Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Možnosti využití technologií HTML5 a CSS3 při tvorbě website

Bc. Tomáš Berák
vedoucí: Ing. Petr Benda, Ph. D

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Berák Tomáš

Informatika

Název práce

Možnosti využití technologii HTML5 a CSS3 při tvorbě website

Anglický název

Possibilities of HTML5 and CSS3 technologies for website development

---

Cíle práce

Cílem práce je popis a analýza nových možností jazyka HTML 5 a CSS3. Dále porovnání s možnostmi stávajících verzí těchto technologií, rozlišení postupů a metodik tvorby webu a analýza podpory těchto řešení v internetových prohlížečích. Analyzované postupy budou realizovány v rámci praktické studie. Na základě zjištěných výsledků bude formulováno ucelené doporučení pro práci s novými technologiemi.

Metodika

Řešení problematiky diplomové práce bude založeno na studiu a analýze odborných informačních zdrojů. Praktická část práce bude zaměřena na vypracování případové studie analyzující možnosti technologií HTML5 a CSS3. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry diplomové práce.

Harmonogram zpracování

06/2013 - Upřesnění osnovy a dílčích cílů práce
07/2013 - 10/2013 Analýza informačních zdrojů a zpracování rešeršní části
10/2013 - Kontrola průběhu práce - 1. zápočet
10/2014 - 02/2012 Vypracování praktické části
02/2014 - Kontrola průběhu práce - 2. zápočet
03/2014 - Finalizace a odevzdání práce

Rozsah textové části
60 - 80 stran

Klíčová slova
HTML, CSS, W3C, webové technologie, webové prohlížeče, webové stránky, webové standardy, redesign, pravidla přístupného webu

Doporučené zdroje informací
Pilgrim, Mark. HTML5: Up and Running. O'Reilly, 2010. 978-0596806026.
Weyl, Estelle, Lazaris, Louis a Goldstein, Alexis. HTML5 & CSS3 For The Real World. 5 : 19, 2011. 978-0980846904.

Vedoucí práce
Benda Petr, Ing., Ph.D.

Termín odevzdání
březen 2014

doc. Ing. Zdeněk Havlíček, CSc.
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

prof. Ing. Jan Hron, DrSc., dr. h. c.
Děkan fakulty

V Praze dne 30.10.2013

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Možnosti využití technologií HTML5 a CSS3 při tvorbě website" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze, dne 28.11.2014 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce Ing. Petru Bendovi, Ph.D za cenné rady a připomínky při tvorbě této práce.

Možnosti využití technologií HTML5 a CSS3 při tvorbě website

Possibilities of HTML5 and CSS3 technologies for website development

Souhrn

Diplomová práce se zabývá novými technologiemi HTML5 a CSS3 a jejich možnostmi využití při tvorbě webových aplikací. Práce je rozdělena na teoretickou a praktickou část.

Teoretická část obsahuje dílčí kapitoly, které popisují důležité historické milníky webových aplikací, jejich definice a principy fungování, problematiku webových prohlížečů a popis stávajících standardů. Dále je v práci popsána problematika nových technologií HTML5 a CSS3, která je doplněna o praktické příklady jejich využití.

Praktická část práce je orientována na klíčové funkcionality technologií HTML5 a CSS3. Pomocí technologie HTML5 je vytvořena webová aplikace, ve které jsou implementovány klíčové pilíře této technologie. Výsledná aplikace je analyzována a zhodnocena.

Klíčová slova: HTML, CSS, W3C, webové technologie, webové prohlížeče, webové stránky, webové standardy, redesign, pravidla přístupného webu

Summary

This thesis deals with new technologies HTML5 and CSS3, and its potential for creating web applications. The work is divided into theoretical and practical part.

The theoretical contains partial chapters that describe the important historical milestones of Web applications, their definitions and principles of operation, the issue of web browsers and description of existing standards. The theoretical part also describes the problems of new technologies HTML5 and CSS3, which is supplemented by practical examples of their used.

The practical part is focused on the key functionality of technology HTML5 and CSS3. Using HTML5 web application is created, which are implemented key pillars of this technology. The resulting application is analyzed and evaluated.

Keywords: HTML, CSS, W3C, web technologies, web browsers, web pages, web standards, redesign, rules of accessibility

Obsah:

1 Úvod

Webové aplikace jsou možná největším vynálezem dvacátého století v odětví internetových technologií. Za posledních dvacet let změnily způsoby vykonávání práce, studia a v neposlední řadě také ovlivnily způsoby chování lidí. Moderní doba vyžaduje efektivnost, rychlost a dostupnost zpracování informací a to je hlavním důsledkem progresivního vývoje internetových technologií.

Základním pilířem webových aplikací je značkovací jazyk HTML. Pomocí této technologie jsou definovány základní struktury samotných aplikací. Tedy rozdělení webového dokumentu do částí jako jsou záhlaví, navigační menu či prostor pro samotný obsah. Funkcí HTML je tedy způsob reprezentace definovaného textu na webové stránce.

Technologie CSS je jazykem, který je v úzké symbióze s jazykem HTML. Jedná se o jazyk, který neovlivňuje obsah dokumentu, ale definuje jeho striktní strukturu. Reprezentuje tedy způsoby zobrazení dokumentu na výstupním zařízení. Webové aplikace tedy díky kaskádovým stylům nejsou pouhým textem.

Přetrvávajícím problémem u nově vznikajících technologií je podpora webových aplikací u internetových prohlížečů. Rychlost vývoje aplikací a množství internetových prohlížečů má za následek omezenou podporu či dokonce nefunkčnost některých standardů těchto jazyků.

Nejnovější technologie s označením HTML5 a CSS3 přináší mnoho nových funkcí. To však přináší mnoho problémů a rizik v podobě změny logiky vytváření webových aplikací a podpory těchto technologií webovými prohlížeči. Tato práce se právě zabývá těmito technologiemi, které by se měly stát webovými standardy právě v době psaní této práce.

2 Cíle práce a metodika

2.1 Cíl práce

Cílem práce je popis a analýza nových možností jazyka HTML5 a CSS3. Dále porovnání s možnostmi stávajících verzí těchto technologií, rozlišení postupů a metodik tvorby webu a analýza podpory těchto řešení v internetových prohlížečích. Analyzované postupy budou realizovány v rámci praktické studie. Na základě zjištěných výsledků bude formulováno ucelené doporučení pro práci s novými technologiemi.

2.2 Metodika

Řešení problematiky diplomové práce bude založeno na studiu a analýze odborných informačních zdrojů. Praktická část práce bude zaměřena na vypracování případové studie analyzující možnosti technologií HTML5 a CSS3. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry diplomové práce.

3 Přehled řešené problematiky

3.1 World Wide Web

Před pětadvaceti lety spatřil světlo světa první návrh systému, z něhož se následně vyvinula služba World Wide Web a její zkratka www. Internetová služba vznikla původně jako prostředek ke sdílení vědeckých informací. První návrh systému předložil Brit Tim Berners-Lee před 25 lety, 13. března 1989 na půdě Evropské organizace pro jaderný výzkum (CERN) se sídlem v Ženevě.

V polovině roku 1980 přišel do této mezinárodní vědecké organizace jako konzultant britský softwarový inženýr Timothy Berners-Lee, který tehdy pro svou osobní potřebu naprogramoval jednoduchý elektronický katalog, přístupný odkudkoliv prostřednictvím počítačové sítě. Jednotlivé listy propojoval systém odkazů, díky nimž se daly snadno vyhledat související informace. Odtud byl jen krůček k vytvoření systému internetových stránek, mezi nimiž lze jednoduše přecházet pomocí hypertextových odkazů. Koncem roku 1981 však Bernersi-Leemu vypršel původní šestiměsíční kontrakt s CERNem a odešel pracovat pro firmu Image Computer Systems. Teprve po návratu do Ženevy v polovině dekády svůj někdejší nápad rozpracoval, takže jej na jaře 1989 mohl CERNu nabídnout jako řešení dlouhodobých problémů se sdílením dat mezi vědeckými týmy. (čtk, a další, 2013)

Spolu s Robertem Cailliauem a několika dalšími spolupracovníky navrhl Berners-Lee jazyk pro vytváření webových stránek (HyperText Markup Language, zkráceně HTML) a definoval protokol pro přenos stránek internetem (HyperText Transfer Protocol, HTTP). Berners-Lee byl i autorem programu, jenž sloužil zároveň jako editor i prohlížeč stránek. Když Berners-Lee zvažoval jak program pojmenovat, napadaly ho názvy jako Mine of Information (Důl na informace) či Information Mesh (Informační spleť). Nakonec ale padla volba na World Wide Web (Celosvětová síť). Tento název se posléze v mírně modifikované podobě (s mezerami mezi slovy) stal označením pro celou službu, zatímco samotný prohlížeč byl překřtěn na Nexus. (čtk, a další, 2013)

Koncem roku 1990 byl do provozu uveden první webový server na světě. Měl adresu info.cern.ch a běžel na počítači NeXT, který v CERNu pietně uchovávají dodnes. Stránky byly čistě textové a obsahovaly návod pro zájemce o pořízení vlastního webu. Své

servery začaly záhy zřizovat evropské i americké vědecké instituce. V listopadu 1992 bylo po celém světě 26 serverů. O rok později jejich počet přesáhl pět set a koncem roku 1994 už jich bylo deset tisíc kusů, přičemž se k nim připojovalo na deset milionů uživatelů. Příčinou tohoto raketového růstu bylo rozhodnutí Evropské organizace pro jaderný výzkum, která na jaře 1993 poskytla veřejnosti možnost využívat zcela zdarma.
(čtk, a další, 2013)

Novináři se Bernerse-Leeho v minulosti opakovaně ptali, zda nelituje rozhodnutí poskytnout web uživatelům bezplatně. Vždy se jim přitom dostalo negativní odpovědi. V interview z října 2009 Berners-Lee napůl v žertu prohlásil, že zpětné lituje jen jedné věci, a to rozhodnutí použít dvojité lomítko pro oddělení názvu protokolu od domény webu v adresách webových stránek. Věcně vzato to nebylo nezbytné a někteří uživatelé si na to prý později stěžovali. Obligátní formulku „http:“ a dvě lomítka ale dnes už stejně většina prohlížečů doplňuje do adresního řádku automaticky. (čtk, a další, 2013)

Za svůj přínos světu informačních technologií byl Timothy Berners-Lee mnohokrát oceněn a v roce 2004 jej britská královna dokonce povýšila do šlechtického stavu. Práci na rozvoji webu se věnuje dodnes, ať už coby vysokoškolský profesor, nebo jako předseda organizace World Wide Web Consortium (W3C), která definuje nové a rozvíjí stávající technické standardy webu. Berners-Lee je rovněž znám jako otec myšlenky takzvaného sémantického webu, na němž nebudou informace uloženy a strukturovány nahodile jako dnes, nýbrž podle standardizovaných pravidel usnadňujících jejich vyhledání a zpracování. (čtk, a další, 2013)

3.1.1 W3C

World Wide Web Consortium neboli W3C je mezinárodní společenství, které obsahuje velké množsví členských či průmyslových organizací a společně pracují na standardech pro World Wide Web. (Christensson, 2010)

Posláním W3C je vést web k jeho plnému potenciálu pomocí příslušných protokolů a směrnic. Toho je dosaženo především tím, že vytváří a publikuje webové standardy. Zavedením webových standardů vytvořených W3C, mohou hardwarový a softwarový výrobci zajistit kompatibilitu s nejnovějšími webovými technologiemi. Například většina

webových prohlížečů obsahuje standardy W3C, což jim umožňuje interpretovat nejnovější verze HTML a CSS kódu. (Christensson, 2010)

Kromě standardů HTML a CSS, W3C také poskytuje standardy pro webové grafiky (jako jsou například PNG obrázky), ale také i pro audio a video určené pro webové stránky. Organizace dále vyvíjí standardy pro webové aplikace či webové skriptování. Navíc také připravuje ochranu osobních údajů a bezpečnostní pokyny, které by měli uživatelé dodržovat. (Christensson, 2010)

3.2 HyperText Transfer Protocol

Protokol HTTP (HyperText Transfer Protocol) je aplikační protokol určený k přístupu a výměně informací v prostředí distribuovaných hypermediálních informačních systémů. Je používán od roku 1990 v rámci globální iniciativy WWW (World Wide Web), v níž je použit pro přenos hypertextových WWW stránek ve formátu HTML (Hyper Text Markup Language) mezi WWW (Web) klienty a servery. Dnes je protokol využíván jako základní generický protokol sítě internet a vnitropodnikových sítí intranet a používá se i ke zpřístupňování služeb jiných aplikačních protokolů (FTP, SMTP, aj.). Původní verze, využívaná v prních WWW prohlížečích a serverech, byla označena jako HTTP/0.9, dnes je však rozšířenější všeobecnější verze HTTP/1.0 a vyšší. (Burian, 2014)

Protokol HTTP je koncipován jako jednoduchý bezestavový objektově orientovaný protokol (stateless), pracující v režimu otázka / odpověď (request / response). HTTP vyžaduje pro přenos zpráv (request / reply) použití spolehlivé transportní služby. Umožňuje proto použití libovolného transparentního protokolu. Zatím je však upřednostňován protokol TCP s vyhrazeným aplikačním portem TCP na straně http serveru. Protokol umožňuje přenášet informace v různorodém formátu. HTTP používá k adresování požadovaných informačních zdrojů adresové schéma uznačované jako URL (Uniform Resource Locator), které adresuje nejen příslušný server, ale i požadovaný dokument, přístupový protokol, případně data pro autorizaci přístupu. (Burian, 2014)

3.2.1 Metody způsobu komunikace

Protokol HTTP verze 1.0 definuje příkazy GET, HEAD a POST. Verze 1.1 protokolu HTTP rozšiřuje HTTP 1.0 o příkazy PUT, DELETE a TRACE. Minimální implementace protokolu HTTP musí obsahovat tři příkazy: GET, HEAD, POST. (Grygarek, 2009)

GET

Příkaz GET načte požadované informace, které jsou identifikované pomocí URL adresy ze serveru. S použitím pole hlavičky If-Modified-Since je možné implementovat načtení dat, pokud byly změněny od stanoveného data. Pomocí této informace je možné se rozhodnout, jestli data načtená v cache od minulého použití jsou aktuální, nebo mají být načtena znovu. (Grygarek, 2009)

Syntaxe: GET <URL> HTTP/1.0
Příklad: GET http://www.google.com HTTP/1.0

HEAD

Metoda HEAD je identická jako metoda GET s tím rozdílem, že server nevrací žádné tělo zprávy. Metainformace obsažené v hlavičce odpovědi na požadavek příkazu HEAD jsou stejné jako u příkazu GET. Tato metoda je často používána pro testování dostupnosti nebo aktuálnosti změn odkazů. Pokud je použito pole hlavičky If-Modified-Since, je ignorováno. Syntaxe je stejná jako u příkazu GET. (Grygarek, 2009)

POST

Metoda POST se používá k odeslání informací na server. Nejčastější použití je odeslání dat vyplněného formuláře na server. POST je vytvořena, aby pomohla zajistit ostatním metodám tyto funkce (Grygarek, 2009):

- Komentář k existujícím zdrojům.
- Odeslání zprávy na vývěsku, konference, emailového seznamu nebo podobných skupin.

- Zajištění bloku dat jako výsledek odeslání formuláře.
- Rozšíření databáze pomocí operace append.

Aktuální funkce prováděné metodou POST jsou odvozeny serverem a obvykle závisejí na požadavkové URI. Pomocí URI se rozlišuje operace s prováděnými daty.

Při použití metody POST je vyžadováno správně vyplněné pole hlavičky Content-Length. Server může odpovědět chybovou zprávou 400 (bad request), pokud nedokáže určit velikost obsahu zprávy. Aplikace nesmí cachovat odpovědi na příkaz POST, protože nemají žádnou možnost jak zjistit, zda server v budoucnu na stejný požadavek odpoví stejně. (Grygarek, 2009)

3.2.2 Stavový kód a stavové hlášení

Stavový kód je číselná hodnota skládající se ze tří číslic. Odpovídá výslednému kódu, který servervy generoval při pokusu o zpracování požadavku. Stavový kód je určen pro programy a aplikace, zatímco text, který jej doprovází, je určen pro lidské čtenáře. První číslo stavového kódu definuje kategorii výsledného kódu. Všechny kódy se dají rozdělit na následující kategorie (Jakel, 2002):

Tabulka č. 1: Kategorie stavových kódů

Kategorie	Rozsah stavových kódů	Popis
Informační	100 - 199	Zprávy definované konkrétní aplikací.
Úspěch	200 - 299	Požadavek byl úspěšně zpracován.
Přesměrování	300 - 399	Klient musí pro konečné zpracování požadavků vykonat určitou činnost. O této činnosti se uživatel nemusí dovědět.
Chyba klienta	400 - 499	Problémy na straně klienta.
Chyba serveru	500 - 599	Problémy na straně serveru.

pozn.: * Agregace a popis stavových hlášení.

Zdroj: (Jakel, 2002)

Rozpis některých dvojic stavových kódů - stavové hlášení včetně popisu akce, která s nimi souvisí:

Tabulka č. 2: Granularizace stavových kódů

Stavový kód	Popis
100 Continue	Klient může pokračovat v zasílání požadavku.
101 Switching Protocols	Server mění protokol.
200 OK	Operace proběhla bez chyby, požadavek je úspěšně splněn.
201 Created	Výsledkem požadavku je nově vytvořený objekt.
202 Accepted	Byl přijat asynchronní požadavek. Požadavek byl správně akceptován, odpovídající činnost se však ještě zatím nemusela provést.
204 No Content	Požadavek byl úspěšný, ale jeho výsledkem nejsou žádná data pro klienta.
300 Multiple choises	Požadovaný zdroj se dá získat z několika různých míst. V odpovědi se vrací seznam všech možností.
301 Moved Permanently	Požadovaná adresa URL se trvale přesunula na novou adresu URL. Všechny další odkazy musí použít tuto novou URL.
302 Moved Temporarily	Požadovaná adresa URL se dočasně přesunula na novou adresu URL. Všechny další odkazy mohou používat dosavadní URL.
304 Not Modified	Podmíněný požadavek byl správně zpracován, dokument však od udané doby nebyl modifikován.
400 Bad Request	Server nerozumí požadavku, klient jej musí opravit a poslat znovu.
401 Unauthorized	Jestliže byl původní požadavek klienta anonymní, musí být nyní autentizován. Pokud už požadavek byl autentizován, pak byl přístup odepřen.
403 Forbidden	Server nemůže požadavku vyhovět, autorizace nebyla úspěšná.
404 Not Found	Server nenašel zadanou adresu URL.
405 Method Not Allowed	Použitá metoda není přípustná pro dosažení požadovaného objektu.
406 Not Acceptable	Požadovaný objekt není k dispozici ve formátu podporovaném klientem.
408 Request Timeout	Klient nedokončil odesílání požadavku v časovém limitu.
410 Gone	Požadovaný objekt byl trvale odstraněn.
415 Unsupported Media Type	Požadavek obsahuje data v serveru neznámém formátu.
500 Internal Server Error	Na serveru došlo k neočekávané chybě.
501 Not Implemented	Tento požadavek server nepodporuje.
502 Bad Gateway	Proxy server nebo brána obdržely od dalšího serveru neplatnou odpověď.
503 Service Unavailable	Server dočasně nemůže nebo nechce zpracovat požadavek. Většinou když je přetížený nebo se provádí údržba.
505 HTTP Version Not Supported	Server nepodporuje verzi HTTP v daném požadavku.

Pozn.: * Detailní popis stavových kódů.

Zdroj: (Jakel, 2002)

3.3 Webové technologie

Dříve webové prohlížeče sloužily téměř výhradně k pasivnímu prohlížení webového obsahu. V posledních letech s rozvojem cloud-computingu ovšem z webu vykrystalizovala jakási univerzální platforma dostupná na širokém spektru zařízení. Současné webové aplikace jsou s největší pravděpodobností jen předzvěstí dalšího technologického posunu. (Macich, 2013)

Je to logické, protože mohou fungovat na opravdu pestré paletě zařízení, aniž by záleželo na jejich hardwarové architektuře či přítomném operačním systému. Vedle osobních počítačů jde o mobily, tablety, chytré televizory a další černou elektroniku, stacionární i přenosné herní konzole, chytré přenosné přehrávače multimédií nebo čtečky elektronických knih. (Macich, 2013)

Nicméně potenciál webových aplikací prozatím stále určuje podpora nových webových standardů a webových technologií v prohlížečích pro osobní počítače. (Macich, 2013)

3.3.1 Značkovací jazyky

Anglicky markup language, označují soubor pravidel (konvencí), který určuje způsob použití značek při kódování textu. Takový jazyk musí specifikovat přesně, které značky je možné používat, které z nich jsou povinné, dále způsob odlišení značek od vlastního textu a hlavně význam jednotlivých značek. (Sklenák, 2001)

Historicky se slovo markup používalo pro popis anotace nebo jiných značek, které byly vloženy přímo do textu, a které byly určeny pro sazeče, aby věděl, jak má být daný úsek vytištěn / vysázen. Například zvlněné podrtržení znamenalo pokyn pro použití tučného písma, zvláštní symboly označovaly text k vynechání, k tisku jinou velikostí nebo jiným písmem atd. postupem času se příprava textů pro formátování a tisk zautomatizovala, a proto se i pojem markup začal používat v obecnějším smyslu pro všechny druhy značek (markup codes), které jsou vkládány do elektronického textu pro ovlivnění formátování, tisku nebo jiného zpracování textu. (Sklenák, 2001)

Kódování textu pomocí značek je možné vymezit jako způsob explicitního ovlivnění interpretace textu. Na nejnižší úrovni lze takto chápat každý tištěný text – obsahuje interpunkční znaménka, velká písmena, určitý způsob rozmístění textu na stránce, mezery mezi slovy – to vše lze považovat za určité značky, které napomáhají čtenáři rozpoznávat jednotlivá slova a rozlišovat sktrukturální jednotky textu (záhlaví, odstavce, věty). Kódování textu pro počítačové zpracování tedy není nic jiného, než určitý způsob vyjádření myšlenek autora. Je to proces explicitního vyjadřování všeho domnělého i implicitního tak, aby uživatel byl nasměrován, jak má daný text interpretovat. (Sklenák, 2001)

SGML

Standard Generalized Markup Language (SGML) je určen pro naplnění výše uvedených požadavků (viz 3.1.1 Značkovací jazyky). Další požadavky je nutno řešit na aplikační úrovni. Kromě zachycení struktury textu, slouží SGML jako metajazyk pro popis nového markup jazyka a prostředek pro vyjádření obsahu textu. Jak se přitom bude s textem a jeho vyznačenými částmi zacházet, záleží vždy na použité aplikaci. (Sklenák, 2001)

SGML zavádí pojem „typ dokumentu“ a dála pak „definici typu dokumentu“ (dále jen DTD). Na dokumenty se nahlíží podobně jako na jiné objekty, které jsou dnes počítačově zpracovány, a proto lze hovořit o jejich typech. Typ dokumentu je formálně definován vymezením jeho částí a jejich strukturou. Například zpráva (report) může být definována tak, že se skládá z názvu, obvykle i autora, abstraktu a posloupnosti odstavců. Ovšem samotný název, pokud není následován abstraktem a odstavci textu, nelze považovat z formálního hlediska za zprávu. (Sklenák, 2001)

DTD obsahuje mimo jiné definici všech elementů, které lze v dokumentu použít, jejich přípustných hodnot a atributů, které lze u daného elementu použít. Tato definice je díky SGML zapsána ve zcela standardizované podobě, a proto může být snadno zpravována počítači. Z tohoto pohledu si lze SGML představit jako značkovací jazyk a DTD jako program zapsaný v SGML. (Sklenák, 2001)

DTD pouze definuje, jakou může mít dokument syntaxi – jména elementů, jejich používání. Sémantiku (význam) jednotlivých elementů je nutné definovat odlišněji. Například pro HTML obsahuje jeho specifikace popis použití všech elementů – autor HTML dokumentu tak ví, k čemu který element slouží. Pro prohlížeč je zase naopak důležité vědět, jak má obsah jednotlivých elementů zobrazit – tuto definici může mít v sobě zabudovanou napevno, nebo může využívat například styly, které definují vzhled jednotlivých elementů. (Sklenák, 2001)

HTML

HTML neboli HyperText Markup Language je značkovací jazyk, který je určen k vytváření dokumentů, obsahujících hypertextové odkazy a pokročilejší formátování. (Písek, 2010)

HTML poskytuje elementy, pomocí nichž lze vytvářet, formátovat nebo upravovat webové stránky následujícími způsoby (Písek, 2010):

- Nastavení vzhledu nebo velikosti zvoleného textu, použití tučného písma či kurzívy.
- Vložení obrázků všech tvarů a velikostí, použití animací ve formátu GIF, což jsou série obrázků sloučených do jediného souboru.
- Vytvoření formulářů pro zasílání emailů či reakci na dotazník.
- Vytvoření tabulek pro lepší kontrolu nad dokumentem a jeho formátováním.
- Definování barev bozadí pro HTML dokument, celou tabulku, řádek tabulky, nebo dokonce pro jednotlivou buňku.
- Vkládání odkazů pro další sekce WWW stránek, na dokumenty na jiných stránkách, video a audio soubory.

Z výše uvedeného si je možné povšimnout, že HTML má značně široké možnosti použití, ale nadruhou stranu je tu celá řada omezení, která mohou začínajícího tvůrce webových stránek nepříjemně překvapit. Je nutné si uvědomit, že pro vytvoření některých funkcí, kterými dnes běžně webové stránky disponují, si se samotným HTML není možné vystačit. (Písek, 2010)

HTML je určeno pro dokumenty se statickým obsahem. To znamená, že nelze vytvářek dokumenty, jejichž obsah by se automaticky měnil. Pomocí HTML nelze vytvářet dynamicky se měnící nabídky, vysouvací menu ani nic podobného. Proto je jazyk HTML kombinován s dalšími webovými technologiemi. (Písek, 2010)

HTML prošlo za svůj život několika fázemi. První verze HTML, 1.0, se objevila v roce 1990 a neoficiální verze HTML+ byla představena ve druhé polovině roku 1993. Rysy HTML+ zahrnovaly formuláře, tabulky a obrázky s popisem, ale neobsahovaly formátování odstavců ani úpravy textu. HTML+ obsahovalo 78 elementů, z nichž mnohé již dnes v HTML nenajdeme. Mnohé zastaralé elementy definovaly komponenty dokumentu, například výpisky, poznámky a podtitulky. (Písek, 2010)

HTML 2.0, uvedené na trh v roce 1994, bylo první verzí, která měla formální specifikaci a stala se oficiálním standardem. Tato verze obsahovala celkem 49 elementů.
V květnu 1996 bylo uvedeno na trh HTML 3.2, které dnes považujeme za pravého nástupce HTML 2.0, jinými slovy obchází HTML 3.0. HTML 3.2 přidalo 19 nových prvků, zachovalo tabulky a atributy pro obtékání textu z HTML 3.0 a zapracovalo četná rozšíření pro Netscape. (Písek, 2010)

HTML verzi 4.0 se během jejího vývoje přezdívalo Cougar. Tato verze přinesla podporu prvku OBJECT, který má velký význam pro vkládání obrázků a multimédií. Dále HTML 4.0 podporuje kaskádové styly, úpravy formulářů a tabulek, skriptování na straně klienta, internacionalizaci – tj. rozpoznání jazyků, které obsahují zvláštní znaky v abecedě nebo se například čtou zprava doleva a další zvláštní znaky pro matematické operace a profesionální publikování. (Písek, 2010)

V současné době se pracuje na specifikaci HTML 5.0. Přestože je zatím k dispozici jen jeho pracovní verze (není zatím standardizována W3C), již teď je jasné, že v této verzi bude řada užitečných novinek. (Písek, 2010)

XML

Jazyk XML (Extensible Markup Language) umožňuje v dokumentech používat libovolně pojmenované elementy, ale v praxi často zjistíme, že je velice užitečné dopředu znát, jaké informace se v dokumentu mohou vyskytovat. Schémata dokumentu XML umožní formálně zapsat, jaké informace se v dokumentu mohou vyskytovat. Schémata dokumentů XML umožní formálně zapsat, jaké elementy a atributy jsou v dokumentu povolené. (Kosek, 2009)

Primární význam schémat leží v jejich použití pro formání definici značkovacích jazyků. Výhoda formalizované definice spočívá v tom, že je jednoznačná a znemožňuje různé intepretace, na rozdíl od definice popsané v přirozeném jazyce.

Vzhledem k tomu, že schéma jednoznačně definuje, jak může dokument XML vypadat, je možné jej samozřejmě použít i pro validaci. Ostatně se jedná asi o nejčastější použití schémat. Validace je proces, při kterém se ověřuje, zda nějaký konkrétní dokument XML vyhovuje všem omezením definovaným ve schématu. Další výhodou validace je ulehčení vývoje aplikací. Ty nemusí při čtení dat z XML provádět kontrolu správného vstupu, protože většinu řiadných chyb odhalí validace. (Kosek, 2009)

Specifikace jazyka XML popisuje DTD (Document Type Definition), což je jazyk pro popis schématu dokumentu procházející ještě z jazyka SGML. DTD díky tomu mají jednu vekou výhodu – podporuje je velké množství aplikací. To je však současně jejich jedinou výhodou. (Kosek, 2009)

S rozšiřováním XML začínaly být nedostatky DTD stálepatrnější. Za největší nedostatek je možné označit nulovou podporu pro jmenné prostory. Jmenné prostory jsou mechanismus, který umožňuje k jednomu dokumentu XML kombinovat více sad značek – například do webové stránky v XHTML je možné vložit obrázek v SVG a matematickou rovnici zapsanou v MathML (Mathematical Markup Language). (Kosek, 2009)

Druhým nedostatkem je nemožnost určení datového typu pro obsah elementů a atributů. XML bylo primárně navrženo pro značkování dokumentů textové povahy. XML se však začalo masově používat i na výměnu strukturovaných dat mezi informačními systémy. V dokumentech, jako je například faktura, už zdaleka vše není jen textem.

Aby byla validace skutečně účinná, je potřeba omezit nejen strukturu vzájemného zanoření elementů, což DTD neumí, ale i obsah těchto elementů jejich datovým typem. (Kosek, 2009)

Předchozí důvody byly natolik závažné, že se již v době vzniku standardu XML začalo pracovat na několika nových schémových jazycích, které měly překonat výše popsaná omezení. (Kosek, 2009)

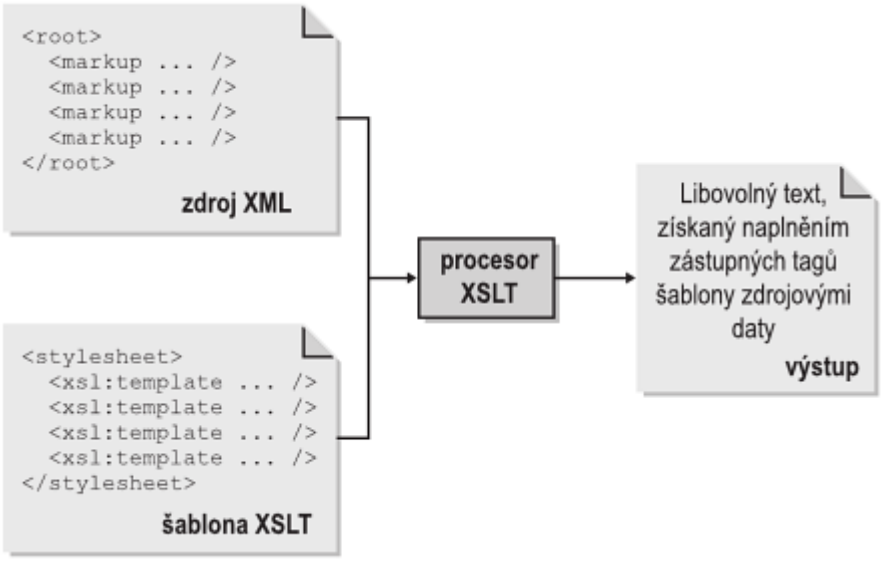
Uznávaným standardem se v květnu 2001 stal jazyk XML schéma přijatý jako doporučení konsorcia W3C. Po DTD je to dnes nejpoužívanější schémový jazyk a podporují jej všechny velké firmy a mnoho projektů open-source. Jediným problémem W3C XML schémat je poněkud složitá specifikace a některá poněkud temná a nejasná zákoutí jazyka. (Kosek, 2009)

XSLT

XSLT (eXtensible Stylesheet Language Transformations) je značkovacím jazykem založeným na XML, který slouží k převodu XML dokumentů na jiné textové formáty. V současné době je nejobvyklejším použitím XSLT převod XML dokumentu na jiný XML dokument, což pomáhá zmrnit následky nekompatibiitynávrhů XML dokumentů. (Kosek, 2009)

XSLT je tedy proces, který kombinuje dva dokumenty XML – zdrojový soubor a skript – a tak vytvoří dokument třetí. Výsledkem může být opět dokument XML, ale také stránka HTML nebolibovolný textový soubor. (Skonnard, a další, 2006)

Zdrojový dokument musí splňovat jen jediný požadavek – musí se jednat o správně strukturovaný dokument XML. Skript musí být platným dokumentem XML, který obsahuje logiku transformace, vyjádřenou prostřednictvím prvků slovníku XSLT. Na skript XSLT je možné nahlížet jako na posloupnost šablon. Každá šablona přebírá jako vstup jeden či více zdrojových elementů a vrací určitý výstupní text, složený z literálů a transformovaných vstupních dat. (Esposito, 2004)



Obrázek č. 1: Přehled procesu transformace XSLT

Zdroj: (Esposito, 2004)

CSS

Ve svých počátcích v roce 1990 byla jediným cílem jazyka HTML prezentace obsahu v čisté, strukturované formě, pouze s několika značkami, tak, aby bylo zřejmé, co je nadpis, co odstavec atd. Již v těchto dobách se předpokládalo, že způsob zobrazení se bude řídit stylovými předpisy. První webový prohlížeč na světě – NeXT – byl navržen tak, aby výsledné formátování dokumentu bylo řízeno jednoduchým stylovým předpisem. (Cyroň, 2005)

Později si začali mnozí autoři stežovat, že nemohou nikterak ovlivnit vzhled svých HTML stránek, a proto byl již v roce 1994 vypracován první návrh „Cascading Style Sheets“. Bohužel se vžak v téže době objevil webový prohlížeč Netscape 1, což s odstupem doby byla pro web katastrofa. Firma Netscape totiž místo podpory stylů zabudovala vlastní vlastnosti pro řízení přímo do značek HTML. Toto řešení započalo éru nekompatibility webových stránek i prohlížečů, a hlavně zanesení původně čistého a strukturovaného kódu HTML dokumentu různým formátovacím „balastem“. (Cyroň, 2005)

Postupně přišli na trh webových prohlížečů další výrobci, kteří v tomto trendu pokračovali, a HTML kód se tak zanášel dalšími značkami a jejich atributy, které nesloužily ničemu jinému, než k řízení vzhledu dokumentů. V jazyce HTML byly k dispozici i tabulky, a to výhradně k prezentaci tabulkových dat. Jenže autoři brzy přišli na to, že tabulky se dají využít k rozvržení stránek a „nalili“ do buněk sloupce textu a dalšího obsahu. Autoři byli později „obohaceni“ o tzv. rámce, pomocí kterých bylo možno vzhled prezentace na obrazovace zobrazit v té době nevídaným způsobem. (Cyroň, 2005)

Později začalo být i výrobcům prohlížečů jasné, že tudy cesta nevede. Jelikož v té době CSS však ještě nebyly nikým standardizovány, implementace CSS se v prohlížečích příliš neshodovaly. První verze specifikace standardizačního konsorcia W3C – CSS1 – byla uvedena v roce 1996. Za otce dnešních CSS je považován HíkonWium Lie, dnešní vývojář internetového prohlížeče Opera. (Cyroň, 2005)

Formátování pomocí CSS lze formátovat nejen tradiční HTML dokumenty, ale též dokumenty XML. Tato možnost je velmi důležitá, protože XML dokumenty sice dokáží zobrazit současné prohlížeče bez problémů, avšak bez jakéhokoliv formátování. S výjimknou stylování pomocí XSL, je použití jedinou metodou, jak dokument vhodně zobrazit. (Cyroň, 2005)

Současný trend při tvorbě HTML dokumentů směřuje k tomu, že veškeré rozmístění prvků je svěřeno CSS a nikoliv dříve obvyklým tabulkám a rámcům. Takto formátovaný dokument dokáží bez problému zobrazit nejen všechny moderní prohlížeče, ale i prohlížeče o generaci starší. Přístupnost obsahu prezentace pro zastaralé prohlížeče pak vyplývá automaticky z kvalitně strukturovaného dokumentu. (Cyroň, 2005)

Změna vzhledu prezentace, v níž je i rozmístění prvků řízeno CSS, je potom maličkostí. Snadno změníme zcela vzhled naší prezentace, a to nejen barvy a písmo, ale například i zarovnání či rozměry prvků a jejich umístění. (Cyroň, 2005)

Kromě standardních vlastností, definovaných ve specifikaci CSS, prohlížeče často podporují další nestandardní vlastnosti. Všechny možné hodnoty patří do některé z těchto kategorií (Cyroň, 2005):

- Standardní vlastnosti CSS: Vlastnosti, jejichž možné hodnoty a možnosti užití odpovídají v současnosti platné specifikaci CSS 2.1.
- Připravované vlastnosti CSS: Jedná se o vlastnosti, o nichž se zmiňuje stále ještě nehotová technologie CSS3, u kterých je dosti velká šance, že nakonec ve specifikaci CSS3 budou zakomponovány. Ovšem z pohledu CSS 2.1se jedná o hrubý nestandard a současná verze validátoru (programu na tvorbu validity) označí tyto za nevalidní (odporující specifikaci CSS 2.1).
- Odmítnuté vlastnosti CSS: vlastnosti, které byly postoupeny konsorciu W3C ke schválení v rámci specifikace CSS3, a byly buďto odmítnuty, nebo je u nich mizivá šance, že nakonec ve specifikaci CSS3 budou zakomponovány.
- Proprietární vlastnosti CSS: hrubě nestandardní rozšíření prohlížečů.

3.3.2 Databázové systémy

Data jsou prakticky jedinou „věcí“, kterou je možné na internetu nabízet. Návštěvník přichází na web z jediného důvodu – dozvědět se informaci, která je mu poskytnuta pomocí webového prohlížeče.

Ať už se jedná o data ve formě různých článků nebo o data v podobě uživatelských jmen a hesel, vždy pro všechny tato data je nutné najít vhodný způsob uložení na serveru.

Data je možné rozdělit do dvou kategorií:

1. Data generující se na výstupu.
2. Data, které je nutné před uživatelem skrýt.

Taková data jsou důležitá pro běh serveru jako takového a každá informace, kterou je o těchto datech možná získat, je nežádoucí. Jedná se tedy o data pro připojení k databázi, uživatelská jména a hesla, o struktury tabulek, které jsou vytvářeny a podobně. (Ponkrác, 2007)

Databáze není ve svém původním významu nic jiného než místo, ve kterém jsou uložena data. Zpracováním a přístupem údajů v databázi je obvykle pověřen nějaký program. Programu spravujícímu data se často říká DBMS (Database Management System), v překladu SŘBD (System Řízení báze dat). DBMS tedy provádí veškeré zpracování, ukládání a získávání dat. (Ponkrác, 2007)

Naprostá většina DBMS staví na tzv. relačním modelu dat. V tomto modelu jsou data uspořádána do databázových tabulek, kde každá tabulka uspořádává údaje určitého druhu. (Ponkrác, 2007)

Většina běžných databází pracuje na principu klient / server. Jinak řečeno databáze pracuje jako služba, kde se celého procesu účastní dvě strany. První je takzvaný server, tedy program, který nabízí své služby ostatním - DBMS, tj. program pro správu dat. Tento program umí základní práci s daty, tedy spravovat, číst, měnit data, hledat v datech a podobně. Běžný uživatel ovšem o tomto serveru ani neví, protože používá program, tzv. klienta, který na práci s daty používá právě server. Základní smysl spočívá v tom, že každý program dělá pouze jednu věc, a tu dělá co nejlépe. Pokud nějaký program potřebuje pracovat s daty, používá volání databázového serveru, který potřebnou práci s daty vykoná za něj. (Ponkrác, 2007)

Jakákoliv práce s databází se tedy skládá zpravidla z následujících operací (Ponkrác, 2007):

- Navázání spojení s databázovým serverem.
- Zaslání příkazu na práci s daty.
- Vyzvednutí výsledných dat.
- Ukončení spojení s databázovým serverem.

Pro komunikaci s databázovým serverem existuje speciální jazyk zvaný SQL (Structured Query Language). Jazyk SQL má zasebou poměrně dlouhý vývoj. Jeho prototyp byl poprvé implementován v roce 1994 společností IBM. Od té doby se stále vyvíjí, a to zejména v případě přidávání nových technologií. V roce 1986, kdy se dočkal

standardizace organizací ANSI, se SQL stal univerzálním nástrojem pro programování databází. (Ponkrác, 2007)

SQLite

SQLite reprezentuje relační databázový systém, jenž je obsažen v jedné malé knihovně. Jedná se o systém, který je velmi šetrný k paměti a poměrně rychlý. Další plus této relační databáze je v její snadné použitelnosti v rámci aplikace, jehož SQLite nepracuje na principu klient / server, jako jiné databázové systémy (např. MS SQL, kde jedatabázový server spuštěn jako samostatný proces), ale představuje knihovnu, jež se přilinkuje k aplikaci a lze ji přes jednoduché rozhraní používat. Využívání SQLite je také výhodné z důvodů jeho šíření pod licencí public domain, jež reprezentuje volné využívání bez nutnosti jakékoliv platby za tento systém. (Vávrů, a další, 2013)

MySQL

MySQL je databázový server vytvořený komerční firmou, který se etabloval jako všudypřítomný server pro webové aplikace. MySQL není ani nejlepší databází, ani nejkvalitnější, a dokonce není ani za všech okolností zadarmo. Přesto, že za její používání pro PHP na webové stránky není nutné platit, při použití databáze MySQL v komerčním prostředí ano. (Ponkrác, 2007)

Ve srovnání s jinými databázovými systémy patří MySQL spíše k těm jednoduchým, ale zase je nenáročný na zdroje počítače a u některých operací nabízí i zvýšení rychlosti. (Ponkrác, 2007)

Samotný PHP dokáže pracovat s mnoha databázovými systémy a MySQL je jen jednou z možností, kterou je možné využít. (Ponkrác, 2007)

PostreSQL

PostreSQL je plnohodnotným relačním databázovým systémem s otevřeným zrojovým kódem. Má za sebou více než 20let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. (Štědroň, 2009)

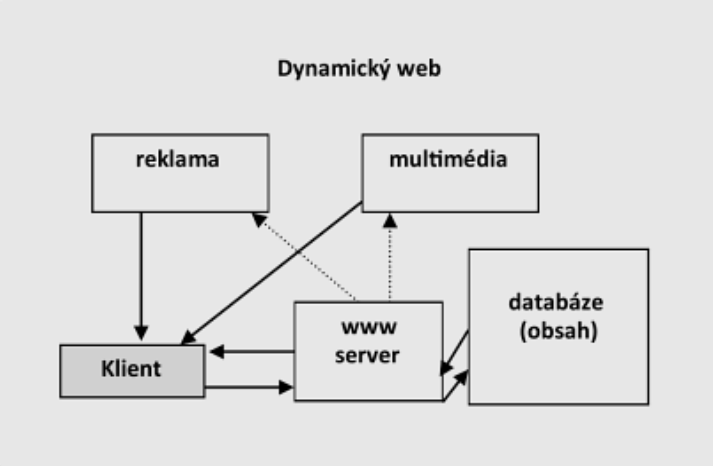
PostgreSQL je šířen pod BSD licencí, která je nejliberálnější ze věech open source licencí. Tato licence umožňuje neomezené používání, modifikaci a distribuci PostreSQL. PostreSQL je možno šířit se zdrojovými kódy nebo bez nich. (Štědroň, 2009)

3.3.3 Statické a dynamické weby

Existují v zásadě dva typy webů z hlediska jejich technického řešení – statické a dynamické. Oba se skládají z webových stránek, které jsou zobrazovány návštěvníku-uživateli, ale velice zásadně se liší tím, jak jsou tyto stránky tvořeny. (Bednář, 2011)

Statický web se skládá z hotových webových stránek, které jsou umístěny na webovém serveru. Tyto stránky jsou odesílány klientovi, který je zobrazuje ve svém webovém prohlížeči. Webový server tak v podstatě slouží jako sklad stránek, který je sám o sobě pasivní a publikace se nijak neúčastní. (Bednář, 2011)

Dynamický webový server pracuje odlišně. Webové stránky se na něm fyzicky nenachází. Každá z nich je vytvořena až na základě požadavku klienta, který je chce zobrazit. Stránka je zkompletována z komponent, které se nacházejí na různých místech, a následovně je odeslána klientovi. Na webovém serveru funguje program, který fyzicky provádí kompletaci webových stránek, jejichž obsah zde většinou není možné naleznout. Ten se totiž nachází na databázovém serveru. Program, který zodpovídá za tvorbu dynamického webu, se nazývá správce obsahu (anglicky Content Management System, zkráceně CMS). (Bednář, 2011)



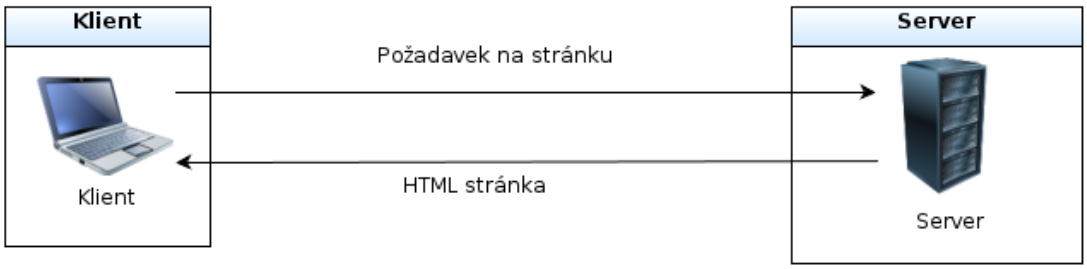
Obrázek č. 2: Dynamický web

Zdroj: (Bednář, 2011)

Rozdíl mezi statickým a dynamickým webem, který by byl dán pouze jejich principy z pohledu uživatele, takřka neexistuje. Z pohledu toho, kdo na webu publikuje informace, ale může být značný. V podstatě platí, že má-li být na statickém webu publikována nová informace – článek, je potřeba více nebo méně tuto informaci manuálně implementovatdo grafické šablony webu, fyzicky umístit na server a pomocí odkazů propojits okolím tak, aby byla pro uživatele webu dosažitelná. I když je možné tuto činnost do jisté míry automatizovat, v podstatě z velké části spoléhá na práci programátora. (Bednář, 2011)

První weby byly založeny na principu statických prezentací. Autoři museli současně znát a umět používat celou řadu technologií, aby dokázali publikovat své články. Protože web byl tvořen řadou stránek na serveru, znamenala každá změna vzhledu, nebo například jen potřeba přidat na stránku reklamu, poměrně náročnou práci. Nedá se říct, že by to bylo nutné, ale z praktických důvodů bylo jednoznačně vhodné, aby byl web udržován a aktualizován pouze z jednoho místa, myšleno z jednoho fyzického počítače. Pokud by na něm pracovali dva a více editorů z více než jednoho místa a snažili se současně například publikovat nové články, mohlo by to mít za následek vážné poškození celého webu anáslednou nedostupnost pro čtenáře. (Bednář, 2011)

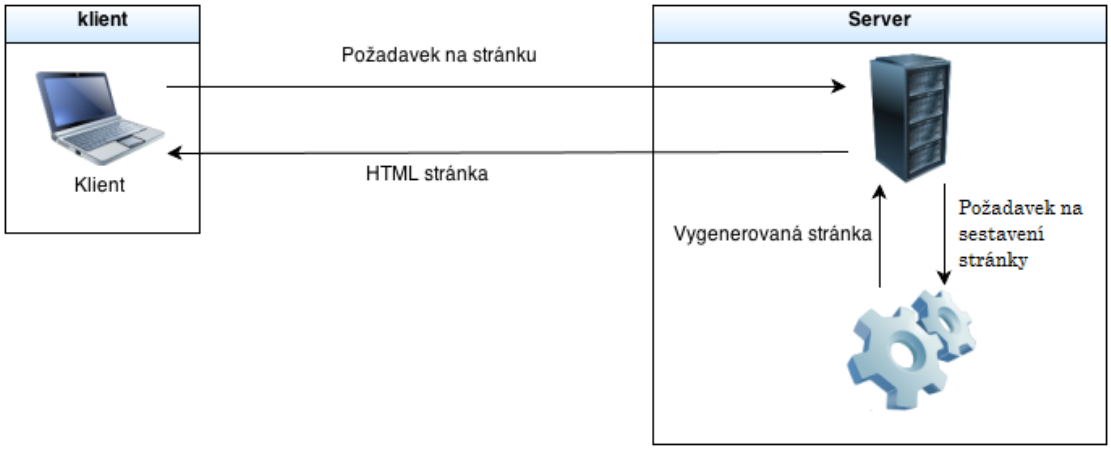
Statická publikace je vhodná spíše pro prezentace firem i médií a je třeba říci, že pro tento účel se na mnoha místech používá dodnes. Pro provoz on-line média je ale z výše popsaných důvodů nepoužitelná. Proto začala on-line média poměrně rychle hledat cesty, jak komlikovaný a na odbornou práci náročný postup statického publikování zjednodušit a především zrychlit. (Bednář, 2011)



Obrázek č. 3: Princip fungování statického webu

Zdroj: (Čápka, 2013)

Dynamický web je tedy obvykle tvořen rozšířením statických webů o prvky a elementy, které se vyhodnotí a vytvoří až po určité uživatelské akci (například vyplnění formuláře). Typickými zástupci technologií používané na tvorbu dynamického webu jsou PHP, PERL, či ASP. Možné je však používat i další technologie, jakýmu jsou například Java, JavaScript a jiné. Převážná většina dynamických webů je implementována pomocí technologie PHP. Principem je vykonání scriptu - příkazu s danými vstupními daty, které může zadat i uživatel, a následné vyhodnocení a sestavení statické stránky, která se zobrazí v okně webového prohlížeče. Technologie, která dynamický web představuje, se dělí především na serverově a klientsky orientované. To podle toho, kde se dynamické skripty vyhodnocují. U serverových technologií je skript vyhodnocený na straně serveru a prohlížeči se tak odešle výsledná statická stránka, sestavená obvykle z (X)HTML, PHP či ASP. Od klienta všechny technologie spočívají v odeslání zdrojového kódu prohlížeče, který tak danou technologii musí rozpoznat, vyhodnotit skripty a až potom sestavit výslednou stránku (JavaScript). Hlavní výhodou dynamického webu je tedy možnost interakce mezi webem a uživatelem. Nevýhoda spočívá jen ve vyšší složitosti a nutnosti znalostí internetových technologií a programování. (Bednář, 2011)



Obrázek č. 4: Princip fungování dynamického webu

Zdroj: (Čápka, 2013)

3.4 Webové prohlížeče

Internetový prohlížeč nebo také anglickým slovem browser je počítačová aplikace, která slouží k prohlížení webových stránek, označovaných jako WWW (World Wide Web). Internetový prohlížeč může mít daleko více funkcí, do prohlížečů jsou integrovány např. aplikace například pro rychlou komunikaci, telefonování přes internet, překladače atd. (Procházka, 2010)

Program umožňuje komunikaci se sadou protokolů HTTP (Hyper Text Transfer Protocol) a HTTP server v rámci zpracování přijatého kódu (HTML, XML, apod.), který podle daných standardů zformátuje a zobrazí webovou stránku na obrazovce počítače. Textové prohlížeče zobrazují obvykle stránky jako velmi jednoduše formátovaný text. Grafické prohlížeče zobrazují formátování stránky včetně zobrazení obrázků. Pro zobrazení některých zvláštních součástí stránky, jako jsou animace vytvoření v technologii Flash nebo Javové applety, je třeba prohlížeč doplnit o specializované zásuvné moduly. (Procházka, 2010)

3.4.1 Rozdělení

Rozdělení webových prohlížečů je založena na tzv. renderovacím jádře. Princip fungování prohlížečů, resp. jejich vykreslovací jader je poměrně sofistikovaným způsobem. V okamžiku vyvolání nějaké aktivity jsou prohlížeči zaslány pakety s informací o zdrojovém kódu načítané stránky, které jsou dále poslány renderovacímu jádru v takovém pořadí, v jakém jsou přijímány. Společně s touto informací jsou renderovacímu jádru sděleny informace o rozlišení prohlížeče a tedy i informace o tom, jak velké pole může zobrazit. Renderevací „motor“ začne zpracovávat informaci ze zdrojového kódu pomocí HTML značek a sestaví si pomyslný strom, ze které pak sestaví samotnou stránku. Stránka však v tomto okamžiku ještě není kompletní, protože se čeká ještě na dodatečné soubory, jako například obrázky. Podle rozlišení monitoru a velikosti okna internetového prohlížeče sestaví finální rozmístění objektů na webové stránce a tím i finální zobrazení uživateli. (Kasner, 2013)

Trident

Trident patří mezi renderovací jádra s nejdelší historií. Jedná se o produkt Microsoftu a je pod licencí MS-EULA k dispozici i ostatním vývojářům. Toho využívají některé méně známé browsery a další jako např. Maxthon nebo Lunascape, které ho využívají v kombinaci s ostatními jádry. Primární uplatnění ale má v prohlížeči Internet Explorer. Ten byl ve svých dřívějších verzích značně kritizován pro nedodržování standardů, což se značně zlepšilo od verze 7, pro kterou Microsoft značně Trident přepracoval. Zajímavostí je, že tento engine je použit v Průzkumníku ve Windows 98, Me a 2000 a také pro Ovládací panely ve Windows XP. (Kasner, 2013)

Obrázek č. 5: Nová podoba loga IE

Zdroj: (Kasner, 2013)

Gecko

Gecko je kompletně open-source engine vyvíjený společnostmi MozillaCorporation a MozillaFoundation, které původně vznikly právě pod záštitou Netscape Communications Corporation. Byl určen pro prohlížeč Netscape, ale dnes je nejvíce rozšířený díky Firefoxu a další produktům od společnosti Mozilla, z nichž nejznámější je e-mailový klient Thunderbird. Také ho můžeme najít například v linuxové verzi správce fotografií Picasa od Googlu. (Kasner, 2013)

Obrázek č. 6: Gecko a MozilaFirefox

Zdroj: (Kasner, 2013)

WebKit

Přestože se WebKit osamostatnil jako open-source teprve v roce 2005, má za sebou už delší historii. Vznikl totiž jako odnož KHTML, renderovacího enginu z projektu KDE. Na jeho vývoji se v minulosti podílely například společnosti Apple, Google, Nokia nebo Blackberry a v současnosti se jedná o nejrozšířenější engine na světě. Hlavní podíl na tom má fakt, že je využit v prohlížečích Google Chrome a Safari. Stejně tak je ale základem i vestavěných prohlížečů na Symbianu nebo na Androidu a pro renderování internetového obsahu ho využívá i Playstation 3, Steam nebo čtečka knih Amazon Kindle. (Kasner, 2013)

Obrázek č. 7: WebKit

Zdroj: (Kasner, 2013)

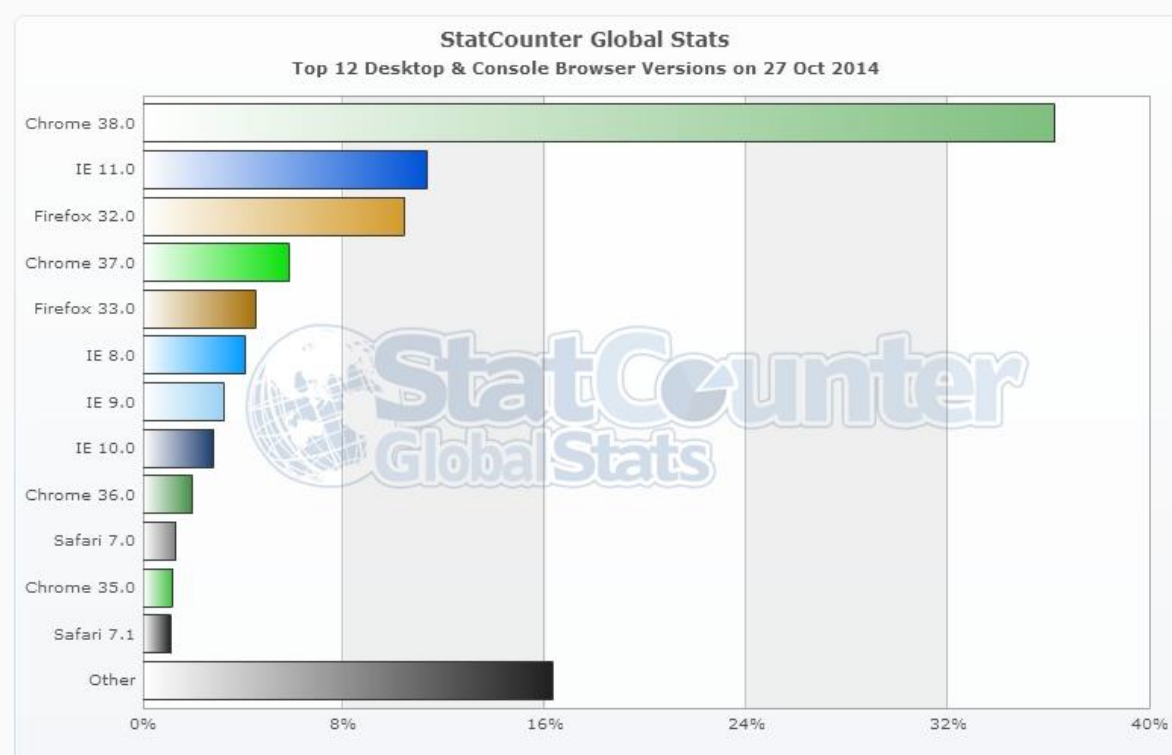
Presto

Presto patří mezi jádra s privátní licencí. Je vyvíjeno společností Opera pro své produkty, konkrétně pro desktopový prohlížeč od verze 7 a dále pak pro mobilní prohlížeče Opera Mobile a Opera Mini. Poprvé bylo představeno v roce 2003. (Kasner, 2013)

Obrázek č. 8: Presto

Zdroj: (Kasner, 2013)

Podle zdroje StatCounter byl k 27.10.2014 nejvíce používaným prohlížečem na světě Chrome ve verzi 38. Na dalších místech se umisťují prohlížeče jako například Firefox či Internet Explorer.



Obrázek č. 9: Světová statistika využití webových prohlížečů

Zdroj: (StatCounter, 2014)

3.5 Technologie HTML5

HTML5 je revize Hypertext Markup Language (HTML), standardního programovacího jazyka pro popis obsahu a vzhledu webových stránek.

HTML5 byl vyvinut s cílem vyřešit problémy s kompatibilitou, které ovlivňují současný standard, tedy HTML 4.0. Jeden z největších rozdílů mezi HTML5 a předchozími verzemi této normy je, že starší verze HTML vyžadují proprietární pluginy a API (Aplication Program Interface). To je důvod, proč webové stránky, které byly vytvořeny a testovány v jednom prohlížeči nemusejí být správně reprezentovány v jiném prohlížeči. HTML5 poskytuje jedno společné rozhraní, aby bylo nakládání s jeho prvky snadnější. Není tak nutné například instalovat Flash plugin v HTML5. (Rouse, 2014)

Jedním z cílů HTML5 je podpora multimdií pro mobilní zařízení. Byly zavedeny nové syntaktické funkce pro podporu videa, audia a canvas tagy. HTML5 také přináší nové funkce, které mohou skutečně změnit způsob, jakým uživatelé pracují s dokumenty. (Rouse, 2014)

3.5.1 DOCTYPE

Každý HTML dokument musí začínat prohlášením, dekrarující o jakou verzi HTML dokumentu se jedná - povinný údaj každého HTML dokumentu.

HTML 4.0 Strict:

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

HTML 4.0 Transitional

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

HTML5

```
<!DOCTYPE html>
```

3.5.2 Strukturální elementy

HTML5 je výrazným zásahem do způsobu, jakým programátoři vytváří webové aplikace. Nejen, že upravuje stávající sémantický význam u HTML značek, ale zavádí mnoho nových, revolučních funkcionalit.

Většina prvků při tvorbě WWW stránky je spojována do logických bloků oddělených elementem <*div*>. Tento element sám o sobě nenese žádný sémantický význam a slouží tak jen k rozdělení stránky pro lepší formátování. Těmto elementům je možné přidávat atributy typu *class* či unikátní atributy typu *id*, pomocí kterých je možná kategorizace elemetů *div*.

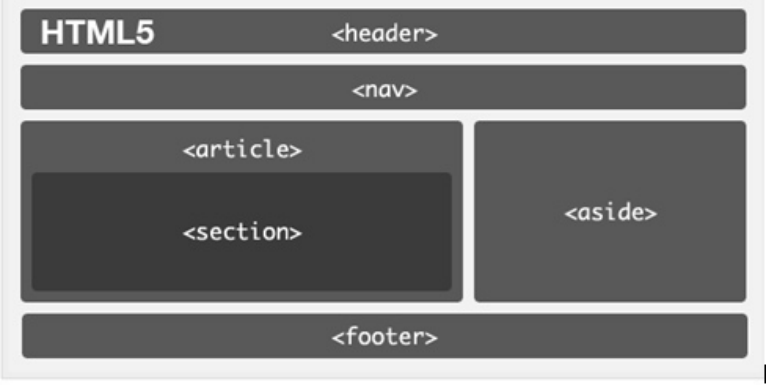
Obrázek č. 10: Struktura dokumentu HTML 4.0

Zdroj: (Patel, 2013)

HTML5 zavádí nové elementy, jejichž cílem je sémantické popsání různých částí dokumentu (T.N.Sharma, a další, 2012):

- <*article*>: Definuje článek.
- <*aside*>: Definuje obsah spojený s hlavním obsahem stránky.
- <*command*>: Definuje příkazové tlačítko, které uživatel může vyvolat.
- <*detail*>: Určuje další podrobnosti, které uživatel může zobrazit nebo skrýt.
- <*summary*>: Definuje viditelný nadpis pro <*detail*> element.
- <*figure*>: Určuje samostatný obsah, jako ilustrace, diagramy, fotografie, výpisy kódu, atd.

- <*figcaption*>: Definuje titulek pro <*figure*> prvek.
- <*footer*>: Definuje zápatí dokumentu.
- <*header*>: Definuje záhlaví dokumentu.
- <*hgroup*>: Skupiny nadpisů <h1> až <h6> elementů.
- <*mark*>: Definuje označený / zvýrazněný text.
- <*nav*>: Definuje navigační odkazy.
- <*progress*>: Představuje průběhu postupu.
- <*section*>: Definuje sekci dokumentu.
- <*time*>: Definuje datum / čas.



Obrázek č. 11: Struktura dokumentu HTML5

Zdroj: (Patel, 2013)

3.5.3 Audio a Video

Nové elementy <*audio*> a <*video*> poskytují podporu pro přehrávání audio a video médií bez nutnosti plug-inů. Video kodeky a zvukové kodeky se používají pro zpracování videa a zvuku a různé kodeky poskytují různé stupně komprese a kvality.

Různé typy prohlížečů nepodporují stejné formáty médií v jejich implementací HTML5 pro video a audio, a to především z důvodu problémů s patenty. Oblast formátů médií na webu značně trpí patentového práva v mnoha zemích, včetně zemí, USA a EU. (Wald, a další, 2013)

Pro přehrání videa pomocí HTML5, je tedy nutné pro každý prohlížeč použít jiný typ formátu:

```
<video controls>
        <source src="nazev.webm" type="video/webm">
        <source src="nazev.mp4" type="video/mp4">
        <p>Chyba, Váš prohlížeč nepodporuje HTML5 video!</p>
</video>
```

WebM

WebM je nativně podporován v prohlížečích obsahující jádro Gecko (Firefox), WebKit (Chrome) a Presto (Opera). Do ostatních prohlížečů je možné instalovat plugin (rozšíření) pro podporuju tohoto formátu videa či audia. (Wald, a další, 2013)

Rozpoznání je možné podle definování typu multimédia:

- *video/WebM* - WebM mediální soubor, který obsahuje video (a možná i audio).
- *audio/WebM* - WebM mediální soubor obsahující pouze zvuk.

Ogg Theora

Jedná se o video a audio kodek, který podporují taktéž pouze prohlížeče s jádrem Triden, WebKit, a Presto. Formát není podporován v aplikaci Internet Explorer. (Wald, a další, 2013)

Rozpoznání je možné podle definování typu multimédia:

- *video/ogg* - WebM mediální soubor, který obsahuje video (a možná i audio).
- *audio/ogg* - WebM mediální soubor obsahující pouze zvuk.

H.264

Audio a video kodek H.264 je nativně podporován prohlížeči Internet Explorer, Safari a Chrome. Opera tento formát nepodporuje. (Wald, a další, 2013)

Browser	H.264	Ogg/Theora	WebM
Internet Explorer	✓	✗	✗
Firefox	✗	✓	✓
Chrome	✓	✓	✓
Safari	✗	✗	✗
Opera	✗	✓	✓

Obrázek č. 12: Kodeky podporované prohlížeči

Zdroj: (nhinkle, 2011)

3.5.4 Local storage

Místní úložiště (local storage) uchovává data bez časového omezení na straně klienta podobně jako cookies (malé množství dat, odeslané serverem, pro dočasné uložení na straně klienta). Rozdíl je však v tom, že local storage dokáže uchovávat velké množství informací. Cookies jsou totiž omezeny velikostí a prohlížeč je odesílá zpět na webový server pokaždé, když se klient vyžaduje nové. HTML5 local storage zůstavají uloženy na klientovi a webové stránky k nim mohou přistupovat pomocí technologie JavaScript. (Trivedi, a další, 2014)

3.5.5 Web SQL

Web SQL je specifikace, která pokrývá ukládání a přístup k datům přes SQL databázi na straně klienta. To umožňuje, že webové stránky komunikují s daty přímo na klientovi a mohou tak pracovat v režimu offline, tedy bez přístupu k síti internet. (Trivedi, a další, 2014)

3.5.6 Drag & Drop

HTML5 přichází s Drag and Drop technologií, která umoňuje dynamicky přenášet soubory z či mimo prohlížeč, jako jsou například fotografie a jiné. Přináší tak nativní podporu přímo do prohlížeče, takže je mnohem snazší ovladání na zařízení, jako jsou mobilní telefony. (Trivedi, a další, 2014)

3.5.7 Input types

HTML element <*input*> se používá k vytvoření interaktivních ovládacích prvků pro webové formuláře, prostřednictvím kterých uživatel komunikuje. Sémantika <*input*> se značně liší v závislosti na hodnotě atributu *type*.

Input type elementy v HTML4 (W3C, 2014):

- *text*: Definuje zadávání textu pomocí pole.
- *password*: Definuje pole pro heslo (znaky jsou maskovány).
- *checkbox*: Checkbox umožní uživateli vybrat nula nebo více z předdefinovaných hodnot.
- *radio*: Definuje přepínač pro výběr z mnoha předdefinovaných hodnot.
- *submit*: Definuje tlačítko odeslat (pro odeslání formuláře).
- *image*: Definuje obrázek jako tlačítko odeslat (submit).
- *reset*: Definuje tlačítko reset (vynulování všech hodnot pro formuláře na výchozí hodnoty).
- *button*: Definuje potvrzovací tlačítko.
- *hidden*: Definuje skryté vstupní pole.
- *file*: Určuje typy souborů, které server přijímá, které mohou být předloženy prostřednictvím nahrání souboru.

Nové input type elementy HTML5 (W3C, 2014):

- *search*: Jednořádkové textové pole pro zadávání vstupního řetězce.
- *email*: Pole pro editaci e-mailové adresy. Vstupní hodnota je validována na správnost tvaru emailové adresy.
- *url*: Pole pro editaci URL (Uniform Resource Locator) adresy. Vstupní hodnota je validována na správný tvyr URL adresy.
- *tel*: Pole pro zadání telefonního čísla.
- *number*: Pole pro zadání desetinného čísla.
- *range*: Ovládání pro zadání čísla, jehož přesná hodnota není důležitá.
- *date*: Ovládání pro zadání data (rok, měsíc a den, bez času).

- *month*: Ovládání pro zadání měsíce a roku, bez času.
- *week*: Ovládání pro zadání čísla týdne a roku.
- *time*: Ovládání pro zadání časové hodnoty bez času.
- *datetime*: Řízení pro zadávání data a času (hodiny, minuty, sekundy a milisekundy) na základě UTC (Coordinated Universal Time) časového pásma.
- *datetime-local*: Ovládání pro zadání data a času, bez časového pásma.
- *color*: Řízení pro určení barvy.

3.5.8 Canvas

Canvas element je element definovaný v HTML kódu pomocí atributu šířky (*width*) a výšky (*height*). Nicméně implementace canvas elementu se neobejde bez použití canvas API (Aplication Program Interface). Toto API rozhraní je napsáno technologií JavaScript, která obsahuje přístupy pro kompletní sadu funkcí pro kreslení, což umožňuje dynamicky generovat grafiku. (Gerchev, 2014)

Zásadní vlatnosi canvas elementu (Gerchev, 2014):

- **Interaktivita -** Canvas je plně interaktivní. Může pružně reagovat na aktivity uživatele při použití klávesnice, myši nebo dotykových událostí. Vývojář tak není omezen pouze na statickou grafiku.
- **Animace -** Každý objekt typu canvas může být animovaný. To umožňuje vytvoření různých úrovní animace. Od jednoduchých grafických objektů až po velice sofistikované grafické objekty.
- **Flexibilita –** Je možné vytvářet jakékoliv grafické prvky. Je možné definovat čáry, tvary, texty, obrázky, atd - s nebo bez animace. Navíc je možné přidání videa či audia přímo do canvas aplikace.
- **Prohlížeč / Podpora platformy -** HTML5 Canvas je všemi hlavními prohlížeči a je dostupý pro širokou škálu zařízení, včetně stolních počítačů, tabletů a chytrých telefonů.

- **Popularita** - Canvas popularita rychle a neustále roste, tudíž je dobrá dostupnost zdrojů.
- **Je to webový standard** - na rozdíl od Flash a Silverlight, je canvas otevřená technologie, která je součástí HTML5.
- **Rozvíjet jednou, spustit všude** - HTML5 Canvas nabízí skvělou přenositelnost. Po vytvoření, na rozdíl od Flash a Silverlight aplikace může běžet prakticky kdekoli - od největších počítačů až po mobilní zařízení.
- **Volně přístupné vývojové nástroje** - Základní nástroje pro tvorbu HTML5 canvas aplikací jsou jen prohlížeč a dobrý editor. Navíc, existuje mnoho volně šiřitelných knihoven a nástrojů, které umožňují pokročilé návrhy.

3.6 Technologie CSS3

CSS3 (Cascding Style Sheets) je specifikace, která se v době psaní této práce neustále mění v čase. Některé části specifikace jsou považovány za stabilní a byly implementovány do moderních prohlížečů, ostatní části by měly být považovány za experimentální a byly jen částečně zavedeny v různé míře. Zbývající části specifikace jsou stále teoretické návrhy a nebyly implementovány vůbec. Některé prohlížeče si vytvořili své vlastní CSS vlastnosti, které nejsou popsány v žádné specifikaci CSS3 a možná ani nikdy nebudou. To vše znamená, že je důležité pochopení, jak je možné CSS3 použít při stylyzování webových stránek a to jak nyní, tak do budoucnosti při procesu normalizace. (Gasston, 2014)

3.6.1 Selektory

Selektory jsou srdcem CSS. Přestože původní specifikace CSS1 měla specifikováno pouze 5 selektorů, CSS2 rozšířila tento „sortiment“ o dalších 12. CSS3 jde ještě dál a to zhruba zdvojnásobením celkového počtu těchto atributů. (Gasston, 2014)

Selektory je možné rozdělit do dvou kategorií. První kategorií jsou prvky, které jsou obsaženy přímo na definovaném elementu (<*p*> prvky odstavců a *href* atributy pro cíle odkazů). Do této kategorie spadají *class*, *type* a *href* atributy. Druhá kategorie obsahuje pseudo-selektory, které působí na prvky nebo informace, které je umístěny mimo strom

dokumentu (například první písmeno odstavce nebo poslední dítě z rodičovského elementu). (Gasston, 2014)

3.6.2 Textové fonty

Ve specifikaci CSS3 byla definována řada nových fontů, semantický označovaných jako *@font-face* (De Silva, 2010). Umožňuje také definování a použití jiných typů fontů, které nejsem přímo specifikované v CSS3. Text lze tak prezentovat uživateli v libovolém písmu. Volnost, kterou nepochybně CSS3 v tomto směru přináší, může působit komplikace. Definování nových stylů se realizuje prostřednictvím extreního souboru (*.ttf), kde velikost souboru může komplikovat rychlost načítání webových stránek. Další komplikací může být autorský zákon, který se může vztahovat na některé fonty, a není možné takto chráněné fonty volně použít. V poslední řadě některé fonty nemají podporu pro českou znakovou sadu. (Janovský, 2014)

3.6.3 Text wrapping

Jedná se o vlastnost definující, jak mají být reprezentovány konce řádků v toku textu. Algoritmus je tedy zaměřen na to, jak se má zachovat, když se daný text nevejde do definovaného objektu. Na výběr je, zda řádky mohou být přerušeny na libovolném místě (možnost *normal*), pro řádky, které nemají být zalomeny (možnost *none*), povolení zalomení řádků mezi dvěma libovolnými znaky (možnost *unrestricted*) či poslední možností, kde o zalomení rozhoduje prohlížeč podle toho, zda existují následující znaky, které je možné zalomit (možnost *suppress*). (Meiert, 2006)

3.6.4 Text stroke

Pomocí této funkcionality je možné textu dát další rozměr. Nově je možné definovat textu šířku (*width*) barvy písma. To znamená, že je možná definovat nejen velikost písma, ale také velikost barevné škály přímo na textu – tzn. šířku barvy. V době psaní této práce je na trhu k dispozici pouze jediný prohlížeč, založený na WebKit jádře – Google Chrome. Proto je prozatím tato funkcionalita definována v CSS3 jako *-webkit-text-stroke*. (Harris, 2011)

Obrázek č. 13: Příklad text-stroke

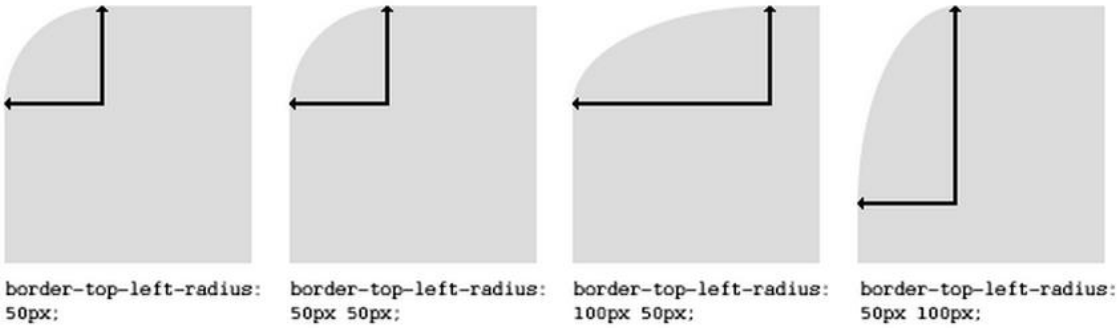
Zdroj: Autor

3.6.5 Skupiny sloupců

CSS3 představuje tzv. vícesloupcové rozvržení dokumentu. To umožňuje rozdělit jeden prvek na tři, čtyři nebo více sloupců. Umožňuje určit počet sloupců, mezer mezi sloupci a přidat řádek mezi sloupci. Vícesloupcové rozvržení je simulace, které by se dalo přiblížit např. rozložení textu v novinách a časopisech, kde jeden dlouhý příběh „proudí" z horní do dolní části jednoho sloupce, z horní části do dolní části druhého sloupce, a tak dále. Avšak není možné nastavení různé šířky pro každý sloupez individuálně, všechny mají stejnou šířku. Využití více sloupců je převážně žádoucí při použití dlouhých textů. Místo jednoho opravdu dlouhého seznamu, je možné text více granularizovat do více sloupců a ušetřit tak cenný vertikální prostor. Základní syntaxe je velice jednoduchá. Použitím vlastnosti *column-count* je možné nastavení počet sloupců, *column-gap* vlastnosti nastavit mezery mezi sloupci a sloupci, pravidlo *column-rule* pro nastavení „čáry" pro oddělení sloupců. (McFarland, 2013)

3.6.6 Zaoblené rohy

Rounder corners, neboli zaoblené rohy, mohou být vytvořeny nezávisle pomocí čtyř samostatných definic - *border-bottom-left-radius, border-top-left-radius,* atd. Nebo na všech čtyřech rozích současně pomocí zkráceného zápisu - *border-radius*. Velikost každé definice může být definována pomocí jedné nebo dvou hodnot, které jsou vyjádřeny buď absolutně (pomocí definování velikosti v pixelech - px), nebo relativně (definice pomocí procent). Jsou-li definovány dvě hodnoty, jsou využity v pořadí nejprve horizontální a pak vertikální, což určuje zakřivení rohu. (Storey, 2012)



Obrázek č. 14: Příklad definování zaoblených rohů

Zdroj: (Storey, 2012)

3.6.7 Stíny

Stíny jsou dalším společným rysem moderních webových řešení. Stíny přidávají prvek sytosti, ale mohou také zlepšit čitelnost (při správném použití) titulků či nadpisů. *Text-shadow* atribut byl technicky součástí CSS2, ale teprve nedávno byl podpořen hlavními prohlížeči. (Harris, 2011)

Text-shadow atribut má čtyři základní parametry (W3C, 2014):

1. *offset-x*: Určuje vzálenost na ose x (z leva do prava). Definuje tedy horizontální vzálenost stínu od textu. Kladná hodnota pohybuje stínem doprava a záporná hodnota pohybuje stínem doleva od textu.

2. *offset-y*: Ukazuje, jak na ose y (zhora-dolů), bude stín vzdálen od původního textu. Pozitivní hodnota pohybuje stínem dolů, záporná hodnota pak stínem nahoru od textu.

3. *blur*: Upřesňuje rozmazání poloměru stínu. Pokud je hodnota 0px, stín není rozmazán a vypadá stejně jako v základní definici. Obecně platí, že velikost rozmazání odpovídá hodnotě z nejdelší *offset* (vzdáleností). To umožňuje, aby byl stín rozpoznatelný a nestal se tak rozmazaným.

4. *color*: Definuje barvu stínu. Standardní barva stínu je šedá, ale pro tvorbu speciálních efektů je připravena velká škála barev. Rozmazání (*blur*) má tendenci

zesvětlit stín. Pokud je nutné aplikovat velké rozmazání stínu, je možné použít totožnou barvu jako u textu. V případě, že stín nebude rozmazaný moc, je žádoucí barvu zesvětlit pro lepší čitelnost textu.

Velikost stínu se určuje nepřímo s kombinací vzdáleností stínu a jeho rozmazáním. Pro dosažení chtěného stínu je tedy nutné experimentovat. Speciálním využitím může být použití stínu pro zvýraznění (vystoupení) textu pomocí nanesením kontrastní barvy. Všechny poslední modely prohlížečů, kromě Internet Explorer, podporují funkcionalitu *text-shadows.* Při definici ve stylech CSS3 tak nejsou vyžadovány speciální prefixy. (Harris, 2011)

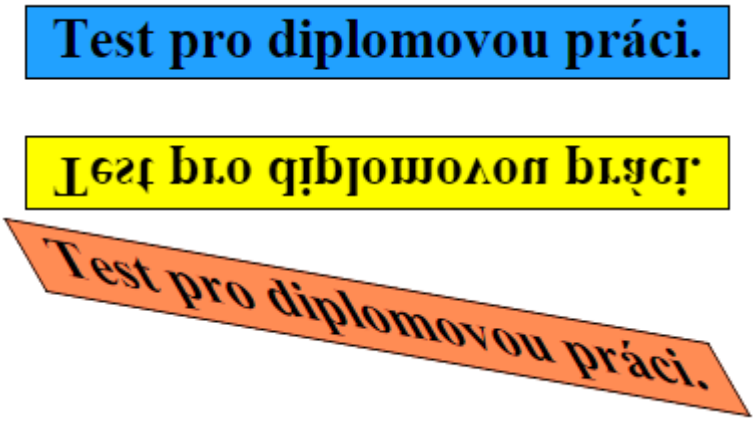
Obrázek č. 15: Příklad stínování textu

Zdroj: Autor

3.6.8 Transformování

Použitím CSS3 transformace je možné objekty libovolně otáčet, měnit velikost či zkosit. To může být nápomocí pro vyniknutí objektů umístěných na webu a zabránění tak jejich „nudnosti“. Atribut umožňuje použít matematické transformace na kteroukoli část stránky, tedy části označené v HTML kódu jako <*div*>. Při použití transformace na element, je nutné použít jeden nebo více z následujících parametrů a popsat tak typ transformace (Hogan, 2011):

- *translate*: Posune objekt z výchozí polohy.
- *rotate*: Otočení obrázku kolem středu osy.
- *scale*: Změní velikost objektu. Standardní proporcionální změna velikosti a to jak horizontální, tak vertikální.
- *skew*: Umožňuje naklonit prvek pod nějakým úhlem.



Obrázek č. 16: Příklad transformace textu

Zdroj: Autor

3.6.9 Gradient

Nový přírůstek do CSS3 je schopnost definovat *gradient* (prolnutí z jedné barvy do druhé) jako obrázek na pozadí stánky. Pro definici gradientu je použít jiných CSS3 vlastností, a to definování pomocí RGBA barev. V předchozích verzích CSS, bylo pozadí definováno přímo pomocí anglických názvů barev, pomocí Hexa kódu nebo barevné definice RGB (Red, Green, Blue), kde se definují jednotlivé hodnoty pro červenou, zelenou a modrou. Tyto definice barev jsou samozřejmě stále zcela platné v CSS3, ale je možné použít nové pravidlo, RGBA (Red, Green, Blue, Alpha), které je analogické RGB. RGBA umožňuje definovat tzv. Aplha kanál, který definuje velikost průhlednosti (anglicky opacity). (Harris, 2011)

Obrázek č. 17: CSS3 Gradient

Zdroj: Autor

4 Vlastní řešení

Praktická část bude zaměřena na stěžejní problematiku HTML5. Revoluční vlastností této technologie je přesun dat ze serverové strany na stranu klienta. Pomocí technologií ze skupiny HTML5 budou realizovány možnosti uložení dat na straně klienta. Díky tomuto způsobu ukládání dat je možné vytvářet tzv. offline aplikace, tedy aplikace které nutně nepotřebují přístup k síti internet a budou také předmětem praktické části. Bude vytvořena webová aplikace, která bude obsahovat uživatelský formulář s možností nastavení uživatelského pozadím, bude dostupná na adrese http://berry.ic.cz a bude součástí této práce. Na webové aplikaci budou demonstrovány výše uvedené způsoby ukládání dat a bude provedeno otestování v nejpoužívanějších internetových prohlížečích. Nejdříve je však nezbytně nutné vysvětlit, co se pod termínem HTML5 skrývá.

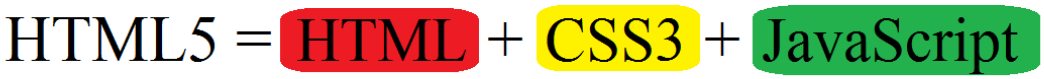
Nástup HTML5 je dlouhodobý proces, který započal v roce 2007, a finální specifikace by měla být představena v blízké době, tedy v době psaní této diplomové práce. Avšak není hned nezbytné začít bezhlavě přepisovat webové aplikace novou technologií. Na druhou stranu je dobré mít povědomí, kudy daná technologie povede a jaké možnosti a rizika sebou přináší.

Během vývoje HTML5 docházelo k velkému zájmu ze strany programátorů o tuto technologii a stalo prakticky nevyhnutelné, a totiž že pod označení HTML5 se schovalo nepřeberné množství specifikací a technik, které s klasickým HTML jako takovým - značkovacím jazykem pro hypertext, nemá prakticky nic společného. Vzniká tedy nový „buzzword“ (termín označující obecnou lidskou známost) pro tuto technologii a je nutné si říci, co si pod pojmem HTML5 představit.

Příčinou vzniku tohoto buzzwordu má porovnávání HTML5 s technologií FLASH. Dokonce někteří CEO (výkonný ředitel, ang. Chief Executive Office) velkých společností se nechali slyšet, že upřednostňují technologii HTML5 před již velice rozšířenou technologií FLASH (Zeldman, 2011). Dokáže ale samotné HTML5, tedy značkovací jazyk, nahradit tuto technologi? Bez nějakých hlubších znalostí o dané problematice je odpověď jasná – NE. Otázkou, která se nevyhnutelně nabízí, je zda dokáže HTML5

v kombinaci s ostatními technologiemi nahradit FLASH. Ano, v kombinaci s CSS3 a technologí JavaScript je to možné.

Vzniká tedy jednoduchá rovnice, která říká, co se skrývá pod pojmem HTML5.



Obrázek č. 18: Definice technologie HTML5

4.1 Webová aplikace

Webová aplikace bude založena na práci s daty, které budou v dalších částech této práce zpracovávány na straně klienta. Bude vytvořen tedy HTML5 dokument, kde uživatel bude mít možnost odeslat zprávu a budou od něj požadovány údaje o jméně, telefonním čísle a emailu. Maximální velikost zprávy bude nastavena na 160 znaků. Pole s informací o jméně, emailu a zprávě budou povinnými poli. Aby byla webová prezentace líbivější, bude implementována možnost přepnutí barevného pozadí uživatelem.

Formuláře jsou nezbytnou součástí při tvorbě interaktivního webu. Jsou interaktivní, protože jsou po koncovém uživateli vyžadovány patřičné informace podle toho, k čemu je formulář určen.

Před samotným vytvořením formuláře je nejprve nutné si nadefinovat HTML5 stránku jako takovou. Jak se definice HTML5 stránky liší od předchozích verzí je popsáno v kapitole 3.5.1 DOCTYPE.

Vzniká tedy kostra HTML5 stránky, kde je nadefinována znaková sada pro český jazyk (*lang="cs"*), způsob čtení textu zleva do prava (*dir="ltr"*) a znakové kódování (*charset="utf-8"*). Takovýmto nadefinováním se pak řídí celá webová stránka.

```
<!DOCTYPE html>
<html lang="cs" dir="ltr">
<head>
  <meta charset="utf-8" />
</head>
<body>

</body>
</html>
```

Obrázek č. 19: Struktura HTML5 dokumentu

Nyní je již možné začít s vytvořením formuláře. Sekce, která je definována jako *<body></body>*, je určena pro definování základní kostry webu. Do této části je vytvořena podkostra, značená jako s jednoznačnou identifikací pomocí přiřazení *id="contact-form"*. Takto nadefinované části webové stránky jsou pro lepší orientaci přidány dvě úrovně nadpisů *(<h1></h1>, <h2></h2>)* pro popsání týkající se problematiky a unikátně identifikovatelná část, pro příslušný formulář.

```
<div id="contact-form">
  <h1>Formulář</h1>
  <h2>DP - praktická část (Tvorba formuláře)</h2>
  <form id="myForm">
  </form>
</div>
```

Obrázek č. 20: Definice formuláře v HTML5

Prvním vstupním prvkem formuláře je jméno uživatele, které je označeno elementem *<input type="text"/>*. Jedná se o standardní element využívaný i v ostatních verzích HTML. Aby uživatel věděl, že do této kolonky je nutné uvést i příjmení, bylo mu napovězeno pomocí elementu *placeholder="Josef Novák"* (Pozn. Jedná se o fiktivní osobu) a je tak ihned zřetelné, že je vyžadováno celé jméno. Dále je pole označeno jako *required*, tedy vyžadovaný. Skutečnost je uživateli představena pomocí zobrazení hvězdičky přímo u kolonky. Bez zadání této hodnoty neproběhne samotné odeslání formuláře. Jelikož je prvek umístěný ve formuláři jako první, při prvotním načtení stránky je umístěn kurzor právě do tohoto prvku (*autofocus="autofocus"*) a uživateli je tak zpřístupněno vyplňovaní bez předchozí aktivity.

```
<label for="name">
  Jméno: <span class="required">*</span>
</label>
<input  type="text" id="name" name="name"
        placeholder="Josef Novák"
        required="required"
        autofocus="autofocus"
/>
```

Obrázek č. 21: Definice položky: jméno

Druhým údajem v pořadí je informace o uživatelském emailu. HTML5 pro definici emailu obsahuje atribut označený jako *<input type="email"/>*. Nový element oproti starším verzím odstraňuje nutnost vytvoření regulárního výrazu pro ověření validity emailové adresy. Stejně tak jako u kolonky se jménem je uživateli napovězen tvar, jaký je vyžadován pro správné zadání (*placeholder="josef.novak@seznam.cz"*) a v poslední řadě je kolonka označena taktéž jako *required*, tedy vyžadována.

```
<label for="email">
  Email: <span class="required">*</span>
</label>
<input  type="email" id="email" name="email"
        placeholder="josef.novak@seznam.cz"
        required="required"
/>
```

Obrázek č. 22: Definice položky: email

Další informací, která není tentokrát definována jako povinná, je informace o kontaktním čísle. HTML5 má pro tento případ definici *type="tel"*. Při vyplňování této hodnoty pomocí osobního počítače typu notebook či desktop je hodnota reprezentována jako text. Rozdíl je patrný až při přístupu z mobilního zařízení, jako je například tablet či chytrý telefon, kde při zadávání telefonního čísla dojde k vysunutí klávesnice s možností zadání čísla. Dochází tak ke komfortu při zadávání údajů.

```
<label for="telephone">Telefon: </label>
<input type="tel" id="telephone" name="telephone" />
```

Obrázek č. 23: Definice položky: telefon

Poslední hodnoutou, která bude opět vyžadována, bude text zrávy. Opět se jedná o vyžadovanou informaci, tedy typ *required*. Element pro tento typ informace je zvolen jako *type="message"*. V tomto případě se nabízí také definování prvku pomocí *type="text"*, jako je tomu v případě jména. Označení kolonky jako *message* však umožňuje uživateli využívat klávesy *ENTER*, pro lepší formátování textu, jako je například tvorba odstavců. Při použití definice jako *text* by si formulář reagoval pokusem o odeslání informací a to je v tomto případě nežádoucí. Takto definovanému prvku je možné nastavit minimální (*data-minlength*) a maximální (*data-maxlenght*) počet znaků uvedených ve zprávě. Při nesplnění definovaných podmínek není formulář zpracován a je nutné zprávu přizpůsobit. Jelikož je tato hodnota definována jako *required*, není potřeba definovat spodní hranici počtu znaků a je tak definovaný pouze maximální počet znaků, tedy 160.

```
<label for="message">
   Zpráva: <span class="required">*</span>
</label>
<textarea id="message"
          name="message"
          placeholder="Zpráva může obsahovat maximálně 160 znaků"
          required="required" data-maxlength="160">
</textarea>
```

Obrázek č. 24: Definice položky: zpráva

Prvky pro vyžadující informace jsou již definovány, nyní je potřeba definovat prostředek pro odeslání formuláře. Element *<input type="submit"/>* definuje graficky jednoduché tlačítko, které je možné doplnit textem pomocí hodnoty, tedy například *value="Odeslat!"*. V případě, že všechna zadaná data z formuláře jsou validní (*jméno, telefon* atd.), tedy splňují definovanou strukturu, znakové omezení atd., dojde po stisknutí tlačítka myši na tento objekt o pokus k odeslání informací. V této fázi kliknutí na tlačítko pouze smaže data z formuláře, jelikož není definováno, jak se data mají zpracovat.

Přidána je také informace, jednoznačně identifikována v HTML jako *id="req-field-desc"*, která vysvětluje význam znaku hvězdička (*), tedy že jsou položky označeny tímto znakem povinná (*Povinná pole!*).

```
<input type="submit" value="Odeslat!" id="submit-button" />

<p id="req-field-desc"><span class="required">*</span> Povinná pole!</p>
```

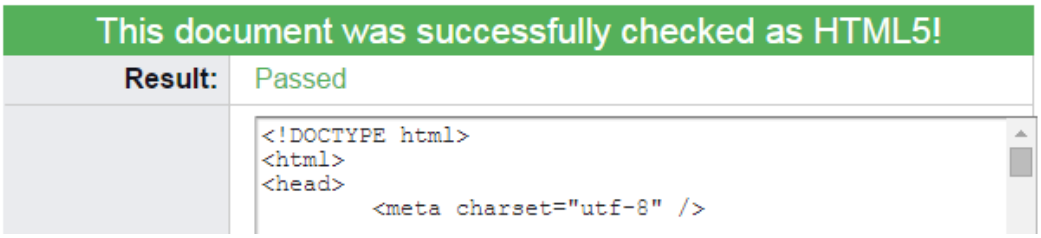
Obrázek č. 25: Definice tlačítka a povinných polí

Pro další část praktické části je připravena také možnost definování barevného pozadí uživatelem. Tato možnost je uživateli zprostředkována pomocí rozbalovacího seznamu, které je unikátně označené jako *id="color_list*. Seznam je rozdělen na dvě skupiny. První skupinou, označenou jako *<optgroup label="Základní barvy">*, je možnost volby absolutní základní barvy, které jsou přímo definovány v HTML5 standardu. Druhou možností, označenou jako *<optgroup label="Gradient">*, je možnost výběru barevného gradient pozadí (problematika popsána v kapitole 3.6.9 Gradient).

```
<select id="color_list">
  <optgroup label="Základní barvy">
    <option value="white">Bílá</option>
    <option value="gray">Šedá</option>
    <option value="orange">Oranžová</option>
    <option value="purple">Fialová</option>
    <option value="blue">Modrá</option>
    <option value="green">Zelená</option>
    <option value="red">Červená</option>
  </optgroup>
  <optgroup label="Gradient">
    <option value="(linear, left top, left bottom, color-stop(0%,rgba(2,2,2,1)),
                    color-stop(100%,rgba(173,173,173,1))"
    >Gradient</option>
  </optgroup>
</select>
<input type="button" value="Nastavit barvu pozadí"/>
```

Obrázek č. 26: Definice seznamu barev

V této fázi je provedena kontrola validity HTML5 kódu. V případě, že by kód nebyl validním, důsledkem by mohlo být špatná interpretovatelnost v různých prohlížečích.



Obrázek č. 27: Validace HTML5 dokumentu

Zdroj: (W3C, 2014)

Takto definovaná HTML5 stránka vypadá po spuštění v internetovám prohlížeči následovně:

Obrázek č. 28: Zobrazení čistého HTML5 dokumentu v prohlížečí

V druhé části vytváření formuláře je nutné přehledně danou webovou stránku prezentovat. Pro nastylování je využito funkcionality z rodiny HTML5 a tou je CSS3. Připojení stylů je realizováno externím souborem. Je tedy nutné přidání této informace do HTML5 kódu do části označené jako *<head></head>*.

```
<link href="css/style.css" rel="stylesheet" type="text/css" media="screen" />
```

Obrázek č. 29: Definice kaskádových stylů

Díky unikátnímu definování částí webové aplikace v HTML5 kódu pomocí označení *id=““*, je nyní možné takto definovanou část stylizovat nezávisle na sobě. U této problematiky je však nutné brát zřetel na webové prohlížeče. Na trhu se nachází prohlížeče, které mají odlišná renderovací jádra (viz kapitola 3.4.1 Rozdělení), a je tak žádoucí definovat každou skutečnost zvlášť pro každé jádro, resp. prohlížeč. Prohlížeče používající jádro WebKit používají v CSS3 prefix -webkit-. Prohlížeče typu Gecko pak

prefix -moz. Nesprávným definováním kaskádových stylů může být špatná interpetovatelnost prohlížečem, či dokonce její nefunkčnost. Proto je důležité neopomenutí této skutečnosti při stylizování webových aplikací.

U formuláře je využita funkcionalita CSS3 tzv. zaoblených rohů (kapitola 3.6.6 Zaoblené rohy) označována jako *border-radius*, která je definována hodnou 15px. Jelikož je uvedena pouze jedna hodnota, znamená to, že vechny rohy definované v části označené jako *id="contact-form"*, budou mít na ose X a ose Y zaobleny rohy na tuto hodnotu. Podstatnou částí je umístění formuláře na webové stránce, ta byla nastavena jako *position: relative;* to má za následek, že celý tento blok bude zarovnán relativně podle velikosti okna v prohlížeči. Znamená to, že v případě, že uživatel bude dynamicky měnit velikost okna, formulář se bude automaticky přizpůsobovat těmto změnám.

```
#contact-form {
   background-color:#F2F7F9;
   width:465px;
   padding:20px;
   margin: 100px auto;
   border: 6px solid #8FB5C1;
   -moz-border-radius:15px;
   -webkit-border-radius:15px;
   border-radius:15px;
   position:relative;
}
```

Obrázek č. 30: Definice stylů formuláře

Výsledná webová stránka po implmenetaci kaskádových stylů pak vypadá následovně:

Obrázek č. 31: Webová aplikace

4.2 Local Storage

Nadefinovaný formulář (viz kapitola 4.1 Webová aplikace) bude nyní rozšířen o funkcionalitu lokálního úložiště, tzv. local storage (viz kapitola 3.5.4 Local Storage). Tato funkcionalita HTML5 je naprosto revolučním řešením při implementaci webů. Pomocí ní je možné ulehčit práci webovému serveru a zpříjemnit práci uživatelům webových stránek. Local storage se hodí k ukládání jakýchkoliv necitlivých osobních dat

na straně klienta. Výhoda se ukrývá v uložení dat na klientovi a tak zabráněním jejich zpracování na serveru, kde si uživatel nemusí být jistý, jakým způsobem jsou data zpracována.

Implementace local storage je realizováno skriptovací technologií z řady HTML5 – JavaScriptem, který umožňuje dynamizaci webových stránek. Implementace JavaScriptu se realizuje v hlavičče HTML5 kódu, tedy v *<head></head>* pomocí značky *<script type="text/javascript"></script>*.

V první fázi je umožněno na webové aplikaci libovolně měnit grafiku pozadí pomocí předdefinovaných hodnot z minulé kapitoly (4.1 Webová aplikace). Nejprve je nutné vytvořit funkci, která nastaví pozadí. Ta dále rozhoduje, zda uživatel již danou stránku navštívil a modifikoval pozadí, či se uživatel na dané stránce nachází poprvé. Znamená to tedy kontrolovat na straně klienta, zda již nedošlo k uložení nějaké hodnoty pro příslušnou funkcionalitu (*localStorage.bgColor.lenght* == 0). V případě, že není k dispozici žádná takováto uložená hodnota, tedy že délka hodnoty je rovna nule, je nastavena barva pozadí na základní hodnotu, tedy bílou (white). V opačném případě, tedy že uživatel již provedl konfiguraci, je nastaveno pozadí na příslušnou barvu a také je tato hodnota nastavena v předdefinovaném seznamu, ze kterého uživatel vybíral danou konfiguraci.

```
$(document).ready (
    function loadBackground(){
        if (localStorage.bgColor.length == 0)
            {
                localStorage.bgColor = "white";
            }
        else
            {
                document.body.style.background = localStorage.bgColor;
                document.getElementById("color_list").value = localStorage.bgColor;
            }
    }
)
```

Obrázek č. 32: Definice načtení local storage

Pro první použití, tedy možnosti nastavené barvy pozadí, je nutné tuto akci vyvolat. V předchozí kapitole (4.1 Webová aplikace) bylo definováno tlačítko pro možnost nastavení barevného pozadí z předdefinovaného seznamu. Na toto tlačítko je nově implementována aktivita, která po vyvolání akce, tedy po stisku (*onClick*), zavolá definovanou funkci (*setBackground*).

```
<input type="button" value="Nastavit barvu pozadí" onClick="javascript:setBackground();" />
```

Obrázek č. 33: Definice nastavení pozadí

Po zavolání funkce je načtena hodnota barvy, kterou si uživatel vybral a také její popisek, který danou hodnotu interpretoval. Obě hodnoty jsou uloženy do local storage a po aktualizaci stránky či budoucí znovu otevření stránky je načtena poslední známá konfigurace.

```
function setBackground(){
    localStorage.bgColor = document.getElementById("color_list").value;
    document.body.style.background = localStorage.bgColor;
}
```

Obrázek č. 34: Definice ukládání do local storage

Analogicky je možné uchovávat data z formuláře. Nicméně v tomto případě je nežádoucí uchovávat data po odeslání formuláře. Odlišností od načítání hodnot z předdefinovaného seznamu je, že je nežádoucí vyžadovat po uživateli po každé změně informací ve formuláři stisknutí tlačítka pro uložení. Je tedy potřeba identifikovat unikátní jména prvků formuláře (*#name, #email, #telephone, #message*), tedy definovanými *id* (viz kapitola 4.1 Webová aplikace), a pomocí nich při jakékoliv změně (*.on('keyup')*) ukládat (*setItem*) právě aktuální zadaný text obsažený v dané buňce. Protože však je ukládání realizováno agregovaně, tedy pro všechny položky z formuláře, je potřeba dynamicky ukládat unikátní označení buňky (*$(this).attr('id')*) formuláře a její hodnotu (*$(this).val()*).

```
$('#name, #email, #telephone, #message').on('keyup', function(){

    <!--console.log('key up');-->
    <!--ulozeni do locaStorage-->
    localStorage.setItem($(this).attr('id'), $(this).val());

});
```

Obrázek č. 35: Definice technologie HTML5

Funkcionalita local storage je implementována v tomto případě, aby byla data zachována pro případ, že dojde k nechtěné aktualizaci stránky či popřípadě k fyzickému zavření prohlížeče. Je tedy nutné při znovunačtení stránky naplnit všechny (*$(this).val(localStorage.getItem($(this).attr('id')));*) položky formuláře posledními známými daty z local storage pomocí jejich unikátních hodnot *id*.

```
$('#name, #email, #telephone, #message').each(function(){

    <!-- pokud nastane refresh stranky, zobrazim posledni data-->
    console.log('stranka byla aktualizovana a jsou nactena data');
    $(this).val(localStorage.getItem($(this).attr('id')));

});
```

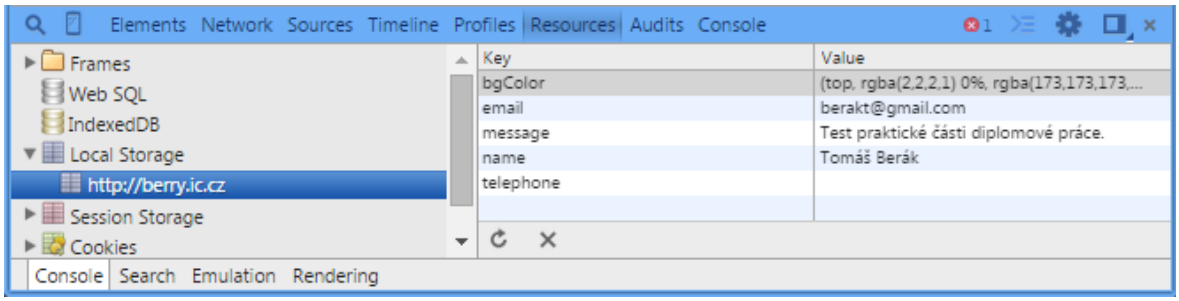
Obrázek č. 36: Definice načtení z local storage

Jak bylo řečeno na začátku této kapitoly, je nežádoucí uchovávat uložená data z formuláře po jeho odeslání. Aktivita odeslání je vyvolána sisknutím tlačítka *Odeslat!*, které je součástí formuláře s identifikátorem *id="myForm"*. Znamená to tedy, že při stisku tlačítka je nutné vymazat všechny uložené hodnoty pro všechny položky z local storage (*localStorage.removeItem($(this).attr('id'));*) a také provést reset těchto polí ve formuláři (*$('#myForm')[0].reset();*).

```
$('#myForm').submit(function(s){
    s.preventDefault();
    console.log('potvrzuji a vymazavam localStorage');
    <!--vymazu localstorage, pokud jsou data odeslana -->

    $('#name, #email, #telephone, #message').each(function(){
        localStorage.removeItem($(this).attr('id'));
        $('#myForm')[0].reset();
    });

});
```

Obrázek č. 37: Definice vymazání local storage

Všechny aktuálně uložené hodnoty local storage je možné zjistit pomocí prohlížeče, kde je webová aplikace spuštěna.



Obrázek č. 38: Local storage

4.3 Web SQL

Webové databáze (viz kapitola 3.5.5 Web SQL) jsou hostované a přetrvávající uvnitř internetového prohlížeče, tedy na klientovi. Tím umožňují vytváření aplikací se sofistikovanější strukturou dotazů. Práce s databázemi je poněkud komplikovanější než např. s local storage, nicméně se jedná o stejný princip fungování. Nespornou výhodou Web SQL je pak, že dokáží pracovat i v režimu offline.

V této části bude opět využita naprogramovaná webová aplikace (viz kapitola 4.1 Webová aplikace), pomocí které bude implementována Web SQL databáze tak, aby se konfigurační data pro formulář ukládala na stranu klienta, tedy přímo v prohlížeči, a byly tak dostupná například i právě v offline režimu.

Web SQL je založena na transakčním přístupu. Tedy není možné provádět SQL příkazy mimo transakce. To znamená, že při probíhající transakci jsou využívané objekty v databázi blokovány pro ostatní procesy. Aby daná transakce věděla, jakou databázi má využít, a tedy k jakým datům má být přistupováno, je nejprve nutné databázi definovat. Do pomocné proměnné jsou nadefinovány parametry databáze. Je definována databáze „Test“, která bude mít maximální kapacitu 5 MB.

```
var databaseOptions = {
        name: "Test",
        version: "1.0",
        displayName: "Hello world",
        maxSize: 5 * 1024 * 1024
    };

    var database = openDatabase(
        databaseOptions.name,
        databaseOptions.version,
        databaseOptions.displayName,
        databaseOptions.maxSize
    );
```

Obrázek č. 39: Definice Web SQL databáze

Web SQL databáze realizuje databázové transakce pomocí čtyř proměných. První z nich je samotný příkaz, který má databázový stroj vykonat, tedy například operace INSERT, UPDATE či DELETE. Druhou proměnnou je hodnota parametru, která má být zpracována. Tedy například hodnota parametru vkládané do databázové tabulky. V případě, že definovaná operace je vykonána úspěšně, je použit třetí, výstupní parametr. V opačném případě je využit čtvrtý parametr, tedy pro neúspěšné zpracování. V tomto případě je vytvořena funkce, která nám do logu prohlížeče tuto informaci ohlásí. Konkrétně vypíše chybovou zprávu a její kód.

```
function errorHandler(transaction, error){
    console.Log('ERROR:' + error.message +', code: ' + error.code);
    return true;
};
```

Obrázek č. 40: Definice logovací funkce

Stejně tak, jako při využití local storage (viz kapitola 4.1 Local storage), je nutné při první načtení aplikace ověřit dostupnost databázové tabulky, se kterou má aplikace pracovat, popřípadě jí vytvořit (*CREATE TABLE IF NOT EXISTS*). Tabulka obsahuje primární klíč *id*, který je nastaven na hodnotu AUTOINCREMENT, tedy automaticky navyšující. Dále obsahuje hodnotu *value*, ve které jsou ukládány názvy hodnot barevného pozadí, které jsou nakonfigurováno uživatelem.

```
$(document).ready (
    function loadBackground(){
        database.transaction(
            function( transaction ){
                transaction.executeSql(
                    "CREATE TABLE IF NOT EXISTS background (" +
                        "id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
                        "value TEXT NOT NULL" +
                    ");"
                );

            }
        );
```

Obrázek č. 41: Struktura tabulky background

Protože je nastavena velikost databáze na 5MB, je nežádoucí uchovávat historické řady o nastavených hodotách pozadí. Je tedy provedena transakce, která odmaže (*DELETE*) všechna historická data kromě poslední (*id <> MAX(ID)*), tedy aktuální.

```
database.transaction(
    function(transaction){
        transaction.executeSql(
            'DELETE FROM background where id <> (SELECT MAX(id) FROM background);',
            null, null, errorHandler
        );
    }
);
```

Obrázek č. 42: Vymazání historických dat tabulky background

V případě, že bylo k aplikaci v minulosti již přistoupeno a byla provedena konfigurace pozadí, je načtena (*SELECT*) poslední známá hodnota (*MAX(ID)*) z databáze a dále je hodnota textově zobrazena v seznamu pro výběr (grafické zpracování aplikace viz

kapitola 4.1 Webová aplikace). V logu internetováho prohlížeče je pak inromace o načítané hodnotě zprostředkována uživateli.

```
database.transaction(
        function(transaction){
            transaction.executeSql(
                'SELECT value FROM background WHERE id = (SELECT MAX(id) FROM background);', [],
                function(transaction, result){
                    for (var i=0; i < result.rows.length; i++){
                        var row = result.rows.item(i);
                        console.log('Načítám poslední známou konfiguraci, hodnota pozadí: ' + row.value);
                        document.getElementById("color_list").value = row.value;
                        document.body.style.background = row.value;
                    }
                },
                errorHandler
            );
        }
);
```

Obrázek č. 43: Načtení nastaveného pozadí

Při vyvolání konfigurace pozadí, ať už se jedná o prvnotní změnu či úpravu aktuálního nastavení, je nutné tuto informaci uchovat v databázi. Znamená to tedy, že při vyvolání uložení prostřednictvím tlačítka *Nastavit barvu pozadí* (viz kapitol 4.1 Webová aplikace), je uložena (*INSERT*) právě nastavená hodnota v poli *color_list* (viz kapitol 4.1 Webová aplikace) do databáze.

```
function setBackground()
    {

        var hodnota = document.getElementById("color_list").value;
        console.log('Vkládám hodnotu: ' + hodnota);
        database.transaction(
                function(transaction){
                    transaction.executeSql(
                        'INSERT INTO background (value) VALUES (?);', [hodnota], null, errorHandler
                    );
                }
            );
```

Obrázek č. 44: Zapsaní zvoleného pozadí do tabulky background

Po provedené změně (*INSERT*), je nutné požadované nastavení uživateli zprostředkovat. Tedy nastavit pozadí na poslední známou hodnotu (*MAX(id)*) z databáze pomocí transakce typu *SELECT*.

```
database.transaction(
    function(transaction){
        transaction.executeSql(
            'SELECT value FROM background WHERE id = (SELECT MAX(id) FROM background);', [],
            function(transaction, result){
                for (var i=0; i < result.rows.length; i++){
                    var row = result.rows.item(i);
                    console.log('Aktuální hodnota pozadí: ' + row.value);
                    document.body.style.background = row.value;
                }
            },
            errorHandler
        );
    }
);
```

Obrázek č. 45: Načtení zvoleného pozadí

Analogicky jako při ukládání konfigurace pozadí, je možné uchovávat data určena pro formulář. Při otevření webové aplikace je provedena kontrola na existenci databázové tabulky (*CREATE TABLE IF NOT EXISTS*). Protože tabulka udržuje informace o zadaných hodnotách do různých formulářových prvků (*jméno*, *telefon*, atd.) je nutné přidat další sloupec (*kod*), pomocí kterého dokáže databáze jednoznačně rozlišit, v jakém sloupci je realizována změna.

```
database.transaction(
    function( transaction ){
        transaction.executeSql(
            "CREATE TABLE IF NOT EXISTS form (" +
              "id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
              "kod TEXT NOT NULL," +
              "value TEXT NOT NULL" +
            ");"
        );

    }
);
```

Obrázek č. 46: Strunktura tabulky form

Při vyplňování formuláře, je uchovávána (*INSERT*) každá provedená změna v kolonkách pro *jméno*, *email*, *telefon* a *zprávu* (struktura formuláře viz kapitola 4.1 Webová aplikace).

To má ovšem za následek vkládání velkého množství informací. Pro příklad, že by byla zadávána hodnota „*abcd*“ je postupně informace uchovávána následovně:

- a
- ab
- abc
- abcd

Při rozsáhlých textech, zadávaných do polí formuláře, není žádoucí držet v databázi všechny takto provedené změna a proto je opět držena právě poslední (*MAX(id)*) vložená hodnota (*INSERT*). Ostatní hodnoty jsou z databáze odmazány (*DELETE*).

```
$('#name, #email, #telephone, #message').on('keyup', function(){
    var kod = ($(this).attr('id'));
    var hodnota = ($(this).val());
    console.log('Vkládám hodnotu: ' + kod + '=' + hodnota);
    database.transaction(
            function(transaction){
                transaction.executeSql(
                    'INSERT INTO form (kod, value) VALUES (?, ?);', [kod, hodnota], null, errorHandler
                );
            }
        );
    database.transaction(
            function(transaction){
                transaction.executeSql(
                    'DELETE FROM form WHERE id NOT IN (SELECT MAX(id) FROM form GROUP BY kod);',
                    null, null, errorHandler
                );
            }
        );
});
```

Obrázek č. 47: Proces vkládání hodnot do tabulky form

Všechna uložená data je nutné mět zprostředkovat uživateli a to kdykoliv, kdy je vyvolán přístup na webovou aplikaci. Tedy například po aktualizaci stránky či jejím opětovném načtení. Data jsou v databázi webového prohlížeče po celou dobu, dokud nedojde k jejich fyzickému smazání.

Je tedy nutné načítat poslední uložená data pro každou položku formuláře. Pomocí sloupce *kod*, kde je držena informace o názvu položky ve formuláři, tedy například jméno, heslo atd., jsou načtena data a jsou pomocí příkazu *getElementById* nastaveny příslušné buňce formuláře. Pokud je tedy ve sloupci *kod* uložena hodnota „*jmeno*“ a jeho *value*, tedy

poslední uložená hodnota, rovna „*abcd*“, je zobrazena uživateli v kolonce „*Jméno:*“ hodnota „*abcd*“. Analogicky jsou nastavena zbývající pole.

```
$('#name, #email, #telephone, #message').each(function(){
    database.transaction(
        function(transaction){
            transaction.executeSql(
                'SELECT id, kod, value FROM form WHERE id IN (SELECT MAX(id) FROM form GROUP BY kod);', [],
                function(transaction, result){
                    for (var i=0; i < result.rows.length; i++){
                        var row = result.rows.item(i);
                        console.log(
                            'Aktuální uložená data formuláře: '  + 'id: ' + row.id + ' ; '
                            + 'kod: ' +  row.kod + ' ; ' + 'hodnota: ' + row.value
                        );
                        if (row.kod == 'name')
                            {
                                document.getElementById('name').value = row.value;
                            }
                        else if (row.kod == 'email')
                            {
                                document.getElementById('email').value = row.value;
                            }
                        else if (row.kod == 'telephone')
                            {
                                document.getElementById('telephone').value = row.value;
                            }
                        else if (row.kod == 'message')
                            {
                                document.getElementById('message').value = row.value;
                            }
                        else
                            {
                                console.log('Neznámá hodnota!!!!');
                            }
                    }
                },
                errorHandler
            );
        }
    );
    <!-- pokud nastane refresh stranky, zobrazim posledni data-->
    console.log('stranka byla aktualizovana a jsou nactena data');
});
```

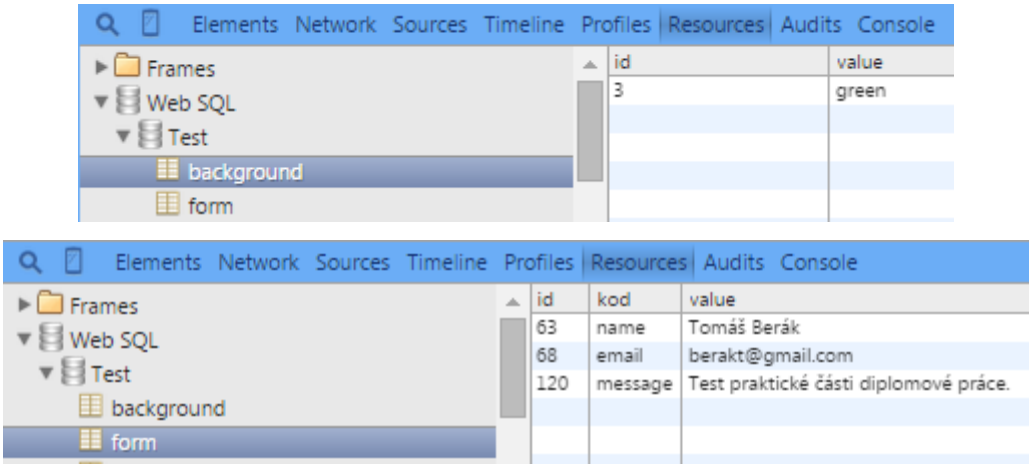
Obrázek č. 48: Načtení hodnot formulářem

Poslední aktivitou, kterou je možné vyvolat, je odeslání formuláře. V takovém případě dojde ke smazání všech hodnot z databáze a dojde k nastavení formuláře a výchozí nastavení. Tedy tak jako by uživatel na stránku přistoupil poprvé.

```
$('form').submit(function(s){
    console.log('potvrzuji a vymazavam databázi');
    <!--vymazu dtb, pokud jsou data odeslana -->

    $('#name, #email, #telephone, #message').each(function(){
        database.transaction(
            function(transaction){
                transaction.executeSql(
                    'DELETE FROM form;', null, null, errorHandler
                );
            }
        );

        $('#myForm')[0].reset();
    });
});
```

Obrázek č. 49: Proces odeslání dat formuláře

Všechny aktuálně uložené hodnoty Web SQL aplikace je možné zjistit pomocí prohlížeče, kde je webová aplikace spuštěna.



Obrázek č. 50: Web SQL

4.4 Offline aplication

Pomocí problematiky local storage (viz kapitola 3.5.4 Local storage) nebo problematiky Web SQL (viz kapitola 3.5.5 Web SQL) je možné pracovat s webovými aplikacemi i v tzv. offline režimu. Tedy ve stavu, kdy není nutně vyžadován přístup k síti internet. To umožňuje další flexibilitu uživateli, jako je například možnost práce v místech, kde není dostupný mobilní internet či jakákoli jiná alternativa sítě.

Offline webové sránky nicméně potřebují prvotní přístup na webovou aplikaci prostřednictvím internetu. Je totiž nutné mít dostupné stránky, obrázky, soubory atp. pro budoucí offline využití. Jedná se tedy o zprostředkování všech potřebných souborů, které se uloží do CACHE (dočasné) paměti zařízení.

Jelikož komunikaci a tak i zprostředkování webové aplikace klientovi řídí webový server, je nutné tuto informaci předem definovat. Soubor CACHE manifestu (*.htaccess*) informuje webový server o skutečnosti, že se mají prostředkovat klientovi potřebná data, tedy uložit do dočasné paměti prohlížeče. To je zajištěno informací v souboru „*text/cache-manifest*“

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

Obrázek č. 51: Obsah souboru .htaccess

V HTML5 dokumentu je nutné zmínit webovému prohlížeči skutečnost, že v případě nedostupnosti webové aplikace má zobrazit CACHE soubory, které načetl při prvotním načtení online. Znamená to tedy, že v HTML5 kódu je nutné atributu *<html>* přidělit element, který odkazuje na soubor s vlastností typu offline application (*manifest=“localstorage.manifest“).*

```
<!DOCTYPE html>
<html manifest="localstorage.manifest">
<head>
```

Obrázek č. 52: Definice .manifest v HTML5

První řádek souboru .manifestu obsahuje vždy „*CACHE MANIFEST*“. Druhou částí souboru je pak informace o tom, jaké soubory se mají načítat při připojení k webové aplikaci. Jde tedy o seznam URL adres (Uniform Resource Locator) konkrétních dat. Je tedy definována stránka s HTML5 kódem webové aplikace (*offlineAplication.html*), dále informace o manifest souboru (*localstorage.manifest*), kaskádových stylech (*style.css*) a JavaScriptu (*jquery.js*). Druhá část souboru obsahuje informaci, jak se má zachovat prohlížeč, který v online režimu zobrazuje obrázek (*online.png*) a není uveden v první části, tedy v definici pro stažení potřených souborů offline aplikace. Klientovi je tímto

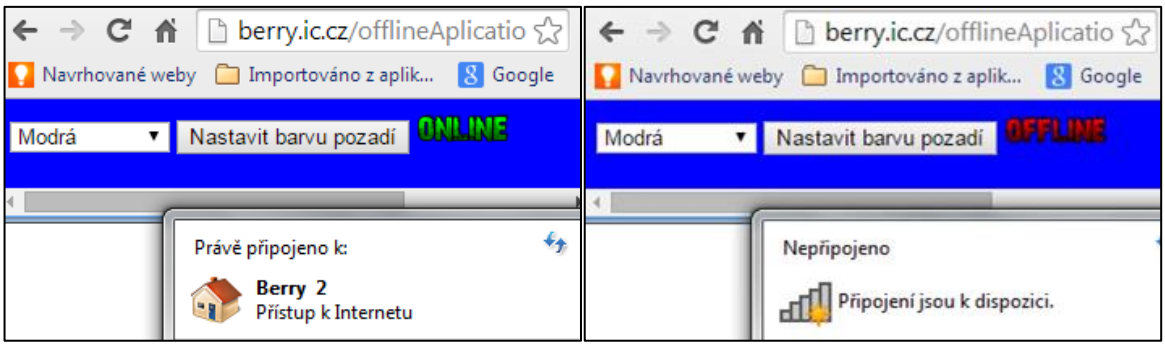
způsobem sdělena informace, že pokud je v režimu offline, má nahradit obrázek s informací o online stavu (*online.png*) obrázkem s informací o offline stavu (*offline.png*).

```
CACHE MANIFEST

CACHE:
offlineAplication.html
localstorage.manifest
../css/style.css
../js/jquery.js

FALLBACK:
http://berry.ic.cz/images/online.png http://berry.ic.cz/images/offline.png
```

Obrázek č. 53: Obsah souboru .manifest

Jakmile je takto připravená offline aplikace, při prvním připojení dojde ke stažení všech souborů uložených v CACHE *manifestu* pomocí události *downloading*. Prohlížeč čeká do té doby, než mu bude serverem oznámena událost *cached*, tedy že jsou všechny soubory uloženy na klientovi. Při dalším online připojení je kontrolováno, zda se vsouboru manifestu neudála nějaká změna. Dále probíhá kontrola, zda jsou soubory uloženy na straně klienta. V případě, že jsou obě kontroly provedeny beze změny, je tato skutečnost oznámena klientovi událostí *noupdate*. Problém ale nastává, když se změna odehraje na úrovni zdrojového kódu souboru. V tomto případě není stažen soubor nový, kde je změna realizována, ale je použit starší, již uložený soubor na straně klienta (cache soubor). Autor v tomto případě nenašel způsob, jak daný problém řešit.

Aplikace má tedy dynamické chování, které je závislé na přístupu k síti internet. Může být tedy prezentována dvěma způsoby, tedy typem online a offline.



Obrázek č. 54: Online a offline aplikace

4.5 Testování ve vybraných prohlížečích

Na základě implementovaných funkcionalit HTML5 je také nutné otestovat internetové prohlížeče. K dipozici je nejen mnoho prohlížečů postavených na odlišném způsobu fungování (viz kapitola 3.4.1 Rozdělení), ale také mnoho jejich verzí. Na základě statistiky nejpoužívanějších prohlížečů (viz kapitola 3.4.1 Rozdělení) bude test realizován pro nejvíce zastoupené prohlížeče. Tedy pro Google Chrome s verzí 38.0, Mozila Firefox verzí 32.0, Internet Explorer ve verzích 8.0 a 11.0. Výsledky budou moci nastat pouze dva, tedy zda je funkcionalita plně funkční (stav *OK*) či funkční není (stav *NOK*).

Testovány budou funkcionality implementované v praktické části této práce:

- CSS3 – Gradient
- CSS3 – Zaoblené rohy
- HTML5 – Local storage
- HTML5 – Web SQL
- HTML5 – Offline Aplication

Tabulka č. 3: Hodnocení testů prohlížečů

	Gradient	**Zaoblené rohy**	**Local Storage**	**Web SQL**	**Offline Aplication**
Google Chrome 38.0	OK	OK	OK	OK	OK
Mozila Firefox 32.0	OK	OK	OK	NOK	OK
Internet Explorer 11.0	OK	OK	OK	NOK	OK
Internet Explorer 8.0	NOK	NOK	OK	NOK	NOK

Pozn.:* OK – funkční, NOK – nefunkční

Dle očekávání si nejlépe vedla nejnovější verze prohlížeče Google Chrome, ve kterém jsou implementovávy všechny testované funkcionality. Důvodem může být silná podpora technologie HTML5 ze strany Google. Naopak nejhůře si vedla starší verze Internet Exploreru ve verzi 8.0 od Microsoftu. Výsledek však není překvapivý, jelikož se jedná o verzi prohlížeče z roku 2008. Aktuální verze prohlížeče je ve verzi 11.0, která si vedla podstatně lépe.

5 Závěr

Cílem práce bylo představení a realizace nových funkcionalit pomocí nových technologií HTML5 a CSS3. Teorická část pojednává o historickém vývoji webových technologií, popisuje současné technologie a zavádí nové pojmy HTML5 a CSS3.

Na teoretickou část navazuje praktická, která vysvětluje pojem HTML5. Dále pak implementuje problematiku uchovávání a zpracování dat na straně klienta a testuje tyto funkčnosti v internetových prohlížečích.

V první části práci byla vytvořena webová aplikace, která obsahovala novinky HTML5 a CSS3. Byl vytvořen uživatelský formulář s možností odeslat autorovi aplikace libovolný text. Byla také implementována možnost výběru uživatelského pozadí. Napsaná webová aplikace byla testována na validitu kódu, tak aby výsledky praktické části byly co nejpřesnější. Stránka byla stanovena jako validní přes webové stránky W3C. Pomocí této šablony byly dále otestovány dvě nové funkcionality CSS3, konkrétně problematika Grandient a zaoblených rohů. Test byl proveden ve čtyřech nejpoužívanějších prohlížečích na světě. Bylo dokázáno, že nejnovější internetové prohlížeče podporují obě tyto funkcionality. Výjimka nastala u starší verze prohlíže Internet Exporeru ve verzi 8.0, kde testované problematiky nejsou podporovány.

Pomocí validní webové aplikace byla ve druhé části práce implementována jedna z funkcionalit HTML5, která zpracovává data na straně klienta, tzv. Local storage. Výslednou aplikací byly otestovány čtyři nejpoužívanější prohlížeče, které se umístily na předních místech statistiky teoretické části. Všechny čtyři testované prohlížeče obstály tímto testem s pozitivním výsledkem. Funkcionalita local storage byla bezproblémově funkční.

V další části práce byla pomocí vytvořené aplikace implementována technologie HTML5 tzv. Web SQL. V tomto případě bylo testováno, jak jsou nejpoužívanější internetové prohlížeče připraveny na tuto funkcionalitu. Výsledek byl opačný než v předchozím případě. Funkcionalita byla shledána jako naprosto funkční pouze u prohlížeče Google Chrome ve verzi 38.

V poslední části byla pomocí webové aplikace implementována problematika Offline application, pomocí které dokáže být webová aplikace funkční i bez přístupu na síť internet. Opět byla problematika testována pomocí čtyř nejpoužívanějších prohlížečů na

světě. Zde byl výsledek pozitivnější než u problematiky Web SQL. Pouze starší verze Internet Exploreru ve verzi 8.0 od Microsoftu byla zhodnocena jako nefunkční.

Na základě provedených zjištění lze konstatovat, že přestože je očekáváno vydání standardizace HTML5 a CSS3 právě v době psaní této práce, nejsou výše uvedené problematiky zcela podporovány ve všech prohlížečích. Jediný prohlížeč, který podporuje všechny tři testované zmíněné funcionality je Google Chrome ve verzi 38.

Závěrem lze tedy říci, že všechny aktuální verze prohlížečů podporují technologii HTML5 pouze částečně. Využití HTML5 a CSS3 by autor doporučil pouze v případě tvorby struktur webu a tvorbě grafiky. Implementace zpracování dat na straně klienta není zatím zcela podporováno a mohlo by tak dojít ke komplikacím při přístupu uživatele k webovým aplikacím.

6 Seznam použitých zdrojů

Bednář, Vojtěch. 2011. *Internetová publicistika.* Praha : GRADA Publishing, a.s., 2011. ISBN: 978-80-247-3452-1.

Burian, Pavel. 2014. *Internet inteligentních aktivit.* 1. Praha : Grada Publishing, a.s, 2014. Sv. I. ISBN 978-80-247-5137-5.

Cyroň, Miroslav. 2005. *CSS - kaskádové styly.* Praha : Grada Publishing a.s., 2005. ISBN: 80-247-1420-5.

Čápka, David. 2013. ITnetwork.cz. [Online] 2013. [Citace: 12. říjen 2014.] http://www.itnetwork.cz/tutorial-uvod-do-asp-dot-net.

čtk a msc. 2013. E15.cz. [Online] 2013. [Citace: 10. říjen 2014.] http://e-svet.e15.cz/internet/ctvrtstoleti-world-wide-web-jeji-tvurce-by-ted-nejradsi-zrusil-dvojite-lomitko-1068421.

De Silva, Malin . 2010. SiliconStation. [Online] 2010. [Citace: 21. listopad 2014.] http://siliconstation.com/new-css3/.

Esposito, Dino. 2004. *XML – efektivní programování pro .NET.* Praha : Grada Publishing a.s., 2004. ISBN: 80-247-0775-6.

Gasston, Peter. 2014. *Book of CSS3.* 2. San Francisco : William Pollock, 2014. ISBN: 978-1-59327-580-8.

Gerchev, Ivaylo. 2014. sitepoint. [Online] 2014. [Citace: 20. listopad 2014.] http://www.sitepoint.com/html5-canvas-tutorial-introduction/.

Grygarek, Petr. 2009. Katedra informatiky FEI VŠB-TUO. [Online] 2009. [Citace: 1. říjen 2014.] http://www.cs.vsb.cz/grygarek/kotasek/htp03.htm.

Harris, Andy. 2011. *HTML5 For Dummies Quick Reference.* Indianapolis : Wiley Publishing, Inc., 2011. ISBN: 978-1-118-01252-9.

—. 2011. *HTML5 Multimedia Develop and Design.* Berkeley : Peachpit Press, 2011. ISBN: 0-321-79393-5.

Hogan, Brian. 2011. *HTML5 and CSS3: Develop with Tomorrow's Standards Today (Pragmatic Programmers).* USA : NC: Pragmatic Bookshelf, 2011. ISBN: 978-1-93435-668-5.

Christensson, Per. 2010. TechTerms.com. [Online] 2010. [Citace: 8. říjen 2014.] http://www.techterms.com/definition/w3c.

Jakel, Milan. 2002. interval.cz. [Online] 2002. [Citace: 10. říjen 2014.] http://interval.cz/clanky/stavove-kody-a-hlaseni-v-odpovedi-protokolu-http/.

Janovský, Dušan. 2014. Jak psát web. [Online] 2014. [Citace: 21. listopad 2014.] http://www.jakpsatweb.cz/css/font-face.html.

Kasner, Zdeněk. 2013. SWMAG.cz. [Online] 2013. [Citace: 18. 10 2014.] http://www.swmag.cz/1161/jadra-prohlizecu-srovnani/.

Kosek, Jiří. 2009. *PHP a XML.* Praha : GRADA Publishing, a.s., 2009. ISBN 978-80-247-1116-4.

Macich, Jiří. 2013. ROOT.cz. [Online] 2013. [Citace: 9. říjen 2014.] http://www.root.cz/clanky/moderni-prohlizece-a-webove-technologie-blizke-budoucnosti/.

McFarland, David Sawyer. 2013. *CSS3: The Missing Manual.* Sebastopol : O'Reilly Media, 2013. ISBN: 978-1449325947.

Meiert, Jens. 2006. CSS 3: Texteffekte. [Online] 24. březen 2006. [Citace: 21. listopad 2014.] http://meiert.com/de/publications/articles/20060324/. ISSN: 1614-3124.

nhinkle. 2011. superuser.com. [Online] 2011. [Citace: 17. listopad 2014.] http://blog.superuser.com/2011/04/12/spring-2011-browser-roundup/.

Patel, Bina. 2013. eofax. [Online] 2013. [Citace: 14. listopad 2014.] http://www.graduateprojects.zofaxwebsolutions.com/ZofaxWebSolutions-Graduate-Software-Computer-MCA-BE-BTECH-Techology-You-Learn.aspx.

Písek, Slavoj. 2010. *HTML, začínáme programovat.* 3. Praha : GRADA Publishing, a.s., 2010. ISBN 978-80-247-3117-9.

Ponkrác, Miroslav. 2007. *PHP a MySQL bez předchozích znalostí.* Brno : Computer Press, 2007. ISBN: 978-80-251-1758-3.

Procházka, David. 2010. *První kroky s internetem.* Praha : GRADA Publishing, a.s., 2010. ISBN: 978-80-247-3255-8.

Rouse, Margaret. 2014. TechTarget. [Online] 2014. [Citace: 13. listopad 2014.] http://searchsoa.techtarget.com/definition/HTML5.

Sklenák, Vilém. 2001. *Data, informace, znalosti a Internet.* 1. Praha : C. H Beck, 2001. ISBN 80-7179-409-0.

Skonnard, Aaron a Gudgin, Martin. 2006. *Essential XML Quick Reference.* Praha : Grada Publishing a.s., 2006. ISBN 80-247-0972-4.

StatCounter. 2014. StatCounter. [Online] 2014. [Citace: 27. říjen 2014.] http://gs.statcounter.com/#desktop+console-browser_version-ww-daily-20141027-20141027-bar.

Storey, David. 2012. CSS3.info. [Online] 2012. [Citace: 22. listopad 2014.] http://www.css3.info/preview/rounded-border/.

Štědroň, Bohumír. 2009. *Open Source software - ve veřejné správě a soukromém sektoru.* Praha : GRADA Publishing, a.s, 2009. ISBN: 978-80-247-3047-9.

T.N.Sharma, Bhardwaj, Priyanka a Bhardwaj, Manish. 2012. Differences between HTML and HTML 5. *International Scholarly Open Access Research.* [Online] 1. září 2012. [Citace: 17. listopad 2014.] http://www.ijceronline.com/papers/Vol2_issue5/AR02514301437.pdf. ISSN 2250-3005.

Trivedi, Priya a Harneja, Sanya. 2014. HTML5 – the new standard for Interactive Web. *International Scholarly Open Access Research.* [Online] 10. listopad 2014. [Citace: 17. listopad 2014.] http://internationaljournalofresearch.org/index.php/ijr/article/download/810/764. ISSN 2348-6848.

Vávrů, Jiří a Ujbányai, Miroslav. 2013. *Programujeme pro Android.* Praha : GRADA Publishing, a.s., 2013. ISBN: 978-80-247-4863-4.

W3C. 2014. W3C validator. [Online] 2014. [Citace: 23. listopad 2014.] http://validator.w3.org/check.

—. 2014. W3Schools. [Online] 2014. [Citace: 18. listopad 2014.] http://www.w3schools.com/cssref/css3_pr_text-shadow.asp.

Wald, Mike, a další. 2013. International Journal of Information Technology & Computer Science. [Online] 1. červen 2013. [Citace: 22. listopad 2014.] http://ijitcs.com/volume%209_No_3/mike.pdf. ISSN: 2091-1610.

Zeldman, Jeffrey . 2011. zeldman.com. [Online] 28. leden 2011. [Citace: 21. listopad 2014.] http://fireballed.org/linked/2011/01/30/zeldman/. ISSN: 1534-0309.

7 Seznam obrázků

8 Seznam tabulek