ROZLOŽENÍ

Úvod do jednoduchého rozložení (tabulky, listy, ...) a pokročilého rozložení pomocí moderních technologií (flexbox, grid, ...).

GRID

Grid, společně s flexem, patří mezi moderní rozložení webů. Flex se spíše zaměřuje na řádkové rozložení a na vedlejších osách dělá minimum. Grid je spíše taková moderní tabulka. Jak si pamatujete, tak nevýhody tabulek jsou takové, že prvky uvnitř nemůžeme přesouvat tak, aby aby na telefonu byly jinde, než tabulka říká. Grid nám toto dovoluje a věci můžeme různě prohazovat.

Grid má tedy dvě dimenze - řádky a sloupce. Grid zapneme na prvku tak, že mu nastavíme vlastnost `display` na hodnotu `grid`. Grid dělíme, jak flex, na obal a jednotlivé prvky.

Gridu musíme minimálně definovat řádky a nebo sloupce. Řádky nastavujeme pomocí `grid-template-rows` a sloupce pomocí `grid-template-columns`. Hodnoty mohou být jakékoliv a jednotlivé řádky/sloupce rozdělujeme mezerou.

```
.container {
    display: grid;

    grid-template-columns: 50px 30px 40px; /* tři sloupce s velikosti 50px, 30px, 40px */
    grid-template-rows: 10px 60px 20px; /* tři řádky */
}
```

Jednotlivé předměty vkládáme dovnitř. Prvně se plní sloupce (tj. první řádek) a poté řádky. Pokud chceme, aby prvky zabrali celé místo, které jim gridem bylo přiděleno, musím nastavit výšku a šířku na 100%. HTML například vypadá:

```
<div class="container">
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
</div>
```

Když přidáme do gridu další věci, tak se již mimo definice budou přidávat na nové řádky. Definice se potom zvětší na `auto`, což znamená maximální hodnotu na dané dimenzi. Mezery mezi jednotlivými řádky a sloupci můžeme nastavovat pomocí vlasto `gap`, `row-gap` a `column-gap`.

Pro jednotky definice řádků a sloupců můžeme taktéž používat jednotku `fr`, která reprezentuje část místa v gridu. Když máme např. 1fr 2fr 1fr v definice, máme 4 místa a prvnímu a třetímu dáme 1/4 a druhý bude mít 2/4. Například tedy:

```
.container {
    display: grid;
    width: 300px;
```

```
    grid-template-columns: 4fr 1fr 2fr;
    grid-template-rows: 10px 60px 20px;

    gap: 30px;
}
```

Věci v gridu můžeme zarovnávat (když nepoužijeme 100% na výšku a šířku) pomocí vlastností `align-items` a `justify-items`. Obě mají zvyklé hodnoty.

Samotný grid můžeme pozicovat pomocí `justify-content` a `align-content`. Tyto hodnoty se používají tehdy, když je možné, že grid je menší než to, co mu rodič dal.

V definicích můžeme jednotlivé hodnoty obalovat do funkcí `repeat` a `minmax`. `repeat(n, value)` způsobí, že se hodnota `value` bude `n`-krát. Když v definici použijeme např. hodnotu `12px repeat(4, 1fr) 3fr`, reálně tam bude hodnota `12px 1fr 1fr 1fr 1fr 3fr`.

`minmax(min, max)` nám zapne buď hodnotu `min` a nebo `max`, dle toho, kolik místa mu dá kontejner. Hodnota nikdy však dané místo nepřesáhne. Často používáme např. `minmax(0, 1fr)`, který nám řekne, že každý sloupec musí mít stejné místo.

Každému prvku kontejneru můžeme ručně nastavit v jakém řádku a sloupci se má zobrazit. Dokonce můžeme říct, od jakého do jakého řádku či sloupce se zobrazí. Na to máme složené vlastnosti `grid-row` a `grid-column`. Například:

```
.item.longer {
    grid-column: 1 / 3; /* zobrazí se na sloupci 1 a 2 */
}
.item.higher {
    grid-row-start: 1;
    grid-row-end: 3; /* stejné jako grid-row: 1 / 3; */
}
```

Jedna z nezmíněných části gridu, kterou se nebudeme učit, jsou gridové plochy (grid area), které zjednodušují vkládání předmětů do gridu. Více o nich naleznete v dalších zdrojích.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.