שיעור 2 Svcollege css

הקדמה chat

### \*\*תגיות וקונבנציות ב-CSS\*\*

ב-CSS משתמשים במגוון \*\*תגיות (Selectors)\*\* וקונבנציות לשמות של \*\*מאפיינים (Properties)\*\* וערכים. הנה רשימה מסודרת של הדברים המרכזיים:

---

## \*\*1. בוחרים אלמנטים (Selectors)\*\*

כדי לעצב אלמנט ב-HTML, משתמשים בבחירה שלו:

### \*\*בחירת תגיות HTML\*\*

```css

h1 {

color: blue;

}

p {

font-size: 18px;

}

```

👉 משפיע על \*\*כל\*\* האלמנטים מסוג `h1` או `p` בדף.

### \*\*בחירת מחלקות (Class)\*\*

```css

.title {

font-weight: bold;

}

```

👉 ייעודי לאלמנטים שיש להם `class="title"` ב-HTML.

### \*\*בחירת מזהים (ID)\*\*

```css

#main-heading {

text-align: center;

}

```

👉 משפיע \*\*רק על אלמנט אחד\*\* עם `id="main-heading"`.

### \*\*בחירת אלמנטים בתוך אחרים\*\*

```css

div p {

color: red;

}

```

👉 ישפיע על \*\*כל הפסקאות (`p`) שנמצאות בתוך `div`\*\*.

### \*\*בחירת ילדים ישירים בלבד\*\*

```css

div > p {

color: green;

}

```

👉 ישפיע \*\*רק על פסקאות (`p`) שהן ילדים ישירים של `div`\*\*.

### \*\*בחירת אלמנט צמוד (Sibling Selector)\*\*

```css

h1 + p {

color: gray;

}

```

👉 הפסקה הראשונה אחרי `h1` תהיה אפורה.

---

## \*\*2. קונבנציות של שמות מאפיינים (Properties)\*\*

כל המאפיינים ב-CSS כתובים בפורמט \*\*`property: value;`\*\*.

לדוגמה:

```css

color: red;

font-size: 20px;

```

### \*\*קטגוריות עיקריות של מאפיינים:\*\*

#### \*\*🖌 צבעים וסטיילים\*\*

| מאפיין | תיאור |

|---------|-------------|

| `color` | צבע הטקסט |

| `background-color` | צבע הרקע |

| `opacity` | שקיפות (0 עד 1) |

| `border` | גבול מסביב לאלמנט |

| `box-shadow` | הצללה סביב האלמנט |

#### \*\*🔠 טקסט ופונטים\*\*

| מאפיין | תיאור |

|---------|-------------|

| `font-family` | סוג הגופן (Arial, sans-serif וכו') |

| `font-size` | גודל הגופן |

| `font-weight` | עובי הגופן (bold, normal, 100-900) |

| `text-align` | יישור טקסט (left, center, right) |

| `line-height` | גובה שורה (ריווח בין שורות) |

| `letter-spacing` | ריווח בין אותיות |

#### \*\*📐 גודל ומיקום\*\*

| מאפיין | תיאור |

|---------|-------------|

| `width` | רוחב האלמנט |

| `height` | גובה האלמנט |

| `max-width` / `max-height` | מקסימום גובה/רוחב |

| `min-width` / `min-height` | מינימום גובה/רוחב |

| `margin` | רווח חיצוני מסביב לאלמנט |

| `padding` | רווח פנימי בתוך האלמנט |

#### \*\*📍 מיקום ותצוגה\*\*

| מאפיין | תיאור |

|---------|-------------|

| `display` | סוג התצוגה (`block`, `inline`, `flex`, `grid`) |

| `position` | סוג המיקום (`relative`, `absolute`, `fixed`, `sticky`) |

| `top` / `left` / `right` / `bottom` | קביעת מיקום האלמנט |

| `z-index` | שכבתיות (מה יהיה מעל/מתחת) |

| `overflow` | איך להתמודד עם תוכן חורג (`hidden`, `scroll`, `auto`) |

#### \*\*🎨 עיצוב מתקדם\*\*

| מאפיין | תיאור |

|---------|-------------|

| `border-radius` | פינות מעוגלות |

| `box-shadow` | הצללה סביב האלמנט |

| `text-shadow` | הצללה על טקסט |

| `filter` | פילטרים כמו `blur` ו-`grayscale` |

---

## \*\*3. יחידות מידה נפוצות\*\*

ב-CSS יש כמה יחידות מידה:

### \*\*יחידות מוחלטות\*\*

| יחידה | תיאור |

|---------|-------------|

| `px` | פיקסלים (הנפוצה ביותר) |

| `cm`, `mm`, `in` | ס"מ, מ"מ, אינצ'ים |

### \*\*יחידות יחסיות\*\*

| יחידה | תיאור |

|---------|-------------|

| `%` | אחוזים מגודל האלמנט ההורה |

| `em` | יחסית לגודל הגופן של האלמנט |

| `rem` | יחסית לגודל הגופן הבסיסי של הדף |

| `vw`, `vh` | אחוז מרוחב/גובה המסך |

---

## \*\*4. אנימציות ומעברים\*\*

CSS מאפשר ליצור אנימציות ללא JavaScript!

### \*\*מעבר (Transition)\*\*

```css

button {

background-color: blue;

transition: background-color 0.3s ease-in-out;

}

button:hover {

background-color: red;

}

```

👉 הצבע ישתנה בכיווץ חלק תוך 0.3 שניות.

### \*\*אנימציה מלאה (Animation)\*\*

```css

@keyframes slideIn {

from {

transform: translateX(-100%);

}

to {

transform: translateX(0);

}

}

div {

animation: slideIn 1s ease-in-out;

}

```

👉 האלמנט יחליק פנימה מהצד.

---

## \*\*5. שימוש במשתנים ב-CSS (CSS Variables)\*\*

ב-CSS ניתן להגדיר משתנים כדי לשפר את ניהול הצבעים והגופנים:

```css

:root {

--primary-color: blue;

--font-size: 18px;

}

h1 {

color: var(--primary-color);

font-size: var(--font-size);

}

```

👉 כל שינוי בצבע או גודל יבוצע רק במקום אחד (`:root`), מה שמקל על התחזוקה.

---

## \*\*סיכום\*\*

🔹 ב-CSS יש \*\*בוחרים (Selectors)\*\* כדי להחיל עיצובים על אלמנטים.

🔹 קיימים \*\*מאפיינים (Properties)\*\* לקביעת צבעים, טקסט, גודל, מיקום וכו'.

🔹 אפשר להשתמש \*\*ביחידות מידה\*\* כמו `px`, `%`, `vw` ו-`em`.

🔹 ניתן ליצור \*\*מעברים (Transitions)\*\* ואפילו \*\*אנימציות (Animations)\*\*!

🚀 \*\*השלב הבא:\*\* ללמוד \*\*Flexbox\*\* ו-\*\*CSS Grid\*\* כדי לשלוט על פריסת האלמנטים בדף!

## \*\*הקדמה ל-JavaScript\*\*

\*\*JavaScript (בקיצור JS)\*\* היא שפת תכנות המשמשת כדי להוסיף \*\*אינטראקטיביות\*\* ו\*\*דינמיות\*\* לאתרים.

בעזרתה ניתן לגרום לאלמנטים ב-HTML להגיב ללחיצות, עכבר, מקלדת, טעינת דף ועוד.

---

## \*\*🔹 למה משמשת JavaScript?\*\*

✅ \*\*שינוי תוכן בדף\*\* (למשל, עדכון טקסט באופן דינמי)

✅ \*\*תגובה לפעולות המשתמש\*\* (לחיצות כפתור, תנועת עכבר, הקלדה)

✅ \*\*עבודה עם טפסים\*\* (בדיקות נתונים לפני שליחה)

✅ \*\*יצירת אנימציות\*\* ושינוי עיצוב ב-CSS

✅ \*\*שליחת וקבלת נתונים מהשרת (API, Fetch, AJAX)\*\*

✅ \*\*בניית אפליקציות שלמות (Front-End + Back-End עם Node.js)\*\*

---

## \*\*📌 חיבור JavaScript לדף HTML\*\*

כדי להפעיל JavaScript בדף, ניתן להכניס את הקוד בשלוש דרכים:

1️⃣ \*\*בתוך תג `<script>` ישירות ב-HTML\*\*

```html

<script>

alert("שלום עולם!");

</script>

```

2️⃣ \*\*בתוך קובץ חיצוני (`script.js`) ולחבר אותו ל-HTML\*\*

```html

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

```

3️⃣ \*\*בתוך תגית HTML באמצעות `onclick`\*\*

```html

<button onclick="alert('לחצת על הכפתור!')">לחץ כאן</button>

```

---

## \*\*🔹 משתנים ב-JavaScript\*\*

ב-JavaScript יש שלוש דרכים להגדיר משתנים:

```js

var x = 10; // ישן, לא מומלץ

let y = 20; // משתנה שניתן לשנות

const z = 30; // משתנה קבוע (אי אפשר לשנות)

```

🔹 \*\*`let`\*\* – מגדיר משתנה שיכול להשתנות.

🔹 \*\*`const`\*\* – מגדיר משתנה קבוע (לא ניתן לשינוי).

🔹 \*\*`var`\*\* – ישן, פחות מומלץ בגלל בעיות התחום שלו.

---

## \*\*🔹 סוגי נתונים (Data Types)\*\*

| סוג נתון | דוגמה |

|------------|------------|

| מספרים (Number) | `let age = 25;` |

| טקסט (String) | `let name = "דני";` |

| בוליאני (Boolean) | `let isStudent = true;` |

| מערך (Array) | `let colors = ["אדום", "כחול", "ירוק"];` |

| אובייקט (Object) | `let person = {name: "יוסי", age: 30};` |

דוגמא:

```js

let age = 25;

let name = "רוני";

let isStudent = true;

let colors = ["אדום", "כחול", "ירוק"];

let person = { name: "יוסי", age: 30 };

console.log(name); // מדפיס "רוני"

console.log(colors[0]); // מדפיס "אדום"

console.log(person.age); // מדפיס 30

```

---

## \*\*🔹 פונקציות (Functions)\*\*

פונקציה היא קטע קוד שניתן להפעיל שוב ושוב.

```js

function sayHello() {

console.log("שלום עולם!");

}

sayHello(); // מפעיל את הפונקציה

```

פונקציה עם פרמטרים:

```js

function greet(name) {

console.log("שלום " + name + "!");

}

greet("דני"); // מדפיס: שלום דני!

```

---

## \*\*🔹 תנאים (If-Else)\*\*

```js

let age = 18;

if (age >= 18) {

console.log("אתה מבוגר");

} else {

console.log("אתה קטין");

}

```

---

## \*\*🔹 לולאות (Loops)\*\*

### \*\*לולאת `for`\*\*

```js

for (let i = 0; i < 5; i++) {

console.log("חזרה מספר " + i);

}

```

### \*\*לולאת `while`\*\*

```js

let count = 0;

while (count < 3) {

console.log("ספירה: " + count);

count++;

}

```

---

## \*\*🔹 אירועים (Events)\*\*

JavaScript מאפשר להגיב לפעולות המשתמש, כמו לחיצות עכבר או הקלדה.

```html

<button onclick="changeText()">לחץ עליי</button>

<p id="text">טקסט רגיל</p>

<script>

function changeText() {

document.getElementById("text").innerHTML = "הטקסט השתנה!";

}

</script>

```

🔹 כשמשתמש לוחץ על הכפתור – הפסקה משתנה.

---

## \*\*🔹 עבודה עם ה-DOM (Document Object Model)\*\*

🔹 \*\*DOM\*\* הוא מבנה של כל ה-HTML בדף. JavaScript מאפשר לשנות אותו.

```js

document.getElementById("demo").innerHTML = "טקסט חדש!";

document.querySelector("h1").style.color = "red";

```

- `document.getElementById("id")` – בחירת אלמנט לפי מזהה (`id`).

- `document.querySelector("tag/class/#id")` – בחירת אלמנט לפי תגית, מחלקה (`class`) או מזהה (`id`).

---

## \*\*🔹 טיפול בטפסים\*\*

```html

<input type="text" id="username">

<button onclick="showName()">הצג שם</button>

<p id="result"></p>

<script>

function showName() {

let name = document.getElementById("username").value;

document.getElementById("result").innerHTML = "שלום " + name;

}

</script>

```

🔹 לוקח את השם מהשדה ומציג אותו.

---

## \*\*🔹 Fetch API - קבלת נתונים מהשרת\*\*

```js

fetch("https://jsonplaceholder.typicode.com/users")

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error("שגיאה: ", error));

```

🔹 מבצע בקשה לשרת ומחזיר נתונים בפורמט JSON.

---

## \*\*🔹 סיכום\*\*

- \*\*JavaScript\*\* מוסיפה \*\*אינטראקטיביות\*\* לאתר.

- ניתן להשתמש ב\*\*משתנים, פונקציות, תנאים ולולאות\*\*.

- אפשר \*\*לשנות תוכן בעמוד (DOM)\*\* בעזרת `document.getElementById`.

- JavaScript \*\*מתקשרת עם השרת\*\* (API, Fetch).

- מאפשרת \*\*אירועים\*\*, כמו לחיצות והקלדה.

🚀 \*\*השלב הבא:\*\* ללמוד על \*\*ES6 (arrow functions, promises, async/await)\*\* ולהתחיל לעבוד עם \*\*React\*\*!

שיעור 2 בן

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Document</title>

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

<link rel="stylesheet" href="main2.css">

</head>

<body>

<ol>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

</ol>

<ul>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

<li> test1 1</li>

</ul>

</body>

</html>

יצרנו טבלה מסוימת

ועשינו לה בcss קווים

ol,li,ul{

border: 2px solid black ;

}

הראנו איך לעגל תמונות בcss

להוסיף בורדר רדיוס 2 px solid black

img{

width: 100px;

border-radius: 4ch;

}

הסברנו למה צריך div כדי לסדר את המסך התגית <div> היא קיצור של "division" (חלוקה). היא משמשת ליצירת אזורים/קבוצות של תוכן ב-HTML. <div> אינו משפיע על העיצוב בפני עצמו, אבל מאפשר לקבץ אלמנטים יחד ולעצב אותם עם CSS.

🔹 למה <div> עוזר פה?

✔ אפשר לעצב את כל מה שבתוך ה-div יחד.

✔ ניתן להוסיף רקע, גבולות, ריווח ויישור.

div.article-kontiner{

border: 2px solid black;

}

כך ניראה בcss והוספנו תנושא של רייט טו לפט <html lang="en" dir="rtl">

הדיבים יתנו לנו אופציה תת מרווחים בין האלמנטים לכן ניצור את הבוקס מודל מה שזה אומר כל דבר יש בו קופסה לכן נוסיף פדדינג שזה אומר לתת מרווחים בתוך הקופסה וגם את הקופסה עצמה אפשר להוזיז קודם כל יש אפשרות לרווח את הדיב עצמ ו ויש עוד אופציה לרווח את הדברים בפנים לדוגמא טקסט

ויש גם מארגין שהוא מרחיק את הקצוות של הקופסא מהמשתמש לא משתמשים במארגין בדרך כלל

פדיינג בלוק זה למעלה למטה ואם אני רוצה מצד שמאל ומצד ימין ינלייין

להוסיף טראנזישין לכפתור והוא אומר כמה זמן יקח לדברים להשתנות

טראנספורם שאומר

איך לשנות את המצביע כשהוא על הכפתור למצביע נוסיף כורסור פוינטר cursor: pointer/grab; כמובן בנוסיף בהווור

button{

background-color: #420bc2;

color: aliceblue;

border: none;

padding: 6px,20px,6px,20px;

border-radius: 8px;

transition: all 10ms;

}

button:hover{

background-color: #8c6cd4;

transform: scale(1.2);

cursor: pointer ;

}

button:active {

transform: scale(0.95) translateY(2px);

box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.2);

}

אז ראינו שיש אפשרות להקטין ולהגדיל את גודל הכפתור ואת אפשרות הלחיצה עליו ונעשה את זה באקטיב מכיון שאנחנו רוצים ליסינר שישנה ברגע שזה קורה יש עוד פונקציה שקוראים לה פוקוס שהיא רק מתקבעת על זה

איך להעביר את כל המשתנים קונטרול די ואז משנים אותם ביחד

סיכום – מתי להשתמש בכל position?

relative – אם צריך להזיז אלמנט בלי לשבור את המבנה.

✅ absolute – כשצריך למקם אלמנט באופן מוחלט מעל הורה כלשהו.

✅ fixed – כשצריך לנעול אלמנט על המסך (כמו תפריט קבוע).

✅ sticky – כשצריך לשמור כותרת / תפריט בראש המסך בעת גלילה.

כשנירצה להוסיף את הכפתור על משהו מסוים נגדיר לו פוזישן ובתוך הפוזישן נגדיר מה נירצה שיעשה

אם נירצה שהתפריט ישאר כל הזמן למעלה גם אם אנחנו נירד לסוף האתר אזי נוסיף את המושג סטיקי בפוזישן

מה זה position: sticky; ב-CSS?

🔹 sticky הוא מצב מיוחד של position שמשלב בין relative ל-fixed.

🔹 האלמנט מתנהג כרגיל (relative) עד שמגיע לנקודה מסוימת בגלילה – ואז נשאר דבוק למקום כמו fixed.

🔹 הוא נצמד רק בתוך הקונטיינר שלו ולא מחוץ לו.

למה sticky לא עובד לפעמים?

❌ אין top, bottom, left או right → חייב להגדיר לפחות אחד מהם.

❌ האלמנט נמצא בתוך overflow: hidden או overflow: auto → צריך להסיר את זה מהקונטיינר.

❌ הקונטיינר שלו קטן מדי → ברגע שהוא יוצא מהקונטיינר, הוא מפסיק להיות sticky

מתי להשתמש ב-sticky?

✅ תפריטים שנשארים בראש רק כשגוללים.

✅ טבלאות עם כותרות נצמדות.

✅ עיצובי ניווט פנימיים באתרים (למשל, רשימות של נושאים בצד).

בוא נעבור על אנימציות

יש אפשרות להוסיף לבד @keyframs ניתן לה שם באונס לדוגמא סוגרים מסולסלים ושם נחלק את הזמנים לאחוזים וניפתח לכל זמן סוגרים