ANIMATION-CSS

```
#id{
    animation: NAME Xs infinite||X
}
```

משך זמן מחזוריות

```
@keyframes NAME{
    from{attribute :value}
    to(attribute:value}
}
```

Media Queries

שימושי בכדי להפוך את האתר לרספונסיבי.
נוכל להגדיר שבטווחי רוחב שונים)מסכים שונים(האלמנטים יוצגו בצורה שונה.

קיימות 2 שיטות עבודה נפוצות:

Mobile First Approach - בניית האתר למובייל, ומשם כלפי מעלה למסכים גדולים יותר.
Desktop First Approach - בניית האתר למסך גדול, ומשם כלפי מטה למסכים קטנים יותר.

הגישה הראשונה נפוצה יותר היות וכניסה לאתרים מהמובייל יותר נפוצה כיום מכניסה לאתרים מהמחשב.

****היות ומדובר על רספונסיביות)עיצוב(זה יירשם ב CSS**

Media Queries

CSS Breakpoints: where & how many?

אין צורך לכתוב כללים לכל גודל מסך אפשרי, ניתן להצביע על ארבע קבוצות של גדלי מסך:

- קטנים וכולל 768 פיקסלים)סמארטפונים(
- גדולים מ 768 פיקסלים)מכשירים קטנים, טאבלטים(
- גדולים מ 992 פיקסלים)מסכים בינוניים)
- גדולים מ 1200 פיקסלים)מסכים גדולים)

Media Queries

Mobile First Approach

כאן יהיה עיצוב הבסיס.

```
// 768 העיצוב הדיפולטיבי ייבנה לגודל של מתחת

@media (min-width: 768px){
  // כאשר האתר בגודל מינימלי של 768 פיקסלים הוא יוצג לפי הכללים שרשומים כאן
}
@media (min-width: 1200px){
  // כאשר האתר בגודל מינימלי של 1200 פיקסלים הוא יוצג לפי הכללים שרשומים כאן
}
```

עבודה עם min אומרת שאנו עוברים ממסכים קטנים לגדולים.

Media Queries

Desktop First Approach

```
// העיצוב הדיפולטיבי ייבנה לגודל של מעל 1200

@media (max-width: 1200px){
  // כאשר האתר בגודל מקסימלי של 1200 פיקסלים הוא יוצג לפי הכללים שרשומים כאן
}
@media (max-width: 768px){
  // כאשר האתר בגודל מקסימלי של 768 פיקסלים הוא יוצג לפי הכללים שרשומים כאן
}
```

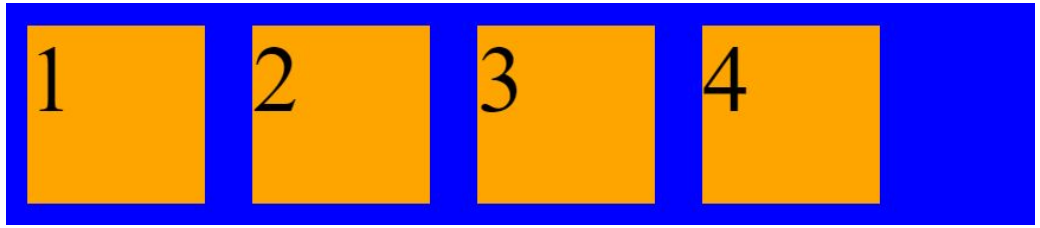
כאן יהיה עיצוב הבסיס.

עבודה עם max אומרת שאנו עוברים ממסכים גדולים לקטנים.

Flexbox Model

מודל "תיבה גמישה" מקל על תכנון ארכיטקטורת הפריסה של האלמנטים באתר כך שהוא יהיה רספונסיבי וגמיש.

- נגדיר אלמנט אב (container) שיכיל מספר אלמנטים בתוכו.
- ה container יקבל את הכלל display: flex ובכך האלמנטים שבתוכו יממשו את מודל flexbox.
- את הכללים שנראה בשקפים הבאים יש לתת ל container.



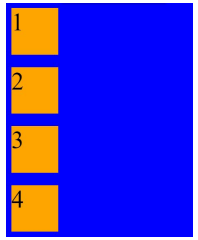

```
<div class='flexboxContainer'>
    <div class='flexItem'>1</div>
    <div class='flexItem'>2</div>
    <div class='flexItem'>3</div>
    <div class='flexItem'>4</div>
</div>
```

```
.flexboxContainer{
    display: flex;
    background-color: blue;
}
.flexItem{
    background-color: orange;
    width: 150px;
    height: 150px;
    margin: 20px;
    font-size: 80px;
}
```

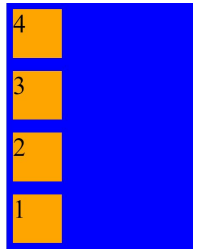
Flexbox Model

Flex-direction- קובע את הציר הראשי והאם הילדים יהיו בעמודה או שורה

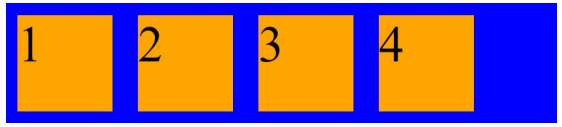
flex-direction: column



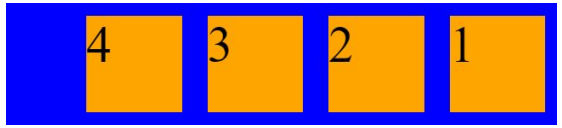
column-reverse



row



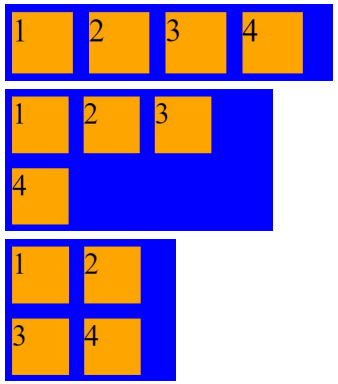
row-reverse



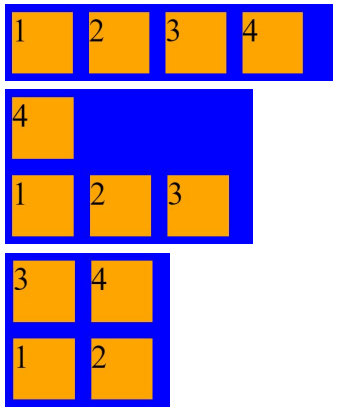
Flexbox Model

flex-wrap – מגדיר אם האלמנטים צריכים "לגלוש" או לא בהתאם לרוחב המסך.

flex-wrap: wrap



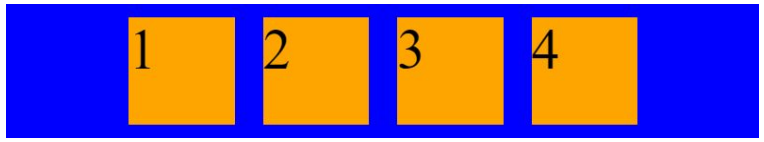
wrap-reverse



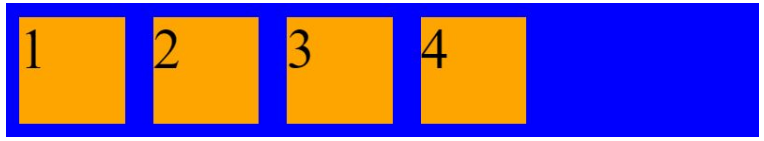
Flexbox Model

justify-content - מגדיר את מיקום הילדים על הציר הראשי

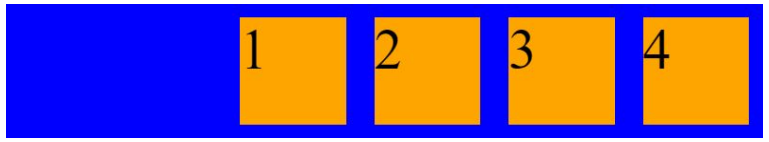
justify-content: center



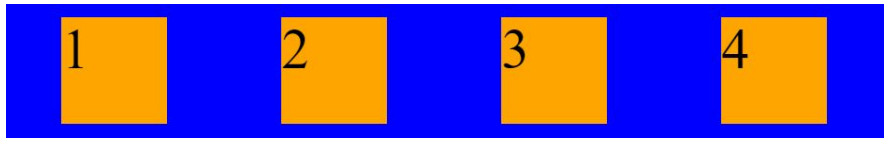
flex-start



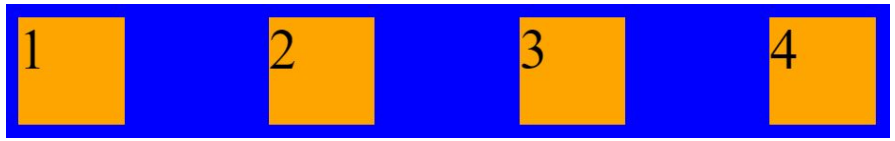
flex-end



space-around



space-between

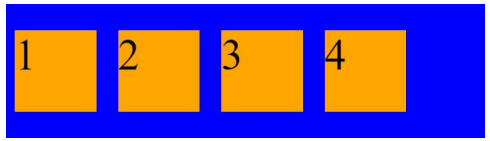


Flexbox Model

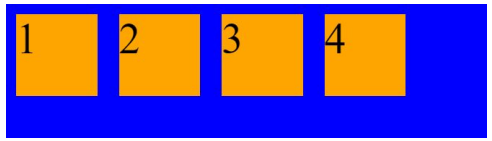
מגדיר את מיקום הילדים על הציר המשני – align-item

`align-items:`

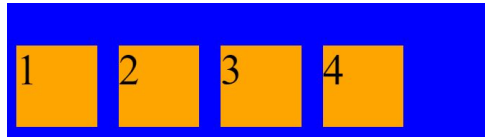
`center`



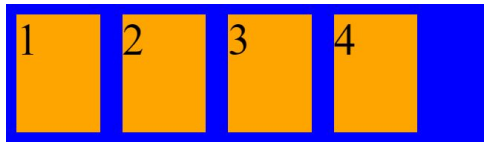
`flex-start`



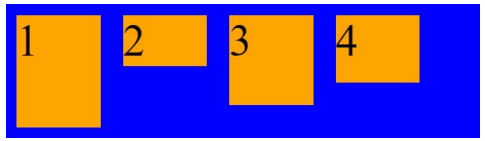
`flex-end`



`stretch`



`baseline`



Flexbox Model

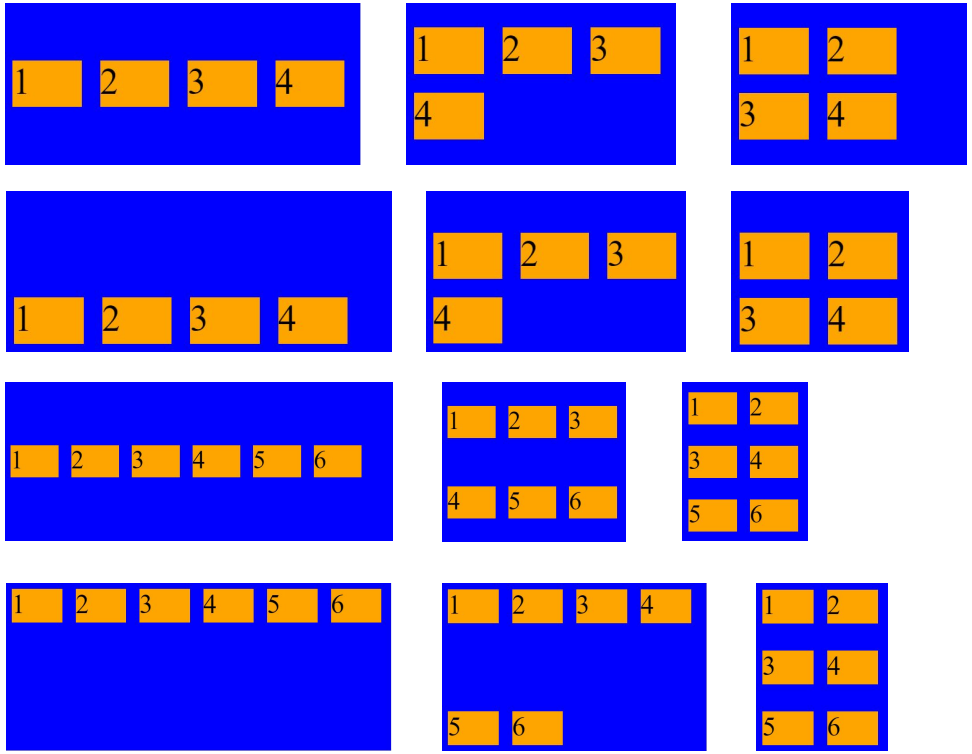
align-content – מגדיר את מיקום הילדים על הציר המשני, כולל שורות מרובות

align-content: center

flex-end

space-around

space-between



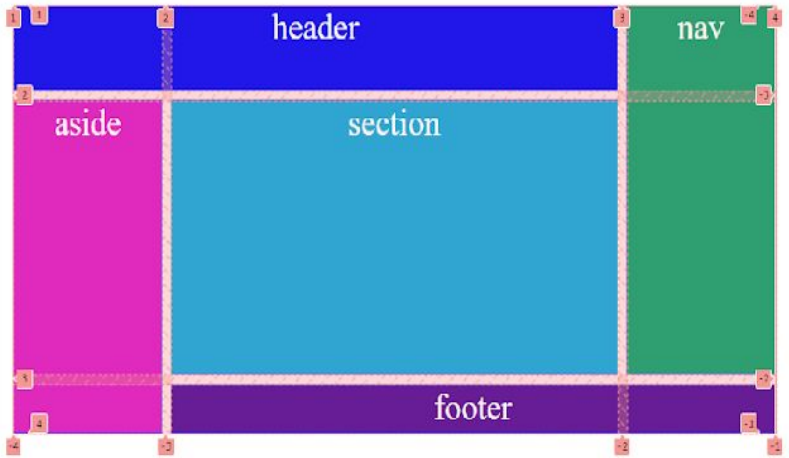
Grid Css

FlexBox	Grid
• מאפשר למקם אלמנטים במימד אחד (שורה/עמודה).	• מאפשר למקם אלמנטים בשני מימדים.
• קיימת תלות בסדר האלמנטים ב html .	• ללא תלות בסדר האלמנטים ב html.
• שימוש נפוץ בעבודה על אלמנט בודד (כדוגמת navbar).	• שימוש נפוץ בעיצוב וחלוקת כל העמוד.
• נתמך בכל הדפדפנים.	• נתמך בכל הדפדפנים.

דו מימדי

vs

חד מימדי



display : grid

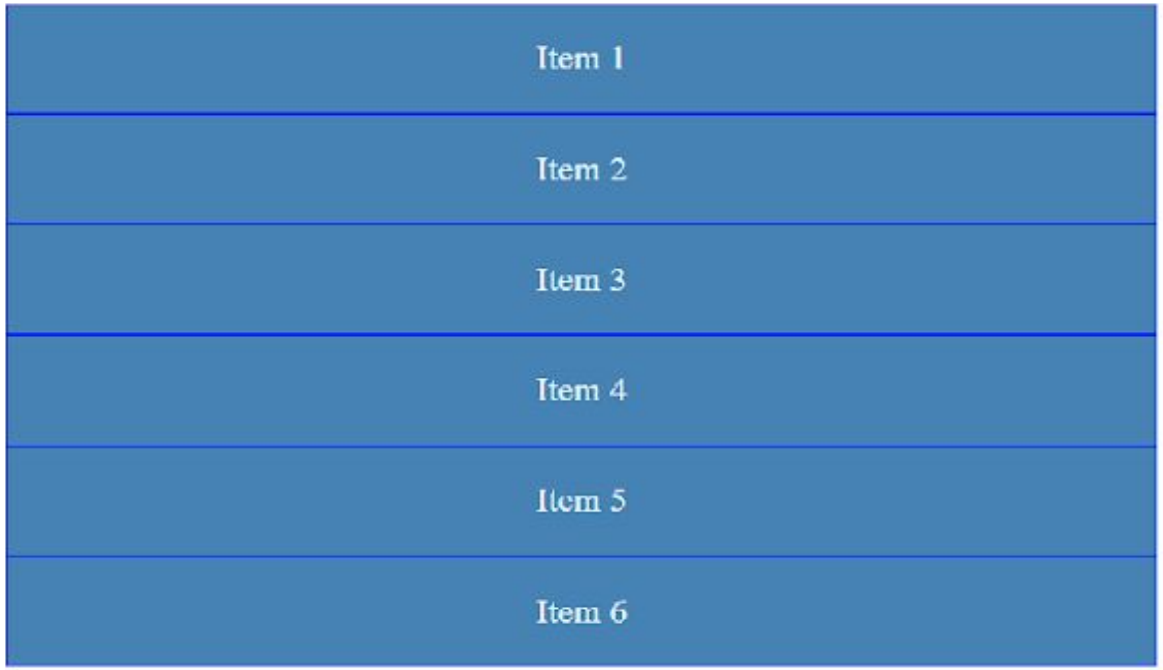
- בדומה לFlexBox עלינו להגדיר אלמנט "אב" (לדוגמא: container) שמכיל בתוכו מספר אלמנטים(Child elements).
- ה"אב" container יקבל **display : grid** ובכך נוכל לממש את היכולות של Grid על האלמנטים שבתוכו .

html

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

css

```css
.container {
    display: grid;
}

.item {
    background: steelblue;
    color: aliceblue;
    font-size: 20px;
    padding: 20px;
    border: 1px blue solid;
    text-align: center;
}
```



Grid - עימוד דו מימדי

** על מנת שנוכל לבצע עימוד דו מימדי עלינו להגדיר את השורות והעמודות שיחולו על האלמנטים (child elements).

```
.container {
    display: grid;
    grid-template-columns: 20% 30% 50%;
    grid-template-rows: 60px 60px;
}
```

	20%	30%	50%
px60	Item 1	Item 2	Item 3
px60	Item 4	Item 5	Item 6

```
.container {
    display: grid;
    grid-template-columns: 20% 30% 50%;
    grid-template-rows: 80px 40px;
}
```

px80	Item 1	Item 2	Item 3
px40	Item 4	Item 5	Item 6

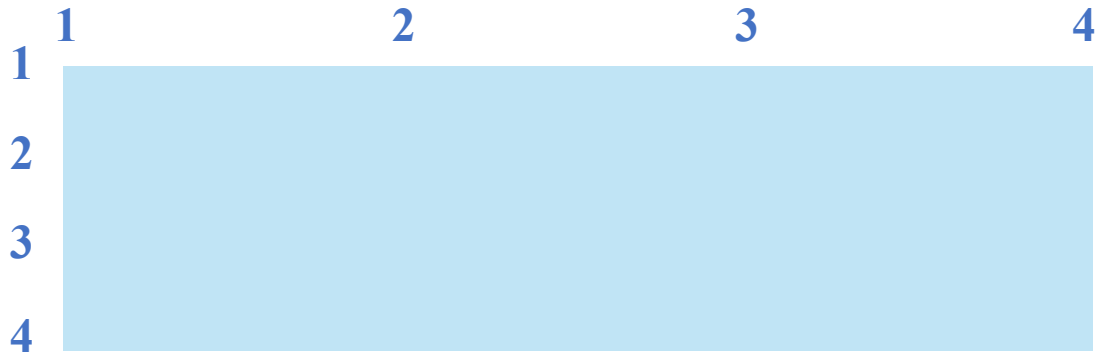
שליטה במיקום וגודל האלמנטים

( )nth-child

grid-column-start && grid-column-end

grid-row-start && grid-row-end

- היכן יתחיל והיכן יסיים (child)ניתן לקבוע עבור כל אלמנט (רשת) grid בעזרת ה.
- **( )nth-child** - מייצג את האלמנט (child) שנבחר (number || odd || even).
- מהיכן מתחיל? **grid-column-start** והיכן מסתיים? **grid-column-end.**
- מהיכן מתחיל? **grid-row-start** והיכן מסתיים? **grid-row-end.**
- אשר ממוספרים (רשת) grid נקודות הציון של ההתחלה והסיום יהיו בעזרת קווי המתאר של.
- קיימים גם בשורות וגם בעמודות grid קווי המתאר ב.



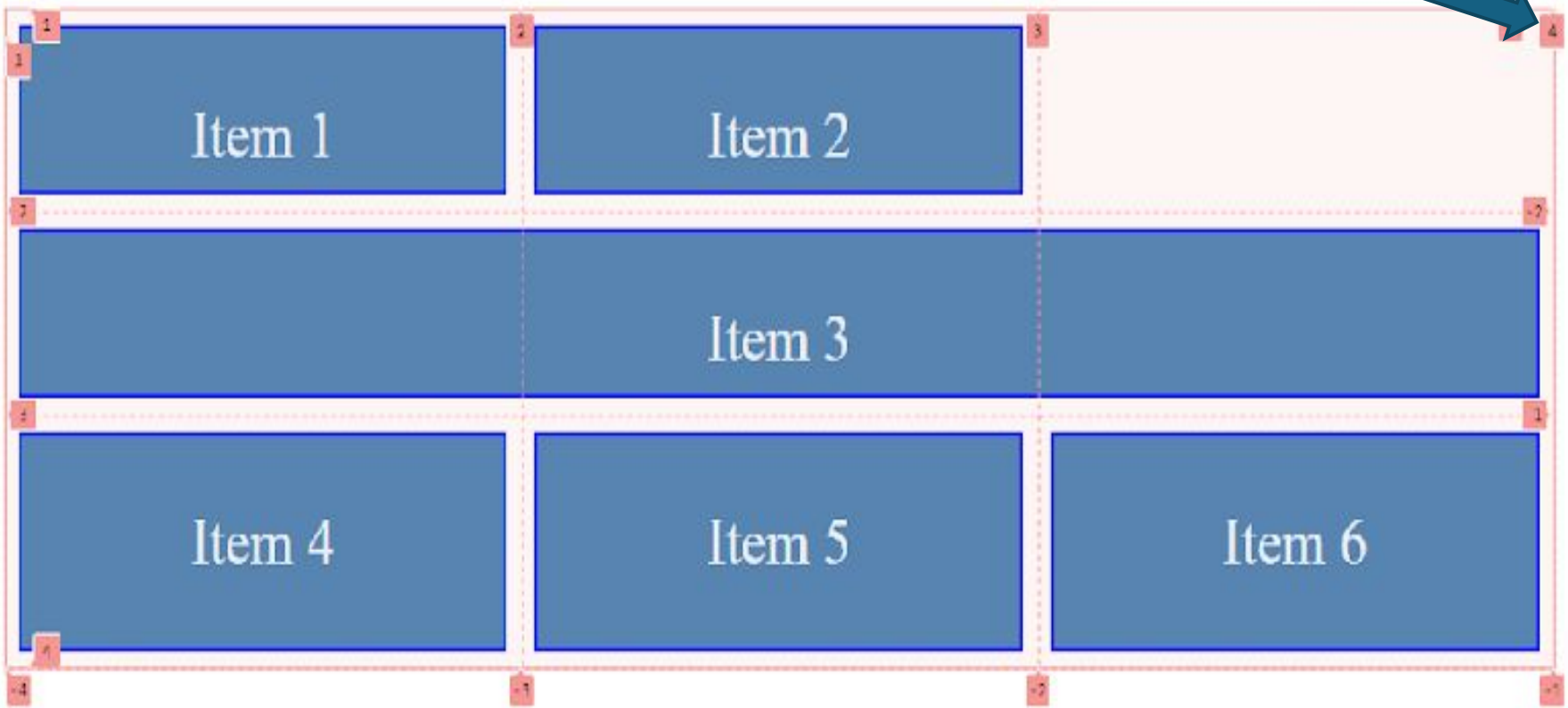
Grid

שליטה במיקום וגודל האלמנטים

( )nth-child

grid-column-start && grid-column-end

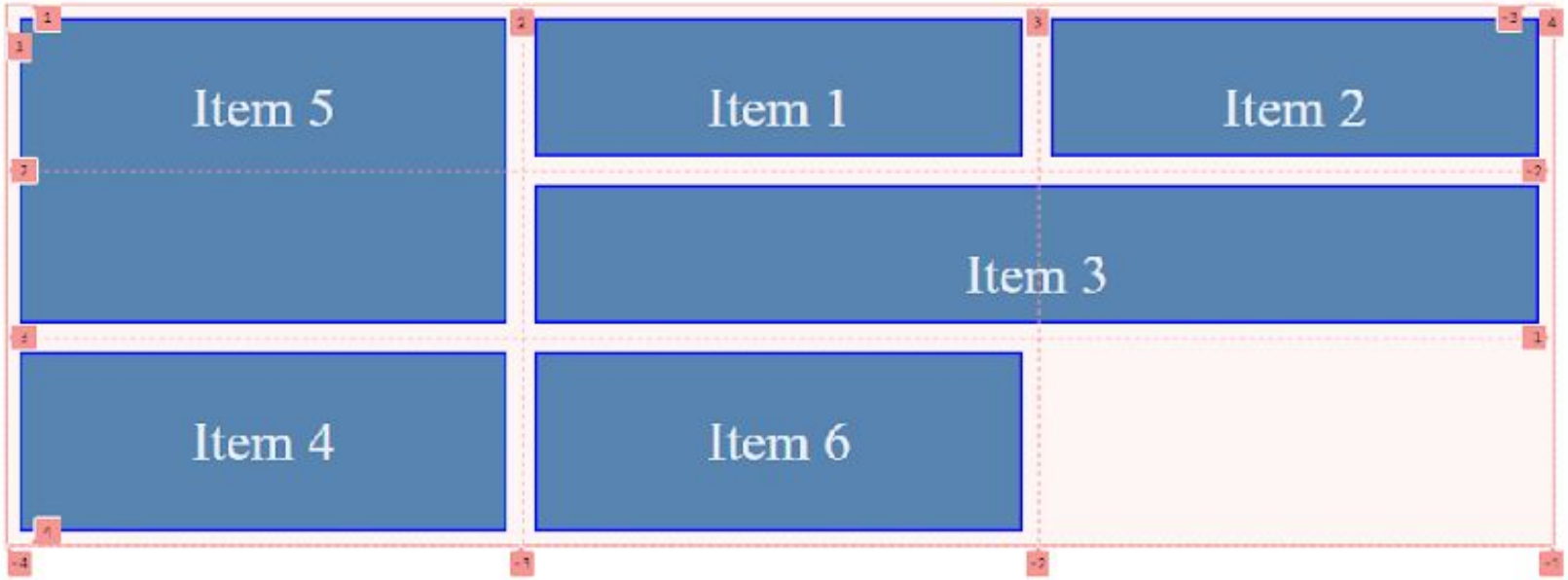
```
.container {
    display: grid;
    grid-template-columns: 33% 33% 33%;
    grid-template-rows: 60px 60px 60px;
}

.container>.item:nth-child(3) {
    grid-column-start: 1;
    grid-column-end: 4;
}

.item {
    background: steelblue;
    color: aliceblue;
    font-size: 20px;
    padding: 20px;
    border: 1px blue solid;
    text-align: center;
    margin: 5px;
}
```



שליטה במיקום וגודל האלמנטים

grid-column && grid-column
grid-row && grid-row
(number / number - דרך מקוצרת לכתיבה)

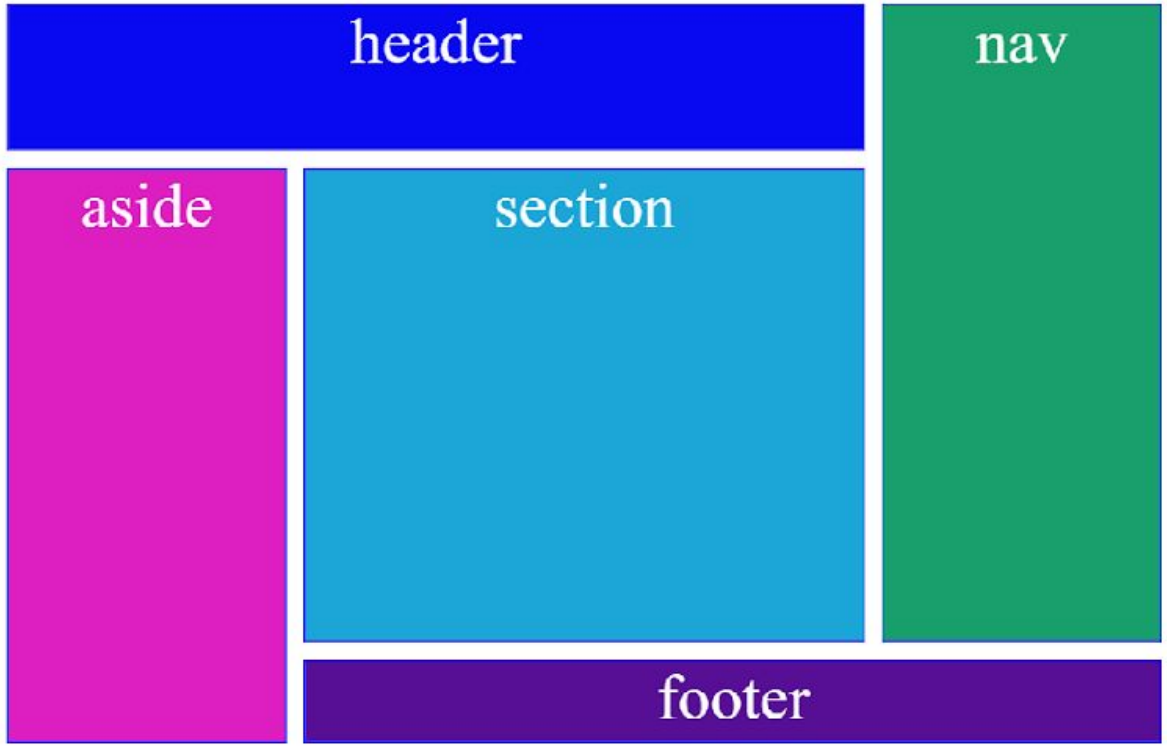
```
.container>.item:nth-child(5) {
    grid-row: 1/3;
    /* מתדיל ב 1 ומסיים ב 3 */
}

.container>.item:nth-child(3) {
    grid-column: 2/4;
    /* מתדיל ב 2 ומסיים ב 4 */
}

.item {
    background: steelblue;
    color: aliceblue;
    font-size: 20px;
    padding: 20px;
    border: 1px blue solid;
    text-align: center;
    margin: 5px;
}
```



grid-template-areas & gap

```
.container {
    padding: 10px;
    display: grid;
    grid-template-areas: "header header nav"
                         "aside section nav"
                         "aside footer footer";
    grid-template-rows: 70px 1fr 40px;
    grid-template-columns: 1fr 2fr 1fr;
    height: 350px;
    gap: 4px;
}


header,
nav,
aside,
section,
footer {
    border: 1px solid rgb(68, 68, 246);
    color: white;
    text-align: center;
    font-size: 30px;
}

header {
    grid-area: header;
    background-color: rgb(9, 9, 241);
}

nav {
    grid area: nav;
    background-color: rgb(25, 159, 108);
}

aside {
    grid area: aside;
    background-color: rgb(221, 10, 191);
}

section {
    grid-area: section;
    background-color: rgb(28, 166, 216);
}

footer {
    grid-area: footer;
    background-color: rgb(86, 15, 149);
}
```

- **ניתן ליצור בסיס למבנה העמוד grid-template-areas** בעזרת התכונה
- grid-area :name ראשית, עלינו להגדיר כל אלמנט על ידי שם שניתן לו
- **נגדיר את האזורים בעמוד על בסיס השמות שנתנו לאלמנטים grid-template-areas** שנית, בתכונה של
- לביצוע מרווח בין אלמנטים gap ניתן להשתמש בתכונה margin בדומה/תחליף ל.



סוף שיעור 3

שאלות??