KAPITOLA 2

Techniky rozvržení

V této kapitole:

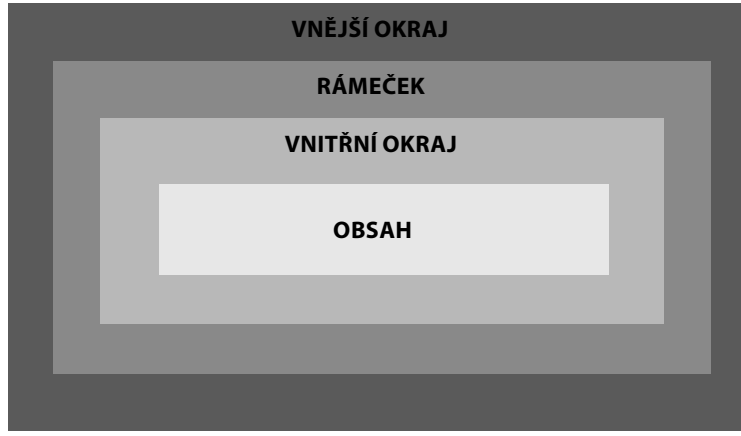
- Box model
- Blokové versus řádkové elementy
- Zkrácené versus běžné vlastnosti jazyka CSS
- Rozvržení založené na obtékání
- Pozicování v jazyce CSS
- Responzivní web design
- Intuitivní rozměry s vlastností box-sizing
- Přidáváme další styly za účelem rozvržení
- Obtékané obrázky posledních receptů
- Styly pro rozvržení záhlaví
- Rozvržení propagační fotografie
- Rozvrhujeme zápatí
- Rozvrhujeme sekci „Nejoblíbenější“
- Kam se bude ubírat budoucnost rozvržení založených na stylech

V této kapitole si představíme několik technik rozvržení. Praktickou zkušenost získáme, až je budeme implementovat do kostry, kterou jsme si vytvořili v kapitole 1, „Úvod do jazyka CSS“.

Než s tím začneme, musíme si popsat ještě některé základní koncepce, které jsme si v předchozí kapitole neukázali.

Box model

Termín box model většinou označuje neviditelnou obdélníkovou oblast, kterou prohlížeč vytváří okolo každého elementu jazyka HTML. Tuto oblast tvoří čtyři základní komponenty, jejichž význam snadněji pochopíte pomocí diagramu z obrázku 2.1.



Obrázek 2.1. Grafická reprezentace box modelu

Velikost komponent box modelu je na tomto diagramu poněkud přehnaná. Jedinou velkou a dobře viditelnou oblastí elementu jazyka HTML bývá většinou obsahová oblast. Díky tomuto zveličenému diagramu si však lépe popíšeme jednotlivé komponenty box modelu, přičemž budeme postupovat zevnitř směrem ven:

- **Obsah:** obsahová část modelu v sobě skrývá samotný obsah (jak jinak). Obsahu jsme se lehce dotkli v kapitole 1, „Úvod do jazyka CSS“, když jsme si popisovali elementy jazyka HTML. Obsahem může být text, obrázky nebo cokoliv jiného viditelného, co chceme zobrazit uživateli na webové stránce.
- **Vnitřní okraj:** vnitřní okraj elementu určujeme prostřednictvím vlastnosti `padding`. Vnitřním okrajem rozumíme volnou oblast okolo obsahu. Velikost vnitřního okraje můžeme definovat pro každou stranu zvlášť (například levý vnitřní okraj lze specifikovat deklarací `padding-left: 20px`) nebo pro všechny strany najednou – kupříkladu deklarací `padding: 20px 10px 30px 20px`. Při nastavování velikosti vnitřního okraje na všech čtyřech stranách zaráz používáme tzv. **zkrácenou vlastnost**. Později v této kapitole si předvedeme další zkrácené vlastnosti. Vlastnosti jazyka CSS, které přijímají více hodnot (jako vlastnost `padding`) mají obvykle stejné pořadí stran – horní straně odpovídá první hodnota a pak se přesouváme po směru hodinových ručiček až k levé straně, kterou reprezentuje čtvrtá hodnota. V tomto případě má tedy horní vnitřní okraj velikost 20 pixelů, pravý vnitřní okraj velikost 10 pixelů, spodní vnitřní okraj velikost 30 pixelů a levý vnitřní okraj velikost 20 pixelů.

- **Rámeček:** rámeček elementu nastavujeme pomocí vlastnosti `border`. Jedná o zkrácenou vlastnost, která v sobě spojuje vlastnosti `border-width`, `border-style` a `border-color`. Oranžovou přerušovanou čáru o tloušťce 4 pixely bychom tudíž nakreslili okolo elementu deklarací `border: 4px dashed orange`.
- **Vnější okraj:** poslední komponentou box modelu je vnější okraj. Vnější okraj se podobá vnitřnímu okraji, a dokonce ho definujeme podobnou syntaxí (například `margin-left: 15px` nebo `margin: 10px 20px 10px 20px`). Na rozdíl od vnitřního okraje ale není vnější okraj součástí elementu. Vnější okraj vytváří volnou oblast mezi cílovým elementem a okolními elementy.

Každý element na webové stránce má tyto komponenty box modelu. Někdy se výchozí vlastnosti těchto komponent mohou lišit u některých typů elementů. Například formuláře mají určitou výchozí šířku a výšku, i když tyto vlastnosti nenastavíme. V kapitole 1, „Úvod do jazyka CSS“, jsme si řekli, že neuspořádané seznamy (elementy `ul`) mohou mít předdefinované vnitřní a vnější okraje z interní šablony stylů webového prohlížeče.

Blokové versus řádkové elementy

Dále bychom si měli vysvětlit jednu důležitou koncepci jazyka HTML – většina elementů tohoto jazyka spadá do některé ze dvou kategorií, a to buď **blokové** nebo **řádkové**. Blokový element je spíše strukturální s cílem vytvořit určité rozvržení, kdežto řádkové elementy se obvykle nacházejí uvnitř blokových elementů a splývají s textem.

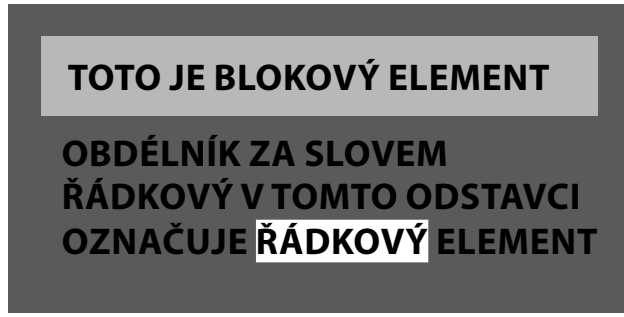
K blokovým elementům patří kupříkladu elementy `div`, `p` a `section`, zatímco mezi řádkovými elementy můžeme najít například elementy `span`, `b` a `em`. To bylo jen několik málo příkladů.

V kódu jazyka CSS můžeme změnit výchozí chování těchto elementů, a to pomocí vlastnosti `display`:

```
span {
  display: block;
}
```

Jak se asi chová blokový jinak oproti řádkovému elementu, není tak těžké uhádnout. Pochopení tohoto rozdílu je pro začínající vývojáře jazyka CSS velmi důležité. Obrázek 2.2 znázorňuje tento rozdíl.

Blokové elementy mají určité vlastnosti. Pokud jim nenastavíme konkrétní šířku, vždy se vodorovně roztahují na šířku svého rodičovského kontejneru a svisle se roztahují na výšku svého obsahu. Blokovému elementu tedy nemusíme nastavovat šířku, dokud například nechceme, aby byl užší, než je oblast vymezená jeho rodičovským elementem. Zcela výjimečně jsme donuceni nastavit blokovému elementu výšku – většinou ho necháme zvětšit se na velikost jeho obsahu.



Obrázek 2.2. Znázornění rozdílu mezi blokovým a řádkovým elementem

Již víme, že blokové elementy jsou strukturálními elementy, a proto standardně začínají až pod elementy, které zapíšeme do kódu jazyka HTML dříve. To je klíčová část pochopení rozdílu mezi blokovými a řádkovými elementy.

Řádkové elementy se totiž běžně chovají jako věty, slova a písmena v odstavcích. Z obrázku 2.2 je patrné, že řádkový element splývá přirozeně s textem a většinou obsahuje pouze text nebo jiné řádkové elementy. Z tohoto důvodu na řádkové elementy obvykle aplikujeme vlastnosti jazyka CSS, které ovlivňují vzhled textu. Například vlastnosti `line-height` a `letter-spacing` jsou vlastnosti určené pro řádkové elementy. Tyto vlastnosti tedy nebudou mít vliv na samotné blokové elementy.

Kromě toho není možné řádkovým elementům nastavovat šířku a výšku a také budou ignorovat horní a spodní vnější okraje. Na druhou stranu levé a pravé vnitřní a vnější okraje u nich definovat můžeme.

Pro účely rozvržení webové stránky se může výjimečně stát, že budeme potřebovat, aby se element choval z části jako blokový element, ale z části rovněž jako řádkový element. To je možné, pokud elementu nastavíme vlastnost `display` s hodnotou `inline-block`:

```
.priklad {
  display: inline-block;
}
```

Tímto způsobem získáme to nejlepší z obou světů – element zůstane součástí textu a budeme na něho moct aplikovat textové vlastnosti jazyka CSS, ale současně mu bude možné nastavovat šířku, výšku a vnější okraje stejně jako blokovému elementu. Dále v této kapitole použijeme tento řádkově-blokový element pro stylování hlavní navigace naší ukázkové stránky.

Zkrácené versus běžné vlastnosti jazyka CSS

Neméně důležité je porozumět rozdílu mezi zkrácenými a běžnými vlastnostmi jazyka CSS. Řekli jsme si, že deklarace v jazyce CSS se skládá z vlastnosti, dvojtečky, hodnoty a středníku.

Zkrácené vlastnosti se však mírně liší. Zkrácená vlastnost ve skutečnosti přijímá sadu hodnot, přičemž každá z nich odpovídá nějaké běžné vlastnosti jazyka CSS. Ukažme si proto příklad, který jsme si již představili – zkrácenou vlastnost `border`:

```
.priklad {
  border: dashed 2px blue;
}
```

Výše uvedené pravidlo stylu obsahuje jedinou deklaraci, ale lze ho přepsat i takto:

```
.priklad {
  border-style: dashed;
  border-width: 2px;
  border-color: blue;
}
```

Z tohoto srovnání je jasné, proč jen zřídka narazíme na běžné vlastnosti, které určují vzhled rámečku. Je mnohem jednodušší použít zkrácenou vlastnost – navíc si vystačíme s kratším kódem, což ve velkém projektu bude mít menší pozitivní vliv na rychlost načítání stránky a mnohem větší vliv na údržbu kódu.

Jestliže v deklaraci zkrácené vlastnosti vynecháme některou hodnotu, jí příslušná běžná vlastnost si obnoví svou výchozí hodnotu. Nyní spojíme dvě pravidla stylů, abychom zjistili, jak to funguje v praxi:

```
.priklad {
  border-style: dashed;
  border-width: 2px;
  border-color: blue;
}

.priklad {
  border: solid;
  color: green;
}
```

V tomto případě používáme stejný selektor u dvou rozdílných pravidel stylů. V kapitole 1, „Úvod do jazyka CSS“, jsme se naučili, že druhé pravidlo stylů má v takové situaci přednost před prvním – přepisuje tedy stejné vlastnosti z prvního pravidla stylu.

V prvním pravidle stylu definujeme všechny tři běžné vlastnosti pro rámeček – specifikujeme mu modrou přerušovanou čáru o tloušťce 2 pixely. Co je však výsledkem uplatnění obou pravidel stylů? Rámeček získá šířku 3 pixely (výchozí tloušťka viditelného rámečku) a zelenou barvu místo modré. Je tomu tak z toho důvodu, že pomocí zkrácené vlastnosti `border` definujeme styl rámečku `solid` (plná čára), ale už nedefinujeme hodnoty pro zbývající dvě vlastnosti `border-width` a `border-color`. Z tohoto příkladu je také patrné, že vlastnost `color` neurčuje jen barvu textu, ale také výchozí barvu rámečku.

Tento příklad se může zdát složitý, ale zkuste s ním trochu experimentovat a určitě ho za chvíli pochopíte. Měli byste si odnést ponaučení, že pokud nenastavíte všechny hodnoty zkrácené vlastnosti, chybějící běžné vlastnosti získají své výchozí hodnoty – nedědí je od existujících pravidel stylů.

Většinou to nezpůsobuje žádné problémy, ale v některých situacích můžeme získat neočekávané výsledky, jak popisuje článek „A Primer on the CSS Font Shorthand Property“[11].

Další důležitý poznatek je, že u některých zkrácených vlastností právě uvedená poučka neplatí – chybějící běžné vlastnosti se nevracejí ke svým výchozím hodnotám, ale dědí aktuálně nastavované hodnoty. Typickým příkladem jsou vlastnosti `margin` a `padding`:

```
.panel {
  padding: 20px 10px 15px;
}
```

Všimněte si, že teď specifikujete jen tři hodnoty. Na začátku této kapitoly jste se dozvěděli, že hodnoty mají být čtyři a představují postupně velikost vnitřního okraje na horní, pravé, spodní a levé straně. Tentokrát čtvrtá hodnota (velikost levého vnitřního okraje) chybí, ale prohlížeč automaticky nastaví 10 pixelů, což je velikost opačného okraje (druhá hodnota). Kdybyste vynechali dvě hodnoty, velikost spodního okraje by se shodovala s velikostí horního okraje podobně, jako by se shodovaly velikosti levého a pravého okraje.

Stejný princip platí také pro zkrácené vlastnosti `margin`, `border-color` a `border-width`. Následující dvě pravidla stylů se tedy zcela shodují:

```
.priklad {
  margin: 10px 20px 10px 20px;
}

.priklad {
  margin: 10px 20px;
}
```

V první deklaraci specifikujeme explicitně všechny čtyři hodnoty (pro vlastnosti `margin-top`, `margin-right`, `margin-bottom` a `margin-left`). Ve druhé deklaraci vynecháváme hodnoty pro

11 *http://www.impressivewebs.com/a-primer-on-the-css-font-shorthand-property/*

vlastnosti `margin-bottom` a `margin-left`, ale to nic nemění na tom, že stejně zdědí hodnoty vlastností `margin-top` a `margin-right`. Níže uvedené deklarace jsou rovněž duplicitní:

```
.priklad {
  border-width: 10px 10px 10px 10px;
}

.priklad {
  border-width: 10px;
}
```

Uvnitř první deklarace rovněž uvádíme všechny čtyři hodnoty, ale v druhé deklaraci necháváme na prohlížeči, aby zkopíroval tři chybějící hodnoty z jediné hodnoty, kterou jsme mu uvedli, a tou je šířka horního rámečku.

Měli byste se naučit používat zkrácené vlastnosti co nejdříve. Zbavíte tím svůj kód zbytečných znaků, díky čemuž bude čitelnější.

Rozvržení založené na obtékání

Pro začátek si představíme techniku rozvržení, která jako jediná funguje ve všech webových prohlížečích a nepoužívá tabulky jazyka HTML (tabulky nejsou vhodné pro rozvržení stránky[12]). Existuje několik nových technik, které si ukážeme dále v této kapitole. Ty však podporují jen některé moderní prohlížeče, a tudíž rozvržení pomocí obtékání je stále nezbytné, pokud chceme, aby naše stránky vypadaly dobře i ve starších prohlížečích (například v prohlížeči Internet Explorer ve verzi 7, 8 a 9). Dokonce i moderní prohlížeče neměly v době psaní této knihy dokonalou podporu těchto nových technik, takže obtékání stále kralovalo způsobům rozvržení stránky pomocí jazyka CSS.

U naší webové stránky Vyhledávač receptů existuje ideální místo, kde můžeme uplatnit rozvržení založené na obtékání – oblast s hlavním obsahem. Tuto oblast bychom chtěli svisle rozdělit do dvou sloupců, jak ukazuje obrázek 2.3.

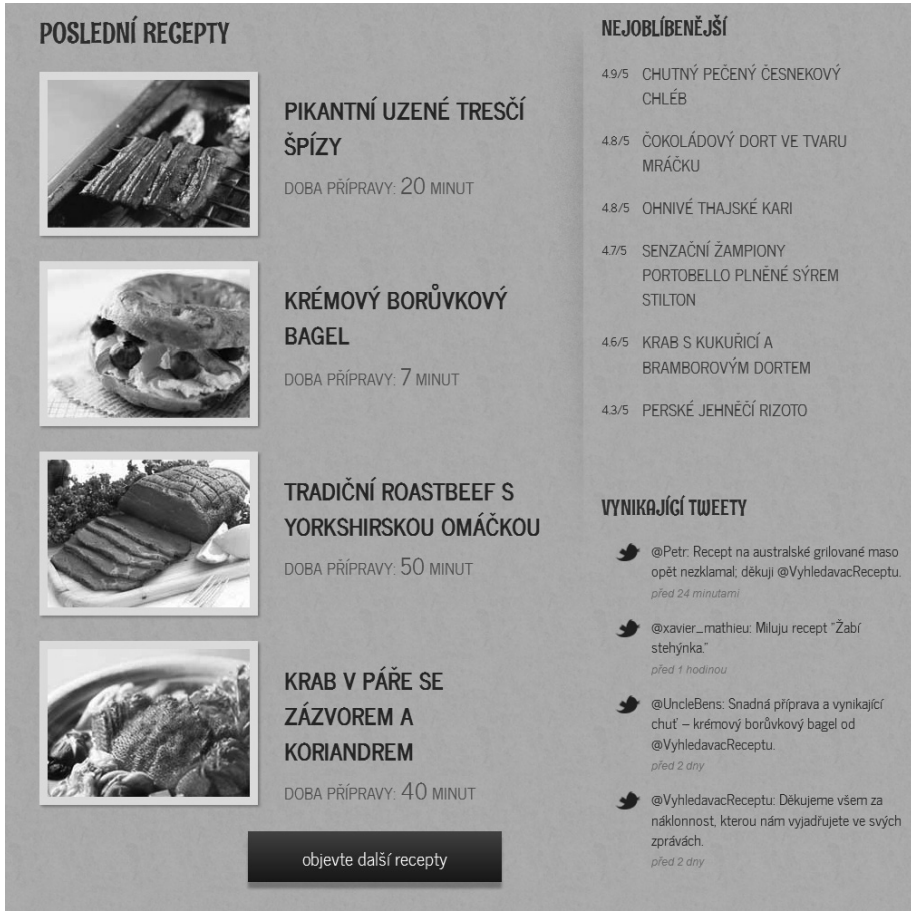
Přidejme tedy několik kaskádových stylů, abychom rozvrhli tuto část stránky:

```
.hlavni {
  width: 1020px;
  margin: 0 auto;
}

  .posledni {
    width: 640px;
```

12 *http://stackoverflow.com/questions/83073/why-not-use-tables-for-layout-in-html*

```
  float: left;
}
...
.postranni-panel {
  width: 360px;
  float: right;
}
```



Obrázek 2.3. Dvousloupcová oblast vyhledávače receptů

Tyto selektory jsme si přichystali v kapitole 1, „Úvod do jazyka CSS". V dokumentu HTML máme element s třídou `hlavni`, který obsahuje oba tyto sloupce. Na předchozím kódu je zajímavé především to, jakou deklaraci vlastnosti `margin` přidělujeme elementu s třídou `hlavni`.

Dříve jsme si řekli, že `margin` je zkrácená vlastnost, a výše uvedená deklarace odpovídá následující:

```
.hlavni {
  margin: 0 auto 0 auto;
}
```

Pro jednoduchost jsme vynechali poslední dvě hodnoty, jelikož prohlížeč je stejně sám doplní. Zatímco hodnota `0` je zřejmá (element s třídou `hlavni` nebude mít žádný horní a spodní vnější okraj), s hodnotou `auto` jsme se ještě nesetkali.

Pokud přiřadíme hodnotu `auto` levému a pravému okraji elementu, u nějž jsme explicitně specifikovali šířku, umístíme tento element vodorovně na střed jeho rodičovského elementu. Neumísťujeme však na střed obsah elementu, ale samotný element. Rodičovským elementem elementu s třídou `hlavni` je element `body`, jenž nemá žádnou explicitně nastavenou šířku. V důsledku této deklarace vlastnosti `margin` se tedy bude element s třídou `hlavni` nacházet vždy na středu okna webového prohlížeče, a to bez ohledu na jeho šířku.

Tuto techniku si ideálně nechejte vytetovat na hřbet ruky, ať ji nezapomenete – budete ji potřebovat často. Funguje ale jen při umísťování elementů na střed ve vodorovném směru. Umísťování elementů na střed svisle je v jazyce CSS o něco těžší.[13]

Tip: Jednotky v jazyce CSS

U příkladů v této knize jste si možná všimli, že některé číselné hodnoty mají příponu `px`. Tato přípona označuje pixel, přičemž se jedná o jednotku jazyka CSS. Dalšími jednotkami jsou kupříkladu `%` (procenta), `em` nebo `rem`. Více se o jednotkách kaskádových stylů dozvíte v následující kapitole.

Dále v naší šabloně stylů vybíráme dva elementy uvnitř elementu s třídou `hlavni`. Prvním z nich je levý sloupec, který má třídu `posledni` (bude sloužit jako přehled posledních receptů). Druhým sloupcem je element `aside`, jemuž jsme přidělili třídu `postranni-panel`.

V této kapitole jsme si vysvětlili rozdíly mezi blokovými a řádkovými elementy. Oba naše sloupce jsou blokové elementy, takže standardně se zobrazují pod sebou, a ne vedle sebe tak, aby vyplňovaly prostor vymezený jejich rodičovským elementem (v tomto případě se jedná o element s třídou `hlavni`), což bychom potřebovali.

Abychom dosáhli dvousloupcového rozvržení obsahu, používáme vlastnost `float`, s jejíž pomocí sdělujeme prohlížeči, na jakou stranu naší stránky bychom chtěli umístit jednotlivé elementy. Vlastnost `float` přijímá některou z těchto hodnot: `none`, `left`, `right` nebo `inherit`.

Požadovaného výsledku tudíž dosahujeme tak, že používáme deklaraci `float: left` pro levý sloupec a deklaraci `float: right` pro pravý sloupec. Navíc definujeme šířku obou sloupců tak, aby vypadaly podobně jako na našem návrhu z programu Photoshop.

V této fázi by měl obsah elementu s třídou `hlavni` vypadat podobně jako na obrázku 2.4.

13 *http://blog.themeforest.net/tutorials/vertical-centering-with-css/*

Obrázek 2.4. Po přidání vlastnosti float jsme získali dva sloupce

Naše stránka je zatím poněkud nevýrazná, že? Dosud jsme nenapsali mnoho stylů, takže současný vzhled je spíše výsledkem načtení souboru Normalize.css spolu s výchozími deklaracemi webového prohlížeče (například modrý text odkazů).

Rušíme obtékání

Rušit obtékání musíme u téměř všech stránek, v nichž používáme rozvržení založené na obtékání. Aby bylo jasnější, proč tomu tak je, přidejme dočasně dvě deklarace k elementu s třídou `hlavni`, což je element, který obsahuje naše obtékané sloupce:

```
.hlavni {
  width: 1020px;
  margin: 0 auto;
  outline: solid 1px red;
```

```
  background: green;
}
```

Poslední dvě deklarace jsou pouze dočasné a mají za úkol demonstrovat problém s obtékáním. Obrázek 2.5 ukazuje, co se stane, když přidáme tyto styly a obnovíme stránku.

Obrázek 2.5. Obtékané sloupce způsobí, že se jejich rodičovský element smrskne

Očekávali bychom, že pod obsahem elementu s třídou `hlavni` se objeví zelené pozadí a okolo něj červený obrys. Vidíme však pouze 2 pixely vysokou červenou čáru nahoře tohoto elementu.

Stalo se to právě proto, že obsahuje obtékané dceřiné elementy. Pokud element obsahuje obtékané elementy a žádné neobtékané elementy, sbalí se, jako by v něm vůbec nic nebylo. Postačí nám tedy zrušit některé z deklarovaných obtékání a zelené pozadí a červený obrys se objeví.

Tuto situaci tedy vyřešíme tak, že „zrušíme“ obtékání. Vložíme proto následující kód do našeho souboru *styly.css*:

```
.cf:before,
.cf:after {
  content: " ";
```