



VZHŮRU DO **(RESPONZIVNÍHO)** WEBDESIGNU

Martin Michálek

Obsah

Na úvod

Kapitola 1: Proč vlastně responzivní webdesign?

Kapitola 2: Úvod do tvorby webu

Kapitola 3: Dokument jako základ

Kapitola 4: Viewport na mobilech

Kapitola 5: Rychlosť načítania

Kapitola 6: Obrázky a ďalšie médiá

Kapitola 7: Návrh rozhrania v ére mobilov

Kapitola 8: Design komponent rozhrania

Kapitola 9: Media Queries a layout

Kapitola 10: Responzívna navigacia

Kapitola 11: Testovanie responzívnych webov

Na záver

Na úvod

Milá čtenářko, milý čtenáři,

když jsem začal přemýšlet nad knihou o responzivním designu, měl jsem v hlavě naprosto jasno. Dodržím postup, kterým jsem napsal první e-book – „Vzhůru do CSS3“. Vezmu tedy imaginární otvírák na konzervy, otevřu si hlavu a na klávesnici vysypu technické návody, tipy a triky z oblasti responzivního designu.

V případě „Vzhůru do (responzivního) webdesignu“ to ovšem dopadlo jinak. Psát o technologiích a zamlčet jejich vazbu na design je totiž v případě responzivního designu nemožné.

Když například chci psát o technologii Media Queries, musím dodat designérské přemýšlení nad volbou breakpointů a nad postupem návrhu nejprve pro mobily, Mobile First. Pokud bych to neudělal, vývojář bude psát kód způsobem, který mu zbytečně zkomplikuje život. Designérovi zase nedojde, proč vývojář zvedá obočí nad některými jeho nápady. HTML, CSS a design jsou zkrátka spojené nádoby.

Měl jsem v tom znovu jasno: Kniha bude o implementaci a *návrhu* responzivních uživatelských rozhraní! Jenže jak se brzy dozvíté, přívlastek *responzivní* už dnes považuji za přežitý. Responzivní je (nebo by měl být) každý web. Proč to přídavné jméno vůbec používat?

Držíte v ruce knížku, která by se mohla jmenovat jen „Vzhůru do webdesignu“. A když z webdesignu vezmeme jen oblast návrhu a implementace uživatelského rozhraní, byla by to naprostá pravda.

V částech o designu se ale zaměřuji především na přechod od webů navržených pro počítače k webům fungujícím na všech zařízeních. Na *responzivní* fázi webdesignu. Proto se v názvu objevila závorka.

Přeji vám příjemné čtení!

Martin Michálek

Děkuji!

Ještě jednou díky za zakoupení e-booku „Vzhůru do (responzivního) webdesignu“. Vážím si vaši důvěry. Díky ní mohu e-book dále vylepšovat a rozšiřovat.

Pokud se k vám náhodou dostal jinou cestou, nebojte – nic není ztraceno. Stále můžete dát vše do pořádku koupí licence.

vrdl.cz/ebook-responzivni

Autor



Martin Michálek je frontend designér na volné noze s takřka dvacetiletou praxí. Navrhoje, implementuje a radí s uživatelskými rozhraními responzivních webů. V poslední době spolupracoval se značkami, jako je Česká televize, VašeČočky.cz nebo Svobodná Evropa.

O moderních technologiích napsal e-book „Vzhůru do CSS3“. Přiležitostně píše pro Zdroják.cz nebo Lupa.cz, jednou také na Smashing Magazine. Spravuje Vzhůru dolů, blog určený zejména profesionálním webovým vývojářům. Vede spolek Frontendisti.cz, který se zaměřuje na vzdělávání webařské komunity, organizuje srazy a publikuje přednášky.

Kdykoliv může, je se ženou a dětmi v pražském Kunratickém lese. V jeho sousedství také napsal následující řádky.

Pro koho knížka je a jak ji číst?

Během psaní „Vzhůru do (responzivního) webdesignu“ jsem myslел hlavně na tři skupiny webařů a webařek:

1) Vývojáři a kodéři

Vy z knížky využijete největší část. Získáte přehled o postupech a technologiích používaných pro tvorbu dnešních responzivních webů. Ovšem – každý frontend kodér a skoro každý vývojář dotváří, nebo dokonce navrhoje i nějaké části uživatelských rozhraní. Mezi řádky a v netechnických kapitolách se tedy můžete inspirovat tipy pro tvorbu a úpravy responzivních rozhraní.

2) Designéři, grafici, marketéři

Vaši pozornosti doporučím hlavně kapitoly o statistikách, rychlosti načítání, návrhu rozhraní v éře mobilů, procesu návrhu UI a responzivních navigacích. Věřím ale, že vás nalákám i do dalších textů a získáte tak hrubý přehled o technologických základech, na kterých weby společně stavíme.

3) Začátečníci a méně zkušení

Text knihy jsem psal tak, aby se dal číst i jako učebnice tvorby webových uživatelských rozhraní. Vynechávám všechny pokročilé nástroje, nezmíňuji složité techniky a občas sám skřípu zuby nad tím, jak moc musím zjednodušovat. Přeji si ale, aby kniha dávala smysl i lidem, kteří se k tvorbě webu dostali teprve nedávno a třeba ještě neví, do které ze škatulek „designéři“ a „vývojáři“ se časem zařadí.

Co předpokládám, že čtenáři umí?

Pro pochopení technických částí budete potřebovat dobrou znalost HTML a CSS a hrubý přehled o moderních technologiích zmíněných v e-booku „Vzhůru do CSS3“. Netechnické části předpokládají jen to, že máte základní přehled o procesu návrhu webu.

To podstatné vyzkoušíme na příkladu

V textu často ukazují principy, které se špatně vysvětlují na umělých ukázkách. Proto se knihou jako červená nit táhne příklad.

Kvůli zachování tempa vyprávění jsem se samozřejmě i v procesu

jeho tvorby dopustil mnoha zjednodušení. Na příkladu ale ukážu mnoho věcí:

1. Definici potřeb uživatelů a cílů webu
2. Přípravu obsahu webu
3. Výběr barvy a typografie
4. Návrh a nakódování dokumentové vrstvy designu
5. Implementaci responzivních médií
6. Návrh komponenty uživatelského rozhraní
7. Rozvržení stránky

Jednotlivé kroky, ale i hotovou verzi si samozřejmě můžete stáhnout, zkoušet nebo využít pro vaše potřeby.

Co je nového?

Čtete „Vzhůru do (responzivního) webdesignu“ ve verzi 1.2 z listopadu 2018. Tady je seznam obsahových novinek:

- Nový text: Zobrazování webů na chytrých hodinkách Apple Watch.
- Nový text: Jak nastavovat breakpointy?.
- Nový text: Breakpointy v CSS.
- Přepsaná kapitola o viewportech: Viewporty, meta značka pro viewport a nový text o zjištění velikosti okna.
- Přepsaný text o jednotkách v CSS.
- Requiem za mrtvé technologie: V e-booku už nepíšu o Internet Exploreru 10 a starších, prohlížeči Android Browser, nebo operačním systému Windows Phone.
- Opravdu hodně drobných úprav, vylepšení a vyleštění.

Hlášení případných chyb a náměty na vylepšení pro příští verze velmi vítám: martin@vzhurudolu.cz.

A teď už přeji příjemné čtení.

Kapitola 1: Proč vlastně responzivní webdesign?

Nevím, jak vás, ale mě jako dítě pohádka „O perníkové chaloupce“ vážně štvala. Před usnutím jsem si zkoušel představit ten temný les. To šlo. Jenže malebný palouk uprostřed? A domek s perníkovou střechou?! Okolnosti byly podezřelé a jasně naznačovaly, že v chaloupce může dojít k jistým obtížím. A stejně si tam Jeníček s Mařenkou klidně vlezli. To nikdy nepochopím.

My se poučíme. Než otevřeme dveře webdesignu, pořádně prozkoumáme jeho okolí.

Čím webdesign prochází a co jej v nejbližší době čeká? Podívejme se na změny, které se v technologickém světě udaly a zřejmě ještě udějí a které se nějak promítly do profesních životů nás nebohých webařů.

1. Příchod mobilů není jedinou změnou, na kterou reagujeme. Ale je tou nejvýznamnější.
2. Nové jsou i displeje enormně velké. Nevíte, proč na ně po rád zapomínáme?
3. Malá zařízení navíc mohou mít neuvěřitelné vysoká rozlišení. Ještě čerstvěji jsou na trhu také hybridní zařízení, umožňující ovládání dotykem i myší.
4. Prohlížeče se množí jako myši ve sklepě perníkové chaloupky.
5. Z temného lesa se ke všemu vynořují nová, k internetu připojená zařízení.
6. V každé části vám ukážu statistiky s aktuálními trendy, ale

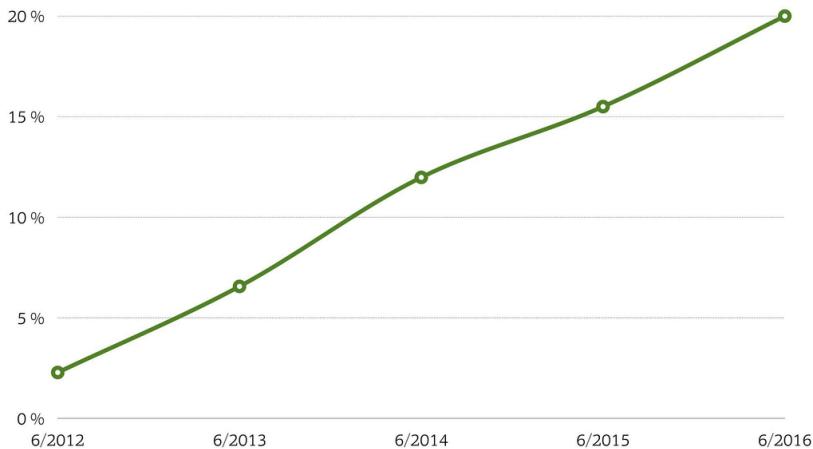
musím zmínit ještě ultimátní argumentaci. Googlem. Za chvíli ji uslyšíte.

7. Nakonec se v části o webech mobilních, responzivních a adaptivních zaměříme na porovnání různých přístupů k tvorbě webu. Který je lepší, co myslíte?

Budeme mít hotovo, ještě než Jeníček s Mařenkou dostanou kakao a teplou bábovku... *Teplou bábovku...* V noci! No nebylo to krajně podezřelé?

Mobily, jsou tady mobily

Proč promeškat příležitost ukázat si graf s rostoucí návštěvností z mobilních zařízení? Jdeme na to.



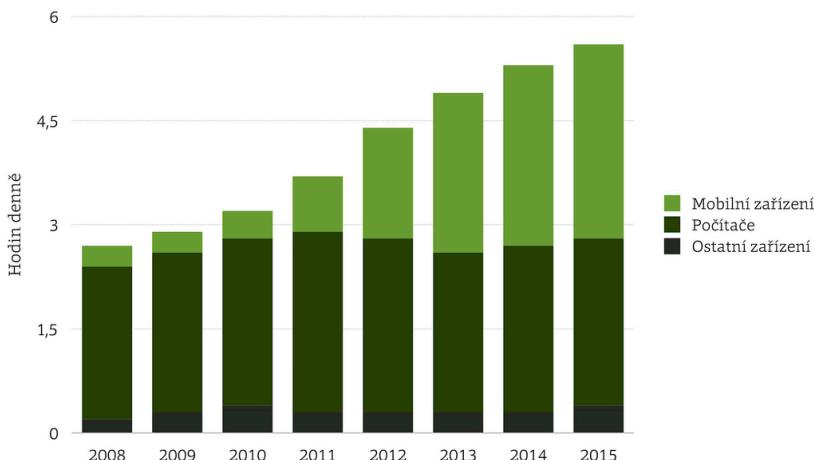
Podíl mobilních prohlížečů na celkovém počtu zobrazení stránek k polovině roku 2016.
Zdroj: Gemius SA, gemiusTraffic, Rankings.cz

Čísla z Rankings.cz ukazovaly asi pětinový podíl na zhlédnutých stránkách velkých webů v ČR. Tam se mobily a tablety samozřejmě

nezastaví.

Na vyspělejším Západě se podíl mobilů dostal už přes polovinu a dále se zvyšuje. Ani mezi českými weby už dnes nejsou výjimečné kousky s nadpoloviční návštěvností z mobilních zařízení. Ostatně než sledovat obecná data, dívejte se raději na čísla z vašich projektů.

V další statistice zase můžeme vidět, jak moc narostl počet hodin, které lidi denně tráví na mobilních zařízeních. Z dvaceti minut v roce 2008 na téměř tři hodiny v roce 2015.



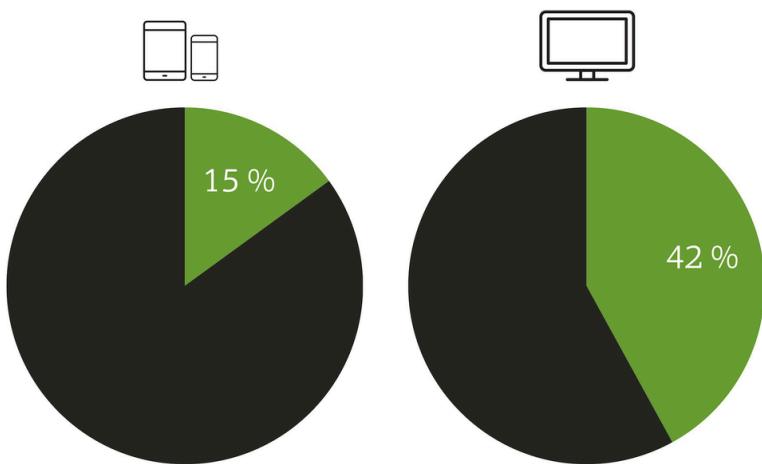
Podíl času tráveného na mobilních zařízeních roste, ovšem ne na úkor počítačů. V roce 2015 to bylo už 51 % z průměrných 5,6 hodiny denně. Zdroj: eMarketer. vrdl.in/ikr1p

Ale vy už dávno víte, že mobily jsou pro Web zásadní, že ano? Ještě alespoň několik praktických odkazů:

- Česká čísla mám z Rankings.cz.
- Aktuální světové statistiky hledejte na Statcounter.com.
- Návod, jak vytáhnout vlastní čísla z Google Analytics:
vrdl.in/gamobily.

Pro péči o ty malé úplně zapomínáme na velké

Vlastně se my webdesignéři chováme jako správní rodiče. Pipláme se s těmi roztomilými, mrňavými telefony. Jenže jsme zapomněli, že z našich dalších dětí, desktopových monitorů, mezičím vyrostli pěkní čahouni. Nebo vlastně cvalíci, protože se nám roztahli hlavně do šířky.



Na blog Vzhůru dolů v lednu 2017 přišlo 42 % návštěv s rozlišením obrazovky 1921 pixelů a širším. Z tabletů a mobilů jen 15 %

Na webech jiného klienta, cestovky Rekrea s populačním vzorkem bližším českému průměru, evidujeme kolem 20 % uživatelů cvalíkovitých monitorů. To je stejný podíl jako mají přistupující z mobilních zařízení.

I váš web může být na velkých rozlišeních špatně čitelný kvůli nedostatečné velikosti písma nebo prostě jen nevyužívá potenciál

širokého okna. Nespokojte se s tím, že na velkých displejích tvoří rozhraní webu úzkou nudli uprostřed prázdné plochy okna prohlížeče. Jedním z možných řešení, totiž zvětšováním rozvržení webu pomocí jednotky **rem**, se budeme zabývat v kapitole [typografie](#).

Vývoj: podíl konvenčních tabletů klesá, hybrydy rostou

Z jakých zařízení budou lidé v nejbližších letech přistupovat na naše weby méně a z jakých více? Srovnání se dneškem by vypadalo asi takto:

Prodeje klesají	Prodeje rostou
Počítače ovládané jen myší, dnešní tablety	Hybridní notebooky, profí tabletov, velké telefony, velmi malé notebooky

Podle odhadů agentury Gartner to totiž vypadá, že minimálně do roku 2018 budou klesat prodeje tradičních notebooků a desktopových počítačů. Stejně tak ale tabletů. Porostou jen prodeje chytrých telefonů a malých přenosných notebooků. vrdl.in/807ya

Neřekl bych ale, že se budeme se středně velkými displeji setkávat méně často. Rostou prodeje větších chytrých telefonů – phabletů – a prodávají se právě i ty velmi malé notebooky.

Gartner také ukazuje, jak se zvyšují prodeje hybridních notebooků, které je možné ovládat myší i dotykově. Mezi lety 2014 a 2015 o celých 70 procent. vrdl.in/scx2m

Další rostoucí skupinou jsou profesionální tablety jako iPad Pro. Jde o velké „dotykáče“, ke kterým občas připojíte klávesnici. V řadě

manažerských, administrativních, ale i designérských profesí dle mého názoru z velké části nebo úplně nahradí notebooky.

Co se týká škály velikostí obrazovky, musíme holt z těchto důvodů počítat se všemi, které nás napadnou. Nový je také kombinovaný způsob ovládání u hybridních notebooků: tabletom a myší. Ale ani to by nás responzivní webaře překvapovat nemělo.

CSS rozlišení

Na školeních se stále setkávám se strachem mnohých webových tvůrců z ohromných rozlišení displejů posledních modelů mobilů. Výrobci dnes udávají až FullHD plochu pro zobrazování. To je 1920 × 1080 pixelů. Znamená to, že se na pětipalcových displejích budou weby zobrazovat stejně jako na velkých monitorech a budou jen zmenšené do trpasličí velikosti?

Netrapme se tím. Nás webové tvůrce totiž hardware rozlišení vůbec nezajímá. Prohlížeče jej pro nás přepočítávají do „CSS rozlišení“.

V tabulce na příkladu několika zařízení ukazují hardware a CSS rozlišení vybraných modelů telefonů:

Zařízení	Hardware rozlišení	CSS rozlišení
iPhone 4	640 × 960	320 × 480
Google Nexus 7	800 × 1280	604 × 966
HTC One	1080 × 1920	360 × 640
Xiaomi Mi3	1080 × 1920	270 × 480

Vykreslení CSS rozlišení do hardware rastru pak obstarají samy prohlížeče.

„Retina“ displeje

CSS rozlišení má naprostá většina dnešních mobilních zařízení a občas nějaký ten notebook. Většinou se tyto obrazovky označují jako „Retina“ displeje. To není úplně přesné, protože jde o marketingový název dnešních displejů na zařízeních Apple. Nepřesnosti mi ale nevadí. Hlavně, že si budeme rozumět.

px je „CSS pixel“

CSS jednotka „pixel“ dříve vždy odpovídala hardwarovému pixelu, tedy skutečně fyzickému objektu. Dnes už to neplatí. **px** je „CSS pixel“, jednotka, která se na každém zařízení vykresluje do jiného počtu hardwarových pixelů. Detailní technikálie tady ale nejsou tak důležité jako důsledky.

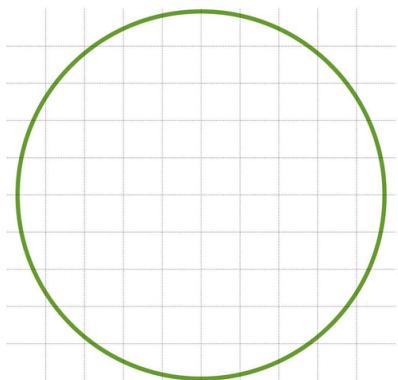
Nesprávně vložená bitmapová grafika bude rozostřená

Dejme tomu, že jsme si v grafickém programu připravili obrázek kružnice. Odtamtud ji vyexportujeme ve výšce a šířce 10 pixelů do formátu PNG. Teď ji vložíme do stránky:

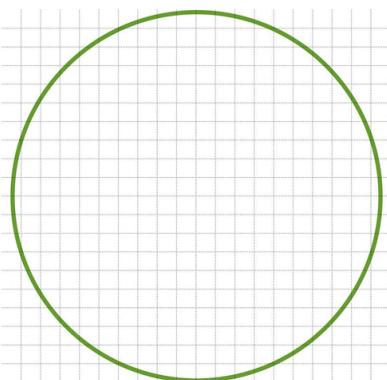
```

```

Běžný displej



„Retina“ displej



Hustota hardwareových pixelů je na „Retina“ displeji násobně vyšší

Na běžném displeji se soubor s obrázkem obsahující data pro 10×10 pixelů vykreslí do mřížky 10×10 hardwareových pixelů. Tady je svět ještě v pořádku.

Jenže na „Retina“ displeji je potřeba vykreslit obrázek do mřížky 20×20 . Tady vznikají problémy, protože prohlížeč má data jen pro poloviční počet pixelů. Druhou polovinu si tedy musí vymýšlet. Snaží se to dělat chytře, ale obrázek bude působit více či méně rozostřeně.

Řešení: používejte SVG a responzivní obrázky

Řešením je používat co nejvíce vektorové grafiky. Ta neobsahuje konkrétní pixely, ale informaci o křivkách, které pak prohlížeč vykreslí dokonale ostře. Využívejte formát SVG, o kterém píšu v kapitole o obrázcích.

U fotografií a obecně bitmapových formátů (PNG, JPG...) je

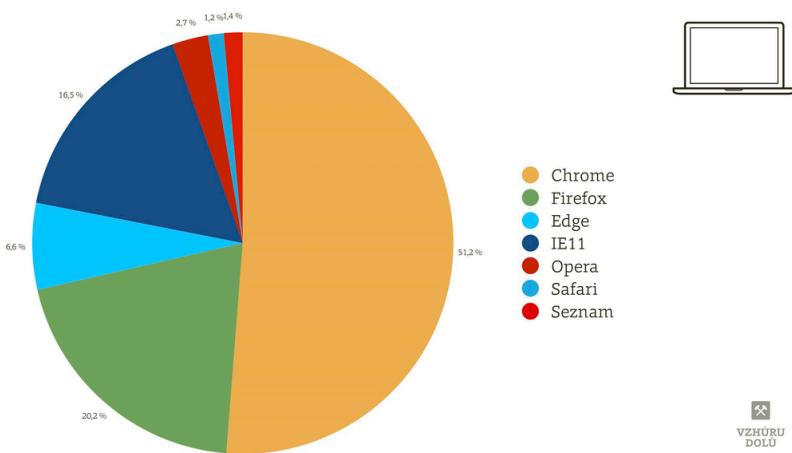
k dispozici více řešení. Píšu o nich opět v kapitole o obrázcích.

Úroda prohlížečů

Podívejme se, jak to vypadá na trhu prohlížečů pro počítače a také jejich sourozence pro mobilní zařízení.

Na desktopu: vede Chrome, staré Explorery vymřely

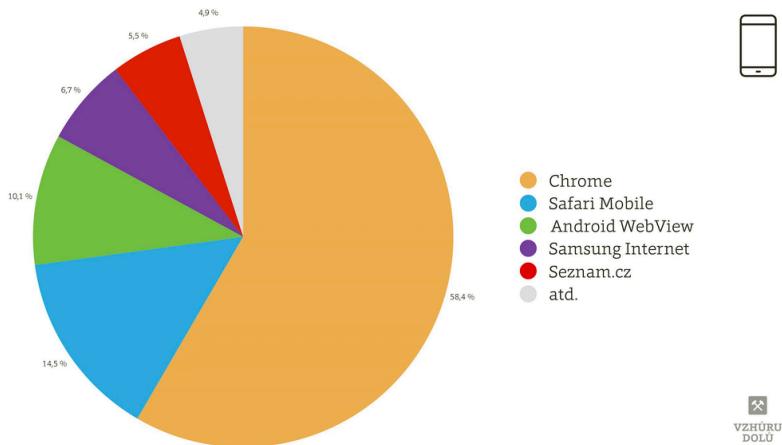
Když vezmeme trh počítačů, mezi prohlížeči už jednoznačně vede Chrome, následovaný Firefoxem a Explorerem 11. Opera, Safari nebo Edge to dotáhly na podíly jen v rádu jednotek procent. Explorery verzí 10 a starší už prakticky vymřely. A hlavně: prohlížečů je pořád docela hodně.



Obrázek: Podíl desktopových prohlížečů. Zdroj: Google Analytics pro weby cestovky Rekrea v červnu 2018

Mobilní zařízení: Chrome a Safari

Na mobilech musíme počítat s Google Chrome a Safari. Menší podíl mají další odvozeniny jádra Chromium, jako je Samsung Internet nebo Android Brower. U Samsungu ale čísla jen odhaduji, protože spolehlivé měření zatím nemáme. Ještě menší podíl pak vykazují Internet Explorery či Edge na platformách Windows Phone nebo Opera Mobile.



Obrázek: Podíl mobilních prohlížečů. Zdroj: Google Analytics pro weby cestovky Rekrea v červnu 2018

Moc se neví, že na iOS se jádro Safari používá i v Google Chrome a všech dalších prohlížečích. Raději to zmíním už tady. Vykreslování tedy obvykle není potřeba na iOS kontrolovat v Chrome a zároveň Safari.

Ještě méně se ví o nástupu nových malých prohlížečů. Znáte například Samsung Brower? V Česku také nemůžete minout stále významnější prohlížeč od Seznamu.

Sledujte statistiky podílů prohlížečů ve vaší cílové skupině. Na Vzhůru dolů pak najdete aktuální čísla. vrdl.cz/p/prohlizece

Argumentace Googlem

Občas se stává, že statistiky jako přesvědčovací technika nezabírají. Je tu ale ještě jeden způsob, jak váhavé přesvědčit o nutnosti změn při tvorbě webů. Říkám mu „argumentace Googlem“.

Divím se totiž, že ještě nemáme stovky variant následujícího suchého webdesignérského vtipu. Jeho scénář by vypadal asi takto:

V místnosti sedí tvůrci webů a diskutují o nějakém tématu. Časem se rozdělí na dva nesmiřitelné tábory a zdá se, že domů odejdou bez shody, utvrzení ve svých názorech.

Ke konci sezení se otevřou dveře a vstoupí Google. Všichni ztichnou a napjatě jej sledují. Google vlídně pozdraví a pomalu a elegantně se usadí na volnou židli. Pak začne mluvit. Tichým a vyrovnaným hlasem řekne, co si myslí. Krátce nato se weboví tvůrci zvednou a potichu odejdou. Teď už mají jednotný názor. Ten, který jim sdělil Google. Všichni jdou spolu na pivo, jen Google zamíří domů přemýšlet nad dalšími tématy. Pro mě vtipná scéna, pro někoho smutný fakt.

Odkazování na doporučení Google funguje. Ta firma je totiž stále nejmocnějším dodavatelem návštěvníků na naše weby a diskutovat s ní moc smysl nemá.

Podobné je to s tématy, která budeme rozebírat v této knize. Google byl jednou z prvních velkých firem, které přesně pochopily význam mobilních zařízení pro web. Jeho doporučení jsou v zásadě plně v souladu s tím, co vám na následujících stránkách budu tvrdit.

Podívejme se na některé jeho kroky a doporučení z poslední doby:

- Radí stavět web jako responzivní, tedy s jednou URL adresou pro všechny typy zařízení. vrdl.in/googlerwd
- Od ledna 2017 znevýhodňuje weby s obtěžujícími překryvnými vrstvami, které jsou špatně použitelné hlavně na mobilech. vrdl.in/googlepopup
- V průběhu let 2017 a 2018 začal posuzovat některé weby hlavně podle toho, jak vypadají na mobilech. vrdl.cz/b/73-google-mobile-first

Mohli bychom pokračovat dále. Google chápe web jako *zejména* mobilní, a je tedy poměrně oddaným fanouškem filozofie návrhu nejprve pro mobily (Mobile First), o které budu psát v druhé polovině knihy.

Co se týká doporučení Google, je to přesně dle slov klasika Járy Cimrmana: „Můžeme s tím nesouhlasit, můžeme proti tomu protestovat, ale je to asi tak jediné, co s tím můžeme dělat.“

Osobní poznámka na závěr. Argumenty Googlem ve své praxi používám velmi nerad a také v knížce je budu používat jen jako doplňující. Dávám přednost přímým odkazům na statistiky a průzkumy, které ostatně i lidé z Googlu pro svá rozhodnutí a doporučení používají.

Závěr této části trochu odlehčím a zkusím si z křišťálové koule zavěštit, kam se prostředí responzivního webu posune v příštích letech.

Zařízení z budoucnosti: co je po mobilech?

K internetu jsou prý už připojené i stromy v Amazonii, takže

lednička s webovým prohlížečem by nás překvapovat neměla. Je to ale použitelné? Máme s tím jako autoři webů počítat?

Na jakých zařízeních budou lidé používat prohlížeče a zobrazovat naše weby za pár let? Cítím krásnou příležitost se historicky ztrapnit a jdu vývoj odhadnout jen podle svého nejlepšího vědomí.

Budem si na nich prohlížet weby?

Televize: NE

I vaše chytrá domácí televize nejspíš nějaký ten prohlížeč obsahuje. Moc je ale lidé nepoužívají. Zakopaný pes vězí v uživatelském ovládání. Budoucnost v případě televizí vidím v gestech a hlasovém rozhraní. Interakce s běžnými webovými stránkami na televizi budou do té doby nepohodlné. To ale neznamená, že nevznikají nativní TV aplikace vytvářené pomocí webových technologií. Ostatně i HbbTV aplikace jsou vyrobené webovými technologiemi.
vrdl.in/hbbtv

Ledničky: NE

Lednička bývá v domácnostech jedním z center dění, takže si tam nějaký dotykový displej představit umím. Jenže jaká je jeho přidaná hodnota oproti přenosnému telefonu nebo tabletu? Problém je také s jeho svislou polohou. Ovládání nebude příjemné. Specializované aplikace mohou obstát, ale konzumace a interakce s běžnými weby uživatele bavit nebude. Ne, prohlížeče se v ledničkách, tak jak je známe třeba z tabletů, myslím neujmou. Ale lednička s prohlížečem existuje, to ne že ne. Samsung Family Hub Refrigerator:
vrdl.in/udp4y.

Kapesní herní konzole: ANO

Myslím ty malé potvůrky do ruky. Jsou to jen převlečené chytré telefony, prohlížeče obsahují a není důvod na nich weby nepoužívat.

Auta: ANO

S rozmachem (polo)automatického řízení poroste význam velkých dotykových obrazovek. Nejspíš v těch místech, kde teď v autě máte výstupy z ventilátorů. Číst si Blesk.cz v dopravní zácpě? Já to dělat nebudu, ale dává mi to dokonalý smysl. Jen doufejme, že ta auta budou fakt chytré, aby zvládla řídit, protože nás od palubních displejů už nic neodtrhne. Mají to už odzkoušeno v Tesla Motors: vrdl.in/rlwn0.

Chytré hodinky: Trochu ANO

„Zkoušel jsem, ale myslím, že je to nepoužitelné, a masovému přijetí nevěřím.“ psal jsem zde pro první verzi knížky. Jenže autoři míní, ale uživatelé mění. V textu o webech na hodinkách od Apple uvidíte, že lidé si už dnes naše délka začínají prohlížet i na hodinkách.

Revoluce nejspíš ne. Weby budou ale ještě více responzivní

Jedině že by na trh ještě vstoupily pračky s displejem a prohlížečem. Ale ne, dělám si legraci.

Dotykové panely v autech jsou vlastně větší tablety a přenosné herní konzole zase něco jako chytré telefony. Vypadá to tedy, že velké novinky nás ze strany displejů v nejbližší době nečekají.

Responzivní weby a aplikace se spíše skrze hardware naučí lépe využívat znalosti o prostředí prohlížeče, zařízení a uživatele. Mohou zjistit stav baterie, světelné podmínky v okolí zařízení a lépe využít geolokaci a polohu zařízení v reálném světě. Hezky to rozebírá Una Kravets v článku „Rethinking Responsive Design“. vrdl.in/ozlu9

Sledujte hlasová uživatelská rozhraní

A když už se tak odvážně historicky ztrapňuji, příští velký technologický posun vidím v hlasovém uživatelském rozhraní.

Sami asi víte, že hlasový vstup se stále zlepšuje i ve stávajících zařízeních. V domácnostech nebo třeba v autech je pro hlasové ovládání naprosto ideální prostředí. Moje předškolní děti například ovládají hlasem YouTube na tabletu zcela bez potíží.

Sledujte zařízení jako třeba Google Home. Nebo si na YouTube pusťte představení Amazon Echo. youtu.be/KkOCeAtKHIC

Tam už moc příležitostí pro designéry rozhraní webových stránek nebude. Omlouvám se. Ale nebojte, mluvící rozhraní ta naše obrazovková ani zdaleka nenahradí. Tuhle knížku ještě neodkládejte, to vážně ne. Weby budou dále sloužit jako zdroj informací. I pro tyhle mluvící potvory.

Ted' se už z této futurologické odbočky dostáváme zpět k responzivním webům.

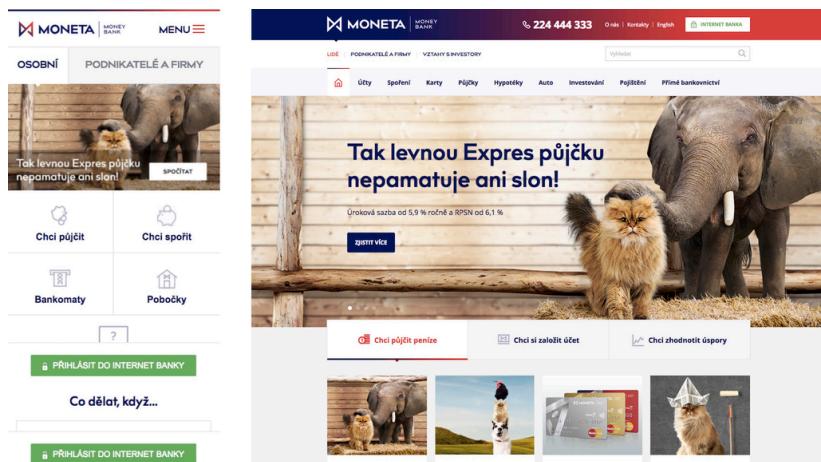
Weby responzivní, adaptivní, mobilní

Vlastně dnes máme jen dvě možnosti, jak udělat web pro všechna zařízení: adaptivní web, nebo zvláštní verzi webu pro mobilní zařízení. Kam zmizely weby responzivní? Víte, většina z nich dnes

patří do kategorie webů *adaptivních*, ale budeme jim dále říkat responzivní. Asi bych vám měl něco vysvětlit, že? Čtěte dále.

Mobilní web

Občas se říká „m tečka web“. Ano, jsou to ty weby, které pro přístupy z počítačů používají adresu jako www.example.cz a pro přístupy z mobilů něco jako m.example.cz. Z pohledu návštěvníka i provozovatele jsou to vlastně dva weby, které servírují stejný nebo případně i upravený obsah.

The image shows two side-by-side screenshots of the Moneta website. On the left is the mobile version, featuring a large image of an elephant and a fluffy cat, with text in the center reading "Tak levnou Express půjčku nepamatuje ani slon!" (Such a cheap Express loan no one even remembers!). Below this are several call-to-action buttons: "Chci půjčit", "Chci spořit", "Bankomaty", "Pobočky", and "Co dělat, když...". At the bottom are two more "PRIHLÁSIT DO INTERNET BANKY" buttons. On the right is the desktop version of the website, which has a similar layout but includes a navigation bar with links like "Účty", "Spaření", "Karty", "Půjčky", etc., and a search bar. Both versions feature the same background image of the elephant and cat.

Příklad mobilního webu: m.moneta.cz a www.moneta.cz

Výhoda: Mobilní web může být relativně rychle hotový

Pokud máte složitý desktopový web a teprve uvažujete o řešení pro mobilní uživatele, může to být na rozdíl od velkého responzivního redesignu časově i finančně o dost méně náročné.

Výhoda: Serverová detekce se může hodit

Pro někoho může být výhodou serverová detekce zařízení.

Výjimečně se může stát, že chcete určité skupině uživatelů poslat výrazně jinou verzi designu nebo obsahu. Dříve se tak řešil problém existence velmi starých chytrých telefonů, které dnešní frontend technologie nezvládaly – třeba Nokií se Symbianem. Drobné detekce na serveru dělá dnes i řada responzivních webů. U nich je ale drtivá většina frontend kódu pro všechny návštěvníky totožná.

Nevýhoda: Dvě URL adresy a nutnost udržovat vazby mezi nimi

Je potřeba řešit situace přesměrování mezi jednou a druhou verzí. Vyhledávače, jak jim říct o existenci dvou webů? Jako správce webu dvě verze nechcete, věřte mi. Pokud už ale v takové situaci jste, udělejte vazbu mezi oběma verzemi dobře. Nečekali byste to, ale odkážu vás na doporučení Google: vrdl.in/os3y1

Nevýhoda: Co se středně velkými displeji?

Co s tablety? U speciálních mobilních webů často vznikne problém – mobilní web bude na tablety moc jednoduchý a desktopový moc složitý.

A co dále? Občas se argumentuje tím, že speciální mobilní web se může načítat rychleji. Hlavně pokud je desktopová verze datově oplácená. V žádném případě to ale neznamená, že nelze udělat velmi rychlý responzivní web. Naopak! V páté kapitole o tom něco málo píšu.

Jak vidíte, tohle řešení má řadu nevýhod a považuji ho za dlouhodobě neudržitelné.

V krátkodobém horizontu může být „m tečka“ web docela záchrana, ale v dlouhodobém se údržba dvou webů prodraží. Ukážu to na

příkladu.

Příběh Scuk.cz: Když je něco vymyšlené pro desktop, těžko z toho uděláte responzivní web

Před lety jsem jako kodér spolupracoval na projektu Scuk.cz známého foodbloggera, pana Cuketky. scuk.cz

Dnes už je Scuk responzivní, ale v roce 2010 ještě mobily nebyly v Česku tak důležité. Proto jsme, jako všichni v té době, vyrobili jen desktopovou verzi. Pár let po spuštění ovšem v naší cílové skupině podíl mobilních uživatelů vzrostl. Responzivní web jsem tehdy udělat uměl, takže jsme se bavili o jakémsi „zresponzivnění“ desktopového webu. Nakonec jsme to ale zavrhlí.

Desktopový Scuk.cz byl totiž *vymyšlený* pro desktop. Když je něco vymyšlené pro desktop, těžko z toho uděláte responzivní web. Musíte to vymyslet znovu.

Scuk byl postavený na mapách Google, takže rozhraní bylo poměrně složité a řešení z pohledu tehdejších frontend technologií relativně těžkopádné. Na mobilech, kde potřebujete rychlé načtení a na pohodlnost ovládání má uživatel vyšší nároky, by bylo ještě těžkopádnější.

The image shows a side-by-side comparison of the Scuk.cz website. On the left is a smartphone displaying the mobile version, which has a dark theme with a red header and a sidebar menu for categories like 'Vše', 'Restaurace', and 'Kavárny'. On the right is the desktop version, featuring a light orange header with the 'Scuk.' logo and a search bar. Below the header is a map of Central Europe with several numbered locations (10, 6, 56, 374) indicating the presence of Scuk. The desktop version also includes a sidebar with a 'MALL.CZ' button.

Původní verze Scuk.cz a zpětně dodělaná mobilní verze

Nakonec jsme se rozhodli vytvořit ještě jeden web: právě „m tečka“ verzi pro mobily. Už v době příprav jsme ale věděli, že bude dočasná. Po několika letech se oba staré weby zahodily a vznikl nový, jednotný responzivní web. Už beze mě, protože naše domácnost se v té době rozrůstala o děti. Ale povedl se, že ano?



Poslední, již plně responzivní generace Scuk.cz

A teď už konečně obracím kormidlo směrem k našim responzivním webům. Hurá!

Responzivní web

Jde o web, který reaguje *responzivně*, tedy rychle a jednoznačně, na změny prostředí. Technicky je to web, kterému pro to, aby se přizpůsobil všem zařízením, stačí jedna aplikace. Dle původní definice si vystačí s pružným layoutem a se změnami pro konkrétní skupiny rozlišení obrazovky realizovanými pomocí Media Queries. O tom si ale ještě budeme povídат v šesté kapitole.

Responzivní web vlastně odstraňuje nevýhody speciálního mobilního webu, které jsem jmenoval výše:

- Není potřeba spravovat více verzí webové aplikace.
- Všechna zařízení sdílejí stejné URL adresy, takže není potřeba řešit vztahy mezi nimi.

Jen pozor, bez nevýhod to není: Responzivní redesign vyžaduje nezanedbatelné množství energie pro všechny lidi, kteří se účastní procesu vývoje a správy webu. Často je nutná zcela zásadní změna v pracovních postupech nejen designérů a vývojářů, ale prakticky všech, kdo se na webu, jeho obsahu a marketingu podílejí.

Responzivní redesign VašeČočky.cz: rok a půl práce

Dalšího klienta, firmu Maternia, se mi hned v počátcích spolupráce (někdy v roce 2015) povedlo přesvědčit, aby myšlenku na mobilní web opustil. Rok a půl jsme pak pracovali na pořádném responzivním redesignu všech jeho e-shopů. Nyní ale má dlouhodobě udržitelné řešení, které jedním kódem obsluhuje všechna relevantní zařízení.

The screenshot displays the responsive website VašeČočky.cz. At the top, there's a header with a logo, a search bar, and a phone number (800 114 1118). Below the header, the desktop version shows a navigation menu with links like 'Kontaktní čočky', 'Rostoky', 'Slnečné brýle', 'Příslušenství', and 'Borskové stříšky'. A sidebar on the left offers a 'Jednodenní čočky' section with a contact form and a photo of Eva Várgová. The main content area shows a grid of contact lens products with their names and prices: BioTrue ONEday (675 Kč skladem), Biotrue ONEday (30 čoček) (675 Kč skladem), 1 Day Acuvue Moist (30 čoček) (499 Kč skladem), Focus Dailies Progressives (30 čoček) (599 Kč skladem), DAILIES AquaComfort Plus (30 čoček) (429 Kč skladem), 1 Day Acuvue Triflex (30 čoček) (599 Kč skladem), Softlens Daily Disposable (30 čoček) (449 Kč skladem), and DAILIES Total 1 (30 čoček) (499 Kč skladem). The mobile version on the right shows a similar layout but with a more compact grid of products.

VašeČočky.cz jako příklad responzivního webu

Responzivní web je to, čemu dávám vždy přednost a čemu (to byste nevěřili!) se budeme také v dalších textech nejvíc věnovat.

V boji za úměrnou délku textů jsem vynechal menší argumenty,

které ve prospěch responzivních webů používám. Pokud svým šéfům, kolegům nebo klientům ještě stále potřebujete předkládat důvody pro přechod na responzivní web, poříďte si výbornou argumentační příručku „Going Responsive“ od Karen McGrane. Ta je důvodům pro responzivní web a ošemetnostem přechodu na něj věnovaná celá. vrdl.in/goingrwd

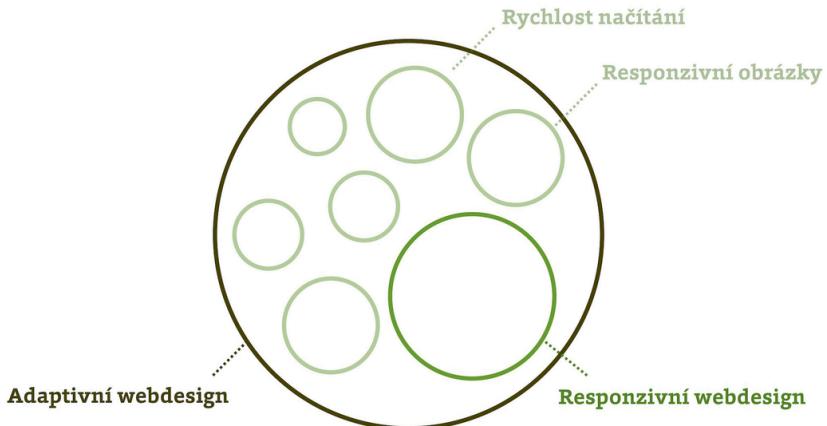
Tady bychom mohli skončit, ale dlužím vám ještě poznámku k přídavnému jménu *adaptivní*.

Adaptivní web

Technické prostředky původní definice responzivního designu dnes už k výrobě dobrého webu nestačí. Dnes už nestačí přizpůsobit rozvržení stránky a mediální obsah pružnému rozvržení.

Máme zde také řadu situací, kdy je potřeba měnit věci už na serveru: Často například v kontextu řešení rychlosti načítání. Můžeme také potřebovat poslat každému zařízení jiný obrázek. Nebo zajistit, aby telefonní číslo bylo aktivní jen na mobilech. Zájemci najdou řešení toho posledního v článku „Průvodce odkazy na telefonní čísla“ na Vzhůru dolů. vrdl.cz/b/57-href-tel

Téhle širší kategorie řešení můžeme říkat adaptivní webdesign.



Do adaptivního webdesignu patří kromě technik responzivního designu také například řešení rychlosti načítání nebo technologie responzivních obrázků

Původní definice responzivního designu už prostě dnešním webům nestačí.

Responzívni webdesign = adaptivní webdesign = webdesign

Mnohé dnešní weby tedy nejsou responzivní, ale adaptivní. Jenže lidé je jako responzivní pořád označují. Proč? Protože si na to zvykli a my už s tím nic nenaděláme. Pro potřeby zdejších textů tedy budeme dál mluvit o responzivním designu a k jeho implementaci užívat technik designu adaptivního.

Ono na tom vlastně v dlouhodobé perspektivě nezáleží. Fáze webdesignu, ve které bylo potřeba rozlišovat responzivní a „normální“, snad už brzy skončí. Všechny weby budou responzivní, takže to přídavné jméno můžeme přestat používat.

Slovem *responzivní* se totiž dnes už hlavně vyhraňujeme vůči předchozímu způsobu tvorby webů. *Responzivní* je tedy více pojmenování pro aktuální způsob tvorby. Pro aktuální etapu webdesignu. A tahle etapa skončí.

Za pár let už slova „responzivní“ nebo „adaptivní“ potřebovat nebude. Všechny weby budou responzivní, jen budované prostředky adaptivního designu. Nebude responzivní webdesign, zůstane zase jen *webdesign*.

Mimochodem, adaptivní webdesign popsal a příklady moc hezky doplnil Aaron Gustafson v knížce, jejíž název byste neuhádli.

„Adaptive Web Design“. adaptivewebdesign.info

Pojmologii už ale uzavřeme. V další kapitole budeme pracovat na příkladu konkrétního (responzivního) webu, takže nás čeká praxe. Začít ale musíme ze široka, protože rozumný návrh (responzivního) uživatelského rozhraní vzniká až na základě informací získaných z analytické fáze projektu. Víte vy co? Pojdme si raději povídět něco o tom, jak se dneska tvoří weby. Slibuji, že to bude stručné.

Zapamatujte si

- Přístupy z mobilních zařízení měly v roce 2016 na webech v ČR v průměru pětinový podíl. Od té doby se leccos změnilo, ale růst téhle statistiky ne.
- Nesmíme zapomenout na velké displeje. Jejich podíl je podobným mobilním zařízením.
- Roste podíl větších dotykových zařízení, hybridních notebooků a velkých dotykových tabletů.
- Klesá prodej dnešních tabletů a nedotykových počítačů.
- Hardwarové rozlišení nás, webaře, nezajímá. Navrhujeme a kódujeme pro přepočítané „CSS rozlišení“ obrazovky.

- Prohlížečů je na trhu hodně a weby je potřeba testovat ve většině z nich.
- Nevěřte cizím statistikám. Sledujte vlastní Google Analytics.
- Google doporučuje responzivní design a v mobilech vidí budoucnost.
- Kromě kapesních herních konzolí a displejů v autech teď jiná nová zařízení pro zobrazování webů nechystají.
- Všechny responzivní weby jsou adaptivní. Ale říkáme jim responzivní, protože se to ujalo.
- „M tečka“ weby mohou být fajn dočasné řešení. Dlouhodobě udržitelné jsou ale jen responzivní řešení.

Kapitola 2: Úvod do tvorby webu

Vy zkušenější víte, že se dnešní webdesign nedá pojmenovat jediným člověkem. Já se třeba ze židle frontend designéra jakžtakž vyznám v návrhu uživatelského rozhraní, jazycích HTML a CSS a souvisejících témaech. Kolegům a kolegyním uživatelským výzkumníkům, grafikům, copywriterům, marketérům, vývojářům a serverovým administrátorem vidím pod ruce jen málo.

Napsat knížku o celém webdesignu vlastně už ani není možné. I tady se budeme zabývat jen úzkou výsečí: designem a implementací webového uživatelského rozhraní. Vynecháme všechny rybníky s tenkým ledem, ve kterých bych se jistojistě utopil.

Webdesign jako obor si tedy pro potřeby knížky můžeme zjednodušit.

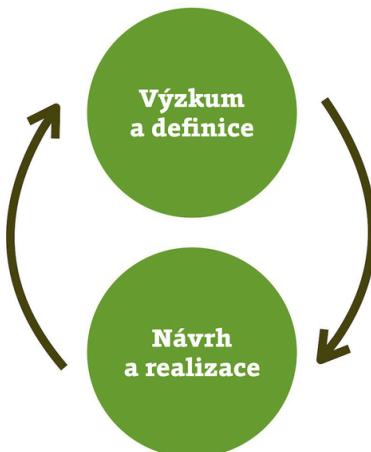
1. Do pěti minut pak dostaneme celý proces tvorby webu.
2. Analytickou práci a průzkum předcházející tvorbě webu si shrneme do User Centered Design Canvasu
3. Design Canvas aplikujeme na příkladu.
4. V knihách se to má hemžit pohádkovými bytostmi, takže obsah nám vyčaruje Imaginární copywriterka.

Základy procesu tvorby webu

V dalších kapitolách budeme navrhovat a psát kód jednoduché responzivní webové stránky. Při návrhu, ale i implementaci budeme

dělat rozhodnutí, která by v praxi vycházela z předchozích fází procesu návrhu webu.

Celý proces tvorby webu zjednodušíme do dvou kroků. Ony to vlastně kroky nejsou. Je to nekonečný koloběh, jehož fáze se v praxi velmi prolínají.



Zjednodušený koloběh tvorby webu. V knize se budeme zabývat hlavně fází řemeslnou: návrhem a realizací

Průzkum a definice

Zjišťujeme potřeby a cíle klienta a jeho zákazníků. Analyzujeme jeho konkurenci a pozici na trhu. Díváme se na jeho cílové skupiny.

V této fázi pracují hlavně designéři, produktoví manažeři, datoví, marketingoví a brandingoví analytici, copywriteři a také samozřejmě lidé na straně klienta.

Návrh a realizace

Na základě předchozí fáze navrhнемe drátěné modely rozhraní, prototypy a grafické zpracování. Pracují tady weboví designéři, grafici, frontend kodéři, programátoři. Prostě řemeslníci. A, ano, hlavně této fázi se v textech věnuji.

Ve chvíli, kdy odtud vyjde něco hmatatelného, přecházíme opět do fáze průzkumu a definice. Testujeme výsledky s uživateli, analyzujeme podle dat z Google Analytics nebo prostě podrobujeme vlastnímu kritickému pohledu. Vyhodnocujeme. Začali jsme tím další kolečko průzkumu, definice a následného návrhu s realizací.

Takhle stručně mně to pro potřeby knížky stačí. Výhrady zkušených webařů, nespokojených s mou zjednodušující interpretací, tady asi slyšíte až k vám domů. My ale můžeme pokračovat dál, protože ty zvědavé pošlu k dalším zdrojům.

Další zdroje o procesu návrhu webu

- Jiří Sekera a Petr Kosnar detailně popisují proces návrhu produktů, aplikací a služeb ve svém kurzu „Human-Centred Design: Design zaměřený na člověka“. Je dostupný zdarma na Seduo.cz a skvěle použitelný i pro weby. seduo.cz/human-centred-design
- Pokud v procesu tvorby webu plavete vy nebo vaši klienti, velmi doporučuji stručnou, ale atraktivně pojatou knihu Jana Řezáče „Web ostrý jako břitva“. houseofrezac.com/kniha
- Raději něco hutnějšího? Poctivá kniha o designérském procesu „Dobrý designér to všechno ví“ je dobrá volba i pro vás zkušenější.

pixy.cz/kniha-dobrydesigner

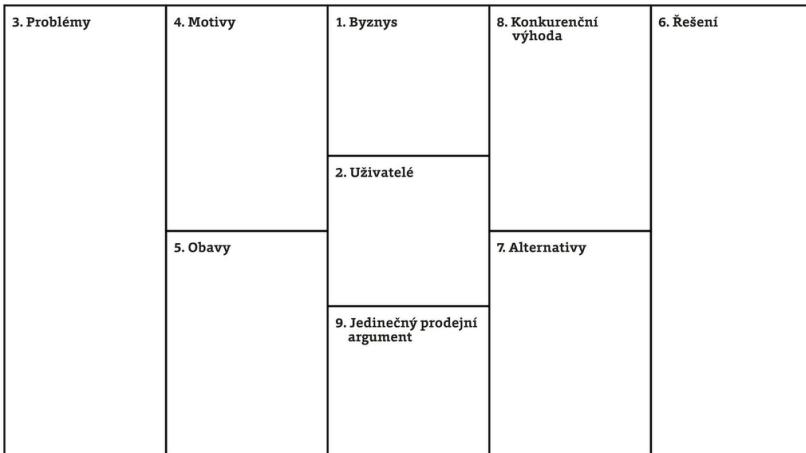
Fáze průzkumu a definice nějaký čas zabere, ale není předmětem této knihy. Budeme předpokládat, že jsme si jí prošli. Pro potřeby projektu, jejž za chvíli otevřeme, se nám pak bude hodit nástroj, který průzkum umí shrnout na jeden list papíru.

User Centered Design Canvas

Správný designér hledá řešení, jež pomáhá uživatelům, ale také cílům člověka, firmy nebo instituce, pro kterou web navrhuje.

Výstupů výzkumné fáze projektu bývá u větších projektů více: persony, uživatelské scénáře, mapy webu, definice klíčových ukazatelů výkonu (KPI) a mnoho dalších. User Centered Design Canvas je ale nástroj, který to celé dokáže shrnout na jedné stránce.

UCD Canvas může u větších projektů posloužit jako shrnutí, provázející projektem všechny členy týmu. U těch menších se obejdeme bez dalších výstupů.



User Centered Design Canvas. ucdc.therectangles.com

User Centered Design Canvas má čtyři hlavní oblasti:

1. Typy uživatelů

Všechny kategorie možných uživatelů webu nebo produktu.

2. Vlastnosti uživatelů

Problémy, motivy a obavy uživatelů.

3. Vlastnosti byznysu

Konkurenční výhody, alternativy k produktu a řešení problémů uživatelů, která nabídneme.

4. Jedinečný prodejní argument

Důvod, proč by uživatelé měli koupit zrovna váš produkt, a ne ten konkurenční.

Možná jste si všimli, že mluvíme o variaci na známý Lean Canvas – zjednodušený podnikatelský plán na jedné straně papíru. Ano, je to tak. Ten už rámec obsahu knihy překračuje skutečně hodně, takže nabídnou alespoň odkaz. leancanvas.cz

Teď, když umíme jednoduché výstupy z výzkumné fáze shrnout do jedné stránky, pojďme se zaměřit na fázi návrhu a realizace uživatelského rozhraní.

Proces návrhu uživatelského rozhraní

V knize a na příkladu ukazují proces tvorby webů, o kterém by mnozí profesionální kolegové řekli, že je „exotický“. Nevyužívám nástroje pro tvorbu předběžných návrhů webů a skoro neotevím Photoshop a jiné kreslicí programy. Velkou část práce dělám přímo v prohlížeči. Pojďme si říct, jak přesně budeme v knize postupovat a proč vlastně takto specificky.

V procesu návrhu a implementace uživatelského rozhraní se budeme pohybovat v kolečku „Návrh a realizace“ ze zjednodušeného grafu procesu tvorby webu, který znáte z předchozí podkapitoly o základech procesu tvorby webu.

Navazujeme tedy na kolečko „Průzkumu a definice“. Některé výstupy z této fáze jsme si shrnuli do User Centered Design Canvasu. Dále bychom z ní dostali například přípravu obsahu pro web, návrh jeho informační architektury, uživatelské scénáře, analýzu klíčových slov a další materiály.

Jak obvykle vypadá návrh uživatelského rozhraní?

Nejběžnější proces tvorby rozhraní v dnešních webařských týmech vypadá asi takto:

1. Načrtnou se wireframy
2. Nakreslí se grafika
3. Nakóduje se to

Každou funkci obvykle zastává jiný člověk. Často se jednotliví aktéři ani nepotkají (a na dálku na sebe navzájem nadávají). Konvenčnímu procesu kromě nedostatku mezioborové spolupráce vytýkám také neefektivitu. Například připomínky ke grafice a její globální změny se v dnešních grafických editorech zapracovávají pomalu a složitě.

Z pohledu frontendisty mně na konvenčním procesu také vadí, že se takto vznikající weby do svého přirozeného prostředí – prohlížeče – dostanou příliš pozdě. U designérů a grafiků, ale často už i v krocích před nimi padají rozhodnutí, která nejsou otestovaná prohlížečem a přizpůsobená technologickým možnostem. Kodér pak už jen nadává – však to znáte.

Podívejme se teď na můj postup. Je specifický a v běžném webovém studiu se asi jen tak neujme. Ale není dobrého nebo špatného postupu. Důležitý je vždy výsledek: Jak funguje výsledný produkt, kolik času vám proces zabral a zda pro vás bylo příjemné jím procházet. Následující postup práce je navíc dost jednoduchý na to, aby jej mohli převzít začátečníci. A profesionály třeba v lecčems inspiruje.

Tři a tři kroky: proces návrhu uživatelského rozhraní

Vytvoříme dokumentovou základnu, pak procházíme jednotlivé komponenty a pomocí skicování, prototypování či jiných vhodných nástrojů je navrhujeme. Potom nebo mezitím můžeme vymyslet rozvržení celé stránky.

Krok 1: Dokumentová základna

Obsah vložíme do prohlížeče a vyznačíme jej sémantickými HTML značkami. Dostaneme základní, dokumentovou vrstvu stavby webu.

Dále připravíme vizuální design společný pro celý web. Vybereme písma, barvy, stupně velikosti, styl grafiky ikon, ale i dostatečně robustní technickou základnu. Podrobně se dokumentu věnuji ve třetí kapitole.

Krok 2: Komponenty

Komponentu zde berte jako menší nebo větší součást rozhraní webu. Od tlačítka a formulářového pole až po komplexnější skupiny, jako je záložková navigace nebo patička webu. Proces tady ještě rozdělím na tři kroky. Obvykle u mě probíhá takto:

1. *Skicování* Rychlý brainstorming nad možnostmi řešení s tužkou, fixy a papírem.
2. *Prototypování*
Převedení předvybraného řešení do formy testovatelné v prohlížeči. Nad HTML prototypy obvykle dále iteruji a vybrušuji je až do finální podoby.
3. *Aplikace grafického stylu*
Ten už máme připravený od dokumentové základny.

Asi z toho vidíte, že proces návrhu a implementace komponent uživatelského rozhraní zabere nevíce času. O komponentách píšu v osmé kapitole.

Krok 3: Rozvržení webu

Už během přípravy dokumentové základny a komponent je vhodné vymýšlet systém pro layout webu. Jeho implementací se zabývám v deváté kapitole.

Proč to v knížce ukazuje právě takto?

Hlavně proto, že se na tom dobře ukazují principy responzivního designu. A že to nevyžaduje složité nástroje a je to vhodné pro

začátečníky. Ale řeknu vám i další důvody.

Rozhraní se brzy dostane do prohlížeče

Výstupy z Photoshopu mohou být dokonalé. Dokonalé a iluzorní. Jak říká klasik: „Péesdék“ vypadají přesně tak, jak web *nikdy* vypadat nebude.

Není to jen proto, že prohlížeče vykreslují jinak. Ve statickém kreslicím nástroji neodzkoušíte responzivitu, interakce, animace a další rozměry webového média. Proto může být velmi obohacující dostat web do prohlížeče už v raných fázích projektu. A už z něj neodcházet.

Komponenty a znovupoužitelnost

To, že jsme ve webdesignu začali pracovat se stránkami jako nejmenšími prvky, ze kterých skládáme weby, je omyl, který nás bude ještě dlouho mrzet. Stránky jsou prostě pořád příliš komplexní entity na to, aby je bylo možné navrhnout i implementovat dostatečně kvalitně a efektivně.

Neměli bychom navrhovat stránky, ale systémy komponent. Znovu použitelných komponent. Ze „stránkových“ výstupů grafických programů není možné systém vyčist, i kdyby tam byl. Obvykle tam ale ani žádný není. Systém znova použitelných komponent pro každý větší web by ovšem šetřil čas i peníze všem.

Krátké iterace, ne vodopád

Spolupráce designéra a kodéra je obvykle velmi neefektivní. Designér dlouho pracuje na návrzích obsahujících unikátní stránky webu. Provádění změn ve Photoshopu nebo Sketchi ke všemu není nijak příjemné. Kodér je pak zase dlouho převádí do HTML, CSS a Javascriptu.

Je to vodopádový proces, známý z klasických výrobních linek, kde

bylo potřeba v každé fázi produkt dokonale vybrousit, aby do té následující vplul bez újmy. Digitální médium ovšem kodérům i designérům nabízí možnost pracovat společně, na menších celcích, než jsou stránky, v kratších iteracích a rychleji. Škoda, že toho tak málo využíváme.

Nepotřebujete k tomu složité nástroje

Stačí vám prohlížeč, editor kódu, tužka, papír – a hlava. Jedna z cílových skupin knihy jsou začátečníci, ale ani profíky nechci zbytečně zatěžovat prototypovacími nebo návrhářskými nástroji, které třeba nemusejí používat.

Kde se to hodí a kde ne?

Uvedené workflow je ovšem náročné na intenzitu mezioborové spolupráce a čas. Myslím, že se hodí pro všechny, kteří zvládnou kódovat a alespoň trochu přitom myslet designérsky. Použil bych jej ve všech podoborech webdesignu, které nejsou závislé na „prodeji PSD“. Proces nebo jeho prvky je možné využít při tvorbě webových aplikací nebo během dlouhodobé práce na produktu.

U klientské, agenturní práce si jej zatím neumím představit. U většiny zde zapadajících projektů nás čeká ohromný kus práce: Vysvětlit klientům, že to, za co platí, není pár krásných obrázků z Photoshopu. Že by měli chtít platit za systémy komponent. A že weby jsou doma v prohlížečích.

Zdroje

Pokud by vás téma alternativních pracovních postupů zaujalo, věnujte svou pozornost následujícím odkazům.

- *Přednáška Stephena Haye „Responsive Design Workflow“*

Stephen Hay je pro mě asi nejzajímavější zdroj procesů pro tvorbu responzivních webů. Vydal i stejnojmennou knížku. youtu.be/6e3m9qRj67o

- *Přednáška Ryana Singera „Designing from start to finish“*

Jako mnohé z vás, i mě velmi inspirovaly procesy v Basecamp.com (dříve 37signals). vrdl.in/singer

- *Můj text „Design webů v prohlížeči“*

Přepis přednášky z WebExpo 2015. Obsahuje zde uvedený postup, aplikovaný při redesignu e-shopu VašeČočky.cz. vrdl.cz/b/38-design-v-prohlizeci

- *Kniha Brada Frosta „Atomic Design“*

V textu, který právě čtete, nezvládnut jít takhle do hloubky, ale systémy designu mají ve webdesignu velkou budoucnost, vězte mi. atomicdesign.bradfrost.com

Tím se dostáváme k ukázkovému příkladu, kterým se v knize budeme zabývat. Představíme si jej tím, že si pro něj sestavíme UCD Canvas.

Příklad: E-shop s vybavením pro děti z lesních školek

Jaký zvolit příklad pro knihu? Nebudete mi věřit, ale tohle byla jedna z těžších částí práce na ní.

Neměl by to být až moc jednoduchý prezentační web. Ne všichni ale dělají komplexní webové aplikace. Někde mezi statickým webem a aplikací leží e-shopy, tedy prezentační weby s výraznými aplikačními prvky v podobě objednávkového procesu. OK, bude to elektronický obchod.

Jaký e-shop to ale, do prkýnka, bude? V zemi, kde jich je pomalu více

než nakupujících? Ono se ale často stačí podívat přes rameno své milé ženě a zamyslet se, jaké problémy při online nakupování řešíme. Řešíme problémy s dětmi.

Oba naši chlapečkové chodili nebo chodí do skvělé kunratické lesní školky. Jsou pořád venku a nadmíru jim to prospívá. No a víte, jak se to říká: Špatné počasí neexistuje, je to vždy jen problém špatného oblečení.

Proto žena často nakupuje pohorky, nepromokavé kalhoty a bundy s vodním sloupcem, o kterém se mně nikdy ani nesnilo. Je z ní úplná expertka na miniaturní outdoorové vybavení. Jenže získat ty vědomosti nebyla úplná sranda. Navíc nakupuje na různých e-shopech a moc by se jí líbilo, kdyby vše vyřešil jen jeden. Právě teď jsme doma v obýváku jeden fiktivní založili. Jmenuje se *ForestKid.cz* a v knížce mu uděláme prima web, uvidíte.

Pojďme na první krok. Vezměme prázdnou tabulku našeho UCD canvasu a načrtněme si do ní tržního prostředí našeho e-shopu.

UCD Canvas pro fiktivní e-shop ForestKid.cz

1. Byznys

ForestKid.cz, e-shop zaměřený na potřeby pro děti z lesních školek.

2. Uživatelé

V prvé řadě jsou to rodiče dětí z lesních školek, zejména maminky. Následně také aktivní rodiče, kteří se školními dětmi chodí po horách, jezdí na kole nebo třeba na vodu. Méně pak příbuzní těchto dětí, kteří zde hledají vhodné dárky. Obvykle to budou lidé z větších měst.

3. Problémy, které uživatelé řeší

Výběr kvalitního outdoorového vybavení pro děti není velký. Obvykle je potřeba věci shánět na různých specializovaných e-shopech, často přímo u výrobců. To zdržuje, a navíc to prodražuje nákup kvůli poštovnému. Na internetu je také velmi málo obsahu, který by nakupujícím pomohl s rozhodováním, takže většina doporučení je z „offline světa“ výměnou informací mezi rodiči.

4. Motivy uživatelů

Ušetření na poštovném, získání informací, ušetření času nákupem na jednom místě.

5. Obavy uživatelů

Není vybavení příliš drahé? Je nový e-shop dostatečně důvěryhodný? Má smysl kupovat kvalitní věci? Jaký je ekologický dopad výroby a distribuce zmíněného vybavení?

6. Řešení problémů, která obchod nabízí

Veškeré vybavení se prodává na jednom místě a s jedním poštovným. Součástí e-shopu je bohatý obsah: vlastní testy vybavení, blog o novinkách z oboru, uživatelské recenze a fórum.

7. Alternativy k ForestKid

Přímá konkurence (např. OutdoorKids.cz, OutdoorBaby.cz) má širokou nabídku, ale horší komfort nákupu a chybí jim obsah sloužící k rozhodování. Specializované obchody (např. Fare.cz, VeselaNohavice.cz) nabízí jen jeden typ produktů a postrádají nezávislé hodnocení. Cílová skupina občas využívá také obecné e-shopy jako Sportisimo.cz, hlavně jejich výprodejové akce.

8. Jedinečný prodejní argument

ForestKid.cz je jediný e-shop s celým sortimentem outdoorového vybavení pro děti, který vám zároveň poradí a nakoupíte na něm

opravdu snadno.

Ted' to rádně zjednodušíme

Fajn, osm bodů design canvasu bychom měli. Co by mělo následovat? Příprava obsahu pro samotný web, návrh jeho informační architektury, příprava uživatelských scénářů, analýza klíčových slov a další kroky.

Tady ovšem zmáčkneme tlačítko pro rychlý posun filmu vpřed a tím opět proces tvorby webu trošku zjednodušíme. Představíme si, že tohle všechno máme hotovo a díky *Imaginární copywriterce* se nám vyloupl obsah jedné konkrétní stránky, kterou budeme dál webdesignérsky opečovávat.

Nejvhodnější je začít od základního obsahového prvku, což je v případě e-shopu produkt.

Příklad: Obsah pro stránku detailu produktu

Jako ukázkový produkt pro náš proces návrhu uživatelského rozhraní jsem vybral tyhle krásné kotníkové boty značky Fare, v jejichž poněkud zabahněných variantách se v Kunratickém lese prohánějí mladí Michálci.



Ukázkový produkt, se kterým budeme v příkladu fiktivního e-shopu pracovat. Zdroj: fare.cz

Jak budu v knize často zmiňovat, jednotlivé fáze procesu návrhu je výhodné co nejdříve vidět v podobě prezentovatelné v prohlížeči. U textu to nebude tak složité, viďte?

Dva nástroje: Markdown a CodePen

Pro formátování textu je možné využít nějaký vizuální HTML editor typu Dreamweaveru, přímou editaci kódu nebo – když zavřeme obě oči a zatajíme dech – třeba export z Wordu.

Já používám jednoduché značkování pomocí *Markdownu*. Práce s ním je daleko rychlejší než s HTML a převést jej do jakéhokoliv složitějšího formátu je velmi snadné. [wikipedia.org/wiki/Markdown](https://en.wikipedia.org/wiki/Markdown)

Využijeme také *CodePen*. Jednoduchý online editor, kde si stránku napíšete a otestujete bez potřeby složitějších nástrojů. Jak brzy uvidíte, je to pro mě nepostradatelný pomocník. Všechny

demonstrační příklady v knize jsou umístěné na něm. codepen.io

Obsah příkladu převedeme rovnou do kódu

Velikosti

23, 24, 25, 26, 27, 28, 29, 30

ks Přidat do košíku

Celoroční, třeková, kotníková obuv šněrovací. Vrch je z hovězího nubuku v kombinaci s textilním , vnitřek obuvi je z textilního podšívkového materiálu s výbornými sociální schopnostmi.

Díky speciální membráně, která je mezi vrchem a podšívkou, zůstává noha suchá a dobré klimatizována. Podsev je z termoplastického kaučuku s možným dezenem, vyvýšená část podeše v přední části chrání spíci při běžném zakopávání.

Přednosti je bandážování kotníků a límce, vyjmateLNá vkládací stélka a pevný opatek i špic. Obuv je opatřena náštítkem proti nosákovosti vody.

Vnitřní délka obuvi: č. 23 - 154 mm, č. 24 - 161 mm, č. 25 - 167 mm, č. 26 - 174 mm, č. 27 - 181 mm, č. 28 - 188 mm, č. 29 - 195 mm, č. 30 - 202 mm.

Vyrábí FARE, kvalitní česká obuv

Fare je český výrobce, který se zaměřuje za kvalitní dětskou obuv. Firma z Valašských Klobouků se za 25 let na trhu stala držitelem řady cen a ocenění za vynikající design, kvalitu i inovaci. Více o [značce Fare](#).

Hodnocení zákazníků ★★★★☆

95 % (průměr ze 4 hodnocení).

★★★★☆ Jirina H. Ostrava

„Po sezóně denodělno nošené obuvi v lesní školce můžeme říct, že jsme se v kvalitě nezklamali. Jsou to reálně dobré vypadající, i když jsou využívány boty. Příště bychom jen kupují bez kančíků.“

★★★★★ Jana M., Praha - Kunratice

„Fare je naše oblíbená znacka. Boty perfektně sedí a mají výdrž.“

[Další hodnocení](#)

Použil jsem CodePen a šup! Máme tady první iteraci stránky detailu produktu našeho e-shopu. cdpn.io/e/MJwGXK

The screenshot shows the CodePen interface with the following HTML code:

```
<h2>Velikosti</h2>
<ul>
  <li>23, 24, 25, 26, 27, 28, 29, 30</li>
</ul>
<input type="text" value="ks" /> Přidat do košíku
```

Below the code, there are two reviews from users Jirina H. Ostrava and Jana M., Praha - Kunratice, both giving 5 stars. At the bottom, there are two buttons: "CSS" and "JS".

Už tenhle náhled (prakticky bez CSS stylů) můžeme nějak hodnotit či testovat. Nejlépe se nám to bude dělat v malé velikosti okna, protože tam půjdou problémy obsahu vidět nejlépe.

Pracujeme tady jen s čistým, strukturovaným textem. Pro mediální obsah nebo složitější interaktivní elementy si ve stránce jen vyhradíme místo pomocí šedivých zástupných ploch. Lépe pak uvidíme, zda je délka obsahu, a tedy i stránky snadno vstřebatelná, nebo ne.

V této fázi máme neopakovatelnou příležitost odpovědět si na otázky týkající se textu. Jen tady jsme odstínění od vzhledu, tedy barev, typografie, grafiky a celkového layoutu – rozvržení stránky.

Také nás nerozptylují globální navigační a komunikační prvky webu, jako je společná hlavička a patička.

Testujeme kvalitu obsahu samotného a jeho hierarchii.

Konfrontujeme text s našimi znalostmi uživatelů a zájmů firmy, které jsme promítli do design canvasu. Nechybí nebo nepřebývá na stránce něco? Má obsah správné pořadí? Je obsah srozumitelný a přesvědčivý pro cílovou skupinu?

Pojďme tento moment nepromeškat a pořádně se na texty podívat.

Našel jsem v nich pár problémů, ale nejdřív vám rovnou ukážu druhou iteraci obsahu:

Velikosti a dostupnost

	Velikost	U vás
23	154 mm	pozití
24	161 mm	pozití
25	167 mm	za 4 dny
26	174 mm	pozití
27	181 mm	dodášte vypredané
28	188 mm	pozití
29	195 mm	pozití
30	202 mm	za 4 dny

Jak si doma změnit velikost?

1 ks Přidat do košíku

Doprava a platba: Nad 1 500 Kč zdarma. Jinak 80-150 Kč po celé ČR. Platba kartou, převodem nebo vložkou zdarma.

Více

Vyrábí FARE, kvalitní česká obuv

Fare je český výrobce, který se zaměřuje na kvalitní dětskou obuv. Firma z Valašských Klobouků se za 25 let na trhu stala držitelem řady cen a ocenění za vynikající design, kvalitu i inovaci. Více o [značce Fare](#).

Proč ForestKid?

1. Pomůžeme a poradíme
2. Všechny produkty jsou v tom, co prozíváme. Můžete se nás zeptat na [fórum](#), publikujeme [blog](#).
3. Doprava zdarma a na jednom místě
- Dnes máme skladem 2 546 položek. Outdoorové vybavení pro děti? Jděte rovnou na ForestKid.
3. Doprava zdarma nad 1 500 Kč
- Ano, je to tak. Většina našich zákazníků dopravu platit nemusí.
4. Slevy pro stálé klienty.

1

2

```

• HTML (Markdown)

# Velikosti a dostupnost

| Velikost | U vás | |
|---|---|---|
| 23 | 154 mm | pozití |
| 24 | 161 mm | pozití |
| 25 | 167 mm | za 4 dny |
| 26 | 174 mm | pozití |
| 27 | 181 mm | dodášte vypredané |
| 28 | 188 mm | pozití |
| 29 | 195 mm | pozití |
| 30 | 202 mm | za 4 dny |

[Dok se doma změnit velikost?]()

<Form>
<input type="number" value="1">
ks
<input type="submit" value="Přidat do košíku">

<small>
<>Doprava a platba:</>
Nad 1 500 Kč zdarma. Jinak 80-150 Kč po celé ČR. Platba kartou, převodem nebo vložkou zdarma.
<a href="#">Více</a>
</small>
• CSS
• JS

```

Změny, které jsme provedli v druhé iteraci práce s obsahem. Doplňili jsme dostupnost velikostí a informace o ceně dopravy. cdpn.io/e/XpbBJy

První problém: chybějící informace o dostupnosti velikostí bot

Návštěvníka také chceme přehledněji informovat o vnitřní velikosti bot, která je u koupě přes internet důležitá. A poslední související změnou je to, že jsme přidali doporučení ohledně zařízení na domácí

přeměření velikosti nohy. Nejsme totiž jeden z těch e-shopů, kterým je jedno, co si zákazník koupí.

Druhý problém: chybějící skladová dostupnost jednotlivých velikostí

Doplňme také stručné informace o ceně dopravy a o platbě, která by mohla u nového e-shopu budit pochyby. Plná verze bude samozřejmě standardně v obchodních podmínkách.

Třetí problém: pořadí informací na stránce

Kód produktu a jeho zařazení v kategoriích by neměly stát tak vysoko v hierarchii dokumentu. Podobné je to s detailním popisem produktu. Náš e-shop se má chlubit širokou skladovou nabídkou a na rozdíl od ostatních nabízet znalosti usnadňující rozhodnutí o koupi. Vzpomeňme na výstupy z našeho design canvasu.

Řekněme, že v dokumentu už žádné zásadní chyby nevidíme. Hurá! Můžeme otevřít bránu do veselého světa barev, do světa webové grafiky.

Zapamatujte si

- Proces tvorby webu probíhá v cyklech „Výzkum a definice“ a následný „Návrh a realizace“.
- User Centered Design Canvas stručně shrnuje výzkumnou fázi projektu.
- Když si obsah stránky vložíme do okna mobilního prohlížeče, zviditelní se nám nešvary, které bychom jinak neviděli.

Kapitola 3: Dokument jako základ

Když jsem někdy před rokem 2000 dělal své první weby za peníze, zadání bylo jednoduché: Vezmi tenhle tištěný katalog a převed' jej do HTML. Vzniklý web měl pevně dané rozměry a všelijak napodoboval vzhled katalogu, který jsem měl položený vedle monitoru s rozlišením 800 na 600 pixelů. Ano, byl to středověk. A webdesign byl v té době nevolníkem tiskařiny.

Responzivní design je v jistém smyslu emancipační vlna, která webdesign osvobozuje z područí starších médií. Dneska už například jasně víme, že weby nemohou mít fixní rozměry.

Ale také jsme zjistili, že webový design není možné navrhovat a implementovat pixel po pixelu, tak jak jsem to dělal při převodu tištěných katalogů. Zjistili jsme, že webovému designu více sedí tvorba skládáním a vrstvením.

V téhle zásadní kapitole si rozpitváme první ze dvou základních vrstev designu i technologie – dokumentovou vrstvu.

1. Podíváme se, co nám může přinést vrstvení designu a jak to vychází z technických specifik webu.
2. Vývojářům a začínajícím designérům naservírujeme nutné základy grafického designu.
3. Blíže se podíváme na typografiu, ale také dopady responzivního webdesignu na ni.
4. Dám vám pár tipů na zdroje pro grafické elementy.
5. Vymyslíme základní vizuální styl pro fiktivní e-shop

ForestKid.cz.

6. Poprvé se v knize ponoříme do lehkých technikálií: do jednotek, které na webu můžeme používat. Na ně navážeme technikami zvětšování stránky a komponent a plně responzivní typografií.
7. Představím vám nástroje, které používám pro technickou implementaci dokumentové vrstvy webu.
8. Na konci kapitoly převedeme grafický charakter dokumentu do příkladu. Těšíte se tak jako já?

Vrstvení

Tak bych jedním slovem popsal svůj proces návrhu a implementace webových rozhraní. První z vrstev je dokumentový základ.



Dokumentová vrstva obsahuje společné elementy grafického designu: barvy, typografii, velikostní stupnici, grafický styl a další prvky. Z nich pak vychází komponenty uživatelského rozhraní, obohacené o layout a chování

Na obrázku vidíte opravdu jen nejzákladnější dělení. Dokument

obsahuje mnoho dalších vrstev, což si ukážeme ke konci kapitoly. A komponenty? I ty můžeme dělit do mnoha hierarchií, bylo by to ale už nad rámec téhle knížky.

Musíme se ale shodnout na vrstvení. To totiž považuji za jednu ze základních (a skvělých!) vlastností média, kterému říkáme web. Je to podstata fungování webových technologií a ty dávají mantinej designu. Stavění od dokumentové vrstvy je fajn ještě z jednoho důvodu – umožní nám rozfázovat proces návrhu tak, abychom se nezabývali příliš mnoha problémy najednou. A abychom nezačínali od konce.

Web je vrstvený z podstaty

To, že vidíme nějakou webovou stránku, je možné jen díky správnému fungování skrytých vrstev. Technicky řečeno: zobrazení HTML stránky vyžaduje URL adresu, která vyžaduje HTTP protokol, který je zase postavený na vrstvě TCP/IP.

My ale tak hluboko nepůjdeme. Potřebujeme znát hlavně vrstvení tří hlavních technologií pro tvorbu webu: HTML, CSS a Javascriptu.

Chování

Javascript

Prezentace

CSS

Obsah

HTML

*HTML slouží k vyznačení toho nejdůležitějšího: obsahu, jeho struktury a významu.
CSS nastavuje vzhled a Javascript zase definuje chování stránky*

Dobře, ale co s tím?

Vrstvení zlepšuje kompatibilitu

Je to skvěle vymyšlené. Když selže Javascript, zobrazí se stylovaný obsah. A když selže i CSS, dostanete alespoň obsah.

Nikdy nevíte, kdo a s jakým vybavením přijde zrovna na váš web. Rozdelení obsahu, vzhledu a chování je výhodné i z pohledu kompatibility. I ten nejexotičtější nebo prastarý prohlížeč vám při dodržení správného postupu zobrazí strukturovaný obsah. Jeremy Keith takto na WebExpo 2015 citoval Jake Archibalda:

Když selže výtah, je nepoužitelný. Když ale selžou jezdící schody, stanou se z nich prostě schody. Měli bychom budovat jezdící schody, ne výtahy.

Jeremyho přednáška „Enhance!“, která se těmito principy detailně

zabývá, je dostupná online. [slideslive.com/38894415/enhance](https://www.slideslive.com/38894415/enhance)

Weby závislé na Javascriptu? Opatrně s tím

Některé webové aplikace na přítomnost Javascriptu plně spoléhají. V HTML kódu pak často bývá jen odkaz na skript a až ten obstarává vygenerování obsahu a stylů. Všechny vrstvy zajišťuje Javascript. I obsah a prezentace jsou na něm závislé.

Někde je to z pohledu efektivity výhodné, někdy to dokonce ani jinak nejde. Jen je dobré znát nevýhody takového řešení.

V Javascriptu stačí jedna chyba, a nefunguje vám žádná z vrstev. Jako další nepříjemný bonus „dostanete“ horší indexování vyhledávači a horší celkovou přístupnost obsahu.

Vrstvení je výhodnější pro indexování vyhledávači

I přes to, že se Google už o obsahu vygenerovaném Javascriptem leccos naučil, kvalitní HTML podklad je spolehlivější zdroj. Nemluvě o tom, že další roboti (třeba ten Seznamu nebo Facebooku) ke dni psaní neumí obsah generovaný Javascriptem indexovat vůbec.

Pokud by vás zajímaly detaily o poněkud nedokonalém indexování javascriptových stránek, podívejte se na přednášku Jana Tichého „Vyhledávače a Javascript“. youtu.be/kU_M1elyunw

Vrstvení zpřístupňuje obsah širší cílové skupině

Přístupnost. Široká škatule, do které patří i kompatibilita a vyhledávače, ale také usnadnění přístupu hendikepovaným uživatelům. A pozor, nejsou to jen lidé s oční vadou nebo jiným fyzickým omezením. Momentálně „hendikepovaná“ může být i zdravá dvacetiletá studentka, které některé rozšíření v prohlížeči zablokuje zrovna váš javascriptový soubor. Tohle je web – takové věci se stávají.

Studie prokázaly, že přístupný web se lépe používá všem

návštěvníkům. Ať vás proto nenapadne přístupnost podceňovat. Nepřístupný nebo hůře přístupný web vás může připravit o celou řadu návštěvníků a možných zákazníků.

Někdy se stačí podívat hned na první řádku HTML kódu. Mluví o typu *dokumentu*. Heydon Pickering v knize „Inclusive Design Patterns“ říká:

<!DOCTYPE html> slouží jako důležitá připomínka toho, že i když navrhujete interakčně složité a dynamické rozhraní, stále prostě jen vkládáte obsah do okna prohlížeče.

Vrstvené CSS: kodér potřebuje vidět systém, ne vnější znaky systému

Pokud má web vrstvení v DNA, pak CSS je takový malý vrstvící maniak. Pozor na něj. Následuje nenápadný kousek kódu, který ale dost přesně popisuje důvody, proč vás s těmi vrstvami tak otravují. Mají totiž vliv na designérské procesy:

```
/* Dokument */
html      { color: navy  }
table     { margin-bottom: 1em }

/* Komponenty */
.table    { font-size: 80% }
.table-bg { background: ivory }
```

Pro celý web si jako barvu písma nastavíme námořnickou modř. Všem tabulkám pak v dokumentové základně nastavíme spodní vnější okraj na velikost písma. Tabulky mimochodem mezičím přebraly globální barvu písma. V další vrstvě, v komponentách, pak konkrétním skupinám tabulek pojmenovaným `.table` ještě

zmenšujeme výchozí velikost písma. A jejich variaci `.table-bg` ještě barvu dáme slonové kosti.

A všechny ty krásné vizuální vlastnosti pak bude mít komponenta zapsaná v HTML takto:

```
<table class="table table-bg">  
    ...  
</table>
```

Komponenta `table` prostě zdědila styly z dokumentu a je drobně upravená modifikací `table-bg`.

Ted' uvedu CSS kód, který by mohl napsat kodér bez jasně daných pravidel pro designérskou základnu webu:

```
/* Špatně: komponenty bez společné základny */
```

```
.table {  
    color: navy;  
    font-size: 80%;  
    margin-bottom: 1em;  
}  
  
.table-bg {  
    color: navy;  
    font-size: 80%;  
    margin-bottom: 1em;  
    background: ivory;  
}
```

I když bude výsledek stejný jako v předchozí ukázce, kodér bude z takového řešení brzy nešťastný. Sami vidíte, že je tam spousta opakujícího se kódu. Takové řešení se bude špatně měnit i špatně číst a zpětně upravovat. Ve výsledku bude jen náchylnější k chybám a bude zpomalovat práci.

Kodér tedy potřebuje systém, a to systém vrstvený a skládaný.

Dostane ho ale od designéra?

Designéři, navrhujte od dokumentu, nikoliv od layoutu

Asi většina webů dnes vzniká v kreslicím nástroji, jako je Photoshop nebo Sketch. Grafici si otevřou prázdný dokument, plochu jako ruzyňská přistávací dráha...

A první věc, kterou tam udělají, je rozvržení. Layout. Až pak se přes komponenty rozhraní propracují k obsahu a jeho vlastnostem. Poznáváte se? Nedivím se, je to intuitivní proces, který vám nenápadně nadiktoval nástroj, který používáte.

Kodér, který pak výsledný soubor z Photoshopu zpracovává, musí namísto převodu grafického systému do CSS tupě přepisovat, co vidí. Do technologie převádí vnější znaky systému, ne systém samotný.

Zkusme si v knížce ukázat postup, který jde tvorbě vrstveného systému na ruku. Když už máme v příkladu vyladěný obsah, vybereme písma, barvy a celkový grafický charakter. Prostě vizuální základnu. Pak teprve navrhneme komponenty typu navigace, a až nám obsah začne přetékat „z Ruzyně“, teprve pak jej zalomíme layoutem.

Grafický design

Grafický designér navrhuje firemní identitu, reklamní plochy, obaly, aplikace nebo (chvíle napětí!) weby.

Pojďme se o této části návrhu webu pobavit alespoň v nejzákladnějších obrysech. Otevřeme si k tomu například aktuální

web jednoho z nejzkušenějších českých webových grafických designérů, Jiřího Tvrda.



Web Jiřího Tvrda. Je vlastně děsně jednoduchý. Ale díky kvalitě jednotlivých prostředků velmi účinný. tvrdek.cz

Prostředky, které Jiří použil k sestavení kompozice, nejsou nijak složité:

1. **Styl vzhledu**

Grafický styl, jakým jsou vyvedeny prvky stránky. V době psaní ještě stále letí minimalistický „flat“ styl, pokud byste potřebovali příklad.

2. **Barvy**

Dobře vybrané barvy nebo jejich kombinace chytí toho správného uživatele za srdce.

3. **Typografie**

Písmo a jeho kombinace jsou jedním ze základních stavebních kamenů vizuálního výrazu.

4. **Multimédia**

Fotky, ilustrace, videa nebo třeba koláčové grafy, když chcete.

5. Rozvržení

Layout jak celé stránky, tak rozmístění prvků v rámci jednotlivých komponent.

Kromě této základní čtyřky mají designéři k dispozici další prostředky. Například interakce a animace. Ty na obrázku nevidíte a v některých případech je grafičtí designéři ani nenavrhuji. Může to být práce specializovaného interakčního designéra nebo prostě frontend kodéra.

Je asi jasné, že velmi záleží na kvalitě každé z těchto čtyř komponent. Takže dobře navržený web s nekvalitními fotografiemi prostě fungovat nebude, čehož jsme každý den nešťastnými svědky. Každý ze čtyř prostředků, snad kromě layoutu, podléhá módě. Ale to, jak dohromady fungují, by nemělo být výsledkem grafikova aktuálního rozpoložení pod vlivem módy a měkkých drog. Ty jsou povoleny, ale i tak by grafika měla být důsledkem promyšleného procesu.

Grafika se může líbit, ale musí sloužit!

Jeden z nejčastějších omylů webdesignérů a jejich klientů. Musím to zopakovat i já. Cílem webové grafiky není *lívost* nebo *podbízivost*. Grafika má sloužit účelu, pro který web tvoříme.

Barvy, použitá písma, styl fotografií nebo rozvržení vyvolávají ve čtenáři nějaký pocit. Nesmí to být náhodný pocit nebo dobrý pocit u špatných lidí, jako je grafik nebo klient. Pokud se klientovi web líbí, je to fajn. Když ale neplní potřeby cílové skupiny, je to průšvih.

Jak to zjistit? Předám tady slovo Janu Řezáčovi, přednímu UX konzultantovi:

Názory se nepočítají. Jak tedy přistoupit k designu objektivně? Víte, jak má váš design působit na návštěvníky. A to můžete otestovat.

Jan to na svém webu rozebírá spolu s dalšími častými chybami při návrhu a posuzování grafiky. Pro rychlé posouzení grafiky doporučuje pětisekundový nebo kartičkový test.

houseofrezac.com/grafika

Grafik versus designér

Takže už všichni víme, že barvy, fonty a tlačítka nemají být jen hezké, ale také účel plnící. Proto považuji za dobré zmínit rozdíl mezi často splývajícími pojmy „grafik“ a „designér“.

Hezky to popsal Otto Bohuš:

*Grafik navrhuje tiskoviny, sází brožury, maluje ikonky...
Prostě dělá grafiku. To je v naprostém pořádku. Práci sponustý grafiků jen tiše obdivuji. Vy (zadavatelé) ale hledáte webového designéra.*

Více v jeho článku „Co by měl každý vědět o tom, jak se dělá web: 2. díl“. ottocopy.cz/jak-udelat-web

Zkrátka každý dobrý webový grafik je zároveň designér, protože nedělá jen „hezké věci“, ale snaží se vyřešit nějaký klientův problém.

Bylo to moc stručné, já vím. Zájemce o hlubší průzkum vizuálního designu bych asi v dalším kroku poslal do kapitol o typografii, barvách a layoutu online knihy „Designing for the Web“ od Marka Boultona. designingfortheweb.co.uk

Typografie na webu

Pojďme si projít základní množinu znalostí o využití písma na webu, zmínit pár častých chyb a dvakrát podtrhnout hlavní pravidlo pro stavbu responzivního rozvržení stránky. Předtím ale ještě zmíním jeden účel písma, na který se často zapomíná.

Písmo v nás vyvolává emoce

Než se začteme, může nám typ písma (spolu s dalšími prostředky vizuálního designu) sdělit informaci, co od webu očekávat. Jsem na webu seriózního magazínu, užitného webu typu e-shopu, nebo na stránkách Déčka, určeného pro děti? Asi je jasné, že tohle všechno je možné sdělit nebo zpochybnit mimo jiné i volbou písma. Je toho ale mnohem více.

Pro potřeby předání základní úrovně typografických znalostí mně tady připadá lepší začít z druhého konce. Chybami.

Časté typografické chyby

Vypadají triviálně, ale weby jsou jich plné. Příliš dlouhé řádky, špatný kontrast a nesprávné znaky.

1) Příliš dlouhé řádky

Wikipedie je smutným rekordmanem v délce řádku. Řádek by obecně neměl obsahovat více než 75 znaků, aby oči nepřeskakovaly na řádky sousedící. Ještě o tom budu psát.

Technologie [editovat | editovat zdroj]

Pro realizaci webových stránek se používají zejména technologie **XHTML** (pro strukturu a textový obsah) a **CSS** společně s obrázky (**PNG, GIF, JPG**), které trojí grafickou podobu webu. Navíc se někdy používají další technologie umožňující výšší interaktivitu jako např. **JavaScript, SVG, Flash** či **Java applet**. Do webdesignu někdy lze počítat také části tvorby **serverové části aplikací** – programované v jazyčcích jako **PHP, Python, Java** či **ASP** – a záležitosti spojené se zvyšováním úspěšnosti stránky (**SEO, copywriting**).

Používané nástroje [editovat | editovat zdroj]

Počítačem grafické návrhy webu se mohou dělat v běžných grafických editorech. Samotná tvorba kódu výsledné podoby stránek v **XHTML** a **CSS** je však možno provádět „ručně“ v běžných či speciálně navržených textových editorech (náročnější na schopnosti autora, ale lepší kontrola nad výsledkem) nebo v nástrojích umožňujících **WYSIWYG** (vzhledem k tomu, že jsou komfortní, nevyžadují temel žádné znalosti). Existuje také **WYSIWYG** editory ve formě webových aplikací.

Související články [editovat | editovat zdroj]

Webdesign je velmi masy obor a to se projevuje i v přístupu ke vzdělávání. V roce 2012 neexistuje v České republice jediná vysoká škola, která by nabízena kompletní přístup ke vzdělávání webdesignerů (např. Masarykova univerzita má pro webdesign jeden relevantní předmět – **PV219 Seminář webdesignu**). Webdesignerji jsou tedy převážně samouci, a to se výrazně projevuje na vysoké variabilitě kvality jejich výstupů.

Související články [editovat | editovat zdroj]

- [WWW](#)
- [Search Engine Optimization](#)
- [W3C](#)
- [W4D](#)

Videa – přednášky o webdesignu [editovat | editovat zdroj]

- [Jan Resáček - Úvod do webdesignu](#) – přednáška o webdesignu v rámci Bloku expertů Masarykovy univerzity (duben 2012)
- [Jan Resáček - Členecík ve webdesignu](#) – přednáška o aplikaci typografie ve webdesignu v rámci Barcampu 2011 v Brně
- [Jan Sládeček - Responsive design](#) – přednáška o responsivem přístupu ve webdesignu v rámci Barcampu 2011 v Brně

Externí odkazy [editovat | editovat zdroj]

- [Tučka a výjev analýzy na Open Directory Project](#)
- [W3C](#) – Konsorcium W3C dělajícíci na webové standardy (anglicky)

 Tento článek je [příslušnou](#) nebo poslední dílo dležité informace.
Pomozte Wikipedii tím, že jej vnodně [rozšíříte](#). Nevkládejte však bez oprávnění cizí texty.

Wikipedie na počítačovém rozlišení obrazovky

2) Špatný kontrast a další technické parametry

Novinky jsou nejen vysázené Georgií, patkovým písmem s vynikající čitelností pro delší texty, ale také velmi kontrastní barvou. Na českém webu jsou i výrazně horší weby než Zdroják, ale uvádí ho jako hůře čitelnou variantu díky kombinaci několika faktorů.

Novinky.cz

Ve čtvrtek bude zpočátku dne převládat ještě oblačno až zataženo, ale postupně se bude oblačnost protrhávat na polojasno. Teploty budou stoupat ke 14 až 18 stupňů.

V pátek začne na naše území proudit teplý vzduch od jihozápadu. Převládat bude jasno či polojasno s ranními teplotami mezi 8 a 4 stupni, přes den však bude růst šířhat na 19 až 23 stupňů.



Zdroják.cz

I tvořící lidé jako programátoři, designéři, copywritéři, analytici a další se velmi často nechají plynkostí strhnout. Tvoří pak v krátkých úsecích a mezi ním se nechávají rozptylovat emaily, sociálními sítěmi, schůzkami, pracovními chaty a tak dále. To co vznikne v těch krátkých úsecích mezi plynkou prací pak nemůže mít kvalitu výstupů z práce hlubokou.

To asi všichni tak nějak tušíme, že? Cal Newport ale navíc podává přesvědčivé argumenty o tom, že pokud se více věnujete práci, vaše schopnost dělat pocitou hlubokou velmi klesá až zmizí.



Hlubokou práci ocení tvůrčí profese, pro manažery to není

Petr Staniček nedávno přeložil text Paula Grahama „Pracovní režim tvůrců a režim manažerů“, který vlastně mluví o tomtéž z jiné strany:

Manažerský rozvrh je pro šéfy. Vychází z tradičního plánovacího kalendáře s každým dnem rozděleným na hodinové intervaly. [...]

Zdroják.cz má bezpatkové písmo s horším kontrastem a délkom řádků kolem 120 znaků

Kontrast si můžete zkontořovat v nástroji WCAG Contrast Checker.
contrastchecker.com.

3) Nesprávné znaky

Každé rozumné písmo má speciální symboly pro uvozovky (nikoliv symbol palce), pomlčky (nikoliv spojovník, který na klávesnici můžete považovat za minus) nebo výpustku (nikoliv tři běžné tečky).



Leonardo DiCaprio se vsadil se svým hereckým kolegou Tomem Hardym, se kterým si zahrál ve snímku REVENANT Zmrtvýchvstání (2015). DiCaprio si byl jistý, že Hardy za ztvárnění své postavy získá nominaci na Oscara za nejlepší mužský vedlejší výkon. Vítěz sázky měl vytvořit tetování pro druhého. Vzhledem k tomu, že Hardy skutečně nominaci získal, musí si nyní nechat udělat tetování, které mu vytvořil DiCaprio. "Napsal pěkně hnusným rukopisem 'Leo ví všechno', řekl jsem, že si to nechám vytetovat, ale musí to napsat pořádně," svěřil se Hardy britskému časopisu Esquire.

Aktuálně.cz používá nesprávné znaky

Není to žádná buzerace typografických snobů – prostě se to lépe čte. Typografický tahák od Beneš a Michl vám může velmi pomoci.
vrdl.in/am9wu (PDF)

Ideální šířka a výška rádku

Ted' zpozorněte, protože zmíním jeden ze základních designérských principů dnešního (responzivního) webdesignu.

Na příkladu Wikipedie jsem ukazoval, jak se může dlouhý řádek negativně projevit do celkové čitelnosti textu a webu. A není to jen problém Wikipedie.

Platí totiž následující:

- 66 je ideální počet znaků na jedné řádce,
- 45–75 je pak vyhovující rozmezí.

U webů, jako je právě Wikipedie, se čtenářům stává, že snadno ztrácejí aktuálně čtenou řádku. Rychlosť čtení se tím snižuje.

Jsou to pravidla, která zpopularizoval Robert Bringhurst ve své knize „The Elements of Typographic Style“ a která jsou průběžně potvrzována nejrůznějšími studiemi. Ale vyzkoušet si je můžete i sami na sobě.

Na malých displejích však není možné optima dosáhnout.

Doporučení pak říkají s ubývajícím počtem znaků na řádce snižovat i jeho výšku, protože oči častěji přecházejí z jedné řádky na druhou.

Tady je jedno z možných řešení v CSS, které ukazoval Marko Dugonjić ve své přednášce „Responsive Web Typography“ na WebExpo 2014. vrdl.in/rwdtypo

```
/* Méně než 45 znaků */
body { line-height: 1.4 }
p   { margin-bottom: 1.4em }

/* 45–60 znaků */
body { line-height: 1.45 }
p   { margin-bottom: 1.45em }

/* 60–75 znaků */
body { line-height: 1.5 }
p   { margin-bottom: 1.5em }
```

Toto nastavení předpokládá vysázení patkovým písmem a do jednoho sloupce. Drobně se samozřejmě může měnit podle parametrů písma. Jinak to bude pro nepatkové písmo, pro jiný kontrast, pro specifický charakter písma nebo počet sloupců. Nejlépe nám správnou volbu potvrdí poctivé uživatelské testování, ale pro začátek stačí nastavení písem testovat na různých zařízeních a různých lidech ve vašem okolí.

Příliš malý řádkový proklad spojuje sousedící znaky, zhoršuje

čitelnost slov a ve výsledku zpomaluje čtení. Příliš velký zase vypadá jako seznam samostatných položek a nutí uživatele přemýšlet, zda se jedná o souvislý text nebo o nějakou formu odrážek.

Nicméně délka a výška rádku je první designérské pravidlo, na které bychom při návrhu rozhraní měli myslet. Postup návrhu pak ideálně vypadá tak, že zvolíme písmo, získáme obsah a až na těchto dvou nerozlučných přátelích postavíme systém pro layout stránky.

Další zdroje o typografii

Jasně, vnímáte mě dobře. Typografii mám za jeden ze zásadních stavebních kamenů přípravy vizuálu skoro každého webu.

A myslím, že ze všech pěti prostředků grafického designu, které jsem zmíňoval, by právě typografii měli nejvíce rozumět i kodéri a vývojáři. Protože oni jsou často ti „sazeči“, kteří mohou mnohé ovlivnit.

- *Kniha „On Web Typography“*
Skvělá učebnice typografie od Jasona Santa Maria. vrdl.in/76nb2
- *Přednáška „Praktická typografie pro webové kodéry“*
Dan Srb se hezky rozpoval na jedné z akcí Frontendisti.cz.
youtu.be/bJLGEMQ3rnM
- *Online kniha „The Elements of Typographic Style Applied to the Web“*
Bible od Roberta Bringhursta a spoluautorů. webtypography.net

Zdroje elementů grafického designu

U jednodušších webů nebo webových aplikací je možné k návrhu grafického designu přistoupit jako ke skládance.

Servíruji zde zdroje, které pro jednotlivé příslušnosti grafického designu webu používám sám. Pokud opravdu začínáte, držte se zásady „méně je vždy více“.

Písmo

O výběru písma více píšu v následujícím textu o příkladu. Tady alespoň pár odkazů na zajímavé zdroje a tipy k nim.

Systémová písma

Nezapomínejte na ně. I v operačních systémech totiž čekají kvalitní písma, až je využijete. Kvalitních fontů je dnes v operačních systémech dost, ale naprostá většina není široce dostupná. Ve snaze být velmi stručný doporučím hlavně dvě klasiky od Matthewa Cartera, které najdete skoro na každém zařízení – *Georgii* pro čtení a *Verdanu* pro nadpisy a prvky uživatelského rozhraní.

Google Fonts

Asi znáte. Ale pozor: jako každá služba zdarma, i Google Fonts nabízí i řadu nekvalitních nebo pro váš účel nevhodných písem. Pokud o písmech nic moc nevíte, raději si nechte poradit. Typewolf už jich pár vybral za vás. Jen si ověřte podporu českých znaků.
typewolf.com/google-fonts

Kombinování písem

Výběr sady dvou písem pro jedno užití je víceméně vyšší dílčí. Pokud s ním nemáte zkušenosť, hledejte předvybrané sady. Na FontPair skladují přijatelné kombinace právě z Google Fonts.
fontpair.co

Barvy

- Ze všech databází barevných schémat jsem si oblíbil službu Coolors. colors.co
- Colormind má podobné výsledky, jen namísto šikovných uživatelů zapojuje chytrý stroj. colormind.io
- Pro pokročilé je tu pak služba Paletton od Petra Staníčka. paletton.com

Fotky, ilustrace, ikony

- Iconfinder skladuje ikony a jednoduché vektory. Můj velmi častý zdroj. iconfinder.com
- PhotoPin a Pixabay skladují fotografie s licencí Creative Commons. photopin.com pixabay.com
- Adobe Stock pro všechny ostatní obrázky. stock.adobe.com

Dekorace, opakování vzory

- Opakování vzory na ColourLovers.com.
colourlovers.com/patterns
- Dekorační tvary tamtéž. colourlovers.com/shapes

Zdroje, které uvádím, by rozhodně neměly být jakousi internetovou náhradou grafického designéra. Jistě vás nepřekvapí, že mezi specifikací cílů, cílových skupin a výběrem barvy nebo písma je kvantum znalostí, které specializovaní grafici a typografové nosí pod čepicí.

Na druhou stranu tyto zdroje využívají také mnozí profesionální designéři, protože s jejich pomocí šetří čas a hledají u nich inspiraci.

Ted' už pojďme získané znalosti zúročit při návrhu vizuálního designu našeho fiktivního webu.

Příklad: Barvy, písma a grafický styl

Ted' chvíliku prakticky. Vybereme barevné schéma a sadu písem.

Barevné schéma pro odlišení od konkurence

Pro e-shop potřebujeme alespoň tři základní barvy:

- *Neutrální* pro text a bezvýrazné prvky uživatelského rozhraní.
- *Primární* pro aktivní prvky, hlavně primární výzvy k akci. Tahle barva by měla být výrazně odlišná od té neutrální.
- *Doplňková* barva se nám bude hodit pro zvýraznění komunikujících elementů, jako jsou informativní a chybová hlášení.

Z výběru barevného schématu pro zjednodušení nebudeme dělat vědu, přestože i barvy samozřejmě věda jsou. Měli bychom myslit na cílovou skupinu – v našem případě ženy z velkých měst – a samozřejmě do barev nějak promítнуть tematiku e-shopu

Hledáme barevné schéma se třemi a více barvami, které ladí s ženskou cílovou skupinou. A také nás výrazně odlišují od konkurence, která vždy spoléhá na zelené asociace s lesem. ForestKid.cz tedy jako uniforma hajného vypadat nebude.



Vybrané barevné schéma. colorso.co

Jak vidíte z obrázku, barevné schéma je... hodně barevné:

- Zcela nalevo je „infračervená“, barva pro primární výzvy k akci.
- Druhá zleva je „žlutooranžová“, doplnková barva pro komunikující elementy, jako jsou informativní hlášení.
- Uprostřed je „karibský zelená“, doplnková barva pro text, kterou můžeme použít například pro nadpisy sloupců tabulek.
- Následují dva odstíny modré neutrální barvy pro text a neaktivní prvky stránky.

Vybereme písmo: jedno charakteristické a druhé obsahové

Při výběru typografie opět nejprve zvážíme, pro jaké účely budeme písmo potřebovat. V našem případě to zjednodušeně obnáší logo firmy a pak elementy webu: nadpisy, delší texty a navigační prvky uživatelského rozhraní. Posoudíme samozřejmě i vhodnost písem

pro další média, hlavně tisk.

Osobně výběr obvykle začínám buď od kvalitního systémového písma, ke kterému hledám do páru doplňkové písmo. Systémový font nepředstavuje žádnou datovou zátěž, ale jak už jsem zmínil, široce dostupných systémových písem je opravdu málo.

Druhá možnost je najít výrazné a neobvyklé *značkové* písmo a k němu do páru hledat písmo *obsahové*, pro vysázení textů a komponent rozhraní. Tvář našemu e-shopu chci dát právě výrazným písmem, proto padla volba na *Yesева One*. Je nepřehlédnutelné a také „výrazně ženské“, jak píše jeho autor. Písmo pro e-shop s cílovou skupinou maminek jako dělané. Yes, Eva! fonts.google.com/specimen/Yeseva+One

Zbývá nám písmo pro texty a rozhraní. Pro delší texty, které na webu mít jistě budeme (vzpomeňte na plánovaný blog), je vhodnější patkové písmo. Jenže to by pak znamenalo hledat třetí rodinu pro uživatelské rozhraní, kde se patky nehodí kvůli prostorové neúspornosti. Vybraný *PT Sans* je naopak velmi úsporný, taková méně velkorysá *Verdana*. A má hezkou kurzívou, kterou budeme spolu s tučným řezem potřebovat v delších textech.

fonts.google.com/specimen/PT+Sans

Yeseva One

Dětské celoroční trekové boty

PT Sans

Díky speciální membráně, která je mezi vrchem a podšívkou, zůstává noha suchá a dobře klimatizovaná. Podešev je z *termoplastického kaučuku* s **módním dezénem**, vyvýšená část podešve v přední části chrání špici při běžném zakopávání.

Vybraná písma Yeseva One a PT Sans

Přidáno do našeho projektu to vypadá jako v následujícím CodePenu. cdpn.io/e/XpbPmm

Grafický styl: minimalistický s jednoduchými linkami

Vybrali jsme poměrně výrazný barevný i typografický styl. Navíc si pro nás příklad můžeme dovolit předpoklad, že dokážeme zajistit kvalitní obrazový obsah.

E-shop je navíc z principu užitková aplikace. Ikony, tlačítka a další doplňkové grafické elementy by proto neměly poutat příliš pozornosti. Měly by doladit atmosféru a moc nekřičet.

Pro vyhledání grafického stylu jsem využil Iconfinder. Hledal jsem minimalistický styl, dvoubarevné ikonky s obrysem. A našel sady Maxe Gribojedova.

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. Ted' 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. Bud' 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 rádku: 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, nebabíme 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 nastavují 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 určena právě velikostí písma na rodičovském prvku.

Ale vývojářům se s „emky“ samozřejmě pracuje trošku hůř než s „remky“. Ne každý se chce stát pochodujucí 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](https://github.com/niksy/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 autorský nebo uživatelský 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 kočátek, 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 onech 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.

Autorský zoom dokumentu a komponent

Jednotky `em` a `rem` jsou pružné, a tak je možné s jejich pomocí zvětšovat či zmenšovat celý web nebo jeho jednotlivé komponenty.

Zvětšování nebo zmenšování celé stránky

Důležité je sázet pravidla pro základní typografii dokumentu a layout stránky důsledně v `rem` (nebo výjimečně `em`) jednotkách:

```
/* Vzhled komponent: */
h1 { margin-bottom: 1rem }

/* Layout: */
.container { max-width: 30rem }
```

Jak už víte, v `1rem` je uložená vaše výchozí velikost písma. Rozměry takto nastavených vlastností se budou počítat z ní.

Autorsky změnit velikost písma můžeme trvale pro celý web:

```
html {
  font-size: 14px;
}
```

Lusknutím prstu to uděláme také pro konkrétní rozlišení:

```
@media (min-width: 50em) {
  html {
    font-size: 1.25rem;
  }
}
```

Je to například skvělá možnost, jak zvětšit (zoomovat) celou stránku a zlepšit čitelnost obsahu na enormně velkých šířkách okna prohlížeče.

Vzpomeňte si na část první kapitoly o velkých displejích, kde jsme se tím zabývali.

Podobně můžeme zvětšovat či zmenšovat písmo (a pak celý layout) na displejích menších. Každý webový projekt má jinak nastavenou základní typografii, jinak postavený layout, takže k tomu moc obecných rad nejde dát. Ale když budete usilovně zmenšovat a zvětšovat okno a sledovat přitom čitelnost textu, místa vhodná k nasazení celkového zvětšení či zmenšení dokumentu poznáte.

Zvětšování nebo zmenšování komponent

Vezměme sice zjednodušenou, ale o to lépe vše demonstrující komponentu:

```
.component {  
    background: #ccc;  
    padding: 1em;  
}
```

Na jejím místě si můžete představit jen o něco složitější tlačítka, ikony, propracovanější navigaci, fotogalerii a tak dále.

Ted' si stačí nadefinovat obecné pomocné třídy pro zvětšování a zmenšování:

```
.size-sm { font-size: 75% }  
.size-lg { font-size: 125% }
```

O čtvrtinu větší verzi komponenty pak zařídíme takto snadno:

```
<p class="component size-lg">  
    Component  
</p>
```

Pokud chcete takto vytvářet velikostní varianty komponent, musíte dodržet jediné: všechny pružné rozměry sázet v `em`. Použití `rem` by

tady nemělo žádný účinek. Neumí totiž změnit velikost jen pro konkrétní použití komponenty.

em proto používám v komponentách, které mohou měnit velikost podle výskytu. A **rem** ve všech ostatních případech.

Různé komponenty budou vyžadovat různý přístup: Někde budete ještě muset do velikostních tříd přidat pravidlo pro menší výšku rádku, někde to bude stačit tak jako v mé ukázce.

Dává vám to smysl? Kód k tomuto textu je také na CodePenu.
cdpn.io/e/RKQmVG

Elastická typografie

Měnit velikost stránky i jednotlivých komponent už umíme díky předchozí podkapitole a díky jednotkám **em** a **rem**. Při změně velikosti okna se to děje *skokově*.

Co kdybychom ale stránku a komponenty chtěli zvětšovat *plynule*? Prostě elasticky se změnou šířky nebo výšky okna.

K tomu si přizveme dříve už zmíněné jednotky viewportu, hlavně **vw** (setina šířky okna) a **vh** (setina výšky okna).

Elastická (nebo také plně responzivní) typografie je v době psaní textu spíše v počátcích bádání. Nevyřeší zdaleka všechny situace, pro které byste je možná chtěli nadšeně použít. Ale je *cool* i tak. A já ji tady uvedu, i kdyby mě natahovali na elastický skřipec.

Elastická typografie pomocí **vw**

Představte si, že nadpis článku chcete na stránce zvětšovat

a zmenšovat podle šířky okna. To je jednoduché:

```
.heading {  
    font-size: 7vw;  
}
```

Velikost písma bude na sedmi procentech šířky viewportu.

Zmenšíme okno, a vše se zmenší. Zvětšíme ho, a vše se zvětší. Jupí!

Zkuste si to na [CodePenu](https://cdpn.io/e/mAAOLa).

Kód je jednoduchý, funguje ve všech moderních prohlížečích a náhradní řešení pro ty starší bude jednoduché:

```
.heading {  
    font-size: 2em; /* Starší prohlížeče */  
    font-size: 7vw; /* Moderní prohlížeče */  
}
```

Tak proč jsem tedy bručel o nějakých nevýhodách? Nerad kazím oslavu, ale pojďme si představit dvě situace:

1. Jednotka `vw` nezná váš layout. Neví, že jsou v něm fixně nastavené okraje nebo rámečky. Šířka `.heading` tak nemusí být vyjádřitelná v procentech ze šířky okna, na kterou se `vw` odkazuje. A text se prostě na některých viewportech zalomí jinak než na jiných. To vám může vadit.
2. Můžete chtít nastavit minimální a maximální velikost písma. Na mobilech bude sedm procent šířky okna už tak trochu trpasličí velikost, že ano?

První problém dokážu (za jistých podmínek) vyřešit v dalším textu. Ten druhý je zčásti řešitelný nastavením minimální velikosti v `rem`:

```
.heading {  
    font-size: calc(2rem + 7vw);  
}
```

Velikost písma v ukázce nikdy neklesne pod `2rem`. Nadpis se bude při změně šířky okna chovat elasticky. Pokud byste ale chtěli mít typografii důsledně pod kontrolou a zabránit různým zalomením textu na různých velikostech okna, bude se vám více líbit řešení následující.

Elastická typografie podle výšky komponenty

Moje řešení v zásadě nahrazuje jednotku, která v CSS neexistuje, ale pevně doufám, že jednou existovat bude: setinu výšky rodičovského elementu.

Jakmile se totiž odkážeme na rozměry rodiče, při výpočtu velikosti elementu se vezme v potaz layout konkrétní komponenty a text se nám na žádném viewportu nezalomí jinak, než bychom chtěli.

Pokud si onu výšku rodiče představíte jako fiktivní hodnotu `heading-height-percent`, kód by vypadal takto:

```
.heading {  
    font-size: 7 * heading-height-percent;  
}
```

Počítáme teď sedm procent z výšky rodiče, nikoliv šířky stránky. Takže se nám text nezalomí, ani když šířku `.heading` nebude možné vyjádřit v procentech ze šířky okna.

A jak jsme vypočetli tu magickou hodnotu?

```
heading-height-percent =  
(100vw - 2em) /* 1. Layout .heading */  
/ 16 * 9        /* 2. Poměr stran .heading */  
/ 100;          /* 3. Získáme jedno procento */
```

Popíšu to ještě v textu:

1. Layout **.heading**: Prostě ze šířky stránky odečteme postranní okraje komponenty. Tím získáme její přesnou šířku.
2. Šířku **.heading** pak vydělíme poměrem stran komponenty. Díky tomu máme vypočtenou výšku.
3. Výšku **.heading** už pak stačí vydělit stem, abychom získali procentuální údaj.

Ano, řešení to není triviální, a navíc vychází z toho, že znáte poměr stran komponenty. Ale já vám to říkal: Elastická typografie je složitější, než jste si možná myslíeli. Jenže ty výsledky za to stojí, že?

Podrobněji to rozepisují na Vzhůru dolů v článku „CSS řešení: Elastická typografie počítaná v procentech z výšky komponenty“. vrdl.cz/reseni-elasticka-typografie

Pokud jde o tipy na další možná řešení, můžete se obrátit na kolegy ze Smashing Magazine, konkrétně do jejich článku „Truly Fluid Typography With vh And vw Units“. vrdl.in/4g9xs

Máme tedy všechny nezbytné znalosti o typografii a jednotkách v CSS. A máme připravený grafický styl našeho e-shopu. Můžeme tedy začít stavět kód. I ten budeme samozřejmě vrstvit. Ne každý kousek kódu musíme u každého projektu napsat znovu. Proto si ukážeme sadu nástrojů, které ty nejzákladnější vrstvy obstarají za nás.

Blanka a další nástroje pro dokumentovou vrstvu webu

Při formování dokumentu, nevyhnutelném základu každého webu, se může hodit pár nástrojů. Předpokládejme, že máme hotový obsah ve strukturovaném HTML. Co ted?

Seznamte se s Blankou. Žádný tunel to není, nebojte se. Blanka od anglického *blank*: prázdný, čistý, nepopsaný.

Je to má sada nástrojů pro nastavení typografické základny každého webu. Třetí vrstva stavby hned po výchozích stylech prohlížečů a Normalize.css. Dám sem rovnou odkaz, ale nebojte, Blanka pořádně rozpitváme. github.com/machal/blanka-html

Blanka HTML: výchozí šablona prázdného dokumentu

Je to má výchozí šablona pro dokument. Když ji zjednoduším, aby se vešla sem do knížky, vypadá následovně. Nejprve se podíváme na definici typu dokumentu:

```
<!doctype html>
<html class="no-js" lang="cs">
```

Máme zde správný typ dokumentu a nastavený jazyk (`lang="cs"`). Připravili jsme si detekci situace, kdy ve stránce nefunguje Javascript (`class="no-js"`). Můžeme využít při stylování komponent.

Ted' pojďme k hlavičce:

```
<head>
  <meta charset="utf-8">
  <title>____Title____</title>
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <script>
    document.documentElement.className =
      document.documentElement.className.replace("no-js",
"js");
  </script>
</head>
```

Nastavujeme správné kódování znaků (`charset="utf-8"`) a kromě

jiného tady vidíte i správnou meta značku pro viewport. Ještě si o ní povíme v následující kapitole.

A nakonec tělo stránky:

```
<body>
  <a href="#main">skip to main content</a>
  <div class="container">
    <main id="main" role="main" class="content">
      <h1>
        Hello, I'm Blanka!
        <br>Minimal HTML boilerplate.
      </h1>
    </main>
  </div>
</body>
```

Ošetřená je základní přístupnost: Prvek <main> umožňuje uživatelům odečítáčů obrazovky snadný skok na obsah. Ze stejných důvodů nezapomínáme na WAI-ARIA atribut (role="main"). Detaily čtěte u mě na blogu. vrdl.cz/p/html5-struktura

Univerzálnost vyžaduje minimalismus. Pokud chci mít základní kousek kódu jednotný opravdu pro všechny projekty, nesmí být moc složitý. Nechci přemýšlet, kterou řádku pro nový projekt převezmu a která je tam zbytečně.

V Blance toho opravdu moc není. Věřím ale, že vše, co tam je, využijete téměř pro každý váš projekt. Nezkrácenou verzi najdete na Githubu v [blanka.html](https://github.com/machal/blanka-html).

HTML Boilerplate: až moc robustní alternativa

Pokud byste dávali přednost maximalistické variantě, zajímejte se o projekt HTML5 Boilerplate. Pro mě není. Mé projekty se od sebe poměrně dost liší, proto upřednostňuji jednoduché řešení před robustním. Jde ale rozhodně o zajímavý zdroj pro vzdělávání

a inspiraci. html5boilerplate.com

Více k základům HTML nepotřebujeme. Nuda? U stylů to ale bude zajímavější, nebojte.

Výchozí styly prohlížečů

Často se zapomíná, že ještě než napíšeme první řádku CSS, náš dokument už nějaké styly obsahuje. Vždyť prohlížeč musí mít nějaká zadní vrátka, kterými vejde škodolibý skřítek, a klientův vymazlený *dizайн* z wordovského dokumentu vysázený písmem Comic Sans pokazí vědecky vyhlížející stránkou s modrými odkazy vysázenou Times New Roman, že ano?

První vrstvou vzhledu, která se aplikuje na váš dokument, jsou výchozí styly prohlížečů. Ve vývojářských nástrojích je v CSS kaskádě vidíte jako „user agent stylesheet“. Nevidíte? Doporučím vám si jejich zobrazování zapnout. Výchozí styly totiž mají ošklivou vlastnost: V různých prohlížečích mohou mít mírně různá nastavení. Nejprve je vhodné je sjednotit.

Resetování CSS: raději ne

Svého času se v prvním kroku technické práce prakticky na každém webu nasadil CSS Reset od Erika Meyera. Ten vynuloval všechny vnější i vnitřní okraje prvku, čímž jsme získali konzistentně ošklivý Times New Roman a modré odkazy ve všech prohlížečích.

vrdl.in/cssreset

Nevýhoda resetovacího přístupu je (to byste nečekali) v onom *resetování*. Když totiž nějaké vlastnosti „resetujeme“, musíme je v druhém kroku také „setovat“. Nastavit na vysněné hodnoty. Co když nám ale vyhovovalo původní nastavení prohlížečů? To jsme pak udělali dva zbytečné kroky, a styly webu si zkomplikovali hned

na startu.

Normalize.css: sjednocení stylů prohlížečů

Normalize zasahuje jen tam, kde jsou ve výchozích stylech prohlížečů nějaké rozdíly. *Normalizace* stylů, sjednocení jejich vzhledu mezi prohlížeči. Prvním krokem je tedy přidat k dokumentu soubor se stylem Normalize.css. vrdl.in/normal

Díky této druhé vrstvě stylů dokumentu máme výchozí stylování sjednocené napříč prohlížeči. V další vrstvě si už konečně pojďme něco nastavit.

Blanka CSS: typografický základ

Blanka CSS je třetí vrstva stavby stylů webu a má dva hlavní účely:

Sjednocuje vzhled elementů

Normalize.css ladí vzhled napříč prohlížeči, už ale neřeší jednotnost vzhledu uvnitř dokumentu. Třeba levé odsazení u prvků `ul`, `ol` nebo `dd`.

```
ul, ol, dd {  
    padding-left: 1.75rem;  
}
```

Nastavuje základní typografický rytmus

Asi jste si všimli, že mám rád jednoduchost. Aby se mi s dokumentem dobře pracovalo, mají všechny typografické elementy nastavený vnější okraj jen zezdola. Nemusím myslit na horní vnější okraj, nemluvě o vnitřních, které se ve výchozích stylech prohlížečů hojně vyskytují.

Ve stylové Blance je toho více, ale to už si můžete prohlédnout sami

v souboru `blanka.css`. github.com/machal/blanka-html/

```
p, ul, ol, dl, table,  
blockquote, pre, hr, figure {  
    padding-top: 0;  
    padding-bottom: 0;  
    margin-top: 0;  
    margin-bottom: 1rem;  
}
```

Reboot: až moc nastavující alternativa

Opět pro zájemce zmíním i složitější alternativu. Bootstrap 4 přichází s vlastní typografickou základnou, které říká Reboot.
vrdl.in/reboot

Základna obsahuje sjednocení stylu mezi prohlížeči (pro které používáme Normalize.css), sjednocení vzhledu mezi prvky (používáme Blanka.css), ale také jistá nastavení vlastního vzhledu. A právě ne všechna pravidla z poslední kategorie mám v úmyslu využít na všech svých projektech. Nechci jim už v této fázi vnucovat konkrétní písma, konkrétní typografickou stupnici a výšky rádku. To vše vychází ze zvoleného typografického systému a mělo by se psát na míru projektu. O Rebootu jsem ale psal na Vzhůru dolů.
vrdl.cz/b/53-reboot

Když už máme jakžtakž vysázený dokument, můžeme k němu začít přidávat vzhled vlastního webu. Řezy písem, velikostní stupnici, barvy. Promítnou se pak do vzhledu základních prvků dokumentu, nadpisů, odstavců, seznamů, citací, tabulek, formulářových prvků... No však je všechny znáte. Nebo neznáte?

Blanka Type Test

Poslední členka sesterského komanda Blanek vychází právě z toho, že ne všechny HTML elementy musíte znát. Nebo lépe: že si na ně

prostě nemusíte vzpomenout, takže ve stylech je neošetříte.

Kdo z nás nezažil situaci, kdy ve stylech počítal úplně se vším – dokud mu klient přes redakční systém na web nevložil úplně novou kombinaci prvků, která rozbita celý web?

Proto je tu Blanka Type Test, zátěžový test typografie. Je to vlastně dokument obsahující všechny myslitelné i nemyslitelné HTML elementy. V kombinacích, které by nás nenapadly. V zanořených, se kterými jsme nepočítali. V rozměrech a délkách, jejichž představa by vás budila ze sna.

Prostě si stáhněte dokument, přidejte si k němu vlastní CSS a otestujte si všechny možné i nemožné kombinace prvků. Podívejte se na soubor blanka-type-test.cz.html. github.com/machal/blanka-html/

Příklad: hotový dokument

Pojďme do fiktivního e-shopu přidat vlastní typografii, barvy a grafické prvky. Nějak si prostě dokument upravit podle vizuálního stylu ForestKid.cz. Hotový stav příkladu pro tuto fázi si můžete naživo prohlédnout nebo stáhnout na následujících adresách:

- Otevření v prohlížeči: vrdl.in/vdwddok
- Stažení v ZIPu: vrdl.in/vdwddokzip

Co jsme na stránce udělali, nejlépe ověříme pohledem do `style.css`.

Vlastní písma

Hned na první řádce je tohle:

```
@import  
url('https://fonts.googleapis.com/css?  
family=PT+Sans:400,400i,700|Yeseva+One&  
subset=latin-ext');
```

Jasně, přidali jsme naše fonty Yeseva One a PT Sans. Ten druhý ve třech řezech: základním (400), kurzívě (400i) a tučném (700). Pokud vám ta čísla nic neříkají, jde o stupně tučnosti používané také ve vlastnosti `font-weight`.

Sjednocení základní typografie

Znáte už z předchozí podkapitoly. Normalize.CSS vymaže rozdíly ve vzhledu mezi prohlížeči:

```
@import 'vendor/normalize.css';
```

Blanka.CSS sjednotí vzhled napříč HTML prvky.

```
@import 'vendor/blanka.css';
```

Složku jsme pojmenovali `vendor/`, protože jde o projekty externích „dodavatelů“.

Ted' se mrkneme do jiné složky. `core/` obvykle používám pro „programátorské“ části CSS kódu. U složitých projektů tady mívám funkce CSS preprocessorů. V našem případě si vystačíme s proměnnými.

CSS proměnné

Používáme je tady, abychom si zjednodušili práci:

```
@import 'core/variables.css';
```

Z toho bude mít radost naše hlava, protože si nebude muset

pamatovat kódy barev a další často používané údaje.

Definice proměnné vypadá takto:

```
:root {  
    --color-text: #063747;  
}
```

Použití pak například takto:

```
body {  
    color: var(--color-text);  
}
```

Proměnné asi znáte z CSS preprocesorů. Ty jsou pro větší projekty dost užitečné. Píšu o nich v e-booku „Vzhůru do CSS3“ nebo ve starším textu na Vzhůru dolů. vrdl.cz/b/12-css-preprocesory-1

Jak už víte, v knize se chci obejít bez složitějších technikálí a z relativně komplexních CSS preprocesorů bychom toho využili jen málo. Proměnné jsou ale nově přímou součástí kaskádových stylů. A jde s nimi tropit ještě větší legrace než s těmi z preprocesorů. Třeba číst a měnit je z javascriptového nebo HTML kódů. Nastudujte si to na [JeČas.cz. jecas.cz/var](http://jeCas.cz/jecas.cz/var)

Podpora CSS proměnných ovšem není vůbec špatná. Z aktuálních prohlížečů je nezvládají jenom Internet Explorery. caniuse.com/css-variables

Důležité je, že ani v Internet Explorerech se stránka (díky „blbuvzdornosti“ CSS) nerozpadne. Prostě se namísto barev z proměnných použijí barvy výchozí. Texty budou černé, odkazy modré. Vzhledem k rozšířenosti Exploreru verze 11 nám to ale může vadit. Pak doporučuji nasadit automatizované přepočítání proměnných do běžných CSS hodnot. Prohlížeč dostane CSS bez proměnných. Udělejte to pomocí PostCSS. vrdl.cz/p/postcss

Konec technické odbočky, pojďme nastavit další vzhled.

Vlastnosti dokumentu

Velikosti nadpisů a písma, základní styl tabulek, formulářových a dalších prvků.

```
@import 'document/scale.css';  
@import 'document/fonts.css';  
@import 'document/table.css';  
@import 'document/forms.css';
```

To už si nastudujte sami ve staženém příkladu, není to nic složitého.

Pomocné třídy

Pomocné třídy jsou tak trochu mimo ostatní kategorie. Nelze je zařadit do dokumentové vrstvy ani ke komponentám.

```
@import 'helpers/helpers.css';
```

Třída `.focus-only` například styluje prvky, jež mají být viditelné jen při ovládání z klávesnice, třeba odkaz pro skok na hlavní obsah.

```
.focus-only {  
    position: absolute;  
    top: 0;  
    right: 100%;  
}  
  
.focus-only:focus {  
    right: auto;  
}
```

Maximální šířka dokumentu

Vzpomenete si na optimální délku řádku textu [z podkapitoly o typografii](#)? Ideál 66 znaků mám v našem základním písme PT Sans napočítaný kolem **30em**, takže použijeme následující deklaraci:

```
.container {  
    max-width: 30em;  
    margin-left: auto;  
    margin-right: auto;  
}
```

Minimální odsazení ze stran na malých mobilních displejích zase zajistíme touto deklarací:

```
body {  
    margin: 1.5em;  
}
```

Zabývám se tím v `document/document.css`. Tím bychom mohli mít v této fázi hotovo. Dovolte mi ale ještě jednu odbočku pro milovníky pořádku.

Organizace CSS do malých souborů

Často ani profesionální weboví vývojáři moc nehledí na organizaci a způsob psaní CSS kódu. Psaní jakéhokoliv kódu bez jasné štábní kultury ovšem vede ke zbytečným bolehlavům. V knize budeme CSS kód alespoň dělit do malých soběstačných souborů, na které se pomocí zavináčového pravidla `@import` odkazují z hlavního `style.css`. Tvrdím, že je to přehlednější a rychlejší než práce s jedním velkým souborem. Na blogu mám o tomto způsobu dělení článek. vrdl.cz/b/29-organizace-css-2014

Je ale pravda, že z pohledu rychlosti načítání (alespoň na dnes ještě

převládajícím protokolu HTTP/1) je vhodnější posílat prohlížeči CSS v jednom souboru. Šetříme tím dotazy na server, které by na mobilních připojeních velmi oddálily moment vykreslení stránky. CSS preprocesory dělají seskupení importovaných souborů do jednoho automaticky. Udělá to pro vás ale i komponenta `postcss-easy-import` z PostCSS. vrdl.cz/p/postcss

Pěkně jsme si zavrstvili, že?

Vrátím se teď na začátek kapitoly a přinutím vás vzpomenout si na obrázek se dvěma hlavními vrstvami webu: dokumentovou a komponentovou. Pracovali jsme teď na té první, takže se nám povedlo ten krásně jednoduchý obrázek zkomplikovat.

Zjistili jsme, že pod dokumentovou vrstvou máme ještě výchozí styly prohlížečů. V samotném dokumentu pak nejprve sjednocujeme vzhled a pomocí například barev, velikostní stupnice nebo písem nastavujeme vzhled našeho webu. Takže jsme dokumentovou vrstvu rozvrstvili do dalších vrstev.

Ale nekomplikujme to ještě více. Podívejme se na náš e-shop.

A jak teď vypadá rozpracovaná verze ForestKid.cz?

E-shop teď získal základní grafický styl.

Dětské celoroční trekové boty Fare



Obrázky 4:3

Cena: 1 313 Kč s DPH

1085 Kč bez DPH

Velikosti a dostupnost

Velikost	Vnitřní délka	U vás
23	154 mm	pozití
24	161 mm	pozití
25	167 mm	za 4 dny
26	174 mm	pozití
27	181 mm	dočasně vyprodáno
28	188 mm	pozití
29	195 mm	pozití
30	202 mm	za 4 dny

Jak si doma změnit velikost?

1 ks
Přidat do košíku

Proč ForestKid?

- Pomůžeme a poradíme**
Vyznáme se v tom, co prodáváme.
Můžete se nás zeptat na fóru,
publikujeme blog.
- Vše skladem a na jednom místě**
Dnes máme skladem 2 546 položek.
Outdoorové vybavení pro děti? Jdete rovnou na ForestKid.
- Doprava zdarma nad 1 500 Kč**
Ano, je to tak. Většina našich zákazníků dopravu platit nemusí.
- Slevy pro stálé klienty.**
Vyplati se být věrní ForestKid.

Hodnocení zákazníků ★★★★☆

95 % (průměr ze 4 hodnocení).

• ★★★★☆ Jiřina H., Ostrava
„Po sezóně denodenního nošení obuví v lesní škole můžeme říct, že jsme se v kvalitě nezklamali. Jsou to nejen dobré vypadající, ale i dobře vyrobené boty. Přistě bychom jen kupili bez tkaniček.“

Dokument se základním grafickým stylem, barvami a typografií. vrdl.in/vdwddok

Na obrázku trochu kecám a vy zkušenější to víte. Takhle by stránka sama o sobě v mobilu určitě nevypadala. Na obrázku je vidět až stav potom, co jsme prohlížeči oznámili, že je optimalizovaná pro mobilní zařízení.

Udělali jsme to meta značkou pro viewport. I přes to, že jde o jeden řádek HTML kódu, webaři a webařky v něm umí udělat chyby. Pojďme to napravit a o viewportech si něco povědět.

Zapamatujte si

- Web je výhodné tvořit vrstvením: nad obsah pokládat prezentaci a na tu chování.
- I webový design je vrstvený: nad dokumentem jsou komponenty.
- Kodér potřebuje design dostat jako systém. Ne jen vidět jeho vnější znaky.

- Grafika se může líbit, ale musí sloužit.
- Dobrý webový grafik je designér.
- Písmo neslouží jen ke čtení. Také v nás vyvolává pocity.
- Pozor na příliš dlouhé řádky – výrazně nad 75 znaků.
- Kontrolujte kontrast písma a sázejte text pomocí správných znaků.
- Nejdůležitější jednotka je `rem`. `px` je nebezpečné používat pro velikost písma.
- Komponenty i celou stránku je možné někdy zvětšovat či zmenšovat.
- Velké nápisy je možné elasticky zvětšovat pomocí jednotky `vw`.
- Blanka je sada nástrojů pro dokumentovou základnu webu.
- Proměnné v CSS (`--color-text: #063747`) nám ušetří práci.
- Styly rozdělujte do komponent rozdělených do malých souborů.

Kapitola 4: Viewport na mobilech

Smartphony nám přinesly tři zásadní životní radosti: Notifikace chodící ve tři ráno, Facebook plný selfíček – a tvůrcům webů ještě viewporte na mobilech.

Problematika vykreslování do plochy prohlížeče má na mobilních zařízeních svá specifika. Jejich neznalost v důsledku vede k chybám jako je zakázání zoomování obsahu uživatelem nebo používání fixního pozicování. O těch si budeme povídат v kapitole o návrhu rozhraní.

V této kapitole to vezmeme povětšinou technicky:

1. Co je viewport zač a jaké jeho tři typy známe?
2. Jak správně zapsat meta značku pro viewport?
3. Velikost okna prohlížeče v CSS a JavaScriptu?
4. Jak vypadají weby na hodinkách od Apple?

Viewport ve webdesignu: layoutový, vizuální a ideální

Dozvíte se, co je layoutový, co vizuální a co ideální viewport. A také proč je jich tolik.

Co to ale ten *viewport* vlastně je? V kontextu webdesignu jde o označení pro výrez stránky viditelný v okně prohlížeče. Na zařízeních, kde je možné měnit velikost okna (typicky počítačích),

tedy viewport představuje šířku a výšku okna bez rozhraní prohlížeče.

Na mobilních zařízeních potřebují webaři viewportů více, protože obrazovka je malá. Z toho pak vyplývá:

- weby se mu chtejí přizpůsobovat různým způsobem,
- uživatelé zase stránku hodlají zvětšovat nebo zmenšovat.

Od výrobců zařízení jsme tedy dostali dva viewporte a ještě jeden navíc. Jako bonus. Nejdříve ale k historickému kontextu.

Proč neexistuje jen jeden viewport a za co může první iPhone

První iPhone přišel v roce 2006 do situace, kdy byl prakticky každý web navržený jen pro velké displeje. S tím se dítko Steva Jobse snažilo vypořádat zmenšením layoutu webu a přidáním možnosti konkrétní části zvětšovat.

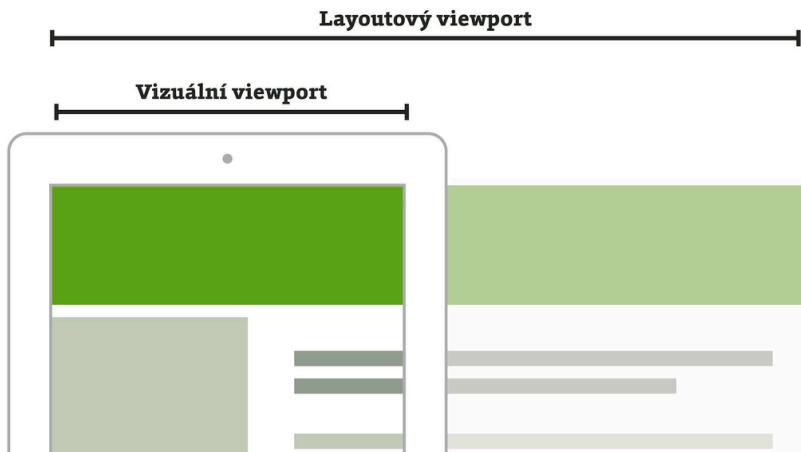
Zároveň v Applu doufali, že začnou vznikat weby přizpůsobené malým obrazovkám. Proto přišli s možností, jak webaři mohou mobilním zařízením sdělit, že pro ně web upravili. My pro to dnes už rutinně používáme meta značku pro viewport:

```
<meta name="viewport" content="width=device-width">
```

Obvykle se tím sjednotí šířka layoutového viewportu se šírkou ideálního viewportu. Je to tedy jednoduché. Ale když uživatel začne stránku zvětšovat nebo zmenšovat, jsme opět v situaci dvou viewportů. Proto je musíme znát.

Zpět tedy k našim viewportům. Budeme používat terminologii Petera-Paula Kocha a rozlišovat viewporty layoutové, vizuální

a ideální. quirksmode.org/mobile/metaviewport



Obrázek: Viewport layoutový a vizuální. Při použití správné meta značky jsou oba při načtení responzivní stránky na všech zařízeních stejně široké

Layoutový viewport

To je plocha, do které se vykresluje layout stránky napsaný v CSS.

Pokud vložíte meta značku pro viewport, má stejné rozměry jako vnitřní část okna prohlížeče. Svou velikost ale mění, když uživatel zoomuje, zvětšuje nebo zmenšuje viditelnou část stránky.

Když na meta značku zapomenete, použije se výchozí šířka layoutového viewportu, která je na dnešních zařízeních vždy 980 pixelů.

Javascriptem zjistíte rozměry layoutového viewportu pomocí `document.documentElement.clientWidth` a `clientHeight`.

K layoutovému viewportu se vztahuje [Media Queries](#) pro velikost

okna `min-width` a `max-width`.

Vizuální viewport

Jedná se o „průzor“, přes který se díváte na layoutový viewport, a tedy stránku samotnou.

Do rozměrů se nezahrnují ovládací prvky prohlížeče. Vizuální viewport zůstává pořád stejný.

Javascriptem zjistíte rozměry vizuálního viewportu pomocí `window.innerWidth` a `innerHeight`. Bez posuvníku pak díky `document.documentElement.clientWidth` a `clientHeight`. Více najdete v [textu o velikosti okna](#).

Ideální viewport

Ideální viewport je trochu z jiné kategorie než dva předchozí. Hodnotu totiž určuje výrobce prohlížeče. Představuje ideální rozměry layoutového viewportu. Počítá se z rozlišení obrazovky v CSS pixelech.

Hodnotu ideálního viewportu získáte, když vydělíte hardwarové rozlišení hodnotou `resolution` (nebo postaru `device-pixel-ratio`).

Takový iPhone X má hardwarové rozlišení kratší strany 1125 pixelů, ale v CSS pixelech je to 375 (Device Pixel Ratio je 3). Ideální viewport je tedy široký 375 pixelů.

Více informací je k dispozici ve zvláštním textu [o CSS pixelu](#).

Po téhle exkurzi do technické teorie se podíváme na její praktické využití v meta značce pro viewport.

Meta značka pro viewport: Vše, co o ní potřebujete vědět

Kdybyste ale moc stáli o podporu všech starších a méně významných kontextů (iOS 8, Windows Phone), volte spíše následující verzi.



Obrázek: Po vložení meta značky pro viewport se z desktopového rozlišení stane mobilní. Zjednodušeně řečeno. Ale prsty v tom mají viewporty

Bez použití meta značky se web vykreslí do výchozího layoutového viewportu, který má většinou šířku 980 pixelů. Web bude vypadat „jako na počítači, jen zmenšený“. S použitím meta značky pro viewport se šířka layoutového viewportu nastaví na velikost rozlišení v CSS pixelech.

Jednoduchá varianta

Dnes už nebude takový problém, pokud použijete následující zápis:

```
<meta name="viewport" content="width=device-width">
```

Když byste ale moc stáli o podporu všech starších a méně významných kontextů (iOS 8, Windows Phone) volte spíše následující verzi.

Varianta s podporou všech zařízení

V HTML hlavičce:

```
<meta name="viewport"  
      content="width=device-width, initial-scale=1">
```

Bez `initial-scale=1` totiž Safari na iOS 8 a starších renderuje stránku do rozlišení, jako by bylo otočené na výšku, i když jej používáme na šířku.

Teoreticky dělá `initial-scale=1` na všech zařízeních totéž co `width=device-width`, ale bez toho druhého chybí Internet Explorer na mobilních Windows 8 stejným způsobem jako osmá a starší verze mobilního operačního systému od Applu.

Jak už jsem na začátku textu naznačil, svět se nezboří, když na tohle zapomenete. Mobilní Windows jsou v roce 2018 i z pohledu uživatelské základny prakticky mrtvá platforma a iOS 8 a starší jsou podobný případ.

Parametry meta značky pro viewport

Do atributu `content` je možné dávat různé vlastnosti a jejich

hodnoty.

width

Nastaví šířku layoutového viewportu v pixelech. Nejčastěji využívaná hodnota `device-width` sjednotí šířku layoutového viewportu se šírkou ideálního viewportu. Takže uživatel nebude muset zoomovat a vaši responzivní stránku uvidí jedna k jedné. Pokud použijete hodnotu, např. `width=400`, nastavíte šířku layoutového viewportu na 400 pixelů. Nenapadá mě ale moc rozumných důvodů, proč to dělat.

initial-scale

Nastaví výchozí zoom, ale také šířku layoutového viewportu. Ve výsledku dělá zápis `initial-scale=1` totéž jako `width=device-width`. Jak už jsem psal: Pokud chcete maximální kompatibilitu, uvádějte oba dva.

user-scalable

Hodnota `no` zakazuje uživateli jakkoliv zoomovat. Prosím, nepoužívejte ji. Zoomování je na mobilních zařízení fakt potřeba. Ať už jde o zvětšení textu v horších světelných podmínkách, nebo jen touhu vidět detaily z nějakého obrázku, přibližování obsahu prostě potřebují všichni uživatelé. Safari na iOS 10 a novějších navíc zákaz zoomování úplně ignoruje. Více o tom píšu [v kapitole o častých chybách](#).

minimum-scale/maximum-scale

Minimální a maximální možný zoom. `maximum-scale=1` ruší možnost přiblížení stejně jako `user-scalable=no`. Opět naléhám – prosím, nepoužívejte to.

shrink-to-fit

Pokud nějaké prvky pozicujete částečně mimo viewport (například pomocí `position: absolute`), na zařízeních s iOS se vizuální

viewport připočítá tak, aby se zobrazil i onen pozicovaný element.

Může se ale stát, že si to takhle nepřejete. Třeba jen chcete, aby byl element částečně oříznutý a mimo viewport. Od iOS 9 můžete použít deklaraci `shrink-to-fit=no`, kterou to zařídíte.

Hezky je to vysvětlené na Stack Overflow, i s ukázkou v CodePen:
[stackoverflow.com/a/33949647/889682](https://codepen.io/alexpapadimitriou/pen/33949647)

Váš meta tag pro viewport by pak měl vypadat takto:

```
<meta name="viewport"  
      content="width=device-width, initial-scale=1.0, shrink-to-  
      fit=no">
```

viewport-fit

Tohle je nová vlastnost, která řeší způsob zobrazování na zařízeních s jinou než hranatou obrazovkou. Jako příklad vezměme chytré hodinky nebo iPhone X a novější. Vlastnost může mít následující hodnoty (už znáte z `background-size`):

- `auto` – výchozí stav, který vše nechává na prohlížeči. U iPhone X a novějších to například odpovídá hodnotě `contain`.
- `contain` – zmenší viewport pro stránku tak, aby byla vidět celá. Jakou barvu vykreslí po stranách, záleží na prohlížeči. U nových iPhonů je to `background-color` z body.
- `cover` – roztahne viewport pro stránku tak, aby nikde „nevýčuhovaly“ neobarvené části rozhraní prohlížeče. S tím rizikem, že kulaté rohy nebo výčnělky na displeji zařízení některé části stránky překryjí.



Pokud má stránka různobarevné pozadí, jako je to u Vzhůru dolů, hodí se do meta značky přidat `viewport-fit=cover`

Více o tom píšu na blogu v článku o iPhone X. vrdl.cz/p/iphone-x

Stručné řešení pro vaše weby: Pro layout s jednobarevným pozadím si jen zkontrolujte nastavení `background-color` na `body`. Pro weby s různobarevnými prvky zabírajícími celou šířku si přidejte parametr do meta značky pro `viewport`:

```
<meta name="viewport"  
      content="width=device-width, viewport-fit=cover">
```

Tipy, triky, zajímavosti

Meta `viewport` raději moc nenastavujte Javascriptem

Hodí se to, jen když nemáte přístup do `<head>`. Teoreticky jde Javascriptem meta značka pro `viewport` i měnit, ale nedělejte to. Je to náročné na překreslování stránky. Vyrobněte raději normální responzivní web s jedním meta tagem pro `viewport`.

Odstanění prodlevy mezi tapnutím a akcí trvající 300 ms

Když budete mít viewport nastavený správně, s hodnotou `width`, aktuální prohlížeče postavené na jádřech WebKit a Blink samy odstraní prodlevu mezi tapnutím a akcí. Starší prohlížeče totiž po tapnutí prstem čekaly, zda nepřidáte prst druhý a nemáte tedy v úmyslu stránku zvětšovat. Více si o tom můžete přečíst na blogu vývojářů WebKitu. vrdl.in/172eg na blogu vývojářů WebKitu.

Zavináčové pravidlo @viewport v CSS

Instrukce pro způsob zobrazování by se měla dávat do CSS, že ano? S logičtěji umístěným zápisem `@viewport { }` přišlo W3C, ale moderní prohlížeče jej zatím nezvládají. Výjimkou je Internet Explorer 11 a Edge, kde je to ale potřeba zapnout. Pravidlo tak využívá jen IE11 v takzvaném „snap“ módu na desktopových Windows. V roce 2018 to tedy podle mě k ničemu není. Psal jsem o tom ve starším článku. vrdl.cz/p/viewport-windows

Weby na WatchOS – pokud máte web optimalizovaný pro viewporte menší než 320px

Chytré hodinky od Applu vynucují zobrazení našich webových dílek na zápěstí uživatelů ve viewportu širokém 320 CSS pixelů. Pokud bychom tomu chtěli zabránit a zobrazit je ve výchozím CSS rozlišení (o šířce 272 nebo 312 pixelů podle typu hodinek), musíme si dupnout následujícím kódem:

```
<meta name="disabled-adaptations" content="watch">
```

Vtipné je, že WatchOS ve výchozím režimu vynucuje přepočítaný viewport uvnitř přepočítaného viewportu. Ale co už – my léta víme, že viewporte na mobilních zařízeních jsou jako teorie relativity: Víme, že existují, víme že jsou složité, ale skoro nikdo jim nerozumí.

Více o [webech na WatchOS](#) píšu ve zvláštní kapitole.

Velikost okna prohlížeče v CSS a JavaScriptu: min-width, innerWidth, clientWidth

Na rozdíl od rozlišení displeje poskytuje velikost okna pro responzivní design webu velmi zajímavé informace.

Kromě velikosti okna můžeme také mluvit o velikosti viewportu. A protože existují minimálně dva různé viewporty, vizuální a layoutový, prohlížeče vracejí v různých vlastnostech různé hodnoty.

JavaScript

Prostřednictvím JavaScriptu se nabízejí dvě možnosti. Vybíráme v zásadě podle toho, zda chceme připočítávat rozměry lišty pro rolování stránky.

Velikost okna s posuvníkem: innerWidth a innerHeight

Vrací šířku a výšku okna v CSS pixelech včetně rolovátka (posuvníku nebo „scrollbaru“), pokud je přítomné.

```
window.innerWidth  
window.innerHeight
```

Kromě okna (`window`) je možné vlastnosti aplikovat i na další podobné objekty: rámy (`frame`) nebo skupinu rámů (`frameset`).

Pokud bychom se bavili o viewportech, těmito vlastnostmi z prohlížeče vytáhneme rozměry vizuálního viewportu.

V Internet Exploreru to funguje od verze 9, jinak ve všech relevantních prohlížečích. Více informací je na MDN.

mdn.io/Window.innerWidth

Velikost dokumentu bez posuvníku: clientWidth a clientHeight

Vrací šířku a výšku prvku v CSS pixelech *bez rozměru rolovátky („scrollbaru“)*, pokud je přítomné.

```
document.documentElement.clientWidth  
document.documentElement.clientHeight
```

Kromě dokumentu je možné vlastnost aplikovat na jakýkoliv jiný HTML prvek, který má layout alespoň typu inline.

Pokud bude řeč o viewportech, vlastnostmi **clientWidth** a **clientHeight** z prohlížeče vytáhneme rozměry *layoutového viewportu*. Raději připomenu, že pokud na mobilních zařízeních zapomeneme na meta značku pro viewport, vrátí nám rozměry výchozího viewportu.

Funguje to ve všech prohlížečích. Více informací je na MDN: mdn.io/Element/clientWidth.

Posuvník, iOS a vyzkoušení

Většina dnešních operačních systémů „scrollbary“ na stránce schovává. Hodnoty **innerWidth** a **clientWidth** aplikované na okno nebo dokument budou díky tomu shodné.

Jednou z podstatných výjimek jsou desktopové Windows, které posuvníky zobrazují a do **clientWidth** pak nepočítají.

Dalším viditelným odlišením je chování prohlížečů na iOS: Pokud je stránka díky přetečení textu širší než viditelný viewport, Safari počítá **clientWidth** pro celý dokument. Chrome na Androidu naproti tomu obě hodnoty ponechává stejné.

Rozdíly mezi **innerWidth** a **clientWidth** si můžete vyzkoušet také na

mém CodePenu: cdpn.io/e/rrXNWO.

Pro zjištění viewportu jsou výhodnější vlastnosti `innerWidth` a `innerHeight`, protože na všech platformách dostanete shodně vypočtená čísla. Pokud potřebujete zjistit velikost prostoru dostupného pro váš design, musíte pracovat s `clientWidth` a `clientHeight`.

CSS

V CSS máme Media Queries, kterými se na šířku nebo výšku ptáme například takto:

```
/* Minimální šířka */  
@media only screen and (min-width: 30em) { ... }  
/* Maximální výška */  
@media only screen and (max-height: 10em) { ... }
```

Tyhle hodnoty se v CSS počítají jako velikost viewportu, tedy včetně šířky posuvníků.

Weby na chytrých hodinkách Apple Watch

Mezi zařízeními, které nějak zobrazují weby, už určitou dobu figurují i chytré hodinky. Ale má smysl se jimi v responzivním designu zabývat? Ano, minimálně od téhle chvíle určitě.

Budou lidé weby používat na chytrých hodinkách?

Občas dostanu obrázek webu Vzhůru dolů z nějakých hodinek, ale bral jsem to spíše za kuriozitu a masovému používání nevěřil.

Do první verze knížky „Vzhůru do (responzivního) designu“ jsem

ostatně napsal:

*Myslím, že je to nepoužitelné, a masovému přijetí nevěřím.
Plocha je pro weby příliš malá na konzumaci obsahu,
natož pak rozumnou interakci s ním.*

Jenže hodinky se zvětšují a způsob jejich ovládání je stále pohodlnější. Stále sice platí, že nevěřím, že používání webů na hodinkách bude *masovou* záležitostí, ale v tom ostatním jsem se pravděpodobně mylil.

Do hry už vstoupil i Apple s hodinkami Watch, respektive operačním systémem na nich – watchOS.

Píšu o tom proto, že Apple je známý tím, že kroky nemívá nepodložené reálnými testy uživatelů a jejich chování. To je něco jiného, než když menší výrobce hodinek do systému *narve* prohlížeč, protože – no proč ne, že?

Počínaje tímto krokem Apple (platí od roku 2018 a watchOS verze 5) je potřeba věřit tomu, že lidé weby na hodinkách nějak používat budou. Rozdíl oproti mobilům nebo desktopu bude pravděpodobně v intenzitě toho používání. V textu vycházím hlavně z prezentace „Designing Web Content for watchOS“ přímo od Applu.
vrdl.in/webwatchos

Ale čtěte dále, všechno se dozvíte.

Pohledem uživatele

Prohlížeč na WatchOS není nainstalovaný jako samostatná aplikace. Jde tam spíše o možnost otevírat sdílené odkazy:

- V aplikacích Mail a Messages vám může přijít odkaz na web.

- Odkaz pak můžete otevřít v prohlížeči, přičemž stránka je v něm uměle upravena.

Stránku je možné základním způsobem používat:

- posouvat stránku dotykem nebo tlačítkem „korunky“,
- dvojitým tapnutím přiblížit obsah,
- posunovat se vpřed a vzad v historii prohlížení (gesto švihnutí od kraje obrazovky),
- číst ve čtenářském režimu (Reader Mode),
- používat jednoduché formuláře.

Technicky

- Prohlížeč je Safari, postavený na Webkitu.
- Některé vlastnosti jsou vypnuty: zmiňuje se hlavně video, Service worker, webové fonty...

Výchozí stav prohlížení: vynucený initial-scale

Pokud to dobře chápou, prohlížeč se snaží weby za každou cenu zobrazit v šířce viewportu 320px. Nejvíce proto, že Apple nevěří, že jsou naše díla připravená na menší rozlišení. Dobře dělá.

Technicky to funguje následovně:

- Vynutí hodnotu vlastnosti initial-scale meta značky pro viewport na 0.49.
- Prohlížeč pak hlásí rozlišení 320×357 (v CSS pixelech).



Obrázek: Jak fungují výchozí adaptace webů na watchOS 5. Zdroj: Apple

Vypnutí výchozího stavu a meta značka disabled-adaptations

V případě, že jste si jistí, že váš web zvládne i menší rozlišení, lze vynucenou adaptaci vypnout:

```
<meta name="disabled-adaptations" content="watch">
```

V takovém případě nebude Safari na watchOS provádět výchozí adaptace a pracovat s běžným rozlišením v CSS pixelech:

- 272 × 340 px pro 38mm hodinky Apple Watch
- 312 × 390 px pro 42mm hodinky Apple Watch

Pojďme se podívat i na další použité technologie. Není to vlastně nic nového a je dobré, že Apple zůstal u standardního a jinde zavedeného kódu.

Následující kousky kódu vám doporučuji přidat na všechny veřejně dostupné obsahové weby. „Riziko“ zobrazení na chytrých hodinkách totiž od letoška poroste.

Open Graph pro náhled obrázku

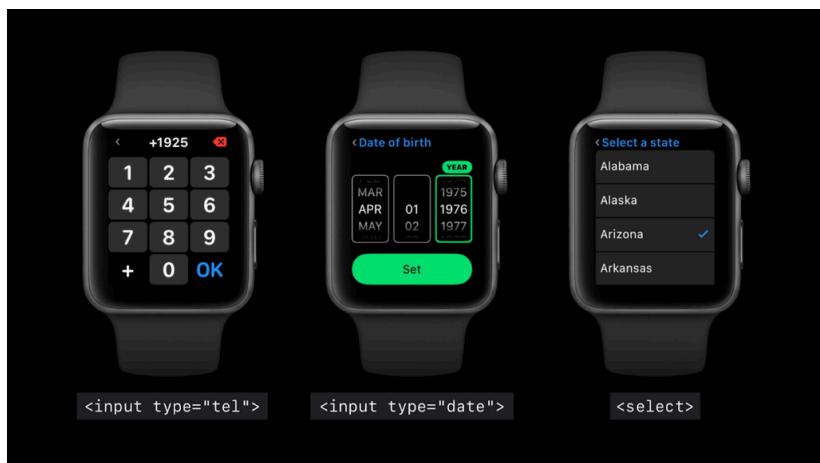
Abyste dosáhli hezkého náhledu odkazu v aplikacích pro práci s e-maily a chatování, přidejte tyhle dvě meta značky:

```
<meta property="og:title" content="Titulek stránky">  
<meta property="og:image"  
content="https://www.example.com/nahledovy-obrazek.jpg">
```

Předpokládám ale, že je už na webu máte, a to kvůli zobrazení náhledů na sociálních sítích nebo v chatovacích aplikacích.

Formuláře

Mile mě překvapilo, že na hodinkách od Applu je možné dělat také interakce s formuláři. Když se ale podíváte na následující obrázek, dává to smysl. Uživatelský vstup je vyřešený vážně hezky:



Obrázek: Formulářové prvky na watchOS 5. Zdroj: Apple

Jako webmasteři jen musíte použít správné typy pro značku `<input>` nebo nativní prvek pro výběr `<select>`:

```
<input type="tel">  
<input type="email">  
<input type="date">  
<select>
```

Opět předpokládám, že to pro vás není nic nového a přidáváte to do každého formuláře, neboť to je přístupné řešení pro všechny možné platformy.

Apple navíc doporučuje přidávat popisek `aria-label` (také standardizovaný), aby uživatel viděl, jaké políčko vyplňuje, když se mu ovládací prvky zobrazí přes celou šířku stránky:

```
<input type="email" aria-label="E-mailová adresa">
```

Mód čtení

Na delších textových stránkách se Safari na watchOS automaticky přepíná do „Reader Mode“. V něm vnučuje stylování stránce tak, aby se dobře četla. To opět není nic nového, protože podobný režim má Safari na všech zařízeních. Jen ho nevnučuje.

Webmasteri mohou pro lepší uživatelský prozitek udělat následující:

- Zavřít hlavní obsah do značky `<article>`.
- K sémantickým informacím o článcích přidat atributy `itemprop`. Vypadá to, že uznávané jsou hodnoty `title`, `author`, `subheading` nebo `pubdate`.
- Používat sémanticky správné HTML značky pro obsah: ``, ``, `<blockquote>`, `<figure>` a `<figcaption>`.

Zapamatujte si

- Layoutový `viewport` je plocha stránky.
- Vizuální `viewport` má šířku a výšku obsahu okna prohlížeče.

- „CSS rozlišení“ je ideální viewport.
- Správné nastavení viewportu v meta značce: `width=device-width`. Ale je to složitější.
- V Javascriptu máme několik možností, jak zjistit velikost okna. Vzpomenete si na vlastnosti, které vracejí velikost okna s posuvníkem?
- Nezapomeňte na ošetření zobrazování webů na chytrých hodinkách.

Kapitola 5: Rychlost načítání

Hrozně rád vařím. Dělám to ale relativně málo, takže jako každý amatér nemám rutinně zvládnuté pracovní postupy a v hlavě všechny recepty. Občas se mi proto stane, že na něco podstatného zapomenu. Jednou v jídle chybí sůl. Jindy čekám, až vykyne těsto bez kvasnic. Prostě zapomnětlivý kuchař.

Lidé od webu se jako zapomnětliví kuchaři chovají v jednom kuse. Na suroviny, které měly být neoddělitelnou součástí už úvodního plánování, si s vyděšeným výrazem ve tváři vzpomenou, až když už je pozdě. Spustí web, a pak to přijde:

- Přemýšlejí, jak získat návštěvnost.
- Nervózně čekají na obsah od klienta.
- Začnou si lámat hlavu nad přístupností pro slabozraké.
- Přikážou kodérovi připravit mobilní verzi.
- Po spuštění webu se vrhnou na „optimalizaci“ rychlosti načtení.

Jenže přátelé, neexistuje nic jako tlačítka s nápisy typu „zapnout SEO“ nebo „zapnout rychlosť“ na hlavě webařky nebo webaře.

Tyto a další aspekty dobrého webu musejí být promýšleny jako nedílná součást hned první, výzkumné a definiční fáze webového projektu. Povídali jsme si o ní v [kapitole o tvorbě webu](#), pokud si vzpomínáte.

Ale zpět k jednomu z těch typických „last minute“ témat, totiž k rychlosti načítání:

1. Ukážeme si, proč je dobré se rychlostí zabývat.

2. Povíme si o Speed Budgetu – maximálních parametrech rychlosti, které svému webu dovolíte.
3. Je tu spousta nástrojů, které vám s hledáním problémů na vašem webu pomohou.
4. Dám vám tipy na konkrétní kroky pro zrychlení webu.

Jdeme na to. Proč nezapomínat na kvasnice a rychlosť načítania?

Rychlostní limity. A proč nedělat „optimalizace“?

Rychlostní limit je metrika, která říká, jaké maximální hodnoty parametrů rychlosti mají mít vaše vstupní stránky. V anglických textech se mluví o „Speed Budgetu“ nebo „Performance Budgetu“.

Jak na nastavení rychlostních limitů?

Rychlostní limity si vložte do tabulky. Kolegům nebo externímu dodavateli pak dejte za úkol jejich dosažení.

1) Vyberte důležité vstupní stránky

Rychlosť je zásadní hlavně u stránek, na které lidé chodí z marketingových kanálů. U e-shopů to bude šablona detailu produktu a jejich seznam. Možná homepage. Málo důležité stránky řešit nemusíte.

2) Vyberte, jaké parametry budete hodnotit

WebpageTest.org. Testoval bych na něm. Nejdůležitější hodnota je *Speed Index*. Další pak *Load Time*, *First Byte*. V textu o nástrojích se dozvítí více.

3) Otestujte parametry nejbližších konkurentů

Seznam konkurentů už nejspíš máte. Ale pokud ne, rychle to napravte. Uložte si je do tabulky i s hodnotami u konkurentů a nastavte si cílové metriky tak, aby byly alespoň o 20 % lepší než u nejrychlejšího webu konkurence. O sestavování Speed Budgetu pěkně píše Daniel Mall v článku „How To Make A Performance Budget“. vrdl.in/perfbudget

Pak už to vše stačí jen zrealizovat, že?

Optimalizace vs. kultura rychlosti

Nemám rád slovo „optimalizace“. Abyste totiž něco mohli „optimalizovat“, musíte to předtím pokazit.

Pokud na rychlosť nemyslíte už v úvodních fázích práce na projektu, bude „optimalizace“ náročná. Když ji má projekt v DNA, je to ve výsledku jednodušší.

Stanovení rychlostních limitů velmi pomůže. Všechny nové vlastnosti, které různí členové týmu běžně do projektů přidávají, se zvažují i pod úhlem limitů.

Klient: „Že bychom na houmpejdž dali automaticky pouštěné videjko přes celou stránku?“ Designér: „OK, ale překročí vám to rychlostní limit. Šestnáctkrát.“

Je důležité, aby vývojáři byli přítomní už v oné přípravné fázi projektu. Alespoň pro konzultace, ale ideálně i s možností zúčastnit se přípravy prototypů. Webový projekt je stejně technický jako designérský. Designéři a marketéři obvykle neznají technická specifika komponent zvažovaných k implementaci. Vývojáři jim rádi připraví testy.

Kontrola dodržování rychlostních limitů by se měla provádět nejen během práce na webu, ale také po každé větší aktualizaci. Dá se to naštěstí zautomatizovat. Buď nějakým vlastním nástrojem, který vytáhne data z analyzátorů, o nichž mluvím v dalším textu, nebo třeba pomocí specializované nástroje Speed Curve. speedcurve.com

Pojďme si teď ukázat nástroje pro analýzu rychlosti.

Proč řešit rychlosť načítání webu?

V Google mají jasno. Tohle prohlásil v roce 2010 Eric Schmidt, tehdejší šéf firmy:

Never underestimate the importance of fast.

Ale nemusíte být Google. Rychlejší načtení vylepší uživatelský prožitek a pomůže tak cílům každého webu.

My lidé to zkrátka rychlé chceme

Uživatelé jsou prostě jen lidé. Jejich trpělivost při čekání na reakci rozhraní po provedení akce není nekonečná.

- Do 100 milisekund: považujeme za okamžitou reakci.
- Do jedné vteřiny: poznáme prodlevu, ale nepřerušíme úkol.
- Nad deset vteřin: ztrácíme pozornost a máme tendenci začít plnit jiný úkol.

Více se dozvíte v článku „Response Times: The 3 Important Limits“ od Jacoba Nielsena. vrdl.in/4o3d7

Rychlosť načtení stránky má vliv na úspěšnost webu

Říkají to alespoň studie a testy, které o tom byly publikovány.

Walmart si například spočítal, že každé zrychlení načtení stránky o vteřinu zvýší konverze jejich webu o dva procentní body. To nebude úplně špatný výdělek, že ano? vrdl.in/ziud9

I v Česku rychlosť zabírá. Portálu Srovname.cz jsem osobně pomáhal se zrychlením důležitých vstupních stránek. Poměrně jednoduchými kroky jsme na mobilních zařízeních zvýšili konverzní poměr o čtvrtinu. vrdl.cz/b/58-rychlosť-srovname-cz

Další analýzy dokazují, že se zrychlení webu pozitivně promítá do všech klíčových ukazatelů: konverzního poměru, počtu zobrazených stránek, počtu návštěv nebo třeba spokojenosti zákazníků. Mnoho studií ze světa najdete na WPO stats. wpostats.com

Rychlosť ovlivňuje řazení inzerátů AdWords a přirozených výsledků ve vyhledávání přes Google

Ano, rychlosť načítání je jedním z řadicích parametrů AdWords kampaní. Už od roku 2008. Na oficiálním blogu AdWords se píše:

Pokud výrazně vylepšíte čas načtení vaší cílové stránky, měli byste vidět lepší Quality Score a nižší nabídky pro minimální cenu za proklik (CPC).

Cituji z článku „Landing page load time now affects keywords' Quality Scores“. vrdl.in/9w2xd

Google říká, že rychlosť webu má vliv na zobrazování ve výsledcích vyhledávání. Na pořadí ve výsledcích má přímý vliv hlavně rychlosť

vygenerování stránky na serveru. Zájemce odkážu na detailní popis na Moz.com. vrdl.in/zhrkp

Od poloviny rok 2018 je také rychlosť nově řadícím signálem v mobilních výsledcích vyhledávání. Google to píše v blogpostu „Using page speed in mobile search ranking“. vrdl.in/f7b80

Rychlosť tedy ovlivňuje nejen konverze, ale i pravděpodobnost, že se na web člověk vůbec dostane.

Rychlosť mobilních sítí se nikdy nevyrovnaná pevnému připojení

Ani zvyšování přenosové rychlosti na 3G/4G sítích nesrovná načítání webů na mobilech s připojením v kancelářích a domácnostech. Proč? Klíčový problém je latence, tedy doba, kterou v průměru trvá spojení mezi serverem a klientem.

Mrkněte se na tuhle tabulku. Je inspirována knihou Ilji Grigorika, High Performance Browser Networking.

Připojení	Max. rychlosť Mbit/s	Latence ms
2G/EDGE	0,1–0,4	300–1000
3G	0,5–5	100–500
4G/LTE	1–50	< 100
WiFi	1–50	< 10

Latence je průměrné zpoždění mezi požadavkem serveru a odpovědí prohlížeče. Doručení každého ze souborů, které web potřebuje, latency zdrží

Zatímco ideální přenosová rychlosť mobilního připojení se může

rovnat pomalejším „wifinám“, zpoždění (latence) pro zaslání každého ze souborů bude vždy horší. Vysvětlení je nasnadě – architektura mobilního internetového připojení je daleko složitější než u toho pevného. Data jdou vzduchem, že ano.

Česko je už sice hezky pokryto LTE sítěmi, jenže síť čtvrté generace zdaleka nemají a jen tak nebudou mít v mobilu všichni.

lte.ctu.cz/pokryti/

Takže: ano, rychlosť načtení je dobré řešit. Pravděpodobně vám na web pomůže dostat více lidí a také vylepší klíčové ukazatele hodnocení jeho úspěšnosti.

Rychlosť je důležitá. Jak ji ale dostat do vašich projektů?

Nástroje pro analýzu rychlosti načtení stránky

Začněte s Google PageSpeed Insights nebo Google Analytics, pokračujte na WebPageTest.org. Vývojářům se také budou hodit vývojářské nástroje v Chrome nebo jiných prohlížečích.

Google PageSpeed Insights

Validátor základních technických problémů, které komplikují rychlosť webu. Zde začněte. Otestujte si tady všechny důležité vstupní šablony.

The screenshot shows the Google PageSpeed Insights interface. At the top, it says "PageSpeed Tools > Insights". Below that are tabs for "PRÍRŮČKY", "REFERENCE", "UKÁZKY", and "PODPORA". The URL "http://vzhuruodolu.cz/" is entered in the address bar, and there's a "ANALYZOVAT" button. The main area shows a score of "32 / 100" for "Souhrn návrhů". There are three sections with suggestions:

- Měli byste opravit:**
 - Zkratit dobu odezvy serveru
 - Ukázat postup opravy
- Zvažte možnost opravy:**
 - Eliminujte v obsahu na okrajem kód JavaScript a CSS blokující vykreslení.
 - Ukázat postup opravy
- Využijte načítání do mezipaměti prohlížeče:**
 - Ukázat postup opravy
- Minifikujte kód HTML:**
 - Ukázat postup opravy
- Počet úspěšně splněných pravidel: 6**
 - Zobrazit podrobnosti

A note at the bottom says: "Uložení výsledků do mezipaměti trvá 30 sekund. Pokud jste na stránce provedli změny, vyčkejte 30 sekund a teprve poté test spusťte znova."

Google PageSpeed Insights zobrazí skóre webu, ale také rovnou návrhy na vylepšení

Dokud v PageSpeed Insights (PSI) nedosáhnete skóre dejme tomu kolem 80 bodů na desktopu i mobilu, nemá smysl učit se další nástroje. Vyřešit je potřeba hlavně červeně zvýrazněné problémy vašeho webu.

Zároveň není nutné bojovat za dosažení stovky. PSI je prostě jen automat, a tak vám strhne body i za rozumné věci. Upozorní vás například na „špatně“ nastavenou dobu kešování měřicího skriptu Google Analytics. Jinak je to ale skvělý nástroj.
developers.google.com/speed/pagespeed/insights

Google Lighthouse

Nepostradatelný nástroj pro analýzu rychlosti, ale také přístupnosti, technického SEO a obecných doporučení.

Je možné jej používat přímo v Google Chrome (DevTools a záložka

„Audits“), ale také mnoha dalšími způsoby. Online verze je zde: developers.google.com/web/tools/lighthouse/run.

Velmi doporučuji právě Lighthouse pouštět pravidelně a řídit se jeho doporučeními.

Google Analytics

U statistik z Google Analytics se mi líbí, jak jsou po ruce marketákům. Mají ale velmi zajímavé využití i pro vývojáře, hlavně když se rozšíří o Trackomatic a Technical Performance Dashboard. vrdl.cz/p/google-analytics-vyvojari

Na přehledy o rychlosti webu se mrkněte do *Chování > Rychlosť > Přehled*. Je potřeba měřit pomocí aktuální verze: Universal Analytics. Analytics ukazují *Časování stránek* (*Page Timings*), ale napříč různými kontexty, jako jsou prohlížeče nebo regiony.

V *Časování uživatelů* (*User Timings*) mohou být vaše vlastní měření – např. jak rychle se načetl konkrétní obrázek. Je to potřeba nastavit. vrdl.in/f3rbx

Standardně se právě pro měření rychlosti používá jednoprocenční vzorek zhlédnutí vaší stránky. Pokud to chcete jinak, je potřeba měřit s nastavením ‘`siteSpeedSampleRate`’: [50.](http://vrdl.in/4bn30) vrdl.in/4bn30

WebPagetest.org

Pro mě nástroj číslo jedna. Dělá pokročilou analýzu a testuje detailněji než Page Speed Insights. Na druhou stranu je průběžné používání WebPagetestu náročnější, protože testy nějakou dobu trvají. WebPagetest také nevede za ruku jako PSI, nedává přímé

rady, jak problém odstranit. A je potřeba jej trochu více poznávat, jelikož rozhraní nepatří mezi nejpřívětivější.

Umožňuje testování z jiné lokality, testování pomalého připojení a v prohlížečích, ve kterých nemáte pokročilé vývojářské nástroje. Třeba v těch mobilních nebo ve starých Internet Explorerech.

The screenshot shows the WebPageTest.org interface. At the top, there's a navigation bar with links for HOME, TEST RESULT (highlighted), TEST HISTORY, FORUMS, DOCUMENTATION, and ABOUT. Below the navigation is a banner for a test of 'Web Page Performance Test for www.srovname.cz' from Prague, Czech Republic - Chrome - Emulated Nexus 5 - 3GSlow - Mobile at 3/21/2015, 3:34:31 AM. The banner includes a 'Need help improving?' link. Below the banner, there are several performance icons: First Byte Time (A), Keep-alive Enabled (A), Compress Transfer (A), Compress Images (F), Cache static (C), and Effective use of CDN (X). A green button labeled '2' points to a 'Speed Index' section which compares 'Document Complete' and 'Fully Loaded' times across 'First View' and 'Repeat View'. A blue button labeled '3' points to a 'Waterfall' chart showing the sequence of requests and their timing. A green button labeled '4' points to a 'Filmstrip View' showing a visual representation of the page's loading stages.

WebpageTest.org: kromě vysvědčení (1) vidíme i Speed Index (2), vodopádový pohled (3) nebo se můžeme prokliknout na filmový pás (4)

Má také API. Je omezené na pár stovek dotazů týdně, což ale vystačí skoro všem.

Co z výsledků WebPagetest mě zajímá?

1) Vysvědčení

Hodnocení v parametrech, které připadají autorům WebPagetest důležité. *First Byte* (jak rychlý je server), nastavení kešování na serveru, komprese obrázků a využití CDN.

2) Speed Index

Speed Index je průměrný čas zobrazení konkrétní stránky v daném prohlížeči, velikosti okna a rychlosti internetu. Čím nižší je váš Speed Index, tím lépe. Nejrychlejší weby dosahují čísel kolem tisícovky. Průměr je mezi pěti a desíti tisíci. Čísla nad deset tisíc jsou na pováženou.

Speed Index je esence rychlosti načítání. Číslo, které můžete porovnávat s konkurenty nebo ho porovnávat před optimalizací a po ní. Takový „pagerank“ pro odborníky na zrychlování webu.

3) Vodopád načítání (Waterfall)

Jak se stahují komponenty stránky? Které z nich blokuje parsování? Užitečný a detailní pohled na postup načítání.

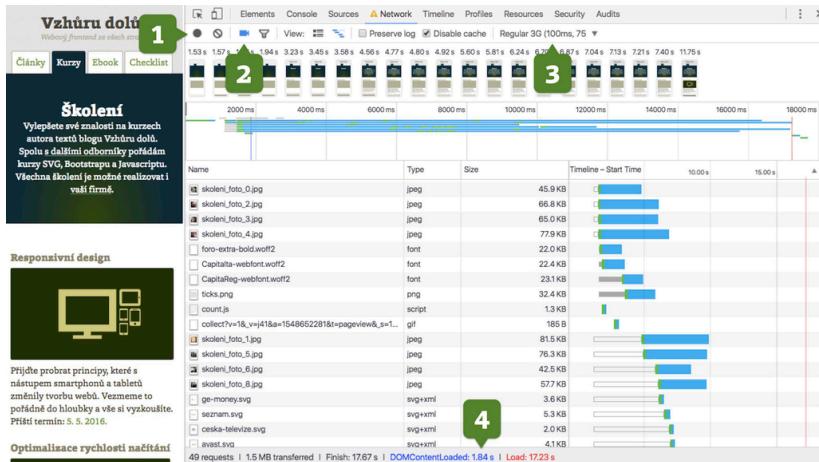
4) Filmový pás (Filmstrip View)

Jak přesně vizuálně probíhalo načítání stránky? To je velmi důležité. Dívám se, jak dlouho trvalo, než se uživateli zobrazil obsah, a také jakým způsobem vykreslování probíhalo. Je to užitečnější než strojově vypočtený *Start Render*, který vidíte v tabulce.

Dívám se samozřejmě i na další čísla. WebPagetest má mnoho zákoutí k prozkoumání, tohle byl ale dobrý začátek. webpagetest.org

Chrome DevTools

Pokročilá analýza a detailní testování procesů načítání v Chrome. Velmi podobný nástroj mají i ostatní moderní prohlížeče.



Chrome DevTools a analýza rychlosti

Vše podstatné je v záložce Network:

1. Zapněte nahrávání.
2. Zapněte snímání filmový pásu s průběhem zobrazování webu.
3. Omezte rychlosť připojení.
4. Modrá je událost *DOMContentLoaded*. Ta je spuštěná ve chvíli kdy bylo HTML načteno a rozparsováno. Červená událost *Load* se spustí, jakmile prohlížeč stáhl úplně vše, včetně obrázků.

Video: [Chrome DevTools: analýza načítání stránky ~ Praktická demonstrace analýzy rychlosti načítání ve vývojářských nástrojích Chrome](#).

GTmetrix

Používám jen jako doplněk, ale někteří mu dají přednost před PageSpeed Insights. Obsahuje totiž analýzu z tohoto nástroje a zároveň ještě dříve velmi známou metodiku YSlow v jednom

reportu.

Umí toho hodně. Ukáže timeline, zvládne emulaci pomalého připojení. Testovací lokality má GTmetrix ale pro ČR horší než WebPagetest.org a s méně možnostmi nastavení. Hezké je monitorování a nastavení připomínek do e-mailu. gtmetrix.com

Tipy pro rychlé načtení webu

Vytvoření rychlého webu vyžaduje širokou škálu dovedností a znalostí. V tomto textu si jen vyklikujeme základní obrysy.

Vždycky musíte zmenšit datový objem prvků stránky, zamezit blokování vykreslování stránky a ideálně ještě zrychlit procesy na serveru. Na protokolu HTTP/1 ještě snížit počet dotazů na server.

Nástroj PageSpeed Insights, který jsem chválil v [textu o nástrojích](#), vás upozorní, pokud váš web následující doporučení porušuje.

1. Zmenšete datový objem

Pravidlo zmenšování dat je evergreen. Platí pro každý web, aplikaci, platí pro HTTP/2 i stále ještě převažující první verzi protokolu HTTP.

Obrázky

Snižte datovou velikost obrázků, zvolte pro ně vhodné formáty.

1. Používejte responzivní obrázky. Píšu o nich v příští kapitole, o [médiích](#).
2. Cokoliv, co lze vyjádřit vektorem (logotypy, ikony, grafy...), uložte do formátu SVG. vrdl.cz/p/svg
3. Pro dekorace používejte CSS, nikoliv obrázky.

4. Používejte kompresi pokročilými nástroji, jako je Kraken.io, JPEGmini.com nebo Guetzli.
5. Zvažte použití formátu WebP namísto JPEG. I když jej umí jen Chrome a Opera, vyplatí se, protože je datově výrazně úspornější (jeho detekci se naučíte v textu o značce <picture>) z následující kapitoly.

Další rady od Google na téma obrázků jsou zde: vrdl.in/od06q

Webové fonty

- Každý font má své řezy (tučný, kurzíva...). Nepoužívejte jich zbytečně moc. Více než pět na jedné stránce je na pováženou. Každý řez zatěžuje stránku desítkami kilobajtů dat navíc.
- Využívejte úsporný formát WOFF2. vrdl.cz/b/50-woff2
- Do fontu dejte jen znaky, které na webu opravdu potřebujete. Vlastnost podporují i Google Fonts. vrdl.in/9763d

Povídání o rychlosti a webfontech od Google: vrdl.in/91bg5

Zdrojové kódy

Zmenšete datovou velikost prvků stránky jako jsou JS, CSS nebo HTML soubory.

- Použijte UglifyJS, CSSNano a podobné nástroje pro automatické zmenšení datového objemu zdrojových kódů.
- Vytvořte si verze velkých knihoven, které obsahují jen používané vlastnosti. Například Bootstrap je možné datově velmi ořezat. getbootstrap.com/customize/

Rady od Google: vrdl.in/kmav0

Lazy loading

Technika, která odkládá načtení datově zatěžujících komponent až na chvíli, kdy je uživatel opravdu potřebuje. Obvykle na moment

narolování stránky na jejich pozici. Zvažte využití téhle užitečné techniky.

Je použitelná pro javascriptové knihovny, vkládané prvky jako videa z YouTube, ale hlavně pro stránky se spoustou obrázků.

vrdl.cz/p/lazy-loading

2. Zamezte blokování vykreslení

Data můžete ušetřit vždy. Dnešní weby ale mívají velké problémy v jiné oblasti: ve špatném způsobu posílání souborů do prohlížeče.

Prohlížeče nedokážou stránku zobrazit, dokud nestáhnou a nezpracují veškerý blokující Javascript (bez parametrů `async` nebo `defer`) a všechny CSS soubory. Na pomalých připojeních to může trvat celou věčnost. Uživatel se pak dívá na bílou obrazovku, i když se mu už spousta prvků stránky načetla.

Javascript

- Používejte parametry `async` a `defer`. Prohlížeči říkají, že na jejich stažení nemusí čekat a stránku může zobrazit už bez nich.
Externí JS soubory bez těchto parametrů nedávejte do hlavičky (`<head>`) nebo těla (`<html>`) stránky. jecas.cz/async-defer
- Rozdělte si Javascript do skupin. *Kritický* kód jako polyfills nebo detekční skripty vkládejte přímo do HTML. Externí *nezbytně blokující* vložte pomocí `<script src="">` rovnou do stránky. *Externí neblokující* pak vkládejte pomocí `<script src="" async>`.
- Pozor na vkládané Javascripty třetích stran. Příkladem budíž měřící kódy nebo A/B testovací nástroje. Pokud je zrovna nevyužíváte, vypínejte je. Zkuste přepsat jejich kód, bývá často velmi špatně optimalizovaný: vrdl.in/3ym50

Další rady od Google: vrdl.in/afzxg

Webové fonty

Když prohlížeče zjistí, že jsou ve stránce webové fonty, většinou na čas schovají obsah stránky. Obvykle je ale u webů vhodnější do načtení webfontů zobrazovat obsah alespoň pomocí systémových písem. Pokud je to i váš případ, doporučím vám knihovnu [FontFaceObserver](#). Zajistí, abyste měli načítání fontů pod kontrolou vy jako autoři. A sjednotí různá chování různých prohlížečů.

[fontfaceobserver.com](#)

CSS

Kritické CSS je implementačně mírně náročnější technika, ale s velkým vlivem na rychlosť zobrazení stránky. Jde o automatické rozdelení CSS na dvě části a vložení té kritické přímo do HTML kódu. [vrdl.cz/b/35-critical-css](#)

Dejte přednost viditelnému obsahu

To, co je na stránce vidět nahoře, se do prohlížeče snažte v externích souborech dostat jako první. Nastavte správné pořadí – využijte vkládání obrázků přímo do HTML nebo CSS. Rady od strýčků a tet z Google: [vrdl.in/aywc5](#)

3. Zrychlete server

Na serveru můžete hodně získat. V celkovém čase potřebném k načtení stránky sice rychlosť serveru hraje menší roli než zpracování frontendu, je ale velmi důležitá. Jak už jsem psal: Hledí na ni Google a považuje ji za jeden ze signálů pro pozici stránky ve výsledcích vyhledávání. Serverové technologie nejsou součástí mé specializace, proto alespoň velmi stručně.

Zrychlete serverový čas

Já vím, dobře se to řekne. Ale renderování stránky přes vteřinu je

velmi nepříjemné a určitě na serveru nějaké rezervy máte. Rady od Google: [vrdl.in/ivy17](#)

Zapněte si gzip

Zdaleka není samozřejmost. Přitom je to jednoduché. Zkontrolujte si to hlavně na sdíleném hostingu. Rady od Google: [vrdl.in/1g9j8](#)

Zajistěte správné kešování v prohlížeči

To je složitější, ale prvky stránky mají na serverech často zapnutá špatná kešovací pravidla. Rady od Google: [vrdl.in/l0rhz](#)

Vyhnete se zbytečným přesměrováním

Každé přesměrování zpomalí dojem z načítání stránky, proto je pokud možno nedělejte. [vrdl.in/46oba](#)

4. Neběží vám server na HTTP/2? Pak ještě minimalizujte počet dotazů

Ve dnech, kdy toto píšu, jsou weby běžící na HTTP/2 ještě stále vzácné. Pro weby servírované pomocí HTTP/1 je proto k uvedeným základním radám potřeba přidat ještě jednu: Minimalizujte počet dotazů na server. Týká se všech externích prvků webu – souborů s CSS a JS kódem, obrázků, webfontů a dalších.

- CSS a JS lze spojovat do balíčků. Najděte si nástroje pro vaši platformu. [vrdl.in/7zcde](#)
- Obrázky zase můžete spojovat do takzvaných „sprajtů“. [jecas.cz/css-sprite](#)
- Menší obrázky můžete vložit přímo do HTML nebo CSS pomocí takzvaného „data URI“. [jecas.cz/data-uri](#)

Na druhé verzi HTTP protokolu to už obvykle není potřeba dělat. Díky vychytávce jménem *multiplexing* je obecně lepší posílat

prohlížeči mnoha malých souborů. Více o HTTP/2, včetně seznamu podporujících hostingů, najdete na Vzhůru dolů. vrdl.cz/p/http-2

To bychom měli. Vytvořili jsme si tady základní mapu problémů, které obvykle zpomalují weby. Máte málo času? Doporučím vám začít s následujícími třemi. Trpí jimi skoro každý web a jejich odstranění bude mít největší efekt.

1. Datová velikost, hlavně obrázků a Javascriptů (bod 1)
2. Javascript blokující vykreslení (bod 2, první odstavec)
3. Webfonty (bod 2, druhý odstavec)

Nebojte, nudit se nebudete. I náprava těchto tří problémů vám nějaký ten den práce zabere.

Jen to, prosím vás, nedělejte jednorázově. Optimalizaci si zautomatizujte a vložte do procesu nahrávání webu na server. Metriky pak kontrolujte průběžně, ideálně opět automaticky.

Zapamatujte si

- Rychlosť „neoptimalizujte“. Zabývejte se jí už na začátku projektu.
- Nastavte si rychlostní limity (Speed Budget) a kontrolujte jejich dodržování.
- Pokud lidé čekají na načtení webu přes deset vteřin, ztrácí pozornost.
- Rychlosť načtení má přímý vliv na úspěšnost webu.
- Kontrolujte weby pomocí Google PageSpeed Insights. Sledujte rychlosť v Google Analytics.
- Zvažte zapnutí HTTP/2 na vašem serveru.
- Redukujte datovou velikost stránky.
- Zkontrolujte, zda web neobsahuje Javascript blokující

vykreslení.

Kapitola 6: Obrázky a další média

Obsah vložený do prohlížeče je z podstaty responzivní. No, nedivte se – je to pravda. Jenže platí pro text, ne už další elementy. Obrázky, tabulky, video... Vsadím se, že takovou věc jste na webu už viděli.

1. Vrátíme se ke kořenům technického pojetí responzivního designu, abychom poznali, proč je potřeba mediálnímu obsahu věnovat takovou péči.
2. Pak se zaměříme na způsoby, jak přinutit mediální obsah k pružnému chování.
3. Zaostříme na obrázky a ukážeme si, jak různým zařízením posílat obrázky podle jejich zásluh.
4. Ukážu vám nejuniverzálnější metodu pro ošetření obrázků, nové atributy značky .
5. Element <picture> možná také neznáte, že?
6. Vektorové SVG donutíme k pružnému chování ve všech prohlížečích, i když v Internet Exploreru to bude docela fuška. Uvažte si silné kafe.
7. A tabulky? Ach, ty responzivní tabulky. Těšte se na fakt hodně možných variant jejich návrhu a implementace.
8. Responzivní grafy vezmeme letem světem, protože tak časté nejsou.
9. Radostně také hlásím návrat ke kódování našeho fiktivního e-shopu. Naimplementujeme si do něj responzivní obrázky.

Marcotteho responzivní design

Původní definice responzivního designu pochází z roku 2010, hlavy a pera Ethana Marcotteho. A je to nadmíru jednoduchá, technicistní myšlenka.

1. Udělejte pružný layout.
2. Doplňte to pružnými obrázky.
3. Přidejte podmínky pro změny layoutu, Media Queries.

Článek z A List Apart nesoucí titul „Responsive Web Design“ je asi nejsdílenější článek v historii webdesignu a k téhle knížce vám asi nic moc nového neřekne. Odkázat na něj ale musím. Ethan Marcotte má můj respekt, protože to vymyslel a zformuloval tak skvěle, že si dnešní webdesign bez „responzivnosti“ už ani neumíme představit.
alistapart.com/article/responsive-web-design



Tři principy responzivního designu: Pružný layout, pružné obrázky, Media Queries

1. Pružný layout

Ethan Marcotte psal článek (a pak ještě knihu) v době, kdy byl nebohý internet přeplněný weby připravenými pouze pro počítače. Skoro všechny měly fixní šířku layoutu, nastavenou v konkrétní pixelové hodnotě. No jen si to představte. Ach, bolí mě už jen to pomyšlení!

Pružný layout naproti tomu mění rozměry podle velikosti okna. Nejčastěji je definovaný v procentech ze šířky okna.

Dnes už bych nikomu nedoporučoval vymýšlet weby jen pro velké displeje. Tehdy to ale nikdo jinak nedělal. V roce 2010 tedy musel Marcotte a jeho tým zbourat něco jako Stalinův pomník tehdejšího webdesignu: Layout fixních rozměrů optimalizovaný pro velké displeje.

O layoutu píšu v samostatné kapitole.

2. Pružné obrázky

Když už máte pružný layout, musíte mu přizpůsobit elementy vevnitř. První věc, která vám ze zmenšeného layoutu vypadne, budou obrázky.

O obrázcích a jiných médiích dále píšu v samostatných textech.

3. Podmínky pro změny layoutu (Media Queries)

Představte si, že na tehdejší počítačový web nasadíte pružný layout a pružné obrázky. Máte? Ted mírně zmenšete okno prohlížeče. Vše funguje hezký, že? Co když ale okno zmenšíte opravdu hodně,

řekněme na velikost malých mobilů? Tam pružný layout nepomůže. Musíme tedy změnit layout nebo ho úplně zrušit. Pomohou nám k tomu Media Queries, které si spolu rozpitváme později, v kapitole o layoutu.

Tímto končíme prohlídku naší kuchyně. Nejzákladnější surovinu, vývar responzivního designu, máme. Další podkapitoly už budou detailně rozebírat responzivní přizpůsobování veškerého možného netextového obsahu.

Principy responzivního webdesignu by se daly ještě zjednodušit. Takhle jej vysvětluji na školeních:

*Vložte obsah do prohlížeče. Zvětšujte nebo zmenšujte okno.
A odstraňujte problémy, které vidíte.*

Ten bonmot je překvapivě pravdivý. Při zmenšování nebo zvětšování okna se musíte vypořádat s nepružnými médiemi, obsahu dát layoutový rámec, vyřešit rychlosť načítání...

Vkládaná média se zachováním poměru stran

Jak zařídit, aby se obrázky, video a prvky vkládané přes `<iframe>` přizpůsobovaly šířce rodičovského elementu a ještě k tomu zachovávaly poměr stran?

Pružné obrázky

To nebude nic složitého. Vezmeme to rovnou od kódu, co říkáte?

```
.content img {  
    max-width: 100%;  
    height: auto;  
}
```

.content img

Selektor by mohl být jednodušší (jen `img`), ale mám ve zvyku pružnost aplikovat jen na obrázky obsahové. Obvykle se nehodí takto ošetřovat například logotyp nebo ikony v hlavičce či patičce stránky.

max-width: 100%

Nastaví šířku obrázku podle šířky rodiče. Neudělá ale ten ošklivý trapas, že by jej roztahoval nad rámec jeho velikosti v pixelech a tím deformoval. To by udělala nezbednější kolegyně mezi vlastnostmi, `width`.

height: auto

Uvést musíme, aby si obrázek zároveň zachoval poměr stran. V opačném případě by se poměrově deformoval.

Vyzkoušet si to můžete na CodePenu: cdpn.io/e/jWebge

Pružné vkládané elementy se zachovaním poměru stran

Vkládáte do stránky kód třetí strany pomocí `<iframe>`, máte tam `<video>` nebo nedejbože Flash v `<object>`?

Většina takto vkládaných elementů třetí strany má jakés-takés responzivní chování zajištěno díky výchově na straně dodavatele kódu. Vám se ale může stát, že i u nich potřebujete zachovat poměr stran.

```
.rwd-object {  
    position: relative;  
    height: 0;  
    padding-bottom: 60%; /* Udává poměr stran */  
}  
  
.rwd-object-in {  
    position: absolute;  
    width: 100%;  
    height: 100%;  
}
```

V HTML to pak bude vypadat asi takto:

```
<div class="rwd-object">  
    <iframe class="rwd-object-in" ...></iframe>  
</div>
```

A zase si to pojďme vysvětlit. To nejzajímavější se děje v kódu třídy `.rwd-object`:

position: relative

Vytvoří nový omezující blok, ve kterém lze absolutně pozicovat.

height: 0

Potřebujeme vynulovat proto, abychom výšku elementu mohli nastavit pomocí vnitřních okrajů prvku.

padding-bottom: 60%

Svislý vnitřní okraj se na rozdíl od vlastnosti `height` počítá ze šířky elementu, takže s jeho pomocí určíme poměr stran rodičovského bloku.

Poměr stran je zde tedy 100 ku 60, takže 5 : 3. Pro poměr 16 : 9 bychom do `padding-bottom` vložili hodnotu 56.25%. Tady je ještě výpočet: $(9 / 16 * 100)$.

.rwd-object-in

Tuto třídu pak aplikujeme přímo na <iframe> nebo jiný vkládaný element. Ten už má za úkol se jen sobecky roztahovat na celou výšku i šířku rodiče.

Máte svrbění si to hned zkoušet? Neváhejte, CodePen se na vás těší.
cdpn.io/e/BdniC

Tímto jsme ošetřili *pružnost* obrázků a vkládaných médií v rámci layoutu. *Responzivnost* se ale definuje šířejí než jen prostou flexibilní reakcí na změnu šířky layoutu. Jdeme na ni. Zaměříme se nejprve na obrázky, což je bezpochyby nejčastější mediální obsah.

Responzivní obrázky

V responzivním designu máme na výběr poměrně hodně řešení možných problémů s obrázky.

Pojďme si nejprve udělat mapu těch problémů:

- *Rychlost načítání*

Velký obrázek pro počítač je zbytečné posílat do mobilu. Datový objem je jeden z největších problémů bránících rychlému načtení.

- *Retina displeje*

Pokud má displej dvojnásobnou a vyšší hustotu hardwarových pixelů, bude tam obyčejná bitmapa vypadat špatně. Znáte to z textu o [CSS pixelu](#).

- *Art direction*

Občas chceme na různá zařízení poslat různé výřezy obrazovky. Celou fotku na počítač a výřez obličeje na mobil například.

- *Velikost okna*

Pro různě velká okna prohlížeče bychom rádi servírovali různé

varianty obrázků.

- *Layout*

Úplně nejraději bychom, aby obrázky znaly layout stránky, protože ten je v responzivním designu velmi variabilní.

Nejčastěji chceme ušetřit datový objem stránky na mobilech nebo poskytnout kvalitní zobrazení pro vysokokapacitní displeje typu Retina.

Ukažme si proto kompletní přehled všech možných řešení, jejich výhod a nevýhod.

Řešení	Rychlosť	Retina	AD	Okno	Layout	Vhodné pro
1. SVG	+	+	-	-	-	vektory
2. 	-	-	-	-	-	cokoliv
3. 2x	-	+	-	-	-	cokoliv
4. kompr.	+	+	-	-	-	fotky
5. 	-	+	+	+	-	cokoliv
6. 	+	+	-	+	-	cokoliv
7. 	+	+	-	-	+	cokoliv
8. <picture>	+	+	+	+	-	cokoliv

Srovnání řešení pro responzivní obrázky. Rychlosť – zohledňuje rychlosť načítání? Retina – zohledňuje vysokokapacitní displeje? AD (Art Direction) – dokážou poslat různé ořezy obrázků na různá zařízení? Okno – umí vybírat obrázky podle velikosti okna prohlížeče? Layout – zohledňuje layout webu?

Ve srovnání jsem leccos zjednodušil. Nevidíte tam, že jednotlivá řešení lze kombinovat. Třeba komprimované obrázky s technikou srcset/sizes. Ale to vám určitě došlo nebo dojde z dalšího textu. A teď už na jednotlivá řešení.

1. Vektory? SVG

Tohle je jednoduché. Máte-li obrázek vyjádřitelný vektorem, prostě z něj udělejte SVG a pošlete jej ve stránce prohlížečům. Pokud to extra nezmrvíte, vektory jsou datově velmi úsporné a automaticky připravené.

O responzivních SVG píšu v jedné z dalších podkapitol.

2. Staré dobré

S bitmapami to bude složitější, ale jednu věc vím jistě. Jeden neoptimalizovaný obrázek vám pravděpodobně stačit nebude.

```
<!-- Bude datově neúsporný nebo  
     ošklivý na Retina displejích: -->  

```

Tahle (ne)technika patří do muzea webového vývoje. Podlaží Počítačová éra.

3. Dvojnásobná velikost obrázku v

Občas se ještě setkávám s řešením, které upřednostňuje Retina displeje. Autoři prostě obrázek vloží ve dvojnásobné fyzické velikosti oproti původnímu:

```
<!-- Bude datově neúsporný (a možná také  
     ošklivý na Retina displejích): -->  

```

Je to samozřejmě nevýhodné pro rychlosť načtení na běžných (ne-

Retina) displejích. Raději vás upozorním, že obrázek nebude datově dvakrát tak velký, ale tři- nebo čtyřikrát. Obsahuje přece čtyřnásobný počet pixelů. Zajímavější to začne být, když obrázku uberec na zobrazovací kvalitě.

4. Razantně komprimované obrázky v

Datový objem i vysokokapacitní displeje můžete v některých situacích vyřešit naráz. Prostě zvětšte pixelovou velikost obrázku a výrazně snížte jeho kvalitu:

```
>
```

Jak vypadá výroba takového obrázku ve třech krocích?

1. Původní obrázek uložte ve výrazně větší pixelové velikosti.
2. Snížte kvalitu exportu někam výrazně pod polovinu.
3. Prohlížeč necháte obrázek převzorkovat na původní velikost.

Komprimované obrázky jsme zkoušeli nasadit na jednom starším projektu. Udělali jsme si testy pro různé kombinace komprese a pixelové velikosti. Nakonec jsme došli k tomu, že obrázky ve dvojnásobné pixelové velikosti a kvalitě komprese nastavené na 30 % měly nejlepší poměr kvality a datového objemu. Ten byl poloviční oproti původní verzi s 80% kvalitou a velikostí stejnou, jako se používá ve stránce. U různých typů obrázků to ale bude různé.

Autoři nápadu, Filament Group, svůj zkušební obrázek vkládali dvaapůlkrát větší a kvalitu JPG snížili na 0 %. Výsledný obrázek se pyšnil opět méně než polovinou datového objemu toho původního.
vrdl.in/z7k34

Asi sami vidíte, že řešení je vhodné jen pro JPG nebo WebP obrázky, kde je možné nastavit ztrátovou kompresi různých úrovní. Typově je pak použití metody vhodné spíše pro fotografie než třeba obrázky s textem, kde by v ostrých hranách mezi barvami byla ztráta kvality viditelná.

5. Vlastní řešení pomocí

Občas je pro responzivní obrázky možné vidět řešení s nahrazováním atributu **src**:

```
>
```

Na velkých displejích pak autoři těchto řešení usilují o zkopírování obsahu **data-src** do **src** pomocí Javascriptu. Ano, prohlížeč pak zobrazí správný obrázek. Takto pracuje například knihovna Response. [responsejs.com](#)

Na pohled elegantní, ale nevýhody to má. Neexistuje totiž způsob, jak prohlížeč odradit od stažení obrázku nalinkovaného v atributu **src**. Proto se v těchto řešeních obrázek sice vymění, ale předtím se už stáhl tento soubor. To není potěšující zpráva pro uživatele čekající na pomalém připojení.

Navíc je nutné naprogramovat i logiku pro další scénáře, které mají responzivní obrázky řešit. Například ony Retina displeje. Logiku, kterou už navíc prohlížeče mají v sobě. Hned k ní dojdeme, ale musíme se rozloučit se starým známým atributem **src**.

Iniciativa Responsive Images Community Group totiž před lety přišla s novými atributy – **srcset** a **sizes** – a také s úplně novým tagem **<picture>**. To jsou řešení, která dnes považuji za standardní,

a pokud je to možné, dávám jim přednost.

6. Atribut `srcset` značky

Hodí se pro scénář s výběrem varianty podle velikosti okna. Do atributu `srcset` uvedete velikostní varianty, které jste si předtím uložili na server:

```
>
```

Všimněte si `w`, takzvaného *deskriptoru*, který nese informaci o šířce obrázku. Proč je tam potřeba? Dobrá otázka, zodpovíme si ji v textu o atributech `srcset` a `sizes`. Těší se na vás hned v další podkapitole.

7. Atribut `sizes` značky

Řešení s atributem `srcset` je fajn, ale zajistí výměnu obrázků jen podle velikosti okna. Obrázky se ale obvykle vyskytují v nějakém prostředí layoutu webu. Proto potřebujeme ještě atribut `sizes`, kterým prohlížeči předáváme onu informaci o layoutu:

```
>
```

Pokud chcete více informací, odkážu vás opět na podrobně rozepsaný materiál o atributech srcset a sizes.

8. Nová značka <picture>

Nový tag <picture> vymysleli pro méně časté scénáře – třeba když potřebujete mít na konkrétních velikostech layoutu jinak oříznuté obrázky:

```
<picture>
  <source media="(min-width: 600px)"
srcset="image_100x100.jpg">
  <source media="(min-width: 1024px)"
srcset="image_300x300.jpg">
    
</picture>
```

Na první pohled méně zkušených očí vypadá užitečněji než atributy srcset a sizes, ale není to pravda. Hodí se opravdu hlavně jen na ty speciální ořezové verze a další méně časté scénáře. Více si přečtete v samostatném textu.

Co ale ještě zmínit chci, je podpora nových atributů a značky <picture> v prohlížečích. Je výborná, nebojte se.

Podpora srcset, sizes a <picture> v prohlížečích

Podporují je všechny moderní prohlížeče. Responzivní obrázky nám chybí hlavně ve všech verzích Exploreru (a taky Android Browseru do čtyřkových verzí Androidu, ale to už nás asi nemusí trápit). caniuse.com/srcset

Obzvlášť IE ve verzi 11 je ke dni psaní textu ještě velmi silně zastoupený. Je však dobré si uvědomit, jaké je v tomto případě chování „nepodporujících prohlížečů“.

První náhradní řešení: přirozené

Použijete parametr `src`, který moderní prohlížeče ignorují, pokud je přítomný `srcset`:

```

```

Druhé náhradní řešení: Picturefill

Javascriptová knihovna, která zařídí fungování atributů `srcset`, `sizes` a značky `<picture>` i ve starších prohlížečích. Jmenuje se Picturefill a považuji ji za dobré řešení, které mám odzkoušené na několika webech. scottjehl.github.io/picturefill

Atributy responzivních obrázků: `srcset` a `sizes`

Nové atributy řeší potřebu autorů stránek zobrazovat v různých kontextech designu různé varianty obrázků.

Změna kontextu v tomto případě nejčastěji vypadá jako změna layoutu pro jinou velikost obrazovky. Může ale jít také o zobrazení stránky na zařízeních s displeji typu Retina (různými poměry `device-pixel-ratio`). Do budoucna třeba ještě o změnu úrovně zoomu na stránce nebo uživatele na pomalém připojení.

Na atributech `srcset` a `sizes` je hezké, že poměrně složité rozhodování, který obrázek ve které situaci použít, necháváme na prohlížeči.

Jako autoři strány mu jen řekneme, jaké varianty obrázku má k dispozici (`srcset`) a jak jsou veliké na jednotlivých breakpointech layoutu (`sizes`).

srcset: Sada zdrojů obrázku a jejich vlastnosti

```
>
```

Prohlížeč tímhle kódem sdělujeme, že jsme předgenerovali obrázek `small_600.png`. Jeho fyzická šířka (při exportu z grafického editoru) je 600 pixelů, což říká zápis `600w`, k němuž se ještě dostaneme. Dále zde máme obrázek `medium_1024.png` v šířce 1024 pixelů a `large_1600.png` v šířce 1600 pixelů. V atributu `src` pak uvádíme náhradní řešení pro prohlížeče, které atribut `srcset` neumí.

Prohlížeč se zde při rozhodování o tom, který obrázek stáhnout a zobrazit, dívá na aktuální šířku okna.

Některé prohlížeče na to půjdou chytřejí a například `small_600.png` budou zobrazovat ještě kousek nad hranicí šestisetpixelového okna, protože na vizuální kvalitě obrázku to neubere.

Prohlížeč vezme v potaz i aktuální `device-pixel-ratio`. Například na zařízení s původním Retina displejem (`device-pixel-ratio=2`) pak v případě, že okno je široké 600 pixelů, stáhne a použije už největší obrázek, `large_1600.png`. Potřebuje obrázek velikosti šířky okna krát `device-pixel-ratio`. Takže kolem 1200 pixelů široký. Obrázek `medium_1024.png` by mu tedy nestačil.

V potenciálu chytrého rozhodování prohlížeče vězí krásu atributu `srcset`. Prohlížeč zváží všechny informace, které má o stavu stránky k dispozici a podle toho vybere nevhodnější obrázek. Vy jako autoři jen vygenerujete dost variant a správně je popíšete.

Demo výše uvedeného kódu mám také na CodePenu. Nejlépe jej vyzkoušíte, když si zmenšíte okno ukázky, obnovíte stránku a pak budete okno postupně zvětšovat. cdpn.io/e/WboGgE

Kolik variant obrázků vygenerovat?

V ukázce mám obrázky tří, to by ale v praxi nestačilo. Obvykle si vypočtu šířku nejmenší a největší možné varianty. Mezi nimi pak nechám vygenerovat obrázky odstupňované po 200 až 400 pixelech. Je to samozřejmě někdy limitováno i úložným místem na serveru, takže tuto radu prosím berte s rezervou.

Detailně si postup hledání variant ukážeme už za chvíli na [příkladu](#).

Deskriptory vlastností obrázků v srcset

Zatím jsem zmínil jen šířku obrázku, tedy deskriptor `w`. Ten budete používat v naprosté většině případů. Deskriptor `x` jej doplňuje pro specifické scénáře použití.

deskriptor w

Udává, jakou pixelovou šířku má ve skutečnosti soubor s obrázkem.

```
<img srcset="  
    small_600.png 600w,  
    medium_1024.png 1024w"  
    width="200" height="200" alt="...">>
```

Pozor, neříká nic o šířce obrázku v rámci layoutu stránky. K tomu dále slouží atribut `width` nebo případně CSS.

deskriptor x

Druhý deskriptor určuje připravenost souboru s obrázkem pro různé poměry `device-pixel-ratio`, například:

```
<img srcset="  
    image.png,  
    image@2x.png 2x"  
width="200" height="200" alt="...">>
```

Tímto zápisem říkám, že `image@2x.png` má prohlížeč použít při `device-pixel-ratio` alespoň 2 a `image.png` ve všech hodnotách menších než 2. Když deskriptor neuvedete, výchozí je `1x`.

Pojďme se teď ještě podívat na atribut `sizes`, který prohlížeč umožní vybírat nejen podle fyzických parametrů souborů s obrázky, ale i podle layoutu vaší stránky.

sizes: Velikost obrázku ve stránce

V praxi totiž tak často nepotřebujeme, aby prohlížeč vybral obrázek podle šířky okna. Spíše podle šířky prostoru pro obrázek v rámci aktuálního layoutu stránky. A právě od toho máme atribut `sizes`:

```
>
```

Sledujte hodnoty v `sizes`. Responzivní layout v ukázce je vymyšlený takto:

- V oknech šířky 768 pixelů a více se obrázek vykresluje do plochy o šířce 300 pixelů (`(min-width: 768px) 300px`).
- Ve všech ostatních velikostech okna, tedy do 767 pixelů, zabere 100 procent šířky okna (`100vw`).

„Volkswageny“, tedy jednotku `vw`, jsme probírali v [textu o jednotkách](#), vzpomínáte?

První vyhovující varianta v `sizes` vyhrává, takže na pořadí záleží. Pozor na to.

Proč si informaci o layoutu stránky prohlížeč nevezme z CSS?

Dobrá otázka. Rozhodnutí, který z obrázků stáhnout a zobrazit, prohlížeče dělají ještě předtím, než znají CSS. Ony v tu chvíli jen rychle parsují HTML a o stylech zatím „nevědí“. Když by na styly čekaly, zpozdí se stažení a zobrazení obrázků. Je to takhle výhodnější pro uživatele. Zkracuje to načtení obrázků a vlastně celé stránky.

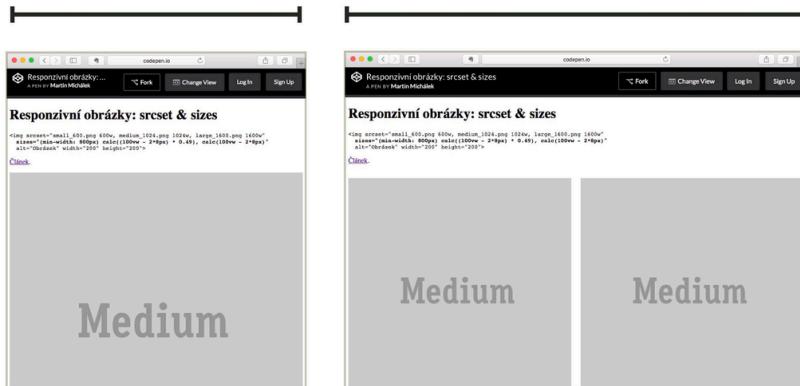
Velikosti obrázků podle layoutu

V responzivním layoutu obvykle přesně nevíme, jaké rozměry budou mít obrázky v rámci konkrétní šířky okna. A právě tady se projeví síla kombinace `sizes` s CSS funkcí `calc()`. Definovat velikost obrázku můžeme relativně k layoutu mezi konkrétními breakpointy.

Pojďme si nejprve vizuálně přiblížit layout pro další ukázku:

0 - 799px

800px a více



Layout příkladu pro demonstraci srcset/sizes

Do 800px breakpointu je to jednoduché: Obrázek zabírá celou šířku layoutu. Nikoliv ovšem šířku okna, a tak musíme odečít výchozí margin u <body>, který mají prohlížeče nastavený na 8px:

`calc(100vw - 2 * 8px)`

Od breakpointu 800px pak musíme vyjít z CSS layoutu, který je zapsaný takto:

```
@media only screen and (min-width: 800px) {  
    .image {  
        width: 49%;  
    }  
}
```

Přepsáno do funkce calc() to vypadá takto:

`calc((100vw - 2 * 8px) * 0.49)`

Ještě v jazyce našeho kmene:

(100 procent šířky viewportu - výchozí margin u <body>)
* 49% šířka obrázku

Takže celý zápis tagu bude vypadat takto:

```

```

Pojďme si pro jistotu ještě shrnout celý zápis:

1. V **src** máme obrázek sloužící jako náhradní řešení pro starší prohlížeče.
2. V **srcset** máme seznam variant obrázku, které jsme předpřipravili a uložili na server.
3. Atribut **sizes** říká: na šírkách okna od 800 pixelů výše bude mít obrázek velikost **calc((100vw - 2 * 8px) * 0.49)**. Ve všech ostatních případech – to znamená do 799 pixelů – pak **calc(100vw - 2 * 8px)**.

Demo na CodePenu: cdpn.io/e/azBmaX

Nezapomínejte prosím na povinný atribut **alt**, který ocení vyhledávače a odečítáče obrazovky pro zrakově hendikepované uživatele.

Doplňte i atribut **height**, který vylepší vykreslování stránky tím, že ještě před stažením obrázku sdělí prohlížeči, jak velký prostor pro něj má v layoutu vynechat. Nevztahujte jej tedy k pixelové výšce obrázku, ale k prostoru stránky, který si přejete pro obrázek rezervovat do chvíle, než se stáhne a zobrazí.

„Hernajs, a proč je to tak složité?“

Nedivím se samozřejmě žádným námitkám vůči estetice a zdánlivě zbytečné složitosti autorské práce s responzivními obrázky. Moje první reakce byly stejné. Když si ale zopakujeme, že informace z CSS prohlížeči v momentě parsování HTML nijak nepomohou, asi bychom došli k řešení stejnemu nebo velmi podobnému. Pokud bychom jej tedy chtěli vymýšlet znova, což nikomu nedoporučuji.

V textu o responzivních obrázcích jsem zmiňoval i další alternativy. `` ale považuji za výchozí řešení. Ta ostatní se hodí pro konkrétní a méně časté scénáře.

Pojďme si rozpitvat jednu z metod pro specifické situace – novou značku `<picture>`.

Nová značka Picture

`<picture>` umožňuje definovat varianty obrázku pro různé stavy v responzivním webdesignu.

Na rozdíl od atributů `srcset` a `sizes` nenecháváme rozhodování na prohlížeči. Vedení tady přebíráme my autoři. Ukážu tady pář scénářů, kdy je to výhodné.

Ukázka zápisu

```
<picture>
  <source media="(min-width: 1024px)" srcset="large.jpg">
  <source media="(min-width: 600px)" srcset="medium.jpg">
  
</picture>
```

V elementech `<source>` uvádím alternativy k výchozímu obrázku, který je v ``.

Prohlížeč vezme vždy první vyhovující obrázek. Je možné tedy obrázky řadit jak od největšího, tak od nejmenšího obrázku. V prvé případě použijte Media Query `min-width`, v druhém `max-width`.
cdpn.io/e/qDmar

Značka `<picture>` přitom tvoří jen obal, zatímco prvky `<source>` jsou jakési molitanové vycpávky nesoucí informaci o alternativách. Veškeré stylování nebo všechni události v Javascriptu je nutné dělat přímo na `` elementu. V každém `<picture>` musí být právě jeden ``.

Kdy se vám může `<picture>` hodit? Hlavně ve dvou situacích:

1. Připravili jste obrázky v různých ořezech. Třeba na mobily chcete poslat čtverce a jinde obdélníky. Zároveň chcete mít pod kontrolou hranice, kdy prohlížeč použije jednu, či druhou ořezovou verzi. Jde o „art direction“, tedy autorské řízení formy a obsahu obrázků.
2. Prohlížečům jste obrázky připravili v různých souborových formátech.

V naprosté většině případů vám bude stačit stará dobrá značka `` s atributy `srcset` a `sizes`.

Art direction: obrázky pro různá rozlišení mají také různý obsah

Máme tři varianty obrázků a prohlížeči chceme přesně stanovit hranice přepínání mezi nimi:

```
<picture>
  <source
    srcset="large_1600.png"
    media="(min-width: 1024px)">
  <source
    srcset="medium_1024.png"
    media="(min-width: 800px)">
  
</picture>
```

Pro okna 1024 pixelů a větší se stáhne a použije obrázek `large_1600.png`, od 800 do 1023 pixelů `medium_1024.png` a pro okna šířky 799 a méně pixelů pak `small_600.png`.

I tady jsem pro vás připravil demo na CodePen. cdpn.io/e/VYPPQQ

V čem se to liší od ``? Příklad, který uvádím výše, je velmi zjednodušený. Museli byste v něm ještě ošetřit displeje typu Retina, tedy různé hodnoty `device-pixel-ratio`. To máte u `srcset` a `sizes` „v ceně“ řešení: prohlížeč to udělá sám. Na druhou stranu tady pomocí jakýchkoliv Media Queries určíte sami hranice mezi variantami. V metodě `srcset` vybírá prohlížeč sám podle layoutu nastaveného v `sizes`.

Jinými slovy: Pokud byste se rozhodli používat `<picture>` pro běžné obrázky, byly by vaše Media Queries v nich uvedené dost složité. Kromě šířky okna by musely zohledňovat velikost obrázku v layoutu a také displeje typu Retina. Pojdeme se ale zaměřit na ty scénáře, kdy se nová značka opravdu hodí.

Podle formátu obrázku

Vybírat obrázky prohlížeče umí i podle formátu. Použijte atribut `type`. Hodí se hlavně pro detekci prohlížečů, které zvládají nový

formát WebP. Ten je mimochodem ještě výrazně datově úspornější než JPEG, ale ke dni psaní jej podporuje jen Chrome a Opera.
caniuse.com/webp

```
<picture>
  <source media="(min-width: 1024px)"
    srcset="large.webp" type="image/webp">
  <source media="(min-width: 1024px)"
    srcset="large.jpg">
  
</picture>
```

Prohlížeč, který umí formát WebP a běží v okně velikosti alespoň 1024 pixelů, stáhne a zobrazí soubor `large.webp`.

Tímto způsobem je také možné udělat pěkné náhradní řešení pro formát SVG:

```
<picture>
  <source type="image/svg+xml" srcset="logo.svg">
  
</picture>
```

Šmytec. O bitmapových obrázcích jsme si toho řekli už dost. Teď vzhůru do vektorů!

Responzivní SVG

Vektorové obrázky jsou fajn. Asi tady není potřeba zahlitit vás celou řadou argumentů pro využívání SVG. Ty si případně nastudujte na Vzhůru dolů. vrdl.cz/p/svg

Používání SVG pro ikony nebo logotypy namísto PNG či GIF obrázků hájím kam vkročím, ale v jedné věci jsou bitmapy zlaté: V přizpůsobování šířce layoutu a zachování poměru stran. To, čeho jsme tak snadno dosáhli v kapitole o pružných obrázcích, u SVG

budeme dělat poměrně složitě.

Abychom ale pochopili, proč není dosažení pružnosti SVG jednoduché jako facka, musíme na světlo Boží vytáhnout jednu překvapivou a možná i nepříjemnou pravdu. Nádech...

SVG není obrázek.

... výdech! No jasně, SVG je vektorový dokument, který vkládáme do stránky. Právě proto jej do HTML můžeme dát nejen ve formě obrázku (``), ale také jako externí zdroj (`<iframe>`, `<object>`) nebo vektory vykreslit přímo (`<svg>`).

Bitmapy mají jasně definovanou výšku i šířku. Je proto velmi snadné jejich poměr stran zachovat. Jaká je ale výška nebo poměr stran dokumentu?

Nastavení šířky a výšky je k ničemu, použijte `viewbox`

Mnoho kodérů se domnívá, že u značky `<svg>` nastaví parametry `width` a `height` – a SVG začne poslouchat. Je to tak ale jen v některých prohlížečích a některých typech vložení do stránky. Celá pravda ovšem je, že nastavením výšky a šířky si mnoho věcí zkomplikujeme.

Vysvětlení je složité, takže vás odkážu na článek, který to rozebírá lépe, než bych to dokázal udělat já. Než se ale do čtení textu „How to Scale SVG“ na CSS Tricks pustíte, ujistěte se, že doma máte dostatečnou zásobu brufenů. css-tricks.com/scale-svg

S brufeny nebo bez, výsledek je stejný. Prostě použijeme parametr `viewbox`:

```
<svg viewBox="0 0 100 50">
```

Kód říká, že výchozí velikost SVG dokumentu je 100 na 50 pixelů, že souřadnicový systém začíná klasicky na bodě nula nula a že si má dokument držet poměr stran.

Jak teď zajistit pružné chování SVG při různých typech vložení do stránky?

Pružné SVG vložené do HTML dokumentu pomocí `<svg>`

V moderních prohlížečích je to v případě dodržení výše uvedeného opravdu jednoduché. Jen kvůli Internet Exploreru musíme přidat obalující značku:

```
<p class="svg-container">  
  <svg viewBox="0 0 900 400" class="svg-content"> ... </svg>  
</p>
```

Třídu `.svg-container` pak kvůli Explorerům nastylujeme metodou pro zachování poměru stran, stejně jako to děláme u vkládaných elementů v textu o obrázcích:

```
.svg-container {  
  position: relative;  
  width: 100%;  
  height: 0;  
  padding-top: 100%; /* Poměr stran 1:1 */  
}  
  
.svg-content {  
  position: absolute;  
  top: 0;  
  left: 0;  
}
```

Podívejte na výsledný CodePen. cdpn.io/e/oYOZwz

Pružné SVG vkládané externě pomocí

Opět do zdrojového SVG přidáme viewbox a zrušíme případně přítomné atributy `width` a `height`. Pak stačí klasicky vložit do stránky:

```

```

Pokud nebudeme nastavovat výšku obrázku, v CSS nemusíme pro moderní prohlížeče nastavovat vůbec nic. Opět jen musíme napravit chování Internet Explorerů:

```
.svg {  
    width: 100%;  
}
```

CodePen s příkladem se na vás těší i tady. cdpn.io/e/VmNbPx

Pružné SVG externě v CSS

Tady je to jednoduché – stačí prostě obrázek umístit na pozadí, zakázat mu opakování a pomocí `background-size: contain` jej rozprostřít do celé šířky rodiče.

```
.svg-icon {  
    background-image: url('icon.svg');  
    background-repeat: no-repeat;  
    background-size: contain;  
}
```

U většiny vektorových obrázků pak chceme definovat poměr stran, který si mají zachovat. Opět si pomůžeme trikem pro zachování poměru stran.

```
.svg-icon  
{  
    ...  
    height: 0;  
    padding-bottom: 100%; /* Poměr stran 1:1 */  
}
```

Tady už není ani žádné speciální nastavení pro Internet Explorer.
Hurá!

Mrkněme se spolu na CodePen. cdpn.io/e/NbmgsPr

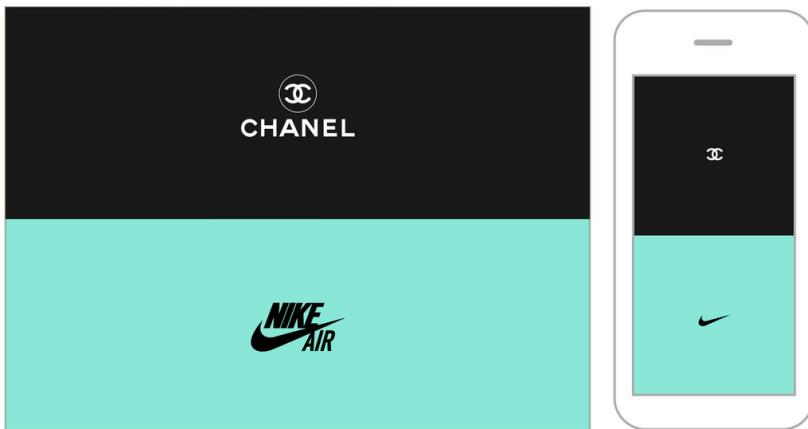
Tím bychom mohli zajištění pružnosti SVG v responzivním layoutu uzavřít pro nejčastěji používané způsoby vložení.

Formát je ale možné do stránky vkládat i dalšími, méně používanými způsoby. Jak zajistit pružnost při vložení do `<object>`, `<iframe>`? Poradí vám vrchní odbornice na SVG, Sara Soueidan, v článku „Making SVGs Responsive with CSS“. vrdl.in/yixth

Ale čtěte dál. To nejlepší teprve přijde. Slíbil jsem povídání o *responzivních* SVG. Zatím jsme se ale bavili jen o těch *pružných*. O těch, které se přizpůsobují šířce rozvržení stránky. Kromě pružného layoutu ale responzivitu definují ještě změny stylování v určitých velikostech obrazovky. Prostě [Media Queries](#).

(Opravdu) responzivní SVG

Ano, uvnitř SVG můžeme Media Queries úplně v pohodě použít. A ano, někdy se podmínky nevztahují k rozměrům celé stránky, ale k rodiči SVG dokumentu.



Využití Media Queries v responzivních SVG najdete na webu „Responsive Logos“. Podívejte se, stojí to za to. responsivelogos.co.uk

K čemu se vztahují Media Queries v SVG?

Podívejme se na jednoduchý příklad, kdy na menších velikostech okna invertujeme barvy loga Vzhůru dolů.

Media Queries prostě napišeme dovnitř kódu vektorového dokumentu:

```
<svg>
  <style>
    @media all and (max-width: 500px) {
      #layer-background { display: none; }
      #layer-text path { fill: url(#Gradient_1); }
    }
  </style>
  ...
</svg>
```

Tady se podmínky v `@media` vcelku logicky vztahují k šířce okna prohlížeče. cdpn.io/e/vyMRPL

Co když ale vložíme SVG soubor obsahující Media Queries externě?

```

```

Tušíte správně, podmínky v @media se pak budou vztahovat k šířce obrázku samotného. cdnp.io/e/zZKzRe

Podmínky budou pracovat jako Container Queries, které bychom ve webdesignu potřebovali jako sůl. Ale nemáme je. Zatím tedy jen u externích SVG. Zmíním je ještě v [kapitole o Media Queries](#).

Tak či tak, mechanismus responzivních SVG má velkou budoucnost: pro ikony, grafy, interaktivní elementy, mapy (!) a další prvky s potřebou měnit hustotu informací nebo formy podle velikosti okna prohlížeče.

A teď už pryč od obrázků a hurá do tabulek a grafů. Slibuji, že to nebude taková nuda, jak to zní.

Responzivní tabulky

Chuck Norris toho zvládne hodně, třeba i rozbrečí cibuli, ale tabulky na webu by mu daly zabrat. No vážně. Však čtěte.

Zejména ty rozsáhlejší mají nehezkou vlastnost, že na menších displejích jsou rozměrově poněkud nezkrotitelné. Pojdíme si představit všechny způsoby, jak lze s tabulkami v dnešním webdesignu zacházet, a vy si jistě vyberete. Tedy pokud nejste Chuck Norris. Ten si vybral, ještě než jsem začal psát.

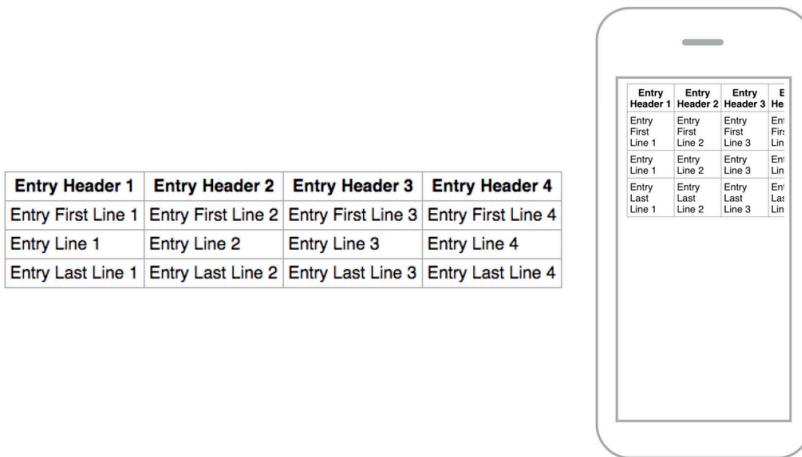
Posun do stran

Nejjednodušší varianta. Prostě tabulce přikážete, ať se roluje do

strany, pokud překročí aktuální šířku okna:

```
.scrollable-table {  
    overflow-x: auto;  
}
```

Uživatel si pak onen posun obstará palcem. Důležité je, aby byla možnost posunu indikována useknutím obsahu zprava. „Scrollbar“, indikátor možnosti posunu, totiž sám o sobě nestačí. Na mobilech nebývá vidět, dokud uživatel na tabulkou nezaútočí prstem.



Obsah tabulky se na malém displeji posouvá do stran

Vyzkoušejte si zmenšit okno v ukázce. cdpn.io/e/ENMezZ

Řešení se hodí hlavně pro tabulky s menším počtem řádků i sloupců a s popisem dat nahore. Nebo také pro tabulky vkládané přes redakční systémy, u kterých nevíte, jak složité budou. Anebo když prostě chcete ušetřit čas na vývoj.

Než si ukážeme propracovanější způsoby práce s responzivními tabulkami, dovolte mi jeden tip na nástroj.

Chytrý plugin: Tablesaw

Tablesaw je jQuery plugin, který zvládá téměř všechny zde popsané možnosti chování responzivních tabulek. Zkrátka švýcarský tabulkový nůž Chuck Norrise. github.com/filamentgroup/tablessaw

Posun do stran s fixním sloupcem

Varianta pro tabulky s popisem dat ve svislém směru a klidně i příšerně moc sloupci s daty samotnými. Na malém displeji prstem posunujete do stran jen sloupce s daty. Popis zůstává na místě.

Movie Title	Rank	Year	Rating	Gross
Avatar	1	2009	83%	\$2.7B
Titanic	2	1997	88%	\$2.1B
The Avengers	3	2012	92%	\$1.5B
Harry Potter and the Deathly Hallows—Part 2	4	2011	96%	\$1.3B
Frozen	5	2013	89%	\$1.2B
Iron Man 3	6	2013	78%	\$1.2B
Transformers: Dark of the Moon	7	2011	36%	\$1.1B
The Lord of the Rings: The Return of the King	8	2003	95%	\$1.1B
Skyfall	9	2012	92%	\$1.1B
Transformers: Age of Extinction	10	2014	18%	\$1.0B



Tabulka s pevně ukotveným prvním sloupcem na mobilu a možností posouvat gestem „swipe“. github.com/filamentgroup/tablessaw

Vyzkoušejte si naživo v CodePenu. cdpn.io/e/qqvJdV

S propracovanějším řešením využívajícím flexbox a další moderní CSS vlastnosti přišel David Bushell v textu „CSS only Responsive

Tables“. vrdl.in/xlpbn

Fixně-posuvné řešení je pak možné doplnit detekcí gesta švihnutí (swipe) pro snadnější a přesnější posouvání sloupečků.

Řešení má mnoho užití. Podmínkou ale je, aby tabulka měla přijatelně nízký počet řádků.

No jo, ale co když máte tabulku toho typu, kterému programátoři říkají „datagrid“? Ta má, potvora, hodně sloupečků, ale také řádků.

Stohování

Datagrid není žádná vzácnost. Každá webová aplikace pro interní systémy je datagridů plná. Je to případ i vašeho projektu? Pak bych vám doporučil přestylovat tabulkou na mobilech do podoby netabulkového, kartičkového zobrazení. Říkám tomu *stohování*.

Movie Title	Rank	Year	Rating	Gross
Avatar	1	2009	83%	\$2.7B
Titanic	2	1997	88%	\$2.1B
The Avengers	3	2012	92%	\$1.5B
Harry Potter and the Deathly Hallows – Part 2	4	2011	96%	\$1.3B
Frozen	5	2013	89%	\$1.2B
Iron Man 3	6	2013	78%	\$1.2B
Transformers: Dark of the Moon	7	2011	36%	\$1.1B
The Lord of the Rings: The Return of the King	8	2003	95%	\$1.1B
Skyfall	9	2012	92%	\$1.1B
Transformers: Age of Extinction	10	2014	18%	\$1.0B



Movie Title	Avatar
Rank	1
Year	2009
Rating	83%
Gross	\$2.7B
Movie Title	Titanic
Rank	2
Year	1997
Rating	88%
Gross	\$2.1B
Movie Title	The Avengers
Rank	3
Year	2012
Rating	92%
Gross	\$1.5B
Movie Title	Harry Potter and the Death

Stohování tabulky na menších displejích. github.com/filamentgroup/tableaw

V nejjednodušší možné CSS implementaci tabulce na menších displejích zrušíme „tabulkovost“:

```
@media only screen and (max-width: 600px) {  
    table, th, td, tr, thead, tbody {  
        display: block;  
    }  
}
```

Na CodePenu je možné zkusit si to i s dalším stylováním.

[cdpn.io/e/bBZmxE](https://codepen.io/e/bBZmxE)

I tak ale čisté CSS řešení nebude dokonalé. Pro tento typ práce s tabulkami budete potřebovat kousek Javascriptu nebo zmíněný plugin.

Stohování se hodí i pro tabulky se složitějším obsahem v buňkách: odstavcový text, formulářové prvky a tak dále.

Změna směru tabulky

Často se stává, že se pro malé displeje hodí jiný směr zobrazení tabulky než u velkých. Podívejte se na obrázek, hned pochopíte.

The image shows a comparison between a full-sized table and its mobile responsive version. On the left is a wide table with 7 columns: Numbers, Names, Values, Dates, Cash Money, Messages, and Buttons. It contains 6 rows of data. On the right is a mobile phone displaying a responsive version of the same table, which is much narrower and has collapsed some columns (like Dates and Cash Money) into dropdown menus.

Numbers	Names	Values	Dates	Cash Money	Messages	Buttons
000000001	Dr. Jayhawk	102	03/30/1940	\$60.42	PAID	Select
000000002	Dr. Jayhawk	137	03/18/1953	\$69.68	PAID	Select
000000003	Dr. Wolverine Longer Text Test	154	03/29/1976	\$86.68	PAID	Select
000000004	Dr. Tarheel	113	03/30/1981	\$63.50	PAID	Select
000000005	Dr. Orange	147	03/30/1987	\$74.73	PAID	Select
000000006	Dr. Who	000	04/08/2013	\$0.00	PENDING	Select

Změna směru tabulky. Vyzkoušejte si to na CodePenu. cdpn.io/rjmyx

Nasazení doporučuji u tabulek, které mají velký počet řádků, ale málo sloupců.

Tak. Řekněme, že nejčastější scénáře jsme vyčerpali. Pojďme se ale podívat i na exotičtější situace.

Odkaz na plnou tabulku

Na mobilech můžete samozřejmě tabulku hodně zjednodušit a přiložit odkaz na plnou verzi.

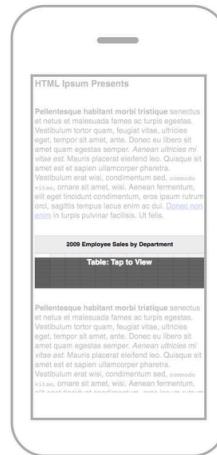
Za fajn nápad také považuji vložit do stránky namísto tabulky jen jakýsi zástupný symbol. Vidíte to na obrázku a zkoumat můžete v přiložené ukázce. jsbin.com/apane6/14

HTML Ipsum Presents

Pelleterisque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tempor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, [commodo vivamus](#), ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. [Donec non enim](#) in turpis pulvinar facilisis. Ut felis.

2009 Employee Sales by Department										
	food	auto	household	furniture	kitchen	bath	flooring	plumbing	electrical	hardware
Mary	190	160	40	120	30	70	40	120	30	70
Tom	3	40	30	45	35	49	30	45	35	49
Brad	10	180	10	85	25	79	10	85	25	79
Kate	40	80	90	25	15	119	40	80	90	25
Donald	45	35	49	30	3	40	30	45	35	49
Mark	49	30	3	40	30	45	35	49	45	35
Samantha	30	45	35	49	49	30	3	40	45	35
Harold	30	30	3	40	45	35	40	30	49	49

Pelleterisque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tempor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, [commodo vivamus](#), ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. [Donec non enim](#) in turpis pulvinar facilisis. Ut felis.



Tabulku na mobilu uprostřed obsahu nahradíme zástupným symbolem. Kliknutím se zobrazí plná verze

Kdy se hodí? Pro složité tabulky uprostřed jiného obsahu, kde ostatní scénáře (stohování, fixní sloupec) selhávají.

Varianta s reprezentací obsahu na mobilech grafem

Z tabulky prostě na mobilu uděláte zdjednodušený graf. Doporučuji nasazovat v kombinaci s odkazem na plnou verzi tabulky. Hodí se opět pro situace se strašlivě komplikovanými daty bez výrazné obsahové hierarchie.

Na mobilech něco vynechat. Ale opravdu?

„Prostě schováme pár sloupečků, a na mobil se to vejde.“ Tahle varianta vypadá, že se sama nabízí. Z technického pohledu budíž. Jenže designér ve mně zvedá obočí i ukazováček, aby vás upozornil na možné nevýhody.

Schovávání obsahu na konkrétních zařízeních je dost nebezpečné.

Jak už jsem argumentoval dříve, stejní lidé se na vaše rozhraní dívají z různých zařízení. Proč by určitý obsah měli na jednom zařízení vidět a jiném ne?

Připomenu to znovu v textu o častých chybách responzivních webů v rámci sedmé kapitoly.

Responzivní grafy

Responzivní grafy nacházejí využití hlavně v rukou demagogických politiků. Grafy, které používají, se přizpůsobují jejich vidění světa.

OK, nebudu vám kazit krásné chvíle s mými texty těmito rádoby vtipnými odbočkami.

Budeme se bavit o opravdových grafech v opravdových responzivních stránkách. Zase tak často se nepoužívají, proto to vezmu letem světem.

Pokud s grafy pracujete často, pak to hlavní co byste si měli odnést je: V responzivním rozhraní nestačí grafy přizpůsobit šířkou layoutu stránky. Obvykle je potřeba změnit hustotu dat. Někdy dokonce připravit zcela jiné zobrazení pro malé a jiné pro velké displeje.

Poradím alespoň několik knihoven, které to s responzivními grafy umí.

Chartist.js

Z mého pohledu nejzajímavější knihovna pro jednodušší grafy. Protože používá formát SVG, umí kromě přizpůsobení velikosti okna také přizpůsobovat obsah grafů. gionkunz.github.io/chartist-js/



Chartist.js se umí menším displejům přizpůsobovat nejen velikostně, ale i hustotou dat a typem zobrazení

Highcharts

Nějaké responzivní možnosti má i tahle populární grafová knihovna. Spíše se ale jedná o přizpůsobení velikosti než o adekvátní změny v obsahu a hustotě formy či obsahu.

highcharts.com/demo/responsive

Chart.js

Velmi populární knihovna, ale grafy vykresluje do prvku <canvas>, takže s responzitou to bude horší. Canvas totiž není elegantně vektorový jako SVG. Šírkou a výškou se ale grafy přizpůsobovat umí.
chartjs.org

Tím jsme se dostali k poslednímu textu kapitoly o mediálním obsahu ve stránkách. Pojdeme se vrátit k příkladu. ForestKid.cz,

vzpomínáte?

Příklad: dokument s přizpůsobivými médií

Abychom použili vědomosti nasité v téhle kapitole, vložil jsem do příkladu veškerý mediální obsah. Podívejme se teď na něj v rozšírení dnešních běžných mobilů.

1 Cena: 1 313 Kč s DPH
1085 Kč bez DPH

2

Velikost	Vnitřní délka	U vás
23	154 mm	později
24	161 mm	3 dny
25	167 mm	dočasně vyprodáno
26	174 mm	později
27	181 mm	později
28	188 mm	později
29	195 mm	později
30	202 mm	za 4 dny

3

4

Příklad před aplikováním přizpůsobivosti médií

Text se chová hezky, ale média nám vystrkují růžky, že ano? Žádný strach, nůžky na ně brát nebudeme. Prostě zařídíme, aby se začala chovat pružněji.

Pro nedočkavce je tady výsledek:

- Otevření v prohlížeči: vrdl.in/vwdmed
- Stažení v ZIPu: vrdl.in/vwdmedzip

Pojďme si teď lidsky popsat, co přesně se změnilo.

1. SVG logo

Na logo aplikujeme trik, který jsme se naučili v podkapitole o responzivním SVG. Odstraníme parametry `width` a `height`. Všechno necháme na jejich kolegovi: `viewBox`. Řešení hledejte přímo v `index.html`.

Pro správné pružné chování v Internet Exploreru ještě doplníme rodičovský kontejner a trik se zachováním poměru stran pomocí `padding-bottom`. To hledejte v souboru `style/ui/logo.css`.

2. Obrázky produktu

První krok je jednoduchý: pružné přizpůsobení velikosti okna. Toho dosáhneme kódem pro obrázky, který jsme se naučili na začátku kapitoly. Najdete jej v souboru `style/media/images.css`.

Druhý krok zajistí, aby se načítaly také správné varianty obrázků na různě velkých oknech a poměrech `device-pixel-ratio`.

Pojďme si tady projít celý proces, protože jinde v knize tuto část popisují spíše obecně.

Vložíme obrázky v maximálním rozlišení

Obsahové obrázky webu se hodí ukládat v co největším rozlišení. Nikdy totiž nevíte, jak budou vypadat displeje budoucnosti. Nám se je podařilo ulovit v šírkách kolem dvou tisíc pixelů.

Najdeme nejmenší a největší velikost

Nejmenší velikost obrazovky aktuálních mobilů je 240 pixelů. Dnes už se moc nedělají, ale nějaký podíl na trhu ještě mají. V tomto rozlišení mají obrázky po odečtení okrajů šířku 192 pixelů.

Zaokrouhlíme si to na 200 pixelů a to bude naše nejmenší varianta.

Maximální šířka layoutu je nastavená na `30em`, což je 540 pixelů. Kvůli „Retina“ displejům budeme počítat s dvojnásobkem, tedy 1080 pixelů. Obrázky je vhodné testovat i na zařízeních s více než dvojnásobným poměrem hardwarových a CSS pixelů, ale mám zkušenost, že dvojnásobek obvykle postačuje.

Vyrobíme varianty a uvedeme je do `srcset`

Jak jsem psal v textu o `srcset` a `sizes`, varianty generují po dvě stě a tři sta pixelech. Můžu to zařídit nějakou automatizací na svém počítači nebo na serveru. Pro jednorázovou práci ale doporučuji výborný generátor variant obrázků „Responsive Image Breakpoints Generator“. responsivebreakpoints.com

Ten varianty chytře vytváří podle minimálního kroku datové velikosti. Když jsem nastavil 30kB, dostal jsem následující verze prvního obrázku:

Šířka v pixelech	Velikost v kB
200	12
442	43
617	72
762	100
903	129
1036	160
1080	180

Zapsáno v kódu to vypadá takto:

```

```

Pro další dva obrázky to bude vypadat trochu jinak. Podívejte se pak do HTML zdroje nebo na odpovídající commit na Githubu.
[git.io/vDVjw](https://github.com/robinhansen/vzhuru-do-responzivniho-webdesignu/blob/main/_06_responsive_design/_01_responsive_images.html)

Nastavíme velikost obrázku v layoutu: sizes

Layout a velikost obrázků v něm jsou v tuto chvíli jednoduše zjistitelné. Stačí vzít vývojářské nástroje, zmenšit okno a postupně ho zvětšovat. Všechny body zlomu layoutu tam krásně uvidíte a z vývojářských nástrojů snadno vyčtete patřičné rozměry.

- Na malých displejích zabírá layout celou obrazovku bez postranních okrajů a obrázek jakbysmet: calc(100vw - 2 * 1.5rem).
- Od šířky okna kolem 530 pixelů už se dále nezvětšuje. Šířka obrázku tam zůstává fixně na 480px.
- Od 640 pixelů je zase obrázek široký 540px.

Teď si tři typy layoutu zapíšme do atributu sizes:

```

```

Prohlížeč uplatní první vyhovující pravidlo, proto jsem typy layoutu řadil od největších obrazovek po nejmenší.

3. Tabulka

V tomto rozlišení ještě vypadá hezky. Ale není tabulky, která by se někde nerozpadla. Stačilo by okno zmenšit o trochu více.

Aplikujeme řešení pro posun do stran z podkapitoly o tabulkách.

Uvidíte to v souboru `style/media/rwd-table.css` a následujícím commitu. git.io/vDwJJ

4. Vkládané video

Stačí vzpomenout na „Pružné vkládané elementy se zachováním poměru stran“ ze začátku této kapitoly. A opět se o slovo hlásí trik s `padding-bottom`, který ostatně v responzivním designu budete potřebovat velmi často.

Tuto věc má v našem případě na starost komponenta `style/media/rwd-object.css`.

Aktuální stav příkladu si můžete naživo prohlédnout nebo stáhnout na následujících adresách.

- Otevření v prohlížeči: vrdl.in/vdwdmed
- Stažení v ZIPu: vrdl.in/vdwdmedzip

Dostali jsme se tedy do stavu naformátovaného dokumentu s ošetřeným mediálním obsahem. Teď už pojďme navrhnout pokročilejší prvky uživatelského rozhraní. A pak i layout. Nejdřív ale znalosti, které byste měli mít, pokud si na to chcete troufnout.

Zapamatujte si

- Trik s `padding-bottom` umožní zachovat poměr stran jakéhokoliv elementu.
- Grafiku, kterou lze vyjádřit vektorem, vkládejte do webu v SVG.
- `` vám pro bitmapové obrázky stačit nebude.
- Zvažte využití formátu WebP.
- Do atributu `srcset` značky `` vkládejte seznam předpřipravených variant obrázků. Do `sizes` patří popis breakpointů layoutu webu.
- Značka `<picture>` slouží hlavně pro posílání různých ořezů a souborových typů na různá zařízení.
- SVG je dokument, ne obrázek.
- Ve zdrojovém kódu SVG nastavujte `viewbox`, nikoliv `width` a `height`.
- Uvnitř SVG můžete používat Media Queries. Je to fajn.
- S responzivními tabulkami vám pomůže knihovna Tablesaw.

Kapitola 7: Návrh rozhraní v éře mobilů

Než si prakticky ukážeme, jaký proces pro návrh používám, musím popsat přístup vyplývající ze zkušeností, průzkumů a dat, které za mým uvažováním stojí.

Kapitola by měla aktualizovat vaše znalosti o návrhu rozhraní. Je zaměřená hlavně na novou přítomnost mobilních zařízení.

Postupovat budeme od obecného ke konkrétnímu:

1. Čtyři základní vlastnosti dobrého responzivního rozhraní.
2. Způsob návrhu „Mobile First“. Doporučuje upřednostňovat mobily?
3. Jak lidé používají nová zařízení? Podíváme se také na hybridní a větší dotyková zařízení.
4. Jaká je minimální doporučená velikost aktivní plochy v rozhraní? Co myslíte?
5. „Pravidlo o třech klicích“ budeme ignorovat. Zavedeme pravidlo o otravných dotycích.
6. Tipy, co dělat a co nedělat v responzivním designu. Mám jich opravdu hodně.
7. Časté desktopové zlozvyky webdesignérů. Například karusely, akordeony a efekty po najetí myši.
8. Je rozumné na mobilech schovávat obsah?
9. A nakonec seznam častých chyb responzivních webařů. Internet je jich plný.

Čtyři principy návrhu responzivního rozhraní

My, dnešní webdesignéři, musíme předpokládat, že:

- všechna zařízení mohou být dotyková,
- jeden člověk se na naše weby dívá z různých zařízení,
- nejčastěji je ovládá tím nejtlustším prstem – palcem.

Když jsem s těmito předpoklady pracoval, dospěl jsem k následujícím čtyřem principům návrhu uživatelských rozhraní.

1) Konzistence

Jeden uživatel se na naše weby dívá přes různá zařízení. Proto pokud to není nutné, nenuťme jej učit se ovládání webu na každém zařízení znova. Rozhraní by ideálně mělo být co nejpodobnější napříč všemi zařízeními.

2) Jednoduchost

Když to přeženu, většina uživatelů umí klikat či sahat palcem, posunovat stránku a mačkat tlačítko pro návrat v historii. Nic dalšího pro jistotu nepředpokládám. Proto se pokud možno snažím vyhnout složitějším ovládacím prvkům, jako jsou například karusely, delší formuláře nebo modální okna.

3) Přirozený proud

To, co je na stránce nahoře, je přirozeně důležité, co uprostřed méně důležité a tak dále. Nejpřirozenější směr konzumace informací

v naší části světa je zleva doprava a shora dolů. Ten samozřejmě můžete změnit vizuálním designem, ale musíte si být jistí, že to děláte dobře. Proto se vyhýbám prvkům, které uživatele nutí vracet se proti přirozenému proudu: například záložkovým navigacím uprostřed stránky.

4) Očividnost

Co oči nevidí, srdce nepálí. Jenže co oči uživatele nevidí, srdce designéra pálit může. Lidé prostě schované prvky neotevírají tak často, jak bychom si přáli. Proto je lepší zobrazit alespoň pář položek hlavní navigace, nebudovat závislost rozhraní na ikonách a na důležitých místech uživatelského rozhraní se vyhýbat vysouvacím nabídкам. Luke Wroblewski to hezky shrnul do anglického „Obvious Always Wins“. vrdl.in/t93f2

Tak, dogma je na světě. A teď si jej pojďme aplikovat na způsob návrhu webů.

Filozofie „Mobile First“

Mobile First je způsob návrhu uživatelského rozhraní, který z pohledu důležitosti staví mobilní zařízení minimálně na úroveň tradičních počítačů s velkými obrazovkami.

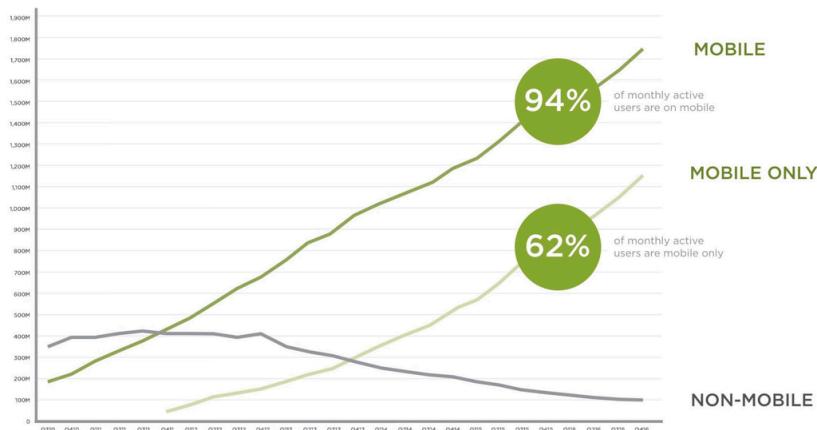
Kromě reflexe nástupu smartphonů na trh má Mobile First ještě jeden – daleko zajímavější – vedlejší účinek. Učí nás navrhovat jednodušší a myslím že i lepší rozhraní.

Autor myšlenky, Luke Wroblewski, ji definoval asi takto:

Designéři, navrhujte nejprve pro mobily. Prudce se šíří mezi uživateli. Nutí zaměřit pozornost na to nejdůležitější. A rozšiřuje naše možnosti.

Mobily budou používanější než desktop. Někde už jsou

Jak už jsem mnohokrát zmínil, podíl mobilů na trhu roste. Uživatelé příručních přístrojů budou jednou tvořit většinu návštěvnosti téměř jistě i na vašem webu. Proto „nejdříve mobily“.

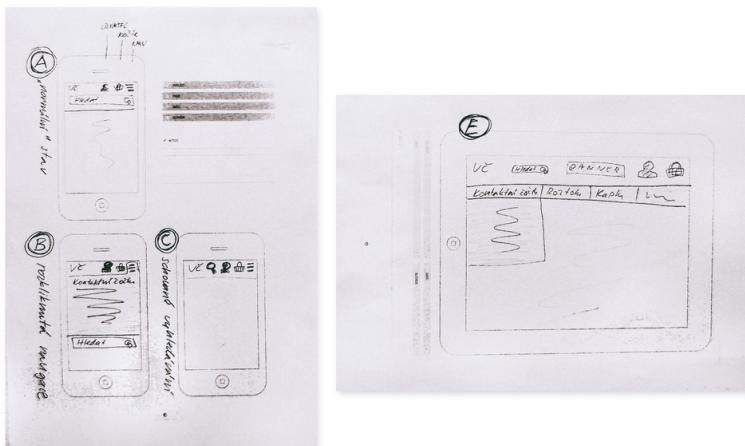


94 % uživatelů přistupovalo ke konci roku 2016 na Facebook přes mobilní zařízení.
62 % jich používalo dokonce výlučně mobilní zařízení. Zdroj: Luke Wroblewski.
vrdl.in/6xnd5

Malý displej nutí designéry zaměřit pozornost na to nejdůležitější

Návrh rozhraní v desktopovém světě počítal s tím, že máme k dispozici velkou plochu obrazovky. Z takto vymyšleného rozhraní se pak ale varianta pro mobily odvozuje velmi špatně.

Pro mnohé designéry, včetně mě, je proto lepší si rozhraní navrhnout nejprve pro ty nejmenší displeje.



Rychlé ruční skici „Mobile First“ návrhů pro VašeČočky.cz

Při skicování mobilního rozhraní jsme omezení plochou. Uživatelské rozhraní pak přirozeně redukujeme na to nejpodstatnější. K přípravě verze pro větší displeje často stačí jen použít běžné techniky responzivního designu: zvětšení nebo mírné přeskládání elementů.

Zjednodušení řešení pak mimo jiné zvyšuje přístupnost webu nebo webové aplikace. Přístup Mobile First pomáhá minimálně ve třech oblastech:

- Zvyšuje srozumitelnost a přehlednost.
- Zavádí snadnější lineární způsob konzumování informací.
- Zvětšuje velikost ovládacích prvků.

Píše to Radek Pavláček, přední expert na přístupnost, ve svém článku „Proč je Mobile First přístup dobrý i pro přístupnost“. vrdl.in/mfprist

Je ale tento nový zjednodušující pohled na návrh jediným důsledkem filozofie „Mobile First“?

Mobily také rozšiřují naše možnosti

Příchod mobilů neznamená pro designéry jen omezení: zmenšení plochy displeje, horší vykreslovací výkon nebo pomalejší internetové připojení.

Znamená také nové možnosti. Mobily máme stále u sebe, poskytují informace o naší poloze a tak dále. Mnoho vlastností dnešních webů a aplikací by bez mobilů vůbec nemohlo vzniknout. Vezměme třeba daleko lepší možnosti lokalizace uživatelů nebo možnost okamžité reakce na newslettery nebo sociální sítě.

Opravdu ale chceme dávat mobily na první místo?

Hlavně ne „Desktop First“!

Uvědomme si, že Wroblewski s myšlenkou přišel v roce 2009, jen dva roky po uvedení prvního iPhonu. Zvolání „Mobile First!“ vzniklo jako reakce na v té době převládající postup. Weby se navrhovaly jen pro velké displeje. Postupem zvaným „Desktop First“.

Mobilní rozhraní se pak vymýšlelo až v implementační fázi, nebo dokonce až po implementaci rozhraní pro velké displeje. Nastávaly

ohromné vývojářské i designérské potíže. Ve výsledku pak velké kompromisy uživatelském rozhraní na mobilech. Že je téma stále žhavé, ukazuje fakt, že v mnoha českých webařských týmech jde stále o aktuální způsob práce. To bolí!

Dnes už nejde o upřednostnění mobilů, ale hledání jednoho řešení vhodného pro všechny

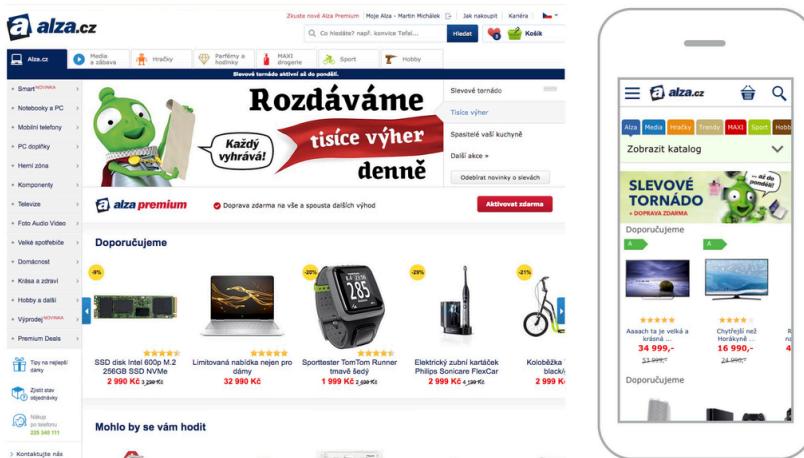
Pojmenování Mobile First vyvolává dojem, že mobilní zařízení je nutné vždy a všude upřednostňovat. Já se ale přikláním k méně vyhraněnému přístupu: všechny typy relevantních zařízení považujeme za důležité.

Pro mě i mnoho dalších je ale efektivnější myslit při návrhu rozhraní nejprve na mobily. Je to těžký začátek, ale šetřím si tak spoustu problémů v další fázích návrhu a implementace.

„Desktop First“ a „Mobile First“ na příkladech

Příkladů řešení „Desktop First“ najdeme v Česku hodně. Vezměme Alzu, která má v době mého psaní dva oddělené weby pro malé a pro velké displeje. Webařský tým Alzy odvádí skvělou práci – stačí se podívat na jejich pozici na trhu. Nesou si s sebou ale zátež desktopového webu. Ten už dnes plně nereflektuje rozšíření mobilních zařízení mezi uživateli.

Jen si například spočítejte, kolik různých navigací má desktopová verze Alzy. Už to samo o sobě nevěstí nic dobrého pro „převod“ do mobilní podoby. V mobilní verzi pak některé navigace „zmizí“, jiné zase vypadají výrazně jinak než na desktopu. Rozhraní tedy z pohledu uživatele trpí nekonzistencí.



Nynější Alza.cz jako příklad „Desktop First“ přístupu. Mobilní verze je samostatný web na jiné doméně. Rozhraní na obou typech zařízení trpí nejednotností

Provozovatelé velkých a úspěšných webů samozřejmě často nechávají zásadní redesign až na chvíli, kdy je nezbytně nutný. Převod do responzivní verze „Mobile First“ v případě Alzy znamená práci na mnoho měsíců až let.

Na druhé straně spektra stojí Maternia, provozovatel e-shopů jako VašeČočky.cz nebo Lentiamo.co.uk a můj vážený klient. Tam se k zásadním změnám rozhodli už při zvažování výroby speciální mobilní verze.

Postupem Mobile First jsme postupně přepracovali celý web. Všechny komponenty rozhraní webu jsou co možná nejjednotnější z pohledu uživatele všech zařízení, ale i z pohledu návrhu a technologie.

Jednodenní čočky

Ráno nastart, večer vydřit. Tak produkuje se jednodenní čočky procházejí. Zde máte možnost ani počítat. Káždý den nové ráno nové par čoček.

Přejdete na stránku s cenou za 1000 Kč za 100 čoček

Eva Vargová
Zákaznický servis

Zobrazit

1 jednodenní (54) | Všechny typy | Všechny výrobce | Všechny čočky | Pokročilé vyhledávání

Produkt	Cena za 100 čoček
Biotrue ONEday (30 čoček)	679 Kč
1 Day Acuvue Moist (30 čoček)	499 Kč
1 Day Acuvue Moist for Astigmatism (30 čoček)	609 Kč
Safelens Daily Disposable (30 čoček) + 30 čoček	449 Kč
Safelens Daily Disposable (30 čoček)	449 Kč
Focus Dailies Progressives (30 čoček)	599 Kč
DAILIES Aquacomfort Plus (30 čoček)	429 Kč
1 Day Acuvue Trifocal (30 čoček)	599 Kč
DAILIES Aquacomfort Plus (90 čoček)	1 229 Kč
Dailies Total 1 (90 čoček)	2 169 Kč
Focus DAILIES All Day Comfort (30 čoček)	429 Kč
Preacle 1 day (90 čoček)	1 262 Kč
Focus Dailies All Day Comfort (90 čoček)	2 169 Kč
Focus Dailies All Day Comfort (180 čoček)	2 169 Kč
Preacle 1 day (30 čoček)	399 Kč
Safelens Daily Disposable (30 čoček)	399 Kč
FreeLook 1 Day (10 čoček) + 10 čoček	369 Kč
Biomedics 1 Day Extra (20 čoček)	519 Kč

Nalezeno dle produktu:

Hledaný výraz: Jednodenní čočky

Ráno nastart, večer vydřit. Tak produkuje se jednodenní čočky procházejí. Zde máte možnost ani počítat. Káždý den nové ráno nové par čoček.

Potřebujete poradit? Ozvěte se mi (po 8-16)
800 114 118 | info@vasecoky.cz

Zobrazit

1 jednodenní (54) | Všechny typy |
Všechny výrobce | Všechny čočky |
Pokročilé vyhledávání

Biotrue ONEday (30 čoček)

VašeČočky.cz jako příklad „Mobile First“ přístupu

Vedlejší, ale podstatnou výhodou je pak jednotnost na úrovni kódu, která Maternii šetří práci a zrychluje vývoj.

Tolik k Mobile First. Původní znění myšlenky hledejte ve slavném Wroblewského článku „Mobile First“. vrdl.in/4slev

V další podkapitole se zaměříme na chování uživatelů.

Lidé a zařízení: jak je ovládají a jak se chovají?

Jak vlastně lidé ta nová zařízení drží, jak je osahávají? Vítejte u mobilní, tabletové a desktopové Kámasútry!

Asi víte, že dotykovost je nový standard. Ale dozvíte se také, že prvky, které mají být snadno dosažitelné, je dobré na mobilech umísťovat na vodorovný střed a na větších zařízeních k pravému

kraji. Pojďme na to.

Všechno je dotykové

Na začátek si dovolím parafrázovat myšlenku Joshe Clarka z jeho skvělé knihy „Designing for Touch“:

Zařízení jakéhokoliv typu může být dotykové. Proto musíme předpokládat, že dotykové bude.

Přesně tak, milí čtenáři, pojďme považovat dotykové ovládání za výchozí stav.

V době, kdy píšu tento text, ještě „dotykáče“ mezi zařízeními přistupujícími na vaše weby nemusejí hrát první housle. Jenže jak už jsem psal dříve, to se brzy změní. Statistiky neúprosně ukazují, že dotyková zařízení jednou snědí počítačové myši i s kabelem.

Ani skupina uživatelů klasických počítačů, zejména notebooků, není nedotčená. Máme tady hybridní zařízení, tedy dotykové stroje s klávesnicí a myší. A jejich prodeje rostou.

Proč nemít dvě verze rozhraní – pro „myšovitá“ a dotyková zařízení?

Bylo by to totiž neefektivní a je dost těžké ta zařízení detekovat.

Nejprve k efektivitě. Představte si, že děláte dvě verze uživatelského rozhraní vaši aplikace. Nevadí vám to? A teď si představte, že to máte všechno platit. Pro nastartování představivosti doporučuji podkapitolu o webech mobilních, responzivních, adaptivních.

Možná si vzpomenete, že jsem proti speciální mobilní verzi webu argumentoval náročností práce i údržby pro designéry i vývojáře.

Ale týká se vlastně všech řemesel souvisejících s webem: uživatelského a technického testování, správy obsahu a dalších. Práce na dvou rozhraních je všechny zpomalí a prodraží.

Z principu je navíc téměř nemožné detektovat dotyková zařízení. Kdyby se lidé dělili na *dotykující* a *myšující*, možné by to jakž takž bylo. Jenže lidstvo je tak trochu zlomyslná parta. Je tu ona rostoucí skupina hybridních zařízení, *dotykujících* a *myšujících* zároveň.

Výjimečně nastává situace, kdy se nějaká detekce hodí. Třeba když chcete pro desktopové rozhraní otevřít prvek po najetí myši. Pak použijte detekční knihovnu Modernizr, která *myšovitá* umí najít.

Obecně se tomu ale snažte vyhnout, protože i tato detekce je nespolehlivá. Uvedu dva z mnoha důvodů. Starší dotyková zařízení například jen technicky emulovala klikání myší, takže je jako dotyková detektovat nelze. A pak tu máme hybridní zařízení. Technicky se jich zeptáte: „Umíš doteky?“ „Ano, umím,“ odpoví. Jenže co když uživatel právě ovládá vaše rozhraní myší? Více o tomto problému najdete v dokumentaci Modernizru, hledejte „touchevents“. modernizr.com/docs

Lidé váš web vidí na více zařízeních

A zase je to tady! Místo toho, aby se nám lidstvo rozumně rozškatulkovalo do dvou táborů – *mobilisté* a *desktopisté* – dělají nám v celé věci neporádek. Představte si, že většina z nich je v obou táborech najednou. Uf!

No tak dobře, teď vážně. Opravdu neexistuje nic jako oddělené tábory mobilních a desktopových uživatelů. Vezměme například studii uživatelů v USA ze začátku roku 2016 od Google:

- 57 % jich využívá více než jedno zařízení;

- 20 % jich dokonce využívá další zařízení, i když zrovna sedí u počítače;
- 39 % lidí vyhledává jen na mobilech, 28 % na různých zařízeních, 32 % jen na počítači;
- 27 % využívá jen mobil a jen 14 % pouze desktop.

Google to měřil na svých amerických uživatelích mezi 18 a 49 lety v prvním čtvrtletí roku 2016. vrdl.in/gdum

Není bez zajímavosti, že uživatelé v průzkumu strávili v průměru 75 minut denně u tabletů, 120 minut u počítače a celých 170 minut pohledem do mobilu. Když držíme mobily, nechodíme snad ani na záchod! Nebo si je na záchod bereme, že ano.

Zařízení nejčastěji držíme jednou rukou a ovládáme palcem

V roce 2013 se Steven Hoober sebral a šel se podívat, jak lidé na ulicích drží své mobilní telefony. Asi vás to nepřekvapí, ale my lidé jsme se nesjednotili ani ve způsobu držení těch malých svítících krabiček.



49 %



36 %



15 %

Výsledek výzkumu Stevena Hoobera: 49 % lidí drželo mobilní zařízení v jedné ruce a šátralo po něm palcem. 36 % lidí dávalo přednost „kolébkovému“ chvatu (držení v jedné ruce a ovládání prstem druhé ruky) a 15 % drželo krabičky obouruč a ovládalo dvěma palci

Hoober ale o držení mobilů zjistil i další věci, které se v zásadě dají zobecnit i pro další dotyková zařízení:

Držíme telefony různě podle kontextu a pozice

Ano, i úchopově přelétaví jsme. Chudáci mobilní telefony. A co teprve tablety!

Dvě třetiny dotyků při držení v jedné ruce se provádí pravačkou

A to i přes to, že leváci tvoří jen asi desetinu, nikoliv zbylou třetinu populace. I my praváci si tedy občas sáhneme levým palcem. Tím méně přesným, mimochodem.

75 % všech interakcí bylo děláno palcem

U jednorukého držení to asi smysl dává, ale palce to vyhrály i u kolébkového chvatu.

Takže my lidé jsme vlastně jen složité mechanismy pro přenášení a ovládání palců. Zdrojový výzkum Stevena Hoobera „How Do Users Really Hold Mobile Devices“ najdete na UXMatters.com.

vrdl.in/m326o

Pozor na malou plochu pohodlně dosažitelných oblastí obrazovky

Prodávají se stále větší chytré telefony, takže bychom při návrhu rozhraní měli myslet na to, že palcem je u nich dosažitelná daleko menší část obrazovky.



Mobily jsou větší, ale plocha ovládatelná palcem se zmenšuje. Větší zařízení totiž také mívají silnější šasi. Uprostřed zvýrazněná zelená je palcem dosažitelná snadno, žlutá hůř a zbylá červená skoro vůbec

Pokud má být prvek rozhraní snadno dosažitelný, bude nejlepší, když jej na mobilech umístíte co nejvíce dolů a co nejvíce doprostřed. Spodní hrana prohlížeče ale pro umístění důležitých prvků dobré místo nepředstavuje. K tomu se ještě vrátím v povídání o fixním pozicování elementů.

Na menších mobilech tedy dělají palce kolem 75 % všech interakcí. Na velkých mobilech kolem 60 %. Uvádí to Josh Clark ve vynikajícím článku „How We Hold Our Gadgets“ na A List Apart, ze kterého budu dále vycházet. vrdl.in/hold

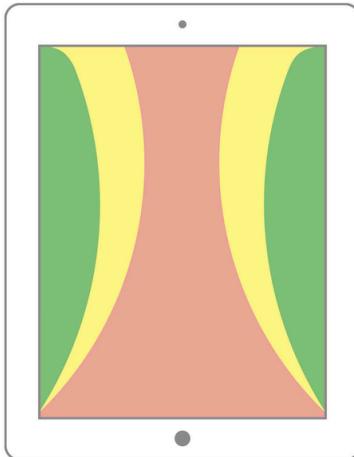
To bychom měli mobily. Jak je to u větších zařízení?

U tabletů značně záleží na jejich velikosti. Josh Clark ve výše odkazovaném článku zmiňuje, že ty menší sedmi- a osmipalcové drží většina uživatelů ještě v ruce a ovládá palci. Větší tablety si zase pokládáme na stůl nebo do klína.

Na otázku, jestli lidé tablety častěji používají na výšku, nebo na šířku, odpovím tak, že to vychází na remízu. Větší tablety používá mírná většina lidí na šířku. Menší na výšku, protože se nám pak lépe drží v ruce.

Malé tablety (7" a 8") držíme častěji obouruč a ovládáme palci

Znovu se zde budu odkazovat na data, která sesbíral Josh Clark. Malé tablety držíme v ruce, ale zóny pohodlného ovládání palci vypadají zcela jinak.



Na malých tabletech držených obouruč jsou palci nejlépe dosažitelné okraje od středu nahoru. Spodní okraj a střed jsou naopak dosažitelné nejhůře

Aktivní prvky, které mají být snadno dosažitelné, bychom tedy měli umísťovat ke kraji. Nejlépe pravému, vzhledem k přesile praváků v populaci.

Zóny pohodlného ovládání na malých tabletech jsou dost v kontrastu s dnes běžným umísťováním málo důležitých prvků na fixní pozice ke krajím obrazovky. Ano, vy lišty z Heureka.cz nebo výzvy ke kliknutí na online chaty, dívám se právě na vás!

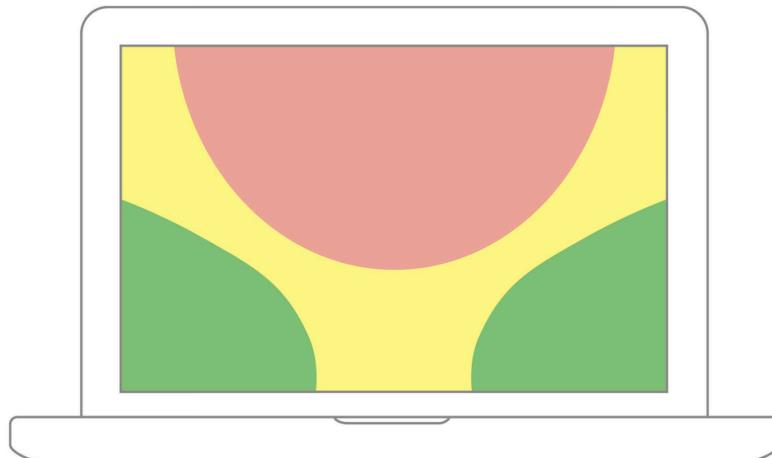
Hybridní zařízení a notebooky s dotykovou obrazovkou: i tady překvapivě vedou palce

I mě, jako člověka nadšeného do dotykových zařízení, překvapilo, že celých 77 % uživatelů notebooků s dotykovou obrazovkou používá častěji doteky než ovládání myší. Vyšlo to alespoň ze studie Intelu, odkazované v Clarkově článku.

O tom, zda lidé budou vůbec někdy chtít ovládat zařízení stojící na

stole doteky, se dlouho vedly spory. Skeptici namítali, že ovládání ukazováčkem člověka nutí držet paže před sebou a to že není pohodlné. Když si vzpomenete na onen vysoký podíl ovládání doteky, budete se asi divit, když prohlásím, že skeptici měli pravdu.

Lidé totiž dotykové obrazovky notebooků a podobných zařízení neovládají ukazováčkem, ale znova palcem. Únavě paží zabraňují tak, že si ruce opřou o stůl pod hranou displeje. Ukazováčkem ohmatávají obrazovky jen uživatelé méně zkušení, kteří většinou časem přejdou znova na palce.



U notebooků s dotykovou obrazovkou a hybridních zařízení jsou palcem nejlépe dostupné plochy spodních rohů

I u těchto zařízení jsou tedy nejsnáze dosažitelné kraje uživatelského rozhraní. Jen nezapomeňte, že je u nich uživatelům potřeba nechat trochu volného místa pro rolování stránky.

V jednom se tedy dotyky na všechna zařízení shodují. Většinou před ostatními prsty upřednostňujeme ovládání palcem. Palce jsou přesné a pro mobilní zařízení univerzálně použitelné prsty.

Primárně bychom tedy měli rozhraní navrhovat pro ně.

Z „touch first“ rozhraní profitují všichni

Nerad bych, aby zapadla důležitá věc. Pokud navrhneme jednotné uživatelské rozhraní a uzpůsobíme je hlavně dotykovému ovládání palci, budou profitovat i majitelé dalších ovládacích prvků: ukazováčků nebo klidně prostředníčků, ale hlavně kurzorů ovládaných myší. Větší aktivní plocha znamená větší pohodlí při ovládání rozhraní.

Minimální plocha aktivní plochy: alespoň centimetr čtvereční

Robert Wadlow byl podle Wikipedie nejvyšším člověkem v historii. Přezdívalo se mu Obr z Altonu. Když navrhoji rozhraní, dost na něj myslím. Hned vysvětlím proč.

Titérné aktivní plochy u uživatelském rozhraní jsou častým hříchem responzivních webů. Ano, kurzorem myší se trefíte skoro na cokoliv.

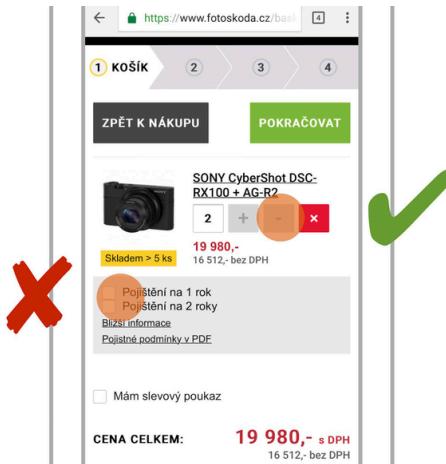
Palec sedmnáctileté dívky bude menší než palec Roberta Wadlowa. Ten totiž měřil 272 cm a vážil okolo 220 kg. Proto raději při vymýšlení rozhraní myslím na Obra z Altonu než na mladé dívky, jestli mi rozumíte.

Co se týká minimální plochy aktivní plochy, moderní webařina se nejčastěji odkazuje na další výzkum Stevena Hoobera, provedený tentokrát ve spolupráci s Patti Shank. Zjistili, že potřebná minimální plocha se různí podle vzdálenosti od kraje obrazovky:

- ve středu obrazovky je 7 čtverečních milimetrů

- na krajích obrazovky je to 11 čtverečních milimetrů

Přiznávám, že ve svých myšlenkách na Obra z Altonu si pravidlo zjednodušuji. Chci aktivní plochu vždy alespoň jeden čtvereční centimetr.



E-shop FotoŠkoda.cz má jeden z těch povedenějších košíků na mobilech. Všechno velké, navigace jednoznačná. Jen prvky v šedivé ploše s „Pojištěním“ by můj palec ani na pětiapůlpatkovém iPhone netrefil. Na výšku alespoň centimetr, prosím

Jak ideální plochu zapsat kódem?

Za předpokladu, že máte správně nastavenou meta značku pro viewport, to dokonce lze zapsat v CSS tak, aby ve všech dnešních mobilních zařízeních byla plocha přibližně centimetr veliká.

Podle Joshe Clarka, kterého cituji v předchozích textech, mají téměř všechna dotyková zařízení rozlišení kolem 160 DPI (CSS pixelů na palec). Přepočtem do centimetrů čtverečních dostaneme při obvyklé výchozí velikosti písma v prohlížečích (16px) tento výsledek:

```
/* 10mm při 160 DPI   63px   4rem */  
.touch { width: 4rem; height: 4rem; }
```

Budou to lidé trefovat palcem? Navrhněte to na plochu centimetru čtverečního. Navrhněte to i pro obra z Altonu.

Na závěr odkážu na zmíněný výzkum Stevena Hoobera a Patti Shank: „*Making mLearning Usable: How We Use Mobile Devices*“. vrdl.in/aug5z

Málo kliků a ještě méně (otravných) dotyků

Víte, my webdesignéři si problémy (jako všichni lidé) občas až moc zjednodušíme. Díky tomu asi před patnácti lety vzniklo „Pravidlo tří kliků“. Doporučovalo, aby veškeré informace byly na vašem webu dostupné maximálně na tři kliknutí myši.

„Pravidlo“ bylo naštěstí mnohokrát vyvráceno:

Uživatelům nevadí extra kliknutí navíc, pokud o nich nemusí přemýšlet.

Přesně tak, tři otravná kliknutí jsou horší než pět jednoduchých. Jak říká klasik Steve Krug už titulem své zásadní knihy: „Nenuťte uživatele přemýšlet.“ Na desktopu máme výzkumy ověřeno, že to platilo a platit bude. A co teprve, když zmenšíme obrazovku!

Tvorba uživatelského rozhraní pro mobily není jednoduchá disciplína. Designér má k dispozici méně prostoru. Uživatel kromě menšího prostoru i zhoršené fyzické podmínky.

Představte si například, že hledáte dopravní spojení v klidu kanceláře. Máte? A teď ve spěchu, v tramvaji. Když zjistíte, že má zpoždění. Tramvaj je přecpaná, pravou rukou křečovitě visíte na

oprátce, kterou vám připravil kat tramvaják. Ten se jako šílený žene za utíkajícím jízdním rádem. Připojení k internetu je tu navíc pomalé. Máte to? Ano, vytvoření dobrého rozhraní pro mobily není jednoduchá disciplína.

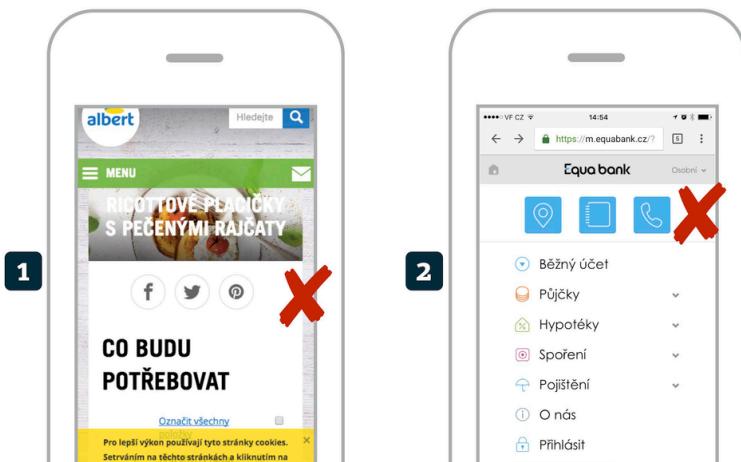
Proto se na mobilech vždy musíme snažit věci zbytečně nekomplikovat a lidem práci maximálně zjednodušovat. Týká se to hlavně míst, kde od nich očekáváte uživatelský vstup nebo složitější interakci. Vzpomeňte si na jeden ze čtyř principů mého dogmatu, o jednoduchosti rozhraní.

Pěknou odmítací argumentaci proti pravidlu tří kliků najdete v textu „Mýtus: Všechny stránky by měly být dostupné na 3 kliky“ na UXGood.cz.vrdl.in/cz1an

Ted' se vrhneme na praktické tipy, co vy na to?

8 tipů pro jednodušší rozhraní na mobilech

Odstraňte zbytečnosti, šetřete ikonkami a rozbalovacími nabídkami. Otevřítejte správné klávesnice, používejte našeptávače. Ale mám i další. Jen čtěte.



Zbytečné ikony sociálních sítí na Albert.cz a podlehnutí ikonománii na EquaBank.cz

1) Dejte pryč všechny zbytečnosti

Na mobilní obrazovce máme k dispozici málo místa, proto tam nechejte jen to opravdu nejdůležitější. O zbytečnosti ikon sociálních sítí budu ještě psát. Na Albert.cz vše zhoršuje ještě pozice nad obsahem. Jakou bude mít uživatel motivaci sdílet obsah hned po přečtení názvu článku?

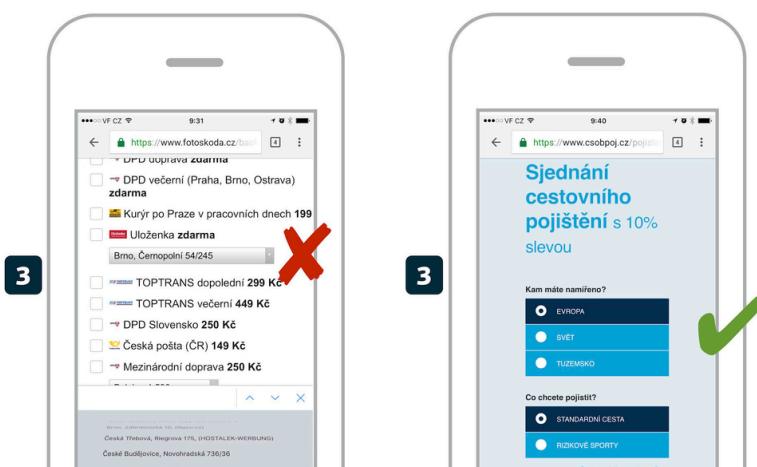
2) Nepodlehněte ikonománii

Mnoho grafiků šetří místo v mobilních rozhraních nadměrným používáním ikon. Ale jak už bylo mnohemkrát prokázáno, většina ikon má velmi nejednoznačný význam.

Podívejme se na web EquaBank.cz na dalším obrázku a udělejme si kvíz: Co má na starosti první, druhá a třetí modrá ikona? První vede do mapové aplikace nebo je to seznam poboček. Najdu tam

i bankomaty? Druhá je... to vážně netuším. Třetí budou asi kontakty. Nebo jen telefon? Bude můj telefon hned volat na zákaznickou linku?

Na většinu ikon se nedá spolehnout. Doplňte je textovými popisy. Více v textu „UX Myth: Icons enhance usability“ na UXMyths.com.
[vrndl.in/7qc2n](http://uxmyths.com/vrdlin/7qc2n)



Špatně použitelný seznam poboček v rozbalovací nabídce na FotoSkoda.cz a chytřejší výběr položek na CSOBpoj.cz

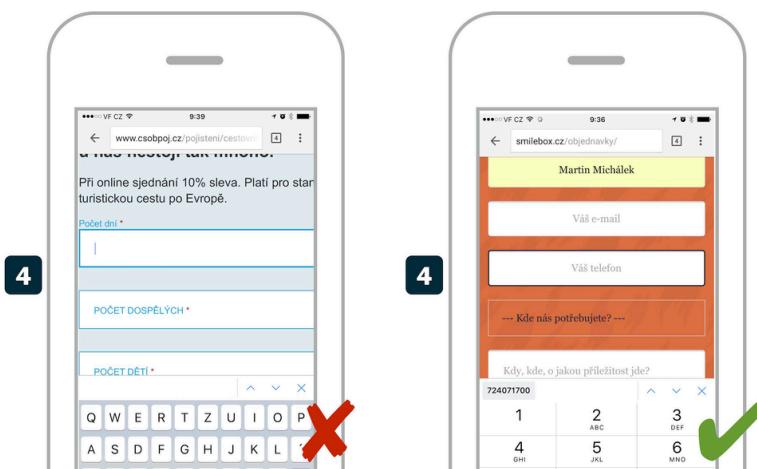
3) Šetřete rozbalovacími nabídkami

S rozbalovacími nabídkami (typu `<select>`) je na mobilech totikéž potíží, až o nich Luke Wroblewski napsal, že je máme použít jen jako poslední záchrannu.

Ani autoři webu FotoSkoda.cz se nevyhnuli nasypání ohromného seznamu poboček Uloženky do rozbalovací nabídky. Zkuste si tam najít tu, která je nejblíž vašemu bydlišti. Třeba v Praze, kde

Samozřejmě všechny ulice znát nemůžete. A představte si počet otravných tapnutí, které takovému rozhraní musíte věnovat. To bolí! Autoři webu CSOBpoj.cz zvolili přívětivější řešení: Rozbalovací nabídky vyměnili za přepínače. Podívejte se na obrázek.

Více také v textu od Luke Wroblewskiego „Dropdowns Should be the UI of Last Resort“.vrdl.in/gad1e



Špatně použitelná alfanumerická klávesnice na CSOBpoj.cz a lepší numerická na SmileBox.cz

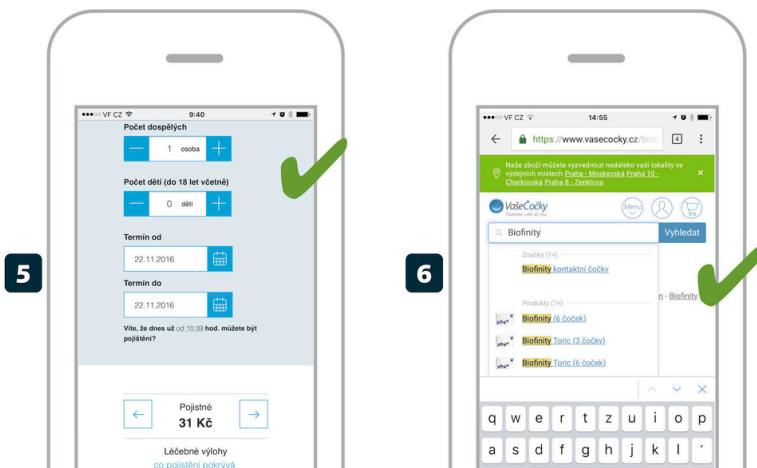
4) Otevřejte pohodlné klávesnice

Web CSOBpoj.cz nutí uživatele vyplnit číselný údaj na alfanumerické klávesnici. SmileBox.cz to má vymyšleno lépe, ten otevře klávesnici numerickou. Podívejte se na obrázek.

Kdykoliv po uživateli chcete vyplnit telefonní číslo, volte specifický typ formulářového pole. Hodí se pro vkládání telefonů (`<input type="tel">`), e-mailů, URL adres nebo na vyhledávací pole. Více

informací najdete na speciální stránce MobileInputTypes.com.
mobileinputtypes.com

A ještě, prosím: Telefonní čísla na stránce vždy na mobilních zařízeních vypisujte jako odkazy. vrdl.cz/b/57-href-tel



Krokovač na CSOBpoj.cz a našeptávač u vyhledávání na VaseCocky.cz

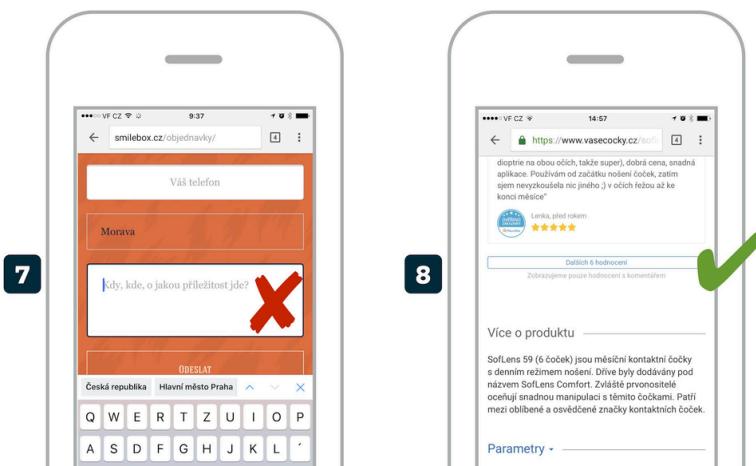
5) Používejte krokovače a další alternativní formulářové prvky

Opět se vracíme k náhradě nešťastného `<select>`. Krokovač (stepper) pomocí běžného HTML neuděláte, ale za vylepšený uživatelský prozítek pár řádků Javascriptu stát může. Zvažte i další alternativní formulářové prvky. Hledejte je rovnou v angličtině: *Radio Group, Button Input, Slider nebo Segmented Control*.

6) Vyhledávání doplňujte našeptávačem

Je velmi dobrým pomocníkem ve formulářových polích, kde je velké množství možných vstupů: Hlavně ve vyhledávání, které je na mobilech kvůli časté nepřítomnosti plnohodnotné navigace velmi důležité.

V HTML pro ten účel existuje prvek `<datalist>`. Jeho využití je ale omezené. Běžné našeptávače jsou v podobně pluginů dostupné pro každý moderní javascriptový framework.



Nutnost psát na SmileBox.cz a rozbalování méně významného obsahu na VaseCocky.cz

7) Nenuťte mobilního uživatele psát

Tohle by stálo na webu SmileBox.cz za vylepšení. Web po uživateli chce ručně vypsat „kde, kdy a jak“ chci přístroj pronajmout. Rychlosť psaní není na mobilu nejvyšší. Problém by lépe vyřešily tři na pár

kliků ovladatelná vstupní pole.

8) Neprotahujte stránku

Spoléhám na to, že uživatelé stránku posunovat umí. To ano. Neznamená to ale, že stránka by měla být dlouhá jako ponožky Pipi dlouhé punčochy. Hezký objev od UX konzultanta Jana Kvasničky vidíte na dalším obrázku.



Pro použití některých stránek bychom potřebovali trošku vyšší telefon. Tady je vidět předkošik na Smarty.cz. Je to modální okno, které se objeví po přidání zboží do košíku. Důležité aktivní prvky jsou červeně orámované. Zdroj: Jan Kvasnička. kvasnickajan.cz

Dlouhá stránka kromě jiného taky odsunuje spodní část rozhraní – patičku a sekundární navigaci v ní. Hledejte alternativní způsoby zobrazení obsahu vkládaného do stránky:

- offcanvas (vysunování obsahu ze strany)
- modální okna nebo karusely (jen pozor na správnou implementaci, zmíním se za chvíli)

- roztahovací akordeony (opět je brzy zmíním)

Když už jsem zmiňoval Jana Kvasničku, vřele doporučuji jeho text a přednášku „Nejčastější chyby při návrhu mobilního a responzivního webu prakticky“. vrdl.in/2tghs

Opusťte desktopové zlozvyky

S pozvolným přechodem většiny uživatelů od myši k dotykovému ovládání přibývá adeptů na zápis do „Červené knihy ohrožených návrhových vzorů“. Pojďme na krátký výlet do pavilonu designérské ZOO, kde je všechny vystavují.

1) Efekt na najetí myši? Už jen jako vylepšení

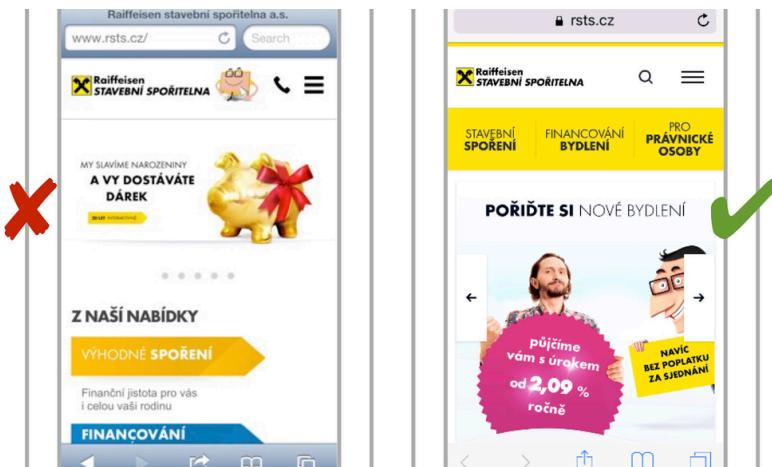
Jak už jsem zmiňoval, jakékoli zařízení může být dotykové, proto musíme dotykové ovládání považovat za výchozí stav. No a na dotykových obrazovkách máme s :hover efekty po najetí myši smělu. Ano, v myší ovládaných zařízeních můžeme přidat vylepšující efekt nebo vrstvu s doplňujícími informacemi. Rozhraní by ale mělo být použitelné bez nich.

2) Karusely jsou fakt složitý dorozumívací prostředek

Karousel je pro designéra i uživatele docela výzva. Z mnoha studií zpochybňujících jejich efektivitu vyberme tu od Erika Runyona. Ten změřil, že ze všech kliknutí na jeho karousel patřilo téměř 90 % jen prvnímu obrázku. Míra prokliku ostatních obrázků se pohybovala mezi dvěma a třemi procenty. vrdl.in/50zuo

Neznamená to, že všechny karusely jsou špatné. Navrhnut dobrý je

ale vážně složité. Osobně po něm sáhnu, až když všechny ostatní možnosti selhávají.



Stará a nová verze karuselu na RSTS.cz. Starší ještě používala na mobilech nepoužitelnou tečkovou navigaci. Nová (vpravo) je fajn. Díky šípkám se lépe ovládá a grafika je uzpůsobená velikosti displeje

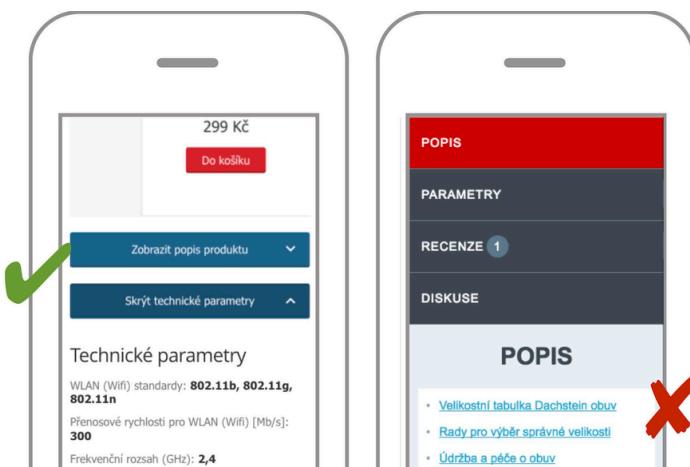
Karusely jsou určitě špatné, když:

- je používáte jako zkrášlující prvek – věc, která se vám líbí;
- je používáte jako univerzální schovávač toho, co na webu nechcete a co si „klient vymyslel“;
- zpomalují načtení nebo zobrazení stránky;
- nejsou použitelné na mobilních zařízeních (malé tečky jako navigace atd.);
- nepropagují obsah na dalších „slajdech“.

Hodně bych si rozmýšlel dnes bohužel běžné použití karuselu pro propagaci aktuálních akcí na úvodní stránce webu. Karusely se ale naopak hodí pro seznamy souvisejících položek. U e-shopu jde třeba o fotografie produktu nebo seznam podobného zboží.

3) Akordeony raději než záložky uprostřed stránky

Záložková navigace uvnitř stránky je na mobilech ke zvážení, protože může otevírat obsah, jehož studium vám ony záložky odsune do nahoře skrytých částí stránky.



CZC.cz má hezky provedený akordeon. Podobný prvek na Mall.cz se ale chová jako záložky: pokud chci po přečtení obsahu pro „Popis“ otevřít „Parametry“, musím posunovat stránku zpět nahoru

I proto mám raději takzvané akordeony. Podporují přirozené plynutí informací shora dolů. Na druhou stranu – akordeony dávají díky výpisu pod sebou informacím (někdy nechtěnou) hierarchii. Jak tedy sami vidíte, při výběru řešení vždy záleží na obsahu, který prezentuje, a celkovém kontextu, ve kterém jej používáte.

4) Uměl by si Obr z Altonu na vašem webu vybrat datum?

Vložení data na mobilech může být pěkná otrava. Hlavně pokud web používá některý z pluginů určených pro počítačové weby. A že je používá každý druhý responzivní web!

Na mobilech můžete využít `<input type="date">`, který otevře nativní výběr data, ale ten taky neřeší všechny potřeby uživatelů. Občas je potřeba udělat vlastní komponentu. Vždy mějte v prvé řadě na paměti ovládání palcem. Obr z Altonu se na vás dívá!

Další tipy od Nielsen Norman Group k výběru data jsem sepsoval na blog. vrdl.cz/b/83-nng-input-date

5) Nespoléhejte na přítomnost globální navigace

Web bez hlavní navigace? Pfff...! Představte si, že byste to nějakému klientovi navrhli před pěti lety. Dnes ale na velmi malých displejích postrádají globální navigaci téměř všechny weby. Prostě se tam nevejde.

Nezbývá než se s tím smířit a na globální navigaci zase tak moc nestavět při návrhu navigačního schématu pro web. U větších webů hraje na mobilech velkou roli vyhledávání. Může být také vhodné stavět úvodní stránku složitějších webů jako rozcestník. Chybějící hlavní navigaci je ale potřeba mít v hlavě při návrhu každé komponenty a každé stránky webu. Téma ještě více [otevřeme v desáté kapitole](#).

6) Modální okna, lightboxy: pozor na správnou implementaci

Modální okna sama o sobě nejsou špatná. Dokonce bych řekl, že jsou na mobilech velmi užitečná. Jen mám asi smůlu – smůlu na weby, které křížek pro zavírání modálního okna schovávají pod horní hranu okna prohlížeče.

Specifickou odrůdu modálních oken, otravná modální okna, dokonce Google považuje za hodné penalizace. Více se dočtete v článku Pavla Ungra „Google od ledna 2017 penalizuje weby s obtěžujícími popupy“. vrdl.in/googlepopup

Modálními okny a lightboxy ukončíme výčet nejvýznamnějších zástupců „Červené knihy ohrožených návrhových vzorů“. Teď se zamyslíme nad schováváním obsahu na mobilech.

Proč na mobilech neschovávat obsah?

Jako příklad si vezmeme tabulkou fotbalové ligy.

Pořadí	Tým	Z	V	R	P	Skóre	Body
1	Plzeň	15	11	3	1	26:9	36
2	Slavia	16	10	5	1	30:12	35
3	Sparta	16	9	5	2	30:13	32
4	Zlín	16	9	4	3	27:18	31
5	Ml.Boleslav	16	8	4	4	28:18	28
6	Teplice	16	6	4	6	18:15	22
7	Karviná	16	6	3	7	22:26	21
8	Slovácko	16	4	6	6	20:25	18
9	Bohemians 1905	15	5	3	7	15:22	18
10	Jablonec	15	4	5	6	23:25	17
11	Dukla	16	4	4	8	21:22	16
12	Hradec	15	5	1	9	17:26	16
13	Liberec	16	3	6	7	11:17	15
14	Brno	16	2	9	5	19:28	15
15	Jihlava	16	2	7	7	11:23	13
16	Příbram	16	2	3	11	12:31	9

Poř.	Tým	Z	Skóre	Body
1	Plzeň	15	26:9	36
2	Slavia	16	30:12	35
3	Sparta	16	30:13	32
4	Zlín	16	27:18	31
5	Ml.Boleslav	16	28:18	28
6	Teplice	16	18:15	22
7	Karviná	16	22:26	21
8	Slovácko	16	20:25	18
9	Bohemians 1905	15	23:22	18
10	Jablonec	15	23:25	17
11	Dukla	16	21:22	16
12	Hradec	15	17:26	16
13	Liberec	16	11:17	15
14	Brno	16	19:28	15
15	Jihlava	16	11:23	13
16	Příbram	16	12:31	9

Tabulka pořadí fotbalové ligy v prosinci 2016. Zdroj obsahu: sport.cz

Pro ty, kdo jsou stejně jako já zcela mimo fotbalové dění, vysvětlím, že „Z“ je počet zápasů, „V“ je počet výher, „R“ remíz a „P“ proher. Tohle v mobilním pohledu úplně chybí. A protože mě už trochu znáte, víte, že tady začnu zvedat obočí.

Vezmeme to ale od začátku. Pojdme si představit, že navrhujeme řešení pro konkrétní web. Víme, že návštěvníci jsou běžní fotbaloví fanoušci. O fotbale něco ví, ale nepotřebují detailní informace jako třeba sázkaři. Obsah tabulky známe a teď vymýslíme řešení designu na malých displejích. Do zařízení s rozlišením 320 nebo nedej bože 240 pixelů se nám celá tabulka nevejde, že? Neodstraníme prostě na mobilech některé sloupečky?

Informace na vašich stránkách mají být dostupné na všech zařízeních

Je pravda, že informace v takové tabulce mají nějakou hierarchii:

Pořadí, tým a počet bodů jsou nejdůležitější. Skóre a počet zápasů hned za nimi. Výhry, remízy a prohry méně důležité a logo týmu na posledním místě. Na obrázku jsem poslední jmenované v mobilním pohledu odstranil. Jenže...

Logo týmu zde sice nese informaci, kterou už můžete získat z názvu týmu, ale je důležité navigačně. Na mobilu se lidem bude bez loga oblíbený tým hledat hůř. Schováním loga jsem sice ušetřil místo, ale zároveň zhoršil uživatelský prozitek.

Výhry, remízy a prohry jsou sice méně důležité, ale... proč je pak máme na velkých displejích? Pokud jsme na základě rozboru cílových skupin došli k tomu, že uživatelé tohoto webu takové informace potřebují, mají tady být na všech zařízeních. Vzpomeňte si, když jsem zmíňoval, že jeden člověk vidí vaše weby na různých zařízeních. Na jiném webu, třeba obecně zpravodajském, by pak nepřítomnost těchto informací smysl dávala.

Jak bych tedy na tabulku na mobilech vyzrál? Použil bych řešení „Posun do stran s fixním sloupcem“ z podkapitoly o tabulkách. Ponechal bych v ní všechny informace a zajistil fixní pozici sloupečků s pořadím, logem a názvem týmů. Ostatní sloupečky bych pak nechal uživatele posouvat do stran.

Vyhnete se častým chybám

Shrňme si tady problémy, kterým se důsledněji věnuji v jiných kapitolách, ale které mají společné jmenovatele v četnosti výskytu na českém a slovenském webu.

1) Pomalu se načítající web

Asi největší hřích dnešních responzivních webů. Rychlosť načítání má vliv na úspěšnost webu a ovlivňuje řazení ve výsledcích vyhledávání Google. Mobilní sítě jsou pomalé a jen tak se nezrychlí. Nezapomeňte rychlosť webu řešit už při návrhu a konzultovat ji v širším týmu. Už jsme se tomu věnovali v kapitole o rychlosti.

2) Navigace schovaná do ikony i tam, kde to není nutné

Responzivní weby často na mobilních obrazovkách schovávají navigace do různých ikon. Z průzkumů ale vyplývá, že ikony bez popisků jsou pro uživatele často nesrozumitelné. Ani samo schovávání navigace není nejlepší nápad. Také si o tom ještě něco povíme.

3) Přizpůsobení jen některým rozlišením

Jak už jsem zmiňoval dříve, dnešní weby se zobrazují v oknech mezi 240 a 2600 pixely. Oba extrémy nejsou příliš časté, ale například podíl obrazovek s velkým rozlišením roste. Při návrhu a testování je potřeba myslet jak na ta nejmenší zařízení, tak právě i na velké stolní displeje.

4) Zakázané přiblížení (zoom)

Uživatelé si zvětšují výřezy obrazovky z mnoha důvodů: kvůli špatnému kontrastu na sluníčku, kvůli snadnějšímu výběru textu nebo se jen chtějí podívat na detail fotografie na stránce. Je to stejně přirozené jako posun stránky nahoru a dolů a občas to potřebují

udělat snad všichni uživatelé.

Rozhodně si nenamlouvezte, že jste web „optimalizovali“ pro mobily a že zoomování potřeba není. Uveďte správnou meta značku pro viewport.

Norma WCAG (doporučení pro přístupné weby) trvá na tom, že vše musí být možné alespoň dvakrát zvětšit. vrdl.in/cbe5f

Prohlížeč Safari na operačním systému iOS verze 10 a novějších už naštěstí „zákaz zoomování“ ignoruje.

5) Neošetřené načítání webfontů

Taky vás števe problikávání obsahu stránky při jejím načítání?

Použití webfontů je v pořádku, ale vývojář musí mít vše pod kontrolou. Různé prohlížeče totiž načítají webfonty různě. Je potřeba si tedy vybrat způsob načítání, který vyhovuje konkrétnímu webu. Pro převzetí kontroly nad načítáním doporučuji malou knihovnu FontFaceObserver. fontfaceobserver.com

6) Vkládání zbytečných sdílecích tlačítek

„Lajkovací“ nebo sdílecí tlačítka Facebooku a dalších sítí jsou na webech velmi často k ničemu. Komplikují uživatelská rozhraní a zpomalují načítání. Funkce sdílení je navíc součástí všech mobilních operačních systémů. Zvažte, jestli vám umístění tlačítek stojí za to. Případně zvolte alternativní, úspornější řešení. Například knihovnu Social Likes. social-likes.js.org

7) Fixně pozicované navigační a propagační elementy

Tady mám poněkud radikální postoj. Navigační a propagační elementy, které při rolování stránky drží pozici, dělají na mobilech více škody než užitku.

Týká se to všech fixně umístěných navigačních lišt i překryvných vrstev s reklamou na newslettery a mobilní aplikace. Nebo v poslední době těch nešťastných tlačítek pro otevření chatu.

Ptáte se proč?

Zmenšují už tak skromný prostor

Telefony s menšími rozlišeními, jako 320×480 nebo dokonce 240×320 pixelů, stále žijí. Problém to ale bude prakticky na každém čtyřpalcovém displeji.

Kolidují s ovládacími prvky prohlížečů

Fixně pozicovanou horní navigaci často překrývá vysunovací lišta s adresním řádkem prohlížeče. Prvky fixně umístěné dole zase kolidují s výsuvnou spodní lištou prohlížeče Safari na iOS.

vrdl.in/houb1

Komplikují posun obrazovky

Postranní fixní lišty zase nevinným palcům vezmou prázdné okraje, kterými jinak spokojeně rolují stránku. Kolikrát se vám stalo, že jste takovou lištu při rolování omylem otevřeli? Mému palci mnohokrát, mohl by vám vyprávět.

Rozbíjí přiblíženou stránku

Jak už jsem psal – zakazovat zoomování považuji za chybu. Co ale udělají vaše fixně pozicované elementy pokaždé, když si uživatel

stránku přiblíží? No jasně, rozbijí layout.

Jsou problematické na méně výkonných zařízeních

Na starších a méně výkonných mobilech vás position: fixed nebude poslouchat a při posunu stránky občas neudrží vámi vysněnou pozici.

Často se argumentuje tím, že fixně pozicovaná navigace umožní uživateli ovládat web na obzvlášť dlouhých stránkách. Na to už jsem namítl, že stránky na mobilech *obzvlášť dlouhé* prostě být nemají, však víte. Na rozumně dlouhých stránkách je návrat zpět nahoru snadný díky možnosti rychlého posunu stránky (tzv. „momentum scrolling“). Tou jsou vybaveny všechny mobilní prohlížeče.

Pro další popis i hlubší argumentaci vás pošlu na další své texty:

- Dvoudílná série „Jak zničit mobilní uživatele?“ na Vzhůru dolů. vrdl.in/lq5b4 a vrdl.in/sacjz
- Můj text „How To Poison The Mobile User“ na Smashing Magazine. vrdl.in/h8n7i

Zapamatujte si

- Vizuální rozhraní musí být konzistentní a jednoduché. Důležité prvky očividné. Dejte přednost přirozenému proudu informací zleva doprava a seshora dolů.
- Přístup „Mobile First“ nutí designéra zaměřit se na to nejdůležitější.
- Nejde nám o upřednostnění mobilů, ale hledání jednoho řešení vhodného pro všechny.
- Dotykové ovládání musíme považovat za výchozí stav.
- Jeden člověk váš web navštěvuje na různých zařízeních.
- Pro ovládání dotykových zařízení lidé používají hlavně palce.

I u hybridních notebooků.

- Aktivní plochy dělejte alespoň jeden čtvereční centimetr velké.
- Uživatelům nevadí extra kliknutí navíc, pokud o něm nemusí přemýšlet.
- Šetřete ikonami, nabídkami v <select> a karusely. Nenuťte uživatele psát delší texty a neprotahujte délku stránky.
- Otevírejte správné softwarové klávesnice, používejte krokováče a vyhledávání s našeptávačem.
- Neschovávejte na mobilech obsah. Jen dekorativní prvky.
- Nevkládejte do webů zbytečně tlačítka sociálních sítí.
- Vyhnete se fixně pozicovaným elementům.
- Nezakazujte zvětšování stránky zoomem.

Kapitola 8: Design komponent rozhraní

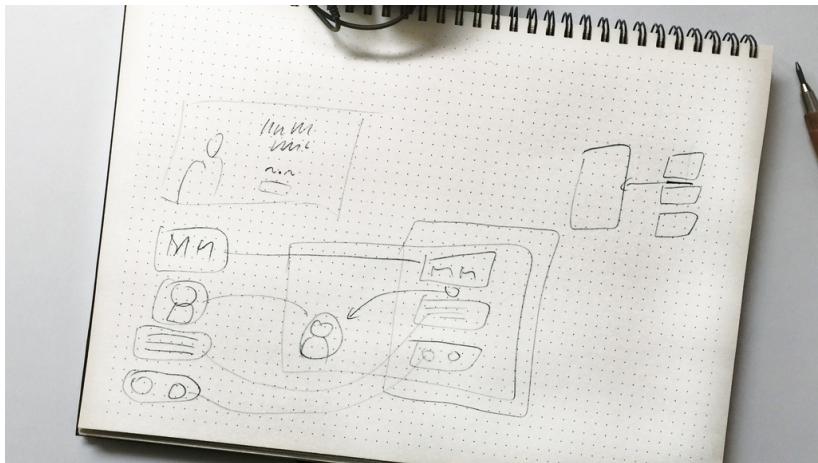
Ted' už přichází na řadu přemýšlení nad návrhem složitějších částí uživatelského rozhraní. Navigace, fotogalerie, hlavičky nebo patičky celých webů. Vsadím se s vámi, že alespoň jeden takový prvek někde na stránce máte.

1. Naučíme se základy skicování.
2. Budu obhajovat testování myšlenky na HTML prototypech.
3. Postup skicování, prototypování a implementace si nakonec projdeme na komponentě fotogalerie v příkladu.

Skicování

Skicování je metoda rychlého zhmotnění myšlenky. Skicuji velmi rád, skicuji tužkou na papír, skicuji v různých situacích a fázích projektu.

Na papír si skicou odložím základní obrysy problému a uvolním mentální kapacity na jeho vyhodnocování nebo na hledání řešení.



Skicování je výborné pro zjednodušení řešení designérských, ale i kodérských problémů. Zpětně těžko říct, co jsem řešil právě tady, ale ušetřilo mi to moc času, to se vsaděte

Skicuji tužkou na papír

Při skicování je důležité, že mi umožňuje soustředit se na daný problém. Existuje celá řada softwarových nástrojů, ale ty mě rozptylují počítačovým prostředím.

Používám prostě tužku, gumu a papír. Když mám na výběr, volím bílý papír ve formátu A3 nebo předtištěné šablony. Na konci textu najdete tipy na nástroje pro tisk šablon.

Jiní skicují popisovačem Sharpie nebo elektronicky v nějaké aplikaci na tabletu. Zkrátka – vezměte, co vám vyhovuje a v čem jste rychlí, a dosáhnete požadované úrovně kvality.

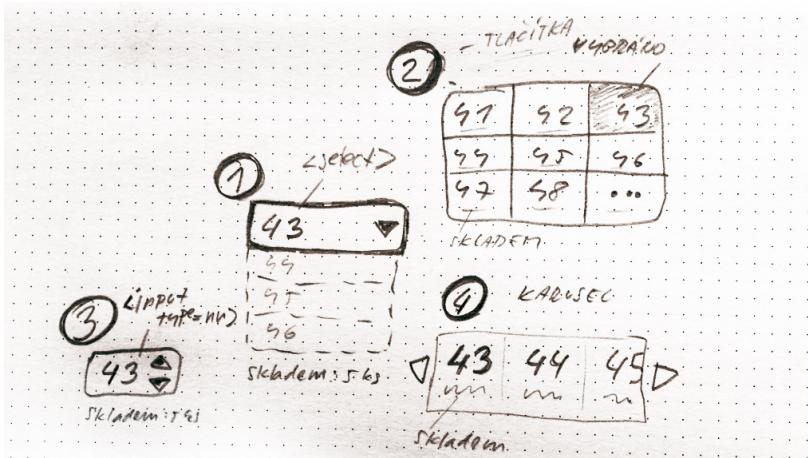
Na rozdíl od uměleckého skicování nemusíte u náčrtů uživatelského rozhraní umět kreslit. Stačí vám jakžtakž zvládnout úsečky,

obdélníky, kružnice a psaní. To by šlo, ne? Pokud si myslíte, že mé skici jsou pěkné, je to proto, že ty úplně první náčrty tady nevidíte. Výhoda skicování je totiž také ve snadném měnění.

Dále se budu zabývat jen specifickým typem skicování: rychlým náčrtům variant designérského řešení.

Skicování variant designérského řešení

Jsem na začátku přemýšlení o designu konkrétní komponenty. Skicování v téhle fázi používám proto, abych vygeneroval co nejvíce variant řešení uživatelského rozhraní. Hlavně žádné hodnocení, žádné přemýšlení. Jen kreslím jako šílený.



Tady jsem hledal optimální způsob prezentace výběru čísla bot v detailu produktu na e-shopech. Tohle už je uhlazenější, prezentovatelnější verze

Mozek zapojím až v další fázi. Dívám se na jednotlivé skici a přemýšlím o jejich výhodách a nevýhodách. Samozřejmě si řešení představím na všech možných zařízeních a beru v potaz zvyklosti,

cílové skupiny a technickou řešitelnost.

Vybranou variantu pak musím ověřit ve světě, ve kterém komponenta bude žít – v prohlížeči. Živým prototypem, na který se podíváme v dalším textu.

Odkazy k dalšímu studiu skicování

- Jako průvodce pro detailnější zkoumání si zvolte Michala Maňáka a jeho článek „Začněte efektivně navrhovat produkty díky skicování“. vrdl.in/8ydb6f
- Styl a úroveň věrnosti skic jsou volitelné. Jak moc se mohou lišit skici designérů v rámci jedné firmy, hezky ukazuje článek „The different sketch styles of the designers at 37signals“ od Jasona Frieda. vrdl.in/pmuio
- Předtištěné šablony mobilních zařízení od SneakPeekIt. sneakpeekit.com
- Pěkný je také Gridzzly, nástroj Rostislava Blahy, který vám umožní tisk požadované mřížky na čistý papír. gridzzly.com

HTML prototypy

Když vyslovíme slovo „prototyp“ v hospodě – a nebude přitom zrovna plná webařů –, asi si okolo sedící představí testovací model nějakého výrobku.

Prototyp je obvykle definován jako *raný* vzorek, který byl vytvořen pro *otestování* myšlenkového konceptu nebo pracovního procesu. Na výsledek testu se buď naváže v reálné výrobě, nebo jej zahodíme, poučíme se a zkusíme to jinak.

Každý dobrý řemeslník věci nejprve promýší a pak teprve vyrábí.

Produkční výroba ale bývá v některých oborech velmi drahá.
Přísluší „třikrát měř, jednou řež“ platí i u webů.

Proto třeba výrobci automobilů, ale právě i webů sahají k nějaké formě zkušebních modelů. Prototypů.

Dvakrát měř, jednou to zkus na prototypu a pak teprve řež

Díky prototypu si můžeme naživo „osahat“ věci, které nám na „papíře“ dávaly smysl. V našem případě hlavně otestovat složité nebo inovativní prvky uživatelského rozhraní. Nebo si zkusit výrobní postupy, které běžně neděláte.

Prototypování je učení na produktu, který je co nejpodobnější tomu cílovému. Způsobů, jak testovat vymyšlené, je ale ve webdesignu hodně. Já tady budu hájit svou oblíbenou cestu – prototypování přímo v HTML, CSS a Javascriptu.

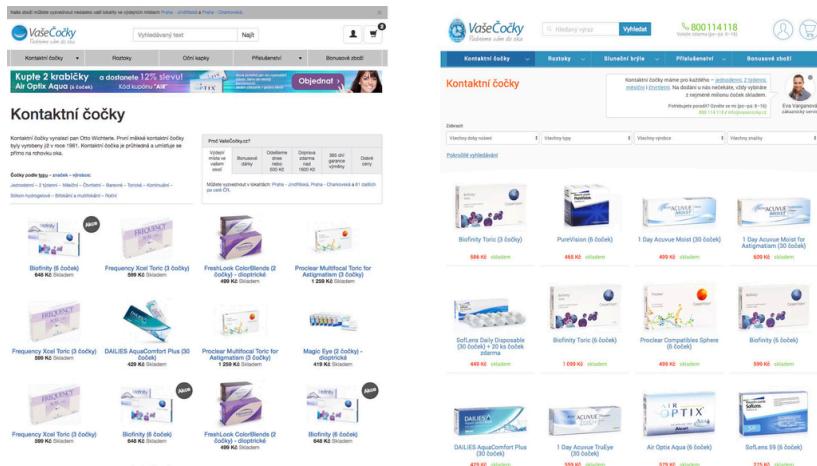
Co je to HTML prototyp a jak se liší třeba od Axure?

Exporty do prohlížečové verze dnes nabízí leckteré „klikací“ nástroje – například Axure RP. Exporty jsou ovšem často limitované tím, co zvládá klikací rozhraní. Plnohodnotné HTML, CSS a Javascript nabídnou vždy výrazně větší možnosti vyjádření.

Pokud vám Axure RP a podobné *wireframovací* nástroje vyhovují, není důvod je měnit. Kromě snadnější práce pro designéra umožňují testovat interakce a další věci. V mnoha týmech mají podobné nástroje nezastupitelnou roli. S HTML prototypováním se nevylučují a mohou se hezky doplňovat.

HTML prototypy, o kterých tady mluvím, jsou prostě plnohodnotné

webové stránky vytvořené technologiemi, které jsou v dnešních prohlížečích po ruce. Jen jsou cíleně zjednodušené – ořezané o některé atributy, které by finální stránka mít měla a prototyp nemusí.



Jeden z raných HTML prototypů VašeČočky.cz a tatař stránka na finálním webu, ke které jsme dospěli iterativním procesem

HTML prototypy mají své výhody

Jsou brzy v prohlížeči, mají volitelnou míru věrnosti a mohou být znova použitelné a snadno udržovatelné.

Brzy v prohlížeči

Vzpomeňte na slovo „raný“ v definici prototypu. Máme problém, který by se složitě řešil na papíře nebo ve Photoshopu? Pak s tím co nejdříve do reálného prostředí – do prohlížeče!

Plnohodnotné

Prototypy tvořené webovými technologiemi jsou ze stejného těsta

jako hotové weby. Ve vybraných aspektech lze dosáhnout absolutní míry věrnosti s konečným produktem.

Znovupoužitelné

Výstupy z jiných prototypovacích nástrojů obvykle po skončení práce zahazujeme. V případě HTML prototypů na ně mohou vývojáři i designéři navázat. Je to pořád jedno těsto. Ale znovupoužitelnost je mince o dvou stranách, jak se dočtete dále.

Snadno udržovatelné

Snadné provádění globálních změn, zbytečné neopakování se... To jsou parametry efektivní práce, kterých je v běžných prototypovacích nástrojích složitější dosáhnout. Co se týká efektivity udržování, je webový kód bezkonkurenční.

HTML prototypy mají samozřejmě také své nevýhody. Ještě chvíli vydržte, povím vám něco o svých zkušenostech s nimi. Nejdříve si ale prototypy pojďme zasadit do kontextu dalších webařských modelovacích nástrojů.

Prototyp versus drátěný model a maketa

Nástroje používané pro demonstraci a odzkoušení nějaké myšlenky můžeme rozdělit do tří skupin:

Drátěný model (wireframe)

- Je to vlastně kostra pro webový design. Ukazuje obsah, možnosti jeho rozložení a případně základní interakce. „Dráťák“ je rychle hotový, ale velmi zjednodušuje. Obvykle je to levná volba, ideální pro úvodní fáze projektů.
- Model nízké věrnosti co do podoby s finálním webem.
- Nejméně časově náročný.
- Nástroje: ruční skicování nebo programy jako UXPin nebo

Balsamiq Mockups.

Statická maketa (mockup)

- Na jednu stranu velmi detailní, jiné atributy prototypu ignorující. Typická maketa je výstup z Photoshopu, který obvykle detailně popisuje vizuální design. Interakce nebo chování v různých rozlišeních statická maketa popisuje naopak zcela nedostatečně.
- Model střední až vysoké věrnosti.
- Časově středně a hodně náročná.
- Nástroje: Photoshop, Sketch.

Interaktivní prototyp

- Důraz na interaktivitu, a tedy například možnost otestovat uživatelské scénáře. Hezký dokáže otestovat i typické frontendové problémy. Vizuální design ale obvykle moc neřeší.
- Opět střední až vysoká věrnost.
- Časově ze všech možností nejnáročnější.
- Nástroje: Naše HTML prototypování nebo z části Axure RP.

Zjednodušeně řečeno: HTML prototypování nabízí nejvyšší možnou míru věrnosti, ale za cenu největší pracnosti.

Výhody pro designéry

Designéři uživatelského rozhraní a weboví grafici mohou ocenit možnost na prototypech testovat následující:

1. Responzivnost

Častý problém maket z grafických programů je statický výstup. UI komponentu není možné otestovat na široké škále zařízení nebo jen rozlišení. HTML prototyp to řeší.

2. Animace

Nástroje pro tvorbu maket jako Photoshop tady nepomohou. Klikací animační nástroje s výstupem do plnohodnotných CSS animací zatím nemáme. I tohle si můžete v rychlosti otestovat s pomocí svého kodéra.

3. Pokročilé interakce

Nemyslím tím jen „kliknu a přejdu najinou stránku“. V HTML se skvěle prototypují třeba složitější ajaxové interakce, klidně spojené s animací.

4. Pokročilé technologie

SVG výplně, výřezy, filtry, efekty... Tady opravdu nevím, jak jinak než přímo v HTML vyzkoušet řešení na nich postavená.

Ohromně ale z HTML prototypování mohou těžit vývojáři, kteří mají na starosti zpracování výstupů designérů. Cílem totiž není jen dostat produkt co nejdříve do prohlížeče, ale také vtáhnout vývojáře do dřívějších fází procesu. Tak, aby zavčasu otestovali možné problémy, které na drátěném modelu ani maketě prostě vidět nejsou.

Frontendisti ošetří své rizikové faktory

Každá správná frontendistka i každý správný frontendista mají prototypování rádi. I pro ně jsem našel pět otázek, na které jim HTML prototyp dokáže dát odpověď. Seznamu, který následuje, občas říkám „Pět obvyklých podezřelých na podkladech od designérů“.

1. Rychlosť načítání

Komponenta vypadá hezky, ale – nezpomalí zásadně načítání stránky? Karusel na úvodní stránku do drátěného modelu šoupnete

raz dva, že? Jen až během prototypování nebo nedej bože finální implementaci zjistíte, že se kvůli němu úvodní stránka načítá fakt pomalu. To nechcete.

2. Výkon v prohlížeči

Nebude se stránka při posouvání „trhat“? Je myšleno i na výkon při práci s načtenou stránkou? To, že klient je nadšený z krásných paralaxových efektů, které chce na webu pouštět zároveň s videem na pozadí, neznamená, že jeho pocity budou sdílet návštěvníci webu. Prototypem zjistíte, jak je jeho myšlenka problematická co do výkonu v prohlížeči.

3. Přístupnost

Jaký dopad bude komponenta mít na přístupnost zrakově postiženými? I karusel může být přístupný. Ale je přístupný zrovna ten váš? Dobré si to otestovat.

4. Zobrazování v exotických prohlížečích

Jak se bude zobrazovat ve starších a exotičtějších prohlížečích? Půjde snadno vymyslet náhradní řešení pro ně? Úžasné SVG efekty, které grafik někde viděl, jsou fajn. Jak to ale poběží v Exploreru 9, jehož uživatelé jsou třeba pro váš projekt ještě stále důležití?

5. Udržovatelnost

Nezkomplikuje řešení navržené designérem (nebo v horším případě klientem) celkovou udržovatelnost projektu? Pokud vaše argumenty nepadají na úrodnou půdu, udělejte rychlý prototyp, na kterém komplikaci se špatnou udržitelností složité knihovny v rámci vašeho projektu ukážete.

Všimněte si, že kromě čtvrtého a pátého bodu, které jsou ryze technické, spadají všechny ostatní do kompetencí dobrého webového designéra.

Klikací prototypovací nástroje jako Axure nám pachatele mezi obvyklými podezřelými najít nepomohou. Kód nemáme od začátku pod kontrolou, takže si na nich rizikové faktory neotestujeme. Výstupy z naklikaných prototypů navíc v produkční fázi projektu použít nemůžeme.

Nevýhody HTML prototypů: časově náročné a vyžadují zkušenější tým

Celou dobu tady o plnohodnotných prototypech básním, takže si teď také pojďme říci, v čem tkví jejich nevýhody:

Jsou časově náročné, a tedy dražší

Nehodí se tedy pro použití kdekoliv a kdykoliv. Rozumný kompromis budu hledat v dalším textu.

Vyžadují určitou zkušenosť na straně designéra i frontendisty

Hlavně u nezkušených frontendistů se může prototypování zbytečně prodražit. Pokud v týmu máte hlavně juniory, dávejte dvakrát pozor, zda se nezaměřují spíše na nástroje nebo věci, které jste na prototypech testovat nechtěli.

Vyžadují intenzivní spolupráci designérů a frontendistů

Prostě si spolu musíte sednout a pracovat na prototypech dohromady. V dělených týmech to často nejde.

Složitější zařazení do workflow větších týmů

Může vám chybět přímé napojení na Axure, Photoshop nebo jiné designérské nástroje. Můžete postrádat možnost diskutovat o konkrétních prvcích rozhraní přímo na webu.

Těžší rozhodování u znovupoužitelnosti

Někdy se dá znovupoužitelností kódu z prototypu dosáhnout, ale

pokud s živými prototypy začínáte, raději na tom netrvejte.

Moje vlastní zkušenost: U responzivního redesignu VašeČočky.cz jsem tak moc stál o znovupoužitelnost a kvalitu kódu prototypu, že jsem v některých jeho částech dosahoval závratně hlemýždího tempa přípravy. Někdy je prostě lepší v zájmu zvýšení rychlosti a lepšího zaměření pozornosti prototypový kód zahodit a pro finální řešení jej napsat znova.

Kdy HTML prototypování použít a kdy spíše ne? Nemusíte prototypovat celé weby

Ano, HTML prototypování je časově nejnáročnější varianta modelování webu. Na druhou stranu umožňuje otestovat velkou část rizikových faktorů na straně designu, frontendu i vývoje obecně. Kód je možné připravovat už v kvalitě využitelné pro produkční nasazení.

Na svých projektech HTML prototypování používám, kdykoliv je to možné. Mluvím do designu rozhraní i frontend kódování, a jak už jsem ukázal, pro obě části mé profesní osobnosti jsou prototypy výborné. Pro HTML prototypování se ale také často rozhodují týmy pracující dlouhodobě na jednom produktu. Tam se vyplatí.

V případě agenturní práce pro mnoho klientů doporučuji HTML prototypy dělat tam, kde zkoušíte nové věci nebo je rizikovost návrhu vysoká.

Ted' jedna odbočka. Podíváme se do blízké budoucnosti návrhu uživatelského rozhraní. Na atomické systémy designu, kde weby netvoříme po stránkách, ale po jednotlivých komponentách.

Atomický design má prototypování v krvi

Ještě poznámka k progresivním směrům návrhu uživatelského rozhraní. Systémy atomických designů jsou tvořeny skládáním menších komponent do větších. Návrh a testování se z velké části odehrávají přímo v prohlížeči, v nástroji Pattern Lab, takže z HTML prototypování dělají neoddělitelnou součást pracovního procesu.

O systémech atomického designu jsem psal na Vzhůru dolů.
vrdl.cz/p/pattern-lab

U systémů atomického designu to bez intenzivní spolupráce designéra s vývojářem nejde. A podobné to je u celého HTML prototypování.

V jakých nástrojích dělám HTML prototypy: Bootstrap a CodePen

Prototypování je dobré dělat v nástroji, který dokážete ovládat rychle a který vám neklade překážky. Svoje nástroje vám tedy vnucovat nechci.

Rychlý online editor CodePen

Editor, kde dělám jednoduché a přímočaré prototypy na pár řádků kódu. Je to rovnou online, takže se to dobře sdílí nebo posílá do mobilních zařízení k otestování.

Frontend knihovna Bootstrap

Většina mých projektů z posledních let vznikla nejprve jako živý prototyp postavený na Bootstrapu. Ten obsahuje dostatečně robustní sadu komponent proto, abych velmi rychle dokázal poskládat prvotní verzi webu k proklikání. S postupným iterativním

vývojem webu se pak postupně jeho komponenty nahrazují komponentami navrženými na míru projektu.

Bootstrap podporuje stavebnicový vývoj. Zároveň dodává řadu principů (prostřednictvím proměnných a mixinů), na kterých pak snadno můžeme stavět své vlastní komponenty.

Video: O HTML prototypování hezky povídá i Adam Kudrna na jedné z akcí Frontendisti.cz. vrdl.in/2t451

Nyní si pojďme znalosti o skicování a prototypech promítat do práce na fotogalerii v našem e-shopu.

Příklad: návrh fotogalerie krok za krokem

Nedivil bych se, kdybyste v této části knihy měli hlavu jako meloun. Jen díky tomu ale máme dost vědomostí, abychom navrhli první komponentu uživatelského rozhraní. A tak nevěšme meloun a pojďme si shrnout, jak budeme postupovat.

1. Do skic si nakreslíme všechny možnosti zobrazení komponenty, které nás napadnou.
2. Varianty vyhodnotíme a vybereme vítěznou.
3. Uděláme *prototyp* vybraného řešení.
4. Povíme si něco o předpokladech pro čistou implementaci.

Jak vypadá fotogalerie před naším designérským zásahem? Fotky jsou prostě seřazeny pod sebe. To nebude to pravé ořechové, že?

Skicování

Ted' prostě, jak jsem popisoval v kapitole o skicování, nabrousíme tužku, přimhouříme oči (to abychom vypadali jako skuteční profíci),

a hlavně – vypneme veškeré tendenze hodnotit to, co vytváříme. Nakreslíme prostě všechny varianty zobrazení fotogalerie, které nás napadnou.

Než si skici ukážeme, měli bychom si rozmyslet, kde se vlastně fotogalerie nachází v kontextu struktury obsahu stránky detailu produktu.

Kde se fotogalerie nachází v hierarchii stránky?

V případě prodeje bot přes internet platí, že obrázek nahradí tisíc slov. Návštěvníci a návštěvnice ForestKid.cz by měli mít možnost si na produkt vytvořit názor hned a bez dalšího klikání. Takže fotky chceme zobrazovat dostatečně velké.

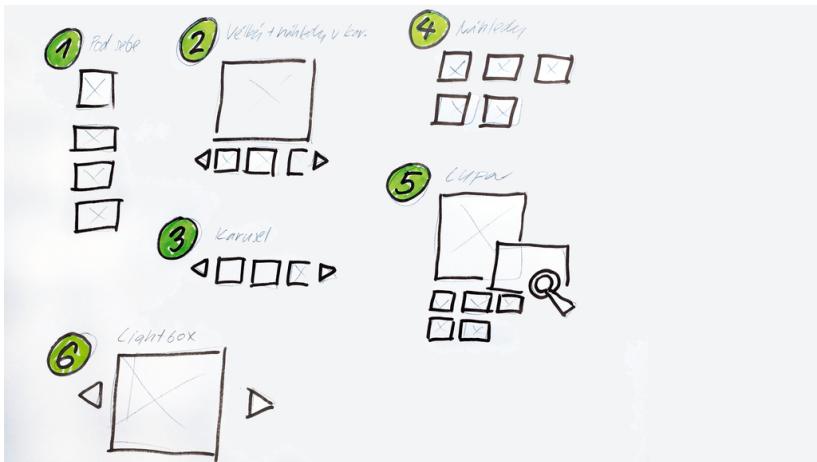
Zároveň jim ale nechceme dát tak zásadní váhu, aby upozadily všechny ostatní informace z horní poloviny stránky. Hlavně pak název produktu a cenu. Řekněme proto, že významem jsou fotografie někde na úrovni názvu produktu a ceny.

Musíme si také vyjasnit, jestli jsme od našich dodavatelů schopní získat dostatečně kvalitní obrázky produktů pro komponentu tak zásadní váhy. V naší hypotetické situaci ano, ale pozor, neplatí to zdaleka pro každý e-shop nebo web obecně.

A ještě na jednu věc nesmíme zapomenout, a sice na variabilitu obsahu. V našem případě vycházím z toho, že všechny fotografie budeme mít produkčně připravené ořezané na bílém pozadí v poměru 1 : 1 a že jich u jednoho produktu bude mezi třemi a patnácti.

Posouzení jednotlivých skic

Zpět ke skicám. Já ze sebe dostal šest možností, co vy?



Šest řešení ve skicích komponenty fotogalerie

1. Výpis pod sebe

Velmi jednoduchý z pohledu ovládání uživateli. V kontextu dalších prvků stránky je ovšem obvykle nepoužitelný. Na malých i velkých displejích odsunuje další informace, jako je cena nebo dostupnost velikostí. Vzpomeňte si také, jak jsem v kapitole o principech návrhu rozhraní bojoval proti otravně dlouhým stránkám na mobilech. Ne, tahle varianta pro nás skvělý e-shop nebude.

2. Středně velký náhled s navigací pomocí karuselu

Vypadá dobře. Šetří totiž prostor na výšku, ale zároveň zobrazuje jednu fotku dostatečně velkou. Na velkých displejích je možné tuto i další varianty doplnit o rozkliknutí do ještě větší verze pomocí lightboxu.

3. Středně velký náhled v karuselu

Ve svislém směru je ještě úspornější. Pokud bychom karusel navrhli správně, může jít o moc prima řešení pro malé displeje. Uživatelé velkých monitorů takovou míru úspornosti neocení,

tam bychom dali přednost řešení jinému. Jednou z našich zásad ale je konzistence rozhraní napříč všemi zařízeními, vzpomeňte si.

4. *Malé náhledy*

Nesplňují náš požadavek zobrazení co největší fotografie bez potřeby klikání.

5. *Lupa*

Pro prohlížení detailu výřezů považuji lupy za uživatelsky komplikované řešení, které je navíc možné pohodlně ovládat jen na *myšovitých* zařízeních.

6. *Lightbox*

Zobrazení fotky přes celou obrazovku samo o sobě použít nemůžeme. Necháme si jej jako možný stav některé z našich variant.

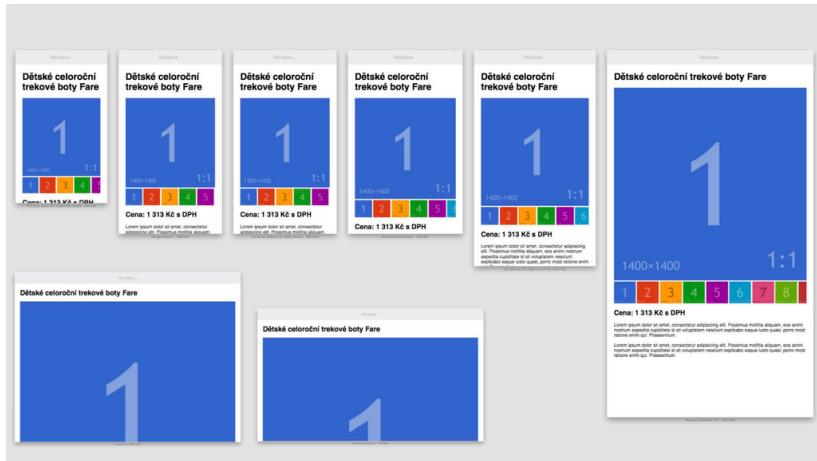
Pro ForestKid.cz jsem vybral druhou variantu se středně velkým náhledem a navigací pomocí karuselu.

Uděláme technický test hypotézy vzešlé ze skicování. Navážeme teď zase na předchozí text o prototypování v HTML.

Prototyp vybrané varianty

Rychlé demo si vytvoříme na už zmíněném online editoru *CodePen* a použijeme přitom generátor zástupných obrázků *Satyr.io*. Výsledek si prohlédneme v *Re:view*, neméně skvělém nástroji pro komparativní testování návrhů v mobilních rozlišeních.

Aktuální stav prototypu vidíte na obrázku nebo naživo na *CodePenu*. cdpn.io/e/JEKxEK.



Komparativní pohled na vybraná rozlišení mobilních zařízení s Androidem přes Re:view

Je vidět, že na zařízeních s vyšším poměrem šířky k výšce uvidí uživatelé při načítání stránky spolehlivě jen navigaci, název produktu a obrázky. Jde samozřejmě o prototyp, takže přesná výška těchto elementů nám v této fázi známá není.

Problém ale zčásti můžeme upravit už teď. Prostě zajistíme, aby se hlavní obrázek nikdy neroztáhl na celou šířku stránky, a zmenšíme tím také výšku komponenty:

```
.gallery-pane {  
    width: 80%;  
    margin-right: auto;  
    margin-left: auto;  
}
```

I tento další krok si můžete vyzkoušet naživo. cdpn.io/e/XpKOxd

Vsadím se ale, že při pohledu na obrázek z testování prototypu vás do očí praštilo ještě něco jiného. Právě to taky od prototypů chceme

– aby obtěžovaly naše oči tím, co naše mozky nedomyslely. Jasně, myslím tím náhledy ze zařízení držených na šířku.

Řešení pro zobrazení na šířku

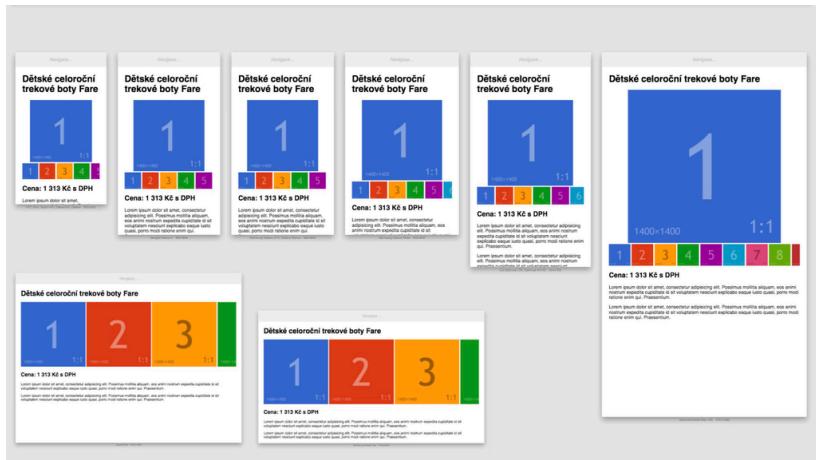
V režimu landscape je vybrané řešení bohužel velmi neuspokojivé. Naše milá uživatelka uvidí v lepším případě jen horní část hlavního obrázku. Když naroluje na malé navigační obrázky, uvidí zase jen jeho spodní část. Tady bohužel musíme sáhnout po jiném řešení. Které ze skicovaných využívalo lépe šířku?

No jasně, třetí varianta, karusel. Ten totiž nevyžaduje roztažení obrázku do plné šířky okna, což zmenší výšku komponenty. Obrázků se do karuselu vejde více vedle sebe, což opět šetří prostor na výšku. Karusel je navíc návrhový vzor, který bychom tak či tak využili pro menší náhledy obrázků v původním řešení. Pravidlo konzistence zde tedy tak moc neporušujeme.

Když kód pro režim na šířku hodně zjednodušíme, bude vypadat následovně:

```
@media only screen and (orientation: landscape) {  
    .gallery-pane img {  
        width: 30%;  
    }  
  
    .gallery-thumbs {  
        display: none;  
    }  
}
```

Využíváme `orientation: landscape`, jednu z [Media Queries](#), o kterých budeme mluvit v další kapitole.



Výsledný prototyp designu fotogalerie

CodePen výsledného prototypu: cdpn.io/e/dNXrMe

Vy zkušenější jste jistě mírně pozvedli obočí nad schováváním jedné části s obrázky pomocí `display: none`. Tohle v produkčním kódu být nesmí, protože by nám mobilní prohlížeče neviditelné obrázky tak či tak stáhly a zpomalily tím nehezky načtení stránky.

Je dobré ale na tomto místě zopakovat, že děláme prototypování *designu*. Určité technické zjednodušení je tady namísto, a to v zájmu zrychlení jednotlivých kroků vývoje.

Snad byly na příkladu dobře vidět výhody prototypů přímo v kódu, o kterých jsem už psal.

- Bylo to *brzy v prohlížeči*, takže jsme na nevýhody původně vybraného řešení nepřišli až při drahé technické implementaci.
- Je to *plnohodnotná* webová stránka, takže jsme mohli testovat jakýkoliv kontext. V tomto případě stačily různě velké obrazovky mobilních zařízení. Stejně tak bychom CodePen

mohli testovat s uživateli.

- Kód je zčásti *znova použitelný* a při následné implementaci jej použijeme jako základ.

Jde ovšem pořád jen o mou designérsko-kodérskou hypotézu, kterou bych měl ověřit uživatelským testováním. To je ale téma, které se nám do knížky už nevejde. Zájemce odkážu na už zmíněný kurz „Human-Centered Design: Design zaměřený na člověka“, který je dostupný zdarma. seduo.cz/human-centered-design

A ještě odkazy na dva zde zmíněné nástroje:

- *Re:view* – skvělé rozšíření Google Chrome pro komparativní testování návrhů v mobilních rozlišeních. re-view.emmet.io/
- *Satyr.io* – vynikající generátor zástupných obrázků pro vaše prototypy. satyr.io

Zapamatujte si

- Skicování je výborné pro usnadnění řešení designérských, ale i kodérských problémů pomocí tužky a papíru nebo tabletu.
- Pokud vám to zkušenosť a rozpočet dovolí, testujte složitější prvky rozhraní na HTML prototypech.
- S prototypováním vám pomůže knihovna Bootstrap a online editor CodePen.

Kapitola 9: Media Queries a layout

Responzivní webdesign by bez layoutu a Media Queries byl jako Bob bez Bobka v klobouku kouzelníka Pokustona. Nesmíme je prostě vynechat.

1. Media Queries, podmínky pro zobrazení, nutně potřebujeme.
Vezmeme je do hloubky.
2. Vy zkušenější oceníte i tipy pro práci s Media Queries.
3. Detailně projdeme způsob tvorby breakpointů očima designéra.
4. Na breakpointy se podíváme také technicky. Jak je implementovat v CSS?
5. Naučíme se základy responzivního layoutu.
6. Rozvržení stránky si pak přidáme i do příkladu.
7. Podíváme se na dokončenou verzi příkladu.

Media Queries

Jde o podmínky, které umožňují aplikovat různá CSS pravidla v různých technických kontextech.

Dejme si rychlý příklad:

```
h1 { font-size: 2em }

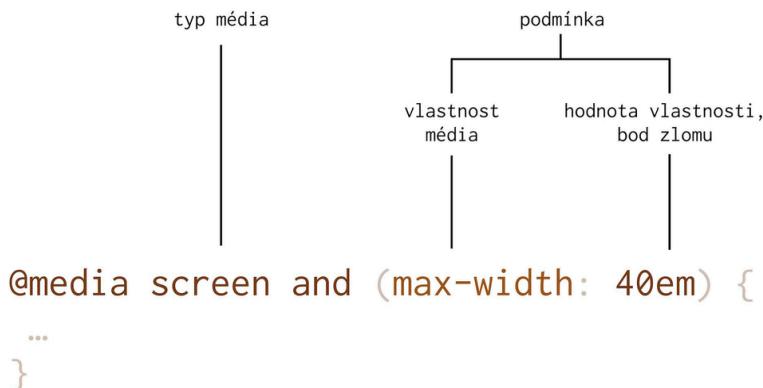
@media only screen and (max-width: 40em) {
    h1 { font-size: 1.5em }
}
```

Nadpis první úrovně zmenšíme pro okna prohlížeče do šířky 40em. Toto jednoduché použití podmínek si můžete vyzkoušet v živé ukázce. cdpn.io/e/Bpjzbz

Z CSS2 budete znát podmínky pro typy médií jako třeba @media print. Norma CSS3 Media Queries je vylepšuje o bližší specifikaci vlastnosti médií.

Anatomie Media Query

Dotaz na medium (anglicky *media query*) se skládá z typu média (*media type*, výchozí je `all`) a podmínky obsahující vlastnosti média (*media features*) s hodnotou nebo rozmezím hodnot.



Anatomie Media Query. Pro zjednodušení jsem odstranil klíčové slovo „only“, které ze zpracování podmínky vyloučí starší Internet Explorery

Body zlomu

V responzivním webdesignu nemůžeme minout pojmem bod zlomu (*breakpoint*), což je hodnota vlastnosti média. O „breakpointech“ mluvíme jako o sadě hodnot pro konkrétní web nebo systém designu. Knihovna Bootstrap má například body zlomu přednastavené takto:

- extra small (šířka okna do 767px)
- small (768–991)
- medium (992–1199)
- large (1200 a více)

Pro konkrétní projekty samozřejmě nebudou tyto konkrétní hodnoty použitelné. Body zlomu je vždy lepší definovat podle obsahu konkrétních komponent než takto centrálně. Body zlomu Bootstrapu berte jen jako ilustraci pojmu.

Tvorbou breakpointů z pohledu designéra se zabývám ve zvláštním textu.

Minimální nebo maximální výška a šířka

Nejčastější podmínky v responzivním designu vypadají jako podmínky pro vodorovný nebo svislý rozměr okna prohlížeče:

```
@media only screen and (min-width: 40em) { ... }  
@media only screen and (max-width: 40em) { ... }  
@media only screen and (min-height: 40em) { ... }  
@media only screen and (max-height: 40em) { ... }
```

V drtivé většině případů se ale pracuje jen s vodorovnými rozměry, s vlastností `width`. Ukážeme si i další možnosti. Předtím se ale naučíme, jak podmínky kombinovat.

Logické operátory

Spojovat Media Queries můžeme pomocí operátoru **and**:

```
@media only screen  
  and (min-width: 30em) and (max-width: 40em) { ... }
```

Podmínka se aplikuje jen na všechna zobrazovací média, tedy ne v tiskové verzi. Navíc musejí mít šířku okna mezi třiceti a čtyřiceti em.

A co „nebo“? Programátoři by možná očekávali **or**, ale používá se čárka:

```
@media only screen and (max-width: 40em), print { ... }
```

Uvedená podmínka se vyhodnotí jako pravdivá, když si uživatel stránku prohlédne na zobrazovacím médiu o šířce viewportu do 40em nebo když bude stránku tisknout. Podle CSS specifikace jde o „seznam oddělený čárkou“, kde se jednotlivé položky seznamu vyhodnocují samostatně. Disjunkce („or“) má proto větší váhu než konjunkce („and“).

Dalším možným operátorem je negace:

```
@media not print { ... }
```

Jen pozor – negace vždy postihuje celou podmínku, nikoliv její určitou část.

Další vlastnosti média

Je jich dost. Od zjišťování orientace zařízení či poměru stran obrazovky přes počet barev až po zjišťování vysokokapacitních displejů. Uvedu jen ty nejpoužitelnější.

Detekce orientace zařízení

Drží uživatel zařízení na výšku, nebo na šířku?

```
@media only screen and  
  (orientation: portrait) {  
    /* Držení na výšku */  
}
```

```
@media only screen and  
  (orientation: landscape) {  
    /* Držení na šířku */  
}
```

Podmínka pro poměr stran obrazovky

Okna s poměrem stran 16 : 9 například zacílíme takto:

```
@media only screen and  
  (aspect-ratio: 16/9) { ... }
```

Existují samozřejmě i varianty pro rozmezí hodnot: `min-aspect-ratio` a `max-aspect-ratio`.

Detekce „Retina“ displejů

Moderní displeje s vyšší hustotou hardwarových pixelů můžeme detektovat takto:

```
@media only screen and  
  (min-resolution: 2dppx) { ... }
```

Aplikuje se, pokud má zařízení poměr mezi hardwarovými a CSS pixely alespoň 2. Ve starších prohlížečích se namísto `resolution` pro totéž používala vlastnost `device-pixel-ratio`.

Poměrů je ale dnes celá řada (1,25; 1,5; 2; 3; 4...). Proto doporučuji namísto podmínky pro vlastnost `resolution` v kombinaci s bitmapovými obrázky využívat vektorový formát SVG. U něj vlastnost `resolution` nepotřebujeme, vektorový obrázek se vykreslí

na všech poměrech **resolution** stejně dobře. vrdl.cz/p/svg

Pokud byste náhodou nerozuměli souvislostem, vzpomeňte si na text o **CSS pixelu** z první kapitoly.

A co další vlastnosti médií?

V textu jsme zvládli ty nejpoužívanější. Z dalších zajímavých budu jmenovat hlavně sadu vlastností pro detekci způsobu ovládání.

Například **@media (hover:hover)**, které se umí prohlížeče zeptat na podporu efektu po najetí kurzoru. Tam se ale čeká na podporu Firefoxu. caniuse.com/media-interaction

Vlastnosti médií existuje ale mnohem více, i když ty ostatní už tak moc použitelné nejsou. jecas.cz/media

Na co si dát u Media Queries pozor?

Nezapomeňte na typ média a raději ani vylučovací slovo **only**.

Nepoužívejte **device-width** a nerozdělujte CSS do souborů podle bodů zlomu.

1. Zápis vynechávající typ média

Tohle může být špatně:

```
@media (min-width: 40em) { ... }
```

Podmínka se pak aplikuje na všechna média. Tedy nejen obrazovková, ale také výstupy pro tisk. Výchozí typ média je totiž **all**, nikoliv **screen**. Tyto dva zápisys jsou tedy ekvivalentní:

```
@media all { ... }  
@media { ... }
```

Klíčové slovo **only** před médiem vyřazuje ze hry staré Explorery (verze šest a starší), které by se jinak tvářily, že podmínce rozumějí.

Občas je krátký zápis bezproblémový, musíte ovšem vědět, co děláte.

Správný zápis vypadá následovně:

```
@media only screen and (min-width: 40em) { ... }
```

2. Cílení na rozlišení obrazovky

Zápis s „device“ cílí na rozlišení obrazovky, nikoliv na velikost okna prohlížeče:

```
@media ... (min-device-width: 40em) { ... }
```

V začátcích responzivního webdesignu toho někteří využívali pro „detekci“ konkrétních zařízení. To ale dobrý postup nebyl, není a nebude. Rozlišení obrazovek je dnes velmi hodně a nedá se z nich vyčist, jestli zařízení patří mezi mobily, tablety, nebo něco jiného.

Zápis podmínky s prefixem `device-` navíc (kvůli cílení na obrazovku, nikoliv okno) znemožňuje testování responzivního layoutu pomocí změny velikosti okna. Vlastně nevím, k čemu by vám dnes mohl být dobrý.

3. Rozdělování CSS podle velikosti obrazovky

Občas ještě někde vídávám:

```
<link rel="stylesheet" href="mobile.css"
      media="max-width: 40em">
```

Vývojáři si dělí kód podle typu zařízení: Například do `mobile.css`, `tablet.css` a `desktop.css`. Kód jedné komponenty rozhraní je pak ale rozdělený do více souborů a dost špatně se to spravuje.

Jediný rozumný hlavní způsob organizace souborů se styly je podle komponent uživatelského rozhraní, jako jsou navigace, tlačítka atd. Píšete totiž kód pro jednu komponentu, nikoliv pro jeden bod zlomu. Pokud je to pro vás nové, čtěte můj blogpost o organizaci CSS.

vrdl.cz/b/29-organizace-css-2014

Media Queries v Javascriptu

V Javascriptu používejte funkci `matchMedia()`, která přijímá stejné podmínky jako CSS:

```
if (matchMedia('only screen and (max-width: 40em)').matches) {  
    // ...  
}
```

Častou chybou je detekce podle šířky nebo výšky okna a následné spoléhání na událost `onresize`. Je to v kódu zbytečně složité a při zmenšování nebo zvětšování okna výkonnostně problematické.

Tipy a triky k Media Queries

V produkčním kódu byste měli u Media Queries mít jednotku `em`. Podíváme se na způsob psaní Mobile First a povíme si něco o budoucnosti – o Element Queries.

Správné je použít `em`, ale autorům se obvykle lépe pracuje s `px`

Často se vedou spory kolem použití jednotek v Media Queries. Správná jednotka je `em`, alespoň ve všech mně známých případech.

Jednotka `px` se nepřizpůsobuje změně velikosti písma, proto nefunguje dobře v situacích, jako je uživatelské zvětšení či zmenšení písma.

A `rem`? S těmi v Media Queries nepracuje korektně Safari, když změníte velikost písma autorský. Takže by v produkčním kódu

našich webů mělo být `em`. Detailní vysvětlení najdete v článku „[PX, EM or REM Media Queries?](#)“ zellwk.com/blog/media-query-units/

Při návrhu bodu zlomu se ovšem na web díváme přes okno prohlížeče. Jeho šířka k naší smůle ale nepracuje s `em`, nýbrž s `px`. Proto obvykle nejdřív v hlavě musíme spustit kalkulačku přepočít z pixelů do `em`. Alespoň v mé případě je ale spouštění kalkulačky náročné na výpočetní zdroje.

Proto se i vám může hodit ve vývojářském kódu používat pixely a pro produkční kód si je nechat přepočít do `em`. Pokud využíváte nějaký automatizační nástroj, je zde plugin „[postcss-em-media-query](#)“.github.com/niksy/postcss-em-media-query

Zanořování Media Queries

Doporučuji vymýlet body zlomu nejlépe vždy pro obsah a design konkrétní komponenty. V kódu bych pak podmínu chtěl vyjádřit jako podmnožinu selektoru komponenty, abych tím i vizuálně v kódu vyjádřil jejich vztah:

```
.el {  
    /* Styly pro vše */  
    @media only screen and (min-width: 20em) {  
        /* Styly pro zařízení splňující podmínu */  
    }  
}
```

Bylo by to krásně přehledné, že? Tento typ zápisu ovšem v prohlížečích nepřipadá v úvahu. Na pomoc byste si museli vzít některý z CSS preprocesorů nebo postprocesor PostCSS.

Možné ovšem je zanořování Media Queries do sebe se selektorem uvnitř:

```
@media only screen and (min-width: 100px) {  
  @media only screen and (max-width: 300px) {  
    .el {  
      color: red;  
    }  
  }  
}
```

Jen pozor, nebude to fungovat v žádném Internet Exploreru, takže spolehlivější je opět využít automat pro zpracování CSS, jako je preprocesor. Živá ukázka různých typů zanořování je na CodePenu.
cdpn.io/e/xEkKd

Psaní kódu stylem Mobile First

Kromě designérské filozofie můžeme o Mobile First mluvit také v souvislosti s psaním CSS kódu:

```
/* „Mobile First“ přístup (lepší): */  
.el {  
  /* Styly pro menší obrazovky */  
}  
  
.el {  
  @media only screen and (min-width: 25em) {  
    display: flex;  
    /* Styly pro větší obrazovky */  
  }  
}
```

Pokud design vaší komponenty nese nálepku „Mobile First“, budete pravděpodobně i její kód psát od varianty pro nejmenší displeje.

Je to ale i obecně výhodnější než opačný přístup, „Desktop First“. Kód pro menší displeje je obvykle jednodušší. A je lepší další deklarace přidávat než odstraňovat nadefinované, což bychom museli dělat u opačného přístupu:

```
/* „Desktop First“ přístup (horší): */
.el {
    /* Styly pro větší obrazovky */
    display: flex;
}

.el {
    @media only screen and (max-width: 25em) {
        display: block;
        /* Styly pro menší obrazovky */
    }
}
```

Element Queries: podmínky týkající se vlastností rodičovského prvku

Media Queries se dotazují vždy jen na parametry okna prohlížeče. To je fajn, když vymýslíme body zlomu pro layout obrazovky. Horší je to pro jednotlivé, v něm rozmištěné komponenty. Zvláště pro ty, které se mohou vyskytovat na různých místech uživatelského rozhraní.

Představme si, že bychom se mohli v CSS ptát na velikost rodičovského elementu. Ano, většinou nás zajímá právě ten. Mohlo by to pak vypadat třeba takto:

```
/* Od šířky 20em naskládej položky vedle sebe: */
@element ".item" and (min-width: 20em) {
    $this {
        display: flex;
    }
}
```

Kód by se aplikoval, pokud by šířka rodiče elementu `.item` byla alespoň `20em`. Ukázka je v kódu javascriptové implementace konceptu – EQCSS.

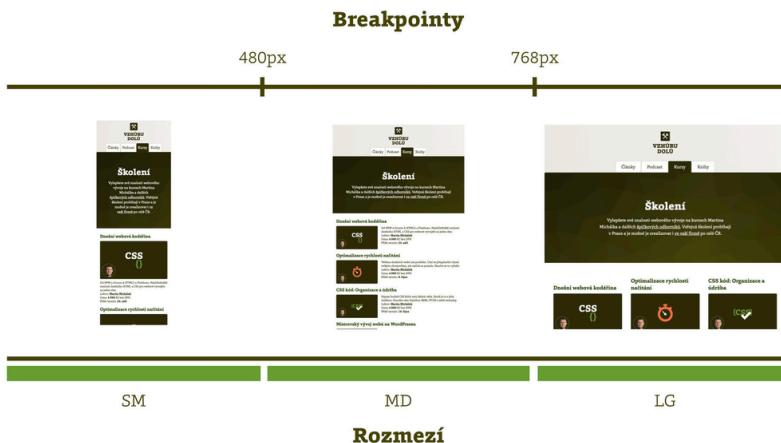
Problém je právě v té závislosti na Javascriptu. Při havárii JS by vám přestalo fungovat CSS. A nebude to také výkonnostně optimální. Specifikace pro Element Queries se teprve rodí, takže pokud nemáte výjimečnou motivaci, doporučuji zatím jen sledovat diskuzi na toto téma. Bližší informace mám na blogu. vrdl.cz/p/element-queries

Breakpointy a rozmezí platnosti responzivního designu

Breakpointy jsou velikosti okna ([viewportu](#)) prohlížeče, ve kterých se může měnit design webu. Česky jim můžeme říkat *body zlomu designu*.

Body zlomu a rozmezí: Bavme se o dvou různých věcech

Všímám si, že se weboví lidé občas neshodnou v definici „breakpointu“.



Bod zlomu (v angličtině „breakpoint“): Konkrétní body, ve kterých se design mění. Samotné hodnoty typu 480px, 640px, 768px a tak dále.

Občas se ale slovem „breakpoint“ označuje ještě jiná věc – rozmezí platnosti designu („range“). Tím jsou pak myšleny skupiny velikosti viewportů a hodnoty typu 0–480px, 481 a více, 1024px a méně a tak dále.

Může být proto matoucí, když použijete frázi „breakpoint pro nejmenší displeje“ a myslíte tím hodnoty 0–480px. Mluvíte totiž o rozmezí. Ale to jen tak na okraj. Pojďme na opravdu důležité téma – jak vlastně body zlomu vymyslet pro konkrétní projekty?

Zapomeňte na zařízení: Z obsahu vyplývá design. Z designu vyplývají breakpointy

Častou chybou je vymýšlet breakpointy „podle zařízení“. Dejme tomu, že chceme oslovit všechny tablety. Usmyslíme si, že to zařídíme následující podmínkou:

```
/* Bod zlomu „pro tablety“ (špatně) */  
@media only screen and (min-width: 640px) and (max-width:  
768px) { }
```

Vypadá to hezky, ale je to konina. Různých rozlišení mobilů i tabletů je tolik, že se nelze na nějaké rozmezí pro tablety nebo mobily spoléhat. V naší ukázce tak některé tablety podmínu splní, jiné zase ne.

Takový Samsung Nexus 10 má rozlišení na delší straně v hodnotě 1280 pixelů, takže podmínu nesplní. Splní ji naopak mnoho chytrých telefonů jako třeba iPhone 6 v režimu na šířku se 736 pixely. Media Queries proto k detekci zařízení vůbec nepoužívejte.

Vždy se při vymýšlení bodů zlomu snažte zaměřit na obsah a jeho

rozvržení na obrazovce. Body zlomu mají vyplynout z obsahu a jeho designu.

Musím tady citovat klasika Stephena Haye:

Začněte s malou obrazovkou a pak zvětšujte okno, dokud se design nerozbije. Tady je čas na breakpoint!

OK, vykašleme se tedy na zařízení a jejich obrazovky. Pořád jsem vám ale ještě neprozradil, jak ty breakpointy vymýšlet. Omlouvám se, že vás takhle napínám. Už na to jdeme. Jen ještě jedno varování.

Ideálně nepřebírejte hotové breakpointy. Odvodte je z designu a cílové skupiny vašeho projektu

Obecně vám samozřejmě doporučím vymýšlet vlastní breakpointy, hlavně ze dvou důvodů:

1. Zohlednění cílové skupiny a zastoupení jednotlivých rozlišení v ní. Může se vám stát, že prefabrikované breakpointy u vás nebudou fungovat, protože v cílové skupině máte například velmi podprůměrný počet uživatelů mobilů a nadprůměrné množství uživatelů velkých displejů. Univerzální breakpointy vycházejí z průměrných dat, což vám může být k ničemu.
2. Přihlédnutí k vlastnímu návrhu uživatelského rozhraní. Platí, že obsah ovlivňuje design a naopak. Pokud je váš design něčím výjimečný (což je tak každý druhý), univerzální breakpointy nemusejí svou roli plnit dobře.

Pojďme se ale na ty prefabrikované body zlomu podívat. Někdy se totiž hodit mohou.

Univerzální breakpointy

Nelámat si hlavu body zlomu na míru je často jediná cesta. Z voleje dám dva příklady:

- Pracujete na univerzálním systému pro tvorbu webů, jako jsou třeba Webnode, SolidPixels nebo framework typu Bootstrapu. Prostě nevíte, jak bude vypadat obsah a design jednotlivých webů.
- Nemáte dost dat. Například proto, že jste v rané fázi návrhu projektu a teprve prototypujete.

Nejčastěji se prefabrikované body zlomu designu přebírají z populárních frontendových frameworků, jako je právě Bootstrap. Ten má přednastavené čtyři hodnoty – *xs* (576 pixelů), *sm* (768), *md* (992) a *lg* (1200).

Lepším řešením může být nastavení podle textu Davida Gilbertsona „The 100% correct way to do CSS breakpoints“.

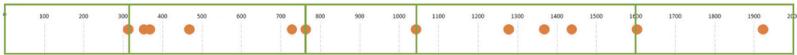
vrdl.in/correctbreakpoints

Tomu se na breakpointech Bootstrapu nelibily dvě věci.

Za prvé – nedělí uživatele do rovnoměrných skupin a obecně spíše vychází ze šířky obrazovek existujících zařízení než z dat od uživatelů.

A ta druhá věc? Není dobré udělat z konkrétního rozlišení konkrétního zařízení breakpoint. Zde se bavíme hlavně o hodnotě 768 CSS pixelů pro bod zlomu „sm“. Určitě jste to sami zažili: Designérka řekne „pro breakpoint 768...“ a vývojář rovnou píše kód zaměřující se přesně na 768 pixelů. Jenže designérka myslela 768 pixelů a výše. A to mohla myslet také rozmezí mezi 768 a 992 pixely, že ano?

Zpět k Davidu Gilbertsonovi. Ten z globálních statistik vytáhl taková čísla, aby bylo rozložení viewportů v jednotlivých rozmezích rovnoměrnější.



Obrázek: Návrh rozložení breakpointů tak, aby v rozmezích byly rovnoměrně zastoupeny různé skupiny zařízení. Oranžové tečky představují nejčastější rozlišení.
Zdroj: David Gilbertson

V tomto kroku jsme došli k těmto hodnotám:

- 600px
- 900px
- 1200px
- 1800px

Jde o lepší řešení než to z Bootstrapu, takže pokud nic jiného nemáte, můžu vám jej doporučit.

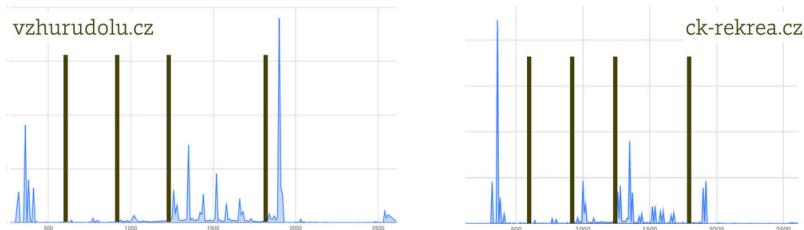
Zůstává zde ale jeden problém. Vězí v oněch *globálních statistikách*. Ty prostě nemusejí pasovat na vaši cílovou skupinu a váš projekt. Úplně nejlepší řešení tedy spočívá v odvození breakpointů z vlastního designu a vlastních dat.

Breakpointy na míru

To, jak je navržené vaše rozhraní, v tuhle chvíli nevím. Můžete ale s mou pomocí zjistit, jak vypadá vaše cílová skupina. Nebo přesněji – rozložení šířek obrazovky v ní.

V zásadě to dnes už jde vytáhnout z Google Analytics. Je to trochu práce, ale určitě se vám to u větších projektů vyplatí. Podíváme se na jeho výsledky. Podrobný návod, jak toho dosáhnout, jsem sepsal do textu „S jakými viewporty uživatelé navštěvují můj web?“. vrdl.cz/viewport-google-analytics

Získáte z něj grafy podobné těmto:



Obrázek: Rozložení breakpointů pro můj web Vzhůru dolů a na webech cestovky Rekrea za první polovinu roku 2018

Z obrázku je hezky vidět, že už tyto dva projekty se v zastoupení cílové skupiny liší. Dejme si výsledky do tabulky s breakpointy z článku Davida Gilbertsona.

Rozmezí v px	Vzhůru dolů	Rekrea
0–599 (xs)	20 %	33 %
600–899 (sm)	2 %	3 %
900–1199 (md)	5 %	13 %
1200–1799 (lg)	38 %	39 %
1800 a více (xl)	33 %	10 %

Tabulka: Zastoupení šířky viewportů v univerzálních rozmezích designu

Z tabulky můžeme například vyčíst, že na projektu *Vzhůru dolů* je hodně důležitá skupina uživatelů s velkými displeji – rozmezí *xl*. U obou projektů jsou pak velmi málo zajímavé skupiny s rozlišeními v rozmezí *sm*. Dává nám to informaci o prioritě jednotlivých rozmezí. Prostě víme, jak moc do jednotlivých skupin investovat energii.

Poznámka k těmto dvěma projektům – ve skutečnosti mají nastaveny body zlomu trochu jinak, podle designu. Zde je uvádím proto, že se výrazně liší v cílové skupině a při hypotetické situaci nasazení univerzálních breakpointů je hezký vidět, že uživatele nerozdělují rovnoměrně a že bychom v dalším kroku museli body zlomu upravovat.

Pro větší projekty vám velmi doporučím vytáhnout si tahle data o rozlišeních obrazovky vašich uživatelů. Pomůže vám to minimálně prioritizovat intenzitu práce pro jejich skupiny odvozené z rozmezí šířek obrazovky.

Tím se dostáváme k dalšímu zajímavého bodu – jak breakpointy pojmenovávat.

Pojmenování: Ideálně abstraktně podle triček

Předpokládám, že breakpointy máte uložené v proměnné preprocessoru. Vezmu tři příklady pojmenování bodu zlomu na 900px:

- *iPad portrait*: Žádná sláva, protože zmiňuje konkrétní zařízení.
Platí totiž určitě i pro jiné tablety než iPad.
- *Tablet portrait*: O něco lepší, ale pořád špatně. Bod zlomu se může týkat také mobilu v landscape režimu nebo zmenšeného okna desktopu.

- **Medium** (nebo v kódu `$breakpoint-md`): Dle mého nejlepší pojmenování. Je abstraktní, takže do komunikace neplete zavádějící konkrétnost. A taky je snadno naučitelný a obvyklý. Kromě výrobců triček používá stupnici `xs` (extra small), `sm` (small), `md` (medium), `lg` (large) a `xl` (extra large) také Bootstrap a další frontendové frameworky.

Tolik k pojmenování. Důležité je, že ať jsou breakpointy pojmenované jakkoliv, měl by se na jejich názvosloví domluvit celý tým.

Lokální a globální body zlomu

Obsahové body zlomu nejčastěji definují podle obsahu konkrétních komponent. Říkám jim *lokální breakpointy* (občas též *komponentové breakpointy* nebo kdysi *mikrobreakpointy*). Hodně názvů pro stejnou věc, že ano? Jde ale o totéž: odlišení bodů zlomu a rozmezí platnosti designu, které platí pro celou aplikaci, od těch, které platí jen pro její malou část – obvykle právě komponentu.

Lokální

Jako příklad vezměme záložkovou navigaci, ve které je určitý počet položek, proto layout zapínáme až od určité hodnoty:

```
/* tabs.scss: */  
$tabs-breakpoint: 260px;  
  
@media only screen and (min-width: #{$tabs-breakpoint}) {  
    .tabs { display: flex; }  
}
```

Globální

To ale neznamená, že nepotřebujete body zlomu *globální*. Ty se nejčastěji hodí pro nastavení layoutu stránky, ale přebírají je

i jednotlivé komponenty:

```
/* variables.scss: */  
$md-breakpoint: 600px;  
  
/* tabs.scss: */  
@media only screen and (min-width: #{$md-breakpoint}) {  
    .tabs { font-size: 1.3rem; }  
}
```

V dalším textu se podíváme na to, jaké jsou možnosti implementace breakpointů a rozmezí v kódu.

Responzivní breakpointy: Realizace v kódu (CSS, Sass i PostCSS)

Ukládání breakpointů a rozmezí do proměnných preprocesoru velmi doporučuji, protože to zpřehlední kód a zefektivní psaní.

Příklady níže využívají CSS preprocesoru Sass v SCSS syntaxi. Ale podíváme se také na PostCSS (a CSSnext) nebo očekávaný vývoj specifikací.

Jednoduše v proměnných

Následuje příklad ze zdrojáků Vzhůru dolů.

Definice breakpointů:

```
$vd-screen-sm: 600px;  
$vd-screen-md: 768px;  
$vd-screen-lg: 1100px;
```

Definice rozmezí:

```
$vd-screen-sm-up: "(min-width: #{$vd-screen-sm})";  
$vd-screen-md-up: "(min-width: #{$vd-screen-md})";  
$vd-screen-lg-up: "(min-width: #{$vd-screen-lg})";  
  
$vd-screen-sm-down: "(max-width: #{$vd-screen-sm - 1})";  
$vd-screen-md-down: "(max-width: #{$vd-screen-md - 1})";  
$vd-screen-lg-down: "(max-width: #{$vd-screen-lg - 1})";
```

A ještě použití:

```
@media #{$vd-screen-sm-up} { }
```

Jak vidíte, zápis použití je díky specifikům Sassu poněkud krkolomnější a celkově jednoduchá implementace vám u větších projektů nemusí stačit. Pojďme se podívat na další, propracovanější metody.

Pomocí mixinů

Další možnost je vytvořit si mixiny pro práci s rozmezími platnosti mixinů. Pojďme rovnou k použití, tam je to vidět přímočařeji.

```
/* Breakpoint "sm" a větší šířka viewportu: */  
@include media-breakpoint-up(sm) { }  
  
/* Breakpoint "lg" a menší šířka viewportu: */  
@include media-breakpoint-down(lg) { }  
  
/* Jen rozmezí následující za breakpointem "sm" */  
@include media-breakpoint-only(sm) { }  
  
/* Vše mezi breakpointy "sm" a "lg": */  
@include media-breakpoint-between(sm, lg) { }
```

Definování mixinů si případně nastudujte ve zdrojácích Bootstrapu.
[vrdl.in/bootstrapbreak](#)

Nevýhoda? Před použitím si musíme naprogramovat celou řadu

uvedených mixinů. Nebo použít Bootstrap, což vám jen kvůli správě breakpointů rozhodně nedoporučuji. Je tady ale pár knihoven, které umí spravovat právě jen breakpointy.

Pomocí knihovny: Sass MQ

Knihovna vám zařídí kompletní správu breakpointů a rozmezí. Nejprve si na definujeme seznam bodů zlomu, včetně jejich pojmenování. Následuje příklad z dokumentace:

```
$mq-breakpoints: (
  mobile: 320px,
  tablet: 740px,
  desktop: 980px,
  wide:    1300px
) !default;
```

Následují možnosti použití, odpovídající předchozímu příkladu. Prostě zavoláte mixin `mq()`:

```
/* Breakpoint "mobile" a větší šířky viewportu: */
@include mq($from: mobile) { }

/* Breakpoint "tablet" a menší šířky viewportu: */
@include mq($until: tablet) { }

/* Jen rozmezí následující za bodem zlomu "tablet" */
@include mq(tablet) { }

/* Vše mezi breakpointy "mobile" a "desktop": */
@include mq(mobile, desktop) { }

/* Vlastní breakpoint: */
@include mq('only screen and (min-width: 1440px)') { }
```

Je navíc možné pracovat s vlastními breakpointy, takže způsob volání zůstává konzistentní napříč celým CSS:

```
@include mq('only screen and (min-width: 1440px)') { }
```

Funkcí a možností využití je zde ale více. Chcete-li více informací, odkážu vás na web knihovny: github.com/sass-mq/sass-mq

Všechna zmíněná řešení mají jednu poměrně citelnou nevýhodu: Používají preprocessory, takže zavádějí nové jazykové prvky do kódu projektu, což může zhoršovat jeho čitelnost.

Ukažme si tedy ještě dvě možnosti, které jsou blíže k čistému CSS. Ale rovnou říkám, že zatím vám jejich využití nedoporučím.

Pomocí Media Queries Level 4, což bohužel zatím neumí prohlížeče

Tohle uvádí hlavně proto, abychom viděli, kam nás vede vývoj standardů. Konsorcium W3.org ve čtvrté verzi specifikace Media Queries chystá „Range Context“, což je zjednodušený zápis platnosti dotazu:

```
@media (width > 320px) { }
@media (width <= 320px) { }
@media (400px <= width <= 700px) { }
```

Pokud je mi známo, v době psaní článku tohle žádný prohlížeč neumí. Pro více informací utíkejte do specifikace na w3.org/TR/mediaqueries-4.

Pomocí PostCSS, což je bohužel jen polovičatý preprocessory

Mnoho vývojářů se nechalo zlákat postprocesorem PostCSS a snaží se jím nahrazovat preprocessory. V mnoha případech to dává smysl, ale zrovna u definice breakpointů bych se touhle cestou nevydával.

Zápis definice i použití vypadá nadějně:

```
@custom-media --small-viewport (max-width: 30em);  
@media (--small-viewport) { }
```

Ovšem pozor – dnešní prohlížeče tomu nerozumí. Potřebujete tedy CSSnext, transpilátor budoucího CSS do stylů, kterým rozumí dnešní prohlížeče. CSSnext v případě uvedeného zápisu ale vychází z draftu (!) specifikace Media Queries 5. To je věc, která se může ještě mnohokrát změnit a do prohlížečů dorazí... až naprší a uschně.

V PostCSS bude ve srovnání s preprocesory navíc dost složité připravit logiku, kterou pro práci s kódem v Media Queries potřebujete. Tuhle cestu tedy zatím nedoporučuji. Pro zájemce je zde ještě můj článek o PostCSS (a CSSnext): vrdl.cz/p/postcss

Můj závěr je tedy jasný:

- Pokud můžete, využijte preprocesor a Sass MQ nebo podobnou malou knihovnu.
- Těšte se na Media Queries čtvrté generace.

Responzivní layout

Pro rozvržení celé stránky nebo jednotlivých komponent máme několik technických možností. Jako výchozí vám doporučím používat flexbox, ale projdeme si je všechny. Používáte rozvržení do mřížky? Za momentík vám povím něco i o něm.

Flexbox

Flexbox se prohlížeče naučily relativně nedávno, ale buďme za něj rádi, protože jde o první pořádný nástroj pro tvorbu layoutu v CSS.

Oproti „float“ a jiným technikám ze staré školy má řadu výhod.

Sám flexbox používám jako výchozí variantu pro jakékoliv v něm realizovatelné rozvržení stránky nebo její komponenty. Podpora v prohlížečích je téměř plná.

Vezměme jednoduché dvousloupové rozvržení stránky:

```
<div class="layout">
  <div class="col col-main"> ... </div>
  <div class="col col-complementary"> ... </div>
</div>
```

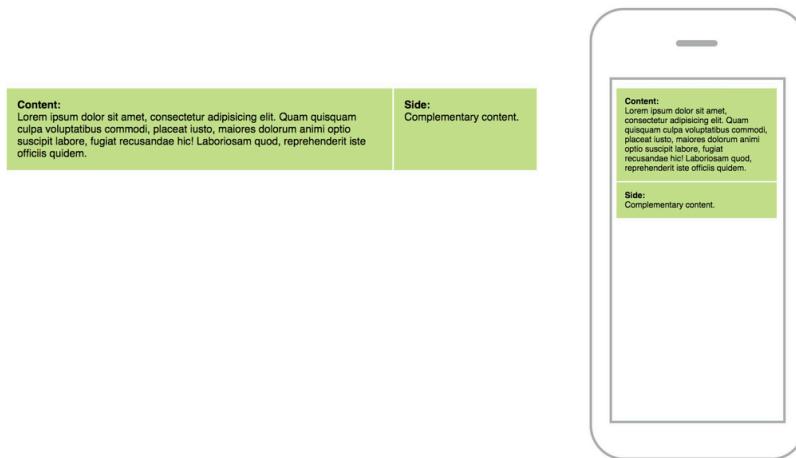
Sloupce bychom chtěli rozdělit na tři a jednu čtvrtinu šířky.

Jednoduchá krása flexboxu je pak vidět v CSS kódu:

```
.col-main { flex: 3; }
.col-complementary { flex: 1; }
```

Pomocí Media Queries, které už známe, pak layout nastavíme jen pro obrazovky od určité šířky:

```
@media only screen and (min-width: 40em) {
  .layout {
    display: flex;
  }
}
```



Jednoduchý responzivní dvousloupcový layout tří ků jedné realizovaný flexboxem.
Živá ukázka je pro vás připravená na CodePenu. cdpn.io/e/wobjoq

Flexbox je plný užitečných vlastností, ale kvůli tomu relativně složitý. Pro jeho studium doporučuji buď jednodušší online příručku na Vzhůru dolů, nebo svůj e-book „Vzhůru do CSS3“, kde jej podrobně vysvětluji i s příklady. vrdl.cz/p/css3-flexbox
[a vrdl.cz/ebook](http://vrdl.cz/ebook)

Brzy už i Grid Layout

Flexbox je velmi mocná technologie pro jednosměrný typ layoutu. Pokud ale potřebujeme rozvržení po vodorovné i svislé ose, s flexboxem se pracuje hůře. Pro ty účely nově připravilo konsorcium W3.org koncept rozvržení do mřížky, Grid Layout.

V době vydání knihy už jej některé prohlížeče podporují ve svých veřejných verzích. Na implementaci „gridu“ ale pracují prohlížeče všechny, takže jej už brzy budeme moci začít používat.

Podívejte se na specifikaci nebo web „Grid by Example“ od Rachel Andrew, kterou znáte z Webexpa 2016. w3.org/TR/css-grid-1/ a gridbyexample.com/examples/

Layout pomocí zastarávajících metod: `float`, `position:absolute`, `display:table`, `display:inline-block`...

Nic proti nim. Dlouho nám pomáhaly a tímto jim děkujeme. Je ale dobré vědět, že jde o techniky, které pro tvorbu rozvržení nebyly vymyšleny, takže mají mnoho nevýhod. Když můžete použít flexbox, použijte jej. Pokud flexbox použít nemůžete, inspirujte se návrhovými vzory pro layout na webu „This is Responsive“ od Brada Frosty. vrdl.in/rtpk3

Vícesloupcový layout pomocí vlastnosti `column`

Primárně slouží k zalamování textů do sloupců na širších displejích. Prostě k „novinové“ sazbě. Může se ale hodit na širších obrazovkách pro dodržení optimální šířky rádku, na což si jistě vzpomínáte z kapitoly o základech typografie. Na webu se to často nepoužívá, ale hodit se může. vrdl.cz/p/css3-multicolumn

Stručný průlet technickými možnostmi bychom tímto mohli uzavřít. Pojďme si ale ještě doporučit jednu netechnickou, designérskou metodu.

Tip: používejte mřížku

Mřížka (nebo také „grid“) rozděluje plochu stránky do sloupečků a jednotlivé komponenty rozhraní jsou pak do nich zarovnány.

Je to skvělý nástroj, protože výrazně zrychluje práci kodéra. Opět je to věc, kterou se webařina naučila z tisku. Všechny rozumné noviny a knihy jsou do nějaké mřížky vysázeny. Layout do mřížky díky své pravidelnosti usnadňuje uživatelům, aby stránku pochopili.

The screenshot shows the 'Getting started' section of the Bootstrap website. At the top, there's a navigation bar with links like 'Bootstrap', 'Getting started', 'CSS', 'Components', 'JavaScript', 'Customize', 'Themes', 'Expo', and 'Blog'. Below the navigation, the main title 'Getting started' is displayed in large, bold, white font. A sub-headline 'An overview of Bootstrap, how to download and use, basic templates and examples, and more.' follows. To the right, there's a sidebar with a 'Free' badge and a call-to-action: 'Learn to build better software & a more collaborative team! Get your copy now!' Below this, there's a list of links: 'Download', 'What's included', 'Compiling CSS and JavaScript', 'Basic template', 'Examples', 'Tools', 'Community', 'Disabling responsiveness', 'Migrating from 2.x to 3.0', 'Browser and device support', 'Third party support', 'Accessibility', 'License FAQs', and 'Translations'. The central area contains three main sections: 'Bootstrap' (with a 'Download Bootstrap' button), 'Source code' (with a 'Download source' button), and 'Sass' (with a 'Download Sass' button). Each section has a brief description below its title.

Vizualizovaný grid v Bootstraru, který toto rozvržení zpopularizoval. Jde ale o prastarý typografický koncept. getbootstrap.com

Kromě toho je práce s pravidelnou mřížkou pro vývojáře efektivnější. Pokud použijete některou z hotových knihoven v CSS, bude vám stačit pracovat s třídami v HTML a nebudeste muset každý kus layoutu stylovat zvláštním CSS kódem. V Bootstraru 4 například rozdělení stránky na dvě poloviny zařídíme takto jednoduše:

```
<div class="row">
  <div class="col"> ... </div>
  <div class="col"> ... </div>
</div>
```

Více informací o systému pro tvorbu rozvržení do mřížky

v Bootstrapu 4 najdete opět na Vzhůru dolů. vrdl.cz/p/bootstrap-4-grid

Příklad: rozvržení stránky

V příkladu žádný složitý layout nepotřebujeme. I tak si ale stavbu rozvržení stránky projdeme krok za krokem. Využijeme přitom totiž spoustu znalostí z předchozích částí knihy.

Struktura stránky

Ta teď vypadá následovně:

```
<div class="container">
  <header role="banner">
    <!-- Logo stránky -->
  </header>
  <main class="layout-container" role="main">
    <!-- Hlavní část stránky -->
  </main>
  <section class="layout-why" role="region">
    <!-- Sekce „Proč ForestKid?“ v patičce -->
  </section>
</div>
```

Pokud bychom se na zoubek podívali hlavní části stránky (`.layout-container`), je rozdělená takto:

```
<div class="layout-heading">  
    <!-- Název produktu -->  
</div>  
<div class="layout-main">  
    <div class="layout-photos">  
        <!-- Fotografie -->  
</div>  
    <div class="layout-text">  
        <!-- Text -->  
</div>  
</div>
```

Všimněte si, že došlo k určitému přeskupení sekcí. Sekci „Proč ForestKid?“ jsme přesunuli až na konec. Týká se celého webu, nikoliv konkrétního produktu a její pozice tomu prostě neodpovídala.

Až dosud jsme pořadí stránky přizpůsobovali malým displejům. Tuto změnu jsme udělali na základě přípravy pro layout ve větších velikostech okna prohlížeče. Pracujeme v *iteracích* a přímo v prohlížeči, kde jsou úpravy snadné. Špatné rozhodnutí z dřívějších fází vývoje nebude působit takové problémy.

V HTML ukázce pro zjednodušení vynechávám další potřebné atributy. Například `role`, které zlepšují přístupnost zařízeními pro odečítání obrazovky. Přidají prvkům stránky význam, který samy o sobě nenesou. Jak definovat strukturu v HTML5, píšu ve zvláštním článku na blogu. vrdl.cz/p/html5-struktura

Rozvržení hlavního obsahu flexboxem

Fotografie (`.layout-photos`) a text (`.layout-text`) rozdělíme do dvou stejně širokých sloupců.

Začneme tím, že jejich rodiče označíme za nositele layoutu:

```
.layout-main {  
    display: flex;  
}
```

Už to samo o sobě vykouzlí rozvržení do dvou sloupců. Flexbox ale vyjde ze šířky obsahu, takže na některých velikostech okna vám sloupce vykreslí různě široké. A to nechceme. Oba sloupce prostě rozdělíme na polovinu šířky rodiče:

```
.layout-text {  
    width: 50%;  
}  
  
.layout-photos {  
    width: 50%;  
}
```

Tohle rozvržení ale nechceme na mobilech. Proto ještě musíme vymyslet bod zlomu, od kterého layout nasadíme.

Bod zlomu a Media Query

Responzivní webdesignér zvětšuje a zmenšuje okno prohlížeče stejně často jako kuchař míchá vařečkou. Po přípravě layoutu tedy pomocí zmenšování okna hledáme minimální šířku, ve které layout dobře funguje. Díváme se, zda se nám nebortí důležité komponenty, ale posuzujeme i délku typografické rádky. Z [kapitoly o typografii](#) víme, že by měla být mezi 45 a 75 znaky. Tady hlídáme délku u jediného delšího textu na stránce: popisu produktu.

Minimální šířka okna, ve které layout funguje, je `800px`. Hodnota je však velmi blízko `768px`, což je menší rozlišení iPadů. Jak jsem psal v [kapitole o principech](#) návrhu responzivního rozhraní, snažím se o konzistenci rozhraní a obecně dost nerad těmto tabletům servíruji výraznějiný layout v režimu na výšku než v poloze na šířku. Je to

jedna z mála výjimek, kdy na můj výběr bodu zlomu mají vliv rozlišení zařízení. Většinou je ale lepší dávat přednost výběru podle obsahu komponenty nebo stránky. To už také víte z [kapitoly o tipech k Media Queries](#).

Tady tedy testuji, zda bych nedokázal layout udělat tak, aby fungoval už od 768 pixelů. A fungovat by tam měl.

V kapitole o Media Queries také píšu, že je lepší breakpointy nastavovat v jednotkám `em`. Do těch si naši `px` hodnotu musíme přepočít:

$$\begin{aligned} & 768\text{px} \\ & \div 16\text{px} \text{ (základní velikost písma)} \\ & = 48\text{em} \end{aligned}$$

A tím se dostáváme k výsledné podmínce pro nasazení layoutu.

```
@media only screen and (min-width: 48em) { ... }
```

Pokud jste pozorně četli [kapitolu o Media Queries](#), zápisu byste měli bez problémů rozumět.

Layout tedy máme hotový.

Příklad: s přimhouřením očí hotovo

Ano, je to tak. Příklad můžeme pro naše potřeby považovat za hotový. V [textu o přípravě fotogalerie](#) jsme si ukázali proces návrhu skicováním a prototypováním ukázkové komponenty. Než jsme se dostali sem, imaginární designér přemýšlel, skicoval, prototypoval a kódoval všechny ostatní komponenty uživatelského rozhraní. Až z něj kouřilo.

ForestKid

Dětské celoroční trekové boty Fare



Vel. Vn. délka **U vás**

<input type="radio"/>	23	154 mm	poštou
<input type="radio"/>	24	161 mm	poštou
<input type="radio"/>	25	167 mm	za 4 dny
<input type="radio"/>	26	174 mm	poštou
<input type="radio"/>	27	181 mm	na mý vyprodáno
<input type="radio"/>	28	188 mm	poštou
<input type="radio"/>	29	195 mm	poštou
<input type="radio"/>	30	202 mm	za 4 dny

ForestKid

Dětské celoroční trekové boty Fare



Vel. Vn. délka **U vás**

<input type="radio"/>	23	154 mm	poštou
<input type="radio"/>	24	161 mm	poštou
<input type="radio"/>	25	167 mm	za 4 dny
<input type="radio"/>	26	174 mm	poštou
<input type="radio"/>	27	181 mm	na mý vyprodáno
<input type="radio"/>	28	188 mm	poštou
<input type="radio"/>	29	195 mm	poštou
<input type="radio"/>	30	202 mm	za 4 dny

1 313 Kč
IČO: 259 000 000, IČD: 2002 KZ

Doprava a platba: Nad 1 500 Kč zdarma, jinak 80-150 Kč
Poštovné: ČR, Platba kartou, předovení i dohru zásilky.
Vše

Jak vybrat správnou velikost?
[Video](#)

Hotové produkty zákazníky ForestKid.cz 95 %.
(Během ze 4 hodin)

Přidat do košíku

Takhle stránka příkladu vypadá v celé své kráse

Poslední krok ukázkové stránky si můžete stáhnout a otevřít zde na následujících odkazech.

- Otevření v prohlížeči: vrdl.in/vdwdhot
- Stažení v ZIPu: vrdl.in/vdwdhotzip

Hotové to je a není

Dokončili jsme stránku, ale web zůstává nehotový. Jak víte, už na začátku procesu jsem zjednodušoval. Udělal jsem obsah a pak prototyp jen pro jednu stránku celého webu a vynechal jsem stránky nákupního procesu, úvodní stránku a celý blog a diskuze. To je obsahová struktura, kterou bychom jako celek měli mít v hlavě před začátkem prací. Kromě celých sekcí webu vynechávám také globální komponenty jako hlavičku s primární navigací a patičku, kde obvykle sídlí sekundární navigace.

Je to jen příklad, doma to raději nezkoušejte

Příklad jsem do knihy přidal, abych prakticky ukázal důležité principy responzivního designu a procesy uvažování s ním spojené.

Výsledek ale rozhodně není perfektní. UX designéři a marketéři by určitě našli chyby už v počátečním posuzování cílových skupin v design canvasu. Profesionální copywriteři by mi mnohé vytkli k textům. Zkušení grafici zase ve výsledku uvidí můj nedostatek citu pro detail a smyslu pro grafické vtípky. A hravost by zrovna u tohoto typu projektu byla na místě.

Výtky by byly v pořádku, nedokonalostí jsem si vědom. Na reálných projektech jsem jen kolečkem v širším týmu a zmíněné obory nechávám kolegům a kolegyním expertům. Pokud snad s webdesignem začínáte, rozhodně se nesnažte vše zvládnout sami. Víc hlav udělá lepší weby, věřte mi.

Zapamatujte si

- Pokud můžete, layout navrhujte do mřížky.
- Body zlomu se snažte nastavovat na míru projekt, podle obsahu konkrétních komponent, podle designu a podle cílové skupiny.
- Správu bodů zlomů si usnadňte CSS preprocesorem.
- Media Queries mějte v produkčním kódu vždy v jednotkách `em`.
- CSS nerozdělujte podle velikostí obrazovky, ale podle komponent rozhraní.
- V Javascriptu změnu velikosti okna detekujte pomocí funkce `matchMedia()`.
- Pokud to jde, kód vždy pište způsobem Mobile First.
- Naučte se flexbox a časem i CSS Grid Layout.

Kapitola 10:

Responzivní navigace

Navigace je paní Columbová mobilního webdesignu. Poručík Columbo o ní mluví, ostatní tuší, že někde bude. Ale jen málokdo ji dokáže bez problémů najít.

1. V první části kapitoly budu pochybovat, zda je schovávání paní Columbové vůbec rozumné.
2. V druhé části se vám pak pokusím představit všechny možné návrhové vzory responzivních navigací.

Mobilní navigace: potřebujeme hamburgery?

Hlavní navigační oblasti webů jsme zvyklí dělat složité. Tak složité a strukturované, že se na malé mobilní obrazovky nevejdou. Proto je tam schováváme a opatřujeme vypínačem.

Jenže ono to schovávání není zase tak dobrý nápad. Pojďte, vezmeme to z gruntu a pozastavíme se také u ikony hamburgeru, oblíbeného tématu diskuzí.

Nejdříve mi ale dovolte jednu zásadní otázku.

Proč jsme vlastně navigace neschovávali už v době čistě desktopových webů?

Protože navigace musí na webech plnit minimálně tři úkoly:

1. *Mapa*

Uživatel by měl z navigace snadno pochopit strukturu webu a najít díky ní, co hledá.

2. *Ukazatel*

Uživatel by měl vědět, kde se na mapě aktuálně nachází.

3. *Reklama*

Zájmem provozovatele webu je, aby uživatel nepřišel o nic zajímavého. A právě na to může také hlavní navigace upozorňovat.

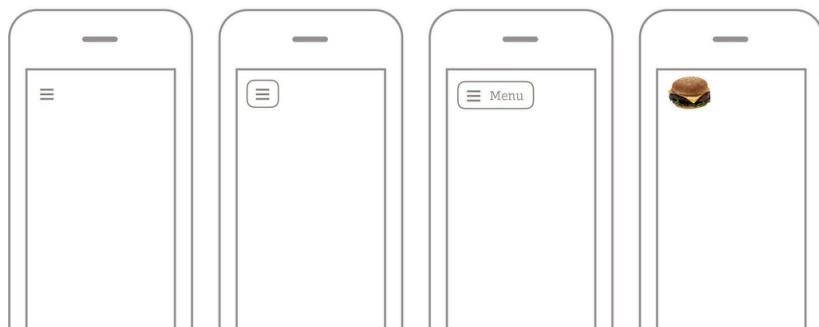
Zeptejme se teď sami sebe: Jak dobře tyto tři úkoly plní schovaná navigace? Že je neplní vůbec? Bingo!

Moc špatný
hamburger

Špatný
hamburger

Dobrý
hamburger

Moc dobrý
hamburger



Dobré a špatné hamburgery

Řešení pro ideální svět stručných navigací na velkých displejích je jednoduché – prostě bychom navigaci neschovávali. Jenže mnozí z nás navrhují weby se složitými navigacemi a mnozí uživatelé mají zařízení s velmi malými displeji.

Jak uvažovat při návrhu hlavní navigace?

Snažte se ji zjednodušit a zobrazit alespoň její část. Pokud se stane, že na některých zařízeních navigaci uživatelé neuvidí, navrhujte web tak, aby na ni nemuseli spoléhat.

1. Navrhněte navigaci tak, aby byla co nejjednodušší

Ano, už při vymýšlení struktury webu aktivujte režim Mobile First. Na šedesát osm položek ve třech úrovních hlavní navigace raději zapomeňte. Jako designéři můžete uživatelům dát i jiné možnosti, jak procházet složité struktury webů: Jako příklad jmenujme vyhledávání nebo katalogové procházení.

2. Navrhněte web, jako by tam navigace nebyla

Je pravděpodobné, že na těch nejmenších displejích budeme muset skoro vždy část navigace schovat. Proto se musíme naučit vymýšlet weby bez ní. Navigační schéma duplikujte v obsahu nebo třeba dejte do úvodní stránky něco jako mapu webu.

3. Pokud to jde, na mobilu ji prostě zobrazte celou

Svět je plný webů s navigací o čtyřech položkách, které používají hamburger jen proto, že to je „in“. Pokud je rozlišení dostatečně široké, navigaci zobrazte.

Jestliže navigaci nedokážete zobrazit celou, zvažte jiné návrhové vzory než prosté zapínání a vypínání. Až pokud žádná z těchto možností nezabrala, volte návrhový vzor s vypínačem navigace. Ano, ten, kterému podle vzhledu ikony říkáme „hamburger“.

Webdesign ikonu hamburgeru potřebuje

Ikona hamburgeru a schovávání navigace jsou v poslední době pod palbou kritiky. Kromě skrývání důležitého obsahu se jako nevýhoda uvádí nízká srozumitelnost ikony běžným uživatelům.

Ale na některých webech se „hamburgeru“ nedá vyhnout. Proto tvrdím, že ji webdesign jako obor potřebuje.

O ikonách je známo, že trvá nějakou dobu, než se mezi uživateli zavedou. Lidé se nenařodili ani se znalostí ikon pro *play*, *pause* a *stop* na hudebních přehrávačích.

Jedním z hlavních důvodů, proč jsme schopni symboly (play, pause a stop) používat bez textového značení, je skutečnost, že si svou cestu mezi komunikační zkratky naší kultury našly díky neustálému opakování na magnetofonech a videopřehrávačích.

Píše to Andy Budd v článku „In defence of the hamburger menu“. vrdl.in/28gc0

Ikona hamburgeru je nová a pro autory webů je výhodné, abychom ji (stejně jako ikonu pro *play*) uživatele naučili. Dosáhneme toho ale jen konzistentním používáním na webech. Prostě tím, že tři vodorovné čárky budou vždy znamenat totéž: otevření seznamu navigačních položek.

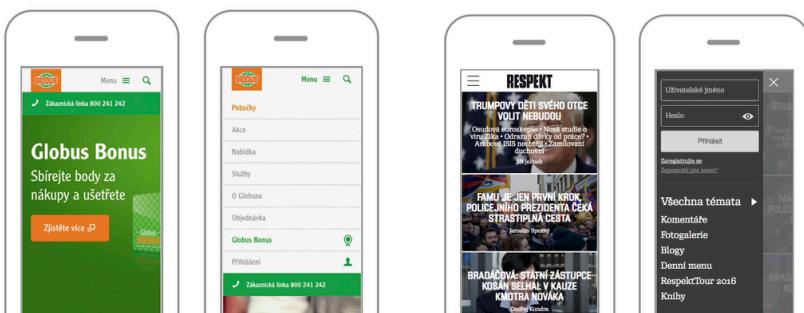
Otevírá hamburger opravdu obsah, který reprezentuje?

Každá ikona je zjednodušenou abstrakcí obsahu, který následuje po

její aktivaci. V případě ikony hamburgeru jde o seznam položek řazených pod sebou. Odpovídá to ale třeba víceúrovňové navigaci nebo celé liště sekundárního obsahu, na který se dostanete kliknutím na hamburger například u jinak výborného webu Respektu?

Dobrý
Ikona odpovídá
rozbalovanému obsahu

Horší
Ikona neodpovídá
rozbalovanému obsahu



Bylo by lepší, kdyby „hamburger“ vždy otevíral totéž, tedy seznam položek. Bohatší obsah, jaký otevřá na webu Respektu, by bylo vhodnější ikonou neopatřovat

Dobře míněné rady pro správný návrh ikony otevírající navigaci

Takže – na některých webech je možné navigaci zobrazit celou i na mobilech. Autoři jiných webů zase použijí chytřejší navigační vzor, jako je prioritizace položek.

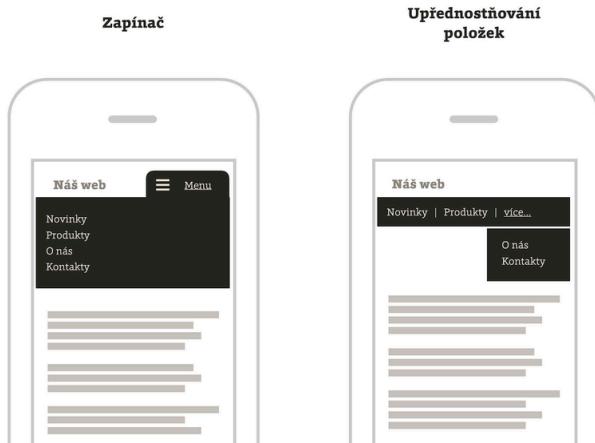
Zbývá nám tu množina webů, u kterých je schovávání navigace nebo její části nevyhnutelné. Takže ano, hamburger je dobrý, protože jej někdy dost nutně potřebujeme. Je dobré ale při návrhu myslit na následující:

1. Ikonu prosím nezneužívejte pro typ obsahu, který nepředstavuje.
2. Ikonu opatřete textovým popiskem „Menu“, aby byla přístupná i uživatelům, kteří hamburger neznají. Respektive vždy popiskem, který přesně popisuje obsah, například „Kategorie“ nebo „Recepty“.
3. Ikona by měla mít vzhled tlačítka. Z testů vyplývá, že lidé pak snadněji pochopí, že jde o aktivní prvek rozhraní.
exisweb.net/menu-eats-hamburger

Návrhové vzory responzivních navigací

Projdeme si osm možností, jak navrhnut navigaci na responzivních webech. Od uživatelsky složitých a implementačně jednoduchých až po ty, ehm, dobré.

Většina webů si ale vystačí se dvěma: upřednostňujícím a přepínacím vzorem. Podívejme se nejprve na ně.



Hlavní návrhové vzory pro responzivní navigace: zapínač (také přepínač nebo Toggle) a upřednostňování se schováváním (Priority+)

Upřednostňování položek (Priority+)

Se zmenšující se šírkou obrazovky se zmenšuje i počet položek v navigaci. Na mobilních šírkách obrazovky pak zůstává opravdu jen to důležité. Ostatní položky navigace jsou obvykle dostupné na rozbalení.

Technicky je možné tento návrhový vzor řešit i dost jednoduše jen pomocí CSS. cdpn.io/e/adeMzP

Přepínač (Toggle)

Celou navigaci na malých displejích schováte do tlačítka. Nejčastěji s ikonou „hamburgeru“. Je to nejpoužívanější typ responzivní navigace, jenže má svá úskalí a často je vhodnější použít jiný. Jeho nevýhodou je hlavně schování celé navigace, která tím přestává

zastávat role, jež má na webu hrát. O schovávání navigace a hamburgeru jsem už psal.

Návrhový vzor *přepínač* zpopularizoval například framework Bootstrap a najdete jej na většině dnešních responzivních webů.

Speciální varianta přepínače: vyjíždění do obrazovky (Off Canvas)

Chování, které znáte z nativních aplikací. Nejobvyklejší je vyjíždění ze strany, které je vhodné pro schovávání bohatého obsahu. V tom případě ale není dobrý nápad použít ikonu hamburgeru. Podívejte se na ukázky od Luka Wroblewskiego. lukew.com/ff/entry.asp?1569

Víceúrovňové navigace: speciální varianta a speciální úroveň ošemetnosti

Oba už zmíněné návrhové vzory je samozřejmě možné použít i pro víceúrovňové navigace. Jen opatrně, protože uživatelské ovládání na mobilech nemusí být nejpříjemnější. Řešení vždy testujte.

Demo víceúrovňové navigace na pěkném webu „Adventures in Responsive Navigation“. vrdl.in/t2n7r

Konverze do jiného stylování

Pokud v navigaci zvládnete mít jen nízké jednotky položek, můžete je zobrazit všechny a na malých obrazovkách jen upravit stylování.

Přestyllování a skok do patičky

Navigace je v HTML někde dole. Na malých displejích je umístěná v patičce. Na velkých displejích se pomocí stylů umístí do hlavičky. Implementačně je *skok do patičky* asi nejméně náročná volba. Uživatelsky ovšem dle mého názoru málo přívětivá. Umístění

navigace je takto nekonzistentní napříč zařízeními. Dnes už bych skok na běžné weby nepoužil. Snad jen v opravdu jednoduchých případech. vrdl.in/9em7w

Konverze do <select>

Navigaci pomocí Javascriptu na mobilech proměňte v nativní seznam položek. Výhodou je opět jednoduchá implementace. Nevýhodou horší uživatelská přívětivost seznamu položek na mobilních zařízeních. Ukázka je na CSS Tricks. vrdl.in/navsel

Navigace jen v patičce

Na některých webech hlavní navigaci v hlavičce nepotřebujete. Žádná speciální péče o mobilní zařízení pak není potřebná.

Vzor „nohy nahoru a nedělat nic“

Raději zmíním i tento „návrhový vzor“. Jak jsem psal v [textu o ikoně hamburgeru](#), často se navigace na mobilech schovává zbytečně. Když v ní máte velmi málo položek, tohle je nejlepší řešení: uvařit si kafe, dát nohy nahoru a nedělat nic.

Video: [Responzivní navigace](#) ~ Jaký typ responzivní navigace vybrat? Projdeme si osm návrhových vzorů pro navigace na responzivních webech a ukážeme si příklady ze skutečných webů.

Zapamatujte si

- Navigace slouží také jako ukazatel, mapa webu a reklama na obsah. Schovaná navigace tyto účely neplní.

- Když to jde, zredukujte počet položek v navigaci a snažte se i na mobilech zobrazit alespoň její část.
- Kromě „hamburgeru“ s přepínačem zvažte i návrhový vzor Priority+ nebo zobrazení nejdůležitějších položek navigace.
- Ikonu „hamburgeru“ doplňte popiskem reprezentujícím její význam („Menu“).
- „Hamburger“ prosím používejte konzistentně pro účely otevření seznamu navigačních položek.

Kapitola 11: Testování responzivních webů

Vraťme se teď společně na začátek, do první kapitoly o nových displejích a zařízeních. Při tvorbě responzivních webů díky ní počítáme s mobily, tablety a počítači. Počítáme s hybridními laptopy nebo třeba profesionálními tablety, jako je iPad Pro.

Počítáme se zvětšujícími se desktopovými displeji. Počítáme s mnoha prohlížeči v nich a desítkami jejich verzí. Myslíme i na různé typy displejů, různé rychlosti připojení a mnoho dalších aspektů poměrně nehostinného prostředí webových tvůrců. Jak ale svůj web dobré otestovat a nezbláznit se z toho?

Test responzivních webů nemusí být žádný *trest*. Je dobré vědět, že nemusíte vlastnit všechna zařízení světa. Stačí mít komplexitu webového prostředí v hlavě už při návrhu designu a psaní kódu. A tamtéž mít uložené znalosti pro hledání a odstraňování chyb. V poslední kapitole na vás čeká text o testování a ladění responzivních webů. Vzhůru do něj!

Testování a ladění webů na zařízeních

Jak si poradit s dnešní škálou prohlížečů a zařízení? A jak s nepřítomností pořádných vývojářských nástrojů na mobilech? Svatý grál neznám, ale pár tipů ze své praxe bych měl.

Mé testování je čtyřfázové:

1. Prototypování
2. Vývojářský desktopový prohlížeč
3. Alternativní prohlížeče pomocí nástroje BrowserStack
4. Reálná zařízení

1) Prototypování (na CodePenu)

O prototypování jsem už psal [ve zvláštní kapitole](#). Ještě než začnu něco finálně implementovat, procházím touto první, prototypovací fází. Na prototypu si vyzkouším nejkritičtější designérské i kodérské problémy. Používám CodePen, kde je to za chvíliku hotové a výsledek můžu rychle otestovat ve všech možných prohlížečích. Dále pak Bootstrap nebo prostě HTML, CSS a Javascript.

2) Vývojářský desktopový prohlížeč (s Chrome DevTools)

V téhle fázi trávím samozřejmě nejvíce času. Kvůli DevTools používám Chrome.

Občas se podívám do ostatních desktopových prohlížečů: Firefoxu, Exploreru, Safari a Edge. Méně často do Opery, která obvykle renderuje stejně jako Chrome.

Protože se ale bavíme hlavně o mobilních zařízeních, v Chrome mám puštěný Device Mode. Neznám lepší nástroj na emulaci všeho možného mobilního a díky tomu v Chrome trávím během procesu návrhu i kódování webu nejvíce času.

Najdete ho pod nenápadnou ikonkou mobilu nalevo od hlavního menu. Nebo pod zkratkou **Ctrl/Cmd+Shift+M**, když máte DevTools otevřené (**Ctrl/Cmd+Shift+I**).

Obsahuje přednastavené profily zařízení, emulaci pomalého mobilního internetu, emulaci uživatelského zoomování a další vychytávky.

Alternativy v ostatních prohlížečích

Něco podobného existuje ve Firefoxu (Responsive Design View – **Ctrl/Cmd+Alt+M**) nebo v Safari, a dokonce v Edge (**F12 / Záložka „Emulation“**). Alternativy jsou ale dle mých zkušeností slabší.

Při testování v Chrome si navíc odladíte nejpopulárnější desktopový i mobilní prohlížeč. Ano, Firefox i Edge mají mobilní bratry, na které bychom neměli zapomínat, ale jejich podíl je malý na to, abyste z nich mohli udělat primární zařízení pro jednodušší testování mobilů.

V Chrome obvykle oknem prohlížeče hýbu a upravuji kód, dokud se mně design alespoň trochu nelibí ve *všech* rozlišeních.

V další fázi potřebuji testovat v reálnějším prostředí. Ano, jde o jiná vykreslovací nebo javascriptová jádra. Pomůže BrowserStack nebo simulátory.

3) Alternativní prohlížeče (pomocí BrowserStacku)

BrowserStack je výborný nástroj, který mi ušetřil spoustu času. Rovnou upozorním, že za live verzi, kterou používám, zaplatíte kolem pěti set korun na osobu měsíčně. Tak, a tím bychom vyčerpali výčet jeho nevýhod.

Bezplatná alternativa existuje, ale je složitější. Simulátory a emulátory nejsou multiplatformní a zaberou moc času při instalaci, správě i spouštění.

BrowserStack naproti tomu:

- běží v prohlížeči a je naprosto multiplatformní,
- nabízí svižnější čas startu i přepínání mezi prohlížeči,
- nevyžaduje vaši pozornost při instalaci a aktualizaci,
- poskytuje možnost testování na reálných zařízeních.

Takže už možná chápete, proč ta pětistovka měsíčně nemusí vůbec bolet. [browserstack.com](#)

Video: [BrowserStack](#) ~ Jak testovat web ve všech prohlížečích a nemuset řešit virtuály a emulátory.

Kdybyste chtěli levnější alternativu, mrkněte se na [CrossBrowserTesting.com](#). Jsou tam jen emulátory a je to pomalejší. Stojí to přibližně sedm stovek, ale v této ceně je i generování screenshotů a spouštění Selenium testů, které jsou u BrowserStacku za další peníze.

Následující část čtěte, jen pokud hodně chcete šetřit a máte vysokou odolnost proti bolesti. V opačném případě přejděte k testování na reálných zařízeních.

Simulátory a emulátory (levná alternativa k BrowserStacku, kterou nedoporučuji)

Mobilní Chrome jakžtakž odpovídá tomu desktopovému, takže potřebujete otestovat hlavně mobilní Safari.

Simulátorem je možné na Macu testovat iOS (nebo také watchOS a tvOS), ale také mobilní Safari. [vrdl.in/sgo6c](#)

4) Testování na fyzických zařízeních

Pozor, simulátory ani BrowserStack vám nestačí! Proč?

- Web si musíte osahat vlastním palcem. Klikání myší

v simulátoru tohle nenahradí.

- Simulátory nenasimulují problémy s výkonností.
- Občas se stane, že v reálném zařízení se věci vykreslují jinak než v simulátoru.

Pokud to myslíte s responzivními weby vážně, určitě k ruce potřebujete ještě nějaká reálná zařízení.

Jaká zařízení si pro testování pořídit? Nejlépe všechna! Že vám to rozpočet nedovolí? Mně taky ne, takže do začátku vás může inspirovat seznam zařízení, na kterých testuji svou práci. Budu je řadit podle toho, jak důležitá mi připadají pro testování dnešních webů.

Budete potřebovat hlavně smartphony i tablety z obou nejrozšířenějších platform (Android a iOS). Nezapomínejte prosím ani na starší zařízení. Jsou menší a méně výkonná. Uživatelská základna je velmi fragmentovaná. Nevěřte tomu, že všichni vaši uživatelé vlastní poslední „ajfoun“.

Telefony:

- iPhone 6 Plus s iOS 10 jako zástupce moderních phabletů.
- Samsung Galaxy S III Mini s Androidem 4, Chrome a Android Browserem. Podprůměrný starý Android.
- iPhone 4 s iOS 7 jako zástupce starých a pomalu vykreslujících iOS zařízení.
- Vodafone 945 se starým Androidem 2.1 a rozlišením 240×400 pixelů. Ano, i na vaše weby se možná občas dívám přes jeden z nejhorších mobilů, jaké si umíte představit.

Tablety:

- iPad Mini s iOS 8. Jeden z menších tabletů, přitom plnohodnotně použitelný (a u nás doma používaný). Také velmi prodávaný.

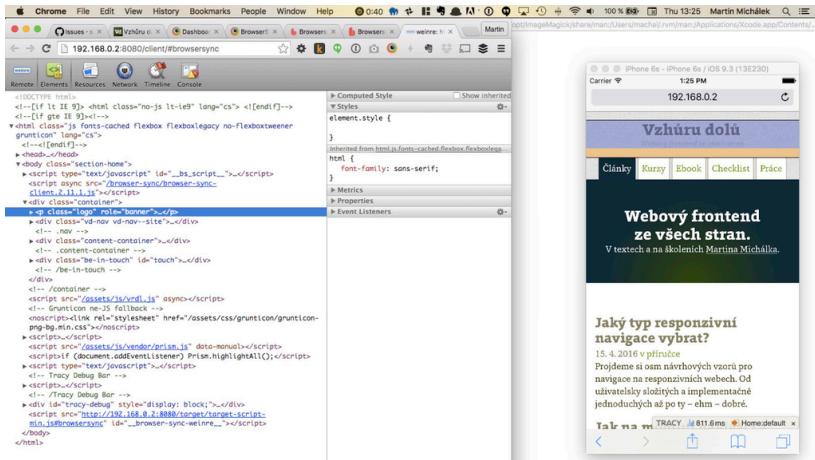
Určitě doporučuji pořídit, hlavně kvůli otestování dostatečné velikosti aktivních prvků v uživatelském rozhraní. Rozlišení 1024×768 na sedmi palcích: Garantuji, že se váš grafik zapotí.

- Tablet Lenovo TAB 2. Klasický desetipalec s nejnovějším Androidem.
- Sencor Element 7 s Androidem 4.1, nechutně pomalým prohlížečem a rozlišením 480×800 pixelů. A také displejem prasklým od našeho mladšího chlapečka. To je prosím simulace těžkých podmínek pro prohlížení webu.

A nezapomeňte na *guerilla* testování – takové prodejny Alzy nebo Datartu bývají plné zařízení, na kterých si můžete leccos vyzkoušet. Z pohledu designéra je zajímavé každé nové zařízení, takže kdybyste se kolem mě s nějakým takovým vyskytli, pravděpodobně vám ho na chvíli ukradnu a budu na něm své weby testovat.

Browsersync a multiplatformní „DevTools“

Browsersync je obecně velmi přínosný nástroj pro testování responzivních webů. Ted nás ale zajímá hlavně jeho schopnost poskytnout něco jako *DevTools*, ovšem multiplatformně. Obsahuje nástroj jménem Weinre, se kterým je ladění webů na mobilech velmi příjemné. vrdl.cz/p/browsersync



Browsersync v kombinaci s BrowserStack. Browsersync a Weinre umožňují kombinovat platformy. Takže třeba Firefox na Macu propojit s Explorerem na Windows Phone. Tedy ne, že byste to v roce 2018 moc často potřebovali

Weinre jsou ale jen něco jako DevTools. Plnohodnotným vývojářským nástrojům dnešních prohlížečů konkurovat nemohou.

Propojení bratrských prohlížečů

- *Mobilní Chrome*

S jeho desktopovým sourozencem propojíte od Androidu verze 4. Mobilní zařízení stačí propojit USB kabelem a pak ještě nastavíte pár věcí podle následujícího návodu. vrdl.in/7ztbj

- *Mobilní Safari*

S desktopovým Safari propojíte samozřejmě jen na Macu. Musíte si také nainstalovat Xcode, což obecně doporučuji, pokud na Macu děláte jakoukoliv vývojařinu. Po propojení zařízení kabelem se pak podívejte do nové položky v menu prohlížeče Safari, která se bude jmenovat „Developer“. V druhé sekci jsou připojená zařízení. Následuje podrobnější návod. vrdl.in/u60bs

Zapamatujte si

- Prohlížečů, verzí platform a modelů zařízení je hodně.
Nespoléhejte na ty, které vám leží na stole.
- Prototypujte. Už to vám odchyti řadu problémů.
- Testujte. Ve vývojářských nástrojích prohlížeče, emulátorech i zařízeních.
- Když navrhujete rozhraní, nutně si jej musíte vyzkoušet i na různých fyzických zařízeních.
- Pokud můžete, zaplatě si BrowserStack.

Na závěr

Právě jste dočetli „Vzhůru do (responzivního) webdesignu“!

Díky, že jste knize věnovali svůj čas, a jsem napjatý, zda se vám líbila.

Zpětná vazba mi moc pomůže

Moc rád si přečtu váš názor nebo zpětnou vazbu. Pomohla vám knížka? Vylepšili byste ji něčím? Pište mi na e-mail, na Twitter nebo na Facebook.

I když jsem udělal vše pro to, abych se chybám vyhnul, nezbývá než se smířit s tím, že i v tomto textu nějaké zůstaly. Moc mi pomůže, když je nahlásíte. Všichni čtenáři dostanou jednou za čas aktualizovanou elektronickou verzi.

- E-mail: martin@vzhurudolu.cz
- Facebook: facebook.com/vzhurudolu
- Autor na Twitteru: twitter.com/machal

Na Twitteru nebo Facebooku můžete použít hashtag „#VzhuruDoWebdesignu“.

Můj první e-book: „Vzhůru do CSS3“

Získáte v něm hlubší technické znalosti o webové koděřině: nástroje jako NPM nebo Grunt a nové CSS3 vlastnosti, od kulatých rohů po

flexbox. Oba e-booky koncipuji tak, aby se doplňovaly.
vzhurudolu.cz/ebook

Kam dál?

- Blog Vzhůru dolů. vzhurudolu.cz
- Veřejná a individuální školení autora a spolupracujících expertů. vzhurudolu.cz/kurzy
- Autor též poskytuje poradenství firmám i jednotlivcům.
vzhurudolu.cz/martin

Poděkování

Velmi děkuji všem recenzentům, kritikům, rejpalům... Prostě kolegům a kolegyním, kteří trpělivě četli, hlásili chyby, upozorňovali na nejasnosti a ptali se „proč?“ tam, kde jsem se sám sebe zeptat zapomněl.

Zásadním způsobem mi pomohli knihu posunout zejména Dan Srb a Jirka Sekera. Děkuji také Janu Kvasničkovi, Danu Střelcovi, Zuzaně Šumlanské, Radku Pavlíčkovi, Danu Duranovi, Janu Polzerovi, Michalu Miklášovi, Tomáši Musiolovi, Kristině Vořanské a Honzovi Sládkovi.

Knížka by nikdy v této podobě nevyšla, nebyt podpory (a přiměřené kritiky) mé ženy Jano, díky!

Za poskytnutí materiálů k produktům ukázkového e-shopu děkuji panu Jaroslavu Naňákovi ze společnosti Fare.

Foto autora je od [Jana Forejta](#).

Za připomínky k verzím 1.1 a 1.2 děkuji těmto dobrým duším: Ivan

Boukal, Jiří Černý, Jakub Honíšek, Michal Horáček, Martin Kavík,
Ondřej Konečný, Michal Maňák, Lubomír Merta, Mário Raček,
Vojtěch Sláma, Ladislav Suchánek, Zuzana Štočková a její studenti,
Martin Pešout. Díky!

Martin Michálek

Vzhůru do (responzivního) webdesignu

Obrázky a schémata: Martin Michálek a Daniel Střelec

(DanielStrelec.cz)

Grafická úprava: Martin Michálek

Obálka: Petr Šťastný (Raist.cz)

Jazyková korektura: Petr Behún (Proofreading.cz)

Web: Martin Michálek, Daniel Střelec a Přemek Koch

V knize jsem použil písma Capita a Foro od Dietera Hofrichtera
(Hoftype.com)

Verze 1.2, listopad 2018