**תרגיל בית 6**

1.

2. מדוע כאשר מוסיפים border לאלמנט הורה הוא כולל בתוכו את הmargin של הילד?

יש תופעה שנקראת "קריסת margin " - שאומרת שכאשר יש אלמנט אב שלבן שלו יש margin, האב לא כולל בשטחו את ה margin של הבן וה margin של הבן כביכול עף החוצה והוא נראה כ margin של האב וזה כביכול אשליה.

אבל- ברגע שנוסיף לאב border או padding – הוא הופך להיות block formatting context באופן חלקי , ואז הקריסה לא מתבצעת , והאב מכיל את ה margin של הילדים, ומגדיל את שטחו.

הוספת border או padding הופכת את ההקשר הגרפי של האב, ומפסיקה את הקריסה של הmargin החוצה – זו התנהגות רשמית לפי תקן CSS..

טיפ למניעה קבועה של Margin Collapse:

div {

display: flow-root;

}

זה יוצר **block formatting context חדש** ומונע לחלוטין קריסת מרג'ינים של ילדים.

6.  .align-content: center;

7. אמת המידה fr היא יותר נכונה וריספונסיבית.

מה ההבדל בינהם?

% = אם אנחנו נותנים מידות באחוזים ברגע שאנחנו עוברים את ה100 אחוז הטבלה תצא מחוץ לגבולות של אלמנט האב ואז צריך להתחיל לחשב, ואת זה אנחנו רוצים למנוע. יש עוד צורה שאפילו שאנחנו לא עוברים את ה100 אחוז אם אנחנו מוסיפים רווחים בין השורות(gap ) זה מוסיף שטח והטבלה יוצאת מגבולות האלמנט.

Fr = "fraction"

יחידת ה-fr ב-CSS משמשת בעיקר כאשר עובדים עם קונטיינרים גמישים (flexbox) או רשתות (grid) כדי לקבוע את המידות יחסית לחלל הפנוי. הנה הסבר כיצד היא פועלת:

**איך זה עובד?**

1. **חלוקת חלל**: כאשר אתה מגדיר מידות בעזרת יחידת fr, אתה למעשה מחלק את החלל הפנוי בין האלמנטים בקונטיינר. כל אלמנט שיקבל ערך fr יקבל חלק מהחלל בהתאם לערך שהוקצה לו.
2. **דוגמה לשימוש ב-Grid**: נניח שיש לנו קונטיינר Grid עם שלושה קולונות, ואנחנו רוצים שהקולונה הראשונה תתפוס 1fr, השנייה 2fr והשלישית 1fr: css .container { display: grid; grid-template-columns: 1fr 2fr 1fr; /\* 1/4, 1/2, 1/4 מהחלל \*/ }
3. **חלוקת החלל**: במקרה הזה:
   * הקולונה הראשונה תקבל רבע מהחלל.
   * הקולונה השנייה תקבל חצי מהחלל.
   * הקולונה השלישית תקבל רבע מהחלל.
4. **התאמה דינמית**: אם תוסיף או תסיר אלמנטים מהקונטיינר, יחידות ה-fr יתאימו אוטומטית כדי למנוע רווחים פנויים, והקולונות יתעדכנו בהתאם לחלל הפנוי.

**דוגמה מעשית:**

<div class="container">

<div class="item">קולונה 1</div>

<div class="item">קולונה 2</div>

<div class="item">קולונה 3</div>

</div>

.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr;

}

.item {

background-color: lightblue;

border: 1px solid blue;

padding: 10px;

}

במקרה הזה, הקולונה השנייה תהיה רחבה יותר משתי הקולונות האחרות.

8. **justify-content**

מזיז את **כל** טבלת הgrid או שורת הflex בחלל האב.

אפשרויות שונות כוללות:

* **flex-start**: המספרים יתחילו מהקצה השמאלי.
* **flex-end**: המספרים יסתיימו בקצה הימני.
* **center**: המספרים יתמקדו במרכז.
* **space-between**: חלוקה שווה של חללים בין המספרים.
* **space-around**: חלל סביב האלמנטים, כולל תחילת וסוף הקונטיינר.

**Justify-items**

מאפיין זה קובע כיצד האלמנטים בתוך קונטיינר Grid צריכים להתמקם בתוך תאי הגריד שלהם על ציר ה-X (האופקי). המאפיין משפיע על האלמנטים עצמם ולא על הקונטיינר. אפשרויות כוללות:

* **start**: מיקום האלמנט בצד השמאלי של התא.
* **end**: מיקום האלמנט בצד הימני של התא.
* **center**: מיקום האלמנט במרכז התא.
* **stretch**: האלמנט יתמלא לחלוטין את התא (זוהי ברירת המחדל).

**align-content**

מאפיין זה קובע כיצד השורות או העמודות בקונטיינר (Flex או Grid) מתמקמות בתוך החלל הפנוי על ציר ה-Y (האנכי). במילים אחרות, הוא משפיע על המרווח בין שורות או עמודות כאשר יש חלל פנוי בקונטיינר. אפשרויות שונות כוללות:

* **flex-start**: השורות יתחילו מהקצה העליון.
* **flex-end**: השורות יסתיימו בקצה התחתון.
* **center**: השורות יתמקדו במרכז.
* **space-between**: חלוקה שווה של חללים בין השורות.
* **space-around**: חלל סביב השורות.
* **stretch**: השורות יתמלאו לחלוטין את החלל הפנוי (זוהי ברירת המחדל).

**align-items**

מאפיין זה קובע כיצד האלמנטים בתוך קונטיינר (Flex או Grid) מתמקמים לאורך הציר האנכי (Y). הוא משפיע על המיקום של האלמנטים עצמם, ולא על הקונטיינר. אפשרויות שונות כוללות:

* **flex-start**: האלמנטים יתחילו מהקצה העליון.
* **flex-end**: האלמנטים יסתיימו בקצה התחתון.
* **center**: האלמנטים יתמקדו במרכז.
* **baseline**: האלמנטים יעמדו בקו הבסיס של הטקסט.
* **stretch**: האלמנטים יתמלאו לחלוטין את הגובה של הקונטיינר (זוהי ברירת המחדל).