KAPITOLA 4

Nové přístupy k rozvrhování jazykem CSS

V této kapitole:
- Více sloupců
- Flexbox
- Grid Layout
- Vzdálenější budoucnost

Jakmile napíšete kód jazyka HTML a zvážíte, jak budete optimalizovat své webové stránky nebo aplikaci pro různá zařízení, můžete začít vytvářet rozvržení. Ve specifikaci CSS 2.1 byly možnosti rozvrhování značně omezené. Třebaže chytří návrháři a vývojáři vyždímali z vlastnosti `float` a pozicování maximum, většina webových stránek obsahuje nějakou variantu třísloupcové mřížky. Tento vzor vznikl zejména kvůli omezením dostupných technologií a ne kvůli tomu, že by to vyžadoval obsah.

Jazyk CSS3 nabízí nové vlastnosti pro tvorbu flexibilních rozvržení, přičemž využívá znalostí z mnoha století psaných a tištěných materiálů, ale také zohledňuje schopnosti obrazovek. S jejich pomocí můžou návrháři lépe zobrazovat obsah a uživatelská rozhraní aplikací můžou lépe reagovat na cílová zařízení.

Upozornění: Třebaže prohlížeče již implementují spoustu nových funkcí, měli byste je považovat za experimentální a uvědomit si, že se můžou měnit, dokud konsorcium W3C nedokončí standardizační proces. Podrobné informace o podpoře v aktuálních informacích najdete v příloze A, jejíž aktualizovaná verze bude k dispozici na adrese *http://modernwebbook.com/*.

Více sloupců

Jelikož webové stránky mají své kořeny ve vědeckých dokumentech, používají velmi jednoduchý vzor rozvržení textu – vše se nachází v jediném nepřerušovaném sloupci, podobně jako text v textovém procesoru nebo editoru. Je tomu tak zejména proto, že web je dynamický, velikosti písma se značně liší a do hry vstupuje řada dalších odlišností, takže je velmi obtížné umístit obsah přesně. Při tisku pracujete s pevnými velikostmi písmen a známým počtem znaků, a proto můžete rozvrhovat text na stránce flexibilněji.

Vezměte si libovolný tištěný časopis nebo noviny a určitě narazíte na text rozdělený do několika sloupců – obvykle do dvou, někdy do tří nebo více. Sloupcové rozvržení umožňuje umístit na stránku více textu, aniž byste museli obětovat čitelnost.

Až donedávna nebylo možné napodobit tento styl na webu bez použití jazyka JavaScript, ale modul pro rozvržení do více sloupců (Multi-column Layout Module) řeší tyto nedostatky. Buďme upřímní – tento styl se pravděpodobně hodí více pro tisk než pro obrazovku, na níž může být nepříjemné posouvat obsah nahoru a dolů. V některých situacích může být ale použití více sloupců přínosné.

Vlastnosti pro více sloupců usnadňují rozdělování vloženého obsahu do sloupců. Na daný element jednoduše aplikujeme vlastnost `column-count`, které předáme celé číslo označující počet sloupců. Například pomocí níže uvedeného kódu bychom rozdělili obsah do tří sloupců:

```
E { column-count: 3; }
```

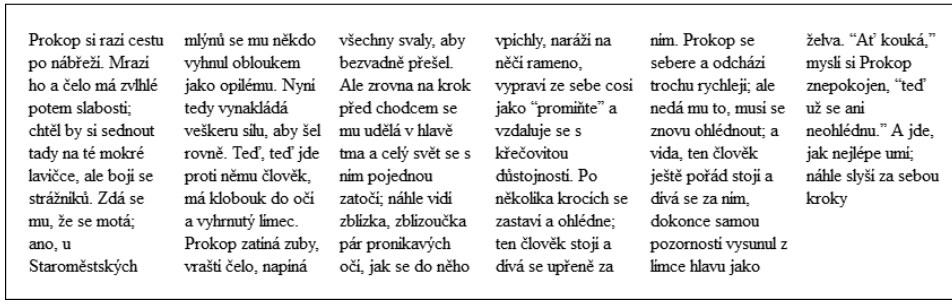
Obsah je nyní rozdělený do tří sloupců, mezi nimiž je malá mezera. Kdybyste chtěli definovat sloupce přesněji, mohli byste nastavit jejich šířku prostřednictvím vlastnosti `column-width`:

```
E { column-width: 120px; }
```

Předchozím kódem vytváříme co nejvíce sloupců o šířce 120 pixelů s mezerami mezi nimi, a to tak, aby se vešly do svého rodičovského elementu. Předpokládejme kupříkladu, že rodičovský element má šířku 600 pixelů – prohlížeč vytvoří 4 sloupce o šířce 120 pixelů, mezi nimiž bude mezera o velikosti přibližně `3em`, a navíc zbyde nějaké volné místo. Prohlížeč toto volné místo rovnoměrně rozdělí mezi jednotlivé sloupce tak, aby vyplnil celou šířku rodičovského elementu. To znamená, že hodnota vlastnosti `column-width` není absolutní, ale vyjadřuje spíše minimální hodnotu.

Obrázek 4.1 srovnává tyto dva přístupy. V horním příkladu vytváříme tři sloupce pomocí vlastnosti `column-count`, zatímco ve spodním příkladu nastavujeme vlastnosti `column-width` hodnotu `120 px`, ale skutečná šířka sloupců se mírně liší, aby lépe vyplnily rodičovský element.

Prokop si razí cestu po nábřeží. Mrazí ho a čelo má zvlhlé potem slabosti; chtěl by si sednout tady na té mokré lavičce, ale boji se strážníků. Zdá se mu, že se motá; ano, u Staroměstských mlýnů se mu někdo vyhnul obloukem jako opilému. Nyní tedy vynakládá veškeru sílu, aby šel rovně. Teď, teď jde proti němu člověk, má klobouk do očí a vyhrnutý límec. Prokop zatíná zuby, vrašťí čelo, napíná všechny svaly, aby bezvadně přešel. Ale zrovna na krok před chodcem se mu udělá v hlavě tma a celý svět se s ním pojednou zatočí; náhle vidí zblízka, zblízoučka pár pronikavých očí, jak se do něho vpíchly, naráží na něčí rameno, vypraví ze sebe cosi jako "promiňte" a vzdaluje se s křečovitou důstojností. Po několika krocích se zastaví a ohlédne; ten člověk stojí a dívá se upřeně za ním. Prokop se sebere a odchází trochu rychleji; ale nedá mu to, musí se znovu ohlédnout; a vida, ten člověk ještě pořád stojí a dívá se za ním, dokonce samou pozornosti vysunul z límce hlavu jako želva. "Ať kouká," mysli si Prokop znepokojen, "teď už se ani neohlédnu." A jde, jak nejlépe umí; náhle slyší za sebou kroky.



Obrázek 4.1 Porovnání sloupců rozvržených pomocí vlastnosti column-count (horní) a column-width (spodní)

Vlastnosti `column-width` a `column-count` je možné aplikovat na stejný element. Pokud to uděláte, z vlastnosti `column-count` se stane maximální možný počet sloupců. V takovém případě prohlížeči sdělujete: „Udělej sloupce o šířce 120 pixelů až do maximálního počtu tří sloupců a přitom změň jejich velikost tak, aby vyplnily rodičovský element.“ Případně byste mohli použít zkrácenou vlastnost `columns`, které předáváte dvě hodnoty reprezentující vlastnosti `column-width` a `column-count`:

```
E { columns: 120px 3; }
```

Jestliže rozdělíte obsah do sloupců bez pevné výšky, prohlížeč automaticky distribuuje řádky v jednotlivých sloupcích takovým způsobem, aby potenciálně zůstalo prázdné místo vespod rodičovského elementu. V případě, že element bude mít definovanou výšku, můžete si vybrat styl toku textu pomocí vlastnosti `column-fill`.

Její výchozí hodnotou je `balance`, s níž prohlížeč distribuuje řádky rovnoměrně. Alternativní chování spočívá ve vyplňování sloupců sekvenčně do jejich maximální výšky. Poslední sloupec bude mít pravděpodobně méně řádků a spoustu nevyužitého místa. Pokud vám to vyhovuje, můžete použít klíčové slovo `auto`.

Mezery a linky

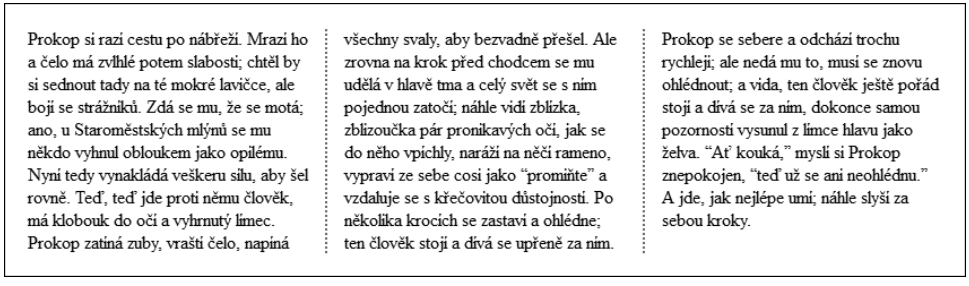
Už víte, že mezi jednotlivými sloupci jsou mezery. Velikost těchto mezer můžete měnit prostřednictvím vlastnosti `column-gap`. Ta přijímá délkovou hodnotu a upravuje mezeru mezi sloupci na požadovanou délku. Zde je příklad:

```
E { column-gap: 2em; }
```

Mezi sloupce můžete také vkládat **linky** pomocí vlastnosti `column-rule`. Výsledný efekt vypadá podobně jako u vlastnosti `border` z jazyka CSS 2.1. Vyžaduje rovněž stejné tři hodnoty – pro šířku, styl a barvu. Toto nastavení ale platí jen pro svislou linku, ne pro všechny čtyři strany. Šedou linku o šířce 2 pixely vložíte mezi sloupce následujícím způsobem:

```
E { column-rule: 2px dotted gray; }
```

Jakmile spojíte tuto vlastnost s vlastností `column-gap`, prohlížeč rozdělí mezeru na obou stranách linky rovnoměrně. V tomto případě byste získali mezeru o velikosti `2em`, takže na každé straně linky o šířce 2 pixely by se nacházela mezera široká `1em`, jak lze vidět na obrázku 4.2.


Prokop si razí cestu po nábřeží. Mrazí ho a čelo má zvlhlé potem slabosti; chtěl by si sednout tady na té mokré lavičce, ale boji se strážníků. Zdá se mu, že se motá; ano, u Staroměstských mlýnů se mu někdo vyhnul obloukem jako opilému. Nyní tedy vynakládá veškeru sílu, aby šel rovně. Teď, teď jde proti němu člověk, má klobouk do očí a vyhrnutý límec. Prokop zatíná zuby, vrašťí čelo, napíná všechny svaly, aby bezvadně přešel. Ale zrovna na krok před chodcem se mu udělá v hlavě tma a celý svět se s ním pojednou zatočí; náhle vidí zblízka, zblízoučka pár pronikavých očí, jak se do něho vpichly, naráží na něčí rameno, vypraví ze sebe cosi jako "promiňte" a vzdaluje se s křečovitou důstojností. Po několika krocích se zastaví a ohlédne; ten člověk stojí a dívá se upřeně za ním. Prokop se sebere a odchází trochu rychleji; ale nedá mu to, musí se znovu ohlédnout; a vida, ten člověk ještě pořád stojí a dívá se za ním, dokonce samou pozorností vysunul z límce hlavu jako želva. "Ať kouká," myslí si Prokop znepokojen, "teď už se ani neohlédnu." A jde, jak nejlépe umí; náhle slyší za sebou kroky.

Obrázek 4.2 Nastavení mezer a linek mezi sloupci

Obalování a zalamování

Sloupce jsou užitečné, pokud pracujete s textem nebo jiným řádkovým obsahem, ale určitě budete chtít také používat objekty a blokové elementy, které lze jen obtížně umísťovat do sloupců. V těchto případech vám pomůže dvojice souvisejících vlastností.

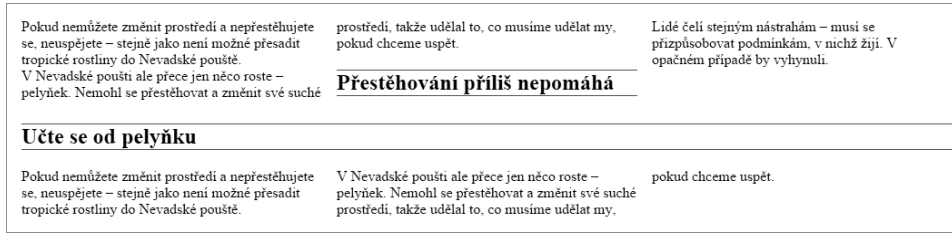
První z nich je `column-span`, jenž se skvěle hodí, pokud chcete přerušit tok sloupců novým elementem, který je bude obalovat všechny. Přípustnou hodnotou je klíčové slovo `all` nebo `none` – to znamená, že nový element obaluje buď všechny sloupce, nebo žádný.

```
F { column-span: all; }
```

Jakmile použijete hodnotu `all`, tok textu se přeruší před elementem, na nějž ji aplikujete, a pokračuje za ním, jak lze vidět na obrázku 4.3.

Jestliže máte elementy uspořádané do mřížky, můžete narazit na situaci, že konec sloupce způsobí zalomení uprostřed daného elementu.

To není příliš ideální, pokud máte například podnadpis, jenž by se mohl rozdělit do dvou sloupců. Této situaci lze zabránit prostřednictvím skupiny vlastností pro určování místa zalamování. Všechny tyto vlastnosti – `break-after`, `break-before` a `break-inside` – fungují na podobném principu.



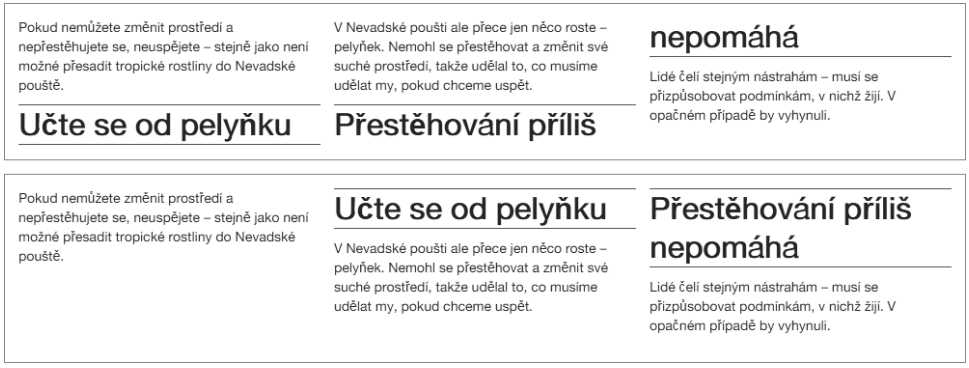
Obrázek 4.3 Druhý podnadpis obaluje všechny sloupce a přerušuje tok obsahu

Pro následující příklad zvolíme vlastnost `break-before`. Jejími hodnotami sdělujeme prohlížeči, zda chceme zalamovat sloupec před elementem. Hodnota `column` zalamuje obsah před elementem (je-li to nutné). Hodnota `avoid-column` (nebo jen `avoid`) zakazuje prohlížeči zalomit obsah, pokud to není nevyhnutelné. Výchozí hodnota `auto` nechává na prohlížeči, aby určil nejlepší způsob rozvržení sloupců.

Pokud vám není jasné, jak tato vlastnost funguje, prohlédněte si ukázkový soubor `zalamovani-sloupcu.html`. Zde je krátký příklad:

```
h2 { break-before: column; }
```

Výsledky můžete vidět na obrázku 4.4. Horní část používá hodnotu `auto`, a proto jsou podnadpisy vložené uprostřed sloupců. V dolní části s hodnotou `column` je vždy zalomení před elementy `h2`.



Obrázek 4.4 Vliv hodnoty column u vlastnosti break-before lze vidět na spodním výstupu

Vlastnosti `break-after` a `break-middle` fungují podobně, ale zalamují obsah na jiných místech (názvy těchto vlastností mluví samy za sebe).

Flexbox

Pokud vytváříte spíše webovou aplikaci než obsahově bohaté webové stránky, a to zejména tehdy, když má spoustu tlačítek a jiných prvků uživatelského rozhraní nebo spoustu formulářů a interaktivních oblastí, určitě oceníte nový modul Flexbox (Flexible Box).

Jádrem modulu Flexbox je XUL – jazyk používaný k tvorbě rozvržení prohlížeče Firefox. To znamená, že se zaměřuje na uživatelská rozhraní. Flexbox umožňuje flexibilní změnu velikosti elementů, aby se lépe vešly do vyhrazeného prostoru. Navíc je možné elementy přeuspořádávat a otáčet.

Syntaxe Flexboxu prodělala spoustu změn v uplynulých letech, ale konečně je stabilní a má rozsáhlou podporu, proto se o ní můžeme bez obav pobavit.

Použití modelu Flexbox

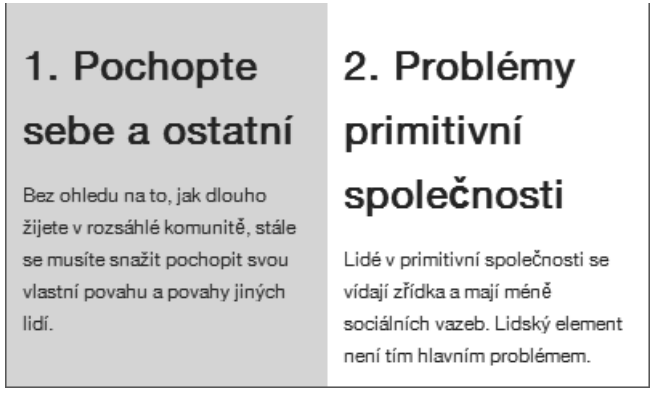
Abyste mohli použít model Flexbox, vytvořte si **flex-obal**. Jedná se rodičovský element, který bude obsahovat všechny **flex-položky**. Flex-obal definujete pomocí nové hodnoty pro vlastnost `display`:

```
E { display: flex; }
```

Teď už máte flex-obal, ale ptáte se, k čemu je dobrý? To zjistíte, až do něho přidáte dceřiné elementy. V ukázkovém souboru `flexbox.html` najdete dvě dceřiné položky uvnitř obalujícího elementu.

```
<div id="obal">
  <div id="potomek1">...</div>
  <div id="potomek2">...</div>
</div>
```

Na obrázku 4.5 si všimněte, že oba dceřiné elementy jsou umístěné vedle sebe a mají stejnou šířku, přestože jste nepoužili vlastnost `float` ani pozicování. To je výchozí chování flex-položek (dceřiných elementů flex-obalu) – jsou umístěné do řádku uvnitř obalujícího elementu. Navíc jsou uspořádané ve směru zápisu jazyka pro daný dokument, takže pro jazyky čtené zleva doprava (čeština, angličtina atd.) směřuje řádek také zleva doprava.



Obrázek 4.5 Dceřiné položky flex-obalu jsou automaticky rozmístěny vodorovně

Tento směr můžete změnit prostřednictvím vlastnosti `flex-direction` na obalujícím elementu. Její výchozí hodnotou je `row` a ta zaručuje výše popsané standardní chování, zatímco hodnotou `column` byste umístili položky svisle – tj. do sloupce. K dispozici je rovněž řada dalších hodnot, jak ostatně uvidíte za okamžik.

```
E {
  display: flex;
  flex-direction: column;
}
```

S tímto kódem budou vaše flex-položky vypadat podobně jako při běžném rozvržení, ale budou pod sebou.

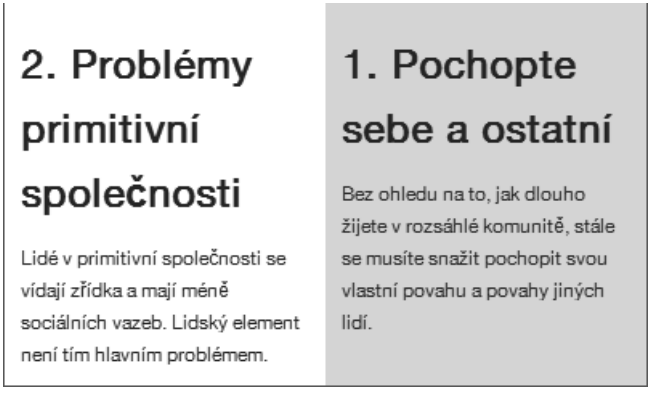
Změna uspořádání obsahu

Další skvělou funkcí modelu Flexbox je, že umí rychle měnit uspořádání zobrazených položek, a to bez ohledu na jejich fyzické umístění v kódu dokumentu. Například v předchozím příkladu jsme měli dvě položky v řádku, ale co kdybychom chtěli, aby element s identifikátorem `potomek2` předcházel elementu s identifikátorem `potomek1`?

Můžete to udělat prostřednictvím vlastnosti `flex-direction` a hodnoty `row-reverse`. Ta obrací uspořádání flex-položek, jak můžete vidět na obrázku 4.6.

```
E { flex-direction: row-reverse; }
```

Hodnota `column-reverse` dělá to samé, ale pro flex-položky zobrazené ve sloupcích – obrací jejich pořadí, ale ve svislém směru. Zkuste experimentovat s hodnotami vlastnosti `flex-direction` ve výše uvedeném ukázkovém souboru, abyste si vyzkoušeli, jak funguje.



Obrázek 4.6 Hodnota row-reverse převrací uspořádání flex-položek

Pomocí vlastnosti `flex-order` můžete jít za hranice pouhého obráceného uspořádání – můžete vytvořit zcela nové vzory uspořádání. Tuto vlastnost lze aplikovat na flex-položky, ale ne na flex-obal. Její hodnotou je číslo, jež reprezentuje **řadovou skupinu**. To znamená, že položky se stejnou hodnotou jsou seskupené dohromady. Kromě toho jsou uspořádané podle své řadové skupiny – všechny položky s nejmenší řadovou skupinou budou první, následují položky s druhou nejmenší řadovou skupinou atd. Všechny položky bez explicitně uvedené hodnoty budou zobrazené jako první, protože výchozí hodnotou je `0`.

Položky se stejným číslem řadové skupiny jsou dodatečně řazené na základě jejich uspořádání v kódu dokumentu. Uspořádání podle řadových skupin se může zdát na první pohled zmatené, proto následuje příklad se čtyřmi flex-položkami:

```
<div id="obal">
  <div id="potomek1">...</div>
  <div id="potomek2">...</div>
  <div id="potomek3">...</div>
  <div id="potomek4">...</div>
</div>
```

Bez jakéhokoliv nastavení se dceřiné elementy zobrazí přesně tak, jak jsme je zapsali do dokumentu – element `#potomek1`, `#potomek2`, `#potomek3` a `#potomek4`. Nyní je přeuspořádáme pomocí různých hodnot vlastnosti `flex-order`:

```
#potomek1 { flex-order: 2; }
#potomek2, #potomek4 { flex-order: 3; }
#potomek3 { flex-order: 1; }
```

S těmito pravidly stylů budou položky umístěné v následujícím pořadí – `#potomek3`, `#potomek1`, `#potomek2` a `#potomek4`. Položka `#potomek3` bude první, jelikož má nejnižší řadovou skupinu, za ní bude následovat položka `#potomek1` s další vyšší skupinou a nakonec položky `#potomek2` a `#potomek4`, které mají řadovou skupinu 3. Položka `#potomek4` je poslední, protože je také poslední v dokumentu.