Žilinská univerzita v Žiline
Elektrotechnická fakulta

28260220131028

ANALÝZA A IMPLEMENTÁCIA MNOŽINY TECHNOLÓGIÍ HTML 5

BAKALÁRSKA PRÁCA

2013
Michal Uhrín

Žilinská univerzita v Žiline
Elektrotechnická fakulta
Katedra riadiacich a informačných systémov

ANALÝZA A IMPLEMENTÁCIA MNOŽINY TECHNOLÓGIÍ HTML 5

BAKALÁRSKA PRÁCA

Študijný program: Automatizácia
Študijný odbor: 5.2.14 Automatizácia
Školiace pracovisko: Žilinská univerzita v Žiline, Elektrotechnická fakulta, Katedra riadiacich a informačných systémov
Vedúci bakalárskej práce: Ing. Peter Holečko, PhD.

Žilina, 2013
Michal Uhrín

KATEDRA RIADIACICH A INFORMAČNÝCH SYSTÉMOV
ELEKTROTECHNICKÁ FAKULTA
ŽILINSKÁ UNIVERZITA V ŽILINE

Akademický rok 2012/2013 Evidenčné číslo: B 28/2013

ZADANIE BAKALÁRSKEJ PRÁCE

Meno študenta: Michal UHRÍN

Študijný odbor: Automatizácia

Téma bakalárskej práce: Analýza a implementácia množiny technológií HTML 5

Pokyny pre vypracovanie bakalárskej práce:

1. Charakteristika jazyka HTML 5, jeho komponentov a súvisiacich technológií.
2. Komplexné porovnanie technológií HTML 4 a HTML 5 z hľadiska funkcionality, výkonu, efektivity, bezpečnosti a ďalších parametrov.
3. Prehľad nových prvkov množiny technológií HTML 5 a ich implementácia.
4. Perspektívy a smer vývoja v oblasti hypertextových značkovacích jazykov.

Vedúci bakalárskej práce: Ing. Peter Holečko, PhD.

Dátum odovzdania bakalárskej práce: 3. 5. 2013

Žilina, 24. 10. 2012

prof. Ing. Juraj Spalek, PhD.
vedúci katedry

Abstrakt

Bakalárska práca sa zameriava na analýzu nového štandardu HTML5 a príbuzných technológií a porovnáva ho s predchádzajúcou verziou HTML 4. Obsahuje potrebné informácie k tvorbe moderných webových dokumentov alebo dynamických stránok pomocou natívnych API. Záverom sa dotýka smerovania hypertextových značkovacích jazykov a zabezpečenia spätnej kompatibility v starších prehliadačoch. Možnosti štandardu demonštruje vytvorená webová aplikácia a prehľad funkcií s výstrižkami kódov použiteľnými v dennej praxi.

Kľúčové slová:

HTML5, Web 2.0, WebStorage, Canvas, API

Žilinská univerzita v Žiline, Elektrotechnická fakulta
Katedra riadiacich a informačných systémov

ANOTAČNÝ ZÁZNAM – BAKALÁRSKA PRÁCA

Meno a priezvisko: Michal Uhrín **Akademický rok:** 2012/2013

Názov práce: Analýza a implementácia množiny technológií HTML 5

Počet strán:	*49*	**Počet obrázkov:**	*7*	**Počet tabuliek:**	*5*
Počet grafov:	*0*	**Počet príloh:**	*5*	**Počet titulov použ. lit.:**	*22*

Anotácia v slovenskom jazyku:

Bakalárska práca analyzuje štandard HTML5, jemu príbuzné technológie a porovnáva ho s predchádzajúcou verziou HTML 4. Obsahuje potrebné informácie k tvorbe moderných webových dokumentov alebo dynamických stránok pomocou natívnych API. Možnosti štandardu demonštruje webová aplikácia a prehľad funkcií s výstrižkami kódov použiteľnými v dennej praxi.

Anotácia v anglickom jazyku:

Bachelor thesis analyzes HTML5 standard with associated technologies and compares it with the previous version of HTML. It contains a knowledge needed for creating a modern web documents and dynamic web pages using native APIs. Possibilities of HTML are demonstrated in a web application and in a structured overview containing code snippets suitable for daily use.

Kľúčové slová: HTML5, Web 2.0, WebStorage, Canvas, API

Vedúci bakalárskej práce: Ing. Peter Holečko, PhD.
Recenzent: Ing. Emília Bubeníková

Dátum odovzdania práce: 3. mája 2013

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

Skratka	Anglický význam	Slovenský význam
API	Application Programming Interface	Aplikačné programovacie rozhranie
CSS	Cascading Style Sheets	Kaskádové štýly
DOM	Document Object Model	Objektový model dokumentu
DTD	Document Type Declaration	Definícia typu dokumentu
HTML	HyperText Markup Language	Hypertextový značkovací jazyk
HTMLWG	HTML Working Group	Pracovná skupina HTML
HTTP	HyperText Transfer Potocol	Hypertextový prenosový protokol
IETF	Internet Engineering Task Force	Komisia techniky Internetu
SGML	Standard Generalized Markup Language	Štandardizovaný zovšeobecnený značkovací jazyk
URL	Uniform Resource Locator	Jednotný vyhľadávač zdrojov
W3C	World Wide Web Consortium	Konzorcium svetovej siete
WHATWG	Web HyperText Application Technology Working Group	Pracovná skupina technológii webových internetových aplikácii
XHTML	Extensible HTML	Rozšíriteľný HTML

ÚVOD

Internet je mocný nástroj. Od svojho vzniku prešiel veľkou evolúciou z pohľadu dostupného obsahu aj technológií. V poslednej dekáde sa prudko rozšíril a stal sa ľahko dostupným informačným prostriedkom v každej domácnosti. Denne ho využívajú milióny ľudí na prezeranie dokumentov, multimediálneho obsahu, písomnú či hlasovú komunikáciu, zábavu a osobnú prezentáciu. Neodmysliteľnou súčasťou webového obsahu je jazyk HTML, ktorý tvorí základ webových dokumentov, respektíve stránok a v súčasnosti veľmi rozšírených webových aplikácií. Jeho vývoj spočiatku kopíroval rozmach internetu, no začiatkom tretieho tisícročia ustal a vyzeralo to, že ho nahradia novšie technológie. Po takmer desaťročnej prestávke začala postupne prenikať na svetlo sveta nová verzia jazyka HTML, ktorá sa snaží vytvoriť pevný základ moderného webu, ktorý stále chýba. S uvoľňovaním konečných verzií pridružených špecifikácií a rastúcou implementáciou prehliadačmi sa jednotlivé funkcie využívajú čoraz častejšie.

Cieľom našej bakalárskej práce je priblížiť nový štandard HTML5, popísať jeho komponenty a ich použitie a následne ho porovnať s predchádzajúcou verziou HTML4. Na začiatku priblížime historický vývoj jazyka HTML, ktorý má napomôcť lepšiemu pochopeniu jeho súčasného smerovania. Ďalej uvedieme základné pojmy úzko súvisiace s HTML. V druhej kapitole popisujeme nové značky a atribúty HTML5. U niektorých tiež porovnávame ich pozmenený význam voči predchádzajúcej verzii. Tretiu kapitolu venujeme implementácii API obsiahnutých v špecifikácii. Štvrtú kapitolu venujeme súčasným možnostiam nasadenia v projektoch, bezpečnosti a smerovaniu štandardu HTML. V praktickej časti sa zameriavame na implementáciu vybraných funkcii HTML5 vytvorením jednoduchej webovej aplikácie a prehľadu základných funkcii, ktorý môže byť použitý pri tvorbe vlastných HTML dokumentov.

Čitateľovi má práca priniesť základný prehľad o HTML5 spolu so znalosťou implementácie niektorých funkcii, ktoré sú k dispozícií už dnes. Takisto má možnosť sa oboznámiť s obmedzeniami použitia, ktoré vyplývajú z nekonzistentného prostredia moderného Webu.

1 PREDSTAVENIE HTML5

HTML5 je aktuálne veľmi skloňované slovíčko, ktoré so sebou nesie určité množstvo dezinformácie. V nasledujúcej kapitole by sme chceli uviesť na pravú mieru, čo to HTML5 vlastne je. Pre lepšie pochopenie súčasnej situácie a postavenia na trhu začneme historickým prehľadom jazyka HTML.

1.1 Historický vývoj HTML

HTML vznikol ako formátovací jazyk dokumentov prenášaných protokolom HTTP v sieti internet. Začiatkom 90.tych rokov ho vytvoril Sir Tim Berners-Lee úpravou medzinárodne známeho SGML, ktorého zápis je členený pomocou značiek. Prepojenia sveta pomocou siete internet za účelom zdieľania vzdialených informácii dostalo názov World Wide Web (ďalej len Web). Na prezeranie dokumentov písaných v HTML slúžili spočiatku terminálové, neskôr grafické programy, nazývané prehliadače. V roku 1993 prehliadač Mosaic rozšíril funkcie HTML o možnosť vkladať do dokumentov obrázky a formulárové polia. Komunita Webu neustále rástla a na usmernenie ďalšieho vývoja bola pod záštitou IETF vytvorená pracovná skupina HTMLWG. Tá ešte v roku 1994, na základe komunitou okomentovanej pracovnej verzie (tzv. draftu), vydala presnú špecifikáciu HTML 2 vo forme DTD. Koncom roka 1994 bolo sformované World Wide Web Consortium (W3C) na podporu rozvoja Webu a vývoj otvorených štandardov, na čele ktorej stál Tim Berners. Konzorcium bolo sponzorované veľkými firmami ako IBM, Microsoft, Netscape či Hewlett-Packard.

Keďže HTML bol, a stále je, otvorený štandard, môže ktokoľvek zainteresovaný pridať návrh na vylepšenie. Pracovná skupina W3C, ktorá prevzala vývoj od IETF nestíhala spracúvať prichádzajúce návrhy. Na pomoc pri tvorbe štandardov vznikla v rámci W3C HTML Editorial Review Board, redakčná rada, zložená z členov vedúcich spoločností trhu. Proces štandardizácie nestíhal držať krok s dopytom, a tak prehliadače okrem oficiálneho jadra špecifikácie, aby ponúkali viac ako konkurencia a uspokojili tvorcov stránok. Prehliadač Internet Explorer spoločnosti Microsoft obsahoval unikátne rozšírenie ActiveX pracujúce s operačným systémom Windows, prehliadač Netscape zase predložil myšlienku rámov, ktoré umožňovali v jednom okne obsiahnuť viac nezávislých dokumentov vo vlastných rámoch. Tieto nekonzistentnosti implementácií komplikovali štandardizáciu HTML 3.

W3C popri HTML pracovalo aj na ďalších štandardoch, ktoré rozširovali možnosti HTML. Kaskádové štýly (CSS) upravovali vzhľad obsahu dokumentov písaných v HTML, podpora JavaScript zase umožňovala vykonávať jednoduché skripty. Vyvrcholením vývoja mala byť verzia HTML 4, respektíve po konečných opravách v roku 1999 verzia HTML 4.01.

Medzitým W3C začalo pracovať na štandarde XHTML1.0, ktorého základ tvoril jazyk XML. Neobsahoval žiadne nové značky oproti HTML 4.01, ale syntaktické pravidlá prevzaté z XML vyžadovali tvoriť bezchybný zápis, aby sa dokument, respektíve stránka vôbec zobrazili v prehliadači. Nasledovníkom tejto verzie je verzia XHTML 2.0, ktorá podľa utopistických predstáv autorov mala byť nezávislá na používaných zariadeniach, primárne určená na členenie obsahu so silným dôrazom na sémantiku kódu. Od základov pozmenené mechanizmy zápisu však boli v porovnaní s predošlými verziami komplikované a spätná kompatibilita s nimi prakticky neexistovala. Pod vplyvom zložitosti nebol štandard prijatý širokou verejnosťou čo malo za následok ukončenie vývojovej pracovnej skupiny XHTML 2 koncom roka 2010.

Spätne v roku 2004, paralelne s vývojom XHTML 2, bola predstavená vízia takzvaného "Web 2.0", založená na prudkom rozvoji webových technológii a aplikácií. Web prestával byť len zmesou textov, obrázkov a stránok prepojených odkazmi. Aplikácie neboli závislé od operačného systému a mohli bežať kdekoľvek, za predpokladu, že to podporoval prehliadač. Na rozdiel od utopického napredovania W3C sa ako alternatíva odrážajúca skutočné potreby dizajnérov sformovala skupinka vývojárov s cieľom pokračovať vo vývoji HTML nazvaná WHAT WG na čele s Ianom Hicksonom. S podporou tvorcov prehliadačov a otvoreným prístupom začali pracovať na špecifikáciách Web Apps 1.0 a Web Forms 2.0 zahŕňajúcich predošlú verziu HTML 4.01. Spolu s rozšíreniami, dnes známych ako Web Storage či Web Sockets, vytvorili balík HTML5. Koncom roka 2006 W3C už tušilo, že cestou XML (XHTML) budúcnosť, aspoň tá blízka, nevedie. Obnovilo HTML WG a adoptovalo špecifikáciu HTML5 od WHAT WG. Prvú pracovnú verziu vydali v Januári 2008.

V súčasnosti na vývoji HTML pracujú organizácie WHAT WG aj W3C. WHATWG od roku 2011 pracujú na verzii označenej jednoducho HTML vo forme "žijúceho štandardu". Konečné slovo nad špecifikáciou má Ian Hickson. Špecifikácia

organizácie W3C naďalej používa číslovanie verzií, pre jednoduchšiu rozlíšiteľnosť pri implementácii. Konečné slovo nad špecifikáciami má v súčasnosti viacero redaktorov, no do roku 2011 ho mal len Ian Hickson.

1.2 Základné pojmy a charakteristiky HTML5

Z historického vývoja je vidieť, že štandard HTML5 je na rozdiel od predchádzajúcej verzie vyvíjaný predovšetkým na tvorbu dynamických a interaktívnych webových aplikácii. Preto možno pod pojmom HTML5 chápať trojicu úzko súvisiacich technológii: HTML, CSS a JavaScript.

Uvádzame niektoré základné princípy, ktoré podľa dokumentu[11] v skratke charakterizujú štandard HTML5.

- *Spätná kompatibilita* - podpora aspoň základných funkcií v starších prehliadačoch, pričom chovanie zastaraných prvkov je naďalej podporované, čo zabezpečí plynulý prechod na nové verzie.
- *Efektivita* - adoptovanie a prípadné vylepšovanie už existujúcich v praxi používaných riešení miesto vytvárania nových, ktoré by nemali využitie, alebo by boli náročné na implementáciu. Správanie prvkov je kompletne definované v špecifikácii aby nevznikali rozdiely v implementácii medzi prehliadačmi.
- *Jednoduchosť* - zjednodušovanie zápisu a odstránenie závislosti na skriptovaní tam, kde je to možné.
- *Univerzálnosť* - oddeľuje štruktúru obsahu od prezentačnej časti (CSS). Takisto nezávislosť na použitej platforme (stolné počítače, mobilné zariadenia) a operačnom prostredí.
- *Otvorenosť* - zachováva maximálnu transparentnosť procesu štandardizácie a ponecháva možnosť komukoľvek prispieť pri vývoji.

1.2.1 HTML

HTML je značkovací jazyk, ktorý vytvára logickú štruktúru obsahu dokumentov prenášaných v sieti Internet pomocou HTTP, ale aj pre lokálne využitie. Logickou štruktúrou sa myslí usporiadanie do blokov, ktorých obsah spolu súvisí, vytvorenie ich hierarchie a prepojenie s inými časťami dokumentu, alebo inými dokumentmi pomocou hypertextových odkazov. Okrem textu môžu html dokumenty obsahovať aj objekty

iného dátového typu. Pre zlepšenie prístupnosti a optimalizácie obsahuje takisto rôzne meta-dáta, určené pre počítačové programy, ktoré bližšie popisujú časti dokumentov.

1.2.2 CSS

Kaskádové štýly predstavujú jednoduchý nástroj na úpravu vzhľadu HTML a XML dokumentov. Najnovšia verzia CSS3 okrem možností meniť farby, písmo a rozloženie obsahu pridáva doteraz chýbajúce možnosti animácie a prechodov. Do html dokumentu sa pridávajú z externého súboru s príponou *.css, alebo môžu byť obsiahnuté v hlavičke dokumentu vo vnútri elementu `<style>`.

1.2.3 JavaScript

Skript je programový kód, ktorý pred spustením nemusí byť kompilovaný alebo inak predspracovaný. V kontexte Webu sa zvyčajne jedná o kód napísaný v jazyku JavaScript, ktorý je vykonávaný v prehliadači po načítaní stránky alebo pri preddefinovanej situácii. Skripty vytvárajú dynamické stránky, ktoré využívajú informácie relevantné pre užívateľa - zemepisná poloha, osobné nastavenia, orientácia zariadenia a podobne. Interaktívne chovanie pretvára stránky na webové aplikácie, ktoré sa chovajú ako natívne aplikácie v prostredí operačného systému.

JavaScript je objektový skriptovací jazyk vyvinutý spoločnosťou Netscape ako nadstavba štandardného skriptovacieho jazyka ECMAScript zabudovaný v každom modernom webovom prehliadači. Kód je priamo vkladaný do kódu stránok v elementoch `<script>` alebo pripájaný súbormi s príponou *.js.

1.2.4 DOM model

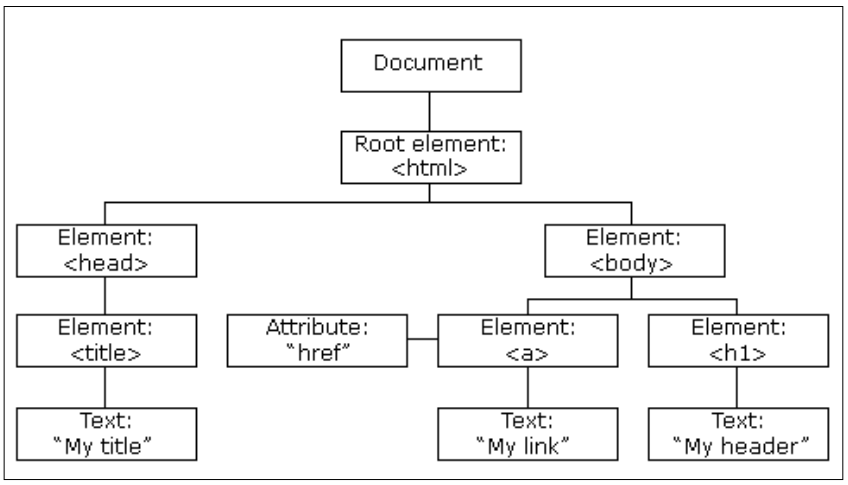
DOM je objektovo orientovaná reprezentácia dokumentu HTML alebo XML. Vzniká ako výstup parsovania HTML dokumentu v prehliadači. Pre prácu s objektmi v strome DOM je definované rozhranie API popísané vo vlastnej špecifikácii.

DOM model obsahuje uzly, predstavujúce časti dokumentu, hierarchicky členené v stromovej štruktúre. Nadradeným objektom stromu je objekt *window* prehliadača, ktorý je najvyšším objektom v BOM (Browser Object Model). Hierarchicky najvyšším objektom DOM je *document*, ktorý reprezentuje samotný obsah stránky. Ďalším objektom je koreňový element, v prípade HTML je to element `<html>` ďalej obsahujúci `<head>` a `<body>`. Takýmto spôsobom je celý dokument opísaný pomocou uzlov. Na

základe BOM a DOM prehliadač vykreslí obsah dokumentu pre užívateľa. Zmenou v DOM sa teda vykonávajú zmeny v samotnom dokumente, čo umožňuje upravovať, pridávať a odstraňovať objekty a dynamicky meniť obsah.

```
<html>
  <head>
    <title>My Title</title>
  </head>
  <body>
    <a href="">My Link</a>
    <h1>My header</h1>
  </body>
 </html>
```

Grafická reprezentácia stromu DOM predchádzajúceho kódu vyzerá nasledovne:



Obr. 1.1 Príklad štruktúry DOM tree HTML kódu [19]

Na zmenu textu nadpisu elementu `<h1>` v predchádzajúcom strome môžeme použiť nasledujúcu metódu, ktorá vráti zoznam všetkých elementov rovnakého typu v dokumente a pomocou vlastnosti vráteného objektu zmeníme obsah:

```
var nadpisy = document.getElementsByTagName('h1');

nadpisy[0].innerHTML = 'Velky nadpis';
```

Tvorba dynamických stránok a aplikácii s vykonávaním na strane klienta (tzv. Front-end) sa nezaobíde bez prístupu a využívaniu DOM modelu, preto je k používaniu pokročilých funkcii HTML5 potrebné aspoň minimálne použitie JavaScriptu.

2 ZMENY MEDZI VERZIAMI

Bežný webový dokument tvorí samotný HTML súbor, ktorého obsah pozostáva z textu a formátovacích značiek, tzv. tagov, ktoré formujú text do stavebných elementov. Značky sú párové, medzi ktoré umiestňujeme text a iné elementy, alebo nepárové, ktoré nesú informáciu len formou vlastností, tzv. atribútov. Atribúty značiek obsahujú v závislosti na type buď hodnotu, alebo svojou prítomnosťou v značke povoľujú určitú vlastnosť, resp. neprítomnosťou zakazujú - nazývajú sa boolean atribúty.

2.1 Zvládanie chýb, deklarácia dokumentu

Obe špecifikácie HTML5 sú dostupné v dvoch verziách. Kompletné verzie [15][17] obsahujú časť primárne určenú tvorcom prehliadačov presne popisujúce očakávané správanie prehliadačov pri prekladaní chybného kódu alebo zastaraných značiek a atribútov, ktoré sa už neodporúčajú používať. Odľahčené verzie špecifikácií [14][16] obsahujú len informácie potrebné pre vývojárov a dizajnérov webových stránok a aplikácii. Urýchľuje to implementáciu nových funkcií do prehliadačov, pretože už nemusia vymýšľať ako ošetriť chyby a zároveň to neobmedzuje dizajnérov a programátorov, aby používali striktný bezchybný zápis, ako tomu bolo pri verziách založených na XML.

Predchádzajúce verzie HTML mali okrem špecifikácie aj presnú strojovú definíciu zvanú DTD (Document Type Definition) opisujúcu pravidlá vykresľovania elementov pre prehliadače ako pozostatok SGML. HTML5 nie je založený na SGML a nevyžaduje odvolanie na DTD, pretože pravidlá vykresľovania sú implementované v prehliadačoch podľa kompletnej špecifikácie. Odráža sa to v zjednodušení deklarácie typu dokumentu.

V HTML 4.01 sa na začiatku dokumentu vložila odvolávka na príslušnú DTD.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

V HTML5 sa uvádza jednoducho,

```
<!DOCTYPE html>
```

a hovorí prehliadaču, že má dokument vykresliť v štandardnom móde, tzn. presne podľa špecifikácie. Štandardný mód používajú všetky moderné prehliadače.

2.2 Pravidlá zápisu kódu

Niektoré pravidlá z XHTML, ktoré sa medzi dizajnérmi neformálne zaužívali uniformujú zápis, takže je všeobecne ľahšie čitateľný. Syntax HTML5 nevyžaduje uvádzať značky `<html>`, `<head>` ani `<body>`, značky a atribúty je možné písať ľubovoľnou veľkosťou písma, nepárové značky netreba uzatvárať, hodnoty atribútov netreba obaľovať do úvodzoviek pokiaľ neobsahujú vzorec, ktorý obsahuje znaky ' , " , < , > alebo = . Jednoducho povedané HTML5 neviaže autorov striktnými pravidlami zápisu kódu a zároveň poskytuje spätnú kompatibilitu pri prechode zo starších verzií.

2.3 MIME typy

MIME, skrátene pre Multipurpose Internet Mail Extensions, je internetový štandard používaný k opisu obsahu súborov, iných ako text v ACSII, pripájaných k emailu. Pod pomenovaním Internet media type alebo Content-type je používaný v HTML a XML pri vkladaní externých súborov do dokumentu. Je určený typom, podtypom a voliteľnými atribútmi.

Na úpravu vzhľadu sa používa len CSS, preto prehliadače automaticky predpokladajú použitie CSS a nevyžadujú presné určenie typu pri použití v dokumentoch. Rovnako je to u JavaScriptu. Zápis sa zjednodušil nasledovne:

```
5:    <link rel="stylesheet" href="css/style.css">
      <script src="moj_script.js"></script>
```

miesto dlhého:

```
4.01: <link rel="stylesheet" type="text/css" href="css/style.css">
      <script src="moj_script.js" type="text/javascript"></script>
```

Informácia o použitej znakovej sade obsahu, ktorá slúži na správne zobrazenie znakov dokumentu sa takisto zjednodušila.

```
4.01: <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5:    <meta charset="UTF-8">
```

Použitím popísaných skrátených verzii definícii parametrov dokumentu môžeme napísať šablónu s jednouchou štruktúrou oddeľujúcu informačnú a obsahovú časť dokumentu. Rozdiel oproti verzii HTML 4.01 vidieť v nasledujúcej tabuľke.

Tab. 2-1Porovnanie štruktúry dokumentu HTML 4.01 a HTML5

HTML 4.01	HTML5
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <title>Dokument</title> <link rel="stylesheet" type="text/css" href="css/style.css"> <script src="mojScript.js" type="text/javascript"></script> </head> <body> <!-- obsah dokumentu --> </body> </html>	<!DOCTYPE html> <html lang="sk"> <head <meta charset="UTF-8"> <title>Dokument</title> <link rel="stylesheet" href="style.css" media="all"> <script src="mojSkript"></script> </head> <body> <!-- obsah dokumentu --> </body> </html>

2.4 Elementy a atribúty

Jedným z princípov HTML5 je natívne, bez závislosti na inej technológii, podporovať čo najviac funkcii, ktoré vývojári a dizajnéri využívajú. Po roku 2000 narastal dopyt po dynamických funkciách, ktoré by priniesli väčšiu interaktivitu. Tie však neboli dostupné v HTML/XML a vznikali náhrady písané v rôznych jazykoch a následne vkladané do stránok formou zásuvných modulov a skriptov. V rozmachu bol AJAX, FLASH a prvé JavaScriptové aplikácie. Implementácie sú rôznorodé, od prehrávačov mediálnych súborov cez hry až po drobné pomôcky na výber dátumu vo formulároch. Niektoré z nich boli adoptované do špecifikácie, čím sa zjednodušila ich implementácia.

Osobitný význam sa kladie na sémantický opis dokumentov. Význam značiek bol pozmenený a použitie niektorých značiek bolo prehodnotené a následne boli označené ako zastarané, takže by sa v budúcnosti nemali používať. Prehliadače ich však budú naďalej podporovať. Nahrubo členený obsah pomocou `<div>` je teraz možné opísať aj radou nových sémantických značiek.

2.4.1 Štruktúrové elementy

Nové značky, ktoré pomáhajú presnejšie popísať štruktúru obsahu dokumentu. Ich význam je podchytený v špecifikácii, aby bolo ich použitie jednotné.

<article> predstavuje úplnú alebo nezávislú kompozíciu dokumentu, aplikácie alebo stránky, ktorá môže byť znova využitá. Príkladom je článok, príspevok v blogu, alebo komentáre.

<section> reprezentuje časť, sekciu dokumentu alebo aplikácie. Obsah sekcie je tematicky zoskupený, zvyčajne označený nadpisom. V prípade, že je možné použiť element, ktorý by obsah lepšie opísal, treba ho použiť miesto `<section>`. Je primárne určený na zápis, ktorý sa má objaviť v obryse dokumentu, takže pri použití na štylizáciu obsahu treba použiť `<div>`.

<nav> predstavuje časť stránky, ktorá obsahuje prepojenia na iné časti stránky alebo na iné stránky. Je primárne určená pre hlavnú navigáciu, nie pre každé zoskupenie odkazov v stránke.

<aside> reprezentuje časť stránky, ktorej obsah aspoň čiastočne súvisí s obsahom v blízkosti elementu `<aside>`, a môže byť osamostatnený bez straty významu. Môže byť použitý na reklamu, citácie z článku a na ostatný obsah oddelený od hlavného obsahu. Nemal by byť použitý na rozširujúce informácie v texte.

<hgroup> element obaľuje úrovne nadpisov `<h1>` až `<h6>`, pričom v obryse dokumentu maskuje vnorené úrovne podnadpisov, ktoré by inak narušili obrys dokumentu. Názov sekcie v obryse je daná hodnotou najvyššej úrovne vnoreného nadpisu. Element bol vylúčený zo špecifikácie no pre úplnosť uvádzame chovanie elementu na nasledujúcom príklade:

Tab. 2-1 Funkcia elementu <hgroup>

zápis:	obrys dokumentu:
<hgroup> Kapely Tubatanka Hudbička.sk </hgroup>	1. Hudbička.sk

<header> by mal obsahovať úvod alebo navigačné prvky. Zvyčajne obsahuje nadpisy článkov a komentárov. V prípade umiestnenia v čele dokumentu obsahuje logo a vyhľadávací formulár. V každej sekcii by mal byť najviac jeden krát. Nezobrazuje sa v obryse dokumentu, pretože nevytvára novú sekciu.

<footer> predstavuje ukončenie sekcie, ktorej je súčasťou. Zvyčajne obsahuje údaje o autorovi, prepojenia súvisiace s dokumentom, licenčné a autorské údaje, element <address>. Takisto ako u <header>, hlavným účelom je pomôcť autorovi vytvoriť zápis, ktorý sa ľahko udržiava a vizuálne štylizuje, pričom nevytvára novú sekciu. V prípade, že je umiestnený v sekcii <body>, vzťahuje sa k obsahu celej stránky.

2.4.2 Obrys dokumentu

Obrys dokumentu (prehľad) je štruktúra dokumentu, pomocou ktorej prehliadače vedia vytvoriť jeho obsah, podobný tomu v knihách. Je tvorený nadpismi, názvami tabuliek a formulárov alebo ostatných prvkov dokumentu. V HTML5 navyše pomocou nových štruktúrových elementov <article>,<aside>,<nav>,<section>, ktoré sú radené do kategórie sekciového obsahu. Vytvorený obsah je následne používaný pre orientáciu po stránke pomocnými technológiami (čítačky obrazoviek, hlasová navigácia), alebo spracované vyhľadávačmi pre skvalitnenie výsledkov vyhľadávania. Tvorba obrysu dokumentu bola pred HTML5 tvorená výlučne pomocou rôznych úrovní nadpisov, kde <h1> predstavuje najvyššiu úroveň a <h6> najnižšiu. Obrys pozostáva z jednej alebo viacerých vnorených sekcii, pričom každá sekcia by mala obsahovať nadpis a ľubovoľný počet podsekcií.

Pravidlá, ktoré opisujú správanie elementov pri tvorbe obrysu:

- prvý nadpis v sekcii určuje názov sekcie, v ktorej je vnorený
- ďalšie nadpisy rovnakej alebo nižšej úrovne už tvoria novú sekciu
- <article>,<aside>,<nav>,<section>,<h1> až <h6> vytvárajú novú sekciu

Existujú tzv. sekciové korene, ktoré majú svoj vlastný obrys. Pokiaľ sú navzájom vnorené, sú v obryse nadradeného koreňa skryté. Toto správanie je napríklad vidieť pri elemente <body> v ukážke (Tab. 2-2). Medzi korene patria nasledujúce elementy: <blockquote>, <figure>, <details>, <fieldset>, <td> a <body>.

Tab. 2-2 Sekciové korene

zápis:	obrys dokumentu:
<body> názov stranky <article> názov článku začiatok článku... <blockquote> Mary Wardová Never slovám, ak za nimi nestoja skutky. </blockquote> ...pokračovanie článku </article> </body>	1. názov stránky 1. názov článku

Pri zrovnaní v príklade (Tab. 2-3) je vidieť, že zápis obrysu v HTML5 je oveľa komplikovanejší, no pozitívom je, že každá sekcia vytvorená pomocou sekciových elementov umožňuje použiť všetky úrovne nadpisov bez závislosti na úrovni ktorá by inak mala hierarchicky nasledovať. Nabáda k tomu aj špecifikácia [16]:

"Sekcie môžu obsahovať akékoľvek úrovne nadpisov, no dôrazne odporúčame autorom používať buď <h1> elementy alebo úroveň adekvátnu k úrovni vnorenia sekcie."

Negatívom je strata vizuálnej hierarchie nadpisov v dokumente, čo môže pri vypnutom CSS na úpravu vzhľadu znížiť prehľadnosť v dokumente. Pre zachovanie kompatibility voči starším prehliadačom je vhodné voliť kompromis použitím nových elementov s dodržaním hierarchie úrovní nadpisov, tak ako uvádza špecifikácia.

Tab. 2-3 Porovnanie obrysov v HTML4 a HTML5

zápis v HTML5:	zápis v HTML 4.01:
<body> Hudbička.sk <nav> Rubriky RockPop Underground </nav> <section>	<body> Hudbička.sk Rubriky RockPop Underground Novinky Spevák ušiel z pódia <!-- obsah článku -->

```
    <h1>Novinky</h1>
      <article>
        <h1>Spevák ušiel z pódia</h1>
        <!-- obsah článku -->
          <h2>komentáre</h2>
      </article>
      <article>
        <h1>Turné kapely Žumpa!</h1>
        <!-- obsah článku -->
          <h2>komentáre</h2>
      </article>
  </section>
</body>
```

```
      <h4>komentáre</h4>
    <h3>Turné kapely Žumpa!</h3>
    <!-- obsah článku -->
      <h4>komentáre</h4>
</body>
```

Zápisom štruktúry v tabuľke (Tab. 2-2) vznikne v oboch prípadoch nasledujúci obrys:

```
1. Hudbička.sk
      1. Rubriky
            1. RockPop
            2. Underground
      2. Novinky
            1. Spevák ušiel z pódia
                  1. komentáre
            2. Turné kapely Žumpa!
                  1. komentáre
```

2.4.3 Ostatné nové elementy

Okrem štruktúrových elementov pribudli do špecifikácie aj elementy, ktoré implementujú správanie prvkov dosahované náhradnými riešeniami. Podpora v prehliadačoch je rôzna, preto pre súčasnú podporu v prehliadačoch odporúčame stránku CanIuse.com[1], ktorá je pravidelne aktualizovaná.

`<command>` slúži na špecifikovanie príkazu, na ktorý sa môžu ostatné časti, tlačidlá a odkazy dokumentu odvolať. Podporuje ho len prehliadač Mozilla Firefox. Pokiaľ ho neimplementujú ostatné prehliadače, bude vylúčený zo špecifikácie.

`<details>` slúži na zobrazovanie a skrývanie vnoreného obsahu. Nadpis, ktorý sa má zobraziť pred rozbalením definuje vnorený element `<summary>`. Ak sa vynechá, prehliadače zobrazia predvolený text. Podporujú ho prehliadače Chrome, Safari 6+ a mobilné prehliadače Android browser 4+ a Blackberry browser 10.

`<figure>` je určený na obalenie mediálneho obsahu (obrázku, videa, grafu) s popisom obsahu uloženým vo vnorenom elemente `<figcaption>`.

[1] http://caniuse.com

```
<figure>
      <img src=balon.jpg alt="teplovzdusny balon">
      <figcaption id=balon>Teplovzdušný balón</figcaption>
</figure>
```

<mark> zvýrazňuje text medzi značkami, preto je používaný na zvýraznenie hľadaného reťazca po použití vyhľadávača alebo na zvýraznenie reťazca slov v bloku textu vzhľadom na aktuálnu činnosť užívateľa, ako je napríklad filtrácia obsahu. Do kódu ho dynamicky pridáva JavaScript alebo aplikácia na strane servera.

<meter> slúži na zobrazenie konečného výsledku formou sklenenej banky.

<progress> slúži na vizualizáciu práve prebiehajúceho procesu, napríklad načítania, preberania súboru alebo spracovania výsledku. Efektívne využitie je v kombinácii s JavaScriptom, ktorý dynamicky mení parametre hodnoty s postupom procesu.

<ruby> <rt> <rp> pomôcka pri zápise formátovania východných jazykov. Samotný element je riadkový, ale spolu s `<rt>` a `<rp>` je možné vyskladať znaky východných jazykov podľa výslovnosti do blokov nad seba, pričom sa ako celok sú stále v rozsahu jedného riadka.

<time> umožňuje vyjadriť dátum a čas v ľubovoľnom človekom čitateľnom formáte, no zároveň uchováva aj počítačom čitateľnú číselnú verziu v atribúte `datetime`. Hodnota následne môže byť formátovaná pomocou JavaScriptu podľa lokálneho formátu času alebo použitá vyhľadávačmi pri použití časovej relevancie. Boolean atribút `pubdate` použije časový údaj ako dátum zverejnenia/pridania v rámci elementu `<article>`. V prípade, že `<time>` nie je súčasťou elementu `<article>`, atribút `pubdate` sa vztiahne na celý dokument. Vhodné napríklad na komentáre článkov, dátumy transakcii či dátumy aktualizácii.

Príklad času vo formáte HH:MM:SS.ms (oddeľovač dátumu a času je ' T '):

```
<time datetime="2012-10-28T18:20:15.014+01:00">28. Októbra 2012  šesť
hodín večer</time>
```

Ak `<time>` neobsahuje `datetime` atribút, obsah medzi značkami `<time>` músi byť platný dátový reťazec. Ak obsahuje `datetime`, tak hodnota atribútu `datetime` musí byť platný dátový reťazec.

<wbr> vkladá do textu značku, kde by sa mal zalamovať v prípade zmenšenia rozmeru okna prehliadača aby nevzniklo neželané vodorovné rolovanie. Využitie je hlavne pri dlhých textoch. Je ekvivalentný so znakmi `&#8203;` a `&shy;`.

<keygen> element slúži na obsluhu generovania kľúčového páru. Pri odoslaní formulára je súkromný kľúč lokálne uložený a verejný kľúč je zaslaný na server. Každý prehliadač ponúka iné možnosti zabezpečenia kľúča formou roletky. Atribútom `keytype` sa špecifikuje typ vygenerovaného kľúča (rsa, dsa) a atribút `challenge` špecifikuje samotný reťazec, ktorý je zaslaný spolu s verejným kľúčom.

<output> účelom tohto elementu je prijať a zobraziť výsledky výpočtu. Mal by byť použitý tam, kde má užívateľ vidieť hodnotu, ale nenarábať s ňou, a kde má by hodnota vypočítaná z ostatných vložených dát formulára. Príkladom môže byť výpočet celkovej sumy v košíku internetového obchodu po zarátaní dane a poštovného. Výstupná hodnota je vložená medzi párovými značkami elementu. Na aktualizáciu hodnoty je vhodný JavaScript. Element má atribút `for`, ktorý odkazuje na identifikátory `id` formulárových polí, ktorých hodnoty boli použité pri výpočte.

2.4.4 Pozmenené elementy

Zmena definície významu elementov sa nesie v duchu sémantiky, tzn. aby kód vystihoval charakter obsahu prioritne nad vizuálnou interpretáciou obsahu.

<address> v HTML 4 slúžil na zobrazenie kontaktných údajov autora dokumentu, pretože nepoznal element `<article>`. V HTML5 sa môže vyskytovať v stránke viac krát, konkrétne v každom elemente `<article>`, a tak uvádzať informácie aj o autorovi článku. Nemal by obsahovať poštové údaje, pokiaľ nie sú súčasťou kontaktných údajov autora.

<cite> v HTML 4 popisoval citáciu alebo referenciu na iný zdroj (knihu, skupinu alebo individuálnu osobu). v HTML5 reprezentuje názov diela (napríklad knihy, básne, piesne, filmu, hry, …) a nemôže byť použitý na označenie mena osoby. Táto zmena vzbudzuje vlnu kontroverzie a ide proti princípu spätnej kompatibility.

```
<p><cite>Mona Lisa</cite> od Leonarda da Vinci je v Paríži.</p>
```

<dl> zoznam pojmov spolu so združeným elementom `<dt>` (pojem) a `<dd>` (význam) mohli byť v HTML 4 použité na dialóg. Špecifikácia HTML5 tento spôsob použitia

zakazuje. Taktiež sa zmenil názov z *definition list* na *description list* (opisný zoznam). Mali by byť použité na rôzne páry názov-hodnota ako sú napríklad často kladené otázky a odpovede (FAQ).

** <i>** - špecifikácia rozlišuje význam týchto dvoch elementov aj napriek totožnému zobrazeniu. Zatiaľ čo zvýrazňuje časť vety, ktorá mení jej význam, element <i> vyznačuje frázu, konkrétne pomenovanie či technický výraz.

```
<p><i>Legovaná oceľ/i> je oceľ s vyšším obsahom prímesí.... </p>
<p>Stretneme sa <em>dnes podvečer</em>.</p>
```

<hr> v HTML4 plnil úlohu vodorovnej čiary. Po sémantickej úprave od seba oddeľuje významové celky sekcie.

<ol> - samotný význam elementu sa nezmenil, no pribudli dva nové atribúty. Atribút start definuje, od akej hodnoty má začať číslovanie zoznamu. Boolean atribút reversed zase mení smer akým je zoznam číslovaný. Použitie vidíme napríklad v číslovaní výsledkov štatistík a podobne.

```
<ol start="10" reversed>
  <li>Pavol Rybníček </li>
  <li>Rudolf Rýchly</li>
  <li>Tomáš Durič</li>
</ol>
<p>aktuálne poradie pretekárov bude
upravené po skončení etapy </p>
```

```
10. Pavol Rybníček
9.  Rudolf Rýchly
8.  Tomáš Durič

aktuálne poradie pretekárov bude
upravené po skončení etapy
```

<s> element prešiel sémantickou zmenou. Podobne ako element *<strike>* zobrazuje stredom preškrtnutý text. Naznačuje však, že preškrtnutý text už nie je presný alebo platný. Použiteľné napríklad pri cenovom výpredaji na označenie pôvodnej ceny.

<small> element prešiel kompletnou zmenou významu. Pôvodne malý text teraz znamená drobnú tlač, ktorá zahŕňa licenčné podmienky alebo autorské práva.

**** predstavuje zvýšenú dôležitosť významu v kontexte, ale na rozdiel od nemení význam vety. Vnáranie viacerých elementov zvyšuje dôležitosť.

```
<p><strong>Pozor! Môže spôsobiť epilepsiu.</strong></p>
```

Element **<b>** len zvýrazňuje časť textu bez vnášania dôležitosti, zmeny prejavu alebo zafarbenia. Napríklad na označenie kľúčových pojmov v abstrakte, názvov výrobkov v recenzii a iné.

```
<p>Vždy sa veľmi teším na maminu <b>škoricovú bábovku<b>.</p>
```

2.4.5 Nové atribúty

contenteditable - boolean atribút, ktorý povoľuje úpravu obsahu. Zmeny môžu byť ukladané na strane užívateľa alebo odosielané na server, napríklad na prispôsobenie zobrazenia špecifickému pre užívateľa alebo úpravu článkov.

draggable, dropzone - dvojica atribútov slúži na označenie elementov v dokumente, ktoré môžu byť pretiahnuté. Atribút `draggable` povoľuje pretiahnutie a zatiaľ prehliadačmi nepodporovaný atribút `dropzone` označuje element, ktorý môže prijať ťahané objekty. Viac k použitiu v sekcii Drag and Drop.

hidden - boolean atribút, ktorý skrýva element, v ktorom je použitý. Znamená to, že element už nepotrebuje byť zobrazený, ostáva však použiteľný ostatnými časťami stránky na rozdiel od alternatívneho riešenia pomocou CSS (`display: none;`). Jeho prítomnosť sa dá meniť pomocou JS, čo umožňuje vytvoriť dynamické chovanie stránky. Napríklad prihlasovací formulár a podobne. Nepodporovaný v IE.

spellcheck - boolean atribút, ktorý povolí alebo zruší kontrolu pravopisu elementu.

tabindex - atribút svojou hodnotou určuje, v akom poradí majú byť zameriavané elementy (`focus`), zväčša formulárové polia, pomocou klávesy `tab`.

accesskey - atribút mapuje klávesovú skratku elementu, ktorá ho zameria alebo aktivuje. Skratky sú použiteľné v kombinácii `alt + { skratka }` vo všetkých prehliadačoch okrem Opery. Atribút nie je nový, no oproti HTML4 je použiteľný na každý element - globálny.

```
<!--po stlačení skratky prejde na adresu, na ktorú odkazuje prepojenie -->
<a href="http://Hudbička.sk/rockPop.html" accesskey="r">RockPop</a>
```

async - atribút elementu `<script>`. Umožňuje načítanie skriptu paralelne (na pozadí) so zvyšným obsahom stránky bez toho, že by obmedzoval jeho načítanie a hneď po načítaní ho spustí. Podmienkou použitia je externý skript, nie skript vo vnútri HTML kódu. Je vhodné ho použiť, keď skript nie je závislý na iných zdrojoch alebo elementoch HTML a skoré načítanie by mohlo obmedziť jeho funkčnosť. Pri použití `async` nie je možné použiť atribút `defer` a opačne. Atribút `defer` zabezpečuje, že sa skript načíta až po načítaní ostatného obsahu dokumentu. V prípade použitia starších prehliadačov, respektíve prehliadačov, ktoré atribút `async` nepodporujú je vhodné použiť oba atribúty. V súčasnosti ho nepodporuje len Internet Explorer <9 a Opera.

2.4.6 Zastarané elementy a atribúty

Niektoré elementy boli označené ako zastarané a nemali by sa používať. Ich efekty sú dosiahnuteľné iným spôsobom, zväčša pomocou CSS. Je možné ich aj naďalej používať, pretože ich správanie je stále dokumentované v špecifikácii. Prehľadu je venovaná osobitná sekcia v špecifikácii [18].

2.5 Formuláre

Nové elementy, atribúty a vstupné typy formulárov uvedené v novej špecifikácii z pohľadu užívateľa neprinášajú nič nové. Ich správanie je už roky implementované dizajnérmi pomocou JavaScriptu alebo AJAXu. HTML5 ich implementáciu však robí jednoduchšiu. Validovaním (overovaním) obsahu v prehliadači bez nutnosti komunikácie so serverom je správanie rýchlejšie, respektíve plynulejšie ako náhradné riešenia.

Vzhľad jednotlivých prvkov nie je v špecifikácii popísaný, takže každý prehliadač používa vlastnú prezentáciu. Je tu možnosť prispôsobiť vzhľad pomocou CSS selektorov. V niektorých prípadoch to môže viesť k neprehľadnosti, takže prispôsobenie treba zvážiť. Užívatelia by nemali ostať pri pohľade na formulár zmätený. Príklady prispôsobenia pomocou selektorov sú v praktickej časti.

2.5.1 Nové vstupné typy

Okrem pôvodných dátových typov elementu `<input>` pribudlo trinásť nových dátových typov. Pomáhajú bližšie špecifikovať dátový obsah pri validácii a takisto vylepšujú prístupnosť zo strany skript. Mobilné zariadenia vedia podľa nich upraviť užívateľské rozhranie pre rýchle a efektívne vypĺňanie polí. Prehliadačom nepodporované vstupné typy sú brané ako `type="text"`.

Tab. 2-4 Prehľad vstupných typov formulárov

pôvodné dátové typy		**nové dátové typy**		
button	password	search	week	color
checkbox	radio	email	url	range
file	reset	datetime	datetime-local	time
hidden	submit	tel	number	

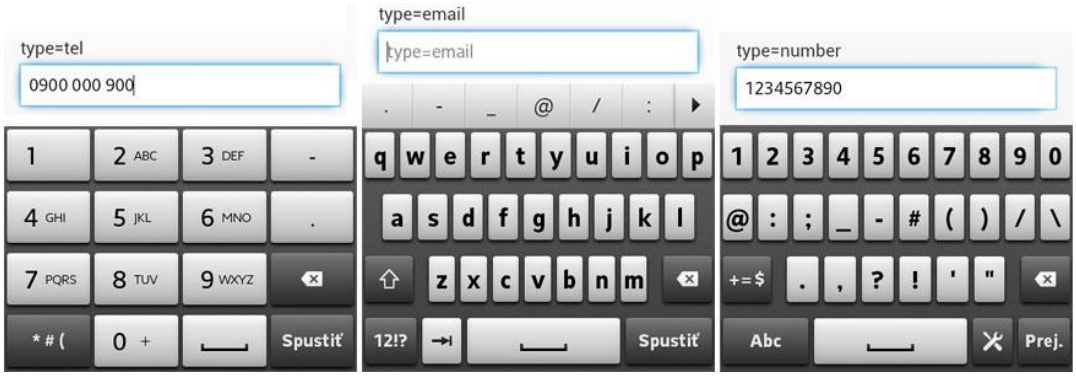
search - prehliadač očakáva reťazec pre vyhľadávanie. Rozdiel voči typu ***text*** je len štylistický, preto podporujúce prehliadače dopĺňajú k pravému okraju tlačidlo pre vymazanie obsahu.

email - prehliadač overí či vstupná hodnota je v platnom formáte emailovej adresy, neoveruje existenciu adresy. Umožňuje načítať viac adries oddelených čiarkou pri použití voliteľného boolean atribútu `multiple`.

url - slúži na vkladanie URL adries. Podobne ako email, vizuálne sa neodlišuje, no na mobilných zariadeniach sa upraví virtuálna klávesnica pre ľahšie zadávanie. Niektoré prehliadače poskytujú aj overovanie správneho formátovania.

tel - definuje pole pre vloženie telefónneho čísla. Formát čísla je možné naznačiť pomocou atribútu `placeholder` a presne definovať pomocou atribútu `pattern`.

number - definuje pole na vklad číselných hodnôt. V pravej časti poľa sa nachádzajú tlačidlá na zvyšovanie alebo znižovanie hodnoty. Hodnotu je možné zadať buď ručne pomocou klávesnice alebo navoliť pomocou tlačidiel. Atribút `step` udáva krok zvyšovania alebo znižovania hodnoty tlačidlami, `min` a `max` rozsah hodnôt. Dotykové zariadenia zobrazia pri zadávaní numerickú klávesnicu. Niekoľko ukážok z mobilnej verzie prehliadača v systéme Android:



Obr. 2.1 Zmena rozloženia klávesnice (Android)

date - poskytuje natívny kalendár, ktorý uľahčuje zadávanie dátumu. Formát je závislí na prehliadači, no obsahuje len deň, mesiac a rok. (2008-05-13)

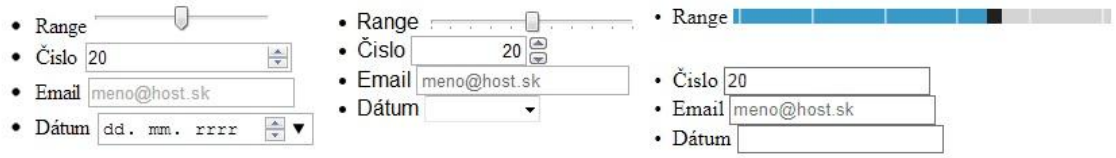
month, week - uchovávajú informáciu len o mesiaci vo formáte (2012-10) alebo o týždni (2012-W16)

time - vstup času v 24 hodinovom formátovaní (00:00-23:59)

datetime,datetime-local - rozširuje dátum o časový údaj. V zápise sú oddelené znakom T. `(2012-11-02T19:21+1:00)`

range - miesto poľa sa zobrazí bežec, ktorým užívateľ nastaví vstupnú hodnotu. Nezobrazuje presnú hodnou, takže je vhodný na menej presné zadávanie. V prípade konkrétneho zadania hodnoty je lepši použiť typ `number` alebo použiť na zobrazenie hodnôt CSS.

color - farebná paleta s možnosťou výberu farby v hexadecimálnom formáte.



Obr. 2.2Porovnanie typov v Chrome, Opera, IE 10

2.5.2 Nové atribúty formulárov

placeholder - atribút umožňujúci zobraziť pomôcku, ktorá užívateľovi naznačí aké vstupné dáta by mal zadať. Text zmizne ako náhle užívateľ vloží prvý znak, a znova sa objaví keď zmaže obsah. Vývojári túto funkciu doteraz implementovali pomocou JavaScriptu. Pokiaľ je text dlhší ako šírka poľa, je vhodné pomôcku zapísať do atribútu `title`, ktorý sa zobrazí pri podržaní kurzora nad poľom.

required - boolean atribút, ktorý hovorí prehliadaču aby odoslal formulár len ak sú údaje v poli vyplnené. Pole teda nemôže ostať prázdne a takisto nemôže obsahovať nepovolené znaky pre daný vstupný typ. Ak sú povinné polia formulára vyplnené nesprávne alebo sú ponechané prázdne, formulár počká s odoslaním a presunie kurzor k prvému nevyhovujúcemu poľu. Prehliadače Opera, Firefox, Chrome a IE10 tiež upozornia užívateľa pomocou varovných správ. Atribút môže byť použitý v akomkoľvek elemente okrem typov `button, range, color a hidden`.

```
<input type="text" name="Meno" required placeholder="Michal">
```

autofocus - atribút nastaví kurzor do zvoleného poľa formulára po načítaní stránky. Pre správne fungovanie smie mať túto vlastnosť len jeden element v stránke. Správanie možno vidieť napríklad vo vyhľadávači Google - ihneď po načítaní stránky sa kurzor nastaví do vyhľadávacieho poľa aby užívateľ mohol bez zdržania zadať reťazec na

vyhľadanie. Táto vlastnosť však môže mať aj negatívny efekt pre používateľov pri nesprávnom použití, preto treba použitie vopred zvážiť.

```
<input  type="email" id="email" name="email" autofocus
placeholder="janko@gmail.com" >
```

autocomplete - automatické vyplňovanie formulárov je jedna z funkcii prehliadača. Nie vždy však túto možnosť chceme mať k dispozícii, najmä z bezpečnostných dôvodov. Tento atribút zakazuje alebo povoľuje prehliadaču vyplniť formulár údajmi. Prehliadač predpokladá, že hodnota parametra `autocomplete` je ON pokiaľ v zápise nepoužijeme OFF.

```
<input id="email" name="email" type="email" autocomplete="off">
```

pattern - atribút umožňujúci zadať bežný výraz, ktorý užívateľ musí dodržať aby boli vstupné dáta overené ako platné. Jazyk použitý na písanie vzorov (`pattern`) je založený na Perl podobný jazyku JavaScript. Základy jazyka nie sú predmetom tejto práce, preto pre viac informácii, použite vo vyhľadávači reťazec „regular expressions“. Pri použití vzoru je vhodné upozorniť užívateľa na požiadavky pri zadávaní v `placeholder, title` alebo osobitne v blízkosti poľa.

Príklad: heslo, ktoré musí mať dĺžku od 6 do 12 znakov
\S – akýkoľvek znak okrem medzery; {6,12} – počet znakov od 6 do 12:

```
<input type=”password” id=”password” pattern=”\S{6,12}”>
```

disabled - boolean atribút, ktorý môže byť použitý s ktorýmkoľvek prvkom formulárov okrem `<output>`. Elementy s týmto atribútom majú neaktívny obsah a nie sú odosielané so zvyškom formulára. Prehliadače zabránia zamerania (focus) na tieto elementy. Môže byť použitý napríklad na zablokovanie tlačidla odoslať kým nie sú vyplnené všetky požadované polia formulára.

readonly - atribút podobne ako `disabled` znemožňuje užívateľa editovať obsah, môže však dostať focus a je odosielaný spolu so zvyškom formulára. Príklad použitia je pevne daný názov predmetu pri odosielaní komentáru alebo emailu.

```
<input type="text" id="predmet" value="ABC" readonly>
```

multiple - umožňuje výber alebo vklad viacerých hodnôt. V predchádzajúcich verziách sa tento atribút vzťahoval len na element `<select>`. V HTML5 môže byť použitý aj vo vstupných poliach typu `email` alebo `file`. Umožní to zadať viac emailových adries oddelených čiarkou alebo načítať viac súborov.

list, <datalist> - Zlučuje správanie elementov `<input>` a `<select>`. Atribútom `list` umiestnenom v značke `<input>`, ktorý obsahuje identifikátor `id` zoznamu možností `<datalist>`, sa možnosti zo zoznamu previažu so vstupným poľom. To umožní používateľovi vybrať si jednu z predvolených možností, alebo napísať vlastnú, ktorá sa nevyskytuje v zozname. V prípade, že prehliadač nepodporuje element `<datalist>`, elementu `<input>` sa to nedotkne. Položky zo zoznamu vkladáme ako vnorené elementy `<option>` s atribútom `value` - zobrazený text položky.

```
<input type="text" id="krajina" name="krajina" list="ponuka">
      <datalist id="ponuka">
            <option value="Slovenská Republika">
            <option value="Česká Republika">
      </datalist>
```

2.6 Microdata

Stránky a aplikácie sú určené predovšetkým pre ľudí. Podľa toho formulujeme aj ich obsah. HTML5 upravuje značky aby zápis boli viac sémantický, čiže významový. Opis len výhradne pomocou značiek však ľuďom moc nepovie, a strojom poskytne len základný prehľad o obsahu. Aby sme poskytli úplné informácie o obsahu, alebo aspoň o jeho potrebných častiach, treba siahnuť za hranice HTML.

Micro-dáta, ako sú tieto dodatočné informácie všeobecne nazývané, sú bežné pojmy tematicky zoskupené v slovníkoch podľa rôznych kritérií. Odvolaním sa na ne z kódu vedia stroje, hlavne vyhľadávače, presnejšie pochopiť a rozlíšiť obsah dokumentov. Pred HTML5 existovali dva spôsoby, ako tieto nadbytočné, no užitočné informácie vložiť do zápisu. Použitím atribútov `class` (microformats[2]) alebo pomocou atribútov `property` a `content` (RFDa [3] od W3C). Obe riešenia sú stále používané.

[2] http://microformats.org/

[3] http://www.w3.org/TR/rdfa-in-html/

Špecifikácia HTML5 pridáva vlastnú sadu atribútov pre bližší opis obsahu dokumentov, ktoré sa pridávajú do už existujúcich elementov zápisu.

itemscope boolean atribút, ktorý označí oblasť, v ktorej sa nachádzajú položky na bližšie popísanie obsahu.

itemtype definuje, aký slovník sa použije pri popise položiek. Uvádza sa spolu s atribútom `itemscope`. Obsahuje URL s umiestnením slovníka.

itemprop priraďuje vlastnosť zo slovníka k časti obsahu, ktorá ju čo najpresnejšie opisuje.

Slovník s položkami si možno vytvoriť, ale pre univerzálne použitie odporúčame použiť niektorý z už existujúcich slovníkov. Vyhľadávače Google, Bing, Yahoo! a Yandex používajú pri rozpoznávaní obsahu slovníky z lokality Schema.org[4], ktorú vo vzájomnej kooperácii vytvorili a spravujú. Slovníky z nej sú používané aj v RDFa.

Ako príklad sme použili popis knihy v internetovom obchode, na opis ktorej sme použili vlastnosti zo slovníka Book[5] a Offer[6]z lokality Schema.org.

viditeľné informácie:

```
Facebook efekt
David Kirkpatrick, vydavateľstvo Eastone Books, 2011
Cena: 18,99€
O knihe: Aká je skutočná pravda o vzniku najpopulárnejšej siete Facebook?...
Detaily:
      Jazyk: slovenský jazyk
      Počet strán:360
      ISBN: 9788081091889
```

---

zápis s Microdata:

```
<div itemscope itemtype="http://schema.org/Book">
  <h2><span itemprop="name">Facebook efekt</span></h2>
  <h3><span itemprop="author">David Kirkpatrick</span>, vydavateľstvo
      <span itemprop="publisher">Eastone Books</span>,
      <span itemprop="datePublished">2011</span></h3>
  <div itemprop="offers" itemscope itemtype="http://schema.org/Offer">
      Cena: <span itemprop="price">20€</span>
  </div>
  <p>O knihe:<span itemprop="about"> Aká je skutočná pravda o vzniku
```

[4] http://schema.org

[5] http://schema.org/Book

[6] http://schema.org/Offer

```
  najpopulárnejšej siete Facebook?...</span></p>
  Detaily:
       <ul>
         <li>Jazyk: <span itemprop="inLanguage">slovenský jazyk</span></li>
         <li>Počet strán: <span itemprop="numberOfPages">360</span></li>
         <li>ISBN: <span itemprop="isbn">9788081091889</span></li>
       </ul>
</div>
```

2.7 Atribúty data-*

Na ukladanie dodatočných údajov do kódu, ktoré sa dajú ďalej využívať pred HTML5 slúžili atribúty `class` a `rel`. Mnohokrát sa to nezaobišlo bez konfliktu medzi vzhľadom a funkcionalitou stránky. HTML5 pridáva nový atribút vyhradený práve za účelom uchovávania týchto vlastných údajov, pre ktoré nie je k dispozícii vhodnejší atribút alebo element. Počet priradených atribútov k elementu nie je obmedzený.

Použitie vidieť na nasledujúcom príklade:

```
<article data-time="45" data-popis="Interval" id="ukazka">00:00:45 |
Interval</article>
```

Časť za "`data-`" udáva názov dát a reťazec medzi " " udáva hodnotu. Prístup k takto uchovaným dátam je natívny pomocou vlastnosti objektu `dataset` vo všetkých prehliadačoch okrem Internet Explorer, kde prístup k dátam treba riešiť metódami `getAttribute()` a `setAttribute()`.

```
var blok = document.getElementById('ukazka');

//natívny prístup
popis = blok.dataset.popis; //čítanie
blok.dataset.popis = 'iny popis'; //zápis

//náhradné riešenie
popis = blok.getAttribute('data-popis'); //čítanie
blok.setAttribute('data-popis', 'iny popis'); //zápis
```

3 HTML5 API

Ako sme v úvode naznačili, špecifikácia HTML5 zahŕňa okrem statického opisu obsahu dokumentov aj rôzne API (Application Programming Interface) na manipuláciu s objektmi v DOM modeli a ich využitie na tvorbu interaktívnych webových aplikácii.

V čase písania práce bolo k dispozícii dovedna 61 špecifikácii popisujúcich API využiteľné k tvorbe webových aplikácií pre stolné aj mobilné zariadenia. V tabuľke (Tab. 3-1) uvádzame prehľad API ktoré sú opísané v špecifikácii HTML5 a pár tých, ktoré úzko súvisia s HTML5, no sú opísane vo vlastných špecifikáciách.

Tab. 3-1Prehľad API

Špecifikácia HTML5	Vlastné špecifikácie
MediaElement API	SVG
Text track API	File API
History	Web GL
Location	HTML Media Capture
Applicaton cache	MathML
Drag and Drop	Indexované DB
Web workers	Web Cryptography API
Server-sent events	Geolocation
Web sockets	Canvas 2D
Cross-document messaging	...
Web strorage	
Canvas 2D	

3.1 MediaElement API

Neodmysliteľnou súčasťou webových stránok sú dnes už nepochybne videá a hudobné súbory. Pred uvedením HTML5 boli tieto multimediálne prvky vkladané do stránok pomocou zásuvných modulov softvérových dodávateľov v elemente `<embed>`. Najznámejší dodávatelia sú Adobe (Flash), QuickTime, RealPlayer a Silverlight. Ich

zásuvné moduly však podliehajú licenciám. HTML5 natívne podporuje vkladanie videa a zvukových súborov do obsahu pomocou nových značiek `<video>` a `<audio>` čím do istej miery odstraňujú závislosť od zásuvných modulov a ich implementácia je zrozumiteľná a jednoduchá. Každá minca má však dve strany. So zavádzaním do praxe to je s novými elementmi komplikovanejšie. Neexistuje priama kontrola nad obsahom z pohľadu autorských práv, pretože zdrojové súbory sú uvedené priamo v kóde. Tento problém sa dá odstrániť napríklad dynamickým pridávaním zdroja média do kódu. Takisto možnosti manipulácie s videom nie sú rozvinuté na takej úrovni ako už niekoľko rokov vyvíjané zásuvné moduly.

3.1.1 Princíp

Elementy `<video>` a `<audio>` pracujú s kontajnermi. Kontajnery sú obalom dátových tokov, zvukových alebo obrazových, a ďalších meta-dát ako sú informácie o interpretovi, názov diela či titulky. Dátové toky sú v surovom stave veľmi veľké a preto sa na zmenšenie ich veľkosti používajú rôzne kódeky. Prehrávače potom musia pred samotným prehratím dátové toky najprv dekódovať. Existujú desiatky kódekov, ktoré kladú rozdielne nároky na dekódovací výkon. HTML5 využíva pre video H.264,Theora, VP8 a pre zvuk ACC, OGG a MP3. Kombinácie týchto kódekov sú použité v troch kontajneroch, ktoré sa líšia podporou prehliadačov.

H.264 je kódek vytvorený skupinou MPEG určený pre kompresiu videa s vysokou kvalitou. Špecifikácia je rozdelená na viac profilov, ktoré sú podľa kvality určené pre iné koncové použitie. Jeho použitie na kódovanie je spoplatnené licenčnými poplatkami. Práva vlastní Google, Microsoft aj Apple.

Priaznivci otvorených riešení, konkrétne organizácia Xiph.Org Foundation, vyvinuli kódek *Theora*. Dosahuje podobnú kvalitu ako H.264, jeho použitie je bezplatné, nie je však široko rozšírený.

Spoločnosť Google vyvinula otvorený kódek *VP8*, ktorý je kvalitou podobný H.264. Podporuje ho okrem väčšiny prehliadačov aj Adobe Flash Player, no nie je podporovaný spoločnosťou Apple a ich produktmi.

Rovnaké problémy so širokou použiteľnosťou sú aj medzi zvukovými kódekmi. Advanced Audio Coding (AAC) od firmy Apple je kvalitou a možnosťou viacerých profilov podobný kódeku H.264. Ponúka vyššiu kvalitu zvuku pri rovnakej veľkosti súboru ako MP3, ale pri použití podlieha licenčným poplatkom. Zástupca otvorených

riešení medzi zvukovými kódekmi je Vorbis (OGG), ktorý je používaný v kombinácii s obrazovými kódekmi Theora a VP8. Jeho slabou stránkou je malá hardvérová podpora. Najpopulárnejším kódekom je MP3, ale keďže aj naň sa vzťahujú patenty, takisto nie je podporovaný všetkými prehliadačmi.

Vyústenie tejto komplikovanej situácie prináša tieto tri mediálne kontajnery:

- MP4 - video kóduje pomocou H.264 a zvuk kodekom AAC
- WebM - video kóduje pomocou VP8 a zvuk kodekom Vorbis
- Ogg - video kóduje pomocou Theora a zvuk kodekom Vorbis

Tab. 3-2 Tabuľka podpory kontajnerov prehliadačmi (zdroj: caniuse.com)

prehliadač \ kontajner	MP4	WebM	Ogg	MP3	AAC	Ogg
Internet Explorer	✓9+	×	×	✓	×	×
Firefox	4.0+	✓	✓	×	✓	✓
Google Chrome	✓	✓	✓	✓	✓	✓
Apple Safari 5+	✓	×	×	✓	✓	×
Opera 10.6+	×	✓	✓	×	✓	✓
iOS Safari	✓	×	×	✓	✓	×
Android Browser	✓	2. 3+	×	✓	✓	×
Blackberry Browser	✓	×	×	ak odkaz na stiahnutie		

3.1.2 Video

Video sa do stránky vkladá medzi značky `<video>` a `</video>`. Pomocou atribútov `height` a `width` sa nastavia rozmery oblasti prehrávania, no na pomer strán videa to nemá žiaden vplyv. Video sa zarovná na stred definovanej oblasti a nezdeformuje sa. Podporované hodnoty rozmerov sú celé čísla veľkosti *integer* v obrazových bodoch (px).

K prehrávaniu je možné pridať základné ovládacie prvky ako sú prehrať, zastaviť, stíšiť alebo ukazovateľ polohy stopy. Na zapnutie tohto jednoduchého natívneho ovládania slúži boolean atribút `controls`.

Pre odstránenie závislosti od prehliadača treba vložiť viac zdrojov videa, ktoré sa vkladajú pomocou značiek `<source>` a definíciu kontajnerového formátu atribútom `type`, ktorý obsahuje typ kontajnera a použitý obrazový a zvukový kódek. Pre

užívateľov, ktorý používajú prehliadač nepodporujúci element `<video>` je slušné vložiť odkaz na stiahnutie, aby si súbor mohli prehrať mimo prehliadača v prostredí operačného systému.

```
<video width=“320“ height=“240“>
       <source src=“video01.webm“ type='video/webm; codecs=”vp8, vorbis” '>
       <source src=“video01.mp4“ type='video/mp4; codecs=”avc1.42E01E, mp4a.40.2”'>
       <source src=“video01.ogv“ type='video/ogg; codecs=”theora, vorbis”'>
       <p>Váš prehliadač nepodporuje element video.</p>
       <a href="video01.mp4">Stiahnuť video súbor</a>
</video> <!-- poradie <source> som testoval, a toto funguje bezproblémovo -->
```

Ostatné vlastnosti prehrávania sa nastavujú nasledujúcimi atribútmi:

`autoplay` video sa prehrá hneď ako je dostupné. Treba však myslieť na to, že podobne ako atribút `autofocus` vo formulároch, nemusí používateľovi vyhovovať a samovoľné spustenie videa môže byť otravné. Špeciálne v prípade, ak je vo videu použité hlasné audio.

`muted` video sa prehrá bez zvukovej stopy. Zvuk sa dá povoliť ovládacím prvkom.

`loop` po ukončení prehrávania sa video začne prehrávať od začiatku až kým ho užívateľ nezastaví alebo neopustí stránku.

`poster` statická ukážka obsahu. Snímok z videa alebo obrázok vytvorený pre lepšiu orientáciu medzi viacerými videami. Atribút `src` obsahuje cestu k obrázku.

`preload` Atribút má tri hodnoty, ktoré upravujú funkciu. Hodnota `auto` načíta videá a príslušné meta-dáta hneď ako je to možné. Načítanie by nemalo spomaliť chod zvyšku stránky. Hodnota `none` zabráni načítaniu videa aj meta-dát a až po vyžiadaní prehrávania od užívateľa povolí načítanie obsahu. Hodnota `metadata` povolí načítanie `poster` obrázka a informácii o veľkosti videa a kódekoch.

Na vloženie videa, ktoré sa má opakovane prehrávať stíšené po spustení užívateľom, no zobraziť informácie pred spustením slúži tento zápis:

```
<video poster="nahlad_01.jpg" preload="metadata" loop muted controls>
      <source src="video01.webm" type="video/webm">
      <source src="video01.mp4" type="video/mp4">
</video>
```

3.1.3 Text Track

Podobne ako v stolných video prehrávačoch, ani v HTML5 nechýba podpora titulkov a nadpisov videa. Natívne prehrávače v prehliadačoch ich zatiaľ prehrávajú len v experimentálnych verziách, no pokročilé JavaScriptové knižnice pre prehrávače vedia zápis prečítať pomocou TextTrack API[7] a pracovať s vloženými údajmi.

Do videa sa externé stopy pridávajú vnoreným atribútom `<track>`, ktorý má nasledujúce štyri parametre:

src cesta k zdroju súboru; podporovaný je formát Web Video Text Track .vtt

srclang jazyk obsahu súboru

label názov v rámci kódu pre identifikáciu stopy

kind druh stopy (titulky, nadpisy, kapitoly, popisy, meta-dáta)

```
<track src="titulky.vtt" srclang="SK" label="titulkySK" kind="subtitles">
```

Web Video Text Track je štandardizovaný formát, ktorý vychádza z formátu `.srt`, no môže obsahovať aj popisy, nadpisy a meta-dáta čo ho robí univerzálnym v rámci HTML. Príklad zápisu titulkov v súbore WebVTT:

```
WEBVTT
                                        medzera oddeľujúca stopy
1                                       číslo stopy
00:00:05.000 --> 00:00:15.150           časový interval zobrazenia
Ahoj, nepočul som ťa prichádzať.        text, podporuje html formátovanie
```

3.1.4 Audio

Vkladanie zvukových súborov je rovnako jednoduché ako vkladanie video súborov, ale používajú sa značky `<audio>` a `</audio>`. Možnosti prehrávania sa dajú prispôsobiť atribútmi `autoplay`, `loop` a `preload` rovnako ako pri videu. Pre väčšiu kompatibilitu v prehliadačoch sa tiež odporúča použitie viac kontajnerových formátov.

```
<audio>
      <source src="zvuky/heavy_riff03.ogg" type="audio/ogg">
      <source src="zvuky/heavy_riff03.mp3" type="audio/mpeg">
      <p>Váš prehliadač nepodporuje element audio.</p>
      <a href="zvuky/heavy_riff03.mp3 ">Stiahnuť zvukový súbor</a>
</audio>
```

[7] http://developers.whatwg.org/the-video-element.html#text-track-api

Výhodou používania HTML5 videa/audia oproti vkladaniu cez zásuvný modul je, že video je súčasťou webu čo dovoľuje upravovať elementy a jeho komponenty pomocou JavaScriptu a CSS. Naproti tomu fakt, že každý prehliadač používa vlastnú sadu ovládacích prvkov môže narúšať vzhľad stránky. Tento problém odstraňuje kombinácia metód z MediaElement API, objektov médií z DOM a CSS, ktorými je možné vytvoriť vlastné ovládacie prvky a spojiť ich s videom. Prehľad možností prístupných cez API je na stránke W3C[8]. V súčasnosti už existuje mnoho hotových riešení, ktoré sa len vkladajú do stránok spolu s potrebnými JavaSkript knižnicami (http://mediaelementjs.com/) a poskytujú spätnú podporu pre staršie prehliadače.

3.2 Application cache.

S možnosťou pracovať bez pripojenia na internet sa otvárajú nové možnosti pri tvorbe webových aplikácii. Pri požiadavke na prehliadač, aby zobrazil stránku alebo spustil aplikáciu prehliadač komunikuje som serverom, načíta obsah pomocou internetového spojenia a zobrazí ho. Aplikačná cache (úložisko) umožní, že vybrané časti stránky alebo aplikácie budú fungovať užívateľom aj bez pripojenia na internet, čo je značnou výhodou najmä pre mobilné zariadenia, ktorých pripojenie je často obmedzené dátovým prenosom.

K tomu aby stránka alebo aplikácia fungovala off-line stačí vyplniť manifest (prehlásenie), ktorý prehliadaču povie, ktoré časti si má uložiť do úložiska, a umiestniť ho na webový server spolu so zvyškom stránky alebo aplikácie. Do manifestu vieme zapísať, ktoré súbory sa majú zálohovať, ktoré sa nemajú, a ktoré majú byť načítané zo servera pri dostupnom pripojení k internetu, no v prípade výpadku spojenia majú byť použité zálohované. Dovoľuje uložiť celú stránku vrátane všetkých zdrojov (HTML, CSS, skripty, médiá). Príkladom zálohy je náhradná stránka, ktorá sa užívateľovi ukáže pri pokuse o pripojenie do komunikačnej služby v režime off-line. O sledovanie stavu súborov sa stará objekt `applicationCache`, ktorý obsahuje metódy na správu dát z úložiska.

[8] http://www.w3.org/2010/05/video/mediaevents.html

Prvým krokom je vytvorenie manifestu. Je to súbor s ľubovoľným názvom typu `.appcache`, ktorého štruktúra je pevne daná. Súbor sa umiestni kdekoľvek na server a odkáže sa na jeho umiestnenie v značke `<html>` každého html súboru na servreri.

Podmienkou je, aby server poznal MIME typ *appcache*. Ak ho nepozná, treba ho pridať. Napríklad na server založený na Apache sa pridá vložením nasledovného riadku do súboru `.htaccess`:

```
AddType text/cache-manifest .manifest
AddType text/cache-manifest .appcache
```

Prehliadač sa pri prvej návšteve opýta, či môže uložiť súbory stránky alebo aplikácie do aplikačnej pamäte. Nie je podporovaná v Internet Explorer 9 a nižšie.

štruktúra súboru **.appcache* (# zakomentuje riadok) :

```
CACHE MANIFEST
#verzia 1.32 27.11.2012 - verzia manifestu (voliteľné)

CACHE:
#cesty súborov ktoré sa majú uložiť do úložiska

NETWORK:
#cesty súborov ktoré sa nemajú uložiť do úložiska
#znak * načíta všetky súbory, ktoré nie sú definované v iných častiach

FALLBACK:
#cesty súborov ktoré sa majú zobraziť v prípade on-line nedostupnosti
#skladá sa z 2 parametrov: originál{medzera}náhrada
#napr: images/online.jpeg images/offline.jpeg
```

Cesty k súborom sa uvádzajú vo vzťahu k umiestneniu manifestu (*.appcache)

Pridanie manifestu do html súborov:

```
<!DOCTYPE html>
<html manifest="/medzipamat.manifest">
...
</html>
```

Aby to nebolo také jednoduché, prehliadače umožňujú len obmedzenú dátovú kapacitu na uloženie. Veľkosti sa líšia v závislosti na prehliadači, no minimum, ktoré poskytuje každý prehliadač je 5MB. Veľké súbory teda nie je efektívne označovať na ukladanie, ale je lepšie pre ne použiť `FALLBACK` riešenie so správou o nedostupnosti.

Vlastnosť objektu `window applicationCache.status` z API vráti jeden zo šiestich stavov cache ako číselnú hodnotu (0-5) porovnaním starej a novej verzie manifestu. Hodnota 4 znamená že je dostupná aktualizácia. Aktualizáciu vykonáva prehliadač vždy pri načítaní alebo obnovení stránky, no len v prípade, že sa obsah manifestu zmenil. Pri

zmene obsahu stránky je preto vhodné zmeniť verziu v manifeste aby sa pri načítaná stránky spustila aktualizácia, alebo aplikácia môže zavolať vždy pri spustení metódu `swapCache()`, ktorou manuálne požiada prehliadač o aktualizáciu úložiska.

3.3 Web storage

Web Storage je API na lokálne ukladanie jednoduchých dát vo formáte kľúč-hodnota vo webovom prehliadači podobne ako cookies, no s menšími bezpečnostnými rizikami.

Cookies sú malé dátové súbory ukladajúce sa do zariadenia užívateľa. Obsahujú dáta v pároch kľúč-hodnota (meno, Janko), ktoré pri komunikácii so serverom upravujú chovanie stránky podľa užívateľa. Napríklad sú to hodnoty pre automatické vypĺňanie formulárových polí. Okrem týchto informácii obsahujú informácie o dobe platnosti, ceste k súboru a názov domény, pre ktorú je platná. Cookies sú posielané v každej HTTP požiadavke čo z nich robí ľahký cieľ zneužitia tretími stranami odchytávaním informácií v prípade nezabezpečeného spojenia.

Rozdiel medzi Web Storage (miestneho úložiska) a Cookies je spôsob akým sú využívané. Informácie, ktoré v sebe uchovávajú neslúžia pre server, ale sú dostupné len pre prehliadač. Na komunikáciu so serverom je stále nutné používať Cookies. Zatiaľ čo Cookies majú vyhradené 4KB miesta, Web Storage umožňuje použiť priestor až 5MB.

Web Storage ponúka dve možnosti uloženia dát - `sessionStorage` a `localStorage`. Dáta vytvorené v `sessionStorage` sú dostupné len po dobu relácie, čiže po dobu prístupu na stránke. Po zavretí okna, t.j. ukončení relácie sa dáta zmažú. Zabráni sa prenikaniu dát medzi oknami, čo môže spôsobiť nepríjemnosti typu viacnásobná objednávka. Výnimkou sú vyskakovacie okná z okna relácie, ktoré môžu dediť dočasne uložené údaje.

`LocalStorage` sprístupňuje údaje aj medzi reláciami, pretože ich ukladá na disk a odstraňuje ich len na pokyn užívateľa alebo kódu v stránke. Uložené údaje sú tak ako Cookies uložené pre každý prehliadač osobitne, takže údaje z Google Chrome nie sú dostupné napr. v prehliadači Opera.

Na zápis, čítanie a mazanie slúžia nasledujúce metódy:

```
localStorage.setItem(' názov kľúča ', ' hodnota '); //zápis
localStorage.getItem(' názov kľúča ' ); //čítanie
```

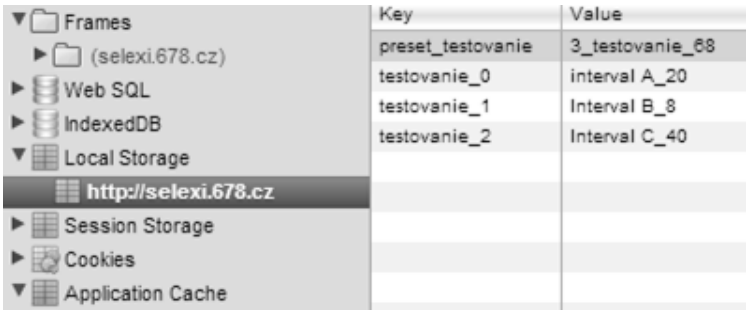
```
localStorage.removeItem(' názov kľúča '); //vymaže konkrétny kľúč
localStorage.clear(); //vymaže celú pamäť

sessionStorage.setItem(' názov kľúča ', ' hodnota '); //sessionStorage
```

Metóda `getItem()` spracúva údaje v dátovom type *string*. Pred číselnou manipuláciou je nutné údaje pretransformovať na číselný dátový typ, napr. *integer*:

```
var vyska = parseInt(localStorage.getItem("vyska"));
```

Zápis do úložiska je možný aj skrátenou formou priamym zápisom hodnoty ku kľúču, ale je to podmienené existenciou kľúča pred zápisom. Vstupná hodnota je tiež *string*. Nasledujúca skrátená forma je vhodná na aktualizáciu už existujúcich záznamov:

```
localStorage.testovanie_2 = 'Interval C_40';
```



Obr. 3.1 Prehliadanie obsahu localStorage v režime pre vývojárov (Chrome)

Čo sa bezpečnosti týka, prehliadač sprístupní úložisko len pre požiadavky prichádzajúce z rovnakej domény ako stránka alebo aplikácia. Pri možnej hrozbe falošnej adresy pôvodcu prehliadač nevie overiť, že požiadavka skutočne prichádza z rovnakej domény. Na Obr. 3.1 je vidieť dáta uložené pre doménu *http://selexi.678.cz*.

3.4 Canvas

Element `<canvas>` je virtuálne plátno vo vnútri html dokumentu, do ktorého je pomocou Canvas 2D API možné kresliť rôzne tvary a text. Obsah je rastrový - každý vykreslený bod (pixel) má presne určenú polohu, farbu a priehľadnosť. Veľkosť obrázka závisí od rozlíšenia zariadenia a pri zväčšovaní dochádza k zníženiu kvality.

Pred začiatkom kreslenia je nutné definovať rozmery plátna pomocou atribútov `height` a `width` a priradiť plátný identifikátor `id`. Pre orientáciu, no hlavne pre

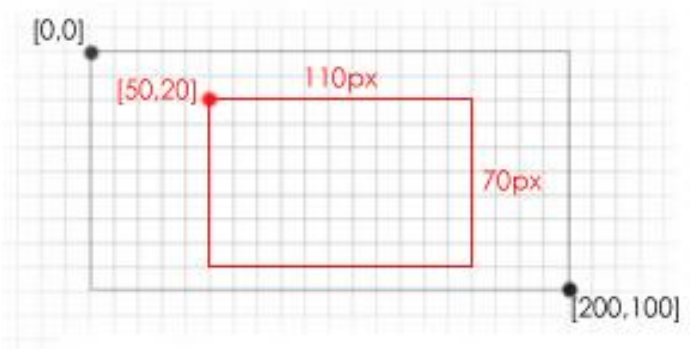
vytváranie obsahu plátna slúži súradnicový systém X,Y s počiatkom [0,0] v ľavom hornom rohu a rozmermi definovanými v značke elementu. Pomocou identifikátora pristupujeme k objektu elementu `<canvas>` v DOM strome a následne k jeho kontextu, ktorý obsahuje metódy na kreslenie a zobrazenie obsahu.

Implementácia prázdneho plátna o rozmeroch 200x100 px do dokumentu a priradenie 2d kontextu:

```
<canvas width="200px" height="100px" id="platno">
	<!--náhradné riešenie-->
	Váš prehliadač nepodporuje element &lt;canvasgt;
</canvas>
<script>
		//inicializácia plátna
	var canvas = document.getElementById("platno");
		//priradenie kontextu
	var kontext = canvas.getContext("2d");
</script>
```

Na vykreslenie čierneho obrysu obdĺžnika ako na obrázku nižšie by sme použili nasledujúcu vlastnosť a metódu:

```
kontext.strokeStyle = 'red'; //farba orámovania
kontext.strokeRect(50, 20, 110, 70); //x,y,sirka,vyska ->orámovanie
```



Obr. 3.2 súradnicový systém canvas

Na rozdiel od SVG, `<canvas>` možno uložiť ako obrázok použitím nasledujúcej funkcie, ktorá ho po kliknutí na element `canvas` otvorí ako rastrový obrázok:

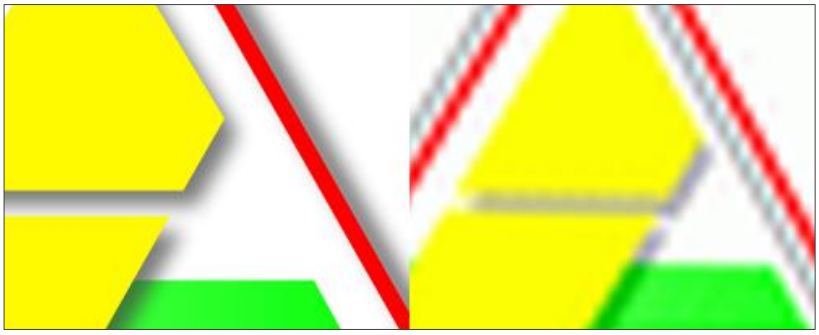
```
canvas.onclick = function () {
	window.location = canvas.toDataURL('image/png');
};
```

Špecifikácia API[9] je rozsiahla a ponúka veľa možností na tvorbu grafiky a na manipuláciu s existujúcou grafikou v dokumente. Záverom ešte poznamenáme, že pri animovaní je nutné nové snímky kompletne vykresliť, takže s rastúcim rozlíšením plátna narastá čas vykreslenia. Je vhodný na komplexné, matematicky náročné animácie, na manipuláciu s videom/obrázkami v reálnom čase v podobe rôznych filtrov.

3.5 SVG

Scalable Vector Graphic (SVG) je formát, ktorý umožňuje opísať vektorovú grafiku pomocou XML. Jeho sila je v zachovaní kvality obrazu pri zmene veľkosti. Podobne ako Canvas 2D podporuje kreslenie tvarov, prechodov, vzorov a prostredníctvom DOM aj animácie. Špecifikácia SVG[10] je udržiavaná konzorciom W3C, pričom v súčasnosti pracujú na rozšírenej verzii SVG2.

Do html dokumentov sa pred HTML5 vkladal len cez elementy `<embed>`,`<object>` a `<iframe>`. V HTML5 pribudol element `<svg>`, ktorý umožňuje aj prácu s grafikou v rámci kódu. Pre statické vkladanie do dokumentu bez podpory skriptovania je stále dostupný cez `<object>`, pre nové prehliadače (IE 9+) cez `<svg>`. Podpora základných filtrov a animácii je podľa CanIuse[11] dostupná všade okrem IE.



Obr. 3.3 Porovnanie kvality SVG/Canvas 2D grafiky pri zmene pôvodného rozmeru

[9] http://developers.whatwg.org/the-canvas-element.html

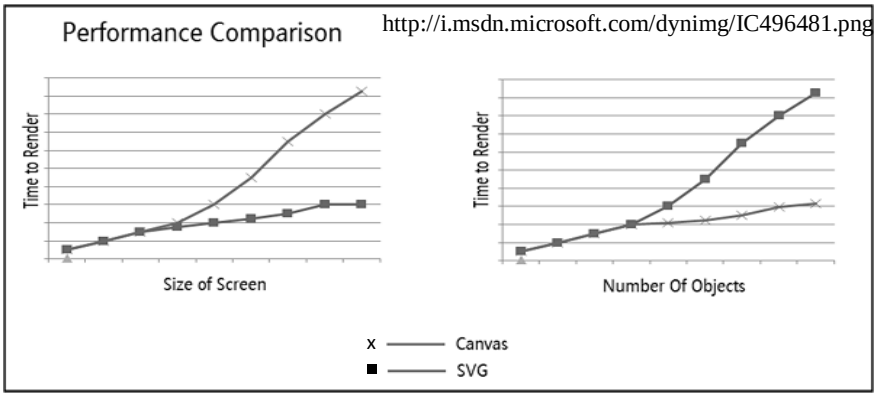
[10] http://www.w3.org/TR/SVG11/

[11] http://caniuse.com/#search=svg

Pre porovnanie s efektivitou zápisu `canvas`, stačí na vykreslenie obdĺžnika tento jednoduchý kód:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <rect x="50" y="20" width="110" height="70"
      style="fill:none;stroke:red;stroke-width:1;"/>
</svg>
```

Napriek tomu, že SVG nepatrí do špecifikácie HTML5, odjakživa je veľmi príbuzná s HTML. Spolu s Canvas 2D tvoria potrebnú sadu nástrojov na tvorbu interaktívnych prvkov stránok a aplikácii. Nie sú však vhodné na každé použitie, čoho dôkazom je graf (obrázok 3.4) z článku[12], v ktorom autori testovali rôzne implementácie Canvas a SVG. Z výsledkov testu je vidieť, že výpočtová náročnosť a jej úmerná doba vykreslenia je väčšia u Canvas, no pri komplexnej kompozícii je pomalší SVG.



Obr. 3.4 Porovnanie výkonu Canvas 2D a SVG

3.6 Drag and Drop

Pred HTML5 sa funkcie "potiahni a pusť", tak ako ich poznáme z prostredia operačného systému, dali implementovať pomocou knižníc jQuery, MooTools alebo Dojo. HTML5 natívne umožňuje presúvať či prenášať dáta vo vnútri prehliadača. Špecifikácia definuje mechanizmus založený na udalostiach (events), JavaScript API a príslušné atribúty na označenie elementov v zápise. Zdrojový element môže byť každá časť stránky, napríklad obrázok, súbor alebo odkaz. Obrázky, odkazy a text je možné ťahať štandardne. U ostatných elementov to umožňuje atribút `draggable` s hodnotou *`true`*; hodnota *`false`* naopak zakazuje ťahanie objektu.

[12] http://msdn.microsoft.com/en-us/library/ie/gg193983(v=vs.85).aspx

Cieľová oblasť pretiahnutia, ktorá prijíma ťahané objekty nemôže byť akýkoľvek element (napr. obrázok), spravidla je do `<div>` alebo formulárové pole. Správanie pri ťahaní a púšťaní je ďalej opísane udalosťami a JavaScriptom, ktorému sa nedá vyhnúť ani pri vytvorení len základnej funkcionality v stránke, minimálne do času, kým prehliadače neimplementujú atribút `dropzone`, ktorý charakterizuje akciu, ktorá sa má vykonať pri pustení ťahaného objektu v cieľovej oblasti (*`copy, move, link`*).

Udalosti môžeme rozdeliť do dvoch skupín:

udalosti na ťahanom elemente: `dragstart, drag, dragend`

udalosti oblasti pretiahnutia: `dragenter, dragleave, draghover, drop`

Počas ťahania sú dáta dočasne uložené v objekte `dataTransfer`, podobne ako pri vystrihnutí súborov v operačnom systéme Windows sa súbory uložia do zásobníka, kde čakajú na ďalšie spracovanie. Naraz môže držať viacero objektov rôznych rovnakých aj rôznych dátových typov. Implementácia v prehliadačoch Internet Explorer je vo verziách 8 a 9 paradoxne obmedzená, aj napriek tomu, že táto funkcia korení práve v starých verziách Internet Explorer. V ostatných stolných prehliadačoch je Drag and Drop kompletne podporovaný. Na dotykových mobilných zariadeniach nie je podporovaný. V kombinácii s File API[13] je možné pracovať aj so súbormi z operačného systému.

3.7 Ostatné API

Podrobnejšie priblížiť všetky API nie je z dôvodu rozsahu možné. V nasledujúcej časti preto len stručne priblížime niekoľko ďalších rozhraní.

3.7.1 History

Pri zmene adresy sa v histórii prehliadača vytvára záznam, takže navigačnými tlačidlami prehliadača sa užívateľ prepína medzi adresami, respektíve stavmi stránok. Metódy obsiahnuté v History API umožňujú manipulovať so stavom stránky, takže aj dynamicky meniaci sa obsah stránky/aplikácie sa na základe akcií užívateľa môže zaznamenávať ako nový záznam stavu v histórii. Umožní to návrat späť bez straty dovtedy vykonaných zmien.

[13] http://www.w3.org/TR/FileAPI/

Zmena obsahu pri stránke komunikujúcej so serverom vyvolá pri zmene informácii kompletné prekreslenie okna prehliadača a prenosu kompletnej stránky z webového servera. S použitím metód History API prehliadač, respektíve aplikácia, požiada webový server len o časť stránky, ktorá by sa pri vykonaní akcie (po stlačení tlačidla/kliknutí na odkaz) vykonala bez načítania a prekreslenia celého obsahu. Využitie je rôznorodé od formulárových polí až po hry aplikácie.

3.7.2 Web Workers

Web Workers ("webový robotníci") zohrávajú kľúčovú úlohu pri tvorbe komplexných webových aplikácii. Umožňujú vykonávanie časti JavaScriptového kódu na pozadí paralelne s hlavným oknom prehliadača, čím vlastne predstavujú procesné vlákno. Znamená to, že vykonávaný skript neovplyvňuje výkon a správanie aplikácie. Typickým príkladom môže byť aplikovanie filtra v elemente `<canvas>`. Výpočet trvá niekoľko sekúnd, počas ktorých by bol chod aplikácie spomalený alebo pozastavený. Tiež by nebolo možné proces prerušiť. Web Workers tento proces v tichosti obslúžia na pozadí a výsledok, v tomto prípade filtrovaný raster, vrátia hlavnej aplikácii.

Práva k obsahu Web Workers sú limitované obmedzeným prístupom k DOM stromu aplikácie, preto sa všetky vstupné dáta musia poslať robotníkom pred začatím spracúvania. Komunikácia s hlavnou aplikáciou je výlučne pomocou správ (`.postMessage();`). Viac informácii a prehľad možností je v špecifikácii[14].

3.7.3 Web Sockets

Web Sockets je rozhranie, ktoré definuje obojsmernú plne duplexnú komunikáciu formou správ vymieňaných medzi klientom a serverom v reálnom čase. Pracuje podobne ako TCP/IP. Na rozdiel od iných metód (HTTP polling) je komunikácia po vytvorení spojenia možná kedykoľvek až do ukončenia spojenia, nie len na výzvu zo strany servera (polling). Pretože strany neustále počúvajú a netreba posielať okrem správ ďalšie nadbytočné hlavičky, spojenie má veľmi nízku odozvu a záťaž na pripojenie. Nadviazanie spojenia prebieha cez HTTP vytvorením tunela, ktorý obchádza proxy, no následná komunikácia využíva protokol WS alebo jeho zabezpečenú verziu

[14] http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html

WSS. Na to aby bolo možné komunikovať sú potrebné prehliadač a server podporujúce Web Sockets. Porty sú totožné ako pri HTTP (80) / HTTPS (443) protokole.

V súčasnosti existujú hotové riešenia, ktoré obsahujú okrem Web Sockets viacero komunikačných metód (Comet, AJAX). V prípade výpadku spojenia alebo použitia prehliadača bez podpory Web Sockets prepnú na náhradnú metódu komunikácie. Príkladmi využitia je úprava dokumentu viacerými užívateľmi súčasne bez preťaženia servera alebo vzniku nekonzistentného obsahu, alebo webová hra pre viac hráčov.

3.7.4 Web Messaging

V HTML4 nie je možné z bezpečnostných dôvodov obsah stránok z jednej domény poslať alebo poskytnúť stránke z inej domény. Zabraňuje to zámene a odchytávaniu posielaných dát treťou stranou počas prenosu, no na druhej strane to zabraňuje cielenej výmene dát medzi dvomi dôveryhodnými doménami. Mnoho webových aplikácii využíva informácie z viacerých subdomén alebo domén (Facebook, Twitter), no pre zabezpečenie týchto dátových prenosov musia využívať zložité triky, ktoré odhaľujú potenciálne slabiny.

HTML5 prináša `postMessage()` API, ktoré vytvára rozhranie pre výmenu dát skriptov bežiacich na odlišných doménach pomocou vlastností `data`, `origin`, `lastEventId`, `source a ports` udalosti `message` na strane prijímateľa. K získaniu pôvodu odosielateľa slúži vlastnosť `origin`, nesúca URL adresu odosielateľa. Následné overenie či sa skutočne jedná o uvedeného odosielateľa už musí zabezpečiť vývojár. Samotné HTML5 toto overenie k výmene správ nevyžaduje. Messaging API využíva aj už spomínané rozhranie Web Workers.

3.7.5 WebGL

WebGL je multiplatformový otvorený webový štandard pre nízkoúrovňové grafické API založené na OpenGL ES 2.0 implementovaný do HTML5 pomocou elementu `<canvas>`, natívne bez použitia zásuvných modulov. Niekedy sa označuje aj ako Canvas 3D. Vyvíja ho tím Khronos Group[15] spolu s pracovnou skupinou WebGL zahŕňajúcou spoločnosti Apple, Google, Mozilla a Opera. Umožňuje vytvárať pôsobivé priestorové vizualizácie, hry a aplikácie.

[15] http://www.khronos.org/webgl/

4 BUDÚCNOSŤ HTML

Už v úvodnej časti sme spomenuli, že prehliadače nepokrývajú všetky body špecifikácie, a tak sa ľahko stane, že funkcie, ktoré bez problémov fungujú v niektorých prehliadačoch nefungujú v iných. S čistým svedomím môžeme konštatovať, že všetky najnovšie verzie stolných a mobilných prehliadačov podporujú, značnú časť špecifikácie. Za predpokladu, že by užívatelia pravidelne aktualizovali svoj obľúbený prehliadač na najnovšiu verziu, problémy spojené s ošetrovaním správania nepodporovaných funkcií HTML5 a pridružených technológii náhradnými riešeniami by odpadli.

4.1 Podpora HTML5 v starších prehliadačoch

Realita je však úplne niekde inde, a tak treba pri tvorbe webových aplikácii v prvom rade myslieť na koncových užívateľov, ktorý používajú nekompatibilný prehliadač alebo majú z nejakého dôvodu vypnutý JavaScript. Do istej miery si vývojári poradili aj s týmto problémom, a to konkrétne pomocou náhradných riešení (takzvaných polyfill a shims), ktoré emulujú podporu nepodporovaných funkcií v prehliadačoch. Polyfill môže byť jednoduchá JavaScriptová knižnica, ktorá pridáva podporu CSS3 selektorov, ale takisto to môže byť prepracované riešenie založené na technológii FLASH, ktoré sprístupní elementy `<audio>` a `<video>` v prehliadačoch Internet Explorer 6.

Jedným z najpoužívanejších riešení je HTML5shiv[16], ktoré pridáva podporu nových štruktúrových elementov do starších prehliadačov (IE 6-9, Mozilla 3.x, Safari 4.x). V prípade, že je potreba viac než len podpora nových elementov, tvorcovia HTML5shiv pracujú aj na rozšírenej verzii nazvanej Modernizr[17]. Oproti HTML5shiv obsahuje navyše detekciu podpory jednotlivých funkcií HTML5 a CSS3, na základe ktorej sa v kóde dá odvolať na náhradné riešenie.

[16] https://code.google.com/p/html5shiv/

[17] http://modernizr.com/

Ešte komplexnejším riešením je Webshims[18], ktoré využíva Modernizr a jQuery. Webshims pridáva podporu štruktúrových elementov, formulárov, canvas, webstorage, geolocation a mediálnych elementov `<audio>` a `<video>`. Prednosťou tohto riešenia je automatická detekcia podpora funkcii a sprístupňovanie len tých nepodporovaných. Znamená to, že užívateľ dostane natívne správanie prednostne pred náhradným.

Použitie je veľmi jednoduché pridaním prepojení so skriptami do html súboru. Pred použitím len niektorých funkcii je možné si nakonfigurovať vlastný balík pre Modernizr.

```
<script src="js/jquery-1.8.2.min.js"></script>
<script src="js/modernizr-custom.js"></script>
<script src="js/webshim/polyfiller.js"></script>
```

Posledným krokom je zvoliť, ktoré funkcie má Webshim emulovať.

```
<script>
      // emulovať všetky nepodporované funkcie
      $.webshims.polyfill();
      // emulovať len formuláre a canvas
      $.webshims.polyfill('forms canvas');
</script>
```

V prípade, že sa náhradný kód (polyfill) odstráni, aplikácia stále pobeží v prehliadačoch podporujúcich hlavný kód aplikácie, takže to nenarúša budúcnosť stránky alebo aplikácie.

Pri tvorbe kompatibilnej aplikácie postopujeme v nasledujúcich krokoch:

1. Vytvoriť aplikáciu v súlade so špecifikáciami
2. Detegovať podporu použitých funkcii v prehliadači (Moderzniz, Webshims)
3. Ak prehliadač nepodporuje funkciu, pomôcť si "polyfill-om"

4.2 Bezpečnosť HTML5

S množstvom noviniek, ktoré prináša HTML5, vzrastá aj počet potenciálnych bezpečnostných slabín, ktoré hackerom otvárajú nové možnosti zneužitia informácií nič netušiaceho užívateľa. Okrem už spomínanej výmeny správ medzi rôznymi doménami

[18] http://afarkas.github.com/webshim/demos/

sú potenciálnou oblasťou útokov aj nové atribúty predovšetkým určené k obohateniu celkovej funkcionality stránok, ktoré môžu byť zneužité. Napríklad atribút `autofocus`, ktorý sme predstavili v kapitole 2.5.2, môže byť zneužitý k "ukradnutiu" pozornosti na okno nesúce podvodný obsah alebo spúšťajúce nebezpečný skript.

```
<input onfocus="vykonajNiecoZle()" autofocus>
```

HTML5 rovnako ako predošlé verzie nie je bezpečné riešenie. Aj napriek tomu, že obsahuje relatívne ošetrené funkcie, ich nesprávne použitie odkrýva útočníkom možnosti zneužitia. Bezpečnostné opatrenia (napr. používanie HTTPS protokolu čo najviac) a dôkladné testovanie aj naďalej zostávajú dôležitou súčasťou procesu tvorby stránok a aplikácií s využitím HTML. Ucelený prehľad bezpečnostných medzier[19] zahŕňa referenčnú vzorku kódov, ktorým sa treba pri tvorbe vyhnúť. Informácie o bezpečnosti HTML5, ktoré detailne zachytávajú každú funkciu jazyka vrátane API, sú dostupné na portáli html5security[20].

4.3 Smerovanie HTML

Napriek tomu, že HTML5 tu už pár rokov je, implementáciu je možné postrehnúť najviac u nových webov, pretože staršie stále používajú kompletné riešenia napísané v XHTML alebo HTML 4.01. Je to spôsobené tým, že HTML5 netvorí celok, ale je poskladaná z častí, ktoré len štandardizujú už tretími stranami vytvorené funkcie. Najväčším zastúpením trhu sú nepochybne webové aplikácie a redakčné systémy, ktoré s podporou chytrých mobilných zariadení tvoria ideálnu cieľovú skupinu.

W3C plánuje finálnu (odporúčanú) verziu HTML5 na posledný štvrťrok 2014, pričom v dobe písania tejto práce prebiehajú práce na odstraňovaní známych chýb a kompletizuje sa dokumentácia. Dostupná je aj prvá editorská verzia HTML 5.1 od W3C, ktorá by mala nasledovať "HTML 5.0" v roku 2016. Veríme, že s uvedením odporúčanej verzie sa ešte viac firiem začne preorientovávať z využívania modulov na kompletné HTML5 riešenia, pretože má skutočne silný potenciál a veľkú ponuku možností s neustálym rastom.

[19] http://html5sec.org/

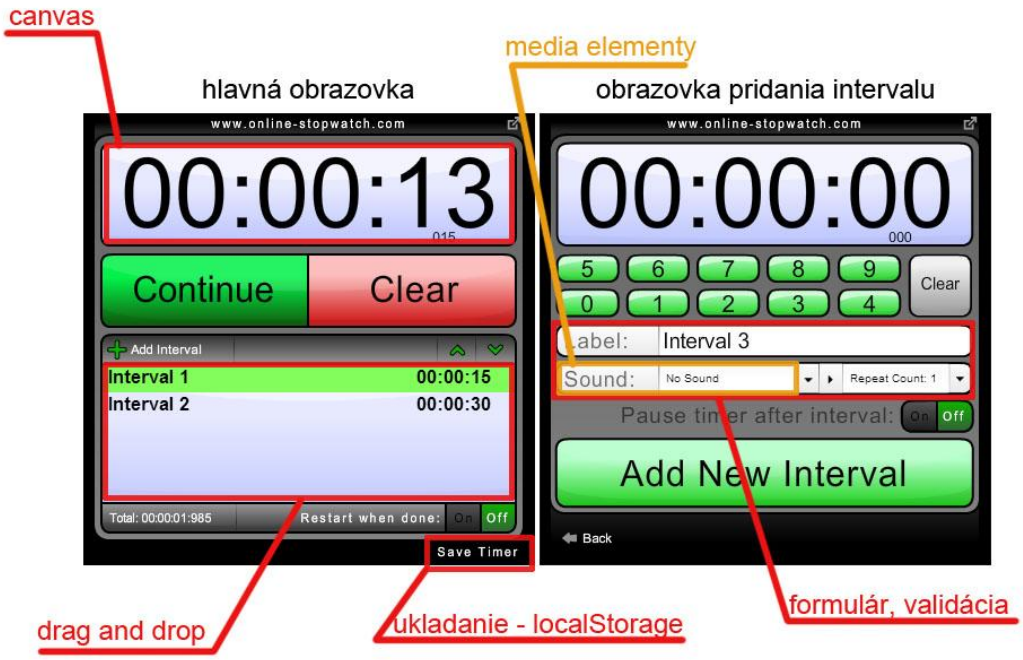
[20] http://html5security.org/

5 PRAKTICKÁ ČASŤ

Pri výbere obsahu praktickej časti sme zvažovali, ako najlepšie ukázať implementáciu HTML5 v praxi a zároveň jej ponechať edukačný charakter. Pre komplexnosť štandardu voči rozsahu práce ho nie je možné v praktickej časti celkom pokryť. Praktickú časť sme preto rozdelili na dva celky, ktoré spolu pokrývajú aktuálne najviac používané funkcie. Prvým celkom je jednoduchá webová aplikácia vytvorená pomocou HTML5 a druhú časť tvorí prehľad štruktúrových prvkov HTML5 s praktickými ukážkami a zdrojovými kódmi, ktorá môže slúžiť ako referencia, či doplnok pri výučbe HTML.

5.1 Analýza aplikácie

Pri samotnom návrhu aplikácie sme sa snažili, aby sme demonštrovali čo najviac z množiny technológií, ktoré obsahuje štandard HTML5. Inšpiráciou bola internetová aplikácia vytvorená v prostredí FLASH, ktorej prostredie je vidieť na Obr. 5.1. Aplikácia funguje ako počítač časových intervalov, ktorý možno využiť hlavne pri cvičení, no môže byť použitý na časovanie a signalizáciu prakticky ľubovoľnej činnosti.



Obr. 5.1 Analýza aplikácie (FLASH)

Pred začatím sme analýzou určili mechanizmy a funkcie použité v pôvodnej aplikácii a priradili sme im niektorú z dostupných ekvivalentných funkcií HTML5. Rozsah použitých funkcii reprezentatívne pokrýva záber možností HTML5. Zoznam použitých funkcií a vlastností s podporu vo webových prehliadačoch zahŕňa Tab. 5-1.

Tab. 5-1 Zoznam použitých funkcií a vlastností

HTML	IE	Firefox	Chrome	Safari	Opera	Android
sémantické elementy	+	+	+	+	+	+
canvas	+	+	+	+	+	+
audio element	+	+	+	+	+	2.3+
webstorage	+	+	+	+	+	+
offl-line uložisko	10+	+	+	+	+	+
drag & drop	základ	+	+	+	+	x
atribút data-	+	+	+	+	+	+
validácia formulárov	x	+	+	+	+	x
nové formulárové prvky	x	+	+	+	+	x
<input type="number">	x	x	+	+	+	+
atribút defer	+	+	+	+	x	3.0+
getElementsByClassName	+	+	+	+	+	+
atribút placeholder	10+	+	+	+	+	+
audio/mpeg (mp3)	+	x	+	+	x	+
audio/ogg	x	+	+	x	+	x
audio/wav	x	+	+	x	+	x
CSS	**IE**	**Firefox**	**Chrome**	**Safari**	**Opera**	**Android**
tiene (box shadow)	+	+	+	+	+	+
prechody (gradients)	10+	+	+	+	+	+
zaoblenie rohov (border	+	+	+	+	+	+
CSS3 selektory	+	+	+	+	+	+
priehľadnosť	+	+	+	+	+	+
classList	10+	+	+	+	+	+

5.2 Prostredie aplikácie

Základom aplikácie je štruktúra vytvorená novými štruktúrovými elementmi a prezentačná vrstva vytvorená s využitím CSS3 s použitím minimálneho počtu externých

obrázkov. Pri spustení aplikácia skontroluje, či už užívateľ predtým používal aplikáciu a uložil si predvoľby obsahujúce zoznamy intervalov do lokálnej pamäte (localStorage). V prípade, že sa nájdu záznamy, načítajú sa do bloku "predvoľby" a môžu byť použité v aktuálnej relácií alebo môžu byť zmazané. Predvoľby sú neprenosné medzi zariadeniami a prehliadačmi v rámci zariadenia. V prípade, že ide o prvé spustenie, prehliadač načíta príslušné súbory z manifestu do úložiska (application cache) a od toho momentu je aplikácia použiteľná aj bez pripojenia k internetu. Obsah manifestu je v Prílohe B.

Obr. 5.2 Popis blokov webovej aplikácie

Na Obr. 5.2 sú naznačené oblasti v prostredí aplikácie, ktoré bližšie popíšeme v nasledujúcej časti.

1. Informačný panel vytvorený pomocou elementu `<canvas>`. Obsahuje vizualizáciu zostávajúceho času pri odpočítavaní pozostávajúcu v ľavej časti z dvoch kruhov a v pravej z textových informácii. Vonkajší kruh reprezentuje celkový zostávajúci čas aktívnych intervalov a menší kruh reprezentuje aktuálne počítaný interval. Prekresľuje sa vždy pri zmene niektorej zobrazovanej informácie pomocou príslušných JavaScript API.
2. Na pridávanie nových intervalov do zoznamu slúži formulárové pole s validáciou, ktoré využíva nový vstupný typ "*number*" a atribúty `placeholder` a `required`. Overenie zaistí, že sa do zoznamu nepodarí vložiť nepomenovaný interval a v

prípade, že užívateľ zadá neplatné časové hodnoty nastavia sa v príslušných poliach nuly. Funkcie overenia sú v Prílohe D. Potvrdenie hodnôt polí je možné aj pomocou klávesy ENTER.

3. Po vložení nového intervalu alebo načítaní z pamäte sa intervaly vložia do tejto oblasti. Poradie intervalov je možné meniť pomocou Drag and Drop API. Každý interval nesie svoj časový údaj v atribúte `data-time` a názov v atribúte `data-popis`, ktoré uľahčujú výpočet času pri počítaní a ukladanie do predvolieb. Správanie pri udalostiach Drag and Drop je v Prílohe C.
4. Vizualizácia režimu aplikácie podľa stavu pripojenia k internetu (on-line/off-line) v závislosti na dostupnosti zdroja. Pri nedostupnom internetovom pripojení sa načíta grafika z aplikačného úložiska, no pri dostupnom pripojení sa načíta z webového servera.
5. Kontajner s predvoľbami, ktoré si užívateľ môže preniesť medzi použitiami aplikácie ich uložením v lokálnom úložisku (localStorage). Podmienkou vytvorenia je súčet času intervalov väčší ako 0 sekúnd a platný názov predvoľby vo vstupnom poli. Použitie predvoľby - pridanie intervalov do zoznamu - sa vykoná kliknutím na žlté tlačidlo.

Pred začatím počítania, ale aj počas už spusteného počítania užívateľ môže prepínať režim počtu cyklov opakovania medzi jednorazovým alebo nekonečným. Po dokončení počítania intervalu aj celého zoznamu sú pomocou Media API prehrané výstražné zvuky, ktoré informujú užívateľa v prípade, že sa nepozerá na aplikáciu alebo ju má v okne prehliadača minimalizovanú.

5.3 Testovanie funkčnosti aplikácie

Po výsledkoch testovaní v prehliadačoch sme museli použiť niekoľko náhradných riešení kvôli chýbajúcej podpore prehliadačov.

- Prehliadač Opera stále neimplementoval podporu atribútu `defer`, ktorý pozdrží načítanie externého skriptu, dokým nie je celá stránka načítaná. Pretože aplikácia pracuje s vytvoreným obsahom (tlačidlá, kontajnery) bolo nutné počkať s inicializáciou, kým nebude celý obsah načítaný. Namiesto umiestnenia celého skriptu do `window.onload()` sme to vyriešili presunutím pripojenia skriptu na koniec tela dokumentu.
- Prehliadač Internet Explorer nepozná vlastnosť `classList`, preto sme dynamické pridávanie a odoberanie tried riešili pridaním náhradného riešenia pomocou vlastnosti `className`. Pre podporujúce prehliadače sme ponechali riešenie s využitím spomínanej metódy.
- Funkcionalita Drag and Drop v prehliadači Internet Explorer 9 a nižšie je obmedzená na elementy `<img>`, text a odkazy. Keďže intervaly sú elementy `<article>`, nie je možné ich presúvať bez použitia náhradných riešení tretích strán.

Testovaním sme overili, že aplikácia je plne funkčná v moderných prehliadačoch:

- Google Chrome, verzia 26.0.1410.64 m
- Opera, verzia 12.15 build 1748
- Mozilla Firefox, verzia 16.0.2; 20.0.1
- Internet Explorer, verzia 10 (okrem `type="number"`)
- Android Web Browser, verzia 4.0

5.4 Zhodnotenie aplikácie

Webová aplikácia spĺňa ciele stanovené na začiatku príce a demonštruje možnosti vytvorenia webovej aplikácie len s použitím natívnych funkcií HTML5. Nakoľko aplikácia nebola primárne navrhnutá pre použitie na mobilných zariadeniach, pri zadávaní informácii na obrazovkách s malým rozlíšením je potrebné priblíženie obrazu. Tento nedostatok by odstránilo použitie Media Queries[21], ktorými je možné priradiť

[21] http://www.w3.org/TR/css3-mediaqueries/

špecifický vzhľad obsahu HTML dokumentov podľa rozmerov okna prehliadača. Použitím JavaScriptovej knižnice jQuery[22] by sa taktiež dala dosiahnuť väčšia spätná kompatibilita a zároveň väčšia kompatibilita naprieč prehliadačmi.

V prílohovej časti sme uviedli niektoré časti kódu, ktoré sú špecifické pre implementáciu funkcií HTML5 a môžu byť znova použité.

5.5 Referenčný dokument

Referenčný dokument, ktorý tvorí druhú časť praktickej časti obsahuje prehľad nových možností HTML5 tematicky rozdelenú do štyroch kapitol a doplnkovú kapitolu, ktorá popisuje niektoré vlastnosti CSS3 použiteľné k obohateniu vzhľadu užívateľského prostredia. Dokument je vytvorený pomocou HTML5 a CSS3, takže samotný prehľad je zároveň aj demonštráciou implementácie. Navigáciu po dokumente zabezpečuje postranné menu. Úvodná obrazovka praktickej časti je v Prílohe E.

Tretia kapitola, približujúca možnosti tvorby grafiky v elemente `<canvas>` obsahuje interaktívne kódové pole. Pomocou metód z Canvas 2D API do poľa môžu užívatelia kresliť na priradenom plátne alebo sledovať dopad zmien parametrov predpripravených ukážok. Touto interaktívnou formou si rýchlejšie osvoja prácu v prostredí `<canvas>` bez nutnosti predchádzajúcich znalostí. Nadobudnuté znalosti môžu ďalej použiť pri tvorbe dynamických grafov alebo grafických prvkov dokumentov.

[22] http://jquery.com/

ZÁVER

Cieľom bakalárskej práce bolo priblížiť nový štandard HTML5 a porovnať ho s predchádzajúcou verziou HTML4. Pre rozsiahlosť špecifikácie nebolo možné detailne popísať každú časť, no vytvorili sme prehľad funkcií a ich implementáciu. V praktickej časti sme demonštrovali použitie vybraných funkcií, ktoré sú rozsiahlejšie popísané v samotnej práci alebo v druhej časti praktickej časti, vo webovej aplikácii. Aplikácie je tvorená prevažne pre stolné prehliadače, ale nakoľko nevyžaduje prístup k sieti internet, môže obohatiť zbierku aplikácii chytrých moderných zariadení. Referenčný dokument obsahujúci prehľad možností môže slúžiť ako referencia pri prechode z verzie HTML 4.01. Jednotlivé funkcie sú stručne popísané a zachytávajú podstatu potrebnú pre pochopenie a okamžité použitie. Stanovené ciele sme tak splnili.

Osobným prínosom práce je nadobudnutý prehľad o najnovších webových technológiách, prehľad špecifikácií a skúsenosti v jazykoch HTML, CSS a JavaScript. Získané vedomosti sme aplikovali pri tvorbe praktickej časti práce. Najmä osvojenie jazyka JavaScript umožnilo použiť aj pokročilé API HTML5, ktoré posúvajú možnosti tvorby natívneho dynamického obsahu.

Počet stránok a aplikácii v súčasnom Webe využívajúcich HTML5 neustále rastie. S prechodom HTML5 na odporúčanú verziu pri zachovaní súčasného stúpajúceho trendu implementácie vytvorí HTML5 silnú platformu nezávislú na zariadeniach konkurujúcu technológii FLASH. Počas písania práce boli elementy v špecifikácii pridávané a odstraňované, takže udržať krok s aktuálnou verziou bolo o to náročnejšie, čo na druhej strane dokazuje, že štandard stále nie je hotový a prispôsobuje sa reálnym požiadavkám Webu.

ZOZNAM POUŽITEJ LITERATÚRY

[1] PILGRIM, M. *Dive into HTML5*. [Online] 2011. [cit. 5. 12 2012.] 254s. Dostupné na internete: <http://diveintohtml5.info/>

[2] LAWSON B. - SHARP R. *Introducing HTML5. 2. vyd.* Berkley, CA : New Riders, 18 Október 2012. 314s. ISBN-13: 978-0-13-279300-1.

[3] Inc., Google. *HTML 5 Rocks*. [Online] [cit. 20. 2 2013.] Dostupné na internete: <http://www.html5rocks.com>

[4] HOGAN, BRIAN P. *HTML5 a CSS3: Výukový kurz webového vývojáře.* Brno : Computer Press, 2012. ISBN: 978-80-251-3576-1.

[5] HICKSON, I. *Web Storage*. [Online] W3C, 8. 12 2011. [cit. 12. 12 2012.] Dostupné na internete: <http://www.w3.org/TR/webstorage/>

[6] GOLDSTEIN A. - LAZARIS L. - WEYL E. *HTML5 & CSS3 for the Real World*. s.l. : Site Point Pty Ltd., 2011. 377s. ISBN: 978-0-9808469-0-4.

[7] CLARK R. - LAWSON B. et al.: *HTML5 Doctor*. [Online] [cit 8. 4. 2013.] Dostupné na internete: <http://html5doctor.com/>.

[8] W3C - HTML Working Group. *W3C - HTML Working Group.* [Online] W3C [cit. 15. 3. 2013] Dostpuné na internete: <http://www.w3.org/html/wg/>

[9] DYER J. MediaElements.js. [Online] GitHub Project. [cit. 5. 12 2012.] <http://mediaelementjs.com/>

[10] ROBINSON, M., *html5doctor*. [Online] 12. Júl 2011. [cit. 4. Február 2013.] Dostupné na internete: <http://html5doctor.com/outlines/>

[11] W3C *HTML design principles* [Online] 26. 11 2007. [cit. 5. 2 2013.] <http://www.w3.org/TR/html-design-principles/>

[12] HICKSON I., GOOGLE. *HTML - Cancvas element.* [Online] WHAT WG, 5. 12 2012. [cit. 5. 12 2012.] <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html>

[13] W3C. *Javascript web APIs*. *W3C.org.* [Online] [cit. 25. 3. 2013.] Dostupné na internete: <http://www.w3.org/standards/webdesign/script>

[14] W3C. *HTML5: Edition for Web Authors.* [Online] 25. 11 2012. [Dátum: 5. 2 2013.] Dostupné na internete: <http://www.w3.org/TR/html5-author/>

[15] W3C. *HTML5 - W3C Candidate Recommendation.* [Online] 17. 12 2012. [cit. 5. 2 2013.] Dostupné na internete: <http://www.w3.org/TR/html5/>

[16] HICKSON, I., SCHWARZ B. *HTML: The Living Standard*. Technická špecifikácia pre webových vývojárov. [Online] WHATWG, Jún 2012. [cit. 20. 4. 2013.] Dostupné na internete <http://developers.whatwg.org/>

[17] WHATWG. *HTML Living Standard - Complete*. [Online] WHATWG, 5. Február 2013. [cit. 5. Február 2013.] <http://www.whatwg.org/specs/web-apps/current-work/>

[18] WHATWG. Living Standard | Obsolete Features. [Online] 5. Február 2013. [cit. 5. Február 2013.] <http://www.whatwg.org/specs/web-apps/current-work/multipage/obsolete.html#obsolete>

[19] W3SCHOOLS.COM, *HTML DOM Tree Tutorial* [online] [cit. 6.4.2013] <http://www.w3schools.com/htmldom/default.asp>

[20] STEVENS L. The Truth About HTML5. 1.vyd. [ePub] Indie Digital Pty Ltd, 2012. 269s.

[21] ZAKAS. C, N. *Professional JavaScript for Web Developers (Programmer to Programmer)* 3.vyd. Indianapolis, IN: John Wiley & Sons, Inc., 2012. 964s. ISBN: 978-1-118-22219-5.

[22] KESSIN, Z. Programming HTML5 Applications. 1.vyd. Sebastopol, CA: O'Reilly Media, Inc. 2012. 142s. ISBN: 978-1-449-39908-5.

ČESTNÉ VYHLÁSENIE

Vyhlasujem, že som zadanú bakalársku prácu vypracoval samostatne, pod odborným vedením vedúceho bakalárskej práce Ing. Petra Holečka, PhD. a používal som len literatúru uvedenú v práci.

Súhlasím so zapožičiavaním bakalárskej práce.

V Žiline dňa 3. 5. 2013

_____

podpis

Prílohy

Zoznam príloh

Príloha A) súbory na priloženom CD:

css		
appStyle	Cascading Style Sheet Document	vzhľad aplikácie
desert	Cascading Style Sheet Document	vzhľad kódových výstrižkov v prehľade
info	Cascading Style Sheet Document	vzhľad prehľadovej časti
files		
bulbMpTri	MPEG Layer 3 Audio File	zvuk dopočítaný interval - mp3 (IE, Chrome, Safari)
bulbVorbis	Ogg Vorbis File	zvuk dopočítaný interval - ogg (Chrome, Firefox, Opera)
bulbWave	Microsoft Wave Sound Format	zvuk dopočítaný interval - wav (Chrome, Opera, Safari, Firefox)
fluteMpTri	MPEG Layer 3 Audio File	zvuk dopočítaný zoznam - mp3
fluteVorbis	Ogg Vorbis File	zvuk dopočítaný zoznam- ogg
fluteWave	Microsoft Wave Sound Format	zvuk dopočítaný zoznam- wav
traffic1	MP4 File	vzorové video z prehľadu - mp4 (IE, Safari)
traffic1	WebM Video	vzorové video z prehľadu - webm (Opera, Firefox, Chrome)
fonts		
WebSymbols-Regular-webfont.eot	EOT File	symbolové písmo (ikonky)
WebSymbols-Regular-webfont	TrueType font file	symbolové písmo (ikonky)
js		
info	JScript Script File	skript pre prehľad
mojScript	JScript Script File	skript pre aplikáciu
prettify	JScript Script File	farebné zvýrazňovanie kódových výstrižkov
run_prettify	JScript Script File	farebné zvýrazňovanie kódových výstrižkov
obrazky		
10px	JPEG image	pozadie elementu <canvas> v prehľade
arrow_bg	JPEG image	grafika k <hr> v prehľade
clanok	JPEG image	zápis článku pomocou nových elementov
grid	JPEG image	pozadie dokumentov
kont	JPEG image	tabuľka formátov
pos_vid	JPEG image	poster pre HTML5 video
tst10_thumb	JPEG image	figure
brows	PNG File	grafika podpory prehliadačov
guma	PNG File	ikonka Guma - canvas
htmlcss	PNG File	validacia HTML5 a CSS3
htmlMuscle_small	PNG File	externy obrazok pre príklad v canvas
offline	PNG File	aplikácia - stav offline
online	PNG File	aplikácia - stav online
save	PNG File	ikonka Save - canvas
valec	PNG File	ikonka Prekresli - canvas
manifest.appcache	APPCACHE File	manifest Application cache
.htaccess	HTACCESS File	pridanie MIME typov pre Application cache
app	Chrome HTML Document	webová aplikácia - počítač intervalov
info	Chrome HTML Document	prehľad funkcii HTML5

Príloha B) manifest Application cache

```
CACHE MANIFEST
# Cache manifest version 1.00
# Pocitac Intervalov

CACHE:
#subory, ktore sa ulozia do cache

app.html
js/mojScript.js
css/appStyle.css
obrazky/grid.png
obrazky/brows.png
obrazky/icons.png
obrazky/htmlcss.png
fonts/WebSymbols-Regular-webfont.eot
fonts/WebSymbols-Regular-webfont.ttf

NETWORK:
#subory, ktore sa vzdy nacitaju z weboveh servera - videa, zvuky
*

FALLBACK:
#nahradne subory pri zlyhani pripojenia
obrazky/online.png obrazky/offline.png
```

Príloha C) Drag and Drop udalosti

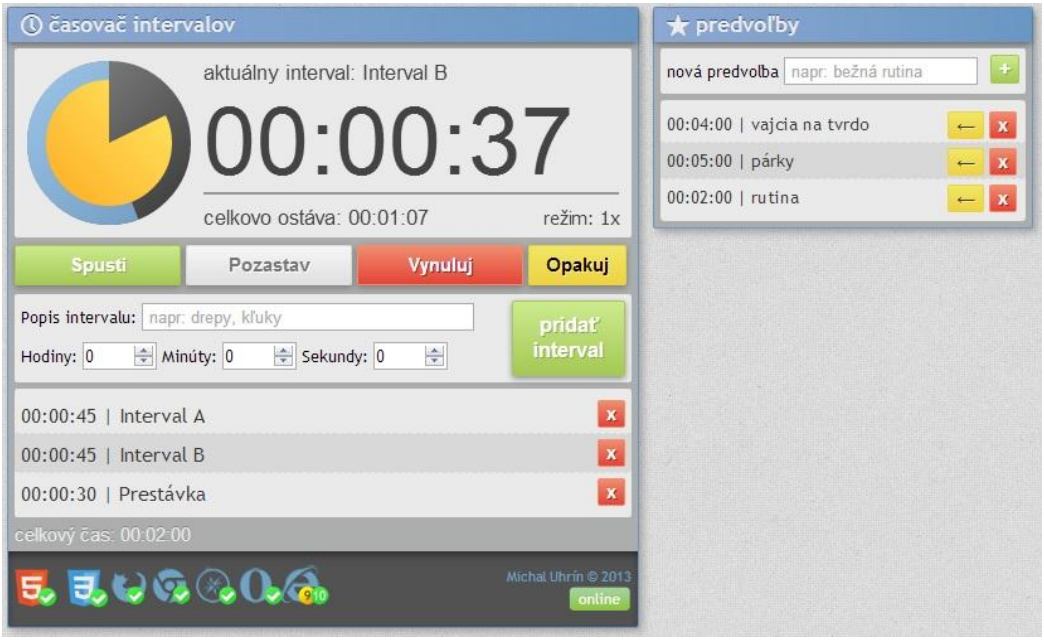
```
//DRAGSTART:
function dStartInterval(target, e) {
    //zamknutie pocas pocitania
    if (cntRunin === false && cntPaused === false) {
        e.dataTransfer.effectAllowed = 'move';
        e.dataTransfer.setData('text', target.outerHTML);
        e.target.style.opacity = '0.4'; //zpriehľadnenie zdrojového intervalu
    }
}

//DRAGEND:
function dEndInterval(target, event) {
    if (cntRunin === false && cntPaused === false) {
        target.parentNode.removeChild(target);
    }
}

//DRAGOVER:
function dOverInterval(e) {
    //zamknutie pocas pocitania
    if (e.preventDefault) { //umozni pustenie - nahrada za dropzone
        e.preventDefault();
    }
    e.dataTransfer.dropEffect = 'move';
    return false;
}

//DRAGENTER:
function dEnterInterval(e) {
    e.target.classList.add('over'); //označenie aktuálneho cieľa pustenia
}

//DRAGLEAVE:
function dLeaveInterval(e) {
    e.target.classList.remove('over'); //odznačenie aktuálneho cieľa pustenia
}

//DROP:
function dDropInterval(target, e) {
    if (e.stopPropagation) {
    e.stopPropagation(); //zabráni prehliadaču presmerovanie (Firefox)
    }
    e.target.classList.remove('over'); //odznačenie aktuálneho cieľa pustenia

    //ak je ciel pustenia prvý v zozname intervalov, vloží sa pred neho
    if(target === poleIntervalov()[0]) {
        target.insertAdjacentHTML('beforebegin',
e.dataTransfer.getData('text'));
    } else {
       //inak vloží ťahaný interval za cieľ spustenia
        target.insertAdjacentHTML('afterend',
e.dataTransfer.getData('text'));
    }
    return false;
}
```

Príloha D) validácia formulára

```
// VALIDATION API:

//VALDIACIA TEXTU:
function validaciaStr(pole, element) {
       //validacia chýbajúcej hodnoty poľa
    if (pole.validity.valueMissing === false) {
        document.getElementById(element).classList.remove('valErr');
        return true;
    } else {
        document.getElementById(element).classList.add('valErr');
        return false;
    }
}

//VALIDACIA CISLA:

//rangeOverflow - overenie na zaklade atrubútu 'max'
//rangeUnderflow - overenie na zaklade atrubútu 'min'
//valueMissing - kontrola chýbajúcej hodnoty
// vlastnosť 'validity' nesie stav overenia poľa

function validaciaNum() {
    //pattern, kontrola medze hranic
    if (arguments[0].validity.rangeOverflow === false &&
       arguments[0].validity.rangeUnderflow === false &&
       arguments[0].validity.valueMissing === false) {
        return true;
    } else {
        return false;
    }
}
```

Príloha E) obrazovky praktickej časti

obrazovka aplikácie (Google Chrome) - režim pauza:



obrazovka prehľadu (Google Chrome):