ROOT.CZ

CSS Grid: revoluce ve web designu?

Pavel Satrapa před 15 hodinami

Když se zhruba před rokem začalo mluvit o novém modulu pro CSS, který umožňuje používat na stránce typografickou mřížku (grid), považoval jsem je nejdřív za módní vlnu. Designéři dostali novou hračku a nadšeně výskají.

Když se později její podpora objevila v řadě prohlížečů a zkusil jsem si na ni sáhnout, musel jsem uznat, že tahle hračka dokáže být vážně užitečná. Po dlouhých letech vnořování `<div>` a všeobecných značkovacích orgií se objevilo něco, co umožňuje dělat psí kusy a zároveň motivuje spíše ke zjednodušení kódu.

Vytknu před závorku, že pojem „grid“ se v typografii používá od 50. let 20. století a jeho českým ekvivalentem je „mřížka“. Nebudu ani trochu kůl a budu se tohoto pojmu držet. S typografickou mřížkou se setkáte typicky u knih typu „obrazová encyklopedie“, kde se míchá text s obrázky a struktura stránky bývá složitá.

Základní myšlenkou typografické mřížky (http://thinkingwithtype.com/grid/) je, že se plocha stránky rozdělí na určitá pole, řekněme pět sloupců a sedm řádků. Toto rozdělení je pravidelné, všechny sloupce jsou stejně široké, všechny řádky stejně vysoké, a konzistentní v celé publikaci. Když se pak na stránky umisťují obrázky a bloky textu, vždy zabírají obdélníkovou oblast tvořenou několika sousedními poli – třeba 2×3. Tyto plochy se mění podle potřeby, takže každá strana může vypadat odlišně, ale jelikož respektují hranice stejné mřížky, i ty odlišně uspořádané strany jsou konzistentní a celkově publikace drží pohromadě.

Webová mřížka funguje podobně, opět rozdělí stránku na pravoúhlou síť polí, která se následně naplňují obsahem. Jsou zde samozřejmě rozdíly, například často nebývá pravidelná. Trochu to připomíná dřevní časy používání tabulek pro uspořádání stránek, ovšem mřížka je elegantnější, silnější a nezamořuje HTML přidanými značkami.

Její specifikaci najdete v dokumentu CSS Grid Layout Module Level 1 (https://www.w3.org/TR/css-grid-1/), který je zatím ještě ve stavu kandidátském, nicméně je považován za prakticky dokončený a ve stávající podobě je i široce

implementován. Modul CSS Grid podporuje valná většina současných prohlížečů (https://caniuse.com/#feat=css-grid). Problém je s Microsoftem, který implementoval ranou pracovní verzi, jež se od konečné specifikace nemálo liší. Internet Explorer už v tomto stavu má zůstat, Edge začal podporovat aktuální definici mřížky od verze 16. Sečteno a podtrženo: není třeba se příliš bát, mřížku už lze nasadit na produkční weby.

Vytvoření mřížky

Nebudeme procházet celou specifikaci a všechny CSS vlastnosti, soustředíme se jen na základní principy fungování a praktické uplatnění. Výchozím bodem je, že prvku, v jehož rámci chcete vytvořit mřížku, přidělíte vlastnost `display: grid`. Máte-li odvahu, může to být přímo `<body>`. Častější asi bude nějaký obalující `<div>`, ale mřížku můžete v zásadě vytvořit v libovolném prvku, čili může pokývat jen část stránky.

Děti příslušného prvku (tedy přímí potomci bez jakýchkoli mezilehlých prvků) se stanou prvky mřížky a umisťují se do jednotlivých oblastí. Toto je důležité – do mřížky lze vkládat jen přímé potomky prvku s `display: grid`. Nemůžete vytáhnout prvek hluboko zanořený v HTML kódu a vložit jej do některé z oblastí.

Kromě samotného zavedení mřížky musíte ještě definovat, jak má vypadat. K tomu slouží vlastnosti `grid-template-columns` a `grid-template-rows`. Z nich vyplývá, kolik sloupců a řádků bude mřížka mít a jaké budou jejich rozměry. Hodnotou jsou mezerami oddělované rozměry. Lze je libovolně míchat, takže například definice

```
grid-template-columns: 200px auto 25%;
```

znamená, že mřížka bude mít 3 sloupce, první bude široký 200 pixelů, poslední 25 % celkové šířky a prostřední se přizpůsobí podle svého obsahu, zpravidla zabere zbylé volné místo. Zajímavou možností je jednotka `fr`. Je „gumová“ a roztáhne se tak, aby zabrala veškeré volné místo. Pokud je gumových rozměrů více, rozdělí si prostor v poměru podle čísel před fr. Definice

```
grid-template-columns: 200px 3fr 1fr;
```

zavádí opět tři sloupce. První bude široký 200 pixelů. Místo, které zbude po jeho odečtení, se rozdělí na čtvrtiny. Tři z nich připadnou druhému sloupci a jedna třetímu. Pozor, `1fr` není totéž co `auto`. `1fr` zabere všechno volné místo, zatímco `auto` se roztáhne podle potřeby. Nejvýrazněji se rozdíl projeví, pokud se jich objeví víc vedle sebe.

```
grid-template-columns: 1fr 1fr;
```

znamená dva stejně široké sloupce, zatímco

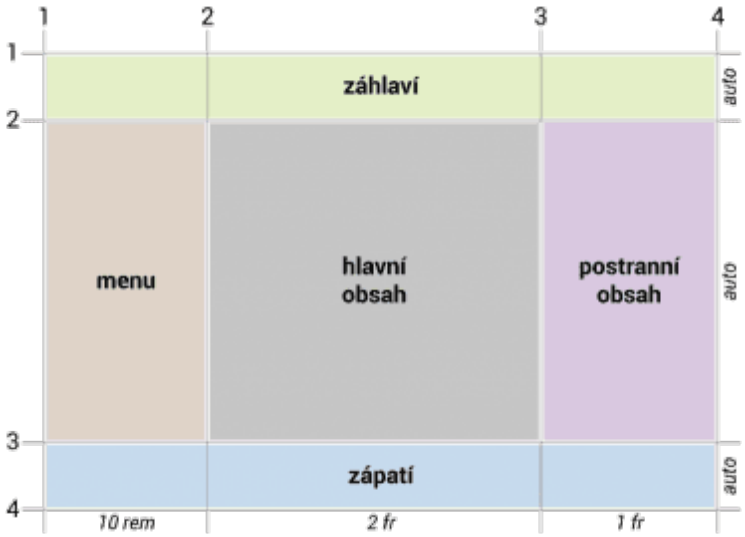
```
grid-template-columns: auto auto;
```

vytvoří dva sloupce, jejichž šířka se může lišit v závislosti na velikosti jejich obsahu.

Uzavřeme tuto část příkladem. Řekněme, že chceme vytvořit stránku se záhlavím a zápatím přes celou šířku, mezi nimi budou tři sloupce – po stranách dva užší s menu a nějakým postranním obsahem, uprostřed pak hlavní obsah. Vychází nám tedy mřížka 3×3. Budeme-li odvážní a vytvoříme ji pro celé tělo stránky, mohla by definice vypadat třeba takto:

```
body {
    display: grid;
    grid-template-columns: 10rem 2fr 1fr;
    grid-template-rows: auto auto auto;
}
```

Definici řádků jsme si ve skutečnosti mohli odpustit, protože `auto` je jejich výchozí výška a přidávají se do mřížky automaticky podle potřeby. Naše mřížka a zamýšlený obsah vypadají asi takto:



Mřížka s prvky stránky

Umisťování prvků

Mřížku máme, pojďme do ní vložit nějaký obsah. To lze provést několika způsoby. Začněme tím nejzákladnějším, kdy přesně stanovíte, odkud kam má sahat který prvek. V CSS mřížce k tomu slouží čísla (fiktivních) linií oddělujících jednotlivé sloupce nebo řádky. Na obrázku výše jsou znázorněny, přestože na stránce vykresleny nebudou. Čísluje se od jedničky, takže například první sloupec sahá od linie 1 po linii 2.

Má-li záhlaví (prvek `<header>`) sahat přes celou šířku stránky, znamená to, že se rozprostírá od sloupcové linie 1 zahajující první sloupec po linii 4 ukončující třetí

sloupec. Zabírá první řádek mřížky, z pohledu řádkových linií od 1 do 2:

```
header {
    grid-column: 1 / 4;
    grid-row: 1 / 2;
}
```

Zápatí je definováno podobně, jen se nachází v posledním řádku:

```
footer {
    grid-column: 1 / 4;
    grid-row: 3 / 4;
}
```

V příkladech předpokládám, že HTML kód stránky je velmi přímočarý a klíčové prvky jsou dětmi prvku `<body>`, tedy že tělo stránky vypadá nějak takto:

```
<body>
    <header>...</header>
    <nav class="mainmenu">...</nav>
    <main>...</main>
    <aside class="sidebar">...</aside>
    <footer>...</footer>
</body>
```

Při umisťování se můžete vyřádit. Lze například jednotlivým liniím při definici řádků nebo sloupců přiřadit jména a při umisťování pracovat se jmény. Za lomítkem také může být hodnota `span` *N*, která znamená, že daný prvek se má roztáhnout přes *N* sloupců nebo řádků. Stejné umístění hlavičky bychom mohli zapsat následovně:

```
header {
    grid-column: 1 / span 3;
    grid-row: 1 / span 1;
}
```

Záporná čísla se počítají od konce. Takže záhlaví přes celou šířku mřížky, bez ohledu na počet jejích sloupců, lze snadno definovat pomocí

```
header {
    grid-column: 1 / -1;
    grid-row: 1 / 2;
}
```

Dávejte si pozor na to, že explicitní umístění může rozšiřovat stávající mřížku. Pokud umístíte obsah do sloupců nebo řádků, které neexistují, budou automaticky vytvořeny. Například pokud při třísloupcové mřížce z příkladu výše umístíte záhlaví

```
header {
    grid-column: 1 / 6;
    grid-row: 1 / 2;
```

```
}
```

přidají se čtvrtý a pátý sloupec se šířkou `auto`. Pokud s tím zbytek návrhu nepočítá, může to obsah stránky dost rozházet. Věnujte proto velkou péči konzistenci čísel řádků a sloupců. Pro automaticky přidávané řádky a sloupce také nefungují záporná čísla, takže pozor na ně.

Důležitým konceptem je automatické umisťování. Přiřadíte-li prvku vlastnost `display: grid`, znamená to, že *všechny* jeho děti budou vloženy do jeho mřížky. Pokud jejich umístění neurčíte sami (například pomocí `grid-column` a `grid-row`), umístí je prohlížeč automaticky do prvního volného pole.

Díky tomu naše definice výše v zásadě stačí na vytvoření stránky podle původních představ. Záhlavím jsme obsadili celý první řádek, zápatím celý třetí a druhý zůstal volný. Když prohlížeč v HTML kódu narazí na hlavní menu (prvek `<nav>`), jehož umístění není stanoveno, umístí jej do prvního volného pole, což je první pole druhého řádku, tedy levý sloupec v hlavním prostoru stránky. Následuje `<main>` automaticky umístěný do prostředního a `<aside>` do pravého pole.

Používat automat pro základní rozvržení stránky není dvakrát šťastné. Pokud někde uděláte drobnou chybu a do stránky se vám vloudí nějaký prvek navíc, snadno se může stát, že jeho automatické vložení úplně rozhází umístění jeho následníků. Pro základní součásti stránky proto doporučujeme vždy explicitně definovat jejich umístění.

Naopak pro stránku typu titulní stránka novin se automat velmi hodí, protože umožňuje do ní jednoduše vkládat texty a obrázky, které automat rozloží po mřížce. Oddělování sekcí pak jednoduše zařídíte tak, že nadpisy (řekněme `<h2>`) roztáhnete přes všechny sloupce mřížky a tím zajistíte, že nadpis sekce zabere vždy celý řádek, bude umístěn pod všechny prvky z předchozí sekce a následující prvky se v mřížce ocitnou až pod ním.

Pomocí `span` můžete narušit pravidelnost. Definice v podobě `auto / span 2` znamená, že první sloupec prvku se má určit automaticky, ale prvek se má roztáhnout přes dva sloupce. Pokud byste sólokapry chtěli umisťovat vždy zcela doleva a věnovat jim prostor 2×2 pole mřížky, mohla by jejich definice obsahovat

```
.solokapr {
    grid-column: 1 / span 2;
    grid-row: auto / span 2;
}
```

Podívejte se na příklad (https://i.iinfo.cz/files/root/51/priklad-nepravidelny-1.html) mřížky s automaticky umisťovanými prvky různých velikostí.

Pojmenované oblasti

Další možností pro rozmístění prvků v mřížce je pojmenovat její jednotlivé oblasti a vkládaným prvkům určit jména oblastí, do nichž mají být umístěny. Tahle metoda je má oblíbená, protože obrovsky usnadňuje responzivní design. V závislosti na šířce okna měníte jen definice sloupců (a případně řádků) a vymezení oblastí. Vše ostatní zůstává beze změny.

Jak to funguje? Na úrovni prvku definujícího mřížku pojmenujete oblasti. Slouží k tomu vlastnost `grid-template-areas`. Hodnotou je mezerami oddělovaný seznam řetězců, z nichž každý odpovídá jednomu řádku (proto je lepší psát je pod sebe). V něm jsou pak mezerami oddělovaná jména oblastí v jednotlivých polích. Stejné jméno se může vyskytovat opakovaně – znamená to, že oblast sahá přes všechna taková pole. Ta musí tvořit obdélník, nelze mít oblast nesouvislou nebo zahnutou.

V případě naší ukázkové stránky by definice vypadala nějak takto:

```
body {
    display: grid;
    grid-template-columns: 10rem 2fr 1fr;
    grid-template-areas:
        "zahlavi zahlavi zahlavi"
        "menu    hlavni  postranni"
        "zapati  zapati  zapati";
}
```

Zavádí pět pojmenovaných oblastí, z nichž oblast `zahlavi` zabírá všechna tři pole prvního řádku, `zapati` celý poslední řádek a zbývající tři oblasti obsazují vždy jen jedno pole. Jména jsou celkem libovolná, samozřejmě je záhodno, aby byla mnemotechnická.

Prvky pak vložíte do pojmenované oblasti pomocí vlastnosti `grid-area`, jejíž hodnotou je příslušné jméno. Výše zmíněné deklarace pro umístění jednotlivých prvků bychom nahradili následujícími:

```
header     { grid-area: zahlavi; }
.mainmenu  { grid-area: menu; }
main       { grid-area: hlavni; }
.sidebar   { grid-area: postranni; }
footer     { grid-area: zapati; }
```

Veškeré reakce na šířku okna (či případné jiné faktory) se omezí na změny v definici mřížky a pojmenovaných oblastí. Řekněme, že pro úzké displeje bychom chtěli stránku uspořádat do jednoho sloupce, kde všechny oblasti budou pod sebou. Navíc odsuneme menu až pod obsah. Teprve když šířka stránky překročí 45 rem, přejdeme na třísloupcové uspořádání. Definice, které takové chování zajistí, by vypadaly nějak takto:

```
body {
```

```
    display: grid;
    grid-template-columns: 1fr;
    grid-template-areas:
        "zahlavi"
        "hlavni"
        "postranni"
        "menu"
        "zapati";
}

header     { grid-area: zahlavi; }
.mainmenu  { grid-area: menu; }
main       { grid-area: hlavni; }
.sidebar   { grid-area: postranni; }
footer     { grid-area: zapati; }

@media ( min-width: 45rem ) {
    body {
        grid-template-columns: 10rem 2fr 1fr;
        grid-template-areas:
            "zahlavi zahlavi zahlavi"
            "menu    hlavni  postranni"
            "zapati  zapati  zapati";
    }
}
```

Příklad takové stránky (https://i.iinfo.cz/files/root/413/priklad-stranka-1.html) předvede, jak se bude chovat ve vašem prohlížeči.

Příjemné je, že dramatické změny rozložení stránky jsou realizovány úpravou pouhých dvou vlastností (případně tří, pokud budete pracovat i s definicí řádků) a vše se nachází pohromadě. Takže je s tím málo práce a riziko, že na něco zapomenete, vytvoříte nekonzistentní definici a stránka se vám rozpadne, je celkem nízké.

Násup

Jak už bylo řečeno, mřížka může sahat přes celou stránku, nebo může pokrývat jen její část. A samozřejmě můžete mít jednu mřížku uvnitř druhé. Například zmiňovaná titulní stránka novin by pravděpodobně měla jednu celostránkovou mřížku, kde by se definovaly základní prvky typu záhlaví, hlavní menu a podobně. Její hlavní obsahová část by pak definovala svou vlastní mřížku, kam by se v dlaždicovém uspořádání automaticky umisťovaly upoutávky na jednotlivé články. Definice prvku `<main>` by obsahovala jak vlastnosti prvku umístěného do mřížky, tak definici nové mřížky pro jeho obsah:

```
main {
    grid-area: hlavni;

    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-column-gap: 0.5rem;
    grid-row-gap: 0.5rem;
}
```

Mřížka se také dobře doplňuje s dalšími technologiemi, jako je například flexbox. Jím by se dalo řešit třeba menu vložené do jedné z oblastí. Obecné pravidlo pro návrh rafinovanějších metod rozmístění prvků: pro dvourozměrné uspořádání bývá vhodnější mřížka, pro jednorozměrné flexbox. Protipříklady se jistě najdou.

Opakování stejných hodnot v definici sloupců (řádků) umí být otravné. Proto je k dispozici zápis `repeat( počet , šířka )`, který vloží příslušný *počet* daných *šířek*. Rozdělení na 4 stejně široké sloupce z příkladu výše by se dalo zapsat

```
grid-template-columns: repeat(4, 1fr);
```

Lze je kombinovat s ostatními. Pokud byste chtěli sloupec široký 10 rem, za nímž následují tři stejně široké sloupce a na konci jeden dvojnásobně široký, použijte

```
grid-template-columns: 10rem repeat(3,1fr) 2fr;
```

Návrh složitěji formátovaného obsahu obvykle vychází z jedné ze dvou základních myšlenek – buď máte předem jasno o počtu sloupců, nebo máte představu o šířce sloupců a chtěli byste, aby se jejich počet přizpůsobil šířce dostupného prostoru.

První případ je jasný, příslušné konstrukce už byly popsány. Pro ten druhý použijte `repeat()`, kde jako počet opakování zadejte `auto-fit` nebo `auto-fill`. Chovají se podobně, vloží co nejvíce sloupců daných rozměrů. Liší se v tom, že pokud ve sloupcích není vložen žádný materiál, `auto-fill` je zachová, zatímco `auto-fit` smaže (formálně zůstanou zachovány, ale mají nulovou šířku a sloučí se i jejich okraje). Čili co nejvíce sloupců šířky 15 rem by zajistila deklarace

```
grid-template-columns: repeat(auto-fit, 15rem);
```

Při pevně zadané šířce slupce ovšem na konci zpravidla vznikne mezera, protože šířka daného prostoru nebývá přesným násobkem šířky sloupce. Proto lze šířku sloupce definovat pomocí `minmax( nejmenší , největší )` a ponechat prohlížeči určitou volnost, aby roztažením sloupců zabral dostupný prostor. Oblíbenou hodní hranicí je 1 fr, která umožní sloupce roztáhnout podle potřeby (ale pořád se snaží jich vložit co nejvíce, takže maximální šířka nepřesáhne dvojnásobek minimální). Předchozí definici bychom mohli vylepšit na

```
grid-template-columns: repeat(auto-fit, minmax(15rem,1fr));
```

Opět se můžete podívat na příklad (https://i.iinfo.cz/files/root/253/priklad-autofit-1.html). Zkuste změnit šířku okna prohlížeče a pozorujte, jak se rozměry i počet sloupců mění, aniž by byly použity obvyklé podmínky `@media`.

Mřížka v praxi aneb podpora

Vraťme se ještě k patálii s Microsoftem a se zpětnou kompatibilitou obecně. Abyste se pojistili, že vlastnosti související s mřížkou nezpůsobí zmatení klientům, kteří ji neznají, je záhodno před jejich použitím ověřit, zda jsou podporovány. Jako první kandidát se nabízí

```
@supports ( display: grid ) { ... }
```

Jenže můžete narazit u MSIE (a starších verzí MS Edge), které si myslí, že tuto vlastnost podporují. Ve skutečnosti znají pracovní verzi z roku 2011 (https://www.w3.org/TR/2011/WD-css3-grid-layout-20110407/). Proto je záhodno podporu testovat na některé z vlastností, které obsahuje až aktuální specifikace. Nabízí se třeba `grid-template-columns`, `grid-template-rows` či `grid-template-areas`. Tyto vlastnosti změnily jména a prohlížeče Microsoftu se k nim nehlásí. O něco kratší je `grid-auto-flow` ovlivňující automatické umisťování. Čili vhodný test může vypadat

```
@supports ( grid-auto-flow: row ) {
    definice mřížky a jejích prvků
}
```

Tady bych ochutnávku mřížkovaného CSS ukončil. Text zdaleka není vyčerpávající, jedná se jen o stručné představení možností. Dobrým zdrojem pro podrobnější studium je A Complete Guide to Grid (https://css-tricks.com/snippets/css/complete-guide-grid/), kde najdete přehled všech CSS vlastností pro rodiče definujícího mřížku i děti do ní vkládané.

Celkově se CSS Grid jeví jako velmi nadějný mechanismus. Kombinuje velkou vyjadřovací sílu s jednoduchým kódem. Po dlouhé době se objevuje v CSS koncept, který inklinuje spíše ke zjednodušování HTML kódu namísto obalování významových prvků dalšími a dalšími `<div>` k dosahování různých efektů. Rozhodně stojí za pozornost.
