Některé z nich jsem tedy pro naše potřeby zakoupil a barevně upravil.

Upravené ikony Maxe Gribojedova, které využijeme pro komunikaci benefitů e-shopu. iconfinder.com/enotmaks

Je to hotové. Teď se vrhneme na kódování. Musíme si předtím povědět, jak je to v responzivním webdesignu s CSS jednotkami.

Jednotky pro tvorbu webu (em, rem, %, px, vh, vw): Kde použít jakou?

Pojďme si tady shrnout všechny CSS jednotky použitelné v dnešním webdesignu. A na příkladu ukázat, k čemu se která hodí.

Za základní jednotku pro svůj způsob práce považuji jednotky relativní k velikosti písma – `rem` a `em`.

Ani ty ale nepovažuji za žádné Supermany mezi CSS jednotkami. Prostě se nehodí na vše. Pro různé účely ještě budeme potřebovat

i Spidermana, Batmana a další jejich kolegy a kolegyně.

Rychlý přehled použitelných jednotek

Zapamatujte si hlavně následující šestici.

Jednotka	Jak počítá rozměr?
`rem`	relativně k velikosti písma na prvku `<html>`
`em`	relativně k velikosti písma na elementu
`px`	přepočtený pixel, CSS pixel
`%`	procenta relativně k rodičovskému elementu
`vw`	procento ze šířky okna prohlížeče
`vh`	procento z výšky okna prohlížeče

Existují samozřejmě ještě další: namátkou `pt`, `ex` nebo `vmax`. Buď je ale jejich využitelnost malá, nebo skoro žádná, takže pro zjednodušení je zatím úplně vynechám.

Které jednotky v jakých situacích použít?

V tomhle textu dost dbám na to, abychom si nerozbili přirozenou dědičnost velikosti písma v prohlížečích. Kromě nás, autorů stránek, si ji totiž může chtít změnit uživatel a občas ji mění i prohlížeče. Proto vám pro různá použití doporučuji různé jednotky:

- Základní velikost písma v dokumentu: %
- Rozměry vycházející z velikosti písma dokumentu: rem
- Rozměry vycházející z velikosti písma rodiče: em
- Media Queries: em
- Výška řádku: číslem bez jednotky
- Rámečky, dekorace: px

- Typografie podle velikosti okna: vw

Je ale možné, že si skoro všude vystačíte s px. K tomu se dostávám na konci textu.

Připravil jsem jednoduché demo, ve kterém představuji všechny nejčastější scénáře nastavování rozměrů v CSS. Projdeme si to v textu, ale je také online: cdpn.io/e/dvdxWG

Základní velikost písma v dokumentu: %

Výchozí velikost písma je v drtivé většině prohlížečů 16px. Existují ale méně významné prohlížeče, které velikost písma nastavují jinak. Prostě tak, aby se nám na konkrétním zařízení lépe četlo. Například prohlížeč v Kindle 3 měl výchozí velikost písma 26px. vrdl.in/16px

Pokud vám výchozí velikost písma nevyhovuje, změňte to pružnými jednotkami, například procenty:

```
html {
  font-size: 125%;
}
```

Nastavíte tak o čtvrtinu větší písmo, než je výchozí. Skoro u všech prohlížečů tedy 20px. V Kindle 3 by to bylo 33px. Je důležité si uvědomit, že je to v pořádku. Pokud prohlížeč nastavuje jinak velké písmo, dělá to z rozumných důvodů.

Proč ne px? Protože lidé si zvětšují písmo v prohlížečích

Pokud bychom už tady použili px, našim milým uživatelům bychom zakázali měnit si výchozí velikost písma v prohlížečích.

Pozor, nebavíme se o „zoomování“, ale o zvětšení velikosti písma pro všechny weby. Taková věc stále existuje v prohlížečích nebo v operačních systémech. A ano, lidé to používají. Asi taky jednou

budeme. Dělají to totiž lidé s horším zrakem nebo třeba jen méně kvalitními displeji.

Vývojář z Archive.org Evan Minto to měřil a zjistil, že velikost písma si v prohlížeči změnila 3 % jejich uživatelů. Jak trefně přirovnává, je to více než podíl návštěvníků používajících Internet Explorer, Edge nebo Operu Mini. Protože chceme vytvářet řešení s co nejširším uživatelským zásahem, neměli bychom to ignorovat. Zdroj: medium.com/@vamptvo/5cfb20831773

Rozměry vycházející z velikosti písma dokumentu: rem

Velikost písma, vnější a vnitřní okraje, ale i další vlastnosti v dokumentu a komponentách prostě nastavuji v `rem`. `1rem` (1 root `em`) obsahuje výchozí velikost písma nastavenou autorem pro dokument a případně ještě upravenou uživatelem nebo prohlížečem, jak už jsme viděli.

Pokud ji na dokumentu nezměníme, platí, že `1rem = 16px`.

```
p { margin-bottom: 1rem; }
h1 { font-size: 2rem; }
```

Odstavcům tímto kódem nastavím spodní vnější okraj na výšku písma. Nadpisy první úrovně budou dvojnásobně velké oproti standardní velikosti písma.

Používat `rem` je výhodné i z pohledu vývojáře:

- V `1rem` máte uloženou základní velikost písma a nemusíte si pamatovat, jestli je to 12, 14, 16, nebo kolik vlastně pixelů.
- Šířka layoutu nastavená v `rem` bude dodržovat optimální délku textu, i když si uživatel písmo zvětší. (Vzpomeňte si na text

o typografii.)
- Díky `rem` je také možné zvětšit celý dokument na konkrétních rozmezích designu. (Budeme rozebírat v části o autorském „zoomování“ dokumentu).

Co když dostávám podklady v px?

Možná jste zvyklí při převodu designu do kódu pracovat v `px`, protože grafici a grafičky dodávají podklady v takových jednotkách. Jak už jsem ale napsal, nastavovat v `px` cokoliv odvozeného od hlavní velikosti písma komplikuje život mnohým uživatelům.

Jak z konfliktu design versus přístupnost ven? Může vám pomoci automatická úprava CSS. Existují minimálně dva pluginy do PostCSS, které kód napsaný v `px` převedou do `rem`:

- Plugin „postcss-line-height-px-to-unitless“ například převede výšku řádku do bezjednotkové podoby: Z `font-size: 16px; line-height: 26px;` udělá `font-size: 16px; line-height: 1.63;`. github.com/makotot/postcss-line-height-px-to-unitless
- Plugin „postcss-pxtorem“ zase zařídí převod pro vybrané vlastnosti. Z `font-size: 16px` prostě udělá `font-size: 1rem`. github.com/cuth/postcss-pxtorem

`rem` tedy považuji za hlavní jednotku pro tvorbu rozhraní. Velmi se nám ale také hodí `em`.

Rozměry vycházející z velikosti písma rodiče: em

Jednotka `em` obsahuje velikost písma elementu, nikoliv dokumentu.

```
html {
  font-size: 100%; /* = 16px */
}

p {
  padding: 1em; /* = 16px */
}

.button {
  font-size: 75%;
  padding: 1em; /* = 12px */
}
```

Vidíte, že `1em` znamená v různých místech dokumentu různé věci. Někde to může být fajn: Třeba v případě, že kódujete komponentu, jejíž velikost *má být* určená právě velikostí písma na rodičovském prvku.

Ale vývojářům se s „emky“ samozřejmě pracuje trošku hůř než s „remky“. Ne každý se chce stát pochodující kalkulačkou pro převod mezi `em` a pixely.

Jen pozor, em není čtverčík

„Čtverčík“ je typografická jednotka, která se počítá ze šířky velkého „M“. `em` se občas nepřesně jako čtverčík označuje. Jenže to by pak bylo pro různá písma různě velké.

Není. W3C `em` definovalo jinak. Jeho velikost v kořeni dokumentu je ve všech prohlížečích a při použití jakéhokoliv písma stejná – `16px`. Pokud to ovšem nezmění někdo ze zákeřné trojice uživatel, prohlížeč či autor stránky. Vysvětlují to například uživatelé diskuze na JakPsátWeb: vrdl.in/oyqwn

Media Queries: em

V textu o Media Queries píšu, proč nepoužít `px` (opět kvůli

zvětšování písma) a `rem` (kvůli chybě v Safari). Proto nám zbývají `em`:

```
@media screen and (min-width: 30em) { }
```

Opět je ale možné použít automatický převod z `px`, protože (i mně) se tady s CSS pixely pracuje lépe. Pomůže plugin do PostCSS jménem „postcss-em-media-query“. github.com/niksy/postcss-em-media-query

Výška řádku: číslem bez jednotky

Hodnota bez jednotky je pro výšku řádku specifická, ale dává naprostý smysl:

```
h1 {
  line-height: 1.5;
}
```

V ukázce je výška řádku prostě jedenapůlnásobek velikosti písma nadpisu první úrovně. A je nám pak úplně jedno, jak si který čert nastaví velikost písma.

Proč je to lepší než nastavení „natvrdo“ v `rem`, `em` nebo `px`? Pokud se autorsky nebo uživatelsky v některém kontextu změní velikost písma, nemusíte pak už měnit nastavení výšky řádku.

Layout: procenta, ale i další jednotky

Pro layout se dobře hodí procenta. Například:

```
.layout-col {
  width: 50%;
}
```

Raději připomínám, že se vždy počítají ze šířky nejbližšího rodičovského prvku.

Použitelných jednotek pro layout je ale více:

- Procenta nebo `vw` se roztahují podle šířky okna, `vh` podle jeho šířky.
- `rem` a `em` podle velikosti písma.
- Ve flexboxu je možné používat také absolutní jednotky (`flex: 1`).
- V CSS Grid zase takzvané podílové jednotky (`grid-template-columns: 3fr 1fr`).
- Občas se hodí i fixní rozměry v `px`.

Rámečky, dekorace: px

Doporučuji `px` používat jen tam, kde potřebujete precizní vyjádření v pixelech. Třeba pro rámečky mezi prvky navigace:

```
.nav-item {
  border-left: 1px solid white;
}
```

Jen pro jistotu připomínám, že už nejde o hardwarový pixel. Psal jsem o tom v první kapitole v části o CSS pixelu.

Typografie podle velikosti okna: vw

Můžete potřebovat i zvětšování a zmenšování velikosti písma podle šířky nebo výšky okna. Pak si vzpomeňte na jednotky `vw` (viewport width) nebo `vh` (viewport height).

Například tento nadpis z příkladu bude mít velikost písma `2rem` a k tomu vždy dvě procenta ze šířky okna:

```
.heading {
  font-size: calc(2rem + 2vw);
}
```

Triky s responzivní typografií se více zabývám v přespříští podkapitole.

A ještě jeden odkaz jako příklad: cdpn.io/e/dvdxWG

Co když chci přesto používat hlavně px?

Nemyslím si, že zemře hodně koťátek, když to uděláte. Použití px je u velké části typů designu na implementaci výrazně pohodlnější.

Přesto se ujistěte, že písmo v návrhu je dostatečně veliké na to, aby je přečetla většina uživatelů. Jako základ se obecně doporučuje alespoň oněch 16px.

Raději se také sami sebe zeptejte, zda vám nevadí nic z následujícího seznamu:

- Uživatelům, kteří si změnili písmo v systému nebo prohlížeči (na Archive.org asi 3 %), se jejich nastavení na vašem webu neprojeví. Zůstává jim možnost zoomovat celou stránku.
- Změna velikosti písma nebude správně reflektována v Media Queries. (Řešíme v tipech k Media Queries).
- V návrhu designu se nepočítá s elastickou typografií, zvětšující se podle viewportu.
- Designér nebo designérka rovněž nepočítali s pružnou změnou velikosti komponenty podle velikosti písma rodiče ani s globální změnou velikosti písma v určitých breakpointech designu.

Způsob práce při návrhu designu, který v knížce ukazuji, by v mnoha položkách tohoto kontrolního seznamu úpěl, skřípal nebo přímo selhal. Budeme se proto v dalších textech px spíše vyhýbat.