

# 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

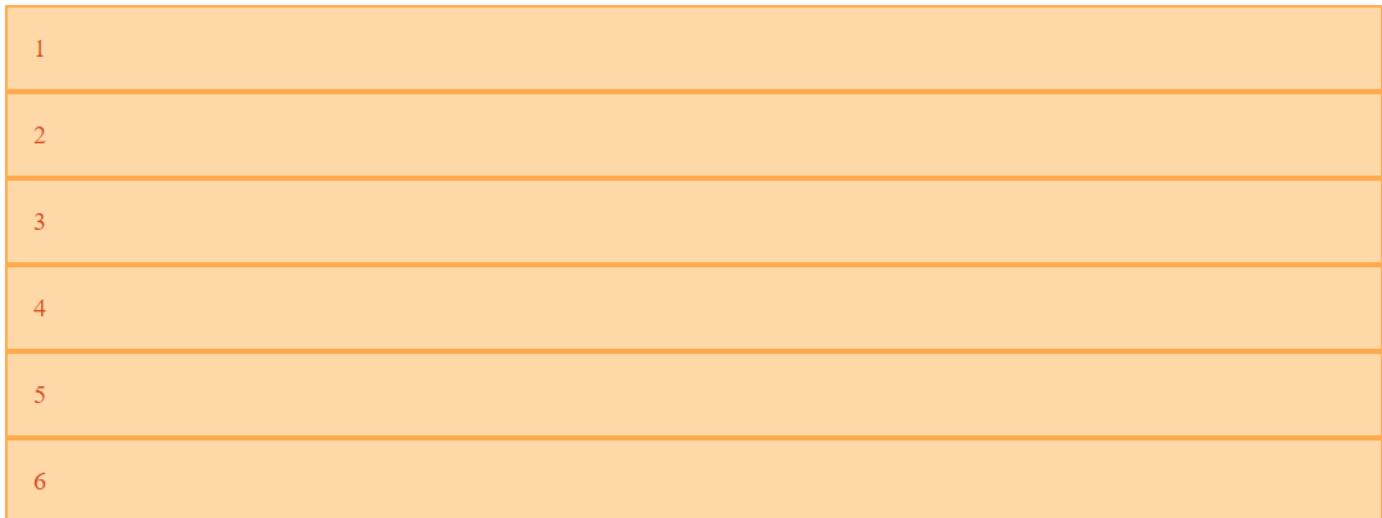
---

- **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>
```

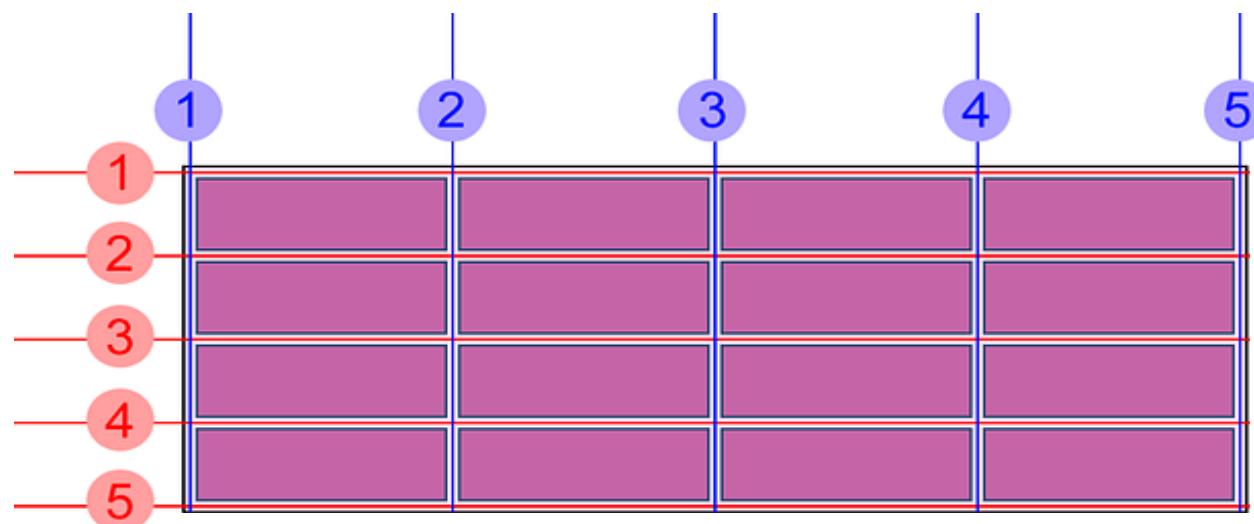
```
.wrapper {  
  display: grid;  
}
```



# 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;  
}
```

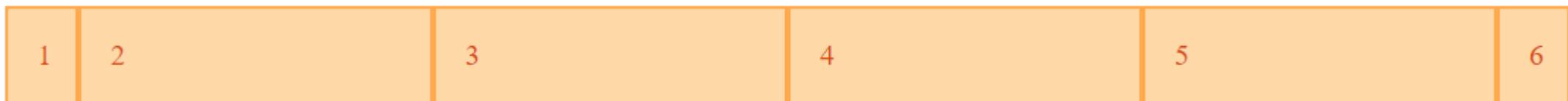
דוגמא

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 להגיד  
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

---

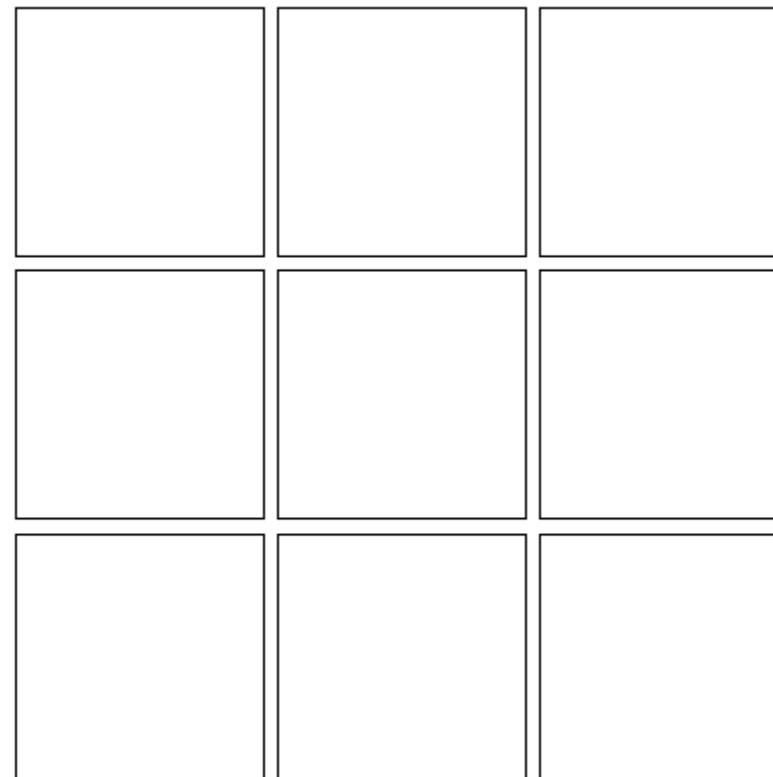
```
.wrapper {  
  display: grid;  
  grid-template-columns: 40px repeat(4, 1fr) 40px;  
  gap: 5px;  
}
```



# תרגיל חימום

---

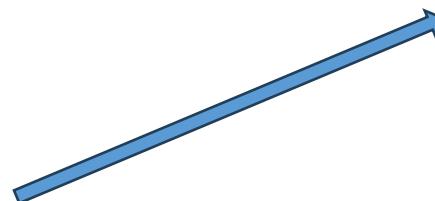
- צייר את הגריד הבא על ידי שימוש ב CSS Grid



# 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>
```

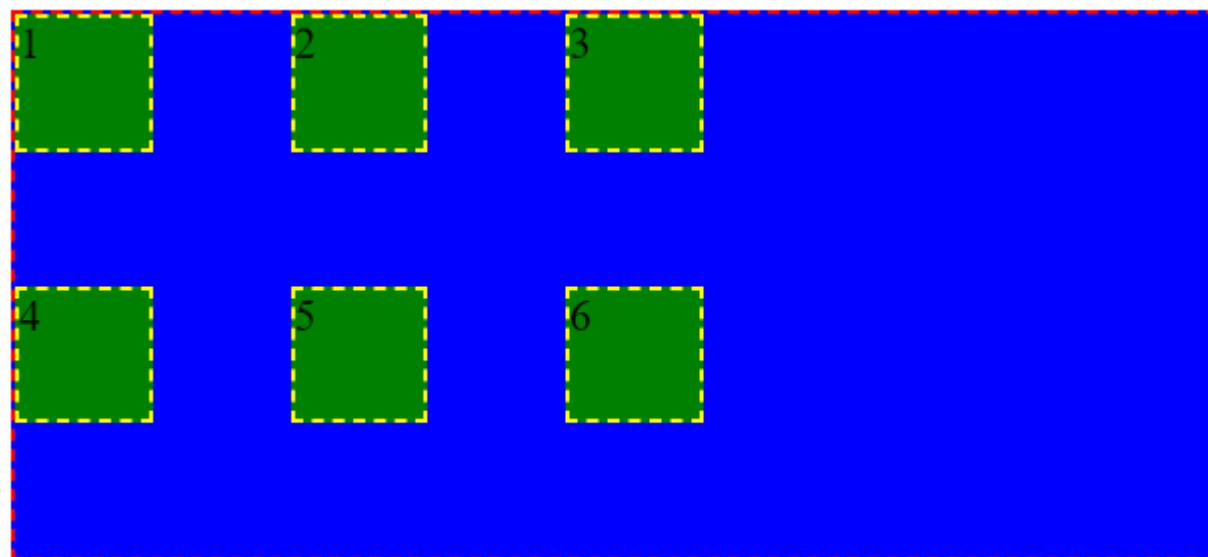
נסתכל על הקונטינר הבא  
המכיל 6 אלמנטים בתוכו

```

<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>

```

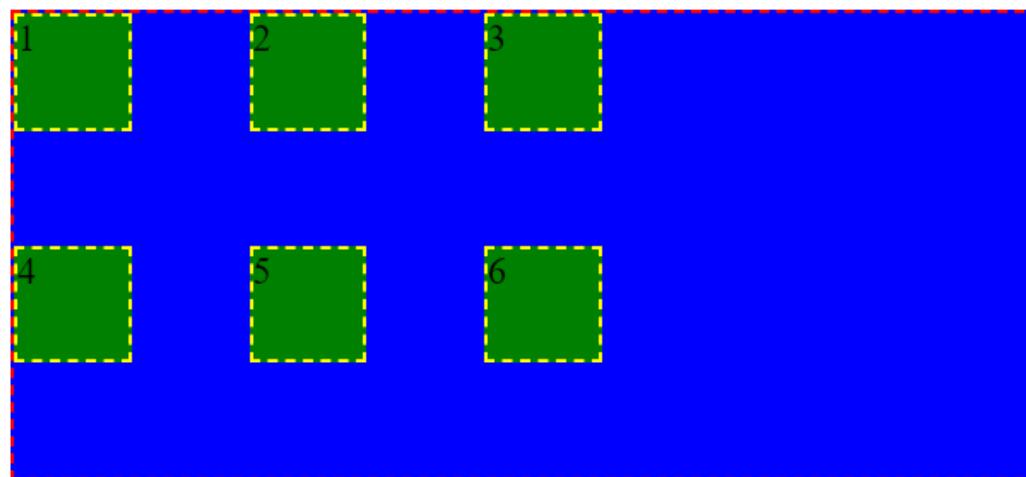
שים לב שה Grid Items במקורה זה אינם  
מלאים ה Cell (שהוגדר ל 100px )  
אל תופס רק ax50 כפי שהגדכנו



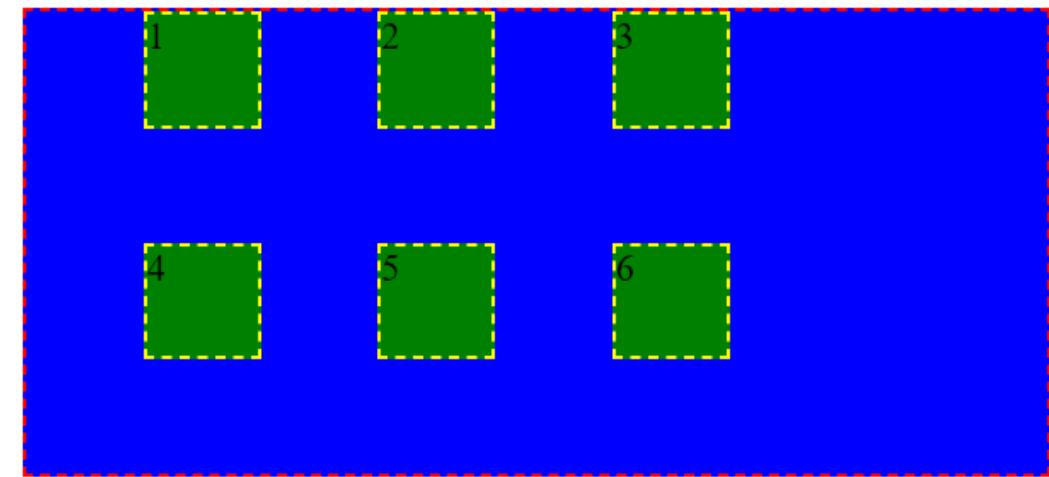
# justify-items

---

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



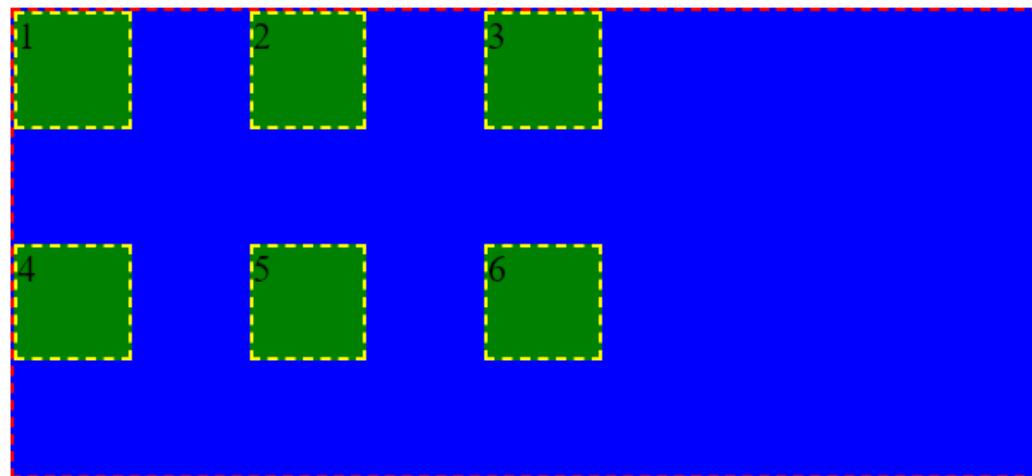
`justify-items: end;`



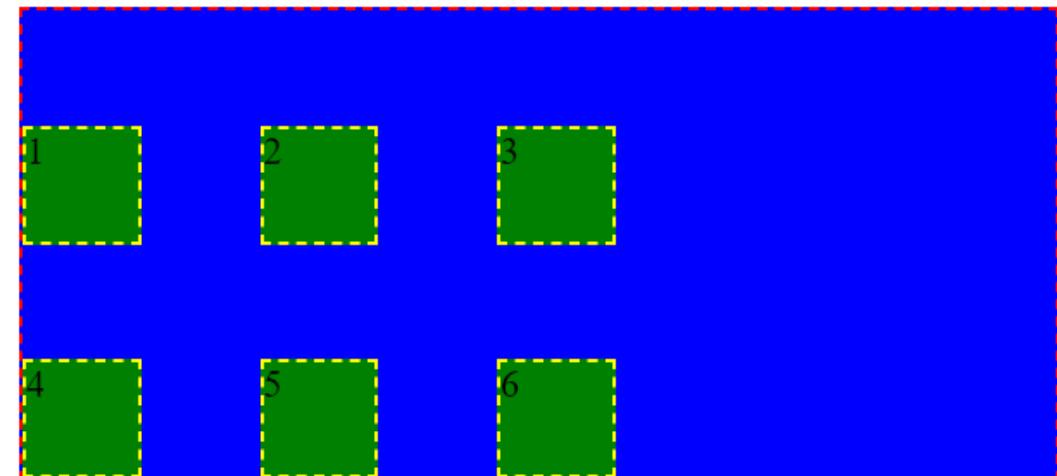
# align-items

---

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

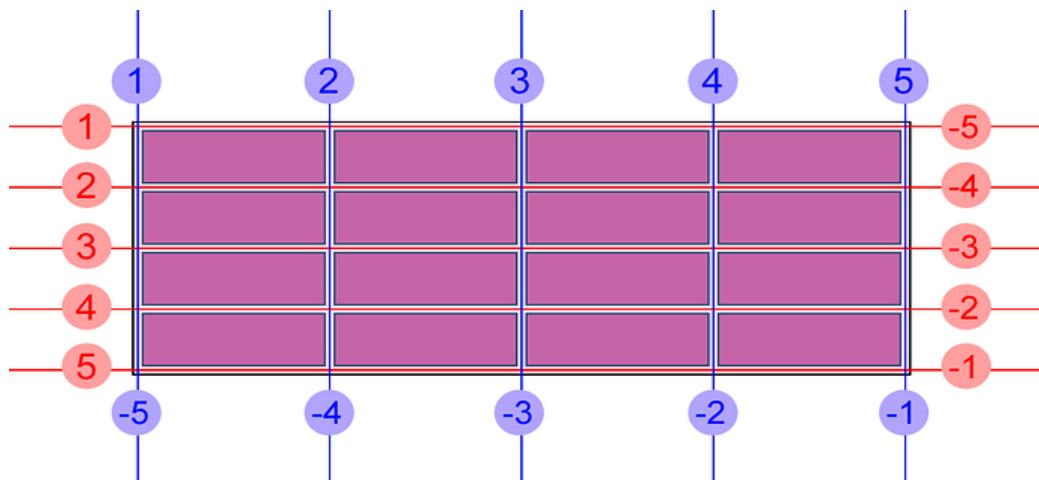


**align-items: end;**



# grid-column-start/end

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



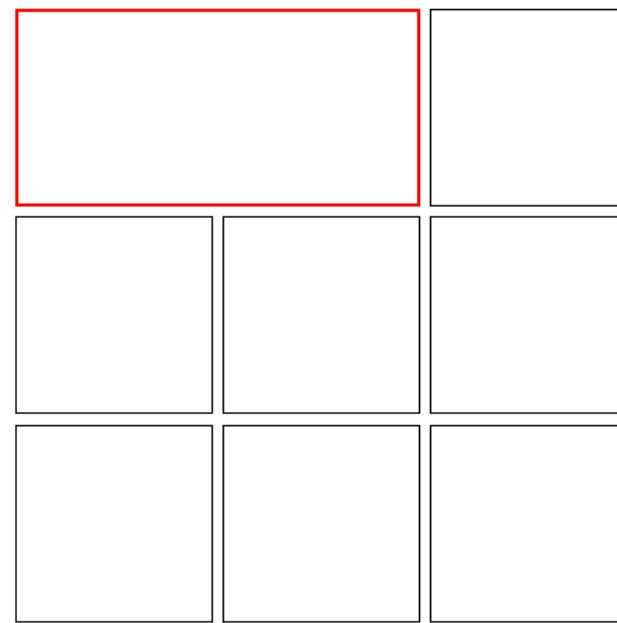
grid-column-start

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;  
}
```

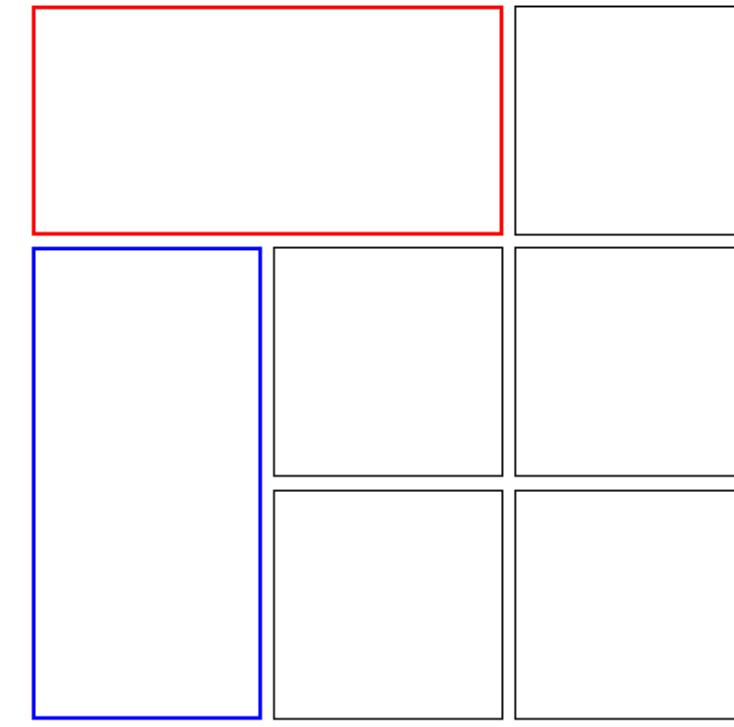


```
.container {  
width: 300px;  
margin: 100px auto;  
display: grid;  
grid-template-columns: 100px 100px 100px;  
grid-template-rows: 100px 100px 100px;  
gap: 5px;  
}
```

```
.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

---

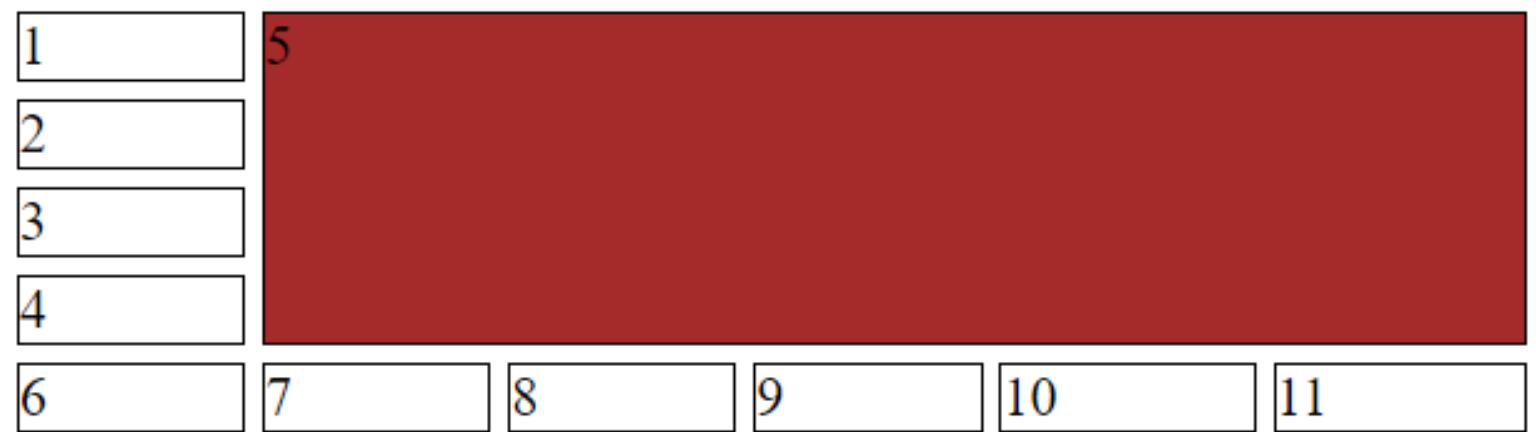
התכונה `grid-area` יכולה לשמש כתחליף מקוצר לתכונות הבאות:

- `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>
```

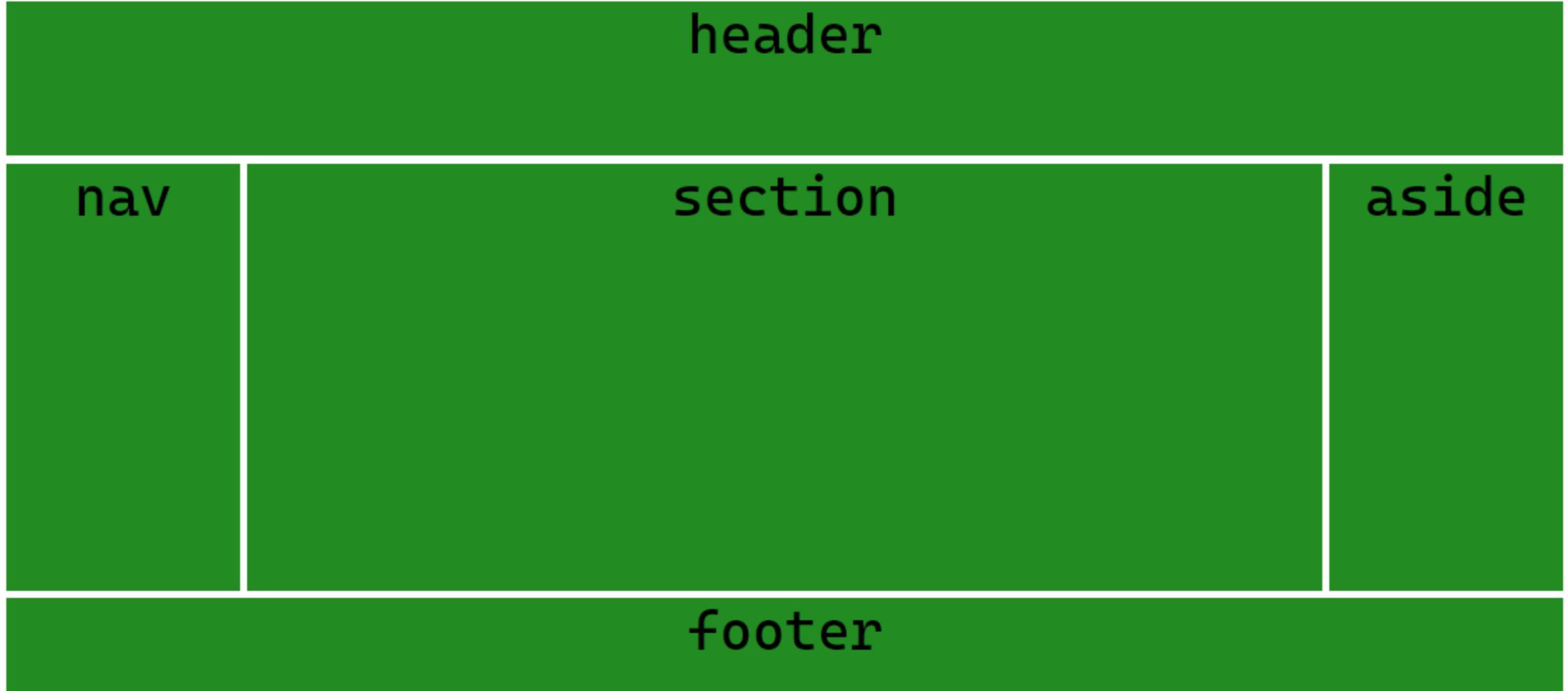
יש להגדיר על הקונטינר של הגRID בצורה בהם

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;  
}
```

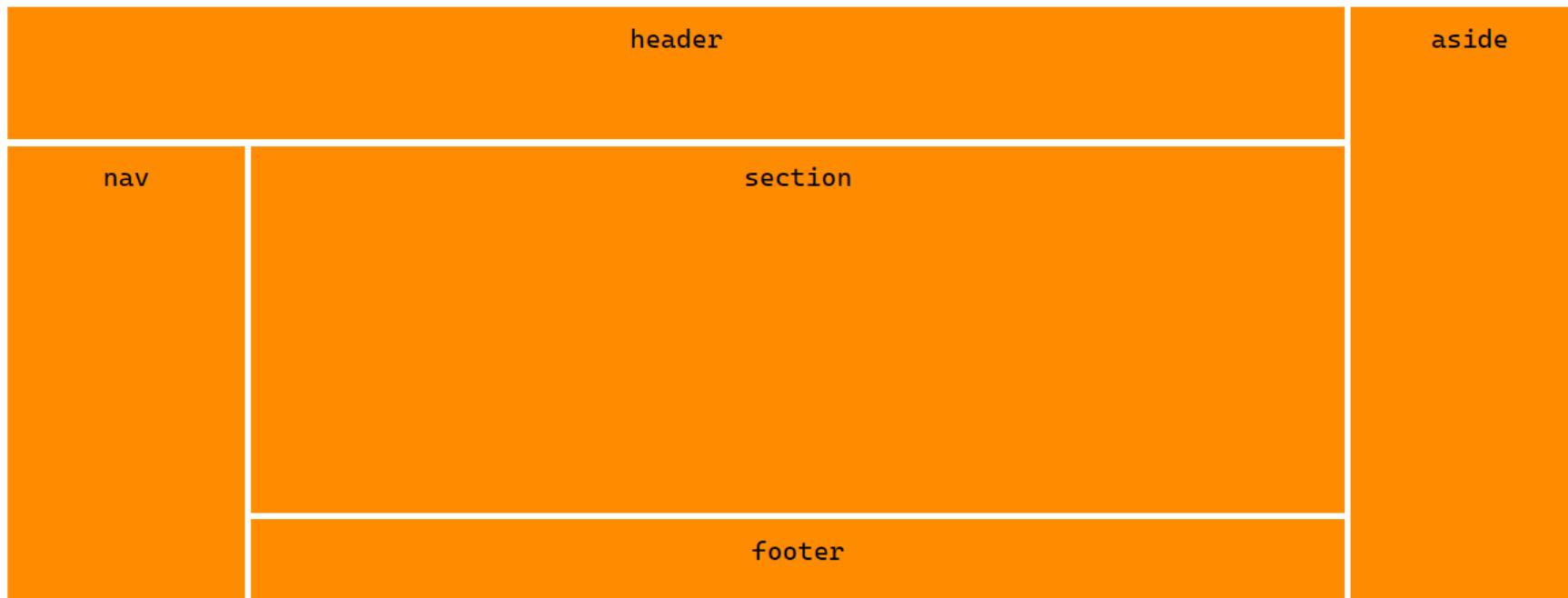
גושים מייצג שורה  
מילה מייצגת עמודה



# 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 ~980 ("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 (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

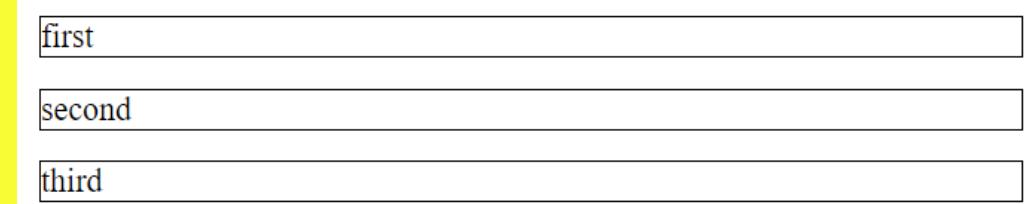
```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-gap: 15px;  
}  
  
.container > li {  
    list-style-type: none;  
    border: 1px solid black;  
}
```



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

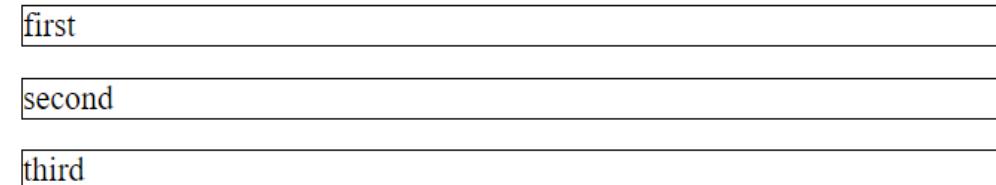
```
@media screen and (max-width: 960px) {  
    .container {  
        grid-template-columns: 1fr;  
    }  
}
```

```
<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:

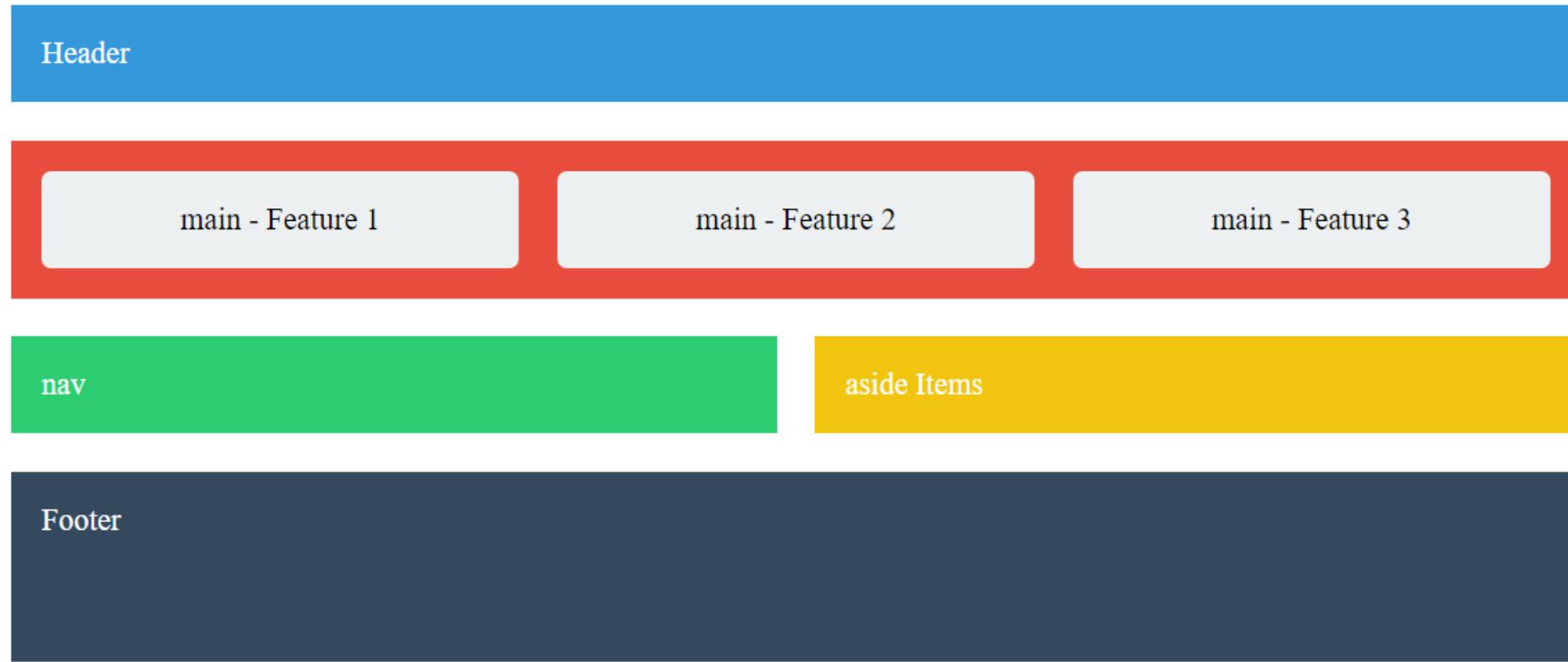
גודול מ x 960px, קטן מ 960px וגודול מ x 480px, קטן מ 480px

העימוד על פי המסכים בשקפים הבאים

# Large screen



# Tablet screen

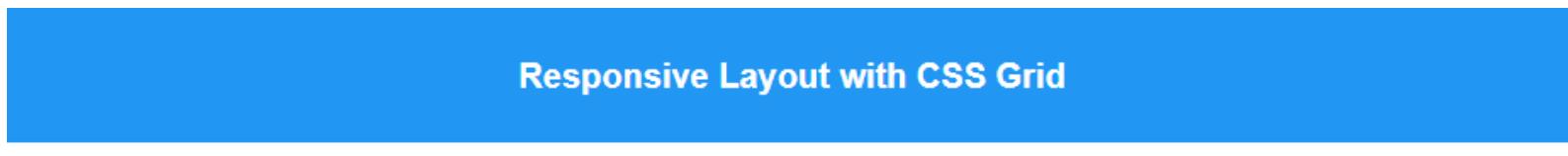


# Smart phone



# 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?