Obsah

1. Rozvrhnutie obsahu webovej stránky – page layout

Návrh rozvrhnutia obsahu webovej stránky patrí medzi najťažšie problémy, s ktorými sa web dizajnér stretne. Problém spočíva v tom, navrhnúť rozvrhnutie tak, aby spĺňalo niekoľko podmienok (ktoré si na prvý pohľad môžu odporovať) súčasne. Rozvrhnutie by malo byť pre používateľa logické, prehľadné, použiteľné a prístupné (ale o tom neskôr).

Techniky rozvrhnutia obsahu webovej stránky

Existuje niekoľko rôznych techník a ich kombinácií, ktoré môžeme použiť. Štyri najpoužívanejšie sú:

- prirodzené rozvrhnutie
  - elementy akceptujú prirodzený tok elementov a vytvárajú tak jednoduché, lineárne usporiadanie stránky,
  - pokiaľ nepovieme inak, aplikuje sa tento prístup,
  - pokiaľ ste doteraz neriešili umiestnenie elementov v okne prehliadača, využívali ste práve prirodzené rozvrhnutie,
- rámce (frames)
  - na rozvrhnutie obsahu sú použité rámce (okno prehliadača je rozdelené na niekoľko častí a v každej časti je zobrazený konkrétny html dokument),
  - HTML5 už rámce nepodporuje, dá sa však využiť element iframe, ktorý má niektoré výhody rámcov,
- tabuľky
  - na rozvrhnutie obsahu sú použité tabuľky,
- CSS
  - na rozvrhnutie obsahu sú použité CSS,
- iné
  - napr. flexbox (toto je len jedna z vlastností CSS), CSS šablóny, CSS frameworky (aj keď tieto možností len využívajú vopred definované štýlopisy, prípadne v kombinácii s JavaScriptom).

Neexistuje jednoznačná odpoveď na otázku, ktorú možnosť použiť a ktorú nie. Záleží od množstva faktorov. Každá možnosť ponúka isté výhody a isté nevýhody.

Ukážeme si, ako navrhnúť rozmiestnenie obsahu webovej stránky pomocou prvých štyroch techník. Použijeme päť typických častí webovej stránky: hlavičku, menu webu (odkazy do webu), obsah konkrétnej stránky webu, iné odkazy (odkazy mimo webu, resp. reklamný priestor) a pätičku. V našom príklade budeme predpokladať nasledovné rozvrhnutie obsahu webovej stránky (viď. obrázok nižšie).

<table>
<tr><td colspan="3">hlavička webu</td></tr>
<tr><td>menu
ponuka 1</td><td>obsah stránky webu
Lorem ipsum dolor sit amet consectetuer eu interdum rutrum Sed nec. Curabitur</td><td>odkaz 1
odkaz 2</td></tr>
</table>

ponuka 2 ponuka 3 ponuka 4	Suspendisse Curabitur vitae vitae lacinia ante et adipiscing velit at. Elit sit id montes augue nec eget dui tincidunt Aenean at. Mattis congue tristique Curabitur sem laoreet pellentesque gravida amet tincidunt In. Congue orci ut id pede velit Ut.	odkaz 3 odkaz 4
pätička webu		

V nasledujúcej časti sa kvôli jednoduchosti vyhneme formátovaniu (farby, pozadia, zarovnania atď.) Ide nám o demonštráciu jednotlivých spôsobov rozvrhnutia obsahu stránky.

Rozvrhnutie obsahu v HTML5

V (X)HTML sa na obalenie jednotlivých častí používajú elementy `div`. V HTML5 sa namiesto bezvýznamového (v zmysle sémantiky) `div` elementu používajú sémantické elementy[1]:

- `header` - hlavička,
  - hlavička sa používa pre úvodný obsah alebo navigáciu, typicky môže obsahovať logo, informácie o autorstve, identifikáciu autora a pod., môže obsahovať vlastné nadpisy, v dokumente môže byť viac častí `header`, napr. jedna pre celý dokument a ďalšie pre jednotlivé sekcie,
- `nav` - navigácia,
  - obsahuje navigáciu v rámci webu alebo stránky, do tejto časti sa umiestňuje len hlavná navigácia a nie všetky odkazy, ktoré sa na stránke nachádzajú,
- `section` - sekcia v dokumente,
  - sekcia zoskupuje tematický obsah, často so samostatným nadpisom, typicky môžeme mať jednu sekciu pre úvod, jednu pre samostatný obsah a jednu pre kontaktné informácie
- `article` pre nezávislý, samostatný obsah, článok
  - táto časť by mala dávať zmysel sama o sebe aj keď ju zo stránky vyberieme ako samostatný obsah, môže obsahovať vlastné nadpisy,m
- `aside` - vedľajší bočný panel,
  - umiestnime sem sekundárny obsah, súvisiace odkazy smerujúce mimo web,
- `footer` – pätička,
  - pätička môže obsahovať informácie o autorských právach, kontaktné údaje, odkaz na mapu stránky, odkazy na súvisiace dokumenty a pod.,

[1] Okrem tu uvedených elementov, HTML5 ponúka aj množstvo ďalších štrukturálnych sémantických elementov. Viac napr. na: https://www.w3schools.com/html/html5_new_elements.asp.

v dokumente môže byť viac častí `footer`, napr. jedna pre celý dokument a ďalšie pre jednotlivé sekcie,

Neexistuje jednoznačné pravidlo, či v `section` majú byť vnorené viaceré `article` alebo naopak, či v `article` majú byť vnorené viaceré `section`. Na existujúcich weboch sa môžete stretnúť s oboma prípadmi.

Ak používame HTML5, je vhodné z hľadiska prístupnosti použiť vyššie uvedené elementy pre obalenie jednotlivých častí stránky. A to aj v prípade ak neriešime rozvrhnutie obsahu stránky.

Prirodzené rozvrhnutie obsahu stránky

Využijeme vo veľkej miere elementy (jazyka (X)HTML)[2], ktoré majú svoj sémantický význam. Stránka je jednoduchá, prehľadná a prístupná.

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/prirodzene.html]

Výhody:

- jednoduché a rýchle,
- stačia nám len elementy (X)HTML,
- dokument je prístupný, ľahko čitateľný,
- zachováva si rovnaký vzhľad aj v textových prehliadačoch,
    - prirodzené rozvrhnutie obsahu webovej stránky vyzerá v textovom prehliadači LYNX asi takto: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/prirodzene.php,[3]
- stránky majú logickú lineárnu štruktúru, pretože ju už z princípu (X)HTML navrhujeme ako lineárnu.

Nevýhody:

- nemožnosť (pokročilého) umiestnenia elementov,
- stránky majú len lineárnu štruktúru,
- pri rozsiahlejšom obsahu sa v prípade dosiahnutia konca dokumentu navigácia a hlavička nezobrazia,
- pri načítavaní iných stránok webu sa opakovane načítavajú rovnaké časti (napr. menu, hlavička a pod.)

[2] Pre vizuálne oddelenie obsahu od hlavičky a päty sme použili element <hr>. Tento element sme v predchádzajúcej časti zaradili medzi prezentačné s odporúčaním, nahradiť ho CSS. HTML5 mu prisudzuje sémantický význam – tematický zlom na úrovni odseku.

[3] Pre zobrazenie stránky v textovom režime môžeme využiť niektorý z online nástrojov, napr.: https://www.textise.net/.

Rámcové rozvrhnutie obsahu stránky

O rámcoch sme ešte nehovorili, spomenuli sme ich len okrajovo ako jednu z možností DTD (X)HTML dokumentu. HTML5 rámce už nepodporuje vôbec. Pri použití rámcov potrebujeme jeden (X)HTML dokument na rozvrhnutie obsahu – rozdelenie okna prehliadača. Definícia rámcov v tomto dokumente spôsobí rozdelenie okna prehliadača na niekoľko výrezov. Do každého výrezu sa potom nahrá príslušný dokument. Definícia rámcového (X)HTML dokumentu pre náš príklad vyzerá nasledovne:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="sk" lang="sk">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2" />
<title>Rámcové rozloženie</title>
</head>
<frameset rows="60,*,60" >
  <frame src="hlavicka.html" name="hlavicka" title="Hlavička stránky" />
  <frameset cols="150,*,150">
    <frame src="menu.html" name="menu" title="Hlavné menu stránky" />
    <frame src="obsah.html" name="obsah" title="Obsah stránky" />
    <frame src="odkazy.html" name="odkazy" title="Odkazy na iné weby" />
  </frameset>
  <frame src="paticka.html" name="pata" title="Päťička stránky" />
  <noframes>
    <body>
      Váš prehliadač nepodporuje rámce, pokračujte <a href="menu.html"
title="Hlavné menu">hlavným menu</a>.
    </body>
  </noframes>
 </frameset>
</html>
```

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/ramce.html]

Výhody:

- jednoducho sa vytvárajú štruktúrované weby,
- nie je potrebné opätovne načítavať časti webu, ktoré sa opakujú (napr. menu, hlavičku, ...), každá z týchto častí je uložená v samostatnom dokumente a pri aktivovaní odkazu sa aktualizuje len obsah jedného z rámcov,
- aj keď rolujeme na stránke (v rámci), navigácia je stále na rovnakom mieste,
- pokiaľ to nezakážeme, používateľ má možnosť prispôsobiť si veľkosť rámcov,
- ľahko sa definuje rozvrhnutie obsahu webovej stránky.

Nevýhody:

- z dnešného pohľadu je to zastaraná technológia, to však neznamená že nepoužiteľná,
- problémy s tlačou dokumentu (nemusí byť jasné, či sa má tlačiť obsah obrazovky alebo konkrétneho rámca),
- ak sa rámce navrhnú zle, stránky budú neprístupné,
- textové prehliadače (príp. mobilné zariadenia) rámce nezobrazia, rámcové rozvrhnutie obsahu webovej stránky vyzerá v textovom prehliadači LYNX asi takto: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/ramce.php,

- adresa v URL riadku sa nemení, nevieme teda priamo lokalizovať konkrétny dokument (napr. pre uloženie do obľúbených položiek),
- niektoré indexovacie stroje nevedia indexovať stránky s rámcami (moderné indexovacie stroje si s týmto poradia),
- ak je výsledkom vyhľadávania (v nejakom vyhľadávacom nástroji) odkaz na konkrétnu stránku, po aktivovaní odkazu prichádzame o okolitý obsah a strácame kontext stránky (nie je napr. prístupné menu),
- rámce sa nemôžu prekrývať,
- v HTML5 už nepodporované.

Všimnime si, že telo dokumentu neobsahuje časť `<body>`. Namiesto neho je použitá definícia rámcov pomocou párového elementu `<frameset>`. Atribút `rows`, resp. `cols` určuje, na koľko riadkov, resp. stĺpcov a s akou veľkosťou sa okno prehliadača rozdelí. Okno môžeme súčasne rozdeliť horizontálne aj vertikálne, použijeme súčasne atribúty `rows` a `cols`. Výšku riadku, resp. šírku stĺpca zadávame v pixeloch, v percentách alebo pomerom k voľnému miestu.

Napr.: `<frameset cols="25%, 150, 2*, *">` rozdelí okno prehliadača na štyri stĺpce. Prvý stĺpec má šírku 25 % šírky okna, druhý 150 pixelov. Šírky tretieho a štvrtého stĺpca sa nastavia v pomere 2:1 zo zvyšnej šírky okna.

Pomocou elementu `<frame>` definujeme vlastnosti jednotlivých výrezov okna prehliadača. Použiť môžeme atribúty:

- *name:* `<frame name="meno_okna" />`
  definujeme meno okna, toto meno použijeme ako hodnotu atribútu `target` elementu `a`, ak potrebujeme definovať, v ktorom okne sa má odkaz otvoriť,
- *longdesc:* `<frame longdesc="popis rámca" />`
  popisujeme obsah rámca, pri krátkych popisoch môžeme využiť aj atribút `title`,
- *src:* `<frame src="url dokumentu" />`
  definujeme URL dokumentu, ktorý sa má nahrať do daného rámca,
- *noresize:* `<frame noresize="noresize" />`
  zakáže zmenu veľkosti rámca používateľom,
- scrolling: `<frame scrolling="auto|yes|no" />`
  automatické, vždy zobrazené, vždy nezobrazené rolovacie lišty daného rámca,
- frameborder: `<frame frameborder="0|1" />`
  neviditeľná, viditeľná hranica medzi rámcami,
- marginwidth: `<frame marginwidth="počet pixelov" />`
  definujeme vzdialenosť obsahu rámca od jeho ľavého a pravého okraja,
- marginheight: `<frame marginheight="počet pixelov" />`
  definujeme vzdialenosť obsahu rámca od jeho horného a spodného okraja.

Namiesto elementu `<frame>` môžeme použiť ďalšiu definíciu `<frameset>` a rozdeliť tak daný riadok resp. stĺpec na ďalšie časti.

Súčasťou hlavnej definície `<frameset>` by mala by časť `<noframes>`. Obsah tohto párového elementu sa zobrazí, ak prehliadač používateľa nepodporuje rámce.

Tabuľkové rozvrhnutie obsahu stránky

V čase, keď CSS ešte neposkytovalo možnosti rozvrhnutia obsahu a podpora CSS v prehliadačoch bola nízka, prišli dizajnéri s myšlienkou využiť (zneužiť) na rozvrhnutie obsahu tabuľky. Tabuľky boli pôvodne určené na prezentáciu tabuľkových dát, ale dali sa využiť aj na rozvrhnutie obsahu. Využitie tabuliek k tomuto účelu sa považuje za jeden z najväčších hackov v oblasti web dizajnu. Nasledujúci dokument schválne nepoužíva žiadne CSS pre prezentáciu. Využívajú sa prezentačné atribúty elementov. HTML5 už nepodporuje tabuľkový layout.

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/tabulka.html]

Výhody:

- jednoducho sa vytvárajú štruktúrované weby,
- tabuľky sú veľmi staré, takže aj staršie prehliadače ich zobrazia správne,
- ponúkajú možnosti ako napr. zarovnanie obsahu bunky vertikálne (na stred), premenlivá a pritom rovnaká výška buniek v riadku.

Nevýhody:

- tabuľky sú určené na zobrazenie tabuľkových dát, nie na rozvrhnutie obsahu, v krajných prípadoch je však „dovolené“ ich takto používať,
- komplikované tabuľky nie sú prístupné,
- textové prehliadače tabuľky nezobrazia, text je prevedený do lineárnej štruktúry, ak je tabuľka navrhnutá nesprávne, výsledný text bude ťažko čitateľný, tabuľkové rozvrhnutie obsahu webovej stránky vyzerá v textovom prehliadači LYNX asi takto: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/tabulka.php,
- tabuľka sa zobrazí (v niektorých prehliadačoch) až potom, keď sa načíta jej celý obsah,
- dnes už nevhodná technológia, pretože ju vieme nahradiť technológiou na to určenou (CSS),
- zobrazenie komplikovaných (vnorených) tabuliek je náročné na výkon počítača,
- nie je možné oddeliť obsah dokumentu od jeho vzhľadu,
- pri načítavaní iných stránok webu sa opakovane načítavajú rovnaké časti.

CSS rozvrhnutie obsahu stránky

V moderných weboch by sa mali na rozvrhnutie obsahu webovej stránky používať výhradne CSS. Schéma rozvrhnutia a techniky obtekania ponúkajú takmer neobmedzené možnosti.

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/css.html]

```
<style type="text/css">
    #kontajner {
        position: absolute;
        width: 798px; /*celkova sirka ramceka bude 798 + hranica = 800px*/
        left: 50%; /*nastavíme pozíciu ľavého okraja elementu*/
        margin: 0;
        margin-left: -400px; /*nastavíme ľavé vonkajšie odsadenie*/
        padding: 0;
        border-style: solid;
        border-width: 1px; /*šírka hranice sa do rozmeru elementu nezapočítava*/
    }

    #hlavicka {
        border-bottom: solid;
        border-width: 1px;
    }

    #menu {
        width: 18%;
        float: left; /*nastavime obtekanie zlava*/
    }

    #obsah {
        width: 63%;
        float: left; /*nastavime obtekanie zlava*/
        border-left: solid;
        border-left-width: 1px;
        border-right: solid;
        border-right-width: 1px;
    }

    #odkazy {
        width: 18%;
        float: left; /*nastavime obtekanie zlava*/
    }

    #pata {
        width: 100%;
        border-top: solid;
        border-width: 1px;
        clear: left; /*zrušíme obtekanie zlava*/
    }
</style>
```

Novšie verzie prehliadačov už správne interpretujú hodnotu `auto` atribútu `margin`. Zarovnať obálku horizontálne na stred stránky je možné aj takto:

```
#kontajner {
         width: 798px;
         margin: auto;
         padding:0;
         border-style:solid;
         border-width:1px;
      }
```

V tomto prípade si prehliadač vonkajšie odsadenie dopočíta sám.

Časti webovej stránky (štruktúra stránky) sme vložili do kontajnera „`kontajner`“. Ďalšie časti sú rozvrhnuté vzhľadom na kontajner, do ktorého sú vložené. Na označenie vlastnosti kontajnerov sme použili identifikátory, keďže každý kontajner sa na stránke nachádza len raz.

Rovnaký efekt môžeme dosiahnuť aj v HTML5. Tu je však rozumné použiť nové sémantické elementy ktoré HTML5 prináša.

[ukážka:https://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/css_html5.html]

Výhody:

- oddelenie obsahu dokumentu od jeho formátovania,
- možnosť vytvárať komplikované (pokročilé) rozvrhnutie obsahu webovej stránky,
- moderný web,
- oproti tabuľkovému dizajnu sa web zobrazuje rýchlejšie, má kratší, štruktúrovanejší a prehľadnejší kód,
- web si zachováva logiku aj v textových prehliadačoch, CSS rozvrhnutie obsahu webovej stránky vyzerá v textovom prehliadači LYNX asi takto: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/css.php,
- možnosť prednostného načítania určitých častí stránky.

Nevýhody:

- je potrebné dobre rozumieť schéme CSS rozvrhnutia a technikám obtekania,
- niektoré prehliadače interpretujú CSS odlišne,
- ťažšia orientácia v CSS kóde,
- pri načítavaní iných stránok webu sa opakovane načítavajú rovnaké časti.

Záver

V každom príklade sme si ukázali, ako vyzerá webová stránka v textovom prehliadači LYNX. Nebolo to ani tak kvôli tomu, že by sme brali ohľad na používateľov textových prehliadačov. Išlo o to, že podobným spôsobom je webová stránka sprostredkovaná napr. nevidomým. Ich čítačka totiž nedokáže interpretovať vizuálnu stránku webu, dokáže interpretovať len textové informácie.

Ukázali sme si ako vytvoriť layout pre päť bežných častí webovej stránky. V praxi sa však môžeme stretnúť aj s inými časťami stránky. Ktoré časti a ako použiť však záleží od konkrétnej situácie.

2. Responzívny dizajn

Bolo by zrejme chybou domnievať sa, že každý návštevník nášho webu má monitor s rozlíšením aspoň 1024×768 pixelov a navrhnúť rozvrhnutie obsahu pre takéto minimálne rozlíšenie.

Ak sa pokúsime meniť veľkosť okna v predchádzajúcich ukážkach zistíme, že pri menších šírkach (<800px) sa obsah stránky do okna nevojde a je potrebné rolovať aj v horizontálnom smere. Takéto správanie nie je žiadúce a mali by sme mu zabrániť.

Počet návštevníkov, ktorí používajú tablet, smartfon, iphone či iné „mini“ zariadenie neustále rastie[4]. Problém nízkeho rozlíšenia sa tak z minulosti opäť vracia. Navyše sa k nemu pridali rôzne pomery strán zobrazovacích zariadení, či orientácia zariadenia. Otázka znie, ako vyriešiť problém, aby sa obsah stránky „vošiel“ do okna prehliadača? Minimálne v horizontálnom smere.

Existuje niekoľko postupov, napr.

- vytvoriť verziu webu pre každé (najčastejšie používané) rozlíšenie displeja (podobne ako v minulosti existovali verzie webu pre mobilné telefóny, resp. verzie webov s diakritikou a bez nej),
    - udržiavať niekoľko verzií s tým istým obsahom je prakticky nemožné, navyše stránky s rýchlo meniacim sa obsahom typu blog, diskusia, stránka na sociálnej sieti a pod. by sa takto vytvárať nedali (zatiaľ neuvažujeme o skriptovaní),
- generovať stránky podľa rozlíšenia displeja zariadenia,
    - pomocou skriptov na strane klienta vieme zistiť rozlíšenie displeja a potom skript na strane servera vygeneruje príslušné rozvrhnutie obsahu, tu sa však spoliehame na skriptovanie na strane klienta, ktoré je plne v jeho moci a prípadne môže byť aj vypnuté,
- systém, ktorý automaticky vyhodnocuje parametre zariadenia a následne mu prispôsobuje dizajn a zobrazenie webovej stránky,
    - Media Queries - táto možnosť sa tu objavila len nedávno s príchodom CSS 3 a vlastnosti Media Queries, túto technológiu podporujú všetky aktuálne verzie prehliadačov (http://caniuse.com/css-mediaqueries),
    - Responsive pixel - vďaka CSS vlastnostiam vw/vh/vmin/vmax vieme prirodzene škálovať veľkosť objektov, využitím funkcie calc() dokážeme veľkosti objektov počítať dynamicky,
- kombinácia predchádzajúcich prístupov.

Prístup, kde systém automaticky prispôsobuje vzhľad webu parametrom zobrazovacieho zariadenia sa nazýva **responzívny dizajn.** Responzívny dizajn kladie zvýšené nároky pri návrhu webu, pretože musíme navrhovať dizajn pre rôzne zariadenia. Vieme však dokonale oddeliť dizajn od obsahu (používame CSS) a nemusíme vytvárať rôzne verzie (X)HTML pre rôzne zariadenia. Navyše môžeme zvýrazniť tie prvky webu, ktoré sú dôležité pre návštevníka používajúceho konkrétne zariadenie.

Pri návrhu responzívneho dizajnu môžeme postupovať dvoma spôsobmi.

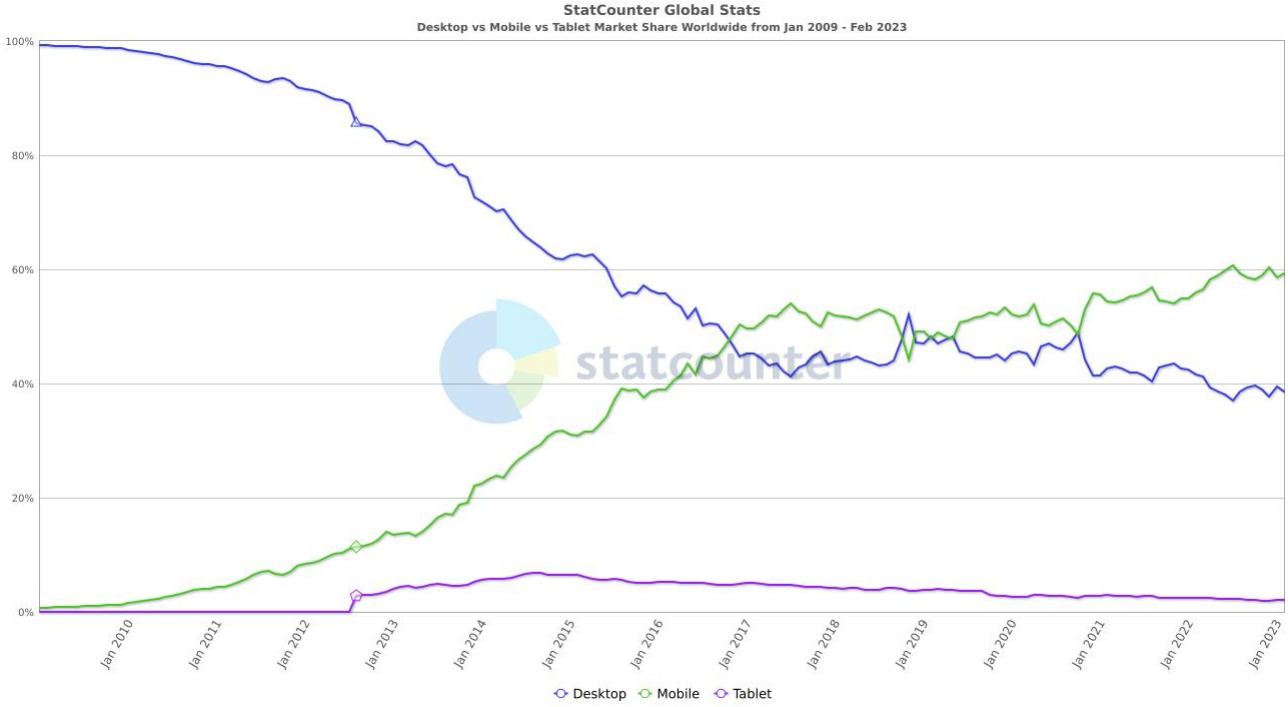
---

[4] Jednu zo štatistík nájdeme na https://gs.statcounter.com/screen-resolution-stats.

Prvý prístup (mobile first) spočíva v tom, že ako prvý navrhneme dizajn pre najmenšie zariadenia (napr. mobilný telefón) do ktorého zahrnieme základnú funkcionalitu a vlastnosti. Potom postupne navrhujeme dizajn pre väčšie zariadenia, pridávame funkcionalitu, interakciu, efekty a pokročilejšie riešenia, ktoré je možné implementovať na väčších a komplexnejších zariadeniach.

Druhý prístup (mobile end) je opačný. Navrhujeme komplexný dizajn od najzložitejšieho (napr. pre desktop s veľkým displejom) zariadenia s množstvom funkcionality a efektov. Potom postupne redukujeme funkcie a efekty a prispôsobujeme dizajn pre menšie a jednoduchšie zariadenia.

Aj keď na prvý pohľad vyzerajú byť obidva princípy rovnocenné, „mobile first“ prístup je výhodnejší. Ak vývojári použijú prístup „mobile end“, majú tendenciu pokročilé prvky dizajnu prenášať aj na nižšie zariadenia. To sa im pravdepodobne nepodarí a tak sa budú musieť vzdať množstva nápadov a vytvárať namiesto nich „rýchle“ riešenia. Koncová verzia (mobile) tak bude pôsobiť skôr ako nedokončený, nevyladený produkt. Preto sa dnes väčšina dizajnérov prikláňa k prístupu „mobile first“. Navyše štatistiky ukazujú, že mobilné zariadenia sú rovnako často požívané ako prístup z desktopu. Dokonca od 11/2016 je mobilný prístup využívanejší viac než prístup z desktopu.



Obrázok 1 Zdroj:
https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-202302

Responzívny dizajn využívajúci Media Queries

Nasledujúci príklad demonštruje jednoduchú verziu webovej stránky, ktorá používa responzívny dizajn.

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responzivny_dizajn.html]

```
<link rel="stylesheet" type="text/css"  href="styly.css" />
<link rel="stylesheet" type="text/css" media="screen and (max-width:539px)"
href="maly.css" />
<link rel="stylesheet" type="text/css" media="screen and (min-width:540px) and (max-
width:959px)" href="stredny.css" />
<link rel="stylesheet" type="text/css" media="screen and (min-width:960px)"
href="velky.css" />
```

K dispozícii sú štyri rôzne CSS súbory[5].

Súbor **styly.css** definuje vzhľad jednotlivých časti webu tak, aby sme ich vizuálne ľahko identifikovali. Na responzivitu tieto vlastnosti nemajú vplyv.

styly.css

```
#kontajner {
    border: 1px solid #000;
    color: #000000;
    background-color: #CCCCCC;
}

#menu {
    background-color: #666;
    text-align: center;
    color: #000000;
}

#menu a {
    padding: 1em;
    margin: 1em;
    background-color: #FFF;
    color: #000000;
}

#obsah {
    background-color: #DDD;
    color: #000000;
    padding: 1em;
    box-sizing: border-box;
}

#reklama {
    padding: 1em;
    box-sizing: border-box;
}
```

Ďalšie štýlopisy definujú rozmiestnenie jednotlivých častí webu pre rôzne šírky zobrazovacieho zariadenia. Každý z nich sa použije pri zobrazení na obrazovke pri konkrétnej šírke obrazovky (presnejšie povedané šírke okna prehliadača). Jednotlivé css

[5] V princípe nemusíme mať štyri štýlopisy. Celú definíciu štýlov môžeme vložiť de jedného štýlopisu. Štýlopis je rozdelený len kvôli prehľadnosti a lepšej manipulácii (napr. jednoducho sa dá niektorý z nich „nepoužiť“ pri testovaní).

súbory vyzerajú nasledovne (prázdne definície je možné odstrániť, tu sme ich ponechali len kvôli možnosti budúcich úprav):

maly.css

```
.stredny, .velky {
    display: none;
}

#kontajner {
}

#menu {
}

#menu a {
    display: inline-block;
    width: 80%;
}

#obsah {
}

#reklama {
}
```

stredny.css

```
.maly, .velky {
    display: none;
}

#kontajner {
}

#menu {
}

#menu a {
    display: inline-block;
    width: 30%;
}

#obsah {
    text-align: justify;
}

#reklama {
}
```

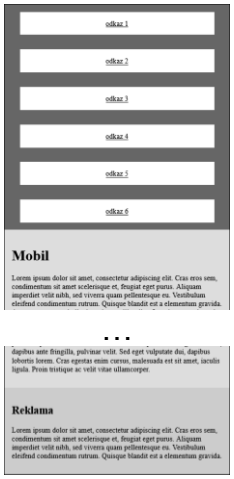
velky.css

```
.stredny, .maly {
    display: none;
}

#kontajner {
    position: absolute;
    width: 950px;
    left: 50%;
```

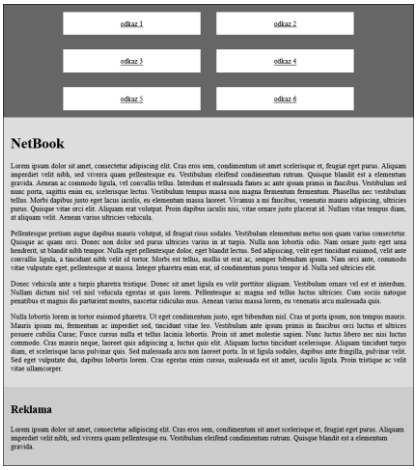
```
    margin-left: -475px;
}

#menu {
}

#menu a {
    display: inline-block;
}

#obsah {
    float: left;
    width: 700px;
    position: relative;
    text-align: justify;
}

#reklama {
    float: left;
    width: 250px;
}
```

Prvá definícia v predchádzajúcich štýlopisoch zabezpečí skrytie elementov (X)HTML dokumentu. V tomto prípade ide o nadpisy, ktoré zobrazujú, pre koho je dané rozlíšenie vhodné.

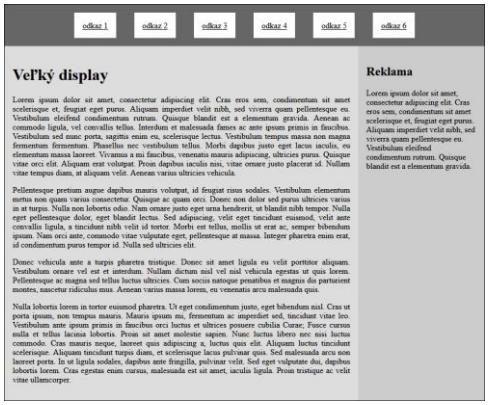
Zvyšná časť je vždy prispôsobená konkrétnemu rozlíšeniu. Podľa šírky sa mení rozloženie obsahu, priestor pre reklamu je presunutý a položky v menu sú prispôsobené šírke okna.



dizajn pre mobil



dizajn pre netbook



dizajn pre veľký displej

Na testovanie responzívneho dizajnu môžete využiť online testovač:
https://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/test_responzivny_dizajn.php

Media Queries ponúkajú aj ďalšie možnosti na identifikáciu parametrov zariadenia:

- `min-width, max-width` – minimálna, maximálna šírka zobrazovacej oblasti zariadenia (okna prehliadača),
- `min-height, max-height` - minimálna, maximálna výška zobrazovacej oblasti zariadenia (okna prehliadača),
- `orientation` – orientácia zariadenia (portrait | landscape),

- `aspect-ratio` – pomer šírky a výšky zobrazovacieho zariadenia,
- `color, min-color, max-color` – farebná hĺbka zariadenia,
- `color-index, min-color-index, max-color-index` – počet farieb zariadenia.

Ďalšie parametre môžeme nájsť na v špecifikácii W3C (http://www.w3.org/TR/css3-mediaqueries/).

Posledným krokom by malo byť testovanie, či sa stránka správa tak, ako si predstavujeme. Okrem klasickej manipulácie s prehliadačom (zmena veľkosti okna) môžeme využiť aj on-line simulátory rôznych zariadení, napr. http://www.responsinator.com/ alebo https://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/test_responzivny_dizajn.php.

Responzivita pomocou Responsive pixel

Pri tomto prístupe berieme do úvahy fakt, že veľké zariadenia (obrazovky) sú pri používaní vzdialené od očí používateľa viac ako menšie zariadenia. Objekty na väčších obrazovkách by tak mali byť väčšie než objekty na menších obrazovkách. Najlepšie si to asi predstavíme na veľkosti písma.

[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responsive_pixel.html]

```
    <style>
        #kontajner {
            max-width: 798px;
            border: 1px solid #000;
            background-color: #CCC;
            margin: auto;
        }
        h1 {
            font-size: min(5vw, 3em);
        }
        p {
            font-size: calc(0.8em + 2vw)
        }
    </style>
</head>
<body>
<div id="kontajner">
    <h1>Nadpis úrovne 1</h1>
    <p>Toto je nejaký text v odseku... </p>
</div>
```

Všimnime si, ako sa nastaví veľkosť písma.

Pri nadpisoch je to 5% šírky obrazovky, ale zároveň nie viac ako trojnásobok aktuálnej veľkosti písma (a samozrejme aj naopak).

Veľkosť textu v odseku sa priebežne vypočítava[6] ako 0,8 aktuálnej veľkosti písma + 2% šírky obrazovky.

[6] V CSS sa dajú využívať aj iné funkcie. Viac napr. na https://www.w3schools.com/cssref/css_functions.asp.

Responzívny dizajn využitím frameworku Bootstrap[7]

Vytvárať CSS pre responzívny dizajn môže byť (a aj je) časovo dosť náročné. Túto cestu si môžeme uľahčiť využitím niektorého z frameworkov (napr. Bootstrap - http://getbootstrap.com/, Foundation - http://foundation.zurb.com/) alebo jednoduchší W3.CSS - https://www.w3schools.com/w3css/). Zjednodušene povedané, layout framework je pomôcka, ktorá nám pre poukladané prázdne okienka pomôže vytvoriť responzívny dizajn.

Tvorcami frameworku Bootstrap sú páni Mark Otto a Jacob Thornton. Vytvorili si ho pre vlastnú potrebu počas vývoja sociálnej siete Twitter. V nasledujúcom texte sa bližšie pozrieme na jednu z výhod jeho použitia. Na návrh responzívneho layoutu.

Samotný framework sa dá stiahnuť zo stránky http://getbootstrap.com/. Archív stačí dekomprimovať a rozbaliť do priečinka, kde máme umiestnený web. Po rozbalení dostaneme nasledujúcu štruktúru priečinkov[8]:

- css
- js

Pre našu potrebu bude užitočný najmä súbor css/bootstrap.css. Súbor css/bootstrap.min.css je jeho minimalizovanou verziou (neobsahuje komentáre, biele znaky a pod.).

Prepojenie s existujúcim (X)HTML dokumentom je už jednoduché.

```
<link rel="stylesheet" href="css/bootstrap.css" />
```

Samotný framework nie je potrebné stiahnuť ako lokálnu verziu. Prilinkovať si môžeme online CDN verziu (Content Delivery Network)[9]. Podobne to robia viaceré weby. Ak teda používateľ narazí na náš web, jeho prehliadač nemusí nanovo sťahovať celý framework, ale použije lokálne uloženú kópiu, ktorú si v minulosti stiahol pri návšteve iného webu. Samotné prilinkovanie online verzie je už jednoduché:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
aFq/bzH65dt+w6FI2ooMVUpc+21e0SRygnTpmBvdBgSdnuTN7QbdgL+OapgHtvPp"
crossorigin="anonymous">
```

Grid systém - mriežka

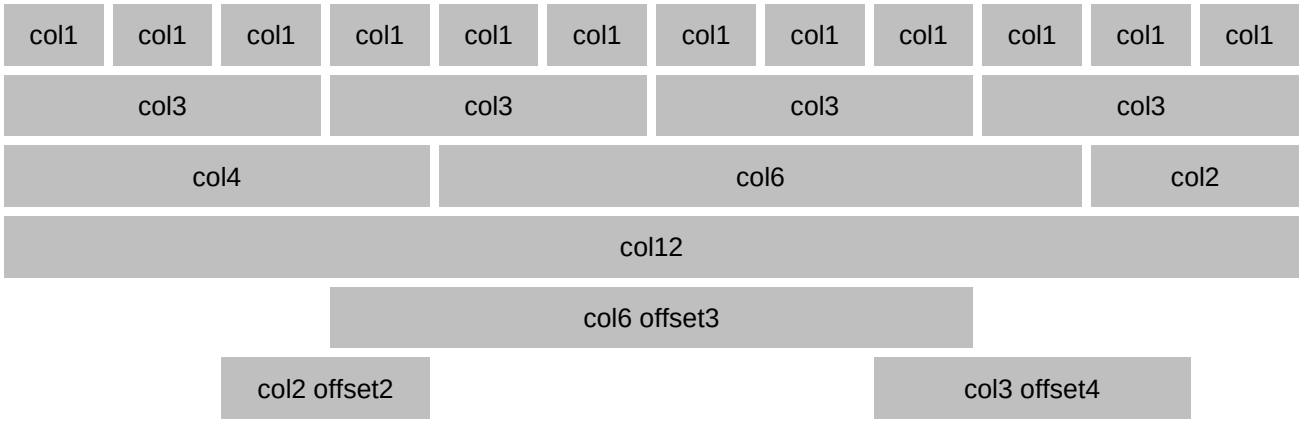
Pre rozvrhnutie obsahu používa Bootstrap mriežku (grid) ktorá rozdeľuje plochu na riadky a každý riadok na 12 stĺpcov. Číslo 12 je zvolené z dôvodu, že ho vieme rozdeliť na 1, 2, 3, 4, 6 alebo 12 častí, čo nám poskytuje dobrú flexibilitu pri vytváraní rozvrhnutia obsahu. Pomocou vytvárania riadkov (row), „spájania“ stĺpcov (col) a „odsadzovania“ stĺpcov (offset) blokov mriežky vieme navrhovať v podstate ľubovoľné rozvrhnutie obsahu stránky.

---

[7] Layout nie je jediná vec, ktorú môžeme prostredníctvom frameworku Bootstrap riešiť. Bootstrap poskytuje podporu pre typografiu, formuláre, tlačidlá, navigáciu a množstvo ďalšieho.

[8] Uvedená štruktúra priečinkov môže byť iná v závislosti na verzii frameworku. V učebnom texte používame verziu 4.1.3

[9] Samotný odkaz a číslo verzie sa môžu meniť. Aktuálne informácie nájdete na https://getbootstrap.com/docs/5.3/getting-started/download/.



Bootstrap používa šesť prefixov na vytváranie stĺpcov pre rôzne veľkosti obrazoviek:

- col extra malé obrazovky (šírka < 576px)
- col-sm malé obrazovky (576px ≤ šírka)
- col-md stredne veľké obrazovky (768px ≤ šírka)
- col-lg veľké obrazovky (992px ≤ šírka)
- col-xl veľmi veľké obrazovky (1200px ≤ šírka)
- col-xxl veľmi veľmi veľké obrazovky (1400px ≤ šírka)

To znamená, že obrazovku vieme rozdeliť niekoľkými spôsobmi a podľa aktuálnej šírky obrazovky sa použije príslušné rozdelenie.

Kontajner

Väčšina riešení začína tým, že definujeme kontajner – element, do ktorého bude vložený celý obsah webovej stránky. Obsah stránky umiestnime do elementu `div` s triedou `container`[10]. Týmto si zabezpečíme, že celý obsah sa „vtlačí“ do kontajnera, ktorého šírka sa odvíja od šírky okna.

- extra malé obrazovky (šírka < 576px) šírka kontajnera None (auto)
- malé obrazovky (576px ≤ šírka) šírka kontajnera 540px
- stredne veľké obrazovky (768px ≤ šírka) šírka kontajnera 720px
- veľké obrazovky (992px ≤ šírka) šírka kontajnera 960px
- veľmi veľké obrazovky (1200px ≤ šírka) šírka kontajnera 1140px
- veľmi veľmi veľké obrazovky (1400px ≤ šírka) šírka kontajnera 1320px

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css"/>
    <title>Responzívny dizajn - Bootstrap framework</title>
</head>
<body>
<div class="container">
```

[10] V HTML5 používajme namiesto sémanticky nevýznamového elementu `div` nové štrukturálne elementy.

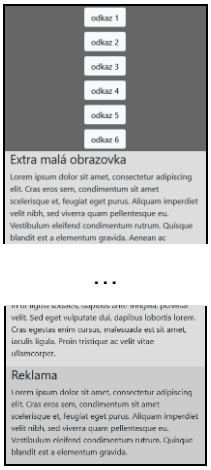
```
</div>
</body>
</html>
```

Riadky a stĺpce v kontajneri

V kontajnerovom div-e potom definujeme samotný obsah stránky rozdelený do riadkov a stĺpcov mriežky Bootstrapu. V prípade, že 12 stĺpcov je pre naše potreby málo, môžeme každý blok opäť rozdeliť na stĺpce (podobne, ako sme rozdelili riadok).Predchádzajúci príklad, teraz ale s využitím frameworku Bootstrap, môže vyzerať nasledovne:

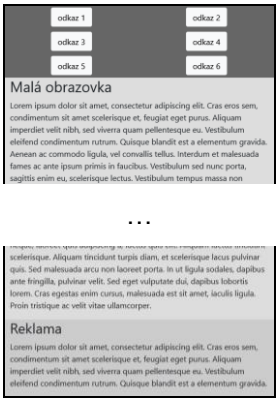
[ukážka: http://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responzivny_dizajn_bootstrap.html]

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css"/>
    <style type="text/css">
        .tlacidlo {margin: 5px; }
        .menu {background-color: #666}
        .obsah {background-color: #DDD}
        .reklama {background-color: #CCC}
    </style>
    <title>Responzívny dizajn - Bootstrap framework</title>
</head>
<body>

<div class="container">
    <div class="row text-center menu">
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class=" btn btn-light tlacidlo">odkaz 1</a>
        </div>
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class="btn btn-light tlacidlo">odkaz 2</a>
        </div>
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class="btn btn-light tlacidlo">odkaz 3</a>
        </div>
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class="btn btn-light tlacidlo">odkaz 4</a>
        </div>
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class="btn btn-light tlacidlo">odkaz 5</a>
        </div>
        <div class="col-12 col-sm-6 col-md-4 col-lg-2">
            <a href="" class="btn btn-light tlacidlo">odkaz 6</a>
        </div>
    </div>
    <div class="row">
        <div class="col-12 col-md-9 col-lg-10 obsah">
            <h1><span class="d-block d-sm-none">Extra malá obrazovka</span>
                <span class="d-none d-sm-block d-md-none">Malá obrazovka</span>
                <span class="d-none d-md-block d-lg-none">Stredne veľká
obrazovka</span>
                <span class="d-none d-lg-block d-xl-none">Veľká obrazovka</span>
                <span class="d-none d-xl-block d-xxl-none">Veľmi veľká obrazovka</span>
                <span class="d-none d-xxl-block">Veľmi veľká obrazovka</span>
            </h1>
            <p>Lorem …</p>
            <p>Pellentesque …</p>
            <p>Donec …</p>
            <p>Nulla …</p>
        </div>
        <div class="co-12 col-md-3 col-lg-2 reklama">
            <h2>Reklama</h2>
            <p>Lorem …</p>
        </div>
    </div>
</div>
</body>
</html>
```



…

mobil



…

tablet na výšku

…

tablet na šírku, notebook

…

veľký display

Všimnime si jednotlivé triedy:

- `row` definuje riadok v mriežke
- `col-*` definuje šírku rámčeka pre extra malé obrazovky, šírka je určená počtom zlúčených stĺpcov mriežky,
- `col-sm-*` definuje šírku rámčeka pre malé obrazovky, šírka je určená počtom zlúčených stĺpcov mriežky,
- `col-md-*` definuje šírku rámčeka pre stredne veľké obrazovky, šírka je určená počtom zlúčených stĺpcov mriežky,
- `col-lg-*` definuje šírku rámčeka pre veľké obrazovky, šírka je určená počtom zlúčených stĺpcov mriežky,
- `btn btn-farby tlacidlo` triedy definujúce vzhľad odkazu ako tlačidlo,
- `d-none, d-{sm,md,lg,xl, xxl}-none` triedy skrývajúce elementy podľa rozlíšenia obrazovky
- `d-block, d-{sm,md,lg,xl, xxl}-block` triedy zobrazujúce elementy podľa rozlíšenia obrazovky

Samozrejme, že vlastnosti definované vo frameworku si môžeme prispôsobiť. Neodporúča sa však zasahovať do samotných súborov Bootstrap-u. Následný prechod na novšiu verziu by bol značne komplikovaný. My sme nové vlastnosti (farby) definovali v hlavičke html dokumentu (odporúčam definíciu presunúť do externého štýlopisu).

Ak sa rozhodneme využiť z frameworku Bootstrap len gridovací (mriežku) nemusíme prilinkovať celý bootstrap. Stačí použiť len bootstrap-grid.css resp. minimalizovanú verziu bootstrap-grid.min.css.

3. Responzívne obrázky

`<img>` je jeden z elementov, ktorý zobrazuje objekt konkrétnej veľkosti (šírka a výška obrázka). V prípade rôznych veľkostí obrazovky by bolo vhodné prispôsobiť aj rozmery zobrazovaného obrázka. Tento cieľ môžeme realizovať niekoľkými spôsobmi.

Automatická zmena veľkosti zobrazenia obrázka

Jednou z možností ako dosiahnuť responzívnosť obrázkov je nastavenie maximálnej šírky obrázka (`max-width`, napr. vzhľadom na rodičovský element, ktorého šírka sa odvíja od šírky okna prehliadača) a automatické dopočítanie jeho výšky (`height`) tak, aby sa zachoval pomer strán obrázka.

[ukážka: http:/di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responzivny_obrazok1.html]

```
<!DOCTYPE html>
<html lang="sk">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
    <title>Responzívny obrázok</title>
    <style>
        .okraj {
            border: #000000 solid 1px;
            margin-bottom: 1em;
        }
        .responzivny {
            max-width: 50%;
            height: auto;
            float: right;
            padding: 1em;
        }
        .zrus_obtekanie {
            clear: both;
        }
    </style>
</head>
<body>

<div class="container okraj">
    <h1>Responzívny obrázok 1</h1>
    <img src="popradske_pleso.jpg" alt="Chata pri Popradskom plese" width="600"
height="450" class="responzivny">
    <p>Lorem ... </p>
    <p>Lorem ... </p>
    <div class="zrus_obtekanie "></div>
</div>
<div class="container okraj">
    <h1>Responzívny obrázok 2</h1>
    <img src="popradske_pleso.jpg" alt="Chata pri Popradskom plese" width="600"
height="450" class="responzivny">
    <p>Lorem ... </p>
    <p>Lorem ... </p>
</div>

</body>
</html>
```

Všimnime si triedu `.responzivny`, ktorá definuje vlastnosti obrázka[11]. Pre samotnú responzívnosť sú dôležité vlastnosti `max-width: 50%` a `height: auto`, ktoré zabezpečia, že obrázok zaberie maximálne 50% šírky rodičovského elementu (alebo svoju maximálnu šírku, podľa toho, čo je menšie) a jeho výška sa automaticky dopočíta. V ukážke sú dva rodičovské kontajnery pre dva obrázky. V druhej ukážke obrázok občas „vylezie“ zo spodnej časti kontajnera. Je to spôsobené tým, že obrázku sme nastavili obtekanie (`float: right`). Zabrániť tomu môžeme tak, že vložíme na koniec kontajnera ešte jeden div, ktorý obtekanie zruší (`<div class="zrus_obtekanie "></div>`). Toto riešenie sme použili pri prvom obrázku.

Rôzne obrázky pre rôzne rozlíšenia zariadenia

V predchádzajúcom riešení sme použili jeden obrázkový súbor, ktorého rozmery zobrazenia sa menili podľa toho, aké veľké je zobrazovacie zariadenie. Nevýhodou tohto riešenia je, že aj pri malých zariadeniach je potrebné preniesť „veľký“ obrázok, ktorý zobrazíme v zmenšenej podobe.

Riešenie tohto problému dosiahneme pomocou elementu `<picture>`. Element `<picture>` prináša vývojárom viac možností pri špecifikácii zdrojov obrázkov. Namiesto jedného obrázku (obrázkového súboru), ktorý podľa potreby zväčšujeme alebo zmenšujeme v okne prehliadača, sprístupnime viac obrázkov (obrázkových súborov) v rôznych veľkostiach. Obrázky sa nemusia líšiť len veľkosťou, ale aj obsahom – napr. rôzne verzie zobrazujú rôzne výrezy z pôvodného obrázka. Element `<picture>` obsahuje dva vnorené elementy: minimálne jeden element `<source>` a práve jeden element `<img>`.

[ukážka: https://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responzivny_obrazok2.html]

```
<picture>
    <source media="(max-width: 210px)" srcset="popradske_pleso_100.png">
    <source media="(max-width: 410px)" srcset="popradske_pleso_200.png">
    <source media="(max-width: 800px)" srcset="popradske_pleso_400.png">
    <img src="popradske_pleso_600.png" alt="Chata pri Popradskom plese">
</picture>
```

Všimnime si jednotlivé elementy `<source>`. Atribút `media` definuje vlastnosti (media query) zobrazovacieho zariadenia. Ak zariadenie vyhovuje požiadavke, použije sa zdroj uvedený v atribúte `srcset`. Prvá splnená požiadavka určuje zdrojový súbor. Na poradí teda záleží. Element `<img>` sa uvádza ako posledný a použije sa, ak sa nepoužije žiadny z predchádzajúcich zdrojov v `<source>`.

Automatický výber obrázka podľa veľkosti zobrazenia

Výber obrázkového súboru môžeme ponechať na rozhodnutí prehliadača. Prehliadaču poskytneme niekoľko zdrojových súborov obrázkov rôznych veľkostí a necháme ho, aby si z nich vybral ten najvhodnejší. Aby to prehliadač dokázal, musíme mu vopred poskytnúť informácie na základe ktorých sa rozhodne.

Poskytneme mu zoznam obrázkov aj s ich šírkou a šírku plochy v okne prehliadača, v ktorej sa obrázok zobrazí.

```
<img srcset="popradske_pleso_100.png 100w,
             popradske_pleso_200.png 200w,
             popradske_pleso_400.png 400w,
```

[11] Framework Bootstrap má pre responzívne obrázky definovanú triedu `.img-fluid` s vlastnosťami: `max-width: 100%; height: auto`.

```
            popradske_pleso_600.png 600w"
     src="popradske_pleso_600.png"
     alt="Chata pri Popradskom plese">
```

V atribúte `srcset`[12] je uvedených niekoľko zdrojových súborov obrázkov aj s ich šírkou. Šírka sa uvádza v pixeloch, ale používa sa jednotka w (presnejšie je to deskriptor) aby bolo jasné, že ide o šírku (**w**idth). Tento rozmer zodpovedá skutočnému rozmeru obrázka v obrázkom súbore. Výška sa neuvádza, pretože sa dopočíta automaticky. Posledný atribút `src` slúži pre prehliadače, ktoré atribút `srcset` nepodporujú.

Ak by mal prehliadač z vyššie uvedenej definície určiť, ktorý obrázok zobraziť, rozhodovať sa môže len na základe šírky zobrazovacieho zariadenia. Pri tomto výbere predpokladá, že obrázok zaberie celú dostupnú šírku zariadenia. To ale často nezodpovedá realite. V skutočnosti obrázok nemusí zaberať celú šírku.

Celý obsah môže byť umiestnený v kontajnery a na veľkých displejoch sa obrázok zobrazí v plnej šírke 600px, na stredných zaberie 40% šírky, na malých displejoch 60% a na tých najmenších ho „natiahneme“ na 100% mínus jeho odsadenie. Ak túto informáciu sprostredkujeme prehliadaču (element `sizes`), bude vedieť vybrať ten najvhodnejší obrázok nie len podľa šírky zobrazovacieho zariadenia ale aj podľa priestoru, v ktorom obrázok zobrazujeme.

[ukážka: https://di.ics.upjs.sk/vyucba/pomocne_materialy/pws/layout/responzivny_obrazok3.html]

```
<img srcset="popradske_pleso_100.png 100w,
             popradske_pleso_200.png 200w,
             popradske_pleso_400.png 400w,
             popradske_pleso_600.png 600w"
     sizes="(max-width: 200px) calc(100vw - 10px),
            (max-width: 400px) 60vw,
            (max-width: 800px) 40vw,
            600px"
     src="popradske_pleso_600.png"
     alt="Chata pri Popradskom plese">
```

[12] Tento element bol zavedený v HTML5.

Použité a ďalšie zdroje

- css Zen Garden: The Beauty in CSS Design
  http://www.csszengarden.com/
- Stichpunkt CSS: Layout ohne Tabellen
  http://www.stichpunkt.de/css/bereiche.html
- intensivstation :: CSS Templates :: Templates
  http://www.intensivstation.ch/templates/
- Dynamic Drive CSS Layouts- Tableless, CSS based templates
  http://www.dynamicdrive.com/style/layouts/
- Open Source Web Design - Download free web design templates
  http://www.oswd.org/
- Media Queries
  http://www.w3.org/TR/css3-mediaqueries/
- Bootstrap - The world's most popular mobile-first and responsive front-end framework
  http://getbootstrap.com/