

# Advanced Javascript

מרצה: תומר שגיא

שיעור 9

26/11/2023

Callback

מבוא לתכנות אסינכרוני

# JavaScript Advanced

## הכרות עם תכנות אסינכרוני ונושאים מתקדמים

במסגרת הנושא נכיר ונעמיק ידע ב JavaScript ונלמד על אפשרויות מתקדמות ותכנות אסינכרוני הכולל קריאות לשרת.



שיעור 9

שיעור 10

שיעור 11

שיעור 12

✓ אופרטור Spread

✓ Shallow copy & Deep copy

✓ פונקציות חץ - arrow

✓ שימוש ב try & catch

✓ פונקציות Callback

✓ מבוא לתכנות א-סנכרוני

✓ קריאות XMLHttpRequest

✓ מחלקת Promise

✓ שימוש ב then - catch

✓ Async and Await

✓ Fetch

✓ Modules

2 שיעורים ספייר



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

# 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

firstName	"gal"
lastName	"lavi"

## getUser

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

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקראו לה <code>handleUserName</code> . פונקציה זו תדע לקבל 2 פרמטרים – <code>firstName</code> ו- <code>lastName</code> .
Ex-2	הפונקציה <code>handleUserName</code> תדע להגדיל את האות הראשונה של כל אחד מהפרמטרים, ותחזיר סטרינג אחד הכולל את השם + שם המשפחה עם רווח ביניהם. נסו לשלוח לפונקציה ערכים שונים והדפיסו את התוצאות לקונסול.
Ex-3	צרו פונקציה חדשה וקראו לה <code>getUser</code> . הפונקציה תבקש מהמשתמש להזין שם פרטי, ולאחר מכן תבקש שם משפחה. הציגו <code>alert</code> למשתמש אשר מברך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>handleUserName</code> .
Ex-4	בתוך הפונקציה <code>getUser</code> , צרו משתנה חדש בשם <code>quickHandleName</code> וכערך תנו לו פונקציית חץ אשר "תחקה" את אופן הפעולה של הפונקציה <code>handleUserName</code> .
Ex-5	הציגו <code>alert</code> למשתמש אשר מברך אותו לאחר טיפול בשם באמצעות שימוש בפונקציה <code>quickHandleName</code> .



# שאלות 1 – פונקציות חץ

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

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

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

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

שאלה 1: מה היתרון העיקרי שבשימוש בפונקציות-חץ ב JavaScript-בהשוואה לביטויי פונקציה רגילים?

שאלה 2: המר את הפונקציה הרגילה הבאה לפונקציה-חץ:

שאלה 3: כתוב מחדש את הפונקציה הבאה באמצעות פונקציה-חץ:

שאלה 3: כתוב מחדש את הפונקציה הבאה באמצעות פונקציה-חץ:  
javascript

שאלה 4: המר את הקוד הבא כך שישמש בפונקציה-חץ וגוף קצר:

שאלה 5: מה התחביר של פונקציה-חץ שאין לה פרמטרים?

# שאלות 2 – פונקציות חץ

- שאלה 1:
- כתוב 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:
- כתוב פונקציה באמצעות פונקציית חץ שבודקת אם מספר נתון הוא זוגי ומחזירה `true` אם כן ו-`false` אם הוא אי-זוגי.
- שאלה 4:
- צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של מחרוזות ומחזירה מערך חדש עם אורכי המחרוזות.
- שאלה 5:
- כתוב פונקציה באמצעות פונקציית חץ שמחשבת את הגורמים של מספר תמידי חיובי נתון.
- שאלה 6:
- צרו פונקציה באמצעות פונקציית חץ שמקבלת מערך של מספרים ומחזירה את סכום כל המספרים החיוביים במערך.
- שאלה 7:
- כתבו פונקציה באמצעות פונקציית חץ שמקבלת מחרוזת כקלט ומחזירה את המחרוזת ההפוכה.
- שאלה 8:
- צור פונקציה באמצעות פונקציית חץ שמקבלת מערך של שמות ומחזירה מערך חדש עם השמות שמתחילים באות "א".
- שאלה 9:
- כתבו פונקציה באמצעות פונקציית חץ שמחשבת את שטח המלבן על פי אורכו ורוחבו שניתנים 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 = _____ => `!_____, שלום`;
```

# שאלות השלם את החסר 2

תרגול 4:  
כתבו פונקציית חץ שבודקת אם מספר נתון הוא זוגי ומחזירה `true` אם כן ו-`false` אם אי-זוגי:

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

תרגול 5:  
המירו את הפונקציה הבאה לפונקציית חץ:

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

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

תרגול 6:  
כתבו פונקציית חץ שמקבלת מערך של מחרוזות ומחזירה מערך חדש עם אורכי המחרוזות:

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



# תרגול שימוש ב-try & catch

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

## divide

num1	50
num2	5

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



# 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"
};
```

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

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

**Program  
Variable**

**Memory**

age

→

5

age

→

```
{
  date_of_birth: "22-4-1988"
}
```

age

→

```
function() {
  return 'my age is 24';
}
```

*'age' points to a function without parameters*

age

→

```
function(n) {
  return 'my age is ' + n;
}
```

*'age' points to a function with parameters*

```
let age;
```

```
age = 5;
```

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

Program  
Variable

Memory

age

→

5

age

→

{

date\_of\_birth: "22-4-1988"

}

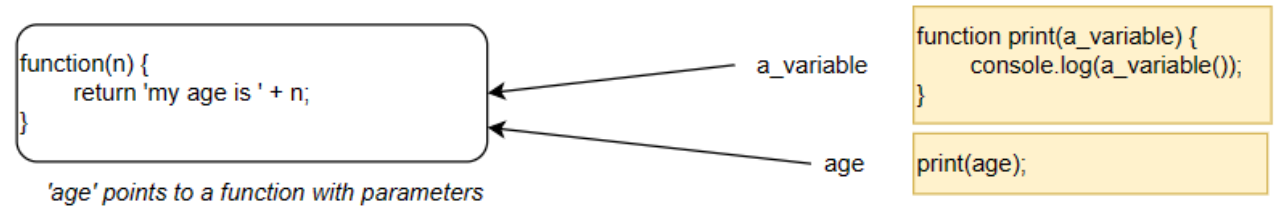
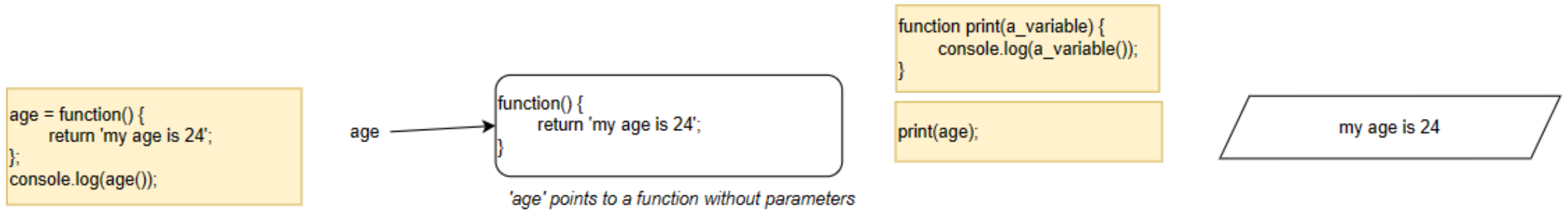
```
function print(a_variable) {  
  console.log(a_variable);  
}
```

```
print(age);
```

```
print(age);
```

5

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



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

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.

```
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);
}

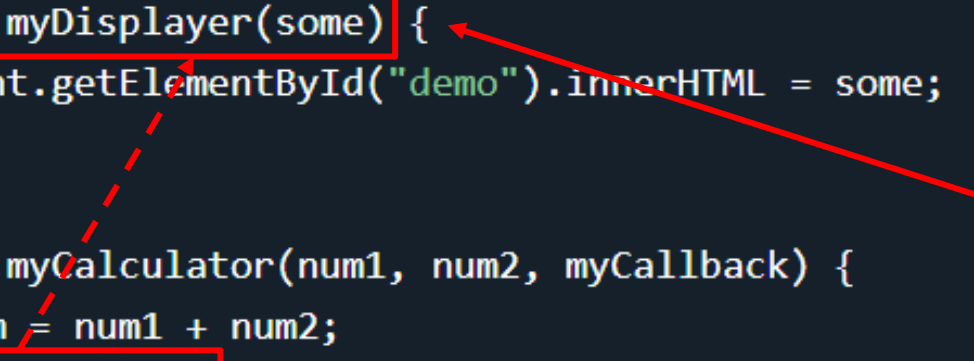
myCalculator(5, 5, myDisplayer);
```

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

1) Send the function's reference / pointer in memory

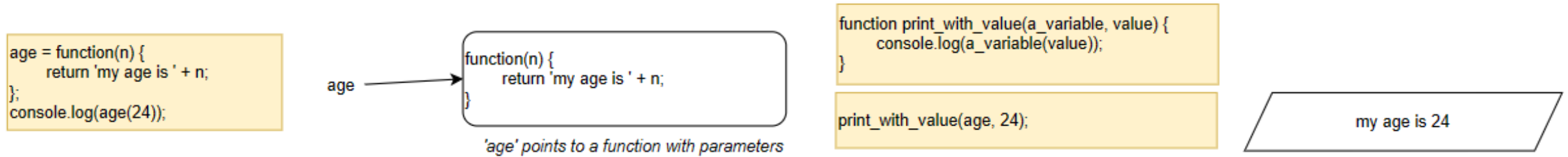
# Parameters in source function and callback function

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

A red dashed arrow points from the 'myCallback' parameter in the 'myCalculator' function to the 'myDisplayer' function definition. A solid red arrow points from the 'myCallback(sum)' call inside 'myCalculator' to the 'some' parameter in '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)



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>

```

# 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 (פונקציות שניתן לקרוא להן מאוחר יותר):
- אחת מדפיסה את השם שלך באמצעות הפונקציה `alert()`
- ואחת מדפיסה את השם שלך באמצעות הפונקציה `document.write()`
- צרו פונקציה 3 שתכלול:
- כפרמטר קלט, פונקציה קולבק (פשוט משתנה שמצביע לפונקציה)
- תציג את ההודעה "מציג את השם שלך" באמצעות הפונקציה `alert()`
- תקרא לפונקציה הקולבק
- קראו לפונקציה השלישית ומועבר אליה שם הפונקציה הראשונה (הפונקציה הראשונה של הקולבק)
- קראו לפונקציה השלישית ומועבר אליה שם הפונקציה השנייה (הפונקציה השנייה של הקולבק)
- אתם אמורים לראות את שמכם מוצג תחילה בתיבת `' , alert()` שם השם יופיע על העמוד.

# פונקציות Callback

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

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

```
function sayThankYou() {  
    alert("Thank You");  
    alert("Welcome back again");  
}
```

פרמטר myFunc שמיועד  
לשם פונקציה שתופעל

```
function totalPrice(price, products, myFunc) {  
    alert(price * products);  
    myFunc()  
}
```

הפעלת myFunc  
שהתקבל כפרמטר

```
totalPrice(10, 5, sayThankYou)
```

שם הפונקציה  
שתופעל בסיום הקוד



# פונקציות Callback

JS\_Advanced\_Callback\_Functions.html

פונקציית callback אנונימית  
שתופעל בעת הפעלת האירוע

```
// 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 שתופעל  
בעת הפעלת האירוע

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

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

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

```
<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>
```

# השלם את החסר 1

תרגול 1:  
צרו פונקציית קולבק שמכפילה שני מספרים ומחזירה את התוצאה. לאחר מכן, צרו פונקציה בשם 'calculate' שמקבלת את הפונקציה הזו כארגומנט ומשתמשת בה כדי לחשב את התוצאה של כפל 5 ו-7.

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

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

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 כארגומנט 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 החסר 2

תרגול 3:

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

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

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

calculateSum(_____);
```

תרגול 4:

צרו פונקציית קולבק שמקבלת מחרוזת ומחזירה את אורך המחרוזת. לאחר מכן, צרו 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(`length האורך הוא: ${}`);
};

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

num1	50
num2	5

## calc

num1	50
num2	5
calcFunc	calcFunc(num1,num2)

## go

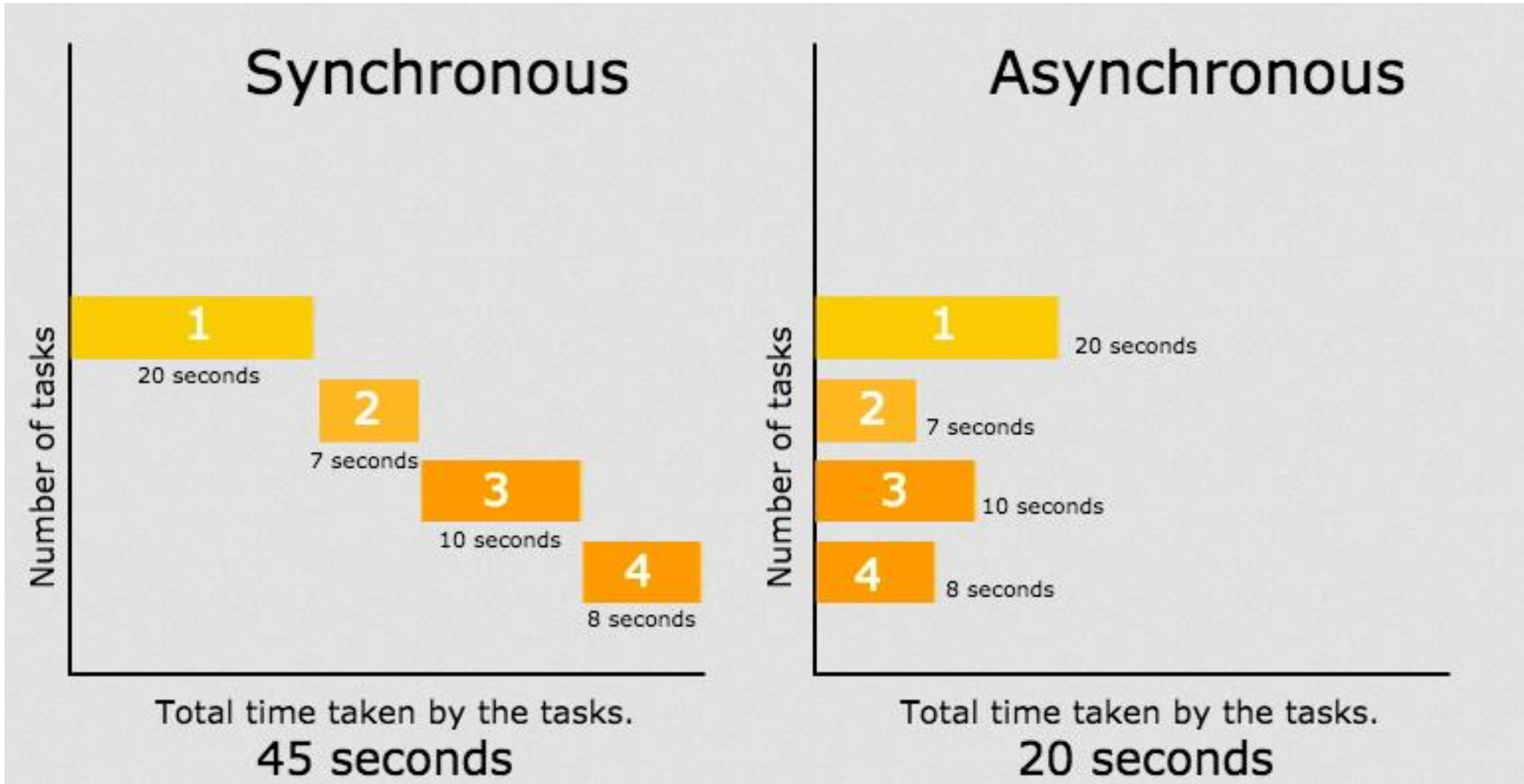
num1	Prompt
num2	Prompt
operator	Prompt

תרגיל	תיאור המשימה
Ex-1	צרו פונקציה חדשה וקראו לה <code>divide</code> . פונקציה זו תדע לקבל 2 פרמטרים - <code>num1</code> ו- <code>num2</code> ותדפיס בקונסולה את תוצאת החילוק של המספרים.
Ex-2	צרו פונקציה נוספת עבור כל פעולה מתמטית בהתאם לסעיף הקודם.
Ex-3	צרו פונקציה חדשה בשם <code>calc</code> . פונקציה זו תקבל כפרמטרים 2 מספרים ופונקציית <code>callback</code> בשם <code>calcFunc</code> . הפונקציה תשתמש בפונקציית ה- <code>callback</code> שנשלחה אליה כפרמטר על מנת לבצע את החישוב על המספרים.
Ex-4	צרו פונקציה חדשה בשם <code>go</code> . הפונקציה תבקש מהמשתמש 2 מספרים ואופרטור ותשמור 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 ה- <code>callback</code> המתאימה.
Ex-5	אתגר: השתמשו בפונקציות חץ אנונימיות במקום להשתמש בפונקציות החישוב שכתבנו בהתחלה.

# מבוא לתכנות א-סנכרוני







# 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 לתכנות א-סנכרוני

JS\_Advanced\_Asynchronous1.html

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

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

```
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..");
```

קריאה לשרת עשויה לקחת  
זמן, אך הקוד ממשיך לרוץ

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

# מבוא לתכנות א-סנכרוני

JS\_Advanced\_Asynchronous2.html

```
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 לקחת זמן,  
אך הקוד ממשיך לרוץ

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

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

2
3
4
1