Soukromá vyšší odborná škola a Obchodní akademie s.r.o.
České Budějovice **Pražská 3.**

Absolventská práce

2003 **Petra Pavlyková**

Prohlašuji že jsem absolventskou práci na téma Tvorba internetových aplikací kaskádovými styly CSS verze 3, vypracovala samostatně a použila jen pramenů, uváděných v bibliografii.

V Českých Budějovicích dne 17. května 2003

Soukromá vyšší odborná škola a Obchodní akademie s.r.o.
České Budějovice Pražská 3.

Studijní obor : Výpočetní technika a programování

Tvorba internetových aplikací kaskádovými styly

CSS verze 3

Absolventská práce

Autor : Petra Pavlyková

Vedoucí absloventské práce : PaedDr. Petr Pexa

Tímto bych chtěla poděkovat PaedDr. Petru Pexovi za odborné vedení, připomínky a cenné rady při vypracovávání absolventské práce.

Obsah

1 ÚVOD

Název internet je snad každému znám, alespoň jako pojem, taktéž i www. Méně lidí už ví, že www, je zkratka tří slov : World Wide Web, ale tato neznalost nikomu nebrání v prohlížení (též surfování) po mnoha stránkách, a to nejen při hledání informací, shánění práce, ale i seznamování, posílání zpráv, nebo krátkých textových zpráv (známých jako sms), či prohlížení obrázků. Nejen pro tyto účely je dnes internet mnoha lidmi využíván každý den. A právě dostupnost a přiblížení široké veřejnosti vedlo tvůrce ke zlepšení a grafickému zpříjemnění vzhledu stránek, na rozdíl od prvopočátků, kdy byly stránky, ale i prohlížeče konstruovány na zobrazování pouhého textu.

Nastal tak vývoj, kdy byly do jazyka HTML mimo jiné přidávány možnosti ovlivňovat grafický vzhled dokumentu. Dnes již díky HTML můžeme vytvořit vzhledově pěknou stránku. Používání HTML má však mnoho nevýhod, mezi největší patří pracnost. Stále opakované nastavování různých vlastností jak písma, tak rámečků, tak vzhledu jednotlivých oken. Kopírováním se jistě tvůrci webu usnadní život, ale musí mít v hlavě již konečnou představu a formu té dané www stránky. To umí asi málokdo. Většina z nás totiž stále zkouší různé kombinace barev, obrázků, pozadí i fontů a tak stále kopírujeme a přepisujeme na mnoha stránkách dokola. Dalším nešvarem je nepřehlednost, orientovat se v textu a tazích, vědět co ke kterému patří, vyžaduje výbornou orientaci v kódu. Ke grafickým úpravám stránky nám výborně poslouží právě kaskádové styly. Pro kaskádové styly je nejčastějším označením zkratka CSS což v originále znamená Cascading Style Sheets (dále jen CSS).

Kaskádové styly, umožňují právě jednoduše a přehledně formátovat písmo, upravovat barvu, pozadí i samotné obrázky. V této publikaci se budu věnovat právě popisu vlastností CSS. A to jejich nejnovější verzí, CSS level 3. Bude zde uvedena i syntaxe, obrázky, příklady. Jestliže si však budete uvedené hned chtít zkusit, musím vás upozornit, že půjdou jen některé základní prvky z levelů 1 a 2, protože většina funkcí CSS je pouze návrhem organizace W3C, která všechny tyto návrhy musí

schválit a zpracovat. Proto vám vlastnosti a funkce CSS3, fungovat v prohlížečích nebudou. Ale jestliže si to chcete vyzkoušet, můžete to zkusit přes emulátor, který je možné si stáhnout na domovské stránce konsorcia w3c, jehož internetová adresa je www.w3.org, kde nalezneme veškeré informace pro tvorbu webu. A najdeme zde i nejnovější příspěvky k CSS3, které v době kdy jsem psala tuto práci ještě nebyly uvedeny.

2 ÚVOD DO CSS

Odkud vlastně kaskádní styly pocházejí? O deset let dříve, než vznikl world wide web, průkopníci v oblasti elektronické dokumentace zjistili významný rozdíl mezi tím, jak dokument vypadá (často mluvíme o vzhledu) a podkladem struktury dokumentu. A tak komplexní elektronický nakladatelský systém má od té doby být realizován do jisté míry tak, aby odděloval informace o tom, jak by ten, který dokument měl vypadat, od dokumentu samého. Ten kdo první předvedl nápad, jak toto realizovat, byl Tim Berners-Lee. Ale v tom horečném spěchu webovské rané exploze, ostatní do jisté míry snadnější, ale také problematičtější způsoby, jak formátovat stránku, tento objev odsunuly do pozadí. Mezi tyto nové způsoby patřily tagy jako `<font>` nebo `<b>` a další elementy HTML. Do dnešních dnů, většina programátorů webovských stránek formátuje vzhled stránek právě tímto způsobem. A většina nástrojů pro web, speciálně WYSIWYG podporuje používání této metody. Ve zkratce, CSS poskytuje prostředky pro autory www stránek, které oddělují vzhled stránek od jejich obsahu. Kaskádní styly jsou vlastně doporučením, jak mají stránky vypadat. Toto doporučení vydává World Wide Web Consortium (dále jen W3C). W3C je asociace investorů webu, univerzit, společností jako je Netscape Communications a Microsoft a expertů z mnoha souvisejících oborů. Byla založena Tim Berners-Leeem (kterého můžeme považovat za zakladatele webu) a existuje proto, aby propagovala a podporovala www. Jedna z jejich rolí je, publikovat doporučení. Tato doporučení můžeme považovat za jakýsi standard. Doporučení obsahují různé aspekty www, od HTML tak i XML apod.

W3C vydalo již 2 hlavní doporuční týkajících se CSS a to: CSS1 a CSS2. V nynější době pracuje i na třetím doporučení a to CSS3.

CSS má několik levelů rostoucí komplikovanost. Úvodní CSS1 se zejména zabýval relativně jednoduchými formátovacími funkcemi, jako color, font, background a podobně. Následovalo CSS 2 (1998), který se zabýval dalšími možnostmi důmyslných rysů, jako například *page-based layout*, podporoval schopné provedení

fontů a dávající volnost tvůrcům, k rozvržení dokumentu bez použití HTML tabulek jako prostředků k umístění prvků na obrazovku.

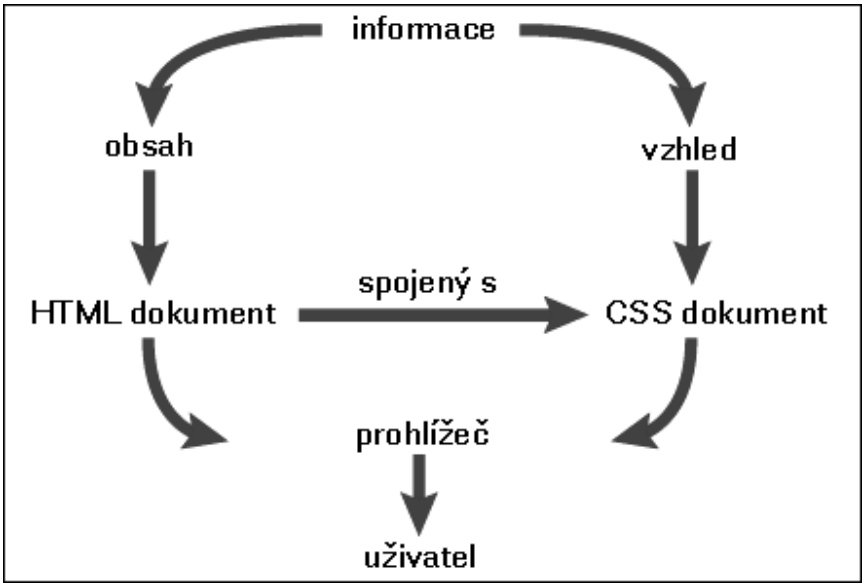
Práce na 3. levelu je v plném proudu. Některé části byli již dokončeny v roce 2001 a 2002 většina zbývajících vlastností bude dokončena během roku 2003. Tento nový level přinese spoustu nových rysů, bude podporovat více typografických zvyklostí, více interaktivity a více prostředků a nápadů.

CSS3 se bude skládat z několika oddělených specifikací, které se nazývají moduly. Rozdělení CSS do modulů umožňuje nástrojům pracovat se všemi moduly najednou. To nám ulehčí realizaci nápadů pro vlastní profily. Moduly nám pomohou výkonněji zlepšovat a upravovat námi vytvořený dokument.

3 JAK VLASTNĚ CSS PRACUJÍ

Již víme, že kaskádní styly jsou jako textové soubory, nebo texty vložené do hlavičky HTML dokumentu, které jakýmsi způsobem pomáhají oddělit text od obsahu. Obsah stránky se dostane do souboru s HTML. A vzhled stránky do listu stylů. Ale jak to, že výsledkem tohoto je webovská stránka v prohlížeči u uživatele?

Myšlenka kaskádních stylů je taková, že je to určitá sada instrukcí a návrhů pro prohlížeč, jak vykreslit stránku. Ale pozor, je to jenom návrh, protože CSS nenutí prohlížeč, aby ukazoval stránku tou konkrétní cestou, on jenom navrhuje prohlížeči, jak by ta stránka mohla vypadat. Toto je důležité rozlišovat.



Obrázek 1. Jak pracují Kaskádní styly

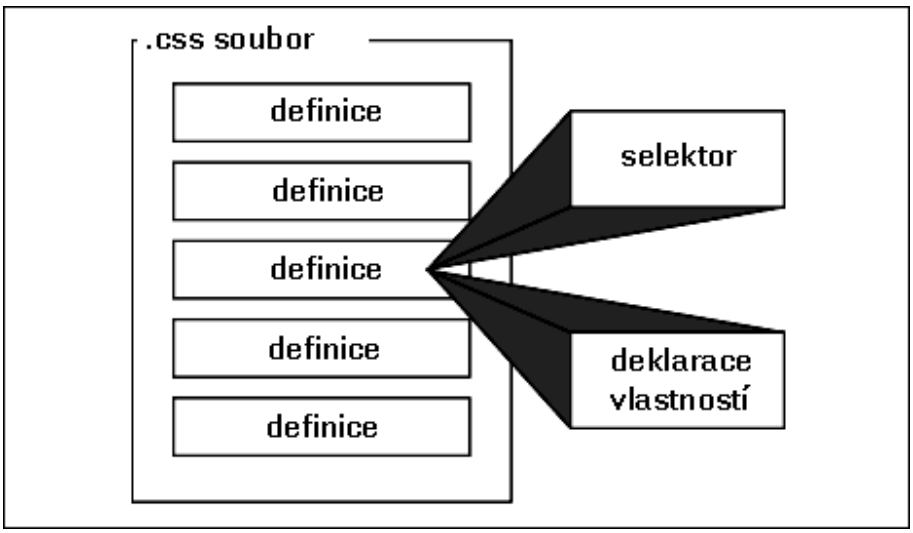
3.1 ELEMENTY KASKÁDNÍCH STYLŮ

Všechny kaskádní styly (je jedno jestli jsou obsaženy v samotném souboru .css, nebo jsou vloženy přímo v hlavičce HTML dokumentu.) jsou sérií instrukcí, které jsou nazývány definicemi. Definice mají za úkol dělat 2 věci (viz obrázek č.2):

1. identifikovat prvky HTML dokumentu, které ovlivňují
2. říkají prohlížeči, jak má tyto prvky vykreslit

podle prvků, například paragrafů, řádek, prvků seznamu apod. V technické terminologie, je jakýkoliv prvek, který je nějak zvýrazněný v HTML tagu.

Část definice která identifikuje prvky stránky, se nazývá selektor. Selektory rozlišují prvky stránky. A část definice, která říká prohlížeči, jak by mohl být vybraný prvek vykreslen, se nazývá deklarace. Deklarace může obsahovat různé množství vlastností, jednotlivé části stylu, které budou aplikovány na vybraný prvek.



Obrázek 2. – Struktura práce stylů

Příklad k vysvětlení výše uvedeného:

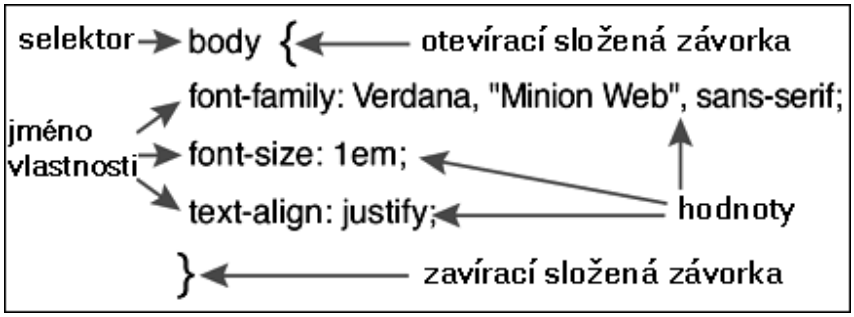
```
body {
font-family: Verdana, Helvetica, sans-serif;
font-size: 1em;
text-align:justify; }
```

Na této jednoduché definici se dá velmi snadno vysvětlit to, co jsem již předeslala (viz obrázek č.3). Selektorem je v tomto případě `<body>`. To znamená, že tato definice bude ovlivňovat tag `<body>`, na každé stránce spojené s touto stránkou stylů.

Tato definice obsahuje deklaraci se třemi vlastnostmi. To znamená, že každý vybraný prvek v ní bude mít ovlivněné tři své vlastnosti.. Každá vlastnost má své jméno, např. *text-align* a svou hodnotu, například *justify*. V tomto případě, font textu v celém těle (body) bude Verdana. Jestliže uživatel nemá na svém počítači font Verdana, prohlížeč použije font Helvetica a jestliže není ani tento font tak se použije defaultní font – sans-serif.

Navíc bude mít písmo v *body* velikost 1em. Velikost 1em si vysvětlíme později. Ten, kdo zná HTML, ví, že není možné nastavit velikost nadpisů v bodech, jedině si vybrat jednu ze 6 poměrných velikostí písma. Toto je další výrazná výhoda kaskádních stylů: mnohem důmyslnější rozvržení stránky a typografická kontrola.

Nakonec je prohlížeč instruován, o tom, že má text zobrazit jako plně vyrovnaný do bloku.



Obrázek č.3 – stavba definice

3.2 SYNTAXE KASKÁDNÍCH STYLŮ

Jak už bylo uvedeno dříve, dlouho neexistovalo nic jako standard www. Výsledkem toho byl vznik velmi shovívavého prohlížeče k jazyku HTML, který není

psán žádnou gramatickou syntaxí. Naproti tomu CSS byli vždy velmi specificky standardizovány. To znamená, že jestliže nedodržíte přesně syntaxi, tak prohlížeč k vám nebude vůbec shovívavý. Při nejmenším vám kaskádní styly nebudou pracovat přesně tak jak byste očekávali. Výše jste mohli vidět, jaké jsou části definic kaskádních stylů a i příklad, jak se dávají jednotlivé deklarace dohromady. S odkazem na výše uvedený příklad budou následovat nějaká základní pravidla, která byste měli dodržovat, jestliže chcete, aby vám vaše kaskádní styly pracovaly, tak jak chcete vy.

1. Každá definice musí obsahovat selektor a deklaraci. Deklarace následuje bezprostředně za selektorem a je obsažena ve složených závorkách.
2. Deklarace je jedna či více vlastností oddělených středníkem-
3. Každá vlastnost má své jméno, za kterým následuje dvojtečka a za ní hodnota té které vlastnosti. Mohou to být rozdílné typy hodnot, ale každá daná vlastnost může nabývat jen určitých hodnot, které jsou nastaveny ve specifikaci. (tato specifikace bude následovat níže)
4. Někdy může vlastnost nabývat číselných hodnot, jako například vlastnost *font-family*. Hodnoty v seznamu mohou být odděleny čárkou a mezerou.
5. Jindy mohou hodnoty mít jednotky, takové jako vlastnost *font-size*, v uvedeném příkladě. V takovém případě nesmí mezera oddělovat hodnotu a jednotku.
6. Tak jako v HTML, volné řádky mohou být použity k lepší čitelnost a přehlednosti vašeho kódu.

4 ZAČLEŇOVÁNÍ STYLŮ

Řekli jsme si, co to kaskádní styly jsou, jak pracují a proč se používají. Ale ještě jsme si podrobně nepopsali, jak se prohlížeč o nich dozví. Jestli kaskádní styly obsahují informace, které řeknou prohlížeči, jak zobrazit jednotlivé stránky, zda se mají kaskádní styly použít, či nikoliv a kde je může najít.

Existují 2 způsoby jak mohou být kaskádní styly propojené s HTML dokumentem. První je ten, že kaskádní styly jsou přímo vložené do hlavičky HTML dokumentu:

```
<style type=”text/css”> </style>
```

Je to jednoduché a prosté, ale ne příliš profesionální. Víme, jak jednoduše, rychle a efektivně lze využít kaskádní styly ke změně vzhledu stránky spojené právě s definicí kaskádních stylů. Ale když použijeme vložené styly, ztrácíme veškeré tyto výhody a v podstatě nemáme oddělen vzhled od obsahu.

Druhým, daleko lepším a efektivnějším způsobem jsou takzvané propojené styly. Stačí propojit v hlavičce HTML dokumentu samotný dokument s definicí kaskádních stylů. Jakmile prohlížeč začne zkoumat stránku, uvidí kaskádní styly a stáhne si stránku s jejich definicí. Poté ji použije ke správnému zobrazení stránky.

```
<head>
<link rel=”stylesheet” href=”moje_styly.css” type=”text/css”>
</head>
```

rel = znamená, že to je odkaz a informuje prohlížeč, o tom, co může očekávat na druhém konci, který se jmenuje style sheet.
type = styly stránky mohou být psány několika různými jazyky. Styly, o kterých mi tady hovoříme jsou CSS. A proto tento atribut říká prohlížeči, o jaký formát stylů se jedná. Typ obsahu je nezbytný.

href = informuje prohlížeč o lokalitě, kde má uvedené styly hledat. Může obsahovat jak absolutní tak relativní cestu.

Samotné kaskádové styly používají příkaz @import, který dělá v podstatě totéž, tedy natahuje styly ze souboru. Uvádím příklad stránky, která natáhne styly ze souboru moje_styly.css a ještě si k tomu navíc přebarví všechny texty uvnitř značky b namodro:

```
<html>
<head>
<title<titulek</title>
<style type="text/css"><!--
  @import url("moje_styly.css");
  b {color:blue;}
--></style>
</head>
<body>
 ...
</body>
```

Pro snazší používání stylů byla do jazyka HTML ve verzi 4.0 přidána úprava, která říká, že většina značek v jazyce HTML může mít přidán svůj individuální styl, a to jako atribut s názvem style. Jsou to tzv. in-line styly:

```
<p style="font-style:italic;"> </p>
```

Tento odstavec bude psán kurzívou

Toto způsobí, že právě jeden odstavec bude právě takovýto odlišný od jiných, v celém dokumentu. Tento způsob ale také nedodržuje základní teorii, toho, proč byly kaskádní styly vymyšleny a to, oddělit vzhled od textu. Ale obrovský význam má používání atributu style v dynamickém HTML. Je totiž možné měnit styly (třeba pomocí JavaScriptu) v závislosti na tom, co čtenář stránky dělá. Ale to už je úplně něco jiného.

5 ELEMENTY V CSS A JEJICH VLASTNOSTI

Elementy jsou v podstatě jednotlivé tagy kódu HTML. Ale kaskádní styly nejsou závislé pouze na tagy HTML, to by zase nebyla zřejmá jejich výhoda a byly by značně omezeny. V CSS jsou vytvořeny nové elementy, které rozšiřují schopnosti předdefinování vlastních prvků kódu HTML. Mezi tyto elementy patří třídy (CLASS), identifikátory (ID) a pseudo-elementy.

5.1 ZÁKLADNÍ ELEMENTY HTML KÓDU

Nejjednodušší je použít CSS v předdefinování vlastností pro již existující HTML tagy.

```
<style>
<!—
body {color : #990000; font-size: 12pt;}
p{font-family:helvetica; text-align:justify}
-->
</style>
```

Tyto tagy se vyznačují vzájemnou dědičností. To znamená, že například tag `<BODY>` je nadřazený všem tagům používaným v jeho těle a tedy všechny tyto tagy dědí nepředefinované vlastnosti tohoto tagu. V našem příkladě to znamená, že tag `<P>` dědí po tagu `<BODY>` barvu a velikost písma. Ale i ostatní tagy, které budou uvedeny dále v HTML dokumentu budou dědit.

Tagy lze seskupovat, tzn. že několik tagů využívá stejná pravidla. Můžeme tagy dát za sebou a oddělit čárkou a předdefinovat jim stejné vlastnosti. Lze ale také nastavovat vlastnosti dle kontextu, toho kterého elementu. Chceme-li na příklad nastavit odkazy v tabulce silně a třeba modře, napíšeme tyto tagy za sebou a oddělíme je mezerou.

```
td a {color : #0099FF; font-weight : bold;}
```

Tento styl zápisu se nazývá kontejnerový. Dá se použít u jakýchkoliv typů tagů a při jejich kombinaci. Jsou však většinou velmi nepřehledné a v případě modifikace nám práci ztěžují a tak se tento zápis moc nedoporučuje.

Základní elementy se dělí do několika skupin. Tyto skupiny určují pro jaké elementy je možné použít ty které vlastnosti CSS. Skupiny máme tyto:

Blokové elementy – jsou takové, za kterými se zalamuje řádka, nebo takové, které vztvářejí urřité bloky. např. `<P>, <H>, <TD>`, apod.

Inline elementy - to jsou takové, které jsou běžnou součástí textu na řádce a není před ani za nimi žádné zalomení řádky. např. `<SPAN>, <I>, <STRONG>`, apod.

Nahrazované elementy – jsou takové, které jsou nahrazeny obsahem a pro jejichž zařazení do okolního textu stránky jsou důležité pouze jejich vlastní rozměry. např. `<IMG>, <OBJECT>`, apod.

5.2 TŘÍDY (CLASS)

Třídy slouží k dodefinování nebo přímo pro předefinování vlastností HTML tagů, ve kterých jsou použity. Třídu definujeme obdobně jako tag, až na to, že před její název předřadíme tečku. Její volání v HTML kódu je prováděno pomocí parametru *CLASS*.

Zde je pro větší názornost uveden příklad, řekněme, že budeme chtít mít stránku, kde písmo v některých zvolených odstavcích bude psáno kurzívou, zatímco v ostatních ho nechceme ovlivňovat.

Nastavit odstavce na kurzívu je velice jednoduché, můžeme to udělat třeba takto:

```
p { font-style: italic; }
```

Problém ale je, že nechceme nastavovat všechny odstavce na kurzívu, ale jenom některé. A právě k tomuto slouží třídy. Musíme tedy náš styl přepsat třeba takto:

```
p.kurz { font-style: italic; }
```

Zápis *p.kurz* znamená, že všechny odstavce, které budou mít třídu *kurz*, budou psány kurzívou. Už nám tedy zbývá jenom ukázat, jak ji budeme v HTML volat:

```
<p class="kurz">Cokoliv kurzívou.</p>
<p>Již normálně.</p>
```

Většina značek v HTML má možnost přidání atributu class, která určuje třídu pro kaskádové styly CSS. Můžeme atribut class přidat k jakékoli značce, u které je možné ovlivnit vzhled. Což jsou prakticky všechny značky s výjimkou několika málo značek: `<html>`, `<head>`, `<meta>`, `<style>`, `<basefont>`, `<title>`, `<base>` a `<param>`. Jinak se dá použít u všech ostatních, to znamená v oblasti mezi značkami `<body>` a `</body>`, a dokonce i u značky `<body>` samotné. Pomocí stylů pak můžeme přesněji cílit naše úpravy vzhledu dokumentu.

Názvy tříd je dobré volit pečlivě tak, aby obsahovaly pokud možno pouze písmena anglické abecedy, číslice a případně pomlčku. Velice to zjednoduší případnou práci s třídami. Prohlížeče v názvech tříd nerozlišují malá a velká písmena, takže napíšeme-li jednou název třídy jako *tucne* a podruhé ji použijeme pod názvem *TuCNe*, mělo by se vše zobrazit bez problémů. Na druhé straně takovéto střídání velkých a malých písmen není dobré a svědčí o neprofesionalitě autora stránky.

Novější norma CSS2 má práci s třídami velmi rozšířenou, a kromě toho umožňuje i další triky. Tato norma dovoluje použít u jedné značky více tříd, které jsou oddělené jednou mezerou:

```
<html>
<head>
<title>titulek</title>
<style type="text/css"><!--
  .modre { color: blue; }
```

```
  .tucne {font-weight: bold;  }
--></style>
<p class="tucne modre">Tento odstavec bude napsán tučně a modrou
barvou
</p>
</body>
</html>
```

V tomto případě má odstavec, který v HTML představuje značka `<p>` dvě třídy. A odstavec bude zobrazen podle obou stylů, pokud si vzájemně nenařizují něco protichůdného.

5.3 IDENTIFIKÁTORY

Identifikátory mají podobné vlastnosti jako třídy. Doplňují nebo mění vlastnosti definovaných tagů. Jejich hlavní využití je ale v jejich spojení s dynamickými prvky HTML. Jeho podstatou je jednoznačně definovat element (identifikovat). Proto se identifikátory používají obvykle pouze pro jeden určitý element v HTML dokumentu, což ale nemusí být podmínka. Syntaxe identifikátorů je taktéž obdobná jako u tříd, avšak na místo tečky před jménem je použita zahrádka.

```
#identifikátor {specifikace}
```

Volání v HTML dokumentu probíhá pomocí atributu ID:

```
<tag id="identifikátor">
```

```
#special {font-color:#FF0000; font-family: helvetica;}
<td id="special">Buňka s identifikátorem special</td>
```

Konkrétní užití identifikátorů je např. při dynamickém vypínání a zapínání viditelnosti některých částí dokumentů, např. textů. Názvy identifikátorů se mohou

shodovat s názvy tříd, což se ale samozřejmě nedoporučuje z důvodů přehlednosti. Při případu, že je použita třída a identifikátor v jednom tagu, tak se to naopak doporučuje. Zde vede shodné pojmenování k větší přehlednosti.

6 SELEKTORY

Selektory reprezentují strukturu. Tato struktura je použita jako podmínka určující které elementy odpovídají selektoru ve stromovém dokumentu nebo jako popis HTML, či XML části, odpovídající struktuře. Mohou být rozsáhlé, od jednoduchých jmen prvků k související reprezentaci. Selektory jsou nyní modulem CSS3 a jsou nezávislou specifikací. Jiné specifikace se mohou nyní odkazovat na tento dokument nezávisle na CSS

6.1 ROZDÍLY OD CSS2

Hlavní rozdíly mezi selektory v CSS2 a těmito Selektory jsou:

- Seznam základních definic (selektor, skupiny selektorů, jednoduché selektory, apod.) je jasnější
- univerzální selektory a vlastnosti selektorů
- nové kombinace
- nové jednoduché selektory obsahující podřetězec odpovídající vlastnostem selektorů a nové pseudotřídy
- nové pseudo-elementy a všechny nyní začínají s „::“, což je nová konvence pseudo-elementů
- přepsání gramatiky selektorů
- profily přidané normy začleněných Selektorů a definování kolekce selektorů, které jsou aktuálně podporovány každou specifikací

6.2 SKUPINY SELEKTORŮ

Když různé selektory sdílí ty samé deklarace mohou být seskupeny do jednoho seznamu a jsou odděleny čárkou.

```
h1 {font-family: sans-serif}
h2 {font-family: sans-serif}
```

```
h3 {font-family: sans-serif}
```

Výše napsaná definice je naprosto ekvivalentní k:

```
h1, h2, h3 {font-family: sans-serif}
```

6.3 JEDNODUCHÉ SELEKTORY

Nejjednoduššími selektory jsou typové selektory. K těmto patří např. `<H1>`. Je dobré si toto uvědomit pro následující kapitoly.

6.3.1 TYPOVÉ SELEKTORY A JMENNÝ PROSTOR (NAMESPACES)

Typové selektory umožňují nepovinné komponenty ve jmenném prostoru. Na začátek bych chtěla ozřejmit spojení jmenový prostor. Můžeme si to představit jako takový externí soubor, který potírá bariéry, v našem případě, kaskádových stylů. A co je zde definováno, platí pro všechny soubory, či stylové předpisy.

Předpony deklarované ve jmenném prostoru, mohou být přiřazeny ke jménu prvk, který je oddělený jmenným prostorem, oddělovačem (|). Jmenný prostor může být nechaný prázdný, znamená to, že selektor znázorňuje elementy se žádným deklarovaným jmenným prostorem.

ns|E elementy se jménem E ve jmenném prostoru ns

***|E** elementy se jménem E v jakémkoliv jmenném prostoru, obsahující ty které jsou mimo jakýkoliv deklarovaný jmenný prostor

|E elementy se jménem E bez jakéhokoliv deklarovaného jmenného prostoru

E jestliže není specifikován žádný jmenný prostor,je toto ekvivalentní s ***|E**. Jinak je ekvivalentní s **ns|E** kde ns je defaultní jmenný prostor.

```
@namespace foo url(http://www.example.com);
      foo|h1 { color: blue }
      foo|* { color: yellow }
      |h1 { color: red }
      *|h1 { color: green }
      h1 { color: green }
```

První pravidlo nastaví jen elementy `h1` v "http://www.example.com" jmenném prostoru.

Druhé pravidlo nastaví všechny elementy v "http://www.example.com" jmenném prostoru.

Třetí pravidlo nastaví jen element `h1` ,mimo jakýkoliv deklarovaný jmenný prostor.

Čtvrté nastaví element `h1` v kterémkoliv jmenném prostoru (obsahující tyto mimo jakýkoliv deklarovaný jmenný prostor).

Poslední je ekvivalentní se čtvrtým pravidlem protože zde není deklarován žádný defaultní jmenný prostor.

6.4 ATRIBUTY SELEKTORŮ

Tyto selektory dovolují reprezentaci vlastností prvkům.

6.4.1 VÝSKYT VLASTNOSTÍ A HODNOTY SELEKTORŮ

CSS2 uvedlo čtyři atributy selektorů. A těmi jsou:

[att] Reprezentující atribut **att**, jakákoliv hodnota atributů.

[att=val] Reprezentující atribut **att** s přesnou hodnotou "val".

[att~=val] Představující atribut **att** jehož hodnota je mezerou oddělený seznam slov, jedno které je přesně "val". Když je tento selektor použit, slova v této hodntě nesmí obsahovat mezeru (od místa, kde jsou odděleny právě mezerou).

[att|=val] hodnota tohoto selektoru může být buď "val" nebo může začínat "val" bezprostředně následovaný mezerou(-). Toto zejména odpovídá subkódu.Jazyk subkódu je nastíněn v pseudotřídě :lang.

Hodnoty atributů musí být identifikátory nebo řetězce. Case-sensitivity jmen atributů a hodnot v selektorech závisí na jazyku dokumentu, ve kterém jsou psány.

Například následující selektor atributů představuje element h1 který má titulek atributu, kterékoliv jeho hodnoty:

```
h1[title]
```

V následujícím případě, představuje selektor element span, jehož atribut třídy má hodnotu přesně "example":

```
span[class=example]
```

Mnohonásobný atribut selektorů může být použit k reprezentaci různých atributů elementů, nebo různé podmínky stejných atributů. V tomto případě selektor je aplikován na element span jehož atribut hello nabývá přesně hodnoty "Cleveland" a jehož atribut goodbye má přesně hodnotu "Columbus":

```
span[hello="Cleveland"][goodbye="Columbus"]
```

Následující selektory ilustrují rozdíly mezi "=" a "~=". První selektor bude například, hodnoty "copyright copyleft copyeditor" atributu rel. Druhý selektor bude reprezentován jen na element s atributem href mající přesnou hodnotu "http://www.w3.org/".

```
a[rel~="copyright"]
a[href="http://www.w3.org/"]
```

Další selektor představuje link jehož atribute hreflang je "fr":

```
link[hreflang=fr]
```

Další link je aplikována na ty, jejichž hodnota atributu hreflang začíná na "en", obsahuje "en", "en-US", a "en-cockney":

```
link[hreflang|="en"]
```

Stejně tak nadcházející selektor zobrazuje element DIALOGUE kdykoliv má jednu ze dvou rozdílných hodnot pro atribut character:

```
DIALOGUE[character=romeo]
```

```
DIALOGUE[character=juliet]
```

6.4.2 PODŘETĚZCE ODPOVÍDAJÍCÍ ATRIBUTŮM SELEKTORŮ

V CSS3 byly přidány tři dodatečné atributy selektorů, které jsou poskytovány pro odpovídající podřetězce v hodnotách atributů:

[att^=val] - atribut att jehož hodnota začíná předponou "val"
[att$=val] - atribut att jehož hodnota končí příponou "val"
[att*=val] - atribut att jehož hodnota obsahuje při nejmenším jednu instanci podřetězce "val"

Hodnoty atributů musí být také identifikátory nebo řetězce Také case-sensitivity názvů atributů v selektorech závisí na jazyku dokumentu, ve kterém stylové předpisy tvoříme.

HTML `object`,odkazující na tag `image`:

```
object[type^="image/"]
```

HTML anchor s attributem `href` jehož hodnota končí ".html".

```
a[href$=".html"]
```

HTML odstavec s atributem `title` jehož hodnota obsahuje podřetězec"hello"

```
p[title*="hello"]
```

6.5 PSEUDOTŘÍDY

Pseudotřídy (nebo pseudoprvky) obsahují prvky jednoho typu, které odpovídají určitým kontextuálním kritériím. Pseudotřída vždy obsahuje dvojtečku (:), která je následována jménem a nepovinnou hodnotou mezi závorkami. Pseudotřídy jsou dovoleny používat u všech jednoduchých selektorů. Některé z nich se vzájemně vylučuj, zatímco jiné mohou být aplikovány současně na tentýž prvek. Pseudotřídy mohou být dynamické, v tom smyslu, že prvek ji může nabýt nebo ztratit zatímco uživatel pracuje se stránkou.

6.5.1 DYNAMICKÉ PSEUDOTŘÍDY

Asi nejznámější příklad bude tag `<A>` tedy odkaz. U toho jsou ustaveny dle CSS2 čtyři kontextuální kritéria a tedy i čtyři pseudotřídy. Navštívené odkazy :visited, aktivní a nenavštívené odkazy :active, propojené odkazy :link a odkazy, nad které umístíme kurzor myši :hover.

V CSS3 přibyly další čtyři a ještě dodatek ke třídám :active a :focus. Pseudotřída :active se aplikuje jakmile uživatel aktivuje link, například mezi tím, co zmáčkneme tlačítko myši a pustíme ho. A pseudotřída :focus se aplikuje v ten moment, kdy je odkaz označen pomocí události na klávesnici nebo jiným způsobem. Tento návrh dodává, ještě, že pouze elementy, jež vložil uživatel a které mají aktivovanou hodnotu, mohou používat právě pseudotřídu :focus nebo :active.

```
a:link      /* nenavštívený odkaz*/
a:visited   /* navštívený odkaz */
```

```
a:hover     /* uživatlské najetí */
a:active    /* aktivní odkaz */
```

Příklad kombinování dynamických pseudo-tříd:

```
a:focus
a:focus:hover
```

6.5.2 CÍLOVÁ PSEUDOTŘÍDA :target

Některá URI (viz dále) se odvolávají na umístění uvnitř zdroje. Tento druh URI končí s „bash“ (#), následovaný identifikátorem

```
*:target { color : red }
*:target::before { content : url(target.png) }
```

6.5.3 JAZYKOVÁ PSEUDODŘÍDA :lang

Jestliže jazyk dokumentu specifikuje jak bude přednastaven jazyk prvků,je možné psát selektory, tak, že reprezentují prvky založené na tomto jazyce. Například v HTML4,je jazyk určený kombinací atributu `lang`,elementem `meta` a popřípadě informací z protokolu (tak jako v hlavičce HTTP). XML užívá atribut nazývaný `xml:lang`, a mohou zde být i jiné dokumenty specifikující jazyk metod pro nastavení jazyka-

Pseudotřída `:lang(C)` popisuje element, jaký je v jazyku C. Zde je C jazyk, tak jak je specifikovaný v HTML 4.01 a v RFC 3066

Tento příklad ukazuje HTML dokument , který je psán ve vlámštině, francouzština nebo němčině. Dva následující selektory zachycují `q` citaci v libovolném elementu, ve všech třech jazycích.

```
html:lang(fr-be)
html:lang(de)
:lang(fr-be) > q
:lang(de) > q
```

6.5.4 PSEUDOTŘÍDY UŽIVATELSKÉHO PROSTŘEDÍ

Pseudotřída :enabled umožňuje autorovi přizpůsobit vzhled prvkům uživatelského rozhraní v nějakém módním stylu. Opak k tomuto je třída :disabled. Pseudotřída :checked, je aplikována na elementy, které uživatel může upravovat, zaškrtávat je apod. Vztahuje se to především na checkboxy a radiobutony. Když je právě takovýto prvek za-škrtnut, pak je právě aktivována tato pseudotřída, která je dynamická. Používá se na všechna media. Poslední je pseudotřída :indeterminate se používá na stejné prvky jako předchozí, ale jen v takovém případě, že tyto prvky jsou i nejsou zaškrtnuté. To může být způsobeno atributem prvku.Pseudotřída je ve stylu listů definována pomocí dvojtečky jako oddělovače (:pseudotřída {specifikace}). Následné volání potom probíhá zcela automaticky a pokud prvek odpovídá některému kontextu automaticky se provede formátování dle daného kontextu.

```
a:visited <color: #FF0000; font-family: helvetica; text-decoration:
none;>
<a href="uvod.html">Zpět na úvod</a>
```

Zde máme předdefinován styl pro tag `<A>`. Pokud tedy dojde k načítání tagu *A*, prohlížeč nejdříve vyhodnotí zda byl zobrazovaný odkaz již navštíven a pokud ano použije automaticky formátování dle definice pseudotřídy. Definice pseudotříd se většinou zapisují s předřazením tagu, ke kterému se vztahují. Je to vhodné obzvláště pro zvýšení přehlednosti stylu listu. V některých případech však může vztahovat pouze k jednomu jedinému tagu. Ovšem pokud použijeme pseudotřídu bez předřazeného tagu, musíme si zároveň uvědomit, že tato pseudotřída se použije ve všech elementech, které tuto pseudotřídu podporují a které odpovídají požadovanému kontextu. Díky tomu se nám mnohdy může stát, že se elementy budou přeformátovávat aniž bychom to chtěli. Proto používání definice pseudotříd bez konkretizace vztažného elementu nemohu doporučit.

6.5.5 STRUKTURÁLNÍ PSEUDOTŘÍDY

Selektory, které zavádějí koncept těchto pseudo-tříd, v tolerovaném výběru, založeném na extra informacích, které leží ve stromu dokumentu nemohou být vyjádřeny jinými jednoduchými selektory nebo kombinací těchto.

6.5.5.1 Pseudotřída :root

Pseudotřída :root zastupuje každý element, který je v dokumentu kořenovým. V HTML 4 je to element HTML. V XML je to kterýkoliv vhodný pro DTD nebo schéma a jmenný prostor pro ten který XML dokument.

6.5.5.2 Pseudotřída :nth-child()

Pseudotřída `:nth-child(an+b)`. Tento zápis říká, že element má an+b-1 sourozence již někde dříve v dokumentovém stromě. Pro daná kladná celá čísla nebo nulové hodnoty. `n`. Jinak řečeno, toto dává potomky elementů do skupin jednotlivých elementů. Toto například dovoluje selektorům adresovat každou jednotlivou řádku a může být použita například ke střídání barev nebo odstavců textu v cyklech po čtyřech. Hodnoty `a` a `b`, musí být nastaveny na nulu, záporné, či kladné celé číslo. Index prvního potomka elementu je 1. `:nth-child()` může mít také hodnotu 'odd' a 'even' pro argumenty. 'Odd' má stejný význam jako 2n+1, a 'even' má stejný výraz jako 2n.

```
tr:nth-child(2n+1) /* každá lichá řádku HTML tabulky */
tr:nth-child(odd) /* ten samý význam */
tr:nth-child(2n) /* každá sudá řádka HTML tabulky */
tr:nth-child(even) /* ten samý význam */

/* Střídání barev v odstavci v CSS */
p:nth-child(4n+1) { color: navy; }
```

```
p:nth-child(4n+2) { color: green; }
p:nth-child(4n+3) { color: maroon; }
p:nth-child(4n+4) { color: purple; }
```

Jestliže se a = 0, není použito žádného opakování, takže například `:nth-child(0n+5)` nastaví pouze pátého potomka Když a=0, není třeba tuto část zahrnovat, tudíž syntaxe se zjednoduší na `:nth-child(b)` a naší poslední ukázku zjednoduší na `:nth-child(5)`.

Jestliže se a = 1, pak může být číslo vynecháno. Tudíž následující příklady znamenají to samé.

```
bar:nth-child(1n+0)  /*  všechny  elementy  bar  jsou  specifikovány
(0,1,1) */
bar:nth-child(n+0) /* to samé */
bar:nth-child(n) /* to samé */
bar /* to samé, ale nižší specifikace (0,0,1) */
```

Když se b = 0, pak je vybraný každý a-tý element.

```
tr:nth-child(2n) /* každá sudá řádka HTML tabulky */
```

Když jsou a i b nulové pseudotřída nepředstavuje žádné elementy ve stromové struktuře dokumentu. Hodnota může být záporná, ale pouze kladné hodnoty an+b, pro n>= 0, mohou být zobrazeny v dokumentové struktuře dokumentu.

html|tr:nth-child(-n+6) /* 6 prvních řádek XHTML tabulky */

6.5.5.3 Pseudotřída :nth-last-child()

`:nth-last-child(an+b)` tento zápis znamená, že prvek, který má nejméně an+b-1 příbuzného někde dále ve stromové struktuře dokumentu. Pro danná kladná celá čísla a nulu.Syntaxe zápisu je stejná jako u předešlé pseudotřídy. Také může obsahovat ´odd´ a ´even´.

```
tr:nth-last-child(-n+2) /* dvě poslední řádky HTML tabulky */
foo:nth-last-child(odd) /* všechny liché řádky ve jmenném prostoru
            foo v rodičovských elementech, počítaných od poslední. */
```

6.5.5.4 Pseudotřída :nth-of-type()

:nth-of-type(an+b) tento zápis znamená, že element, který má an+b-1 má nějakého příbuzného s tím samým názvem někde dříve ve stromové struktuře dokumentu.

```
img:nth-of-type(2n+1) { float: right; }
img:nth-of-type(2n) { float: left; }
```

Tento příklad umožňuje CSS střídat pozice plovoucích obrázků.

6.5.5.5 Pseudotřída nth-last-of-type()

Tato pseudotřída je analogicky opačná k třídě předchozí, tudíž příbuzný prvek existuje někde dále v dokumentu.

```
body > h2:nth-of-type(n+2):nth-last-of-type(n+2)
```

Pomocí tohoto příkladu můžeme na všechny potomky elementu `H2` XHTML dokumentu použít selektor, kromě první a poslední řádky.

6.5.5.6 Pseudotřídy :first-child, :last-child, first-of-type a last-of type

Tyto pseudotřídy jsou stejné jako pseudotřídy :nth-child, :nth-last-child, :nth-of-type, :nth-last-of-type. Můžeme si vybrat, jakou v tom daném případě použijeme.

```
div > p:first-child
ol > li:last-child
dl dt:first-of-type
tr > td:last-of-type
```

V prvním příkladě `p` představuje prvního potomka elementu `div`.

Ve druhém je naopak `li` posledním potomkem seznamu `ol`.

Třetí je definice titulku `dt` uvnitř definice seznamu `dl`, toto `dt` je v dokumentu první svého typu,

A v posledním, čtvrtém příkladě je buňka `td` poslední v řádce tabulky `tr`.

6.5.5.7 Pseudotřídy :only-child a :only-of-type

Tyto pseudo-třídy jsou stejné s :first-child, :last-child nebo nth-child, :nth-last-child a :nth-of-type, :nth-last-of-type, ale s nižší specifikací.

6.5.5.8 5.5.5.7. Pseudotřída :empty

Tato pseudotřída představuje taková prvek, který nemá vůbec žádného potomka.

```
<p></p>
```

`p:empty` je právoplatnou reprezentací na předchozí ukázce.

6.5.6 ZÁPORNÉ PSEUDOTŘÍDY,

Jak již název napovídá, tato pseudotřída reprezentuje, který není popsán žádným argumen-tem.

Následující selektor se vztahuje na všechna tlačítka `button` v HTML dokumentu, která nejsou zakázaná

```
button:not([DISABLED])
```

A níže uvedená skupina selektorů se vztahuje na všechny prvky HTML kromě odkazů.

```
html|*:not(:link):not(:visited)
```

6.6 PSEUDO-ELEMENTY

Pseudoprvky (pseudo-elementy) jsou obdobou pseudotříd a jsou to v podstatě specifické pseudotřídy, které se vztahují ke specifickým kontextům. V CSS3 je zavedena nová syntaxe v uvádění pseudo-elementů a to, že jsou předcházeny dvojicí dvojteček (::), oproti předchozím definicím v kaskádních stylech. A je to z důvodů přehlednosti, aby se viditelněji odlišily od pseudotříd.

V CSS1 jsou definovány dva pseudo-elementy a to ::first-line(řádek) zvýrazní celý první řádek, tak jak si určíme, například, jako v nadcházejícím případě, bude první řádka napsána kapitálkami.

```
p::first-line { text-transform: uppercase }
```

A druhou je ::first-letter(písmeno), díky které se nám zobrazí první písmeno jako iniciálu, nebo ho naopak může udělat menší. V našem případě je ale použito známějšího případu a to iniciály, která bude psána ještě kurzívou a bude tučná.

```
P::first-letter { font-size: 200%; font-style: italic; font-weight:
bold;}
```

Oba tyto pseudo-elementy lze použít i na fiktivní první řádky, či iniciály, jestliže například první dva řádky uzavřeme do tagu p::first-line, pak tyto dvě řádky můžeme opět naformátovat podle vlastních představ.

V CSS3 jsou definovány nové dva pseudo-elementy a to ::selection a ::menu. Selection je aplikován na text, který je označen uživatelem. A :menu je pro autory, kteří pomocí tohoto mohou specifikovat styl a polohu generovaného menu. Tento je potomkem elementů a proto dědí všechny styly od defaultních ::before a ::after, které se používají ke generování obsahu.

7 KASKÁDOVÁNÍ A DĚDIČNOST

Tento modul CSS3 popisuje jak jsou hodnoty přidělovány hodnotám. CSS dovolují různým stylovým předpisům ovlivňovat ztvárnění dokumentu a proces kombinování těchto kaskádních předpisů se nazývá právě kaskádování. Jestliže žádná hodnota nemůže být nalezena během kas-kádování, pak tato hodnota může být zděděna z rodičovského elementu nebo je použita počáteční hodnota.

Hlavní důvod tohoto modulu je přepsat důležité části CSS2 jako module pro CSS3. S očeká-váním počátečních hodnot, všechny rysy popsány zde, rovněž existují v CSS2. Oproti CSS2, se kaskádování změnilo ve dvou případech.

Jeden ze základních principů designu CSS je dovolit jak autorům a uživatelům, ovlivňovat provedení vzhledu dokumentu..

Vstupem kaskádování a dědičnosti jsou:

- Nastavení deklarace, které mohou být aplikovány na prvky, či vlastnosti v kombinací s otázkou. Předpokládá se, že deklarace není používána a není zahrnuta v nastavení.
- Zděděně hodnoty vlastností
- Počáteční hodnoty vlastností.

Výstupem je pak jediná hodnota, známá jako kaskádová hodnota. Prohlížeč musí třídit deklarace podle následujících kritérií, v pořadí důležitosti:

- Důležitost. V CSS3, váha deklarace je založena na původu deklarace a je označena levelem důležitosti. Přečtěte si následující sekce, abyste věděli jak vypočítat důležitost. Deklarace s největší váhou vyhrává.
- Specifikace. Selektory popisují jak vypočítat specifikaci. Deklarace s nejvyšší specifikací vyhrává.
- Pořadí vzhledu. Poslední deklarace vyhrává.

Řadící proces pokračuje dokud je nalezena první vyhrávající deklarace.

7.1 DĚDIČNOST

Dědičnost je cesta rozšiřování vlastností od rodičovských prvků, potomkům. Mnoho vlastností je zděděno, která znamenají, že rodičovské prvky budou automaticky užity bez dalších specifikací. Všechny vlastnosti přijímají klíčové slovo inherit, které explicitně specifikuje, že hodnoty mohou být přineseny od předků. Základní prvek, který nemá žádné předky, používá počáteční hodnotu naproti hodnot zděděných.

Jako obecné pravidlo platí to, že vypočtená hodota vlastnosti, která je zděděná. Vlastnosti mohou být specifikované tak jako ostatní druhy hodnot mohou být místo toho zděděny.

7.1.1 VYPOČTENÍ VÁHY DŮLEŽITOSTI

V pořadí třídění deklarace v procesu kaskádování, váha deklarace musí být známa. V CSS3, váha deklarací je založena na původu deklarace a je označena úrovní důležitosti. Kaskádní styly mohou mít tři rozdílné původy: autor, uživatel a prohlížeč.

- Autor specifikuje kaskádní styly pro zdroj dokumentu podle konvencí jazyka dokumentů. Například, v HTML, kaskádní styly mohou být obsaženy v dokumentu nebo provázány zvenku.
- Uživatel může specifikoval informace stylů pro jednotlivé dokumenty. Například uživatel může specifikovat soubor, který obsahuje kaskádní styly nebo prohlížeč může poskytnout rozhraní, které generuje uživatelův stylový předpis.
- Prohlížeč musí používat standardní stylový předpis, dříve než všechny ostatní stylové předpisy dokumentu.

Každá deklarace CSS může mít dvě různé úrovně důležitosti: normální (který je standardní) a důležitá (která musí být označeny). Syntaxe CSS3 popisuje jak jsou deklarace označovány jako důležité. Váha důležitosti kaskádních stylů z různých předpisů, vzestupně je:

- Kaskádní předpis prohlížeče
- Normální kaskádní předpis uživatele
- Normální předpis autora
- Důležitý předpis autora
- Důležitý předpis uživatele

Standardně, tato strategie dává autorovi deklarace větší váhu, než ten samý u uživatele.

8 PRAVIDLA PRO ZAPISOVÁNÍ KASKÁDNÍCH STYLŮ

Máme již základní vědomosti o CSS, je ale důležité, při jejich používání dodržovat určitá pravidla, aby pak naše stránky vypadaly, tak jak opravdu chceme.
Jak víme, z předchozích kapitol, stylové předpisy (style sheets) mohou být umístěny jak v samo-statném souboru, tak v hlavičce dokumentu v prvku `<style>` nebo přímo v jednotlivých HTML prvcích, v atributu style. Pro základní použití v konečných verzích dokumentů se hodí pouze první způsob. Druhý je vhodný spíše pro ladění a třetí slouží pro zcela výjimečné případy.
Hlavní výhody externího stylového předpisu jsou tři:
1. Jeden externí stylový předpis zajistí stejný styl celému dokumentu(nebo jeho části).
2. Stylový předpis se načte jen s první stránkou a pak se již kešuje.
3. Externí stylový předpis lze snadno ukrýt před staršími prohlížeči, které by ho nedokázaly správně interpretovat.

Dalším pravidlem je nepoužívání obecných HTML tagů zbytečně. S nástupem kaskádových stylů se rozšířily i obecné HTML prvky a `<span>`, které dříve nebyly skoro potřeba. Ale ani pro stránky, plně formátované pomocí CSS nejsou příliš potřeba alespoň ne v přehnané míře. Prvky `<div>` a `<span>` v kombinaci s třídou nebo id se hodí pouze tehdy, když pro danou strukturu neexistuje vhodný HTML tag. V případě blokového `<div>` to jsou nejčastěji takové oblasti stránky, jakými jsou záhlaví, zápatí, hlavní text a okrajový sloupec, v případě řádkového `<span>` to může být například drobná poznámka v textu. Naopak pro menu, zvýrazněný název firmy nebo ukázku zdrojového kódu by vždy měly být použity adekvátní prvky HTML, jejichž specifičnost se vyznačí třídou.

Jedním z častých problémů bývá, kdy použít třídy (class) a kdy identifikátory (id). Pravidlo je jednoduché: Smí-li být daný prvek v dokumentu pouze jednou, je správnější id, v opačném případě se musí použít třída. Systematické a správné používání id zvyšuje přehlednost kódu a jeho odolnost proti chybám. Tomu, kdo se

podívá do kódu, je hned jasné, že prvek označený id smí být v dokumentu maximálně jednou.

Častou chybou je též nevhodné pojmenování tříd. V případě návrhu stránky, bývá identifikátor nazván například `prvninadpis` nebo třída `zelenyodstavec`. Během vývoje stránek se však první nadpis přestěhuje až na pozici pátého a z původně zeleného odstavce se stane modrý. Výsledkem je dokonalý zmatek, ve kterém se nevyzná ani sám autor. Třídy a id proto pojmenovávejme zásadně podle jejich funkce či strukturálního významu, ne podle toho, jak nyní vypadá pro ně definovaný styl.

Typický HTML kód začátečníka v CSS je plný tříd. A může vypadat například takto:

```
<h2 class="nadpisclanku">Nadpis</h2>
<p class="odstavecclanku">...</p>
<p class="odstavecclanku">...</p>
...
<p class="odstavecclanku">...</p>
```

Mnohem efektivnější je řešení pomocí vymezení strukturálních bloků pomocí prvku `<div>` a kontextových selektorů. A předešlý kód může vypadat po takové úpravě takto:

```
<div class="clanek">
  <h2>Nadpis</h2>
  <p>...</p>
  <p>...</p>
  ...
  <p>...</p>
</div>
```

Ve stylovém předpisu pak již jen nadefinujeme pravidla pro kontextové selektory:

```
.clanek h2 {...}
.clanek p {...}
```
A přehlednější a nepřehlcené řešení je hotovo.

Podobně, jako není dobré, když máme příliš mnoho tříd v HTML kódu, není dobrý ani nadbytek názvů tříd. Neustále nové názvy tříd se špatně vymýšlejí a ještě hůře pamatují. CSS přitom nabízí elegantní mechanismus, jak tentýž název třídy použít v různém významu. Například chceme-li odlišně formátovat první písmeno odstavce, první odstavec článku nebo první položku seznamu (pro tento konkrétní účel zapomeňme, že CSS nabízí pseudo-elementy, je to jen příklad). Ve všech třech případech však bude formátování odlišné. Potřebujeme na to tři třídy? Nikoli, stačí nám jedna a tyto tři definice:

```
p span.prvni {...} /* pro první písmeno odstavce */
p.prvni {...} /* pro první odstavec v článku */
li.prvni {...} /* pro první položku seznamu */
```

Je to daleko elegantnější a odpadá problém s vymýšlením stále nových tříd.

Vyšší efektivity, přehlednosti a udržovatelnosti kódu lze často dosáhnout i tím, že do jedné třídy sdružíme vždy ta pravidla, která se používají společně a na některé HTML prvky pak aplikujeme více těchto tříd současně. Pokud tedy například často potřebujeme červené pozadí, podtržené tučné písmo, barevné zvýraznění nebo libovolnou kombinaci těchto tří vlastností, nemusíme připravovat víc tříd než tyto tři:

```
.pozadi {
  background-color: #990000;
}
.tucnepodtrzeno {
  font-weight: bold;
  font-style: underline;
```

```
}
.barevna {
  color: #FF9900;
}
```

V HTML kódu pak již můžeme tyto třídy libovolně kombinovat, například takto:

```
<p class="pozadi tucnepodtrzeno">   /* nebo */
 <span class="tucnepodtrzeno barevna">.
```

K dobrému stylu patří i správné a důsledné využívání toho, že kaskádové styly kaskádují. Je tedy zbytečné psát všechny tyto definice:

```
h1 {
  font-family: "Arial CE", Verdana, sans-serif;
  font-style: italic;
  font-size: 200%
}
h2 {
  font-family: "Arial CE", Verdana, sans-serif;
  font-style: italic;
  font-size: 160% }
```

Jednodušeji, přehledněji a úsporněji můžeme to samé zapsat takto:

```
h1, h2 {
  font-family: "Arial CE", Verdana, sans-serif;
  font-style: italic;
}
h1 {
  font-size: 200%
}
h2 {
```

```
  font-size: 160%
}
```

Kaskádování však nespočívá jen v postupném přepisování či doplňování vlastností. Nové hod-noty lze odvozovat i z rodičovských vlastností. Můžeme tak kaskádovat třeba velikost písma. Používáme-li například pro určení velikosti písma pixel, deklarujme v pixelech velikost jen jednou pro základní písmo a další odstupňování provedeme relativně procenty.

```
body {
  font-size: 14px;
}
h1 {
  font-size: 200%
}
#Uvod {
  font-size: 85%
}
```

Snadno pak na jediném místě můžeme změnit různé velikosti písma po celém dokumentu.

Důležité je dodržovat i kaskádové pořadí. Nezáleží v jakém pořadí zapíšeme elementy do stylového předpisu, jako v tom, jak je budeme volat a přetěžovat. Pokud používáme technologii vkládaných stylů, pak mají nejnižší prioritu propojené styly. Pak následují globální styly a nejvyšší prioritu mají pak in-line styly. Pokud jde o prioritu elementů, tak nejvyšší prioritu mají identifikátory (ID). Za nimi pak třídy (CLASS), poté pseudotřídy nebo pseudo-elementy a nejnižší prioritu mají definice základních elementů HTML. Ale priority můžeme definovat podle sebe.

Při kaskádování si ale musíme dát pozor na sdružené vlastnosti. Uvědomme si totiž, že když například nadefinujeme:

```
p {
  font: 14px sans-serif;
}
```

neříkáme jen to, že písmo bude o velikosti 14px a bude psáno ve fontu sans-serif, ale to, že to napíšeme takto, znamená, že určíme tím i všechno toto:

```
p {
  font-style: normal;
  font-weight: normal;
  font-size: 14px;
  line-height: normal;
  font-family: sans-serif;
}
```

Přehlednost stylového předpisu (myšlen je zejména externí) zvýší a jeho údržbu usnadní vhodné systematické uspořádání. Praktický je například zvyk, zapisovat každé jednotlivé pravidlo na samostatný řádek, vždy ukončený středníkem, namísto dlouhé šňůry pravidel na jediném řádku. Usnadní nám to kopírování pravidel mezi jednotlivými selektory.

Dále je dobré nezapomenout, že i jazyk CSS nabízí komentáře, které jsou vymezené znaky /* (začátek komentáře) a */ (konec komentáře). Pečlivě komentovanému stylovému předpisu budeme po čase mnohem lépe rozumět a snadno v něm nalezneme to, co právě potřebujeme změnit. A patří to k dobrým mravům jak u programátorů, tak u tvůrců webovských stránek. Na konec je dobré si zvyknout i na určitou posloupnost selektorů. Je v zásadně jedno jakou, hlavně že v tom máme nějaký systém, který dlouhodobě dodržujeme.

9 HODNOTY

Tento modul CSS3 popisuje rozličné hodnoty a jednotky, které mohou vlastnosti CSS používat. Popisuje jak specifické vlastnosti, které jsou to, co stylové předpisy obsahují a ty jsou pak zpracovány do vypočtených vlastností a aktuálních vlastností.

V CSS3 je 5 základních typů hodnot. Jsou to - klíčová slova (např. `pitch-range: inherit`), čísla (např. `orphans: 3`), čísla s jednotkovými identifikátory (např. `border-width: 0.2em`), řetězce (např. `content: ´Figure´`), funkce (např. `background: url (http://www.w3.org/ image)`) a speciální případy (např. `color: #F00“ a „font-size:Helvetica`). Většina vlastností akceptuje hodnoty z různých předchozích typů. Každá vlastnost má formální definici o tom, jaké typy hodnot mohou používat, tyto mohou být uvedeny v popisu vlastnosti.

9.1 KLÍČOVÁ SLOVA

V oficiálních definicích hodnot vlastností, jsou klíčová slova zapisována doslovně. Zde jsou pro příklad uvedeny hodnoty definice pro vlastnost *border-collapse* a jejich následné využití:

```
Value: collapse | separate | inherit
table { border-collapse: separate}
```

Všechny vlastnosti CSS3 umožňují použití hodnotu klíčového slova *inherit*. Klíčová slova nejsou nikdy uvedena v uvozovkách ani apostrofech.

9.2 ČÍSLA

Hodnoty číslovek mohou být jak typu integer, tak real. Hodnoty nabývající hodnoty typu integer, jsou předcházeny tagem <integer> a hodnoty nabývající hodnot typu real tagem `<number>`. Obojí jsou definovány jen v desítkové soustavě, mohou nabývat jak kladných, tak záporných hodnot.

9.3 ČÍSLA S JEDNOTKOVÝMI IDENTIFIKÁTORY

9.3.1 DÉLKOVÉ JEDNOTKY

Mnoho parametrů nastavovaných v CSS je hodnota udávající míru nebo vzdálenost. Tyto hodnoty se zapisují jako celá, či desetinná čísla. Ta se oddělují tečko a ne čárkou, jak je všeobecná zvyklost. Mohou nabývat kladných i záporných hodnot. Bezprostředně za číslicí následují jednotky, ve kterých je hodnota zadána, jestliže zde uvedeme mezeru, je to chyba. Tyto jednotky jsou vyjádřeny dvou písmennými zkratkami. Jestliže je číslice uvedena bez své jednotky, je jí implicitně přiřazena hodnota pixelů. Používané jednotky se dělí na absolutní a relativní.

Absolutní jednotky jsou takové, jejichž hodnota je přesně definována a je odvoditelná os základních jednotek tabulky SI. Tyto jednotky jsou uvedeny v následující tabulce. Do této skupiny bývá někdy zařazován i pixel. Pixel je však bezrozměrný a jeho aktuální fyzická velikost je závislá na nastavení parametrů zobrazení monitoru popřípadě i tisku na tiskárně.

cm	Centimetry, 1cm = 0,01m
mm	Milimetry, 1mm = 0,1cm
in	Palce, 1in = 2,54cm = 72pt = 6pc
pt	Body, 1pt = 1/72in = 1/12pc
pc	Pica, 1pc = 12pt

V případech, kde specifikace délky nemůže být podporována, musí prohlížeč převést číslo do přibližné hodnoty.

Relativní jednotky jsou takové jejichž hodnota není přesně definována a závisí na nastavení jiných parametrů. Potomci nemohou dědit relativní hodnoty, dědí většinou vypočítané hodnoty. V tomto příkladě hodnota *text-indent* elementu *H1*, budou 36pt a ne 45pt, jestliže je *H1* potomek elementu *H1*.

```
body{
      font-size: 12pt;
      text-indent: 3em;  /* to jest */
}
H1  { font-size: 15pt }
```

Zde je seznam relativních jednotek uveden v následující tabulce.

px	Pixel, 1pixel = 1 bod obrazovky
em	vztažná jednotka, odpovídá šířce „m“
en	vztažná jednotka, odpovídá šířce „n“
ex	vztažná jednotky odpovídá výšce „x“

9.3.2 PROCENTA

Formát procentuálních hodnot (je předcházen ::definicí `<percentage>`) je v této specifikaci následován číslem a pak znakem ´%´. Výhodou je, že pevně nenastavujeme konečné hodnoty, ale pouze upravuje implicitní či dříve nastavené hodnoty. Je to stejné jako použití v HTML. Jsou ve své podstatě pouze další relativní jednotkou, to znamená, že dle vztažné hodnoty dojde k odvození absolutní hodnoty.

9.3.3 ÚHLY

Úhly jsou značeny `<angle>` a jsou používány u zvukových stylových předpisů. Jsou následovány číslem a identifikací úhlu:

- deg : stupně
- grad : gradiány
- rad : radiány

Mohou také nabývat záporných hodnot. Jsou v rozsahu od 0 do 360 stupňů.

9.3.4 ČASY

Definice času je předcházen tagem <time>, také se používají v mluvených stylech. Po čísle následuje identifikátor času, které mohou být:

- ms : milisekundy
- s : sekundy

Nemohou nabývat záporných hodnot.

9.3.5 FREKVENCE

Frekvence jsou označovány <frequency> a jsou také používány, jako předchozí. A identifikátory těchto jednotek jsou:

- Hz : Hertze
- kHz : kilo Hertze

Nemohou nabývat záporných hodnot.

9.4 ŘETĚZCE

Definice řetězců mohou být psány jak s dvojitými uvozovkami, tak s apostrofy. Uvozovky se nemohou vyskytovat uvnitř jiných uvozovek bez oddělení escape sekvencemi (jako '\"').

Příklady:

```
„toto je ´řetězec´"
„toto je \"řetězec\""
'toto je „řetězec"´
'toto je \´řetězec\´'
```

Řetězce nemohou obsahovat přímo novou řádku. Musíme použít backslash (zpětné lomítko), jako v tomto případě:

```
A[TITLE="ne moc dlou\
hy titulek"]    {/* . . .*/}
A[TITLE="ne moc dlohy titulek"] {/* . . . */}
```

9.5 FUNKCE

9.5.1 URI

URL (Uniform Resource Locators) poskytuje adresy zdrojů na webu a očekávaným novým zdrojem je URN(Uniform Resource Name). Dohromady se tyto dva zdroje nazývají právě URIs(Uniform Resource Identifiers). A právě tato specifikace užívá URI. Hodnoty URI jsou označovány <uri>. Funkcionální notace určená URI v hodnotách je jako při užití URL, tak jako zde:

```
body { background: url(„http://www.obrazky.cz/dracek.gif“) }
```

Uvozovky ani mezery nejsou povinné. Jestliže nechceme používat absolutní cestu ke zdroji, můžeme použít i relativní URI. Pro CSS, základ URI je ve stylových předpisech a ne ve zdroji dokumentu. Pro příklad musíme předpokládat následující pravidlo:

```
body { background: url(„yellow“) }
```

je umístěný ve stylovém předpisu navrženém URI:

```
http://www.priklady.cz/styly/zakladni.css
```

Pozadí zdroje dokumentu body bude vydlaždičkován jakýmkoliv obrázkem, který je popsán
zdrojem navrženým v URI
`http://www.priklady.cz/styly/yellow`

Prohlížeč může zdroje změnit na nedosažitelné nebo nepoužitelné díky URI, které může být nefunkční nebo neexistující.

9.5.2 POČÍTADLA

Definice počítadel je označována identifikátory buď `<counter-increment>` a `<counter-reset>`. Vkládání počítadel se často využívá při automatickém generování obsahu. V CSS2 hodnoty počítadel mohly být uvedeny ve vlastnosti *content*.

Zde je stylový předpis, který počítá odstavce (*P*) pro každou kapitolu (*H1*). Odstavce jsou číslovány římskými číslovkami, následovanými tečkou a mezerou:

```
P {counter-increment: par-num}
H1 {counter-reset: par-num}
P:before {content: counter(par-num, upper-roman) ". "}
```

Počítadla, která nejsou v rozsahu žádného `counter-reset`, jsou vynulovány na nulu pomocí `counter-reset`.

9.6 SPECIÁLNÍ PŘÍPADY

Existují dva společné případy nespadají pod žádné výše popsané. Jsou to *color*, které jsou předcházeny # a *font-family*. *Font-family* jsou stejné jako řetězce, ale jsou u nich očekávány buď uvozovky nebo apostrofy.

9.6.1 BARVY

Při zápisu barev máme dvě hlavní možností zápisu. První možností je zápis pomocí anglického jména v řetězcové notaci. Jména barev jsou definována v normě HTML. Seznam klíčových slov barev jsou:

aqua	světle modrá	**navy**	modrá
black	černá	**olive**	hnědá
blue	modrá	**purple**	fialová

fuchsia	světle fialová	**red**	červená
gray	šedá	**silver**	stříbrná
green	zelená	**teal**	tyrkysová
lime	světle zelená	**white**	bílá
maroon	červená	**yellow**	žlutá

Druhou možností jak zapsat barvu, je zadat přímo barevné složení RGB (Red – Green - Blue) v hexadecimální notaci. Pro tento zápis existují v CSS čtyři syntaktické způsoby. V každém z těchto způsobů odděleně definujeme tři složky barvy, červenou, zelenou a modrou. A to buď pomocí hexadecimálních čísel, dekadických čísel a nebo procentuálního vyjádření.

#rgb	tři číslice Hex soustavy (#F00 – červená)
#rrggbb	šest číslic Hex soustavy (#0000FF – modrá)
rgb(r, g, b)	r, g, b – čísla Dec soustavy 0 - 255
rgb(r%, g%, b%)	r, g, b – procentuální vyjádření barev 0 – 100

10 SPECIFIKOVANÉ, VYPOČTENÉ A AKTUÁLNÍ HODNOTY

Poslední hodnotou CSS3 pro vložené prvky je výsledek tří krokové kalkulace: hodnota je specifikována ve stylovém předpisu (specifické hodnoty), potom je rozhodnuto o absolutní hodnotě, když je třeba (vypočtená hodnota) a nakonec je přeměněna podle ohraničení lokálního prostředí (aktuální hodnota). S těmito hodnotami ale my pracovat přímo nebudeme, proto se o nich zmíním jen okrajově.

10.1 SPECIFIKOVANÉ HODNOTY

Jsou to hodnoty, které jsou specifikované ve stylových předpisech. Konečná deklarace obsahuje specifikované hodnoty. Často tyto hodnoty nepotřebují žádné počítání. Jestliže neexistují žádné konečné deklarace, nejsou ani specifikované hodnoty.

10.2 VYPOČTENÉ HODNOTY

Specifikované hodnoty mohou být absolutní (např. když nejsou specifikovány relativně k jiným hodnotám jako *re*“ nebo 2mm) nebo relativní (např. když jsou specifikovány relativně k jiným hodnotám, tak jako v *auto* nebo „12%“). Pro absolutní hodnoty není třeba žádného počítání k tomu abychom zjistili vypočtené hodnoty.

10.3 AKTUÁLNÍ HODNOTY

Vypočtené hodnoty jsou v podstatě připraveny k používání, ale prohlížeč nemusí být schopen použít tyto v daném prostředí. Například prohlížeč může být schopen vyjádřit ohraničení v pixelech a možná proto musí přibližně vypočítat danou šířku. Aktuální hodnota je vypočtená hodnota po provedených úpravách.

11 TYPY MÉDIÍ A PREFERENCÍ

11.1 ROZLIŠENÍ TYPU MÉDIÍ

Norma CSS level 2 zavedla novou možnost specifického nastavení pro jednotlivé zobrazovací média. To znamená, že můžeme jednu stránku optimalizovat pro zobrazení na klasickém monitoru, televizním přijímači a tiskárně, aniž bychom byli nuceni vytvářet pro každý způsob zobrazení speciální stránku. Stačí, když vytvoříme tři různé definice stylů pro každý typ média a prohlížeče, pak za nás sám vybere ten správný způsob zobrazení. Tato vlastnost je dosti převratná a i když možná v současné době nebude tolik využívána, tak její čas ještě přijde.

11.2 PARAMETR @media

Tento parametr přiřazuje nadefinovaný styl jednotlivým zobrazovacím a zvukovým mediím. Hodnoty *all* – pro všechna média; aural-zvukové syntezátory, norma CSS2 podporuje i tzv. zvukové styly.

11.3 TYP MÉDIA ‘presentation’(prezentace)

Médium obrazovka typicky naznačuje okno na grafickém displeji nebo na počítači. Pro takováto zařízení je typické, že mají vysokou přesnost audio stereo schopnosti a schopnost zobrazovat obsah pokrytý kompletní grafický display, namísto jen v okně. V tomto je podstatný požadavek, aby autorův obsah byl schopen tento mód akceptovat. Prezentace se od projekce liší tím, že není poskytována pro širší obrazovku, zatímco projekce většinou je.

11.4 @preference

CSS @media pravidla dovolují autorům, specifikovat rozdílné vzhledy obsahů, které jsou založeny na tom, jaké médium je poskytnuto nebo jaký druh zařízení je používáno. Mnoho přístrojů podporují reprodukci na multimediálních médiích, typický příklad je počítač, na který je připojena tiskárna a tak podporuje jak tisk, tak

výstup na obrazovku. Takovýto počítač má také většinou audio schopnosti a pak může podporovat i zvukové styly. Pravidlo @preference a vlastnost audio, dovolují autorovi označit které médium je preferováno pro vyobrazení obsahu. Například, obsah, který byl navrhnut k zobrazení a nakonec k tisku, může obsahovat pravidlo @preference v našem stylovém předpise.

```
@preference {
 media: presentation, screen, print;
}
```

Toto pravidlo, říká prohlížeči že se má pokusit vypsat obsah jako prezentaci , která je v tomto případě defaultní a jestliže toto není možné, pak to zobrazit na obrazovku a nakonec připravit pro tisk.

Jediná vlastnost aktuální pro @preference jsou "média", která určuje uspořádanému seznamu v médiích, kterému obsahu je určen. Jestliže prohlížeč podporuje více jak jedno médium, uvedené na seznamu, defaultním se může stát první médium v seznamu, které je na našem počítači podporováno.

12 PŘEHLED VLASTNOSTÍ KASKÁDNÍCH STYLŮ

Následující kapitoly obsahují souhrn všech vlastností jak CSS1,CSS2, tak i CSS3, u kterých jsou v některých případech nové hodnoty, ale především jsou zahrnuty v určených modulech, jiné dokonce tvoří samostatné moduly a proto je zde uvádím všechny. Některé jsou i nezbytně nutné jako základ pro rozšíření v CSS3.

13 CSS3 MODUL COLOR (BARVA)

V CSS1 byla pro barvy definována jediná a základnív vlastnost a to vlastnost *color*. Ta je základem pro všechny ostatní nadstavby. Lze ji aplikovat na všechny elementy, je dědičná a je uvedena v jednom ze tří dříve uvedených formátů. Její implicitní hodnota je závislá na typu a nastavení prohlížeče.

```
em { color: red }
em { color rgb(255,0,0) }
```

V CSS3 je barvě věnován celý jeden samostatný modul, ten popisuje vlastnosti CSS, které dovolují autorovi specifikovat jak barvu popředí, neprůhlednost, či průhlednost prvků. Přídavné vlastnosti dovolují specifikovat barvu profilu. Tento modul také detailně popisuje výše uvedenou vlastnost *color*.

13.1 TRANSPARENTNOST: VLASTNOST opacity (NEPRŮHLEDNOST)

Neprůhlednost může být chápána konceptuálně jako postprocesorová operace, poté, co je prvek vyobrazen v RGBA barvách. Nastavení neprůhlednosti specifikuje smísení vyobrazení v současném složení zobrazení.

- Hodnoty: `<alphavalue> <priority-index>` |*inherit*.
- Implicitní hodnota: 1 none
- Lze ji aplikovat na všechny elementy a na vizuální média.

- Tato vlastnost není dědičná.
- Vypočítané hodnota: stejná jako specifická hodnota po oříznutí hodnoty `<alphavalue>` do rozsahu [0,0,1,0]

Hodnota `<alphavalue>` nabývá decimálních hodnot. [0,0,1,0]. Jakákoli hodnota mimo rozsah 0,0 – plně transparentní, až do hodnoty 1,0 – plně neprůhledný, bude zmenšený do tohoto rozsahu.

Hodnota `<priority-index>`. Doplňková hodnota integeru nebo klíčové slovo-*none*. Tato hodnota vyjadřuje priority pro prvek, používá některé hardwarové urychlení efektu opacity.

13.2 BARVA PROFILU: VLASTNOST color-property

Tato vlastnost povoluje specifikaci zdroje nastavit na jinou, než je defaultní.

- Hodnoty: *auto, sRGB, <název>, <uri>, inherit*
- Implicitní hodnota: *auto*
- Dá se aplikovat na všechny elementy a na vizuální média
- Tato vlastnost je dědičná
- Vypočítané hodnota: specifická hodnota

Hodnota auto je defaultně nastavená. Všechny barvy jsou předdefinovány v sRGB barvách,
bez obsahu dat. Pro obrázky nastavují profil na data toho profilu, který je právě používán.

Hodnota sRGB. Zdroj profilu je převzatý z sRGB. Toto je rozdílné od hodnoty auto, ve kterém je vložený profil převýšen obrázkem.

Hodnota `<název>` odpovídá definované barvě profilu, která je popsána v databázi prohlížeče. Prohlížeč hledá takový barevný profil, který odpovídá hodnotě v tomto atributu. Jestliže není žádný vhodný nalezen, pak je použit vložený profil uvnitř obrázku.

Hodnota <uri> určuje zdroj profilu.

```
/* užití specifického profilu, přesto, že obrázek obsahuje vložený
profil */
IMG { color-profile: url("http://příklady.cz/profily/eg.icm"}
```

13.2.1 VLASTNOST rendering-intent

Tato vlastnost povoluje specifikaci barevného profilu zobrazujícího jiný účel, než je ten standardní. Chování jiných hodnot, než automatických je definována International Color Consorciem (ICC).

- Hodnoty*: auto, perceptual, relative-colorimetric, saturation, absolute-colorimetric, inherit*
- Defaultní hodnota: *auto*
- Dá se aplikovat na všechny prvky a na vizuální média
- Dědičnost: ano
- Vypočítané hodnota: specifická hodnota

Pomocí hodnoty auto, prohlížeč určuje nejlepší záměr, který je založen na obsahu typu. Ostatní hodnoty jsou popsány a spíše se vztahují k ICC, tudíž je zde nebudu uvádět, ani W3C se popisem těchto vlastností nezabývá.

13.2.2 PRAVIDLO ZAVINÁČE: @color-profile

SVG 1.0, představuje @color-profile jako metodu pro seskupování vlastností `color-profile` a `rendering-intent`.

Toto pravidlo, může být použito ke specifikaci popisu barevného profilu. Všeobecný formát je:

```
@color-profile {<color-profile-description>}
/* kde <color-profile-description> má tuto formu: */
deskriptor: hodnota
[...]
```

```
deskriptor: hodnota
```

Každé pravidlo @color-profile upřesňuje hodnotu pro každý barevný profil deskriptoru, jak pro implicitní, tak i pro explicitní hodnoty.

13.3 NOVÉ BAREVNÉ JEDNOTKY

13.3.1 HODNOTY RGBA

Model RGB je zde rozšířen. Zahrnuje filtr alpha, který dovoluje specifikovat průhlednost barvy. Následující příklad definuje tu samou barvu:

```
em { color: rgb(255,0,0)}          /* integerový rozsah od 0 do 255
*/
em { color: rgb(255,0,0,1)}        /*    to    samé    s    explicitní
průhledností 1 */
em { color: rgb(100%,0%,0%)}       /* rozsah float od 0,0% do 100,0%
*/
em { color: rgb(100%,0%,0%,0%)}    /*    to    samé    s    explicitní
průhledností 1 */
```

Na rozdíl do hodnot RGB tyto hodnoty nemají žádné hexadecimální značení. Formát hodnot RGBA ve své notaci je bezprostředně následován hodnotou <alphavalue> a za ní je uzavírací kulatá závorka. Následující příklad specifikuje nový efekt, který je teď díky RGBA možný:

```
p { color: rgb(0,0,255,0.5)}       /* polotransparentní sytě modrá
*/
p { color: rgb(100%,50%,0%,0.1)}   /*   velice   transparentní   sytě
oranžová */
```

Jestliže prohlížeč nepodporuje hodnoty RGBA, tyto mohou být upraveny, tak jako nerozpoznatelné hodnoty pro CSS. Nesmí být používány jako jednoduché RGB hodnoty s ignorací průhlednosti.

13.3.1.1 NOVÉ KLÍČOVÉ SLOVO PRO BARVY: transparent

CSS1 uvedlo klíčové slovo transparent pro vlastnost *background-color*. CSS2 dovoluje použít toto klíčové slovo také pro vlastnost *border-color*. CSS3 dále rozšiřuje použití tohoto klíčového slova na všechny vlastnosti které akceptují hodnotu color. Tímto se zjednoduší definice těchto vlastností v CSS3.

13.3.1.2 HODNOTY BAREV HSL

CSS3 přidává číselné vyjádření jasnosti sytosti odstínu -Hue Saturation Lightness (HSL). To bylo zavedeno z důsledků, že barvy RGB mají následující omezení:

- RGB jsou orientovány hardwarově: to odráží použití jednotky CRT
- RGB jsou méně přizpůsobeny k tisku.
- RGB nejsou intuitivní

Je zde několik různých barevných schémat . Ale výhody HSL jsou, že jsou symetricky stejné jak ke světlosti, tak tmavosti a je velmi jednoduché převést HSL na RGB.

HSL jsou zakódovány ve trojici čísel, které určují (jasnost, sytost, odstín). Jasnost je vyjádřena jako určitý úhel v barevném kruhu (představme si to jako duhu stočenou v kruhu). Podle definice červené = 360 a ostatní barvy jsou rozprostřeny okolo tohoto kruhu. Tak například zelená = 120, modrá = 240, atd. Sytost a odstín jsou vyjádřeny procentuálně. 100% je plná sytost, zatímco 0% je už odstín šedé. 0% odstín je černá, 100% odstín je naopak bílá a 50% odstín je normální. Takže například:

```
* { color: hsl(0, 100%, 50%)}            /* červená */
* { color: hsl(120, 100%, 50%)}    /* zelená */
```

```
* { color: hsl(120, 100%, 25%)}          /* světle zelená */
* { color: hsl(120, 100%, 75%)}          /* tmavě zelená */
* { color: hsl(120, 50%, 50%)}           /* pastelově zelená */
```

Výhodu HSL nad RGB, je že použití HSL je intuitivní: můžeme hádat barvy, které chceme. Je také jednodušší vytváření nastavení barev(ponechání stejné jasnosti a kombinací sytosti a odstínu).

13.3.1.3 HODNOTY BAREV HSLA

Stejně tak jako RGB, mají mají doplněk filtr alpha a tím i RGBA, tak i HSL mají HSLA.

```
* { color: hsl(120, 100%, 50%)}    /* zelená */
* { color: hsl(120, 100%, 50%, 1)} /*   to   samé,   s   explicitní
průhledností 1 */
```

Formát HSLA barevných hodnot ve funkcionální notaci je hsla (následuje jasnost ve stupních, sytost a odstín v procentuální hodnotě a následně alphavalue, následováno). Mezera je použita okolo číselných hodnot. V dalším příkladě je upřesněn efekt , který je nyní možný díky nové *hsla()* notaci:

```
p { color: hsl(240, 100%, 50%, 0.5)}     /*  polotransparentní  sytě
                                         modrá */
p { color: hsl(30, 100%, 50%, 0.1)}      /*   velmi   transparentní
                                         oranžová */
```

13.3.2 KLÍČOVÉ SLOVO X11 PRO BARVY

Seznam barev X11, jsou takové, které jsou podporovány oblíbenými prohlížeči. První sloupeček používá názvy hodnot barev a druhý používá číselné hodnoty, které ale vyjadřují samozřejmě ty samé hodnoty.

- Aqua #00FFFF 0,255,255
- Chocolate #D2691E 210,105,30
- IndianaRed #CD5C5C 205,92,92

Toto je samozřejmě jen malá ukázka, protože celou tabulku nebo přehled si můžete najít jednoduše na internetu a uvádět zde by bylo zbytečné.

13.3.3 CSS SYSTÉMOVÉ BARVY

Kromě schopnosti přiřadit předdeklarované barvy textu, pozadí apod., CSS3 stejně jako CSS2 dovoluje autorům specifikovat barvy ve stylu, který uceluje uživatelské prostředí. Pravidla stylů, berou v úvahu uživatelské předvolby, kterým nabídnou následující výhody:

- Vytváří stránky, které vyhovují uživatelsky definovanému vzhledu a cítění
- Vytváří stránky, které mohou být více dostupné uživatelskému nastavení.
- Pro systémy které nemají odpovídající hodnoty, mohou být tyto zmapovány v nejbližším systémovém atributu, nebo nastaveny na defaultní barvu.

ActiveBorder	barva okraje aktivního okna
ActiveCaption	barva titulkového pruhu
AppWorkspace	barva pozadí mnohonásobného rozhraní dokumentu
Background	pozadí v počítači
ButtonFace	barva povrchu trojrozměrných prvků

ButtonHighlight	tmavý stín trojrozměrných objektů
ButtonShadow	barva stínu trojrozměrných prvků
ButtonText	text tlačítek
CaptionText	text v titulku, minimalizace, posuvník
GrayText	šedivý (neviditelný) text. Tato barva je nastavena na #000 jestliže váš displej nepodporuje sytě šedou barvu
Highlight	barva vybraných položek
HighlightText	barva textu vybraných položek
InactiveBorder	barva okraje neaktivního okna
InactiveCaption	barva neaktivního titulkového pruhu
InactiveCaptionText	barva textu neaktivního titulkového pruhu
InfoBackground	barva pozadí nástrojů pro funkční klávesy
InfoText	barva textu nástrojů funkčních kláves
Menu	barva pozadí menu
MenuText	barva textu v menu
Scrollbar	barva pozadí posuvníku
ThreeDDarkShadow	tmavě šedá barva pro trojrozměrné prvky
ThreeDFace	barva povrchu trojrozměrných prvků
ThreeDHighlight	barva vybraných trojrozměrných prvků
ThreeDLightShadow	světlá barva pro trojrozměrné prvky
ThreeDShadow	tmavě šedá barva pro trojrozměrné prvky
Window	barva pozdí okna
WindowFrame	barva rámu okna
WindowText	barva textu v okně

Například následující ukázka nastavuje barvu popředí a pozadí odstavců na stejnou barvu, jakou má uživatelské okno:

```
p { color: WindowText: background-color: Window }
```

13.3.3.1 VÝHODY UŽIVATELŮ PRO BARVY ODKAZŮ V CSS3

Většina prohlížečů dovoluje uživateli vybrat defaultní barvy pro odkazy, které jsou v různých provedeních. Následující systémové barvy dovolují autorovi explicitně upravit s těmito preferencemi. Jména barev jsou seřazeny shodně se systémovými barvami CSS2, které jsou uvedeny výše.

ActiveHyperlink	pozadí aktivního odkazu
ActiveHyperlinkText	text aktivního odkazu
HoverHyperlink	pozadí odkazu po najetí myši
HoverHyperlinkText	text tohoto odkazu
Hyperlink	pozadí odkazu
HyperlinkText	text odkazu
VisitedHyperlink	pozadí navštíveného odkazu
VisitedHyperlinkText	text navštíveného odkazu

Tak například k nastavení všech odkazů na jejich defaultní barvu navštívených a nenavštívených barev, napišme toto:

```
:link {
  color: HyperlinkText;
  backround-color: Hyperlink
}
:visited {
```

```
  color: VisitedHyperlinkText;
  background-color: VisitedHyperlink
}
```

13.3.3.2 SYSTÉMOVÁ BARVA flavor

Zdůrazněná barva přizpůsobuje uživatelské rozhraní prohlížeči. Prohlížeč může defaultní *flavor* barvu použít přímo díky mechanismu, který používá uživatel. Ale neočekává se, že tato hodnota bude mít nějaký velký smysl na všech platformách a strojích, takže toto berte jen jako doplňující informaci.

```
:focus { outline: 1px solid flavor }
/* nastavuje obrys okolo aktuálně zaměřeného prvku */
```

Ukázka definicí barev pomocí kaskádních stylů pro HTML 4.

```
body {
      color: black;
      background: white;
      color-profile: sRGB;
      rendering-intent: auto
}

/* tradiční barva odkazů pro běžné počítače a prohlížeče */
:link { color: blue; }
:visited { color: purple; }

/* uživatelský výběr odkazů */
:link {
  color: HyperlinkText;
  background-color: Hyperlink;  }
:visited {
```

```
  color: VisitedHyperlinkText;
  background-color: VisitedHyperlink
}
:link:hover,:visited:hover {
  color: HoverHyperlinkText;
  background-color: HoverHyperlink
}
:link:active,:visited:active {
  color: ActiveHyperlinkText;
  background-color: ActiveHyperlink
}

img.object {
      color-profile: auto;
      rendering-intent: auto;
}

/* defaultně zaměřený obrys */
:focus {
      outline: 1px dotted gray;
      outline: 1px solid flavor;
}
```

13.3.4 PROFILY

Každá specifikace, která používá module CSS3 - color musí definovat podskupinu CSS3 rysů barev, ty dovolují a vylučují a popisují lokální význam všech komponent této podskupiny.

14 CSS3 MODUL BACKGROUND (POZADÍ)

Pro styl pozadí prvků slouží skupina vlastností modulu background. Pozadí je zobrazeno na celé ploše oblastí obsahu, výplně a rámečku. Pokud není v CSS specifikováno, stly pozadí dokumentu závisí na prohlížeči.

14.1 VLASTNOST background-color

- Tato vlastnost nastavuje barvu pozadí.
- Lze ji aplikovat na všechny elementy.
- Vlastnost není dědičná.
- Hodnotou je buď *color* v jednom ze dříve uvedených formátů, či hodnota *transparent* průhledné pozadí, dokud prvkům pozadí neurčíme, budou průhledné a pod nimi bude prosvítat pozadí jejich rodičovského prvku (případně i obsah prvků, které překrývají)..
- Implicitní hodnota: *transparent*

```
h1 { background-color: #F00 }
```

14.1.1 VLASTNOST background-image

Tato vlastnost vkládá do pozadí elementu obrázek.

- Lze ji aplikovat na všechny elementy.
- Vlastnost není dědičná.
- Hodnotou je URI cesta k souboru, *none* tedy bez obrázku nebo nově *inherit* – tedy nějaký poděděný obrázek
- Implicitní hodnota: *none*.

```
body { background-image: url("marble.gif") }
p { background-image: none }
```

14.1.2 VLASTNOST background-repeat

Obrázek může být na pozadí umístěn pouze jednou, nebo se může dlaždicově opakovat ve svislém, vodorovném či obou směrech. V místech, kde obrázek není a pod jeho průhlednými oblastmi je zobrazena plocha daná hodnotou *background-color* (případně nic, pokud je hodnota *transparent.*

- Lze ji aplikovat na všechny elementy. Má však smysl definovat ji pouze u elementů, u kterých je nastavena vlastnost *background-image* jinak než na *none*.
- Vlastnost není dědičná.
- Hodnoty: *repeat* (opakuje se po obou osách) *repeat-x* (opakuje se pouze vodorovně) *repeat-y* (opakuje se pouze svisle) *no-repeat* (nedochází k opakovanému vkládání).
- Implicitní hodnota: *repeat*

```
body { background: white url("vlnovka.gif");
 background-repeat: repeat-y;
 background-position: center; }
```

Pomocí tohoto kódu docílíte toho, že z jednoho malého obrázku, kde je vlnovka, uděláte, přes celou stránku dokumentu pěkného hada.

14.1.3 VLASTNOST background-attachment

- Vlastnost určuje zda dochází k pohybu pozadí s elementem nebo zda je pozadí fixováno a pohybuje se pouze element po pozadí.
- Lze ji aplikovat na všechny elementy. Má však smysl definovat ji pouze u elementů, u kterých je nastavena vlastnost *background-image* jinak než na *none*.
- Vlastnost není dědičná.

- Hodnoty: *scroll* (Pozadí se pohybuje společně s elementem) *fixed* (pozadí je fixováno a element se pohybuje po něm).
- Implicitní hodnota: *scroll*

```
body {
  background: red url("vlnovka.gif");
  background-repeat: repeat-y;
  background-attachment: fixed;
}
```

14.1.4 VLASTNOST background-position

- Vlastnost určuje výchozí pozici pro vkládání obrázku do pozadí (udává se X a Y pozice)
- Lze ji aplikovat na blokové a nahrazované elementy. Má však smysl definovat ji pouze u elementů, u kterých je nastavena vlastnost *background-image* jinak než na *none*.
- Vlastnost není dědičná.
- Hodnoty: délkové jednotky (X Y) procenta (vztahují se k velikosti vlastního elementu) pro X pozici *left*, *center*, *right* pro Y pozici *top*, *center*, *bottom*
- Implicitní hodnota: *0% 0%"*tedy *left top*

Klíčová slova se nesmí kombinovat s procenty a naopak.

```
body { background: url("banner.jpeg") right top }     /* 100% 0% */
body { background: url("banner.jpeg") top center }    /* 50% 0% */
body { background: url("banner.jpeg") center }  /* 50% 50% */
body { background: url("banner.jpeg") bottom }  /* 50% 100% */
```

14.1.5 VLASTNOST background-clip

- Tato vlastnost určuje, jestli bude pozadí dokumentu rozšířeno i za okraje dokumentu, či nikoliv
- Lze ji aplikovat na blokové a nahrazované elementy
- Vlastnost není dědičná
- Hodnoty: *border* – pozadí není roztáhnuto za okraje, naopak, při použití druhé hodnoty *padding* – pozadí je roztáhnuto a je transparentní.
- Implicitní hodnota: *border*

14.1.6 VLASTNOST background-origin

- Tato vlastnost určuje, jak bude vlastnost *background-position* vypočítána
- Hodnoty: *border, padding, content*
- Lze ji aplikovat na blokové a nahrazované elementy
- Vlastnost není dědičná

14.1.7 VLASTNOST background-size

Používá se, je-li použita vlastnost *background-image* pak použitý obrázek buď rozšiřuje nebo zužuje.

- Hodnoty: procenta, čísla, či *auto.*
- Lze ji aplikovat na blokové a nahrazované elementy
- Vlastnost není dědičná
- Implicitní hodnota: *auto*

Následující příklad roztáhne obrázek na pozadí tak, aby byl podkladem pro celý obsah:

```
div { background-image: url(plasma.png);
      background-size: 100%;
      background-origin: content}
```

14.1.8 VLASTNOST background-quantity

Používá se, jestliže je použita vlastnost *background-image,* pak touto vlastností určujeme, kolikrát se tento obrázek bude opakovat

- Hodnoty: *infinite* – nastavuje počet opakování na nekonečno. A pak celá čísla určují počet opakování.
- Vlastnost není dědičná
- Lze ji aplikovat na blokové a nahrazované elementy

```
BODY {
      background-image: url(rozum.gif);
      background-repeat: repeat-x;
      background-quantity: 3;
      }
```

Tento příklad říká prohlížeči, že obrázek, který tvoří pozadí bude opakován vertikálně pouze třikrát.

14.1.9 VLASTNOST background-spacing

Pomocí této vlastnosti můžeme velikost mezer mezi obrázky, které se opakují a tvoří tak pozadí našeho dokumentu

- Hodnota je pouze jediná a to nastavení vzdálenosti (*length)*
- Vlastnost není dědičná
- Lze ji aplikovat na blokové a nahrazované elementy

Formátovací model těchto výpočtů je: levý horní roh obrázku = *background-origin* + *background-position* + *background-spacing*

```
BODY {
        background-image: url(decko.gif);
        background-repeat: repeat-x;
        background-quantity: 5;
        background-spacing: 10px 5px;
      }
```

14.1.10 SOUHRNÁ VLASTNOST background

Pomocí této vlastnosti můžeme opět nastavovat hodnoty všem předešlým vlastnostem, týkajících se pozadí. Použití je obdobné, jako u ostatních souhrnných vlastností.

```
body { background: red }
p { background: url("chess.png") gray 50% repeat fixed }
```

15 CSS3 MODUL FONTS (PÍSMO)

Všechny vlastnosti které se týkají vzhledu písma jsou v CSS3 zahrnuty do jednoho modulu. A patří sem i vlastnosti definované jak v levelu1, tak v levelu 2..

15.1 VLASTNOST PÍSMA font-family

Tato vlastnost specifikuje a seřazuje podle důležitosti, seznam názvů rodin písem, tedy seznam písem která se mají použít pro zobrazení textu.

- Lze ji aplikovat na všechny elementy a na generovaný obsah a na vizuální média
- Tato vlastnost je dědičná

Hodnotou této vlastnosti je seznam názvů rodin písma, oddělený čárkami. Názvem rodiny je buďto konkrétní písmo používané v klientu (názvy obsahující mezeru musí být uzavřeny v uvozovkách), nebo obecná rodina. CSS používá tyto definované styly `serif`, `sans-serif`, `monospace`, `fantasy` a `cursive`. Definované styly jsou klíčová slova a proto nemusí být uzavřeny v uvozovkách.

generické jméno	popis
serif	patkové písmo
sans-serif	bezpatkové písmo
cursive	*skloněné písmo - kurzíva*
fantasy	OZDOBNÉ PÍSMO
monospace	neproporcionální písmo

```
body{ font-family: „Arial CE“, „Helvetica CE“, Arial, sans-serif }
p { font-family: ´Times CE´, ´Times New Roman´, serif }
code { font-family: monospace }
```

Pozor na to, že prohlížeč opera nerespektuje písmo nastavené uživatelem a použije písmo jiné.

15.1.1 STYL FONTU, VLASTNOSTI: font-style, font-variant, font-weight a font-stretch

Tyto vlastnosti lze aplikovat na všechny prvky, generovaný obsah a vizuální média a jsou dědičné.

Vlastnost *font-style* má hodnoty *normal*, *italic* a *obligue*. Výchozí hodnota *normal* nastaví základní (neskloněné) písmo, hodnoty *italic* a *obligue* písmo skloněné. S hodnotou *obligue* se použije písmo označené klientem jako *obligue* (obvykle písma s přídomkem *Obligue*, *Slanted* či *Incline*). S hodnotou *italic* se použije písmo označené jako *italic* (obvykle písma s přídomkem Italic, Cursive či Kurziv), pokud není dostupné, použije písmo označené jako *obligue*.

```
p.pozn { font-style: italic }
em {
  font-style: normal;
  font-family: cursive;
}
```

Internet Explorer (IE) 4 a 5 místo kapitálek uměle vytvoří jen verzálky (tj. totéž jako `text-transform: uppercase`). CSS to sice povoluje, ale kapitálky to nejsou.

Vlastnost font variant nabývá hodnot *normal* a *small-caps*. Přepíná mezi základním písmem a kapitálkami, což je styl písma, kde jsou malé znaky vytvořeny zmenšenými velkými znaky, anglicky small caps. Výchozí hodnota *normal* nastaví základní písmo, s hodnotou *small-caps* se text zobrazí v kapitálkách. Kapitálky může klient vytvořit uměli, případně místo nich použít pouze verzálky.

```
h1 { font-variant: small-caps }
```

Vlastnost *font-weight* může mít hodnoty: *normal*, *bold*, *bolder*, *lighter*, 100, 200, 300, 400, 500, 600, 700, 800 a 900. Tato vlastnost nastavuje tučnost písma. Hodnoty 100 až 900 tvoří posloupnost od nejslabšího po nejsilnější písmo. Následující hodnotě musí odpovídat písmo alespoň stejně silné jako hodnotě předcházející. Výchozí hodnota *normal* odpovídá hodnotě 400, hodnota *bold* je totéž jako 700.

Hodnotu *bolder* nepodporuje Netscape Navigator (NN).

Prohlížeč setřídí všechny dostupné řezy použitých rodin písma podle jejich síly a přiřadí jim hodnoty 100 až 900, např. v pořadí Light, Book (obvykle 400), Medium (obvykle 500), Bold (700), Heavy, ExtraBlack. Pokud nějaká hodnota zůstane nepřiřazena, přiřadí se jí nejbližší vhodné písmo. Hodnoty *bolder* a *lighter* použijí nejbližší písmo, které je silnější, respektive slabší než písmo rodičovského prvku. Pokud již žádné takové není k dispozici, písmo se nezmění a prvku se pouze nastaví následující, resp. předchozí hodnota v posloupnosti (pokud ještě není použita krajní hodnota 900 nebo 100).

```
strong { font-weight: bold }  /* 700 */
p { font-weight: 400 }        /*normal */
p span { font-weight: lighter }
```

Vlastnost *font-stretch* může mít hodnoty *normal*, *wider*, *narrower*, *ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded* a *ultra-expanded*. Tato vlastnost volí normální, úzký nebo rozšířený vzhled písma na rozdíl od *font-family*. Absolutní hodnoty klíčových slov jsou seřazení od nejužšího po nejširší vzhled písma:

1. ultra-condensed (středně úzký)
2. extra-condensed (velmi úzký)
3. condensed (úzký)
4. semi-condensed (středně úzký)

5. normal (normální)
6. semi-expanded (středně široký)
7. expanded (rozšířený)
8. extra-expanded (velmi široký)
9. ultra-expanded (málo široký)

Relativní hodnota wider nastavuje hodnotu na následující rozšířenou hodnotu nad zděděnou hodnotou a analogicky relativní hodnota *narrower* nastavuje hodnotu na nejbližší užší.

15.1.2 VELIKOST FONTU, VLASTNOST: font-size

- Tato vlastnost lze aplikovat na všechny prvky, generovaný obsah a vizuální média
- Je dědičná

Vlastnost font-size definuje velikost písma. Tato velikost odpovídá typografickému čtverčíku, některé znaky jej mohou přesahovat. Může mít hodnoty *absolute-size, relative-size, length* a *percentage.*

Absolut-size, čili absolutní hodnota velikosti je definována pomocí klíčových slov, které jsou: *xx-small, x-small, small, medium, large, x-large* a *xx-large.*

Za to, *relative-size*, čili relativní hodnota velikosti je definována klíčovými slovy *larger* a *smaller*, díky kterým můžeme použít písmo větší, či menší. Tak například, pokud rodičovský prvek používá velikost písma *small*, s hodnotou *larger* by se měla použít velikost *medium*. Pokud velikost písma rodičovského prvku není blízko některému z klíčových slov, prohlížeč může velikost nastavit mezi hodnotami z tabulky, nebo vybrat hodnotu nejbližší.

```
p { font-size: normal }
blockquote { font-size: larger }
em { font-size: 150% }
```

```
em { font-size: 1.5em }
```

W3C doporučuje používat především absolutní velikost písma, zadané klíčovými slovy. Specifikace CSS1 určovala koeficient 1,5 pro převod mezi jednotlivými stupni, ten však byl příliš velký. CSS2 jej snížila na 1,2, to však stále působilo příliš velké skoky u menších velikostí. CSS 2.1 nakonec uvádí převodní tabulku, která používá různé koeficienty pro různé velikosti, aby rozdíly mezi nimi byly přiměřené.

Ale pozor na to, že IE4-5 používá základní velikost písma (velikost nestylovaného textu) pro *small* místo *medium*. Písmo o velikosti *medium* (a vůbec písmo zadané absolutní velikostí) je tak v IE vždy větší než v jiných prohlížečích.

Absolutní velikosti	**xx-small**	**x-small**	**small**	**medium**	**large**	**x-large**	**xx-large**	
Koeficient	3/5	¾	8/9	**1**	6/5	3/2	2	3
Odpovídající nadpisy HTML	H6		H5	**H4**	H3	H2	H1	
Odpovídající velikosti značky font	1		2	**3**	4	5	6	7

15.1.3 SDRUŽENÁ VLASTNOST FONT

- Tato vlastnost slouží k nastavení všech vlastností písma současně.
- Hodnoty, kterých může nabývat jsou: *font-style, font-variant, font-weight, font-size, line-height, font-family, caption, icon, menu, message-box, small-caption* a *status-bar*
- Je dědičná
- Dá se použít na všechny prvky a vizuální média

Vlastnost *font* má dvě možné syntaxe. První možností je použití *font* jako sdružené vlastnosti. Lze díky ní definovat všechny vlastnosti písma současně, ve tvaru, který je velmi podobný předpisům písma používaným v typografii. U jednotlivých podvlastností je třeba dodržet uvedené pořadí:

- nejprve se uvádí hodnoty vlastností *font-style, font-variant* a *font-weight* v libovolné pořadí, oddělené mezerou. S tím, že kterákoli z nich (nebo žádná) nemusí být uvedena. Pokud je zde uvedena alespoň jedna hodnota *normal*, použije se pro všechny chybějící vlastnosti.
- za nimi následuje hodnota *font-size*. Tato hodnota je zde povinná.
- volitelně může být hodnota *font-size* následována lomítkem "/" a hodnotou *line-height* (bez mezer). Pokud není tato hodnota uvedena, nesmí zde být ani lomítko.
- nakonec se povinně uvádí hodnota *font-family* (seznam rodin písma).

Sdružená vlastnost *font* tedy musí minimálně obsahovat dvojici *font-size* a *font-family*, ostatní vlastnosti jsou nepovinné. Pokud je vlastnost *font* použita, všechny dílčí vlastnosti, které může obsahovat, jsou nejprve nastaveny na svou výchozí hodnotu a následně jsou předefinovány ty podvlastnosti, které jsou zde uvedeny. Proto, když vynecháme některou z nich, musíme vždy počítat s tím, že bude mít výchozí hodnotu, i když jsme ji předtím definovali. Např.:

```
p { font-weight: bold }
p { font: 12pt/1.2 "Helvetica CE",Arial,sans-serif }
```

Vlastnost *font* zde přiřadí chybějícím podvlastnostem *font-style, font-variant, a font-weight* jejich výchozí hodnoty, což u *font-weight* znamená hodnotu *normal*. Písmo tedy nebude tučné.

```
body { font: normal serif }
```

```
/* chybějící vlastnosti font-style, font-variant, font-weight a
line-height budou mít přiřazenu výchozí hodnotu */
em { font: normal bold smaller inherit }
/* hodnota normal se použije pro font-style i pro font-variant; písmo
bude tučné, menší velikosti, line.height bude mít výchozí hodnotu a
rodina písma se zdědí z rodičovského prvku (inherit) */
p.pozn {font: normal small-caps bolder 12pt/14pt serif }
```

Druhou možnou syntaxí vlastnosti *font* je její použití jakožto standardní vlastnosti. Její hodnotou pak může být jedno z klíčových slov, odkazujících se na písma použitá v systému uživatele:

- *caption* – písmo použité pro ovládací prvky (tlačítka, seznamy atd.)
- *icon* – písmo použité pro popisy ikon
- *menu* – písmo použité pro nabídky
- *message-box* – písmo použité v dialogových rámečcích
- *small-caption* – písmo pro zobrazení malých ovládacích prvků
- *status-bar* – písmo použité ve stavových rámečcích oken

Přiřazení klíčového slova nastaví všechny dílčí vlastnosti podle charakteristiky použitého písma, je ale možné je následně upravit. Pokud v systému není písmo uvedeného typu, klient by je měl nahradit písmem podobným, nebo alespoň svým výchozím písmem.

15.1.4 FONT DECORATION

Vlastnost font decoration je vlastnost CSS3 a popisuje okrášlení, která ovlivňují zobrazení fontů. Přestože je podobná vlastnosti *text-decoration*, je rozdílná v tom, že je úžeji spojena s fontem a tak ho může lépe ovlivňovat

15.1.4.1 VLASTNOST font-effect

- Tato vlastnost ovládacích prvků speciálně ovlivňuje znaky
- Hodnoty jsou *none, emboss, engrave* a *outline*
- Dá se aplikovat na všechny prvky, generovaný seznam a vizuální média
- Je dědičná

Význam hodnot je následující:

none – žádný efekt nebude použit

engrave – znaky budou vypadat tak, jako kdyby byly do stránky vyryty (tento efekt je také občas nazýván jako sunken text)

emboss –znamená naopak, že znaky budou jakoby vystouplé ze stránky

outline – bude vykreslen pouze obrys znaků

Obrázek č.4. Ukázka hodnot engrave, emboss a outline efektu

15.1.4.2 VLASTNOST font-smooth

Tato vlastnost dovoluje autorovi kontrolu nad použitím potlačených roztřepených okrajů, poté co jsou použity.

- Hodnoty: *auto, never, always, absolute-size, length*
- Dá se použít na všechny prvky generovaný obsah a vizuální média

Význam hodnot je následující:

auto – defaultně vyhlazuje text podle systému

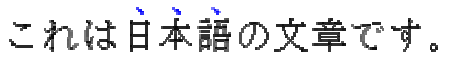
never – nevyhlazuje text

always – vyhlazuje text vždy v jakékoli velikosti fontu

absolute size - a s tím spolená *length* nastavuje vyhlazení textu na zobrazený text, tehdy, když ja aktuální hodnota vlastnosti *font-size*, větší nebo rovna specifické vlastnosti.

15.1.4.3 VLASTNOSTI font-emphasize-style, font-emphasize-position a SDRUŽENÁ VLASTNOST font-emphasize

Vlastnost *font-emphasize-style* určuje styl pro zvýraznění formátování (něco jako diakritická znaménka u nás), aplikované na text. Její hodnoty mohou být *none* (žádné), *accent* (důraz), *dot* (tečka), *circle* (kroužek), *disc* (kolo). Východoasijské dokumenty používají následující symboly nad každým znakem k zvýraznění běhu textu, tak jak vidíme na obrázku číslo 5.:



Obrázek č.5 – ukázka použití vlastnosti *font-emphasize-style* (důraz je vždy označen čárkou nad znakem)

Vlastnost *font-emphasize-position* má hodnoty *before* a *after*. Touto vlastností určujeme pozici diakritických znamének. Hodnoty *before* (před) a *after* (za) jsou chápány relativně k základní lince. Např. v japonštině preferují pozici *before*, viz obrázek č.5. Ale naopak v čínštině dávají přednost hodnotě *after*, viz obrázek č.6:

Obrázek č.6 – použití hodnoty *after* (důraz je vždy označen čárkou pod znakem)

Poslední vlastnost vztahující se k důrazu je sdružená vlastnost *font-emphasize*, pomocí níž můžeme nastavit všechny předchozí vlastnosti pro důraz, společně.

16 CSS3 MODUL WEB FONT (PÍSMO PRO WEB)

Norma CSS2 přinesla nové možnosti pro práci s písmy. Jejich největším nedostatkem byla práce s konkrétními fonty. Pokud totiž uživatel nevlastnil požadovaný font písma neexistoval způsob jak tento font nahradit či dokonce tento font uživateli zprostředkovat. Norma CSS2 proto nabízí čtyři různé způsoby jak se dopracovat k požadovanému fontu či vykreslení písma. Prvním z těchto možností je, že uživatel požadované písmo vlastní (původní metoda CSS1), pak je vše v pořádku a požadované písmo se použije. Další tři nové možnosti však řeší případ kdy uživatel požadované písmo nevlastní. V takovém případě je možno použít metodu "inteligentního hledání". To znamená, že je hledáno písmo, které je co nejvíce podobné požadovanému fontu, jak skutečným vzhledem, tak způsobem vykreslování. Třetí metoda, "metoda syntézy" je schopna navíc toto nejpodobnější písmo ještě upravit, aby výsledné zobrazení co nejvíce odpovídalo vlastnostem původně požadovaného fontu. Poslední čtvrtá metoda umožňuje uživateli načíst požadované písmo přímo z Internetu. CSS2 navíc umožňuje tyto tři metody kombinovat (tzv. progresivní vykreslování). To znamená že v případě nenalezení požadovaného fontu je automaticky hledán nejpodobnější font a zároveň je požadovaný font načítán z Internetu. Jakmile je podobný font nalezen a případně i upraven je požadovaný text přechodně znázorněn náhradním fontem. V okamžiku kdy je pak načten požadovaný font dojde k automatickému překreslení pravým fontem.

Abychom mohly těchto nových vlastností využít musíme pro každé písmo nadefinovat nový specifický parametr *"@font-face"*. Pomocí tohoto parametru definujeme vlastnosti fontu dle kterých je při "inteligentním hledání" vybírán podobný font popřípadě, při "metodě syntézy" i upravován náhradní font. V tomto parametru taktéž definujeme umístění tohoto fontu na Internetu pro načtení písma. Parametr má syntaxi:

```
@font-face { <font-description> }
```

Kde `<font-description >`má takovouto formu:

```
descriptor: hodnota;
descriptor: hodnota;
[...]
descriptor: hodnota;
```

Tak například v tomto příkladě je definován font Robson Celtic a je referencován ve stylovém předpisu v HTML dokumentu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Font test</TITLE>
    <STYLE TYPE="text/css" MEDIA="screen, print">
      @font-face {
        font-family: "Robson Celtic";
        src: url("http://site/fonts/rob-celt")
      }
      H1 { font-family: "Robson Celtic", serif }
    </STYLE>
  </HEAD>
  <BODY>
   <H1> Tento nadpis bude vyobrazen pomocí fontu Robson Celtic </H1>
  </BODY>
</HTML>
```

16.1 KLÍČOVÁ SLOVA (DESKRIPTORY) PRO FONT: font-family, font-style, font-variant, font-weight, font-stretch, font-size

Všechny tyto vlastnosti mají stejná jména jako klasické v modulu FONT a mají i stejné hodnoty. Ale zde tyto vlastnosti nenastavují existující font, ale určují vlastnosti hledaného fontu. Takže zde nebudu znovu uvádět jejich popis, ten naleznete viz CSS3 modul Font.

16.1.1 DESKRIPTOR PRO KVALIFIKACI DAT FONTU: unicode-range

Vlastnost popisuje hodnoty variant Unicode, které má obsahovat písmo hledané pro shodu s původním fontem. Používá se v kontextu *@font-face.* Implicitní hodnota: *U+0 –7FFFFFFF*

16.1.2 DESKRIPTOR PRO ČÍSELNÉ HODNOTY: units-per-em

Vlastnost popisuje délkovou jednotku písma pro použití v dalších parametrech, konkrétně sděluje, kolik jednotek se vejde do čtverčíku em. Používá se v kontextu *@font-face* a hodnotou je číslice bez jednotek.

16.1.3 DESKRIPTOR PRO ODKAZY: src

Odkaz (či seznam odkazů) na soubor obsahující definici požadovaného písma.
Vlastnost lze použít v kontextu *@font-face*
Hodnoty: *url* (odkaz na soubor písma), *format(<string>)* (specifikace druhu písem a platforem,)

- *truedoc-prf* -: True Doc Portable Font Resource *.prf
- *embedded-opentype*: Embedded Open Type *.eot
- *type-1* PostScript Type 1 *.pfb *.pfa
- *truetype*: TrueType *.ttf
- *opentype*: OpenType, TrueType Open *.ttf
- *truetype-gx*: TrueType GX
- *speedo:* Speedo
- *intellifont*: IntelliFont

```
@font-face { font-family:”Robson-Celtic”;
src: url(/ttf/rob-cel) format(truetype)
url(/post/rob-cel) format(type-1)}
```

16.1.4 DESKRIPTORY PRO PÁROVÁNÍ: panose-1, stemv, stemh, slope, cap-height, x-height, ascent, descent

Vlastnost *panose-1,* popisuje požadovaný typ písma podle standardu Panose, dnes zřejmě nejpoužívanějšího standardu pro písma typu latinka. Hodnotou je číslo.

Vlastnost *stemv*, a *stemh,* popisují vertikální (*stemv*) a horizontální (*stemh*) šířku písma. Hodnotou je číslo.

Vlastnost *slope,* popisuje základní úhel sklonu písma od svislé osy. Hodnotou je číslo, vyjadřující úhel.

Vlastnost *cap-height*, popisuje výšku velkých písmen u původního písma. Hodnotou je číslo.

Vlastnost *x-height*, popisuje výšku malých písmen u původního písma. Hodnotou je číslo.

Vlastnost *ascent*, popisuje maximální výšku znaku bez diakritiky. Hodnotou je číslo.

Vlastnost *descent*, popisuje maximální výšku písma pod základnou, bez diakritiky. Některé jazyky totiž používají diakritiku i pod písmenem např. Arabština. Hodnotou je číslo.

16.1.5 DESKRIPTOR PRO SLUČOVÁNÍ: widths, bbox, definition-src

Vlastnost *widths*, určuje šířku znaků, hodnotou je seznam hodnoty *urange* oddělených čárkou, který je následován jednou nebo více číselnými hodnotami. Jestliže použijeme tento deskriptor, musíme použít i deskriptor *units-per-em,* viz. výše.

Vlastnost *bbox*, určuje maximální ohraničení boxů, fontu. Hodnotou jsou čtyři čísla, v pořadí: levá, x-ová, levá y-ová, pravá x-ová a pravá y-ová, ohraničujícího boxu.

Vlastnost *definition-src,* může být buď uvnitř fontu jako popis ve formátovací sadě, nebo může být poskytnutý uvnitř odděleného fontu definice zdroje definovaného pomocí URI. Popis tohoto deskriptoru, však ještě W3C příliš nepřipravila a nedotáhla dokonce.

16.1.6 DESKRIPTORY PRO ZAROVNÁNÍ: baseline, centerline, mathline, topline

Deskriptor *baseline*, je pro použití pro všechny základny fontů. A jestliže je tento nastaven na jinou hodnotu, než implicitní (nula), musí být použit také deskriptor *units-per-em*.

Deskriptor *centerline,* tento je použit pro použití střední základny písma. Jestliže není nadefinována žádná hodnota, prohlížeč může použít různou heuristiku, tak jako střed úsečky výstupu a sestupu hodnoty.

Deskriptor *mathline*, se používá pro základnu u matematických výrazů.

A poslední vlastnost *topline,* se používá pro nejvyšší základnu písma.

Pro dobré porozumění výše uvedených vlastností, je potřeba znát typografické výrazy a základy profesionální sazby.

Máme následující seznam fontů (viz tabulka):

Swiss 721 light	light & light italic
Swiss 721	roman, bold, italic, bold italic
Swiss 721 medium	medium & medium italic
Swiss 721 heavy	heavy & heavy italic
Swiss 721 black	black, black italic, & black #2
Swiss 721 Condensed	roman, bold, italic, bold italic
Swiss 721 Expanded	roman, bold, italic, bold italic

Následující popis fontu může být použit k tomu, aby byl možný ke stažení:

```
@font-face {
    font-family: "Swiss 721";
    src: url("swiss721lt.pfr"); /* Swiss 721 úzký */
    font-style: normal, italic;
```

```
    font-weight: 200;
}
@font-face {
    font-family: "Swiss 721";
    src: url("swiss721.pfr"); /* Klasické Swiss 721 */
}
@font-face {
    font-family: "Swiss 721";
    src: url("swiss721md.pfr"); /* Swiss 721 střední */
    font-style: normal, italic;
    font-weight: 500;
}
@font-face {
    font-family: "Swiss 721";
    src: url("swiss721hvy.pfr"); /* Swiss 721 silná */
    font-style: normal, italic;
    font-weight: 700;
}
@font-face {
    font-family: "Swiss 721";
    src: url("swiss721blk.pfr"); /* Swiss 721 černá */
    font-style: normal, italic;
    font-weight: 800,900; /* uvědomme si, že kurzíva, která by měla
                             být tučná 900px neexistuje*/
}
@font-face {
    font-family: "Swiss 721";
    src: url(swiss721.pfr); /* condensed Swiss 721 */
    font-stretch: condensed;
}
@font-face {
    font-family: "Swiss 721";
    src: url(swiss721.pfr); /* expanded Swiss 721 */
    font-stretch: expanded;
}
```

17 CSS3 MODUL TEXT

Text nese hlavní informační hodnotu dokumentů a obvykle tvoří naprostou většinu obsahu webových stránek. Definování jeho vzhledu je také jednou z nejvyužívanějších funkci CSS. Proto také CSS3 věnovalo textu samostatný modul s mnoha novými vlastnostmi.

17.1 NASTAVENÍ TOKU TEXTU, VLASTNOSTI writing- mode a direction

Vlastnost wtiting-mode určuje tok textu jedné řádky nebo celých odstavců.

- Hodnoty jsou*: lr-tb, rl-tb, tb-rl, tb-lr, bt-rl, bt-lr, lr, rl, tb*
- Je dědičná a dá se použít na všechny prvky a generovaný obsah a vizuální média

Hodnotami jsou počáteční písmena anglických slov a jejich kombinace, určující směr, tzn:

- *l* – left (levo)
- *r* – right (pravo)
- *t* – top (navrchu)
- *bottom* – (vespod)

Takže například kombinace *lr-tb*, znamená, že text bude zobrazen tak jak jsme u nás zvyklí, tedy, že jednotlivé řádky budou psány zleva doprava a celý text shora dolů. CSS3 myslelo hodně i na tvůrce z východu, jako je Japonsko a Čína a tak text může být zformátován například podle čínského vzoru, tedy zprava doleva a zdola nahoru.

Vlastnost *direction* má ten samý význam, ale je to pro určení směru textu, tehdy, když používáme unicode. Má pouze 2 hodnoty a to *ltr* (left to right = zleva doprava) a *rtl* (right to left = zprava doleva).

```
e { writing-mode: lr-tb; direction: rtl;}
```

17.1.1 SMĚR JEDNOTLIVÝCH ZNAKŮ A TOK TEXTU, VLASTNOSTI: glyph-orientation-vertical A glyph-orientation-vertical

Tyto vlastnosti jsou vlastně analogické k těm předchozím, je to určeno právě například pro japonské znaky apod. Obě mají dvě hodnoty a to *auto*, čili automatický nebo druhá je *angle*, čili úhel, který se udává v radiánech. Možnosti stupňů jsou 0, 90, 180 a 270 radiánů.

17.2 VLOŽENÍ A PŘEPSÁNÍ, VLASTNOSTI: unicode-bidi

Tato vlastnost dovoluje převzít kontrolu nad psaním v unicode algoritmech. Nabývá hodnot *normal*, *embed* a *bidi-override.*

17.3 VLASTNOST text-script

- mají hodnoty auto a script
- je dědičná a vztahuje se na všechny prvky a generovaný obsah

Hodnota *auto* znamená, že se použije první znak potomka, který má jednoznačný identifikátoru prvku.

17.4 ZAROVNÁNÍ A SESKUPOVÁNÍ TEXTU, VLASTNOST: text-align

- Vlastnost nastavuje odstavcové zarovnání textu (k levému či pravému okraji, na střed nebo do bloku).
- Lze aplikovat na blokové elementy.
- Vlastnost je dědičná.
- Hodnoty: *left* - text zarovnán k levému okraji, *center* - text zarovnán na střed *right* - text zarovnán k pravému okraji, *justify* - text je zarovnán do bloku pomocí zvětšení mezislovních mezer. A nové hodnoty CSS3: *start, end* - text

je zarovnán k začátku nebo konci vloženého řádku <*string*> – specifikuje řetězec, ve které buňce v řádce tabulky bude zarovnán

- Implicitní hodnota závisí na nastavení prohlížeče (většinou implicitně *left*)

```
div {color: #000; font: small-caps 15pt serif; word-spacing: 15px;
     letter-spacing: 3; text-decoration: blink; text-align: left}
p {font: 200 xx-large "Arial CE", sans-serif; word-spacing: normal;
     letter-spacing: 2mm; text-align: justify}
```

17.4.1 VLASTNOST text-justify

- Hodnoty jsou: *auto, interword, inter-ideograph, distribute, newspaper, inter-cluster* a *kashida*
- Používá se na prvky: *block-level* a *block-elements* a pro vizuální média

Toto je nová vlastnost CSS3, určuje jaký typ bude použit pro zarovnání na poslední řádky, kterým je nastavena vlastnost *text-align-last* a zároveň hodnota *justify* vlastnosti *text-align*. Používá se především na specifické jazyky jakými jsou japonština, čínština, hindština, arabština apod. Takže zde uvádím jen hodnoty které mohou nabývat a ti kteří budou tuto vlastnost potřebovat, mohou informace zjistit na internetové adrese w3.org.

17.4.2 VLASTNOST text-align-last

- hodnoty*: relative, start, end, center, justify, size*
- používá se na prvky: *block-level* a *block-elements* a pro vizuální média

Tato vlastnost popisuje, jak bude poslední řádka vloženého obsahu v bloku, zarovnána. Také se používá pro samostatné řádky bloku, jestliže obsahují jednu řádku, řádka, která je předcházena tagem *< br >,* prvků v obsahu XHTML prvků.
Význam hodnot:

relative – jestliže vlastnost *text-align* je nastavena na hodnotu *justify*, tak poslední řádka bude zarovnána na začátek řádky. Když má jinou hodnotu, bude tato poslední řádka zarovnána stejně, tak jako je hodnota *text-align.*
start, end, center – zarovnání podle příslušné hodnoty.
justify – poslední řádka bude zarovnána podle hodnoty vlastnosti text-justify.
size – obsah řádky je spojený s řádkou, ke které se hodí, na řádce musí být jen jeden druh fontu.Typické použití pro tuto hodnotu je použití pro jednotlivé řádky. Tato vlastnost může také měnit počet řádek v bloku.

Následující příklad ukazuje použití na vlastnosti, které zarovnány hodnotou *distribute*. Tento styl je typický pro východo-asijskou typografii.

```
p.distributealllines {
      text-align: justify;
      text-justify: distribute;
      text-align-last: justify }
```

17.5 MINIMUM A MAXIMUM PRO VELIKOST FONTŮ, VLASTNOSTI: min-font-size a max-font-size

Obě tyto vlastnosti jsou používané pouze s předchozí vlastností *text-align-last*, při použití hodnoty *size*.

Mají hodnoty jen *font-size* a *auto*, kde *font-size* nedovoluje nastavit velikost písma u poslední řádky, menší/větší, než jaká je nastavena pomocí vlastnosti *font-size* ani *min-font-size*. A hodnota *auto* představuje defaultní nastavení písma prohlížečem, například pro latinku je to velikost 8px.

17.6 VLASTNOST: text-justify-trim

- Hodnoty jsou*: none, punctuation* a *punctuation-and-kana,* výchozí hodnota je *punctuation.*
- Používá se na prvky: *block-level* a *block-elements* a pro vizuální média

Pomocí této vlastnosti můžeme ořezávat mezery mezi slovy. Ale opět jen pro jazyky pro nás vzdálené, jako je japonština a pro jejich široké znaky.

Při použití hodnoty *none* se žádné oříznutí neprovede, (viz obr. č.7).

Obrázek č.7 – výsledek použití hodnoty *none* (bez oříznutí)

Při použití implicitní hodnoty *punctuation* se mezera ořízne jen za posledním znakem slova (viz obr. č.8).

Obrázek č.8 – výsledek použití hodnoty *punctuation*

A pomocí hodnoty *punctuation-and-kana* se pomyslná buňka ořízne u všech znaků (viz obr. č.9).

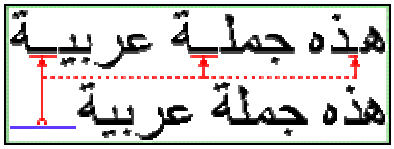
Obrázek č.9 – výsledek použití hodnoty *punctuation-and-kana*

17.7 VLASTNOST text-kashida-space

Toto je další vlastnost, kterou asi moc neoceníme. Má jedinou hodnotu a to procentuální, při čemž základní je 0%.

Kashida je typografický efekt, který je používán v arabském systému písma, který dovoluje roztáhnout znaky v některých pečlivě vybraných bodech, tak jako v následujícím příkladě, kde je první řádka oproti druhé roztáhnuta podle potřeby. Ale

roztažení není docíleno pomocí zvětšení mezer, ale roztáhnutí samotných znaků (viz obr. č.10).



Obrázek č. 10 – horní řádek ukazuje, jak byly znaky roztáhnuty oproti původním

17.8 ODSAZENÍ PRVNÍ ŘÁDKY: text-first-indent

- Hodnoty: *length* a procenta, kde *length* je pevná vzdálenost a procenta jsou procentuálním vyjádřením závislé na šířce bloku.

Určuje, jak nadpis odpovídá odsazení prvního řádku v bloku s větší přesností než vlastnost *line-box*.

17.9 ODSAZENÍ ŘÁDEK U CELÉHO ODSTAVCE: text-block-indent

Hodnoty jsou *none* a *auto* (znamená, že všechny řádky jsou odsazeny na minimální vzdálenost, která je nutná pro to, aby byla první řádka uchována na začátku určeného okraje řádky.

```
:root {text-block-indent: auto}
 p { text-indent: -3em }
```

17.10 ODSAZENÍ VŠECH ŘÁDEK: text-indent

- Vlastnost nastavuje velikost odstavcové odrážky (o kolik bude odsazen první řádek v odstavci).
- Lze aplikovat na blokové elementy.

- Je dědičná.
- Hodnoty: číselná hodnota s jednotkami udávající velikost odsazení, procentuální hodnota vztahující se k šířce blokového elementu.
- Implicitní hodnota: *0*
- Odsazení je aplikováno pouze na blokové elementy, které obsahují více než jeden řádek.

```
div {color: #000; font: small-caps 15pt serif; word-spacing: 15px;
      letter-spacing: 3; text-align: left; text-indent: 25}
p {font: 200 xx-large "Arial CE", sans-serif; word-spacing: normal;
      letter-spacing: 2mm; text-align: justify; text-indent: 15%}
```

17.11 ZALAMOVÁNÍ ŘÁDEK line-break

Hodnoty: *normal* – určuje normální zalamování a *strict* – nastavuje více omezující zalamování, které je standardizováno v Unicode Standard Annex. Což je jen pro zasvěcené a tak to zde dále rozebírat nebudu.

17.12 ZALAMOVÁNÍ SLOV: word-break-cjk, word-break-inside A SOUHRNÁ VLASTNOST word-break

Vlastnost *word-break-cjk*, pomáhá kontrolovat zalamování řádků z pohledu CJK. Možné vlastnosti jsou: *normal* – udržuje všechny skripty CJK pospolu; *break-all* – je stejné jako hodnota *normal*, ale pro skripty CJK, ostatní mohou být zalomeny kdekoliv a hodnota *keep-all* je stejná jako *normal*, pro skripty, které nejsou CJK.

Následující příklad ukazuje styl odstavce, ve kterém mohou být všechny skripty, které nejsou CJK, zalomeny kdekoliv.

```
p.anywordbreaks { word-break: break-all }
```

Vlastnost *word-break-inside* kontroluje dělení slov, uprostřed. Hodnoty mohou být *normal* – ta dovoluje, aby rozdělované slovo stálo v samostatné řádce. Zatímco hodnota *hyphenate* – umožňuje dělení slov jen na povoleném místě

Díky vlastnosti *word-break* můžeme nastavit hodnoty oběma předchozím vlastnostem a to tak, jako u předchozích shrnujících vlastností.

17.13 ROLOVÁNÍ TEXTU wrap-option

- Hodnoty: *wrap, no-wrap, soft-wrap* a *emergency*
- Základní hodnota je: *wrap*

Pomocí této vlastnosti lze určit, jestli náš text bude rolovatelný. Základní hodnota *wrap* nastavuje, že text je možné rolovat. Naopak vlastnost *no-wrap* znamená že text bude možné rolovat pouze v případě, že nebude možně číst celý text na obrazovce. Hodnota *soft-wrap* - text je rolován po posledním znaku který je před koncem-okraje řádky a kde jsou znaky explicitně specifikovány k uchování posunu řádků. A poslední možná hodnota *emergency* - Text je rolován jako pro hodnotu *wrap*, až na to, že algoritmus pro zalomení řádku dovoluje jako poslední instance volbu, která text roluje po posledním znaku, který je poslední před koncem okraje řádky odstavce.

Následující příklad ukazuje jak se budou chovat mezery v elementu *pre* a *p* vlastnost *no-wrap* v XHTML, a v generovaném obsahu.

```
pre { white-space: pre }
p { white-space: normal }
td[nowrap] { white-space: nowrap }
:before, :after {white-space: pre-line }
```

17.14 VLASTNOST text-overflow-mode

Tato vlastnost kontroluje přetečení textu v řádce. Hodnota: *clip* – sepne text s předchozím obsahem. *Ellipsis* – nastaví odrážku na takových místec, kde nastane přetečení textu a vkládají se nakonec, za posledním písmenem odstavce, na které se to

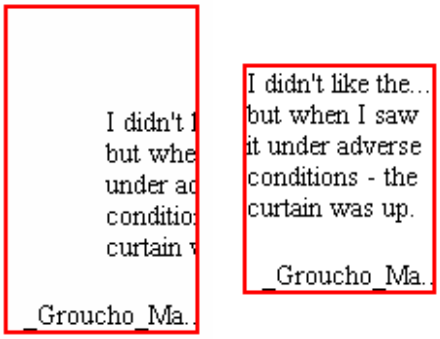
hodí. A *ellipsis-word* – je stejná jako předchozí hodnota, s tím,že zarážka se vkládá za poslední slovo.

Tento text je zde uveden pro následující příklad pro přetečení textu.

```
<blockquote>
<p class="sentence"><span class="nowrap">I didn't like the
play,</span> but then I saw
it under adverse conditions - the curtain was up.
<div class="attributed-to">_Groucho_Marx_</div>
</p>
</blockquote>
```

A zde je ukázka kontroly přetečení textu.

```
blockquote          { width:100px; border: thin solid red; overflow:
hidden;
                      text-overflow-mode:ellipsis;font-size:14px }
span.nowrap        { white-space : nowrap; }
div.attributed-to { position: relative;left:8px }
```



Obrázek č.11,12: přetečení textu a druhý ukazuje, jak tomuto lze zabránit pomocí předchozího kódu.

17.15 SLOVNÍ A MEZISLOVNÍ MEZERY, VLASTNOST letter-spacing

- Vlastnost nastavuje velikost mezery mezi jednotlivými písmeny ve slově.
- Lze aplikovat na všechny elementy.
- Vlastnost je dědičná.
- Hodnoty: *normal* - číselná hodnota s jednotkami (implicitně pixel) a hodnota *length* – ukazuje mezery přidané navíc k normálním mezerám mezi grafické shluky
- Implicitní hodnota: *normal*
- Při použití této vlastnosti musíte počítat i se zvětšením mezislovních mezer

Níže uvedený příklad nastavuje mezery mezi písmeny 0.1 pixelu

```
blockquote { letter-spacing: 0.1em }
```

A v tomto druhém případě, není od prohlížeče požadována žádná větší mezera.

```
blockquote { letter-spacing: 0cm } /* To je to samé jako '0' */
```

17.15.1 VLASTNOST word-spacing

- Vlastnost nastavuje velikost mezislovní mezery v textu.
- Lze aplikovat na všechny elementy.
- Vlastnost je dědičná.
- Hodnoty: *normal* - číselná hodnota s jednotkami (implicitně pixel) a *length* - ukazuje mezery přidané navíc k normálním mezerám mezi slovy
- Implicitní hodnota: *normal* (hodnota *normal* v tomto případě odpovídá číselnému vyjádření 1en – tedy velikosti odpovídající velikosti písmena “n”)

Tento příklad nastavuje velikost mezislovních mezer na 1 čtverčík:

```
h1 { word-spacing: 1em }
```

17.16 VLASTNOST text-autospace

Jestliže tok textu je tvořen ozdobnými znaky a mezery jsou vkládány podle znaků. Tato vlastnost ovládá vytváření mezer při překladu textu. Přidává mezery, které neodpovídají šíři vložených znaků, ale místo toho mezeru o šířce ozdobného znaku.

- Hodnoty: *none, ideograph-numeric, ideograph-alpha, ideograph-space, ideograph-parenthesis*

None – není vkládána žádná speciální mezera navíc.
Ideograph-numeric – vytváří mezeru mezi čísly a ozdobným textem.
Ideograph-alpha – vytváří mezeru mezi normálními znaky latinky a ozdobnými znaky, například řečtina apod.
Ideograph-space – rozšiřuje mezeru vzájemně mezi ozdobnými znaky
Ideograph-parenthesis – rozšiřuje mezeru mezi normálními znaky

Tento příklad ukazuje nastavení mezer mezi číslicemi a ozdobnými znaky.

```
span.autospace { text-autospace:none; }
<span class="autospace">[ideographs]1997[ideographs]</span>
```

17.17 VLASTNOSTI PRO DEKORACI TEXTU text-underline-style, text-line-through-style a text-overline-style

Tyto vlastnosti specifikují styl podtržení a přeškrtnutí. Možné hodnoty jsou: *none* – což znamená, že žádná čára nebude použita. *Solid* – bude použita plná čára. Při použití hodnoty *double* – bude čára dvojitá, při *dotted* – bude čára tečkovaná. *Dashed* – znamená, že styl čáry bude čárkovaný. Při *dot-dash* – bude čára čerchovaná. U použití hodnoty *dot-dot-dash* – je dvojitě čerchovaná. A u hodnoty *wave* – je čára vlnitá.

None - solid – double – dotted – dashed – dot-dash – dot-dot-dash - wave

Obrázek č. 13 ukazuje názorné příklady výše uvedených vlastností

17.17.1 ŠÍŘE ČÁRY, VLASTNOSTI: text-underline-width, text-line-throught-width a text-overline-width

Tyto tři vlastnosti určují šíři čáry pro podtržení a přeškrtnutí. Hodnoty mohou být: *auto* – šíře podtržení je nastavena automaticky podle prohlížeče. *normal* – šířka čáry bude odpovídající k použitému fontu. <*number*> - tloušťka čáry bude vypočtenou hodnotou z elementu *font-size.* <*length*> - šířka bude podle toho jakou velikost nastavíme této hodnotě. <*percentage*> *thin* – je obdobná jako vlastnost number, ale velikost je nastavena procentuálně. *medium* – šíře čáry musí být širší anebo stejně široká jako hodnota *thin.* A poslední hodnota *thick* musí být širší nebo stejně široká jako hodnota *medium.*

17.17.2 BARVA ČÁRY, VLASTNOSTI: text-underline-color, text-line-through-color a text-overline-color

Tyto vlastnosti nastavují barvu čáry. Hodnoty jsou pouze dvě. Buď *auto* je vypočtenou hodnotou vlastnosti *color*, taková jako je nastavená, bude i zde u této hodnoty nastavena implicitně při použití hodnoty auto. A nebo hodnota *color* specifikuje jakoukoli barvu chceme použít.

17.17.3 VLASTNOSTI text-underline-mode, text-line-through-mode a text-overline-mode

Tyto vlastnosti upravují podtržení, či přeškrtnutí, tak že určují, zda budou podtrženy i mezery mezi slovy, či nikoliv. Hodnoty jsou: *continuous* – čára bude pokračovat. *skip-white-space* – mezery nebudou podtrženy. *skip-glyph* – ty znaky, které by se protnuly s čarou, která je podtrhává, podtrženy nebudou. *skip-glyph-and-white-space* – bude stejné jako předešlé, jen rozšířené o mezery, ty také podtrženy nebudou.

17.17.4 OSTATNÍ VLASTNOSTI DEKORACE TEXTU: text-underline-position a text blink

První vlastnost určuje pozici podtržené čáry. Hodnoty: *auto* – čára bude nastavena automaticky podle typu písma. *before-edge* – jestliže je nastavena tato hodnota, bude zarovnána před pozicí čáry určené podle okrajové čáry v boxu. *alphabetic* – podtržení bude zobrazeno pod řádkou v šířce základních písmen naší abecedy. *after-edge* - jestliže je nastavena tato hodnota, bude zarovnána za pozicí čáry určené podle okrajové čáry v boxu.

A druhá vlastnost *text-blink* nastavuje problikávání textu. Existují pouze dvě hodnoty. Hodnota *none*, při které tento efekt aplikován nebude. A hodnota blink nastavuje blikání (není však ještě určeno, zda tuto vlastnost budou některé prohlížeče vůbec podporovat).

17.17.5 SOUHRNÉ VLASTNOSTI PRO DEKORACI TEXTU: text-underline, text-line-through, text-overline a text decoration

Pomocí těchto souhrnných vlastností lze nastavovat hodnoty pro vlastnosti pro dekoraci textu, podobně jako u jiných souhrnných vlastností.

Ukázka použití dekorace textu pro odkaz.

```
a[href] { text-decoration: underline blink }
```

17.18 STÍNOVÁNÍ TEXTU: text-shadow

Pomocí této vlastnosti můžeme stínovat text. Hodnoty jsou pouze *none* – při které žádné stínování nebude zobrazeno a hodnota *shadow* – při které naopak stínování použito bude. U následujícího textu bude stínování použito směrem vpravo a dolu od elementu *h1*.

```
h1 { text-shadow: 0.2em 0.2em }
```

17.19 PŘEVEDENÍ NA KAPITÁLKY: text-transform

Hodnoty: *capitalize* – převádí každé první písmeno v každém slově na kapitálku. *uppercase* – převádí všechna písmena v každém slově na kapitálky. *lowercase* – naopak převádí všechny písmena na malá. A samozřejmě hodnota *none* – nepřevede písmena ani na malá, ani na velká.

```
h1 { text-transform: uppercase }
```

17.20 ZAVĚŠENÁ INTERPUNKCE: hanging-punctuation

Tato vlastnost určí zda interpunkční značka, jestliže je přítomna, může být umístěna vně obsahu a to buď na začátku nebo na konci řádky textu. Hodnoty: *none* – žádné interpunkční znaménko nebude zobrazeno. *start* – znaménko bude na začátku textu, ke kterému znaménko patří. *End* – je přesný opak předchozí hodnoty, tedy znaménko bude na konci textu. A hodnota *both* – je spojením obou předchozích hodnot. Tedy znaménko může být jak na konci, tak na začátku.

Zde je vidět, že CSS3 je velmi odlišné od předešlých levelů. Má spoustu nových vlastností a již existující vlastnosti mají nové hodnoty, či jsou upřesněny jejich významy, také řeší problémy pro ostatní jazyky. CSS3 se hodně přibližuje profesionální sazbě.

18 CSS3 MODUL THE BOX (BOXŮ)

Formátovací model CSS popisuje obdélníkové rámy, které klient generuje pro prvky dokumentu. Základem každého rámu je oblast obsahu (text, obrázek atd.). Kolem ní mohou být volitelně další oblasti: výplňová oblast (padding), oblast rámečku (border) a oblast okraje (margin). Jejich rozměry popisují příslušné vlastnosti CSS a také odpovídající typ formátování. Pro rozměry všech oblastí kolem obsahu, existují samostatné vlastnosti, také jejich rozměr je možné určit na každé straně jiný.

18.1 VLASTNOSTI: display, display-model a display-rule

Vlastnost *display-model* určuje algoritmus, podle kterého budou umístěny potomci prvků, které již tvoří rozhraní dokumentu. A naopak vlastnost *display-rule* určuje jakou funkci mají prvky v rodičovských elementech. A vlastnost *display* je opět souhrnná.

Hodnoty vlastnosti *display-model* jsou: *inline-inside* (uvnitř řádky) *block-inside* (uvnitř bloku), *table* (tabulka – tento modul je teprve připraven k návrhu, ještě neexistuje), *ruby* (ruby – viz. dále).

Vlastnost display-role má hodnot spoustu, zde je jejich přehled: *none* (žádné), *block* (blok), *inline* (v jedné řádce), *list-item* (položka seznamu) *run-in* (tento efekt záleží na tom, jaký bude následovat prvek), *compact* (kompaktní), *table-row* (řádka tabulky), *table-cell* (buňka tabulky), *table-row-group* (skupina řádek tabulky), *table-header-group* (záhlaví tabulky), *table-footer-group* (zápatí tabulky), *table-column* (sloupec tabulky), *table-column-group* (skupina sloupců tabulky), *table-caption* (titulek tabulky), *ruby-text, ruby-base* (ruby základ), *ruby-base-group* (skupina základů ruby), *ruby-text-group* (skupina textu ruby).

Následující kód ukazuje použití různých druhů boxů.

```
<style type="text/css">
```

```
  h3 { display: run-in; margin: 1em 0 }
  h3:after { content: ". " }
  p { display: block; margin: 1em 0 }
  img { display: block; margin: 2em }
  span { display: inline-block; padding: 0.6em;
    font-size: 70%; vertical-align: middle }
</style>
<h3>First heading</h3>
<h3>Second heading</h3>
<p>This paragraph has an image that is displayed as a block:
  <img src="w3c_home" ALT="W3C">
  and also an inline-block:
  <span>
    This element<br>
    has two lines
  </span>
```

18.1.1 KOMBINOVÁNÍ PLOVOUCÍHO ROZHRANÍ A JINÝCH NÁVRHŮ

Ve formátu dokumentu, kde chceme kombinovat více možností rozvržení je třeba použít i jiné doplňující hodnoty k *display-mode* pro návrhy stránek které nejsou definovány pomocí CSS3.

18.1.1.1 VLASTNOST padding

- Vlastnost komplexně nastavuje vnitřní okraje bloku.
- Lze aplikovat na všechny elementy.
- Vlastnost není dědičná.
- Hodnoty: Tato vlastnost má buď čtyři, tři, dva nebo jeden parametr. *padding-top , padding-right, padding-bottom, padding-left*
- V případě že má parametry čtyři nastavujeme okraje v pořadí nahoře, vpravo, dole, vlevo. V případě tří parametrů nastavujeme takto nahoře, vpravo a

zároveň vlevo, dole. V případě dvou parametrů nahoře a zároveň dole, vlevo a zároveň vpravo. A v případě jednoho parametru jsou všechny okraje stejné. Hodnoty parametrů jsou shodné s předchozími vlastnostmi.

- Implicitní hodnota: *0*

18.2 VLASTNOSTI okrajů (margin)

Vlastnosti *margin-top* (shora) , *margin-right* (zprava), *margin-bottom* (zdola), *margin-left* (zleva), nastaví okraje podle svých názvů. Okraje lze nastavit buď automaticky, čili implicitně, nebo vyjádřením vzdálenosti a to buď procentuálně nebo absolutně, číslem.

A souhrnná vlastnost *margin*, která nastavuje současně všechny možné hodnoty najednou.

18.3 VLASTNOSTI box-width a box-height

- Vlastnost nastavující šířku, či výšku elementu
- Lze je aplikovat na blokové a nahrazované elementy.
- Vlastnost není dědičná.
- Hodnoty: *auto* číselná hodnota s jednotkami (implicitně pixel) procentuelní hodnota vztahující se k šířce rodičovského elementu
- Implicitní hodnota: *auto*

```
BUTTON { box-width: 10% }
```

18.4 VLASTNOST box-sizing

- Hodnoty: *content-box* nastavení výšky i šířky přímo boxu s textem, *border-box* – nastavení výšky i šířky pro okraje boxu.
- Lze je aplikovat na blokové a nahrazované elementy.
- Vlastnost není dědičná.

```
BUTTON { border: 2px solid; width: 10% - 2 * 2px }
```

18.5 VLASTNOSTI min-width, max-width, min-height, max-height

Tyto vlastnosti nám dovolují vytvořit určité rozmezí při nastavování šířky elementu boxu. Vlastnost *min-width* určuje minimální a vlastnost max-width maximální šířku boxu. Vlastnost *min-height* určuje minimální a vlastnost *max-height* maximální výšku boxu.

- Lze aplikovat na elementy s vlastností *position: absolute* nebo *relative*.
- Vlastnost není dědičná.
- Hodnoty: *auto* číselná hodnota s jednotkami (implicitně pixel) procentuelní hodnota vztahující se k šířce rodičovského elementu, *inherit*
- Implicitní hodnota: *auto*

18.6 VLASTNOSTI fit, fit-position

Vlastnost *fit* určuje, jaká škála velikostí bude určena, když vlastnosti *width* i *height* budou nastaveny na hodnotu *auto*, nejvíce se používá na obrázky.

- Hodnoty: *fill* – velikost nezáleží na výšce ani šířce, ale box je určen tak,že se dotýká okrajů stránky. *None* – pro objekt nebude použita žádná škála. *Meet* (box udělá tak velký, jak jen to jde, podle jeho šířky nebo výšky). *slice* (to je přesně opak, box udělá tak malý, jak jen to jde)

Pomocí vlastnosti *fit-position* nastavujeme přesněji velikost boxu a to buď procentuálně, nebo absolutně.

18.7 VLASTNOST float

- Vlastnost nastavuje způsob, kterým mohou ostatní elementy obtékat tento element.
- Lze aplikovat na všechny elementy.

- Vlastnost není dědičná.
- Hodnoty: *left, top* (text obtéká horizontálně od shora dokumentu a vpravo – prvek je umístěn v levo) *right, bottom* (anologicky k předchozí hodnotě. Obtékání prvku umístěného na pravém okraji – text obtéká zleva a od začátku boxu), *inside* (stejný efekt jako u *left* nebo *right*, u nestránkovaných dokumentů je to *left*), *outside* (stejný jako u *right* nebo *left* u nestránkových dokumentů je to *right*) *start* (je stejný jako *left, right, top* nebo *bottom*, záleží zde na řízení – např. řízení bude *ltr* pak efekt bude *left, top* apod.) *end* (obdobně jako *start*). *None* (obtékání elementu není povoleno)
- Implicitní hodnota: none

```
<STYLE TYPE="text/css">
  P { width: 24em }
  #L1 { float: left; width: 8em; height: 3em }
  #L2 { float: left; width: 4em; height: 6em }
  #R1 { float: right; width: 6em; height: 9em }
  #L3 { float: left; width: 7em; height: 9em }
  #R2 { float: right; width: 3em; height: 5em }
</STYLE>
<P>
  <IMG ID=L1 SRC="L1.png" ALT="L1">
  <IMG ID=L2 SRC="L2.png" ALT="L2">
  <IMG ID=R1 SRC="R1.png" ALT="R1">
  <IMG ID=L3 SRC="L3.png" ALT="L3">
  <IMG ID=R2 SRC="R2.png" ALT="R2">
  nějaký jakýkoliv text prostě cokoliv ...
</P>
```

18.8 VLASTNOST clear

- Vlastnost povoluje či zakazuje elementu obtékat okolo jiného elementu.
- Lze aplikovat na všechny elementy.

- Vlastnost není dědičná.
- Hodnoty: *none* (element obtéká všechny objekty z obou stran) *left, top* (zakazuje elementu obtékat objekty na levé straně od shora dokumentu) *right, bottom* (zakazuje elementu obtékat objekty na pravé straně od jejich počátku) *both* (znemožňuje elementu obtékat) *inside* (stejný efekt jako u *left* nebo *right*, u nestránkovaných dokumen-tů je to *left*), *outside* (stejný jako u *right* nebo *left* u nestránkových dokumentů je to *right*) *start* (je stejný jako *left, right, top* nebo *bottom*, záleží zde na řízení – např. řízení bude *ltr* pak efekt bude *left, top* apod.) *end* (obdobně jako *start*).
- Implicitní hodnota: *none*

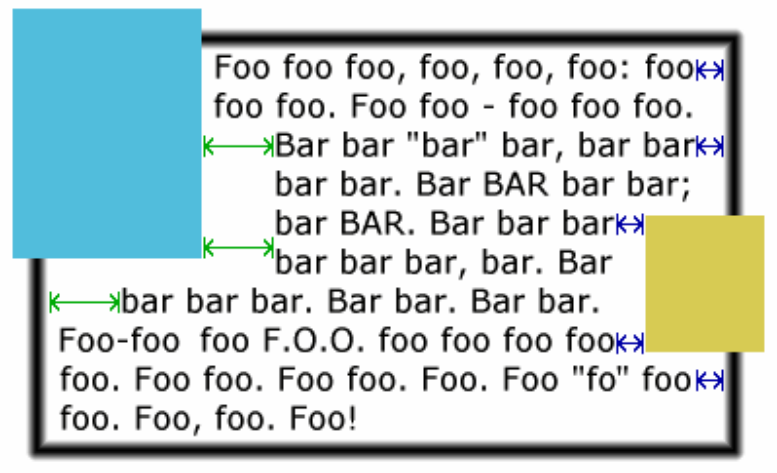
18.9 VLASTNOST clear-after

Někdy potřebujeme zajistit, aby spodní okraj vloženého bloku byl pod obtékaným boxem. Tato vlastnost zvyšuje spodní okraj, tak jak je potřeba.

- Hodnoty jsou: *none* – žádný efekt. *Left, top* – jen levé a horní obtékání bude bráno v úvahu. *Right, bottom* – jen pravé a spodní obtékání se bude brát v úvahu. Inside - stejný efekt jako u *left* nebo *right*, u nestránkovaných dokumentů je to *left*. Outside stejný jako u *right* nebo *left* u nestránkových dokumentů je to *right. Start* - je stejný jako *left, right, top* nebo *bottom*, záleží zde na řízení – např. řízení bude *ltr* pak efekt bude *left, top* apod. *End* (obdobně jako *start*.
- Lze aplikovat na všechny elementy.
- Vlastnost není dědičná.

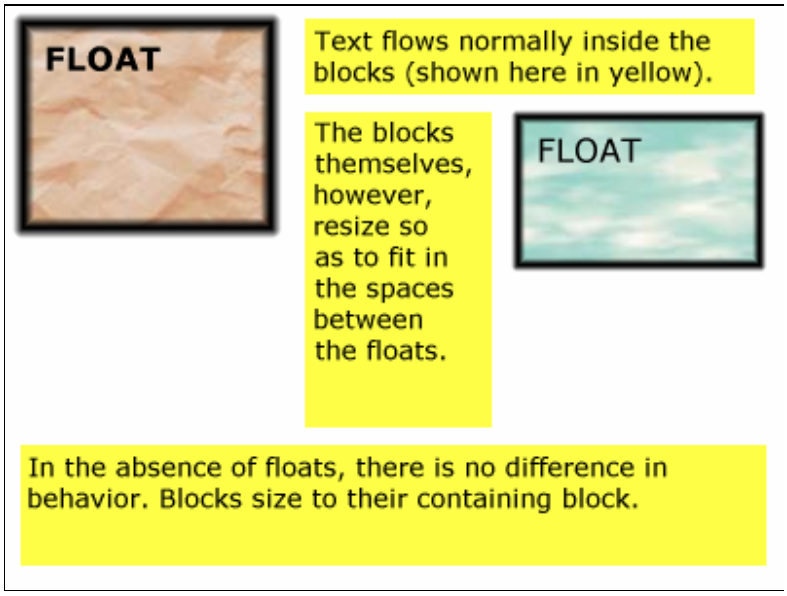
18.10 VLASTNOST float-displace

Tato vlastnost určuje dělení slov při použití obtékání. Hodnoty: *line* – řádkové boxy budou zkráceny a přesunuty tak aby utvořili obtékání. *Indent* – zajišťuje že relativní odsazení je udržováno i přes obtékání.(viz obrázek č.14).



Obrázek č.14 – použití vlastnosti *indent*

Block - Obsahující blokovou šíři užívanou při vodorovném formátovacím modelu.(viz obr. č. 15).



Obrázek č.15 – použití vlastnosti *block.*

Poslední hodnota *block-within-page* – má stejný vliv, jako hodnota *block*, ale bez stanovení, které ruší přizpůsobení šíře bloku, to je děláno samostatně pro každou stránku na které se blok objeví.

Nastavení pravidel pro HTML může vypadat například takto“

```
OL, UL, LI {float-displace: indent}
TABLE {float-displace: block}
```

18.11 VLASTNOST indent-edge-reset

Tato vlastnost určuje který okraj se použije jako odkaz pro nastavení okraje když vypočtené množství odsazení uchovávající hodnoty. Hodnoty: *none* - 'Tento blok neuvede žádnou novou vodicí hranu. *margin-edge border-edge, padding-edge, content-edge* tyto hodnoty mají stejnou hodnotu jako stejnojmenné u textu.

18.12 VLASTNOST overflow

Vlastnost nastavuje způsob práce s elementy, které se nevejdou do nadefinovaného prostoru. Zda element tento prostor přeteče, bude oříznut či jej bude možno skrolovat a tím prohlížet celý. Tato vlastnost má oproti doporučení CSS1 zcela odlišné hodnoty.

- Vlastnost lze aplikovat na všechny elementy s relativní nebo absolutní pozicí či nahrazované elementy.
- Vlastnost není dědičná.
- Hodnoty: *visible* - elementu je povoleno libovolně přetékat stanovený rámeček *hidden* - element je oříznut na velikost rámečku velikost a část zobrazené části závisí na vlastnosti *scroll* - element je možno v menším rámečku skrolovat a tím prohlížet celý. *auto* - typ zobrazení se vybere dle možností uživatelova prohlížeče .
- Implicitní hodnota: *visible* od normy CSS2.

```
BLOCKQUOTE { width : 125px; height : 100px; margin-top: 50px;
            margin-left: 50px; border: thin dashed black }
```

18.13 VLASTNOST overflow-x, overflow-y

Tato vlastnost má úplně stejný efekt, jako předchozí hodnota *overflow* s tím rozdílem, že u vlastnosti *overflow-x* může jen určit, jestli text bude oříznut, a nebo, bude použito skrolování. U vlastnosti *overflow-y* – vertikální skrolování. Obě vlastnosti mají o jednu hodnotu více, a to hodnotu *inherit* – protože jsou zděděny z vlastnosti *overflow*.

18.14 SOUHRNÁ VLASTNOST marquee

Tyto vlastnosti se používají jen tehdy, když prohlížeč používá skrolování pro přetečení.

Vlastnosti pro souhrnnou vlastnost *marquee* jsou: *marquee-style* ,čili určuje styl skrolování a má hodnoty: *none* – obsah se nebude vůbec skrolovat*, slide* – obsah se začíná rolovat vně obsahu boxu, *scroll* - obsah se začíná rolovat vně obsahu boxu a skroluje ho až do posledního řádku*, alternate* - obsah začíná jako viditelný, s jedním okrajem proti okraji boxu a pak se skroluje do té doby , než protější okraj obsahu je proti protějšímu okraji boxu.

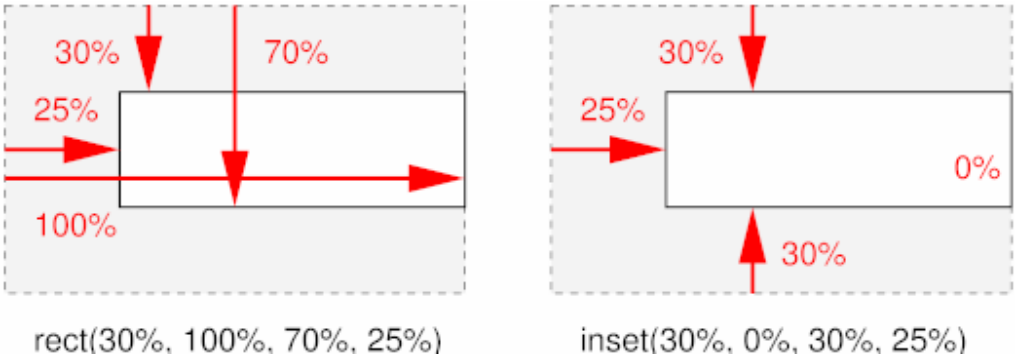
Vlastnost *marquee-direction* – nastavuje řízení rolování. Hodnoty: *forwards* – hodnota závisí na tom, jak je nastavena vlastnost *writing-mode*, *backwards* – obsah se začne rolovat naproti *inline progression direction. ahead* – rolování začne v *block progression direction*. Hodnota *reverse* - rolování začne naproti *block progression.* Hodnota *left* – rolování začne pohybem vlevo, *right* – vpravo, *up* – shora, *down* – dole a poslední hodnota *auto* – nastavuje automatické rolování obsahu.
Vlastnost - *marquee-speed* – určuje rychlost rolování podle hodnot: *slow* – pomalu, *normal* normálně a hotnota *fast* – rychle.

A poslední vlastnost *marquee-repetition* – určuje počet opakování rolování. Hodnota je buď číselná (celé číslo), která vyjadřuje přesný počet opakování a hodnota *infinite* – která značí nekonečné opakování.

```
p { overflow: scroll;
  white-space: nowrap;
  marquee: slide infinite }
```

18.15 VLASTNOST overflow-clip

Tato vlastnost určuje která část obsahu bude viditelná při přetečení. Hodnoty: auto – bude nastaveno automaticky, *rect(trbl)* - Obsah je zkrácený vzhledem k obdélníku daného čtyřmi vyrovnáními z vrchol levého rohu obsahu plochy (nebo dalšího rohu, v závislosti na *writing-mode)*(viz obr. č. 16). A hodnota *inset-rect(trbl)* - Zkracuje vzhledem k obdélníku danému čtyřmi hodnotami a je vyrovnaný ke čtveřici okrajů obsahu oblasti (viz obr. č. 17).



Obrázky č.16, 17 – použití vlastnosti *overflow-clip*

18.16 VLASTNOST visibility

- Vlastnost nastavuje viditelnost elementu.
- Lze aplikovat na všechny elementy.
- Vlastnost není dědičná.

- Hodnoty: *visible* - element je viditelný *hidden* element není viditelný a ani by neměl zabírat žádný prostor v dokumentu *collapse* toto je specifický efekt pro použití v tabulkách, kterým můžete vypouštět řádky a sloupce, při použití na jiné elementy shodný s *hidden*.
- Implicitní hodnota: *visible*

Následující skript způsobí, že odpovídající box, se stane viditelným a druhý skrytý.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<STYLE type="text/css">
<!--
   #container1 { position: absolute;
                 top: 2in; left: 2in; width: 2in }
   #container2 { position: absolute;
                 top: 2in; left: 2in; width: 2in;
                 visibility: hidden; }
-->
</STYLE>
</HEAD>
<BODY>
<P>Vyber podezřelého (suspect):</P>
<DIV id="container1">
   <IMG alt="Al Capone"
        width="100" height="100"
        src="suspect1.jpg">
   <P>Jméno: Al Capone</P>
   <P>Bydliště: Chicago</P>
</DIV>

<DIV id="container2">
```

```
   <IMG alt="Lucky Luciano"
        width="100" height="100"
        src="suspect2.jpg">
   <P>Jméno: Šťastný Luke</P>
   <P>Bydliště: New York</P>
</DIV>

<FORM method="post"
      action="http://www.suspect.org/process-bums">
   <P>
   <INPUT name="Capone" type="button"
          value="Capone"
          onClick='show("container1");hide("container2")'>
   <INPUT name="Luke" type="button"
          value="Luke"
          onClick='show("container2");hide("container1")'>
</FORM>
</BODY>
</HTML>
```

19 CSS3 MODUL BORDER (RÁMEČKŮ)

Vlastnosti obsažené v tomto modulu jsou návrhem pro definování šířky, barvy a stylu rámečků, pro prvky HTML.

19.1 VLASTNOSTI border-color

Vlastnosti nastavují barvu rámečku. Lze je aplikovat na všechny elementy. Má však smysl pouze v případě, že je alespoň jedna strana rámečku viditelná. Vlastnost není dědičná. Hodnotou je barva v jednom z dříve uvedených formátů. Implicitní hodnota: hodnota vlastnosti *color*.

Vlastnosti mohou být: *border-top-color, border-right-color, border-bottom-color, border-left-color*. A ty nastavují barvy jednotlivým ohraničením tabulky, takže tabulka nemusí mít okraj jen červený, nebo modrý, ale například červeno-modro-zeleno-fialový. Toto zavedla již norma CSS2.

19.2 VLASTNOSTI border-style

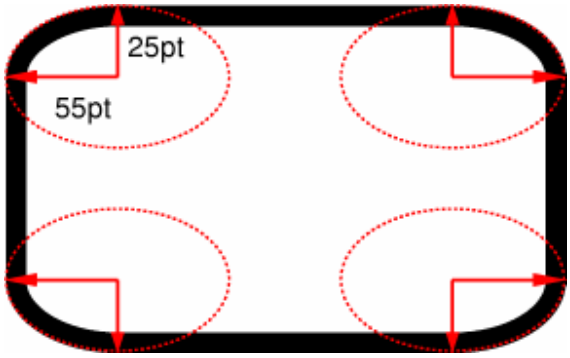
Tyto vlastnosti nastavují způsob vykreslení rámečku (tvar nebo styl). CSS3 umožňuje opět nastavit tento styl u každé čáry zvlášť. Lze je aplikovat na všechny elementy. Vlastnost není dědičná.

- Možné vlastnosti tedy jsou: *border-top-style, border-right-style, border-bottom-style* a *border-left-style.*
- Hodnoty jsou: *none* – žádný styl čáry nebude použit*, hidden* – má stejný efekt, jako *none*, ale jinak by to vypadalo při použití na tabulky*, dotted* – tečkovaná čára*, dashed* – čárkovaná čára*, solid* – jednoduchá čára*, double* dvojitá*, dot-dash* - čerchovaná*, dot-dot-dash* – dvojitá čerchovaná*, wave* - vlnovka*, groove* - vyrytá*, ridge* - vyvýšená*, inset* – vložená čára a *outset* – ozdobná čára.

19.3 VLASTNOSTI border-radius

Pomocí této nové vlastnosti můžeme konečně naše okraje boxů, či tabulek zaoblit. A nejen zaoblit všechny rohy, ale třeba jen jeden, prostě podle naší libosti, tak jak se nám to bude hodit.

- Vlastnosti: *border-top-right-radius, border-bottom-right- radius, border-bottom-left-radius, border-top-left- radius* a *border-radius*.
- Hodnotami jsou dvě čísla, respektive délky, které určují radius čtvrtiny elipsy. (viz obr. 18).



Obrázek č.18 – použití vlastnosti *border-radius*

19.4 VLASTNOSTI OKRAJŮ OBRÁZKŮ

Tyto vlastnosti dovolují autorům přiřadit okrajům různé obrázky a tím vytvořit efektivní vzhled. Existují tři skupiny vlastností *border-image*:

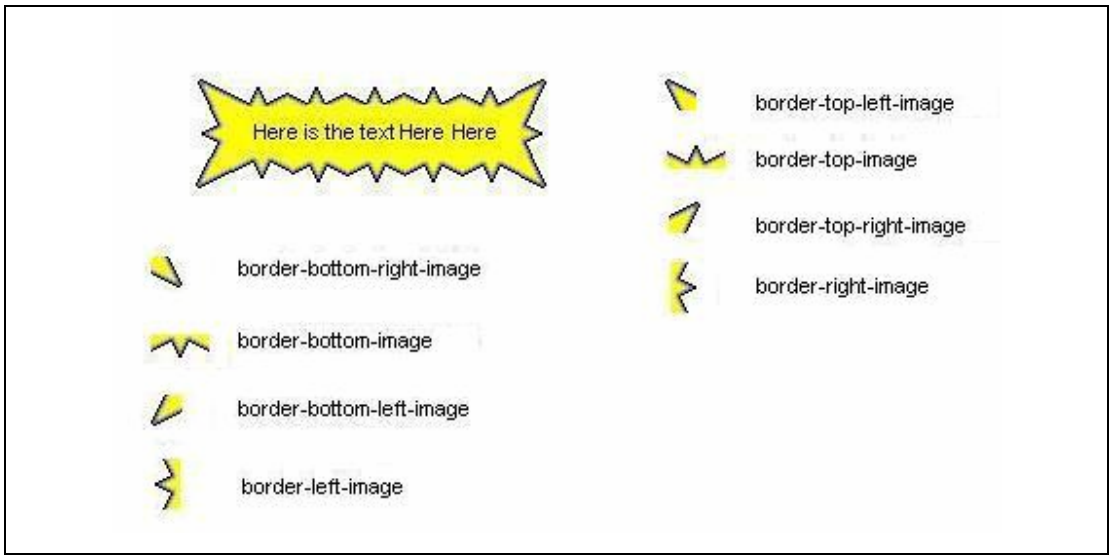
- Specifikace rámečku orázků: *border-image*
- Vhodné okraje obrázků: *border-fit*
- Přetvoření okrajů obrázků: *border-image-transform*

19.4.1 VLASTNOSTI border-image

Vlastnosti *border-top-image, border-right-image, border-bottom-image, border-left-image* a samotnou vlastnost *border-image,* která je opět souhrnnou. Tyto vlastnosti mohou mít tyto hodnoty: adresa obrázku, čili *URI, {1,3} a none.*

Dalšími vlastnostmi jsou: *border-top-left-image, border-top-right-image, border-bottom-right-image, border-bottom-left-image,* tyto mohou nabývat těchto hodnot: opět *uri, continue* – opakování obrázku*, none, inherit.*

Následuje ukázka rámečku, vytvořeným pomocí výše uvedených vlastností.



Obrázek č.19 – použití vlastnosti *border-image*

19.4.2 VLASTNOSTI border-fit

Tyto vlastnosti určují jestli a jak bude obrázek pro okraj „vydlaždičkován“. Možné hodnoty mají následující význam:

clip – umísťuje jednu „dlaždičku“(kousek obrázku), bez potažení a zkracuje přetečení

repeat – opakuje obrázek do té doby, než utvoří celý okraj a zkracuje přetečení.

scale – opakuje obrázek k ohraničení tak blízko vyplněné plochy, jak jen to je možné.

A potahuje všechny dlaždičky nahoru nebo dolů, tak jak je potřeba k vyplnění dané oblasti, bez přetečení.
stretch – roztáhne nebo naopak zúží obrázek, takže jediná dlaždička pokryje celou plochu.
overwrite – umístí jednu dlaždičku, bez potažení, ale přetečení nezkracuje.
overflow – opakuje obrázek do té doby, než utvoří celý okraj, ale nezkracuje přetečení.
space – opakuje obrázek tak blízko k výplni, jak jen to je možné, ale vynechává mezery, takže vydlaždičkování je naprosto přesné.

- Vlastnosti: *border-top-fit-length, border-right-fit-length, border-bottom-fit-length, border-left-fit-length, border-top-fit-width, border-right-fit-width, border-bottom-fit-width, border-left-fit-width* a souhrnná vlastnost *border-fit.*
- Hodnoty: *clip, repeat, scale, stretch, overwrite, overflow, space.*
- Implicitní hodnota přitom je *repeat*

- Dalšími vlastnostmi jsou: *border-top-left-fit-length , border-top-right-fit-length, border-bottom-right-fit-length, border-bottoml-eft-fit-length, border-top-left-fit-wid-th, border-top-right-fit-width, border-bottom-right-fit-width, border-bottom-left-fit-width* a souhrná vlastnost *border-corner-fit.*
- Tyto vlastnosti mohou mít následující hodnoty: *clip repeat scale stretch overwrite overflow space*
- Implicitní hodnota je *overwrite.*

19.4.3 VLASTNOSTI border-image-transform

Tyto vlastnosti jsou použity ke specifikování toho, že obrázek na rohu je přetvářenn pomocí *border-top-left-image* nebo obrázkům na okraji, které jsou přeměněny pomocí *border-top-image* a *border-left-image.* Možné transformace jsou:
none – umístí obrázek bez jakékoliv modifikace.

rotate – otočí obraz tak, aby strana který by by čelila vnitřnímu okraji kdyby byla umístěna na vrchol okraje, zůstává otočena směrem k vnitřnímu okraji.

reflect – „švihne“ obrazem tak, že strany které by byly vnitřními okraji obrazu byly umístěny na vrcholu levého rohu zbývající vnitřní strany. Tento efekt může být modifikován, dalšími hodnotami a to: *reflect-xy, reflect-right, reflect-left* význam těchto hodnot je analogický s hodnotou *reflect.*

Vlastnosti existují pouze dvě, a to: border-corner-image-transform, která má hodnoty: none, reflect i rotate. A vlastnost border-image-transform, jejímiž hodnotami jsou*: none, rotate, reflect-xy, reflect-right, reflect-left*

19.5 VLASTNOST border-break

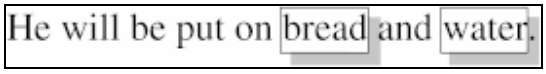
Jak sám název napovídá, pomocí této vlastnosti můžeme upravovat boxy, které mají nějaký okraj, v případě, že jsou přerušeny stránkováním, nebo zalomením sloupce, či inline elementy. V řádce, kde toto přerušení nastalo, může být vložen okraj, nebo tento může být ponechán otevřený. Hodnotou je <*border-style*>, viz výše.

19.6 VLASTNOST box-shadow

Díky této vlastnosti, jeden nebo více stínů může být spojeno s boxem. Jsou taženz kousek od okraje boxu. Vlastností seznam stínů, oddělených čárkou, každá je specifikována třemi délkami, hodnotou a barvou. Jestliže vynecháme délku je nastavena implicitně na 0, jestliže opomeneme barvu, rovná se vypočtené hodnotě vlastnosti *color* (viz výše, modul color). První zadaná délka je horizontálním doplňkem, druhá délka je výrazem vertikálního zobrazení stínu a třetí délka je rozostřeného radiusu. A barva je samozřejmě barvou stínu. Následuje ukázka pro znázornění.

```
span {border: thin solid; box-shadow: 0.2em 0.2em #CCC}
```

A výsledkem je znázorněn na obrázku č. 20:



Obrázek č.20 – použití vlastnosti *box-shadow*

19.7 SOUHRNÉ VLASTNOSTI OKRAJŮ: border-top, border-bottom, border-right, border-left a border

Tyto vlastnosi dovolují souhrně nastavit vzhled jednotlivým okrajům. A pomocí vlastnosti *border,* lze nastavit vše v jednom.

```
h1 { border-bottom: thick solid red url('lines.jpg')}

p { border: solid red }
p {
  border-top: solid red;
  border-right: solid red;
  border-bottom: solid red;
  border-left: solid red
}
blockquote {
  border-color: red;
  border-left: double;
  color: black
}
```

20 CSS3 MODUL LISTS (SEZNAMY)

Seznamy jsou specifickým případem generování obsahu. Zde již nedefinujeme obsah zcela libovolně, pouze si můžeme vybrat z několika přednastavených možností. Navíc je zde ještě jedna odlišnost. Položky seznamů (prvky s *display:list-item)* negenerují pouze jediný hlavní rám, ale mohou generovat i rám doplňkový, do nějž se umístí značka uvozující položky seznamu.

20.1 VLASTNOST: list-style-type

- Vlastnost nastavuje vzhled odrážek (uvozující značky seznamu).
- Lze aplikovat na elementy, které mají *display: list-item*
- Vlastnost je dědičná.
- Hodnoty: *disc* (položka seznamu je uvozena tečkou) *circle* (položka seznamu je uvozena kroužkem) *square* (položka seznamu je uvozena kostičkou) *decimal* (položka seznamu je uvozena číslem, počáteční číslice standardně 1) *lower-roman* (položka seznamu je uvozena římskou číslicí, která je psaná malými písmeny) *upper-roman* (položka seznamu je uvozena římskou číslicí, která je psaná velkými písmeny) *lower-alpha* (položka seznamu je uvozena malým písmenem, počáteční hodnota standardně a) *upper-alpha* (položka seznamu je uvozena velkým písmenem) *none* (bez uvozujícího znaku)
- Implicitní hodnota: *disc*

```
li {font-size: 15pt; list-style-type: square;}
li1 {display: list-item; list-style-type: upper-alpha;
font-size: 9pt; font-weight: bold}
//tento element bude volán jako <LI1>
```

20.1.1 VLASTNOST: list-style-image

- Vlastnost nastavuje místo uvozujícího znaku seznamu obrázek.

- Lze aplikovat na elementy, které mají *display: list-item*
- Vlastnost je dědičná.
- Hodnoty: *none* (obrázek není přiřazen), *uri* adresa požadovaného obrázku
- Implicitní hodnota: *none*

```
LI { list-style-image: url("http://www.ukazky.cz/oval.png") }
```

20.1.2 VLASTNOST: list-style-position

- Vlastnost určuje umístění odrážky seznamu, buďto před textem nebo v textu.
- Lze aplikovat na elementy, které mají *display: list-item*
- Vlastnost je dědičná.
- Hodnoty: *outside* (odrážka umístěna před textem, text se zalamuje k začátku textu prvního řádku) *inside* (odrážka umístěna v textu, text se zalamuje k začátku odrážky)
- Implicitní hodnota: *outside*

```
li {font-size: 15pt; list-style-type: square;
list-style-position: outside}
li1 {display: list-item; list-style-type: upper-alpha;
font-size: 9pt; font-weight: bold; list-style-position: inside}
//tento element bude volán jako <LI1>
```

20.1.3 ZNAČKY - PSEUDO-ELEMENT ::marker

Značky jsou vytvořeny nastavením prvku vlastnosti *display* do položky seznamu. Položky seznamu titulkového písma je v každém druhém respektu, totožném s blokem titulkového písma. Lépe to možná pochopíme pomocí příkladů. První ukazuje je vycentrovaný obsah uvnitř odrážek v boxu s pevnou šíří:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

```
<HTML>
 <HEAD>
  <TITLE>Content alignment in the marker box</TITLE>
  <STYLE type="text/css">
   LI::marker {
     content: "(" counter(counter) ")";
     width: 6em;
     text-align: center;
   }
   LI {
     display: list-item;
     counter-increment: counter;
   }
  </STYLE>
 </HEAD>
 <BODY>
  <OL>
   <LI> Toto je první položka. </LI>
   <LI> Toto je druhá položka. </LI>
   <LI> Toto je třetí položka. </LI>
  </OL>
 </BODY>
</HTML>
```

výsledek může vypadat nějak takto.

```
  (1)    Toto je
         první položka.
  (2)    Toto je
         druhá položka.
  (3)    Toto je
         třetí položka.
```

Následující příklad používá značky k očíslování značek poznámek (odstavců).

Následuje dokument:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
 <HEAD>
  <TITLE>Značky k vytvoření poznámek>/TITLE>
  <STYLE type="text/css">
   P { margin-left: 12 em; }
   P.note::marker {
     content: url("poznamka.gif") "poznamka " counter(note-counter)
":";
      text-align: left;
      width: 10em;
   }
   P.Note {
     display: list-item;
     counter-increment: note-counter;
   }
  </STYLE>
```

```
 </HEAD>
 <BODY>
  <P>Toto je první odstavec v našem dokumentu.</P>
  <P CLASS="Note">Toto je velmi krátký dokument.</P>
  <P>Toto je konec.</P>
 </BODY>
</HTML>
```

může zobrazit něvo takovéhoto.

```
            Toto je první odstavec
            v našem dokumentu.

  Poznámka 1:   Toto je velmi krátký
            dokument.

            Toto je konec.
```

Následujíící příklad ilustruje jak značky mohou být vyrovnány ze svých prvků. Tato HTML aplikace a stylový předpis:

```
-<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE>Marker example 5</TITLE>
  <STYLE type="text/css">
   P { margin-left: 8em } /* Vytvoří mezeru pro počítadla */
   LI::marker { margin: 0 3em 0 0; content: counter(list-item,
lower-roman) "."; }
   LI { display: list-item }
  </STYLE>
 </HEAD>
 <BODY>
  <P> Toto je dlouhý předešlý odstavec ...</P>
  <OL>
   <LI> Toto je první položka
   <LI> Toto je druhá položka
   <LI> Toto je třetí položka
  </OL>
  <P> Toto je dlouhý následující odstavec ...</P>
 </BODY>
</HTML>
```

může zobrazit třeba takto:

```
        Toto je dlouhý předcházející
        odstavec ...

   i.   Toto je první položka
  ii.   Toto je druhá položka.
```

```
 iii.   Toto je třetí položka.

        Toto je následující
        odstavec ...
```

(Poznámka použití implicitního počítadla je inkrementováno.)

20.1.4 SHRNUJÍCÍ VLASTNOST: list-style

- Vlastnost komplexně nastavuje vzhled seznamu.Vlastnosti nastavujeme v pořadí list-style-type, list-style-position, list-style-image, při čemž definování první a zároveň poslední vlastnosti není nutné a postrádá smysl.
- Lze aplikovat na elementy, které mají *display: list-item*
- Vlastnost je dědičná.
- Hodnoty: hodnoty konkrétních vlastností
- Implicitní hodnota: implicitní hodnoty konkrétních vlastností

Nakonec k tomuto modulu uvádím shrnující příklad všech vlastností:

```
/* Stanovení položek menu */
li { display: list-item;
/* counter-increment: list-item; (implied by display: list-item) */
}

/* Nastavení ol a ul tak že vynuluje čítač položek menu */
ol, ul { counter-reset: list-item; }

/* Defaultní typy odrážek pro uspořádaný seznam */
ol { list-style-type: decimal; }

/* Defaultní typy odrážek pro neuspořádaný seznam až do hloubky 3 */
ul { list-style-type: disc; }
ul ul { list-style-type: square; }
ul ul ul { list-style-type: circle; }

/* Typ atributů prvků ol a ul */
```

```
ul[type="disc"] { list-style-type: disc; }
ul[type="circle"] { list-style-type: circle; }
ul[type="square"] { list-style-type: square; }
ol[type="1"] { list-style-type: decimal; }
ol[type="a"] { list-style-type: lower-alpha; }
ol[type="A"] { list-style-type: upper-alpha; }
ol[type="i"] { list-style-type: lower-roman; }
ol[type="I"] { list-style-type: upper-roman; }

/* Počáteční atribut ol prvků */
ol[start] { counter-reset: list-item attr(start, integer, 1);
counter-increment: list-item -1; }

/* Hodnota atributů prvků li */
li[value] { counter-reset: list-item attr(value, integer, 1);
counter-increment: none; }

/* Výše uvedená pravidla nepopisují plně seznamy HTML4, od doby kdy
nepokrývají chování takvových jako okrajů apod. Následující pravidlo
může být použito pro následující účel:
 ol, ul { display: block; margin: 1em 0; padding-left: 2.5em; }
ol ol, ol ul, ul ul, ul ol { margin-top: 0; margin-bottom: 0; }
li::marker { margin-right: 1em; text-align: right; } */
```

Seznamy se předchozí levely příliš nezabývali, ale CSS3 z nich udělal přímo jeden samostatný modul. A nejen to je změna od CSS2. Další jsou:

1. Pseudo element :marker musí být ve tvaru ::marker
2. Není potřeba ukončovat odrážky.
3. Vlastnost *marker-offset* je překonána
4. Odrážky jsou nyní zarovnávány relativně k okraji boxu.
5. Odrážky mají okraje.

21 CSS3 MODUL LINE (ČÁRA)

Nejen pro optické zvýraznění horizontálního dělení obvykle textových bloků se často využívá i vodorovné čáry, pro kterou v XHTML existuje nepárová značka <*hr/*>. Ale tento modul také zahrnuje design základní čáry zarovnání uvnitř každé linky a umístění zapuštěné iniciály. Proto tvůrci CSS3 věnovali lince celý modul. Většina vlastností je úplně nová.

21.1.1 VLASTNOST text-height

Tato vlastnost pomáhá vkládaným boxům určit rozměr (výška v horizontálním toku).

- Hodnoty: *auto* – Průběh rozměru bloku je založen buď na čtverčíkový rozměr, určený prvkem a vlastností *font-height* hodnota nebo výška buňky- (rostoucí i klesající) souvisela s velikostí písma prvku vybraného prohlížečem. *font-size* – rozměr bloku závisí na čtverčíku a velikosti vlastnosti *font-size*. *text-size* - rozměr bloku závisí na výšce buňky a velikosti vlastnosti *font-size. max-size* - rozměr bloku závisí na maximu rozsahu směrem k přednímu-okraji a za okrajem boxu získaného vzhledem k všem potomkům elementů umístěných na stejné lince.
- Použití na inline prvky a rodiče elementů s zobrazeným *ruby-text* (viz. dále)
- Implicitní hodnota: *auto*
- Vlastnost je dědičná

21.1.2 ÚPRAVA ŠÍŘKY ČÁRY, VLASTNOST line-height

Vlastnost nastavující výšku řádky textu (tzv. meziřádkový proklad).

- Hodnoty: *normál* – říká prohlížeči, aby šířku nastavil adekvátně k *font-cize*. Číselná hodnota s jednotkami udávající velikost meziřádkového prokladu; číselná hodnota bez jednotek udávající proklad jako násobek velikosti písma; procentuelní hodnota vztahující se k velikosti písma elementu Ve většině

případů by měl být proklad alespoň 1,2 (čili 120%) velikosti písma aby byla zachována dostatečná čitelnost i pro velká písmena (v případě češtiny i s háčky) a aby se písmena nepřekrývala.

Příklad 3 pravidel, mající stejnou hodnotu:

```
DIV { line-height: 1.2; font-size: 10pt } /* číselná */
DIV { line-height: 1.2em; font-size: 10pt } /* délková */
DIV { line-height: 120%; font-size: 10pt } /* procentuální */
```

21.1.3 VLASTNOSTI line-stacking-strategy line-stacking-ruby, line-stacking-shift, a line-stacking

Tento mechanismus hromadění, kterým linka boxu je předurčena pro každou linku v bloku a pak tyto linky jsou kupené v bloku je řešením jakéhokoliv omezení rozestupů mezi sousedními čarami.

První vlastnost určuje souhrnnou metodu hromadění.

- Hodnoty: *inline-line-height* – velikost hromady je nejmenší hodnota obsahující rozšířený postup bloku, ze všech vkládaných elementů na té lince kdy jsou tyto elementy vhodně uspořádány. *block-line-height* – výška zásobníku odpovídá prvku bloku a hodnotě vlastnosti *line-height*. *max-height* – výška hromady je tak velká jako nejmenší hodnota, která obsahuje rozměr bloku ze všech vkládaných elementů, na té lince, kde jsou tyto elementy vhodně uspořádány. A hodnota *grid-height* – odpovídá nejmenší mnohonásobné velikosti bloku, prvku *line-height*, vypočtená hodnota, která může obsahovat postup bloku, ze všech vložených prvků, na lince, ve které jsou tyto prvky řádně uspořádány.
- Implicitní hodnota: *inline-line-height*
- Tato vlastnost je dědičná.

Další vlastnost, kterou je line-stacking-ruby, určuje metodu hromadění linek pro blokové elementy obsahující poznámku k elementu (prvek se zobrazeným: *ruby-text-container*).

- Hodnoty: *exclude-ruby* - vysvětlivky elementů jsou ignorované pro skládání linek na hromadu. *include-ruby* - Ruby poznámka prvků je považována pro skládání linek na hromadu.
- Implicitní hodnota: *exclude-ruby.*
- Vlastnost je dědičná.

Další vlastností je *line-stacking-shift* - Tato vlastnost určuje skládání linek pro metodu bloku prvků, obsahující elementy se základem-posunu. Možné vlastnosti jsou: *consider-shift* - určující výšku hromady, která zahrnuje přizpůsobenou horní hranu a dolní hranu jakýchkoliv znaků, které mají posunutu základní čáru. Druhou hodnotou je: *disregard-shift* - určuje výšku zásobníku, zahrnuje neposunutou horní hranu a dolní hranu kterýchkoliv znaků, které mají posunutu základní čáru.

Poslední vlastností je souhrnná vlastnost *line-stacking*, která opět umožňuje nastavení všech výše uvedeným vlastnostem, obvyklým způsobem.

21.1.4 VLASTNOST dominant-baseline

Tato vlastnost je užívána pro určení nebo znovu určení okujené-základní čáry tabulky.

- Hodnoty: *auto, use-script* – použití skriptu, *no-change* - dominantní identifikátor základní čáry, základní čára tabulky a základní čára-tabulky velikost písma zůstane stejná jako rodičovský prvek. *reset-size* - dominantní identifikátor základní čáry, základní čára tabulky a základní čára-tabulky velikost písma se mění podle hodnoty *font-size. alphabetic* – dominantní základní linka se nastaví na alfabetickou základnu *hanging* - dominantní základní linka se nastaví na zavěšenou základnu. *ideographic* - dominantní základní čára-identifikátoru je nastaven na ideologickou základní čáru

mathematical – dominantní základní linka se nastaví na základnu matematickou *central* – dominantní základní linka se nastaví na centrální základnu *middle* - dominantní základní linka se nastaví na střední základnu. *text-after-edge* - dominantní základní linka se nastaví na základnu *text-after-edge*. A poslední hodnota *text-before-edge* je logicky nastavena na základnu *text-before-edge.*

- Implicitní hodnota: *auto*
- Dědičnost: ano

21.1.5 VLASTNOST PRO ZAROVNÁNÍ LINKY: alignment-baseline

Tato vlastnost má stejné hodnoty jako vlastnost předchozí:

- Hodnoty: *auto, use-script* – použití skriptu, *no-change* - dominantní identifikátor základní čáry, základní čára tabulky a základní čára-tabulky velikost písma zůstane stejná jako rodičovský prvek. *reset-size* - dominantní identifikátor základní čáry, základní čára tabulky a základní čára-tabulky velikost písma se mění podle hodnoty *font-size. alphabetic* – dominantní základní linka se zarovná k alfabetickou základnu *hanging* - dominantní základní linka se zarovná k zavěšené základně. *ideographic* - dominantní základní čára-identifikátoru zarovná k ideologické základní čáře *mathematical* – dominantní základní linka se zarovná k základně matematické *central* – dominantní základní linka se zarovná k centrální základně *middle* - dominantní základní linka se zarovná k střední základně. *text-after-edge* - dominantní základní linka se zarovná k základně *text-after-edge*. A poslední hodnota *text-before-edge* je logicky zarovnána k základně *text-before-edge.*
- Implicitní hodnota: *baseline*
- Dědičnost: ne

21.1.6 NASTAVENÍ ZAROVNÁVACÍHO BODU: alignment-adjust

Díky této vlastnosti připouští přesnější zarovnání prvků, jako je grafika, nemající základnu-tabulky nebo postrádávající požadovanou základní čáru na své základní lince tabulky.

- Hodnoty: *auto ,baseline* – bod zarovnání je v průsečíku začátku okraje a dominantní základně prvku. *before-edge* - bod zarovnání je v průsečíku začátku okraje a částí před počátkem okraje. *text-before-edge* bod zarovnání je v průsečíku začátku okraje a prvkem *text-before-edge. middle* - bod zarovnání je v průsečíku začátku okraje a středu základny danného elementu. *central* - bod zarovnání je v průsečíku začátku okraje a střední základně danného elementu. *after-edge* - bod zarovnání je v průsečíku začátku okraje a za okrajem prvku. *text-after-edge* - bod zarovnání je v průsečíku začátku okraje a základně *text-after-edge. alphabetic* – bod zarovnání je v průsečíku začátku okraje a alfabetickou základnou *hanging* - bod zarovnání je v průsečíku začátku okraje a zavěšené základně. *ideographic* – bod zarovnání je v průsečíku začátku okraje a ideologické základní čáře *mathematical* – bod zarovnání je v průsečíku začátku okraje a základně
- Implicitní hodnota je *auto.*
- Vlastnost je dědičná.

21.1.7 ZNOVU UMÍSTĚNÍ DOMINANTNÍ ZÁKLADNY baseline-shift

Vlastnost *baseline-shift* dovoluje znovu nastavení dominantní základny, relativně k dominantní základně prvku. Posunutý objekt může být dolním či horním indexem.

- Hodnoty: *baseline* – žádné posunutí nebude použito. *sub* – dominantní základna je posunuta k defaultní pozici dolního indexu. *super* - dominantní

základna je posunuta k defaultní pozici horního indexu. Další hodnoty jsou procentuální a délkové vyjádření posunutí.

- Implicitní hodnota: *baseline*
- Dědičnost: ne.

Tato vlastnost je pouhým náčrtem, jak by tato vlastnost měla vypadat, ale není dokonce ještě rozhodnuto, zda tato vlastnost bude v CSS3.

21.1.8 VERTIKÁLNÍ ZAROVNÁNÍ: vertical-align

Tato vlastnost ovlivňuje vertikální pozicování vložených bloků generovaných vložených levelů uvnitř linie boxu. Následující hodnoty mají význam jen tehdy, když respektujeme rodičovské blokové prvky. Jestliže tento prvek generuje anonymní vložené bloky, pak nemá tato vlastnost žádný efekt, když nemá žádné předky.

- Hodnoty: *auto, use-script* – použití skriptu, *baseline* – zarovnání alfabetické základny prvku, podle rodičovské základny. Když prvek tuto základnu nemá, zarovná se na dolní okraj boxu. *sub* zarovnání k dolnímu indexu prvku rodičovské základny. *sup* - zarovnání k hornímu indexu prvku rodičovské základny *top* - Seřadí část před okrajem rozšířeného vkládaného boxu s předním-okrajem linky boxu *text-top* - seřadí vrchol boxu s předním-okrajem rodičovského prvku fontu. *central* – zarovná střední základnu s předním okrajem rodičovského prvku *middle* – zarovná dominantní základní linku k okraji vloženého elementu. *ottom* – zarovnání zadního okraje rozšířeného boxu se zadním okrajem linky boxu. A poslední hodnotou je *text-bottom* - zarovná vrchol boxu se zadním-okrajem rodičovského prvku fontu.
- Implicitní hodnota: není nadefinována
- Dědičnost: ne

21.1.9 ZAROVNÁNÍ VLOŽENÝCH BLOKŮ inline-box-align

Tato vlastnost určuje která linka z multi-line vložených bloků bude zarovnána k předchozímu nebo následujícímu prvku.

- Hodnoty: *initial* – užívá počáteční linku vložených bloků za účelem zarovnání. *last* – užije poslední linku vloženého bloku kvůli zarovnání. Další hodnotou může být číselné vyjádření kolikátá linka se má použít.
- Implicitní hodnota: *last*
- Dědičnost: ne.)

21.1.10 ZMĚNA VELIKOSTI ZAPUŠTĚNÉ INICIÁLY drop-initial-size a drop-initial value

Pomáhá posadit iniciálu přesně na linku, kde iniciála začíná stejně jako text na této řádce. Prvně jmenovaná vlastnost je jednoduchým nástrojem k zarovnání iniciály, má hodnoty: *initial* – která posune iniciálu na daný řádek. A číselné hodnoty integerového typu, při které je iniciála posunuta k vybrané lince.

Druhá vlastnost kontroluje zanořování iniciály. Hodnoty jsou: *auto* – která bude automaticky nastavena podle prohlížeče. Hodnota *line* – posunutí iniciály bude relativní vzhledem k výšce vybrané linky. Další hodnotou je číselné vyjádření délky zapuštění iniciály. A poslední je procentuální vyjádření zapuštění.

- Použití: pseudo-element *::first-letter*
- Dědičnost: ne
- Implicitní hodnota: auto

21.1.11 ZAROVNÁNÍ INICIÁLY: drop-initial-before-align, drop-initial-before-adjust, drop-initial-before-align a drop-initial-after-adjust

Drop-initial-after-align určuje, jaké zarovnání linky uvnitř n-té linky boxu je definováno s iniciálou. Jedinou hodnotou je právě hodnota vlastnosti *alignment-baseline.*

Drop-initial-after-adjust, nastavuje základní srovnávací bod zarovnání pro iniciálu.

- Hodnoty: *central* – bod zarovnání je průsečíkem konce okraje a centrální základnou. *middle* - bod zarovnání je průsečíkem konce okraje a střední základnou. *after-edge* - bod zarovnání je průsečíkem konce okraje a zadního okraje rozšířeného boxu. *text-after-edge* - bod zarovnání je průsečíkem konce okraje a základny *text-after-edge. ideographic* - bod zarovnání je průsečíkem konce okraje a ideografickou základnou. *alphabetic* - bod zarovnání je průsečíkem konce okraje a alfabetickou základnou. *hanging* - bod zarovnání je průsečíkem konce okraje a zavěšenou základnou. *mathematical* - bod zarovnání je průsečíkem konce okraje a matematickou základnou. A posledními dvěmi hodnotami jsou procentuální a číselné vyjádření zarovnání.

Drop-initial-before-align , nastavuje sekundární bod zarovnání pro iniciálu.

- Hodnoty: *caps-height* – zarovnání podle velikosti kapitálek. A druhou možnou hodnotou je hodnota vlastnosti *alignmetn-baseline*

Drop-initial-before-adjust nastavuje také sekundární bod zarovnání pro iniciálu. Tato vlastnost má však smysl, jen tehdy, když hodnota vlastnosti *drop-initial-size* je nastavena na *auto.*

- Hodnoty: *before-edge* - bod zarovnání je v průsečíku začátku okraje a částí před počátkem okraje. *text-before-edge* bod zarovnání je v průsečíku začátku okraje a prvkem *text-before-edge. middle* - bod zarovnání je v průsečíku začátku okraje a středu základny daného elementu. *central* - bod zarovnání je

v průsečíku začátku okraje a střední základně danného elementu. *after-edge* - bod zarovnání je v průsečíku začátku okraje a za okrajem prvku. *text-after-edge* - bod zarovnání je v průsečíku začátku okraje a základně *text-after-edge. alphabetic* – bod zarovnání je v průsečíku začátku okraje a alfabetickou základnou *hanging* - bod zarovnání je v průsečíku začátku okraje a zavěšené základně. *ideographic* – bod zarovnání je v průsečíku začátku okraje a ideologické základní čáře *mathematical* – bod zarovnání je v průsečíku začátku okraje a základně.

22 CSS3 MODUL RUBY

Ruby je obecně používaný název pro běh textu, které se objevuje v bezprostřední blízkosti jiného toku textu. Vztahuje se k základně a slouží jako poznámka nebo průvodce výslovností spojený s tokem textu. Tento modul právě popisuje použití ruby. Nebudu zde ale popisovat, či vysvětlovat chování a používání ruby. Tak jako v mnoha předešlých modulech (především text a font), je to otázka profesionální sazby. A vysvětlování tohoto není mým úkolem. Já pouze uvedu vlastnosti, které lze použít a jaký to bude mít vliv na zobrazení. Ti kdo chtějí vědět něco více, musí nastudovat publikace týkající se profesionální sazby a zde se dozví, jak je použít při tvorbě internetových stránek.

22.1 POZICOVÁNÍ RUBY, VLASTNOST ruby-positition

Tato vlastnost je používána rodičovskými prvky se zobrazením *ruby-text,* kontrolující pozici ruby textu.

- Hodnoty: *before* – ruby text se objevuje před základnou psaného text. Tento příklad je nejběžnější nastavení použití v jazycích Východní Asie. *after* – ruby text se objeví až za základnou daného textu. Toto je relativně vzácné nastavení pro použití psaných systémů ve Východní Asii. *right* – ruby text se objeví napravo od základny napsaného textu. Na rozdíl od hodnot *before* a *after*, tato hodnota se nevztahuje k směru toku textu.
- Implicitní hodnota: *before*
- Použití na rodičovské prvky ruby textu.
- Dědičnost: ano

22.2 ZAROVNÁNÍ RUBY: ruby-alignment

Tato vlastnost se dá použít na jakékoliv elementy, kontrolující tok textu.

- Hodnoty: *auto* – zarovnání ruby textu bude automatické, podle určeného prohlížeče. *start, left* – obsah ruby textu je zarovnán se začátkem okraje.

center – obsah ruby textu je vycentrován v šíří základny. *end, right* – ruby text bude zarovnán ke konci okraje základny. *distribute-letter* – jestliže ruby text je menší, než základna, pak obsah ruby textu je rovnoměrně rozložen přes šíři základny, s prvním a posledním písmem textu glyfů, je lemováno nahoru s odpovídajícím prvním a poslední základem ozdobné vertikální drážky.. *distribute-space* - jestliže je ruby text menší, než základna, pak obsah ruby textu je rovnoměrně rozložen přes šíři základny s jistým množstvím mezer předcházejících prvnímu a následující poslednímu znaku v ruby textu. *line-edge* – jestliže ruby text nepřiléhá k lince okraje, je obsah zarovnán stejně jako při hodnotě *auto.* Jestliže je přiléhající k lince okraje, pak je ten obsah stále uspořádán jako při *auto,* ale jestliže strana se dotýká konce linky, je obsah seřazen s odpovídajícím okrajem základny.

22.3 VLASTNOST ruby-overhang

Tato vlastnost určuje zda a na které straně, má ruby text povoleno čnět, jakéhokoliv přilehlého textu, navíc k tomu jeho vlastního základu, kdy je ruby text širší než základ ruby textu.

- Hodnoty: *auto* – ruby text může přesahovat text k základně na druhé straně. *start* – text může přesahovat text, který mu předchází. *end* - text může přesahovat text, který za ním následuje. A hodnota *none* – ruby text nemůže přesahovat žádný text, své základny.
- Implicitní hodnota: *auto*
- Lze aplikovat na rodičovské prvky zobrazující ruby text
- Dědičnost: ano.

22.4 VLASTNOST ruby-span

Tato vlastnost kontroluje položené chování poznámky elementů. Dovoluje ostatním jazykům než je XHTML, specifikovat právě toto chování, se zobrazením ruby textu.

- Hodnoty: *attr(x)* – hodnota atributu x, je typu string. Tato hodnota string je vyhodnocena, jako číslo, určující číslo prvků, základny ruby, které jsou zahrnuty prvků poznámek. A hodnota *none*, neurčuje žádné zahrnutí výčtu.
- Implicitní hodnota: *none*
- Aplikuje se na prvky se zobrazením ruby textu
- Dědičnost: ne

Následující příklad ukazuje ukázku XML s použitím vlastnosti *display,* hodnot asociovaných se strukturou ruby a vlastností *span-ruby.*

```
myruby { display: ruby-base; }
myrbc  { display: ruby-base-container; }
myrb   { display: ruby-base; }
myrtc  { display: ruby-text-container; }
myrt   { display: ruby-text; ruby-span: attr(rbspan); }
...
<myruby>
  <myrbc>
    <myrb>05</myrb>
    <myrb>10</myrb>
    <myrb>2003</myrb>
  </myrbc>
  <myrtc>
    <myrt>Měsíc</myrt>
    <myrt>Den</myrt>
    <myrt>Rok</myrt>
  </myrtc>
  <myrtc>
    <myrt rbspan="3">Lhůta vypršení</myrt>
```

```
  </myrtc>
</myruby>
```

23 FILTRY A ZVUKOVÉ STYLY

Ti, kdo již znají kaskádní styly a dočetli až na tuto stránku, čekají, že se dozvědí, něco nového o oblíbených filtrech a poměrně nových zvukových stylů, které se někdy nesprávně zařazují právě k CSS3. Musím vás ale zklamat, protože v době, kdy dokončuji tuto práci, konsorcium w3c nevydalo žádný návrh, týkající se právě filtrů, či zvukových stylů. Ale vzhledem k tomu, že celý level 3, je pouhým návrhem a stále se na něm pracuje, je možné, že se ještě nějaké nové moduly nebo alespoň vlastnosti objeví.

24 ZÁVĚR

Když se začali používat kaskádní styly, nastal konec fádním šedivým stránkám s tabulkami, jednobarevným písmem, stejně značených odkazům apod. Kaskádní styly přinesly nový směr pro formátování webovských stránek a to pomohlo, jak tvůrcům těchto stránek, kteří se snaží o originalitu, pestrost a poutavost, která má lidi upoutat na první pohled, tak jejich návštěvníkům, kteří hodnotí nejdříve vzhled. Formátování pomocí stylů také nevyžaduje žádné hluboké studování nějakého složitého programovacího jazyka. Pomocí příruček a citu pro estetické vnímání lze „vykouzlit“ opravdu oku lahodící stránky během pár hodin.

Technologie CSS 3 řeší problémy profesionální sazby textu. Což je např. vhodné pro tvorbu oficiálních dokumentů, které se mohou stát přímo součástí stránek a není třeba na ně dávat odkaz a umisťovat na server např. ve formátu PDF. Zabývá se také formátováním znaků pro jiné jazyky, než je latinky.

Level 3, kaskádních stylů se od předchozích výrazně liší, především tím, že veškeré vlastnosti zahrnul do samostatných modulů, což má zrychlit práci a především hledání. Lidé, kteří však již s předchozími levely pracovaly, budou zpočátku rozdělením do modulů spíše zmateni, protože syntaxi některých vlastností budou hledat v jiných modulů. Především se jim budou plést moduly background a color, border a box, kam jsou vlastnosti roztříděny podle toho, kam je tvůrci zařadili, zatímco v předchozích verzích byly slučovány ke stejným vlastnostem.

Myslím si, že tyto nejnovější kaskádní styly mají některé již programátorské záležitosti, jakými jsou vypočtené hodnoty a používání dědičnosti. Není to pouze hraní si s barvami, popřípadě se zvukem. Tím se stávají mocným nástrojem pro ty, kdo je budou plně ovládat a tvořit tak profesionální stránky nejen z pohledu funkčnosti, ale i vzhledu a formátování.

Pro ty, kteří s tvorbou stránek začínají a budou chtít používat jen level 3 bez znalosti předchozích, budou tyto začátky nepoměrně těžší. Ale to samozřejmě také

záleží na tom, zda člověk tvořící tyto stránky má alespoň letmou povědomost o programování a rozumí kódu.

CSS3 mě osobně příliš nenadchly, mají sice spoustu nových vlastností, ale v některých případech mi přijde, že je to spíše na škodu věci. Dokument se dá sice zformátovat opravdu dokonale, ale většina lidí tyto vlastnosti vůbec nevyužije, stránky, které jsou přeplněné textem, mohou být sice dokonale upravené, ale na jejich čitelnosti to stejně nepřidá. Myslím si, že se při úpravách kaskádních stylů, autoři měli více zaměřit na tvorbu nových filtrů pro obrázky a také se věnovat médiím. Takové vlastnosti by byly využívány daleko hojněji, ale právě u těchto nenastala téměř žádná změna, jen u médií jsou nějaké malé změn. Je možné, že před tím, než bude CSS3 schváleno a důkladně připraveno pro použití, ještě nějaké moduly vzniknou. Ale v době, kdy dokončuji tuto práci, W3C nemá připraven ani návrh pro nějaké změny těchto vlastností nebo snad nový modul, týkající se filtrů. Ale zde jde pokrok tak rychle, že se jistě brzy nějakých změn dočkáme.

Havelka, Jiří a **Sedlář**, Radek: ***Vytváříme WWW stránky a spravujeme moderní Website, 4. aktualizované vydání.*** Praha, vydavatelství a nakladatelství Computer Press, 2000. ISBN 80-7226-293-9.

Kros, Zbyněk: ***Tvorba internetových aplikací kaskádovými styly, diplomová práce***. České Budějovice, Soukromá vyšší odborná škola a Obchodní akademie, s.r.o., 2001.

Pexa, Petra: ***Tvorba www a wap, kompletní referenční příručka***. České Budějovice, nakladatelství KOPP, 2001. ISBN 80-7232-161-7

Staníček, Petr: ***CSS Kaskádové styly.*** Praha, nakladatelství Computer Press, 2003. ISBN 80-7226-872-4

Webové stránky konsorcia W3C: ***http://www.w3.org***

Webové stránky Interval CZ: ***http://www.interval.cz***