RESPONZÍVNY WEB

Peter Gurský

Responzivita

Media queries

- Aby fungovali na všetkých zariadeniach správne, je potrebné do <head> elementu vložiť tag

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- V CSS si vieme vytvoriť sekciu, ktorá sa aplikuje iba pri splnených kritériách a nahradí tak pravidlá v hlavnej sekcii

```
@media screen and (max-width: 800px){

 /* selektory s pravidlami */

}
```

pravidlá

- Podľa šírky vieportu
  - @media (min-width : 1200px)
  - @media (max-width : 1200px)
- Podľa typu média
  - @media screen
  - @media print
  - @media all
- Podľa orientácie
  - @media (orientation: portrait)
  - @media (orientation: landscape)

Logické operácie

- @media screen and (orientation: portrait)
- @media (min-height: 600px) and (min-width: 800px)
- @media (max-width: 600px) or (min-width: 800px)

@media level 4 a 5

- https://developer.mozilla.org/en-US/docs/Web/CSS/@media
- Zatiaľ biedna podpora prehliadačov
- Od 2020:
  - @media (prefers-color-scheme: dark)
  - @media (prefers-color-scheme: light)
- Od 2023:
  - @media (width < 800px)
  - @media (width >= 900px)
  - @media (400px <= width <= 700px)
- Ďalšie užitočné: color, hover, any-hover, display-mode, prefers-reduced-motion

Bežné zalomenia

- Typické šírky obrazoviek:
  - Mobily 360px - 480px
  - Tablety 481px - 768px
  - Malé monitory 769px - 1024px
  - Väčšie monitory 1025px – 1920px
  - Veľké monitory nad 1920px
- Okná však nemusia byť na celú obrazovku

Vývoj responzívnej stránky

- Desktop-first
  - Začíname so širokou stránkou a upravujeme ju pre menšie obrazovky
- Mobile-first
  - Začíname s malou stránkou a upravujeme ju pre väčšie obrazovky

Kontajnery

- Nastavovanie responsivity elementov na základe veľkosti ich predkov (nie veľkosti viewportu)
- Nastavenie, že element je kontajner a má sa sledovať jeho veľkosť
  - container-type:
    - inline-size (sleduj iba šírku)
    - size (sleduj šírku aj výšku)

```
.content {
   container-type: inline-size;
}
.content .trieda {
   color: red;
}
```

```
@container (inline-size < 600px) {
   .trieda {
      color: purple;
   }
}
```

Kontajnery

- Ak chceme, aby @container pravidlo sledovalo veľkosť iba niektorých kontajnerov, môžeme kontajnery aj pravidlo pomenovať
  - container-name: meno;
- skratka:
  - container: meno / inline-size;

```
.content {
   container-type: inline-size;
   container-name: moj;
}
.content .trieda {
   color: red;
}
```

```
@container moj (inline-size < 600px) {
   .trieda {
      color: purple;
   }
}
```

Flexbox

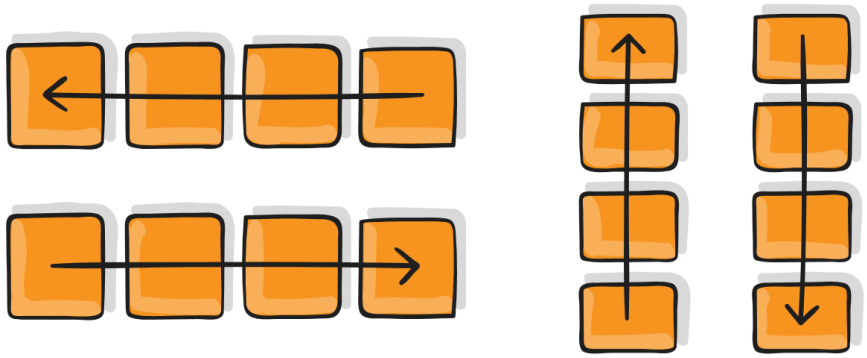
- Metóda na umiestňovanie obsahu po riadkoch alebo stĺpcoch
- Rodičovský komponent hovorí svojim deťom do
  - ich veľkosti
  - ich umiestnení v rodičovi
  - ich umiestnení navzájom
  - distribúcii zvyšného voľného miesta v sebe

Flexbox layout

- základný koncept – kontajner, ktorý rozmiestňuje svoje deti - položky (items)

```
.container {
  display: flex;
}
```

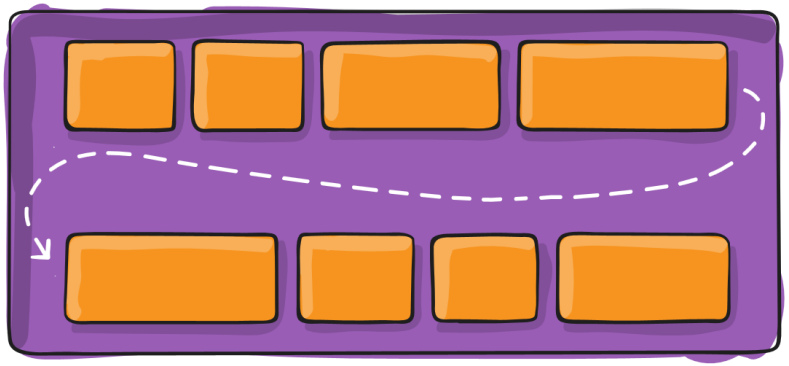
- kontajneru môžeme určiť v akom smere budú položky rozmiestňované (default je row)
  - flex-direction: row | row-reverse | column | column-reverse
  - smer závisí od jazyka (napr. hebrejčina spôsobí, že row ide doľava)
  - zadefinujeme tým **hlavnú os**



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox container

- defaultne sa všetky položky snažia vopchať do jedného riadku/stĺpca. Môžem ich však dať zalamovať – správajú sa ako text
  - flex-wrap: nowrap | wrap | wrap-reverse;
    - wrap-reverse usporiadava riadky zdola nahor

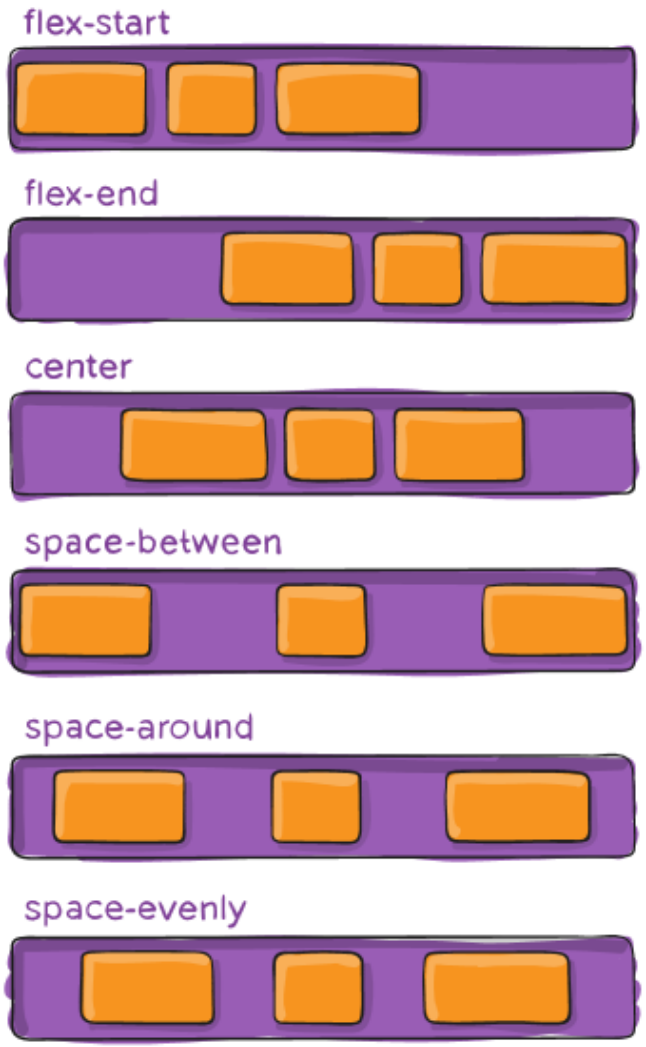


- zjednotený zápis pre direction a wrap
  - flex-flow: <direction> <wrap>
  - default je “row nowrap”

obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox container

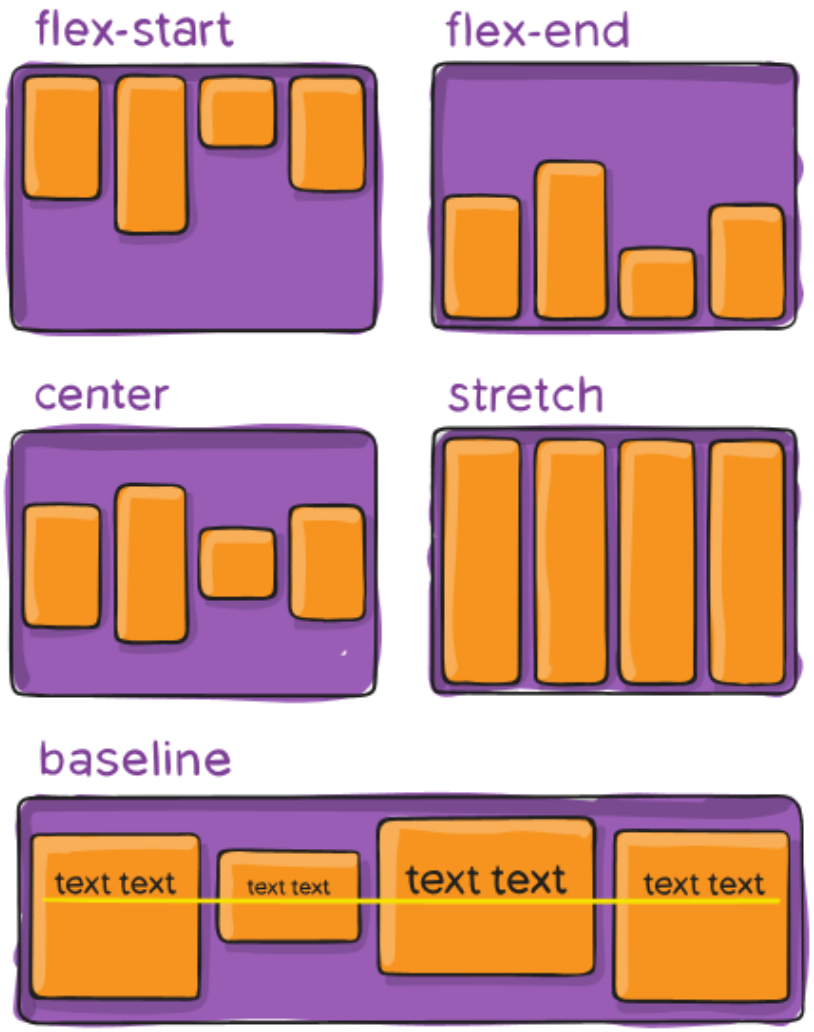
- rozmiestňovanie pozdĺž hlavnej osi
- v kontajneri nastavíme
  - justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly
  - default je flex-start



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox container

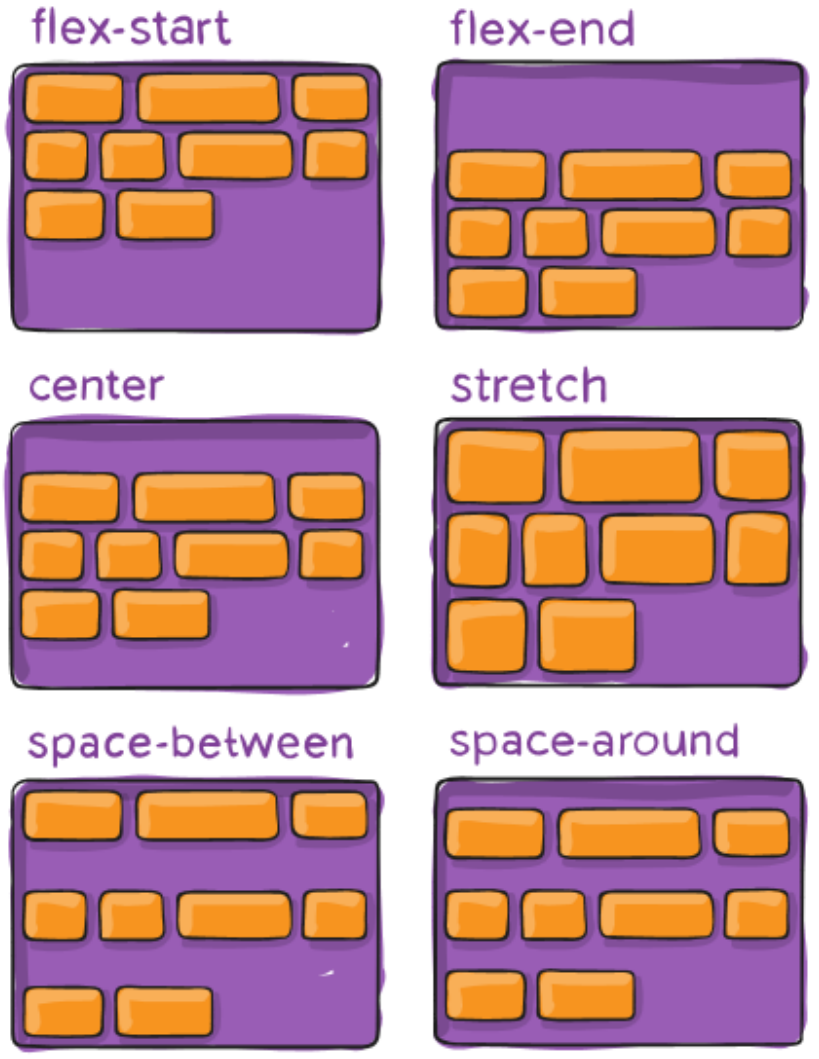
- rozmiestňovanie kolmo na hlavnú os
- v kontajneri nastavíme
  - align-items: stretch | flex-start | flex-end | center | baseline
  - default je stretch



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox container

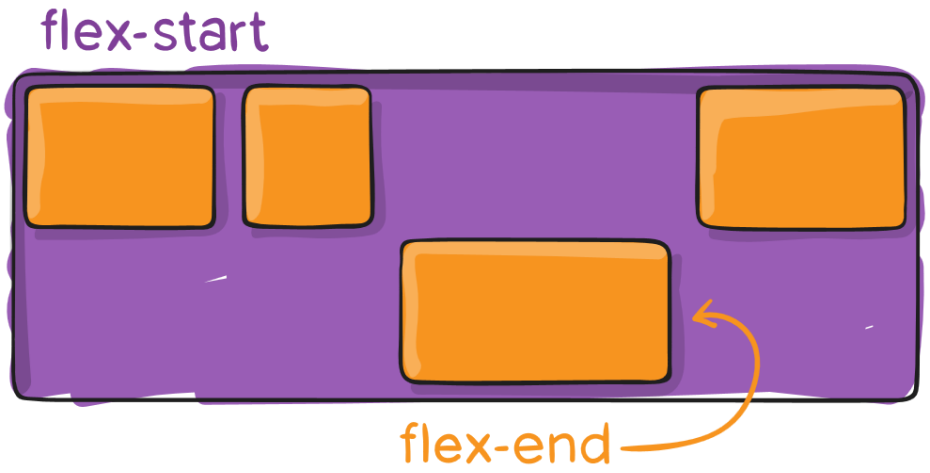
- rozmiestňovanie kolmo na hlavnú os **ak máme wrap**
- riadok je taký vysoký, ako jeho najvyšší item plus koľko si ukradne z voľného miesta
- v kontajneri nastavíme
  - align-content: flex-start | flex-end | center | space-between | space-around | stretch
  - default je stretch (každý riadok kradne rovnako z voľného miesta)



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox item

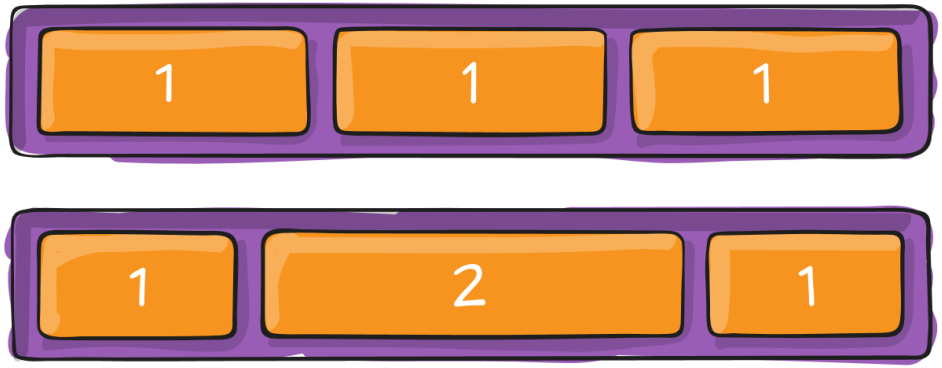
- umiestnenie kolmo na hlavnú os pre konkrétnu položku inak ako pre zvyšok kontajnera
  - align-self: auto | flex-start | flex-end | center | baseline | stretch;



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox item

- prispôsobenie veľkosti položky pozdĺž hlavnej osi v závislosti od voľného miesta, ak je kontajner širší ako súčet pôvodných šírok položiek
  - flex-grow: <číslo>;
  - default je 0 – nezväčšuje sa
  - číslo hovorí aký pomer z voľného miesta si ukradne voči ostatným položkám (3 si ukradne 3x koľko miesta ako 1)



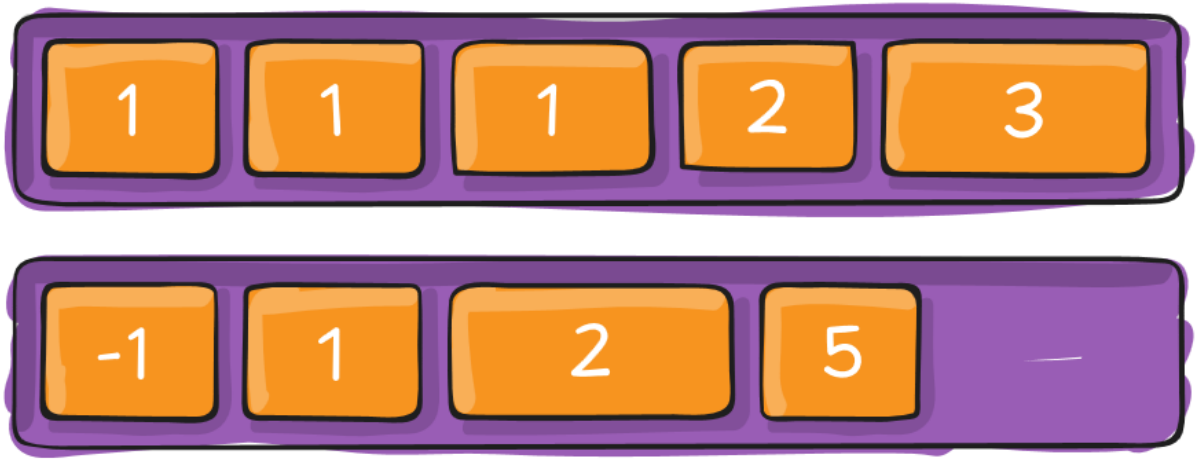
obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox item

- prispôsobenie veľkosti položky pozdĺž hlavnej osi v závislosti od nedostatku miesta, ak je kontajner užší ako súčet pôvodných šírok položiek
  - flex-shrink: <číslo>;
  - default je 1 – zmenšuje sa v pomere 1
  - číslo hovorí ako veľmi sa položka zmenší pri úzkom kontajneri voči ostatným položkám (3 sa vzdá 3x viac miesta ako 1)
- počiatočná veľkosť položky pozdĺž hlavnej osi pred zväčšovaním/zužovaním
  - flex-basis: <veľkosť>| auto;
  - default je auto – ideme podľa originálnej šírky/výšky položky
- kombinácia všetkých troch (odporúčané)
  - flex: none | auto | flex-grow flex-shrink flex-basis;
  - initial (default) už vieme: "0 1 auto“
  - none: 0 0 auto, auto: 1 1 auto;
  - druhý a tretí parameter sú nepovinné

Flexbox item

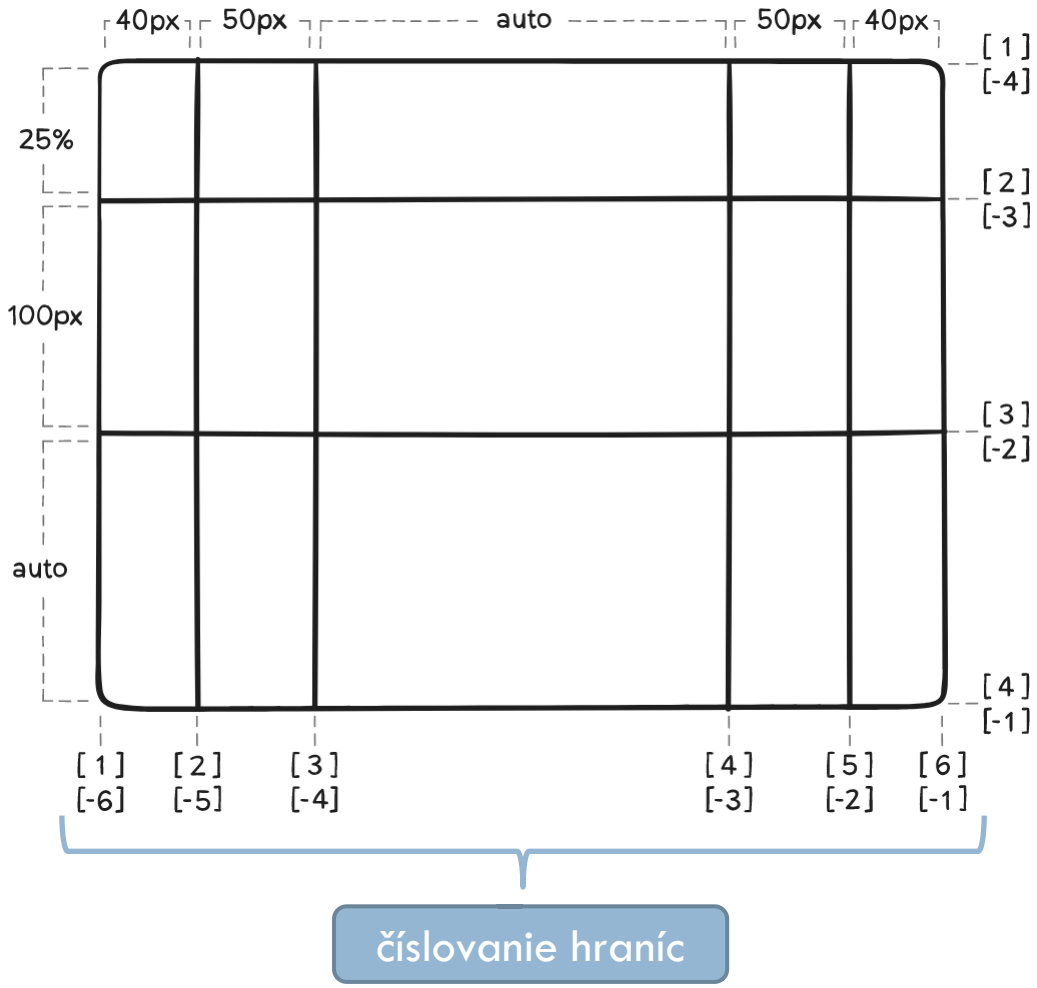
- zmena poradia položiek v kontajneri
  - order: <integer>;
  - default je 0
  - položky najprv zoradí podľa hodnoty order a ak je ich viac s rovnakou hodnotou order ich vzájomné poradie je dané poradím v kontajneri



obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

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;



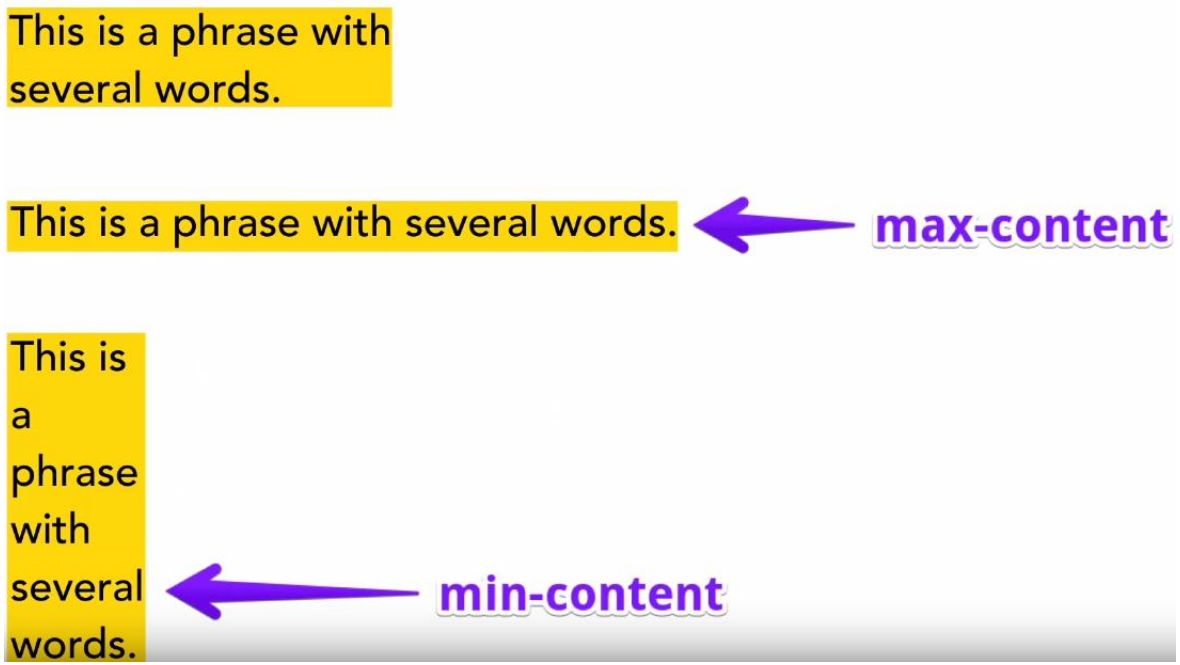
obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

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



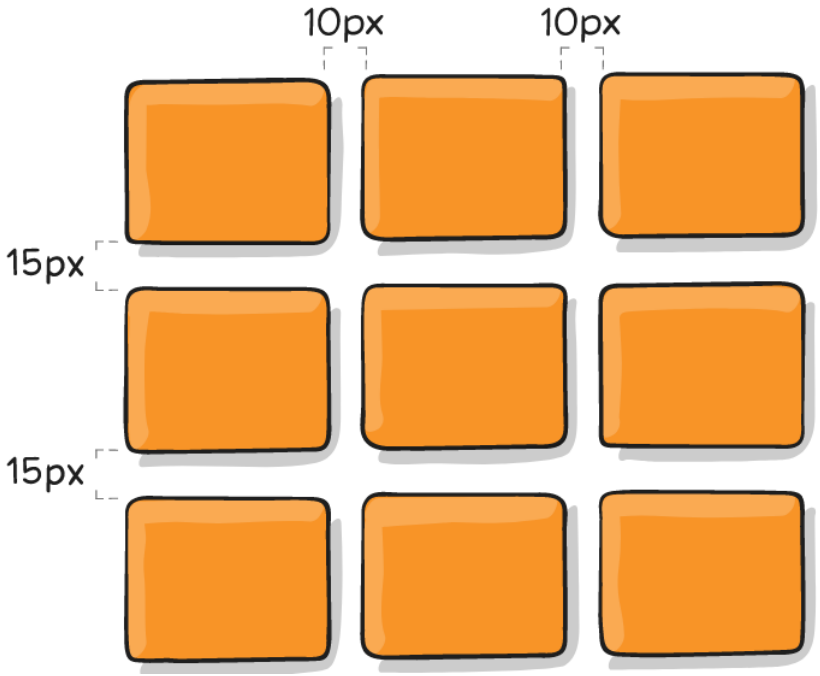
obrázok z: https://www.youtube.com/watch?v=IZ2JX_6SGNI

Grid layout

- repeat(počet, veľkosť) - hromadné definovanie šírok 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-tempate: 1fr 250px 2fr / 1fr 100px
    - grid-teplate-rows: 1fr 250px 2fr;
    - grid-teplate-columns: 1fr 100px;

Grid layout

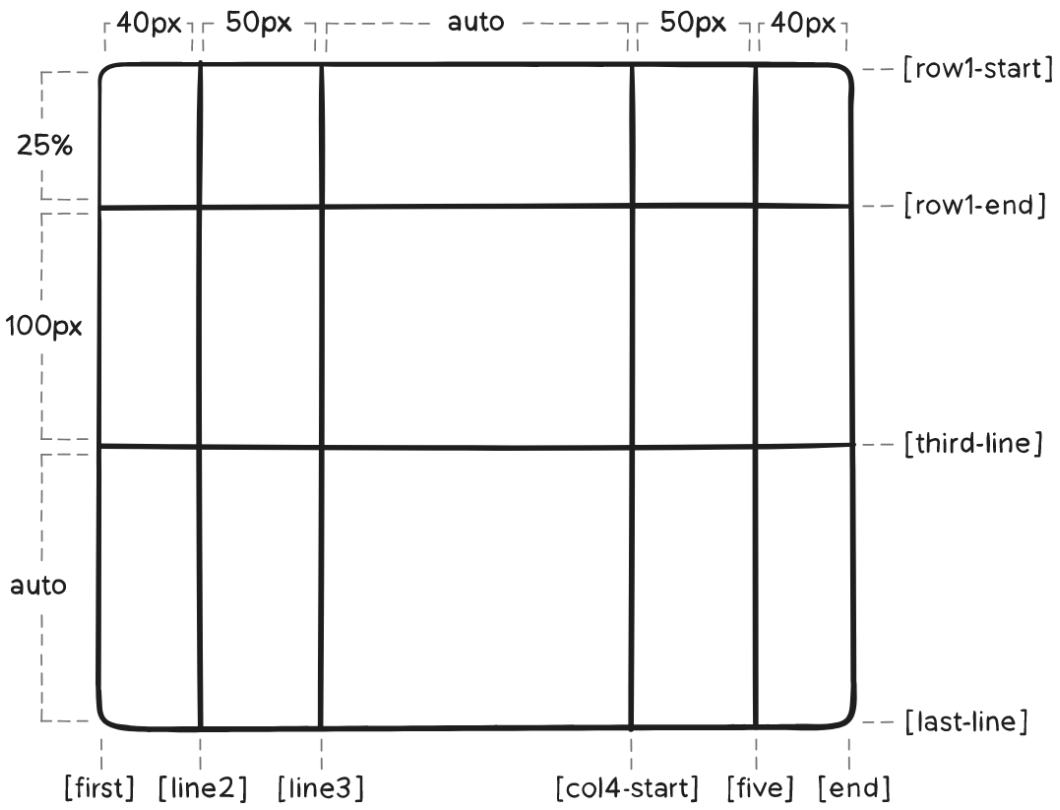
- medzery medzi riadkami a stĺpcami:
  - grid-column-gap: 10px;
  - grid-row-gap: 15px;
- alebo
  - column-gap: 10px;
  - row-gap: 15px;
- alebo spolu cez
  - grid-gap: 15px 10px;
- alebo
  - gap: 15px 10px;
- pre stĺpce aj riadky
  - gap: 12px;



obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

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]; }



obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

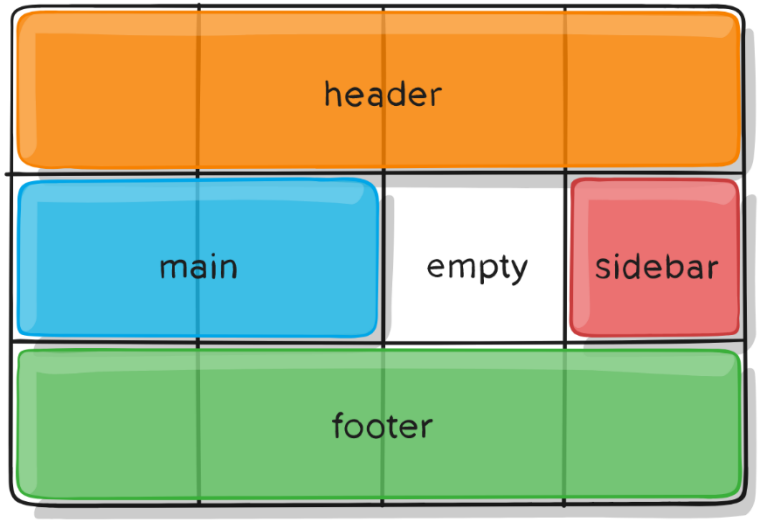
Grid layout - items

- umiestňovanie položiek podľa hraníc
  - definujeme ich pevné hranice zhora, zdola, sprava aj zľava, pričom ak sa niečo z toho neuvedie, 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

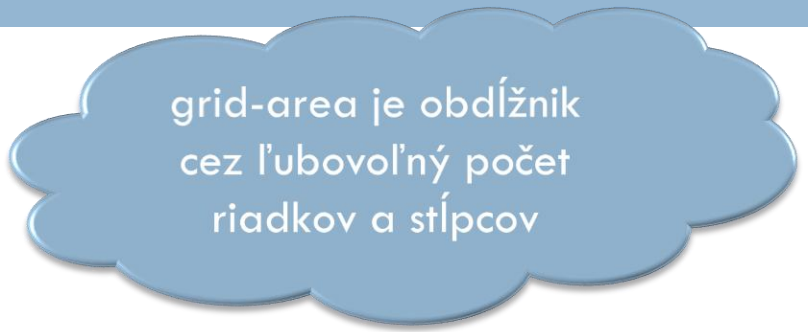
```
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
.item-f {
  grid-area: footer;
}
```

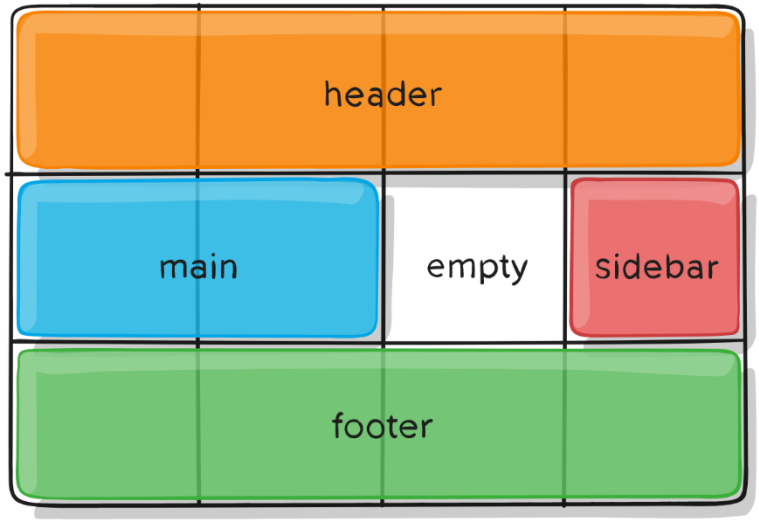


obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

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

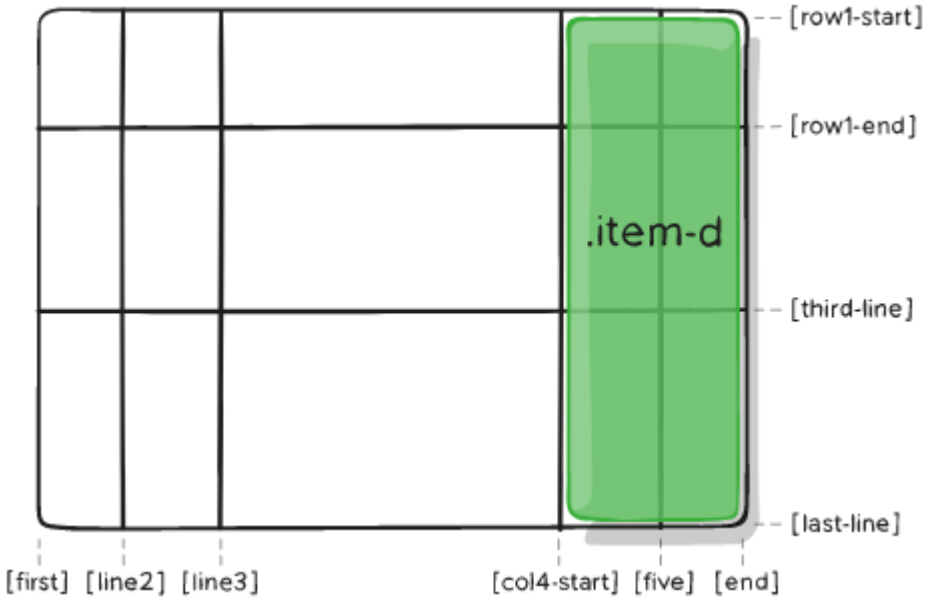




obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

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;



obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

Grid layout – dynamické umiestenie detí

- Ak nešpecifikujeme pre deti kontajnera (položky) ich umiestnenie, vkladajú sa postupne do buniek mriežky. Toto chovanie vieme prispôsobiť:
  - grid-auto-flow: row | column | row dense | column dense
    - row (default) – zapĺňajú sa najprv nezaplnené bunky prvého riadku, potom druhého atď. Ak nie je dosť riadkov pridajú sa automaticky ďalšie
    - column – zapĺňajú 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

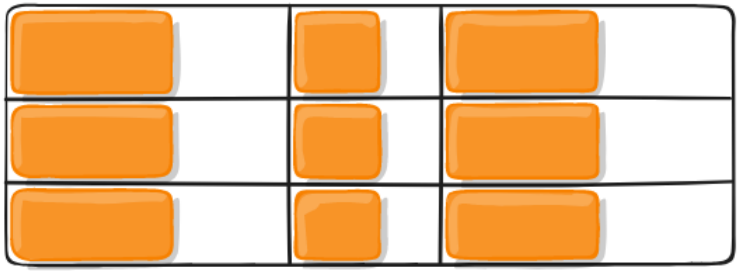
Grid layout – responzívny počet stĺpcov

- Počet opakovaní sa môže prispôsobiť šírke kontajnera
- grid-template-columns: repeat(auto-fill, šírka)
- auto-fill
  - Vyrobí 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

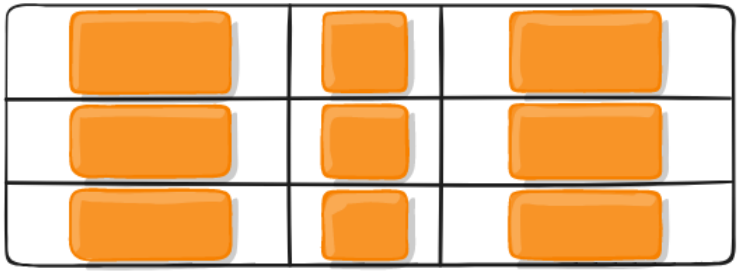
- Zarovnanie všetkých položiek v bunkách/oblastiach kontajnera horizontálne
  - justify-items: start | end | center | stretch (default);

start

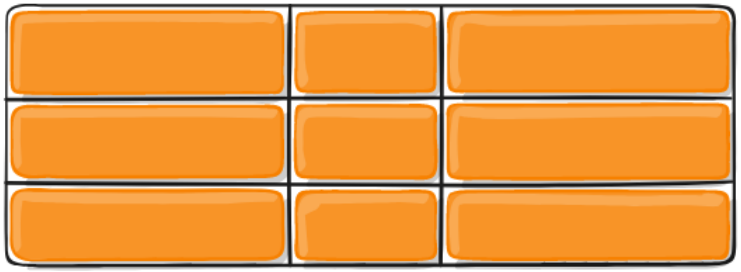


end

center



stretch

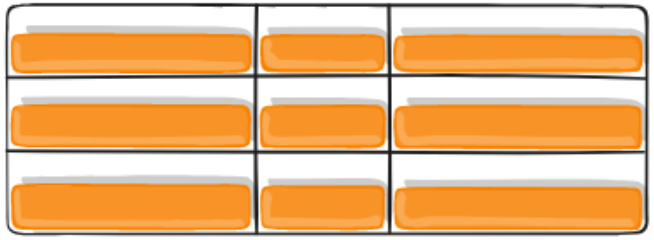


obrázky z: https://css-tricks.com/snippets/css/complete-guide-grid/

Grid layout

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

start



end

center

stretch

- Skrátený zápis pre justify-items a align-items
  - place-items: <align-items> / <justify-items>;

obrázky z: https://css-tricks.com/snippets/css/complete-guide-grid/

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):