אשר יאפשר לקבל [צבע ורקע]          במסנן ה- JS צרו פונקציה חדשה.          הפונקציה תיצור אלמנט חדש ותוסיף אותו למסן.          האלמנט מסוג <code>div</code> ברוחב וגובה של 100 פיקסל ומסגרת רציפה שחורה.          צבע הרקע של האלמנט יהיה בהתאם [צבע ורקע]</p>
demo3	<p><b>הוספת אפשרות להגדיר רוחב וגובה לאלמנט:</b>          הוסיפו למסנן ה- HTML 2 שדות קלט אשר יאפשר לקבל [רוחב אלמנט] [גובה אלמנט]          במסנן ה- JS צרו פונקציה חדשה.          הפונקציה תיצור אלמנט חדש ותוסיף אותו למסן.          האלמנט מסוג <code>div</code> ומסגרת רציפה שחורה.          ברוחב וגובה של האלמנט יהיו בהתאם [רוחב אלמנט] [גובה אלמנט]          האלמנט יהיה בהתאם [צבע ורקע] מהמשימה הקודמת</p>



צورو קובץ לטובת הפרויקט, תחילה מיצירת ממתק המשמש.  
עבור כל משימה צרו מסמן חדש.

תרגיל	תיאור המשימה
4	<p><b>הוספת אפשרות להוסיף תוכן לאלמנט:</b>            הוסיפו למסנן ה- HTML שדה קלט [textarea] אשר יאפשר לקבל [תוכן האלמנט]            במסנן ה- SJ צרו פונקציה חדשה.            הפונקציה תיצור אלמנט חדש ותוסיף אותו למסך.            האלמנט מסוג <code>div</code> ומסגרת רציפה שחורה.            האלמנט יהיה בהתאם [רוחב אלמנט] [גובה אלמנט] [צבע רקע]            מהמשימה הקודמת.            האלמנט יוכל את התוכן בהתאם ל[תוכן האלמנט] שהתקבל</p>
5	<p><b>הוספת אפשרות להוסיף צבע וגודל הגוף האלמנט:</b>            הוסיפו למסנן ה- HTML 2 שדות קלט אשר יאפשרו לקבל [צבע גוף] [גודל גוף]            במסנן ה- SJ צרו פונקציה חדשה.            הפונקציה תיצור אלמנט חדש ותוסיף אותו למסך.            האלמנט מסוג <code>div</code> ומסגרת רציפה שחורה.            האלמנט יהיה בהתאם [רוחב אלמנט] [גובה אלמנט] [צבע רקע] [תוכן האלמנט]            מהמשימה הקודמת.            האלמנט יוכל את התוכן בהתאם ל[צבע גוף] [גודל גוף] שהתקבל</p>



# בונה מסכימים

צورو קובץ לטובת הפרויקט, תחילה מיצרת ממוק משמש.  
עבור כל משימה צרו מסמן חדש.

תרגיל	תיאור המשימה
1	<b>הוספת אפשרות להגדיר את עיצוב המסגרת האלמנט</b> [עובי מסגרת][סרגנון מסגרת][צבע מסגרת]
2	<b>הוספת אפשרות להגדיר את ריווח פנימי וריווח חיצוני לאלמנט</b> [ריווח פנימי][ריווח חיצוני]
3	<b>הוספת אפשרות להגדיר את פינות מעוגלות לאלמנט</b> [עיגול פינות לאלמנט]
4	<b>הוספת אפשרות להגדיר את סוג האלמנט</b> [סוג אלמנט] בחירה מבין האפשרויות מוגדרות: span,h1,h2,h3,p,div (פקד מסותר option )
5	<b>הוספת אפשרות להגדיר צל לאלמנט</b> [ציר X] [ציר Y] [צבע צל]



# בונה מסכימים

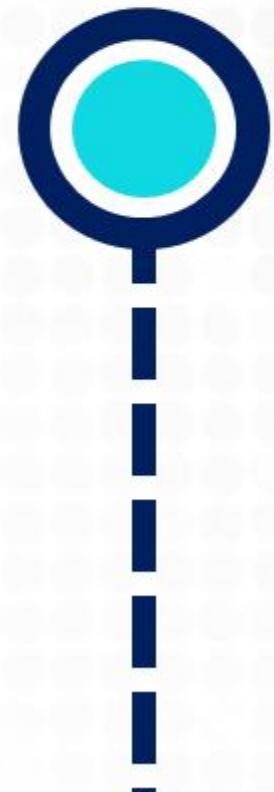
צورو קובץ לטובת הפרויקט, התחילה מיצירת ממתק המשמש.  
עבור כל משימה צרו מסמן חדש.

תרגיל	תיאור המשימה	מסמך
	<b>הוספת מאפיין <code>title</code> הכלול מידע על האלמנט.</b> אתגר: הוספת מועד מתן נוצר האלמנט.	Demo11
	<b>הוספת מאפיין <code>po</code> לאלמנט.</b> חשיבות: ערכו של מאפיין <code>po</code> חייב להיות ייחודי.	Demo12
	<b>הוספת אפשרות למחוק את כל הלוח</b> [עיגול פינוט לאלמנט]	Demo13
	<b>הוספת נפתח שמו (אתגר)</b> המסמך יישמר ב- <code>LocalStorage</code>	Demo14
	<b>הוספת שמירה אוטומטית (אתגר)</b>	Demo15



# User Management

משימת תרגול לניהול משתמשים





## ניהול משתמשים

במסגרת המשימה נבנה מערכת המדממת ניהול משתמשים. המערכת כוללת אפשרות רישום והתחברות של משתמשים לצד ממשק המאפשר לנהל את המשתמשים.



DOM

אובייקטים

מערכות

lolאות

תנאים

משתנים

# ניהול משתמשים

צור טופס הרשמה בהתאם לסקירה הבאה ובצע וידיצה לערבים מתקבלים.  
במידה והערבים תקינים צור אובייקט חדש וاحسن אותו במערך `users`, את המערך יש להדפיס בקונסולה.

הרשמה

לשםישם למשתמש יש לפחות ארבעה ספרות

שם פרטי שם משפחה שם פרטי שם משפחה

המשך



1

# ניהול משתמשים

צורו טבלה בהתאם לסקיצה הבאה והציגו את מערך `users` בטבלה.

The screenshot shows a user management application interface. At the top, there is a registration form titled "הרשמה" (Registration) with fields for "שם פרטי" (First Name), "שם משפחה" (Last Name), "טלפון" (Phone), "דוא" (Email), and "סיסמה" (Password). A "הרשמה" (Register) button is highlighted in blue. Below the registration form is a table titled "רשימת משתמשים" (User List) with columns: "שם פרטי" (First Name), "שם משפחה" (Last Name), "טלפון" (Phone), and "דוא" (Email). The table contains three rows of data:

שם פרטי	שם משפחה	טלפון	דוא
gal	levi	050123456	gal.levi@gmail.com
irriti	irriti	050123456	irriti@gmail.com
dina	dina	050123456	dina@gmail.com



.2

# ניהול משתמשים

הויספו כפטור המאפשר למחוק את המשתמש מהמערך `users` (ומהטבלה).

The screenshot shows a web-based user management system. At the top, there is a registration form titled "הרשמה" (Registration) with fields for First Name, Last Name, Date of Birth (in Hebrew), Gender, and Password. Below this is a table titled "רשימת משתמשים" (User List) with columns for ID, First Name, Last Name, Gender, and Password. Three users are listed:

ID	שם פרטי	שם משפחה	מין	סיסמה
1A123456	gal	levi@gmail.com	נữ	123456
1Q1W2E	snow	snow@gmail.com	זכר	123456
1S1S1SDana	dana	dana@gmail.com	נữ	123456



.3

# ניהול משתמשים

צורך טופס התמחבות בהתאם לסקירה הבאה ובעצם ולידציה לערבים מתקבלים.  
במידה והמנוי אינו מחובר שננו את סטטוס המשתמש למחובר והוא יופיע ברשימה המשתמשים אפשרות ניתוק.

The first screenshot shows the 'הרשמה' (Registration) page with fields for First Name, Last Name, Email, and Password, all marked as required. The second screenshot shows the 'התחברות' (Login) page with fields for Email and Password. The third screenshot shows the 'רשימת משתמשים' (User List) page displaying three users with columns for ID, First Name, Last Name, Email, and Status (Active/Inactive).

ID	שם פרטי	שם משפחה	EMAIL	סטטוס
1	ගֶל	אַרְשָׁיִל	gel.ash@gmail.com	פעיל
2	וִזְרָאֵל	אַרְשָׁיִל	izraiel@gmail.com	לא פעיל
3	דָנָה	אַרְשָׁיִל	dana@gmail.com	פעיל



4

# ניהול משתמשים

הוסיפו ברשימה המשתמשים כפטור המאפשר לעורך משתמש.  
(ניתן להציג את האפשרות לעדכנת משתמש בחלון נפרד, בתחום הטבלה או בתוך הטבלה)

The first screenshot shows a registration form titled "הרשמה" (Registration) with fields for First Name, Last Name, Email, and Password. A "הרשמה" button is at the bottom.

The second screenshot shows a connection form titled "התחברות" (Connection) with fields for First Name, Last Name, and Email. A "הרשמה" button is at the bottom.

The third screenshot shows a "רשימת משתמשים" (User List) page with a table containing three user entries:

ID	שם פרטי	שם משפחה	טלפון	EMAIL	סיסמה
1	gal	lin	0500000000	gal.lin@gmail.com	123456
2	israel	shani	0520000000	israel.shani@gmail.com	123456
3	dana	shani	0515151515	dana.shani@gmail.com	123456



