

הרצאה 1

מבוא ל HTML

קורס : צד ל Koh

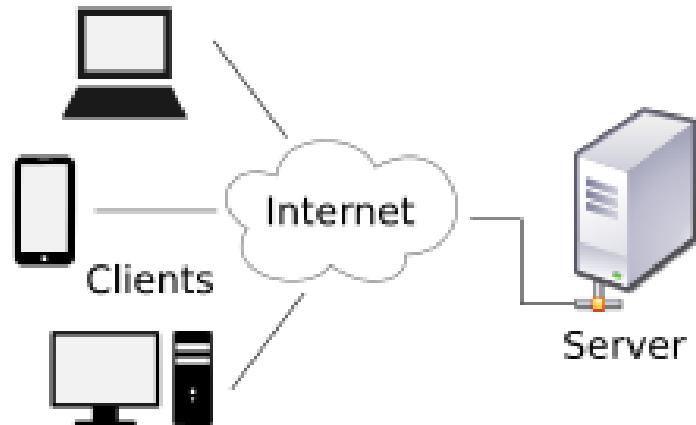
מרצה: יעל סלע זעירא

מבוא לפיתוח צד לקוח

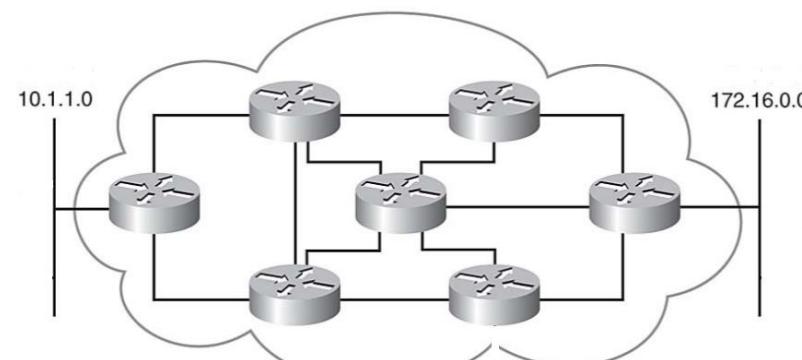
- מבוא לאינטרנט ומודל לקוח-שרת
- הקמת אתר בסביבת visual studio
- מבוא ל-HTML
- בניית דף אינטרנט ראשון
- html element
- style inline
- תגיות בסיסיות נוספות
- Box model

מבנה האינטרנט

אינטרנט - רשת שמחברת מחשבים בכל רחבי העולם כך
שניתן להעביר מידע ממחשב למחשב באמצעות נתבים.



לכל מחשב יש כתובת IP שאליה ישלח המידע,
כתובת IP נראית כך - 194.90.203.10



האינטרנט עובד בדיקון כמו משלוח דואר רגיל כאשר מספקים
כתובת למשЛОח דואר ומעבירים מכתב/חבילה לדואר (הנット).

URL

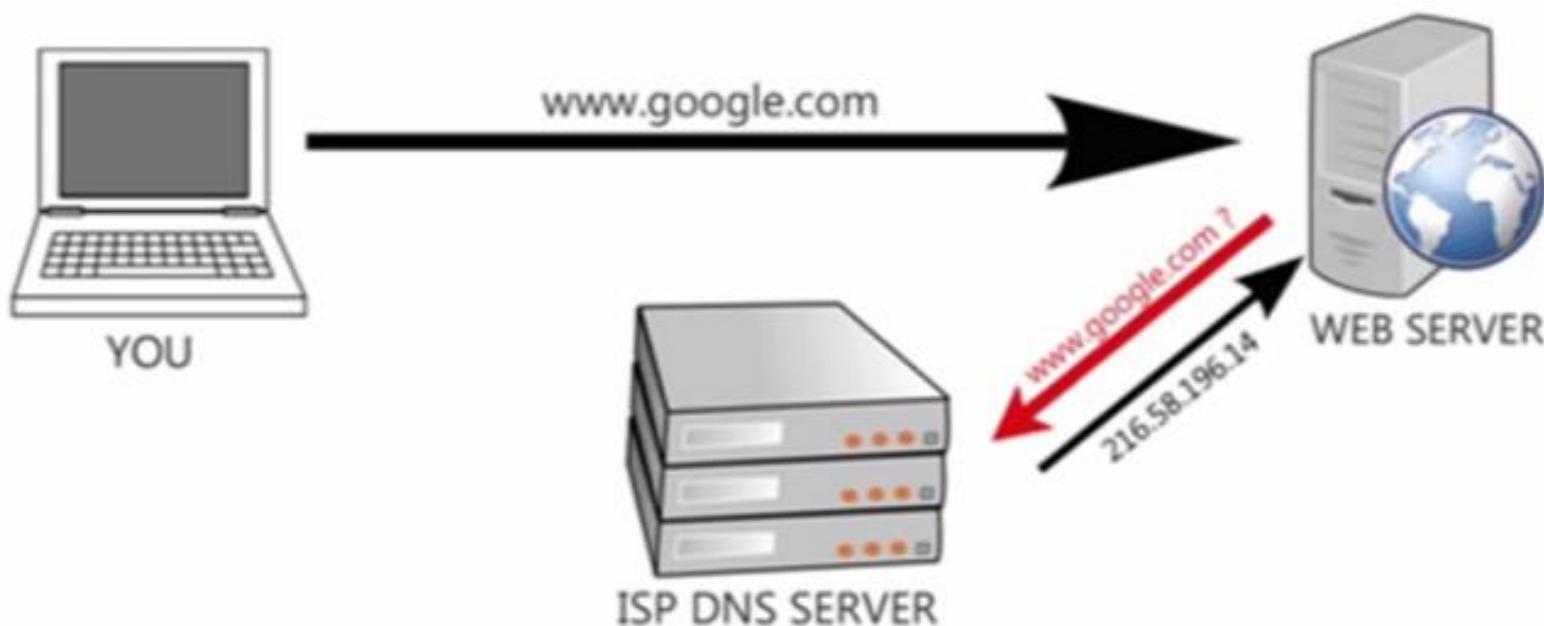
איך הנטבאים ידעו את הכתובת של פייסבוק???
הרי העברנו להם כתובת צו www.facebook.com ולא מהצורה:

194.90.203.10

Uniform Resource Locator - URL – צורה איחידה לרשום כתובות, שנקראת גם כתובת האתר.

DNS - Domain Name System

כתובת האתר בצורת URL תועבר לשרת DNS שיזיר כתובת IP:
בניהם זוכים בקהלות שמות, ה- DNS מבצע המרה בין הכתובת המילולית, אותה זוכר המשתמש, לבין כתובת ה- IP בה למעשה משתמש המחשב על מנת לתקשר עם היעד.



5.10.2021

[https://www.ynet.co.il/digital/
technews/article/bj1rejoef](https://www.ynet.co.il/digital/technews/article/bj1rejoef)

אחרי שעות ארוכות: פייסבוק, וואטסאפ ואינסטגרם חזרו לפועל

התקלה בפייסבוק ובאינסטגרם תוקנה אחרי חצות, יותר משש שעות אחרי הקרישה, וכשעתיים לאחר מכן גם וואטסאפ חזרה לתפקד. "אנחנו מצטערים, תודה על הסבלנות", ציינה פייסבוק - ובהמשך סיפקה הסבר לקרישה הארוכה ביותר שלה מאז 2008: הגדרה לקויה של תצורת הנتابים שלה. מומחה: "דבר נדיר מאוד"

ירם הכהן, מנכ"ל איגוד האינטרנט הישראלי, שמנhall את מרחב צו. ושרתי ה-DNS שלו, מסר: "התקלה של פייסבוק היא חלום הבלחות של כל מי שמנהל שירות אינטרנט וקрайת השכמה לציבור שלא לבסס פעילותו על נוכחות בפלטפורמות בלבד אלא לגונן את הכלים לתקשורת בין אישית ועסקית. התקלה זו, בין אם היא תוצאה של מתקפה ובין אם טעות אנוש, היא חלום הבלחות של כל מי שמנהל שירות אינטרנט".

לדבריו ה-DNS משמש כ"ספר הטלפונים של האינטרנט" והוא חייב להיות זמין כל הזמן. "המשמעות של התקלה היא חסימה מוחלטת של השימוש בשם המתחם com, Facebook.com, לרבות באפליקציות סלולר שאן הן מtabססות על ה-DNS. הנפגעים, מעבר למובן לקבוצת פייסבוק עצמה, הם משתמשים פרטיים ועסקים בשירותי הפלטפורמות פייסבוק, אינסטגרם ווואטסאפ".

כתובת URL

דוגמא:

<http://www.ynet.co.il/home.html>

הפרוטוקול

כתובת האתר ברשות האינטרנט

הקובץ בו הלקוח מעוניין

מה זה פיתוח צד שרת ומה זה פיתוח צד לקוח

צד לקוח:

עימוד ועיצוב הדפים.
יצירת בקשות לשרת עבור נתונים.
כל פעולה שלא מצריכה נתונים מבוסיס הנתונים –
דף שיווקי לדוגמא שהתוכן שלו קבוע.

צד שרת:

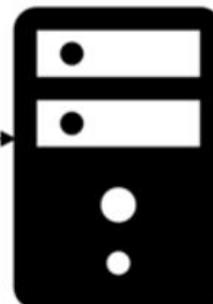
אחראי על ביצוע שאילתות מול בסיס הנתונים לפי בקשות הלקוח.
למשל קבלת שם משתמש וסיסמה וביצוע בדיקה האם קיימים במאגר הנתונים.

צד לקוח

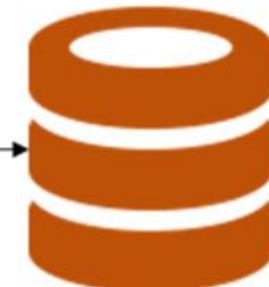


Client

צד שרת



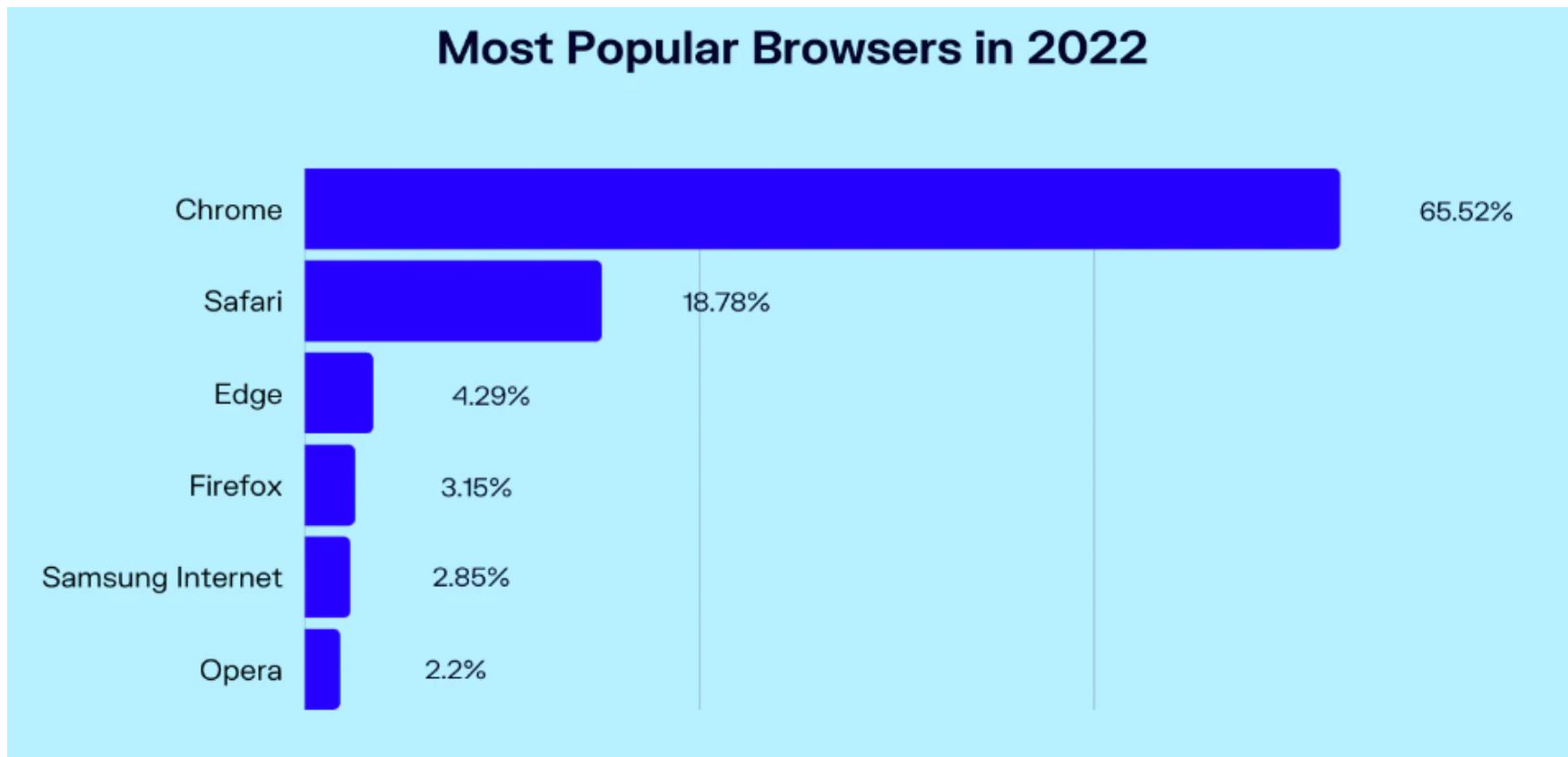
Server



Database

תוכנת דפסן

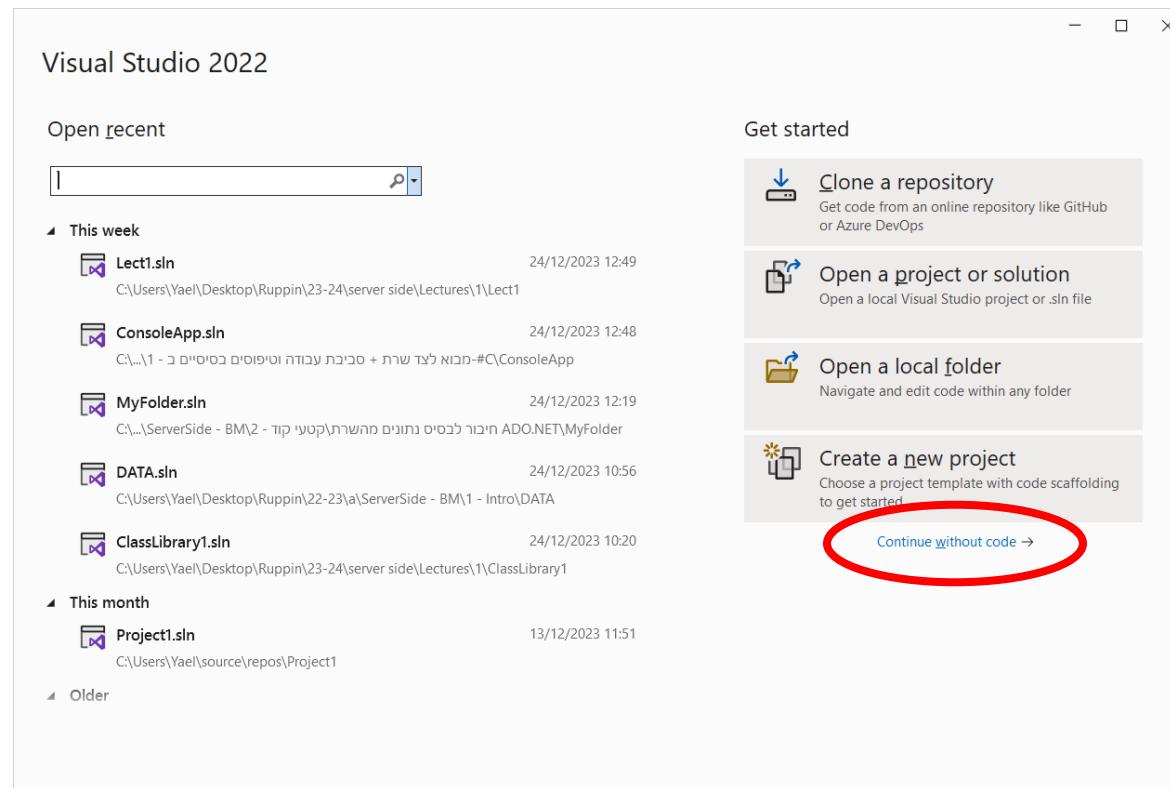
תוכנה היודעת לשלוח בקשות לשרתים ולהציג קבצים המתקבלים ע"י השרת.



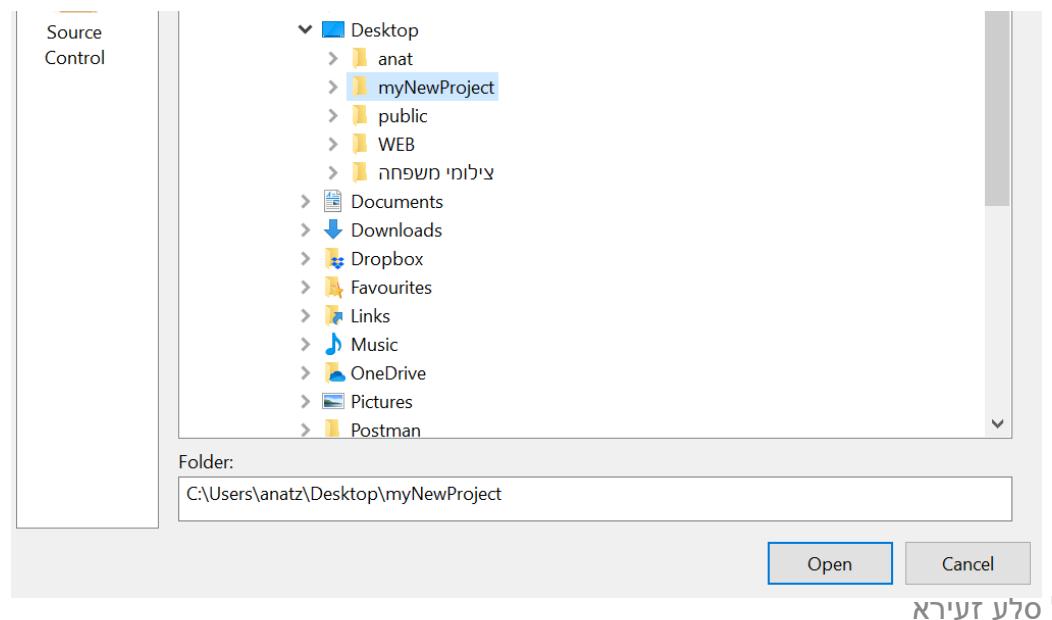
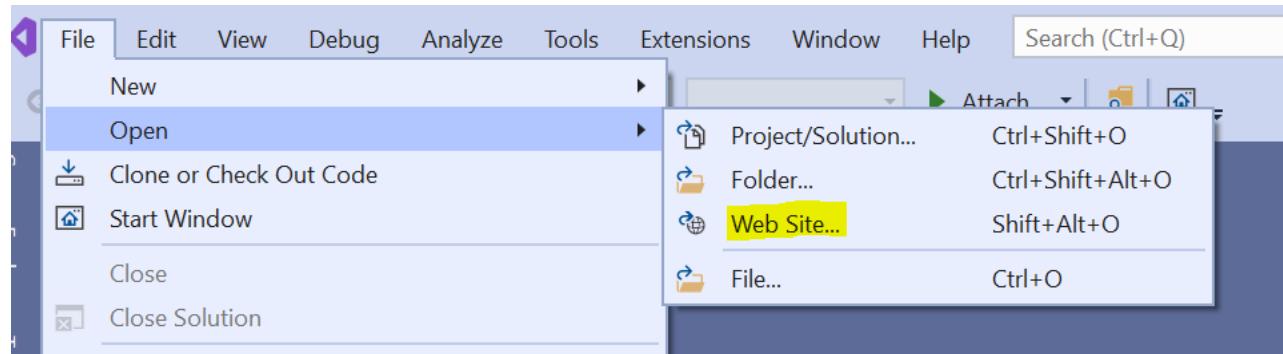
הקמת אתר בסביבה visual studio

1. פתחו תיקייה ריקה במחשב (במקום המתאים לכם)

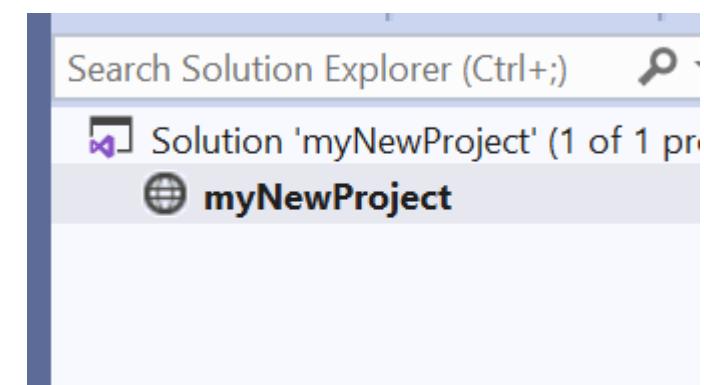
2. היכנסו לsolution visual studio ובסעך הראשון לחצו על - Continue without code



הקמת אתר בסביבה visual studio



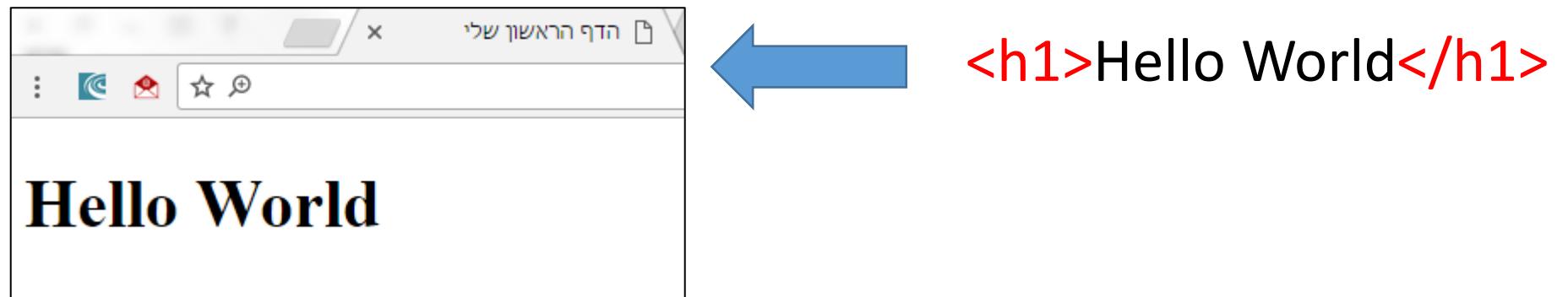
3. בתוכה הסביבה פתחו **web site**
4. בחרו את התיקייה שיצרתם מחשב
5. אז תקבלו את סביבת העבודה שלכם



מבוא ל-HTML

שפת Hyper Text Markup Language אשר מצינית לדפדף מה להציג כאשר הוא קורא את הקוד.
הינה השפה המרכזית בכתיבה דפי אינטרנט.

בשפת HTML כל אלמנט מורכב מותג, למשל התג **h1** מצין כותרת:



אנטומיה של דף HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>My test page</title>
</head>
<body>
  <p>This is my page</p>
</body>
</html>
```

שרירות היסטורית – זה מה שאתם צריכים לדעת על זה

עוטף את תוכן הדף

הגדרות "על" נוספות שיופיעו ב head, כמו המלצות להגדיר גודל מסך ופונקציית זיהוי מסך

כותרת הדף

בתוכו יהיה כל ה content

מבוא ל-HTML

למרבית האלמנטים יש-tag פתוח (`<h1>`) ו-tag סגור (`</h1>`), כאשר ביניהם יבוא הטקסט של האלמנט.

אך יש אלמנטים שלא כמו הדוגמא הבאה:

```

```

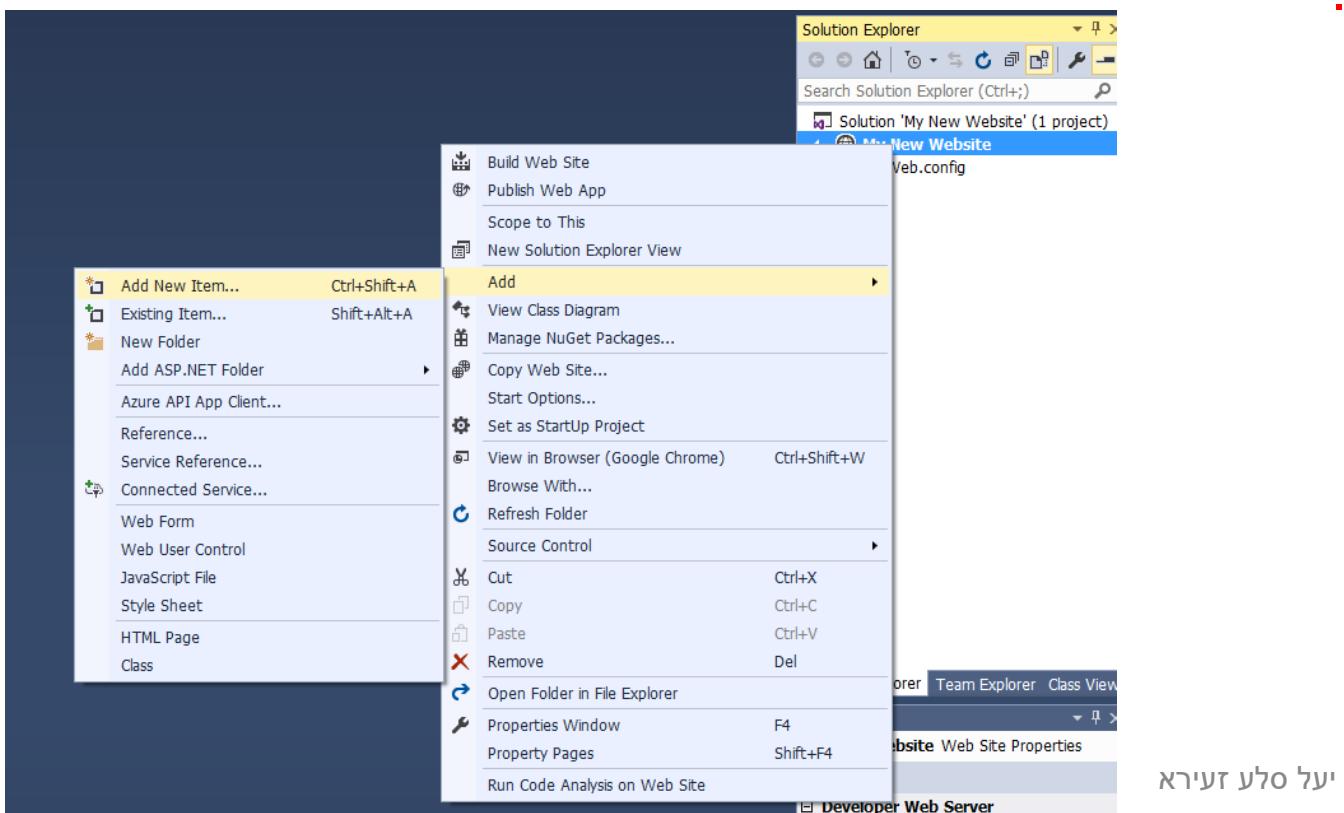
האלמנטים הללו לפעמים נקראים **void elements**

Nesting elements

```
<p>My cat is <strong>very</strong> grumpy.</p>
```

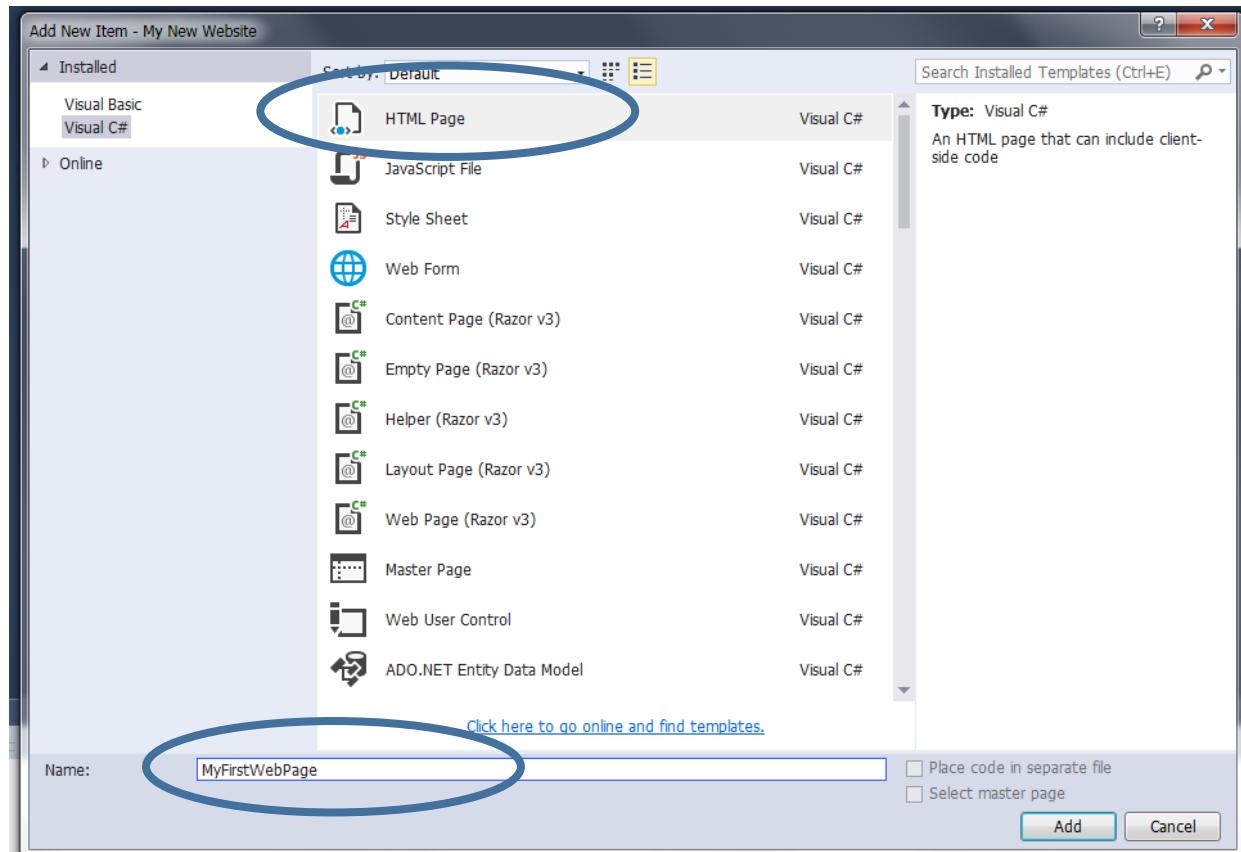
הוספת דף אינטרנט לפרויקט

ישנם הרבה סוגים של דפי אינטרנט – xml, php, aspx, html ועוד הרבה...
כל הדפים הללו עושים שימוש בשפת html.
לא להתבלבל ולקשר את קובץ html לשפת html.



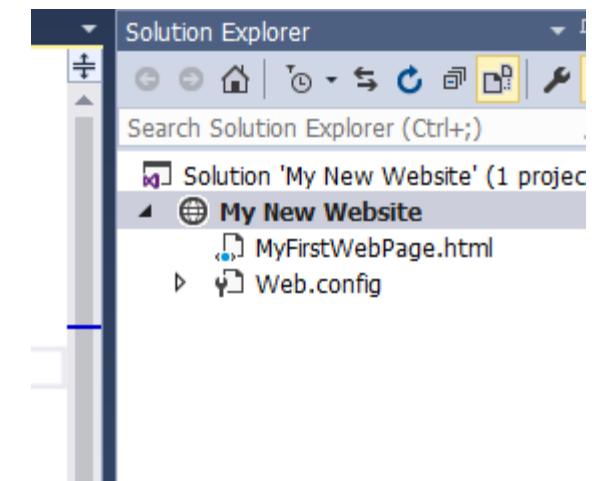
בקורס אנחנו נעבד עם קבצי html בלבד.
מוסיף קובץ html לפרויקט שלנו:
קליק ימני על תיקיית הפרויקט - <
Add New Item <- Add

הוספת דף אינטרנט לפרויקט



נבחר מהרשימה קובץ html
וניתן לו שם בחלק התחתיו –
מומלץ באנגלית ולא רוחחים.

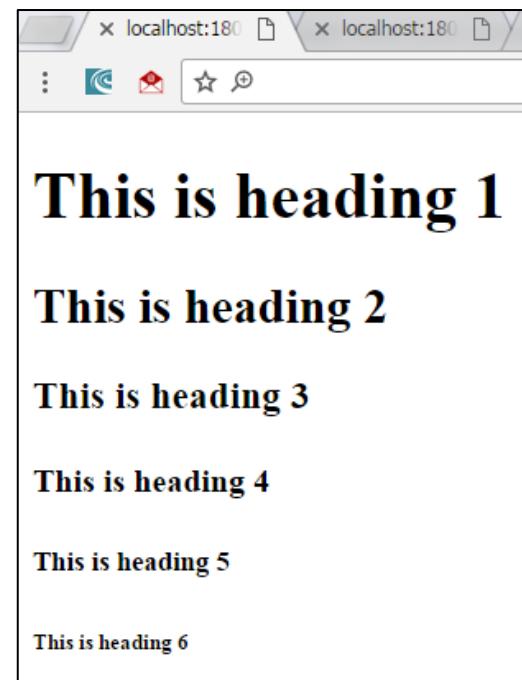
לאחר מכן נלחץ על add להוספה הדף:



תגיות כותרת

משפחת תגיות **h** מצינית הצגת כותרת, **h1-h6**.
h1 – הכותרת הגדולה ביותר, **h6** – הכותרת הקטנה ביותר.

```
|<body>
|  <h1>This is heading 1</h1>
|  <h2>This is heading 2</h2>
|  <h3>This is heading 3</h3>
|  <h4>This is heading 4</h4>
|  <h5>This is heading 5</h5>
|  <h6>This is heading 6</h6>
|
|</body>
```



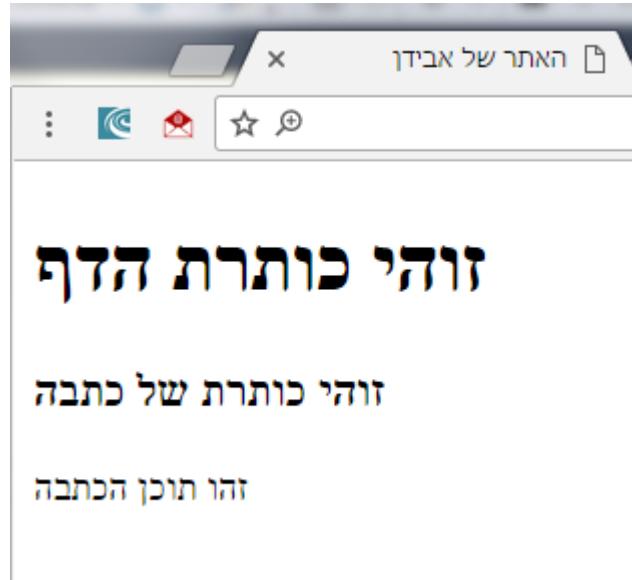
תרגיל ביתה – דף אינטרנט ראשון

הקימו אתר בסביבת visual studio וצרו דף אינטרנט שנראה כך:

* החליפו את שם הדף עם השם שלכם

הכניסה רצף של טקסט אקראי

נסו את תגיית הפסקה <đ>



White space in HTML

```
<p>Dogs are cute.</p>
```

```
<p>
```

Dogs
cute.

are

```
</p>
```

**על מנת להכניס יותר מרוח אחד נוכל להשתמש ב; **

Special Characters

Literal character	Character reference equivalent
<	<
>	>
"	"
'	'
&	&

`<p>`In HTML, you define a paragraph using the `<p>` element.`</p>`

`<p>`In HTML, you define a paragraph using the `<p>` element.`</p>`

HTML Comments

```
<p>I'm not inside a comment</p>
```

```
<!-- <p>I am!</p> -->
```

תבנית הקורס



תגית html

```
<h1 id="header" title="this is header"> the text </h1>
```

תג פתוח ושם

תכונות

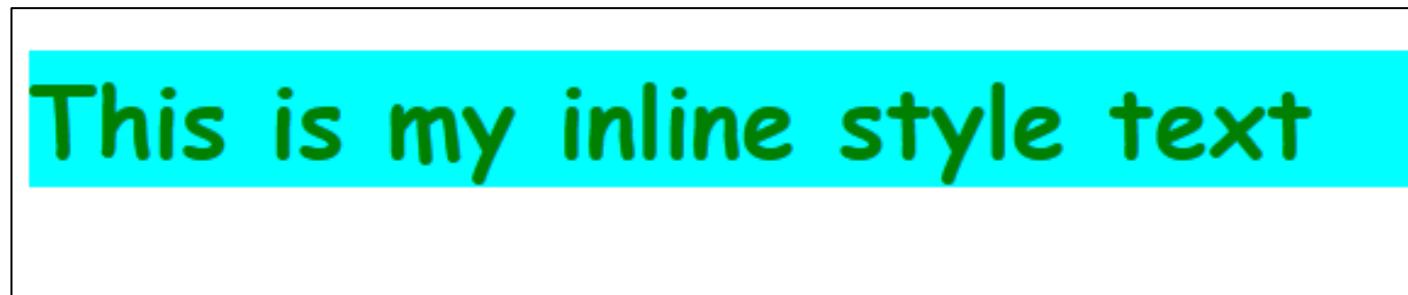
טקסט

תג סגור

- *ישנים פקדים מסוימים שאין להם תג סגור וטקסט
- *לרוב תפקיד הפקד להציג מההו על המסר

תבונת ה-style

תכונת `the-style` קובעת את העיצוב של האלמנט בהיבט של:



- צבע טקסט
- צבע רקע
- פונט
- גודל כתוב
- מיקום הטקסט
- ועוד הרבה..

In-line style

```
<p style="text-indent: 2cm; color: green;">
```

This paragraph has been formatted using the in-line style command.

```
</p>
```

```
<p>
```

This paragraph has not been formatted using the in-line style command.

```
</p>
```

תבונת `style`

למשל קטע הקוד הבא יציג כותרת בצבע אדום:

```
<h1 style="color:red">header</h1>
```

על מנת לשנות מספר תכונות עיצוב נכתב זאת בצורה הבאה:

```
<Element style="attribute1:value; attribute2:value; ..... attributeN:value;">Text</ Element >
```

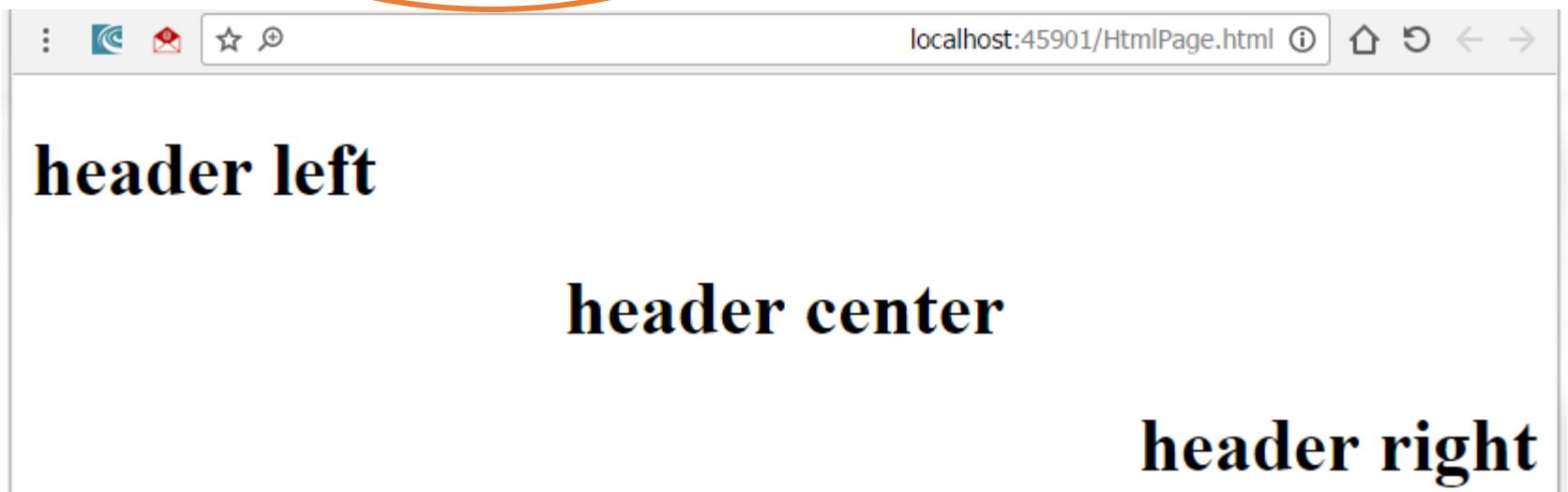


```
<h1 style="color:red;background-color:yellow;font-family:Aharoni">header</h1>
```

תבונת style

```
<h1 style="text-align:left">header left</h1>
<h1 style="text-align:center">header center</h1>
<h1 style="text-align:right">header right</h1>
```

מיקום הטקסט:



כינור

```
<a href="1.html">page 1</a> <br />
```

```
<a href="2.html">page 2</a>
```

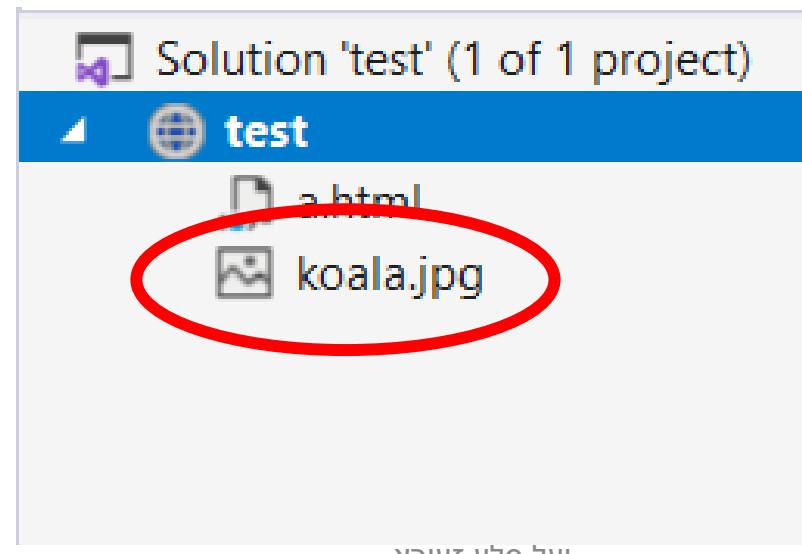
_blank : a newly opened, unnamed window.

```
<a href="2.html" target="_blank">new window</a>
```

img

תג `img` משמש להציג תמונות – קבצים בפורמט `png`, `jpg`, `gif`

התכונה `src` מציין על מיקום התמונה באופן ייחודי לקובץ הקוד ממנו מוצבאים.



```

```

img

על מנת לשלוט בגודל התמונה נוכל להשתמש ב `width, height` או בתכונת `height`

```

```

OR

```

```

img

אפשר להכניס גם כתובת התמונה בראשת - שימושו לב שהلينק מפנה לתמונה

```

```

תגיות נוספות - טקסט

תג span משמש להבלטה של טקסט בתוך טקסט או כתקסט בפני עצמו:

```
<body dir="rtl">  
  <h1> this part is <span style="color:red">important</span> and this is more information</h1>  
</body>
```

localhost:45901/HtmlPage.html ⓘ ⌂ ⌃ ⌁ ⌄

this part is important and this is more information

תגיות נוספות - רשימות

```
<ul>
    <li>קפה</li>
    <li>חלב</li>
    <li>סוכר</li>
</ul>
```



- קפה
- חלב
- סוכר

```
<ol>
    <li>קפה</li>
    <li>חלב</li>
    <li>סוכר</li>
</ol>
```



1. קפה
2. חלב
3. סוכר

תרגיל חיים

1. בקרו באתר הקואלות



2. בקרו באתר הפינגוונים



תגיות נוספות - רשימות

בתוך התג או שנמצא ברשימה ניתן להציג גם תמונות וכל-tag html כרצוננו,
לדוגמא:

```
<ol>
  = <li>
    <a href="http://www.Koalas.com">
      
      <span>בקרו באתר הקואלאות</span>
    </a>
  </li>
  <li>
    <a href="http://www.Penguins.com">
      
      <span>בקרו באתר הפינגוונים</span>
    </a>
  </li>
</ol>
```

1. בקרו באתר הקואלאות



2. בקרו באתר הפינגוונים



תרגיל ביתה – תגיות HTML

עליכם ליצור אתר קניות של רשות סופרים באופן הבא:

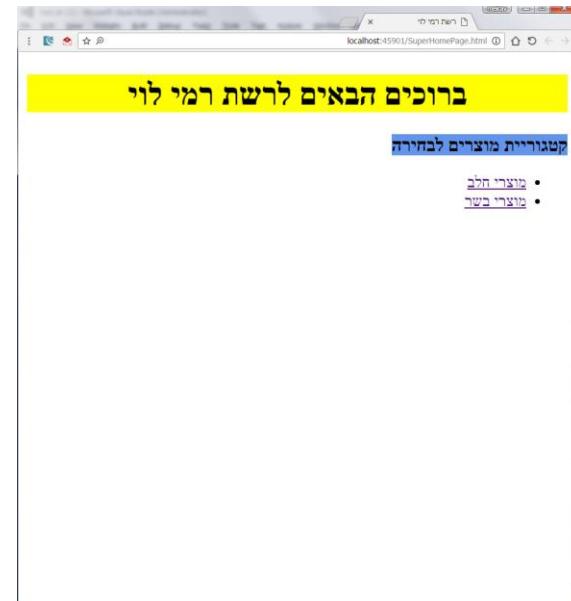
- הדף הראשון יציג את שם הרשות ושתי קטגוריות של מזון (מוצרי חלב ומוצרי בשר).
- בלחיצה על כל קטgorיה יועבר המשתמש לדף המתאים לו ויפיעו 2 תמונות של המוצרים בקטgorיה.

השתמשו בתמונות שנמצאות בתיקיית [pics](#) במודול

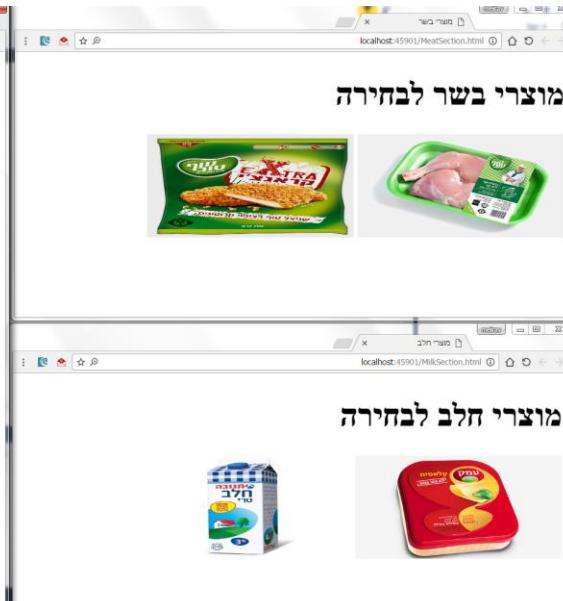
תרגיל ביתה – תגיות HTML

- התמונות יהיו בגודלים זרים
- כותרת הרשת תהיה ממורכצת עם רקע
- כותרת הקטגוריות תהיה עם רקע ולא תתפוך את כל המסר

דף 1



דף 2



מה שונה בתצוגת האלמנטים בקטע הקוד הבא?

```
|<body>  
|  
|<h3 style="background-color:yellow">header 1</h3>  
|<h3 style="background-color:yellow">header 2</h3>  
|<span style="background-color:yellow">span 1</span>  
|<span style="background-color:yellow">span 2</span>  
|<p style="background-color:yellow">paragraph</p>  
</body>
```

רווח מסר ורווח טקסט

מדוע תג `h1-p` תופסים את כל **רווח המסר ?**

ותגי `span` רק את **רווח הטקסט?**

לכל אלמנט `html` יש הגדרות ברירת מחדל עבור מאפיין `display` המגדיר כיצד הוא מוצג:

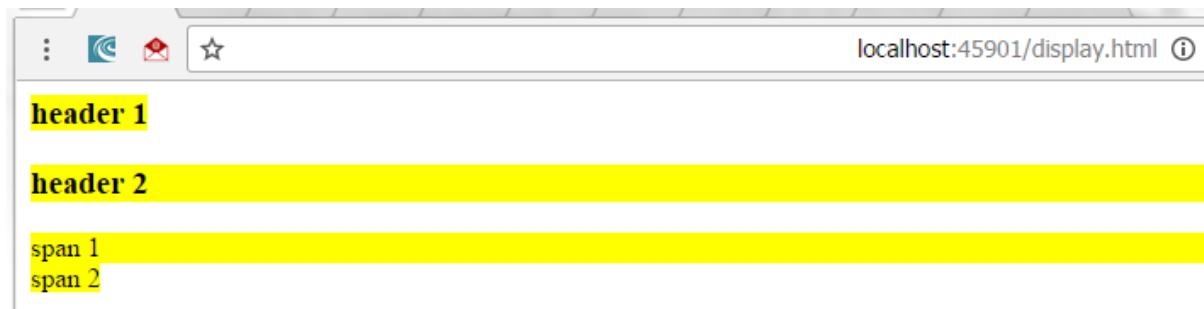
- `tag h` מוגדר ברוחב כל המסר הtekst ואלמנטים אחרים לא יכולים להיות אליו באותה שורה.
- `tag span` מוגדר ברוחב של הטקסט ואלמנטים אחרים יכולים להיות אליו באותה שורה.

display

את הבעה **שבש��ופית** הקודמת ניתן לשנות באמצעות תכונת `display` שנמצאת בתוך תכונת `style`, ערכו התכונה המרכזית הם `none`, `block`, `inline`:

- – האלמנט יתפօס את כל המסר ולא יהיה אליו אלמנטים נוספים בשורה.
- – האלמנט יהיה ברוחב הטקסט ויכולים להיות אליו אלמנטים נוספים באותה שורה.
- – האלמנט לא יוצג במסר.
- או אפשר לעטוף `h` ב `span`
- נקודה פסיק – תחתית יrokeה

```
<h3 style="background-color:yellow; display:inline">header 1</h3>
<h3 style="background-color:yellow;">header 2</h3>
<span style="background-color:yellow; display:block">span 1</span>
<span style="background-color:yellow">span 2</span>
```

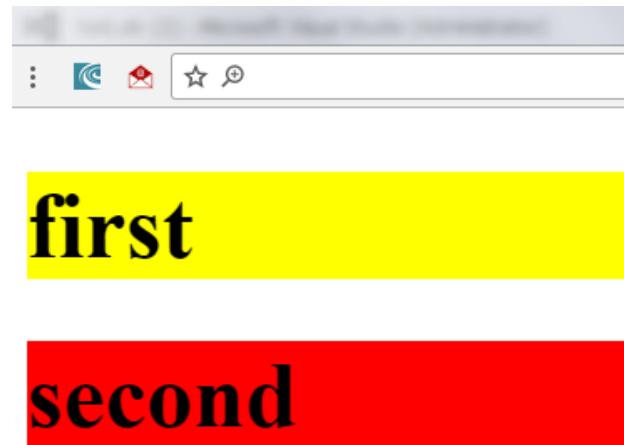


Box Model

Box Model

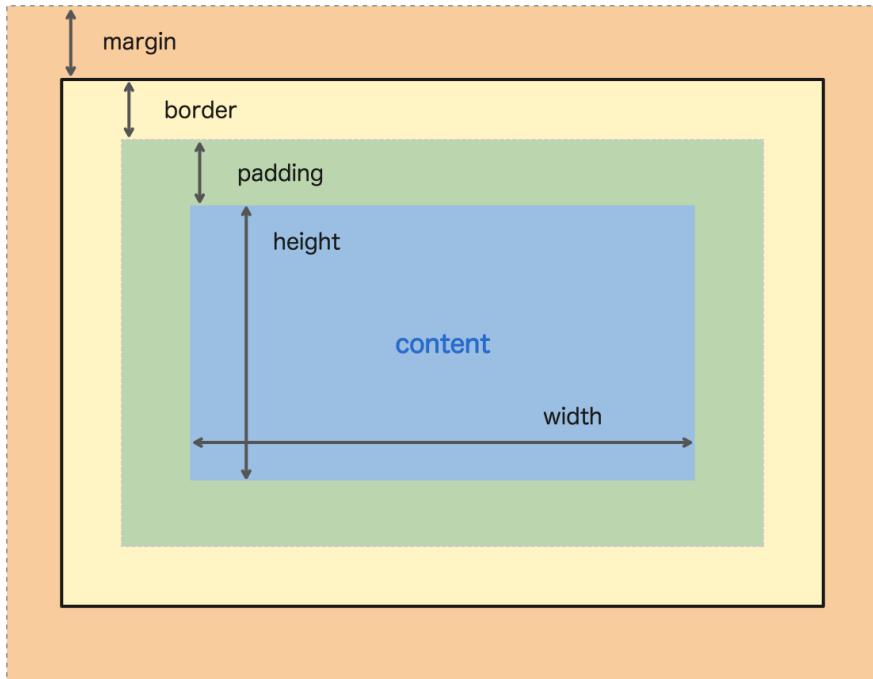
שימוש לב לדבר הבא:

```
<body>
  <h1 style="background-color:yellow">first</h1>
  <h1 style="background-color:red">second</h1>
</body>
```



מדוע יש רווח בין הכותרות?
לא ציינו שום תג `br`
שימוש לב שיש גם רווח למעלה

Box Model



כל אלמנט HTML הוא מעין תיבת :

- הרוחב והגובה שאנו חזו מציינים בתוכנת `height` הם החלק **content**.
- גבול האלמנט.
- המרחק שהאלמנט לוקח מהמסגרת שלו לתוך הכתול **padding**
- רשתה מחוץ לגבול האלמנט (**Margin**)

לכל אלמנט יש ערכי ברירת מחדל: למשל בתגיות `h1` הערכים אינם שוים ל-0

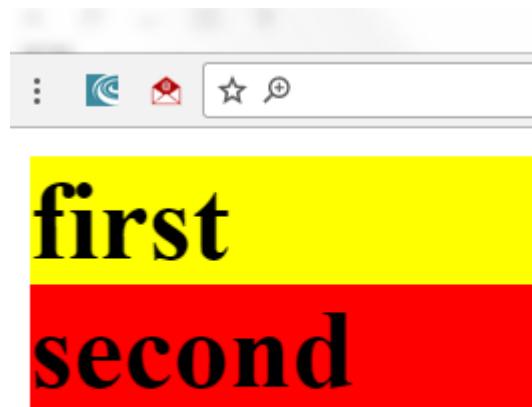
הרוחב שהאלמנט יתפօס בمسך הוא:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

Box Model

איפוא תכונת margin

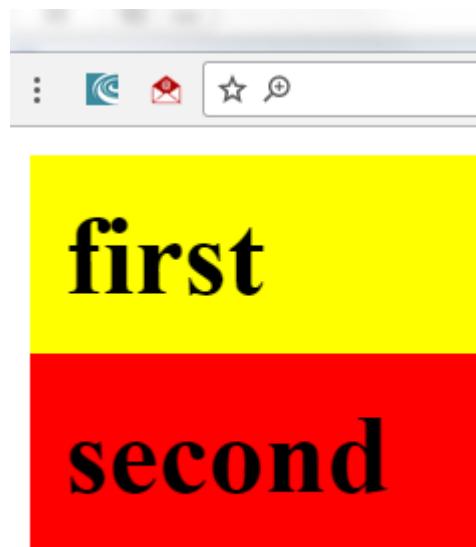
```
<body>
  <h1 style="background-color:yellow; margin:0px">first</h1>
  <h1 style="background-color:red; margin:0px">second</h1>
</body>
```



Box Model

הווגת :padding

```
<body>
  <h1 style="background-color:yellow; margin:0px; padding:10px">first</h1>
  <h1 style="background-color:red; margin:0px; padding:10px">second</h1>
</body>
```



המשר תרגיל ביתה

תנו לכל תמונה את אותו גובה (אל תנתנו רוחב)
הוסיפו עוד תמונות כרצונכם

תנו לכל תמונה

padding •

margin •

border •

מוצרי בשר לבחירה



תרגומ

הרצאה 2

מבנה טבלאי, שימוש בDIV

קורס : צד ל Koh

מרצה: יעל סלע זעירא

טבלה



כאשר נרצה לבנות טבלאי נשתמש ב

<table> <tr> <td>

Table, tr, td

טבלה מורכבת משורות tr ועמודות td

```
<table>
  <tr>
    <td>סטודנט</td>
    <td>צ'יל</td>
  </tr>
  <tr>
    <td>אביב</td>
    <td>95</td>
  </tr>
</table>
```

הוספת גבולות - Table

עבור table פיתחו תכונה שנקראת border והוא מאפיין של ה table מוסיף גבולות לכל האלמנטים:

```
<table border="1" style="border-collapse:collapse">
  <tr>
    <th>סטודנט</th>
    <th>ציון</th>
  </tr>
  <tr>
    <td>אבי</td>
    <td>95</td>
  </tr>
</table>
```



סטודנט	ציון
אבי	95

margin: auto

אם אנחנו רוצים למרכז את הטבלה במרכז הדף, איך נעשה זאת?

1. לשם כך יש את margin: auto שאומר: **שים אותו במרכז (מאוזן)** של מה שאני נמצא בו.

הטבלה אצליינו נמצת בתוך ה body אז הטבלה תהיה במרכז למעלה, אם נרצה שהיא

רווח גם מלמעלה נצרך להוסיף אקס פון margin-top: 20

2. ל table יש גם תכונה של align וונוכל להגדיר אותה כ center

תרגיל חימום

- צרו טבלה עם 3 תמונות של קואלה
- תנו לכל תמונה רוחב, רוחב, padding, margin – ו – border
- מתחת לכל תמונה שימו כותרת (בכל דרך שתבחרו) הכתובת תהיה ממורכזת



קואלה



קואלה



קואלה

טבלה מותאמת אישית

לפעמים אנחנו רוצים להציג טבלה "לא שגרתית":

סיכום ציוני סטודנטים:			
מצטינני נשיא			
שם הסטודנט	ת.ז.	ממוצע	אנו
אבי	144	95	
שי	765	94	

מצטינני דיקן			
שם הסטודנט	ת.ז.	ממוצע	אנו
רונ	872	88	

air
צובעים
שורה בצהוב

סטודנט	קוורט	ציון
אבי	חליליא	90
אבי	פיזיקה	95
כימיה		93

טבלה מותאמת אישית

```
<table border="1" style="border-collapse:collapse">
  <tr>
    <td>
      מצטינני נשיא
    </td>
  </tr>
  <tr>
    <td>שם הסטודנט</td>
    <td>ת.ז</td>
    <td>ממוקע</td>
  </tr>
</table>
```



מצטינני נשיא		
שם הסטודנט	ת.ז	ממוקע

colspan

על מנת שהתא העליון יתפרש על כל התאים, יש צורך להשתמש בתכונת **colspan** על התא שאומרת לו על כמה תאים מעתודות הטרבלה הוא מתפרש:

```
<table border="1" style="border-collapse:collapse">
  <tr>
    <td colspan="3">
      מצטינני נשיא
    </td>
  </tr>
  <tr>
    <td>שם הסטודנט</td>
    <td>ת.ז.</td>
    <td>מזהה</td>
  </tr>
</table>
```

מצטינני נשיא		
שם הסטודנט	ת.ז.	מזהה

טבלה מותאמת אישית

על מנת שהתא העליון יתפרש על שורות, יש צורך להשתמש בתכונת **rowspan** על התא שאומרת לו על כמה שורות משורות הטבלה הוא מתפרש:

```
<table border="1" style="border-collapse:collapse">
  <tr>
    <td rowspan="2" style="text-align: center; vertical-align: middle; font-size: 2em; color: blue; background-color: yellow; padding: 10px; border: none;>אב</td>
    <td style="padding: 10px; border: none;>מצחיק</td>
  </tr>
  <tr>
    <td style="padding: 10px; border: none; text-align: center; vertical-align: middle; font-size: 1.5em; color: red; background-color: yellow; border-top: none;>בב</td>
    <td style="padding: 10px; border: none; text-align: center; vertical-align: middle; font-size: 1.5em; color: red; background-color: yellow; border-top: none;>מאנ</td>
  </tr>
</table>
```

אב	
בב	מאנ

שימוש בהטא נפרש גם על השורה של עצמו בחישוב

תראייל ביתה טבלאות

בחרו לכמ חיה כלשוי, הורידו תמונה שלה מהאינטרנט וצרו עבורה את הדף הבא:

נתונים		תמונה	שם החייה
קטגוריה	ערך		
משקל	200 ק"ג		
מהירות	80 קמ"ש		אריה
תוחלת חיים	15 שנה		

הסרט האהוב עליו : מלך האריות

div

- אלמנט המהווה קונטינר
- מאפשר לסדר אלמנטים במקומות שונים בדף
- Display: block •
- אלמנט בסיסי ביצירת עימוד לדף (layout)

div

Div שאין בתוכו אלמנטים לא יופיע, אלא אם הגדרנו עבורה גובה בצורה מפורשת:

`<div style="background-color:red"></div>` לא יציג כלום//

`<div style="background-color:red; height:30px"></div>`



div

על ידי vip ניתן להציג אלמנטים אשר נמצאים בתוך המיכל.

```
<div style="text-align: center;">Hello div</div>
```

נסתכל על span :

```
<span style="text-align: center;">Hello span</span>
```

בwhats הטעות לא זו ל center מדוע?

div

- נסתכל איך מתנהג DIV
 - צרו DIV ותנו לו רקע
 - רואים מהهو? למה לא?
 - מה צריך להוסיף ← הכניסו את התמונה על מנת לתת לו גובה.
 - הכניסו לתוכו גם כותרת
- ה DIV גדול בהתאם למה שיש בתוכו – נשים על ה DIV כל מה שבתוכו יוז בהתאם.



float

- נולד צורר (למשל בעיתונות) לשים תמונה שייטוף אותה טקסט באופן הבא:

Float

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet.

Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Carol McKinney Highsmith (born Carol Louise McKinney on May 18, 1946) is an American photographer, author, and publisher who has photographed in all the states of the United States, as well as the [District of Columbia](#), and [Puerto Rico](#). She photographs the entire American vista (including landscapes, architecture, urban and rural life, and people in their work environments) in all fifty [U.S. states](#) as a record of the early 21st century.

- נראה בהמשך עוד צורת לימוד `flexbox`.

Clear float

- ראיינו ש float יוצא מה normal flow ואלמנטים אחרים יופיעו סביבו.
- אם אנו רוצים "לנקות" את האלמנטים האחרים מהשפעת ה float ורוצים

שם יופיעו ב normal flow נשתמש ב clear:both

סיכום float

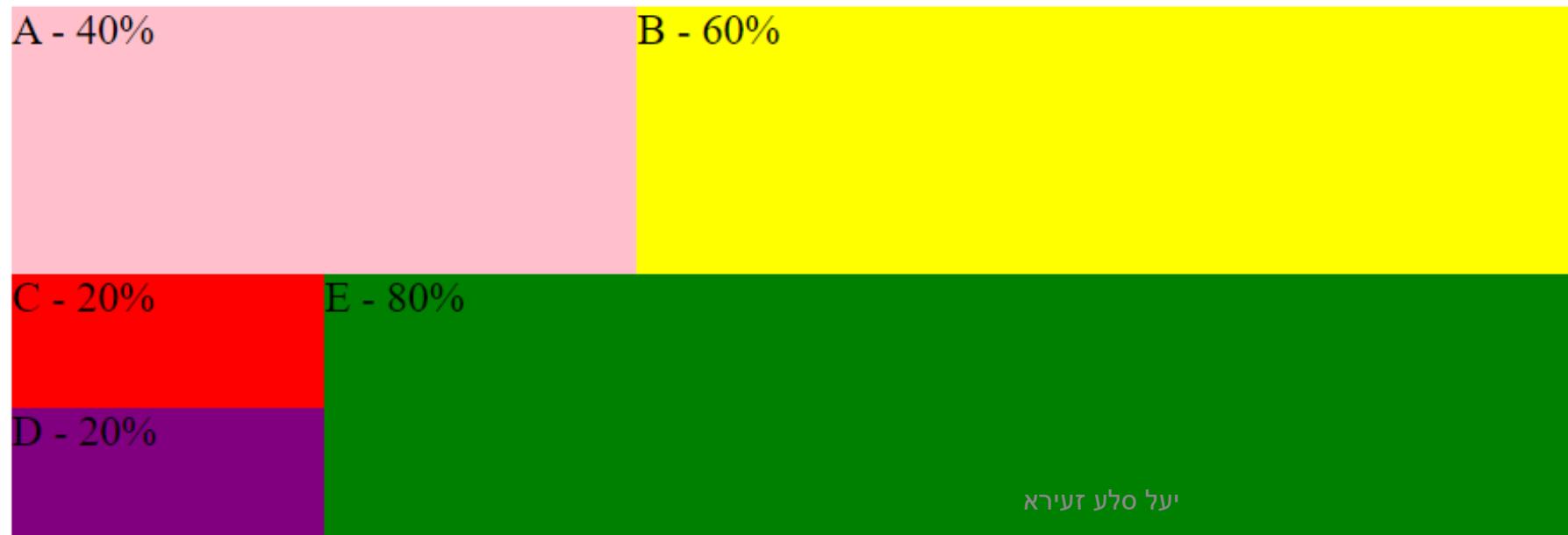
- מציף את האלמנט בדף.
- חשוב לחת גודל לאלמנט שהוא ב float, פיקסלים/אחוזים על מנת לציין כמה שטח האלמנט יתפוצט מרוחב המסר.
- – האלמנט יהיה מימין float:right
- – האלמנט יהיה משמאלי float:left

דוגמא חלוקת div



תרגיל כיתה VII

- יש ליצור את הדף הבא-
- הדף בנווי משתי שורות – שורה ראשונה מכילה שתי קוביות . שורה שנייה מכילה שתי קוביות, כאשר הקובייה השמאלית מחלוקת ל2 קוביות נוספים.
- הקפידו על שימוש בcontainer'ים במקום המתאים.
- גובה כל קובייה – $\text{ax}100$
- רוחב בהתאם לאחוזים



שאלות?

הרצאה 3

CSS

קורס : צד ללקוח

מרצה: יעל סלע זעירא

Cascading Style Sheet

- גליונות עיצוב מדורגים
- שפת עיצוב שהופכת כתיבת דפי HTML לנקייה ופשטונה יותר
- Look & Feel
- קלה ללמידה
- כלי חזק מאוד

יתרונות

- חוסר זמן
- הדף נטען מהר יותר – גורם לכך שיש פחות קוד ולכן הדף עולה מהר יותר
- קל לתחזוקה
- אפשר עיצוב עשיר יותר מאשר רק HTML
- סטנדרט

תגית style בתוך תגית head

בஹמישר נראה שה CSS נכתב
בקובץ חיצוני ומשויך לדף
הנדרש

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
    <style>
        ...
    </style>
</head>
<body>
</body>
</html>
```

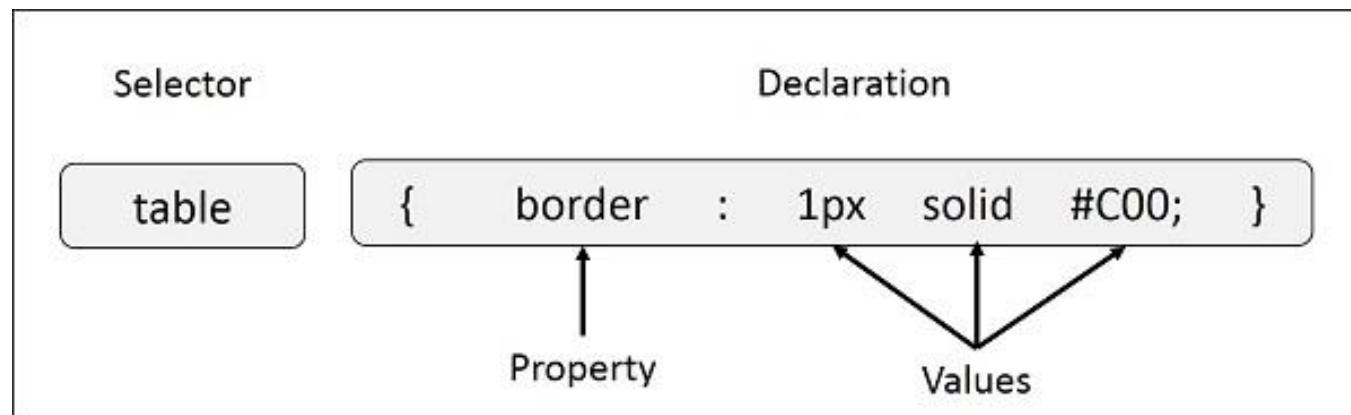
CSS Syntax

```
selector {  
    property: value  
}
```

name/ID/HTML : **Selector**

property : **Property**

value : **Value**



Selector Type – Tag, Class, Id

```
h1{  
    color:red;          tag selector  
}  
  
.red  
{  
    color:red;          class selector  
}  
  
#myH1  
{  
    color:red;          id selector  
}
```

Tag selector

```
<html>
<head>
    <title></title>

    <style>
        h1{
            color:red
        }
    </style>
</head>
<body>
    <h1>header 1</h1>
    <h1>header 2</h1>
    <h1>header 3</h1>
</body>
</html>
```

העיצוב יחול אוטומטית על ה태ג



header 1

header 2

header 3

*קודם עבדנו בرمת השורה של אלמנט ושםנו לכל `h1` את התוכנה באופן ישיר.

Tag selector

ניתן להגיד כמה פעמים שורצים עיצוב לאותו Tag
מה צבע הtekסט והרקע?

header 1

header 2

header 3

*במידה ונדרשו תכונה בהמשך – הערך החדש הוא מה שיוגדר.

Tag selector

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    h1, p{
      color:red;
    }
  </style>
</head>
<body>
  <h1>header 1</h1>
  <h1>header 2</h1>
  <h1>header 3</h1>

  <p>paragraph</p>
</body>
</html>
```

ניתן להגדיר בפעם אחת מספר תגים שונים ע"י הפרדה של פסיק,
למשל:

header 1

header 2

header 3

paragraph

Class selector

```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <h1 class="red">header 1</h1>
    <h1>header 2</h1>
    <h1>header 3</h1>

    <p class="red">paragraph</p>
</body>
</html>
```

מה קורה אם רצים לשים תכונות רק על הפייסקה ועל הcotרת הראשונה?
כלומר רצים להתייחס לאלמנטים מסוימים קבוצה ללא קשר לסוג האלמנט.

- במצב זה ניתן להשתמש בתכונת class (מחלקה) ולהגדיר מחלוקת באותו שם לכל מי שנרצה.

*שם הקבוצה ניתן כרצונכם, כמוון כדי עם משמעות כלשהי.

Class selector

```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>

    <style>
        .red{
            color:red;
        }
    </style>
</head>
<body>
    <h1 class="red">header 1</h1>
    <h1>header 2</h1>
    <h1>header 3</h1>

    <p class="red">paragraph</p>
</body>
</html>
```

לאחר שהגדכנו מחלוקת, ניתן להשתמש בשם שלה בצורה הבאה על מנת לישם תכונות עיצוב:

header 1

header 2

header 3

paragraph

***שימו לב לשימוש בנקודה (.) לפני שם המחלוקת, זה מה שאומר למחשב שמדובר במחלוקת ולא אלמנט כמו שראינו קודם.**

Id selector

```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>

    <style>
        #myH1
        {
            color:red;
        }
    </style>
</head>
<body>
    <h1>header 1</h1>
    <h1 id="myH1">header 2</h1>
    <h1>header 3</h1>

    <p>paragraph</p>

</body>
</html>
```

אפשרה נוספת לחת תכונות עיצוב לאלמנט לפי ת.ז. – id. נסיף לאלמנט תכונה שנקראת id וניתן לה שם כרצוננו (באנגלית).

- נעצב את האלמנט בתג style ע"י שימוש ב-#.

header 1

header 2

header 3

paragraph

*ת.ז. הוא משזה ייחודי לאלמנט ואין לכתוב שני אלמנטים עם אותו id.

*באופן עקרוני ניתן להשיג את זה ע"י שימוש במחלקה רק על האלמנט זהה.

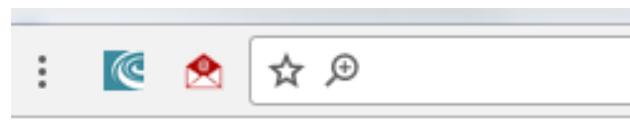
CSS selectors

ניתן לשלב סוגים שונים של סלקטורים, במידה ויש להם תכונות זהות כմובן:

```
.headers, #first, p
{
    background-color:yellow;
    color:greenyellow
}
```

תרגיל ביתה - CSS

יש לישם את העיצוב הבא **ללא** שימוש ב- `:style inline`



My first h1

My second h1

My first p

My first span

*אלמנט יכול לקבל עיצוב מכמה מקומות:

- גם שימוש בתג `class`
- גם שימוש ב-`id`

למה גילוונות עיצוב מדורגים?

```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>

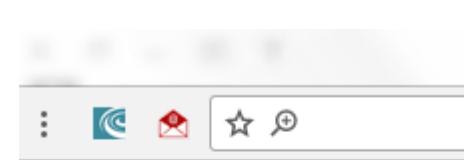
    <style>
        .red
        {
            background-color:red
        }

        h1{
            background-color:yellow;
        }

    </style>
</head>
<body>
    <h1 class="red">header 1</h1>

</body>
</html>
```

מה צבע הרקע של הכותרת?
יש משמעות לעוצמת ה-**selector**
class חזק מtag, גם אם הוא נרשם לפני



למה גילוונות עיצוב מדורגים?

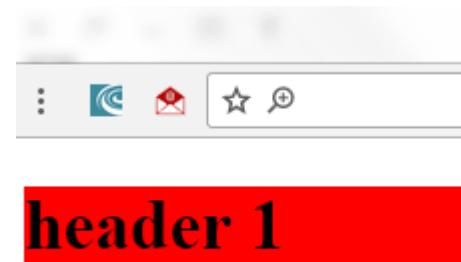
```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>

    <style>
        .red
        {
            background-color:red;
            color:blue;
        }

        .red{
            color:black;
        }

    </style>
</head>
<body>
    <h1 class="red">header 1</h1>
</body>
</html>
```

מה צבע **הtekst** ו**הרקע** של הכותרת?
יש משמעות למי מגיע אחרי מי אשר עוצמת ה-**selector** זהה,
ולתמונה כבר הוגדר ערך



למה גילוות עיצוב מדורגים?

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    #myH1
    {
      background-color:yellow;
    }

    .red
    {
      background-color:red;
    }

  </style>
</head>
<body>
  <h1 class="red" id="myH1">header 1</h1>
</body>
</html>
```

header 1

מה צבע הרקע של הכתובת?
או חזק מ-class

למה גילוונות עיצוב מדורגים?

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>

    #myH1
    {
      background-color:yellow;
    }

    .red
    {
      background-color:red;
    }

  </style>
</head>
<body>
  <h1 class="red" id="myH1" style="background-color:green">header 1</h1>
</body>
</html>
```

מה צבע הרקע של הכותרת?
ברמת האלמנט חזק מ-id Style

header 1

סיכום ביניים – חוקי עיצוב

כל המקומות בהם הגדרנו CSS מתחברים לגילוון אחד לפי סדר שורות.
לאחר מכן מתבצע חישוב לכל תכונת CSS, על סמך החוקים שראינו.

הורשת תכונות שמאלי מנוטים מעל

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    #myDiv
    {
      color:red;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <h1>header 1</h1>
  </div>
</body>
</html>
```

מה צבע הtekst של הכותרת?
צבע הרקע מתקיים מה- div שהכותרת נמצאת בתוכו, הרקע הוגדר ל-div
ולא ישירות לכותרת

header 1

הורשת תכונות שמאליינים מעל

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    #myDiv
    {
      color:red;
    }

    h1{
      color:blue;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <h1>header 1</h1>
  </div>
</body>
</html>
```

מה צבע הטקסט של הכותרת?

גם אם עוצמת הסלקטור של מי שמעליי חזקה יותר באופן טכני, **היא לא קודמת**
لتכונות שמתיחסות ישירות אליו עם משקל נמוך יותר

header 1

סיכום – חוקי עיצוב

כל המקומות בהם הגדרנו CSS מתחברים לגילוון אחד לפי סדר שורות.
לאחר מכן מtbody חישוב לכל תכונת CSS, על סמך החוקים שראינו.

ישיר/עקייף	המשקל	עוצמת עורך נוכחי	תכונה

המשקל של סלקטור Tag – 1
המשקל של סלקטור class – 10
המשקל של סלקטור id – 100
המשקל של style inline – 1000

התכונה עם המשקל הכי גבוה שמצוינת אחרונה היא זאת שתוישם על האלמנט.
התוישות ישירה לאלמנט תמיד חזקה יותר מהתוישות עקייפה.

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

  <style>
    #myDiv
    {
      color:red;
    }

    h1{
      color:blue;
      background-color:yellow;
    }

    .headers
    {
      color:green;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <h1 class="headers">header 1</h1>
  </div>
</body>
</html>
```

לוגמא

בהינתן קטע הקוד הבא נ עובר על השלבים
לקביעת העיצוב של הכותרת:

```
<html>
<head>
  <meta charset="utf-8" />
  <title></title>

<style>
  #myDiv
  {
    color:red;
  }
  h1{
    color:blue;
    background-color:yellow;
  }
  .headers
  {
    color:green;
  }
</style>
</head>
<body>
  <div id="myDiv">
    <h1 class="headers">header 1</h1>
  </div>
</body>
</html>
```

לוגמא

החלק הראשון →

בහינת קטע הקוד הבא עובר על שלבים
לקביעת העיצוב של הכותרת:

תכונה	ערך הנוכחי	עוצמת המשקל	ישיר/עקיף
color	red	100	עקיף
			על סלע זעירא

```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>

    <style>
        #myDiv
        {
            color:red;
        }

        h1{
            color:blue;
            background-color:yellow;
        }

        .headers
        {
            color:green;
        }
    </style>
</head>
<body>
    <div id="myDiv">
        <h1 class="headers">header 1</h1>
    </div>
</body>
</html>
```

דוגמה

בהינתן קטע הקוד הבא נ עובר על שלבים
לקביעת העיצוב של הכותרת:

החלק השני →



תכונה	ערך הנוכחי	יעיר/עירוף	עוצמת המשקל
color	blue	ישיר	1
Background-color	yellow	ישיר על סלע צעירה	1

```

<html>
<head>
    <meta charset="utf-8" />
    <title></title>

    <style>
        #myDiv
        {
            color:red;
        }

        h1{
            color:blue;
            background-color:yellow;
        }

        .headers
        {
            color:green;
        }
    </style>
</head>
<body>
    <div id="myDiv">
        <h1 class="headers">header 1</h1>
    </div>
</body>
</html>

```

דוגמה

בהינתן קטע הקוד הבא נ עובר על השלבים
לקביעת העיצוב של הכותרת:

החלק השלישי

תמונה	ערך הנוכחי	עוצמת משקל	ישיר/עקיף
color	green	10	ישיר
Background-color	yellow	1	ישיר

על סלע זעירא

הרחבה – CSS selectors

The Universal Selectors

```
* {  
color: #000000;  
}
```

כל האלמנטים בדף יהיו שחורים

הרחבה – CSS selectors

The Descendant Selectors

```
ul span {  
color: red;  
}
```

כל span שנמצא בתוך ul הtekst שלו יהיה אדום

הרחבה – CSS selectors

```
h1.black {  
color: red;  
}
```

‘צבע באדום רק אלמנטים מסווג כי בעלי class black’

הרחבה – CSS selectors

The Child Selectors

```
body > p {  
color: red;  
}
```

יצבע באדום רק אלמנטים מסווג ק שהם ילדים ישירים של ה body

הרחבה – CSS selectors

The Attribute Selectors

```
input[type = "text"] {  
color: red;  
}
```

יצבע באדום רק אלמנטים מסווגת שהם מסווג `text`

:hover Selector

```
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    a:hover {
      background-color: yellow;
    }
  </style>
</head>

<body>
<a href="1.html">
Go to with hover
</a>
</body>
```

משנה את ה style רק שהעכבר נמצא על
האלמנט

```
p:hover, h1:hover, a:hover {
  background-color: yellow;
}
```

```
div p {  
    background-color: yellow;  
}
```

הרחבה

<h2>Descendant Selector</h2>

<p>The descendant selector matches all elements that are descendants of a specified element.</p>

```
<div>  
    <p>Paragraph 1 in the div.</p>  
    <p>Paragraph 2 in the div.</p>  
    <section><p>Paragraph 3 in the div.</p></section>  
</div>
```

```
<p>Paragraph 4. Not in a div.</p>  
<p>Paragraph 5. Not in a div.</p>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

```
div > p {  
    background-color: yellow;  
}
```

הרחבה

<h2>Child Selector</h2>

<p>The child selector (>) selects all elements that are the children of a specified element.</p>

```
<div>  
    <p>Paragraph 1 in the div.</p>  
    <p>Paragraph 2 in the div.</p>  
    <section>  
        <!-- not Child but Descendant --&gt;<br/>        <p>Paragraph 3 in the div (inside a section element).</p>  
    </section>  
    <p>Paragraph 4 in the div.</p>  
</div>  
  
<p>Paragraph 5. Not in a div.</p>  
<p>Paragraph 6. Not in a div.</p>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div (inside a section element).

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

הרחבה

```
p.intro {  
    background-color: yellow;  
}
```

- מה קורה במקרה הבא?

```
<h1>Demo of the element.class selector</h1>
```

```
<div class="intro">  
    <p>My name is Donald.</p>  
    <p>I live in Duckburg.</p>  
</div>
```

```
<p>My best friend is Mickey.</p>
```

```
<p class="intro">My best friend is Mickey.</p>
```

My name is Donald.

I live in Duckburg.

My best friend is Mickey.

My best friend is Mickey.

```
<head>

<style>

#header1.callout {

color: red;

}

</style>

</head>

<body>

<div id="header1" class="callout">

<p>this is p1</p>

<p>this is p2</p>

</div>

</body>
```

this is p1

this is p2

```
<head>  
  
<style>  
  
#header1 .callout {  
  
    color: red;  
  
}  
  
</style>  
  
</head>  
  
<body>  
  
<div id="header1" >  
  
    <p class="callout">this is p1</p>  
  
    <p>this is p2</p>  
  
</div>  
  
</body>
```

this is p1

this is p2

```
:not(p) {  
    background: #ff0000;  
}
```

```
<style>  
p {  
    color: #000000;  
}  
  
:not(p) {  
    color: #ff0000;  
}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
  
<p>This is a paragraph.</p>  
<p>This is another paragraph.</p>  
  
<div>This is some text in a div element.</div>  
  
<a href="https://www.w3schools.com" target="_blank">Link to  
W3Schools!</a>
```

הרחבה

This is a heading

This is a paragraph.

This is another paragraph.

This is some text in a div element.
[Link to W3Schools!](https://www.w3schools.com)

```
p:nth-child(2) {  
    background: red;  
}
```

הרחבה

```
<style>  
p:nth-child(2) {  
    background: red;  
}  
</style>  
</head>  
<body>  
  
<p>The first paragraph.</p>  
<p>The second paragraph.</p>  
<p>The third paragraph.</p>  
<p>The fourth paragraph.</p>
```

The first paragraph.

The second paragraph.

The third paragraph.

The fourth paragraph.

```
p:nth-child(odd) {  
    background: red;  
}  
  
p:nth-child(even) {  
    background: blue;  
}
```

```
<p>The first paragraph.</p>  
<p>The second paragraph.</p>  
<p>The third paragraph.</p>
```

The first paragraph.

The second paragraph.

The third paragraph.

תרגיל ביתה

- עבד על קובץ css target ex.html שנה אותו לפי הנקודות במסמך ה word
שירה בצורה הבאה

p11	<i>p13</i>
p12	<i>p14</i>
<i>p21</i>	p23
<i>p22</i>	p24

p31

p32

Colors in CSS

16,777,216 אפשרויות

- כ- 140 צבעים – Color Names •
- – בין (0,0,0) ל (255,255,255), שחור ולבן בהתאם.
- – בין #FFFFFF ל #000000, שחור ולבן בהתאם.

```
p{  
    color: green;  
}  
  
p{  
    color: rgb(0, 255, 0);  
}  
p{  
    color: #00fe00;  
}
```

Font size with px

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      font-size: 40px;
    }

    h2 {
      font-size: 30px;
    }

    p {
      font-size: 14px;
    }
  </style>
</head>
<body>

  <h1>This is heading 1</h1>
  <h2>This is heading 2</h2>
  <p>This is a paragraph.</p>
  <p>This is another paragraph.</p>

</body>
</html>
```

Font size with em

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      font-size: 2.5em; /* 40px/16=2.5em */
    }
    h2 {
      font-size: 1.875em; /* 30px/16=1.875em */
    }
    p {
      font-size: 0.875em; /* 14px/16=0.875em */
    }
  </style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>
  Specifying the font-size in em allows all major browsers to resize the text.
  Unfortunately, there is still a problem with older versions of IE. When resizing the text, it becomes larger/smaller
  than it should.
</p>

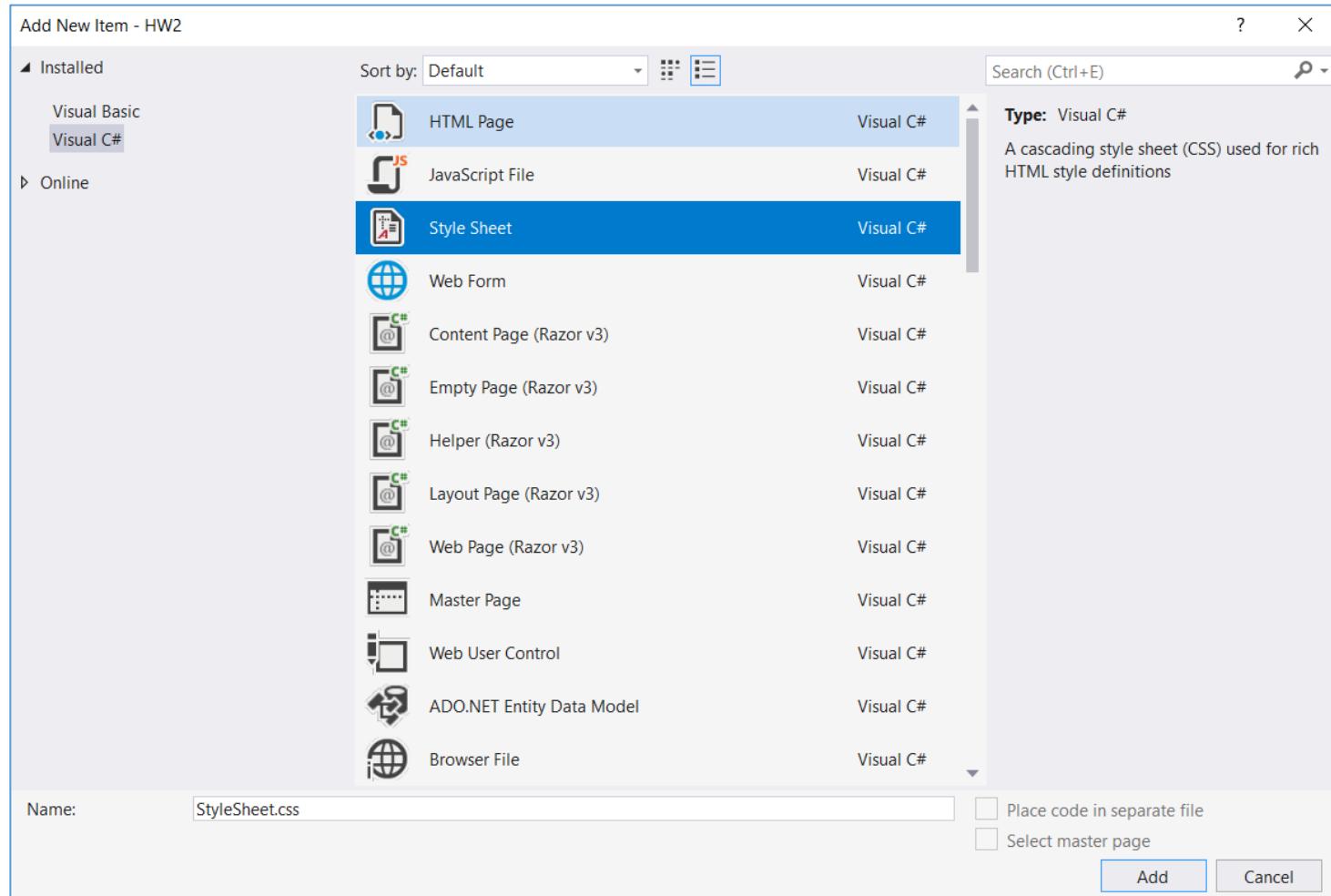
</body>
</html>
```

למשל אם גודל הטקסט הדיפולטיבי
בראזר הוא 16px

CSS Comments

```
/* This is a CSS comment  
It can be broken onto  
multiple lines. */
```

הוצתה CSS לקובץ חיצוני



CSS חיצוני

The screenshot shows a Windows application window with a code editor and a Solution Explorer. The code editor has tabs for 'HtmlPage.html', 'HtmlPage2.html', and 'StyleSheet.css'. The 'StyleSheet.css' tab is active, displaying the following CSS code:

```
1 #yael {  
2     color: red;  
3 }  
4 #yael1{  
5     color: blue;  
6 }  
7 .h1BG {  
8     background-color: yellow;  
9 }  
10
```

The Solution Explorer on the right shows a solution named 'css example' with three files: 'HtmlPage.html', 'HtmlPage2.html', and 'StyleSheet.css', with 'StyleSheet.css' currently selected.

```
<html>
<head>
  <style>
    #yael {
      color: green;
    }
  </style>
  <link href="StyleSheet.css" rel="stylesheet" />
</head>
<body>
  <h1 id="yael" class="h1BG">YAEL</h1>
  <h2 id="yael1" class="h1BG">Yael</h2>
</body>
</html>
```

Style1.css

```
#yael {
  color: red;
}
```

YAEL

Yael

```
<html>
<head>
  <style>
    #yael {
      color: green;
    }
    #yael.h1BG{
      color: pink;
    }
  </style>
  <link href="StyleSheet.css" rel="stylesheet" />
</head>
<body>
  <h1 id="yael" class="h1BG">YAEL</h1>
  <h2 id="yael1" class="h1BG">Yael</h2>
</body>
```

Style1.css

```
#yael {
color: red;
}
#yael1{
color: blue;
}
.h1BG {
background-color: yellow;
}
```

YAEL

Yael

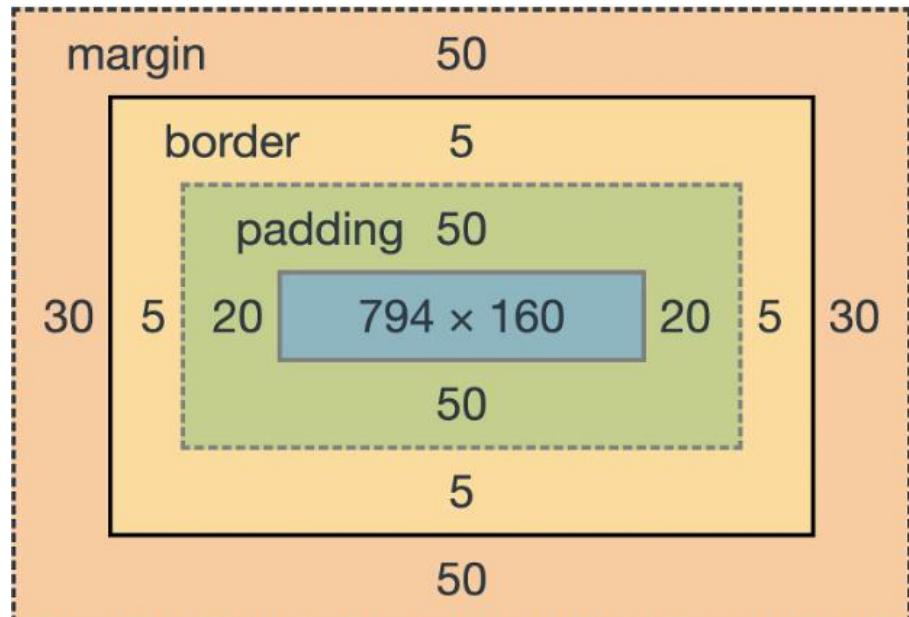
View and change CSS

- Open Chrome DevTool (F12)
- Inspect element
- Add a CSS declaration to an element
- Add a CSS class to an element

لينكيم נוספים

<https://csszengarden.com/pages/alldesigns/>

Box Sizing



- כפי שראינו בהרצאה 1 element-model מקבל גובה ורוחב דיפולטיבי
- אם לאלמנט יש border או padding הם מתווספים לאורך / הרוחב
- לכן עבור 4 תיבות כלומר 4 אלמנטים כאשר כל אחד הוא `width:25%` במידה ויש להם padding או border, לא יכולים יצליחו להיכנס בשורה אחת
- לשם כך נשתמש בתוכנה של `box-sizing`

סוגי Box-Sizing

Content-Box (ברירת המחדל)

- רוחב וגובה האלמנט כוללים רק את תוכן האלמנט.
- ה- padding וה- border מתווספים לרוחב וגובה שנקבעו, כך שהאלמנט בפועל תופס יותר מקום מאשר גדר.

box-sizing: content-box;

Border-Box

- רוחב וגובה האלמנט כוללים את התוכן ה- padding וה- border.
- מאפשר לאלמנט להישאר בתווך הרוחב וגובה שנקבעו, מה שמקל על תכנון הפריסה

box-sizing: border-box;

Example

```
div {  
    box-sizing: content-box;  
    width: 100%;  
}  
Add border and padding but with border-box
```

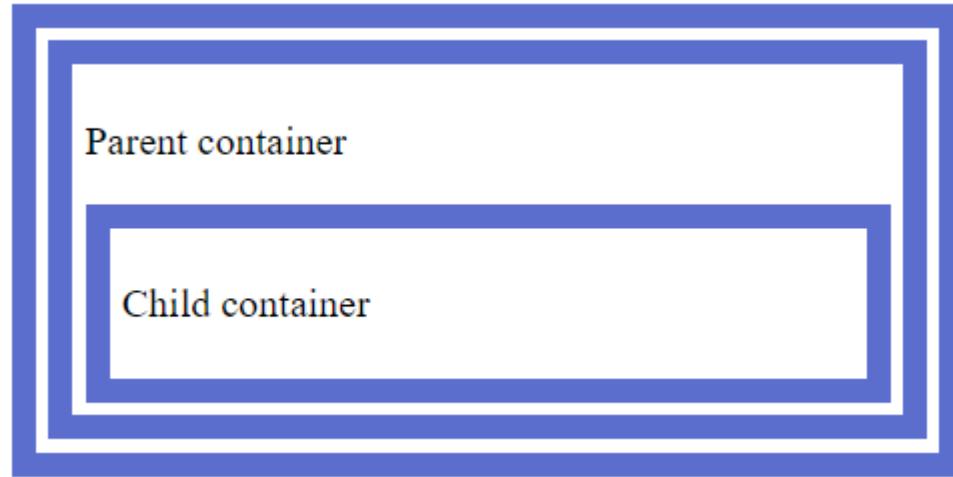
```
div {  
    box-sizing: border-box;  
    width: 100%;  
    border: solid #5B6DCD 10px;  
    padding: 5px;  
}
```

Add border and padding using default `box-sizing: content-box`

```
div {  
    width: 100%;  
    border: solid #5B6DCD 10px;  
    padding: 5px;  
}
```

```
<div class="left">  
    <div>  
        <p>Parent container</p>  
        <div>  
            <p>Child container</p>  
        </div>  
    </div>  
</div>
```

.left { width: 25%; }



box-sizing : border-box;

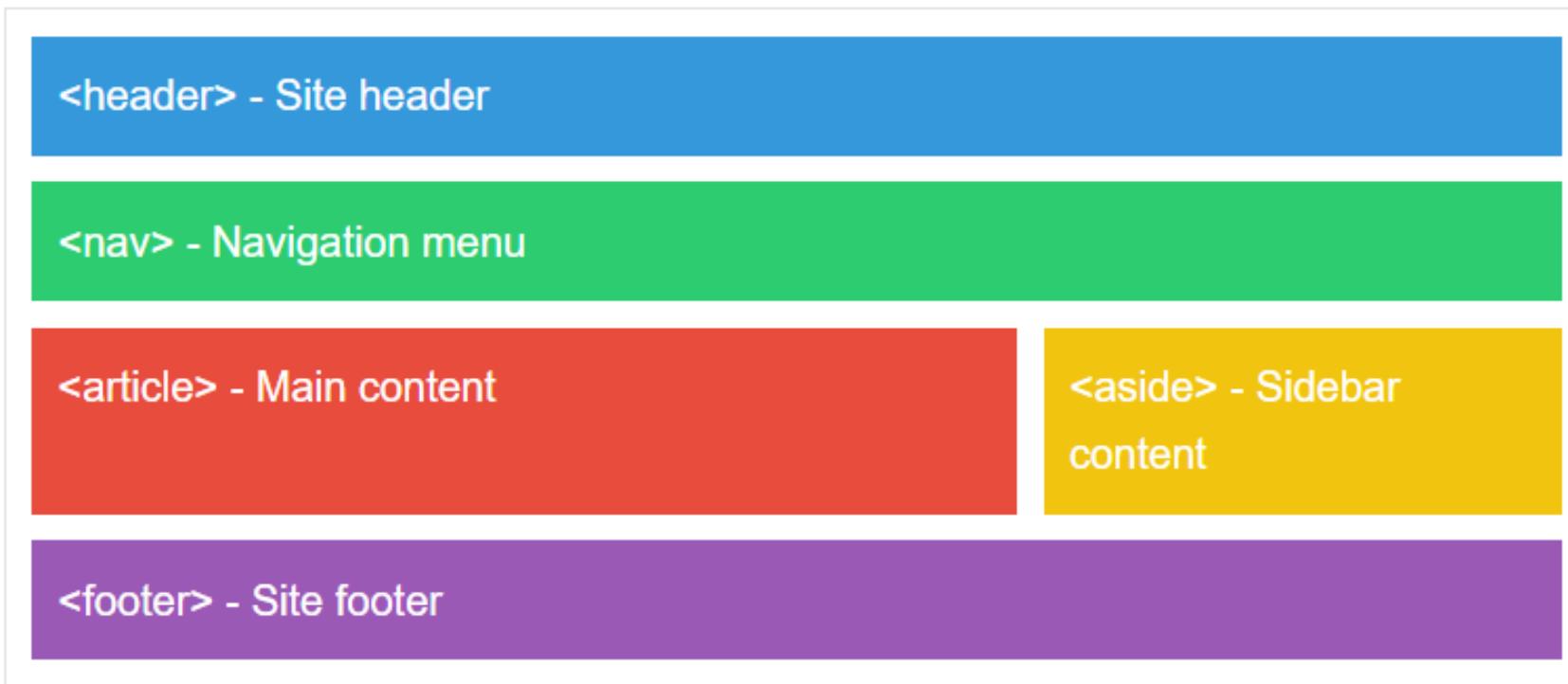
The width and height properties include the content, padding, and border, but do not include the margin. Note that padding and border will be inside of the box



box-sizing: content-box;

This is the initial and default value as specified by the CSS standard. The width and height properties include the content, but does not include the padding, border, or margin

HTML5 Semantic elements



- אלמנטים בעלי שםמשמעותי המתאר את תפקידם
- האלמנטים יתנהגו כזאת
- מספקים מבנה ברור יותר וקריא יותר למכונת SEO
- מאפשר גישה עבור SEO

HTML5 Semantic elements

Elements	Description
<u><article></u>	Defines an independent, self-contained content.
<u><aside></u>	Defines a section with additional information related to the content around the aside element.
<u><dialog></u>	Specifies a dialog box or window.
<u><header></u>	Defines a header of a page or a section.
<u><main></u>	Specifies a document's main content.
<u><nav></u>	Defines a block of navigation links, either within the current document or to other documents.
<u><progress></u>	Displays the progress of the task (progress bar).
<u><section></u>	Creates standalone sections within a webpage containing logically connected content.

Semantic elements

```
<div id="header"></div>  
<div class="section">  
  <div class="article">  
    <div class="figure">  
      <img>  
      <div class="figcaption"></div>  
    </div>  
  </div>  
</div>  
<div id="footer"></div>
```

```
<header></header>  
<section>  
  <article>  
    <figure>  
      <img>  
      <figcaption></figcaption>  
    </figure>  
  </article>  
</section>  
<footer></footer>
```

תרגיל ביתה מתקדם CSS

- בתרגיל הבא נתון לך קובץ שלד עליו לעבוד
- אין לשנות את האלמנטים או להוסיף אלמנטים
- יש להוסיף לקובץ השلد עיצובים בלבד – מחלקות הCSS כבר נתונות לכם עליהם להוסיף רק את התוכנות של כל מחלוקת
- התוצאה הסופית:

The screenshot shows a web page with a purple header containing the text "CSS Practice Exercise". Below the header is a grey navigation bar with links for "Home", "About", "Services", and "Contact". The main content area contains three cards, each with a different background color (white, light orange, and light blue) and rounded corners. The first card is labeled "Card 1" and contains the text: "This is the content of the first card. It showcases the use of CSS to create a simple card layout with a box shadow and rounded corners." The second card is labeled "Card 2" and contains the text: "This is the content of the second card. It also uses the same CSS styles as the first card to maintain a consistent visual appearance." The third card is labeled "Card 3" and contains the text: "This is the content of the third card. It demonstrates the application of CSS to create a clean and modern-looking layout."

Card 1

This is the content of the first card. It showcases the use of CSS to create a simple card layout with a box shadow and rounded corners.

Card 2

This is the content of the second card. It also uses the same CSS styles as the first card to maintain a consistent visual appearance.

Card 3

This is the content of the third card. It demonstrates the application of CSS to create a clean and modern-looking layout.

CSS Practice Exercise

[Home](#) [About](#) [Services](#) [Contact](#)

Card 1

This is the content of the first card. It showcases the use of CSS to create a simple card layout with a box shadow and rounded corners.

Card 2

This is the content of the second card. It also uses the same CSS styles as the first card to maintain a consistent visual appearance.

Card 3

This is the content of the third card. It demonstrates the application of CSS to create a clean and modern-looking layout.

```
<style>
  /* General Styles */
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  /* Header Styles */
  header {
  }

  /* Navigation Styles */
  nav {
  }
  nav a {
  }
  nav a:hover {
  }

  /* Main Content Styles */
  main {
  }
  .card {
  }
  .card:nth-child(2) {

  }
  .card:nth-child(3) {
  }
  .card h2 {
  }
  .card p {
  }

  /* Footer Styles */
  footer {
  }
</style>
```

```
<body>
  <header>
    <h1>CSS Practice Exercise</h1>
  </header>

  <nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Services</a>
    <a href="#">Contact</a>
  </nav>

  <main>
    <div class="card">
      <h2>Card 1</h2>
      <p>This is the content of the first card. It showcases the use of CSS to create a simple card layout with a box shadow and rounded corners.</p>
    </div>

    <div class="card">
      <h2>Card 2</h2>
      <p>This is the content of the second card. It also uses the same CSS styles as the first card to maintain a consistent visual appearance.</p>
    </div>

    <div class="card">
      <h2>Card 3</h2>
      <p>This is the content of the third card. It demonstrates the application of CSS to create a clean and modern-looking layout.</p>
    </div>
  </main>

  <footer>
    <p>&copy; 2024 CSS Practice Exercise</p>
  </footer>
</body>
```

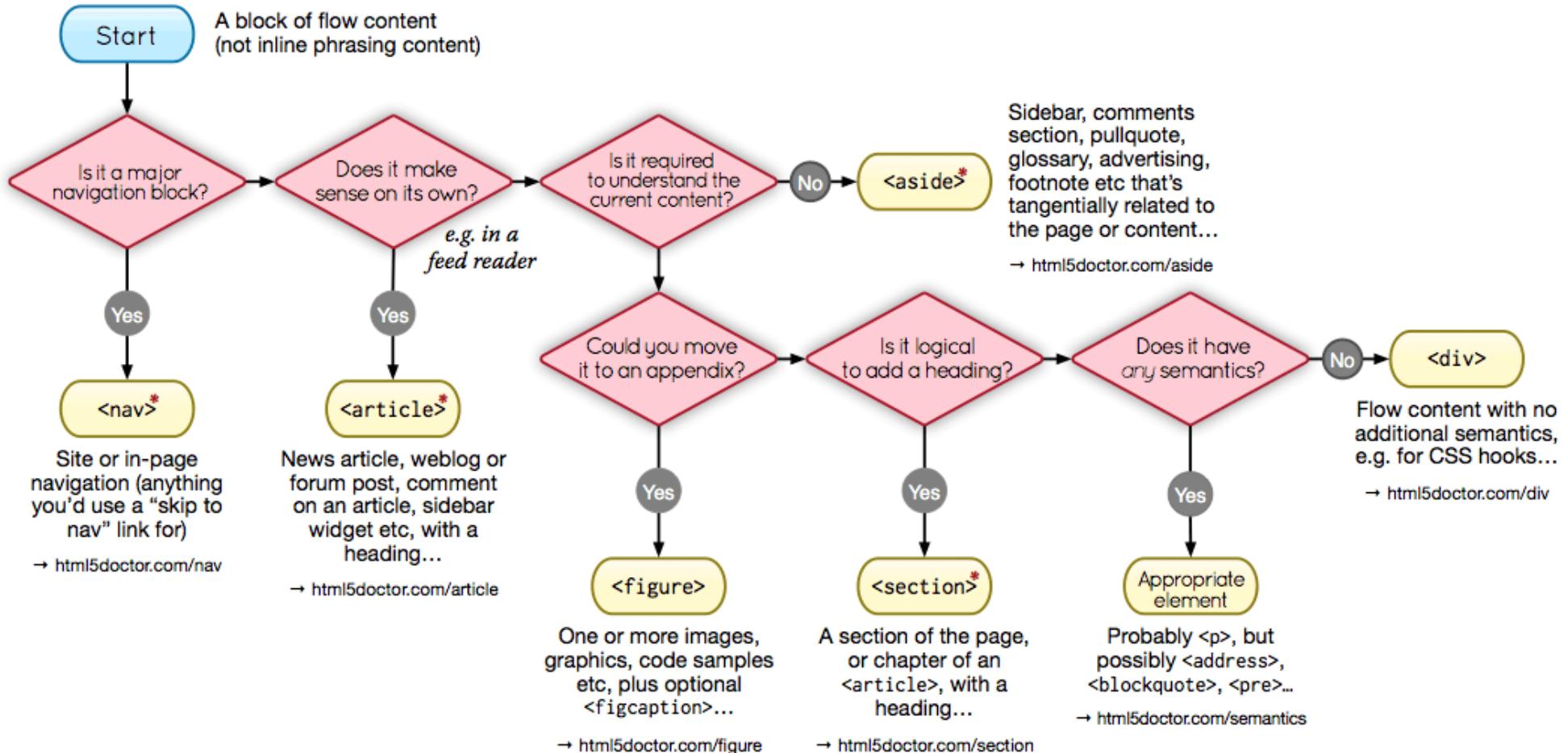


Doctor

HTML5 Element Flowchart

Sectioning content elements and friends

By @riddle & @boblet
www.html5doctor.com



* Sectioning content element

These four elements (and their headings) are used by HTML5's outlining algorithm to make the document's outline
→ html5doctor.com/outline

2011-07-22 v1.5

For more information:
www.html5doctor.com/semantics

Form Elements

Form Elements

עד כה רأינו אלמנטים ללא אינטראקציה עם המשתמש (המשתמש לא נדרש להזין כל ערך או לבחור משהו).

אלמנט טופס מתייחס לאלמנט של HTML שהמשתמש יזין בו ערך, למשל:

- תיבת טקסט
- רשימה נגללת ממנה יבחר ערך
- רשימת צ'ק בוקס לבחירה מרובה
- רשימה לבחירת ערך אחד (radio)
- כפתור לשילוח הטופס

input

הפקד המרכזי הוא פקד `input`.

הפקד מספק הרבה אפשרויות לתצוגה וזאת באמצעות תכונת `type`.

```
<input type="text" value="I'm a text field" />
<input type="password" placeholder="here comes the password" />
<input type="checkbox" id="questionOne" name="subscribe" value="yes" checked />
<input type="radio" id="soup" name="meal" value="soup" checked />
<input type="button" value="This is a button" />
```

input

הפקד המרכזי בפקדי `form`, הוא פקד `input`.

הפקד מספק הרבה אפשרות לציגו וזאת באמצעות תכונת `type`.

```
<input type="text" />  
<br />  
<input type="checkbox" />  
<br />  
<input type="radio" />  
<br />  
<input type="button" />
```



פקד input

תכונה חשובה נוספת לפקד, זו תכונת `value`.

במידה ומדובר בפקד טקסט/כפטור - התכונה מתיחסת לערך שהמשתמש יראה.

במידה ומדובר בפקד צ'ק בוקס/רדיו – התכונה מתיחסת לערך נסתר לשימוש (ע"י המפתח לדוגמא).

```
<span>text</span> <input type="text" value="some text" maxlength="15" disabled />
<br />
<input type="checkbox" value="volvo" /> <span>volvo</span>
<br />
<input type="checkbox" value="bmw" /> <span>bmw</span>
<br />
<input type="checkbox" value="subaru" /> <span>subaru</span>
<br />
<input type="checkbox" value="mazda" /> <span>mazda</span>
<br />
<input type="button" value="button" />
```

The image shows a web page with the following structure:

- A text input field with the placeholder "text" and the value "some text".
- A list of five checkboxes, each with a label:
 - checkbox labeled "volvo" (unchecked)
 - checkbox labeled "bmw" (unchecked)
 - checkbox labeled "subaru" (unchecked)
 - checkbox labeled "mazda" (unchecked)
 - checkbox labeled "button" (checked)

תיבת טקסט

עבור **טקסט** שמצווג כתיבת טקסט יש מספר אפשרויות נוספות, ביניהן:

- **текст** שיוצג בתוך תיבת הטקסט ויעלם עם תחילת הקלדה
- הגבלה על מספר התווים שיוקלדו בתיבה
- **текст** לקריאה בלבד

```
<input type="text" placeholder="first name" maxlength="10" />
```

בדוגמא:

תיבת הטקסט מצוינה למשתמש שיש להכניס שם פרטי.
המשתמש לא יוכל להכניס יותר מ-10 תווים.

label

- תמיד השתמש ב **label** לעטוף טקסט שבא לפני/אחרי הפקד באופן הבא:

Username *

```
<label for="username">Username<span class="required">*</span></label>  
  
<input  
    type="text"  
    id="username"  
    name="username"  
    required  
    minlength="3"  
    maxlength="20"  
    placeholder="Enter your username"  
>
```

label

```
<div>
  <label for="name">Full Name:</label>
  <input type="text" id="name" required>
</div>

<div>
  <label for="email">Email:</label>
  <input type="email" id="email" required>
</div>

<div>
  <label for="age">Age:</label>
  <input type="number" id="age" min="18">
</div>
```

צ'ק בוקס וcpfתורי רדיו

עבור `input` מסווג הנ"ל, תכונה נוספת היא היכולת לסמן מראש את הפקד כלוחץ, מתרצע באמצעות תכונת `checked`.

```
<input type="checkbox" value="car" checked/> volvo  
<br />  
<input type="radio" value="m" checked/> male  
<br />  
<input type="radio" value="f" checked/> female
```



volvo



male



female

שדף יעלה הפקדים יהיו מסומנים.
מה הבעה??

פקדי רדיו – קבוצה מחוברת

בד"כ בשימוש בפקדי רדיו, נרצה בחירה אחת מתוך קבוצה.
על מנת לעשות זאת צריך להגיד לאו פקדים מחוברים כקבוצה, כתוצאה לכך – רק אחד יכול להיות מסומן ברגע נתון.

מתרבץ ע"י שימוש בתכונה name:

```
<input type="radio" value="m" name="gender" checked/> male  male  
<br />  
<input type="radio" value="f" name="gender" checked/> female  female
```

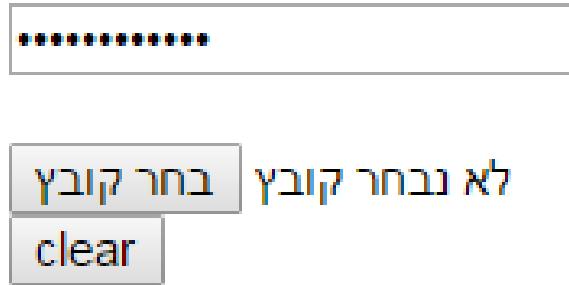
השם בתוכנת ה-name צריך להיות זהה (שייר לקבוצה)
שימו לב שלמרות שני הcptורים מוגדרים כמסומנים – רק אחד סומן עם טיעינת הדף.

פקדי כניסה נוספים

מעבר לסוגי ה-type שראינו יש עוד רשימה ארוכה (חלק נכיר בהמשך הקורס), מספר אפשרויות

נוספות הקיימות בשימוש ורחר.

```
<input type="password" />  
<br /><br />  
<input type="file" />  
<br />  
<input type="reset" value="clear"/>
```



– **type="file"** – יפתח תיבת לבחירת קובץ מהמחשב.
– **type="password"** – מתנהג כמו טקסט רגיל רק שהtekst אינו גלוי לעין.
– **type="reset"** – יחזיר את כל הערכים שהיו לפני שהמשתמש שינה אותם (עובד רק בשילוב אלמנט form)

תכונות נוספות לפקדי input

מעבר לסוגי התכונות שראינו, יש עוד רשימה ארוכה

```
<input type="text" disabled value="123"/> your serial number  
<br />  
<input type="text" autofocus /> insert your id
```

your serial number
insert your id

input – disabled – ניתן לעריכה

input – autofocus – יהיה הפקוד עם טעינת הדף

textarea

פקד נוסף הוא `textarea`, שמאפשר לתייבת הטקסט להיות מספר שורות:

```
<textarea cols="30" rows="2"></textarea>
```



```
hello world,  
hello world
```

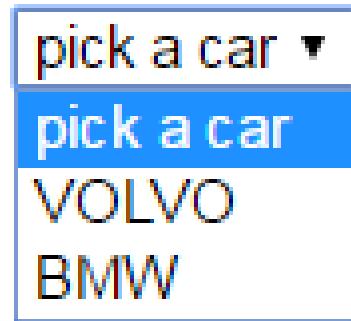
`rows` – מספר השורות שייוו מוצגות למשתמש, אם הטקסט חורג משתי שורות, תופיע אפשרות גלילה.

ניתן גם לשלוט בגודל הטקסט באמצעות תכונות css
(width, height)

select

רשימה נגללת לבחירת ערך מוצג ע"י **תג select**,
פריט ברשימה מוצג ע"י **תג option**.

```
<select>
  <option>pick a car</option>
  <option>VOLVO</option>
  <option>BMW</option>
</select>
```



```
<select>
  <option>pick a car</option>
  <option>VOLVO</option>
  <option selected>BMW</option>
</select>
```



בחירה מחדל ה-**option** העליון יוצג תחילה,
ניתן לחת את תכונת **selected**, והוא יוצג ראשון:

select

ניתן לשלב ערכים נסתרים לכל פריט ברשימה (בדומה לכפתורי רדיו):

```
<select>
  <option value="-1">pick a car</option>
  <option value="v">VOLVO</option>
  <option value="b">BMW</option>
</select>
```

```
<html>
<body>
    <input type="text" value="I'm a text field" />
    <br />
    <input type="password" placeholder="here comes the password" />
    <input type="checkbox" id="questionOne" name="subscribe" value="yes" checked />
    <h1>Choose all the vegetables you like to eat</h1>
    <ul>
        <li>
            <input type="checkbox" value="carrots" checked />
            <label for="carrots">Carrots</label>
        </li>
        <li>
            <input type="checkbox" value="peas" />
            <label for="peas">Peas</label>
        </li>
        <li>
            <input type="checkbox" value="cabbage" />
            <label for="cabbage">Cabbage</label>
        </li>
    </ul>
    <h1>What is your favorite meal?</h1>
    <ul>
        <li>
            <input type="radio" id="soup" name="meal" value="soup" checked />
            <label for="soup">Soup</label>
        </li>
        <li>
            <input type="radio" id="curry" name="meal" value="curry" />
            <label for="curry">Curry</label>
        </li>
        <li>
            <input type="radio" id="pizza" name="meal" value="pizza" />
            <label for="pizza">Pizza</label>
        </li>
    </ul>
    <br />
    <input type="button" value="This is a button" />
</body>
</html>
```

תגיל ביתה

Registration Form

כתוב טופו הרשמה המכיל:

Username • Enter your username

Password • Enter your password

Email • Enter your email

Birthday • dd / mm / yyyy

Button for upload Profile picture •

checkbox of: “Subscribe to our newsletter” •

Register button •

Profile Picture

Choose File No file chosen

Subscribe to our newsletter

Register

Flexbox

קורס : צד ל Koh

מרצה: יעל סלע זעירא

הקדמה Flexbox

- **Flexbox** נותן לנו שימוש נוח לייצרת layout (לבטח כשמדבר ב layout חד מימדי של שורות או עמודות).
- לעיתים ברצוננו להמנע כמה שניתן מתלות בספריות ונכסים חיצוניים (כדוגמת bootstrap) ואני מעוניינים להשתמש רק ביכולות של flexbox – CSS הוא המענה עבורנו.

מודל ה flexbox

קונטינר מסוג flex
ו item שהוא ילד ישיר של קונטינר

Flex items .2

• שני סוגי של "קופסאות" :

Flex container .1 •



“FLEX CONTAINER”



“FLEX ITEMS”

display: flex;

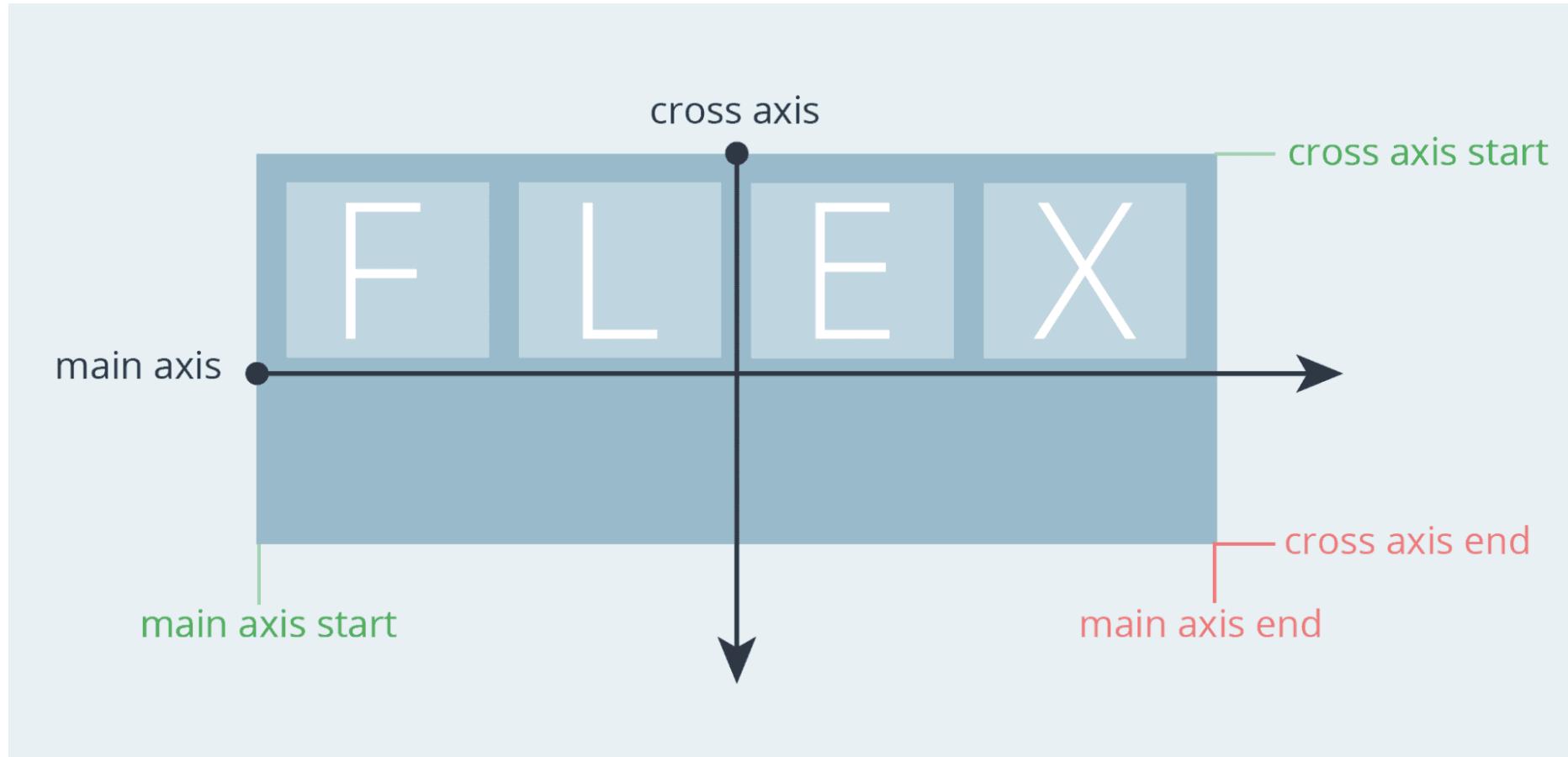
```
<head>
<style>
.flex-container {
  display: flex;
}
</style>
</head>
<body>
<div class="flex-container">
<div>Flex Item</div>
<div>Flex Item</div>
<div>Flex Item</div>
</div>
</body>
```

- על ידי הוספת התכונה `display: flex;` לאלמנט מסוים.

- הוא יאפשר לך **Flex container** **Flex items** וילדיו ל

Flex Item Flex Item Flex Item

Main-Axis and Cross-Axis



מיקום ויישור אלמנטים בציר ה- Y האנכי

align-items

מאפשרת לנו "לשחק" עם האלמנטים אונכית בציר ה y
`align-items: center;`

* אם יש כמה שורות נשתמש ב `align-content`

```
<head>
<style>
.flex-container {
  display: flex;
  border: 1px solid #1a232d;
  height: 200px;
  align-items:center
}
</style>
</head>
<body>
<div class="flex-container">
<div>Flex Item</div>
<div>Flex Item</div>
<div>Flex Item</div>
</div>
</body>
```

Flex ItemFlex ItemFlex Item

align-items:flex-end

Flex ItemFlex ItemFlex Item

מיקום ויישור אלמנטים בציר ה- X האנכי

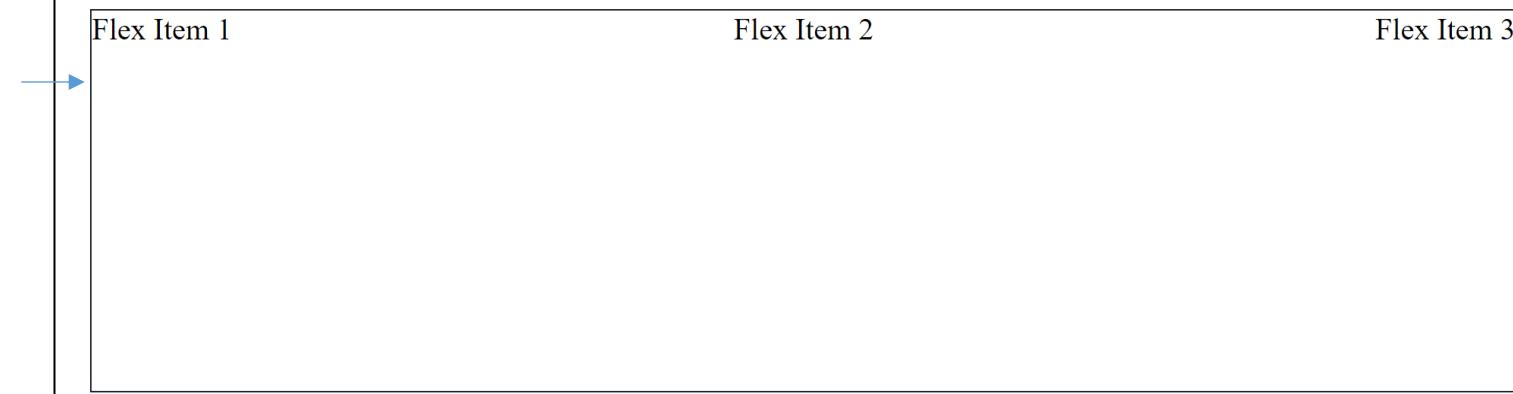
justify-content

מאפשרת לנו "לשחק" עם האלמנטים אופקית בציר ה x

`justify-content: flex-start;`

* ב *Flex Container* אלמנטים מסתדרים בשורה ובתחלת ציר ה- X מכיוון וערך ברירת המחדל של התכונה הוא `flex-start`

```
<head>
<style>
.flex-container {
  display: flex;
  border: 1px solid #1a232d;
  height: 200px;
  justify-content: space-between;
}
</style>
</head>
<body>
<div class="flex-container">
<div>Flex Item 1</div>
<div>Flex Item 2</div>
<div>Flex Item 3</div>
</div>
</body>
```



flex-wrap

- **flex-wrap** – כבירית מחדל אלמנטים לא יגלו לשורה חדשה אם אין להם מקום בקונטינר המכיל אותם.
- אם נרצה שהאלמנטים יגלו לשורה חדשה בקונטינר, יש לשים **flex-wrap: wrap;**

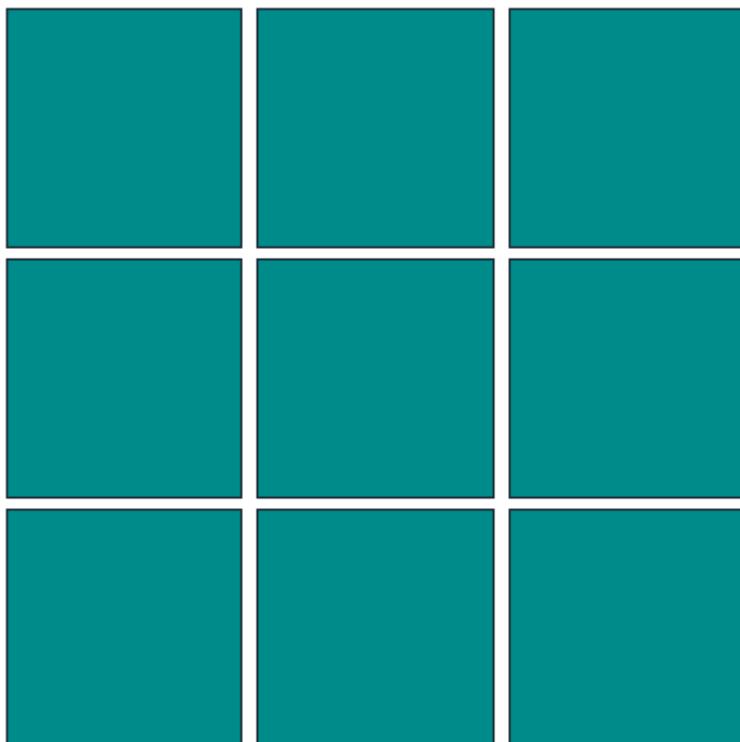


```
<head>
  <style>
    .flex-container {
      display: flex;
      border: 1px solid #1a232d;
      height: 200px;
      flex-wrap: wrap;
    }
    .flex-container > div {
      width: 400px;
      border: 1px dashed black;
    }
    .flex-container > div:last-child {
      width: 900px;
    }
    .flex-container > div:nth-child(odd) {
      background-color: red;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div>Flex Item 1</div>
    <div>Flex Item 2</div>
    <div>Flex Item 3</div>
  </div>
</body>
```



תרגיל ביתה 1

צייר את הגריד הבא על ידי שימוש ב `display: flex`

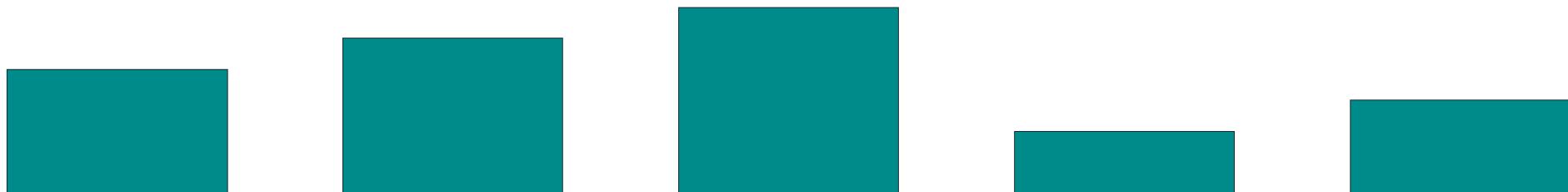


פתרונות תרגילים 1

```
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
.container {
display: flex;
width: 300px;
height: 300px;
flex-wrap: wrap;
}
.container > div {
background-color: darkcyan;
width: 98px;
height: 98px;
margin: 1px;
}
  </style>
</head>
<body>
  <div class="container">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
```

תרגיל ביתה 2

צייר את הסכמה הבאה על ידי שימוש ב `display: flex`



```
.container {  
    display: flex;  
    height: 300px;  
    justify-content: space-between;  
    align-items: flex-end;  
}  
  
.container > div {  
    background-color: darkcyan;  
    border: 1px solid #1a232d;  
    width: 14%;  
}  
  
.container > div:nth-child(1) {  
    width: 14%;  
}  
  
.container > div:nth-child(2) {  
    height: 40%;  
}  
  
.container > div:nth-child(3) {  
    height: 50%;  
}  
  
.container > div:nth-child(4) {  
    height: 60%;  
}  
  
.container > div:nth-child(5) {  
    height: 20%;  
}  
  
.container > div:nth-child(6) {  
    height: 30%;  
}
```

```
<div class="container">  
    <div>40%</div>  
    <div>50%</div>  
    <div>60%</div>  
    <div>20%</div>  
    <div>30%</div>  
</div>
```

תכונות של אלמנט Flex Item

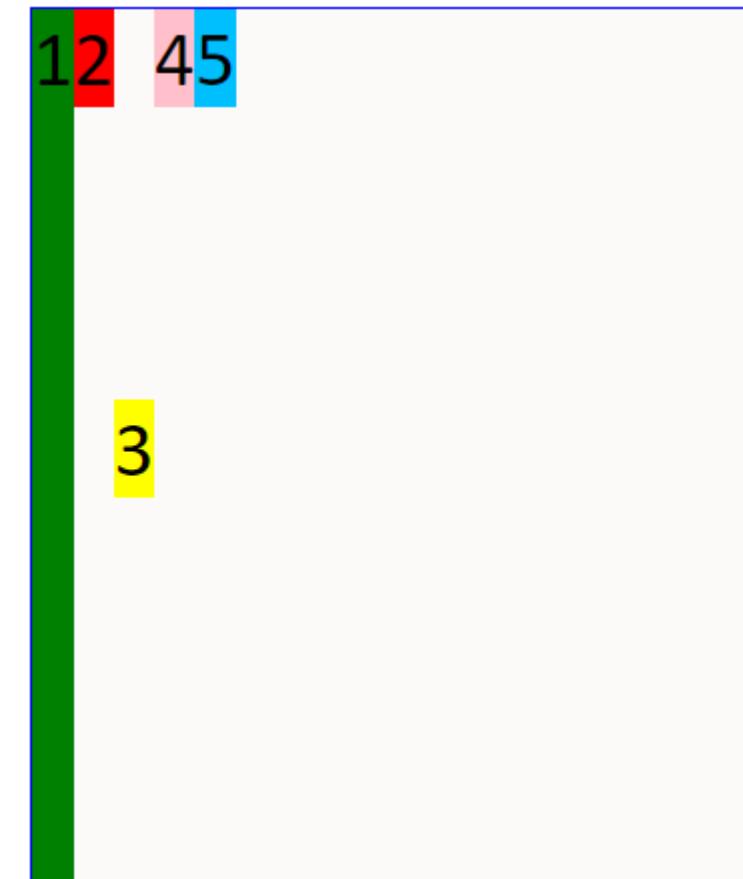
align-self

מאפשרת לשנות מיקום של אלמנט ספציפי בתחום הקונטינר.

מאפשרת לנו לשבור את הגדרת ברירת המחדל הדיפולטיבית של align-items הקיימת לקונטינר עבור אלמנט ספציפי בקונטינר זה.

[flexItemExample.html](#)

```
<head>
  <style>
body{
    font-family: Calibri;
    font-size: xx-large;
}
.main-container {
width: 100%;
height: 350px;
border: 1px solid blue;
background: #fbfaf8;
}
.flex-wrapper {
display: flex;
align-items: flex-start;
}
.flex-wrapper > div:nth-child(1) {
align-self: stretch;
}
.flex-wrapper > div:nth-child(3) {
align-self: center;
}
</style>
</head>
<body>
  <div class="main-container flex-wrapper">
    <div style="background-color: green;">1</div>
    <div style="background-color: red;">2</div>
    <div style="background-color: yellow;">3</div>
    <div style="background-color: pink;">4</div>
    <div style="background-color: deepskyblue;">5</div>
  </div>
</body>
```



תכונות של אלמנט Flex Item

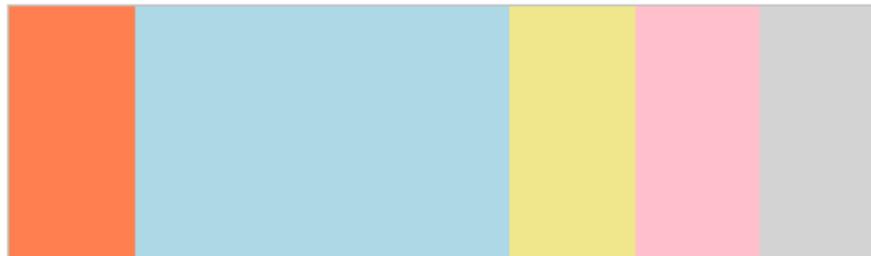
flex-grow

באמצעות התכונה `flex-grow` ניתן לשלוט על כמות השטח שאלמנט ספציפי יתפס ביחס לאחרים.

* מכיוון והערך הוא 'יחס', אם תגדירו את התכונה `200`: `flex-grow: 200` לכל האלמנטים יהיה זה בדיקן
כאילו קבעתם את הערך של `colm` ל-`1`. או אם תנתנו ערך של `flex-grow` רק לאלמנט אחד הוא
יתפס את השטח שהגדרתם לו אל מול היחס הדיפולטיבי שניתן לשאר האלמנטים.

```
<html>
<head>
<style>
#main {
width: 350px;
height: 100px;
border: 1px solid #c3c3c3;
display: flex;
}
#main div:nth-of-type(1) {
flex-grow: 1;
}
#main div:nth-of-type(2) {
flex-grow: 3;
}
#main div:nth-of-type(3) {
flex-grow: 1;
}
#main div:nth-of-type(4) {
flex-grow: 1;
}
#main div:nth-of-type(5) {
flex-grow: 1;
}
</style>
</head>
<body>
<h1>The flex-grow Property</h1>
<div id="main">
<div style="background-color:coral;"></div>
<div style="background-color:lightblue;"></div>
<div style="background-color:khaki;"></div>
<div style="background-color:pink;"></div>
<div style="background-color:lightgrey;"></div>
</div>
<p><b>Note:</b> Internet Explorer 10 and earlier versions do not support the flex-grow property.</p>
</body>
</html>
```

The flex-grow Property



Note: Internet Explorer 10 and earlier versions do not support the flex-grow property.

תרגיל ביתה

צור את התפריט הבא על ידי שימוש ב flex, בנוסף בעבר כבר ישנה הLINK בתפריט לאדם

Home

About

Blog

Careers

Contact Us

```
<div class="container">
    <div><a href="/">Home</a></div>
    <div><a href="/about">About</a></div>
    <div><a href="/blog">Blog</a></div>
    <div><a href="/jobs">Careers</a></div>
    <div><a href="/contact">Contact Us</a></div>
</div>
```

פתרונות תרגיל ביתה

```
html {  
font-family: Calibri;  
}  
body {  
margin: 0;  
}  
.container {  
display: flex;  
background-color: black;  
padding: 10px 0;  
}  
.container > div {  
flex-grow: 1;  
text-align: center;  
background: #ccc;  
color: white;  
margin: 0 5px;  
padding: 5px 15px 5px 15px;  
border: 1px solid white;  
}  
.container > div > a {  
text-decoration: none;  
}  
.container > div:hover {  
border: 1px solid red;  
}  
.container > div > a:hover {  
color: red;  
text-decoration: underline;  
}
```

CSS grid

קורס : צד ל Koh

מרצה: יעל סלע זעירא

CSS Grid

- **CSS Grid** הוא מודול המאפשר ליצור עימוד layout המבוסס רשת (grid) תוך שימוש בעמודות ושורות.

- ראיינו עימוד ב HTML בעזרת floats,divs, tables ועוד תכונות CSS כאלה ואחרות.

- **CSS Grid** מאפשר ליצור מבנה בצורה גריד אשר מתואר ברמת ה **HTML**

* *Flexbox* נועדה ומתאימה יותר עבור עימוד חד מימדי, לעומת שורה או עמודה בלבד, *Grid* לעומת שורה ומתקווה דו מימדי המורכב מעמודות ושורות יחד.

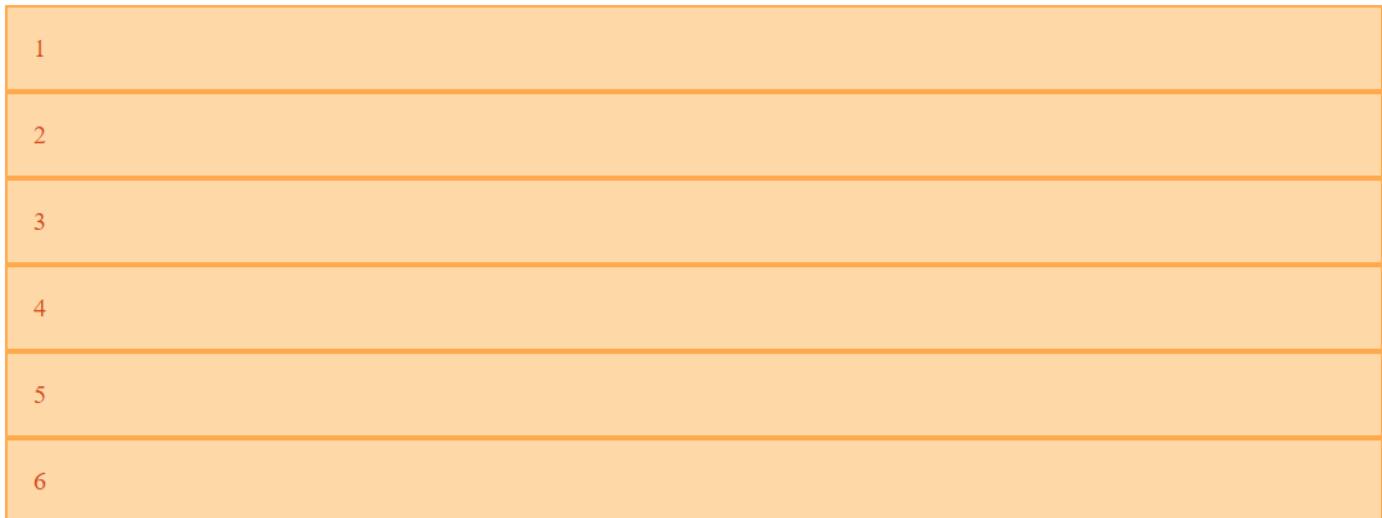
Grid Container

- **Grid container** הוא אלמנט אב המכיל את מתווה הגריד (layout).
- הוא מוגדר באמצעות התכונה **display: grid** על אותו קונטינר.
- **Grid items** - הילדים היוצרים של אותו Grid Container
- **Grid cell** - תא שМОוגדר על ידי הצעטלבות של שורה ועמודה בגריד, כך שאם יש לכם גRID של 3 שורות ו 3 עמודות, יהיו לכם תשעת Cells.

נראה כרגיל – אך בעת כל שורה היא grid item

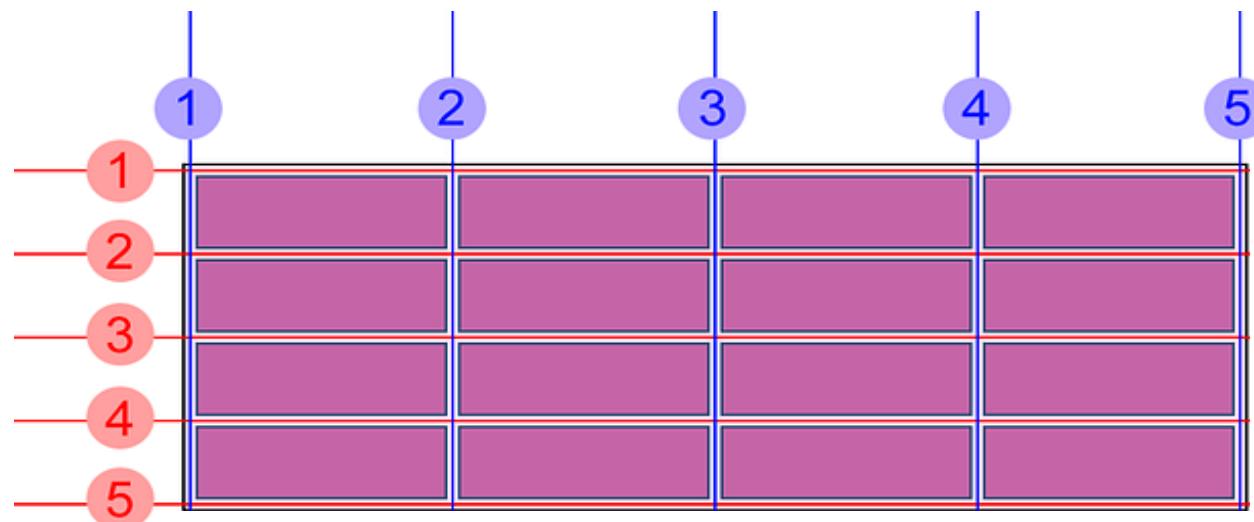
```
<div class="wrapper">  
    <div>One</div>  
    <div>Two</div>  
    <div>Three</div>  
    <div>Four</div>  
    <div>Five</div>  
</div>
```

```
.wrapper {  
    display: grid;  
}
```



Grid Line

קווי grid-line הוא קו שמחלק את הgrid לשורות ועמודות. קווי grid-line ממוקדים מ-1 עד x כמספר השורות או העמודות בgrid עצמו. נשתמש בהגדלה של **grid-template-rows** | **grid-template-columns** כדי להגיד את החלוקת grid.



חלוקת ה Grid לעמודות: grid-template-columns

```
.wrapper {  
    display: grid;  
    grid-template-columns: 200px 200px 200px;  
}
```

דוגמא

דוגמא

```
.wrapper {  
    display: grid;  
    grid-template-columns: 1fr 1fr 1fr;  
}
```

```
.wrapper {  
    display: grid;  
    grid-template-columns: 2fr 1fr 1fr;  
}
```

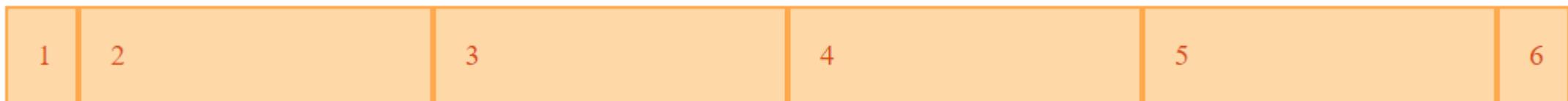
דוגמא

דוגמא

```
.wrapper {  
    display: grid;  
    grid-template-columns: 500px 1fr 2fr;  
}
```

```
.wrapper {  
    display: grid;  
    grid-template-columns: repeat(4, 1fr);  
}
```

```
.wrapper {  
    display: grid;  
    grid-template-columns: 40px repeat(4, 1fr) 40px;  
}
```



חלוקת ה Grid לשורות: grid-template-rows

```
.container {  
display: grid;  
grid-template-columns: repeat(3, 1fr);  
grid-template-rows: 60px 60px 60px;  
}  
.container > div {  
border: 1px solid black;  
}
```

```
<div class="container">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
  <div>4</div>  
  <div>5</div>  
  <div>6</div>  
  <div>7</div>  
  <div>8</div>  
  <div>9</div>  
  <div>10</div>  
  <div>11</div>  
  <div>12</div>  
  <div>13</div>  
</div>
```

1	2	3
4	5	6
7	8	9
10	11	12
13		

חלוקת ה Grid לשורות: grid-auto-rows

```
.container {  
display: grid;  
grid-template-columns: repeat(3, 1fr);  
grid-auto-rows: 60px;  
}  
  
.container > div {  
border: 1px solid black;  
}
```

אם אנו לא יודעים כמה שורות יהיו ונרצה להגדיר גובה לכל השורות שייצרו נוכל להשתמש ב:

grid-auto-rows

1	2	3
4	5	6
7		

חלוקת ה Grid לשורות: grid-auto-rows

```
.container {  
display: grid;  
grid-template-columns: repeat(3, 1fr);  
grid-auto-rows: minmax(100px, auto);  
}  
  
.container > div {  
border: 1px solid black;  
}
```

content – הגובה יקבע לפי ה content
אבל יהיה לפחות 100px

אם אנו לא יודעים כמה שורות יהיו ונרצה להגדיר
גובה לכל השורות שייצרו נוכל להשתמש ב: grid-auto-rows

1	2	3
4	5	6
7		

אפשר גם רק grid-auto-rows: auto; אז הגובה יהיה לפי ה content

```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 5px;  
  grid-auto-rows: minmax(100px, auto);  
}
```

```
<div class="wrapper">  
  <div>One</div>  
  <div>  
    Two  
    <p>I have some more content in.</p>  
    <p>This makes me taller than 100 pixels.</p>  
  </div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

One

Two

I have some more content in.

This makes me taller than 100 pixels.

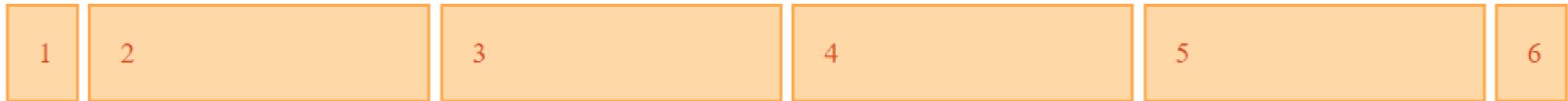
Three

Four

Five

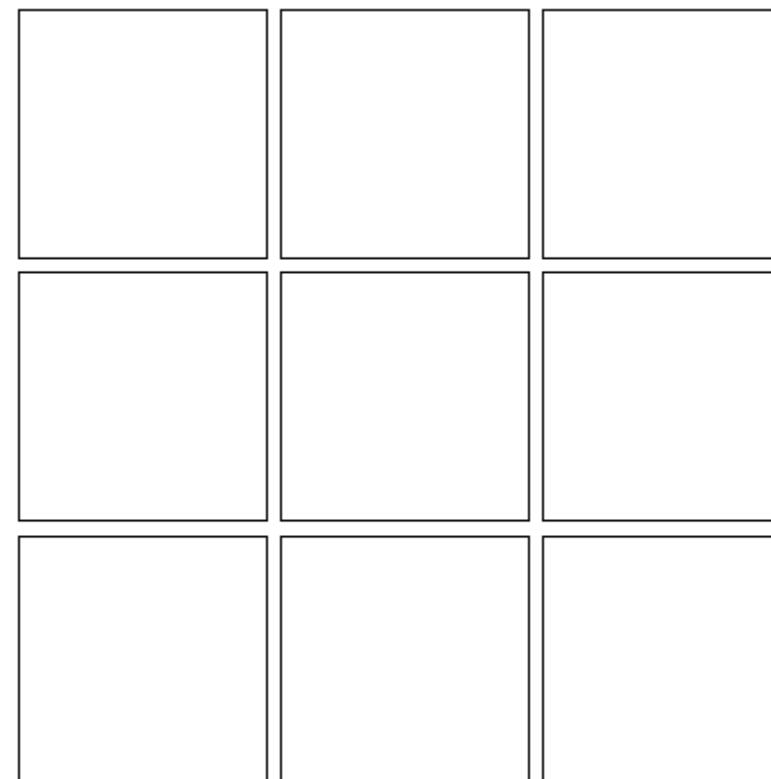
gap

```
.wrapper {  
    display: grid;  
    grid-template-columns: 40px repeat(4, 1fr) 40px;  
    gap: 5px;  
}
```



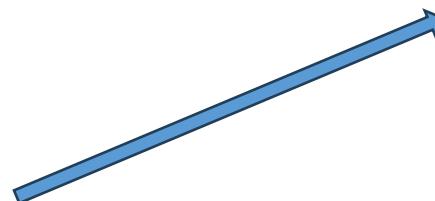
תרגיל חימום

- צייר את הגריד הבא על ידי שימוש ב CSS Grid



Alignment

ניתן מקום את ה Grid Items עצם באמצעות ששת התכונות הבאות:



תכונות אלה משפיעות על ה Grid Container
ארחיקם מתייחסות ל Grid Items כמו גם על Item ספציפי.

- justify-items •
- align-items •
- justify-content •
- align-content •
- justify-self •
- align-self •

```
<div class="my-container">  
    <div class="item">1</div>  
    <div class="item">2</div>  
    <div class="item">3</div>  
    <div class="item">4</div>  
    <div class="item">5</div>  
    <div class="item">6</div>  
</div>
```

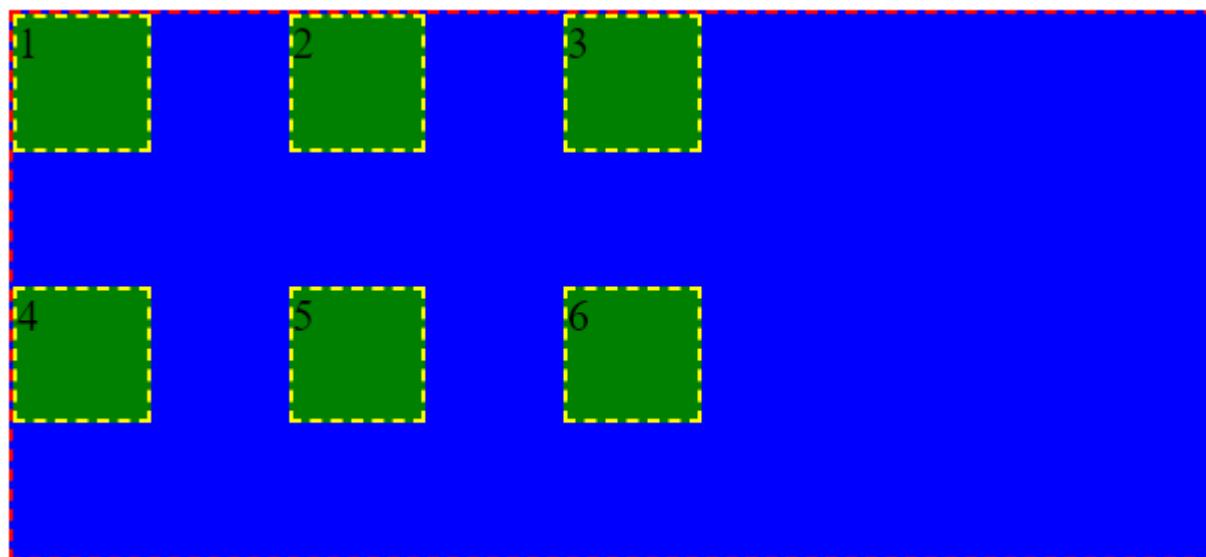
נסתכל על הקונטינר הבא
המכיל 6 אלמנטים בתוכו

```

<style>
.my-container {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  box-sizing: border-box;
  height: 200px;
  background: blue;
  border: 2px dashed red;
}
.my-container div {
  box-sizing: border-box;
  width: 50px;
  height: 50px;
  background: green;
  border: 2px dashed yellow;
}
</style>

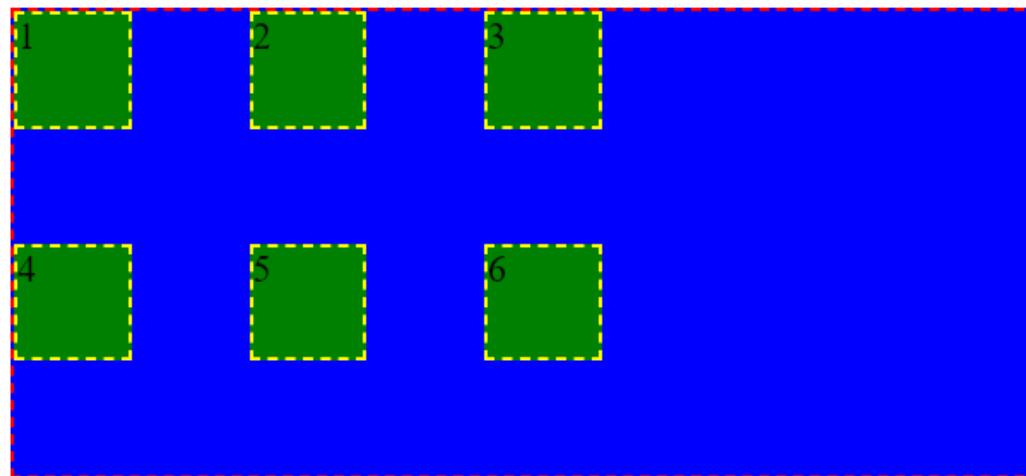
```

שים לב שה Grid Items במקורה זה אינם
מלאים ה Cell (שהוגדר ל 100px)
אל תופס רק ax50 כפי שהגדכנו

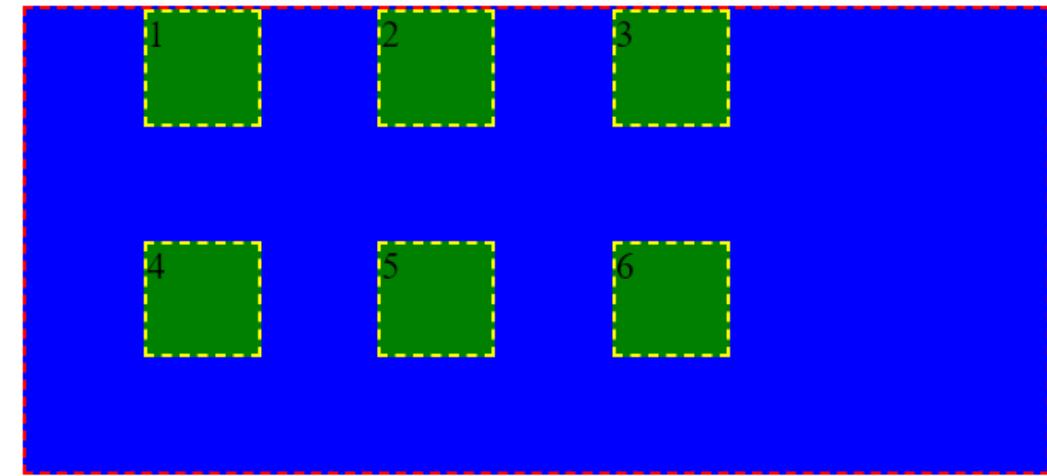


justify-items

משמש לישור ה x (בתוכה העמודה):
start, end, center , stretch

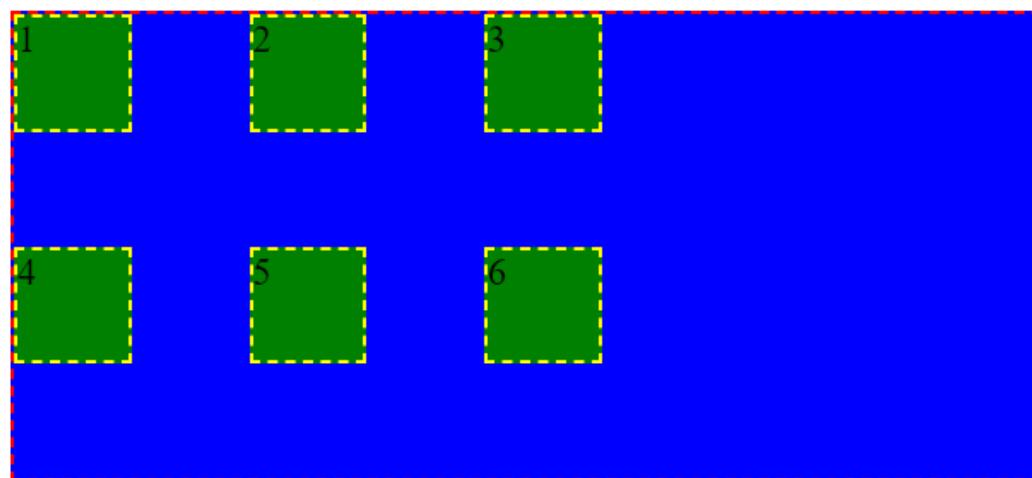


`justify-items: end;`

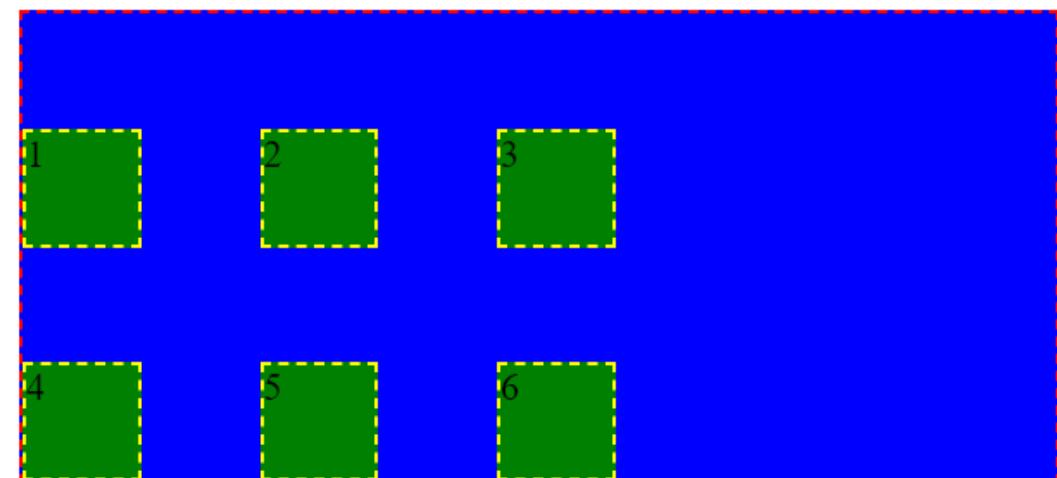


align-items

משמש לישור ה Grid Items על ציר ה y (בתווך השורה):
start, end, center , stretch

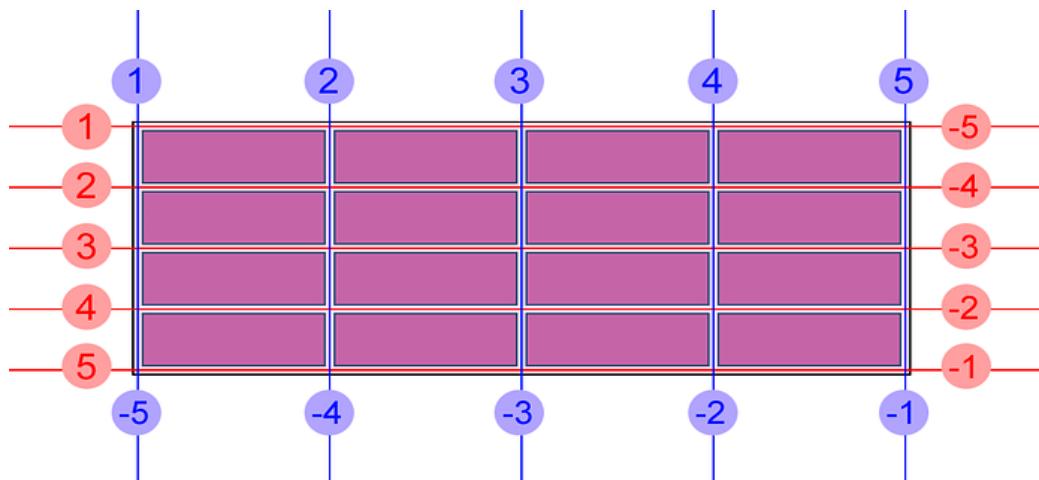


`align-items: end;`



grid-column-start/end

על מנת להגדיר התחלת וסיום של תא מסוים ברמת העמודה, נוסף :



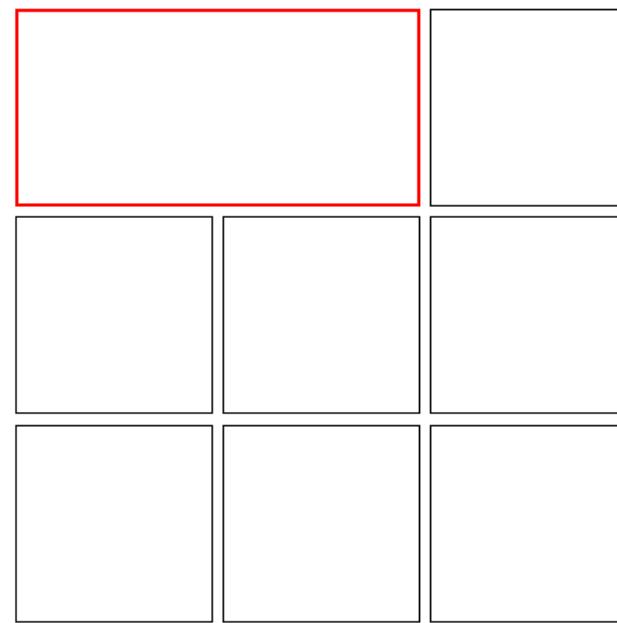
grid-column-start

grid-column-end

```
.container {  
width: 300px;  
margin: 100px auto;  
display: grid;  
grid-template-columns: 100px 100px 100px;  
grid-template-rows: 100px 100px 100px;  
gap: 5px;  
}
```

```
.container > div {  
border: 1px solid #000;  
}
```

```
.container > div:first-child {  
border: 2px solid red;  
grid-column-start: 1;  
grid-column-end: 3;  
}
```

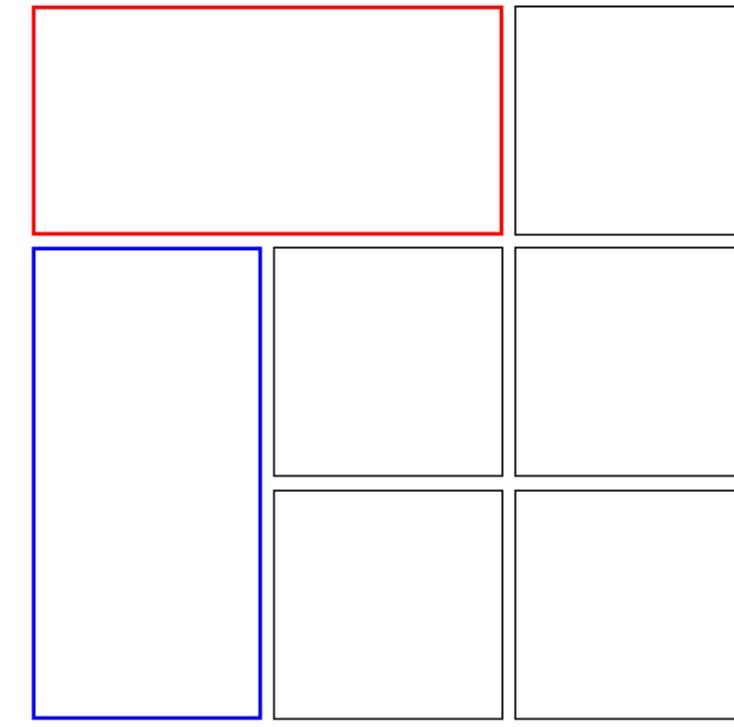


```
.container {  
width: 300px;  
margin: 100px auto;  
display: grid;  
grid-template-columns: 100px 100px 100px;  
grid-template-rows: 100px 100px 100px;  
gap: 5px;  
}
```

```
.container > div {  
border: 1px solid #000;  
}
```

```
.container > div:first-child {  
border: 2px solid red;  
grid-column-start: 1;  
grid-column-end: 3;  
}
```

```
.container > div:nth-child(3) {  
border: 2px solid blue;  
grid-row-start: 2;  
grid-row-end: 4;  
}
```



```
.wrapper
{
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}

.wrapper > div {
  border: 2px solid #ffa94d;
  background-color: #ffd8a8;
  padding: 1em;
  color: #d9480f;
}
```



```
= .box1 {
  grid-column: 1 / 4;
  grid-row: 1 / 3;
}
```

```
= .box2 {
  grid-row: 3 / 5;
}
```

תרגיל ביתה

צור 6 divs וסדר אותם בסדר הבא:



```
.wrapper {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-auto-rows: minmax(100px, auto);  
}
```

grid-area

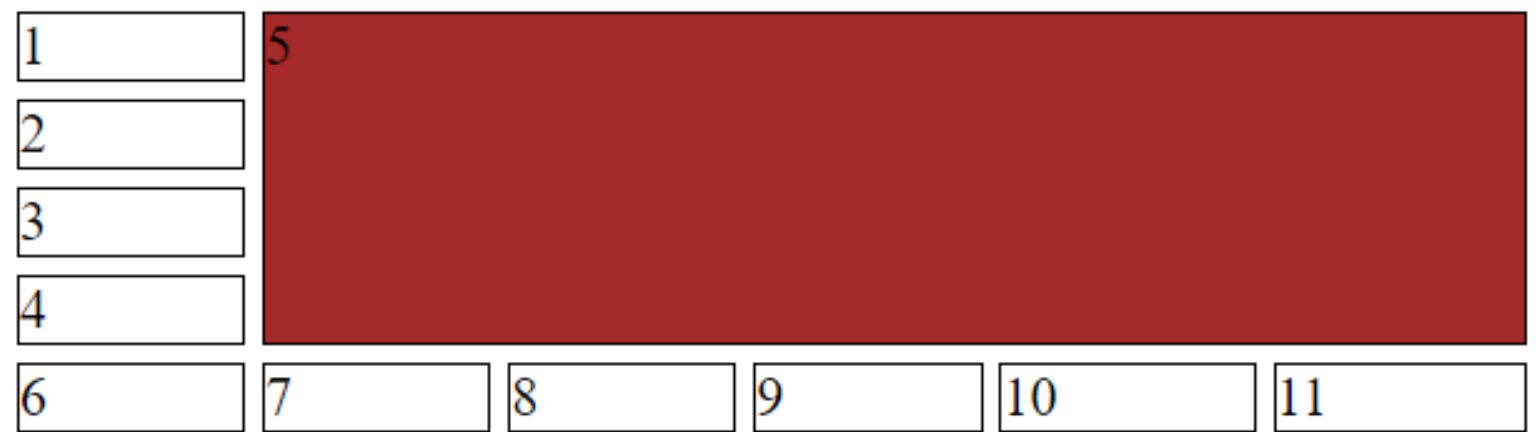
התכונה `grid-area` יכולה לשמש כתחליף מקוצר לתכונות הבאות:

- `grid-row-start`
- `grid-column-start`
- `grid-row-end`
- `grid-column-end`

`grid-area: <row-start> / <column-start> / <row-end> / <column-end>`

```
<div class="wrapper">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
  <div>4</div>  
  <div>5</div>  
  <div>6</div>  
  <div>7</div>  
  <div>8</div>  
  <div>9</div>  
  <div>10</div>  
  <div>11</div>  
</div>
```

```
.wrapper {  
  display: grid;  
}  
  
.wrapper > div:nth-child(5) {  
  grid-area: 1 / 2 / 5 / 7;  
}  
  
grid-area: <row-start> / <column-start> / <row-end> / <column-end>
```



grid-area

```
.wrapper {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: 100px 100px;  
    grid-template-areas:  
        "a a b"  
        "a a b";  
}  
  
.item1 {  
    grid-area: a;  
}  
  
.item2 {  
    grid-area: b;  
}  
.wrapper div{  
    border: dotted red;  
    background-color: antiquewhite;  
}
```

```
<body>  
<div class="wrapper">  
    <div class="item1">Item</div>  
    <div class="item2">Item</div>  
</div>  
</body>
```



יש להגדיר שמות (כראות עיניכם) לאלמנטים בגריד (Grid Items) באמצעות התכונה: `grid-area`

בדוגמא מטה השמות שניתנו הם שמות האלמנטים

```
header {  
    grid-area: header;  
}  
nav {  
    grid-area: nav;  
}  
section {  
    grid-area: section;  
}  
aside {  
    grid-area: aside;  
}  
footer {  
    grid-area: footer;  
}
```

```
<div class="wrapper">  
    <header>header</header>  
    <nav>nav</nav>  
    <section>section</section>  
    <aside>aside</aside>  
    <footer>footer</footer>  
</div>
```

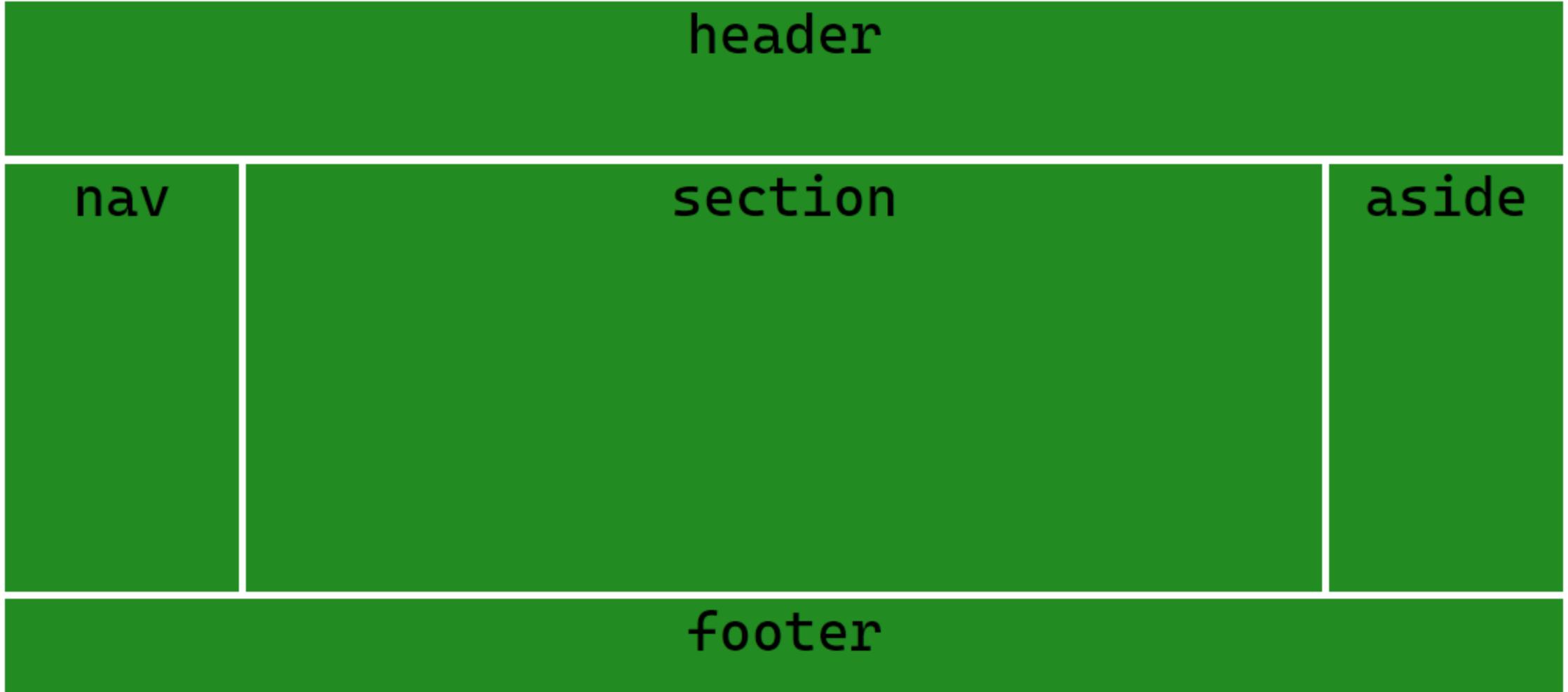
יש להגדיר על הקונטינר של הגRID בצורה בהם

האזורים להם סיפקתם שמות יוצגו :

```
header {  
    grid-area: header;  
}  
nav {  
    grid-area: nav;  
}  
section {  
    grid-area: section;  
}  
aside {  
    grid-area: aside;  
}  
footer {  
    grid-area: footer;  
}
```

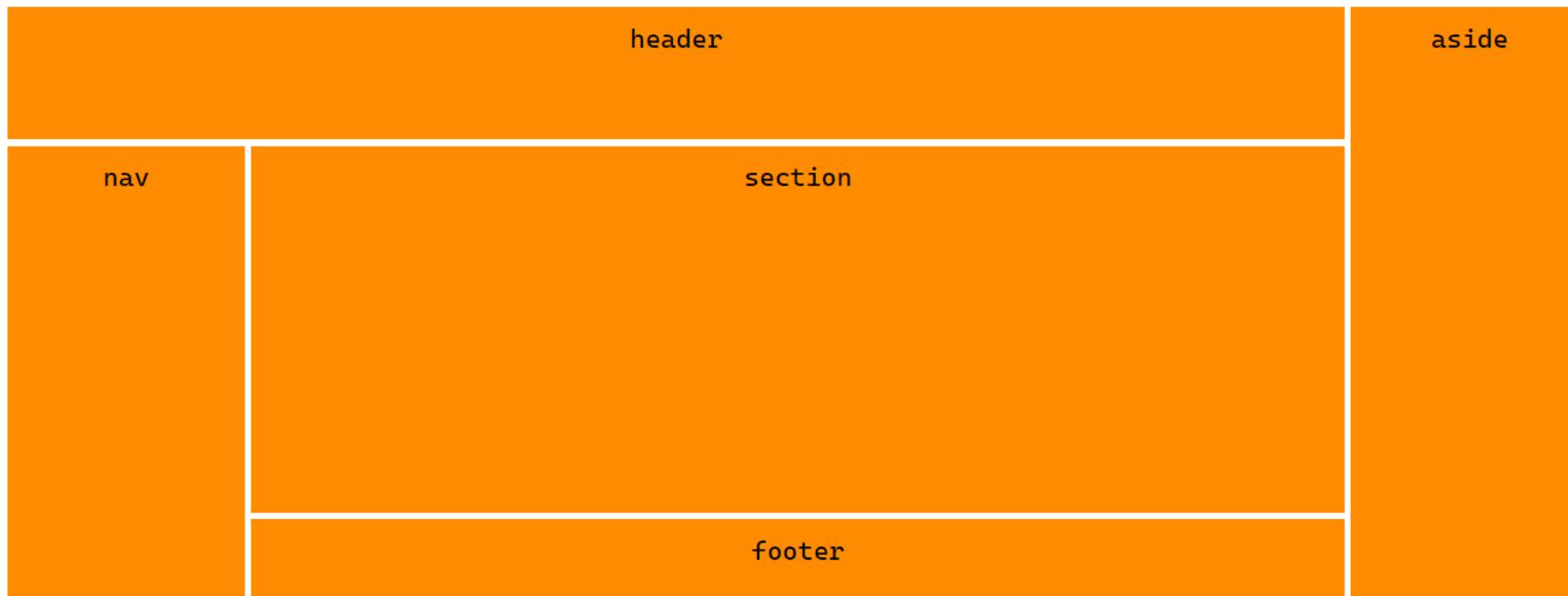
```
.wrapper {  
    display: grid;  
    grid-template-rows: 80px 1fr 50px;  
    grid-template-columns: 15% 1fr 15%;  
    grid-template-areas:  
        "header header header"  
        "nav section aside"  
        "footer footer footer";  
    grid-gap: 4px;  
    height: 360px;  
}
```

גושים מייצג שורה
מילה מייצגת עמודה



תרגיל

- עבוד על exGridTemplateAreasSkelton
- צרו את העימוד הבא לדף:



תגית ה `viewport`

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

width=device-width קובע שרוחב ה `viewport` יתאים לרוחב המסך הפיזי של המחשב
initial-scale=1 מגדיר את רמת הזום ההתחלתית (1=לא זום)

למה צריך את תגית ה `viewport` ?

ללא התגית, דפדףים במכשירים ניידים מתייחסים לעמוד כאילו הוא מיועד לרוחב מסך סטנדרטי של ~960px ("מסך דסקטופ")
ואז תוכן יכול להיראות עזיר על מסכים קטנים והמשתמש יctrar לבעז זום-אין וזום-אאוט כדי לקרוא את התוכן או לטעול אותו.

Media Queries

CSS Media queries מאפשרים לבלוק CSS רק במידה ומתקיימים תנאים מסוימים

```
@media media-type and (media-feature-rule) {  
    /* CSS rules go here */  
}
```

- `all`
- `print`
- `screen`

```
@media screen and (max-width:600px) {  
    body {  
        background: yellow;  
    }  
}
```

אם רוחב המסך צר מ-600 פיקסלים נוסיף בלוק של CSS שישנה את צבע הרקע לצהוב

```
@media print {  
    body {  
        font-size: 12pt;  
    }  
}
```

set the body to 12pt if the page is printed. It will not apply when the page is loaded in a browser.

```
@media screen and (max-width: 600px) {  
    body {  
        color: blue;  
    }  
}
```

max-width make the color blue if the viewport is 600 pixels or narrower

```
@media screen (min-width: 30em) and (max-width: 50em) {  
    body {  
        background-color: lightblue;  
        font-size: 1.2rem;  
    }  
}
```

העיצוב שבתור ה Media Query יחול רק אם רוחב המסך הוא לפחות 30em ועד 50em כולל.
em - יחידת מדידה יחסית לפונט הבסיסי של המסמן

min-width = 30 * 16 = 480px // 30em converted to pixels (assuming 1em = 16px)
max-width = 50 * 16 = 800px // 50em converted to pixels

עיצוב רספונסיבי עם CSS Grid

- CSS Grid מאפשר לנו ליצור עמודים ללא גודלים קבועים אלא על ידי תחימת אלמנטים בגבולות מסוימים

טסך רחוב

first

second

third

סמארטפונ

first

second

third

```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-gap: 15px;  
}  
  
.container > li {  
    list-style-type: none;  
    border: 1px solid black;  
}
```

first

second

third

עד גודל 960px תהיה עמודה אחת, מעל גודל זה יהיו 3 עמודות

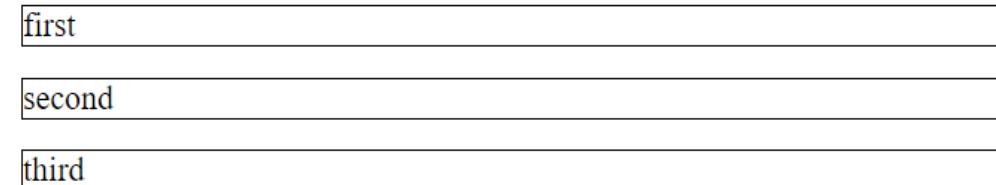
```
@media screen and (max-width: 960px) {  
    .container {  
        grid-template-columns: 1fr;  
    }  
}
```

```
<body>  
    <ul class="container">  
        <li><div>first</div></li>  
        <li><div>second</div></li>  
        <li><div>third</div></li>  
    </ul>  
</body>
```



```
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title></title>           first          second          third
<style>
  .container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 15px;
  }
  .container > li {
    list-style-type: none;
    border: 1px solid black;
  }
  @media screen and (max-width: 960px) {
    .container {
      grid-template-columns: 1fr;
    }
  }
</style>
</head>
<body>
  <ul class="container">
    <li><div>first</div></li>
    <li><div>second</div></li>
    <li><div>third</div></li>
  </ul>
</body>
</html>
```



תרגיל ביתה

השתמש בקובץ השלד exResponsiveCssGridSkelton.html

צור רספונסיביות למסכים בגודל:

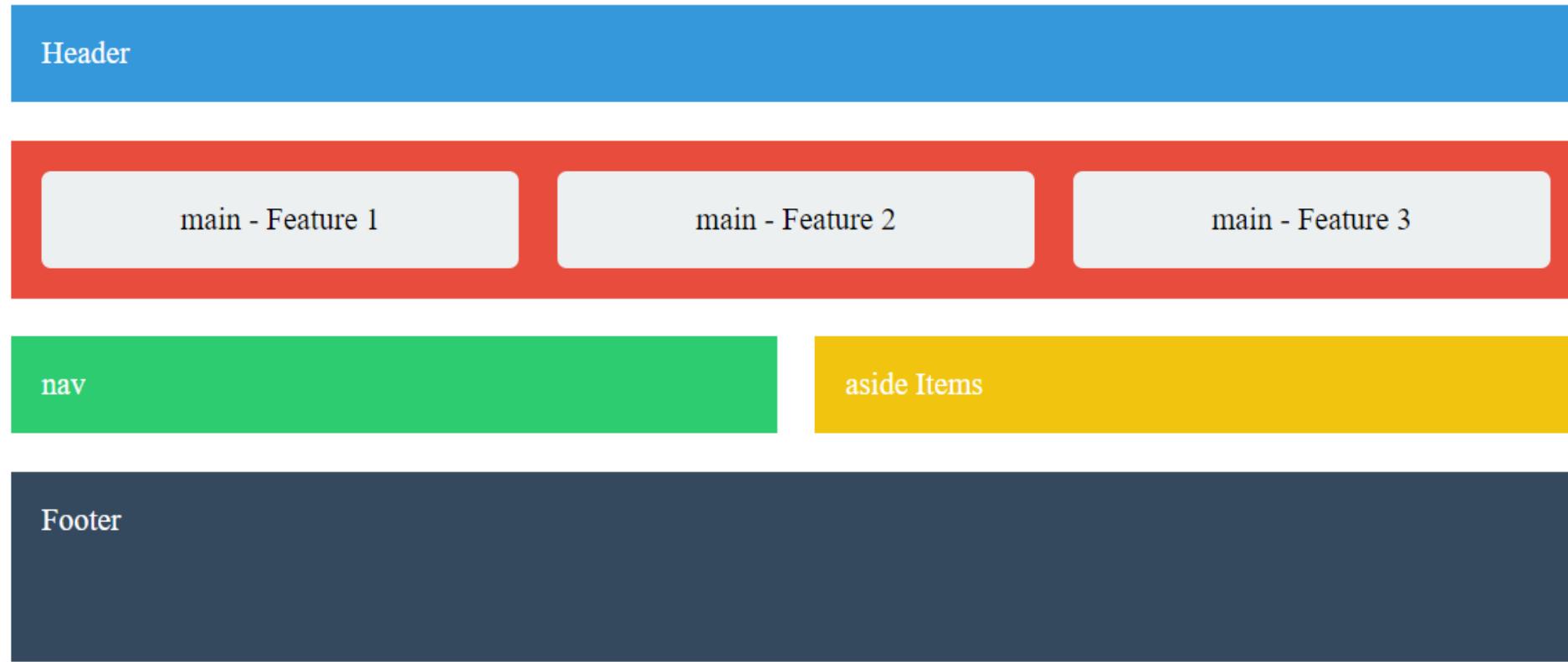
גודול מ x 960px, קטן מ 960px וגודול מ x 480px, קטן מ 480px

העימוד על פי המסכים בשקפים הבאים

Large screen



Tablet screen



Smart phone



```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title></title>
<style>

.main-container {
display: grid;
grid-template-columns: 1fr 3fr 1fr;
grid-template-rows: auto auto 100px;
grid-template-areas:
"header header header"
"nav main aside"
"footer footer footer";
gap: 20px;
}

/* Styles for grid items */
header {
grid-area: header;
background-color: #3498db;
color: white;
padding: 1rem;
}
nav {
grid-area: nav;
background-color: #2ecc71;
color: white;
padding: 1rem;
}
main {
grid-area: main;
background-color: #e74c3c;
padding: 1rem;
display: grid;
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
gap: 20px;
}
```

```
section {  
background-color: #ecf0f1;  
padding: 1rem;  
border-radius: 5px;  
text-align: center;  
}  
  
{
```

```
aside {  
grid-area: aside;  
background-color: #f1c40f;  
color: white;  
padding: 1rem;  
}  
  
{
```

```
footer {  
grid-area: footer;  
background-color: #34495e;  
color: white;  
padding: 1rem;  
}  
  
{
```

```
/* Responsiveness for tablet screens */  
@media screen and (max-width: 960px) {  
.main-container {  
grid-template-columns: 1fr 1fr;  
grid-template-rows: repeat(3, auto) 100px;  
grid-template-areas:  
"header header"  
"main main"  
"nav aside"  
"footer footer";  
}  
}
```

```
@media screen and (max-width: 480px) {  
    .main-container {  
        grid-template-columns: 1fr;  
        grid-template-rows: repeat(4, auto) 100px;  
        grid-template-areas:  
            "header"  
            "nav"  
            "main"  
            "aside"  
            "footer";  
    }  
  
    main {  
        grid-area: main;  
        background-color: #e74c3c;  
        padding: 1rem;  
        display: grid;  
        grid-template-columns: 1fr;  
        gap: 20px;  
    }  
}  
}  
</style>  
</head>  
  


Headernav

main - Feature 1



main - Feature 2



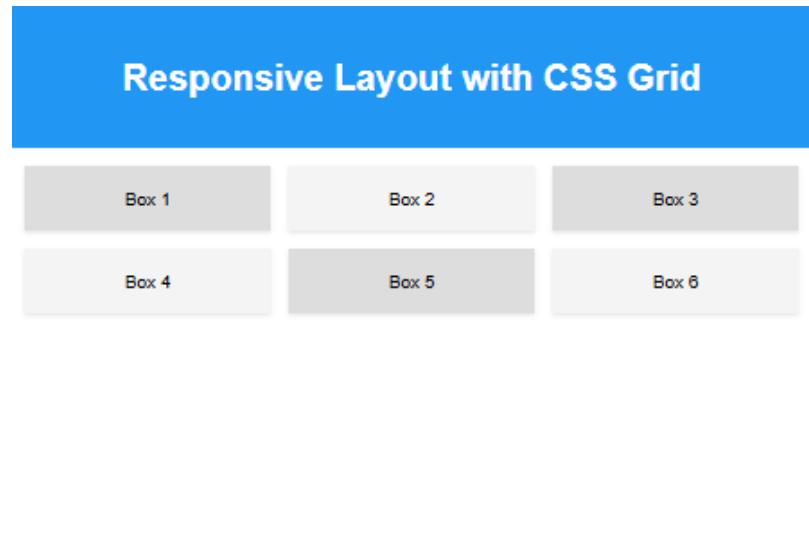
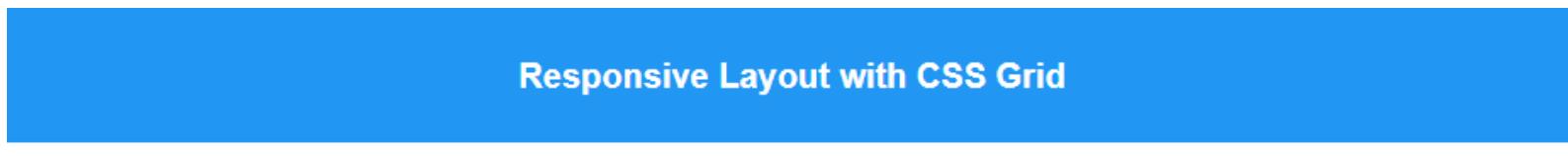
main - Feature 3

aside ItemsFooter


```

האם media query הכרחי?

הרבה פעמים יכולם לתת מענה לrspונסיביות



```
<body>
```

```
  <div class="header">
    <h1>Responsive Layout with CSS Grid</h1>
  </div>
```

```
  <div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
    <div class="box">Box 4</div>
    <div class="box">Box 5</div>
    <div class="box">Box 6</div>
  </div>
```

```
</body>
```

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Responsive Grid Example</title>
<style>
body {
    margin: 0;
    font-family: Arial, sans-serif;
    line-height: 1.6;
}

.header {
    background-color: #2196F3;
    color: white;
    padding: 1rem;
    text-align: center;
}

.container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); /* Auto-adjust
    columns */
    gap: 1rem;
    padding: 1rem;
}

.box {
    background-color: #f4f4f4;
    padding: 1rem;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.box:nth-child(odd) {
    background-color: #ddd;
}
</style>
</head>
```

auto-fit automatically creates as many columns as can fit in the container. If there is extra space, it stretches the columns to fill the space.

minmax(200px, 1fr) This column has a minimum width of 200 pixels. The maximum width is set to 1 fraction unit (1fr), which means it will take up the available free space proportionally along with other flexible columns using the fr unit.

שאלות?

CSS layout and Positioning

Normal Flow

- אפשר להסתכל על הדף כשורות ועמודות.
- התגיות יוצגו לפי הסדר בו הן כתובות אחת אחרי השנייה.
- אלמנטים שהם block יתפסו את כל השורה - כאשר שמייםvis בדף הוא תופס שורה שלמה (כג"ל `k, h` ותגיות נוספות).
- אלמנטים שהם inline יתפסו עמודה - כאשר שמיים `span` בדף הוא תופס עמודה בשורה, ואם נוספים עוד הרבה `span`ים לבסוף הם יתפסו את כל השורה ואז תבוצע ירידת שורה.
- אלמנטים שהם `float` יצפו שמאליה או ימינה מחוץ ל flow normal וישפיעו על האלמנטים האחריהם.

Display property

CSS Syntax

```
display: value;
```

Property Values

Value

inline

block

contents

flex

grid

inline-block

- כל אלמנט בדף הוא קופסא, ה `display` קובע איך הקופסא מתנהגת.
- לכל אלמנט יש `display` בעל ערך `default`, אם ברצוננו לשנותו יש לדרכו אותו ב CSS

block, Inline, inline-block

- **block** מתנהג הפור מ **inline**
- **inline** ו **inline-block** מתנהגים אותו דבר אבל ב **block** ב הגובה
- והרוחב דינמיים
- ל **inline** לא ניתן לתת גובה הוא מקבל את המימדים שלו מהאבא.

visibility vs display

```
h2.a {  
  visibility: visible;  
}
```

```
h2.b {  
  visibility: hidden;  
}
```

```
<head>
<style>
  h2.a {
    visibility: visible;
  }
  h2.b {
    visibility: hidden;
  }
</style>
</head>
<body>
  <h1>The visibility Property</h1>
  <h2 class="a">This heading is visible</h2>
  <h2 class="b">This heading is hidden</h2>
  <p>Notice that the hidden heading still takes up space on the page.</p>
</body>
```

The visibility Property

This heading is visible

Notice that the hidden heading still takes up space on the page.

```
<head>
  <style>
    h1.hidden {
      display: none;
    }
  </style>
</head>
<body>

  <h1>This is a visible heading</h1>
  <h1 class="hidden">This is a hidden heading</h1>
  <p>Notice that the h1 element with display: none;  
does not take up any space.</p>

</body>
```

This is a visible heading

Notice that the h1 element with display: none; does not take up any space.

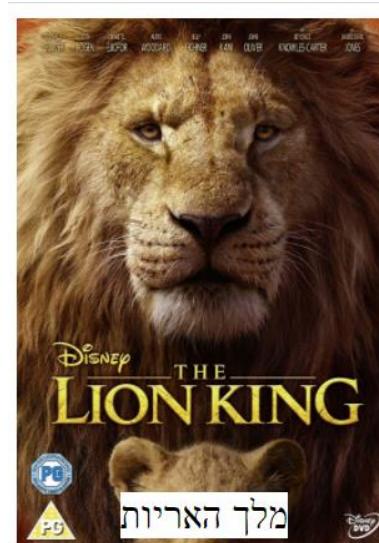
Positioning

- לפעמים נרצה שאלמנטים "יסתרו" לא כמו ב normal flow

מה קורה אם אנחנו רוצים לשים אלמנט מתחתית הדף למרות שעוז לא הגענו עד אליו?



אר מה קורה אם אנחנו רוצים לשים אלמנט על אלמנט? למשל שתיאור התמונה יופיע על גבי התמונה?



Position

מיקום האלמנט בדף:

Static.1

Relative.2

Absolute.3

Fixed.4

Sticky.5

בנוסף עם הגדרת top, right, bottom, and left נקלט את המיקום הסופי.

static

position: static

- כל אלמנט html ב Normal flow מוגדר בדף כתטתי
- הכוונה שהוא תופס את המקום הבא בדף

position:relative

```
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    .positioned {
      position: static;
      background: yellow;
    }
  </style>
</head>
<body>
  <p class="positioned">Hi, nice to meet you</p>
</body>
```

```
<head>
  <meta charset="utf-8" />
  <title></title>
  <style>
    .positioned {
      position: relative;
      background: yellow;
    }
  </style>
</head>
<body>
  <p class="positioned">Hi, nice to meet you</p>
</body>
```

ללא שינוי

position:relative

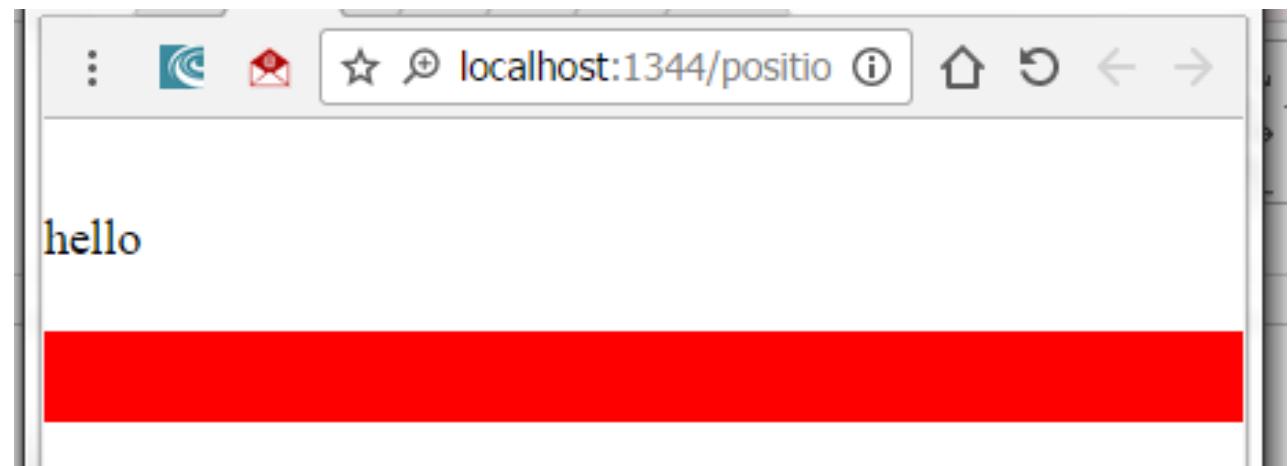
- relative מוציא את האלמנט מהמצב ה static שלו.
- relative בפניהם עצמו לא יעשה שינוי, יש להוסיף: Top, bottom, left, right
- ההזזה תבוצע **יחסית למקומו המקורי של האלמנט** ב normal flow באמצעות תכונות top, left, bottom, right.
- משאיר את המקום של האלמנט "תפוס".
- לא משנה את התכונות הבסיסיות של האלמנט (רוחב, גובה).

[positionRelative2.html](#), [positionRelative1.html](#)

position:relative

זה גם יאפשר לנו להציג אלמנט על אלמנט.

```
|<body>  
|  
| <div style="height:30px; background-color:red; position:relative; top:70px"></div>  
| <span>hello</span>  
  
</body>
```



Position:absolute

– **לא** משאיר את המקום של האלמנט "תפוא". האלמנט הופך להיות "צף" על המסר. משנה את התכונות הבסיסיות של האלמנט, הרוחב והגובה יהיו דינמיים במידה ולא יוגדרו.

```
]<body>
```

```
<div style="height:30px; background-color:red; position:absolute; top:70px; width:100%"></div>
<span>hello</span>
```

```
</body>
```

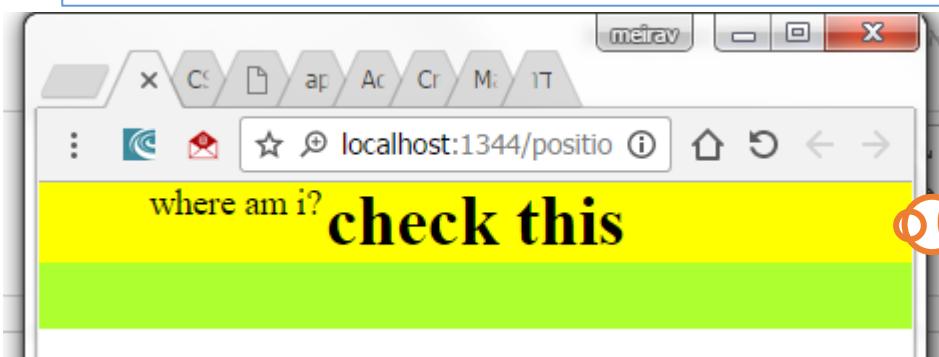


Position:absolute

היכן לדעתכם תופיעת תגית ה-span?

```
<body>
  <h1 style="text-align:center; background-color:yellow; margin:0px">check this</h1>

  <div style="height:30px; background-color:greenyellow">
    <span style="position:absolute; top:0px; left:50px">where am i?</span>
  </div>
</body>
```



במקרה של absolute - האלמנט מבצע חישוב של
היחס לפי "הוורה" הראשון שאינו static.

במקרה זה אין, ולכן הוא מתמוקם לפני Tag body!
*תזכורת – ה default של כל אלמנט הוא static

הערת Z-index

עובד רק על אלמנטים שהם position:

absolute •

relative •

fixed •

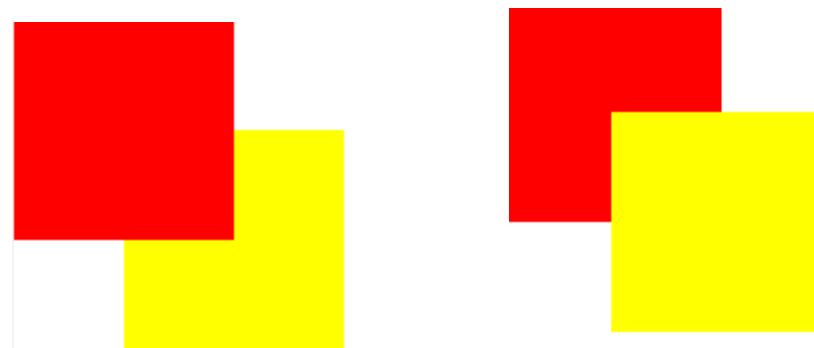
sticky •

flex •

*אם שני אלמנטים חופפים ללא z-index האחרון בקוד יעלה על הקודם לו.

Z-index

- ראיינו שעל ידי absolute ניתן "להציג" אלמנט על המסך, אך מה קורה אם אני רוצה שהוא יוצג מתחת לאלמנט?

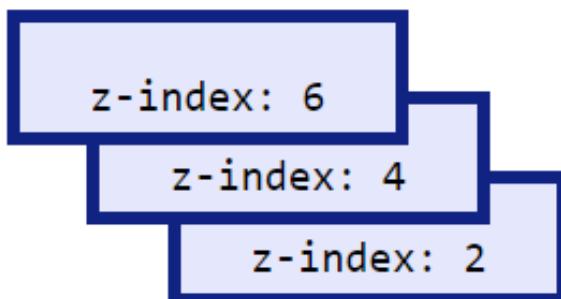


Z-index

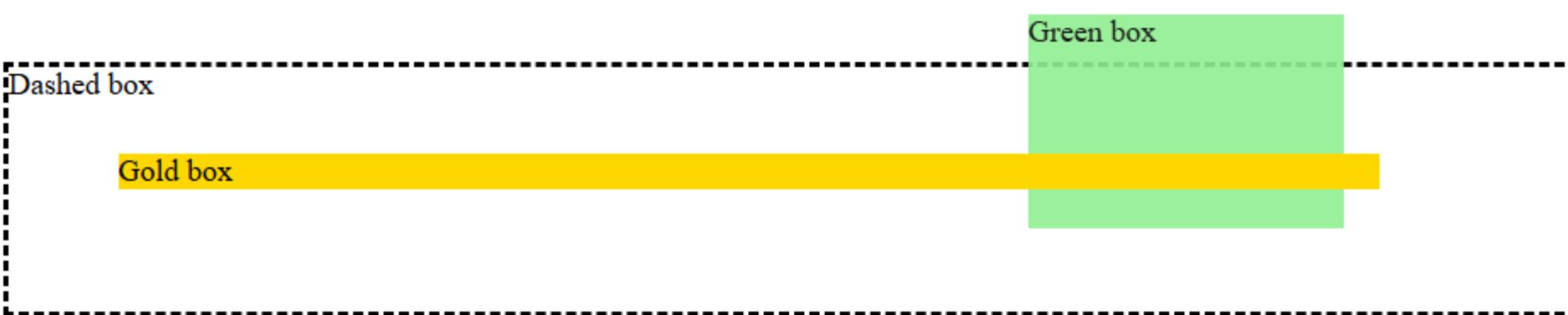
על מנת לשלוט במי שיפיע, נעשה שימוש בתכונת `z-index` שנמצאת בתוך תכונת `style`.

התכונה מקבלת ערך **שלילי או חיובי**

- שלילי יהיה כשרוצים להוריד את ה priority
- כללי אלמנט בעל ערך z גדול מערך של אלמנט אחר יופיע מעל.
- לפי חוק זה ערכי `z-index` של 2 ו 3 עבור שני אלמנטים יתנהגו بدיקן כמו 0000 | 0000 בהתאם.



Z-index



```
.dashed-box {  
position: relative;  
z-index: 1;  
border: dashed;  
height: 8em;  
margin-bottom: 1em;  
margin-top: 2em;  
}
```

```
.green-box {  
position: absolute;  
z-index: 2; /* put .green-  
box above .dashed-box */  
background: lightgreen;  
width: 20%;  
left: 65%;  
top: -25px;  
height: 7em;  
opacity: 0.9;  
}
```

```
.gold-box {  
position: absolute;  
z-index: 3; /* put .gold-box  
above .green-box and .dashed-box  
*/  
background: gold;  
width: 80%;  
left: 60px;  
top: 3em;  
}
```

ZindexExample.html

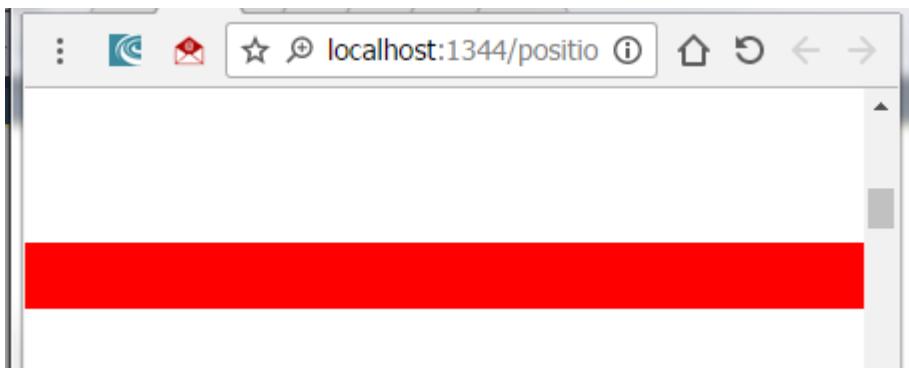
Position:fixed

fixed הוא כמו absolute, רק במקרה זה המיקום יהיה קבוע גם אם נ滾ול את המסך.

```
]<body>
```

```
  <div style="height:30px; background-color:red; position:fixed; top:70px; width:100%"></div>
  <span>hello</span>

  <div style="height:6000px"></div>
</body>
```



נzie את האלמנט **יחסית לחלון הדף**
באמצעות תכונות top, left, bottom, right

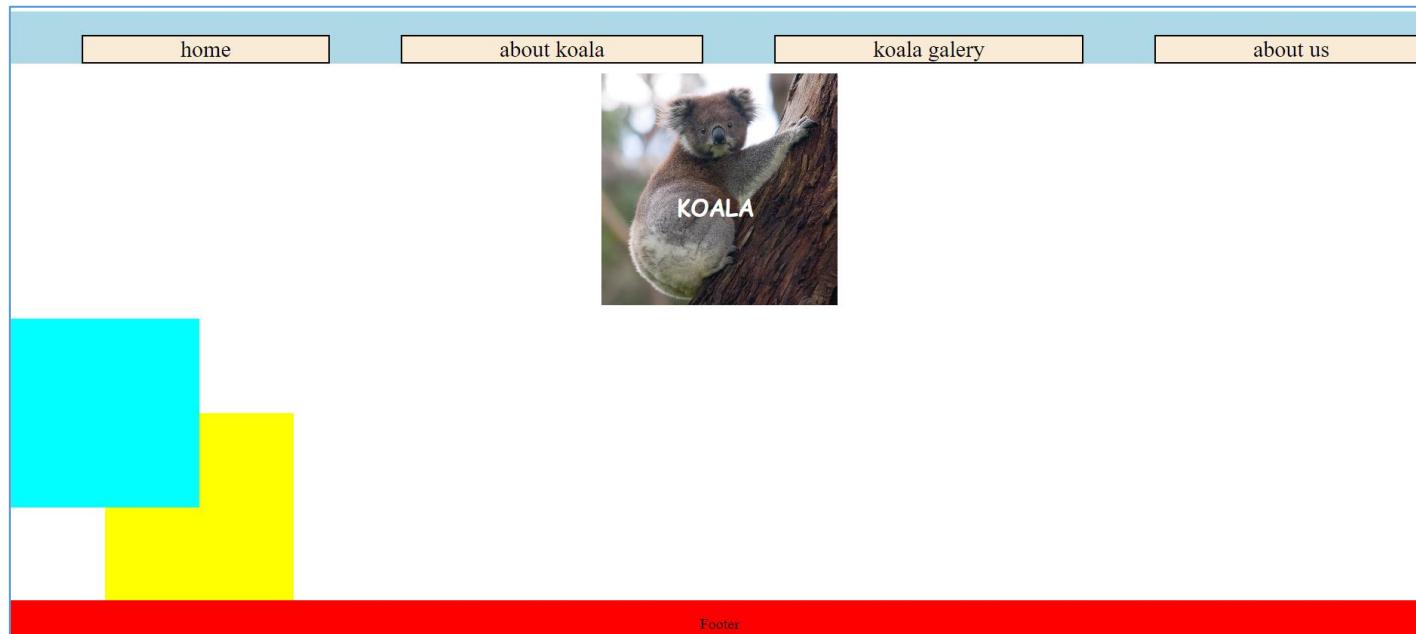
position:sticky

- ממוקם על פי מיקום ה scroll של הדף
- נע בין relative ל fixed, תלוי במיקום ה scroll
- הוא יהיה כמו relative עד נקודה מסוימת ואז הוא "נתקע" במקומו כמו fixed

Sticky.html •

Stickyniceexample.html •

תרגיל ביתה



צור דף חדש המכיל:

- 1) **תפריט עליון** header קבוע השתמשו ב sticky
- 2) מתחתי תמונה עם טקסט ממורכז על גבי התמונה
- 3) מתחתי שתי קוביות בגודל $200px \times 200px$ פנימה ולמטה כשר התחתונה $100px$ מופיעה מעל העליונה
- 4) **תפריט תחתון** Footer קבוע שמוופיע כל הזמן השתמשו ב sticky/fixed

home

about koala

koala galery

about us



Footer

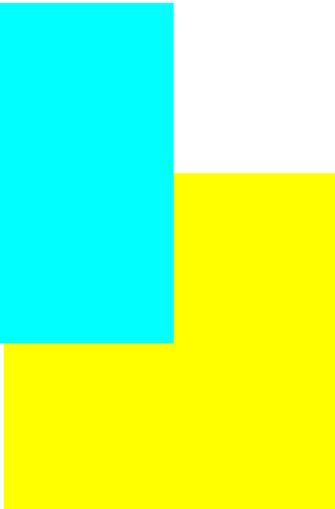
על סלע זעירא

home

about koala

koala galery

about us



Footer

על סלע זעירא

מבוא ל JavaScript

קורס : צד ל Koh

מרצה: יעל סלע זעירא

היום בהרצאה

1. Intro JS רקע כללי

2. אירועים ופונקציות (מבוא)

Variables .3

Data Types .4

getElement .5

querySelector .6

DOM .7

parse .8

Javascript Intro

HTML, CSS, JS, היבת האינטרנט מטרתה להפוך את דף HTML בסיסי לדינמי.

3 שכבות:

שכבה ה HTML

שכבה ה CSS

שכבה ה Javascript

דוגמא דף: JSIntro1.html

Javascript Intro

- שפת תכנות עילית כמו JAVA/C, אך מדובר בשפת script ולא בשפת קומpileציה.
- הקוד נקרא ומחילה לרווח מלמעלה למטה, لكن יש חשיבות לסדר, והתוצאה מייד מוצגת.
- הקוד רץ בצד הלקוח ב browser.

Add Javascript code

```
<script>  
    // JavaScript goes here  
</script>
```

OR

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

<script>

תג script יכול להופיע בתג head ו/או בתג body.

```
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script>
        //code block 1
    </script>
</head>
<body>
    <script>
        //code block 2
    </script>
</body>
</html>
```

alert

לשפה יש פונקציות מובנות, הראשונה שנכיר זו פונקציית alert שמקפיצה חלון התראה במסר.

```
<head>
  <script>
    alert("Hello from javascript")
  </script>
</head>
```

Avoiding alert

1. עוצר את הרצתה javascript עד אשר המשתמש משחרר את ה dialog
 2. לא יידotti למשתמש
 3. חלון מיושן והעיצוב עבورو דל
- **לצרכי debug עדיף להשתמש ב `console.log`**

Avoiding alert

- כאשר בכל זאת רוצים alert כדאי להשתמש בכל modal אחר למשל `<dialog>`

```
<dialog id="myDialog">This is a modern alert!</dialog>
<script>
  const dialog = document.getElementById('myDialog');
  dialog.showModal();
  setTimeout(() => dialog.close(), 2000); // Auto-close after 2 seconds
</script>
```

console.log

```
<head>
  <script>
    console.log("Hello from javascript")
  </script>
</head>
```

function

לפונקציות ב-javascript אין טיפוס החזרה. הגדרת פונקציה נעשית באמצעות המילה
השמורה **function**

```
//func is the name of the function
function func()
{
    //code goes here
}
```

```
func(); //use () on the function name to activate the function
```

function

בתוך תג script ניתן לשים קטעי קוד רבים, בד"כ בתג יי'ו הרבה פונקציות שיחווטו לפעולות שיבצע המשתמש.

קטע הקוד לא יציג את ה alert

```
<script>  
  
    function userClickAlert()  
    {  
        alert("user click!");  
    }  
  
</script>
```

קטע הקוד יציג את ה alert

```
<script>  
  
    function userClickAlert()  
    {  
        alert("user click!");  
    }  
    userClickAlert();  
  
</script>
```

event

הדרך פשוטה ביותר לחווט אלמנט לפונקציה, היא דרך הוסףת **תכונת אירוע לאלמנט** עצמו:

```
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script>
    function userClickAlert()
    {
      alert("user click!");
    }
  </script>
</head>
<body>
  <input type="button" value="click me" onclick="userClickAlert()" />
</body>
</html>
```

יש עוד רשימת אירועים גדולות
שנכירות בהמשך
ודרכים נוספות לחווט אירועים.

תבניות מונחה אירועים

אתרי האינטרנט בנויים על אירועים שמייצר המשתמש.

מרבית דפי האינטרנט שתגיבו אליהם, לא יעשו כלום כל עוד המשתמש לא

ביצע דבר.

**שינוי טקסט
או פריט
ברשימה**
onchange

פוקואו
onfocus

**מעבר
עכבר**
onmouseover

**לחיצה על
עכבר**
onclick

**סוגי און
לחיצת מקש**
onkeydown

חיבור אירוע לאלמנט

- **חיבור אירוע לאלמנט יכול להתבצע בצורה סטטית או דינמית (היום אנו רואים רק את הצורה הסטטית).**
- החיבור של לחיצת כפתור שביצענו עד כה הוא דוגמא לחבר סטטי.
- **נש אלמנטים ועלא תומכו באורוועס מטען ורמאר בז האירוע פונטו**

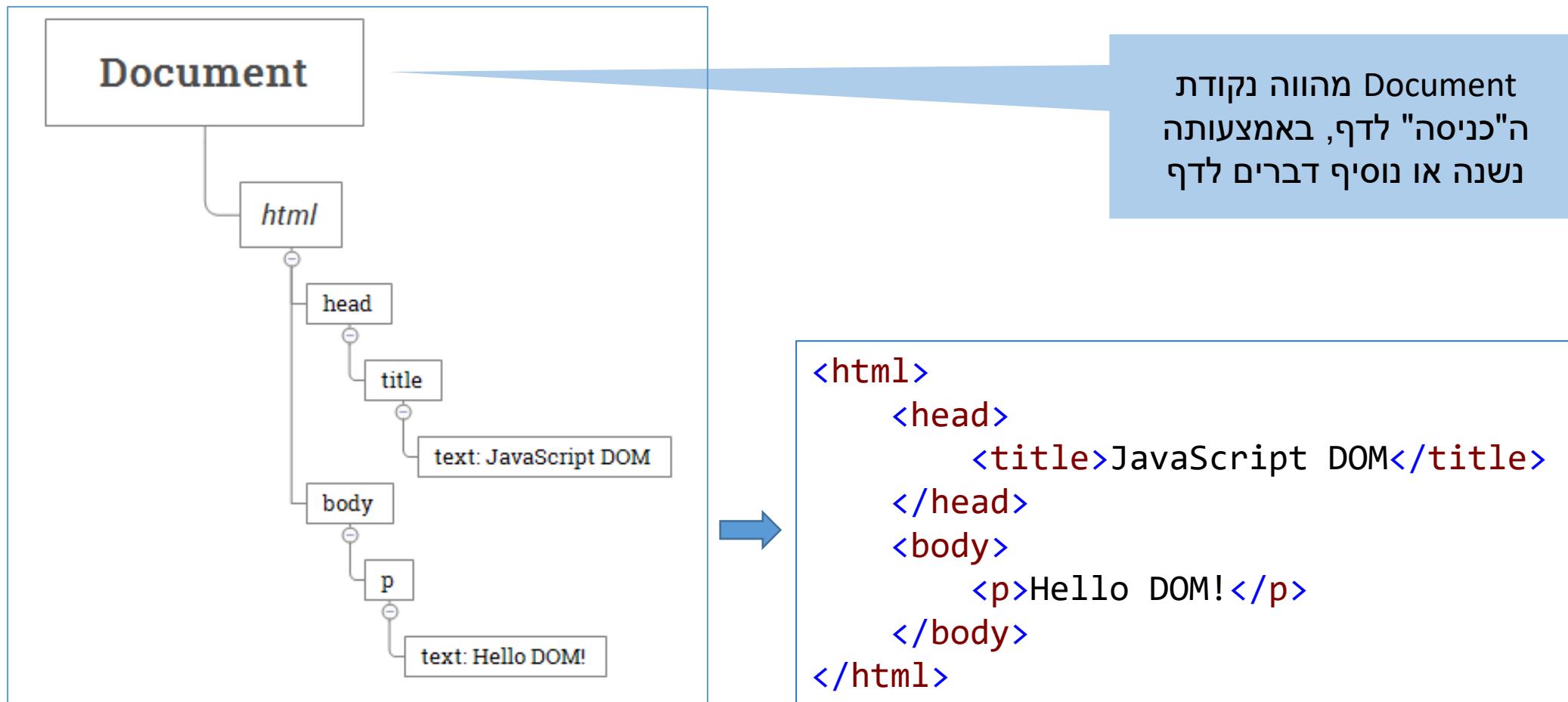
```
<input id="btn" type="button" value="change to red color" onclick="changeColor()"/>
```

לא יופעל

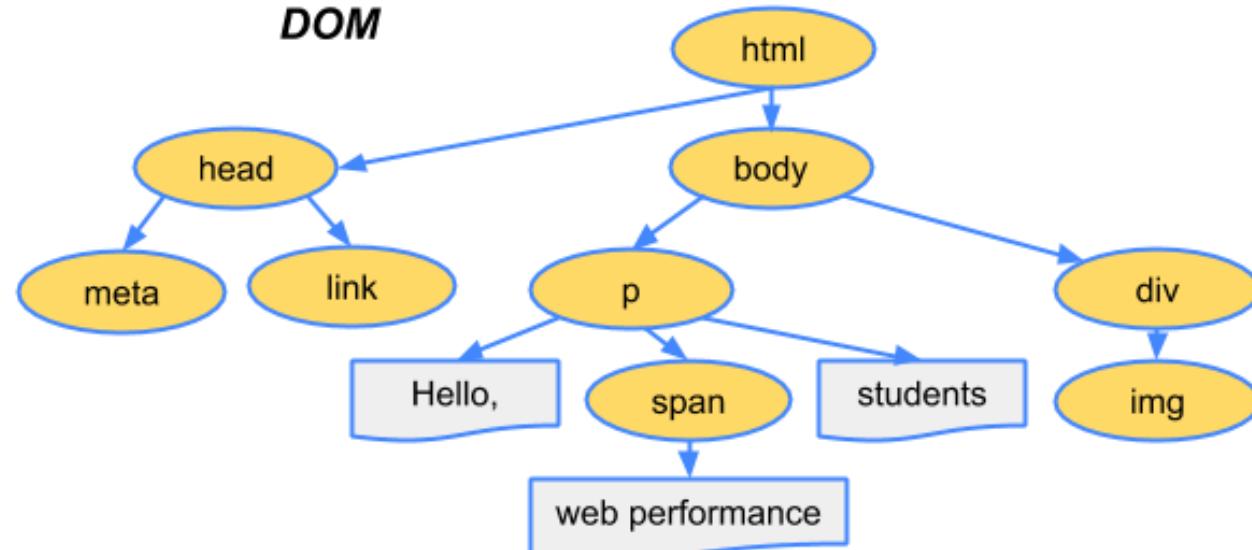
חיבור סטטי

DOM – Document object model

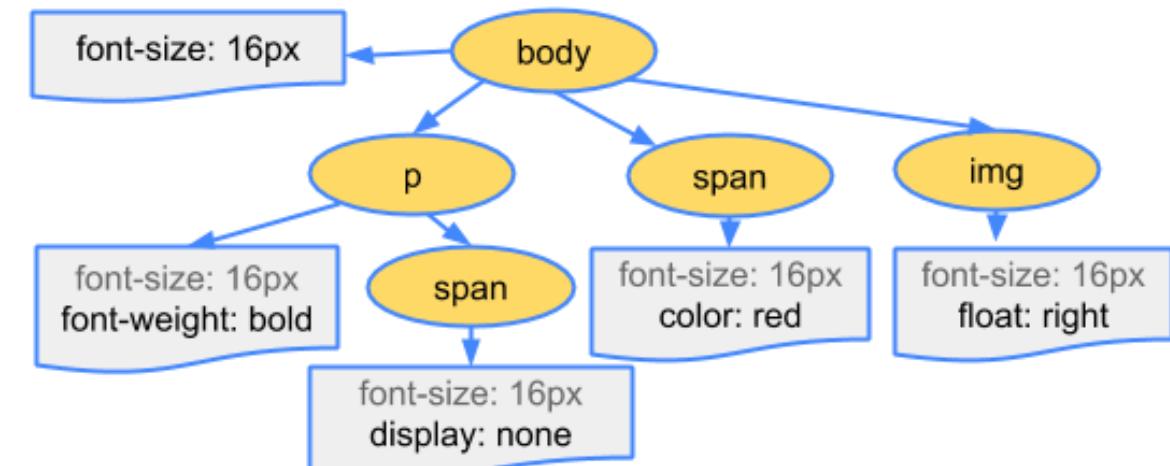
- מודל אותו הבראוזר מייצר טרם עליית הדף. המודל מייצג את כל האלמנטים שנמצאים בדף ה html



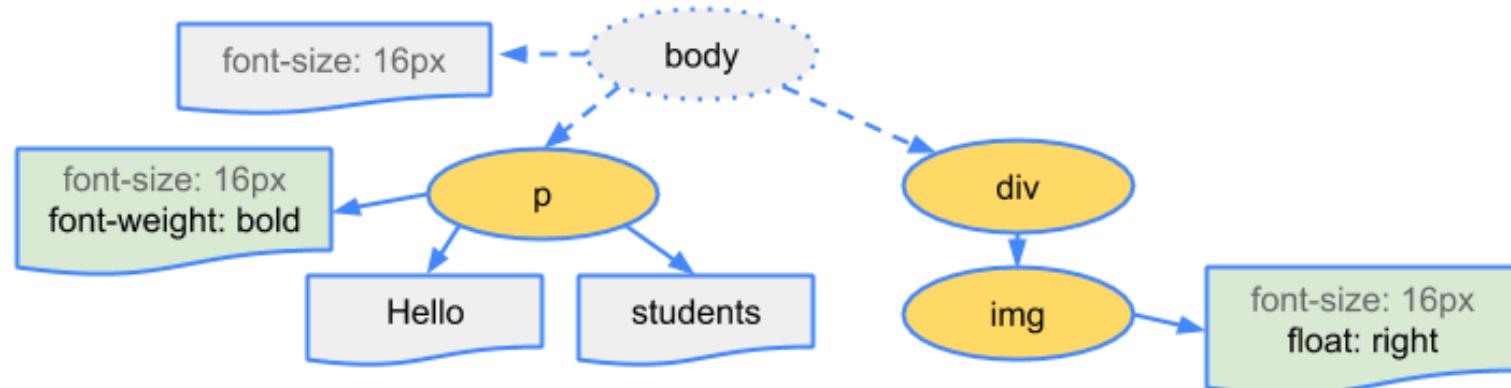
DOM



CSSOM



Render Tree



גישה לאלמנט

- אחות הפקציות בגישה לאלמנטים היא `getElementById`

```
document.getElementById("someId") •
```

- מביאה אלמנט html לפי ה-`id` שלו, כך ניתן לגשת לתכונות האלמנט ולשנות את ערכם.
- `id` של אלמנט הוא `unique`.
- אם ה `id` לא קיים מוחזר `null`.

שינוי אלמנט

נכיה ובלחיצה על כפתור נרצה לשנות את טקסט הcptor – הכניסו לפונקציה:

```
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script>
    function userClick()
    {
      document.getElementById("btn").value = "user click me!";
    }
  </script>
</head>
<body>
  <input type="button" value="click me" onclick="userClick()" id="btn"/>
</body>
</html>
```

שינוי אלמנט

נניח ובלחיצת על כפתור נרצה לשנות את צבע הטקסט של הכפתור
זו תכונה מורכבת (אובייקט) שיש לה תכונות נוספות.

```
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script>
    function changeTextColor()
    {
      document.getElementById("btn").style.color = "red";
    }
  </script>
</head>
<body>
  <input type="button" value="change my text color" onclick="changeTextColor()" id="btn"/>
</body>
</html>
```

על סלע זעירא

onmouseover, onmouseout

שינוי צבע כפטור לפי מעבר עם עכבר ויציאה עם עכבר (onmouseover, onmouseout):

```
<input id="btn" type="button" value="hover me" onmouseover="setColor()" onmouseout="resetColor()"/>

<script>
    function setColor()
    {
        document.getElementById("btn").style.color = "red";
    }

    function resetColor()
    {
        document.getElementById("btn").style.color = "";
    }
</script>
```

onfocus, onblur

שינוי צבע תיבת טקסט כאשר בפוקוס או ביציאה מהתיבה (העכבר בקליק על תיבת הטקסט - blur)

```
<input id="btn" type="text" onfocus="setBackColor()" onblur="resetBackColor()"/>

<script>
    function setBackColor()
    {
        document.getElementById("btn").style.backgroundColor = "red";
    }

    function resetBackColor()
    {
        document.getElementById("btn").style.backgroundColor = "";
    }
</script>
```

onchange

שינוי פריט ברשימה נגלהת (onchange) - עשו alert עם שינוי (select....option)

```
<select onchange="doSomething()">
    <option>1</option>
    <option>2</option>
    <option>3</option>
</select>
```

onkeyup

ברגע עזיבת מקלש מתיבת טקסט שנו את הצבע של הceptor
:(onkeyup)

```
<input id="btn" type="text" onkeyup="setBackColor()"/>

<script>
    function setBackColor()
    {
        var x = document.getElementById("btn");
        x.style.backgroundColor = x.value;
    }

</script>
```

```
//add css class
function myFunction() {
    var element = document.getElementById("btn");
    element.classList.add("myStyle");
}
```

אירועים נוספים בסגנון:

onkeydown – ברגע שceptor נלחץ (כלceptor ירוץ בمعالים עד עזיבת המקלש)

onkeypress – ברגע שceptor נלחץ משמעותו (ירוץ בمعالים עד עזיבת המקלש)

לרשימת אירועים נוספים

תרגיל ביתה 1

- 1) צרו 2 כפתורים רדיוס left ו right, כל לחיצה תזיז את כוורתת הדף בהתאם (רק אחד יכול להיות מסומן).
- 2) צרו תיבת טקסט וכפתור, כאשר בלחיצה על הכפתור יעבר הטקסט שכתוב בתיבה לטקסט הכתוב על הכפתור (י עברו=ימחק מהתיבה).



jsIntroEx1.html

this

לפעמים נרצה להעביר לפונקציה את כל האובייקט שלחצנו עליו:

```
<input type="text" id="txt" />
<input type="button" value="btn" id="happy" onclick="f1(this)" />

<script>
  function f1(cntrl) {
    document.getElementById("txt").value = cntrl.id;
  }
</script>
```

בשביל לדעת תכונות מסוימות שאו בכלל לידע על מי לחצנו.

this

```
<html>
<script>
function colorChanger(el) {
    el.style.backgroundColor = '#007d00';
}
</script>
<body>
    <h2>The JavaScript <i>this</i> Keyword</h2>
    <button onclick="colorChanger(this)">Click to change Me!</button>
</body>
</html>
```

variables

```
<head>
  <title></title>
  <meta charset="utf-8" />
  <script>

    function userClick()
    {
      document.getElementById("btn").value = "user click me!";
      document.getElementById("btn").style.color = "red";
    }

  </script>
</head>
<body>

  <input type="button" id="btn" value="click me" onclick="userClick()" />

</body>
```

- ביצועים – זמן ריצה ארוכים יותר
- קוד אורך ולא נקי

variables

- המילה השמורה `let` מגדירה משתנה
- ב-Javascript **אין הגדרה לטיפוס** - משתנה הוא מיכל לשימירת ערכים
- `x!=X` היא Javascript -
*ניתן להגדיר משתנה גם עם `const` ו-`var` נראה בהמשך

let

- הגדרת משתנה מתבצעת על ידי המילה **let**
- לדוגמה – הוצאה על המשתנה **: message**

```
let message;
```

```
let message;  
message = 'Hello!';  
console.log(message); // shows the variable  
content
```

- cutת נשים לתוכו ערך :

```
let message = 'Hello!'; // define the variable and assign the value  
console.log(message); // Hello!
```

let

```
let user = 'John', age = 25, message = 'Hello';
```

עדיף בצורה הבאה: ארוך אבל יותר קרייא



```
let user = 'John';
let age = 25;
let message = 'Hello';
```



```
let message = "This";
// repeated 'let' leads to an error
let message = "That"; // SyntaxError: 'message' has already been declared
```

לא ניתן להציג על אותו משתנה פעמיים



שמות משתנים

- **שתי מוגבלות:**

1. השם חייב להכיל אותיות או/ו מספרים או/ו את הסימנים `_`, `$`.
2. האות הראשונה חייבת להיות לא ספירה

- **דוגמאות:**

```
let userName;  
let test123;  
let $ = 1; // declared a variable with the name "$"  
let _ = 2; // and now a variable with the name "_"  
  
console.log($); // 1  
console.log(_); // 2
```

*הגדרת משתנה תעבור גם ללא המילה `let` – אך זה תרנגול גורען

- הגדרות שגויות:



```
let 1a; // cannot start with a digit  
let my-name; // hyphens '-' aren't allowed in the name
```

סוגי משתנים

```
let a = 1;           //number
let b = 1.4;         //number
let c = "ff";        //string
let d = 'gg';        //string
let e = [];          //array (the type is object)
let f = {};          //object
let g = true;         //boolean
let h = null;         //null (the type is object)
let i;                //undefined
let l = alert;        //function
```

```
let a = 1;
console.log(typeof a); // number
```

undefined and null

משתנים שהוכרזו אך לא אוטחלו או שאותחלו ב undefined
יתנהגו בצורה הבאה:

- 1) `typeof` returns string 'undefined'
- 2) `==` check with null returns true
- 3) `==` check with undefined returns true
- 4) `====` check with null returns false
- 5) `====` check with undefined returns true

undefined and null

משתנים שאוחחלו בערך null

- 1) `typeof` returns string 'object'
- 2) `==` check with null returns true
- 3) `==` check with undefined returns true
- 4) `===` check with null returns true
- 5) `===` check with undefined returns false

constatnts

- הגדרת קבוע על ידי המילה `const`
- המשתנה – לא ניתן לשינוי

```
const myBirthday = '18.04.1982';
```

```
myBirthday = '01.01.2001'; // error, can't reassing the constant!
```

uppercase constants

- נוהג לחתת לערכים שקשה לזכור והם קבועים וידועים מראש, שמות ב case upper.
- כולם ערכי hard-coded שידועים ניתן להם שמות uppercase

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

// ...when we need to pick a color
let color = COLOR_ORANGE;
console.log(color); // #FF7F00
```

על סלע זעירא

- יתרונות:
- קל לזכור ולהשתמש בו
 - פחות טעויות
 - שם משמעותי ומובן

כלי אבע לשמות משתנים

1. שמות מובנים וקראיים למשל : userName, shoppingCart
2. להימנע מממשתנים שקוראים להם : a, b, c....
3. שמות בעלי משמעות למשל השמות : data, value לא באמת אומרים לנו מהו המשתנה
4. שיחזור משתנים, אין צורך, הגדרו משתנה חדש הבראוזרים היומם מהיריים.

תרגיל ביתה 2

• כתוב פונקציה בשם swap

- הפונקציה מקבלת שני משתנים y , x עם הערךים 111 ו 333
- מבצעת להם swap
- מדפסה לקונסול את x ו y
- וודא שהערכים התחלפו

שאלה

- איזה מהשמות של הקבועים הבאים היה משנה אם בכלל?

```
const BIRTHDAY = '18.04.1982'; // uppercase OK?
```

```
const AGE = someCode(BIRTHDAY); // uppercase OK?
```

פתרונות:
כיוון ש age מחושב ב runtime עדיף יהיה להגדיר אותו באותיות קטנות

פעולות מתמטיות על משתנים

פעולות מתמטיות על טיפוסים מסוג `number` מתנהגות כרגיל.

פעולות מתמטיות על טיפוסים מסוג `string` - כאשר מדובר במספר - מתנהגות כרגיל, להוציא
פעולות חיבור.

```
"5" - "5"; //equal 0  
"5" / "5"; //equal 1  
"5" * "5"; //equal 25  
"5" + "5"; //equal 55
```

```
"5" - 5; //equal 0  
"5" / 5; //equal 1  
"5" * 5; //equal 25  
"5" + 5; //equal 55
```

חיבור מחרוזות בין עצמן או עם מספר – משרשר את הערכים למחוזת אחת:

*אם נבצע פעולות מתמטיות בין שני משתנים, תרגול טוב יהיה להפוך מחרוזת למספר לפני ביצוע הפעולה (<https://dev.to/sanchithasr/7-ways-to-convert-a-string-to-number-in-javascript-4l>)

"hello " + 10; //??

"hello " + 10 + 5; //??

10 + 5 + " hello"; //??

hello 10

hello 105

15 hello

```
<script>
// no error
let message = "hello";
message = 123456;
//number
let n = 123;
n = 12.345;
//special numeric values- Infinity, -Infinity and NaN.
console.log(1 / 0); // Infinity
console.log(Infinity); // Infinity
console.log("not a number" / 2); // NaN,
such division is erroneous

console.log(NaN + 1); // NaN
console.log(3 * NaN); // NaN
console.log("not a number" / 2 - 1); // NaN
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

```
let name = "John";
// embed a variable
console.log(`Hello, ${name}!`); // Hello, John!
// embed an expression
console.log(`the result is ${1 + 2}`); // the
result is 3
let nameFieldChecked = true; // yes, name field is
checked
let ageFieldChecked = false; // no, age field is
not checked
let isGreater = 4 > 1;
console.log(isGreater); // true (the comparison
result is "yes")
let age = 100;
// change the value to undefined
age = undefined;
console.log(age); // "undefined"
</script>
```

משתנה גלובלי

- יוגדר **ישירות בסקריפט** לא בתוך פונקציה
- המשתנה יהיה זמין לכל הפונקציות

String conversion

```
let value = true;  
console.log(typeof value); // boolean  
  
value = String(value); // now value is a string "true"  
console.log(typeof value); // string
```

Numeric conversion

```
console.log("6" / "2"); // 3, strings are converted to numbers
```

```
let str = "123";
console.log(typeof str); // string
```

```
let num = Number(str); // becomes a number 123
```

```
console.log(typeof num); // number
```

```
let age = Number("an arbitrary string instead of a number");

console.log(age); // NaN, conversion failed

console.log(Number(" 123  ")); // 123

console.log(Number("123z")); // NaN (error reading a number at "z")

console.log(Number(true)); // 1

console.log(Number(false)); // 0
```

parseInt()

- | | |
|--|--------|
| a) <code>parseInt("10");</code> | a) 10 |
| b) <code>parseInt("10.00");</code> | b) 10 |
| c) <code>parseInt("10.33");</code> | c) 10 |
| d) <code>parseInt("34 45 66");</code> | d) 34 |
| e) <code>parseInt(" 60);</code> | e) 60 |
| f) <code>parseInt("40 years");</code> | f) 40 |
| g) <code>parseInt("He was 40");</code> | g) Nan |
| h) <code>parseInt("010") ;</code> | h) 10 |

המרת למספרים - שימוש בקוד

```
<html>
<head>
  <title></title>
  <meta charset="utf-8" />

</head>
<body>
  <input type="button" value="conuter" onclick="counterAdd()" />
  <span id="spn">0</span>

  <script>
    var x = document.getElementById("spn");

    function counterAdd()
    {
      x.innerHTML = parseInt(x.innerHTML) + 1;
    }
  </script>
</body>
</html>
```

נביט בתרגיל הכתיבה של ה `:counter`

Boolean conversion

```
console.log(Boolean(1)); // true
```

```
console.log(Boolean(0)); // false
```

```
console.log(Boolean("hello")); // true
```

```
console.log(Boolean("")); // false
```

תוכן אלמנט

```
<input />  
<textarea></textarea>    document.getElementById("inputId").value;
```

```
<div></div>  
<h1></h1>  
<span></span>    document.getElementById("divId").innerHTML;  
<p></p>  
<a></a>  
<ul>  
    <li></li>  
</ul>
```

תרגיל ביתה 3

צרו מונה שסופר קליקים באמצעות כפטור
לחיצה על הcpfutor:

Plus One

0

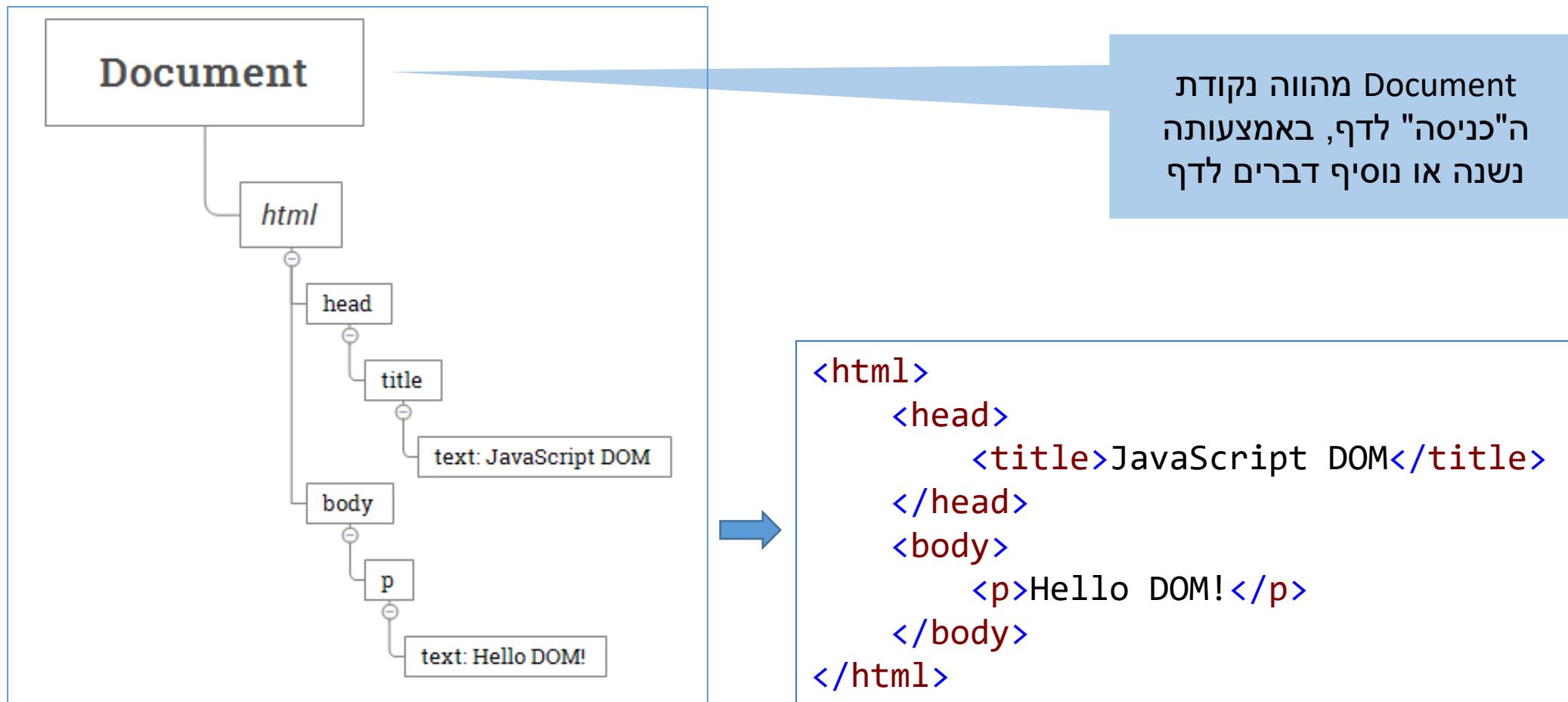
Minus One

- מעלה את המונה ב 1

צרו כפטור נוסף שמוריד קליקים מהמונה
לחיצה על הcpfutor:
• מוריד את המונה ב 1

DOM – Document object model

- מודל אותו הבראוזר מייצר טרם עליית הדף. המודל מייצג את כל האלמנטים שנמצאים בדף ה html



DOM, CSSOM, Script

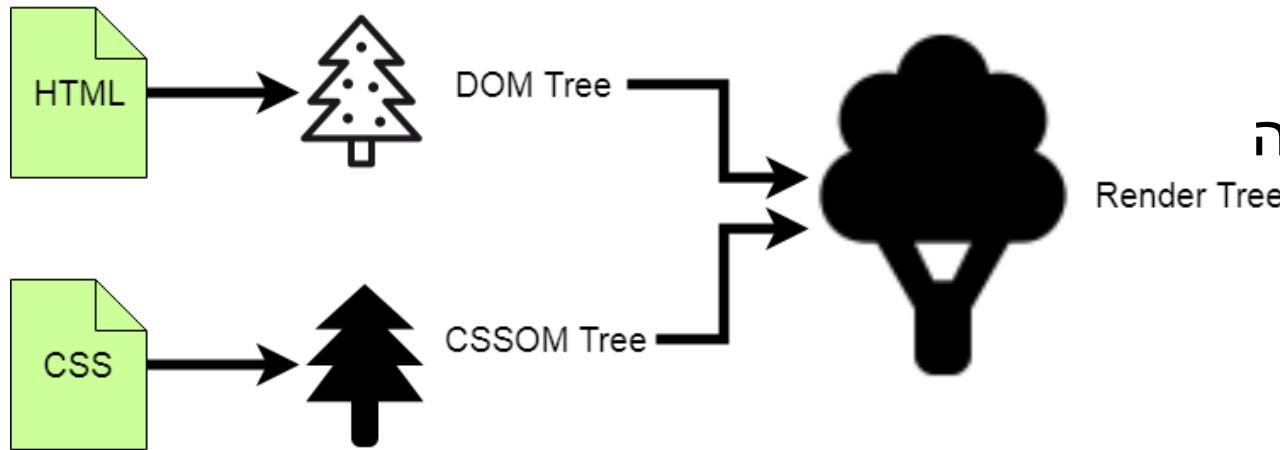
מנוע הקריאה של מסמך HTML ועיבודו לאובייקטים DOM ו-CSSOM מתחילה

מהשורה הראשונה בדף ועד השורה الأخيرة.

כאשר יתקל בקובץ js או תג script – יעצר כל תהליכי הקריאה של המסמך

עד שיושלם הקוד

שלבי טעינה דף HTML

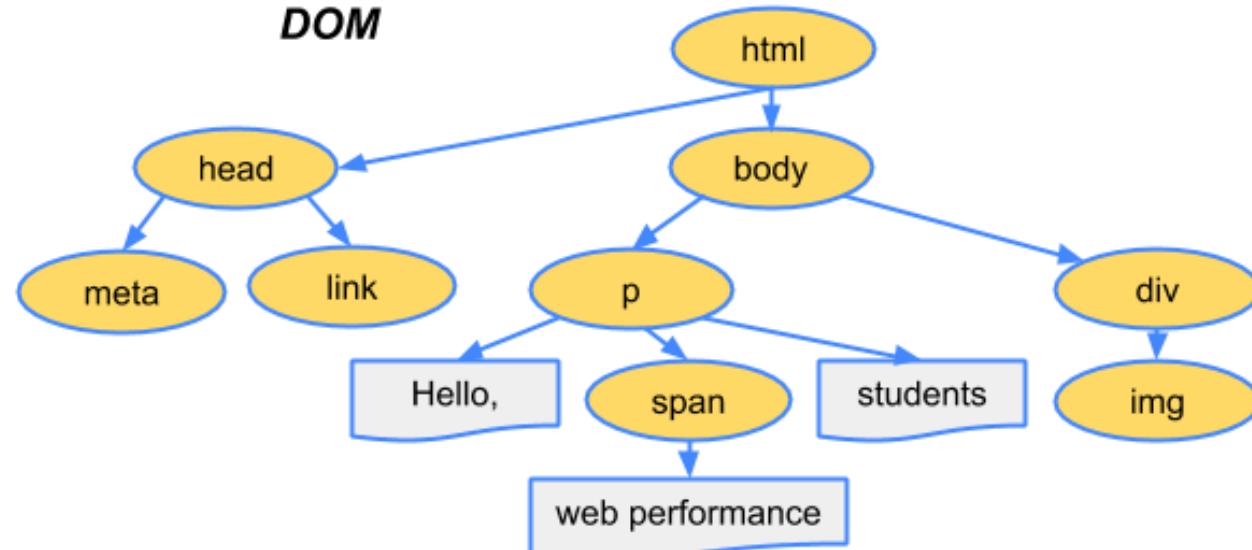


- הדף יוצר:

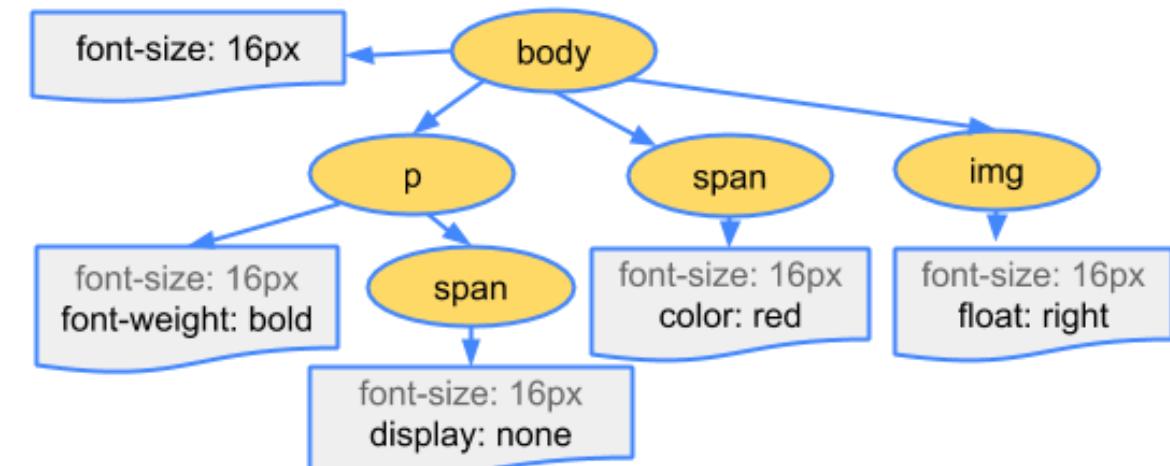
- DOM עבר האלמנטים למסך
- CSSOM עבר קבצי ה CSS

כאשר שני האובייקטים הללו יהיו מוכנים – יחל שלב עיבוד תצוגת המידע למשתמש.

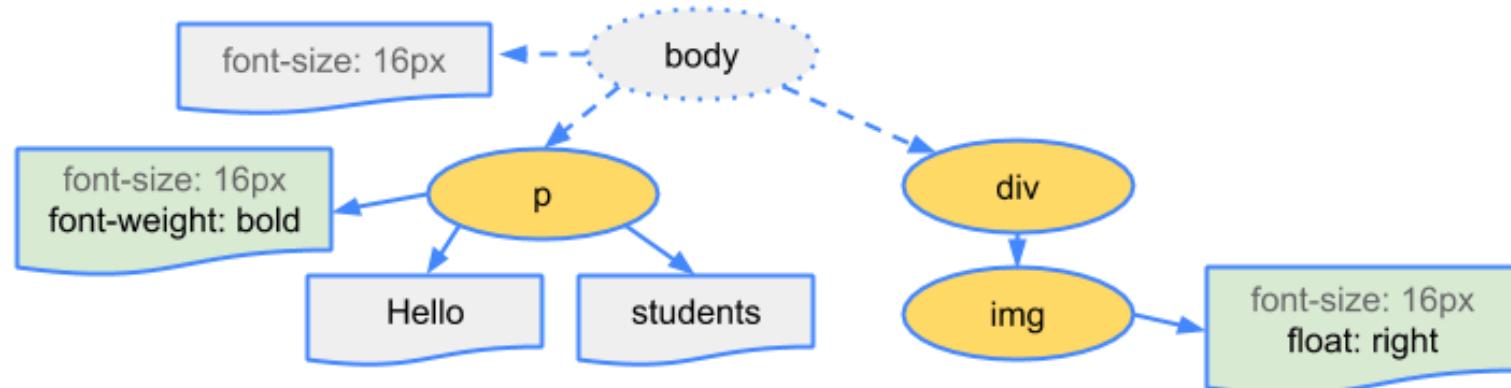
DOM



CSSOM



Render Tree



היכן נשאים ?<script>?

1. בתחום ה <head>

2. לפני הסוגר של ה <body>

```
<head>
<script>
    document.getElementById("num1").value;
</script>
</head>
<body>
    <input type="text" id="num1"/>
</body>
```

✖ Uncaught TypeError: Cannot read properties of null (reading 'value')
at [classEx1.html:9](#)

שלבי טעינה דף HTML

דף HTML יש 3 מצבים בהם הוא נמצא:

1. Loading – המסמן בשלב טעינה וקריאה מייד (יעבוד)
2. Interactive – המסמן סיום להיטען מבחינת אובייקטים DOM ו-CSSOM, בשלב זה למשל יתכן וקראנו-tag `img`, אך לא בהכרח הוצגה התמונה
3. Complete – המסמן מוקן לחלוטין כולל משאבים חזוניים כמו תמונה, סרט וכו'..
באמצעות התוכנה **defer** ניתן להעביר script חזוני שירוץ בסיום שלב interactive (עובד רק על script חזוני)

getElementById

כפי שראינו אם יש לאלמנט פו ניתן לגשת אליו על ידי id או על ידי ה id
ה id חייב להיות unique אחרת יוחזר ערך רנדומלי

```
<div id="elem">  
<div id="elem-content">Element</div>  
</div>  
  
<script>  
// get the element  
let elem = document.getElementById('elem');  
// make its background red  
elem.style.background = 'red';  
</script>
```

מייצר משתנים גלובליים לאלמנטים עם id וכאן ניתן גם לגשת לשירות Javascript

```
<head>
</head>
<body>
  <div id="someElement">getElementById works</div>
  <div id="anotherElement">This also works?!</div>

  <script>
    var someElement = document.getElementById("someElement");
    console.log(someElement.innerHTML); // getElementById works
    console.log(anotherElement.innerHTML); // This also works?!
  </script>
</body>
```

לא משתמשים ישירות ב id בעיקר על מנת להימנע מטעויות – לעוד מידע מוזמנים לקרוא בリンク :
<https://www.sitepoint.com/community/t/should-we-use-getelementbyid/279268>



id="elem-content" window['elem-content']

```
<div id="elem">
<div id="elem-content">Element</div>
</div>

<script>
// elem is a reference to DOM-element with id="elem"
elem.style.background = 'red';

// id="elem-content" has a hyphen inside, so it can't be a variable name
// ...but we can access it using square brackets: window['elem-content']
</script>
```

querySelector

querySelectorAll

מחזירה את כל האלמנטים בתוך elem שתואים ל css אלקטור.

```
<body>
  <p>1</p>
  <p>2</p>
  <p>3</p>
  <p>4</p>
  <p>5</p>
  <div class="note">first div</div>
  <div class="note">second div</div>
  <div class="alert">alert div</div>
  <span id="#test">test span</span>
  <div class="highlighted"><p>hilighted p</p></div>
<script>
  //all of the <p> elements in the document:
  let x1 = document.querySelectorAll("p");
  //class of either note or alert
  let x2 = document.querySelectorAll("div.note, div.alert");
  //inside a container whose ID is test - list of <p> elements whose immediate parent element is a <div>
  //with the class highlighted.
  let x3 = document.querySelector("#test");
  let x4 = x3.querySelectorAll("div.highlighted > p");
</script>
</body>
```

querySelector

يُחזיר את الЭلمنت الرأشون ب css لـ selector, بميـلـيم أـخـرـات يـوـحـزـرـ:

elem.querySelector**All**(nameOfCssSeector)[**0**

لكـنـ، اـمـ انـوـ رـوـصـيـمـ اـتـ الرـأـشـونـ وـرـوـصـيـمـ اـحـدـ بـلـبـدـ نـشـتـمـشـ بـ querySelectorـ بـ

شـهـوـاـ يـوـتـرـ مـهـيـرـ.

```
<html>
  <body>
    <h1>The Document Object</h1>
    <h2>The querySelector() Method</h2>
    <p>Add a background color to the first element with class="example":</p>
    <p class="example">I am a paragraph.</p>
    <p class="example">I am a paragraph.</p>
    <script>
      document.querySelector(".example").style.backgroundColor = "red";
    </script>
  </body>
</html>
```

getElementsBy*

- **getElementsByTagName** – חיפוש לפי Tag

- **getElementsByClassName** – חיפוש לפי class

שתי הפעולות מוחזירות

אוף

אפילו אם מדובר באלמנט אחד

```
<table id="table">
<tr>
  <td>Your age:</td>
<td>
<label>
<input type="radio" name="age" value="young" checked> less than 18
</label>
<label>
<input type="radio" name="age" value="mature"> from 18 to 50
</label>
<label>
<input type="radio" name="age" value="senior"> more than 60
</label>
</td>
</tr>
</table>
```

Your age: less than 18 from 18 to 50 more than 60

```
<script>
let inputs = table.getElementsByTagName('input');
for (let input of inputs) {
  console.log(input.value + ': ' + input.checked);
}
</script>
```

Young: true, mature:false, senior:false

סינון

Method	Searches by...	Can call on an element?
querySelector	CSS-selector	✓
querySelectorAll	CSS-selector	✓
getElementById	id	-
getElementsByName	name	-
getElementsByTagName	tag or '*'	✓
getElementsByClassName	class	✓

```
<div>
    Search the site:
    <input type="text">
    <input type="button" value="Search! ">
</div>
<div id="search-person">
    Search the visitors:
    <table id="age-table">
        <tr>
            <td>Age:</td>
            <td id="age-list">
                <input type="radio" name="age" value="young">less than 18
                <input type="radio" name="age" value="mature">18-50
                <input type="radio" name="age" value="senior">more than 50
            </td>
        </tr>
        <tr>
            <td>Additionally:</td>
            <td>
                <input type="text">
                <input type="text">
                <input type="text">
            </td>
        </tr>
    </table>
    <input type="button" value="Search! ">
</div>
```

חזרה

1. מצא את ה `table` בעל `id="age-table"`
2. כל האלמנטים מסוג `input` בתחום ה `table` זהה
3. ה `td` הראשון בתחום ה `table` זהה
4. כל האלמנטים מסוג `input` שבתחום ה `div` עם ה `id : search-person`

פתרונות

```
<script>

  console.log(document.querySelector("#age-table"));

  console.log(document.querySelectorAll("#age-table input"));

  console.log(document.querySelector("#age-table td:first-child"));

  console.log(document.querySelectorAll("#search-person input"));

</script>
```

תרגיל ביתה

querySelectorClassEx.html •
querySelectorClassEx_Skelton.js •

ספריית Math

ספרייה המשמשת ליצירת מספרים אקראיים, עיגול מספר דצימלי, ערך מוחלט ועוד.

```
<script>
```

```
Math.abs(-3);      //return 3  
Math.ceil(3.3);   //return 4 - rounded up  
Math.floor(3.7);  //return 3 - rounded down  
Math.round(3.3);  //return 3 - rounded to nearest  
Math.random();    //return random number between 0 to 1  
Math.min(1, -1, 3);//return -1 the minimum number  
Math.max(1, -1, 3);//return 3 the maximum number
```

```
</script>
```

הציגת האינדקס והערך של ה current selected value

```
var x = document.getElementById("mySelect");
```

```
document.getElementById("mySelect").selectedIndex = "2";
```

- `Console.log("Index: " + x[x.selectedIndex].index + " is " + x[x.selectedIndex].text);`

האינדקס בבסיס 0

[optionExample.html](#)

תרגיל ביתה 5

- צחו מחשבון ע"י 2 תיבות טקסט בהן יזין המשמש ערכי (הנิיחו שמדובר במספרים).
- בין תיבות הטקסט תהיה רשימה נגללת שימושה 2 פעולות חישוב - מינוס ופלוס (השתמשו ב`zf ==`).
- לאחר התיבות יהיה כפטור `"=` ו-`span` שמציג את התוצאה.

5	+	7	=	12
---	----------	---	----------	----

ברגע לחיצה על הכפטור, על סמך הפעולה ברשימה יבוצע החישוב.

JS2

היום בהרצאה

Conditions – if else .1

2. אופרטורים לוגיים

3. לולאות for, while

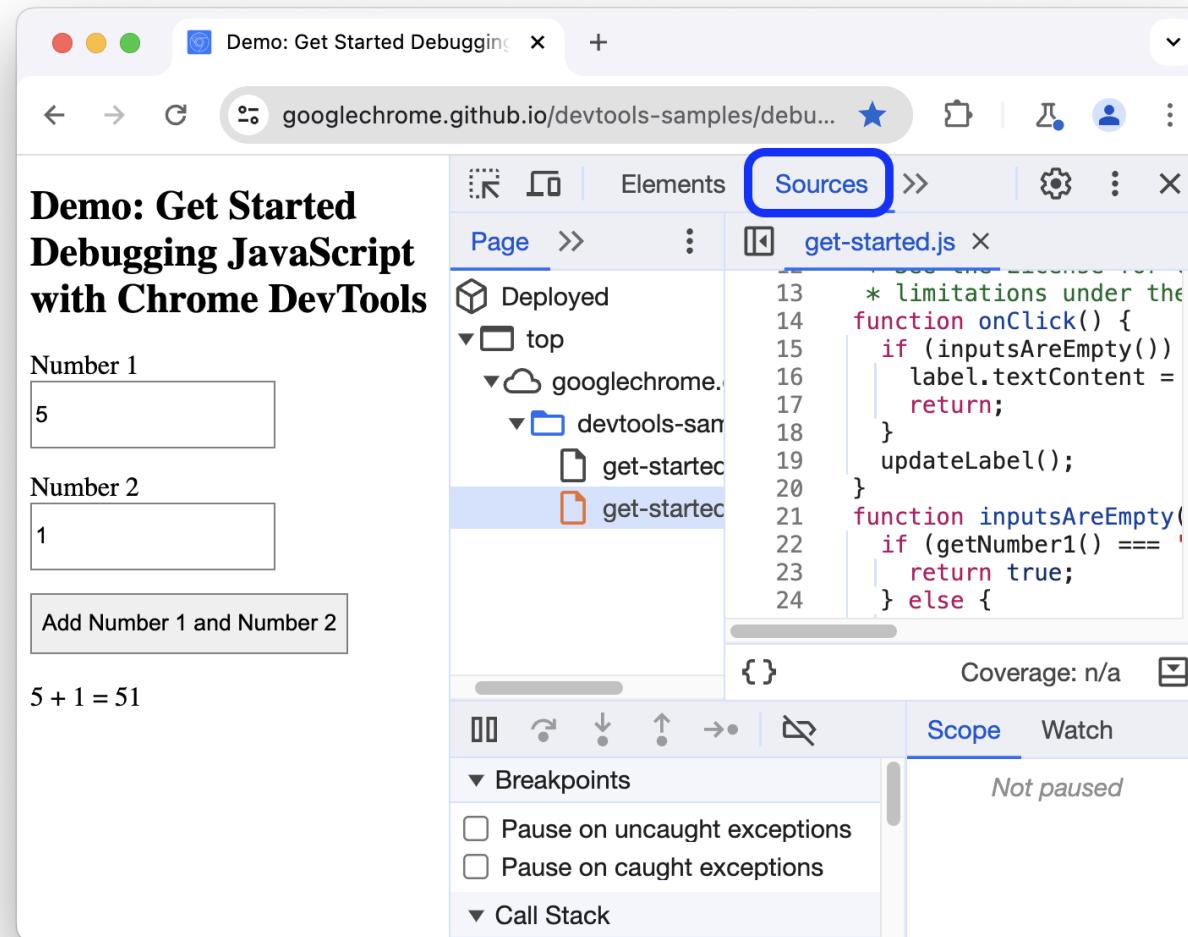
Switch .4

Dynamic body .5

Debug JavaScript

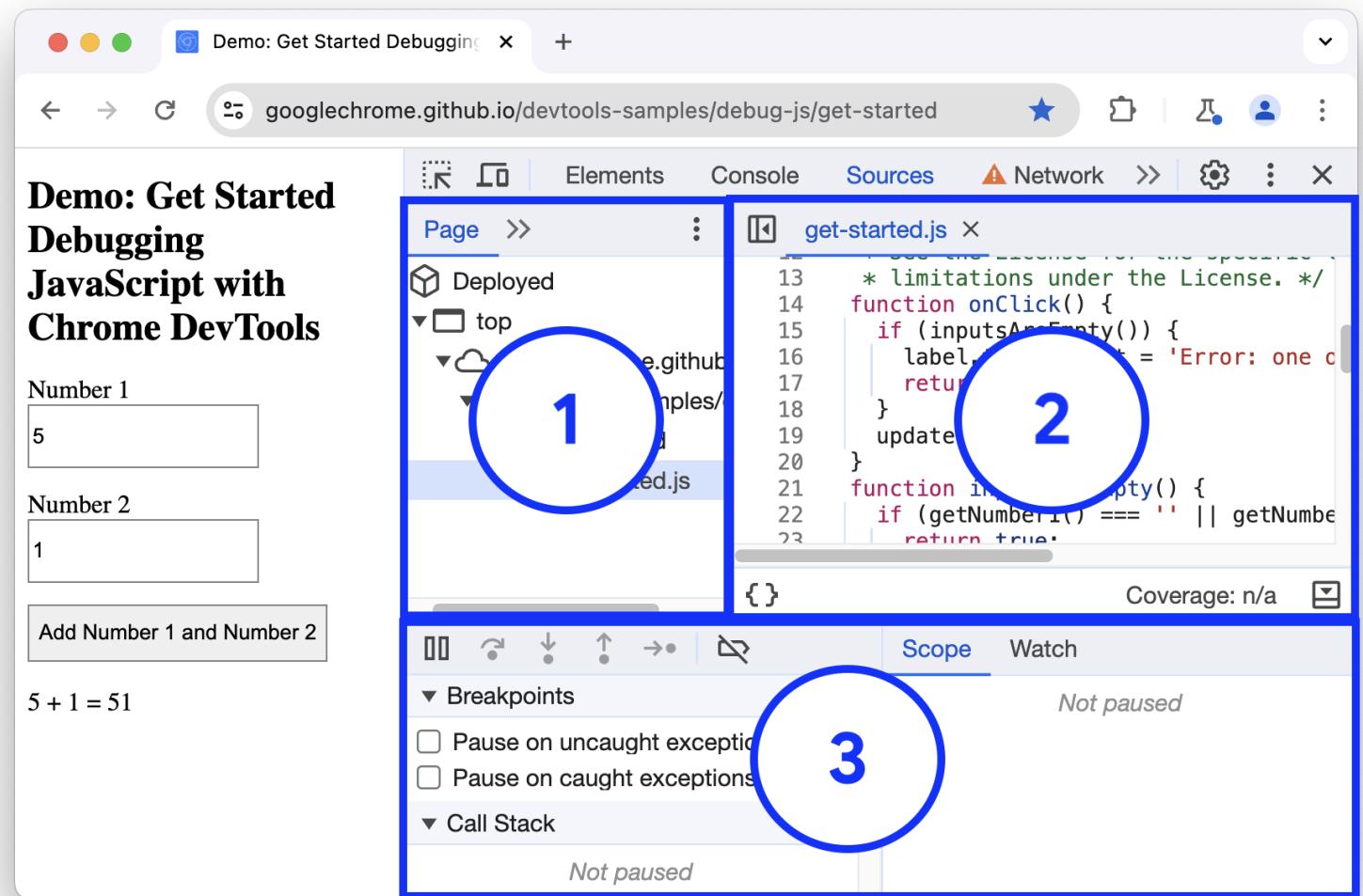
עובד על תרגיל המחשבון

Sources panel UI



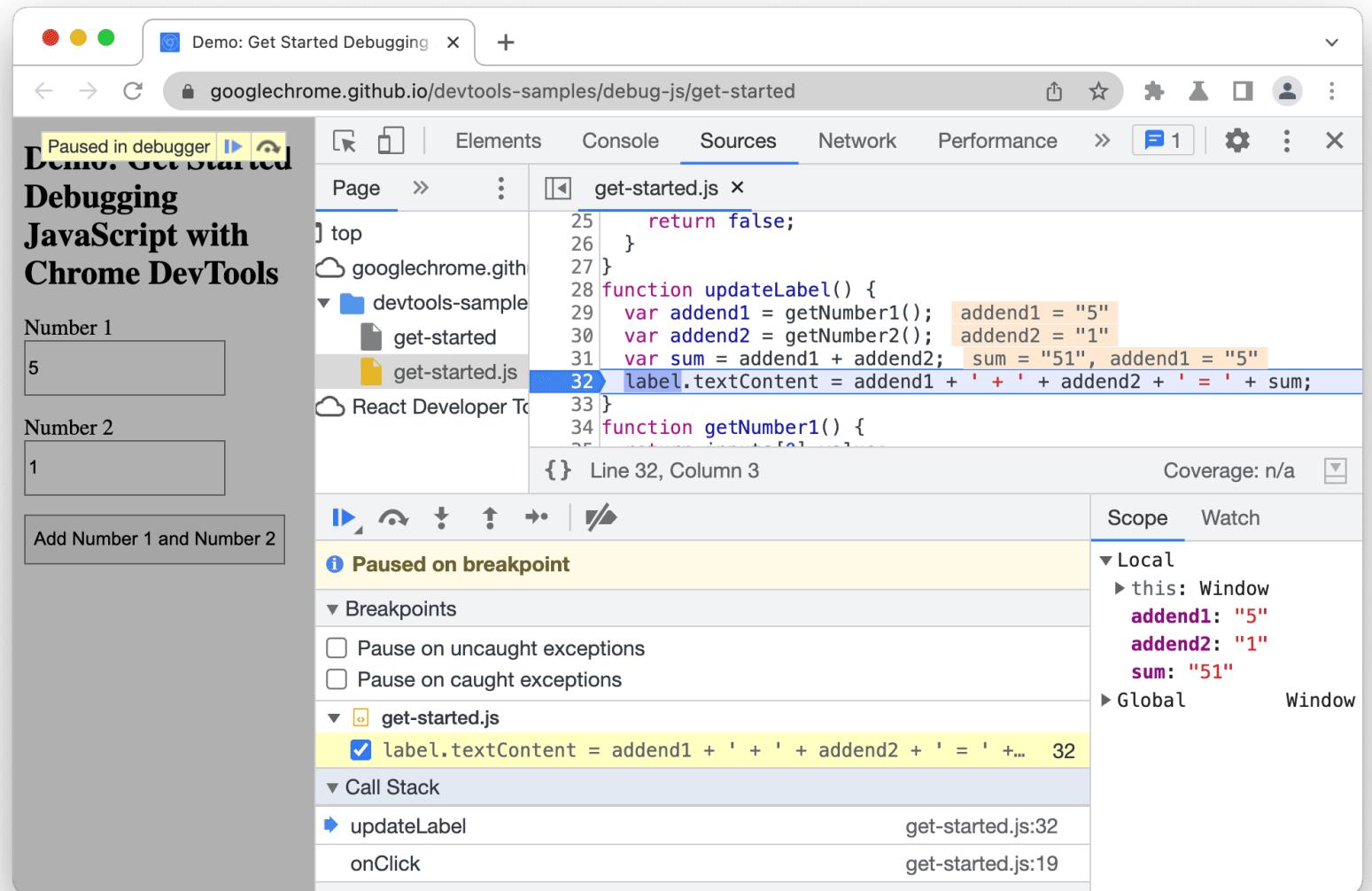
Debug JavaScript

1. File tree
2. The opened file – code editor
3. Debugger section



Debug JavaScript

1. Add breakpoint
2. Refresh the page
3. Step over, Step into
4. The scope window – show you local and global variables
5. Right click add watch



if condition

```
let year = 2022;

if (year < 2022) {
    console.log('Too early...');

}
else if (year > 2022) {
    console.log('Too late');

}
else {
    console.log('Exactly!');

}
```

? : Condition

```
let result = condition ? value1 : value2;
```

```
let age = 18;
```

```
let accessAllowed = (age > 18) ? true : false
```

```
if (age < 3) {  
    message = 'Hi, baby!';  
} else if (age < 18) {  
    message = 'Hello!';  
} else if (age < 100) {  
    message = 'Greetings!';  
} else {  
    message = 'What an unusual age!';  
}
```



```
let message = (age < 3) ? 'Hi, baby!' :  
(age < 18) ? 'Hello!' :  
(age < 100) ? 'Greetings!' :  
'What an unusual age!';  
  
console.log(message);
```

|| (OR), && (AND), ! (NOT), ??

OR

```
let hour = 12;
let isWeekend = true;

if (hour < 10 || hour > 18 || isWeekend) {
    console.log('The office is closed.'); // it is the weekend
}
```

AND

```
if (hour == 12 && minute == 30) {
    console.log('The time is 12:30');
}
```

NOT

```
console.log(!true); // false
console.log(!0); // true
```

a ?? b

if a is defined, then a

if a isn't defined, then b

```
// set height=100, if height is null or undefined  
height = height ?? 100;
```

while

```
while (condition) {  
    // code  
    // so-called "loop body"  
}
```

```
let i = 0;  
while (i < 3) { // shows 0, then 1, then 2  
    console.log(i);  
    i++;  
}
```

for

```
for (begin; condition; step) {  
    // "loop body"  
}
```

skipping loop parts

```
let i = 0; // we have i already declared and assigned  
  
for ( ; i < 3; i++) { // no need for "begin"  
    console.log(i); // 0, 1, 2  
}
```

```
let i = 0;  
  
for ( ; i < 3;) {  
    console.log(i++);  
}
```

This makes the loop
identical to `while (i < 3)`.

var vs. let

- var מאד דומה ל let ב מרבית המקרים ניתן להחליף let ב var ולהפר ולצפות מהדברים לעבוד אותו דבר.
- בעבר היה רק var, כיום משתמשים ב let
- נعمוד על ההבדלים בין השניים

“var” has no block scope

ל “var” אין block scope •

```
if (true) {  
    var test = true; // use "var" instead of "let"  
}  
console.log(test); // console.log: true
```

```
if (true) {  
    let test = true; // use "let"  
}  
  
console.log(test); // ReferenceError: test is not defined
```

“var” has no block scope in loop

```
for (var i = 0; i < 10; i++) {  
    var one = 1;  
    // ...  
}  
  
console.log(i); // 10, "i" is visible after loop, it's a global variable  
console.log(one); // 1, "one" is visible after loop, it's a global variable
```

```
function sayHi() {  
    if (true) {  
        var phrase = "Hello";  
    }  
    console.log(phrase); // works  
}  
sayHi();  
console.log(phrase); // ReferenceError: phrase is not defined
```

“var” has block scope in function

“var” tolerates redeclarations

```
var user = "Pete";  
  
var user = "John"; // this "var" does nothing (already declared)  
// ...it doesn't trigger an error  
  
console.log(user); // John
```

```
let user;  
let user; // SyntaxError: 'user' has already been declared
```

```
- let i = 0;

for (i = 0; i < 3; i++) { // use an existing variable
    console.log(i); // 0, 1, 2
}

console.log(i); // 3, visible, because declared outside of the loop

for (let i = 0; i < 3; i++) {
    console.log(i); // 0, 1, 2
}
console.log(i); // error, no such variable
```

switch

```
switch (x) {  
case 'value1': // if (x === 'value1')  
    ...  
    [break]  
  
case 'value2': // if (x === 'value2')  
    ...  
    [break]  
  
default:  
    ...  
    [break]  
}
```

```
let a = 2 + 2;

switch (a) {
case 3:
console.log('Too small');
break;
case 4:
console.log('Exactly! ');
break;
case 5:
console.log('Too big');
break;
default:
console.log("I don't know such values");
}
```

If there is **no break** then the **execution continues** with the next case **without any checks**.

```
let a = 2 + 2;

switch (a) {
case 3:
console.log('Too small');
case 4:
console.log('Exactly!');
case 5:
console.log('Too big');
default:
console.log("I don't know such values");
}
```

console.log('Exactly!');
console.log('Too big');
console.log("I don't know such values");

grouped two cases

```
let a = 3;

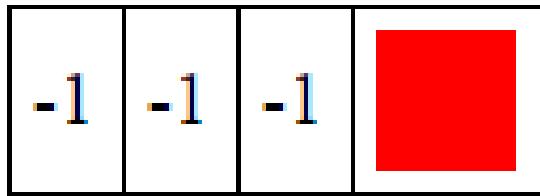
switch (a) {
case 4:
console.log('Right!');
break;

case 3: // (*) grouped two cases
case 5:
console.log('Wrong!');
console.log("Why don't you take a math class?");
break;

default:
console.log('The result is strange. Really.');
}
```

both 3 and 5 show the same message.

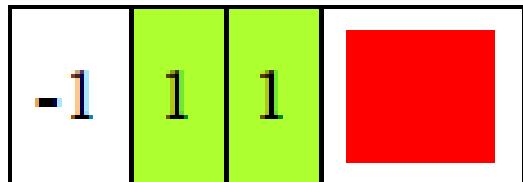
תרגיל ביתה מסכם js



עליכם ליצור 4 תאים בשורה.

ב-3 תאים יופיע המספר 1.-.

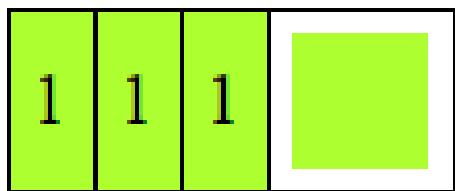
לחיצה על תא עם מספר (-1), המספר הופך ל- (1) ומשתנה לצבע ירוק,
לחיצה נוספת מוחזירה למספר (-1) והצבע נעלם.



בתא הרביעי יופיע vid שצבעו תלוי בשורת התאים:

3 תאים ירוקים (מספר 1) =ה vid צבע ירוק

כל מצב אחר = vid בצבע אדום.



jsBasicsclassEx_greenSquereToRed-SolNoArray.HTML

שינוי ה-DOM

Javascript מספקת לנו את היכולת לשנות את תוכן הדף כרצוננו. הצורה הכי "פешטה" לשינוי התוכן של אלמנט היא באמצעות תכונת `innerHTML`. התכונה מקבלת מחרוזת, אשר תוצג על המסך לפי קוד ה-HTML שרשום בה.

```
<body>
  <div id="myHtml"></div>

  <script>
    document.getElementById("myHtml").innerHTML = "<h1>Hello World</h1>";
  </script>
</body>
```



Hello World

DHTML

באמצעות תכונת `innerHTML` ניתן לדרס את התוכן הנוכחי, להוסיף לו קויים או למחוק את התוכן:

```
<body>
  <div id="myHtml"><h1>Hello World</h1></div>

  <script>
    הוספה תוכן לקוים //
    document.getElementById("myHtml").innerHTML += "<h1>adding header</h1>";
    דרישת הקויים //
    document.getElementById("myHtml").innerHTML = "<h1>switching header</h1>";
    מזיקת התוכן //
    document.getElementById("myHtml").innerHTML = "";
  </script>
</body>
```

איך נבנה את הדף הבא בלחיצת כפתור?

This is a paragraph: 0



This is a paragraph: 1

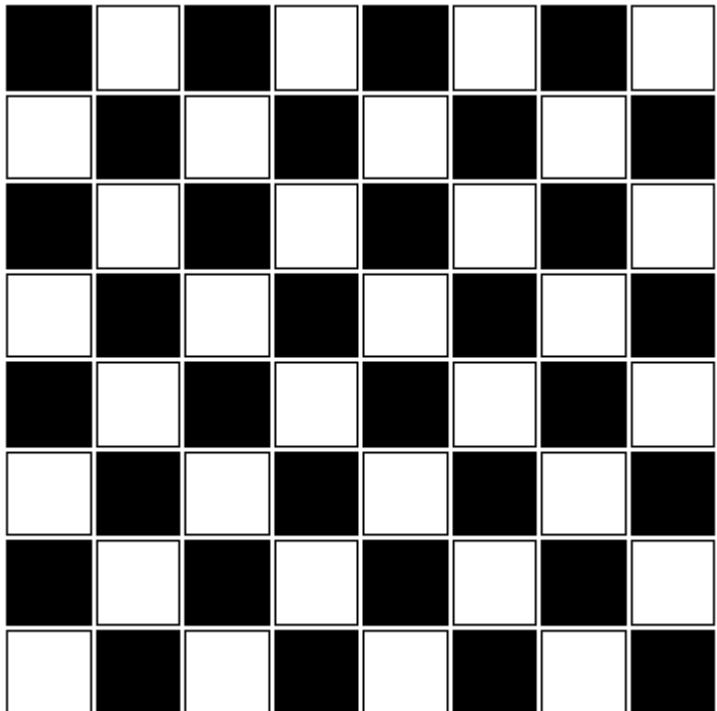


This is a paragraph: 2



```
function showImages() {  
  
    htmlStr = "";  
  
    for (var i = 0; i < 100; i++) {  
        htmlStr += "<div>";  
        htmlStr += "<p>This is a paragraph: " + i + "</p>";  
        htmlStr += '';  
        htmlStr += "</div>";  
    }  
    document.getElementById("ph").innerHTML = htmlStr;
```

תרגיל ביתה



שלב ראשון:

צור את לוח הדמקה הבא באופן דינמי

שלב שני:

לחיצה על תא כלשהו תנסה את צבעו לירוק

*הפתרון שלי הוא באמצעות table מוגזמים
לעשות בדרך אחרת.

שלד לתרגיל כתה דמקה:

```
<style>
  td {
    border: 1px solid #000;
    width: 30px;
    height: 30px;
  }

  .tdBlack {
    background-color: black;
  }

  .tdWhite {
    background-color: white;
  }
</style>
```

```
<body>
  <div id="ph"></div>
</body>
```

```
function colorChanger(el) {
  el.style.backgroundColor = '#007d00';
}
```

פונקציות

function declaration

- הצורה שכבר ראיינו:

```
function sayHi()  
{  
  
    console.log("Hello");  
}
```

function parameters

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

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

let x = myFunction(4, 3);
```

function expression

```
let sayHi = function () {  
    console.log("Hello");  
};
```

- CAN ANO ROAIM MASHTNAH BESHM iHays SHAURK SHLO HOA PONKZIA.
- SHIMO LIB: AIN SHM L PONKZIA, ANO SHMIM AT H PONKZIA CURK SHL MASHTNAH.
- AM NERCHA L KROA L PONKZIA NKRA () iHays UM SGORIIM.
- AM NDFOIS AT MASHTNAH iHays NKBL AT H TAKSUT SHL H PONKZIA.

copy to another variable

```
function sayHi() { // (1) create  
  console.log("Hello");  
}
```

```
let func = sayHi; // (2) copy  
  
func(); // Hello // (3) run the copy (it works!)  
sayHi(); // Hello // this still works too (why wouldn't it)
```

```
let sayHi = function () { // (1) create  
  console.log("Hello");  
};
```

```
let func = sayHi;  
// ...
```

function declaration

בצורה זו שהפונקציה מוגדרת – ניתן לקרוא לפונקציה לפני ההגדרה שלה.

```
sayHi("John"); // Hello, John

function sayHi(name) {
  console.log(`Hello,
${name}`);
}
```

function expression

בצורה זו שהפונקציה היא משתנה, ניתן לקרוא לה רק לאחר הגדרת המשתנה.

```
sayHi("John"); // error! ✘  
  
let sayHi = function (name) { //  
(*) no magic any more  
  console.log(`Hello, ${name}`);  
};
```

arrow function syntax

1. `(param1, paramN) => expression`
2. `(param) => expression`
3. `() => expression`
5. `(param1, paramN) => { statements }`
6. `param => { statements }`

1. (param1, paramN) => expression

```
let sum = (a, b) => a + b;  
console.log(sum(1, 2)); // 3
```

/* This arrow function is a shorter form of:

```
let sum = function(a, b) {  
    return a + b;  
};  
*/
```

2. param => expression

- אם קיים פרמטר אחד ניתן להוריד את הסוגרים ולכתוב כך:

```
let double = n => n * 2;  
console.log(double(3)); // 6
```

3. () => expression

- אם אין פרמטרים אפשר לכתוב סוגרים ריקים

```
let sayHi = () => console.log("Hello!");  
  
sayHi();
```

4. param => { statements }

```
let func = (arg1, arg2, ..., argN) => expression;
```



```
let func = function (arg1, arg2, ..., argN) {
    return expression;
};
```

array

אוסף אלמנטים מסודר האיבר הראשון במקום ה 0

צורה:

```
let arr = [];
```

או

```
let arr = new Array();
```

צורה והשמה:

```
let fruits = ["Apple", "Orange", "Plum"];
```

```
let fruits = ["Apple", "Orange", "Plum"];
console.log(fruits[0]); // Apple
console.log(fruits[1]); // Orange
console.log(fruits[2]); // Plum
```

Replace an element:

```
fruits[2] = 'Pear'; // now ["Apple", "Orange", "Pear"]
```

Add a new one

```
fruits[3] = 'Lemon'; // now ["Apple", "Orange", "Pear", "Lemon"]
```

Array length

```
let fruits = ["Apple", "Orange", "Plum"];  
console.log(fruits.length); // 3
```

Show the whole array

```
let fruits = ["Apple", "Orange", "Plum";  
console.log(fruits); // Apple,Orange,Plum
```

An array can store elements of any type.

```
// mix of values
let arr = ['Apple', 1, true, function () {
console.log('hello'); }];

// get the number at index 1 and then show its name
console.log(arr[1]); // 1

// get the function at index 3 and run it
arr[3](); // hello
```

Access last element

```
- let fruits = ["Apple", "Orange", "Plum"];
```

```
console.log(fruits[fruits.length - 1]); // Plum
```

Shorter syntax

```
let fruits = ["Apple", "Orange", "Plum"];
```

```
// same as fruits[fruits.length-1]
console.log(fruits.at(-1)); // Plum
```

array methods

- **pop** •

- מוציא את האיבר האחרון מהמערך ומחזיר אותו

- **push** •

- מוסיף את האיבר הנתון לסוף המערך

- **shift** •

- מוציא את האיבר הראשון מהמערך ומחזיר אותו

- **unshift** •

- מוסיף את האיבר הנתון לתחילת המערך

push, pop

```
let fruits = ["Apple", "Orange", "Pear"];
console.log(fruits.pop()); // remove "Pear" and console.log
return it
console.log(fruits); // Apple, Orange
```

```
let fruits = ["Apple", "Orange"];
fruits.push("Pear"); //is equal to fruits[fruits.length] =
.....
console.log(fruits); // Apple, Orange, Pear
```

shift, unshift

```
let fruits = ["Apple", "Orange", "Pear"];
console.log(fruits.shift()); // remove Apple and console.log it
console.log(fruits); // Orange, Pear
```

```
let fruits = ["Orange", "Pear"];
fruits.unshift('Apple');
console.log(fruits); // Apple, Orange, Pear
```

array methods

```
var arr1 = [1, "hello", "hi"];

arr1.unshift("11");      //add element to the beginning of the array
arr1.push("11");        //add element to the end of the array

arr1.shift();            //remove the first element from the array
arr1.pop();              //remove the last element from the array

arr1.length;             //the number of elements in the array

arr1[0] = "change";     //set value of the first element in the array or add it if not exists
```

air יראה המערך לאחר כל הפעולות?

[*"change"*, *"hello"*, *"hi"*]

loop over array

Older way “**for**”

```
let arr = ["Apple", "Orange", "Pear"];
for (let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

another form of loop, “**for..of**”

```
let fruits = ["Apple", "Orange", "Plum"];
// iterates over array elements
for (let fruit of fruits) {
    console.log(fruit);
}
```

loop over array

*another form of loop, “**for..in**”*

```
let fruits = ["Apple", "Orange", "Plum"];
for (let i in fruits) {
    console.log(i); // 0, 1, 2
    console.log(fruits[i]); // "Apple", "Orange", "Plum"
}
```

array length

```
let fruits = [];
fruits[123] = "Apple";

console.log(fruits.length); // 124
```

```
let arr = [1, 2, 3, 4, 5];

arr.length = 2; // truncate to 2 elements
console.log(arr); // [1, 2]
```

```
arr.length = 5; // return length back
console.log(arr[3]); // undefined: the values do not return
```

תרגיל ביתה

פלט:

"row 0"

 " 1"

 " 2"

 " 1"

 " 24"

"row 1"

...

...

...

כתב פונקציה המתקבלת את מערך הדו מימדי הבא של מספרים

```
var a = [[1, 2, 1, 24,  
         [8, 11, 9, 4],  
         [7, 0, 7, 27],  
         [7, 4, 28, 14],  
         [3, 10, 26, 7]]];
```

ומשתמשת בלולאה מקווננת על מנת להדפיס את אברי מערך לפי שורות

תרגיל ביתה

- כתוב פונקציה שמקבלת מערך חד מימדי של מספרים ומחרוזות ומדפיסה את האבר שופיע הכי הרבה פעמים וכמה פעמים הוא מופיע .

• **קלט:**

```
var arr1 = [3, 'a', 'a', 'a', 2, 3, 'a', 3, 'a', 2, 4, 9, 3];
```

•

•

פלט:

a (5 times) •

Array forEach

```
const array1 = ['a', 'b', 'c'];
```

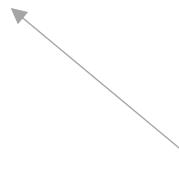
```
array1.forEach((element) => console.log(element));
```

```
// Expected output: "a"
```

```
// Expected output: "b"
```

```
// Expected output: "c"
```

מבצע את הפעולה על כל אחד מהאלמנטים



Converting a for loop to forEach

```
let items = ["item1", "item2", "item3"];
let copyItems = [];
```

```
// before
for (let i = 0; i < items.length; i++) {
  copyItems.push(items[i]);
}
```

```
// after
items.forEach((item) => {
  copyItems.push(item);
});
```

תרגיל ביתה

- פותח את ה skelton בקובץ jewishManDOMClassEx.zip הוסיף את האירועים הרשומים בדף מועתק גם כאן לנוחיותכם:
 1. **onclick** on the **button** - Copy the text to the right div
 2. **onmouseover** on the smiley - Add the image to the right div
 3. **onclick** on the **jewish smiley** – change all the images in the right div to the jewish smiley
 4. **ondblclick** on the **eraser** - remove all the content from the right div

Copy Text



1. onclick on the button - Copy the text to the right div
2. onmouseover on the smiley - Add the image to the right div
3. onclick on the jewish smiley - change all the images in the right div to the jewish smiley
4. ondblclick on the eraser - remove all the content from the right div

JS3

Document Object Model

DOM

- עבור כל דף HTML שהדף טוען הוא יוצר לפני אובייקט מודל הנקרא **Document Object Model - DOM**
- דף מתייחס לדף HTML כאוסף סדור של – **תגיות, שדות וטקסטים** בDOM כל אובייקט צזה נקרא `node`.
האלמנטים מוחזקים בעץ של `nodes`.
- Javascript יכולה לשנות את תוכן הדף על ידי גישה לאובייקטים שלו.
הDOM מאפשר גישה לאובייקטים ובכך מאפשר לחפש, לשנות ולהסרם **אלמנטים**.

fonkzioth DOM

- `setAttribute`
- -----
- `createElement`
- `appendChild`
- -----
- `insertBefore`
- `removeChild`

setAttribute

element.setAttribute(name, value);

```
document.querySelector("div").setAttribute("class", "democlass"); •
```

```
let x = document.querySelector("div").getAttribute("class"); •
```

```
document.querySelector("div").removeAttribute("class"); •
```

setAttribute

- `myAnchor.setAttribute("href", "https://www.mysite.com");`
- `document.getElementById("game-container").setAttribute("class", "game-con");`
- `gameHeader.setAttribute("id", "game-header");`
- `checkbox.setAttribute('checked', 'checked');`
- `myImg.setAttribute('src', 'images/dog.jpg');`

setAttribute

```
• function change() {  
•     document.querySelector("input").setAttribute("type", "button");  
• }  
  
• <body>  
•     <div id="div1">This is static div.</div>  
•     <input onclick="change()" value="Click me" />  
• </body>
```

createElement method

- `let element = document.createElement(-----insert tag name---);`
- Tag name - `div, span, p, h1.....`
- `const newDiv = document.createElement("div");`
- `newDiv.innerHTML = " Good Morning !";`
- `document.body.appendChild(newDiv);`

תרגיל חימום

עם טיענת הדף צרו 3 DIVS, כאשר בכל DIV תופיע כותרת לפי התמונה מטה וצבע ה-DIV יהיה לפי התמונה:



*בתוך ה-body לא יהיה אף אלמנט בצורה סטטית,
אפשר לשים script ב-body

פתרונות

```
<script>
    const colors = ["green", "orange", "red"];

    for (let i = 0; i < 3; i++) {
        let div = document.createElement("div");
        let h = document.createElement("h1");
        div.append(h);
        h.innerHTML = "header " + (i + 1);
        div.style.backgroundColor = colors[i];

        document.body.append(div);
    }
</script>
```

הוספת אירוח בצורה דינמית

addEventListener .1

onclick .2

setAttributes .3

שימוש ב addEventListener

- `let el = document.getElementById("element");`
- `el.addEventListener("click", function () { console.log("Hi");});`

addEventListener

```
element.addEventListener("click", myFunction);
```

```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Hello World";  
}
```

שימוש קצר יותר:

```
element.addEventListener("click", function() {  
    document.getElementById("demo").innerHTML = "Hello World";  
});
```

addEventListener

ניתן להוסיף כמה אירועים לאותו אלמנט

```
<body>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    const element = document.getElementById("myBtn");
    element.addEventListener("click", myFunction1);
    element.addEventListener("click", myFunction2);
    function myFunction1() {
      document.getElementById("demo").innerHTML += "First function was executed! "
    }
    function myFunction2() {
      document.getElementById("demo").innerHTML += "Second function was executed!"
    }
  </script>
</body>
```

- <body>
 <button id="myBtn">Try it</button>
 <p id="demo"></p>
 <script>
 const element = document.getElementById("myBtn");
 element.addEventListener("mouseover", myFunction);
 element.addEventListener("click", mySecondFunction);
 element.addEventListener("mouseout", myThirdFunction);
 function myFunction() {
 document.getElementById("demo").innerHTML += "Moused over!
"
 }
 function mySecondFunction() {
 document.getElementById("demo").innerHTML += "Clicked!
"
 }
 function myThirdFunction() {
 document.getElementById("demo").innerHTML += "Moused out!
"
 }
 </script>
 </body>

נition להוסיף אירועים שונים addEventListener

```
<body>
```

```
  <h1>The Element Object</h1>
```

```
  <h2>The addEventListener() Method</h2>
```

```
  <p>How to pass parameter values with addEventListener().</p>
```

```
  <p>Click the button to perform a calculation.</p>
```

```
  <button id="myBtn">Try it</button>
```

```
  <p id="demo"></p>
```

```
  <script>
```

```
    let p1 = 5;
```

```
    let p2 = 7;
```

```
    document.getElementById("myBtn").addEventListener("click", function () {
      myFunction(p1, p2);
    });
  
```

```
  function myFunction(a, b) {
```

```
    document.getElementById("demo").innerHTML = a * b;
```

```
  }
```

```
</script>
```

```
</body>
```

העברת פרמטרים יכולת לבצע רק דרך

פונקציה פנימית שמקבלת פרמטרים

הfonקצייה מבירה את this
addEventListener

```
<body>
  <button id="myBtn">Try it</button>
  <script>
    document.getElementById("myBtn").addEventListener("click", function () {
      this.style.backgroundColor = "red";
    });
  </script>
</body>
```

onclick

- `let el = document.getElementById("element");`
- `el.onclick = function x() { console.log('Hi'); }`

onclick

```
let el = document.getElementById("element");
• el.onclick = function() { console.log('Hi'); }
```

IX

```
• el.onclick = x;
function x() { console.log('Hi'); }
```

onclick

```
let el = document.getElementById("element"); •
```

מקבל תמיד את ה event

- el.onclick = x;

```
function x() { console.log( event.currentTarget.innerText); }
```

מכיר את this

- el.onclick = x;

```
function x() { console.log( this.innerText); }
```

onclick

```
document.getElementById("para").onclick = fun;  
  
function fun() {  
    document.getElementById("para").innerHTML = "Welcome to Client side class";  
    document.getElementById("para").style.color = "blue";  
    document.getElementById("para").style.backgroundColor = "yellow";  
    document.getElementById("para").style.fontSize = "25px";  
    document.getElementById("para").style.border = "4px solid red";  
}
```

onclick

```
<body>
<button id="myBtn">Try it</button>
<p id="para"></p>
<script>
document.getElementById("myBtn").onclick = fun;

function fun() {
    document.getElementById("para").innerHTML = "Welcome to Client side class";
    document.getElementById("para").style.color = "blue";
    document.getElementById("para").style.backgroundColor = "yellow";
    document.getElementById("para").style.fontSize = "25px";
    document.getElementById("para").style.border = "4px solid red";
}
</script>
</body>
```

onclick

- In previous example - better use **this**

```
<body>
<button id="myBtn">Try it</button>
<p id="para"></p>
<script>
    document.getElementById("myBtn").onclick = function () { fun(this) };

    function fun(elem) {
        elem.innerHTML = "Welcome to Client side class";
        elem.style.color = "blue";
        elem.style.backgroundColor = "yellow";
        elem.style.fontSize = "25px";
        elem.style.border = "4px solid red";
    }
</script>
</body>
```

תרגיל ביתה חלק ב

הוסף אירוע קליק לכל div אשר לחיצה תשנה את צבעו לצהוב

הוסף אירוע דאבל קליק לכל div אשר לחיצה עליו תמחק אותו השתמש
בפונקציה (`remove()`) על האלמנט עצמו להסרתו



פתרונות

```
<body>
```

```
<script>
    const colors = ["green", "orange", "red"];

    for (let i = 0; i < 3; i++) {
        let div = document.createElement("div");
        div.onclick = function () { this.style.backgroundColor = 'yellow' };
        //div.addEventListener("click", function () { this.style.backgroundColor = 'yellow' });
        //div.setAttribute("onclick", "this.style.backgroundColor = 'yellow'");
        div.ondblclick = function () { this.remove() };
        let h = document.createElement("h1");
        div.append(h);
        h.innerHTML = "header " + (i + 1);
        div.style.backgroundColor = colors[i];

        document.body.append(div);
    }
</script>
</body>
```

הציג בכיתה

כתב את כל הֆונקציות שנמצאות עבור אירוע click

```
• <!DOCTYPE html>
• <html>
•   <head>
•     <style type="text/css">
•       #specialDiv {
•         color: red;
•       }
•
•       .colorBlue {
•         color: blue;
•       }
•
•       .buttons {
•         padding: 10px;
•         border: 1px solid grey;
•         border-radius: 4px;
•       }
•     </style>
•     <script>
•       let counter = 0;
•       function getCounter() {
•         counter += 1;
•         updateCounter();
•         return " " + counter;
•       }
•       function updateCounter() {
•         let counterElement = document.getElementById("counter");
•         counterElement.innerText = " " + counter;
•       }
•     </script>
•   </head>
•   <body>
•     <div class="buttons">
•       <input type="button" onclick="addFirstElement();" value="Add element" />
•       <input type="button" onclick="addElementToSpecialDiv();" value="Add element within div" />
•       <input type="button" onclick="addLink();" value="Add a link" />
•       <input type="button" onclick="innerTextVsinnerHTML();" value="innerText VS innerHTML" />
•       <input type="button" onclick="createElementWithCSSClass();" value="Create with CSS class" />
•       <input type="button" onclick="createElementWithStyle();" value="Create with style" />
•       <input type="button" onclick="insertElement();" value="Insert (before) element" />
•       <input type="button" onclick="createBullets();" value="Create bullets" />
•       <input type="button" onclick="addButtons();" value="Add a lot of buttons" />
•       <br><br>
•       <div>
•         Elements added: <span id="counter">0</span>
•       </div>
•       <br>
•     </div>
•     <div id="specialDiv"></div>
•   </body>
• </html>
```

- 1) [addFirstElement](#) - add <p> to body
- 2) [addElementToSpecialDiv](#) - insert <p> to <div id="specialDiv"></div>
- 3) [addLink](#) - add href
- 4) [innerTextVsInnerHTML](#) - add text "This text is with a bolded part." to div
innerHTML / innerText
- 5) [createElementWithCSSClass](#) - add Css class to <p>
- 6) [createElementWithStyle](#) - add style inline to <p>
- 7) [insertElement](#) - insert <p> to <div id="specialDiv"></div> to show first
- 8) [createBullets](#) - create ul..li
- 9) [addButtons](#) - add 100 buttons, click will remove it -
בסוף הרצאה נסיף עלי צעריה

תרגיל ביתה

- **צורך שני תאים טקסט:**
 1. לחיצה על התא הראשון הוא יאפשר לפתיחת
 2. לחיצה על התא השני תחזיר אותו להיות טקסט.
- על ה body להיות ריק – יש ליצור באופן דינמי את האלמנטים באמצעות פונקציות של ה DOM

[change to button](#)

[change to text](#)

[change to button](#)

[change to text](#)

• פתרון DOM_Element.html

תרגיל ביתה מתקדם Ex

Store

Click to Add



Basket

Double click to remove

1. ב click על פרי מהחנות הוסף אותו לסל
2. ב dbclick על פרי מהסל הסר אותו
3. Skelton נמצא בmodel

Objects in JS

Objects

- אובייקט מיוצג על ידי סגירים מסולסלים ובתוכו properties
- key : value צמד של Property
- string הוא Key
- Value יכול להיות כל ערך

```
let user = new Object(); // "object constructor" syntax  
let user = {}; // "object literal" syntax
```

```
let user = {  
    name: "John",  
    age: 30  
};
```

```
console.log(user.name);  
console.log(user.age);
```

```
user.isAdmin = true;
```

```
delete user.age;
```

```
//We can also use multiword property names, but then they must be quoted:  
let user = {  
    name: "John",  
    age: 30,  
    "likes birds": true // multiword property name must be quoted  
};
```

```
// this would give a syntax error  
console.log(user.likes birds); // true
```

```
console.log(user["likes birds"]); // true
```

```
let user = {};  
  
// set  
user["likes birds"] = true;  
  
// get  
console.log(user["likes birds"]); // true  
  
// delete  
delete user["likes birds"];  
  
let key = "likes birds";  
  
// same as user["likes birds"] = true;  
user[key] = true; //OR user.key = true;
```

for...in Loop

- עובר בלולאה על המפתחות של האובייקט

```
for (let key in object) {  
    // executes the body for each key among object properties  
    // key, object[key]  
}
```

תרגיל ביתה

- כתוב פונקציה המקבלת אובייקט שערci המפתחות שלו הם מספרים
ומחזירה את סכום המספרים

```
let grades = {  
    John: 100,  
    Ann: 60,  
    Pete:85  
};
```

- למשל עבור האובייקט:

- יחזיר הסכום 245

פתרונות

```
let grades = {
    John: 100,
    Ann: 60,
    Pete: 85
};

sumProp(grades);

function sumProp(obj) {
    let sum = 0;
    for (let key in obj) {
        sum += obj[key];
    }
}

console.log(sum); // 245
}
```

יצירת אובייקט

- 3 דרכים ליצור אובייקט

1. **ישירות** - על ידי הצהרה אז השמה של ה properties כפי שראינו
2. **динמית** – על ידי שימוש ב `new Object()`
3. **בנאי** - "קונסטרקטור" וממנו יצירה מופעים

.1. הוצאה והשמה

- `let personObj = {
 • firstname: "John",
 • lastname: "Doe",
 • age: 50,
 • tellYourage: function () {
 • console.log("The age is " + this.age);
 • },
 • tellSomething: function (something) {
 • console.log(something);
 • }
 • }
• personObj.tellYourage();
• personObj.tellSomething("Life is good!")`

-
- `personObj.newField = "some data";`
 - `personObj.fullName = personObj.firstname + " " + personObj.lastname;`
 - `personObj.myfunction = function () {`
 - `console.log(this.fullName + " is " + this.age);`
 - `}`
 - `personObj.myfunction();`

2. דינמית - יצירת מופע שימוש ב new Object

- personObj= `new Object();`
- personObj.firstname = "John";
- personObj.age = 50;
- personObj.tellYourage = `function () {`
- `console.log("This age is " + this.age);`
- `}`
- `// You can call then tellYourage function as following`
- `personObj.tellYourage();`

פונקציה מוגדרת מראש

- `function tellYourage() {`
- `console.log("The age is" + this.age);`
- `}`
- `personObj.tellYourage = tellYourage;`

שיםו לב להבדל:

- מציב בשדה פונקציה //

- personObj.tellYourage = tellYourage;

- מציב בשדה ערך מוחזר של פונקציה //

- personObj.tellYourage = tellYourage();

3. "קונסטרקטור"

- // תבנית ליצירת אובייקט מتبצע על ידי function ניתן להסתמך על זה כעל constructor
- `function Person(firstname, lastname, age)`
- {
 - `this.firstname = firstname;`
 - `this.lastname = lastname;`
 - `this.age = age;`
 - `this.tellYourage = function () {`
 - `console.log("This age is " + this.age);`
 - `}`
- }

-
- **מופעים של האובייקט//**
 - myFather = new Person("John", "Doe", 50);
 - myMother = new Person("Sally", "Rally", 48);
 - **ניתן להוסיף תכונות ופונקציות//**
 - myFather.newField = "some data";
 - myFather.fullName = myFather.firstname + " " + myFather.lastname;
 - myFather.myfunction = function () {
 - console.log(this.fullName + " is " + this.age);
 - }

Prototype

- `function MyObject(name, size) {
 • this.name = name;
 • this.size = size;
• }
• // Add a function to the prototype
• MyObject.prototype.tellSize = function() {
 • return "size of " + this.name + " is " + this.size;
• }
• צירת מופע לאובייקט וקריאה לפונקציה שלו //
• let myObj = new MyObject("Sang", "30 inches");
• Console.log(myObj.tellSize());`

תרגיל ביתה

```
1,blue  
1,red  
1,green  
2,blue  
2,red  
2,green  
3,blue  
3,red  
3,green  
4,blue  
4,red  
4,green  
5,blue  
5,red  
5,green  
▶ Array(15)
```

```
let nums = [1,2,3,4,5] •  
let colors = ["blue", "red", "green"] •
```

- 1) צור מערך של אובייקטים המציגים קלפים (השתמש ב `new object` או יצר קונסטרקטור `Card`) כל אובייקט מכיל – מספר וצבע `obj.color` `obj.type`
- 1) על המערך להכיל את כל הקלפים האפשריים, כלומר כל האפשרויות של הצירוף מספר-צבע.
- 1) הדפסו ל `log` את כל אברי המערך

המשר תרגיל ביתה

בנו פונקציה כללית שמקבלת מערך ועושה shuffle – חפשו ברשת פונקציה כזו הבינו אותה והעתיקו אליכם.

בצעו shuffle על המערך והדפינו ל `console.log`

פתרונות

```
let nums = [1,2,3,4,5]
let colors = ["blue", "red", "green"]
let arr = [];

for (let i in nums) {
  for (let j in colors) {
    let obj = new Object();
    obj.type = nums[i];
    obj.color = colors[j];
    arr.push(obj);
    console.log(obj.type + "," + obj.color);
  }
}

function shuffle(a) {
  let j;
  for (let i = a.length - 1; i > 0; i--) {
    j = Math.floor(Math.random() * i);
    x = a[i];
    a[i] = a[j];
    a[j] = x;
  }
  return a;
}
arr = shuffle(arr);
console.log(arr);
```

By Val By Ref in javascript

Primitive & Non-Primitive Data Types

Value Types/ Primitive

Number
String
Boolean
Symbol
Undefined
Null

Reference Types/ Non- Primitive

Object
Array
Function

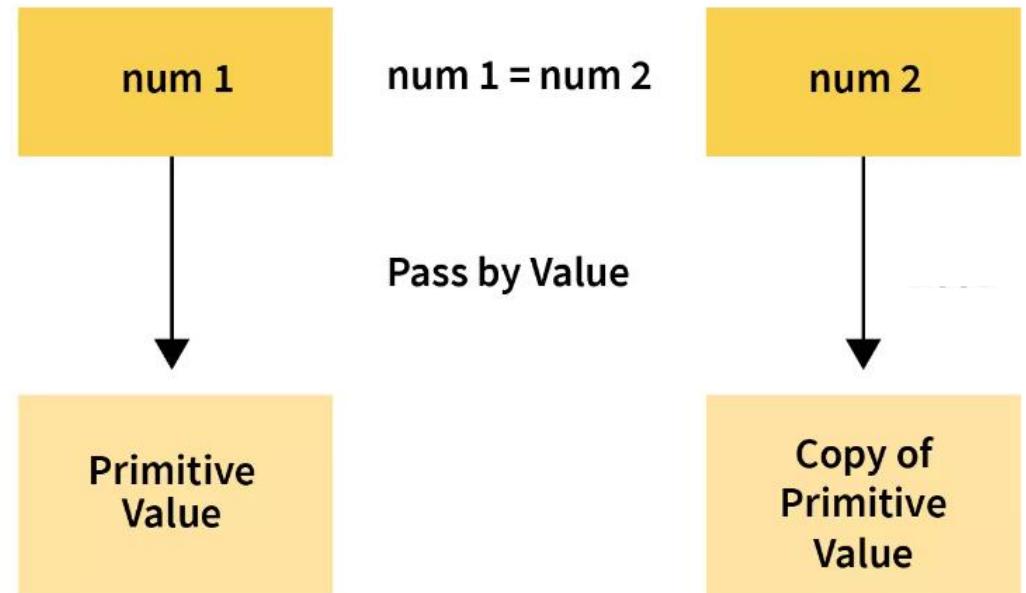
By Val

```
<script>
  let num1 = 70
  let num2 = num1

  console.log(num1) // 70
  console.log(num2) // 70

  num1 = 40

  console.log(num1) // 40
  console.log(num2) // 70
</script>
```



By Val

```
<script>
function multiplication(tmp) {
    tmp = tmp * 50;
    return tmp;
}
let num = 30;
let result = multiplication(num);
console.log(num); // 30
console.log(result); // 1500
</script>
```

tmp	num
1500	30

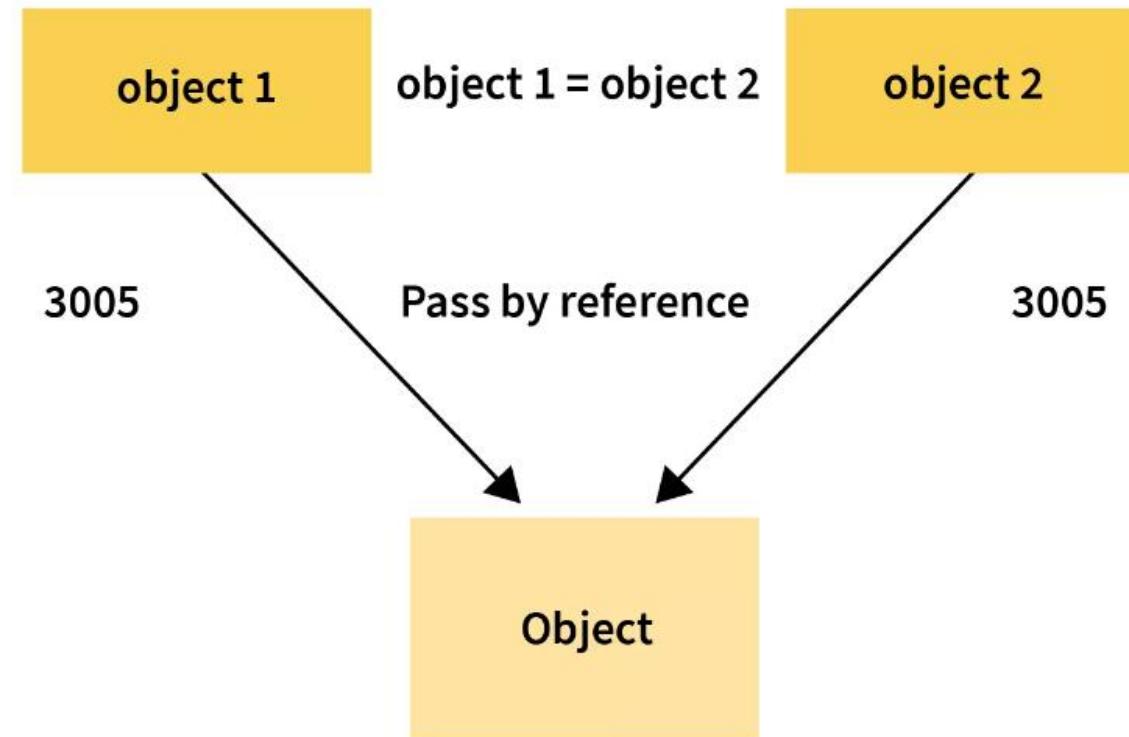
By Ref

```
<script>
  let obj1 = { website: "Yael" }
  let obj2 = obj1;

  console.log(obj1)      // {website: "Yael"}
  console.log(obj2)      // {website: "Yael"}

  obj1.website = "Uri"

  console.log(obj1)      // {website: "Uri"}
  console.log(obj2)      // {website: "Uri"}
</script>
```

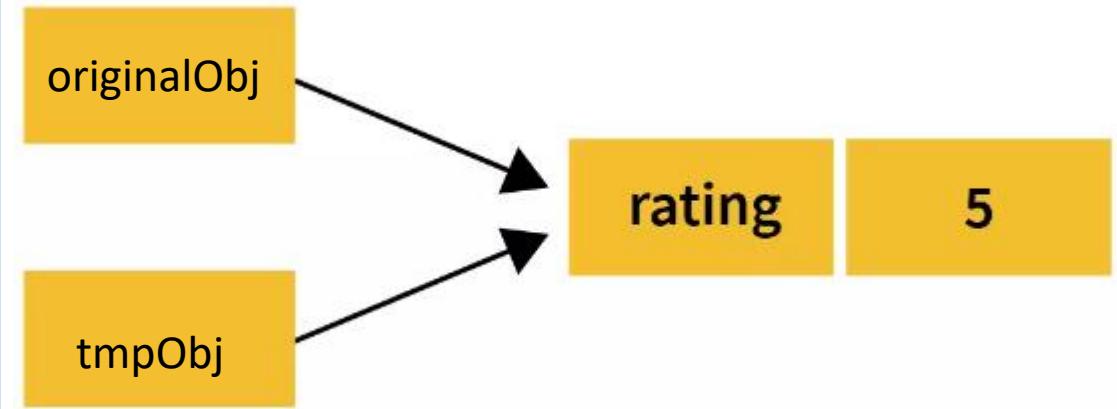


By Ref

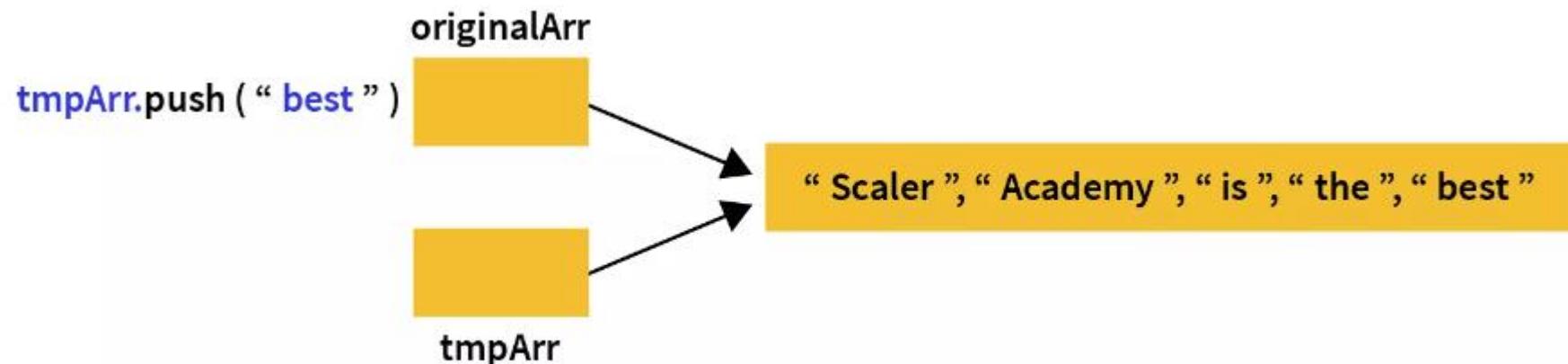
```
<script>
let originalObj = {
    name: "Yael",
    rating: 4.5,
    topic: "JavaScript"
};

function demo(tmpObj) {
    tmpObj.rating = 5;
    console.log(tmpObj.rating);
}

console.log(originalObj.rating);    // 4.5
demo(originalObj);                // 5
console.log(originalObj.rating);    //5
</script>
```



```
let originalArr = ["Client", "Course", "is", "the"];  
  
function pushArray(tmpArr) {  
    tmpArr.push('best')  
    console.log(tmpArr);  
}  
  
console.log(originalArr);      // ["Client", "Course", "is", "the"]  
pushArray(originalArr);       // ["Client", "Course", "is", "the", "best"]  
console.log(originalArr);     // ["Client", "Course", "is", "the", "best"]
```



תרגיל ביתה

צור אובייקט מסוג Deck

שדות האובייקט: מערך של קלפים cards

fonkציות האובייקט:

1. - יצרת חבילה קלפים מהמספרים 1,2,3,4,5 createDeck .1

ומהצבעים 'blue', 'red', 'green' (15 קלפים)

2. - מערבבת את החבילה (fonקציה קיימת בראש חפסו

אותה והעתיקו אליהם)

3. - מקבלת קלף ומוסיפה אותו לחבילה addCard .3

4. - מקבלת קלף ומסירה אותו מהחבילה removeCard .4

- **צור אובייקט מסווג Card**
- שדות האובייקט: color, num
- fonקציות האובייקט:
1. isEqualColor - מקבלת צבע ומחזירה true או false אם הקלף באותו שהתקבל

תרגיל ביתה

- **מהלך התוכנית:**
- צור חבילה קלפים (15 קלפים)
- ערבל אותה על ידי shuffle
- הגרל קלף מהחבילה
- הוציא את כל הקלפים בצבע שהוגרל והעביר אותם לחבילה חדשה.
למשל : אם הוגרל ירוק – כל הקלפים הירוקים יוצאו מחבילת הקלפים
ויעברו לחבילה חדשה שתיצרו.
- בסוף התהליך – בחבילה הישנה יהיו 10 קלפים בחבילה החדשה יהיו 5
קלפים באותו צבע שהוגרל.
- הדפיסו ל `console.log` את חבילות הקלפים

```
<script>
function Card(num, color,) {
this.num = num;
this.color = color;
}
function Deck() {
this.cards = [];
}

Card.prototype.isEqualColor = function (i_color) {
if (this.color == i_color)
return true;
else
return false;
}

Deck.prototype.createDeck = function () {
let nums = ['1', '2', '3', '4', '5'];
let colors = ['blue', 'red', 'green'];
for (let i = 0; i < colors.length; i++) {
for (let j = 0; j < nums.length; j++) {
this.cards.push(new Card(nums[j], colors[i]));
//console.log(this.cards);
}
}
}
Deck.prototype.shuffle = function () {
let i = this.cards.length, j, temp;
while (--i > 0) {
j = Math.floor(Math.random() * (i + 1));
temp = this.cards[j];
this.cards[j] = this.cards[i];
this.cards[i] = temp;
}
}
Deck.prototype.addCard = function (card) {
this.cards.push(card);
}
Deck.prototype.removeCard = function (card) {
let index = this.cards.indexOf(card);
//console.log(index);
this.cards.splice(index, 1);
}

let deckMain = new Deck();
let deckNew = new Deck();

deckMain.createDeck();

let randIndex = Math.floor(Math.random() * deckMain.cards.length);
//console.log(randIndex);
let randColor = deckMain.cards[randIndex].color;

createNewDeckByRandColor(randColor);

function createNewDeckByRandColor(randColor) {
let deckTemp = new Deck();
for (let i = 0; i < deckMain.cards.length; i++) {
if (deckMain.cards[i].isEqualColor(randColor)) {
deckTemp.addCard(deckMain.cards[i]);
}
}

for (let i = 0; i < deckTemp.cards.length; i++) {
deckNew.addCard(deckTemp.cards[i]);
deckMain.removeCard(deckTemp.cards[i]);
}
}

console.log(deckMain.cards);
console.log(deckNew.cards);
</script>
```

שאלות?

JSON Local Storage

היום בהרצאה

- JSON
- localStorage
- include a header across multiple pages
- Query string

JSON (JavaScript object notation)

- **JSON** הוא פורמט לשימרת מידע
- המידע מוחזק בזוגות של key: value עם נקודותים ביניהם
- הזוגות key:value מופרדים על ידי פסיק `,` `,` `,` `,`
- מתחיל ונסגר בסוגרים מסולסלים `{}`
- יכול להכיל מערכים, המערכים מוגדרים על ידי סוגרים מרובעים `[]`

JSON Keys

JSON key is string : •

{"name": "John"} •

JSON Values

- String : {"name": "John"}
- Number : {"age": 30}
- Object : {"employee": {"name": "John", "age": 30, "city": "New York"}}
- Array : {"employees": ["John", "Anna", "Peter"]}
- Boolean : {"sale": true}
- Null : {"middlename": null}

```

let superHeroes = {
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
  ]
}

```

JSON structure

```

console.log(superHeroes.homeTown);
console.log(superHeroes['active']);
console.log(superHeroes['members'][1]['powers'][2]);

```

מה יודפו?

```
let arr = [  
    {  
        "name": "Molecule Man",  
        "age": 29,  
        "secretIdentity": "Dan Jukes",  
        "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]  
    },  
    {  
        "name": "Madame Uppercut",  
        "age": 39,  
        "secretIdentity": "Jane Wilson",  
        "powers": [  
            "Million tonne punch",  
            "Damage resistance",  
            "Superhuman reflexes"  
        ]  
    }  
];
```

הדף את השדה : **"Million tonne punch"**

תרגיל חימום



thai

Bangkok, 100

King Thai, 50

thai house, 35

zozo rest, 50

TomYam house, 48

Masaman, 33



indian

Chapati, 15

Delhi Kitchen, 60

Pushkar delicates, 25

Bombay fire, 30

בצע את הפעולות הבאות על ה JSON הנוכחי בשלד

JSONExPreRestConsole_Skelton

.1 הדפו את סוגי המסעדות ("thai", "indian")

.2 הדפו את סוגי המסעדות ומתחילה הדפו את רשימת המסעדות

לכל סוג ומחירים.

תרגיל ביתה המשך

- יש ליצור באופן דינامي רדיו לכל סוג מסעדה שקיים בJSON
- כאשר המשתמש ילחץ על סוג מסעדה מסוים (checked) יש להציג את שמות כל המסעדות מסוג זה.
- בלחיצה נוספת (unchecked) יעלמו המסעדות מסוג זה.

Cheep Restaurant

Choose a type of restaurant for see list of resturants:

thai indian

Cheep Restaurant

Choose a type of restaurant for see list of resturants:

thai indian

Bangkok: 100

King Thai: 50

thai house: 35

zozo rest: 50

TomYam house: 48

Masaman: 33

JSON במחוזת ובאובייקט

- פורמט json מאד דומה ל javascript אך קל להמיר אובייקטים של JSON ל Javascript ולהפך:

- המרת JSON ל Javascript

- JSON.parse()**

- המרת Javascript ל JSON

- JSON.stringify()**

JSON.Parse

- <body>
- <h2>Creating an Object from a JSON String</h2>
- <p id="demo"></p>
- <p id="demo1"></p>
- <script>
- const txt = '{"name":"John", "age":30, "city":"New York"}'
- const obj = JSON.parse(txt);
- document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
- const text = '{"cars" : ["Ford", "BMW", "Audi", "Fiat"]}';
- const obj1 = JSON.parse(text);
- document.getElementById("demo1").innerHTML = obj1.cars[0];
- </script>
- </body>

תאריכים ופונקציות

- **תאריך יש להביר כמחרוזת**

למשל: "birth": "1986-12-14"

- *(אחכ ניתן להמיר ל date ב javascript על ידי שימוש באובייקט Date)

- **פונקציה יש להביר כמחרוזת**

למשל: "age": "function () {return 30;}"

- *(אחכ ניתן להמיר לפונקציה על ידי שימוש ב eval של javascript)

```
<script>
const txt = `{"name":"John", "age":function () {return 30;}, "city":"New York"}`;
          const obj = JSON.parse(txt);

          // Evaluate the string as a function using eval
          const ageFunction = eval('(' + obj.age + ')');

          // Call the function
          const age = ageFunction();
          console.log(age); // Output: 30

//-----
const txt1 = `{"name":"John", "date":"01/01/2025", "city":"New York"}`;
          const obj1 = JSON.parse(txt1);

          // Evaluate the string as a date
          const date = Date(obj1.date);
          console.log(date);
</script>
```

JSON.stringify

- <body>
- <h2>Create a JSON string from a JavaScript object.</h2>
- <p id="demo"></p>
- <script>
- const obj = {"name": "John", age: 30, city: "New York"};
- const myJSON = JSON.stringify(obj);
- document.getElementById("demo").innerHTML = myJSON;
- </script>
- </body>

העברת נתונים בין דפים

העברת נתונים בין דפים

יש מספר דרכים להעברת נתונים בין דפים בפיתוחצד לקוח, הדרכים העיקריות:

1. **localStorage** – שמירת מידע פר אטר (דומיין) על מחשב הלקוח. מאובטח כלפי יוצרים אחרים – העברת מידע סמויה.

2. **Query string** – העברת נתונים דרך ה-URL - www.mySite.co.il?username=yael (נסתכל רק על דוגמא) משמש להעברת מידע לא מסויים.

3. **cookie** – מידע נשמר על מחשב הלקוח וMOVABR לשרת בכל פניה. (לא ניגע בקורס זה)

localStorage

- מחסן מקומי לשימירת נתונים העובד בצורה של key\value.
- בכל צמד של key\value, הערך **ישמר כמחרוזת**.

The screenshot shows the Chrome DevTools interface with the Application tab selected. In the left sidebar, under the Storage section, 'LocalStorage' is expanded, showing a list of items. One item, 'http://localhost:59445', is highlighted with a red oval. In the main content area, a table displays a single key-value pair: 'lastname' (Key) and 'Smith' (Value), both highlighted with red ovals. The table has columns labeled 'Key' and 'Value'. A red oval also highlights the 'Application' tab at the top of the DevTools.

Key	Value
lastname	Smith

- המחסן בעל נפח אחסון של 5 MB.

localStorage

- לערכים אין תאריך תפוגה עד שהם לא נמחקים.
- גם אם נסגור את הדף ונפתח אותו שוב הערך יהיה קיים (כנ"ל אם נכבה את המחשב).
- כל הדפים באותו הדומיין יכירו את המידע ששמרנו.



localStorage

השפת>User	hbavat>User
<ul style="list-style-type: none">• <code>localStorage.setItem("KEY", VALUE);</code>• <code>localStorage["KEY"] = VALUE;</code>• <code>localStorage.KEY = VALUE;</code>	<ul style="list-style-type: none">• <code>localStorage.getItem("KEY");</code>• <code>localStorage["KEY"]</code>• <code>localStorage.KEY</code>

localStorage

מחיקת ערכים	בדיקה אם המפתח קיים
<ul style="list-style-type: none">• <code>localStorage.removeItem("KEY");</code>• <code>localStorage.clear();</code>	<ul style="list-style-type: none">• <code>if (localStorage["name"] != undefined)</code> <code>name = localStorage["name"] ;</code>

שמירת אובייקט ב localStorage

- ב localStorage ניתן לשמור רק מחרוזות
- על מנת לשמור אובייקט נהפוך אותו למחרוזת json
- על ידי שימוש ב `JSON.stringify(myObject)`
- ```
const myObject = {
 name: "john doe",
 age: 32,
 gender: "male",
 profession: "optician"
}

localStorage.setItem("myObject", JSON.stringify(myObject));
```

# החזרת אובייקט מה localStorage

---

- `let newObject = localStorage.getItem("myObject");`
- `console.log(JSON.parse(newObject));`

# תרגיל ביתה pay

---

1. עבדו על ה Skelton המצורף
2. צרו כפטור עבור כל צעוז
3. לחיצה על הצעוז תוסיף אותו לסל
4. לחיצה על הסל תעביר לדף pay אשר יציג את רשימת הפריטים שנבחרו
5. כל טעינת דף יש לרוקן את העגלה ב pay ולהתחליל למלא מחדש.

Click the buttons to add to the cart

robot

doll

ball

monopoly



Click on the cart image to proceed to payment

clear cart

Click the buttons to add to the cart

robot

doll

ball

monopoly



Click on the cart image to proceed to payment

clear cart

## Your cart:

name: robot price: 100

name: doll price: 70

name: ball price: 30

name: monopoly price: 65

**Total Price = 265**

# הציג לוגין ליזר רשימת יוזרים

---

1. עלינו ליצור דף לוגון עבור מערכת
2. כניסה למערכת דוחשת אימייל וסיסמה
3. המערכת בודקת את האימייל והסיסמה אל מול רשימת היוזרים (רשימת היוזרים נשמרת בlocalStorage)
4. אם היוזר קיים – הוא מועבר לדף הבא
5. אם הוא אינו קיים הוא מקבל הודעה מתאימה

---

# שאלות?

# JSON Part 2

# תרגיל ביתה – חלק א

---

1. הורד מה moodle את קובץ השלד
2. עליינו ליצור דף לוגין עבור מערכת
3. כניסה למערכת דורשת אימייל וסיסמה
4. המערכת בודקת את האימייל והסיסמה אל מול רשימת היוזרים (רשימת היוזרים נשמרת ב `localStorage`)
5. אם היוזר קיים – הוא מועבר לדף הבא
6. אם הוא אינו קיים הוא מקבל הודעה מתאימה

# חלק ב

## Band Information

--select band--

—select band—

Metallica

U2

Abba

Imagine Dragons

### • חלק א

- הוסף לדף הבא מסעיף א רשימת בחירה של להקות JSON.
- בשלד נתון אובייקט JSON המתאים להקות.  
עם עליית הדף עליכם **למלא את שמות** להקות שקיימות בJSON באלמנט select.
- יש לрендר את האלמנטים דינמית ולהכניסם לתוך ה SELECT כמו בדוגמה.

## Band Information

U2

**U2**

U2 are an Irish rock band from Dublin, formed in 1976. The group consists of Bono (lead vocals and rhythm guitar), the Edge (lead guitar, keyboards, and backing vocals), Adam Clayton (bass guitar), and Larry Mullen Jr. (drums and percussion).

צד לקוח - יעל סלע זעירא

### • חלק ב

- בחירה בלהקה תציג את שם הלהקה ואת התיאור שלה מה JSON

# העברת מידע באמצעות Query string

---

העברת מידע דרך ה-URL באמצעות הפרדת כתובות הדף ע"י ?

הנתונים עוברים ע"י צמד של מפתח-ערך, כאשר כל צמד מופרד באמצעות &

למשל, העברת של קניית מוצר:

```
buy milk
```

העברה של קניית מוצר + כמות יחידות:

```
buy 3 units of milk
```

# הברת מידע באמצעות Query string

---

תפישת המידע בדף המתקבל ע"י location.search נותן לנו את החלק מס'ינן השאלה ועד סוף המחרוזת.

location.search → ?product=milk&amount=3

נעשה שימוש באובייקט `URLSearchParams` של שפת js על מנת להחזיר את הערך שאנו רוצים:

\*פונקציית `get` של האובייקטמחזירה את הערך של המפתח

```
let url = new URLSearchParams(location.search.substring(1));
url.get("product");
url.get("amount");
```

```
<body>
buy 3 units of milk
</body>
```

## queryStringProduct.html

```
<html>
<head>
 <meta charset="utf-8" />
 <title></title>
</head>
<body>
 <script>
 console.log("keys:" + location.search); //?product = milk & amount=3
 let spn = document.createElement("span");
 document.body.appendChild(spn);
 spn.innerHTML += "Number of " + url.get("product") + " : ";
 spn.innerHTML += url.get("amount");
 </script>
</body>
</html>
```

# תרגיל ביטה queryString

---

enter your name



הכנס את שמו לחץ על אחד הריבועים

**Welcome!**

בדף אחרי רשום את ההודעה הבאה

Hello, Yael! you chose the color red.

# זמן פונקציות

`setTimeout`

`setInterval`

# setTimeout

---

כasher rozim leheriz at kod ha javascript achri zman mosiyim (miliseconds)

```
setTimeout("console.log('Hello World!')", 2000);
```

herishom log ytbazu laachar shai shniot (seconds 2 = miliseconds 2000)

# שליחת פרמטרים

---

`setTimeout(code, delay)`

`setTimeout(functionRef, delay)`

`setTimeout(functionRef, delay, param1)`

`setTimeout(functionRef, delay, param1, param2)`

`setTimeout(functionRef, delay, param1, param2, /* ..., */ paramN)`

# setTimeout

---

```
<body>
 <h1 id="counter">0</h1>
 <script>
 function increaseCountAfter5Sec() {
 let s = document.querySelector("#counter");
 s.innerHTML = parseInt(s.innerHTML) + 1;
 }

 setTimeout(increaseCountAfter5Sec, 1000)
 </script>
</body>
```

הפונקציה מקבלת:

פונקציה לביצוע

פרמטר נוסף שמצוין עוד כמה זמן  
יש להפעיל את הפונקציה: = 1000  
1 sec

# תזמן פונקציות

---

בדומה לארוע דינامي, ניתן לרשום את הפונקציה לביצוע כר:

```
setTimeout(function () {
 let s = document.querySelector("#counter");

 s.innerHTML = parseInt(s.innerHTML) + 5;

}, 1000)
```

# clearTimeout

---

פונקציית setTimeout מחזירה מספר ייחודי, באמצעות מספר זה ניתן למנוע את ביצוע הפונקציה ע"י

```
<body>
 0
 <script>
 let increaseCount = setTimeout(
 function() {
 console.log("start");
 let s = document.querySelector("#counter");

 console.log(parseInt(s.innerHTML));

 s.innerHTML = parseInt(s.innerHTML) + 1;
 console.log(parseInt(s.innerHTML));

 }, 1000);

 clearTimeout(increaseCount);
 </script>
</body>
```

פונקציית clearTimeout מתקבלת את המספר הייחודי שיצרה setTimeout הפונקציה מקבלת את המספר הייחודי clearIncreaseCountAfter5Sec

```

//Fruit Arrays 1 & 2 declaration & assignment

const fruitArr1 = ['apple', 'mango'];

const fruitArr2 = ['banana', 'papaya'];

//If the fruit name is mango, the clearTimeout method will clear the timeout value and mango fruit will
not be displayed

function checkFruit(fruitArr) {

 for (var i in fruitArr) {

 const timeout = setTimeout(displayFruit, 2000, fruitArr[i]);

 if (fruitArr[i] === 'mango') {

 clearTimeout(timeout);

 }

 }

}

//Function to display the selected fruit name

function displayFruit(fruit) {

 console.log(`The fruit selected is ${fruit}`);

}

//Invoke checkFruit method

checkFruit(fruitArr1);
checkFruit(fruitArr2);

```

# modal

---

כאשר רוצים להציג הודעה ניתן להשתמש ב modal למשל <dialog>

```
<dialog id="myDialog">This is a modern alert!</dialog>
<script>
 const dialog = document.getElementById('myDialog');
 dialog.showModal();
 setTimeout(() => dialog.close(), 2000); // Auto-close after 2 seconds
</script>
```

# setInterval

---

```
<body>
 0
 <script>
 function increaseCountAfter1Sec() {
 let s = document.querySelector("#counter");
 s.innerHTML = parseInt(s.innerHTML) + 1;
 }

 setInterval(increaseCountAfter1Sec, 1000)
 </script>
</body>
```

הfonקציה תבצע את הקוד שוב ושוב לפי  
זמן שקבע:

יצרנו מונה שיגדל בכל שנייה את הערך ב-  
span

\*הפסקת ה Fonקציה ע"י Fonקציית  
.clearInterval

# чикוי setTimeout ו setInvterval

---

באמצעות רקורסיה ניתן לרשום קוד שחוזר על עצמו עם פונקציית `:setTimeout`

```
<body>
 0
 <script>
 function increaseCountAfter1Sec() {
 let s = document.querySelector("#counter");

 s.innerHTML = parseInt(s.innerHTML) + 1;
 setTimeout(increaseCountAfter1Sec, 1000)

 }

 setTimeout(increaseCountAfter1Sec, 1000)
 </script>
</body>
```

# תרגיל פתרים יחד

---

- כתוב count down timer הסופר את הזמן מעכשיו עד ההפסקה

0d 0h 29m 14s

- כאשר מגיעה ההפסקה יופיע

**TIME UP!!**

---

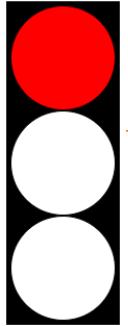
CountDount.html •

# clearInterval

---

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <title></title>
</head>
<body>
 <progress value="0" max="10" id="progressBar"></progress>

 <script>
 let timeleft = 10;
 let downloadTimer = setInterval(function () {
 if (timeleft <= 0) {
 clearInterval(downloadTimer);
 }
 document.getElementById("progressBar").value = 10 - timeleft;
 timeleft -= 1;
 console.log("tick");
 }, 1000);
 </script>
</body>
</html>
```



# תרגיל מסכם



צרו רמזור שעבוד לפי החוקים הבאים:

- אור אדום – 5 שניות
- אור צהוב – 1 שניה
- אור ירוק 5 שניות ולאחר מכן יחזור לצהוב וכן הלאה.

```
table { background-color: black; }
td {
 width: 30px;
 height: 30px;
 border-radius: 50%;
 background-color: white;
}
```

# Callback function

# סינכרוני / אסינכרוני

---

- **סינכרוני** - פעולה אחת פעם אחת בכל זמן, הפעולה הבאה לא תתחל לפני שקדמתה תסתיים.
  - ```
console.log('Hello #1');
console.log('Hello #2');
console.log('Hello #3');
```
- **אסינכרוני** – פעולה שנתקראת ורחב לא תהיה לנו יכולות לדעת מתי הפעולה תסתיים ותחזיר לנו את המידע הרצוי, אם נרצה לעשות שימוש במידע שמתקבל עליינו "להמתין" למידע.
- למשל כאשר אנחנו רוצים לקרוא קובץ מהאינטרנט, אנחנו לא יכולים לעזור את כל התוכנית עד שהקובץ יישען, ולכן משתמש ב *Callback* כדי להפעיל את הקוד שיטפל בנתונים לאחר שהקובץ יישען.

Callback function

- **פונקציה שנשלחת לפונקציה אחרת כארגומנט (פונקציית ה callback)**
- הפונקציה הראשונה קוראת **לפונקציה ה-Callback**. לאחר שהפונקציה הראשונה ביצעה פועלתה.
- דרך רבת עצמה לטפל **בפעולות אסינכרניות (פעולות שלא מתבצעות מיד, כמו קריית נתונים מהשרת או טיימרים)**.
- Callback function נפוצות מאוד ב- JavaScript במיוחד בעבודה עם קוד אסינכרוני, כמו **טעינת קבצים, בקשوت רשת, או ניהול זמן**.

Callback function

```
function operation(val1, val2, callback) {  
    callback(val1, val2)  
}  
  
function sum(a, b) {  
    console.log(a + b)  
}  
  
function divide(a, b) {  
    console.log(a / b)  
}  
  
operation(6, 5, sum)  
operation(9, 3, divide)
```

דוגמה greeting

- פונקציה שמדפיסה ברכה כלשהי.
- פונקציה שמקבלת מידע נרחב יותר על הברכה וקוראת ל `greetingFunc` ה函數 שמדפיסה את הברכה עצמה.

```
function greetingFunc (greetingText)
{
    console.log(`Hi ${greetingText}`);
}
```

```
greetingInfoFunc('John', 'Morning', greetingFunc);
Hi John, Good Morning!
```

```
function greetingInfoFunc(name, dayTime, callbackFunc) {
    const greetingText = `${name}, Good ${dayTime}!`;
    callbackFunc(greetingText);
}
```

```
greetingInfoFunc('John', 'Morning', greetingFunc);
```

יצרנו פונקציה שמדפסה ברכה כלשהי `callbackFunc`

ופונקציה נוספת שמקבלת מידע נרחב יותר על הברכה `greetingInfoFunc`

הfonקציה `greetingInfoFunc` קיבלה ארגומנט את הפונקציה `greetingFunc` וקוראת לה.

פונקציות זהות לשקף הקודם שנכתבו כ :arrow function

```
const greetingFunc = greetingText => console.log(`Hi ${greetingText}`);  
  
const greetingInfoFunc = (name, dayTime, callbackFunc) => {  
    const greetingText = `${name}, Good ${dayTime}!`;  
    callbackFunc(greetingText);  
}  
greetingInfoFunc('John', 'Morning', greetingFunc);
```

דוגמא : square array

```
<script>
function processArray(arr, callback) {
    const result = []; // Create an empty array to hold results
    for (let i = 0; i < arr.length; i++) {
        result.push(callback(arr[i]));
    }
    return result; // Return the transformed array
}

function square(number) {
    return number * number;
}

// Array of numbers
const numbers = [1, 2, 3, 4, 5];

const squaredNumbers = processArray(numbers, square);

// Output the result
console.log(squaredNumbers); // [1, 4, 9, 16, 25]
</script>
```

פונקציה אשר מבצעת פעולה על איבר במערך ומחזירה
מערך חדש עם המכילים את האיברים לאחר הפעולה

תרגיל callback

- ממש את פונקציית המערך forEach

```
const items = ["item1", "item2", "item3"];
const copyItems = [];
```

```
// before forEach
for (let i = 0; i < items.length; i++) {
  copyItems.push(items[i]);
}
```

```
// after
items.forEach((item) => {
  copyItems.push(item);
});
```

פתרונות

Iterates over each element in the array, invoking the callback with the current element and its index.
Does not return a value.

```
function myforEach(array, callback) {  
  
  if (!Array.isArray(array)) {  
    throw new TypeError('First argument must be an array.');//  
  }  
  if (typeof callback !== 'function') {  
    throw new TypeError('Second argument must be a function.');//  
  }  
  
  for (let i = 0; i < array.length; i++) {  
    callback(array[i]);  
  }  
}
```

פתרונות

```
const items = ["item1", "item2", "item3"];
const copyItems = [];
```

```
myforEach(items, function(item){
    copyItems.push(item);
});
```

```
console.log(copyItems);
```

פתרונות

```
const items = [1, 2, 3];
const copyItems = [];

myforEach(items, function (item) {
    copyItems.push(item*item);
});

console.log(copyItems);
```

Callback with setTimeout

כפי שראינו `setTimeout` היא פונקציה אסינכרונית. הרץ את הדוגמא המצורפת להמחשה.

`console.log("Start")` נקראת **imediatly**

הfonkzia שבתוך ה `timeout` מתווספת לתוך אירועים ולא חוסמת את הקוד בהמשך.

```
console.log("Start");

// Schedule a function to be executed after a 2-second delay
setTimeout(() => {
  console.log("Inside setTimeout - Executed after 2 seconds");
}, 2000);

console.log("End");
```

Callback with setTimeout

- ניתן להשתמש ב setTimeOut כדי לדמות פונקציה אסינכרונית, דוגמאות:
 1. היבאת מידע מה server
 2. שליחת הודעה
 3. הורדת קובץ
 4. קניית כרטיס להופעה

המודמה פועלה אסינכרונית setTimeout עם callback של הבאת נתונים משרת מרוחק

```
function fetchData(callback) {
    console.log("Starting data fetch...");

    setTimeout(() => { // Use setTimeout to mimic a delay in an asynchronous operation
        const data = { message: "Data fetched successfully!" };
        console.log("Data fetched.");

        callback(data);
    }, 2000);
}

function processFetchedData(data) { // callback function
    console.log("Processing data: ", data.message);
}

// Initiating the fetch operation, passing the callback function to process the data
fetchData(processFetchedData);

console.log("This will log before data fetch is complete due to async behavior.");
```

המודעה פועלה אסינכרונית של הורדת קובץ

```
function downloadFile(callback) {  
    console.log("Starting file download...");  
  
    setTimeout(function () {  
        console.log("File downloaded successfully.");  
        callback();  
    }, 3000);  
}  
  
function processFile() {  
    console.log("Processing the downloaded file...");  
}  
  
downloadFile(processFile); // קודם מורידים את הקובץ ואז מעבדים אותו
```

Callback hell

- כאשר ישן מספר פעולות אסינכרוניות אחת בשנית, הפתרון הנפוץ הוא לשים Callback בתוך Callback
- זה יכול ליצור קוד לא קרייא ולא נוח לתחזוקה, מה שנקרא **Callback Hell**

Callback hell

- נבצע את הפעולות הבאות:
- 1. Walk the dog
- 2. Clean the kitchen
- 3. Take out the trash

Callback hell

```
function walkDog() {  
    setTimeout(function () {console.log("You walk the dog")}, 1500);  
}
```

```
function cleanKitchen() {  
    setTimeout(function () { console.log("You clean the kitchen") }, 2500);  
}
```

```
function takeOutTrash() {  
    setTimeout(function () { console.log("You take out the trash") }, 500);  
}
```

Callback hell

```
function walkDog(callback) {  
    setTimeout(function () {  
        console.log("You walk the dog");  
        callback();  
    }, 1500);  
}  
  
function cleanKitchen(callback) {  
    setTimeout(function () {  
        console.log("You clean the kitchen");  
        callback();  
    }, 2500);  
}  
  
function takeOutTrash(callback) {  
    setTimeout(function () {  
        console.log("You take out the trash");  
        callback();  
    }, 500);  
}
```

Callback hell

```
walkDog(function () {  
  cleanKitchen(function () {  
    takeOutTrash(function () { console.log("DONE"); });  
  });  
})
```

דוגמה ל-Hell

שלושה setTimeout אחד בתוך אחד השני.

בכל שלב, פעולה מתבצעת ואז מתבצעת פעולה נוספת נספפת שתלויה בה.

הקוד הופך למסובך יותר ככל שימושיפים עוד שלבים, והקינון הופך את הקראיות שלו למורכבת

```
<script>
setTimeout(function () {
    console.log("Step 1 complete.");
    setTimeout(function () {
        console.log("Step 2 complete.");
        setTimeout(function () {
            console.log("Step 3 complete.");
        }, 1000);
    }, 1000);
}, 1000);

</script>
```

Callback hell בעיות ופתרונות

פתרונות:

Promise

Async and wait

בעיות:

- לא קריית

- Debugging בעיית

- ניהול שגיאות בעיית

שאלות?

Closure

צד ל��וח - יעל סלע זעירא

Global Variables

- לפונקציה יש גישה לכל המשתנים שלה
- כמו בדוגמה הבאה : המשתנה a

```
<body>
  <p id="demo"></p>

  <script>
    myFunction();

    function myFunction() {
      let a = 4;
      document.getElementById("demo").innerHTML = a * a;
    }
  </script>
</body>
```

Global Variables

- לפונקציה יש גישה לכל המשתנים המוגדרים מחוץ לפונקציה
- כמו בדוגמה הבאה : המשתנה a

```
<body>
  <p id="demo"></p>
  <script>
    let a = 4;
    myFunction();

    function myFunction() {
      document.getElementById("demo").innerHTML = a * a;
    }
  </script>
</body>
```

Global Variables

- משתנים המוגדרים `var`, `let`, `const` יהיו גלובליים

```
<body>
  <p id="demo"></p>

  <script>
    myFunction();
    document.getElementById("demo").innerHTML = a * a;

    function myFunction() {
      a = 4;
    }
  </script>
</body>
```

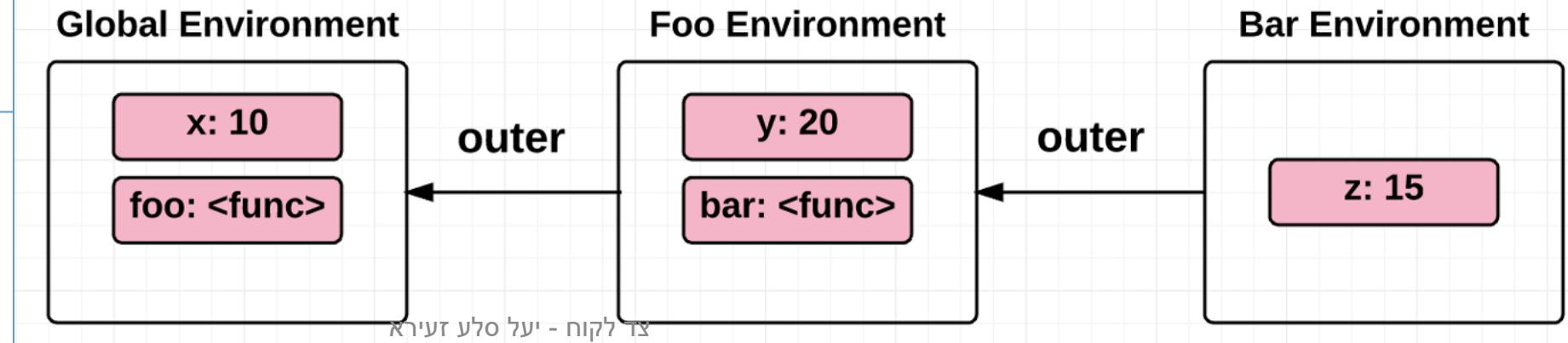
Closure

- Closure היא פונקציה פנימית (inner function) אשר יש לה מצב שעוטף אותה (outer function)
- במילים אחרות clouser נตอน גישה ל:
 - ✓ למשתנים של הפונקציה עצמה
 - ✓ למשתנים של הפונקציה שעוטפה אותה
 - ✓ למשתנים הגלובליים
- פונקציה כזו היא פונקציה ש"זוכרת" את הסביבה שיצרה אותה

Scope

מה ידפו?

```
<script>  
var x = 10;  
  
function foo() {  
    let y = 20;  
    function bar() {  
        let z = 15;  
        return x + y + z;  
    }  
    return bar;  
}  
let a = foo();  
console.log(a())  
</script>
```



מה יודפס?
איזה x ילכוד במקרה זה ?

```
var x = 10;  
  
function foo() {  
    var y = x + 5;  
    return y;  
}  
  
function bar() {  
    var x = 2;  
    return foo();  
}  
console.log(foo());  
console.log(bar());
```

ה scope של פונקציה נקבע לפי המקום שבו היא מוגדרת, לא היכן שהיא נקראת.
מכיוון ש ()foo-מוגדר ב scope-הגלובלי, היא ניגשת ל x-הגלובלי.
casar ()bar קורא ל ()foo מהזיר 15 תור שימוש בערך ה x-הגלובלי של 10 לא ה x-המקומי של ()bar הוא 2

```
<script>
  function sayHello() {
    let say = function () { console.log(hello); }
    // Local variable that ends up within the closure
    let hello = 'Hello, world!';
    return say;
  }
  let sayHelloClosure = sayHello();
  sayHelloClosure(); // 'Hello, world!'
</script>
```

משתנה hello מוגדר אחרי פונקציה אונומית אך לפונקציה עדין יש גישה אליו, זאת מכיוון שהמשתנה כבר הוגדר כמשתנה הנמצא ב scope של הפונקציה בזמן שהפונקציה נוצרה ולכן כבר זמין בזמן הריצעה של sayHello.

אנו רואים שיש לנו גישה למשתנים של הפונקציה העוטפת גם לאחר שהפונקציה חזרה.

```
var x = 10;
function foo(a) {
    var b = 20;

    function bar(c) {
        var d = 30;
        return boop(x + a + b + c + d);
    }

    function boop(e) {
        return e * -1;
    }

    return bar;
}

var moar = foo(5); // Closure
moar(15);
```

```
var x = 10;
function foo(a) {
    var b = 20;

    function bar(c) {
        var d = 30;
        return boop(x + a + b + c + d);
    }

    function boop(e) {
        return e * -1;
    }

    return bar;
}

var moar = foo(5); // Closure
moar(15);
```

הפונקציה `foo(5)` ממחזירה את `bar` לתוך המשנה `moar` קרייה ל `(15) moar` מפעילה את `bar` עם הParmטר `15` , בתוך `e` קיימ כבר הערך `5` מהקרייה ל `foo(5)`.
`bar` מפעילה את `boop` והערך המוחזר הוא:
$$(10+5+20+15+30)*-1$$

Use Cases of Closures

Data Encapsulation

- Closure מאפשר **ניהול המידע** – על ידי הסתרה של משתנים מסוימים, אפשרות **שינוי** משתנים רק **מקום מסוים**

```
function createCounter() {  
    let count = 0;  
  
    function increment() {  
        count++;  
        return count;  
    }  
  
    return increment;  
}
```

```
const counter1 = createCounter();  
const counter2 = createCounter();  
  
console.log(counter1()); // Output: 1  
console.log(counter2()); // Output: 1  
  
console.log(counter1()); // Output: 2  
console.log(counter2()); // Output: 2
```

מתבצעת אנקופולציה למשתנה ה count

הגישה היחידה למשתנה היא על יד

שיבצע הגדולה ויחזיר את הערך.

קריאה ל createCounter ממחזירה את . increment

מספר קריאות ל createCounter יחזרו closures

שמות שונים לכל אחד state שלו.

Use Cases of Closures

Event handlers with preserved state

```
function createGreeter(name) {  
    function greet() {  
        console.log("Hello, " + name + "!");  
    }  
  
    return greet;  
}  
  
const button1 = document.getElementById("button1");  
const button2 = document.getElementById("button2");  
  
button1.addEventListener("click", createGreeter("Alice"));  
button2.addEventListener("click", createGreeter("Bob"));
```

Use Cases of Closures

Functional Programming

אפשר ליצורי פונקציות שמקבלות פונקציות כפרמטרים ומחזירות פונקציות Closure

```
function createMultiplier(factor) {  
    return function (x) {  
        return x * factor;  
    };  
}  
  
const double = createMultiplier(2);  
const triple = createMultiplier(3);  
  
console.log(double(5)); // Output: 10  
console.log(triple(5)); // Output: 15
```

שאלות?

More topics in HTML5

HTML5



HTML הינה הגרסה המעודכנת לשפת HTML

תוספות:

- ניגון קטעי audio ו-video, ציור על canvas
- TAGיות רבות להבנה של הקוד עבור מפתחים ומונע הHIPPOSH
- תכונות חדשות לאלמנטים

HTML5

- HTML5 מציע יתרונות רבים
- מציע פיצרים חדשים מבוססי HTML, CSS, DOM, JavaScript
- טיפול טוב יותר בשגיאות
- מפחית את הצורך ב plugins חיצוניים ועוד....

<input> new types

- range • color
- search • date
- tel • datetime-local
- time • email
- url • month
- week • number

InputTypes.html •

Canvas

- אלמנט המאפשר לעשות ציור (כמו צייר) על ידי שימוש ב javascript
- כמו בתוכנת ציור שכבה דורשת שכבה לפניה
- הקנבס מלבן כאשר הפינה השמאלית העליונה היא נקודת ה 0,0



- ב default גודל הקנבס הנוצר הוא 300px על 150px גובה – ניתן לשנות ב styles כמו גם להוסיף border וcd'. העיצוב הזה יהיה רק על הקנבס עצמו ואין עיצוב הקנבס שקויף

```
<canvas id="myCanvas" width="300" height="200"></canvas>
```

ניתן להוסיף לו border

```
<style>
  canvas {
    border: 1px solid #000;
  }
</style>
```

canvas.getContext

- כדי לגשת לkanbos יש פונקציה הנקראת `getContext()` המגדירה מה י הצייר על הקנبو.
- במקרה שלנו נגדיר

```
const canvas = document.getElementById("tutorial");
const ctx = canvas.getContext("2d");
```

Rectangle

Syntax: `ctx.fillRect(x, y, width, height)`

`x, y`: Coordinates for the top - left corner of the rectangle.

`width, height`: The dimensions of the rectangle.

```
ctx.fillStyle = 'skyblue'; Set fill color
```

```
ctx.fillRect(50, 50, 300, 200); Draws a rectangle
```

Triangle

Creating a triangle involves defining a path and connecting three lines:

1. Begin a new path using `beginPath()`.
2. Move the starting point to the first vertex with `moveTo(x, y)`.
3. Draw lines to the subsequent vertices using `lineTo(x, y)`.
4. Close the path with `closePath()`.
5. Fill the triangle using `fill()`.

Triangle

```
ctx.beginPath(); Start a new path  
  
ctx.moveTo(50, 50); Move to the first vertex  
  
ctx.lineTo(200, 10); Draw line to the second vertex  
  
ctx.lineTo(350, 50); Draw line to the third vertex  
  
ctx.closePath(); Close the path to form a triangle  
  
ctx.fillStyle = 'darkred'; Set fill color  
  
ctx.fill(); Fill the triangle
```

Circle

To draw a circle, use the `arc()` method, which draws an arc or circle by specifying the center point, radius, and start and end angles.

Syntax: `ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise)`

- x, y: Center **of** the circle.
- radius: Radius **of** the circle.
- startAngle, endAngle: The angles defining the start and end **of** the arc **in** radians. Full circle is 0 to $2 * \text{Math.PI}$.
- anticlockwise: Optional boolean, which **if true** draws the arc counter - clockwise.

Circle

```
ctx.beginPath(); Start a new path  
  
ctx.arc(320, 80, 30, 0, Math.PI * 2, true); Draw a full circle  
  
ctx.fillStyle = 'yellow'; Set fill color  
  
ctx.fill(); Fill the circle
```

text

- Set Up Your HTML and Canvas
- Access the Canvas Context
- Set Text Properties
- Draw the Text

```
<body>
  <canvas id="myCanvas" width="600" height="300"></canvas>
  <script>
    function writeTextOnCanvas(canvasId, text) {
      // Get the canvas element and its 2D context
      const canvas = document.getElementById(canvasId);
      const ctx = canvas.getContext('2d');

      // Clear the canvas
      ctx.clearRect(0, 0, canvas.width, canvas.height);

      // Set text properties
      ctx.font = 'bold 48px Verdana';
      ctx.fillStyle = 'teal'; // Text color
      ctx.textAlign = 'center'; // Align text to the center
      ctx.textBaseline = 'middle'; // Align text vertically to the middle

      // Calculate the center position
      const x = canvas.width / 2;
      const y = canvas.height / 2;

      // Draw the text
      ctx.fillText(text, x, y);

    }
    // Example usage
    writeTextOnCanvas('myCanvas', 'Hello, Canvas!');
  </script>
</body>
```

drawImage.html •

Additional Info

- **Stroke vs Fill**: You can also use **strokeRect()** for drawing rectangles with just borders, **stroke()** for outlining paths, and **fill()** to fill them.
- **Colors and Styles**: Use **fillStyle** to set the fill color and **strokeStyle** for the border color.
- **Transforms and Rotations**: Advanced drawings can utilize transformations like scaling, rotating, and translating to manipulate the context state.

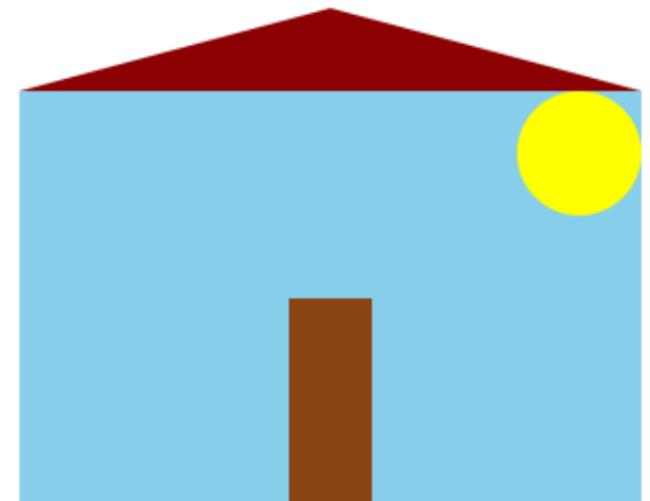
תרגיל ביתה

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Canvas Drawing Exercise</title>
<style>
canvas {
    border: 1px solid black;
}
</style>
</head>
<body>
    <canvas id="myCanvas" width="400" height="400"></canvas>
</body>
</html>
```

- העתק את ה skelton הבא

צייר את התמונה הבאה

- Draw a rectangle top left corner 50,50
- Draw a door $x=180, y=150, \text{width}=40, \text{height}=100$
- Draw a roof using a triangle left corner 50,50, the height of the roof is 50
- Draw a sun using a circle $x=320, y=80, \text{radius} = 30$



Video

```
<video id="video1" width="420"  
src="http:www.w3schools.com/html/mov_bbb.mp4"></video>
```

- controls
- autoplay
- muted

תרגיל

עליכם לישם תוכנה שמנגנת סרט ומאפשרת למשתמש לצלם תמונות מהסרט כאשר הוא רץ,
התמונות יוצגו למשתמש לצד הסרט כמו בתמונה המתוארת.
בכל שורה יהיה מקום ל-5 תמונות וסה"כ ל-20.



שלד פתרון

```
<head>
<style>
    canvas {border: 1px solid black;}
</style>
</head>
<body>
    <video id="video1" width="250" height="200" controls src="http://www.w3schools.com/html/mov_bbb.mp4">
    </video>
    <canvas id="myCanvas" width="250" height="200"></canvas>
    <br />
    <input type="button" value="pic" onclick="pic()" />
    <script>
        var vid = document.getElementById("video1");
        var can = document.getElementById("myCanvas");
        var ctx = can.getContext("2d");
        var x = 0;
        var y = 0;
        function pic() {
            ctx.drawImage(vid, x, y, 50, 50);
        }
    </script>
</body>
```

שאלות?

Fetch API

API

Application Programming Interface

סט של כללים וכליים המאפשרים לתוכנות שונות לדבר זו עם זו ולחולוק מידע
גם אם הן מפותחות על ידי ארגונים שונים וגם אם הן רצות על פלטפורמות
שונות.

Web API Basics

- URL שהAPI חושף, כל endpoint מיצג פונקציה שהAPI יבצע
- XML , JSON - Request and Response Format
- הרבה פעמים יש צורך שרק משתמשים מורשים – Authentication and Authorization
- יכולו לגשת לAPI. לצורך כך משתמשים ב Token או API Key

Fetch API

- javascript מאפשר לנו לבצע קריאות HTTP ב Fetch api
- כל הבראוזרים הפופולריים כיום תומכים ב fetch api
- Fetch api מספק פונקציה גלובלית שניתן להשתמש בה הנקראת `fetch`, והיא דואגת להחזיר resources מה server

Syntax

```
const response = fetch(URL [, options])
```

URL : a URL object that represents the path of the resource to be fetched

Options(optional): Any further options such as:

- Method: The request method is either GET POST PUT DELETE.
- Headers : for example 'Content-Type': 'application/json'
- Body: json string
- Mode
- Credentials
- Cache

How fetch() works

```
<script>
fetch('url')
  .then(response => {
    //handle response
    return response.json()
  })
  .then(data => {
    //handle data
  })
  .catch(error => {
    //handle error
  });
</script>
```

Calling an External API Using Fetch

- **נשתמש ב API חינמי הנקרא JSONPlaceholder API**
- **אינו דורש אימות משתמש או API key**
- <https://jsonplaceholder.typicode.com/guide/>

Fetch

- נשתמש בפונקציה fetch של javascript על מנת לפנות ל אPI חיצוני שיחזיר נתונים,
- את הנתונים נדפיס לוג

```
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    // Check if the response is successful
    return response.json(); // Parse the JSON from the response
  })
  .then(data => {
    // Use the data from the response
    console.log(data);
  })
  .catch(error => {
    // Handle any errors
    console.error('Error:', error);
  });
};
```

Fetch with error handling

```
// Fetch data from the JSONPlaceholder API
fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => {
  if (!response.ok) {

    throw new Error(`HTTP error! Status: ${response.status}`);
  }
  // Return the response data as JSON
  return response.json();
})
.then(data => {
  // Handle the JSON data
  console.log('Fetched Data:', data);
  // Example: Display the titles of the posts
  data.forEach(post => {
    console.log(`Post ${post.id}: ${post.title}`);
  });
})
.catch(error => {
  console.error('There was a problem with the fetch operation:', error);
});
```

Getting a resource

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
      .then(response) => response.json()
      .then(json) => console.log(json)
      .catch(error => {
        console.error('There was a problem with the fetch operation:', error);
      });
    
```

Getting resources

```
fetch('https://jsonplaceholder.typicode.com/posts')

    .then(response) => response.json()

        .then(json) => console.log(json)

            .catch(error => {

console.error('There was a problem with the fetch operation:', error);

});
```

```
<body>
<h1>Post Titles</h1>
<ul id="post-list"></ul>
<script>
// JavaScript code goes here
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    return response.json();
})
.then(data => {
  console.log(data);
  const postList = document.getElementById('post-list');
  data.forEach(post => {
    const listItem = document.createElement('li');
    listItem.innerText = `Post ${post.id}: ${post.title}`;
    postList.appendChild(listItem);
  });
})
.catch(error => {
  console.error('Fetch error:', error);
});
</script>
</body>
```

Sending a resource

כאשר נרצה לשלוח אובייקט ל API נשתמש בקריאה ה regular
אך במקרים אלו, יש להגדיר אובייקט **אפשרויות (options)** שנשלח בצד- ליידע את ה API שהוא
מעבירים נתוניים (שאינם חלק מ Query String)

- **אובייקט ה - options כולל מידע חשוב כפוף:**
`fetch(URL, options[, options])`
- שיטת הבקשה (method)
- סוג הנתוניים (headers)
- תוכן הבקשה (body)

הוספה של אובייקט חדש – נשלח אותו כפרמטר ב body של ה fetch

```
<body>
<script>
fetch('https://jsonplaceholder.typicode.com/posts',
{
    method: 'POST',
    body: JSON.stringify({
        title: 'foo',
        body: 'bar',
        userId: 1,
    }),
    headers: {
        'Content-type': 'application/json; charset=UTF-8',
    },
})
.then((response) => response.json())
.then((json) => console.log(json));
</script>
</body>
```

שאלות?

דוגמא

:Endpoint •

<https://jsonplaceholder.typicode.com/users/2> •

- נבצע קרייה ל API מעלה אשר מחזיר יוזר לפי id
- הדפס לוג
- שנה ב url את 2 למספר אחר

```
<body>
User id=1
<script>
fetch('https://jsonplaceholder.typicode.com/users/2')
    .then(response => {
        if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
        }
        return response.json();
    })
    .then(user => {
        console.log(user);
        console.log(`Name: ${user.name}`);
        console.log(`Username: ${user.username}`);
        console.log(`Email: ${user.email}`);
        console.log(
`Address: ${user.address.street}, ${user.address.suite}, ${user.address.city},
${user.address.zipcode}`);
    })
    .catch(error => {
        console.error('Error fetching user information:', error);
    });
</script>
</body>
```

User id=2

```
<body>
  <script>
    fetch('https://jsonplaceholder.typicode.com/users/1')
      .then(response => {
        if (!response.ok) {
          throw new Error(`HTTP error! Status: ${response.status}`);
        }
        return response.json();
      })
      .then(user => {
        console.log(user);
        console.log(`Name: ${user.name}`);
        console.log(`Username: ${user.username}`);
        console.log(`Email: ${user.email}`);
        console.log(
          `Address: ${user.address.street}, ${user.address.suite}, ${user.address.city},
          ${user.address.zipcode}`
        );
      })
      .catch(error => {
        console.error('Error fetching user information:', error);
      });
  </script>
</body>
```

The End