POZICIOVANIE A LAYOUT V CSS

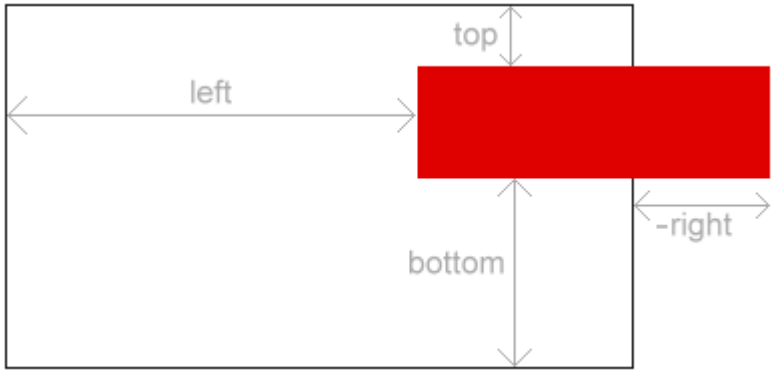
Peter Gurský

Vlastnosť position

- možné hodnoty
  - static – default hodnota
  - relative
  - absolute
  - fixed
  - sticky

Relatívne poziciovanie

- nechá alokované miesto tam, kde bol element pôvodne vyrenderovaný
- samotný element sa posunie
  - top | bottom, left | right




```
.shiftme {
   position: relative;
   color: red;
   bottom: 5px;
   left: 15px;
}
```

```
<div>Lorem ipsum dolor sit amet, <span class="shiftme">consectetur</span> adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</div>
```

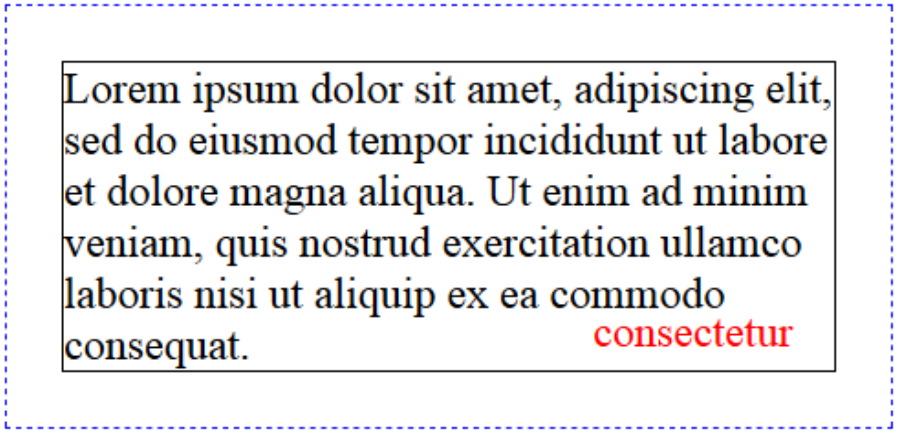
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Absolútne poziciovanie

- vyjme element z úvodného renderovania
- umiestni ho voči najbližšiemu predkovi, ktorý nie je staticky poziciovaný, v našom príklade voči elementu s class-om "parent",
  - ak taký nie je, umiestni ho voči elementu body

```
.grandparent {
   border: 1px dashed blue;
}
.parent {
   position: relative;
   margin: 20px;
   border: 1px solid black;
}
.absol{
   position: absolute;
   color: red;
   bottom: 5px;
   right: 15px;
}
```

```
<div class="grandparent">
   <div class="parent">
      Lorem ipsum dolor sit amet, <span class="absol">consectetur</span>
      adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.
   </div>
</div>
```

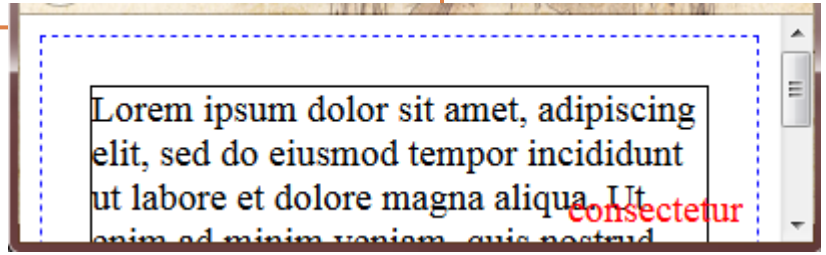


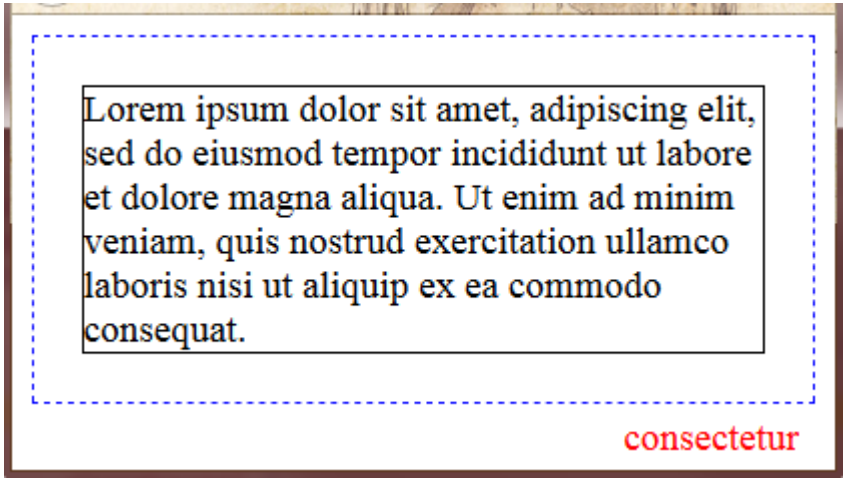
Fixované poziciovanie

- vyjme element z úvodného renderovania
- umiestni ho voči oknu prehliadača (na poziciovanie predkov sa nepozerá)

```
.grandparent {
   border: 1px dashed blue;
}
.parent {
   position: relative;
   margin: 20px;
   border: 1px solid black;
}
.shiftme {
   position: absolute;
   color: red;
   bottom: 5px;
   right: 15px;
}
```

```
<div class="grandparent">
  <div class="parent">
     Lorem ipsum dolor sit amet, <span class="shiftme">consectetur</span>
     adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
     magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
     laboris nisi ut aliquip ex ea commodo consequat.
  </div>
</div>
```





Umiestnenie na stred okna

```
.middle {
   position: fixed;
   border: 1px solid blue;
   width: 15ex;
   height: 5ex;
   line-height: 5ex;
   top: calc(50% - (5ex / 2));
   left: calc(50% - (15ex / 2));
   text-align: center;
}
```

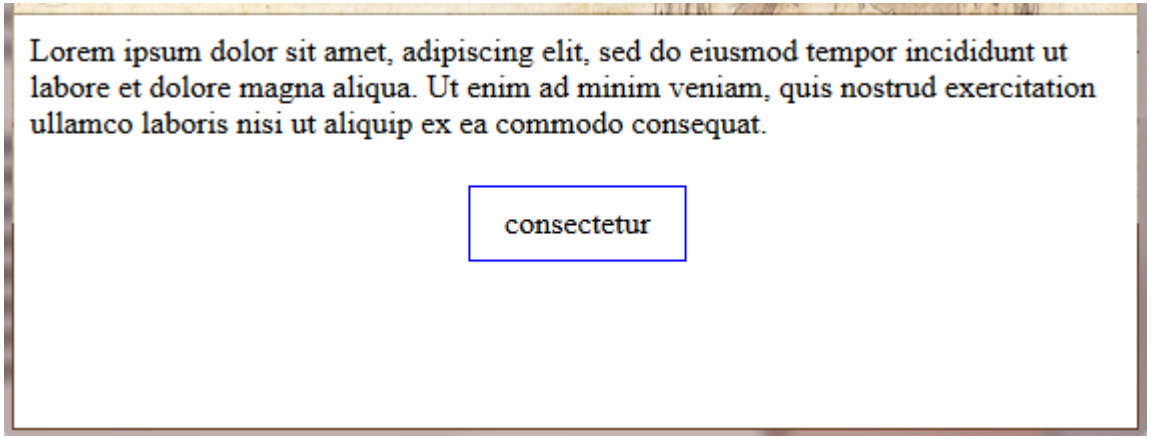
predpokladáme fixnú veľkosť elementu

ak je hodnota rovnaká ako výška elementu, centruje text vertikálne na stred

vyrátame horný a ľavý okraj elementu na základe jeho veľkosti tak, aby stred elementu bol v strede okna

horizontálne centrovanie textu v elemente

Lorem ipsum dolor sit amet, **consectetur** adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



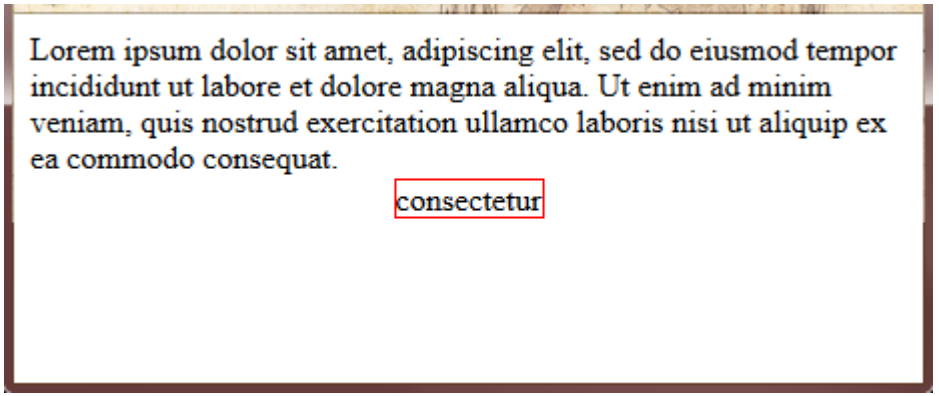
Umiestnenie na stred okna

```
.middle {
   position: fixed;
   border: 1px solid red;
   top: 50%;
   left: 50%;
   transform: translate(-50%,-50%);
}
```

základné umiestnenie s ľavým horným rohom v strede okna prehliadača

posunieme v ose x aj y o polovicu šírky/výšky elementu

Lorem ipsum dolor sit amet, **consectetur** adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



Vlastnosť transform

- -ms-transform: pre IE 9, -webkit-transform pre Safari < 9
- ako hodnota je niektorá z funkcií
  - translate(posunx, posuny)
    - posun ako pri relative, ale ak použijeme %, neberie sa rozmer rodiča, ale elementu samotného
  - rotate(20deg)
    - otočí element o 20 stupňov v smere hodinových ručičiek
  - scaleX(nasobokx), scaleY(nasoboky), scale(nasobokx,nasoboky)
    - zväčší element v osi x a y
  - skewX(20deg), skewY(30deg), skew(20deg, 30deg)
    - zošikmí element horizontálne a verikálne
  - matrix(scaleX,skewY,skewX,scaleY,translateX,translateY)
    - kombinácia všetkých transformácií

poziciovanie cez sticky

- Safari má -webkit-sticky,
- IE a Edge < 16 sticky nepodporujú
- poziciuje sa ako relative pokiaľ nezačne používateľ skrolovať stránku.
- obvykle sa iba nastaví vlastnosť top, na označenie hornej hranice, nad ktorú sa už element nesmie vyrolovať v rámci okna prehliadača

```
.menu {
   position: -webkit-sticky; /* Safari */
   position: sticky;
   top: 0;
}
```

Obtekanie

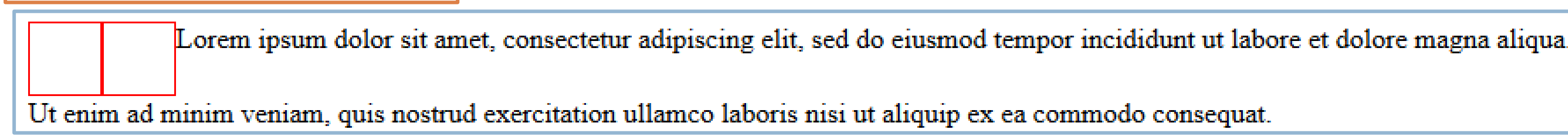
- float: left | right
  - posunie objekt v danej výške dokumentu doľava/doprava pokiaľ nenarazí na ďalší plávajúci element alebo okraj rodičovského elementu
- clear: left | rigth | both
  - ak je v danej výške rodičovského elementu nejaký plávajúci element naľavo / napravo / z oboch strán posunie sa až pod neho,
  - teda nedovolí vedľa seba plávajúci element zľava / sprava / z oboch strán

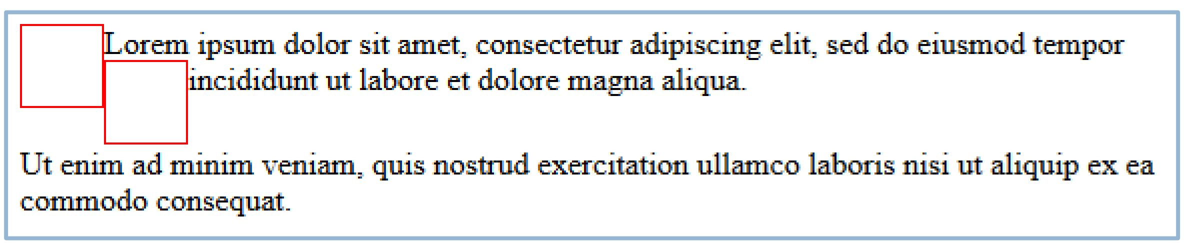
Obtekanie

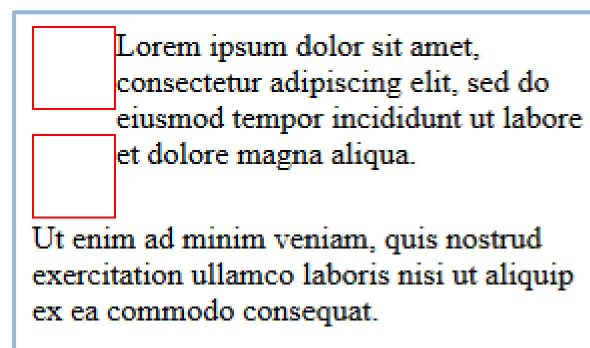
```
.box {
  display: block;
  float: left;
  border: 1px solid red;
  width: 40px;
  height: 40px;
}
.cleared {
  clear: both;
}
```

Lorem ipsum dolor sit amet, **<div class="box"></div>** consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore **<div class="box"></div>** magna aliqua.

<div class="cleared">
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
</div>





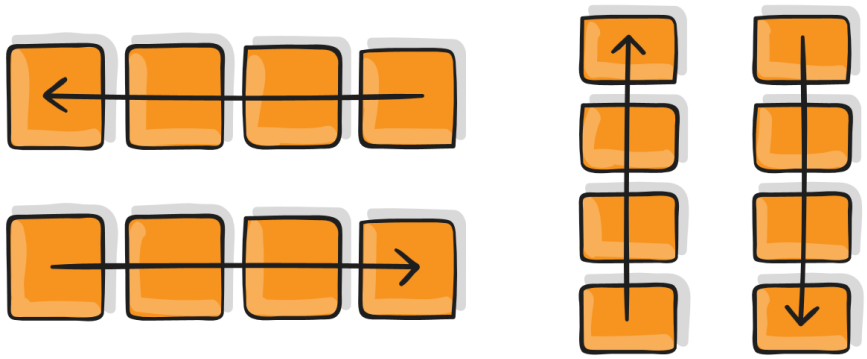


Flexbox layout

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

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

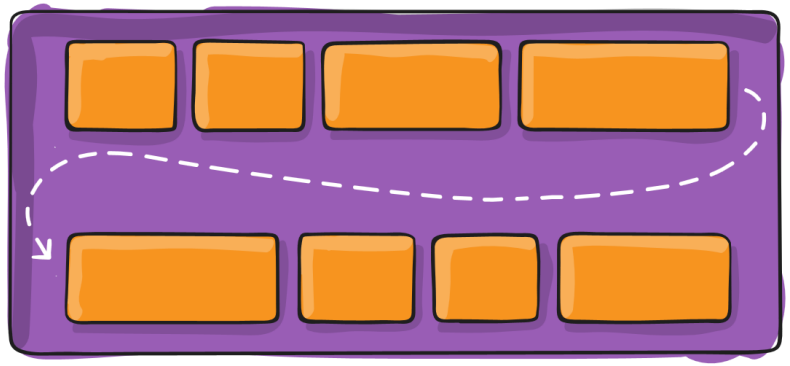
- kontajneru môžeme určiť v akom smere budú položky rozmiestňované (default je row)
  - flex-direction: row | row-reverse | column | column-reverse
  - zadefinujeme tým **hlavnú os**



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

flexbox

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

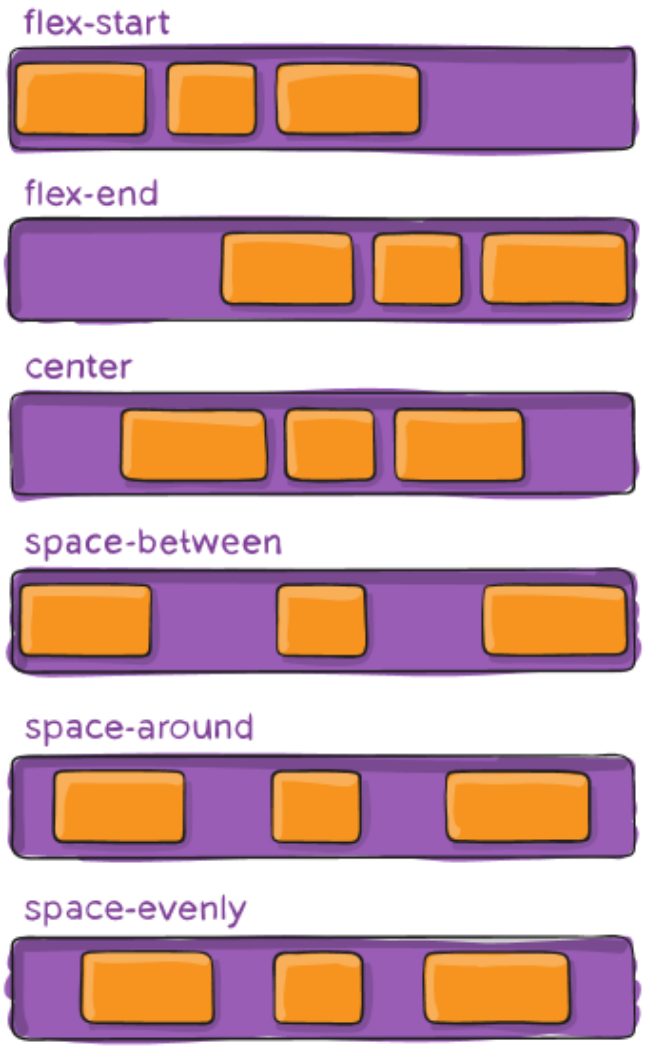


- zjednotený zápis pre direction a wrap
  - flex-flow: <direction> <wrap>
  - default je “row nowrap”

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

flexbox

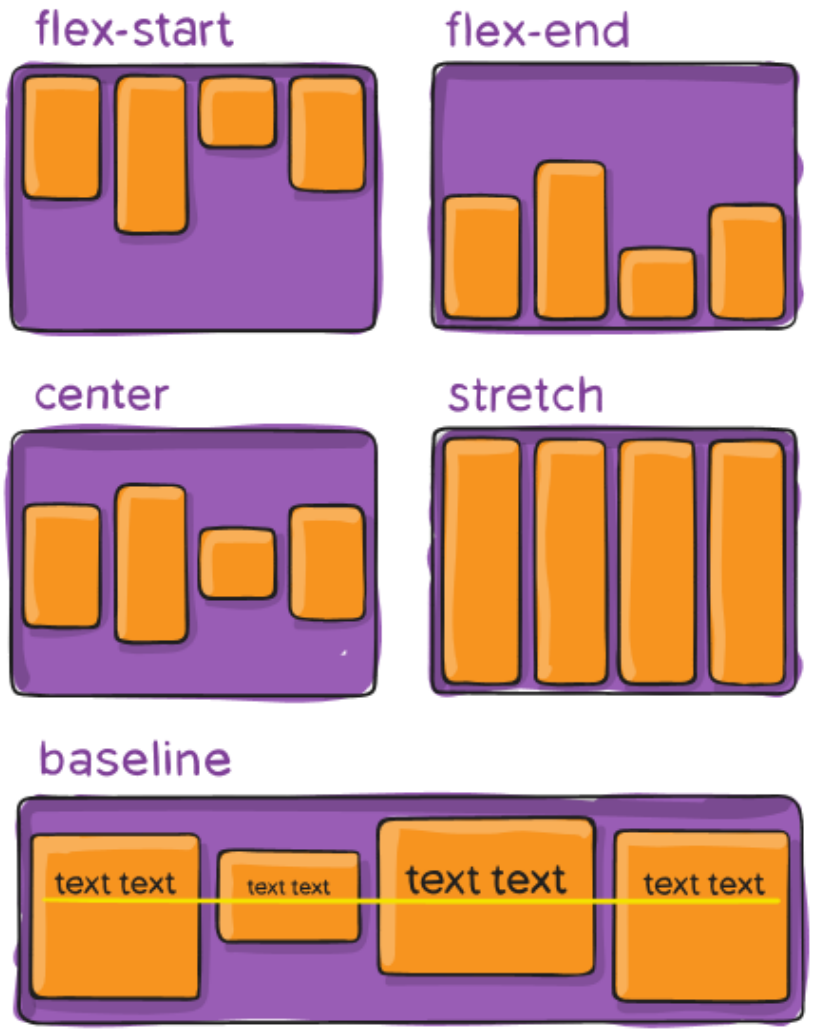
- rozmiestňovanie pozdĺž hlavnej osi
- v kontajneri nastavíme
  - justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly
  - default je flex-start



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

flexbox

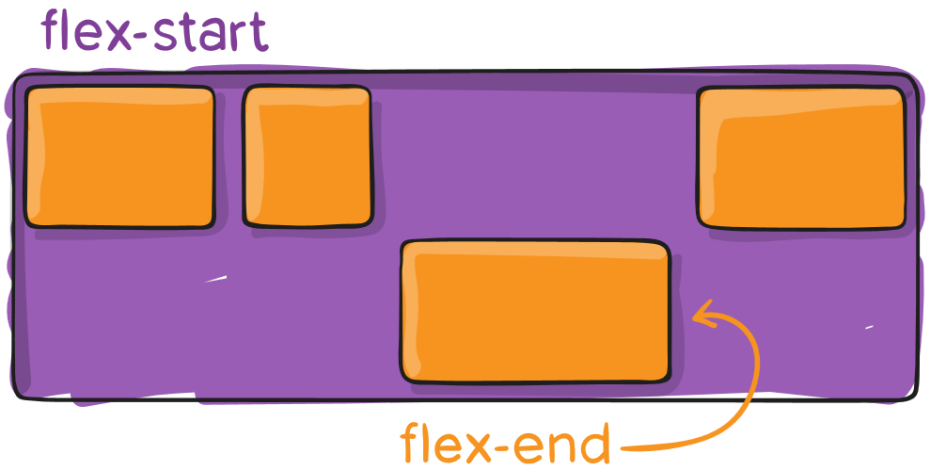
- rozmiestňovanie kolmo na hlavnú os
- v kontajneri nastavíme
  - align-items: stretch | flex-start | flex-end | center | baseline
  - default je stretch



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

flexbox

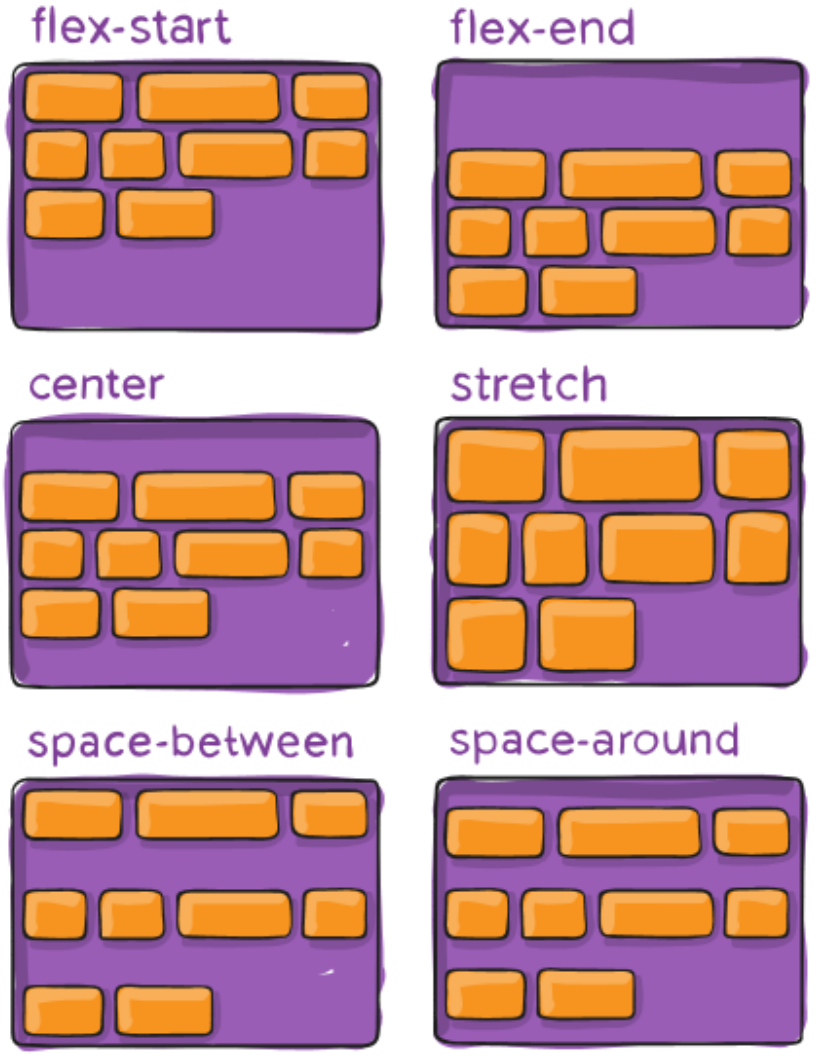
- umiestnenie kolmo na hlavnú os pre konkrétnu položku inak ako pre zvyšok kontajnera
  - align-self: auto | flex-start | flex-end | center | baseline | stretch;



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

flexbox

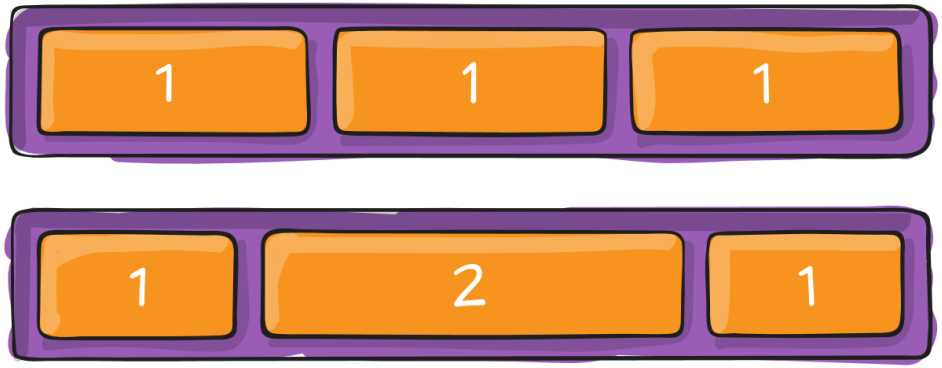
- rozmiestňovanie kolmo na hlavnú os ak máme wrap
- v kontajneri nastavíme
  - align-content: flex-start | flex-end | center | space-between | space-around | stretch
  - default je stretch



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

flexbox

- 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)



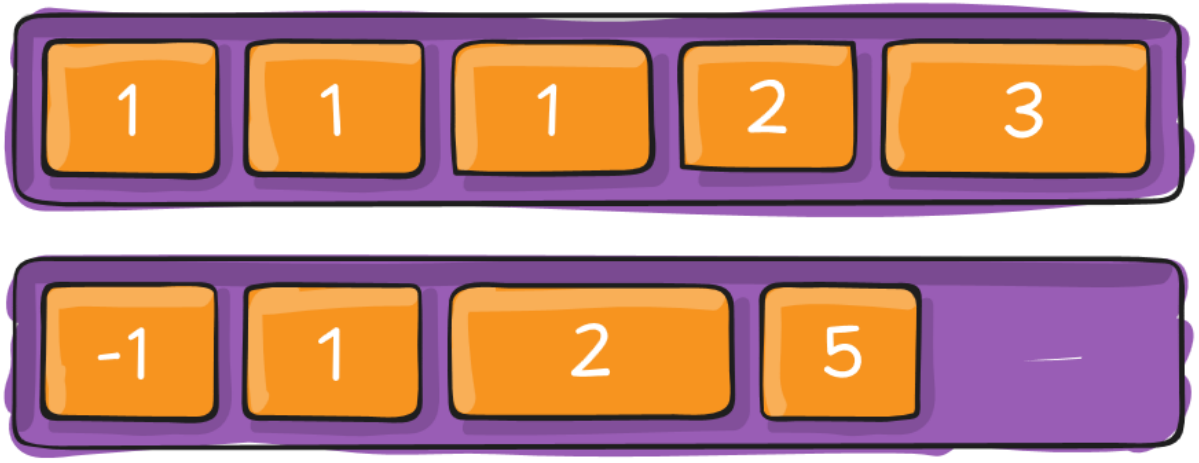
obrázok z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

flexbox

- 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 | flex-grow flex-shrink flex-basis;
  - default už vieme: "0 1 auto“
  - druhý a tretí parameter sú nepovinné

flexbox

- 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/

@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">
```

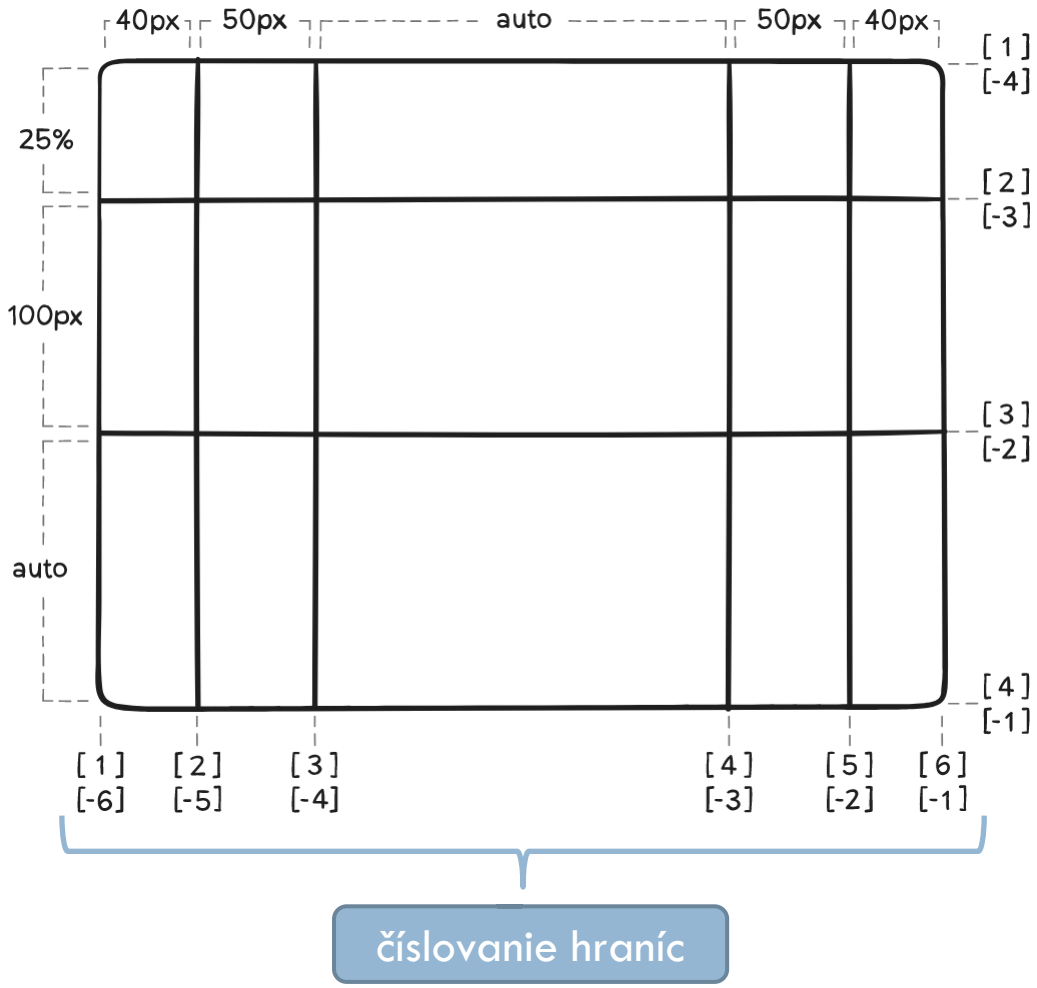
- V CSS si vieme vytvoriť sekciu, ktorá sa aplikuje iba pri splnených kritériách a prepíše tak pravidlá v hlavnej sekcii
  - @media typ_média and vlastnosť_média { … }
    - typ_média
      - screen (obrazovka počítača, tabletu, smartfónu,...), print (tlačiareň), speech (čítač obrazovky), all (všetky uvedené),
    - vlastnosť_média
      - min-width, max-width, min-height, max-height : <číslo>px
      - orientation: portrait (default), landscape

Testovanie responzibility

- Am I Responsive?
  - http://ami.responsivedesign.is/
- Mobile Friendly Test
  - https://search.google.com/test/mobile-friendly
- https://www.webdesignerdepot.com/2017/10/7-free-tools-for-testing-responsive-layouts/

Grid layout

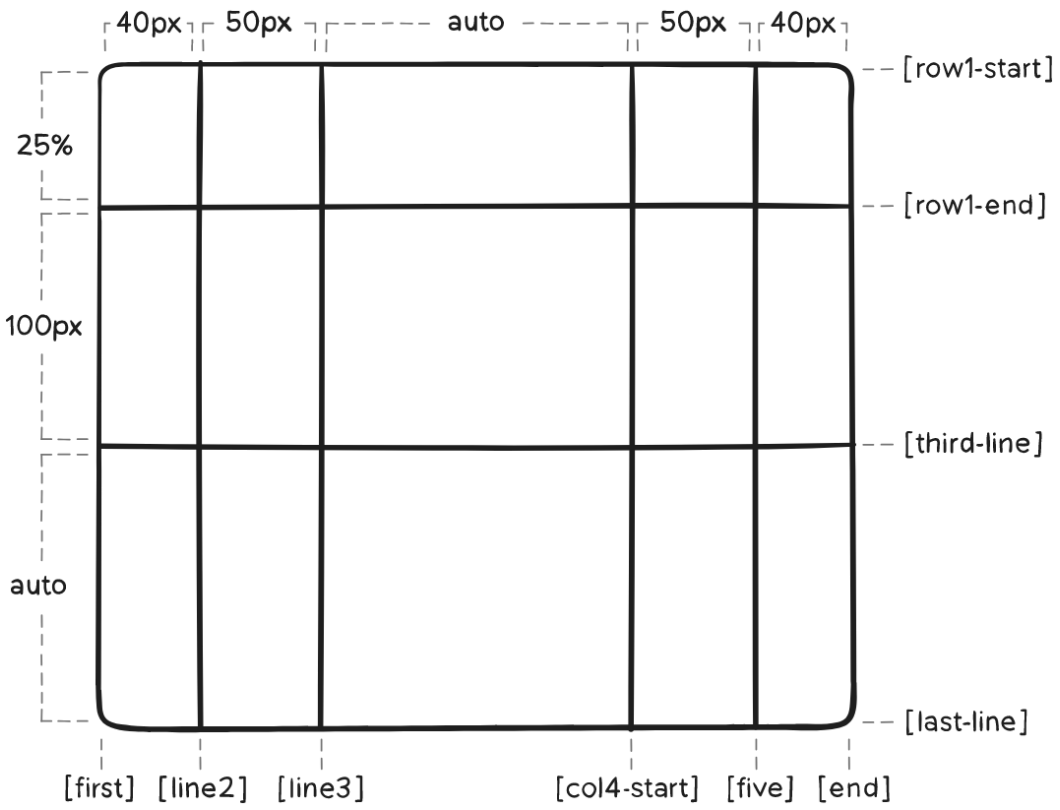
- Na rozdiel od Flexboxu, ktorý je jednorozmerný layoutovací systém – pozdĺž hlavnej osi, grid je dvojrozmerný
- Opäť máme kontajner a položky.
  - v kontajneri nastavíme display:grid;
  - položky sú všetky jeho priame deti
- Kontajneru nastavíme tiež veľkosti riadkov a stĺpcov
  - grid-template-columns: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



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

Grid layout

- 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

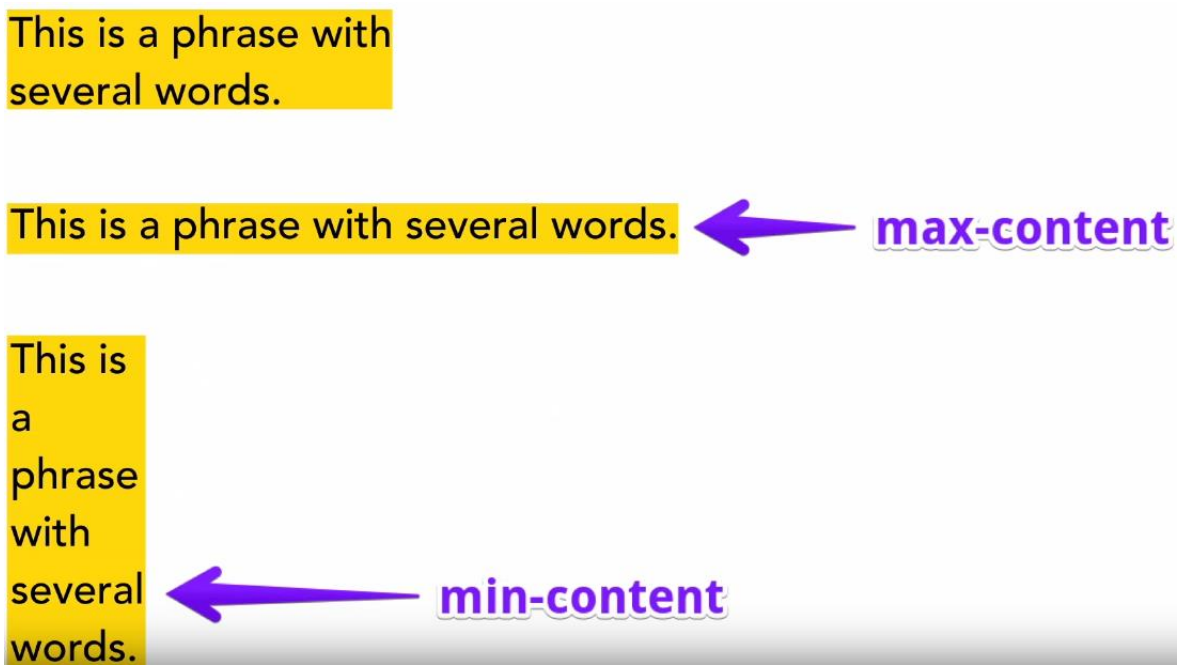
- Extra funkcie a hodnoty
  - ak definujeme šírky riadkov alebo stĺpcov, vieme si ich navyrábať pomocou funkcie repeat(počet, veľkosť)
    - napr. grid-template-columns: repeat(3, 20px [col-start]);
    - ekvivaletne: grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];

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
- Ak chceme skombinovať pomernú a presnú
  - minmax(200px, 1fr);
    - je to pomerná veľkosť 1 ale nie menej ako 200px

Grid layout

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

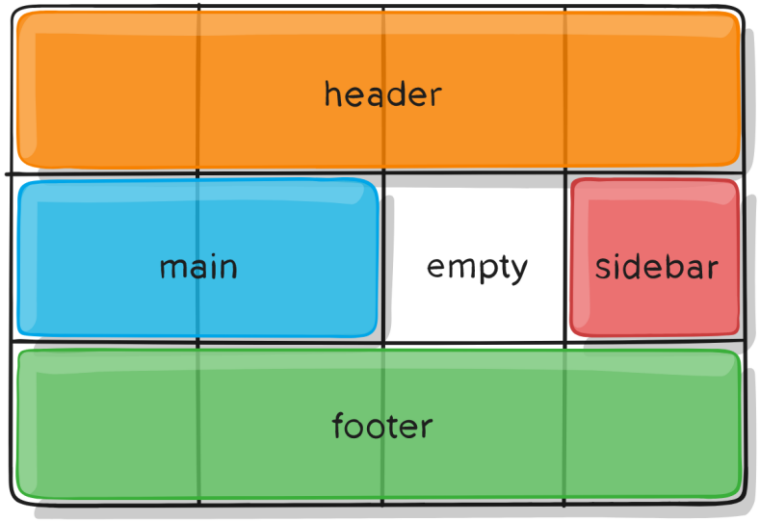


obrázok z: https://www.youtube.com/watch?v=IZ2JX_6SGNI

Grid layout

- Ako uvidíme neskôr položky v kontajneri umiestňujeme medzi hranice na základe čísiel hraníc alebo mien hraníc (prípadne kombináciou)
- Alternatívou je vopred pomenovať oblasti (grid-area) a umiestňovať položky iba na náklade 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;
}
```

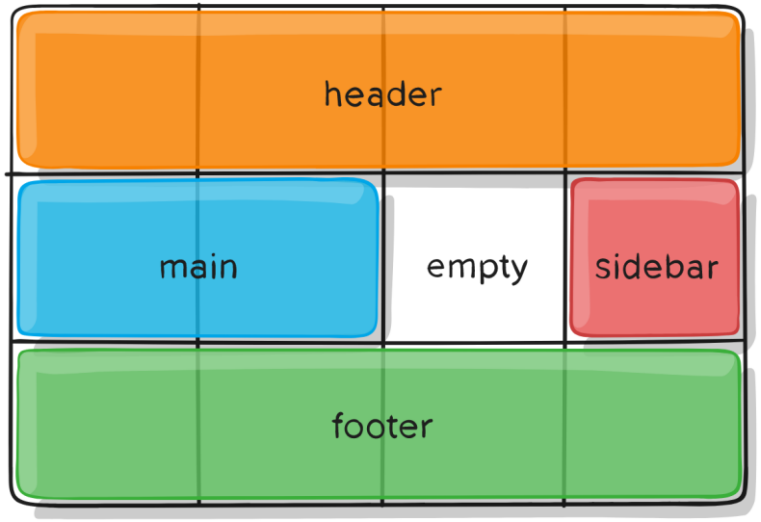


obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

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

grid-area je obdĺžnik cez ľubovoľný počet riadkov a stĺpcov



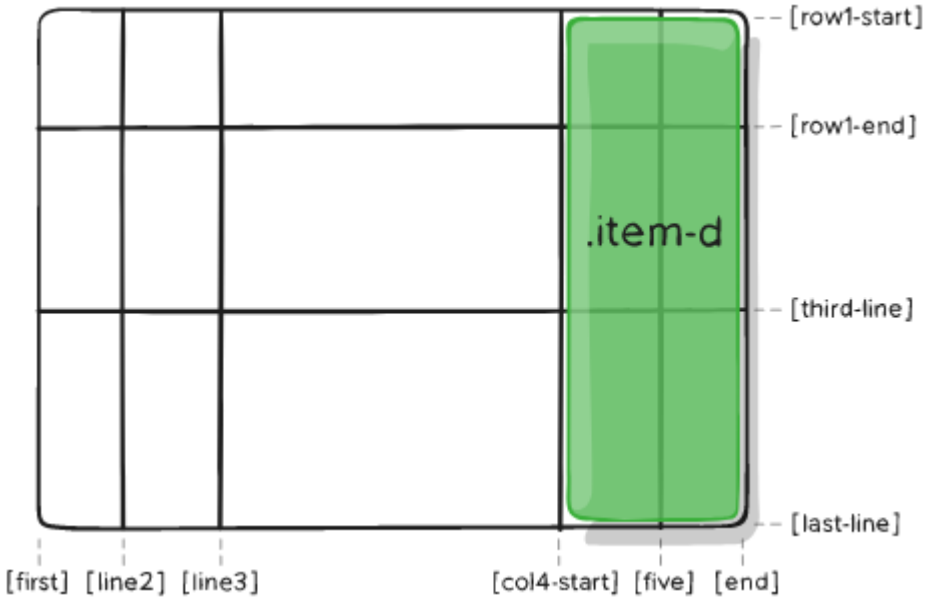
obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

Grid layout

- 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

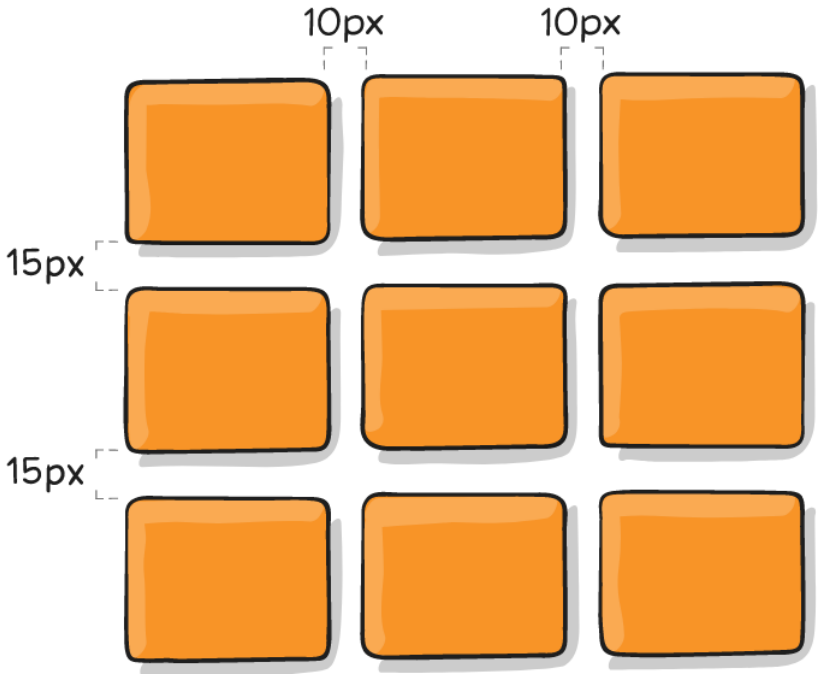
- umiestňovanie položiek do oblasti
  - grid-area: <meno>
  - grid-area:<row-start> / <column-start> / <row-end> / <column-end>;
    - spojenie grid-row a grid-column
  - napr.
    - grid-area: 1 / col4-start / last-line / 6;



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

Grid layout

- medzery medzi riadkami a stĺpcami:
  - grid-column-gap: 10px;
  - grid-row-gap: 15px;
- v najnovších prehliadačoch
  - column-gap: 10px;
  - row-gap: 15px;
- alebo spolu cez
  - grid-gap: 15px 10px;
- v najnovších prehliadačoch
  - gap: 15px 10px;

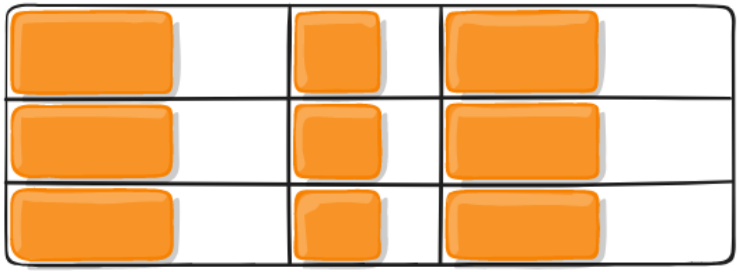


obrázok z: https://css-tricks.com/snippets/css/complete-guide-grid/

Grid layout

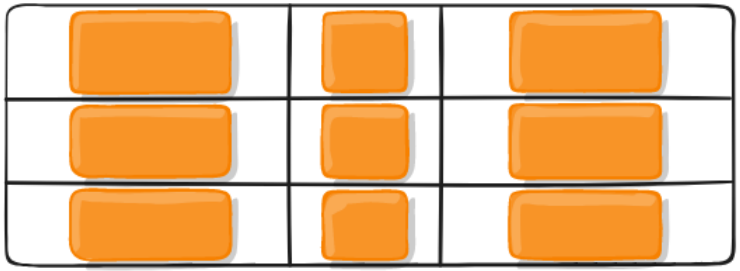
- Zarovnanie všetkých položiek v bunkách/oblastiach kontajnera horizontálne
  - justify-items: start | end | center | stretch (default);

start

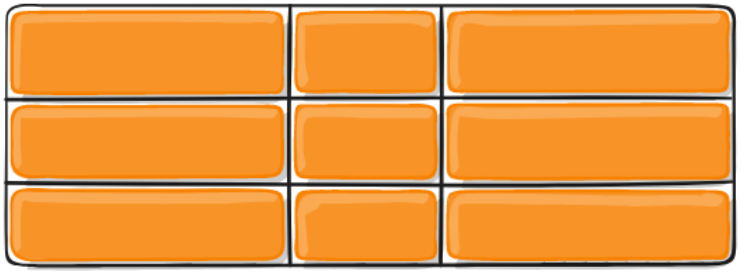


end

center



stretch

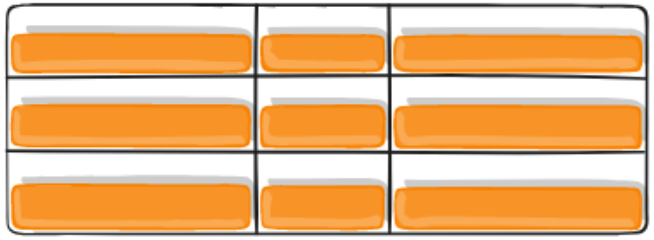


obrázky z: https://css-tricks.com/snippets/css/complete-guide-grid/

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

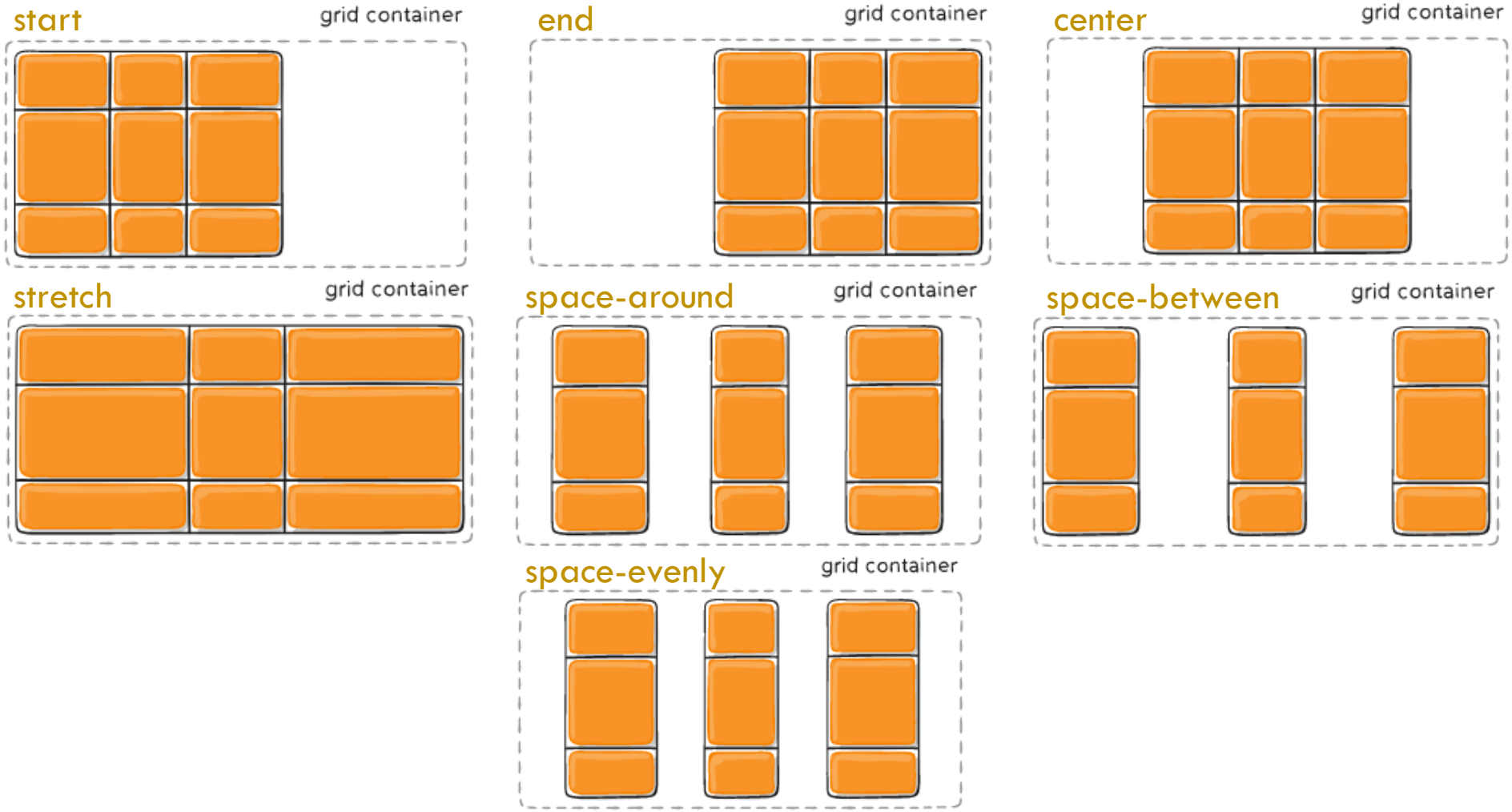
- Ak chceme zarovnávať konkrétnu položku inak, ako zvyšok kontajnera:
  - justify-self: start | end | center | stretch;
  - align-self: start | end | center | stretch;
  - zjednotene:
    - place-self: auto (tak, ako kontajner - default)
    - place-self: <align-self> <justify-self>;
      - ak sa uvedie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



Grid layout

- 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

- 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

Vlastnosť display

- inline
- block
- inline-block
- flex
- inline-flex
- grid
- inline-grid
- none
  - nie visibility: hidden
- contents
  - deti tohto elementu sa stanú deťmi jeho rodiča
- initial
- inherit
- table
- inline-table
- table-caption
- table-column-group
- table-header-group
- table-footer-group
- table-row-group
- table-row
- table-column
- table-cell
- list-item