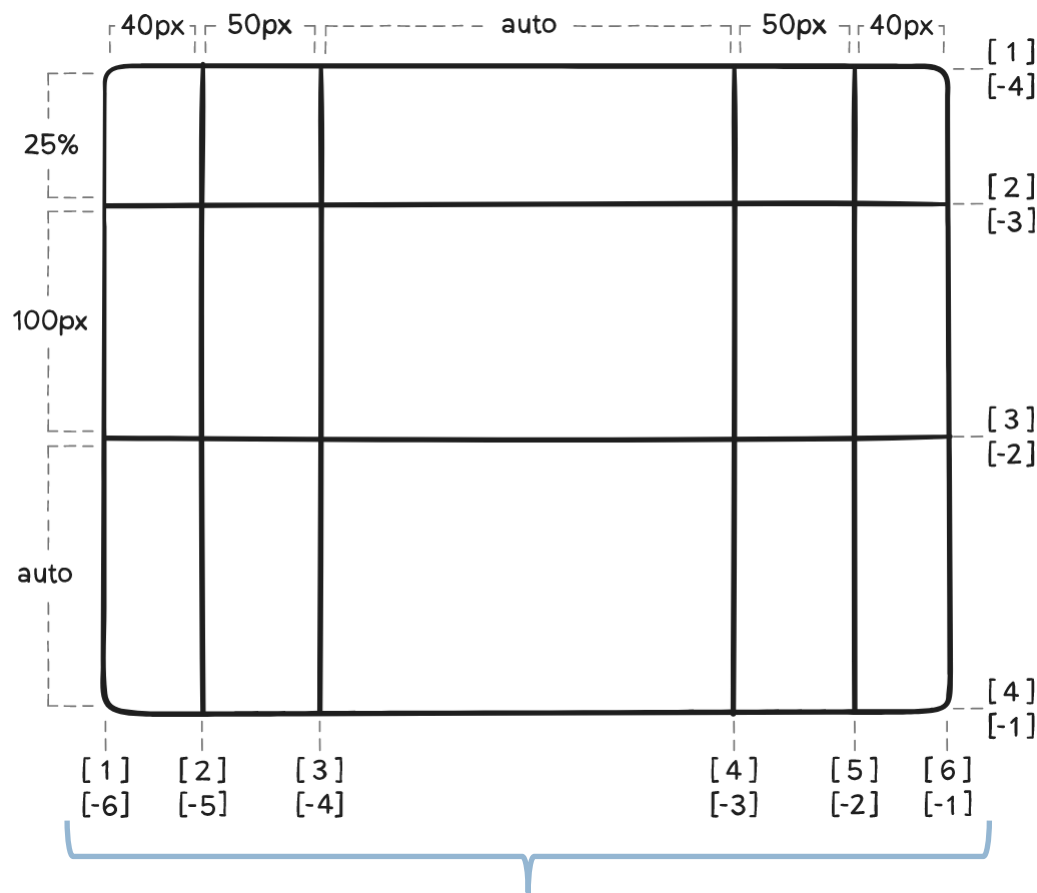


WEBY – CSS 12

Grid layout

Grid layout

- Na rozdiel od Flexboxu, ktorý je jednorozmerný layoutovací systém – pozdĺž hlavnej osi, grid je dvojrozmerný
- Opäť máme kontajner a položky.
 - ▣ v kontajneri nastavíme **display:grid;**
 - ▣ položky sú všetky jeho priame deti
- Kontajneru nastavíme tiež veľkosti riadkov a stĺpcov
 - ▣ `grid-template-columns: 40px 50px auto 50px 40px;`
 - ▣ `grid-template-rows: 25% 100px auto;`



číslovanie hraníc

Grid layout

- Ak chceme rozvrhnúť miesto rovnomerne môžeme použiť pomernú jednotku fr (fraction of free space)
 - ▣ napr. `grid-template-columns: 1fr 50px 2fr 1fr;`
 - vyrobí 4 stĺpce na celú šírku rodiča, kde druhý stĺpec má fixnú šírku 50px a tretí bude taký široký ako prvý a štvrtý dohromady
 - ▣ Podobne ako flex-grow
- Minmax - Ak chceme definovať interval možných dĺžok
 - ▣ `minmax(200px, 1fr);`
 - je to pomerná veľkosť 1, ale nie menej ako 200px
 - ▣ preferovaná je vždy väčšia hodnota
 - `minmax(200px, 300px)`
 - ▣ Podobne ako flex-shrink

Grid layout

- ako dĺžku vieme použiť okrem percentovej, fixnej a pomerej dĺžky aj max-content, min-content a fit-content

This is a phrase with several words.

This is a phrase with several words. ← **max-content**

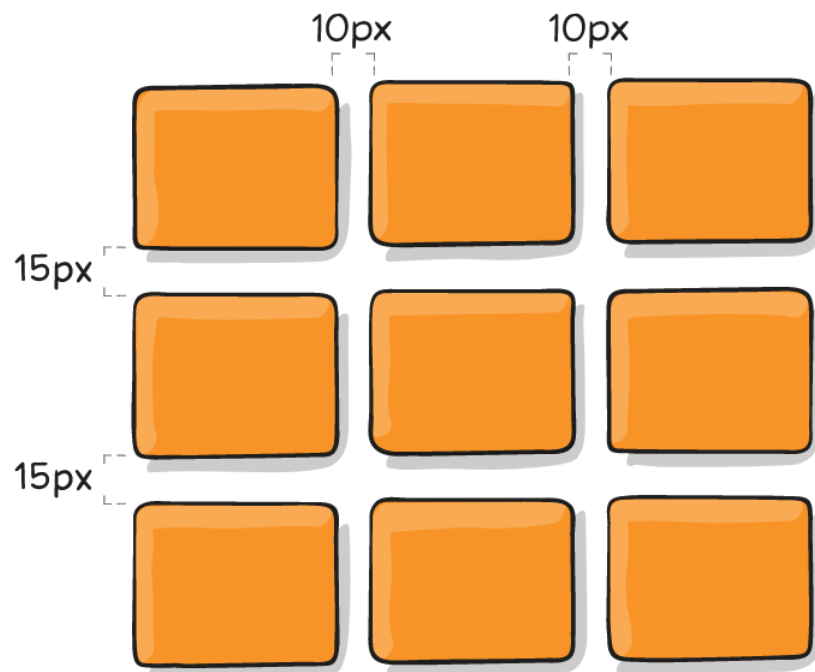
This is
a
phrase
with
several
words. ← **min-content**

Grid layout

- `repeat(počet, veľkosť)` - hromadné definovanie širok stĺpcov, alebo výšok riadkov
 - `grid-template-columns: repeat(3, minmax(20px, 1fr));`
 - ekvivaletne: `grid-template-columns: minmax(20px, 1fr) minmax(20px, 1fr) minmax(20px, 1fr);`
- `fit-content(veľkosť)` – veľkosť medzi min-content a hodnotou v zátvorke, ale nie viac ako max-content
- `grid-template`
 - Spoločný atribút pre riadky aj stĺpce
 - `grid-template: 1fr 250px 2fr / 1fr 100px`
 - `grid-teplate-rows: 1fr 250px 2fr;`
 - `grid-teplate-columns: 1fr 100px;`

Grid layout

- medzery medzi riadkami a stĺpcami:
 - **grid-column-gap:** 10px;
 - **grid-row-gap:** 15px;
- alebo
 - **column-gap:** 10px;
 - **row-gap:** 15px;
- alebo spolu cez
 - **grid-gap:** 15px 10px;
- alebo
 - **gap:** 15px 10px;
- pre stĺpce aj riadky
 - **gap:** 12px;

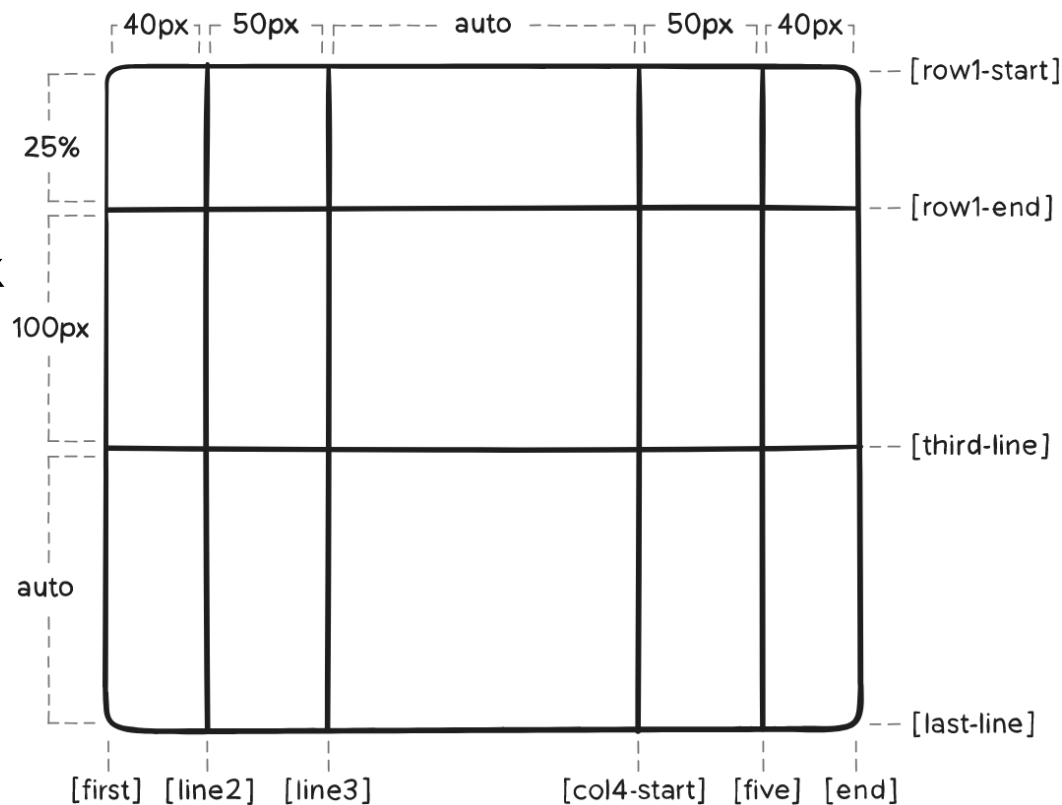


Grid layout

□ Hranice si môžeme aj pomenovať

□ `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]; }`



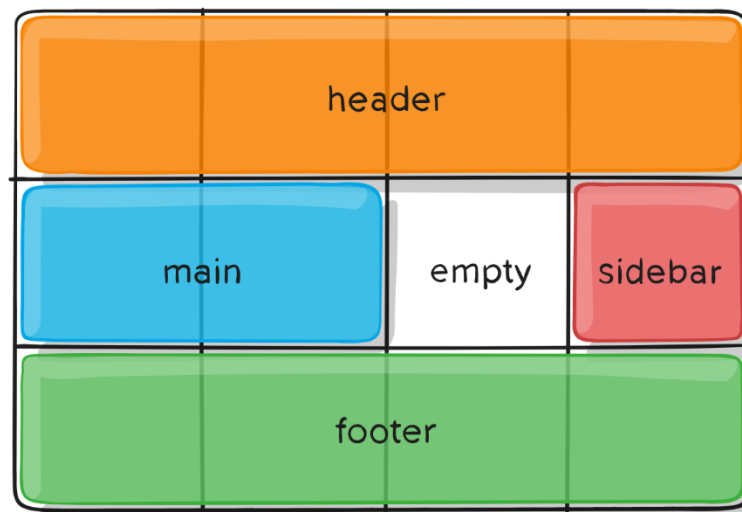
Grid layout - items

- umiestňovanie položiek podľa hraníc
 - ▣ definujeme ich pevné hranice zhora, zdola, sprava aj zľava, pričom ak sa niečo z toho nevedie, použije sa auto
 - ▣ riadok:
 - **grid-row-start:** <number> | <name> | span <number> | span <name> | auto;
 - **grid-row-end:** <number> | <name> | span <number> | span <name> | auto;
 - zjednotene: **grid-row:** <start-line> / <end-line> | <start-line> / span <value>;
 - ▣ stĺpec ekvivaletne cez grid-column-start, grid-column-end a grid-column
 - ▣ number = číslo hranice
 - ▣ name = meno hranice
 - ▣ span number = počet riadkov/stĺpcov položky
 - uvedené maximálne pri jednom: buď pri start-e alebo end-e
 - ▣ span name = až po najbližší riadok/stĺpec s týmto menom, ak ich je viac
 - ▣ auto = automatické uloženie, automatický span alebo span 1

Grid layout

- Alternatívou ku umiestňonaviu medzi hranice na základe čísiel hraníc alebo mien hraníc je vopred pomenovať oblasti (grid-area) a umiestňovať položky iba na základe názvu oblasti

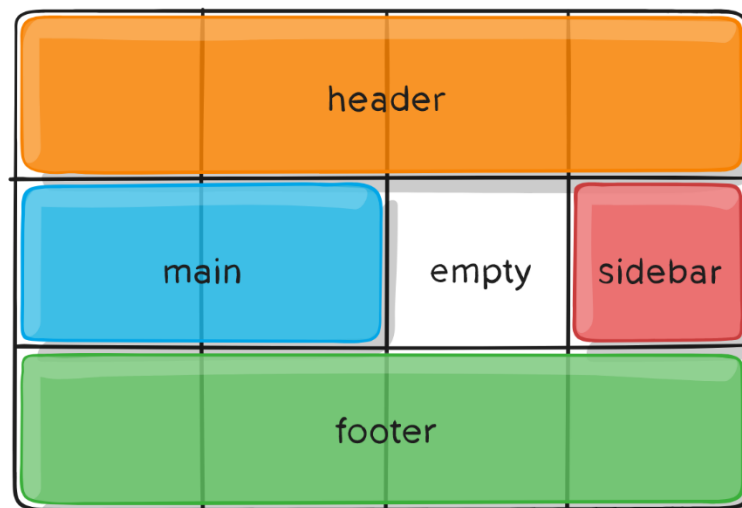
```
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}  
.item-f {  
  grid-area: footer;  
}
```



Grid layout

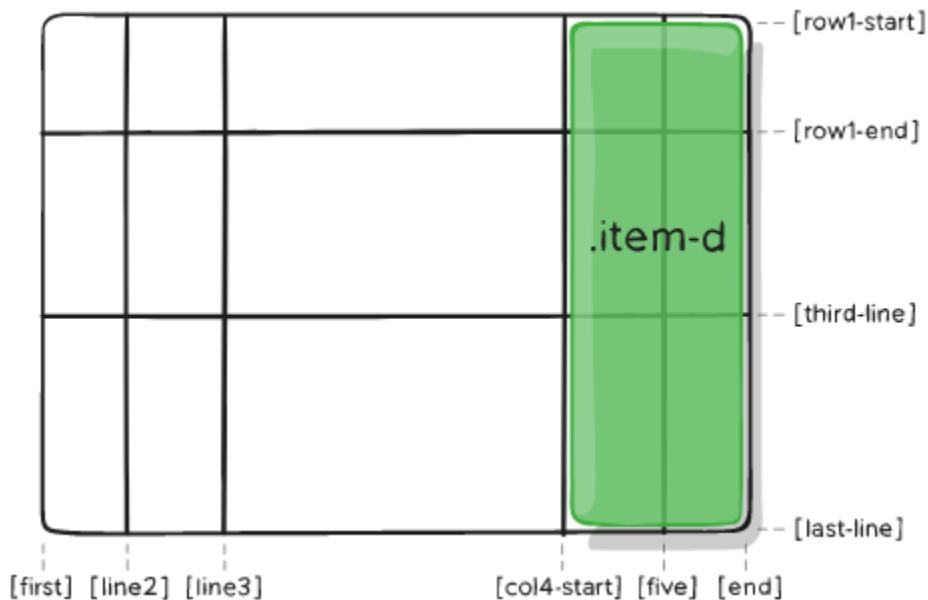
- hranice sú pomenované automaticky
 - ▣ riadky
 - 1 = header-start
 - 2 = header-end = main-start = sidebar-start
 - 3 = main-end = sidebar-end = footer-start
 - 4 = footer-end
 - ▣ stĺpce
 - 1 = header-start = main-start = footer-start
 - 2
 - 3 = main-end
 - 4 = sidebar-start
 - 5 = header-end = sidebar-end = footer-end

grid-area je obdĺžnik cez ľubovoľný počet riadkov a stĺpcov



Grid layout

- umiestňovanie položiek do oblasti
 - ▣ **grid-area:** <meno>
 - ▣ **grid-area:**<row-start> / <column-start> / <row-end> / <column-end>;
 - spojenie grid-row a grid-column
 - ▣ napr.
 - **grid-area:** 1 / col4-start / last-line / 6;



Grid layout – dynamické umiestnenie detí

- Ak nešpecifikujeme pre deti kontajnera (položky) ich umiestnenie, vkladajú sa postupne do buniek mriežky. Toto chovanie vieme prispôbiť:
 - **grid-auto-flow:** row | column | row dense | column dense
 - row (default) – zaplňajú sa najprv nezaplnené bunky prvého riadku, potom druhého atď. Ak nie je dost' riadkov pridajú sa automaticky ďalšie
 - column – zaplňajú sa najprv nezaplnené bunky prvého stĺpca, potom druhého atď. Prípadne sa pridajú ďalšie stĺpce
 - dense – ak sa niektoré časti mriežky museli vynechať, kvôli veľkým položkám, a neskôr sú menšie položky, vložia sa do týchto dier – položky sa tak vykreslia potenciálne v inom poradí, ako sú uvedené v html
- Podobne ako pri flexboxe, vieme položkám zmeniť poradie cez:
 - **order:** <integer>
 - položky najprv zoradí podľa hodnoty order a ak je ich viac s rovnakou hodnotou order ich vzájomné poradie je dané poradím v kontajneri

Grid layout – dynamické rozšírenie mriežky

- Ak je potrebné pridať extra riadky alebo stĺpce, ktoré v definícii mriežky neboli vieme nastaviť ich veľkosť, default je 0
 - **grid-auto-columns:** <veľkosť>
 - **grid-auto-rows:** <veľkosť>
- **grid**
 - zjednotený zápis pre grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns a grid-auto-flow
 - **grid:** <grid-template>
 - funguje ako grid-template
 - **grid:** <grid-template-rows> / [auto-flow && dense?] <grid-auto-columns>?
 - napr. grid: 100px 300px / auto-flow 60px;
 - 2 riadky, automatické stĺpce šírky 60px, grid-auto-flow: column
 - **grid:** [auto-flow && dense?] <grid-auto-rows>? / <grid-template-columns>
 - napr. grid: auto-flow dense 100px / 3fr 1fr;
 - 2 stĺpce, automatické riadky výšky 100px, grid-auto-flow: row dense

Grid layout – responzívny počet stĺpcov

- Počet opakovaní sa môže prispôbiť šírke kontajnera
- `grid-template-columns: repeat(auto-fill, šírka)`
- **auto-fill**
 - ▣ Vyrobití toľko stĺpcov šírky `šírka` koľko vojde do kontajnera bez ohľadu na počet detí
- **auto-fit**
 - ▣ Funguje ako **auto-fill**, ale ak je detí menej ako by vyrobil **auto-fill**, vyrobí len toľko stĺpcov, koľko je detí

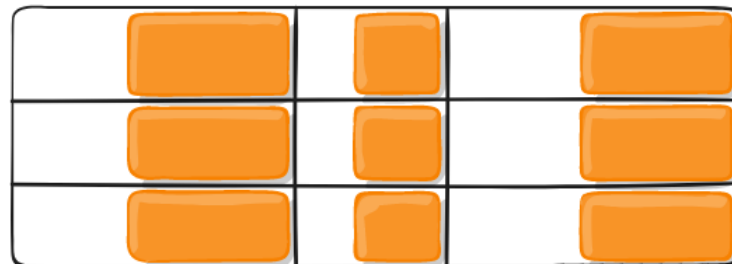
Grid layout

- Zarovnanie všetkých položiek v bunkách/oblastiach kontajnera horizontálne
 - ▣ **justify-items**: start | end | center | stretch (default);

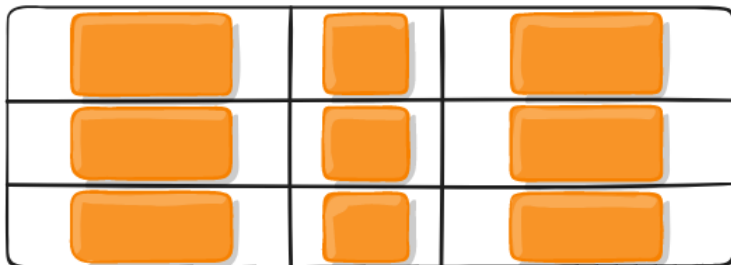
start



end



center



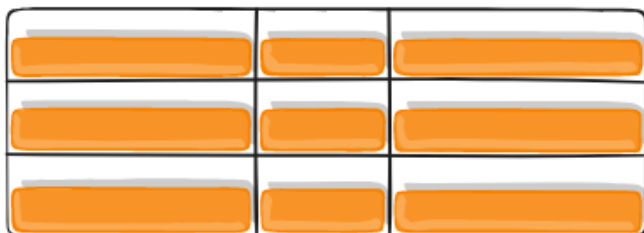
stretch



Grid layout

- Zarovnanie všetkých položiek v bunkách/oblastiach kontajnera vertikálne
 - ▣ **align-items:** start | end | center | stretch (default);

start



end



center



stretch



- Skrátенý zápis pre justify-items a align-items
 - ▣ **place-items:** <align-items> / <justify-items>;

Grid layout

- Ak chceme zarovnávať konkrétnu položku inak, ako zvyšok kontajnera:
 - **justify-self:** start | end | center | stretch;
 - **align-self:** start | end | center | stretch;
 - zjednotene:
 - **place-self:** auto (tak, ako kontajner - default)
 - **place-self:** <align-self> <justify-self>;
 - ak sa uvedie iba jedno, aplikuje sa aj na riadok aj stĺpec

Grid layout

- Ak pri definícii riadkov a stĺpcov použijeme iba fixné alebo percentové jednotky, tak mriežka môže byť aj menšia ako kontajner. Rozloženie riadkov a stĺpcov vo väčšom kontajneri nastavujeme cez:
 - horizontálne:
 - **justify-content:** start (default) | end | center | stretch | space-around | space-between | space-evenly;
 - vertikálne:
 - **align-content:** start (default) | end | center | stretch | space-around | space-between | space-evenly;
 - kombinácia:
 - **place-content:** <align-content> / <justify-content>;
 - **place-content:** <horizontálne aj vertikálne spolu ako 1 hodnota>

Grid layout

□ Horizontálne (vertikálne úplne analogicky):

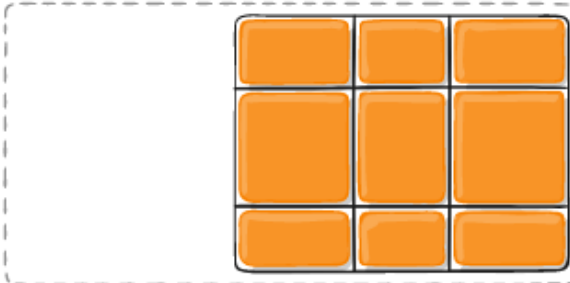
start

grid container



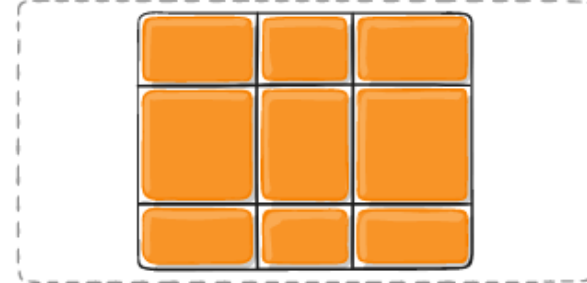
end

grid container



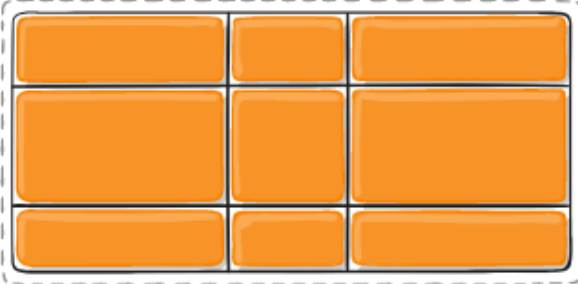
center

grid container



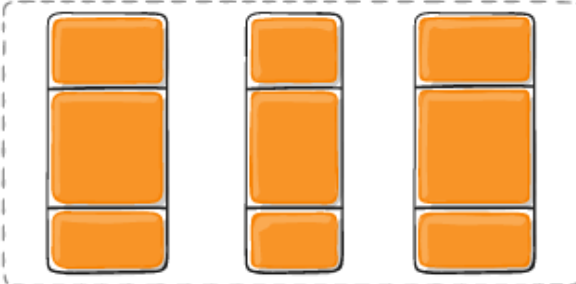
stretch

grid container



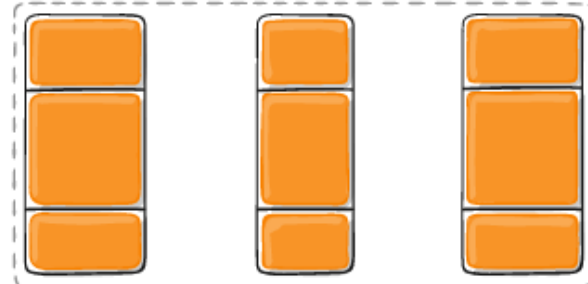
space-around

grid container



space-between

grid container



space-evenly

grid container

