Informatika pro moderní fyziky (8)
HTML a JS - interaktivní dokumenty

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2015/2016
1. prosince 2015

Obsah

- použití cizích API – data v JSON, mashup s GoogleMaps Static API
- základy HTML + ERb

Obsah

Kódování

- znaky v počítači: 1 znak = 1 byte = 256 možností
- cca polovina je “normální text”, zbytek jsou tak trochu speciální znaky
- jsou tam ale jen ‘západní’ znaky, na středoevropské se nedostalo
- co teprv azbuky, japonština, čínština, ...

Kódování

- varianta 1: nahrazovat druhou polovinu znaků tím, co zrovna potřebuju (ěščř...)
- výhoda: nezabírá místo, pořád platí znak=byte, jednoduché řešení
- nevýhoda: pro různé jazyky různá kódování
- další nevýhoda: na leckterý jazyk (neevropský) to naprosto nestačí

Kódování

- varianta 2: přejít na reprezentaci jednoho znaku více byty
- výhoda: podstatně se rozšíří počet znaků, takže není nutné ‘přepínat’
- nevýhoda: zvětšuje velikost textu
- další nevýhoda: míra rozsypanosti čaje při špatné interpretaci se zvyšuje (je možné způsobit i nečitelnost ‘obyčejných’ znaků)

Kódování - obecné problémy

- soubory neobsahují informaci o tom, v jakém kódování jsou, takže je nutné dodávat metadata
- manipulace s kódováním je otravná, obtížná a snadno způsobí problémy
- je nutné synchronizovat/řešit kódování na všech vrstvách aplikace (soubory / databáze / databázový klient / aplikace / server / klient)
- ... pokud existuje 100 různých variant, řeší se to obvykle tím, že se vymyslí nějaká 101. ...

Kódování

- naštěstí to (minimálně v euroamerickém prostředí) už dneska nemusíme řešit, existuje jednotný standard – Unicode (UTF-8)
- starší systémy občas poskytují data v jiných kódováních, je dobré vědět, že pro češtinu se můžete setkat s `Windows/CP1250` a `ISO-9981-2`
- pokud budete chtít pracovat s exotickými jazyky (Afrika, Oceánie), budete asi muset využít UTF-16/32 (naneštěstí opět více variant)
- z historických důvodů je nejednotnost také pro japonská kódování
- to hlavní: používejte UTF-8. Tečka. Nezapomeňte: v editoru, v hlavičce HTML, v hlavičce LaTeXu, všude...

Obsah

JavaScript

- trochu obskurní jazyk bez ladu a skladu
- dá se vkládat do HTML a používá se pro client-side aplikace
- dříve to bylo trochu zlo, dneska se z toho stala
- rozšířenost dalece převyšuje kvalitu

JS frameworky

- vrstva “překrývající” JS a usnadňující základní věci, často zejména manipulaci se stránkou a
- bez nich je JS stěží využitelný
- aktuální standard je jQuery
- množství dalších knihoven a rozšíření

JavaScript v HTML

- event-driven jazyk
- kód se nespouští “jen tak”, ale pouze na základě události
- kliknutí, načtení stránky, uplynutí času, klávesnice ...
- pro nás nejužitečnější `onclick`; např. ...

Schovávání a zobrazování

- každému tagu můžu přiřadit id <img src=”..” id=”image1” >
- v jQuery se potom na příslušný tag odkážu takto:
  `$('#image1')`
- `$('#image1').hide();`
- `<a href="#" onclick="$('#image1').hide();">`
- navíc: je potřeba přidat `return false` na konec handleru
- obdobně `show` a `toggle`
- kousek CSS:
  `<img src="..." style="display:none" >`

JavaScript - jak jQuery nahrát

- je možné se v dokumentu odkázat na jiné JS soubory
- obvykle je to lepší než tam text dokumentu kopírovat... (přehlednost, caching)
- odkaz se vkládá do hlavičky dokumentu (`<head>`)
- `<script src=”jquery-1.4.4.min.js” type=”text/javascript” ></script>`

Úkoly

- doplnit přepínání mezi grafem a tabulkou
- vylepšit z programátorského hlediska (DRY princip a AO/boritá)

Obsah

CSS

- jazyk pro popis vzhledu HTML dokumentu
- v nejjednodušším přístupu definuje styly pro jednotlivé typy elementů
- dále umožňuje definovat tzv. třídy (skupiny elementů, pomocí atributu `class` v HTML) a také styly pro konkrétní elementy (id)
- složitější selekory – vnoření, souslednost atd.

CSS - jednoduchý příklad

- základní selektory: *element*, *.třída* a *#id*
- základní syntaxe *vlastnost: hodnota;*

```
a {
  color: blue;
  text-decoration: none;
  font-weight: bold;
}
#core_map { float:left; }
```

CSS - kam s ním

- přímo k tagu (`<div style="display:none">`) – možná dobré na rychlé ladění/patlání, ale vesměs vždy špatně; jedinou výjimkou je `display:none` pro elementy, které mají být vidět až později, tam to jinak nejde
- v hlavičce (`head`) HTML dokumentu: `<style type="text/css">...</style>`
- v externím souboru (obvykle jediné správné řešení) – pomocí tagu `link` v hlavičce
- my vystačíme se `style` tagem v hlavičcce

Rychlé procvičení

- vezměte si svoje krásné HTML a doplňte do něj trochu toho stylování
- minimum: změnit font (`font-family`), nastavit rozumné velikosti písma a barvy
- zrušit podtrhávání odkazů (`text-decoration`)
- bonus: jiná barva odkazů na přepínání tabulka/graf

Obsah

Zadání dnešní úlohy

- každý den data z 1-9 detektorů (`data/*.csv`)
- detektor má svoji polohu v AZ (VR-1 Vrabec, 8x8 čtvercových pozic) – včetně data je uvedena na prvním řádku CSV souboru
- je potřeba hezky zobrazit na každý den mapu AZ a grafy signálů z detektorů
- viz `html/document.html`

Jak vložit do HTML

- jsou různé metody, jak vložit ze souboru
- my se bez toho v pohodě obejdeme - vložíme přímo (`File.read`)
- v tu chvíli je totiž mj. možné naplácat na SVG objekty javascriptové handlery
- ... tedy na čtverečku můžu mít onclick

Zpracovat data

- pro každý CSV soubor chceme mít graf (gnuplot/png)
- pro každý den mapu (nejdřív obyčejnou, potom klikací, pak třeba s textem)

Další JS chytrosti

- v jQuery už známe `$('#id')`
- ale ve skutečnosti jde použít jakýkoli CSS selektor, takže třeba `$('p')`
- pokročilý CSS selektor – vnoření: `#my_list img` vybere všechny obrázky (`img`) které jsou uvnitř elementu s id `my_list`
- ... použiju v situacích, kdy chci schovat nějakou množinu elementů a pak jeden z nich zobrazit (tj. když mám hromadu obrázků a chci, aby byl vidět jen jeden)

Další CSS chytrosti

- normálně se jednotlivé elementy řadí pod sebe
- můžu místo toho použít tzv. floating, kdy se začnou elementy řadit nalevo nebo napravo
- efekt znáte např. z webových galerií, kde mi fotky vyplní celou šířku okna a jdou po řádcích
- `float:left`
- barvu pozadí nastavím např. `background-color:red` nebo `background-color:#ddddff`

Další SVG chytrosti

- kromě `rect` se bude hodit také `text`
- jako text se zobrazí obsah příslušného elementu
- opět použiju atributy `x`, `y` (levý dolní roh) a můžu přihodit `text-anchor="middle"`, aby to byl dolní prostředek
- pozor, text mi překryje čtvereček, takže budu muset zopakovat onclick!

A to je vše, přátelé!