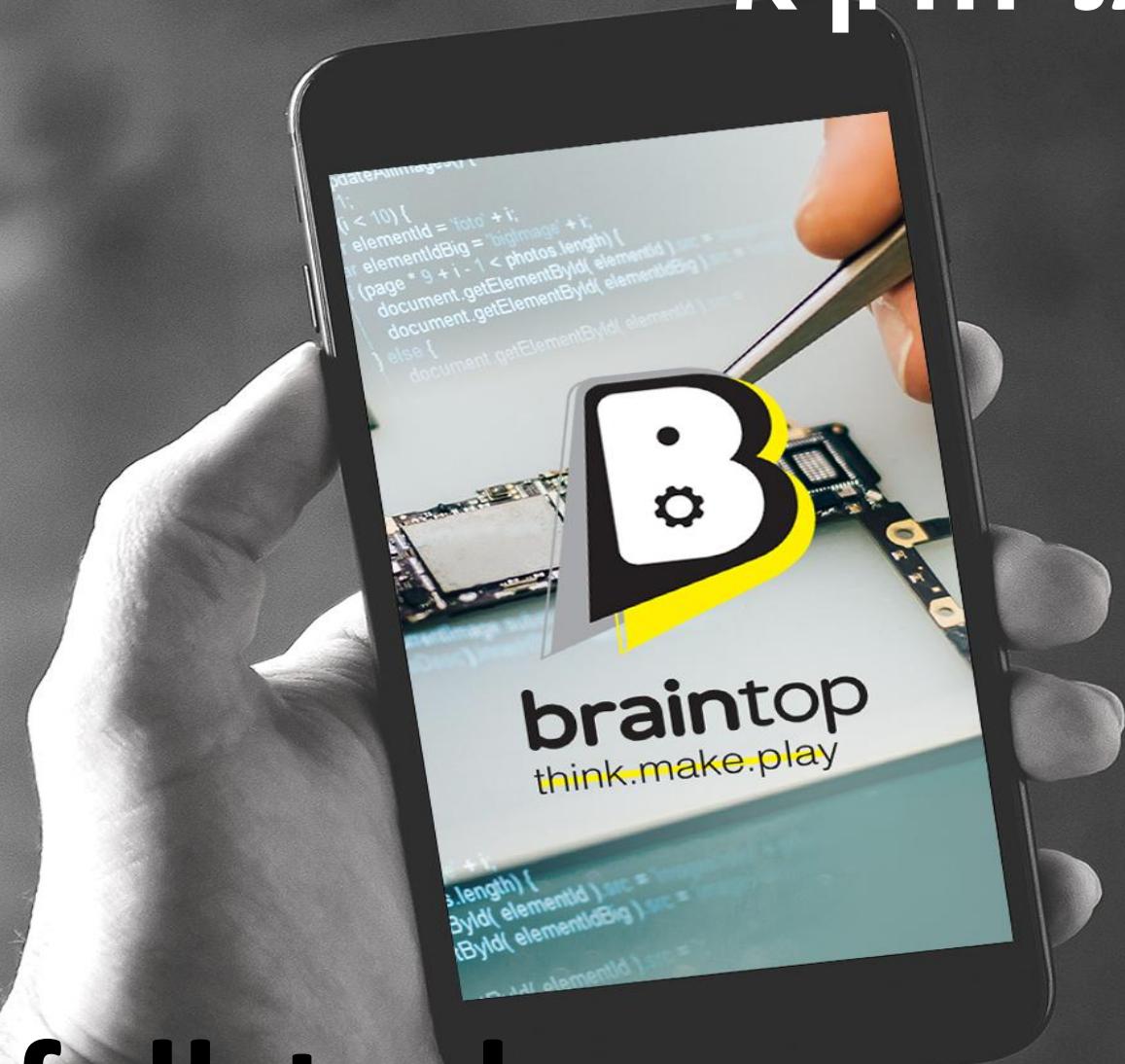


ספר חלק א' JAVASCRIPT



What is fullstack

JAVASCRIPT

תוכן עניינים

3	פרק 1. מה זה fullstack
9	פרק 2. פלט (output)
40	פרק 3. פלט(input)
53	פרק 4. שפט תנאי
82	פרק 5. לולאות for
98	פרק 6. לולאות while
105	פרק 7. ההבדל בין let ו var
114	פרק 8. ערכים
150	פרק 9. פונקציות
175	פרק 10. אובייקטים
191	פרק 11. DOM
212	פרק 12 אירועים (event)
252	פרק 13 (destructuring)

:

מה זה full stack ?

Fullstack מתייחס לפיתוח הרכיבים הקדמיים (צד הלוקח) והן האחוריים (צד השרת) של יישום אינטרנט. מפתח Fullstack מיומן בעבודה הן עם טכנולוגיות החזית, כגון HTML, CSS, JavaScript, וכן גם עם טכנולוגיות הקצה האחורי, כגון שפות תכנות הצד השרת (למשל, Python, Ruby, Node.js) ובסיסי נתונים (למשל, MySQL, MongoDB).

لمפתחי Fullstack יש את הCESSORIM והידע לטפל בכל היבטים של פיתוח אפליקציות אינטרנט, כולל עיצוב מסך משתמש, הטענת פונקציונליות הצד הלוקח, יצירת לוגיקה הצד השרת, אינטראקטיה עם בסיס נתונים ופרישה של האפליקציה לייצור.

על ידי יכולת לעבוד הן על החזית והן על הקצה האחורי, מפתח Fullstack יכולים להתמודד עם משימות פיתוח מקצה לקצה, לשתף פעולה עם צוותים שונים ולהיות בעלי הבנה הוליסטית של ארכיטקטורת היישומים כולה. זה מאפשר להם לבנות יישומי אינטרנט ניתנים להרחבה ועתיריו תכונות שיוכולים להתמודד עם אינטראקטיות של משתמשים, אחסון נתונים ותקשורת עם שירותי חיצוניים.

מה ההבדל בין גישת react + שפת שרת כלשוי ?

הבדלים העיקריים בין גישת ASP.NET לגישת שפת React + שרת הם כדלקמן:

גישת ASP.NET:

- .1. שפה: ASP.NET משתמש בדרך כלל ב-C# כשפה העיקרית עבור פיתוח הקצה האחורי והחזית.
- .2. ארכיטקטורה **מונוליטית**: יישומי ASP.NET נבנים לעיתים קרובות באמצעות ארכיטקטורת מונוליטית, שבה הלוגיקה הקדמית והחלק האחורי מחוברים הדוק בטור אותו בסיס קוד.
- .3. עיבוד בצד השרת: יישומי ASP.NET עושים בדרך כלל מעבדים את ה-HTML הראשוני בשרת ושולחים אותו ללקוח. אינטראקציות ועדכונים עוקבים מטופלים באמצעות חזרות בצד השרת או קריאות AJAX.
- .4. דפוא ASP.NET MVC: עוקב אחר דפוא ה-(MVC) Model-View-Controller, שעזר להפריד חששות ולארגן קוד לרכיבים נפרדים.
- .5. סביבת פיתוח משולבת: הוא ה-IDE העיקרי לפיתוח ASP.NET, המספק סט מקיף של כלים ותוכנות.

מה ההבדל בין גישת react + שפת שרת כלשהו ?

הבדלים העיקריים בין גישת ASP.NET לגישת שפת React + שרת הם כדלקמן:

גישת React + Server Language

- .1. **גמישות שפה:** בגישה שפת React + שרת, יש לך את היכולות לבחור שפות שונות עבור ה-backend (למשל, C, Python, Ruby, C# וко') בהתאם על העדפות שלך ודרישותuproject.
- .2. **משחק משתמש מבוסס רכיבים:** React מתקדם במבנה משחקי משתמש באמצעות ארכיטקטורה מבוססת רכיבים, המאפשרת רכיבי משחק משתמש מודולריים וניתנים לשימוש חוזר.
- .3. **ישומי עמוד בודד:** לעיתים קרובות משתמשים ב-React לבניית "ישומי עמוד בודד" (SPAs), כאשר ה-HTML הראשון נתון פעם אחת, ואינטראקציות עוקבות מטופלות מצד לקוח באמצעות JavaScript.
- .4. **RESTful APIs:** בגישה זו, ה-(React) frontend מתקשך עם ה-backend (שפת השרת) באמצעות משקי API של RESTful, מה שמאפשר ארכיטקטורה **מנוטקמת** ומאפשר הפרדה טוביה יותר של חששות.
- .5. **מערכת אקוולוגית עשירה:** ל-React יש מערכת אקוולוגית גדולה ופעילה עם מגוון רחב של ספריות, כלים ותמייה קהילתית, המאפשרת למפתחים למן פתרונות מובנים מראש ולשפר את פרודוקטיביות הפיתוח.

מה ההבדל בין גישת react + שפת שרת כלשהו ?

ლסיאם

לשתי הגישות יש את החזקות שלהן והן מתאימות לתרחישים שונים. גישת ASP.NET מזינה מסגרת מקיפה וכלים לבניית יישומי אינטרנט תור שימוש ב-C# על פני ה-frontend וה-backend. מצד שני, גישת שפת השרת React + מספקת גמישות בבחירה השפה, מתמקדת בפיתוח ממש משתמש מבוסס רכיבים וחובקת ארכיטקטורה מנוקחת ומודולרית יותר. הבחירה בין שתי הגישות תלולה בגורמים כמו דרישותuproject, מומחיות הצוות, שיקולי ביצועים והעדפות אישיות.

בקורס זה נלמד את גישת + שפת שרת

הקורס מחלק ל 4 חלקים

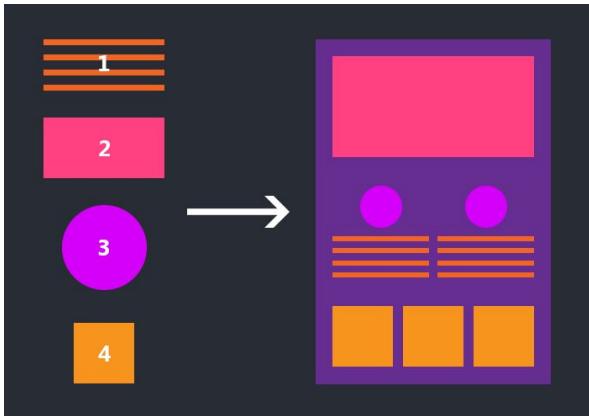
html, css, js	.1
צד שרת (גמיש - ניתן בעתיד לבחור שפת שרת אחרת)	.2
צד לקוח react	.3
פרויקט מסכם	.4

יתרון 1 של nodejs על שאר השפות ? ב nodejs אתה כותה ב js ואין צורך ללמוד שפה נוספת.

עבור סעיף 2 ניתן לבחור במקום nodejs בצד השרת :

- python ,ruby , java, c# ,

דוגמא לארכיטקטורת קומפוננטות ב-React



קומפוננטה היא אחד המרכיבים הבסיסיים של React.

ב-React, אתה מפתח את האפליקציות שלך על ידי יצרת קומפוננטות (חלקים) לשימוש חוזר שאתה יכול להשוב עליהם כקביות לגו עצמאיות.

בחינה מושגית, קומפוננטות הם כמו פונקציות JavaScript (הנקרא "props") ומחזירים רכיבי React שמתארים את מה שצሪיך להופיע על המסך.

אתה יכול לחלק ממשק משתמש לקומפוננטות ולעבוד עליהם לרכיב אב, שייהיה ממשק המשמש הסופי שלך.

זכור את חוק הקומפוננטה - לכל רכיב תפקיד אחד!
אם יש לו כמה, אז צריך להפריד אותו לקומפוננטות נוספות!

Famous websites built with React

facebook



PayPal

ATLASSIAN

Dropbox



NETFLIX



output

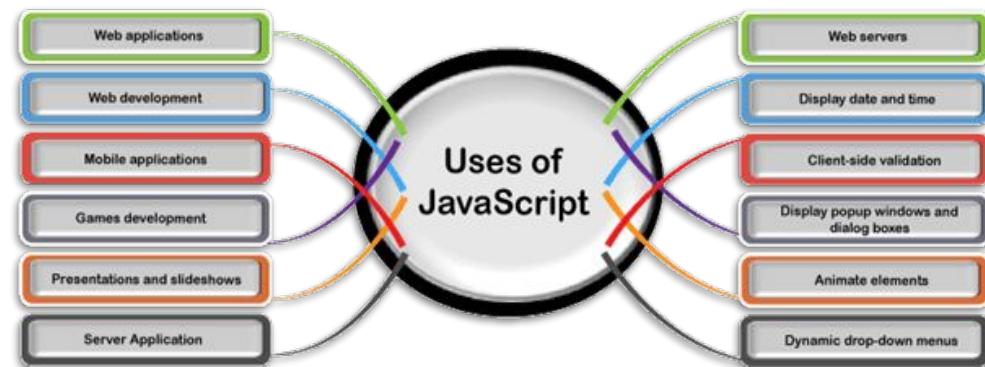


JAVASCRIPT

אודות JavaScript

מהו JavaScript ומדוע אנו משתמשים בו?

- JavaScript היא שפת תכנות המשמשת הן בצד הלקוח והן בצד השרת המאפשרת לרך להפוך את דפי האינטרנט לאינטראקטיביים.
- כאשר HTML ו-CSS הם שפות שנוטנות מבנה וסגנון לדפי אינטרנט, JavaScript נותן לדפי אינטרנט אלמנטים אינטראקטיביים המסייעים המשתמש. לsicום, JavaScript מוסיף התנהגות לדפי אינטרנט.
- Javascript היא שפת התכנות הפופולרית ביותר בעולם.
- JavaScript היא שפת התכנות היחידה שמקורה בדף האינטרנט.



הויסף JavaScript לקוד שלך

```
<html>  
  <body>  
    <h1>The script element</h1>  
    <script>  
    </script>  
  </body>  
</html>
```

איך משתמשים ב-JavaScript ואיפה?

אנו יכולים להכניס קוד JS ב-2 דרכים:

1. הוסיף בתוך `<body>` באמצעות תג `<script>`.

2. צור קובץ JS נפרד וחבר אותו.

3. שימוש ב-<script>

פתח קובץ HTML והוסיף את התג בתוך הגוף:

קובץ נפרד .1

עלינו להציג על קובץ JavaScript חיצוני באמצעות התוכנה `src`.

תג הסקריפט יראה כך:

```
<script src="myscripts.js"></script>
```

התוכנה `src` מציינה על קובץ URLஇதோ (like `src="/scripts/example.js"`)

12

פלט -
Output -

Output - console.log()

מה זה `console.log()` ?

השיטה `log()` מדפסה הודעה למסוף האינטרנט. ההודעה עשויה להיות כל מה שציר להציג למשתמש.

אייה ולמה אנחנו משתמשים בו?

השיטה `log()` שימושית למטרות בדיקה. זה נקרא "ניפוי באגים". למשל מספר שאנו מחשבים או הودעת בדיקה כדי לבדוק אם אנחנו מدلגים על קוד שורה.

air `console.log()` ?

```
<script>
    console.log("hello world");
</script>
```

Bonus - Run this code:

```
alert("I am alert")
```

היכן אנו רואים את ה-`log()` ?

השיטה מדפסה כל דבר בקונסולת הדפדפן. חפש כלים למפתחים או פשוט תפריט כלים בדף שלך.
הקש F12 כדי לפתוח אותו ולחץ על "קונסול" כדי לראות את הפלט.

עשה זאת בעצמך 1

לתרגול השלימו את התרגילים הבאים.

- 1.) כתוב תוכנית שמציגה את שمر בקונסולה.
- 2.) כתוב תוכנית שמרתת את הכתובת שלר בקונסולה.
- 3.) כתוב תוכנית שמרתת את שמר על תגית של פסקה עם `id=details`.
- 4.) כתוב תוכנית שמרתת את הכתובת שלר על תגית של פסקה עם `id=details`.

Document.getElementById

הגדרה ושימוש:

השיטה `getElementById()`מחזירה רכיב עם ערך מוגדר. קיבל את האלמנט עם המזהה שצויין.

השיטה `null` (`getElementById()`)מחזירה אם האלמנט אינו קיימ.

השיטה `Id` (`getElementById()`) היא אחת השיטות הנפוצות ביותר ב-DOM. הוא משמש כמעט בכל פעם שאתה רוצה לקרוא או לערוך אלמנט HTML.

הערה: כל מזהה צריך להיות ייחודי, אבל אם קיימים שני אלמנטים או יותר עם אותו מזהה, `getElementById()`מחזירה את הראשון.
דוגמה עם סגנון:

```
<html>
  <body>
    <h1 id="demo">The Document Object</h1>
    <h2>The getElementById() Method</h2>
    <script>
      document.getElementById("demo").style.color = "red";
    </script>
  </body>
</html>
```

The Document Object

The `getElementById()` Method

Document.getElementById

דוגמה עם :innerHTML

```
<html>
  <body>
    <h2>The getElementById() Method</h2>
    <p id="demo"></p>
    <script>
      document.getElementById("demo").innerHTML = "Hello World";
    </script>
  </body>
</html>
```

The getElementById() Method

Hello World

פלט סיכון

עד כה ראיינו 3 אפשרויות פלט

.1 - פלט לკונסולה - `console.log`

.2 - התראה - `alert`

.3 - פלט לפסקה או `div` – `document.getElementById('').innerHTML`
ספציפי מזהה

משתנים

משתנה הוא מיכל שמכיל את כל הנתונים שנזקן. אי אפשר ליצור משתנה שמתחל במספרים. ברירת המחדל של המשתנה אינה מוגדרת.
ישנן 4 דרכי להכריז על משתנה JavaScript:

1. `var`
2. `let`
3. `const`
4. `nothing`

דוגמאות לסוגי הנתונים הנפוצים ביותר:

- String - text values
- Number - numeric values
- Boolean - true/false
- Null - no value

Data Types

ישנם מספר סוגים של נתונים כולל :

- 1.) Numeric / floating point numbers - `var age = 100`
- 2.) String - sequence of characters used for text - `var name = "Asaf Amir"`
- 3.) Boolean - logical type that can be true or false - `var open = true; true , false`
- 4.) Null- empty value, no value - Null- ערך ריק, ללא ערך
- 5.) Undefined - ערך שנלקח על ידי משתנה שעדין לא הוגדר - `var fullname;`
- 6.) Symbol(ES2015) - ערך ייחודי ואינו ניתן לשינוי
- 7.) BigInt(ES2020) - מספרשלם גדול יותר ;

של דינמי - אין צורך להגדיר ידנית את סוג הנתונים

הצהרות משתנים

מתי להשתמש ב-var

הצהר תמיד על משתני JavaScript עם `const`, `var`, או `let`.

מילת המפתח `var` משמשת בכל קוד JavaScript מ-1995 עד 2015. מילות המפתח `let` ו-`const` נוספו ל-JavaScript ב-2015.

אם אתה רוצה שהקוד שלך יפעל בדף ישן יותר, עליך להשתמש ב-`var`.

מתי להשתמש ב-const

אם אתה רוצה כלל כללי: תמיד הcriiz על משתנים עם `const`.

אם אתה חושב שהערך של המשתנה יכול לשינויו, השתמש ב-`let`.

למשתנים המוצזרים עם מילת המפתח `const` יש ערכים קבועים ואינם ניתנים לשינוי.

דוגמאות Var and Let

```
var salary;
var x = 10;
var y = 5;
var name="Daniel";

var a;
var b;
a = 5;
b = 4;

var sum;
sum = a + b; //9
```

שם חוקי למשתנים

שם משתנה יכול להתחיל באות או בקוו תחתון

- Good: firstname,_age
- Bad: 100num

שם משתנה יכול להכיל אותיות, מספרים וקווים תחתוניים

- Good: age_12
- Bad: age*12

כשאנחנו מcriזים על המספר חשוב לתת לו שם עם משמעות כדי שנזכור מה מסמל המשתנה.

Variable - JavaScript in HTML

```
<html>
```

```
<body>
```

```
    <h1>hello javascript</h1>
```

```
    <script>
```

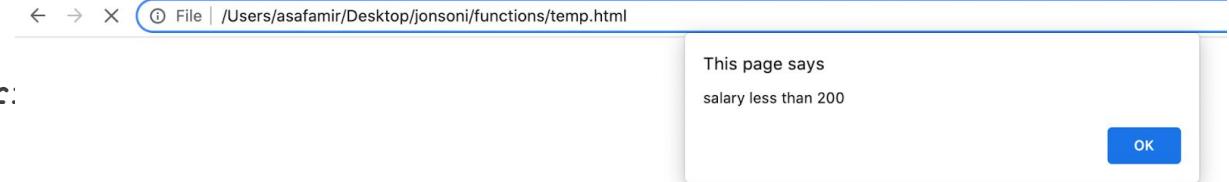
```
        var salary=100
```

```
        alert("salary is "+salary)
```

```
    </script>
```

```
</body>
```

```
</html>
```



Variable - JavaScript in HTML

```
<html>
  <head>
    <title>
    </title>
    <script>
      var salary=100;
      alert(salary);
    </script>
  </head>
  <body>
    <h1>hello javascript</h1>
  </body>
</html>
```

דוגמאות פלט - המשך

11

```
<html>
<body>
    <h1>hello javascript</h1>
    <p id="output"></p>
    <script>
        var a = 5;
        var b = 6;
        var sum = a+b;      document.getElementById("output").innerText=sum;
    </script>
</body>
</html>
```

זרקה - injection

Numeric

```
<html>
  <body>
    <h1>hello java script</h1>
    <script>
      var integer = 3.14;
      var bank = -1;
      alert("bank=" + bank + " integer=" + integer)
    </script>
  </body>
</html>
```



String

```
<html>
  <body>
    <script>
      var str = "My name is Asaf"
      var address = "I live in Holon city";
      var quote1 = "I think, he is \"beautiful\"."
      var quote2 = 'I think, he is "beautiful".'
      alert(str)
      alert(address)
      alert(quote1)
      alert(quote2)
    </script>
  </body>
</html>
```

Boolean

```
<html>
  <body>
    <script>

     isOk = true;

      isRun = false;
      alert(
      alert(
        isRun
      )
    )
  </script>
  </body>
</html>
```



Typeof

```
<html>
<body>
    <h1>hello alert</h1>
    <script>
        var name = "oren";
        var t = typeof(name)
        console.log(t); //string
    </script>
</body>
</html>
```

Operators -

אופרטורים מאפשרים לנו לבצע פונקציות מתמטיות על המשתנים שלנו.

Examples:

```
total=num+3;  
total=total-5;  
average=sum/5;  
Sum=[(sum+8)*x+9]*n/2;  
x=x%5 // We will get the remainder of dividing by 5
```

Operator	Description
+	חיבור - Addition
-	חיסור - Subtraction
*	מכפלה - Multiplication
**	Exponentiation (ES2016)
/	חילוק - Division
%	Modulus (Division Remainder)
++	הגדלה באחד - Increment
--	הקטנה באחד - Decrement

קיצור דרר ב-javascript

קיצור דרר

total+=3

total-=3

total*=3

total=total / 3

רגיל

total=total + 3

total=total - 3

total=total * 3

total=total / 3

קיצור דרר ב-javascript

קיצור

++

--

rangle

total++

total--

דוגמאות לקיורי דרר

```
var x = 5
```

```
X++; //now x is 6
```

```
var y = 100
```

```
y--;//now y is 99
```

תרגול

מה התשובה ?

```
<html>
  <body>
    <h1>hello alert</h1>
    <script>
      var a = 5;
      var b = "4";
      var s = a + b;
      console.log(s);    ???????????
    </script>
  </body>
</html>
```

תרגול - המשך

```
<html>
  <body>
    <h1>hello alert</h1>
    <script>
      var a = 5;
      var b = "4";
      var s = a + b;
      alert(s); 54
    </script>
  </body>
</html>
```



תרגול - המשך

```
<html>
  <body>
    <h1>hello alert</h1>
    <script>
      var a = 5;
      var b = parseInt("4");
      var s = a + b;
      alert(s); //9
    </script>
  </body>
</html>
</html>
```



שרשור מחרוזת

```
<html>
  <body>
    <h1>hello alert</h1>
    <script>
      var age = 8;
      var sentence = "I am " + age + " years old";
      alert(sentence);
    </script>
  </body>
</html>
```



תרגול משתנים ואופרטור

- .1 צור משתנה מספרי בשם "age" עם הערך 11;
- .2 הגדל את הערך של "age" ב-1 (השתמש בקיצור דרכו).
- .3 צור משתנה מחרוזת בשם "name" עם השם שלו;
- .4 הדפס את סוג המשתנה "name".
- .5 הכפל את הערך של "age" ב-2 (השתמש בקיצור דרכו). לאחר מכן הדפס את הערך.
- .6 חלקו את הערך של "age" ב-3. לאחר מכן הדפס את הערך.

פתרון תרגול משתנים ואופרטור

```
1 <html>
2   <body>
3     <script>
4       var age =11;
5       age++;
6       var name = "Daniel";
7       console.log(typeof(name));
8       age*=2;
9       console.log(age);
10      age/=3;
11      console.log(age);
12    </script>
13  </body>
14 </html>
```

string	a.html:7
24	a.html:9
8	a.html:11
>	

INPUT

JAVASCRIPT



Functions

מהי פונקציה?

פונקציה היא גוש קוד שנועד לבצע משימה מסוימת.

מבצעת כאשר "משהו" מפעיל אותה (קורא לה) JavaScript פונקציית

התחבר ב-javascript :

- פונקציית JavaScript מוגדרת עם מילת המפתח **function**, ולאחריה שם, ואחריו סוגרים () .

שמות פונקציות יכולים להכיל אותיות, ספורות, קווים תחתיונים וסימני דולר (אותם כללים כמו משתנים).

הסוגרים עשויים לכלול שמות פרמטרים מופרדים בפסיקים: (פרמטר1, פרמטר2,...)

הקוד שיבוצע, על ידי הפונקציה, ממוקם בתחום סוגרים מסולסלים: {}

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

Note: Inside the function, the arguments (the parameters) behave as local variables.

Return:

כאשר JavaScript מגיע למשפט `return`, הפונקציה תפסיק לפעול.

פונקציות לרוב מחשבות ערך הוצאה. ערך הוצאה "מוחזר" בחרזה ל"מתקשר".

תסתכל על הדוגמה:

```
let x = myFunction(4, 3); // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;           // Function returns the product of a and b => now x=12
}
```

למה פונקציות?

כדי לעשות שימוש חוזר בקוד: הגדר את הקוד פעם אחת, והשתמש בו פעמים רבות.

³ זה אומר להשתמש באותו קוד פעמים רבות עם ארגומנטים שונים, כדי לייצר תוצאות שונות.

Input

קבלת ערך למשתנה מהמשתמש.

ה נבחן 2 דרכיים לקבל קלט מהמשתמש:

1. using input type = text
2. prompt

Input type="text "

ה-tag <input> מצין שדה קלט שבו המשתמש יכול להזין נתונים. אלמנט זה משמש עבור טפסי אינטרנט.

בדוגמה הבאה אנו משתמשים ב-`event` onclick(). אירע ה-`event` מתרחש כאשר השימוש לחץ על אלמנט.

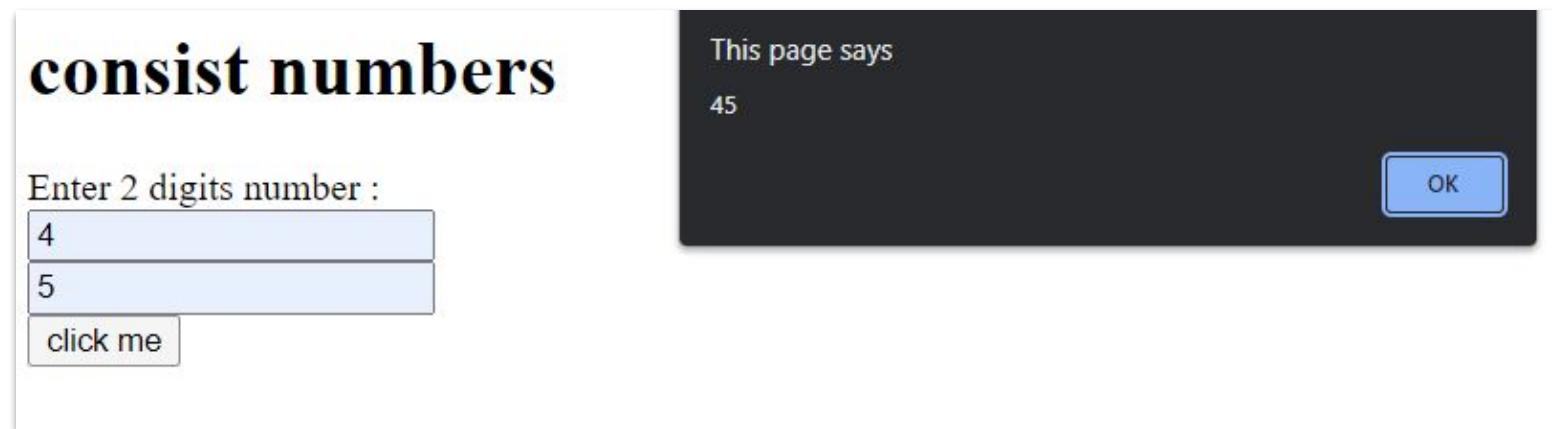
בדוגמה הבאה אנו משתמשים ב-`document.getElementById.value` כדי לקבל את הערך מקלט.

```
<html>
<head>
<title>if else</title>
</head>
<body>
    <h1>Input</h1>
    Enter 2 digits number : <br>
    <input type="text" id="num1" placeholder="print number"><br />
    <button onclick="sum()">click me</button>
</body>
<script>
    function sum() {
        var num1=document.getElementById("num1").value;
        alert(num1);
    }
</script>
</html>
```



Input type="text " exercise

צור עמוד בעל 2 כניסה למספרים. לחיצה על כפתור מתריעה על מספר חדש המורכב מהמספרות שהוזנו. לדוגמה:



Input type="text " exercise solution

צור עמוד בעל 2 כניסה למספרים. לחיצה על כפתור מתריעה על מספר חדש המורכב מהמספרות שהוזנו.

```
<html>
<body>
    <h1>consist numbers </h1>
    Enter 2 digits number : <br>
    <input type="text" id="num1" placeholder="print number"><br />
    <input type="text" id="num2" placeholder="print number"><br />
    <button onclick="sum()">click me</button>
    <script>
        function sum(){
            var num1=document.getElementById("num1").value;
            var num2=document.getElementById("num2").value;
            var sum = num1 + num2
            alert(sum);
        }
    </script>
</body>
</html>
```



Input type— Continued

שmeno לב שחיבור בין 2 מספרים ממירה אותם למחזוזות (2+1=12). כדי לתקן את זה אנו משתמשים ב- "parseInt".

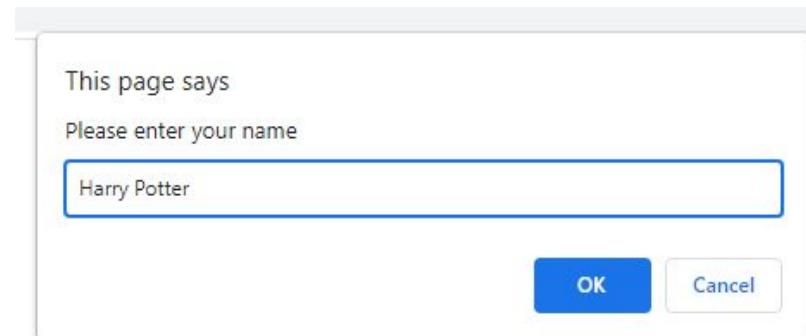
```
<html>
  <body>
    <h1>Input </h1>
    Enter 2 digits number : <br>
    <input type="text" id="num1" placeholder="First Number"><br />
    <input type="text" id="num2" placeholder="Second Number"><br />
    <button onclick="sum()">click me</button>
    <script>
      function sum() {
        var num1=document.getElementById("num1").value;
        var num2=document.getElementById("num2").value;
        var sum = parseInt(num1) + parseInt(num2);
        alert(sum);
      }
    </script>
  </body>
</html>
```

parseInt converts
(casting)string to number.

Prompt

השיטה `prompt()` מציגה תיבת דיו-שיכון המבקשת מהמשתמש קלט.

השיטה `prompt()` מחזירה את ערך הקלט אם המשתמש לוחץ על "אישור", אחרת היאמחזירה `null`.



Example

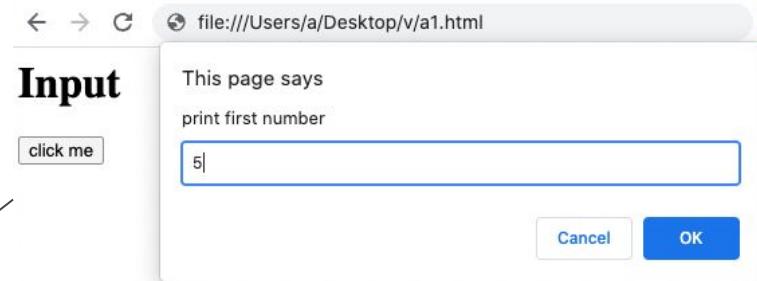
לתרגול יש לבצע את התרגיל הבא:

- 1.) כתוב תוכנית שבוחרת מספרים מהמשתמש ומציגה את הנקודות שלהם באמצעות הטראה.

Example solution

```
<html>
  <head>
    <title>Input</title>
  </head>
  <body>
    <h1>Input </h1>
    <button onclick="sum()">click me</button>
  </body>
  <script>
    function sum(){
      var num1 = prompt("print first number");
      var num2 = prompt("print second number");
      var num3 = prompt("print third number");
      var sum = parseInt(num1) + parseInt(num2) + parseInt(num3);
      alert(sum);
    }
  </script>
</html>
```

parseInt converts
(casting)string to
number.



קלט סיכום

- קבלת ערך למשתנה מהמשתמש

זאת למדנו 2 דרכים לקבל קלט מהמשתמש.

1. prompt
2. using input type = text

Do it yourself 2-7 (go to the LMS)

IF ELSE

JAVASCRIPT



if, else, and else if

לעתים קרובות מאד כאשר אתה כותב קוד, אתה רוצה לבצע פעולות שונות עבור החלטות שונות.

אתה יכול להשתמש בהצהרות מותניות בקוד שלך כדי לעשות זאת.

ב-JavaScript יש לנו את ההצהרות המותניות הבאות:

- **if** - השימוש כדי לציין גוש קוד לביצוע, אם תנאי שצוין נכון
- **else** - השימוש כדי לציין גוש קוד לביצוע, אם אותו תנאי הוא שקר
- **else if** - השימוש כדי לציין תנאי חדש לבדיקה, אם התנאי הראשון הוא שקר
- **switch** - השימוש כדי לציין בלוקים חלופיים רבים של קוד לביצוע - (בונוס: קרא על זה בבית).

•

נלמד על שלושת ההיגדים המותנים הראשונים.

if statement

הצהרה if מעריכה את התנאי בתוך סוגרים () .

.1 אם התנאי מוערך כ-true, הקוד בתוך הגוף של if מבוצע.

.2 אם התנאי מוערך כ-false, הקוד בגוף ה-if ידלג.

הערה: הקוד בתוך {} הוא הגוף של הirection if.

Syntax:

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

Example:

```
var age = 20;  
if( age > 18 ) {  
    alert("Qualifies for driving");  
}
```

Condition is true

```
let number = 2;  
if (number > 0) {  
    // code  
}  
//code after if
```

Condition is false

```
let number = -2;  
if (number > 0) {  
    // code  
}  
//code after if
```

else statement

להצחת if יכול להיות סעיף optional.

Syntax:

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

Example:

```
var age = 20;  
if( age > 18 ) {  
    alert("Qualifies for driving");  
} else {  
    alert("Is not qualified for driving");  
}
```

Condition is true

```
let number = 2;  
if (number > 0) {  
    // code  
}  
else {  
    // code  
}  
// code after if
```

Condition is false

```
let number = -2;  
if (number > 0) {  
    // code  
}  
else {  
    // code  
}  
// code after if
```

else if statement

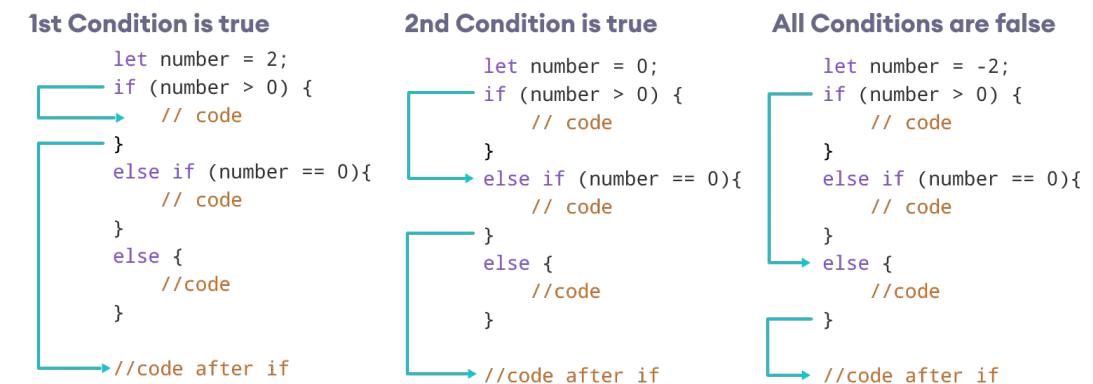
ההצהרה `else if` משמשת לביצוע בלוק קוד בין שתי חלופות. עם זאת, אם אתה צריך לבחור בין יותר משתי חלופות, אם...אחר אם...אפשר להשתמש.

Syntax:

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

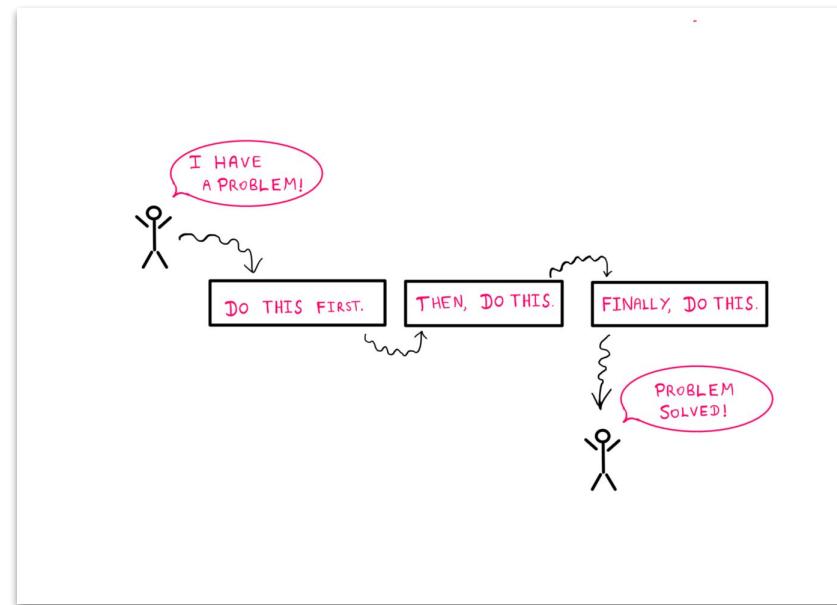
Example:

```
var age = 16;  
  
if( age > 18 ) {  
    alert("Qualifies for driving");  
} else if(age > 15) {  
    alert("Can take driving lessons");  
} else {  
    alert("Is not qualified for driving");  
}
```



Finding a Minimum Between Two Numbers

אלגוריתם- הפתרון לכל בעיה דורש פעולות בסדר מסוים.
פתרון שמודדר על ידי אותו פעולות נקרא אלגוריתם.



האלגוריתם לפתרון:

קלט: num1,num2

בדוק: אם $num1 < num2$ מאשר פלט: num2

בדוק: אחרת אם $num2 < num1$ מאשר פלט: num1

אחר מאשר פלט: שווה

תרגיל: פתרו את זה בעצמכם!

Finding a Minimum – Solution

```
<html>
  <body>
    <h1>if else</h1>
    </body>
    <script>
      var x=300;
      var y=100;
      if(x<y) {
        alert("min is " + x);
      }
      else if(y<x) {
        alert("min is " + y);
      }
      else {
        alert("They are equals")
      }
    </script>
  </html>
```

Logical Operators

To build complex logical expressions there are several logical operators:

- 1.) AND **&&**
- 2.) OR **||**
- 3.) NOT **!**

Boolean Phrase

ביטוי בוליאני "שואל שאלה" שההתשובה עליה היא "כן" או "לא". זה יכול לבוא גם בצורה של שאלה אמיתיית או לא נכוןה.

כמה דוגמאות:

האם הערך של x גדול מערךו של y?
האם שתי מחרוזות טקסט שוות בעיצובן?

Boolean Example

```
<html>
  <body>
    <h1>Hello Boolean</h1>
    <script>
      var a = true;
      if (a==true) {
        }
      if (a) {
        }
    </script>
  </body>
</html>
```

Truth Table

x	y	x&&y	x y	!x
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

בדיקה קלט באמצעות IF

```
<html>
<body>
    <h1>if else </h1>
    Enter 2 digits number : <br>
    <input type="text" id="num1" placeholder="print number"><br />
    <button onclick="validateForm ()">click</button>
    <script>
        function validateForm ()
        {
            var num1=document.getElementById ("num1").value;
            if (num1 >= 10 && num1 < 100)
            {
                alert ("GOOD INPUT");
            }
            else
                alert ("ERROR INPUT ");
        }
    </script>
</body>
</html>
```

Ternary Operator

```
<html>
  <body>
    <h1>Hello Ternary</h1>
    <script>
      var a = 8, b = 9 ;
      var isEqual = a==b ? "hi" : "by";
    </script>
  </body>
</html>
```



The screenshot shows a browser window with the title bar indicating the file path: "/Users/asafamir/Desktop/jonsoni/functions/temp.html". The main content area of the browser displays the text "Hello Ternary" in a large, bold, black font.

כתב תוכנית שבוחרת 2 מספרים ומתריעים שהוא גדול יותר.

כתב תוכנית שמקבלת מספר בן 2 ספרות כקלט ומדפיסה את סכום הספרות של המספר.

אלגוריתם:

- 1.) חלץ את ספרת האחדות לפי $10 \bmod 10$ -> ייחדות.
- 2.) חלץ את העשרות על ידי חלוקה ב-10 -> עשרות.
- 3.) לאחר מכן נוסיף עשרות + ייחדות -> סכום.
- 4.) סכום alert.

Solution

```
<html>
<body>
    <h1>if else </h1>
    Enter 2 digits number : <br>
    <input type="text" id="num1"><br />
    <button onclick="check()">check</button>
    <script>
        function check()
        {
            var num1=parseInt(document.getElementById("num1").value);
            if (num1 >= 10 && num1 < 100)
            {
                var units=num1%10;
                var tens=num1/10;
                var sum=units+tens;
                alert(Math.floor(sum));
            }
            Else
                alert("ERROR INPUT ");
        }
    </script>
</body>
</html>
```



The **Math.floor()** function returns the largest integer less than or equal to a given number. For example:

```
console.log(Math.floor(5.95));
// expected output: 5
```

```
console.log(Math.floor(-5.05));
// expected output: -6
```

כתב תוכנית שתקבל קלט מהמשתמש ותודיעו לר אם המספר מתחלק ב-2.

הבדל בין if ו-if else

```
if (x < 5)
    console.log("5 is larger than x");
if(x>5)
    console.log("5 is smaller than x");
```

Example 2 : x=12

```
if (x > 9)
    console.log(" x is larger than 12");
if (x>7)
    console.log("x is smaller than 12");
if (x>5)
    console.log("x is smaller than 12");
```

```
if (x < 5)
    console.log(" 5 is larger than x");
else if (x>5)
    console.log("5 is smaller than x");
```

Example 2 : x=12

```
if (x > 9)
    console.log(" x is larger than 9");
else if (x>7)
    console.log("x is smaller than 9");
else if (x>5)
    console.log("x is smaller than 9");
```

כתב תוכנית שמקבלת ממספר המזל של המשתמש.
אם מספר המזל הוא 7, המשתמש יתריע על "\$1,000,000\$"

`==` vs(`value`) `vs` `====(type and value)`

```
'' == '0' // false  
'' == 0 // true  
false == 'false' // false  
false == '0' // true  
false == undefined // false  
false == null // false  
null == undefined // true
```

```
var a = [1,2,3];  
var b = a; // == [1,2,3];  
  
var c = { x: 1, y: 2 };  
var d = { x: 1, y: 2 };  
  
var e = "text";  
var f = "te" + "xt";  
  
a == b // true  
a === b // true  
  
c == d // false  
c === d // false  
  
e == f // true  
e === f // true
```

== VS === – Continued

```
"abc" == new String("abc")
// true

"abc" === new String("abc")
// false

true == 1; //true, because
'true' is converted to 1 and
then compared

"2" == 2; //true, because
"2" is converted to 2 and
then compared
```

```
true === 1; //false
"2" === 2; //false

var a = { x: 1, y: 2 };
var b = { x: 1, y: 2 };

var c = a;

var ab_eq = (a === b); //
false (even though a and b
are the same type)

var ac_eq = (a === c); //
true
```

מספר אקראי

```
var n = Math.random() ← Random number 0 - 1
```



```
var n = Math.random() * 100 ← Random number 0 - 100
```



```
n = Math.floor(n)
```

דוגמה לשינוי צבעים

```
<html>  
  
<body>  
  
    <button onclick="disko()">disco</button>  
  
    <p id="num1">0</p>  
  
    <script>  
  
        function disko(){  
  
            var n = Math.random()*100  
            n = Math.floor(n)  
            document.getElementById("num1").innerText = n  
            if(n > 0 && n<30){  
                document.body.style.backgroundColor="red"  
            }  
        }  
    </script>  
</body>  
</html>
```

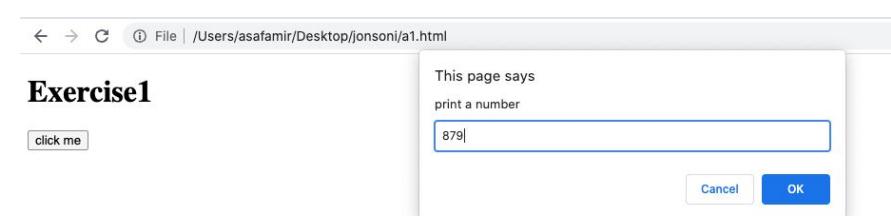
```
if(n >= 30 && n<60){  
    document.body.style.backgroundColor="blue"  
}  
  
if(n >= 60){  
    document.body.style.backgroundColor="pink"  
}  
</script>  
</body>  
</html>
```

דוגמה לשינוי צבעים

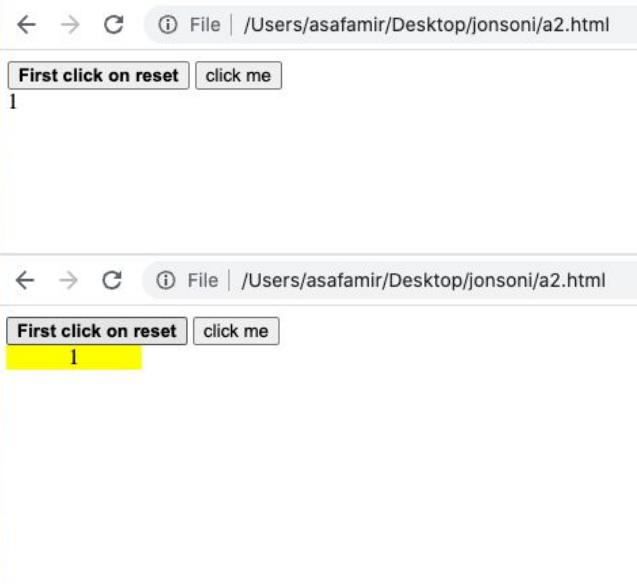


78

1. Create an html page named index.html
 - a. Add a button with the caption click me
 - b. Add a paragraph with id = "details"
 - c. Write a script that receives the user's lucky number from the prompt.
 - d. If the lucky number is 777 the user will see "1,000,000\$" display with a blue color with a yellow background, with a 1px solid green border. else display your name with a yellow color with a black background, with a 5px solid red border
 - e. Clicking the button will activate the function



reset



עשה זאת בעצמך - 5

1. Create an html page named index.html
 - a. Add a button with the caption click me
 - b. Add button with reset caption
 - c. Add a paragraph with id = "details"
 - d. Add a function called reset The function will initialize the div width to be 10px with a yellow background
 - e. Add a function called enlarge The function will increase the div background by 10px
 - f. If `document.getElementById("details").style.width` is more than 500px then the background color of the page will be brown else blue



עשה זאת בעצמך - 6

1. Add a button with the caption click me
 - a. Add a button with the caption click me
 - b. Add a paragraph with id = "details"
 - c. Write a script that receives 2 numbers from the prompt.
 - d. If the Sum of numbers bigger than 100 show the sum of numbers on paragraph with id = "details" with blue color **else** show the sum of numbers on paragraph with id = "details" with red color
 - e. Call the functions by pressing a button.

← → ⌂ ⌂ File | /Users/asafamir/Desktop... ☆

Exercise3

1

← → ⌂ ⌂ File | /Users/asafamir/Desktop... ☆

Exercise3

122

עשה זאת בעצמך - 7

הוסף כפטור עם הכיתוב me click me .1

a. הוסף ``, ``, ``, ``

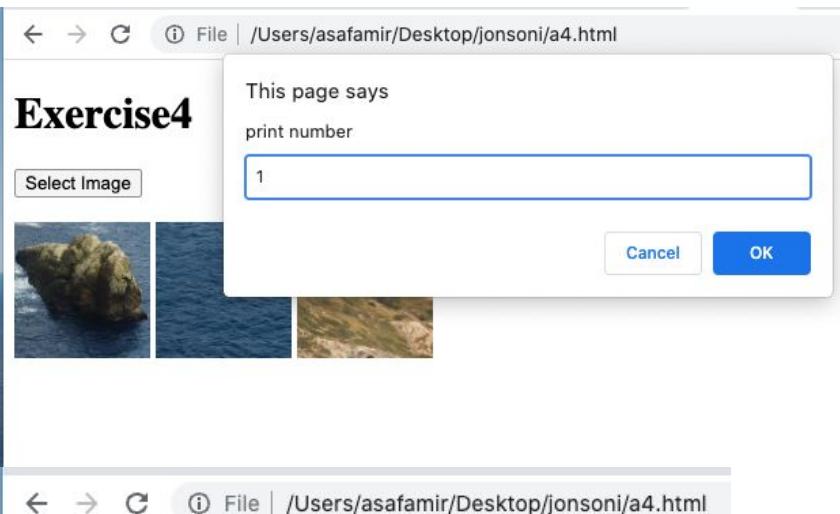
b. כתוב סקריפט שמקבל מספר מה `.prompt`

c. אם המספר קטן מ-100 הצג את כל התמונות

d. אחרת אם המספר שווה ל-100 הצג תמונה ראשונה ושנייה

e. אחרת אם המספר גדול מ-100 מציג רק את התמונה הראשונה

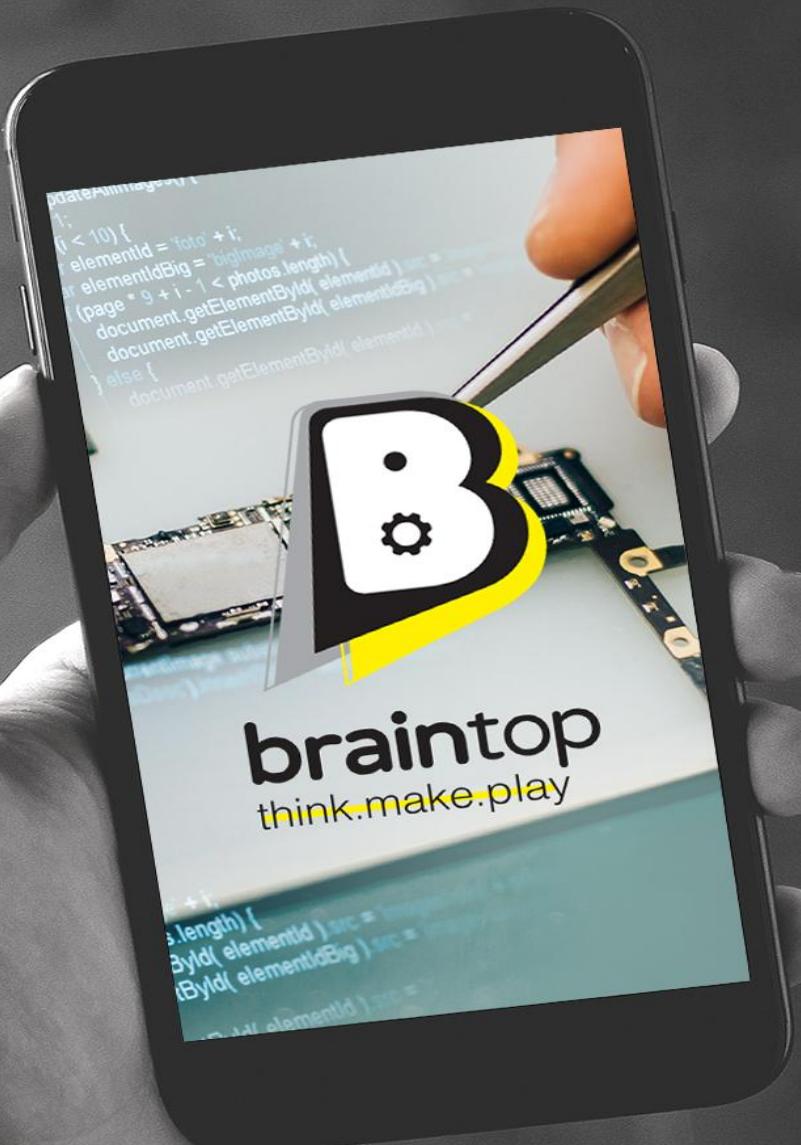
f. קרא לפונקציות על ידי לחיצה על כפטור.



Exercise4



FOR



JAVASCRIPT

For Loops

```
for (init expr; end expr; counter expr )  
{  
    statements  
}
```

1 לוגמַן

```
<html>
<body>
    <h1>for loops example 1</h1>
    <button onclick="print0to10()">start</button>
    <script>
        function print0to10(){
            for(var i=0;i<10;i++)
            {
                console.log(i);
            }
        }
    </script>
</body>
</html>
```

לigma 1



ליגה 2 - 10 loops to 0

```
<html>
<body>
    <h1>for loops 10 to 0</h1>
    <button onclick="print10to0 ()">start</button>
    <script>
        function print10to0 () {
            for (var i=10;i>=0;i--)
            {
                console.log(i);
            }
        }
    </script>
</body>
</html>
```

ליגה 2

A screenshot of a browser's developer tools interface, specifically the Console tab. The title bar shows the file path: /Users/asafamir/Desktop/jonsoni/loopsfor/abc/lesson-60-for/js-fc. The main content area displays the heading "for loops 10 to 0" followed by a "start" button. To the right, the Console tab is active, showing the following output:
10
9
8
7
6
5
4
3
2
1
0

דוגמא 4 - מופרים זוגיים

```
<html>
  <body>
    <h1>for loops</h1>
    <button onclick="printEven()">print even</button>
    <script>
      function printEven() {
        for(var i=0;i<10;i++)
        {
          if(i%2 == 0){
            console.log(i);
          }
        }
      }
    </script>
  </body>
</html>
```

צובר: מטרתו לסקום הערכים – `sum`.
מונה: מטרתו לדעת כמה פעמים הופיע אירוע מסוים - `counter`.

דוגמה 4 ספירה - ספור כמה מספרים גדולים מ-20

```
<html>
<body>
    <h1>for loops</h1>
    <button onclick="howManyBiggerThan20 () " >how many bigger than 20 </button>
    <p id="count"></p>
    <p id="numbers"></p>
    <script>
        function howManyBiggerThan20 () {
            var counter = 0;
            for(var i=0;i<4;i++) {
                var n = prompt("print num ")
                n = parseInt(n)
                if(n > 20) {
                    counter++;// counter
                }
            }
            document.getElementById("count").innerText = counter
        }
    </script>
</body>
</html>
```



עשה זאת בעצמך 1

1. Add a button with the caption count even
 - a. Add a paragraph with id = "details"
 - b. Write a script that receives 4 numbers from the prompt.
 - c. The script will count how many numbers are even(for loop)
 - d. The script will display the number of even numbers on the paragraph id = details
 - e. Call the functions by pressing a button.

צוברים - Accumulates

```
<html>
<body>
    <h1>for loops</h1>
    <button onclick="sumOfNumbers ()">1+2+3 + ...9</button>
    <p id="sum"></p>
    <script>
        function sumOfNumbers () {
            var sum = 0;
            for(var i=1;i<10;i++) {
                sum += i
            }
            document.getElementById("sum").innerText = sum
        }
    </script>
</body>
</html>
```

עשה זאת בעצמך 2

1. הוסף כפטור עם הכתוב סכום של 5 מספרים
 - a. הוסיף פסקה עם `sum = pi`
 - b. כתוב סקריפט שמקבל 5 מספרים מההנחיה.
 - c. התסריט יחשב את סכום המספרים
 - d. התסריט יציג את סכום המספרים על מזזה הפסקה = פרטימ
 - e. התקשר לפונקציות על ידי לחריצה על כפטור.

- הוסף לחץ עם הכתוב **чисוב סכום, ממוצע ומעלת 1000**
1. הוסיף פסקאות עם `id = "above1000"`, `sum = id - i`, `average = sum / id`.
 2. כתוב תסריט שמקבל 5 משכורות מההנחיה.
 3. התסריט יחשב את סכום המשכורות, ממוצע המשכורות, יספור כמה משכורות מעל 1000
 4. הסקריפט יציג את הסכום, הממוצע ומעל 1000 של הפסקה הנוכחית
 5. התקשר לפונקציות על ידי לחיזה על כפטור.

For Inside For

```
<html>
<body>
  <h1>for in for</h1>
  <button onclick="example()">click me</button>
  <p id="output"></p>
  <script>
    function example() {
      var str = ""
      for(var i=0;i<10;i++)
      {
        for(var j=0;j<10;j++)
        {
          str+="*";
        }
        str+="\n"
      }
      document.getElementById("output").innerText =
    }
  </script>
</body>
</html>
```

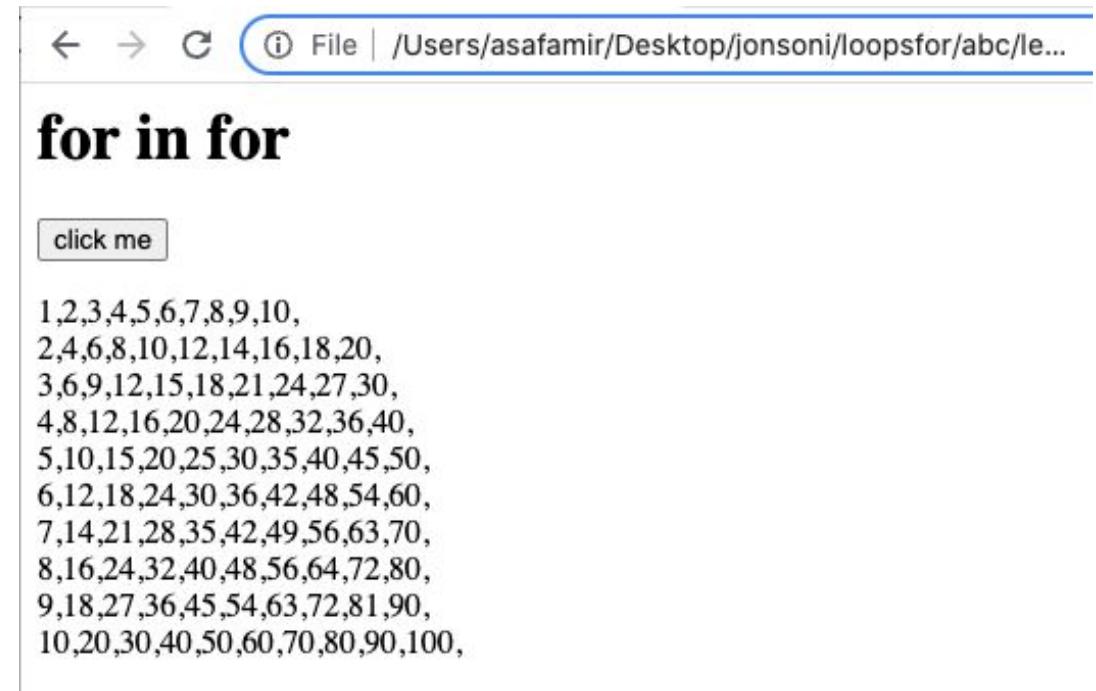


For Inside For – Continued

כתב תוכנית שתציג את לוח הכל.

פתרונות | For Inside For Practice –

```
<html>
<body>
  <h1>for in for</h1>
  <button onclick="example()">click me</button>
  <p id="output"></p>
  <script>
    function example() {
      var str = ""
      for(var i=1;i<=10;i++)
      {
        for(var j=1;j<=10;j++)
        {
          str+=i*j + ", ";
        }
        str+="\n"
      }
      document.getElementById("output").innerText = str;
    }
  </script>
</body>
</html>
```



The screenshot shows a browser window with the URL `/Users/asafamir/Desktop/jonsoni/loopsfor/abc/le...`. The page title is "for in for". There is a single button labeled "click me". Below the button, the output of the script is displayed as a long string of numbers separated by commas and newlines, representing the product of all pairs of integers from 1 to 10.

1,2,3,4,5,6,7,8,9,10,
2,4,6,8,10,12,14,16,18,20,
3,6,9,12,15,18,21,24,27,30,
4,8,12,16,20,24,28,32,36,40,
5,10,15,20,25,30,35,40,45,50,
6,12,18,24,30,36,42,48,54,60,
7,14,21,28,35,42,49,56,63,70,
8,16,24,32,40,48,56,64,72,80,
9,18,27,36,45,54,63,72,81,90,
10,20,30,40,50,60,70,80,90,100,

WHILE

JAVASCRIPT



לולאת while היא לולאה שתמשיך לפעול כל עוד תנאי מסוים מתקיים.

```
while (loop expression)
```

```
{
```

```
    loop expression
```

```
}
```

למה להשתמש ב `while` ?

נשתמש בולאה `for` כאשר נרצה לבצע כמה פעולות מסוימת של פעולות מספר פעמים ידוע. נשתמש ב-`loops while` כאשר מספר הפעמים שהולאה מבוצעת אינו ידוע.

כתב תוכנית שבודחת מספר שלם והפלט הוא מספר הספרות של המספר.

אלגוריתם:

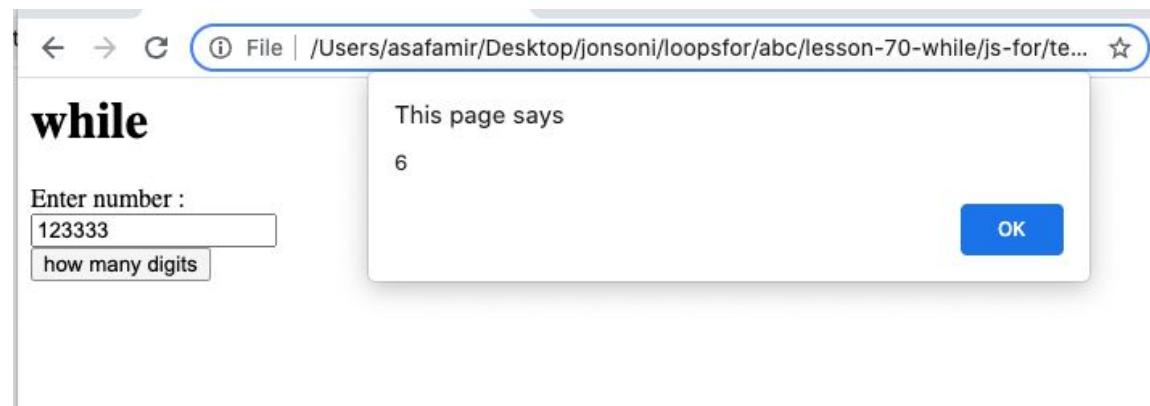
- 1.) חיתוך ספרת היחידה.
- 2.) הגדל את המונה ב-1.
- 3.) בצע את משימות המשנה הללו אם המספר גדול מ-0.

בחירה המשתנים:

- a.) num - ישמור את המספר שהתקבל ואת המספר לאחר הקיצוץ.
- b.) count - כמה ספרות נחתכו.

Example 1 – Solution

```
<html>
<body>
<h1>while </h1>
Enter number : <br>
<input type="text" id="num1" placeholder="print number" /><br />
<button onclick="check()">how many digits </button>
<script>
    function check()
    {
        var count=0;
        var num1=parseInt(document.getElementById("num1").value);
        while (num1 !=0)
        {
            num1=Math.floor(num1/10);
            count++;
        }
        alert(count);
    }
</script>
</body>
</html>
```



lolaha - תרגול

- a. כתוב תוכנית שמדפיסה מספרים מ-5-1 (רק עם while).
- b. כתוב תוכנית שמדפיסה מספרים מ-10 עד 1 (רק עם while).

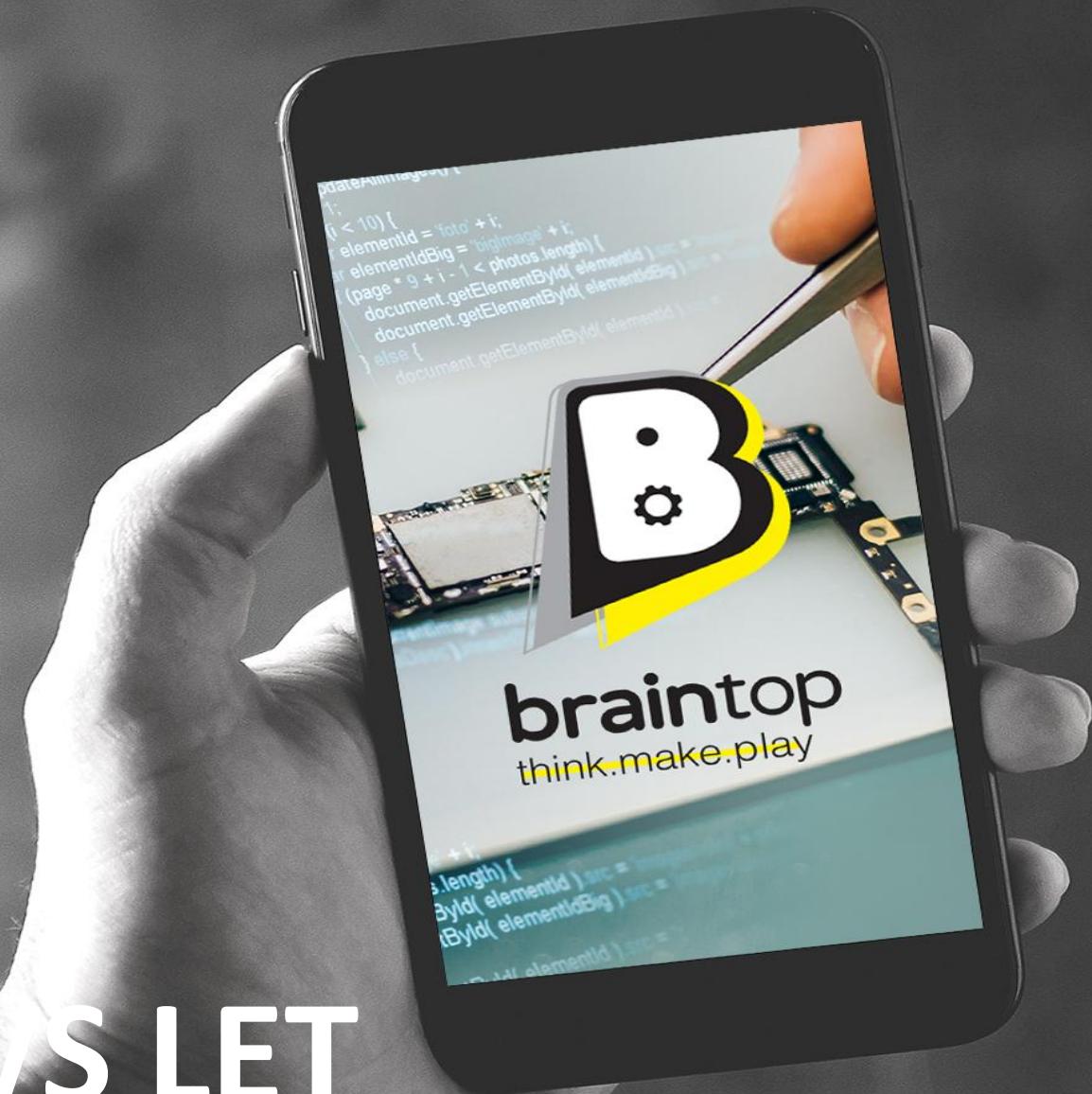
ლולאה - פתרון תרגול

```
1 // program to display numbers from 1 to 5
2 // initialize the variable
3 var i = 1, n = 5;
4
5 // while loop from i = 1 to 5
6 while (i <= n) {
7     console.log(i);
8     i++;
9 }
```

```
1 // program to display numbers from 10 to 1
2 // initialize the variable
3 var index = 10;
4
5 // while loop from i = 1 to 5
6 while (index!=0) {
7     console.log(index);
8     index--;
9 }
```

CONST VS LET

JAVASCRIPT



הצהרת **var** מכריזה על משתנה בהיקף פונקציה או בהיקף גלובלי, באופן אופציוני מתחל אותו לערך.

המשפט **let** מכריז על משתנה מקומי בהיקף בлок, אופציוני מתחל אותו לערך.
ה**const** הם בהיקף בлок, בדומה למשתנים המוצהרים באמצעות מילת המפתח **let**. לא ניתן לשנות את הערך של קבוע באמצעות הקזאה מחדש, ולא ניתן להציג מחדש.

```
<html>
  <body>
    <script>
      var x = 1;
      if (x == 1) {
        var x = 2;
        console.log(x);
        // expected output: 2
      }
      console.log(x);
      // expected output: 2
    </script>
  </body>
</html>
```

הצחרת `var` מכריזה על משתנה בהיקף פונקציה או בהיקף גלובלי, באופן אופציוני מתחילה אותו לערך.

הצחרות `var` מעובדות לפני ביצוע קוד כלשהו.

ההיקף של המשתנה המוצחר עם `var` הוא הקשר הביצוע הנוכחי שלו והסיגריות שלו, שהוא הפונקציה המקיפה והפונקציות המוצחרות בתוכה, או, עבור משתנים שהוכרזו מחוץ לכל פונקציה, גלובליות.

הצחרות משתנים כפולות באמצעות `var` לא יפעלו שגיאה, אפילו במצב קפדי, והמשתנה לא יאבד את ערכו, אלא אם תבוצע הקצהה נוספת.

Strict mode mdn

הערה: **לפעמים** תראה את ברירת המחדל, מצב לא קפדי המכונה "sloppy mode". זה לא מונח رسمي, אבל שים לב לזה, ליתר בטחון.

המצב strict של JavaScript, שהציג ב-5 ECMAScript, הוא דרך להציג גרסה מוגבלת של JavaScript.

Browsers not supporting strict mode will run strict mode code with different behaviour from browsers that do, so don't rely on strict mode without feature-testing for support for the relevant aspects of strict mode.

Strict mode code and non-strict mode code can coexist, so scripts can opt into strict mode incrementally.

More about strict mode -

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

var - 'use strict'

```
<html>
  <body>
    <script>
      'use strict';
      function foo() {
        var x = 1;
        function bar() {
          var y = 2;
          console.log(x); // 1 (function `bar` closes over `x`)
          console.log(y); // 2 (`y` is in scope)
        }
        bar();
        console.log(x); // 1 (`x` is in scope)
        console.log(y); // ReferenceError in strict mode, `y` is scoped to `bar`
      }
      foo();
    </script>
  </body>
</html>
```

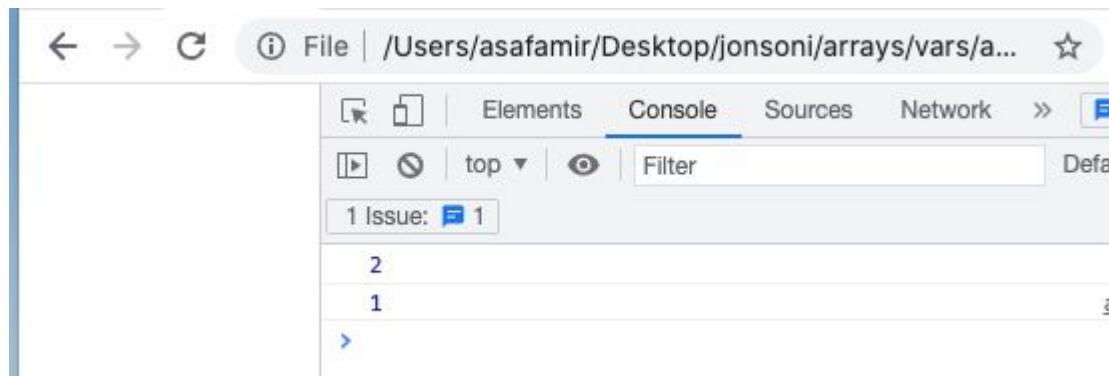
var declarations, wherever they occur, are processed before any code is executed. This is called hoisting, and is discussed further below. The scope of a variable declared with var is its current execution context and closures thereof, which is either the enclosing function and functions declared within it, or, for variables declared outside any function, global..

Duplicate variable declarations using var will not trigger an error, even in strict mode, and the variable will not lose its value, unless another assignment is performed

let

```
<html>
  <body>
    <script>
      let x = 1;
      if (x == 1) {
        let x = 2;
        console.log(x);
        // expected output: 2
      }
      console.log(x);
      // expected output: 1
    </script>
  </body>
</html>
```

let allows you to declare variables that are limited to the scope of a block statement, or expression on which it is used, unlike the var keyword, which declares a variable globally, or locally to an entire function regardless of block scope. The other difference between var and let is that the latter is initialized to a value only when a parser evaluates it (see below).



Let - scoping rules

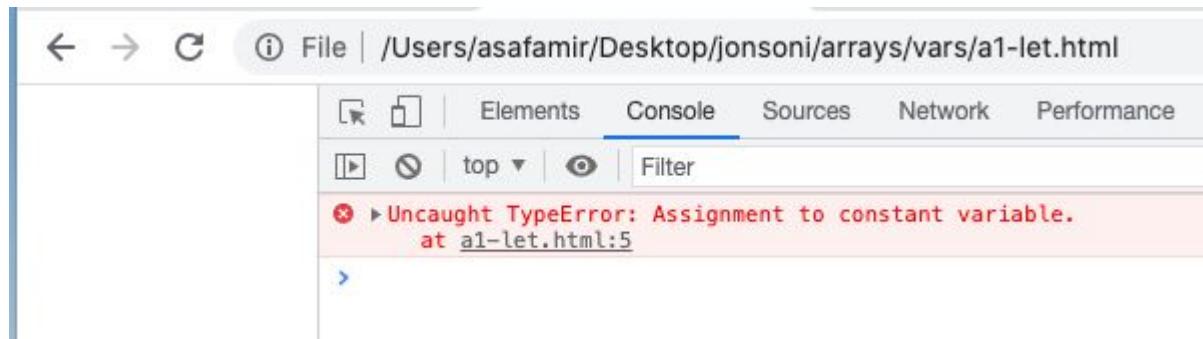
```
<html>
  <body>
    <script>
      function varTest () {
        var x = 1;
        {
          var x = 2; // same variable!
          console.log(x); // 2
        }
        console.log(x); // 2
      }
      function letTest () {
        let x = 1;
        {
          let x = 2; // different variable
          console.log(x); // 2
        }
        console.log(x); // 1
      }
    </script>
  </body>
</html>
```

Variables declared by let have their scope in the block for which they are declared, as well as in any contained sub-blocks. In this way, let works very much like var. The main difference is that the scope of a var variable is the entire enclosing function:

const

```
<html>  
  <body>  
    <script>  
      const number = 42;  
  
      number = 99;  
  
      console.log(number);  
    </script>  
  </body>  
</html>
```

Constants are block-scoped, much like variables declared using the [let](#) keyword. The value of a constant can't be changed through reassignment, and it can't be redeclared.



const - scoping rules

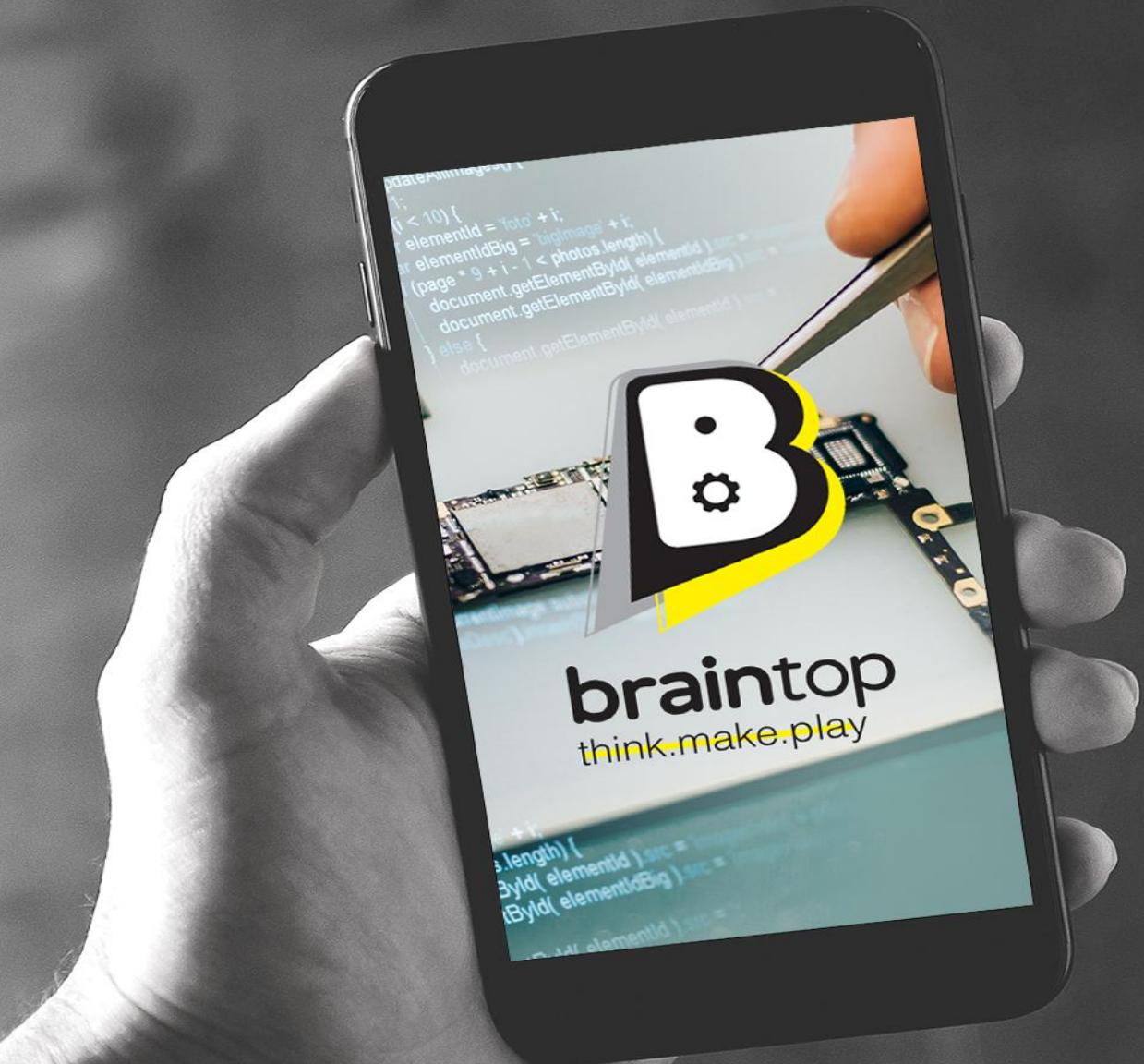
```
<html>
  <body>
    <script>
      // define MY_FAV as a constant and give it the value 7
      const MY_FAV = 7;
      // this will throw an error - Uncaught TypeError: Assignment to constant variable.
      MY_FAV = 20;
      // MY_FAV is 7
      console.log('my favorite number is: ' + MY_FAV);
      // trying to redeclare a constant throws an error
      // Uncaught SyntaxError: Identifier 'MY_FAV' has already been declared
      const MY_FAV = 20;
      // the name MY_FAV is reserved for constant above, so this will fail too
      var MY_FAV = 20;
      // this throws an error too
      let MY_FAV = 20;
    </script>
  </body>
</html>
```



Constants can be declared with uppercase or lowercase, but a common convention is to use all-uppercase letters.

ARRAY

JAVASCRIPT



אם נרצה לכתב תוכנית שמקבלת משכורת של 2000 עובדים ומדפיסה אותם.
איך אנחנו עושים את זה?

אפשרות אחת היא להגיד 2000 משתנים ולא תחל אותם בرمת השכר, אולי זה גוזל זמן.
במקום זאת נוכל להשתמש במערך.

```
<html>
<body>
<h1>Array example 1</h1>
<script>
  var colors = ["red", "blue", "brown"];
  console.log(colors);
  var mixedArr = ["dog", 3, "cat", 9, 12];
</script>
</body>
</html>
```



עד עכשו כדי ליצג 10 משכורות של עובדים הינו צריכים לקבוע:

```
var salary1, salary2, ... , salary10;
```

במקום זאת נגיד מערך של vars:

```
var arr = [] // empty array
```

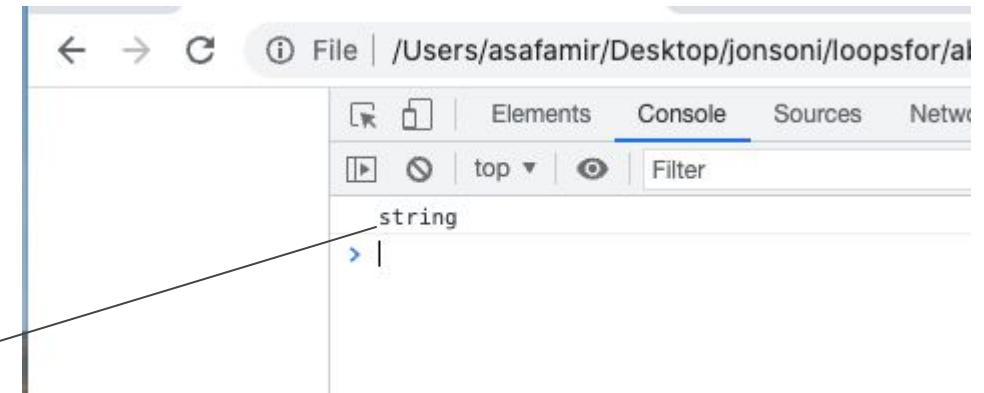
כמה דברים שכדי לזכור: כמה דברים שכדי לזכור:

- 1.) התייחסות לאיברי המערך נעשית באמצעות אינדקס.
 - 2.) האיבר הראשון באינדקס יהיה 0
 - 3.) נתיחס ל-var הראשון באופן הבא: משכורת [0]
 - 4.) האינדקס האחרון יהיה [גודל מערך פחות 1]
 - 5.) ל-var האחרון אנו מתייחסים כדלקמן: salary [9]
- דוגמא לולאה:

```
for (var i=0; i<10; i++)  
    salary[i]=0;
```

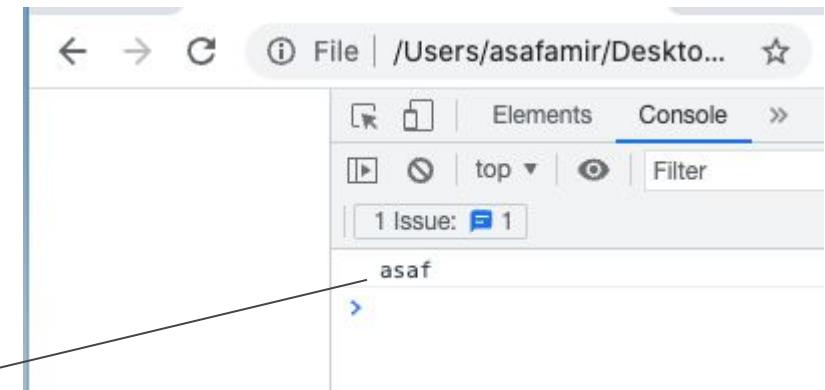
Initialization - אתחול

```
<html>
<body>
<script>
    var arr= [ ];
    arr[0]=12;
    arr[1]="string";
    console.log(arr[1]) //print string
</script>
</body>
</html>
```

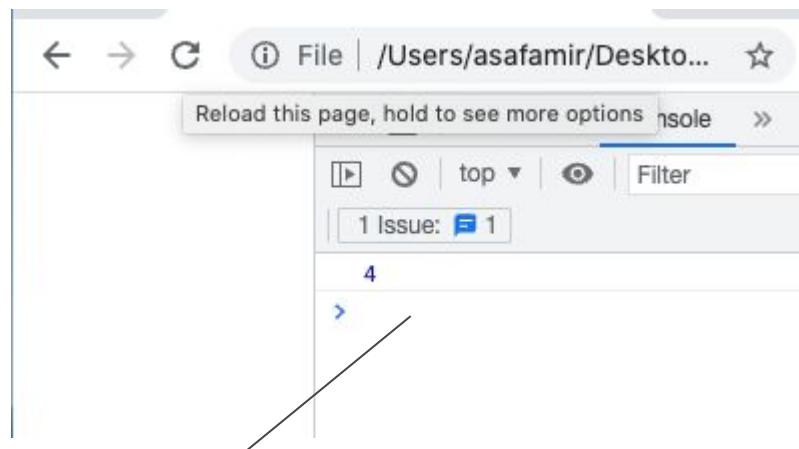


Initialization - אתחול

```
<html>
  <body>
    <script>
      var arr = new Array();
      arr[0]=12;
      arr[1] = "asaf";
      console.log(arr[1]); //print asaf
    </script>
  </body>
</html>
```



Initialization - אתחול



```
<script>

    var arr=[1,4,7,12];

    console.log(arr[1]) //print 4

</script>

</body>

</html>
```

אל תבללו את המספר בשורת הגדרה של המערך!

```
var arr=[1,2,3,4,5,6,7,8,9]
```

Arr[8] is the last index on the array

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

Property - Meta information about object.

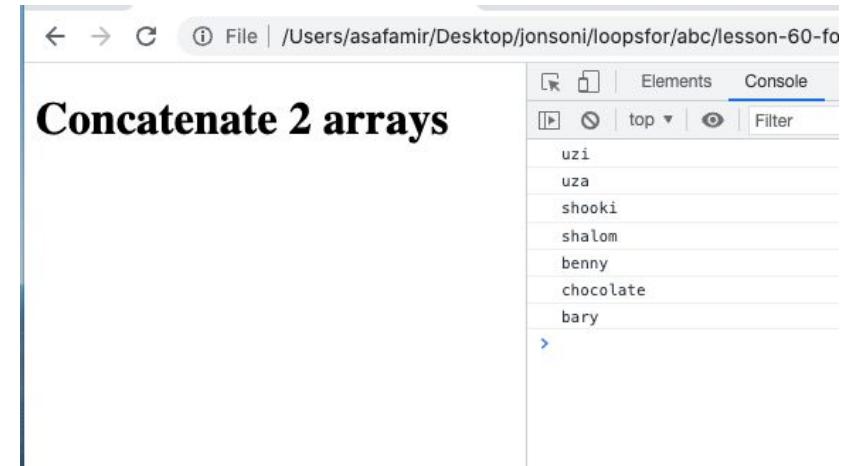
Method - Functions belonging to object.

More - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice#syntax

Concat

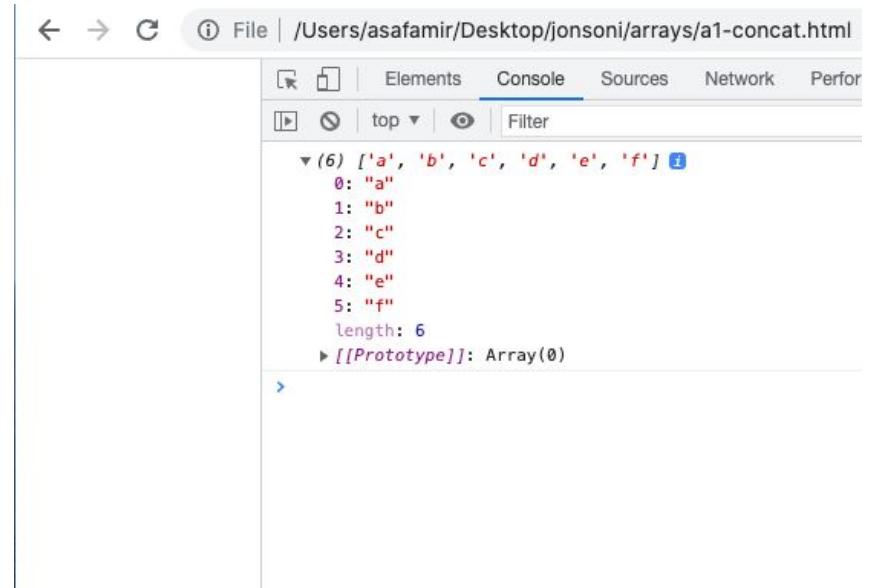
```
<!DOCTYPE html>
<html>
<body>
    <h1>Concatenate 2 arrays</h1>
    <script>
        var arr1=new Array("uzi","uza","shooki");
        var arr2=new Array("shalom","benny","chocolate","bary");
        var all=arr1.concat(arr2);
        for(var i=0;i<all.length;i++)
        {
            console.log(all[i]);
        }
    </script>
</body>
</html>
```

השיטה `concat()` משמשת למזג שני מערכים או יותר. שיטה זו אינה משנה את המרכיבים המקוריים, אלא מחזירה מערך חדש.



Concat mdn examples

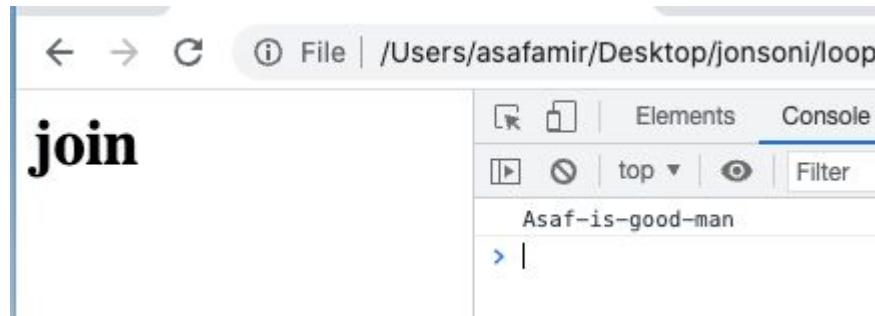
```
<html>
  <body>
    <script>
      const array1 = ['a', 'b', 'c'];
      const array2 = ['d', 'e', 'f'];
      const array3 = array1.concat(array2);
      console.log(array3);
      // expected output: Array ["a", "b", "c", "d", "e", "f"]
    </script>
  </body>
</html>
```



Join

השיטה `join()` יוצרת ומחזירה מחוצת חדשה על ידי שרשור כל האלמנטים במערך (או באובייקט דמי מערך), מופרדים בפסיקים או במחוצת מפריד שמצוינה. אם לערך יש רק פריט אחד, הערך זהה יוחזר ללא שימוש במפריד.

```
<html>
<body>
    <h1>join</h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        var str=arr.join("-");//return string object : Asaf-is-good-man
        console.log(str);
    </script>
</body>
</html>
```



Join mdn example

```
<html>
  <body>
    <script>

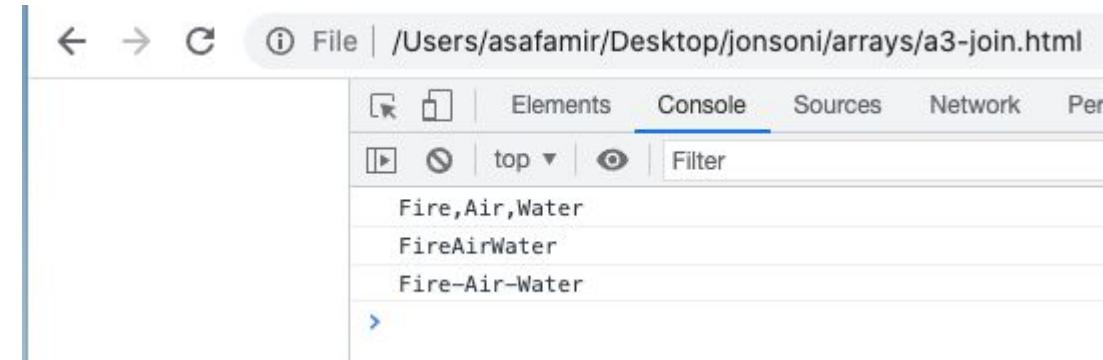
      const elements = ['Fire', 'Air', 'Water'];

      console.log(elements.join());
      // expected output: "Fire,Air,Water"

      console.log(elements.join(' '));
      // expected output: "Fire Air Water"

      console.log(elements.join('-'));
      // expected output: "Fire-Air-Water"

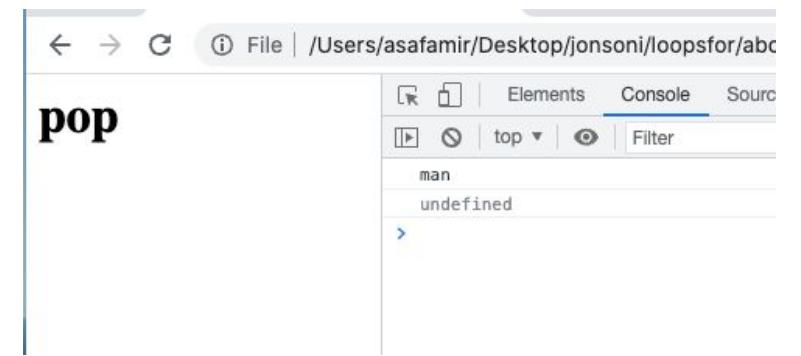
    </script>
  </body>
</html>
```



Pop

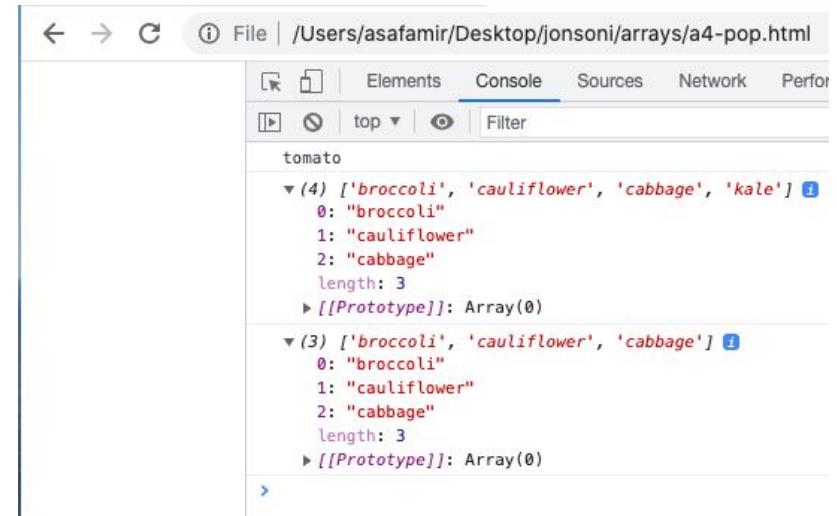
השיטה `pop()` מסירה את האלמנט האחרון ממערך ומחזירה את האלמנט זהה. שיטה זו משנה את אורך המערך.

```
<html>
<body>
    <h1>pop</h1>
    <script>
        var arr=new Array("Asaf", "is", "good", "man");
        var strpop=arr.pop();
        console.log(strpop);
        document.write(arr[3]); //undefined
    </script>
</body>
</html>
```



Pop mdn examples

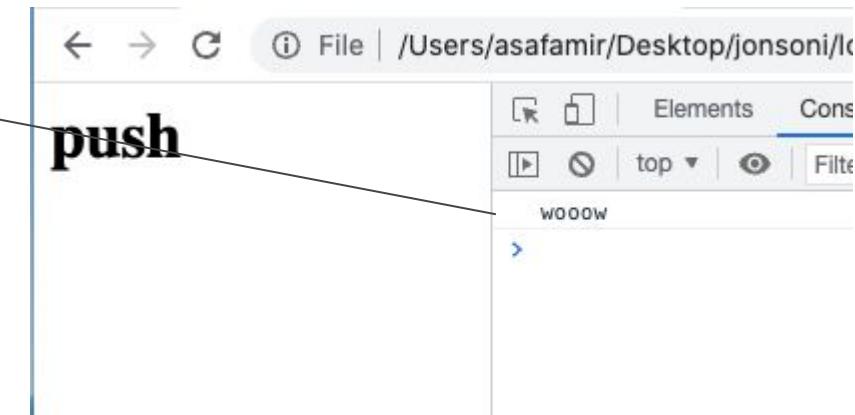
```
<html>
  <body>
    <script>
      const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];
      console.log(plants.pop());
      // expected output: "tomato"
      console.log(plants);
      // expected output: Array ["broccoli", "cauliflower", "cabbage", "kale"]
      plants.pop();
      console.log(plants);
      // expected output: Array ["broccoli", "cauliflower", "cabbage"]
    </script>
  </body>
</html>
```



Push

שיטה `push()` מוסיף אלמנט אחד או יותר לסופם של מערך ומחזירה את האורך החדש של המערך.

```
<html>
<body>
    <h1>push</h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        arr.push("woooow");
        console.log(arr[4]);
    </script>
</body>
</html>
```



Push mdn examples

```
<html>
  <body>
    <script>
      const animals = ['pigs', 'goats', 'sheep'];
      const count = animals.push('cows');
      console.log(count);
      // expected output: 4
      console.log(animals);
      // expected output: Array ["pigs", "goats", "sheep", "cows"]
      animals.push('chickens', 'cats', 'dogs');
      console.log(animals);
      // expected output: Array ["pigs", "goats", "sheep", "cows", "chickens",
      "cats", "dogs"]
    </script>
  </body>
</html>
```

The screenshot shows the browser's developer tools console tab. It displays two arrays related to the 'animals' variable. The first array, at index 4, contains the initial values: 'pigs', 'goats', 'sheep', and 'cows'. The second array, at index 7, shows the result of the push operation, containing all the original elements plus the new ones: 'pigs', 'goats', 'sheep', 'cows', 'chickens', 'cats', and 'dogs'. Both arrays have a length of 7.

```
4
  ▾ (4) ['pigs', 'goats', 'sheep', 'cows']
    0: "pigs"
    1: "goats"
    2: "sheep"
    3: "cows"
    4: "chickens"
    5: "cats"
    6: "dogs"
    length: 7
  ▷ [[Prototype]]: Array(0)

  ▾ (7) ['pigs', 'goats', 'sheep', 'cows', 'chickens', 'cats', 'dogs']
    0: "pigs"
    1: "goats"
    2: "sheep"
    3: "cows"
    4: "chickens"
    5: "cats"
    6: "dogs"
    length: 7
  ▷ [[Prototype]]: Array(0)
```

Reverse

השיטה reverse() הופכת מערך במקומות. אלמנט המערך הראשון הופך לאחרון, ואלמנט המערך האחרון הופך הראשון.

```
<html>
<body>
    <h1>reverse</h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        arr.reverse();
        var strreverse=arr.join("-");
        console.log(strreverse)
    </script>
</body>
</html>
```



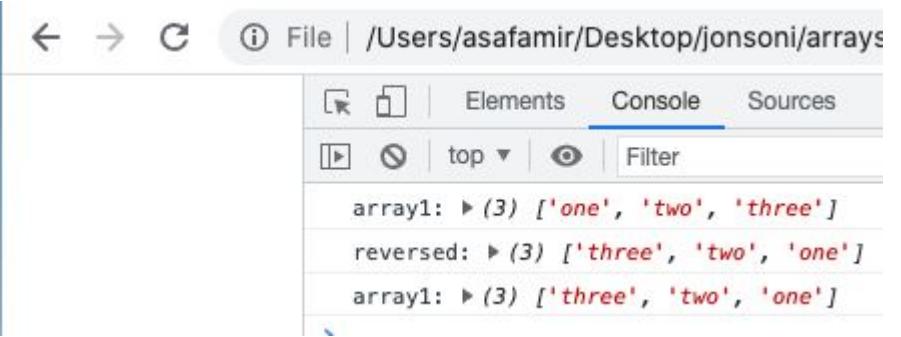
Reverse mdn examples

```
<html>
  <body>
    <script>
      const array1 = ['one', 'two', 'three'];
      console.log('array1:', array1);
      // expected output: "array1:" Array ["one", "two", "three"]

      const reversed = array1.reverse();
      console.log('reversed:', reversed);
      // expected output: "reversed:" Array ["three", "two", "one"]
      // Careful: reverse is destructive -- it changes the original array.

      console.log('array1:', array1);
      // expected output: "array1:" Array ["three", "two", "one"]

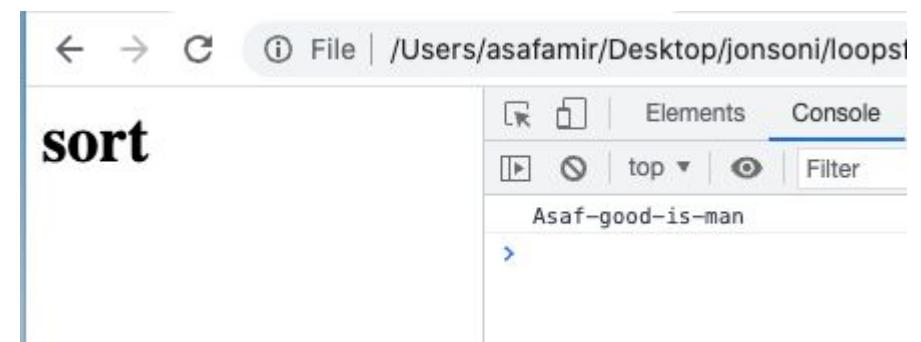
    </script>
  </body>
</html>
```



Sort

השיטה `sort()` ממיינת את האלמנטים של מערך במקומם ומחזירה את המערך המקורי. סדר המין המוגדר כברירת מחדל הוא עולה, בניו על המרת האלמנטים למחוזות, ולאחר מכן השוואת הרצפים שלהם של ערכי ייחודת הקוד של UTF-16.

```
<html>
<body>
    <h1>sort</h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        arr1.sort();
        var strSort=arr.join("-");
        console.log(strSort);
    </script>
</body>
</html>
```



Sort mdn examples

```
<html>
  <body>
    <script>

      const months = ['March', 'Jan', 'Feb', 'Dec'];

      months.sort();

      console.log(months);

      // expected output: Array ["Dec", "Feb", "Jan", "March"]

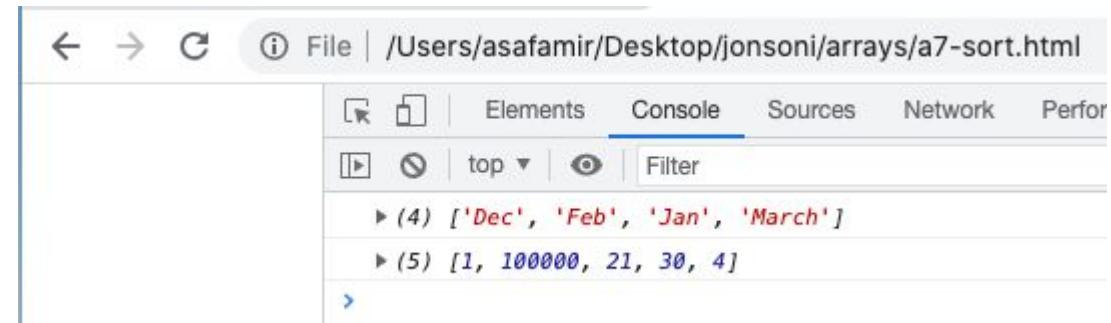
      const array1 = [1, 30, 4, 21, 100000];

      array1.sort();

      console.log(array1);

      // expected output: Array [1, 100000, 21, 30, 4]

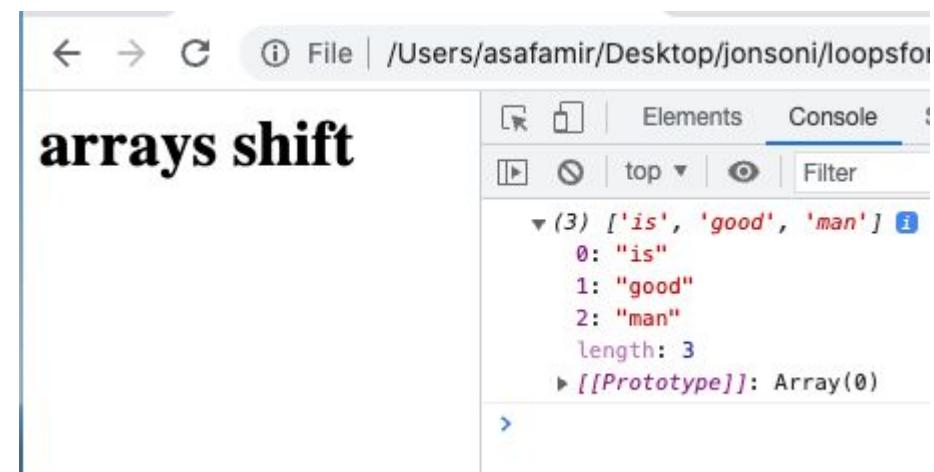
    </script>
  </body>
</html>
```



Shift

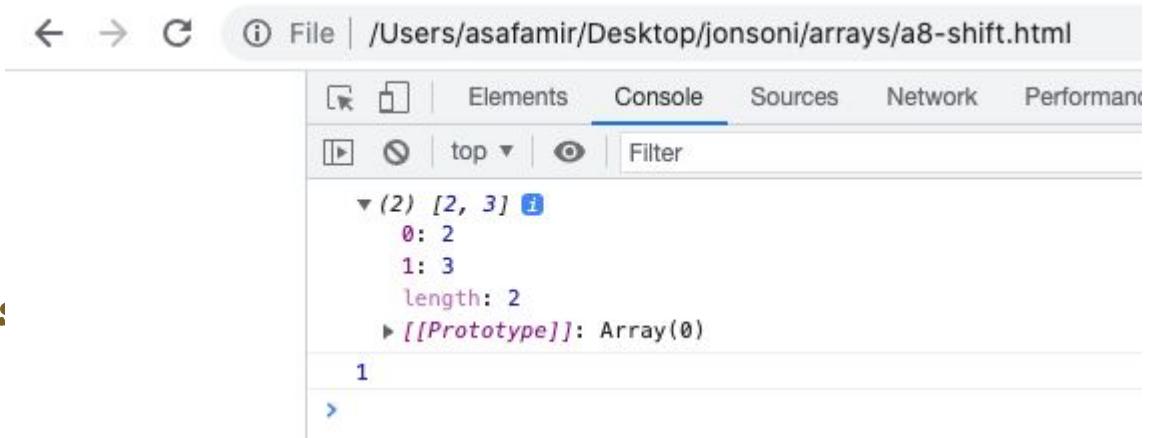
שיטה shift() מסירה את האלמנט הראשון ממערך ומחזירה את האלמנט שהוסר. שיטה זו משנה את אורך המערך.

```
<html>
<body>
    <h1>arrays shift </h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        arr.shift();
        console.log(arr);
    </script>
</body>
</html>
```



Shift mdn examples

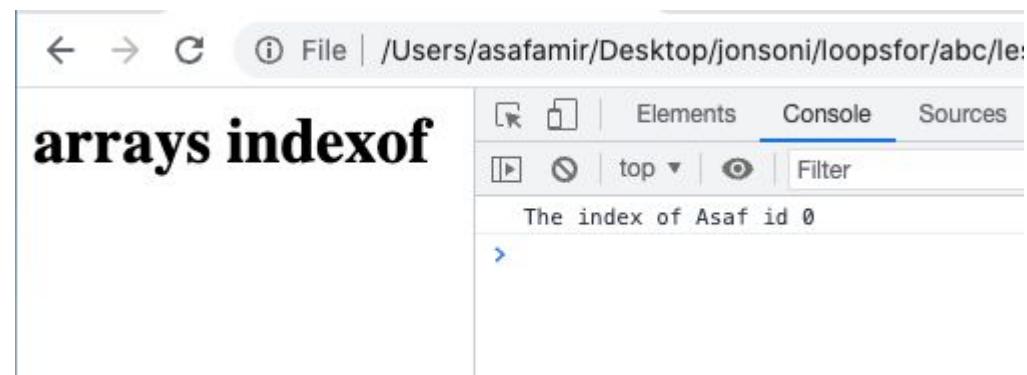
```
<html>
  <body>
    <script>
      const array1 = [1, 2, 3];
      const firstElement = array1[0];
      console.log(array1);
      // expected output: Array [2, 3]
      console.log(firstElement);
      // expected output: 1
    </script>
  </body>
</html>
```



IndexOf

השיטה `indexOf()` מחזירה את האינדקס הראשון שבו ניתן למצוא אלמנט נתון במערך, או 1- אם הוא אינו קיים.

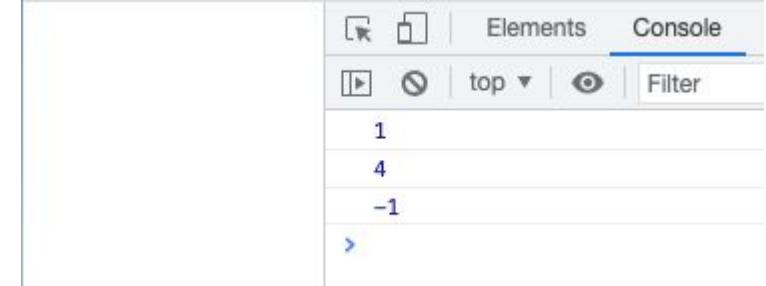
```
<html>
<body>
    <h1>arrays indexof </h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        var pos = arr.IndexOf("Asaf");
        console.log("The index of Asaf id " + pos);
    </script>
</body>
</html>
```



IndexOf mdn examples

```
<html>
  <body>
    <script>
      const beasts = ['ant', 'bison', 'camel', 'duck', 'bison'];
      console.log(beasts.indexOf('bison'));

      // expected output: 1
      // start from index 2
      console.log(beasts.indexOf('bison', 2));
      // expected output: 4
      console.log(beasts.indexOf('giraffe'));
      // expected output: -1
    </script>
  </body>
</html>
```



Splice

שיטה splice() משנה את התוכן של מערך על ידי הסרה או החלפה של אלמנטים קיימים /או הוספת אלמנטים חדשים במקום. כדי לגשת לחלק ממערך מבלי לשנות אותו, ראה slice().

```
<html>
<body>

<h1>Find index of array and remove it from array </h1>

<script>

    var arr=new Array("Asaf","is","good","man") ;

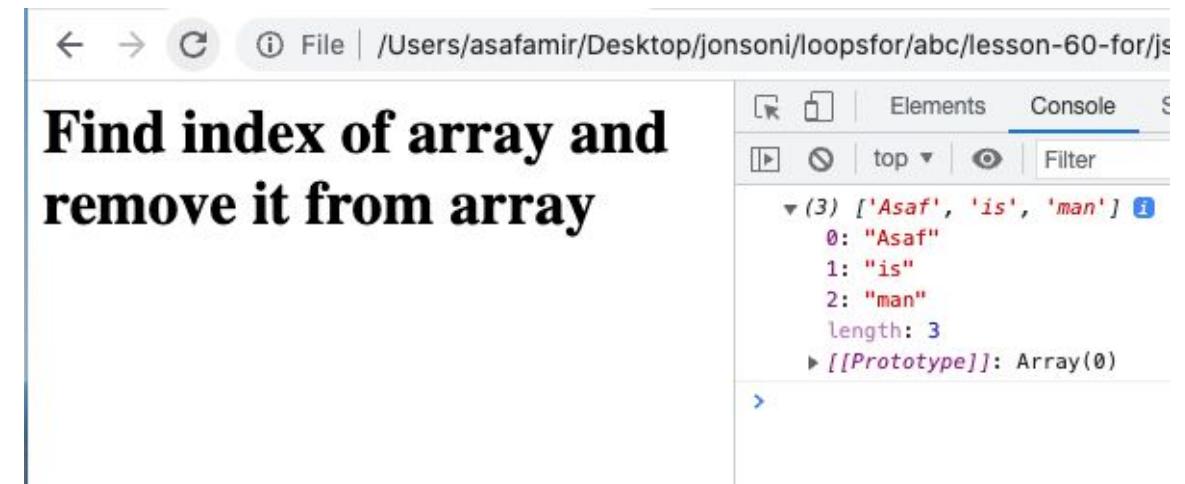
    var pos = arr.indexOf("good") ;

    arr.splice(pos,1) ;

    console.log(arr)

</script>

</body>
</html>
```



**Find index of array and
remove it from array**

Splice mdn examples

```
<html>
  <body>
    <script>
      const months = ['Jan', 'March', 'April', 'June'];

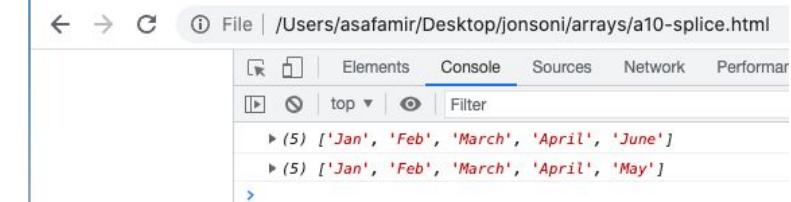
      months.splice(1, 0, 'Feb');
      // inserts at index 1

      console.log(months);
      // expected output: Array ["Jan", "Feb", "March", "April", "June"]

      months.splice(4, 1, 'May');
      // replaces 1 element at index 4

      console.log(months);
      // expected output: Array ["Jan", "Feb", "March", "April", "May"]

    </script>
  </body>
</html>
```



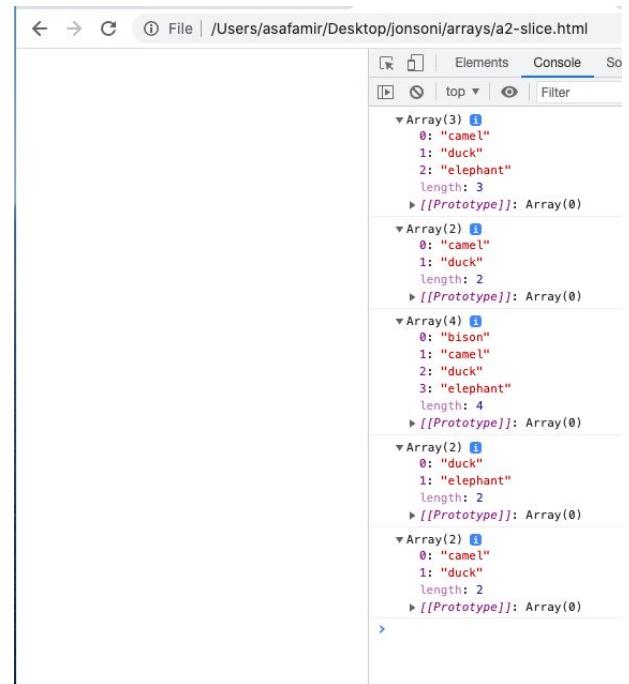
Slice

השיטה `slice()`מחזירה עותק רדוֹד של חלק ממערך לתוכן אובייקט מערך חדש שנבחר מההתחלת ועד הסוף (סוף לא כולל) כאשר התחלת והסוף מייצגים את האינדקס של פריטים במערך זה. המערך המקורי לא ישתנה.

```
<html>
<body>
    <h1>Arrays slice</h1>
    <script>
        var arr=new Array("Asaf","is","good","man");
        var arr = arr.slice();
        console.log(arr);
    </script>
</body>
</html>
```

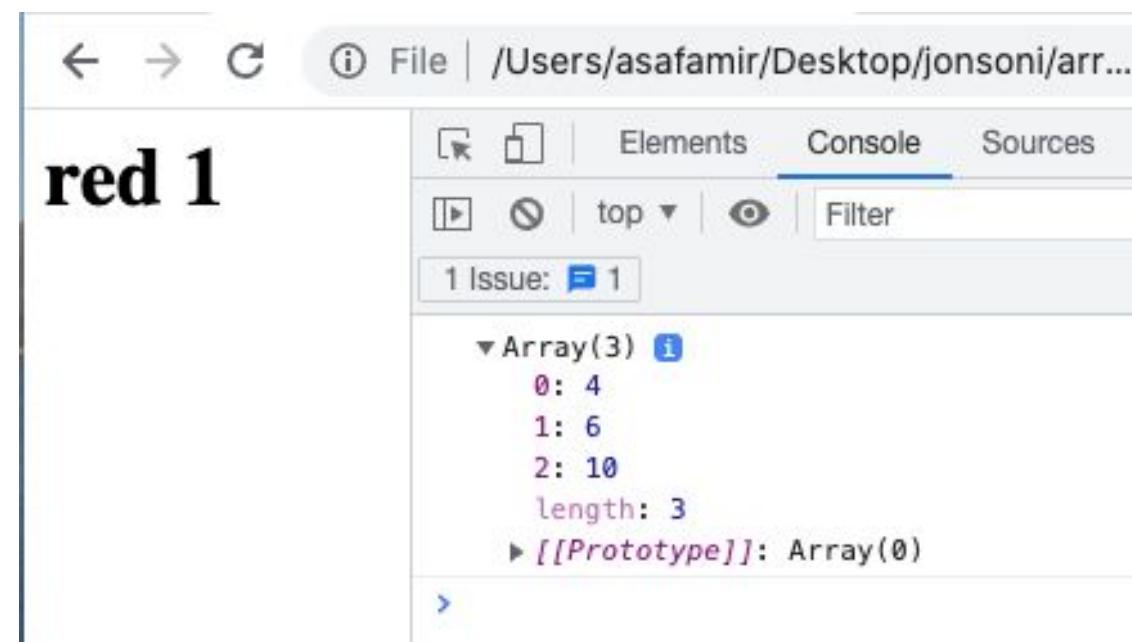
Slice examples

```
<html>
  <body>
    <script>
      const animals = ['ant', 'bison', 'camel', 'duck',
'elephant'];
      console.log(animals.slice(2));
      // expected output: Array ["camel", "duck", "elephant"]
      console.log(animals.slice(2, 4));
      // expected output: Array ["camel", "duck"]
      console.log(animals.slice(1, 5));
      // expected output: Array ["bison", "camel", "duck",
"elephant"]
      console.log(animals.slice(-2));
      // expected output: Array ["duck", "elephant"]
      console.log(animals.slice(2, -1));
      // expected output: Array ["camel", "duck"]
    </script>
  </body>
</html>
```



עשה זאת בעצמך 1

1.) כתבו סקRYPT המאתחל מערך עם המספרים 4,6,10. הדפו את המערך לkonsoleה.



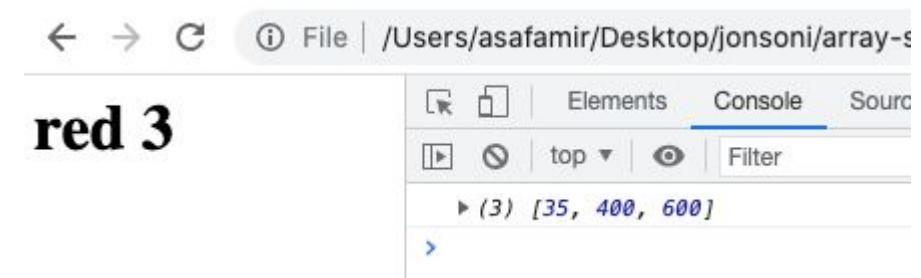
עשה זאת בעצמך 2

כתוב סקRYPT המאתחל מערך עם המספרים 4,6,10. הדפו את הצד ההפוך של המערך לקונסולה.



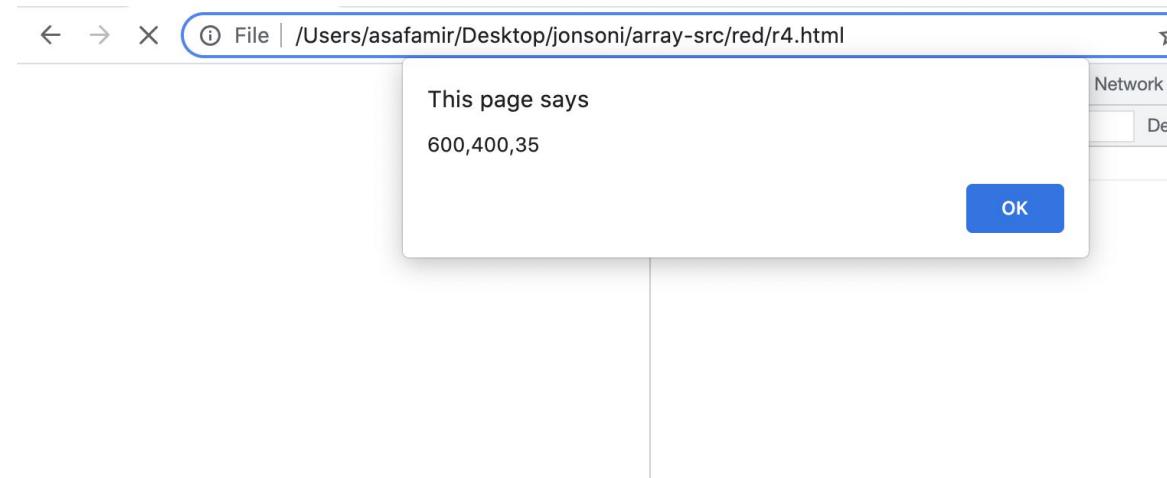
עשה זאת בעצמך 3

כתב סקריפט המאתחל מערך עם המספרים 35, 400, 600. הדפו את המערך לקונסולה. הציג בהתראה את המספרים בסדר עולה.



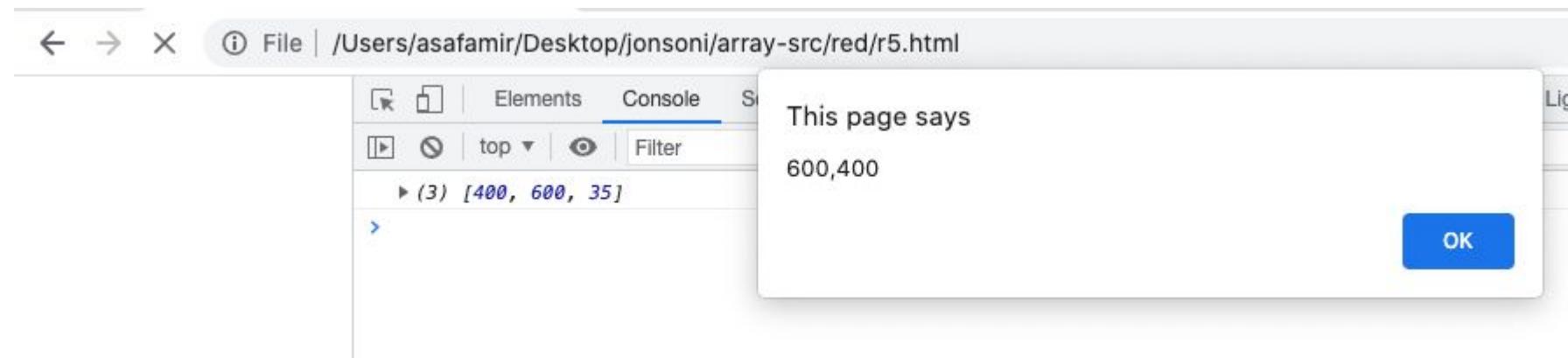
עשה זאת בעצמך 4

Write a script that initializes an array with the numbers 400,600,35. Print the array .to the console. display in alert the numbers in an ascending order



עשה זאת בעצמך 5

כתב סקRYPT המאתחל מערך עם המספרים 400,600,35. הדס את המערך לkonsoleה. מחק את האלמנט האחרון במערך והציג בהתראה את המספרים בסדר יורד.



עשה זאת בעצמך 6

כתב סקRYPT המאתחל סט עם המספרים 1,2,3 והדפס את הסט לkonsole.



1. Add a button with the caption Add numbers to array
 - a. Add a paragraphs with id = "numbers"
 - b. Declare empty array (let arr=[])
 - c. Write a script that receives 5 numbers from the prompt and push them to arr.
 - d. The script will display on id = "numbers" the arrays with comma between every number



i. Hint :

```
for(var i=0;i<arr.length;i++) {
    document.getElementById("numbers").append(arr[i] + ", ")
}
```

- e. Call the functions by pressing the button.



FUNCTIONS

JAVASCRIPT



פונקציות

כבר למדנו על פונקציות, מה הן ולמה אנחנו משתמשים בהן.
בפרק זה נרחיב על פונקציות.

ב-JavaScript ישן 4 דרכי שונות לכתיבה של פונקציות:

- 1.) Function declaration
- 2.) Function expression
- 3.) Arrow function
- 4.) Immediate function

Function Declaration

הצhardt פונקציות היא הדרך המסורתית להגדיר פונקציה.
אנו מתחילהים להציג באמצעות מילת המפתח "function". לאחר מכן נכתוב את שם הפונקציה ולאחר מכן פרמטרים.

```
<html>
  <body>
    <script>
      // Function declaration
      function add(a, b) {
        console.log(a + b);
      }
      // Calling a function
      add(4, 5);
    </script>
  </body>
</html>
```

Expressions Function

ביטוי פונקציות הוא דרך נוספת להגדיר פונקציה ב-JavaScript. CAN ANO MAGDIRIM FONKZIA BAAMZUOT MASHTNA OMACHSONIM AT HURK HMOHZER BAOTU MASHTNA.

```
<html>
  <body>
    <script>
      // Function Expression
      const add = function(a, b) {
        console.log(a+b);
      }

      // Calling function
      add(4, 5);

    </script>
  </body>
</html>
```

Expressions Function

היות אונונימית - היא לא חייבת להיות בעלי שם.
הגדר את ריבוע הפונקציה כך:

```
const square = function(number) {  
    return number * number  
}  
  
var x = square(4) // x gets the value 16
```

Expressions Function

עם זאת, ניתן לספק שם עם ביטוי פונקציה.
מתן שם מאפשר לפונקציה להתייחס לעצמה, וגם מקל על זיהוי הפונקציה של stack traces בdebugger.

```
const factorial = function fac(n) {  
    if(n>0 && n<=1){  
        return 1;  
    } else {  
        return n*fac(n-1);  
    }  
}  
console.log(factorial(3)) // = 6
```

Factorial Formula

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 1$$

$$1! = 1$$

$$2! = 2 \times 1 = 2$$

$$3! = 3 \times 2 \times 1 = 6$$

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

Expressions Function

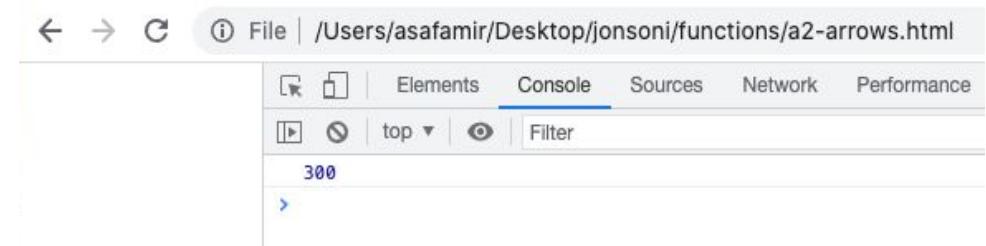
ב-JavaScript, ניתן להגדיר פונקציה על סמך תנאי. לדוגמה, הגדרת הפונקציה הבאה מגדירה את myFunc רק אם num שווה ל-0:

```
var myFunc;  
var num = 0;  
if (num == 0) {  
    myFunc = function() {  
        return num++;  
    }  
    myFunc();  
}  
console.log(num); //1
```

Arrow Function

פונקציות החיצים הוצגו בגרסת ES של JavaScript. הוא משמש לקיצור הקוד.

```
<html>
  <body>
    <script>
      const calcBonus = (bonus) => bonus + 100;
      const finalBonus = calcBonus(200)
      console.log(finalBonus); //300
    </script>
  </body>
</html>
```



Arrow Function

```
<html>
  <body>
    <script>
      const priceCar = (sits) => sits*1000;

      const finalPrice = priceCar(4)

      console.log(finalPrice);

      // Remove the argument parentheses

      const priceCar = sits => sits*1000;

    </script>
  </body>
</html>
```

Arrow Function

```
<html>
  <body>
    <script>
      const priceCar = (sits) => sits*1000;

      const finalPrice = priceCar(4)

      console.log(finalPrice);

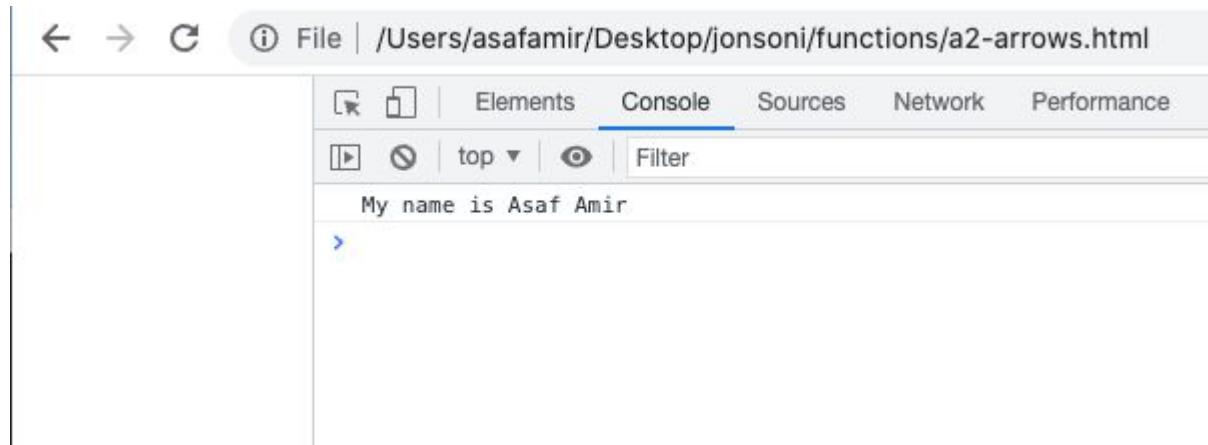
      // Remove the argument parentheses

      const priceCar = sits => sits*1000;

    </script>
  </body>
</html>
```

Arrow Function

```
const fullName = (firstname, lastname) => {  
  const fullname = "My name is " + firstname + " " + lastname  
  return fullname;  
}  
  
const myName = fullName("Asaf", "Amir");  
  
console.log(myName);
```



Immediately Function

פונקציה מיידית היא זו שמתבצעת ברגע שהיא מוגדרת.

```
<html>
  <body>
    <script>
      (function sum(a, b) {
        var r = a + b;
        console.log(r);
      }(3, 5));
    </script>
  </body>
</html>
```

What Will Print?

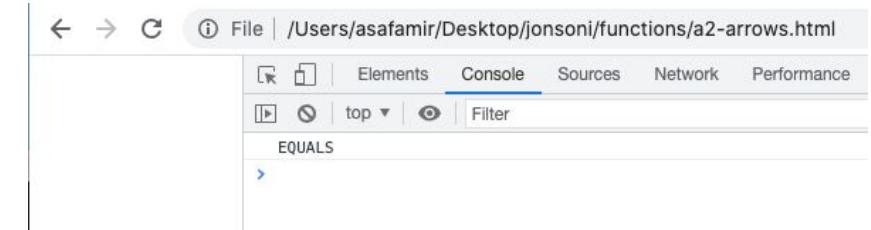
```
<html>
  <body>
    <script>
      max();
      function max() {
        a > b ? console.log(a) : console.log(b);
      }
      var a = 5, b = 6 ;
    </script>
  </body>
</html>
```

What Will Print?

```
<html>
  <body>
    <script>
      max();
      function max() {
        a > b ? console.log(a) : console.log(b); // undefined
      }
      var a = 5, b = 6;
    </script>
  </body>
</html>
```

Immediately Functions – Continued

```
<html>
  <body>
    <script>
      var theBiggest = (function(n1, n2) {
        if(n1 > n2)
          return n1;
        else if(n1 < n2)
          return n2;
        else
          return "EQUALS";
      }(a,b));
      var a = 6;
      var b = 7;
      console.log(theBiggest); //EQUALS
    </script>
  </body>
</html>
```



עשה זאת בעצמך 1

א. כתוב פונקציית Declaration שמקבלת 3 מספרים ומחזירה את המספר המקסימלי.

```
{} function maxBetween3Numbers(n1, n2, n3)
```

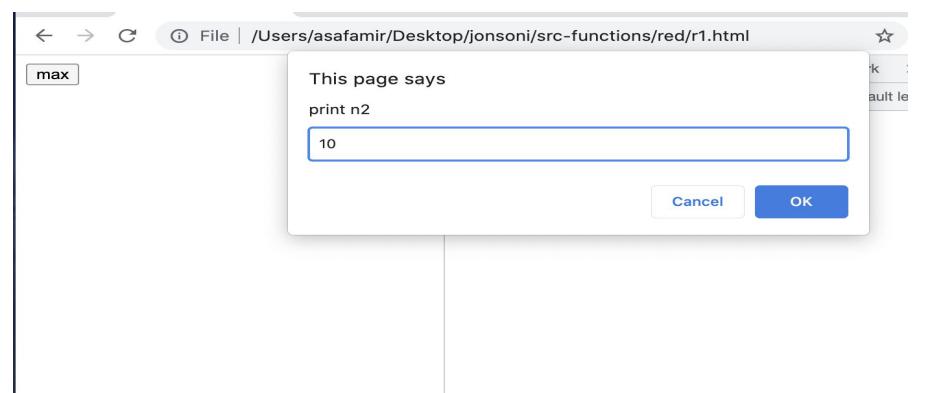
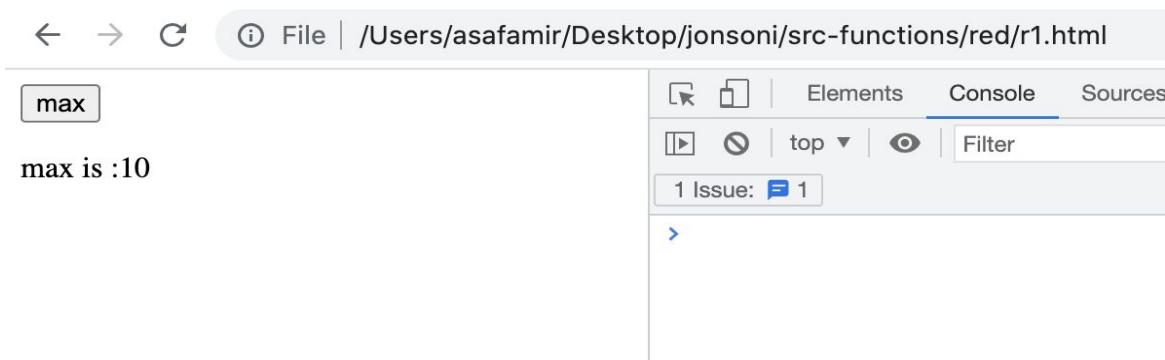
ב. הוסיף כפטור עם הכיתוב max .find max

.a. הוסיף פסקאות עם max = id

.b. כתוב סקריפט שמקבל 3 מספרים מה prompt .prompt

.c. הסקריפט יציג על id max (עליר להשתמש בפונקציה מחלק א').

.d. קרא לפונקציה על ידי לחריצה על כפטור.



166

עשה זאת בעצמך 2

א. כתוב פונקציית **expression** שמקבלת 3 מספרים ומחזירה את המספר המקסימלי.

```
{} function maxBetween3Numbers(n1, n2, n3)
```

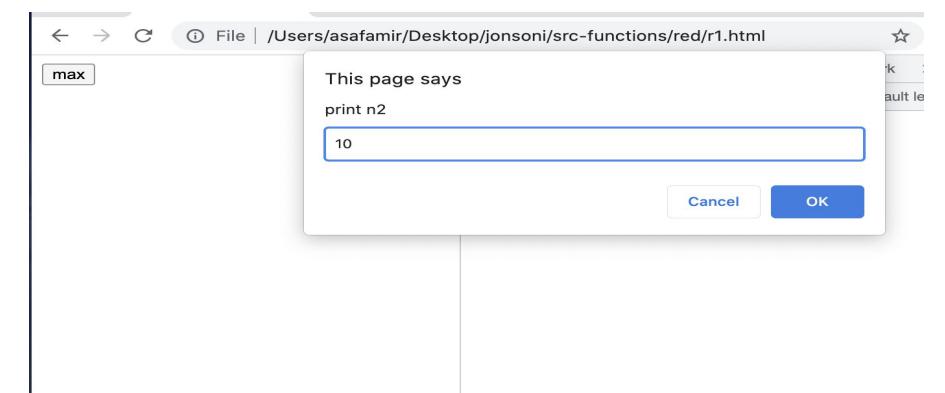
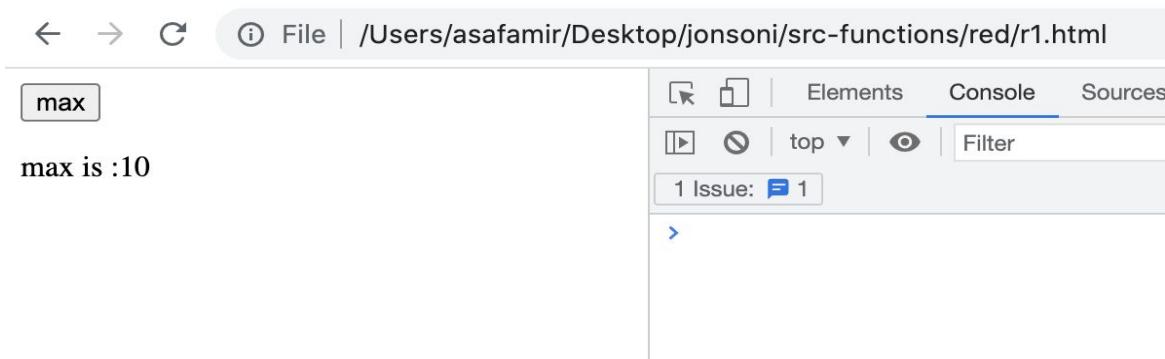
ב. הוסף כפטור עם הכיתוב `max`.

.a. הוסיף פסקאות עם `max = id`

.b. כתוב סקריפט שמקבל 3 מספרים מה `prompt`.

.c. הסקריפט יציג על `max = id` המספר המקסימלי (עליר להשתמש בפונקציה מחלק א').

.d. קרא לפונקציה על ידי לחיצה על כפטור.



עשה זאת בעצמך 3

א. כתוב פונקציית **arrow** שמקבלת 3 מספרים ומחזירה את המספר המקסימלי.

```
{} function maxBetween3Numbers(n1, n2, n3)
```

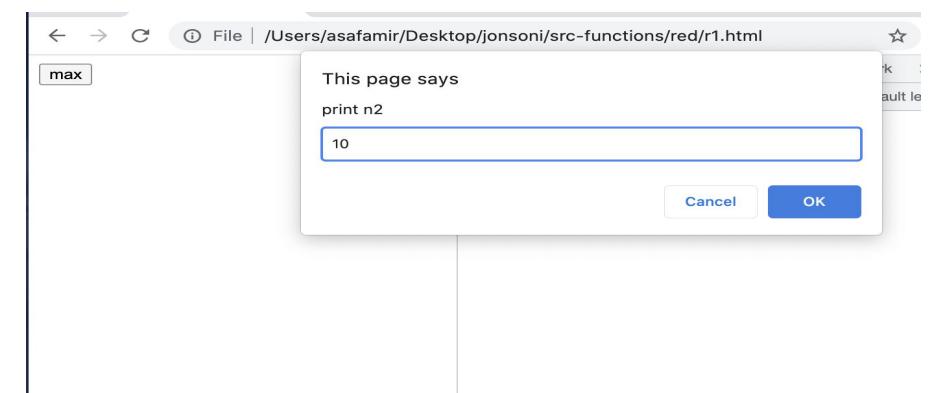
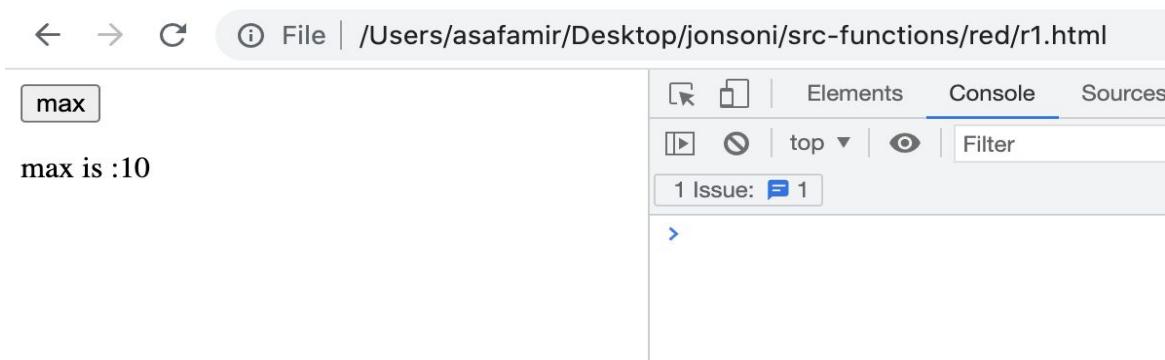
ב. הוסיף כפטור עם הכיתוב `max`.

.a. הוסיף פסקאות עם `max = id`

.b. כתוב סקריפט שמקבל 3 מספרים מה `prompt`.

.c. הסקריפט יציג על `max = id` המספר המקסימלי (עליר להשתמש בפונקציה מחלק א').

.d. קרא לפונקציה על ידי לחריצה על כפטור.



- A. כתוב פונקציית Declaration שמקבלת מערך ותחזיר את המספר המקורי.
- B. הפעיל את הפונקציה מ-A באמצעות `arr=[1,6,2,9]` ובצע `alert` על המספר המקורי.

- A. כתוב פונקציה **expression** שמקבלת מערך ומציגה בהתראה את המספר המרבי.
- B. קרא לפונקציה **m-A** באמצעות `[1,6,2,9]=arr` ובצע **alert** על המספר המקורי

- A. כתוב פונקציית **arrow** שמקבלת מערך ומציגה בהתראה את המספר המרבי.
- B. קרא לפונקציה מ-A באתריות `[1,6,2,9]` alert arr ובצע על המספר המלא.
- C.

- A. כתוב פונקציה **declaration** שמקבלת מערך ומציגה בהתראה את המספר המרבי.
- B. קרא לפונקציה מ-A באמצעות `[1,6,2,9]=arr` ובצע `alert` על המספר המקורי

- A. כתוב פונקציה **expression** המתקבלת מערך ומספר. הפונקציה מחזירה true אם המספר קיים במערך.
- B. קרא לפונקציה m-A באמצעות `[1,6,2,9] arr=` ו`-6 = num`, והתריע(`num` קיים במערך)

- A. כתוב פונקציה **arrow** המקבלת מערך ומספר. הפונקציה מחזירה true אם המספר קיים במערך.
- B. קרא לפונקציה מ-A באמצעות `[1,6,2,9] arr=[1,6,2,9]` והתՐיע(`num`) אם `num` קיים במערך

- A. כתוב פונקציית **declaration** מקבלת מערך ומספר. הפונקציה מחזירה true אם המספר קיים במערך.
- B. קרא לפונקציה מ-A באמצעות [arr=[1,6,2,9] ו- num = 6, והתריע(alert) אם num קיים במערך

OBJECT

JAVASCRIPT



Object

1. JavaScript is designed on a simple object-based paradigm.
2. An object is a collection of properties, and a property is an association between a name (or *key*) and a value.
3. A property's value can be a function, in which case the property is known as a method.
4. In addition to objects that are predefined in the browser, you can define your own objects.
5. This chapter describes how to use objects, properties, functions, and methods, and how to create your own objects.

Object

- JavaScript מתוכנן על פרדיגמה פשוטה מבוססת אובייקטים.
- אובייקט הוא אוסף של מאפיינים, ומאפיין הוא שיר בין שם (או מפתח) לערך.
- ערך מאפיין יכול להיות פונקציה.
- בנוסף לאובייקטים המוגדרים מראש בדף, אתה יכול להגדיר אובייקטים משלך.
- פרק זה מתאר כיצד להשתמש באובייקטים, מאפיינים, פונקציות, וכי怎ן ליצור אובייקטים משלך.

Object and properties

לאובייקט JavaScript יש מאפיינים המשווים אליו. ניתן להסביר תכונה של אובייקט כמשתנה שמוצמד לאובייקט. מאפייני אובייקט זהים בעצם למשתני JavaScript רגילים, למעט הקובץ המצוরף לאובייקטים. המאפיינים של אובייקט מגדירים את המאפיינים של האובייקט. אתה ניגש למאפיינים של אובייקט באמצעות סימן נקודה פשוט:

`objectName.propertyName`

Object

```
<html>
  <body>
    <script>
      var player = new Object();
      // לא חובה

      player.name = "Mike";
      player.level = 3;
      player.points = "8";
      console.log(player);

    </script>
  </body>
</html>
```

כמו כל משתני JavaScript, גם שם אובייקט (שיכול להיות משתנה רגיל) וגם שם המאפיין הם תלויי רישיות. אתה יכול להגדיר מאפיין על ידי הקצתה לו ערך. לדוגמה, בואו ניצור אובייקט בשם שחקן וניתן לו מאפיינים בשם make, name, level ונקודות באופן הבא:

Object

```
<html>
  <body>
    <script>
      let player = {
        name: "Mike",
        Level: 3,
        Points: 8
      }
      console.log(player);
    </script>
  </body>
</html>
```

כמו כל משתני JavaScript, גם שם אובייקט (שיכול להיות משתנה רגיל) וגם שם המאפיין הם תלויי רישיות. אתה יכול להגדיר מאפיין על ידי הקצתה לו ערך. לדוגמה, בואו ניצור אובייקט בשם שחקו וניתן לו מאפיינים בשם make, name, level ונΚודות באופן הבא:

Object

הדוגמה שלמעלה יכולה להיבוט גם באמצעות object initializer, שהוא רשימה מופרדת בפסיק של אפס או יותר זוגות של שמות מאפיינים וערכים משוייכים של אובייקט, המוקפת בסוגרים מסווגלים ({}):

מאפיינים לא מוקצים של אובייקט הם undefined (and not null).

player.Color; // undefined

```
<html>
  <body>
    <script>
      let player{
        name : "Mike";
        level : 3;
        points : "8";
      };
      console.log(player);
    </script>
  </body>
</html>
```

Object

Pack all attributes under one entity.

ניתן לגשת למאפיינים של אובייקט JavaScript או להגדיר אותם גם באמצעות סימון סוגרים (לפרטים נוספים ראה [property accessors](#)).

אובייקטים נקראים לעיתים מרכיבים אסוציאטיביים, מכיוון שכל מאפיין משoir לערך מחרוזת שניית להשתמש בו כדי לגשת אליו.

לדוגמה, תוכל לגשת למאפיינים של אובייקט myCar באופן הבא:

```
player["name"] = "Mike";  
player["level"] = 3;  
player["points"] = "8";
```

Creating new objects

ל-JavaScript יש מספר אובייקטים מוגדרים מראש. בנוסף, אתה יכול ליצור אובייקטים משלך. אתה יכול ליצור אובייקט באמצעות [object initializer](#). אובייקט תחליה תוכל ליצור פונקציית בנאי ולאחר מכן ליצור אובייקט המפעיל את הפונקציה הזו בשילוב עם האופרטור החדש.

התחבר של אובייקט באמצעות [object initializer](#) הוא:

```
var obj = { property_1: value_1, // property_# may be an
identifier...
            2:                 value_2, // or a number...
            // ....,
            'property n': value_n }; // or a string
var myHonda = {color: 'red', wheels: 4, engine: {cylinders: 4,
size: 2.2}};
```

Creating new objects

הדוגמה הבאה יוצרת את `myHonda` עם שלושה מאפיינים. שימוש לב שמאפיין המנוע הוא גם אובייקט עם מאפיינים משלו.

```
var myHonda = {color: 'red', wheels: 4, engine: {cylinders: 4, size: 2.2}};
```

יצירת אובייקטים חדשים - באמצעות קונסטרוקטור

לחלופין, ניתן ליצור אובייקט בשני שלבים הבאים:

1. הגדר את סוג האובייקט על ידי כתיבת פונקציית בנאי. ישנה מוסכמה חזקה, עם סיבה טובה, להשתמש באוט ריאשונית גדולה.
2. צור מופע של האובייקט עם חדש.

כדי להגדיר סוג אובייקט, צור פונקציה עבור סוג האובייקט המציינת את השם, המאפיינים והשיטות שלו. לדוגמה, נניח שאתה רוצה ליצור סוג אובייקט עבור מכוניות. אתה רוצה שסוג זה של אובייקט יקרא מכונית, ואתה רוצה שהוא שי含ן לו מאפיינים כמו דגם ו שנה. כדי לעשות זאת, תכתב את הפונקציה הבאה:

```
function Car(make, model, year) {  
    this.make = make;  
    this.model = model;  
    this.year = year;  
}
```

שימוש לב שימושו של this כדי להקנות ערכים למאפייני האובייקט על סמך הערכים שהועברו לפונקציה.

יצירת אובייקטים חדשים - באמצעות קונסטרוקטור

```
function Car(make, model, year) {  
    this.make = make;  
    this.model = model;  
    this.year = year;  
}
```

שימוש לב לשימוש `this` כדי להקנות ערכים למאפייני האובייקט על סמך הערכים שהועברו לפונקציה.

עתה אתה יכול ליצור אובייקט בשם `mycar` באופן הבא:

```
var mycar = new Car('Eagle', 'Talon TSi', 1993);
```

הצירה זו יוצרת את `mycar` ומקצתה לה את הערכים שציינו עבור המאפיינים שלה. `mycar.year` הוא המספר השלם 1993, וכן הלאה.

יצירת אובייקטים חדשים - באמצעות קונסטרוקטור

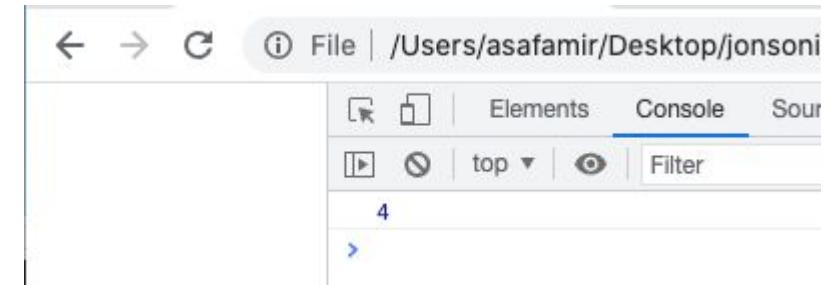
אתה יכול ליצור כל מספר של אובייקטי מכוניות על ידי קריאות `new`. לדוגמה

```
var kenscar = new Car('Nissan', '300ZX', 1992);  
var vpgscar = new Car('Mazda', 'Miata', 1990);
```

Object + function

```
<html>
  <body>
    <script>
      var player = {
        name: "Asaf",
        level: 3,
        points: 8,
        updateLevel : function() {
          this.level++;
          return this.level;
        }
      }

      console.log(player.updateLevel());
    </script>
  </body>
</html>
```



עשה זאת בעצמך 1

ויקט باسم מחשב ותן לו מאפיינים בשם מחיר, צבע ו זיכרון.

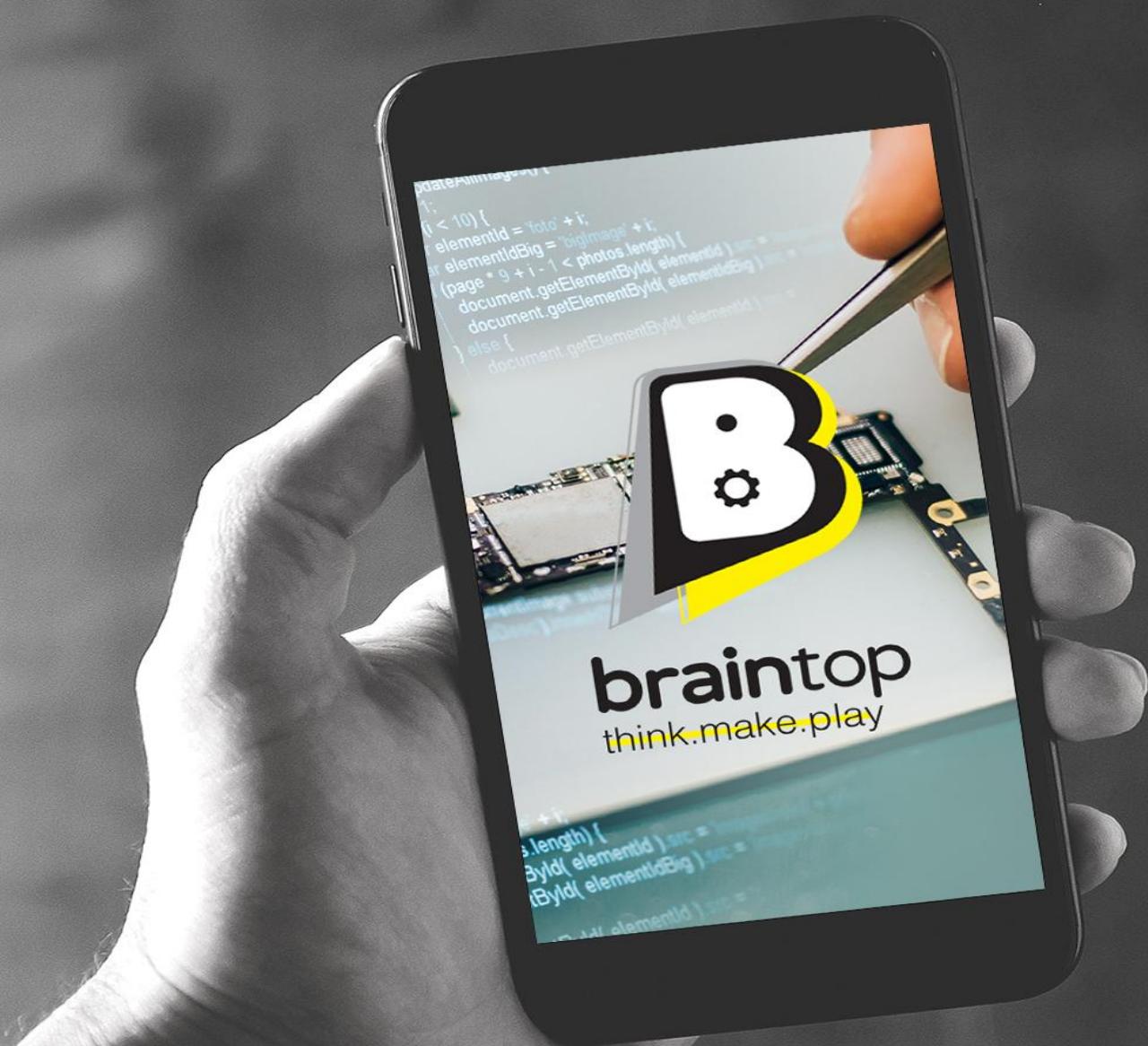
עשה זאת בעצמך 2

1. צור תבנית(מחלקה, פונקציה) בשם Person באמצעות בנאן ותן לו מאפיינים בשם פרטי שם, שם משפחה, גיל.
2. צור 2 מופעים של האובייקט
 - a. מופע ראשון 1c: מייקל, ג'ורדן, 49
 - b. מופע שני 2c : לארה, 74, bili

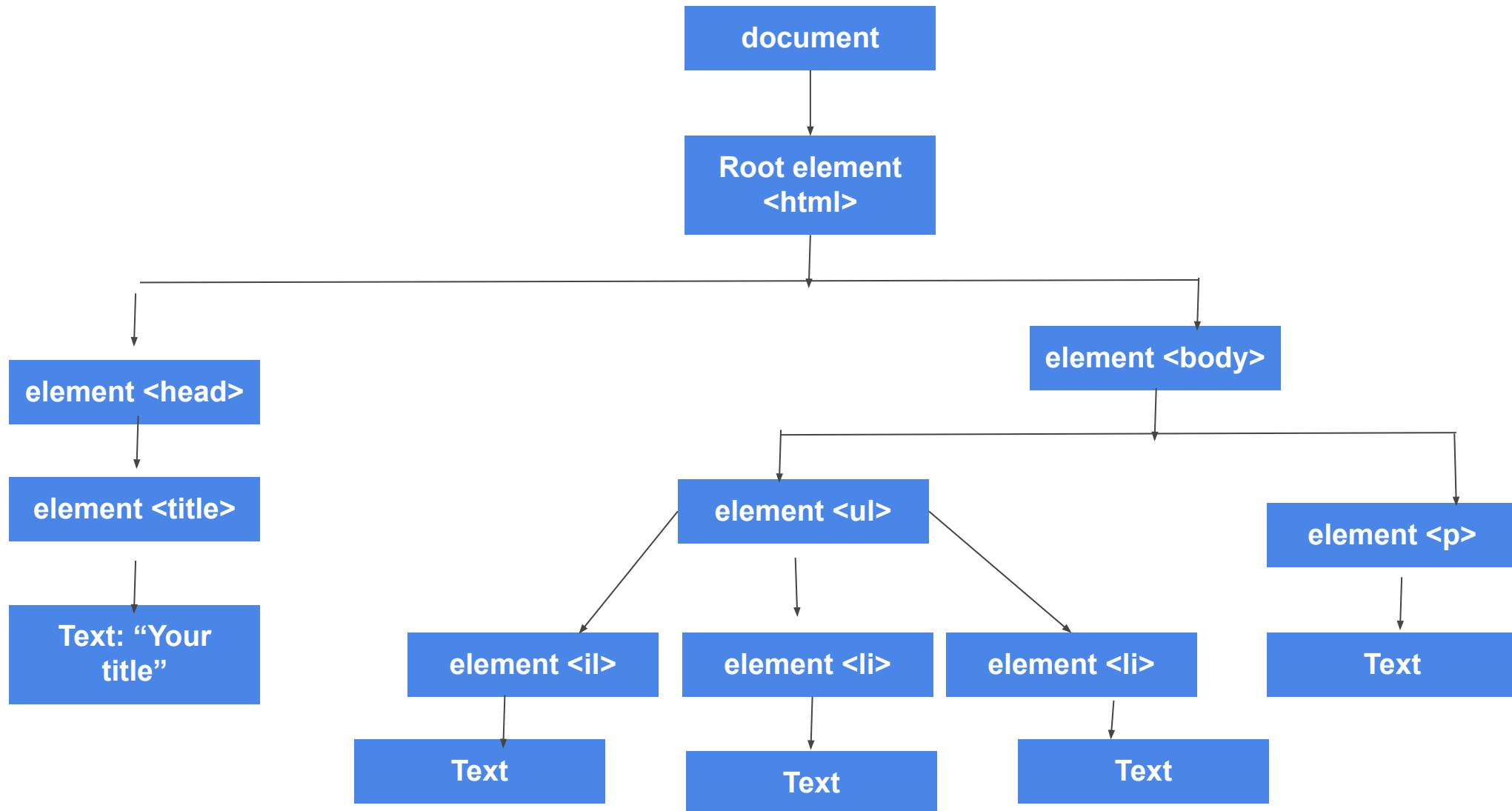
**בפרק הבא נמשיר ללמידה עוד נושאים על אובייקטים, ונקציות
ומרכיבים. בפרק הבא נעבור לדיאן בדום.**

DOM

JAVASCRIPT



DOM



DOM (Document, Object, Model)

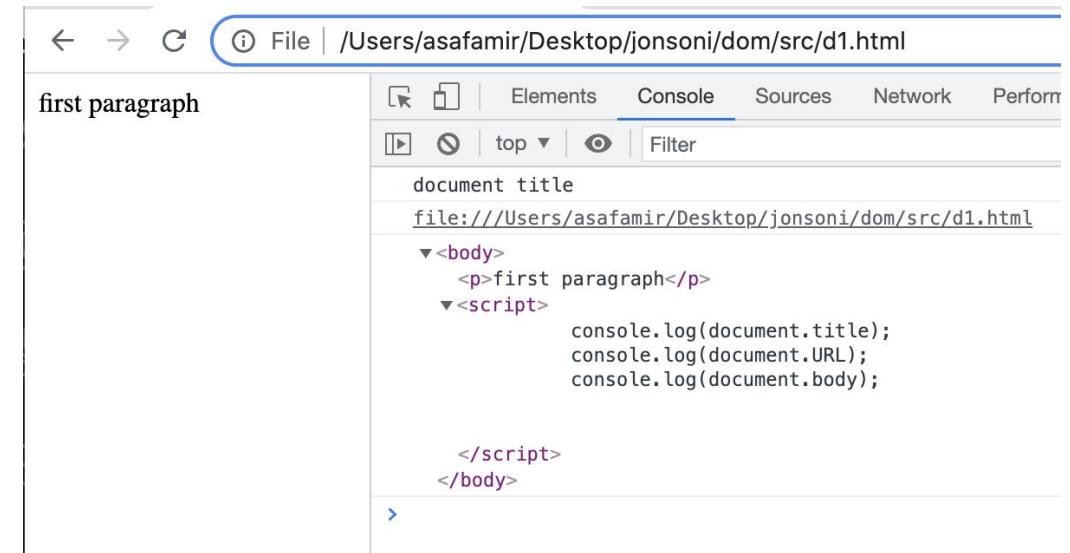
מהו document ? הדף עצמו

דוגמאות ל objects

h1 is an object, page is an object , p is an object

דוגמאות לתוכנות document

- 1.) `document.title; // document title`
- 2.) `document.body;// The body element`
- 3.) `document.URL; // The document url`



The screenshot shows the browser's developer tools open to the 'Console' tab. The URL in the address bar is `/Users/asafamir/Desktop/jonsoni/dom/src/d1.html`. The console output displays the results of the three code snippets:

```
document title
file:///Users/asafamir/Desktop/jonsoni/dom/src/d1.html
<body>
  <p>first paragraph</p>
  <script>
    console.log(document.title);
    console.log(document.URL);
    console.log(document.body);

  </script>
</body>
```

פונקציות שיש לobjects:

- 1.) getTitle
- 2.) getLastParagraph
- 3.) getAllelements

דוגמאות לפונקציות חשובות

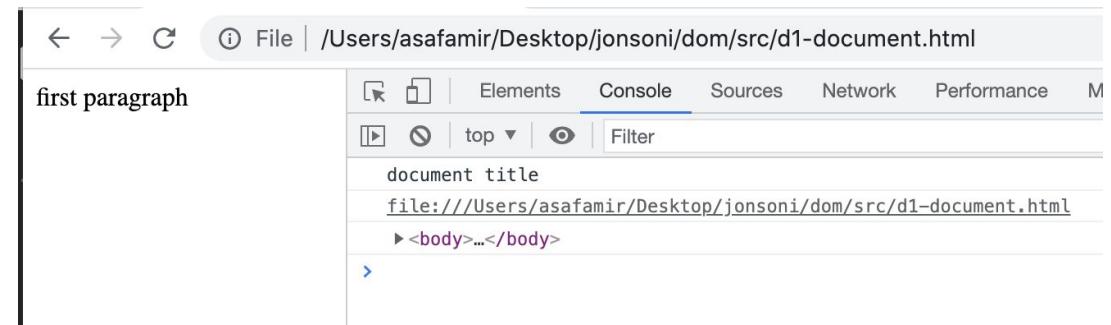
- 4.) getElementById("id");
- 5.) getElementByClassName("css class");
- 6.) getElementByTagName("html tag");
- 7.) getElementsByTagName

Functions

Name of function	Description
<code>getElementById</code>	The Document method getElementById() returns an Element object representing the element whose <code>id</code> property matches the specified string
<code>querySelector</code>	The Document method querySelector() returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.
<code>querySelectorAll</code>	The Document method querySelectorAll() returns a static (not live) NodeList representing a list of the document's elements that match the specified group of selectors.
<code>Node.removeChild</code>	The <code>Node.removeChild()</code> method removes a child node from the DOM and returns the removed node.
<code>addEventListener</code>	The EventTarget method addEventListener() sets up a function that will be called whenever the specified event is delivered to the target.
<code>getElementsByName</code>	The <code>getElementsByName</code> method of Document interface returns an HTMLCollection of elements with the given tag name.
<code>parentNode</code>	<code>parentNode</code> is the parent of the current node. The parent of an element is an Element node, a Document node, or a DocumentFragment node.

Document

```
<html>
  <head>
    <title>document title</title>
  </head>
  <body>
    <p>first paragraph</p>
  </body>
<script>
  console.log(document.title);
  console.log(document.URL);
  console.log(document.body);
</script>
</html>
```

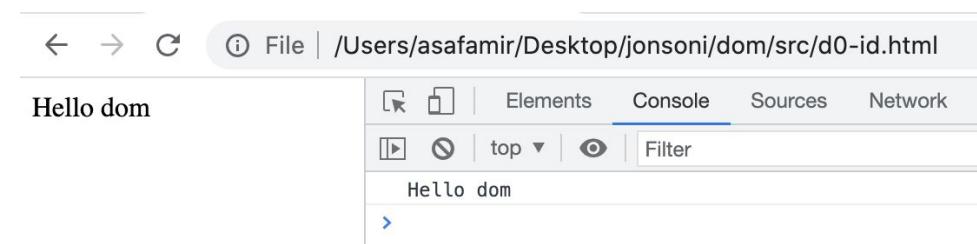


getElementById

```
var element=document.getElementById("p1");
```

We get a reference to the element and can change it. Or go to its children.

```
<html>
  <head>
    <title>document title</title>
  </head>
  <body>
    <p id="p1">Hello dom</p>
  </body>
<script>
  console.log(document.getElementById("p1").innerText);
</script>
</html>
```



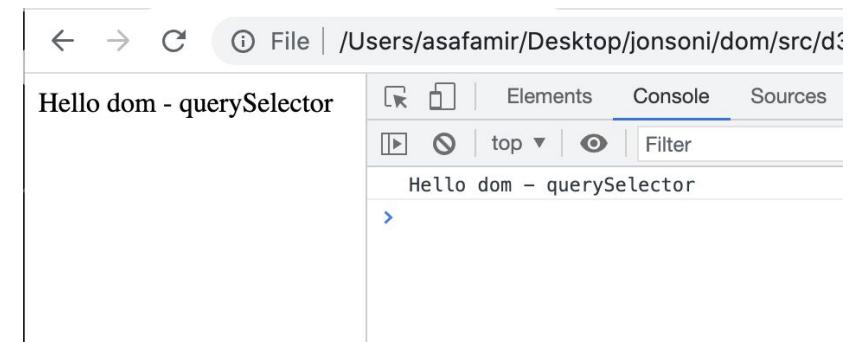
200

querySelector

```
var element=document.querySelector("#p1");
```

We get a reference to the element and can change it. Or go to its children.

```
<html>
  <head>
    <title>document title</title>
  </head>
  <body>
    <p id ="p1">Hello dom - querySelector</p>
  </body>
<script>
  console.log(document.querySelector("#p1").innerText);
</script>
</html>
```

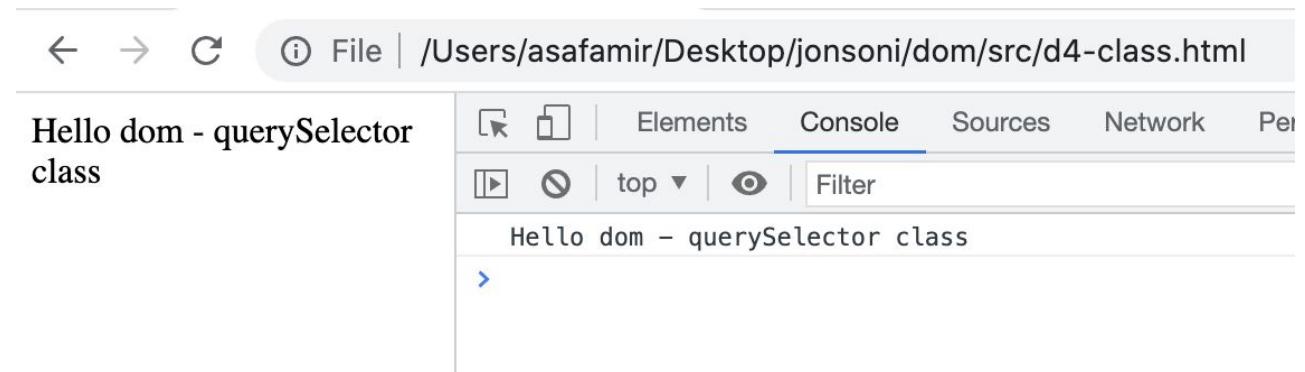


querySelector **class**

```
var element=document.querySelector(".p1");
```

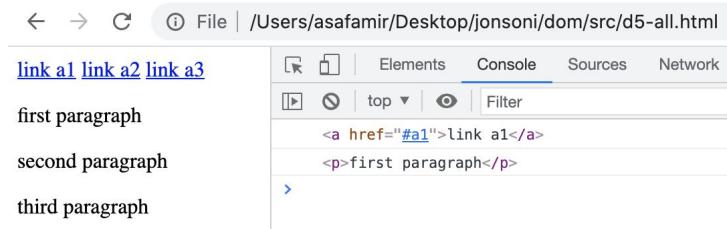
We get a reference to the element and can change it. Or go to its children.

```
<html>
  <head>
    <title>document title</title>
  </head>
  <body>
    <p class="p1">Hello dom - querySelector class </p>
  </body>
<script>
  console.log(document.querySelector(".p1").innerText);
</script>
</html>
```



First Element / all Elements

```
<html>
  <head>
    <title>document title</title>
  </head>
  <body>
    <div class="main-nav">
      <a href="#a1">link a1</a>
      <a href="#a2">link a2</a>
      <a href="#a3">link a3</a>
    </div>
    <div class="post-content">
      <p>first paragraph</p>
      <p>second paragraph</p>
      <p>third paragraph</p>
    </div>
  </body>
<script>
  console.log(document.querySelector(".main-nav a")); //only first a element
  var arr = document.querySelectorAll(".post-content p"); //array of all paragraphs
  console.log(arr[0]); //first paragraph
</script>
</html>
```



First Element Matching Specified Selector

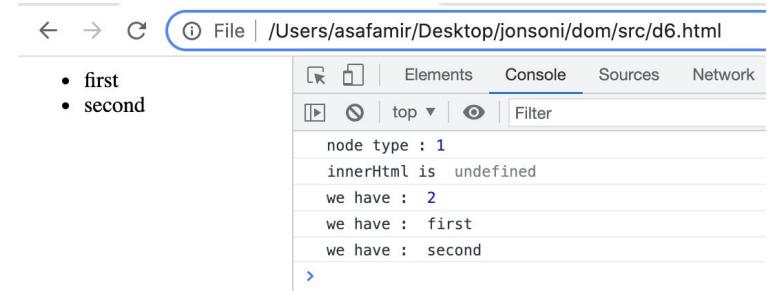
Some examples of getting the first element to match a specified selector.

1.) `document.querySelector(".main-nav a"); //get first a element`

2.) `document.querySelectorAll(".post-content p"); // get all paragraph`

getElementById / getElementByTagName

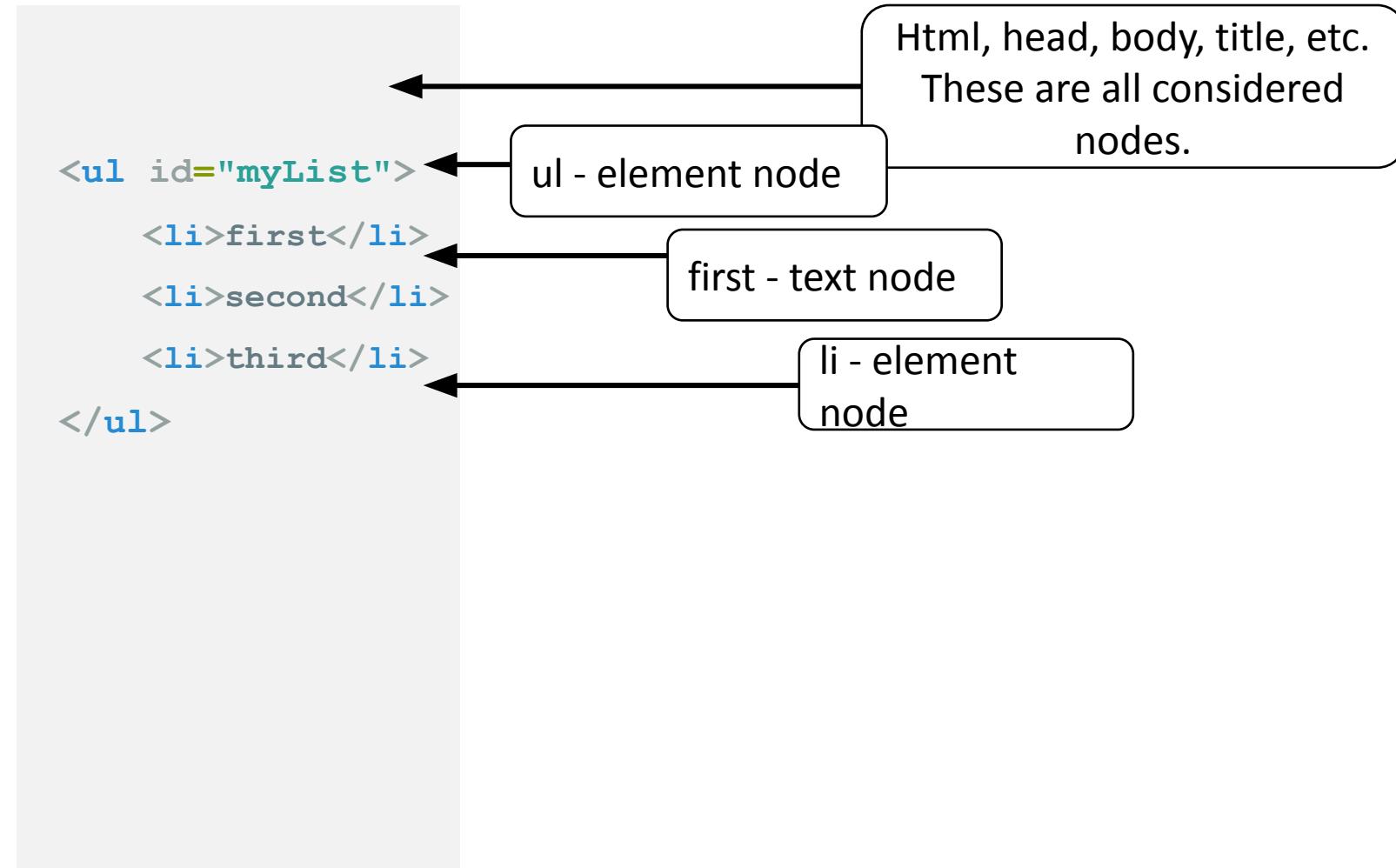
```
<html>
  <head>
    <title>getElementById / getElementsByTagName </title>
  </head>
  <body>
    <ul id="navlinks">
      <li>first</li>
      <li>second</li>
    </ul>
  </body>
<script>
  var element=document.getElementById("navlinks");
  var myLiItems=document.getElementsByTagName("li"); //return array
  console.log("node type :",element.nodeType);//1
  console.log("innerHTML is ",myLiItems.innerHTML);//undefined
  console.log("we have : ",myLiItems.length);//2
  console.log("we have : ",myLiItems[0].innerText);//first
  console.log("we have : ",myLiItems[1].innerText);//second
</script>
</html>
```



getElementByTagName

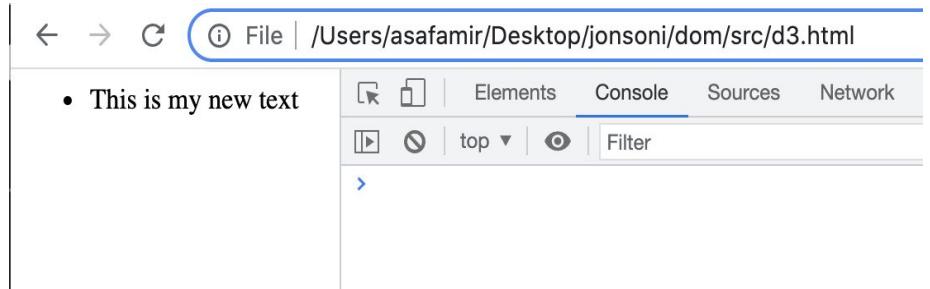
```
var myLiItems=document.getElementsByTagName("li"); //return array
console.log("node type :",element.nodeType);//1
console.log("innerHTML is ",myLiItems.innerHTML); //undefined
console.log("we have : ",myLiItems.length);//2
console.log("we have : ",myLiItems[0].innerText); //first
console.log("we have : ",myLiItems[1].innerText); //second
```

Node and Elements



Create a DOM element

```
<html>
  <head>
    <title>Immediately function</title>
  </head>
  <body>
    <ul id="myul"></ul>
    <script>
      let newLi=document.createElement("li");
      newLi.innerText = "This is my new text";
      let element=document.getElementById("myul");
      element.appendChild(newLi);
    </script>
  </body>
</html>
```



- 1.) Create the Element
- 2.) Add element to document

Add Class/ Remove class

```
<html>
  <head>
    <title>add class</title>
    <style>
      .hidden{
        display: none;
      }
    </style>
  </head>
  <body>
    <button onclick="hide()">hide paragraph</button>
    <button onclick="show()">show paragraph</button>
    <p class="p1">Hi, 🍀</p>
    <script>
      function show(){
        let p1=document.querySelector(".p1").classList.remove("hidden");
      }
      function hide(){
        let p1=document.querySelector(".p1").classList.add("hidden");
      }

    </script>
  </body>
</html>
```

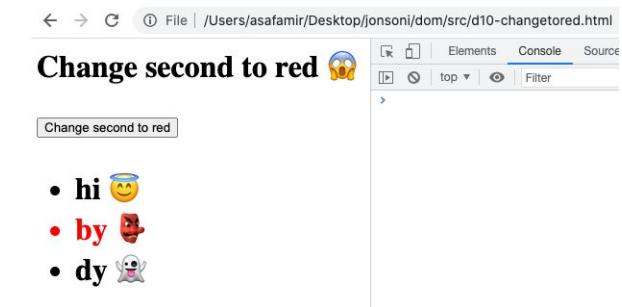


Do it yourself 1

Given the following html page. Add to the page a function called **changeSecondItemToRed** that change the color of the second item in the list to red.

```
<html>
  <body>
    <h1>Change second to red 😱</h1>
    <button onclick="changeSecondItemToRed()">Change second to red</button>
    <ul id="myFriends">
      <li>hi 😊</li>
      <li>by 🦸</li>
      <li>dy 🧟</li>
    </ul>
    <script>
      function changeSecondItemToRed() {
      }
    </script>
  </body>
</html>
```

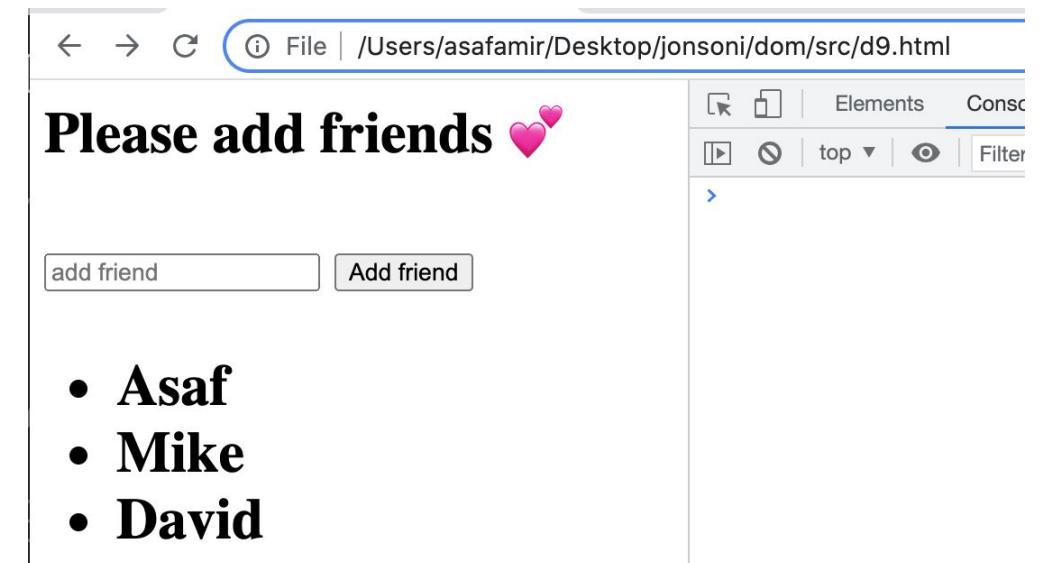
(Hint: **read about .style and solve with it**)



Do it Yourself 2

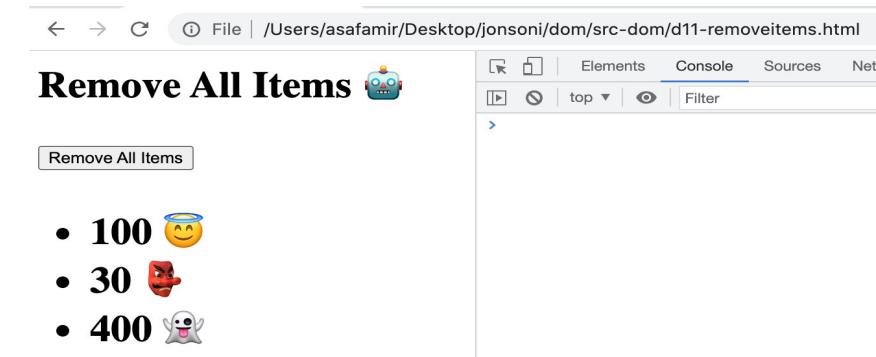
Given the following html page. Add to the page a function called `addFriend` that adds to the list (`ul`) when the user inputs

```
<html>  
  <body>  
    <h1>Please add friends ❤</h1>  
    <input type="text" id="friend" placeholder="add friend">  
    <button onclick="addFriend ()">Add friend</button>  
    <ul id="myFriends"></ul>  
    <script>  
      function addFriend(){  
      }  
    </script>  
  
  </body>  
</html>
```



Remove items from dom

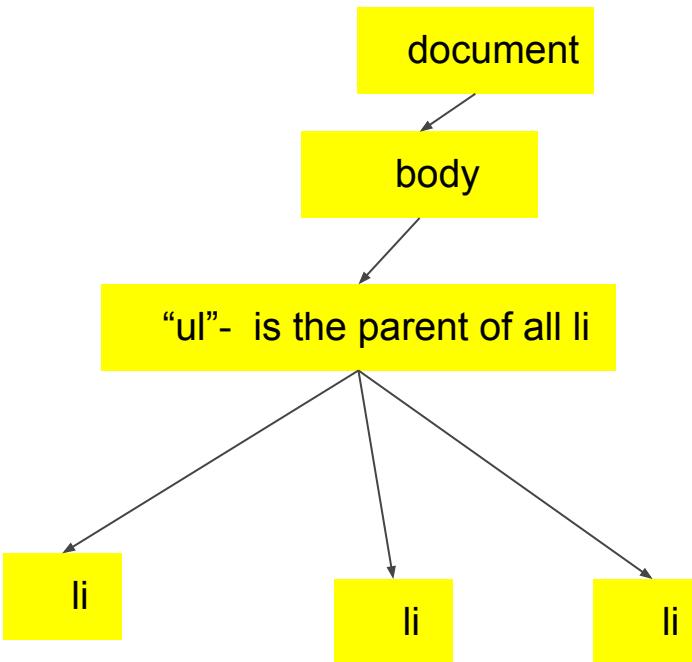
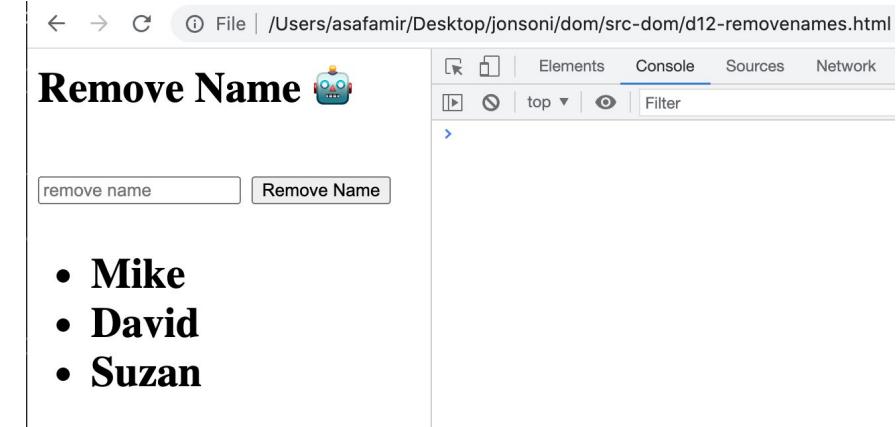
```
<html>
  <body>
    <h1>Remove All Items <img alt="robot icon" style="vertical-align: middle;"></h1>
    <button onclick="removeAllItems ()">Remove All
      Items</button>
    <ul id="items">
      <li>100 <img alt="angel emoji" style="vertical-align: middle;"></li>
      <li>30 <img alt="demon emoji" style="vertical-align: middle;"></li>
      <li>400 <img alt="ghost emoji" style="vertical-align: middle;"></li>
    </ul>
    <script>
      function removeAllItems () {
        var list = document.querySelectorAll ('#items li');
        for (var i=0; li=list[i]; i++) // you can use length {
          li.parentNode.removeChild (li);
        }
      }
    </script>
  </body>
</html>
```



Do it Yourself 3

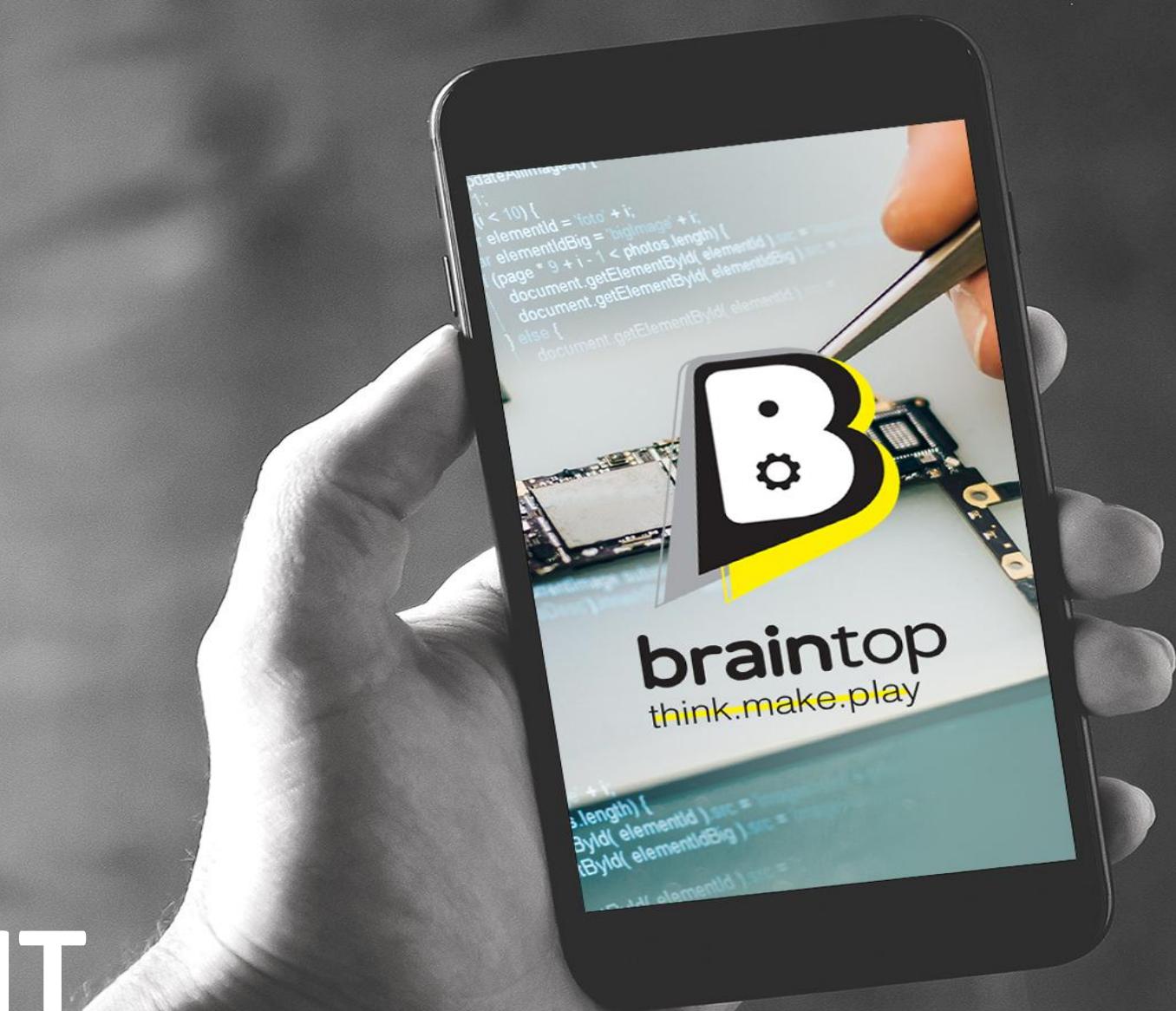
Given the following html page. Add to the page a function called `removeFriend`. The function remove items from list according to user inputs.

```
<html>
  <body>
    <h1>Remove Name 🤖</h1>
    <input type="text" id="name" placeholder="remove name">
    <button onclick="removeNames ()">Remove Name</button>
    <ul id="items">
      <li>Mike</li>
      <li>David</li>
      <li>Suzan</li>
    </ul>
    <script>
      function removeNames () {
      }
    </script>
  </body>
</html>
```



EVENT

JAVASCRIPT



אירוע הוא פעולה (או התרחשות) המתרחשת במערכת שאתה מתכונת.

כasher אנו מתייחסים לאינטרנט, אירועים מופעלים בתוך חלון הדפדפן, ונוטים להיות מחוברים לפריט ספציפי השוכן בו - זה עשוי להיות אלמנט בודד, קבוצת אלמנטים, מסגר HTML שנטען בכרטיסיה הנוכחית, או את כל חלון הדפדפן. ישנים סוגים רבים ו שונים של אירועים שיכולים להתרחש. לדוגמה:

- המשמש בוחר אלמנט מסוים או מרחף את הסמן מעל אלמנט מסוים.
- המשמש בוחר מקש במקלדת.
- המשמש משנה את גודלו או סגור את חלון הדפדפן.
- דף אינטרנט מסתיים בטעינהו.
- מוגש טופס.
- סרטון מופעל, מושהה או מסתיים.
- מתרחשת שגיאה.

More examples of events: <https://developer.mozilla.org/en-US/docs/Web/Events>

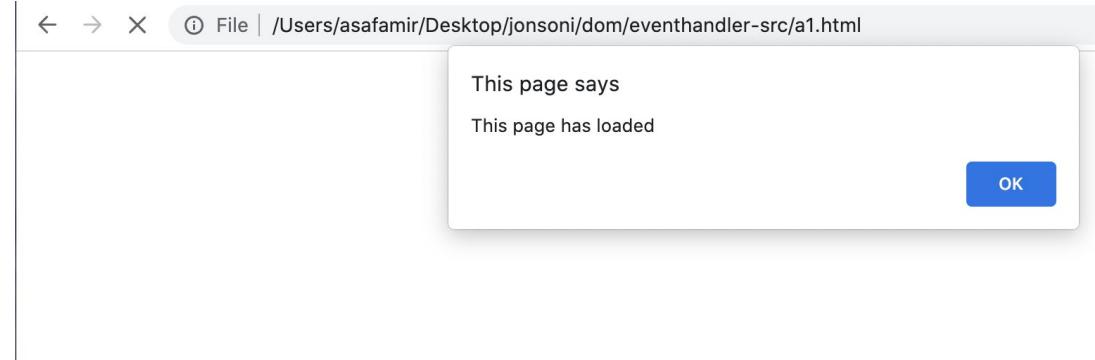
Event	Description
onclick	Click on an element
onmouseover	When the cursor comes over an element
onmouseout	The user moves the mouse away from an HTML element
Onblur	Events related to elements gaining and losing focus
Onfocus	The onfocus event occurs when an element gets focus
onchange	An HTML element has been changed
onkeydown	The user pushes a keyboard key
onkeyup	The onkeyup event occurs when the user releases a key (on the keyboard).
onload	The browser has finished loading the page

כמה אירועים אחרים יכולים לכלול:

- 1.) אירוע מדיה - אודיו/וידאו.
- 2.) אירוע התקדמות - מעקב אחר דפדף מתמשך.
- 3.) אירוע מעבר CSS - התחלת/הפעלה/סיום של מעבר.

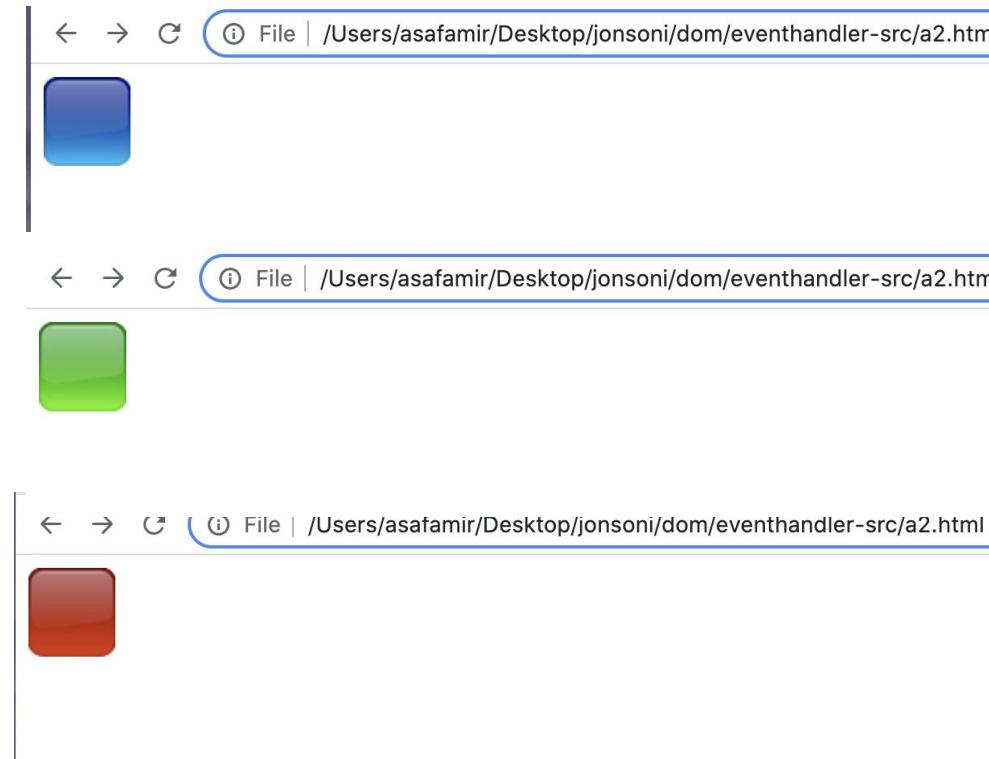
Event Handler Onload

```
<!DOCTYPE >  
  
<html>  
  <body onload="alert('This page has loaded');">  
    here is my text  
  </body>  
</html>
```



כמה דרכים לקוד אירועים

```
<!DOCTYPE >
<html>
<body>
    
</body>
<script>
    var img=document.getElementById("imgId");
    img.onclick= function(){
        img.src="images/green.png";
    };
    img.onmouseover= function(){
        img.src="images/red.png";
    };
</script>
</html>
```



כמה דרכים לקוד אירועים

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    </body>
    <script>
      function myFunction () {
        alert ("hello myFunction");
      }
      document .addEventListener ('click',myFunction );
    </script>
  </html>
```



onBlur and onFocus

```
<html>
<body>
<button id = "btn1" onfocus="myOnFocus (this)"
onblur="myOnBlur (this)">onfocus/onblur</button>
<script>
function myOnFocus (obj) {
    obj.style.background="orange";
}
function myOnBlur (obj) {
    obj.style.background="yellow";
}
</script>
</body>
</html>
```



כasher anu lochzim uli - onFocus

כasher anu ozzbim at tibat hataktot l'mashl - onBlur

1. בtag'ת של ה element

```
<div id="div1" onchange="f1()" class="btn">this is green div</div>
```

2. בצורה דינמית

```
function divFunction(event) {
  //alert("hello divFunction");
  document.getElementById("div1").style.background = "yellow";
}
```

```
document.getElementById("div1").addEventListener("change", divFunction);
```

דגש : ה event מועבר בכל מקרה לפונקציה

3. על ידי רפרנס לאלמנט עצמו

```
<script>
var img = document.getElementById("div1");
img.onclick = function () {
  img.src = "images/green.png";
}</script>
```

גם אם הפרמטר event לא יועבר כפרמטר. הוא יוכר בתוך הפונקציה.

```
function divFunction(event) {
  //alert("hello divFunction");
  document.getElementById("div1").style.background =
    "yellow";
```

```
let btn = document.getElementById("btn");  
btn.addEventListener("click", (event) => {  
    changeContent(event, "newContent");  
} );
```

1. בצורה דינמית

```

function divFunction(event) {
  //alert("hello divFunction");
  document.getElementById("div1").style.background = "yellow";
}

document.getElementById("div1").addEventListener("change", divFunction);

```

דגש : ה event מועבר בכל מקרה לפונקציה

גם אם הפרמטר `event` לא יועבר כפרמטר. הוא יוכר בתוך הפונקציה.

```

function divFunction() {
  console.log(event);
  document.getElementById("div1").style.background = "yellow";
}

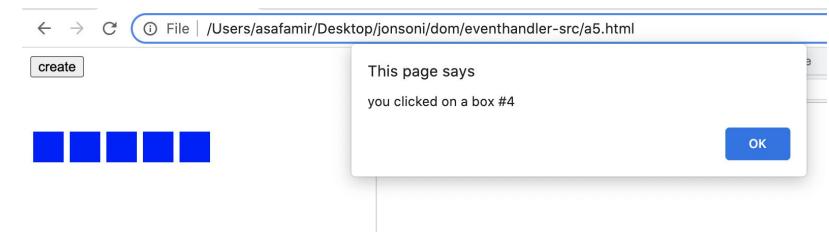
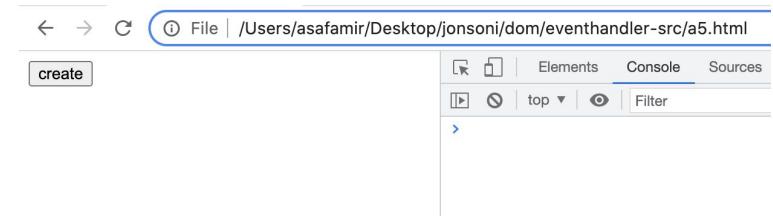
```

1. בצורה דינמית

```
function divFunction(event) {  
    //alert("hello divFunction");  
event.target.style.background = "yellow";  
  
}  
  
document.getElementById("div1").addEventListener("change", divFunction);
```

יצירת כפתורים באופן דינמי

```
<html>
<body>
    <button onclick="createButtons (5,'blue') ">create</button>
    <section style="margin-top: 50px;"></section>
    <script>
        function createButtons (num, color)
        {
            for(let i=0;i<num;i++ ){
                var div = document.createElement ('div');
                div.style.backgroundColor = color;
                div.style.height = "30px";
                div.style.margin="3px";
                div.style.width = "30px";
                div.style.cssFloat = "left";
                div.style.cursor = "pointer";
                div.draggable = "true";
                div.onclick = function (){
                    alert('you clicked on a box #' + i);
                }
                document.getElementsByTagName ('section')[0].appendChild (div);
            }
        }
    </script>
</body>
</html>
```



הדף של קלט משתמש

```
<!DOCTYPE html>
<html lang="en">
<body>
<input placeholder="Enter some text" id="input1" name="name" class="" />
<p id="values"></p>
<script>
const input = document.querySelector("input");
input.addEventListener("input", updateValue);
function updateValue(event) {
  const log = document.getElementById("values");
  console.log(event.target.value);
  let txt = event.target.value;
  input.style.color = txt.length > 10 ? "green" : "red";
  log.textContent = event.target.value;
}
</script>
</body>
</html>
```



Do it yourself 1

Given the following HTML page. Add a function called `blurFunction` to the page. When the user leaves the text box, the background color will change to red.

```
<html>
  <body>
    <h3>Change Background Color of Textbox</h3>
    Enter your name:
    <input type="text" id="myInput"
      onblur="blurFunction(this)">

    <script>
      function blurFunction(obj) {
      }
    </script>
  </body>

</html>
```

← → ⌂ ⌂ File | /Users/asafamir/Desktop/jsons

Change Background Color of Textbox

Enter your name:

← → ⌂ ⌂ File | /Users/asafamir/Desktop/jsons/dom/src-dor

Change Background Color of Textbox

Enter your name:

Do it yourself 2

Given the following HTML page. Add a function called **focusFunction** to the page. When the user is focused on the text box, the background color will change to yellow

```
<html>
  <body>
    <h3>Change Background Color of Textbox</h3>
    Enter your name:
    <input type="text" id="myInput"
onblur="blurFunction(this)" onfocus="focusFunction(this)">
    <script>
      function blurFunction(obj) {
      }
      function focusFunction(obj) {
      }
    </script>
  </body>
</html>
```

← → ⌂ ⓘ File | /Users/asafamir/Desktop/jsons

Change Background Color of Textbox

Enter your name:

← → ⌂ ⓘ File | /Users/asafamir/Desktop/jsons/

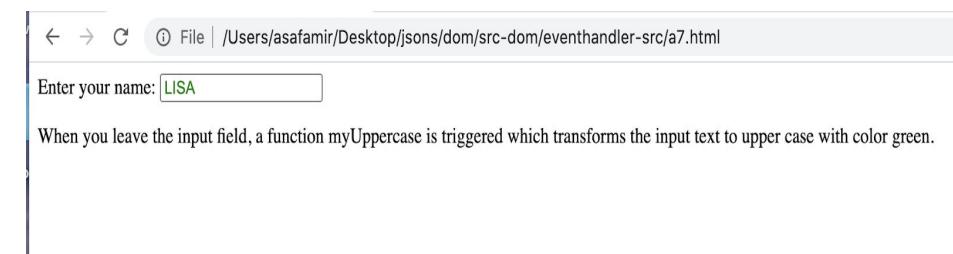
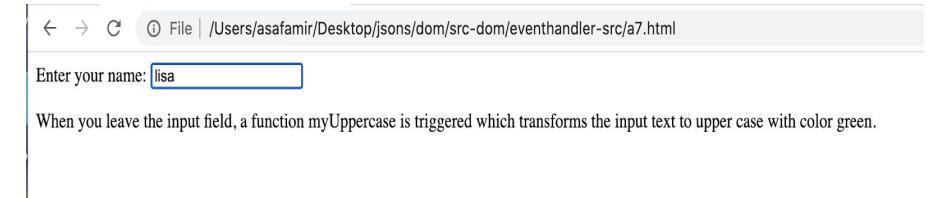
Change Background Color of Textbox

Enter your name:

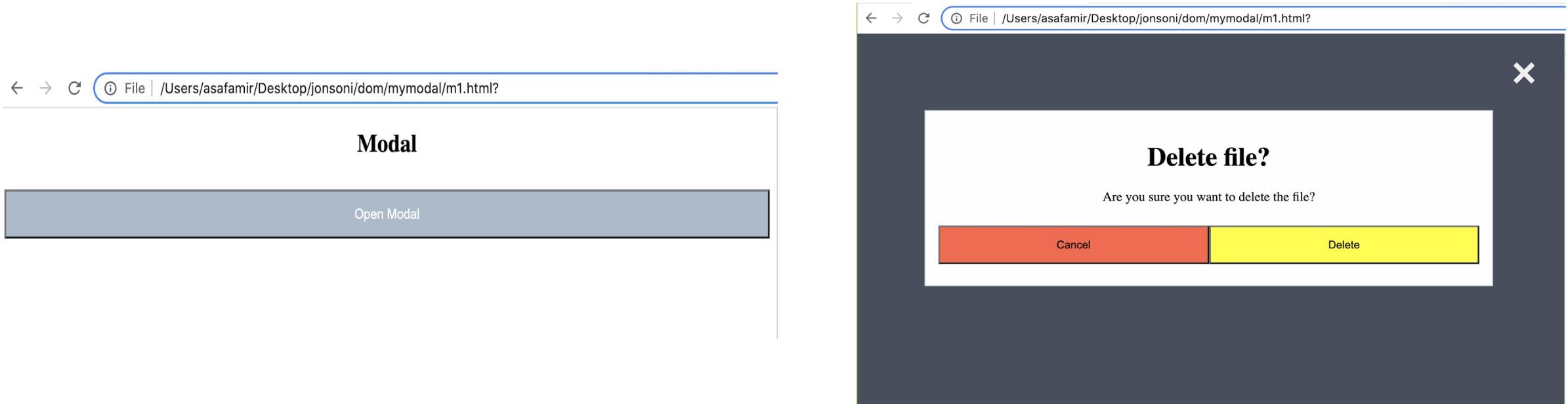
Do it yourself 3

Given the following HTML page. Add a function called myUppercase to the page. When the user leaves the input field, the function myUppercase is triggered which transforms the input text to uppercase with color green.

```
<!DOCTYPE html>
<html>
  <body>
    Enter your name:
    <input type="text" id="firstname" onblur="myUppercase(this)">
    <p>When you leave the input field,
      a function myUppercase is triggered which transforms the
      input text to upper case with color green</p>
    <script>
      function myUppercase(obj)  {
      }
    </script>
  </body>
</html>
```



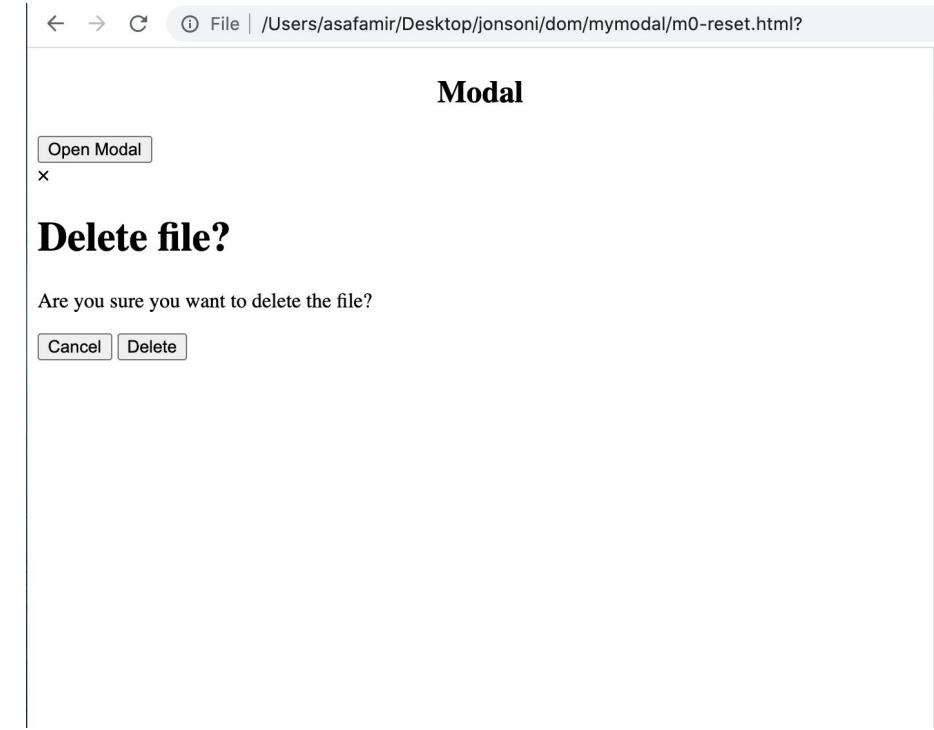
פרויקט דוגמא modal



Modal example reset - html file

```
<!DOCTYPE html>
<html>
<body>
<h2 style="text-align: center;">Modal</h2>
<button class="showModal">Open Modal</button>
<div id="myModal" class="modal">
  <span
    onclick="document.getElementById('myModal').style.display='none'">x</span>
  <form class="modal-content">
    <div class="container">
      <h1>Delete file?</h1>
      <p>Are you sure you want to delete the file? </p>
      <div class="clearfix">
        <button class="btncancel">Cancel</button>
        <button class="btndelete">Delete</button>
      </div>
    </div>
  </form>
</div>

</body>
</html>
```



Modal example reset - Add the following css

```
<style>
* {box-sizing: border-box;}
button {
    background-color: #aabbcc;
    color: white;
    padding: 14px 20px;
    margin: 10px 0;
    cursor: pointer;
    width: 100%;
}
.btncancel, .btndelete {
    float: left;
    width: 50%;
}
.btncancel {
    background-color: tomato;
    color: black;
}
.btndelete {
    background-color: yellow;
    color: black;
}
```

```
.container {
    padding: 16px;
    text-align: center;
}
.modal {
    display: none;
    position: fixed;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: #474e5d;
    padding-top: 50px;
}

/* Modal Content Box */
.modal-content {
    background-color: #fefefe;
    margin: 5% auto 15% auto;
    border: 1px solid #aabbcc;
    width: 80%;
}
```

Modal example reset - Add the following css

```
hr {  
    border: 1px solid #f1f1f1;  
    margin-bottom: 25px;  
}  
/* Close Button (x) */  
.close {  
    position: absolute;  
    right: 35px;  
    top: 15px;  
    font-size: 50px;  
    font-weight: bold;  
    color: #f1f1f1;  
}  
  
.close:hover,  
.close:focus {  
    color: #f44336;  
    cursor: pointer;  
}  
  
/* Clear floats */  
.clearfix::after {  
    content: "";  
    clear: both;  
    display: table;  
}  
</style>
```

דוגמה מודאל שלב 1

```
הכריז על modal var modal = document.getElementById('myModal')  
כתב את הפונקציה () showModal () שמצוירה את הסגנון של  
block modal id=  
כתב את הפונקציה () hideModal () שמצוירה את הסגנון של  
none modal id=
```

דוגמה מודאל שלב 1 - פתרון

הכריז על `var modal` בהתייחס ל-`id=mymodal` 1.

כתב את הפונקציה `showModal()` שמצויגה את הסגנון של `block` 2.

כתב את הפונקציה `hideModal()` שמצויגה את הסגנון של `none` 3.

```
var modal = document.getElementById('mymodal');

function showModal() {
    document.getElementById('mymodal').style.display='block'
}

function hideModal() {
    document.getElementById('mymodal').style.display='none'
}
```

דוגמה מודאל שלב 2

הוסף מאזינים לאירועי klik ללחוץ על פונקציה showModal.

הוסף מאזינים לאירועי klik ללחוץ על פונקציה btnCancel שקורא להצגה hideModal

הוסף מאזיני אירוע klik ללחוץ על פונקציה btndelete שקורא להצגה hideModal

דוגמה מודאל שלב 2 - פתרון

הוסף מאזינים לאיורוּי קליָק ללחצֵן class=showModal הקורא לפונקציה hideModal
הוסף מאזינים לאיורוּי קליָק ללחצֵן class=btnCancel שקורא לפונקציה hideModal
הוסף מאזיני איורוּי קליָק ללחצֵן class=btndelete שקורא לפונקציה hideModal

```
var modal = document.getElementById('myModal');

document.querySelector(".showModal").addEventListener("click",showModal)
document.querySelector(".btncancel").addEventListener("click",hideModal)
document.querySelector(".btndelete").addEventListener("click",hideModal)

function showModal() {
    document.getElementById('myModal').style.display='block'
}

function hideModal() {
    document.getElementById('myModal').style.display='none'
}
```

דוגמה מודאל שלב 3

הוסף מאזינים אונימיים לאירועי הקלדה לтиיעוד, שמקש האזנה ללחיצה מטה. כאשר המשתמש לוחץ על Escape, המודאל ייסגר.

דוגמה מודאל שלב 3 - פתרון

הוסף מאזינים אונוניים לאירוע הקלדה לтиיעוד, שמקש האזנה להחיצה מטה. כאשר המשתמש לוחץ על Escape, המודאל ייסגר.

```
document.addEventListener ('keydown', function (e) {  
  if (e.key === 'Escape') {  
    hideModal ();  
  }  
}) ;
```

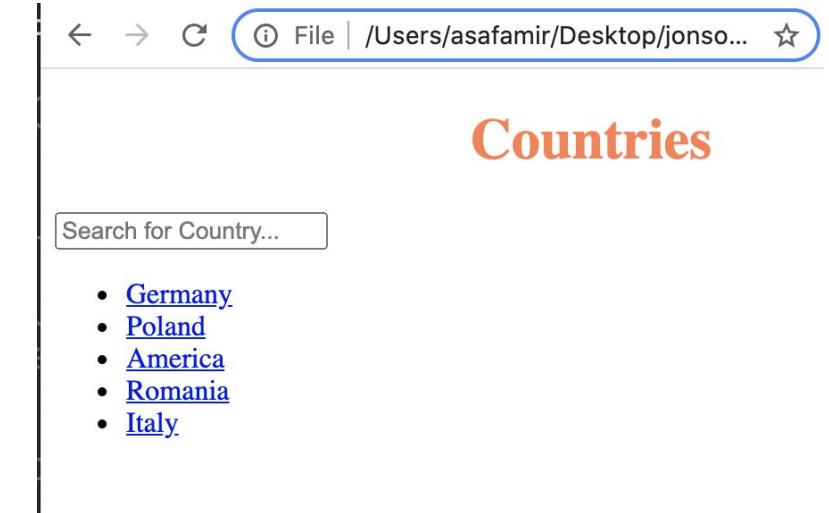
חיפוש פרויקט דוגמה

search



Search example reset - html file

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<h1 style="color:coral;text-align:center;">Countries</h1>
<input type="text" id="userInput" onkeyup="search ()"
placeholder="Search for Country..." title="Type
Country">
<ul id="countryUl">
<li><a href="#">Germany</a></li>
<li><a href="#">Poland</a></li>
<li><a href="#">America</a></li>
<li><a href="#">Romania</a></li>
<li><a href="#">Italy</a></li>
</ul>
</body>
</html>
```



הבא-CSS הופף את ה-Search example reset

```
<style>
* {
  box-sizing: border-box;
}

#userInput {
  background-position: 10px 12px;
  background-repeat: no-repeat;
  width: 100%;
  font-size: 16px;
  padding: 12px 20px 12px 40px;
  border: 1px solid rgb(46, 8, 218);
  margin-bottom: 20px;
}

#countryUl {
  list-style-type: none;
  padding: 0;
  margin: 0;
}

#countryUl li a {
  border: 1px solid rgb(40, 5, 236);
  margin-top: -1px;
  background-color: #c1e90d;
  padding: 12px;
  text-decoration: none;
  font-size: 18px;
  color: black;
  display: block
}

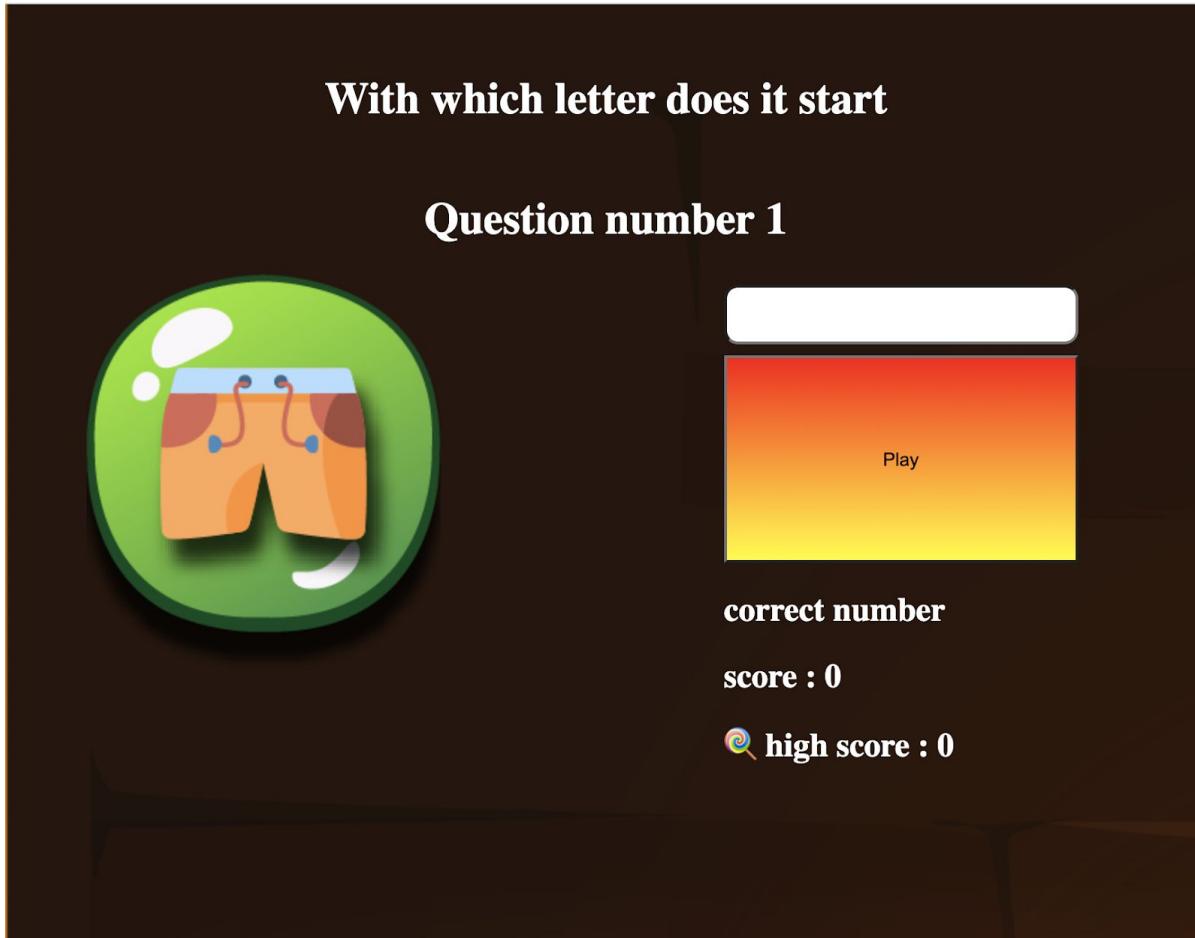
#countryUl li a:hover:not (.header) {
  background-color: #ee0b0b;
}
</style>
```

דוגמה לCHIPOSH JS

```
<script>
function search() {
    var input, filter, ul, li, a, i, value;
    input = document.getElementById("userInput");
    filter = input.value.toUpperCase();
    ul = document.getElementById("countryUl");
    li = ul.getElementsByTagName("li");
    for (i = 0; i < li.length; i++) {
        a = li[i].getElementsByTagName("a")[0];
        value = a.textContent || a.innerText;
        if (value.toUpperCase().indexOf(filter) > -1) {
            li[i].style.display = "";
        } else {
            li[i].style.display = "none";
        }
    }
}
</script>
```

Mini project 1

With which letter does it start



Before we start : Random numbers

```
<html>
  <body>
    <script>
      function getRandomInt(max) {
        return Math.floor(Math.random() * max);
      }
      console.log(getRandomInt(3));
      // expected output: 0, 1 or 2
      console.log(getRandomInt(1));
      // expected output: 0
      console.log(Math.random());
      // expected output: a number from 0 to <1
    </script>
  </body>
</html>
```

Mini 1 project reset - html file

```
<html>
  <body class="backGroundBody" >
    <h1>With which letter does it start </h1>
    <h1 class="countQuestion">Question number 1</h1>
    <div class="imgGuess">
      
    </div>
    <div class="input">
      <input type="text" id="input" >
      <button class="btn play" >Play</button>
    </div>
    <div class="output">
      <h2 class="message">correct number </h2>
      <h2 class="score">score : 0</h2>
      <h2 class="highscore">🔍 high score : 0 </h2>
    </div>
    <div class="playAgain hidden" >
      <button>Game over-lets play again </button>
    </div>
  </body>
</html>
```

File | /Users/asafamir/Desktop/jonsoni/dom/hat-game/a0-reset1.html

With which letter does it start

Question number 1



Play

correct number

score : 0

🔍 high score : 0

Game over-lets play again

Mini project reset - Add the following css

<style>

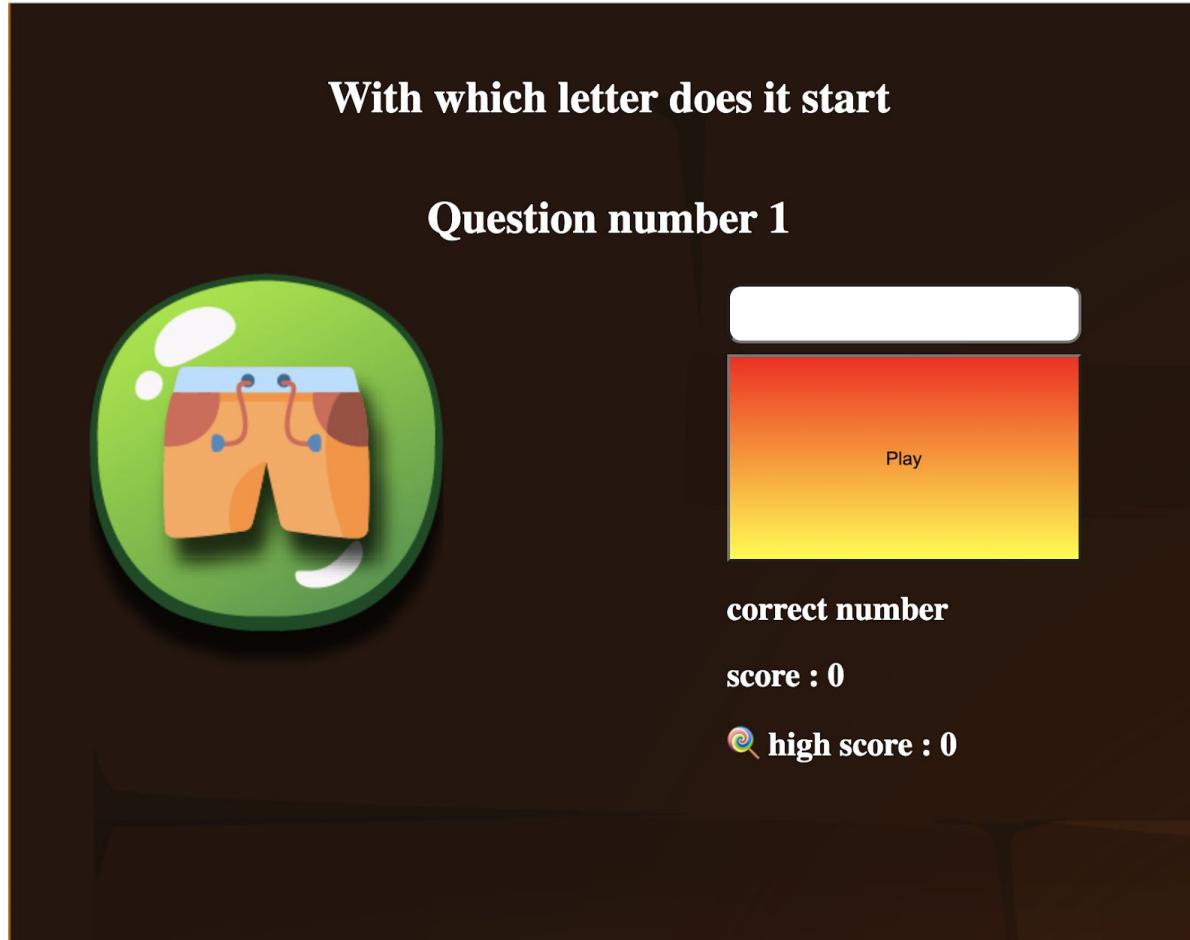
```
body {  
    background-color: #de840f;  
}  
.backGroundBody{  
    background-image: url("images/back.png");  
}  
h1{  
    text-align: center;  
    color: white;  
    margin-top: 50px;  
}  
img{  
    margin-left: 10%;  
}  
.input{  
    float :left;  
    display: block;  
    width: 30%;  
}  
.output{  
    float :left;  
    display: block;  
    color:white;  
    width: 30%;  
  
}  
.imgGuess{  
    float :left;  
    display: block;  
    color:white;  
    width: 60%;  
}
```

```
.imgGuess img{  
    width: 50%;  
}  
.btn{  
    width: 100%;  
    height: 150px;  
    background-image: linear-gradient(red,  
yellow);  
}  
input[type=text] {  
    width: 100%;  
    padding: 12px 20px;  
    margin: 8px 0;  
    box-sizing: border-box;  
    border-radius:10px ;  
}  
.playAgain{  
    float: left;  
    width: 100%;  
    display: block;  
    margin-left: 45%;  
}
```

Mini project reset - Add the following css

```
.playAgain button {  
    margin: 0;  
    position: absolute;  
    left: 50%;  
    background-image: linear-gradient (rgb(9, 255, 0), yellow);  
    -ms-transform: translate(-50%, -50%);  
    transform: translate(-50%, -50%);  
    width: 30%;  
    height: 60px;  
    margin-top: 10px;  
    margin-bottom: 10px;  
}  
.hidden {  
    display: none;  
}  
</style>
```

Mini project phase 2 - After adding css



250

הוראות מיני פרויקט 1

פתחו קובץ HTML בשם mini.html ונוסיפו לו את קוד ה-HTML וה-CSS
פתחו תיקייה בשם תМОנות ושים בה את 3 התמונות הללו.
הגדיר 5 משתנים גלובליים בסקריפט

```
var arr = ["images/hat.png", "images/boat.png", "images/short.png"];  
var arrWord = ["hat", "boat", "shirt"];  
var counter = 0;  
var i = selectRandomImage();  
var score = 0;  
var highScore = 0;  
document.body.classList.add('backGroundBody');  
let btnPlayAgain = document.querySelector('.playAgain');
```

הוראות מיני פרויקט 2

כתב פונקציה בשם `selectRandomImage`, הפונקציה תעשה את הפעולות הבאות:

1. תנסה את הטקסט ב-`countQuestion` ל"מספר שאלה" + מונה (`counter`)
2. קיבל מספר אקראי בין 0 ל-3 למשתנה שנקרא אינדקס
3. מציג את התמונה ממערך `arrWord` באlement שהזיהה ה `id` שלו הוא `imgToGuess`
4. הפונקציה מחזירה את האינדקס למשתנה הגלובלי בשם `var`
5. רענן את העמוד מספר פעמים ובדקו אם התמונה משתנה.

מינি פרויקט הוראה 2 - פתרון

```
<script>
    var arr = ["images/hat.png", "images/boat.png", "images/short.png"];
    var arrWord = ["hat", "boat", "shirt"];
    var counter = 0;
    var i = selectRandomImage();
    var score = 0;
    var highScore = 0;
    document.body.classList.add('backGroundBody');
    let btnPlayAgain = document.querySelector('.playAgain');

    function selectRandomImage() {
        document.querySelector(".countQuestion").textContent= "Question number " + counter
        let index = Math.floor(Math.random() * 10%3);
        console.log(index);
        document.getElementById("imgToGuess").src = arr[index];
        return index;
    }
</script>
```

הוראות מיני פרויקט 3

כתב פונקציה בשם **play()** הפונקציה תעשה את הפעולות הבאות:

1. Increment counter by 1
2. If counter <= 3
 - a. Declare let word = arrWord[i] // arrWord and i are globals var
 - b. Declare let letter = document.getElementById("input").value; // user input value
 - c. If !letter than document.querySelector(".message").textContent = "No Letter 😠"
 - i. And decrement to the user 2 points
 - d. Else If letter==word[0] document.querySelector(".message").textContent = "correct Answer 🤝"
 - i. increment the user 5 points
 - e. Else If letter!=word[0] document.querySelector(".message").textContent = "not correct Answer 🤢"
 - i. decrement to the user 3 points
 - f. Reset value of document.getElementById("input").value to empty ("")
 - g. Display correct score on document.querySelector(".score").textContent
 - h. If counter < 3 than i = selectRandomImage();
3. If counter > 2 {} // we will write on the next phases
4. Add event listener click to button play () below the play function
document.querySelector(".play").addEventListener('click', play);

מיני פרויקט הוראת 3 - פתרון

```
function play() {
    counter++;
    if(counter<=3) {
        let word = arrWord[i];
        var letter = document.getElementById("input").value;
        //var imgLetter = document.getElementById("imgToGuess").na
        if(!letter){
            document.querySelector(".message").textContent = "No Letter 🤪"
            score-=2;
        }
        else if(letter == word[0]) {
            document.querySelector(".message").textContent = "correct Answer 🎉"
            score+=5;
        }
        else if(letter != word[0]) {
            document.querySelector(".message").textContent = "correct Answer 🤢"
            score-=2;
        }
        document.getElementById("input").value="";
        document.querySelector(".score").textContent = "score : " + score;
        if(counter<3) {
            i = selectRandomImage();
        }
    }
    if(counter>2) {
    }
}
//event listener
document.querySelector(".play").addEventListener('click', play);
```

Add function
play and see
the solution

הוראות מיני פרויקט 4

Add 2 function : **showbtnPlayAgain,hiddenbtnPlayAgain** above **selectRandomImage ()**

```
//hidden and show button game over play again

const showbtnPlayAgain = function () {
    btnPlayAgain.classList.remove('hidden');
};

const hiddenbtnPlayAgain = function () {
    btnPlayAgain.classList.add('hidden');
};
```

הוראות מיני פרויקט 5

Add game over function : **function gameOver()**

1. If score>highScore then save highScore and display it on
document.querySelector(".highscore").textContent with "🍭 high score " +
score
2. Remove class backGroundBody with
document.body.classList.remove('backGroundBody');
3. Call showbtnPlayAgain();
***Call the function gameover on play function in the if statement - if counter > 2**



פתרון מיני פרויקט הוראה 5

```
function gameOver() {  
    if(score>highScore) {  
        highScore= score;  
        document.querySelector(".highscore").textContent="🔍 high score " + score;  
    }  
    document.body.classList.remove('backGroundBody');  
    showbtnPlayAgain();  
}
```

Call the function gameover on play function in the if statement - `if counter > 2`

```
if(counter>2){  
}  
|
```

```
if(counter>2){  
    gameOver()  
}  
|
```

הוראות מיני פרויקט 6

Add to playAgain selector EventListener click that call to function reset

```
document.querySelector(".playAgain").addEventListener('click', reset);
```

Add reset function : **function** function reset()

1. document.body.classList.add('backGroundBody');
2. Counter = 0
3. i = selectRandomImage() // get new random picture
4. Reset score to 0
5. document.getElementById("input").value="";
6. hiddenbtnPlayAgain();

הראות מיני פרויקט 6

```
//reset game and play again

document.querySelector(".playAgain").addEventListener('click', reset);

function reset() {

    document.body.classList.add('backGroundBody');

    counter = 0;

    i = selectRandomImage(); // get new random picture

    score = 0;

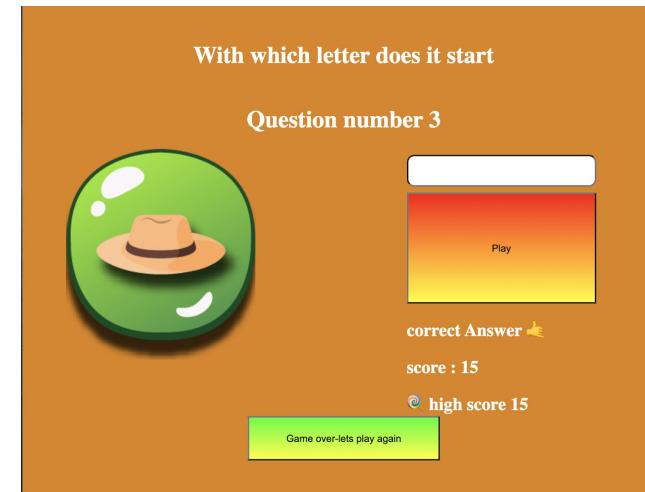
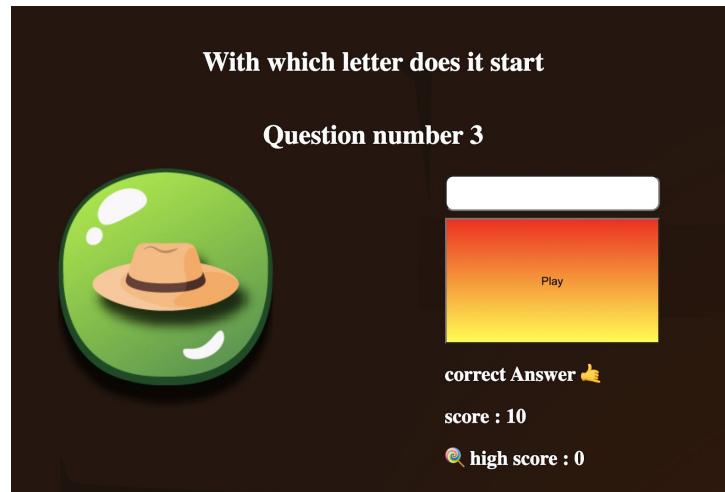
    document.querySelector(".score").textContent =0;

    document.getElementById("input").value="";

    hiddenbtnPlayAgain();

}
```

מיני פרויקט הרץ את הפרויקט :)



DATA STRUCTURE AND ALGORITHMS

JAVASCRIPT



Destructuring assignment array 1

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

```
<html>
```

```
<body>
```

```
<script>
```

```
let a, b;  
[ a, b ] = [ 1, 2 ];  
console.log(a);  
// expected output: 1  
console.log(b);
```

```
</script>
```

```
</body>
```

```
</html>
```

Destructuring assignment array 2

```
<html>
  <body>
    <script>
      let a, b;
      [a, b] = [1, 2];
      console.log(a); // 1
      console.log(b); // 2

      [a, b] = [1, 2, 3, 4, 5];
      console.log(a); // 1
      console.log(b); // 2

      ({ a, b } = { a: 1, b: 2 });
      console.log(a); // 1
      console.log(b); // 2

    </script>
  </body>
</html>
```

Destructuring assignment array 3

```
<html>
  <body>
    <script>
      //Basic variable assignment
      const foo = ['one', 'two', 'three'];
      const [red, yellow, green] = foo;
      console.log(red); // "one"
      console.log(yellow); // "two"
      console.log(green); // "three"

    </script>
  </body>
</html>
```

Destructuring assignment array 4

```
<html>
  <body>
    <script>
      //Assignment separate from declaration
      //A variable can be assigned its value via destructuring,
      //separate from the variable's declaration.

      let a, b;
      [ a, b ] = [ 1, 2 ];
      console.log(a); // 1
      console.log(b); // 2

    </script>
  </body>
</html>
```

Destructuring assignment array 5

```
<html>
  <body>
    <script>
      //Default value
      //A variable can be assigned a default,
      //in the case that the value unpacked from the array is undefined.
      let a, b;
      [ a=5, b=7 ] = [ 1 ];
      console.log(a); // 1
      console.log(b); // 7

    </script>
  </body>
</html>
```

Destructuring assignment array 6

```
<html>
  <body>
    <script>
      //Swapping variables
      let a = 1;
      let b = 3;

      [a, b] = [b, a];
      console.log(a); // 3
      console.log(b); // 1

      const arr = [1,2,3];
      [arr[2], arr[1]] = [arr[1], arr[2]];
      console.log(arr); // [1,3,2]

    </script>
  </body>
</html>
```

Destructuring assignment array 7

```
<html>
  <body>
    <script>
      //Parsing an array returned from a function
      //It's always been possible to return an array from a function. Destructuring can make
working with an array return value more concise.
      //In this example, f() returns the values [1, 2] as its output, which can be parsed in
a single line with destructuring.
      function f() {
        return [1, 2];
      }
      let a, b;
      [a, b] = f();
      console.log(a); // 1
      console.log(b); // 2
    </script>
  </body>
</html>
```

Destructuring assignment array 8

```
<html>
  <body>
    <script>
      //Ignoring some returned values
      function f() {
        return [1, 2, 3];
      }
      const [a, , b] = f();
      console.log(a); // 1
      console.log(b); // 3
      const [c] = f();
      console.log(c); // 1
    </script>
  </body>
</html>
```

Destructuring assignment object 1

```
<html>
  <body>
    <script>
      const user = {
        id: 42,
        isVerified: true
      };

      const {id, isVerified} = user;

      console.log(id); // 42
      console.log(isVerified); // true
    </script>
  </body>
</html>
```

Destructuring assignment object 2

```
<html>
  <body>
    <script>
      //Assignment without declaration
      let a, b;
      ({ a, b } = { a: 1, b: 2 });
    </script>
  </body>
</html>
```

Destructuring assignment object 3

```
<html>
  <body>
    <script>
      // Assigning to new variable names
      const o = {p: 42, q: true};
      const {p: foo, q: bar} = o;
      console.log(foo); // 42
      console.log(bar); // true
    </script>
  </body>
</html>
```

Destructuring assignment object 4

```
<html>
  <body>
    <script>
      // Default values
      const {a = 10, b = 5} = {a: 3};
      console.log(a); // 3
      console.log(b); // 5
    </script>
  </body>
</html>
```

Destructuring assignment object 5

```
<html>
  <body>
    <script>
      // Unpacking fields from objects passed as a function parameter
      const user = {
        id: 30,
        displayName: 'Raz',
        fullName: {
          firstName: 'Mike',
          lastName: 'Laris'
        }
      };
      function userId({id}) {
        return id;
      }

      function whois({displayName, fullName: {firstName: name}}) {
        return `${displayName} is ${name}`;
      }
      console.log(userId(user)); // 30
      console.log(whois(user)); // "Raz is Mike"
    </script>
  </body>
</html>
```

The spread operator array (...) 1

Spread syntax (...) allows an iterable such as an array expression or string to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.

```
<html>
  <body>
    <script>
      let arr = [30, 40, 50];
      let oldWayArr = [5, 6, arr[0], arr[1], arr[2]];
      console.log(oldWayArr);

      let newWayArr = [10, 20, ...arr];
      console.log(newWayArr);
      console.log(...newWayArr);
      console.log(10, 20, 30, 40, 50);

    </script>
  </body>
</html>
```

The spread operator array (...) 2

```
<html>
  <body>
    <script>
      let arr = [30, 40, 50];
      // Copy array
      const arrCopy = [...arr];
      console.log(arrCopy) //30 40 50
    </script>
  </body>
</html>
```

The spread operator array (...) 3

```
<html>
  <body>
    <script>
      let arr1 = [1,2,3]
      let arr2 = [4,5,6]
      // Join 2 arrays
      let arrJoin = [...arr1, ...arr2];
      console.log(arrJoin);
    </script>
  </body>
</html>
```

The spread operator array (...) 4

```
<html>
  <body>
    <script>
      // Iterables: arrays, strings, maps, sets. NOT for objects
      let strContinental = 'Europe';
      let letters = [...strContinental, ' ', 'and Asia'];
      console.log(letters);
      console.log(...strContinental);
    </script>
  </body>
</html>
```

The spread operator array (...) 5

```
<html>
  <body>
    <script>
      let rankPlayers = [
        prompt("1-5 what do you think about Mikel"),
        prompt("1-5 what do you think about Lisa"),
        prompt("1-5 what do you think about Luis"),
      ];
      console.log(rankPlayers);
      function average(n1,n2,n3){
        return (parseInt(n1)+parseInt(n2)+parseInt(n3))/3
      }
      console.log(rankPlayers);
      console.log(average(rankPlayers[0], rankPlayers[1], rankPlayers[2]))
      console.log(average(...rankPlayers))

    </script>
  </body>
</html>
```

The spread operator (...) object 1

```
<html>
  <body>
    <script>
      //Basic assignment
      const user = {
        id: 42,
        isVerified: true
      };
      const {id, isVerified} = user;
      console.log(id); // 42
      console.log(isVerified); // true
    </script>
  </body>
</html>
```

The spread operator (...) object 2

```
<html>
  <body>
    <script>
      //Spread in object literals
      let obj1 = { foo: 'bar', x: 42 };
      let obj2 = { foo: 'baz', y: 13 };

      let clonedObj = { ...obj1 };
      // Object { foo: "bar", x: 42 }

      let mergedObj = { ...obj1, ...obj2 };
      // Object { foo: "baz", x: 42, y: 13 }

    </script>
  </body>
</html>
```

The spread operator (...) object 3

```
<html>
  <body>
    <script>
      let shift = {
        shiftId: '1',
        workPlace: 'L.A',
        workers: [],
        start:"",
        end:"",
        // ES6 enhanced object literals
        startShift(startTime = '8:00') {
          console.log(`Shift started! ${startTime}`);
        },
        endShift(endTime = '0:00') {
          console.log(`Shift ended! ${endTime}`);
        },
        workerForShift(w1, w2, w3) {
          console.log(
            `Worker at first shift ${w1}, ${w2} and ${w3}`
          );
        }
      };
      // Objects
      let newShift = { Country: "England", ...shift, shiftManager: 'Beker' };
      console.log(newShift);
      let shiftCopy = { ...newShift };
      shiftCopy.workers = [ 'Mad', 'Sharon' ];
    </script>
  </body>
</html>
```

```
onsole.log(shiftCopy.workers);
console.log(shiftCopy.shiftId);
shiftCopy.startShift("10:00")
let arrWorkers = [prompt("print first
worker"),prompt("print first worker"),
prompt("print first worker")]
shiftCopy.workerForShift(...arrWorkers)
```

```
</script>
</body>
</html>
```

The rest parameter 1

```
<html>
  <body>
    <script>
      //syntax

      //A function definition's last parameter can be prefixed
      //with "..." (three U+002E FULL STOP characters),
      //which will cause all remaining (user supplied)
      //parameters to be placed within a standard JavaScript array.

      // Only the last parameter in a function definition can
      // be a rest parameter.

      function f(a, b, ...theArgs) {
        // ...
      }
    </script>
  </body>
</html>
```

The rest parameter 2

```
<html>
  <body>
    <script>
      //A function definition can have only one ...restParam.
      //foo(...one, ...wrong, ...wrong)

      //The rest parameter must be the last parameter in the function definition.
      //foo(...wrong, arg2, arg3)
      //foo(arg1, arg2, ...correct)
    </script>
  </body>
</html>
```

The rest parameter 3

```
<html>
  <body>
    <script>
      //Using rest parameters
      //In this example, the first argument is mapped to a and the second to b,
      // so these named arguments are used as normal.

      //However, the third argument, manyMoreArgs,
      // will be an array that contains the third,
      //fourth, fifth, sixth ... nth – as many arguments
      //that the user includes.
      function myFun(a, b, ...manyMoreArgs) {
        console.log("a", a)
        console.log("b", b)
        console.log("manyMoreArgs", manyMoreArgs)
      }
      myFun("one", "two", "three", "four", "five", "six")
      // a, "one"
      // b, "two"
      // manyMoreArgs, ["three", "four", "five", "six"] <-- notice it's an array
    </script>
  </body>
</html>
```

The rest parameter 4

```
<html>
  <body>
    <script>
      //Argument length
      //Since theArgs is an array, a count of its elements is given by the length property.
      function fun1(...theArgs) {
        console.log(theArgs.length)
      }
      fun1()          // 0
      fun1(5)         // 1
      fun1(5, 6, 7)  // 3
    </script>
  </body>
</html>
```

The short circuiting 1 good to know

```
<html>
  <body>
    <script>
      a || (b * c); // evaluate `a` first,
      then produce `a` if `a` is "truthy"
      a && (b < c); // evaluate `a` first,
      then produce `a` if `a` is "falsy"
      a ?? (b || c); // evaluate `a` first,
      then produce `a` if `a` is not `null` and not
      `undefined`
      a?.b.c; // evaluate `a` first,
      then produce `undefined` if `a` is `null` or
      `undefined`
    </script>
  </body>
</html>
```

Short-circuiting is jargon for conditional evaluation. For example, in the expression `a && (b + c)`, if `a` is [falsy](#), then the sub-expression `(b + c)` will not even get evaluated, even if it is in parentheses. We could say that the logical disjunction operator ("OR") is "short-circuited". Along with logical disjunction, other short-circuited operators include logical conjunction ("AND"), nullish-coalescing, optional chaining, and the conditional operator. Some more examples follow.

`a || (b * c);` // evaluate `a` first, then produce `a` if `a` is "truthy"
`a && (b < c);` // evaluate `a` first, then produce `a` if `a` is "falsy"
`a ?? (b || c);` // evaluate `a` first, then produce `a` if `a` is not `null` and not `undefined`
`a?.b.c;` // evaluate `a` first, then produce `undefined` if `a` is `null` or `undefined`

The short circuiting 2 good to know

```
<html>
  <body>
    <script>
      3 > 2 && 2 > 1
      // returns true
      3 > 2 > 1
      // Returns false because 3 > 2 is true, then true is converted to 1
      // in inequality operators, therefore true > 1 becomes 1 > 1, which
      // is false. Adding parentheses makes things clear: (3 > 2) > 1.

    </script>
  </body>
</html>
```

The Nullish coalescing operator (??)

```
<html>
  <body>
    <script>
      //Nullish coalescing operator (??)
      const foo = null ?? 'default string';
      console.log(foo);
      // expected output: "default string"

      const baz = 0 ?? 42;
      console.log(baz);
      // expected output: 0
    </script>
  </body>
</html>
```

The nullish coalescing operator (??) is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand.

This can be contrasted with the logical OR (||) operator, which returns the right-hand side operand if the left operand is any falsy value, not only null or undefined. In other words, if you use || to provide some default value to another variable foo, you may encounter unexpected behaviors if you consider some falsy values as usable (e.g., "" or 0). See below for more examples.

The nullish coalescing operator has the fifth-lowest operator precedence, directly lower than || and directly higher than the conditional (ternary) operator.

The Nullish coalescing operator (??)

Syntax :

leftExpr ?? rightExpr

```
<html>
  <body>
    <script>
      const nullValue = null;
      const emptyText = ""; // falsy
      const someNumber = 42;

      const valA = nullValue ?? "default for A";
      const valB = emptyText ?? "default for B";
      const valC = someNumber ?? 0;

      console.log(valA); // "default for A"
      console.log(valB); // "" (as the empty string is not null or undefined)
      console.log(valC); // 42
    </script>
  </body>
</html>
```

The Nullish coalescing operator (??)

```
<html>
  <body>
    <script>
      // Assigning a default value to a variable
      // Earlier, when one wanted to assign a default value to a variable, a common pattern was to use
      // the logical OR operator (||):
      let foo;
      // foo is never assigned any value so it is still undefined
      let someDummyText = foo || 'Hello!';

    </script>
  </body>
</html>
```

The Nullish coalescing operator (??)

```
<html>
  <body>
    <script>
      //However, due to || being a boolean logical operator,
      // the left hand-side operand was coerced to a boolean
      //for the evaluation and any falsy value (0, '', NaN, null, undefined)
      // was not returned. This behavior may
      // cause unexpected consequences if you consider 0,
      // '', or NaN as valid values.

      let count = 0;
      let text = "";
      let qty = count ?? 42;
      let message = text ?? "hi!";
      console.log(qty);      // 42 and not 0
      console.log(message); // "hi!" and not ""

    </script>
  </body>
</html>
```

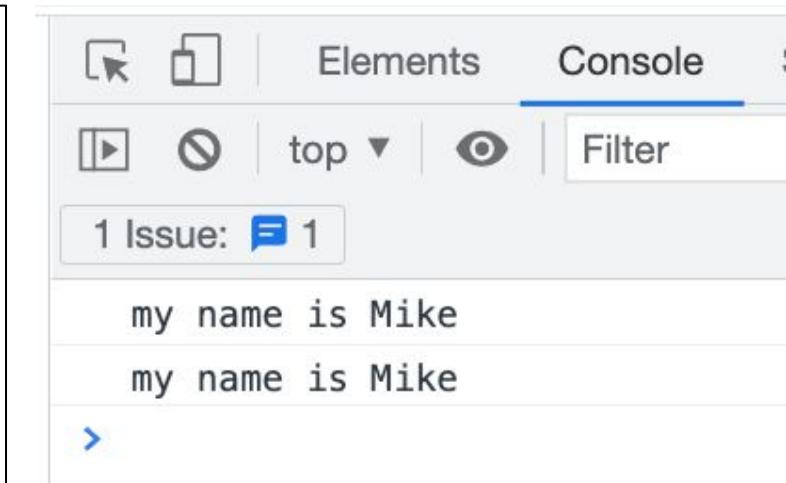
The Nullish coalescing operator (??)

```
<html>
  <body>
    <script>
      //Short-circuiting
      // Like the OR and AND logical operators,
      // the right-hand side expression is not evaluated
      // if the left-hand side proves to be neither null nor undefined.
      function A() { console.log('A was called'); return undefined; }
      function B() { console.log('B was called'); return false; }
      function C() { console.log('C was called'); return "foo"; }

      console.log( A() ?? C() );
      // logs "A was called" then "C was called" and then "foo"
      // as A() returned undefined so both expressions are evaluated
      console.log( B() ?? C() );
      // logs "B was called" then "false"
      // as B() returned false (and not null or undefined), the right
      // hand side expression was not evaluated
    </script>
  </body>
</html>
```

Usage of the backtick character (`) in JavaScript

```
<script>  
  
    var name = "Mike"  
  
    console.log("my name is " + name)  
  
    console.log(`my name is ${name}`)  
  
</script>
```



Looping array

```
<html>
  <body>
    <h1>Looping Array</h1>
    <script>
      let restaurant = {
        timeShift1: '8:00-16:00',
        timeShift2: '16:00-00:00',
        shiftsWorkers: [
          [
            'Mike',
            'James',
            'John',
            'Michael',
            'William',
            'David',
          ],
          [
            'Joseph',
            'Thomas',
            'Charles',
            'Christopher',
            'Daniel',
            'Mark',
          ],
        ],
        totalTipShift1: 4000,
        totalTipShift2: 5000,
        tipped: ['Mike', 'James', 'Michael', 'David', 'Thomas'],
        date: '12/12/2021'
      };
    </script>
  </body>
</html>
```

```
const [a,b] = [...restaurant.shiftsWorkers];
const allWorkers = [...a, ...b]
console.log(allWorkers)
for(const item of allWorkers){
  console.log(item)
}
for(const item of allWorkers.entries()){
  console.log(item)
}
for(const [i,el] of allWorkers.entries()){
  console.log(` ${i+1}. ${el}`)
}

</script>
</body>
</html>
```

Chaining object

```
<html>
  <body>
    <h1>Looping Array</h1>
    <script>
      let restaurant= {
        timeShift1: '8:00-16:00',
        timeShift2: '16:00-00:00',
        shiftsWorkers: [
          [
            'Mike',
            'James',
            'John',
            'Michael',
            'William',
            'David',
          ],
          [
            'Joseph',
            'Thomas',
            'Charles',
            'Christopher',
            'Daniel',
            'Mark',
          ],
        ],
        totalTipShift1:4000,
        totalTipShift2:5000,
        tipped: ['Mike', 'James', 'Michael', 'David','Thomas'],
        date: '12/12/2021',
        address:{city:"NY", street:"Arguban 6"}
      };

      </script>
    </body>
  </html>
```

```
console.log(restaurant.address.city)
```

Looping object keys and values

```
<html>
  <body>
    <h1>Looping Array</h1>
    <script>
      const days = ['monday', 'tuesday', 'wednesday', 'thuesday',
'friday', 'saturday', 'sunday'];
      let restaurant= {
        timeShift1: '8:00-16:00',
        timeShift2: '16:00-00:00',
        shiftsWorkers: [
          [
            'Mike',
            'James',
            'John',
            'Michael',
            'William',
            'David',
          ],
          [
            'Joseph',
            'Thomas',
            'Charles',
            'Christopher',
            'Daniel',
            'Mark',
          ],
        ],
        totalTipShift1:4000,
        totalTipShift2:5000,
        tipped: ['Mike', 'James', 'Michael', 'David','Thomas'],
        date: '12/12/2021',
        address:{city:"NY",street:"Arguban 6"},
        opendays :[days[0],days[3],days[4]]
      };
    </script>
  </body>
</html>
```

```
const properties = Object.keys(restaurant)
console.log(properties)
for(let day of Object.values(restaurant.opendays)) {
  console.log(day)
}
</script>
</body>
</html>
```

The **Set** object lets you store unique values of any type, whether primitive values or object references.

Description

Set objects are collections of values. You can iterate through the elements of a set in insertion order. A value in the Set may only occur once; it is unique in the Set's collection.

Set has, delete and add

```
<html>
  <body>
    <script>
      const mySet1 = new Set()
      mySet1.add(1)          // Set [ 1 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add('some text') // Set [ 1, 5, 'some text' ]
      const o = {a: 1, b: 2}
      mySet1.add(o)

      mySet1.add({a: 1, b: 2}) // o is referencing a different object, so this is okay

      mySet1.has(1)          // true
      mySet1.has(3)          // false, since 3 has not been added to the set
      mySet1.has(5)          // true
      mySet1.has(Math.sqrt(25)) // true
      mySet1.has('Some Text'.toLowerCase()) // true
      mySet1.has(o)          // true

      mySet1.size            // 5

      mySet1.delete(5)        // removes 5 from the set
      mySet1.has(5)          // false, 5 has been removed

      mySet1.size            // 4, since we just removed one value

      console.log(mySet1)
      // logs Set(4) [ 1, "some text", ..., ... ] in Firefox
      // logs Set(4) { 1, "some text", ..., ... } in Chrome
    </script>
  </body>
</html>
```

Set add

```
<html>
  <body>
    <script>
      const mySet1 = new Set()
      mySet1.add(1)          // Set [ 1 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add('some text') // Set [ 1, 5, 'some text' ]
      const o = {a: 1, b: 2}
      mySet1.add(o)

      mySet1.add({a: 1, b: 2}) // o is referencing a different object, so
this is okay

    </script>
  </body>
</html>
```

Set has

```
mySet1.has(1)          // true
mySet1.has(3)          // false, since 3 has not been added to the set
mySet1.has(5)          // true
mySet1.has(Math.sqrt(25)) // true
mySet1.has('Some Text'.toLowerCase()) // true
mySet1.has(0)          // true

mySet1.size            // 5
```

Set delete

```
mySet1.delete(5)      // removes 5 from the set
mySet1.has(5)         // false, 5 has been removed

mySet1.size           // 4, since we just removed one value

console.log(mySet1)
// logs Set(4) [ 1, "some text", ..., ... ] in Firefox
// logs Set(4) { 1, "some text", ..., ... } in Chrome
```

All file set together

```
<html>
  <body>
    <script>
      const mySet1 = new Set()
      mySet1.add(1)          // Set [ 1 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add(5)          // Set [ 1, 5 ]
      mySet1.add('some text') // Set [ 1, 5, 'some text' ]
      const o = {a: 1, b: 2}
      mySet1.add(o)

      mySet1.add({a: 1, b: 2}) // o is referencing a different object, so this is okay

      mySet1.has(1)          // true
      mySet1.has(3)          // false, since 3 has not been added to the set
      mySet1.has(5)          // true
      mySet1.has(Math.sqrt(25)) // true
      mySet1.has('Some Text'.toLowerCase()) // true
      mySet1.has(o)          // true

      mySet1.size            // 5

      mySet1.delete(5)        // removes 5 from the set
      mySet1.has(5)          // false, 5 has been removed

      mySet1.size            // 4, since we just removed one value

      console.log(mySet1)
      // logs Set(4) [ 1, "some text", ..., ... ] in Firefox
      // logs Set(4) { 1, "some text", ..., ... } in Chrome
    </script>
  </body>
</html>
```

Iterating set - key and values

```
// iterate over items in set  
  
// logs the items in the order: 1, "some text", {"a": 1, "b": 2}, {"a": 1, "b": 2}  
for (let item of mySet1) console.log(item)  
  
  
// logs the items in the order: 1, "some text", {"a": 1, "b": 2}, {"a": 1, "b": 2}  
for (let item of mySet1.keys()) console.log(item)  
  
  
// logs the items in the order: 1, "some text", {"a": 1, "b": 2}, {"a": 1, "b": 2}  
for (let item of mySet1.values()) console.log(item)  
  
// logs the items in the order: 1, "some text", {"a": 1, "b": 2}, {"a": 1, "b": 2}  
// (key and value are the same here)  
for (let [key, value] of mySet1.entries()) console.log(key)
```

All Iterating set

```
// converting between Set and Array

const mySet2 = new Set([1, 2, 3, 4])

mySet2.size // 4

// [...mySet2] // [1, 2, 3, 4]
```

set and arrays - remove duplicate elements

```
<html>
  <body>
    <script>

      let myArray = ['value1', 'value2', 'value3']

      // Use the regular Set constructor to transform an Array into a Set
      let mySet = new Set(myArray)

      mySet.has('value1')      // returns true

      // Use the spread operator to transform a set into an Array.
      console.log([...mySet]) // Will show you exactly the same Array as myArray

    </script>
  </body>
</html>
```

set and arrays

```
<html>
  <body>
    <script>
      let myArray = ['value1', 'value2', 'value3']

      // Use the regular Set constructor to transform an Array into a Set
      let mySet = new Set(myArray)

      mySet.has('value1')      // returns true

      // Use the spread operator to transform a set into an Array.
      console.log([...mySet]) // Will show you exactly the same Array as myArray
    </script>
  </body>
</html>
```

set and arrays - remove duplicate elements

```
<html>

  <body>

    <script>

      // Use to remove duplicate elements from the array

      const numbers = [2,3,4,4,2,3,3,4,4,5,5,6,6,7,5,32,3,4,5]

      console.log([...new Set(numbers)])

      // [2, 3, 4, 5, 6, 7, 32]

    </script>

  </body>

</html>
```

set and string

```
<html>
  <body>
    <script>
      let text = 'India'
      const mySet = new Set(text) // Set(5) { 'I', 'n', 'd', 'i', 'a' }
      mySet.size // 5
      //case sensitive & duplicate omission
      new Set("Firefox") // Set(7) { "F", "i", "r", "e", "f", "o", "x" }
      new Set("firefox") // Set(6) { "f", "i", "r", "e", "o", "x" }
    </script>
  </body>
</html>
```

Map

The Map object holds key-value pairs and remembers the original insertion order of the keys. Any value (both objects and) may be used as either a key or a value.

```
<html>
  <body>
    <script>
      const map1 = new Map();
      map1.set('a', 1);
      map1.set('b', 2);
      map1.set('c', 3);
      console.log(map1.get('a'));
      // expected output: 1
      map1.set('a', 97);
      console.log(map1.get('a'));
      // expected output: 97
      console.log(map1.size);
      // expected output: 3
      map1.delete('b');
      console.log(map1.size);
      // expected output: 2
    </script>
  </body>
</html>
```

Map

```
<html>
  <body>
    <script>
      //The correct usage for storing data in the Map is through the set(key, value) method.

      const contacts = new Map()

      contacts.set('Jessie', {phone: "213-555-1234", address: "123 N 1st Ave"})
      contacts.has('Jessie') // true
      contacts.get('Hilary') // undefined
      contacts.set('Hilary', {phone: "617-555-4321", address: "321 S 2nd St"})
      contacts.get('Jessie') // {phone: "213-555-1234", address: "123 N 1st Ave"}
      contacts.delete('Raymond') // false
      contacts.delete('Jessie') // true
      console.log(contacts.size) // 1
    </script>
  </body>
</html>
```

Map - using the map object

```
<html>
  <body>
    <script>
      //Using the Map object
      const myMap = new Map()
      const keyString = 'a string'
      const keyObj = {}
      const keyFunc = function() {}
      // setting the values
      myMap.set(keyString, "value associated with 'a string'")
      myMap.set(keyObj, 'value associated with keyObj')
      myMap.set(keyFunc, 'value associated with keyFunc')
      myMap.size // 3
      // getting the values
      myMap.get(keyString) // "value associated with 'a string'"
      myMap.get(keyObj) // "value associated with keyObj"
      myMap.get(keyFunc) // "value associated with keyFunc"
      myMap.get('a string') // "value associated with 'a string'"
                           // because keyString === 'a string'
      myMap.get({}) // undefined, because keyObj !== {}
      myMap.get(function() {}) // undefined, because keyFunc !== function () {}
    </script>
  </body>
</html>
```

Map - using Nan as map keys

```
<html>
  <body>
    <script>
      //Using NaN as Map keys
      // NaN can also be used as a key.
      // Even though every NaN is not equal to
      // itself (NaN !== NaN is true),
      // the following example works because NaNs are
      // indistinguishable from each other:
      const myMap = new Map()
      myMap.set(NaN, 'not a number')
      myMap.get(NaN)
      // "not a number"

      const otherNaN = Number('foo')
      myMap.get(otherNaN) // "not a number"
    </script>
  </body>
</html>
```

Map - Iterating map

```
<html>
  <body>
    <script>
      // Maps can be iterated using a for..of loop:
      const myMap = new Map()
      myMap.set(0, 'zero')
      myMap.set(1, 'one')
      for (const [key, value] of myMap) {
        console.log(key + ' = ' + value)
      } // 0 = zero      // 1 = one
      for (const key of myMap.keys()) {
        console.log(key)
      } // 0// 1
      for (const value of myMap.values()) {
        console.log(value)
      } // zero// one
      for (const [key, value] of myMap.entries()) {
        console.log(key + ' = ' + value)
      } // 0 = zero // 1 = one
    </script>
  </body>
</html>
```

Map - relation with array object - good to know

```
<html>
  <body>
    <script>
      //Relation with Array objects
      const kvArray = [['key1', 'value1'], ['key2', 'value2']]

      // Use the regular Map constructor to transform a 2D key-value Array into a map
      const myMap = new Map(kvArray)

      myMap.get('key1') // returns "value1"

      // Use Array.from() to transform a map into a 2D key-value Array
      console.log(Array.from(myMap)) // Will show you exactly the same Array as kvArray

      // A succinct way to do the same, using the spread syntax
      console.log([...myMap])

      // Or use the keys() or values() iterators, and convert them to an array
      console.log(Array.from(myMap.keys())) // ["key1", "key2"]
    </script>
  </body>
</html>
```

Map - cloning map - good to know

```
<html>
  <body>
    <script>
      //Cloning and merging Maps
      // Just like Arrays, Maps can be cloned:
      const original = new Map([
        [1, 'one']
      ])

      const clone = new Map(original)

      console.log(clone.get(1))          // one
      console.log(original === clone) // false (useful for shallow comparison)
    </script>
  </body>
</html>
```

Map - cloning map - good to know

```
<html>
  <body>
    <script>
      //Maps can be merged, maintaining key uniqueness:
      const first = new Map([
        [1, 'one'],
        [2, 'two'],
        [3, 'three'],
      ])
      const second = new Map([
        [1, 'uno'],
        [2, 'dos']
      ])
      // Merge two maps. The last repeated key wins.
      // Spread operator essentially converts a Map to an Array
      const merged = new Map([...first, ...second])
      console.log(merged.get(1)) // uno
      console.log(merged.get(2)) // dos
      console.log(merged.get(3)) // three
    </script>
  </body>
</html>
```

Map - merge map - good to know

```
<html>
  <body>
    <script>
      //Maps can be merged with Arrays, too:
      const first = new Map([
        [1, 'one'],
        [2, 'two'],
        [3, 'three'],
      ])
      const second = new Map([
        [1, 'uno'],
        [2, 'dos']
      ])
      // Merge maps with an array. The last repeated key wins.
      const merged = new Map([...first, ...second, [1, 'eins']])
      console.log(merged.get(1)) // eins
      console.log(merged.get(2)) // dos
      console.log(merged.get(3)) // three
    </script>
  </body>
</html>
```

Summary - Array, Set, Map

data come from :

1. Programmer
2. UI
3. external source

where to store it ?

simple list == > array or sets

array

- array - when you need order list of values.

set

- sets when u need to work with unique values or you need to remove duplicate

Summary - Array, Set, Map

map

maps : better performance

maps : keys can have any data type

maps : easy to iterate

maps : easy to compute size

object

objects : more traditional key value.

objects : keys can have only string type.

objects : use when you working with json