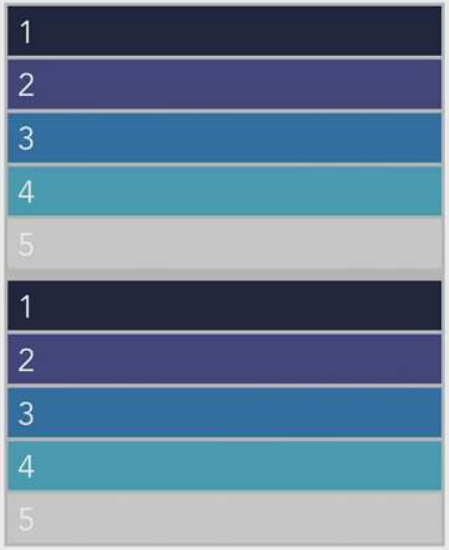
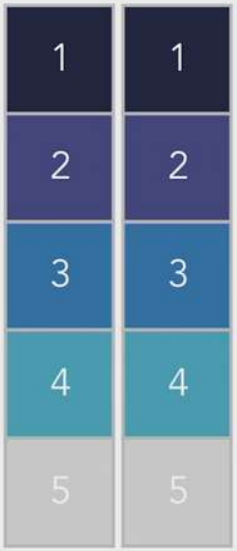
CSS3 – CSS Grid Layout

- Ez egy 2-dimenziós elrendezési rendszer, ami egy rácsot (azaz oszlopokat és a sorokat) használ .
- Jól együtt képes működni a Flexbox –szal.
- A rácsra mint szülőelemre (rácstartály – Grid Container), és a rács elemeire (gyermekeire) CSS szabályokat alkalmaz
- Alapok:
  - Meg kell határozni egy tároló elemet (konténert) a rács számára: ezen elem esetén:
    display: grid vagy *inline -grid;*

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

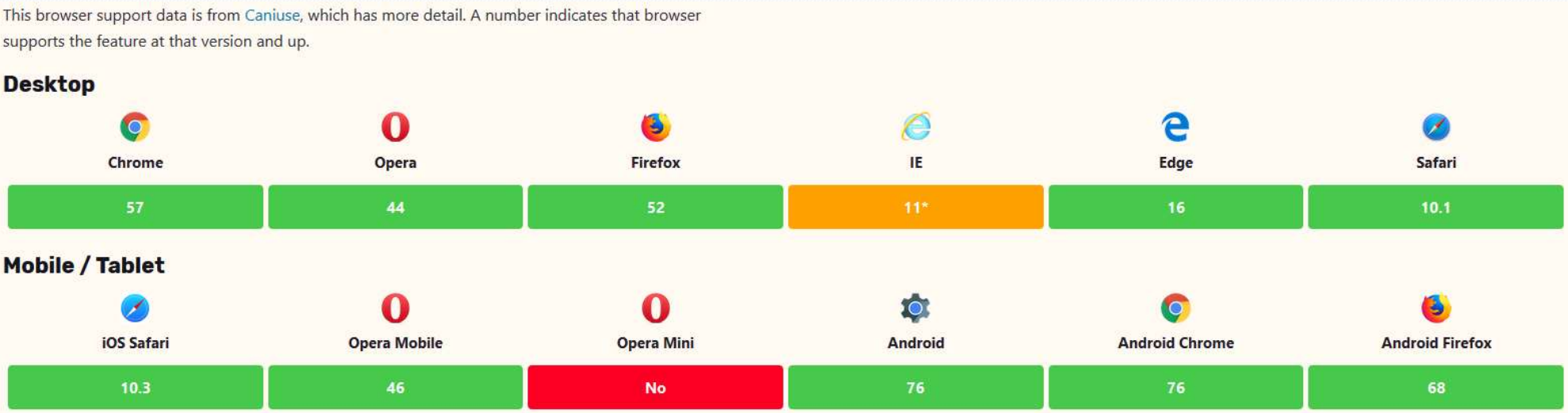



```
.grid-container {
  display: inline-grid;
}
```



CSS3 – CSS Grid Layout

- Be kell állítani a sorokat és az oszlopokat a *grid-template-columns:* és a *grid-template-rows* tulajdonságokat használva.
- El kell helyezni az elemeket a rácsban. Hasonlóan a FlexBox-hoz az elemek sorrendje nem számít.

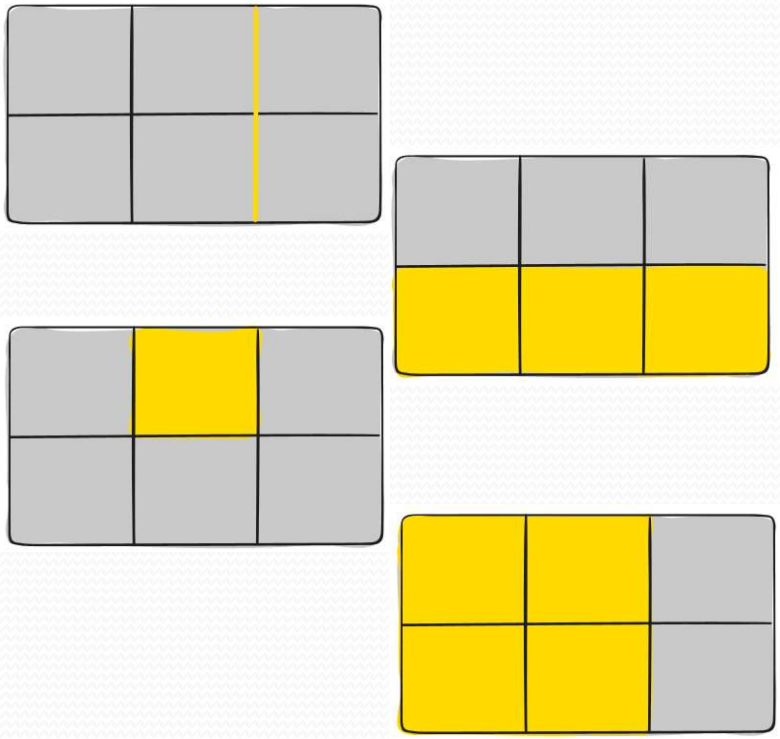


CSS3 –CSS Grid Layout

CSS Grid Layout -Alapfogalmak

- Grid Container – Azon elem, mely *display: grid (vagy inline-grid);* tulajdonságú, ez a közvetlen szülője minden grid-item –nek. (container)
- **Grid Item :** A container közvetlen gyereke. (item, ~~sub-item~~)
- **Grid Line :** A rács (grid) szerkezetét alakítják ki lehetnek vízszintesek és függőlegesek *(pl.: 3. függőleges (oszlop) rácsvonal*
- **Grid Track :** A két szomszédos rácsvonal közötti rész *(pl.: 2 és 3. vízszintes (sor) rácsvonal közötti rész)*
- **Grid Cell :** Ez a rács egyetlen "egysége„ cellája. Két szomszédos sor- és két szomszédos oszloprács közötti rész. *(pl.: 1. és 2. sorrács vonal, valamint a 2. és 3. oszloprács vonal közötti rész.)*
- **Grid Area :** Ez egy rács terület tetszőleges számú cellából állhat. Két sor- és két oszloprács közötti rész. *(pl.: 1. és 3. sorrács vonal, valamint a 1. és 3. oszloprács vonal közötti rész.)*

```
<div class="container">
    <div class="item"></div>
    <div class="item">
        <p class="sub-item"></p>
    </div>
    <div class="item"></div>
</div>
```

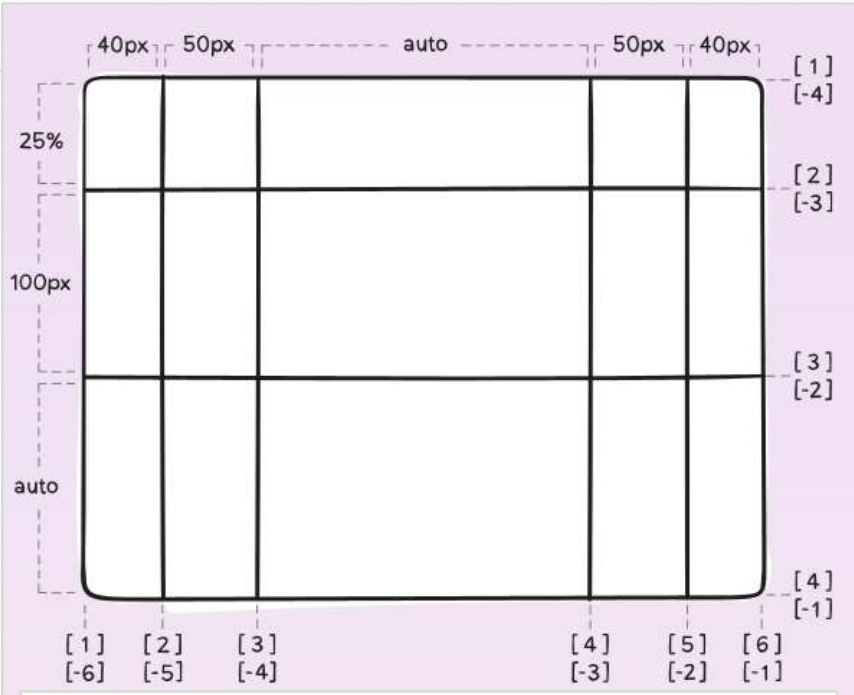


CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- *display: grid | inline-grid* : Rács elrendezést hoz létre
  - *grid*- blokk szintű rácsot hoz létre
  - *inline-grid* - létrehoz egy inline-szintű rácsot
- *grid-template-columns, grid-template-rows* : Rács oszlopainak és sorainak szóközzel elválasztott listájának megadására szolgál.
  - <*track-size*> - a sáv méretének megadás (érték, százalék, törtrész)
  - <*line name*> - a sávokat elválasztó vonal neve

```
.container {
  grid-template-columns: <track-size> ... | <line-name> <track-size> ...;
  grid-template-rows: <track-size> ... | <line-name> <track-size> ...;
}
```

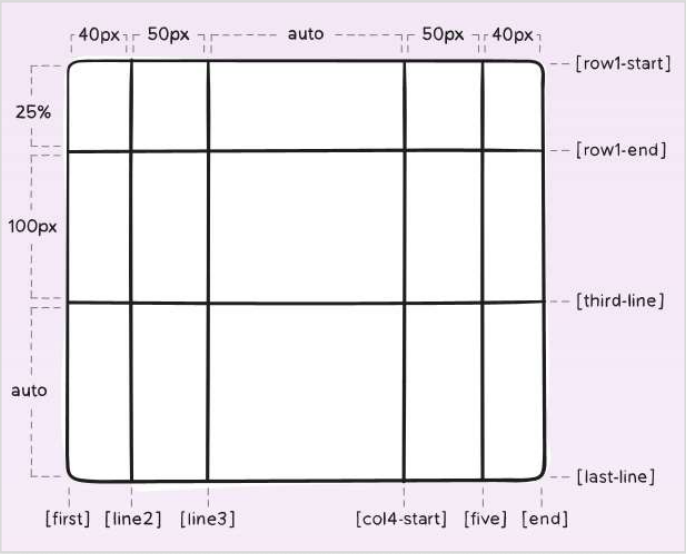



```
.container {
  grid-template-columns: 40px 50px auto 50px 40px;
  grid-template-rows: 25% 100px auto;
}
```

CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- Egy vonalnak lehet több neve is
- Ha a szerkezet tartalmaz ismétlődő részeket akkor használható a repeat() fügv-nyel történő megadás
- Ha több vonal is ugyanazt a nevet kapta akkor a neve után megadott *számmal* lehet hivatkozni a megfelelőre
- A **fr** egység lehetővé teszi, hogy a sáv méretét a rács szabad helyének adott hányadára állítsuk be. A szabad helyet minden nem rugalmas elem után kiszámítódik.
  - 1. példa: az elemeket a rácstartály szélességének egyharmadára állítja
  - 2. példa: az elemek számára rendelkezésre álló szabad terület nem tartalmazza az 50 képpontot




```
.container {
  grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];
}

.container {
  grid-template-columns: repeat(3, 20px [col-start]);
}

.container {
  grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];
}

.item {
  grid-column-start: col-start 2;
}

.container {
  grid-template-columns: 1fr 1fr 1fr;
}

.container {
  grid-template-columns: 1fr 50px 1fr 1fr;
}
```

```
.container {
  grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
}
```

CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

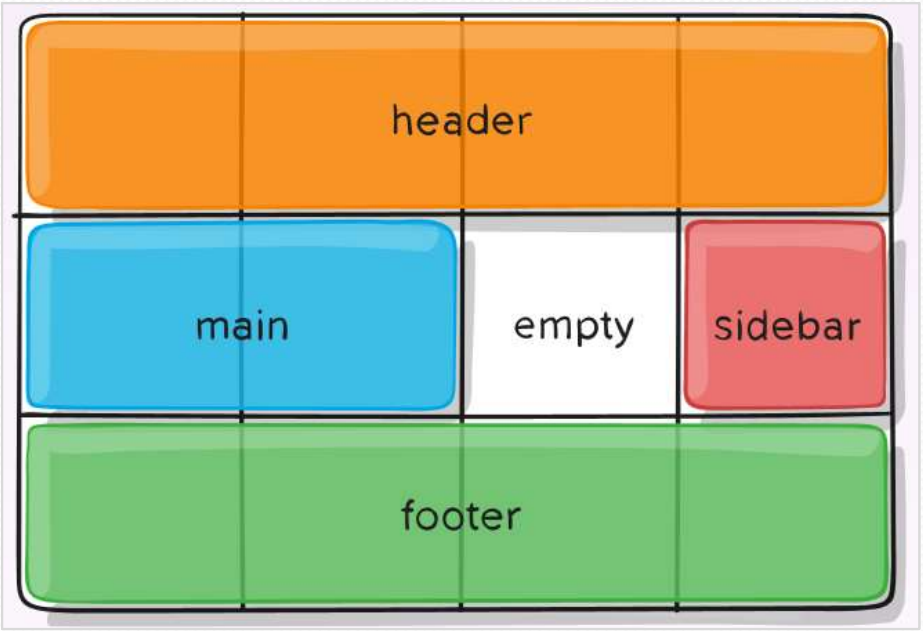
- *grid-template-areas:* Egy rács területet határoz meg olyan módon, hogy a terület nevével hivatkozunk a területhez tartozó cellákban. A *grid-area* tulajdonságot használva
  - <*grid-area-name*> - az adott terület neve
  - **.** – üres rács cella
  - *none* – nem definiált terület

```
.container {
  grid-template-areas:
    "<grid-area-name> | . | none | ..."
    "...";
}
```

Példa:

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- *grid-template: (rövidalak)* a *grid-template-columns, grid-template-rows, grid-template-areas* egy deklarációban
  - *none* – mindhárom értéket a kezdeti értékre állítja
  - <grid-template-rows> / <grid-template-columns> – az tulajdonágokat a megadott értékre állítja be, a területek nem állítja semmilyenre.

```
.container {
  grid-template: none | <grid-template-rows> / <grid-template-columns>;
}
```

  - Példa:

```
.container {
  grid-template:
    [row1-start] "header header header" 25px [row1-end]
    [row2-start] "footer footer footer" 25px [row2-end]
    / auto 50px auto;
}
```

ekvivalens

```
.container {
  grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end];
  grid-template-columns: auto 50px auto;
  grid-template-areas:
    "header header header"
    "footer footer footer";
}
```

CSS3 –CSS Grid Layout

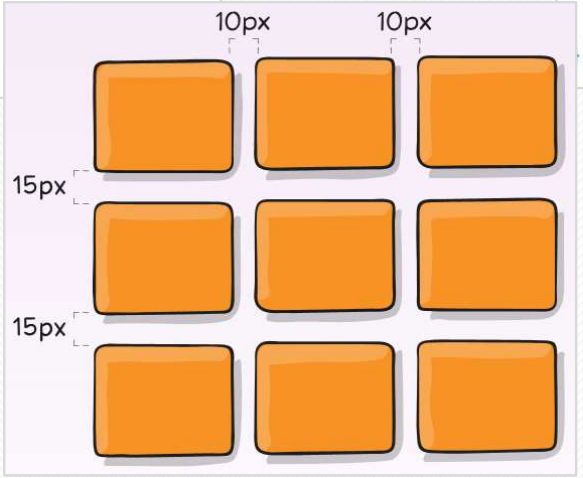
A Szülő (Grid Container) tulajdonságai

- *grid-column-gap, grid-row-gap:* Megadja a rácsvonalak méretét az oszlopok/sorok között. A sorok/ oszlopok előtt és után nem állíthatja be a gap-et.
  - <*line-size*> - vonal mérete

```
.container {
  grid-column-gap: <line-size>;
  grid-row-gap: <line-size>;
}
```

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  grid-column-gap: 10px;
  grid-row-gap: 15px;
}
```

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  grid-gap: 15px 10px;
}
```



- *grid-gap: (rövidalak)* a *grid-column-gap, grid-row-gap* egy deklarációban

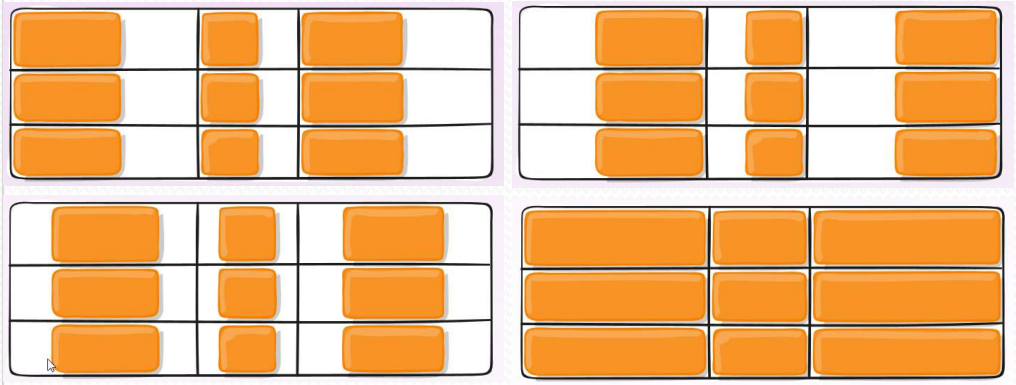
```
.container {
  grid-gap: <grid-row-gap> <grid-column-gap>;
}
```

CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

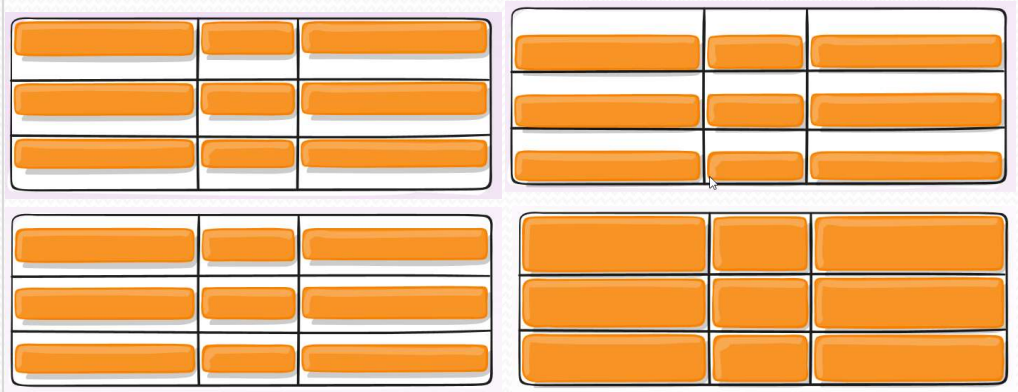
- *justify-items:* Igazítja a rács elemeit a cellán belül - vízszintesen . Ez az érték az összes rács elemre vonatkozik a konténerben.
  - *start* – a cellájuk kezdőéléhez igazít
  - *end* - a cellájuk záró éléhez igazít
  - *center* - a cella közepére igazít
  - *stretch* - kitölti a cella teljes szélességét (aé)

```
.container {
  justify-items: start | end | center | stretch;
}
```



- *align-items:* Igazítja a rács elemeit a cellán belül- függőlegesen. Ez az érték az összes rács elemre vonatkozik a konténerben.
  - *start* – a cella kezdőéléhez (felső) igazít
  - *end* - a cellájuk záró éléhez (alsó) igazít
  - *center* - a cella közepére igazít
  - *stretch* - kitölti a cella teljes magasságát (aé)

```
.container {
  align-items: start | end | center | stretch;
}
```

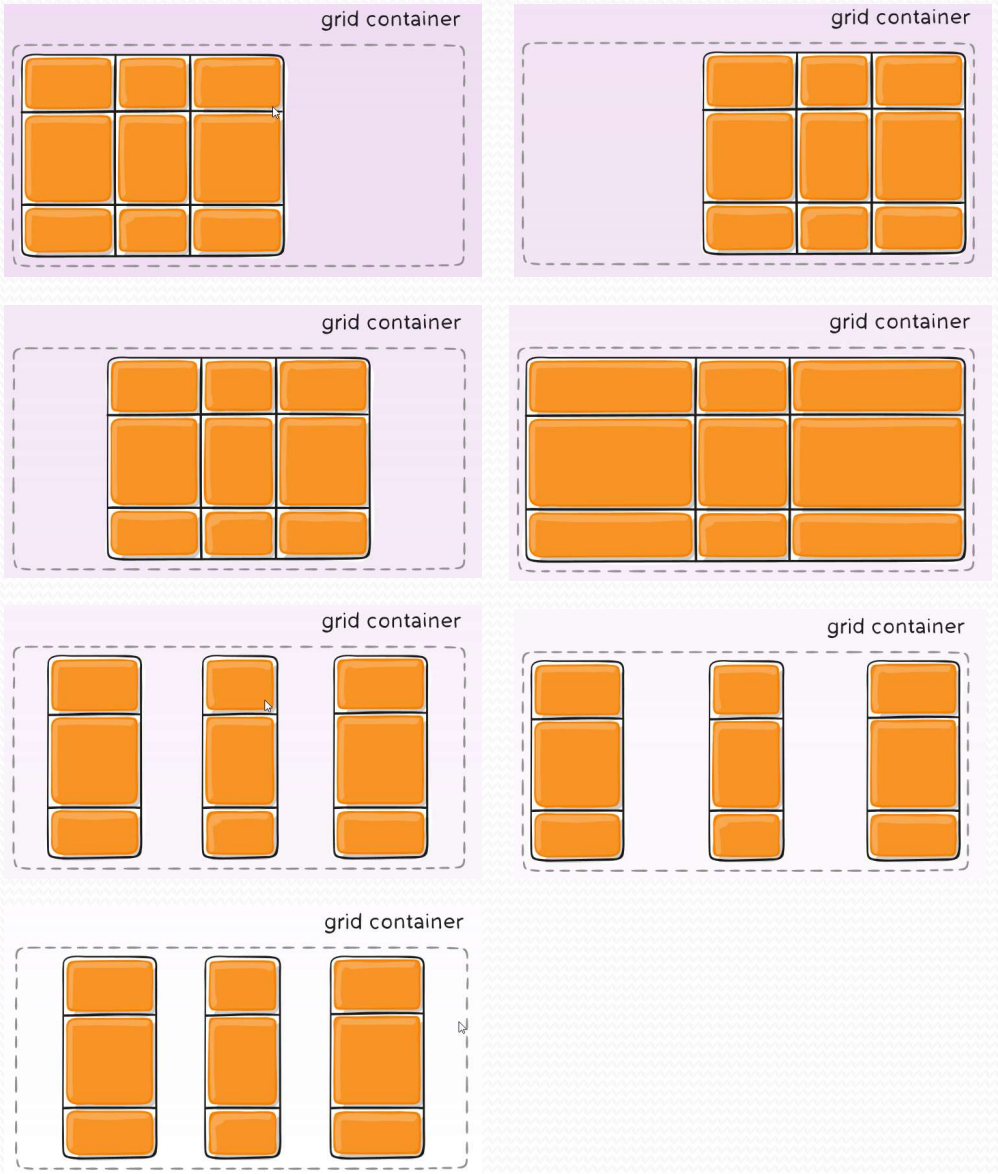


- *place-items:* Az align-items és a justify-items együttes beállítás
  - Beállítás: <align-items> / <justify-items>

CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- *justify-content:* A tartályon belüli vízszintes igazítása a rácsnak
  (Ha a rács teljes mérete kisebb, mint a konténer (a rácselemek nem flexibilisen vannak megadva) akkor állítható a ács igazítása a tartályon belül.)
  - *start* – a konténer kezdőéléhez igazít
  - *end* - a konténer záró éléhez igazít
  - *center* - a konténer közepére igazít
  - *stretch* - kitölti a konténer teljes szélességét
  - *space- around* - egyenletes nagyságú helyet hagy rácselemek között úgy, hogy a végén félegységnyi helyet hagy.
  - *space-between* - egyenletes nagyságú helyet hagy rácselemek között úgy, hogy a végén nincs hely hagyva
  - *space-evenly* - egyenletes nagyságú helyet hagy rácselemek között a végeket is beleértve

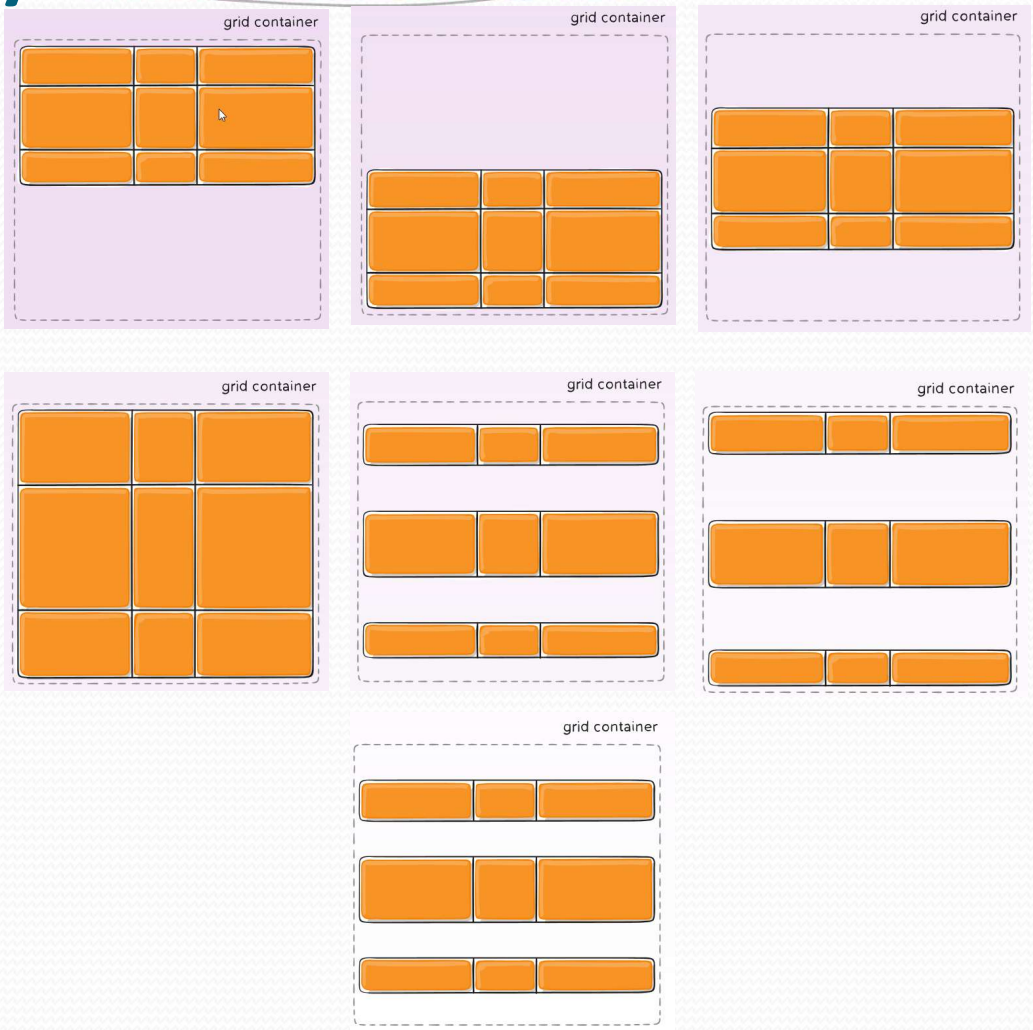



```
.container {
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```

CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- *align-content:* A tartályon belüli függőleges igazítása a rácsnak
  (Ha a rács teljes mérete kisebb, mint a konténer (a rácselemek nem flexibilisen vannak megadva) akkor állítható a ács igazítása a tartályon belül.)
  - *start* – a konténer kezdőéléhez (felső) igazít
  - *end* - a konténer záró éléhez (alsó) igazít
  - *center* - a konténer közepére igazít
  - *stretch* - kitölti a konténer teljes magasságát
  - *space- around* - egyenletes nagyságú helyet hagy rácselemek között úgy, hogy a végén félegységnyi helyet hagy.
  - *space-between* - egyenletes nagyságú helyet hagy rácselemek között úgy, hogy a végén nincs hely hagyva
  - *space-evenly* - egyenletes nagyságú helyet hagy rácselemek között a végeket is beleértve




```
.container {
  align-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```

- *place-content:* Az align-content és a justify-content együttes beállítás
  - Beállítás: <align-content> / <justify-content>

CSS3 –CSS Grid Layout

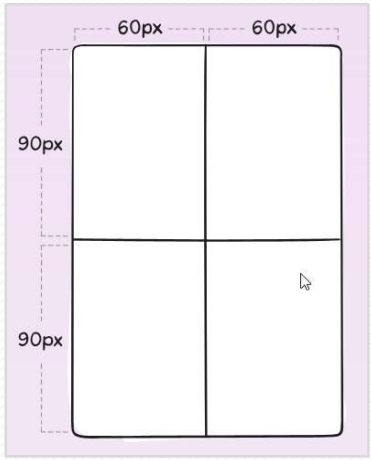
A Szülő (Grid Container) tulajdonságai

- *grid-auto-columns, grid-auto-rows:* Megadja az automatikusan-generált rács(implicit rács) sávjait. Az implicit sávok akkor jönnek létre, ha több rács elem van, mint amennyi cella a rácsban, vagy amikor egy rács elem kerül az explicit rácson kívül.
  - *<track-size>* - lehet a rács szabad helyének hossza, százaléka vagy töredéke (a fr egységben)

```
.container {
  grid-auto-columns: <track-size> ...;
  grid-auto-rows: <track-size> ...;
}
```

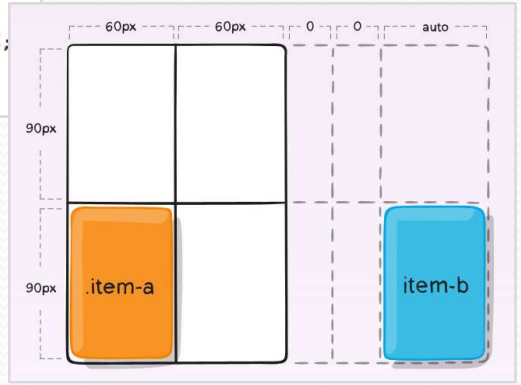
```
.container {
  grid-template-columns: 60px 60px;
  grid-template-rows: 90px 90px;
}
```

Megadhatjuk a rács elemeket csupán a rácssorok és rácsoszlopok megadásával. Pl.:

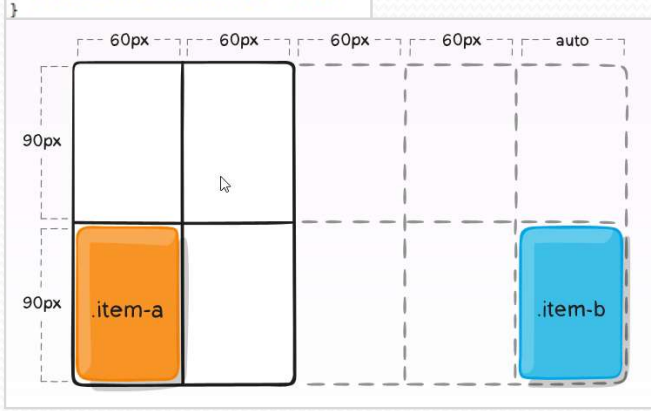


Az .item-b az 5. oszloppal kezdődik, és a 6. oszlop végén fejeződik be, de soha nem határoztuk meg az 5. vagy a 6. rácsvonalát . Mivel nem létező rácsvonalakra hivatkozunk, ezért nulla szélességű implicit sávok jönnek létre a résekben.

```
.item-a {
  grid-column: 1 / 2;
  grid-row: 2 / 3;
}
.item-b {
  grid-column: 5 / 6;
  grid-row: 2 / 3;
}
```




```
.container {
  grid-auto-columns: 60px;
}
```



Ha megadjuk az grid-auto-columns/rows értékeket megadva akkor megadhatjuk ezen implicit sávok szélességét és magasságát

CSS3 –CSS Grid Layout

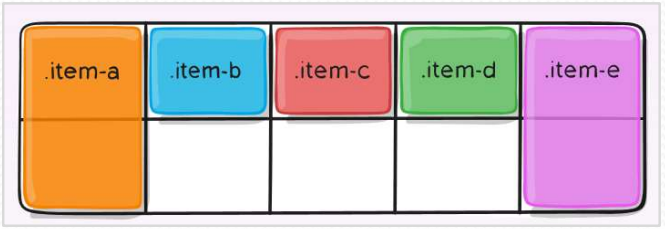
A Szülő (Grid Container) tulajdonságai

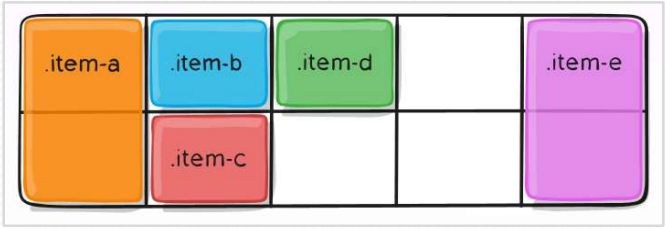
- *grid-auto-flow:* Ha vannak olyan rács elemek, amelyeket nem helyezünk el explicite a rácson kifejezetten tesz a rácsra, akkor az automatikus elhelyezés algoritmus ha be van bekapcsolva, automatikusan elhelyezi ezen elemeket. Ez a tulajdonság szabályozza az automatikus elhelyezés algoritmust.
- <*row*> - a sorokat egymás után töltse ki
- <*column*> - a oszlopokat egymás után töltse ki
- <*dense*> - a később érkező kisebb elemekkel töltse ki a rácsban található lyukakat

```html
<section class="container">
    <div class="item-a">item-a</div>
    <div class="item-b">item-b</div>
    <div class="item-c">item-c</div>
    <div class="item-d">item-d</div>
    <div class="item-e">item-e</div>
</section>
```

```css
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: row;
}
.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}
.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```

```css
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: column;
}
.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}
.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```





CSS3 –CSS Grid Layout

A Szülő (Grid Container) tulajdonságai

- *grid: (rövidalak)* a *grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, grid-auto-flow* egy deklarációban
  - ***none*** – minden értéket a kezdeti értékre állítja
  - **<grid-template >** ugyanúgy működik mint *grid-template* sorthand
  - **<grid-template-rows> / [grid-auto-flow && dense?] <grid-auto-columns>?** Beállítja a *grid-templates-rows*-ban megadott értéket a sorokra/ jel után megadott *grid-auto-flow* étéket beállítja az oszlopokra, ha *a dense* is meg van adva akkor alkalmazza a megfelelő algoritmust és végül, ha a *grid-auto-colums* nincs megadva akkor az *auto* értéke érvényesül
  - **[grid-auto-flow && dense?] <grid-auto-rows>?/ <grid-template-columns>** Beállítja a *grid-templates-colums* -ban megadott értéket az oszlopokra, ha van megadott *grid-auto-flow* étéket beállítja az sorokra, ha *a dense* is meg van adva akkor alkalmazza a megfelelő algoritmust és végül, ha a *grid-auto-rows* nincs megadva akkor az *auto* értéke érvényesül
  - Példák ekvivalens beállításra:

```
.container {
    grid: 100px 300px / 3fr 1fr;
  }
```

⇕

```
.container {
    grid-template-rows: 100px 300px;
    grid-template-columns: 3fr 1fr;
  }
```

```
.container {
    grid: auto-flow dense 100px / 1fr 2fr;
  }
```

⇕

```
.container {
    grid-auto-flow: row dense;
    grid-auto-rows: 100px;
    grid-template-columns: 1fr 2fr;
  }
```

```
.container {
    grid: auto-flow / 200px 1fr;
  }
```

⇕

```
.container {
    grid-auto-flow: row;
    grid-template-columns: 200px 1fr;
  }
```

```
.container {
    grid: 100px 300px / auto-flow 200px;
  }
```

⇕

```
.container {
    grid-template-rows: 100px 300px;
    grid-auto-flow: column;
    grid-auto-columns: 200px;
  }
```

CSS3 –CSS Grid Layout

- További példa ekvivalens beállításra:

```
.container {
    grid: [row1-start] "header header header" 1fr [row1-end]
          [row2-start] "footer footer footer" 25px [row2-end]
          / auto 50px auto;
  }


.container {
    grid-template-areas:
      "header header header"
      "footer footer footer";
    grid-template-rows: [row1-start] 1fr [row1-end row2-start] 25px [row2-end];
    grid-template-columns: auto 50px auto;
  }
```

Speciális függvények és kulcsszavak

1. Sorok és oszlopok méretezésekor használható az az összes megszokott hosszúságegység, pl.: *px, rem,%* stb.
2. De használhatunk olyan kulcsszavakat is, mint a *min-content, a max-content, auto* és talán a leghasznosabb törtrész is – *fr*

```
grid-template-columns: 200px 1fr 2fr min-content;
```

3. Használható olyan függvények is, amelyek ‚segíthetik a méretek rugalmas beállítását.
   Például: egy oszlopot szélességét 1fr értékre állítjuk, de nem zsugorodik tovább mint 200px,

```
grid-template-columns: 1fr minmax(200px, 1fr);
```

4. Van ismétlés () függvény. Pl.: 10 oszlop elkészítése: `grid-template-columns: repeat(10, 1fr);`
5. Ezek kombinálhatók is pl.: `grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));`

CSS3 –CSS Grid Layout

A gyerek elemek (Grid Items) tulajdonságai

float, display: inline-block, display: table cell, vertical-align és column-* tulajdonságok nincsennek hatással a rács elemeire.

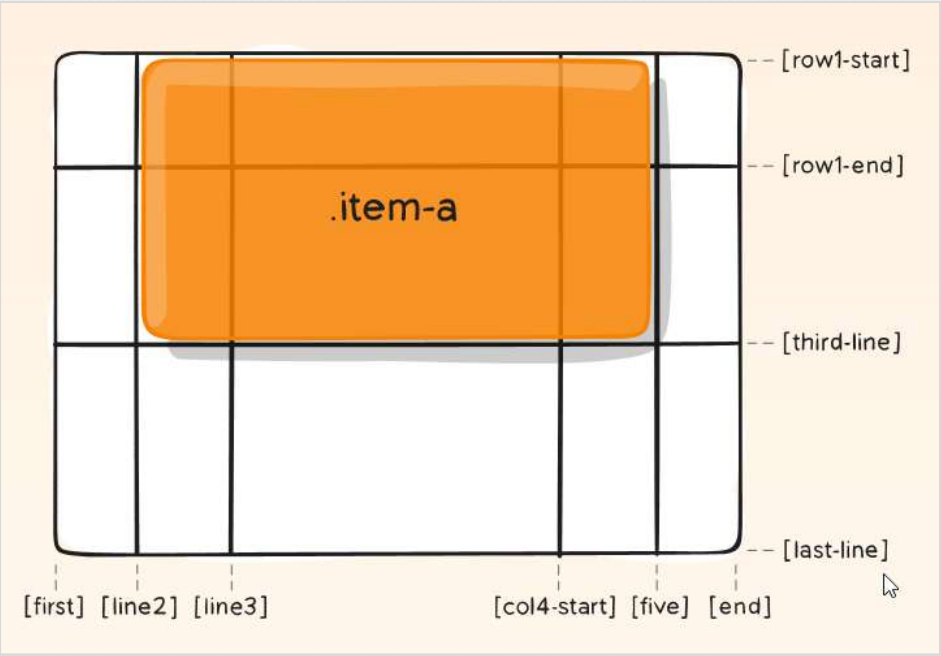
- *grid-column-start, grid-column-end, grid-row-start, grid-row-end :* a rács elem helyét a adhatjuk meg velük a rácson rácsvonalak segítségével.
  - *<line>* - a sávokat elválasztó vonal neve vagy sorszáma
  - *span <number>* - azon sávokat száma amit az elem lefed a rácsban
  - *span <name>* - azon rácsvonal neve ameddig az elem lefedi a rácsot
  - *auto* – automatikus helyfoglalás

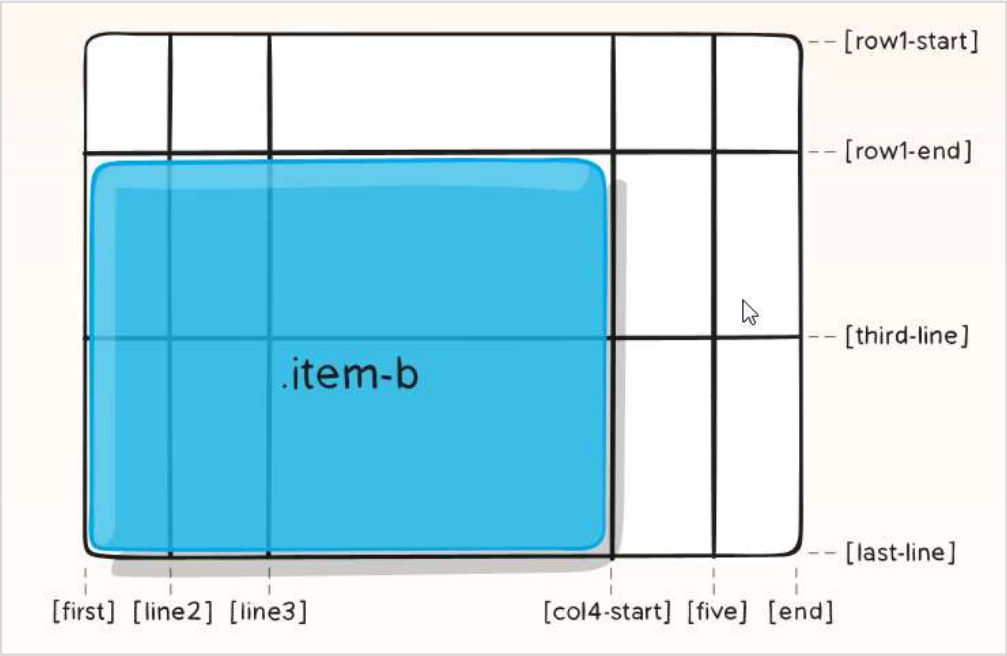
```
.item {
  grid-column-start: <number> | <name> | span <number> | span <name> | auto;
  grid-column-end: <number> | <name> | span <number> | span <name> | auto;
  grid-row-start: <number> | <name> | span <number> | span <name> | auto;
  grid-row-end: <number> | <name> | span <number> | span <name> | auto;
}
```

CSS3 –CSS Grid Layout

```
.item-a {
  grid-column-start: 2;
  grid-column-end: five;
  grid-row-start: row1-start;
  grid-row-end: 3;
}
```

```
.item-b {
  grid-column-start: 1;
  grid-column-end: span col4-start;
  grid-row-start: 2;
  grid-row-end: span 2;
}
```





- Ha nem adtak meg *grid-column-end / grid-row-end* értéket, akkor az elem alapértelmezés szerint 1 sávot fog lefedni.
- Az elemek átfedhetik egymást . Az átfedés *z-index* segítségével adható meg.

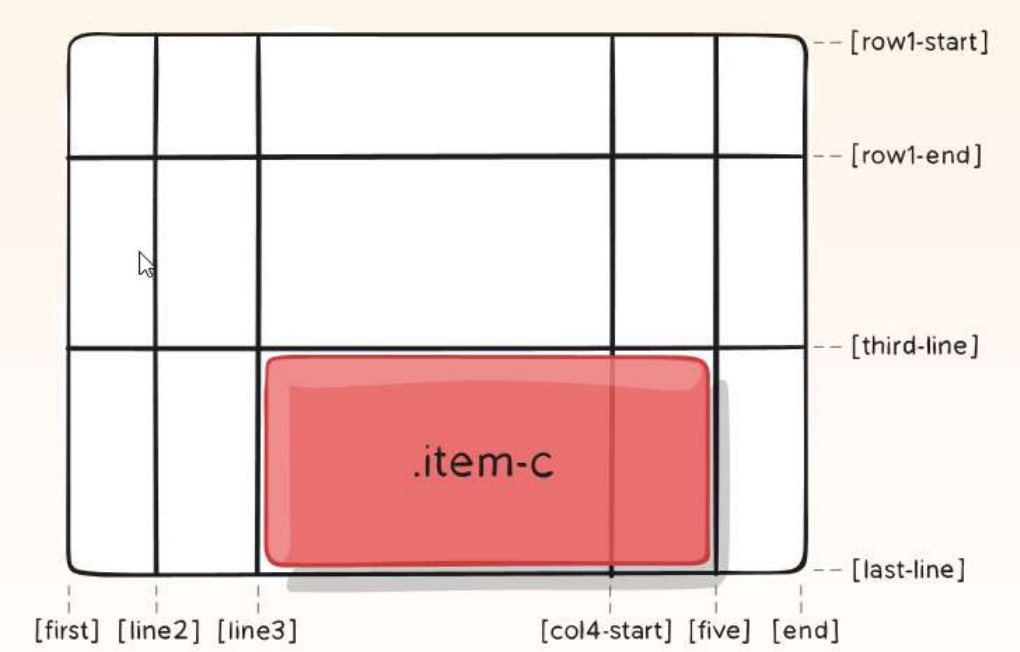
CSS3 –CSS Grid Layout

A gyerek elemek (Grid Items) tulajdonságai

- *grid-column, grid-row (rövid alak):* a rács elem helyét a adhatjuk meg velük a rácson rácsvonalak segítségével rövid formában.

```
.item {
  grid-column: <start-line> / <end-line> | <start-line> / span <value>;
  grid-row: <start-line> / <end-line> | <start-line> / span <value>;
}
```

```
.item-c {
  grid-column: 3 / span 2;
  grid-row: third-line / 4;
}
```



CSS3 –CSS Grid Layout

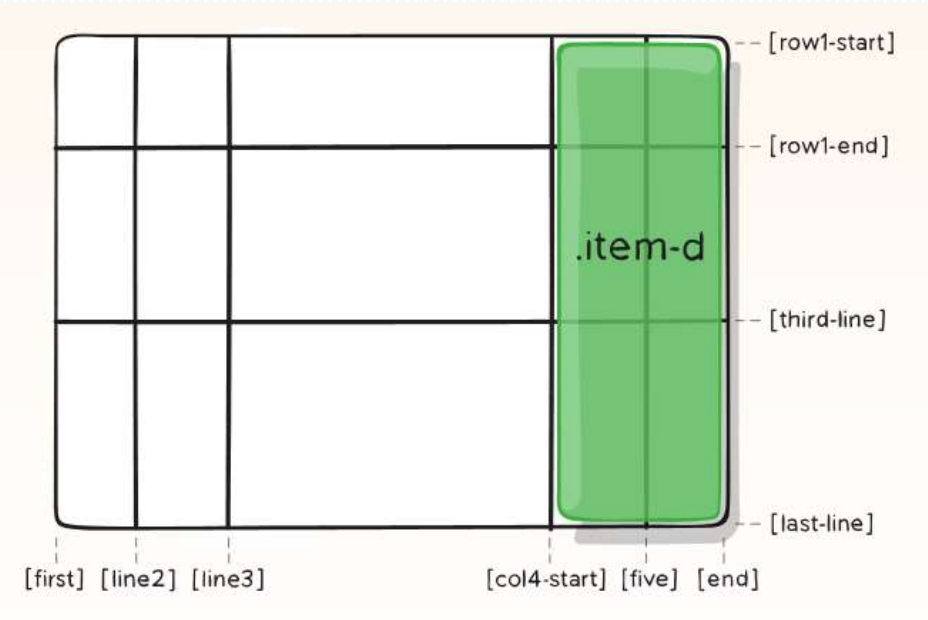
A gyerek elemek (Grid Items) tulajdonságai

- *grid-area:* A grid-template-area –vel létrehozott elem nevének megadására, vagy nagyon rövid elem megadásra használható.

```
.item {
  grid-area: <name> | <row-start> / <column-start> / <row-end> / <column-end>;
}
```

```
.item-d {
  grid-area: header;
}

.item-d {
  grid-area: 1 / col4-start / last-line / 6;
}
```

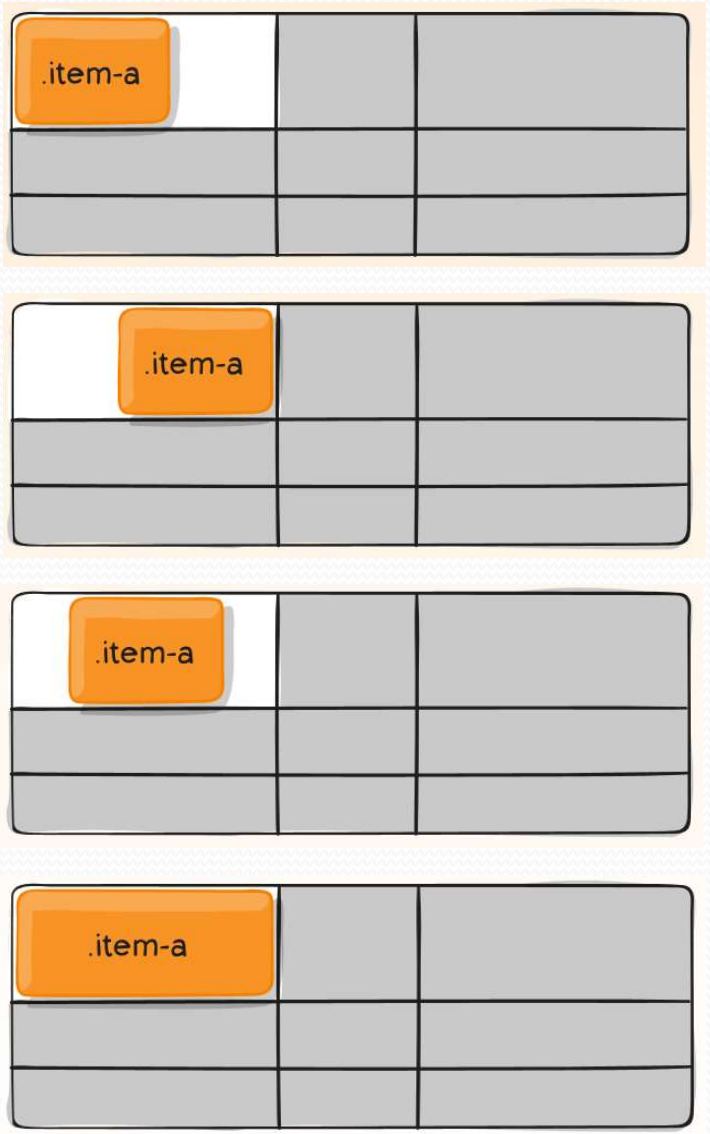


CSS3 –CSS Grid Layout

A gyerek elemek (Grid Items) tulajdonságai

- *justify-self:* egy rácselemen belüli vízszintes igazítás.
  - start - a rács elemet a cella kezdő-oszlop-éléhez igazítja.
  - end - a rács elemet a cella záró-oszlop-éléhez igazítja.
  - center - a rács elemet a cella közepére igazítja
  - stretch - kitölti a cella teljes szélességét (ez az alapértelmezés)

```
.item {
  justify-self: start | end | center | stretch;
}
```

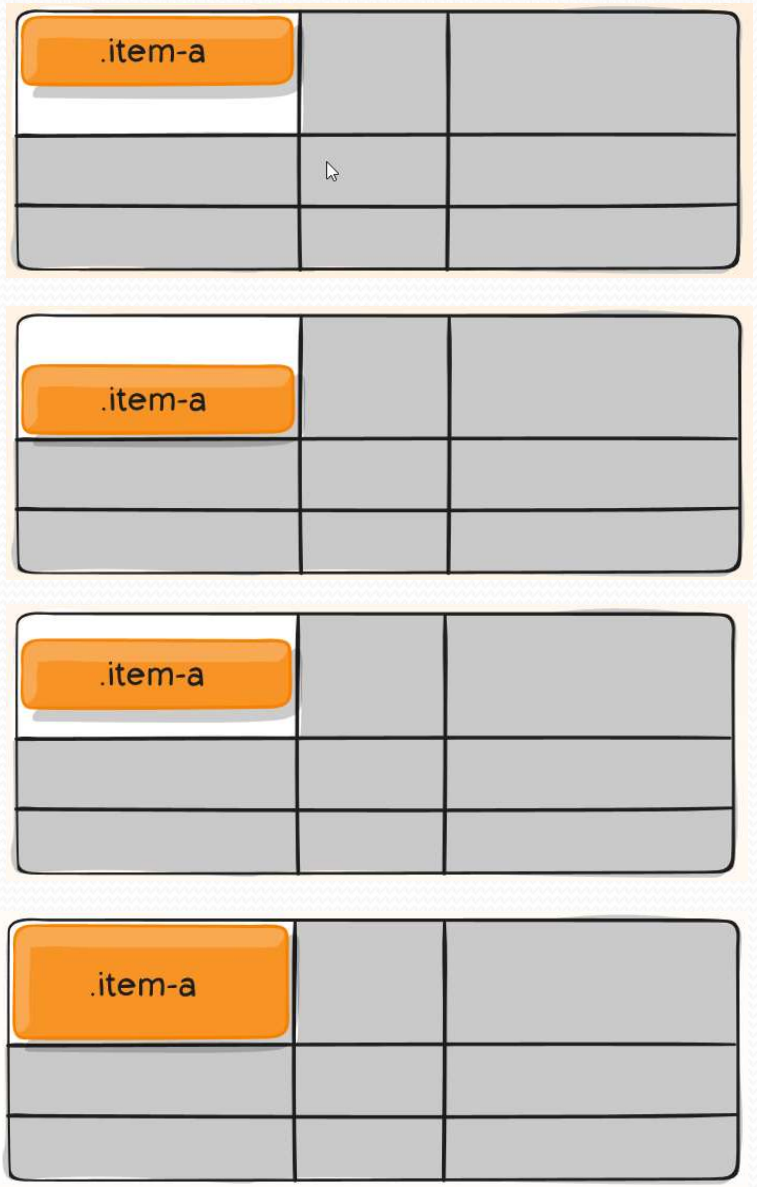


CSS3 –CSS Grid Layout

A gyerek elemek (Grid Items) tulajdonságai

- *align-self:* egy rácselemen belüli függőleges igazítás.
  - start - a rács elemet a cella kezdő-sor-éléhez igazítja.
  - end - a rács elemet a cella záró-sor-éléhez igazítja.
  - center - a rács elemet a cella közepére igazítja
  - stretch - kitölti a cella teljes szélességét (ez az alapértelmezés)

```
.item {
  align-self: start | end | center | stretch;
}
```

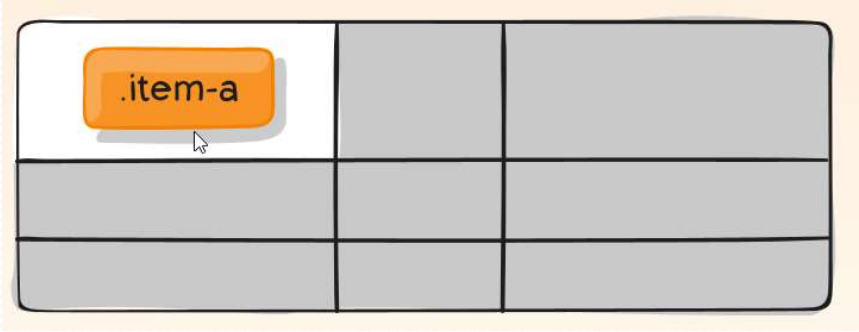


CSS3 –CSS Grid Layout

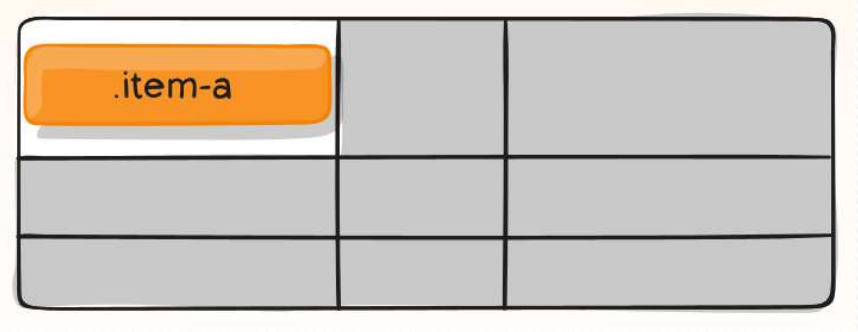
A gyerek elemek (Grid Items) tulajdonságai

- *place-self:* egy rácselemen belüli vízszintes és függőleges igazítás megadása egyben
  - *auto* - a rács elemet az elrendezési módnak megfelelően igazítja.
  - *<align-self>/<justify-self>* - a rács elemet a függőlegesen/vízszintesen igazítja, ha csak egy értéket adunk meg akkor minkét igazítás ennek megfelelő lesz

```
.item-a {
  place-self: center;
}
```

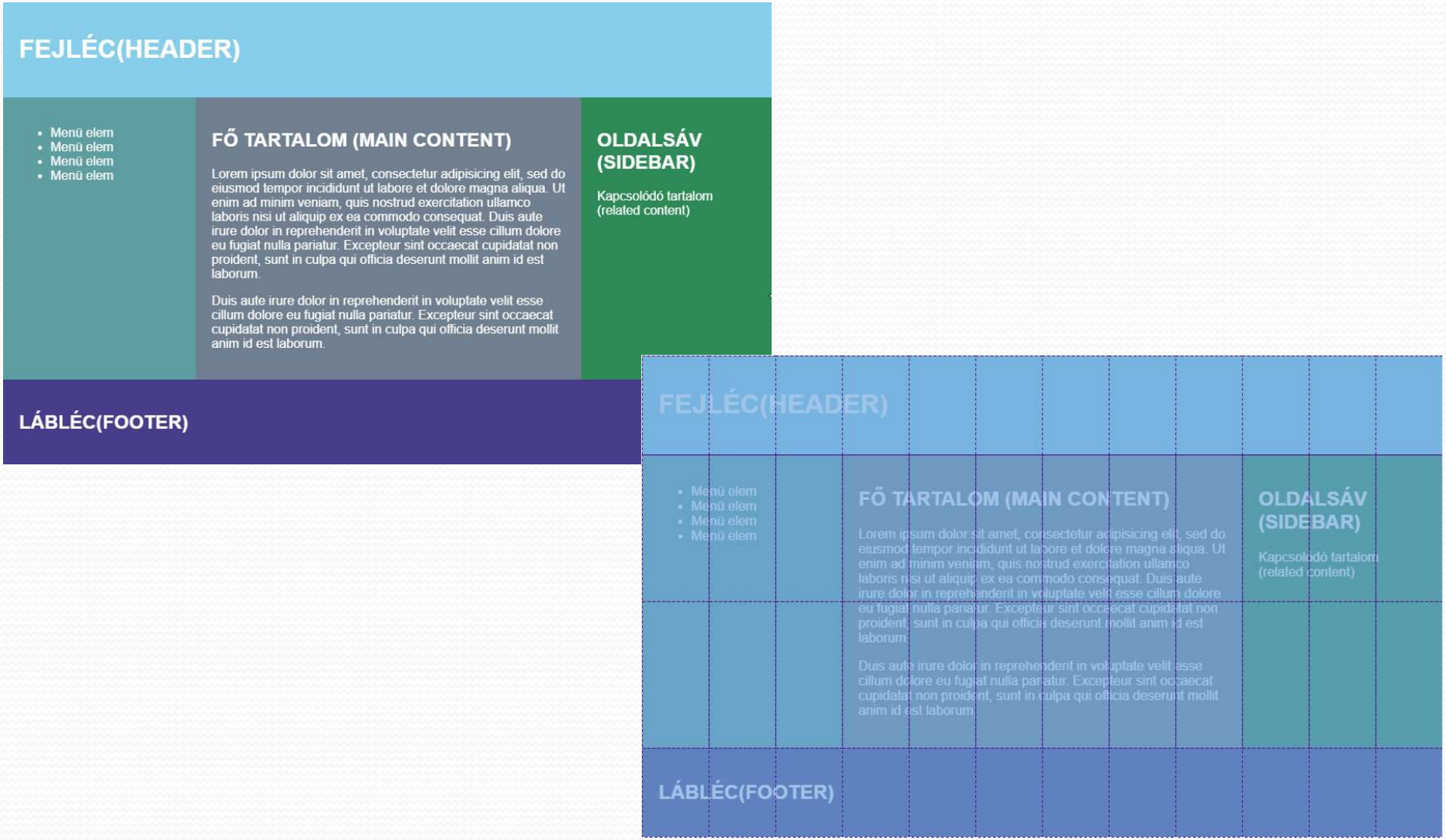



```
.item-a {
  place-self: center stretch;
}
```



CSS3 - Példa

CSS3 – Grid - Layout



CSS3 - Példa

CSS3 – Grid - Layout

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Layout (Flex/Grid) </title>
    <!--<link rel="stylesheet" href="flexbox.css">-->
    <link rel="stylesheet" href="grid.css" >

  </head>
  <body>
    <div class="container">
      <header>
        <h1>Fejléc(header)</h1>
      </header>
      <main id="main-content">
        <h2>Fő Tartalom (main content)</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
        <p>Duis aute irure dolor in reprehenderit in voluptate velit esse ci
      </main>
      <nav>
        <ul>
          <li>Menü elem</li>
          <li>Menü elem</li>
          <li>Menü elem</li>
          <li>Menü elem</li>
        </ul>
      </nav>
      <aside>
        <h2>Oldalsáv (sidebar) </h2>
        <p>Kapcsolódó tartalom (related content)</p>
      </aside>
      <footer>
        <h2>Lábléc(footer)</h2>
      </footer>
    </div>
  </body>
</html>
```

```css
html {
  box-sizing: border-box;
}

*, *:before, *:after {
  box-sizing: inherit;
}

body {
  padding: 0;
  margin: 0;
}

h1, h2 {
  text-transform: uppercase;
}

/* CSS Grid */

.container > * {
  padding: 20px;
}

.container {
  color: white;
  font-family: helvetica, arial, sans-serif;
  display: grid;
  grid-template-columns: repeat(12, minmax(0, 1fr));
}

header {
  background: skyblue;
  grid-column: 1 / 13;
}
main {
  background: slategray;
  grid-column: 4 / 10;
}
nav {
  background: cadetblue;
  grid-column: 1 / 4;
}
aside {
  background: seagreen;
  grid-column: 10 / 13;
}

footer {
  background: darkslateblue;
  grid-column: 1 / 13;
}
```