Webové technológie 5

Aplikácie Webových technológií

Peter Bednár

Responzívne stránky – Media Queries

- Media Queries patria spolu s Flexbox zobrazením medzi základné nástroje návrhu responzívnych stránok, ktoré sa automaticky prispôsobia rôznym zariadeniam

- Media Queries umožňujú definovať pre jednu stránku rôzne kaskádne štýly v závislosti od typu média na ktorom je stránka zobrazená (tzn. pre rôzne veľkosti a orientáciu obrazoviek, zobrazenie pre tlač a pod.)

- Zapisujú sa ako logické podmienky

Media Queries – štruktúra

- Základná štruktúra:

```
@media typ média and vlastnosť 1 and vlastnosť 2 and ... {
   CSS nastavenia
}
```

- Typ média
  - Určuje typ zariadenia pri ktorom sa majú použiť dané CSS nastavenia
  - Najbežnejšie sú `screen` – zobrazenie na obrazovke, `print` – zobrazenie pre tlač, `reader` – zobrazenie pre audio čítačku
  - Je možné zadať aj `all`, alebo negáciu (napr.: `not screen` – všetky okrem obrazovky atď.)

Vlastnosti zobrazovacieho média (1)

- Testované vlastnosti médií zahŕňajú ich absolútnu veľkosť, rozlíšenie, orientáciu, atď.

- `width`, `min-width`, `max-width`,
- `height`, `min-height`, `max-height`
  - Rozmery zobrazovacieho okna (viewportu) v absolútnych jednotkách – px, mm, cm, in (palce)
- `device-width`, `device-min-width`, `device-max-width`,
- `device-height`, `device-min-height`, `device-max-height`
  - Fyzické rozmery obrazovky v absolútnych jednotkách

Vlastnosti zobrazovacieho média (2)

- `resolution`, `min-resolution`, `max-resolution`
  - Fyzické rozlíšenie média v jednotkách dpi (*dot per inch* – počet fyzických bodov zobrazenia na jeden palec)
  - Rôzne prehliadače môžu rozlíšenie vyhodnotiť inak v závislosti na priblížení stránky, ktoré si nastaví používateľ
- `aspect-ratio`, `min-aspect-ratio`, `max-aspect-ratio`
  - Pomer strán medzi šírkou a výškou, napr. 16/9, 4/3 alebo 16/10
- `orientation`
  - Orientácia zobrazovacieho okna na výšku (`portrait`), alebo na šírku (`landscape`)

Media Queries – príklad (1)

- Môžu byť použité pre podmienečné načítanie CSS súboru cez element `<link/>`, napr.:

```
<link href="subor.css" rel="stylesheet"
      media="screen and (max-width: 600px)" />
```

CSS nastavenia zo súboru sa aplikujú iba ak má zobrazovacie okno menej než 600px (vrátane)

- Podobne je možné importovať jeden CSS súbor do druhého príkazom `@import`, napr.:

```
@import url("subor.css") screen and (max-width: 600px)
```

Media Queries – príklad (2)

- Je možné aplikovať tie isté CSS nastavenia pre viacero Media Query výrazov oddelených čiarkou, napr. priamo pri definovaní CSS štýlov:

```
@media print and (min-width: 210mm),
  screen and (min-width: 768px) and (max-width: 1024px)
{ /* aplikované CSS nastavenia */
  body { font-size: 10px; ... }
}
```

CSS nastavenia sa aplikujú iba pri zobrazení pre tlač s minimálnou šírkou strany 210 mm, alebo pri zobrazení na obrazovke so šírkou od 768 px do 1024 px (vrátane)

Animácie v CSS - prechody

- Je možné definovať animáciu pri ktorej sa plynulo mení jedna hodnota CSS vlastnosti na inú (napr. farba, veľkosť, pozícia, atď.)
- Animácia môže byť spustená premiestnením myši na daný element, alebo funkciou JavaScriptu

- Parametre samotnej animácie sa definujú ako CSS vlastnosti

- Vlastnosť `transition-property` určuje, ktoré formátovacie vlastnosti elementu sa budú meniť
  - Zoznam podporovaných vlastností: https://www.w3.org/TR/css3-transitions/#animatable-properties

Vlastnosti prechodov

- `transition-duration`
  - Určuje ako dlho bude animácia trvať v s alebo ms
- `transition-delay`
  - Určuje oneskorenie začiatku prehrávania animácie v s alebo ms po jej aktivovaní
- `transition-timing-function`
  - Určuje priebeh animácie, napr. `linear` – hodnoty sa budú meniť rovnomerne rovnakou rýchlosťou, `ease` – na začiatku rýchla, v strede pomalá a potom zasa rýchla zmena, `ease-in` – začne pomaly a postupne zrýchľuje, `ease-out` – začne rýchlo a postupne spomaľuje atď.

Animácie prechodov – príklad (1)

- Skrátený zápis:

```
transition: property duration delay timing-function
```

```
div {
  width: 100px;
  transition: width 2s
              250ms linear;
}

div:hover {
  width: 400px;
}
```

Počiatočná hodnota

Definícia animácie prechodu

Zmenená hodnota po premiestnení myši nad element

Animácie prechodov – príklad (2)

- Je možné animovať aj zmenu viacerých vlastností naraz uvedením viacerých nastavení

```
div {
  width: 100px;
  font-size: 1em;
  transition-property: width font-size
  transition-duration: 1s 2s
}

div:hover {
  width: 400px;
  font-size: 4em;
}
```

Zložitejšie animácie - zreťazenie prechodov (1)

- Je možné vytvoriť zložitejšie animácie zložené z viacerých prechodov medzi hodnotami
- Základná štruktúra:

```
@keyframes názov {
    from { CSS vlastnosti }
    10%  { CSS vlastnosti }
    ...
    to   { CSS vlastnosti }
}
```

Názov animácie

Hodnoty CSS v danom čase

Rozdelenie priebehu animácie na rámce - časové úseky (začiatok `from = 0%`, koniec `to = 100%`)

Zložitejšie animácie - zreťazenie prechodov (2)

- Priradenie animácie elementu je pomocou vlastnosti `animation-name`, časovanie animácie sa nastavuje podobne ako pri samostatných prechodoch

```
@keyframes farby {
   from { color: white; }
   50%  { color: red;   }
   to   { color: blue;  }
}
div {
  animation-name: farby;
  animation-time: 2s;
  animation-timing-function: ease
}
```

Definícia animácie

Priradenie elementu

Časovanie

Vlastnosti animácií

- `animation-iteration-count`
  - Počet, koľkokrát sa bude animácia opakovať (predstavená hodnota 1), `infinite` – stále opakovať
- `animation-direction`
  - Smer prehrávania vždy od začiatku do konca (`normal`), alebo každý druhý krát prehrávať od konca k začiatku (`alternate` – má význam iba ak je počet opakovaní > 1)
- `animation-fill-mode`
  - `none` – (prednastavená hodnota) na začiatku aj na konci animácie sa vlastnosti nastavania na pôvodné hodnoty (nastavené mimo animácie), `backwards` – na začiatku sa nastavia hodnoty prvého rámca, `forwards` – na konci sa nastavia hodnoty posledného rámca, `both` – kombinácia `backwards`/`forwards`

2D/3D transformácie

- Obdĺžnikovú oblasť do ktorej je zobrazený element je možné na obrazovke transformovať rôznymi funkciami ako napr. rotovať, posunúť, zmeniť pomer strán a pod.
- Vlastnosť `transform`
  - Ako hodnota sa nastavuje transformačná funkcia (resp. postupnosť funkcií)
  - Zoznam funkcií: http://www.css3maker.com/css3-transform.html
- Príklad:
  - `transform: rotate(90deg)` – rotácia o 90°
  - `transform: scale(1,2) translate(10px,-15px)` – dvojnásobné zväčšenie výšky a posunutie o 10px napravo a 15px hore

2D/3D transformácie

- Obdĺžnikovú oblasť je možné transformovať aj v 3D priestore, napr. funkcia `rotateZ()` pre rotáciu okolo osy Z, atď.
- Transformácie je možné kombinovať s animáciami, napr.:

```
div {
  width: 100px;
  height: 100px;
  transition: transform 2s;
}

div:hover {
  transform: rotate(90deg)
             scale(2,2);
}
```

Definovanie animácie prechodu (počiatočná hodnota je bez transformácie)

Konečná transformácia po skončení animácie

Normalizácia štýlov

- Medzi prehliadačmi existujú rozdiely v prednastavených hodnotách CSS vlastností, tzn. tá istá stránka môže byť zobrazená inak

- CSS knižnica normalize
  - Zjednocuje rozdiely rôznych prehliadačov v prednastavených vlastnostiach
  - https://necolas.github.io/normalize.css/
- CSS knižnica reset
  - Resetuje všetky vlastnosti
  - http://meyerweb.com/eric/tools/css/reset/

Normalizácia štýlov - použitie

- Príslušný súbor štýlov pre normalizovanie (normalize.css), alebo reset (reset.css) je potrebné pridať ako prvý odkaz pred ostatnými štýlmi, ktoré chceme na stránke aplikovať
- Príklad:

```
<head>
  <link rel="stylesheet"
        href="css/normalize.css" />
  <link rel="stylesheet"
        href="css/vlastny_styl.css" />
</head>
```

CSS Preprocesory

- Zjednodušujú písanie s zmeny zložitých CSS štýlov
- Automaticky generujú parametrizované CSS štýly, sú určené hlavne na spúšťanie na servery (na klientovi iba počas ladenia štýlov)

- SASS
- http://sass-lang.com/

- LESS
- http://lesscss.org/

LESS - použitie

- Pre generovanie štýlov na strane klienta je potrebné:
  1. Vytvoriť súbor štýlov v LESS a pridať naň odkaz cez element `<link/>`
  2. Pridať na stránku odkaz na skriptový súbor less.js (resp. less-min.js) cez element `<script>`, pri načítaní stránky skript automaticky preloží LESS súbor do CSS nastavení, ktoré sa aplikujú na stránku

```
<head>
  <link rel="stylesheet/less" type="text/css"
        href="styl.less" />
  <script src="less.js"></script>
</head>
```

LESS - príklad

```
@error-color: red;

.set-size(@size) {
  heigth: @size
  & when (@size > 15px) {
    width: @size * 2;
  }
}

div {
  color: @error-color;
 .set-size(10px)
}
p {
 .set-size(20px)
}
```

LESS umožňuje definovať premenné a makrá (tzv. *mixins*) s podmienkami, ktoré sa vyhodnotia a vložia podľa parametrov

Vygenerovaný CSS:

```
div {
  color: red;
  height: 10px;
}
p {
  height: 20px;
  width: 40px;
}
```

Prehľad CSS vlastností (1)

- Vlastnosti pozadia
  - `background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position`
- Vlastnosti písma
  - `font-family`, `font-size`, `font-style`, `font-variant`
- Vlastnosti textu
  - `color`, `text-align`, `vertical-align`, `text-intent`, `text-decoration`, `text-transform`, `white-space`
- Vlastnosti zoznamov
  - `list-style-type`, `list-style-image`

Prehľad CSS vlastností (2)

- Vlastnosti tabuliek
  - `border-collapse`, `border-spacing`, `caption-side`
- Veľkosť okrajov, vlastnosti ohraničenia a zobrazenie kontúry
  - `padding`, `margin`, `border-width` , `border-style`, `border-color`, `border-radius` , `outline-width` , `outline-style`, `outline-color`, `outline-offset`
- Rozmery elementu
  - `width`, `width-min`, `width-max`, `height`, `height-min`, `height-max` , `box-sizing`
- Presah obsahu
  - `overflow`

Prehľad CSS vlastností (3)

- Viditeľnosť, prekrývanie a obtekanie
  - `visibility`, `z-index` , `float` , `clear`
- Umiestnenie
  - `display`, `position`, `left`, `right`, `top`, `bottom`
- Flexbox
  - `flex-direction`, `flex-wrap`, `justify-content`, `align-items`
- Animácie
  - `transition-property`, `transition`, `@keyframes`, `animation-name`, `animation`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode`
- Transformácie
  - `transform`

JavaScript

Jazyk JavaScript (1)

- Skriptovací jazyk – nie je potrebné preložiť program do spustiteľného kódu, interpret priamo načíta a vykoná zdrojový kód
- Bez-typový! – tzn. do jednej premennej môžete priradiť napr. číslo a neskôr reťazec, nekontroluje sa správne priradenie
- Funkcionálny – s funkciou môžete pracovať ako s ľubovoľnou hodnotou, napr. ju môžete predať ako parameter inej funkcii
- Prototypový – môžete definovať objekty, ktoré majú dátové vlastnosti a metódy – funkcie pracujúce nad vlastnosťami

- V súčasnosti štandard pre programovanie skriptov, ktoré sa spúšťajú v prehliadači

Jazyk JavaScript (2)

- Navrhnutý v roku 1995 (prvé použitie v prehliadači v roku 1996 Netscape 2)
- Syntax bol inšpirovaná jazykom C a Java (ale ide o úplne odlišný jazyk než Java!)
- Štandardizovaný ako jazyk ECMAScript organizáciou ECMA International
  - V súčasnosti najviac podporovaná verzia 6
  - Špecifikácia: https://www.ecma-international.org/ecma-262/6.0/

JavaScript - príklad kódu

```
/*
Funkcia sum spočíta prvky
zadaného poľa
*/
function sum(a) {
    var r = 0;
    for (var i = 0; i < a.length; i++) {
        r += a[i];
    }
    return r;
}

var pole = [10, 20, 30];
var x = sum(pole);
var f = sum;
var y = f([30, 20, 10]);
```

Premenné, parametre a návratová hodnota funkcií sa definujú bez typov

Do premennej môžeme priradiť funkciu, ktorú potom môžeme zavolať cez názov premennej

Čísla

- Čísla je možné zapísať desiatkovo, alebo hexadecimálne, napr.:
  `1234`, `123.45`, `123.45e-10`, `0xffaa00` (alebo `0xFFAA00`)
  - Nerozlišuje sa medzi celými číslami a desatinnými (všetky sú reprezentované ako desatinné čísla s dvojitou presnosťou 64 bitov)
- Operátory `+`, `-`, `*`, `/`, `%` (zvyšok po delení), `++`, `--`
- Špeciálne hodnoty
  - `NaN` – ak hodnotu nemožno reprezentovať ako reálne číslo (napr. odmocnina z -1)
  - `Infinity`, `-Infinity` – kladné a záporné nekonečno
- Testovacie funkcie
  - `isNaN(x)` – vráti `true`, ak je hodnota `NaN`, inak `false`
  - `isFinite(x)` – vráti `true` ak je hodnota konečné číslo, tzn. nie je +/- nekonečno, alebo `NaN`

Reťazce

- Reťazce sú reprezentované ako postupnosti znakov v UTF-16 kódovaní
- Zapisujú sa ohraničené " alebo ', špeciálne znaky je potrebné nahradiť sekvenciou \znak podobne ako v C alebo Jave, napr. \', \", \n (nový riadok), \t (tabulátor) atď.
- Reťazce je možné spájať operátorom +, napr.

  `"prvý" + " druhý" = "prvý druhý"`

  `1 + 2 + "tri" = "3tri"` – od prvého reťazca sa výsledok prevedie na reťazec
- Dĺžka reťazca

  `"prvý".length`

  `"".length = 0` – prázdny reťazec

Metódy a funkcie reťazcov

- K reťazcom je možné pristupovať ako k objektom s metódami
- Základné metódy
  - `str.charAt(index)` – vráti znak na danej pozícií (indexovanie je od 0)
  - `str.substring(from, to)` – vráti podreťazec od pozície `from` do `to`, alebo do konca reťazca ak sa vynechá parameter `to`
  - `str.startsWith(s)`, `str.endsWith(s)` – vráti `true`, ak reťazec začína prefixom `s`, alebo končí sufixom `s`
  - `str.toLowerCase()`, `str.toUpperCase()` – vráti nový reťazec prevedením všetkých písmen na malé, alebo veľké
  - `str.trim()` – odstráni prázdne znaky zo začiatku a konca reťazca
- Prevedie reťazca na číslo: `num = parseInt("123");`

Boolovské hodnoty

- Nadobúdajú hodnoty `true` a `false`, používajú sa hlavne pri logických výrazoch v podmienkach
- Logické spojky a operátory
  - `x && y` spojka *a* – ak je `x` `false`, vráť `false` (`y` sa nevyhodnocuje)
  - `x || y` spojka *alebo* – ak je `x` `true`, vráť `true` (`y` sa nevyhodnocuje)
  - `!x` negácia
  - Zložitejšie výrazy je potrebné uzatvárať do zátvoriek aby sa určilo správne poradie vyhodnotenia

Operátory porovnania

- Aritmetické porovnania
  - `<`, `>`, `<=`, `>=` – pred porovnaním sa vykoná typová konverzia, tzn. napr. prázdny reťazec sa prevedie na `0`

    `"" < 2 = true`
  - hocijaké porovnanie s `NaN` je `false`
- Rovnosť a nerovnosť hodnôt
  - `==`, `!=` – pred porovnaním sa vykoná typová konverzia, tzn. napr. `"123" == 123 = true`
  - `===`, `!==` nevykonáva sa žiadna konverzia, hodnoty nie sú rovné ak sa nezhoduje ich typ

Premenné, konštanty, `null` a `undefined`

- Premenné sa vytvárajú kľúčovým slovom `var` a uvedením názvu premennej

```
var x;
```

- Ak premennej nebola priradená žiadna hodnota, jej hodnota bude nedefinovaná, tzn. `x === undefined`
- Premennú je možno priamo inicializovať, napr.:

```
var x = 123;  // typ premennej sa mení podľa
x = "ahoj";   // priradenej hodnoty
```

- Premenným je možné priradiť hodnotu `null`, ktorá má význam "žiadna hodnota", platí `null !== undefined`
- Konštanty je možné definovať slovom `const`, napr.:

```
const PI = 3.141592654;
```