Flexbox   
víme, že se dá udělat rozložení na stránce nastavením css vlastnosti position u jednotlivých prvků, ale s tím jak se v čase kódování webů vyvíjí, tak se rozšiřují s zjednodušují i možnosti, jak rozložení webů dělat.   
Takže není potřeba už používat tabulky, nebo floaty.   
V současné době se pro rozložení prvků na stránce nebo v sekcích používá flexbox nebo grid.   
Flexbox je ideální pro rozložení prvků v sekcích (navbar, galerie fotek etc.), ale dá se s ním i udělat layout celého webu.   
Pro složitější webový layout se doporučuje použití gridu. Ten na daWeb probírat nebudeme, ale určitě se s ním v budoucnu setkáte, případně si nasdílíme zdroje, abyste se ho mohly v průběhu doučit.

Flexbox se vždy týká obalového elementu – kontejneru a jeho potomků – položek. V AJ se jim pak říká flex-container a flex-items.

Flexbox znamená, že položky v daném kontejneru jsou flexibilní, pružné a tak i vyplňují volný prostor v kontejneru. Podle našeho zadání.

Je potřeba mít vždy na paměti, že jsou vlastnosti, které nastavujeme kontejneru – obalu a jiné, které nastavujeme jednotlivým flex-items.

Pokud máme obsah, který chceme nastylovat jako flexbox, tak vždy musíme zajistit, že je obalený v kontejneru.

Kontejneru potom nastavíme vlastnost **display:flex;**

Toto nám zajistí, že se nám děti tohoto kontejneru poskládají vedle sebe (respektive podél jedné osy) a stanou se z nich flex-items, to znamená, že určitým způsobem vyplňují místo v kontejneru. Zároveň jim můžeme nastavovat další vlastnosti a ovlivnit jak to místo vyplňují.

Flexbox je jednosměrný – to znamená, že má hlavní osu – buď je to řádek nebo sloupec / row nebo column / horizontální nebo vertikální

Další vlastnost na kontejneru je **flex-direction.** – tak právě nastavíme, jaká bude hlavní osa – jestli row, nebo column. Podle té se nám potom prvky také seřadí.   
**row, row-reverse, column, column-reverse**

Dále je na kontejneru možné nastavit, jak se prvky uvnitř něj zarovnají:

**Justify-content a align-items**

Justify content zarovnává prvky uvnitř podle hlavní osy:

**Flex-start** – defaultní – na začátek osy

**Flex-end** – na konec osy

(dále je tam start a end – writing mode a left a right)

**Cente**r – do centra

**Space between** – prvky jsou rozmístěny rovnoměrně na ose, první prvek úplně na začátku, poslední prvek úplně na konci

**Space around** – prvky jsou rozmístěny rovnoměrně na ose, a kolem nich je vždy stejně volného prostoru na každé straně (pro každý prvek zleva i zprava)

**Space evenly** - prvky jsou rozmístěny rovnoměrně na ose a mezi dvěma prvky je vždy stejné místo

https://css-tricks.com/almanac/properties/j/justify-content/

**Align-items** podle vedlejší.

Stretch (default)

Flex-start/start/self-start – začátek vedlejší osy

Flex-end/end/self-end konec

Center

Baseline

Poslední na kontejneru je flex-wrap:

Nowrap / defaultní

Wrap – normálně jak čteme tak se zalamuje

Wrap-reverse – zalamuje se naopak

Align content – pokud máme hodně obsahu na několika řádcích, tak jak se tyto zarovnávájí.

Dostáváme se k položkám – flex-items

Order – defaultně je to nula!! Takže pokus nastavíme nějakému 1, tak bude na konci

# Lekce 03

Opakování – úkol na container a order.

## ORDER

* Dále: prvky ve flexboxu mají takovou šířku, jaký je obsah.
* Někdy je ale kontejner větší a my ho chceme vyplnit prvky, ne jen prvky rozmístit s mezerami **CODE**
* Můžeme tedy pomocí vlastnosti určit, jak flex-items budou volné místo zabírat.

## FLEX GROW

Flex-grow nám určuje, kolik volného prostoru položka zabere.

jako hodnotu má číslo (bez jednotky), nebere negativní čísla   
defaultně 0 – prvky se na volný prostor neroztahují  
pokud dáme JEDNOMU PRVKU flex grow 1 – zabere všechno volné místo.

pokud jinému dáme flex-grow 2 – zabere dvakrát tolik volného místa – NEBUDE DVAKRAT VETSI **CODE**

.third {

  flex-grow: 1;

}

.second {

  flex-grow: 2;

}

.first {

  flex-grow: 3;

}

* Flex-shrink   
    
   definuje, zda se prvek bude smršťovat
* Opět jako hodnotu má číslo (bez jednotky), nebere negativní čísla
* defaultně 1 – smršťuje se, aby se vešel do divu
* pokud má 2, tak se bude smršťovat 2x rychleji, nebude poloviční
  + **second flex-shrink 2**   
    Comment
* pokud dáme 0 – obsah prvku se nebude smršťovat.
  + **First flex shrink 0**
  + Ukázka
  + Comment

## FLEX BASIS

* Pokud nastavíme šířku, bude se to chovat jinak u flex direction row a jinak u column. Stejně tak height.
* protože můžeme střídat, která osa je hlavní ve našem flexbox kontejneru, je lepší nastavovat vlastnost **flex-basis**.
* Defaultní nastavení je auto – to znamená tolik, kolik má prvek obsahu.
* Je možné nastavit specifickou šířku ALE! Je důležité myslet na flex-shrink and flex-grow, takže se to dá spíš chápat jako „v ideálním případě“
* často se používají procenta – 50%, nebo 33%, nebo calc

pro vyplnění divu - .flex-item {

  flex-basis: 20%;

}

* není to stejně jako u všech flex grow 1!!!

## Flex:

Kombinuje flex-grow flex-shrink a flex-basis. Je to šikovné, ale často je lepší to rozepsat i pro někoho dalšího, kdo bude kód číst.

.first {

  flex: 0 0 50%;

}

Align-self

Přijímá stejné hodnoty jako align-items, zarovnává polořku podle vedlejší osy, přebíjí align items na kontejneru

Nastavit výšku kontejneru a dát align items center – align self na prvek

.flex-container {

  display: flex;

  flex-direction: row;

  height: 500px;

  align-items: center;

}

.first {

  align-self: stretch;

}