

Informační zdroje: www.jakpsatweb.cz
 kniha: Vytváříme www stránky

1. Tvorba WWW stránek

- WWW stránky prezentované na Internetu jsou programovány v jazyce HTML (HyperText Markup Language). Internetové prohlížeče (př. Explorer) přeloží zdrojový kód a dokáží je zobrazit.

HTML dokument – běžný textový soubor s příponou *.htm, *.html

1. **Tvorba:** 1. Běžný textový editor
2. HTML editor - speciální program pro tvorbu stránek

Wysiwyg editory - tvorba vzhledu, př. jako ve Wordu, sporná kvalita vytvořeného kódu (tvoří se sám)
př. Microsoft FrontPage, Dreamweaver, Adobe GoLive nebo NVU

Strukturní editory – přímá tvorba kódu, ale program napovídá, doplňuje apod.
př. HomeSite, UltraEdit, české EasyPad, HTML Editor a PSPad

2. Základní struktura uložení:

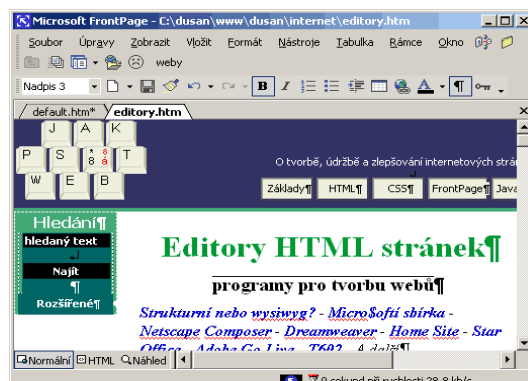
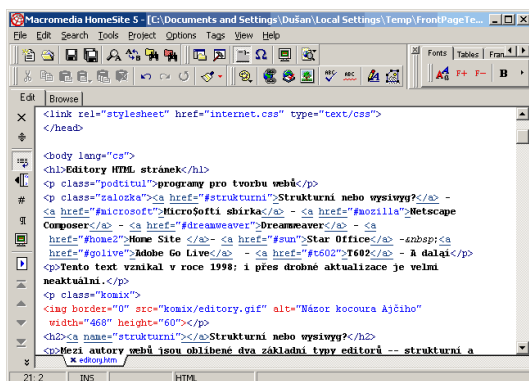
- 📁 Složka HTML - 📁 Složka image (picture) – obrázkové přílohy
- 📄 index.htm - první (hlavní) stránka
- 📄 str1.htm - stránka odkazu
- 📄 str2.htm - atd.

3. Základní struktura dokumentu:

Tagy – značky podle kterých prohlížeč překládá a zobrazuje stránky

- a) **párové** – začátek a konec př. Tento text bude tučný!
- b) **nepárové** – pouze jeden př. <P> - ukončí odstavec
-
 - zalomení řádku

```
<HTML>
<HEAD>
<TITLE>Moje WWW stránka</TITLE>
</HEAD>
<BODY>
- vlastní obsah WWW stránky
</BODY>
</HTML>
```



2. Zápis tagu

Popis	Vývoj příkladu
Značka tagu začíná levou ostrou závorkou.	<
Za ní následuje jméno tagu, před kterým nesmí být mezera.	<font
Mohou následovat atributy. Před každým musí být alespoň jedna mezera.	<font color
Za atributem se píše rovnítko a hodnota v uvozovkách. Vše bez mezer.	<font color="blue"
Atributů může být několik.	<font color="blue" size="6"
Značka končí pravou ostrou závorkou.	
Následuje vlastní text, který se zobrazí	Modrý velký text
Element končí ukončovací značkou s lomítkem a bez atributů	Modrý velký text

3. Základní pravidla (syntaxe) tvorby HTML

HTML má několik málo zásad, které je dobré zmínit.

- Nezáleží na velikosti písem, <body> je totéž co <BODY>
- V adresách a jménech souborů záleží na velikosti písmen, nesmějí tam být mezery a čeština.
- Tagy, které prohlížeč nezná, jako by nebyly.
- Na začátku tagu nesmí být mezera, třeba < br> je špatně.
- Dvě mezery po sobě (nebo víc) mají stejný význam jako jedna mezera.
- Konec řádku ve zdroji se chápe jako mezera.
- Jména atributů je dobré dávat vždy do uvozovek, ale není to úplně nutné, pokud uvnitř nejsou mezery.
- Speciální znaky typu © se do zdroje zadávají jako posloupnost znaků *&něco*; například *©*; , pevná mezera je * *;
- Poznámka se do zdroje vkládá mezi značky <!-- a --> <!-- Toto je poznámka, která se nezobrazí. -->
- Stránky by měly být jednoduché, přehledné, srozumitelné a s určitými estetickými zásadami (formátování, barevné kombinace).
- Je nutno dodržovat přehlednost zdrojového kódu (mezery, tabulátory).
- Tagy lze do sebe většinou libovolně vnořovat a kombinovat.

4. Formátování těla dokumentu

Počáteční tag: <BODY atribut="hodnota" atribut="hodnota">
 -tělo dokumentu

Koncový tag: </BODY>

1. Vložení obrázku na pozadí dokumentu:

 <BODY background="obrazek.gif">

2. Nastavení barvy na pozadí dokumentu:

 <BODY bgcolor=green> - red, blue, gold aj.

3. Definice barvy textu

 <BODY text=yellow> - red, blue, gold aj.

4. Definice barev hypertextových odkazů:

 <BODY link=black> - barva odkazu, standardní je modrá

 <BODY vlink=black> - barva navštíveného odkazu

 <BODY alink=black> - aktivní odkaz (na který jsi klikl myší)

5. Definice okraje:

 <BODY leftmargin=20> - levý okraj (right, top, bottom)

5. Možnosti zápisu barvy

Barva se dá v HTML zapsat pěti způsoby:

Způsob zápisu	Příklad: červené písmo	Poznámka
<i>Jménem</i> v angličtině		Existuje mnoho "pojmenovaných" barev.
<i>Procentuálním</i> RGB zápisem		rgb znamená Red Green Blue (v prohlížečích Opera a Mozilla to funguje jenom v CSS)
<i>Desetiným</i> RGB zápisem		
<i>Šestnáctkovým</i> RGB zápisem		Tento způsob je nejjistější, nejpoužívanější a nejlepší.
<i>Zkráceným šestnáctkovým</i> RGB zápisem		Jenom v případě, že se všechny dvojice cifer shodují

6. Formátování odstavce

1. Zalomení řádku

 - zalomení řádku (text začne na novém řádku, jako klávesa Enter)

2. Zalomení odstavce

<P> - ukončí odstavec, podobný jako
, ale navíc vynechá jeden řádek

<P ALIGN="right"> - následující odstavec je zarovnán vpravo (center, left, justify)

3. Centrování odstavce

<CENTER>Tento text bude na stránce vycentrován na střed</CENTER>

5. Formátování textu

- značení vychází z běžných textových editorů

1. Druh písma

- jednotlivé tagy lze do sebe vnořovat a libovolně kombinovat

Tučné	 ...
<i>Kurziva</i>	<I> ... </I>
<u>Podtržené</u>	<U> ... </U>
Přeškrtnuté	<S> ... </S>
Dolní index	_{...}
Horní index	^{...}
Blikající text	<BLINK> ... </BLINK>

2. Barva, velikost a typ písma

Tento text bude formátován podle nastavení parametru tagu FONT.

FACE	typ písma (př. Times New Roman, Arial, Courier New)
SIZE	velikost písma (od -7 do 7)
COLOR	barva písma

3. Nadpisy

- šest předdefinovaných velikostí nadpisů

<H1>Nadpis první úrovně </H1> - největší nadpis

<H2>Nadpis druhé úrovně </H2>

<H6>Nadpis šesté úrovně </H6> - nejmenší písmo

4. Oddělovací čára <HR>

Další parametry:	<HR SIZE=5> (rozmezí 1-10)	- šířka čáry
	<HR WIDTH=120> (hodnota v bodech)	- délka čáry
	<HR WIDTH=50% > (hodnota v %)	
	<HR ALIGN="center" >	- zarovnání
	<HR NOSHADE >	- bez stínu

6. Tvorba seznamů

1. Seznam s odrážkami

- položky jsou odděleny grafickým znakem

```
<UL TYPE=disc>          - začátek seznamu
  <LI>první položka </LI>
  <LI>druhá položka </LI>
  <LI>třetí položka </LI>
</UL>                   - ukončení seznamu
```

Parametr k tvaru odrážky:	<UL TYPE="disc">	- ● kolečko
	<UL TYPE="circle">	- ° kroužek
	<UL TYPE="square">	- ■ čtvereček

2. Číslovaný seznam

- automaticky dosadí číslo a doplní řadu

```
<OL>                    - začátek seznamu
  <LI>první položka </LI>
  <LI>druhá položka </LI>
  <LI>třetí položka </LI>
</OL>                  - konec seznamu
```

Parametry k číslovaným seznamům:

<OL TYPE=A>	- typ řady (arabská abeceda – a, A; čísla: římská – i, I; arab. – 1)
<OL START=5>	- počáteční hodnota seznamu (př. od čísla 5)
<OL CONTINUE>	- pokračování číslování z předešlého seznamu
<OL ALIGN="right">	- zarovnání seznamu

7. Tvorba tabulek

- mocný nástroj jazyka HTML, do buněk lze vkládat kromě textu také odkazy, obrázky aj., často se používají pro vzhled celé struktury dokumentu.

Příklad:

```
<TABLE>                - začátek tabulky
<TR><TH>záhlaví </TH></TR>    - zvýraz.
<TR><TD>buňka </TD> </TR>    - 1. řádek
<TR><TD>buňka </TD> </TR>    - 2. řádek
<TR><TD>buňka </TD> </TR>    - 3. řádek
</TABLE>                - konec tabulky
```

Parametry k tagu <TABLE>:

```
<TABLE BORDER=3>        - tloušťka rámečku
<TABLE WIDTH=60%>      - šířka tabulky (v % nebo bodech př. WIDTH=400)
<TABLE BGCOLOR="blue"> - barva pozadí tabulky
<TABLE CELSPACING=5>   - vzdálenost mezi buňkami (v bodech)
<TABLE CELLPADDING=3> - vzdálenost textu v buňce od okraje
```

Parametry k tagům <TR> <TD> <TH>:

```
ALIGN="left"            - horizontální zarovnání obsahu buňky (center, right, justify)
VALIGN="middle"         - vertikální zarovnání obsahu buňky (top, middle, bottom, baseline)
ROWSPAN=2               - sloučení (prodloužení) buněk v řádku
COLSPAN=3               - sloučení (prodloužení) buněk v sloupci
```

8. Tvorba odkazu (hyperlinku)

str. 134

Odkaz (hyperlink) – základní prvek WWW stránek, může to být text, obrázek aj., většinou bývá podtržen, je barevně odlišný a kurzor se při najetí na něj změní (nejčastěji ruka s prstem).

1. Internetový odkaz na jinou stránku

a) odkaz na jinou Internetovou stránku:

`Celý tento text je odkaz na portál seznam.cz `

b) odkaz na jinou svou vlastní stránku:

`Celý tento text je odkaz na svoji další stránku uloženou v souboru autor.htm `

2. Odkaz v rámci dokumentu

a) adresace části dokumentu:

`Celý tento text je adresa kde se zastaví vyhledávání `

b) odkaz na adresovanou část:

`Tento odkaz nalistuje stránku na adresovanou část Kapitola 10 ` - symbol # je nutný (angl. klávesnice, SHIFT+3)

`target="_blank"` - odkaz se načte do nového okna

`title="Odkaz"` - pojmenování bubliny při najetí myší na odkaz

3. Odkaz na externí soubor

- jakýkoliv soubor př. rar, doc, xls aj., prohlížeč se zeptá na stažení nebo otevření př.

`Celý tento text je odkaz na externí soubor tabulka.xls `

9. Vložení obrázku

str. 135-137

- grafické objekty jsou na stránkách WWW neodlučnou součástí, je však třeba dbát na rychlost načítání, která se snižuje velikostí souborů.

Používané formáty:

- *.gif - některé lepší vlastnosti př. průhlednost, postupné načítání
- *.jpg - větší komprese (menší soubory)

Parametry tagu IMG:

SRC= pic/foto.jpg	- adresa umístění obrázku (zdroje)
ALT="moje fotka"	- text pokud se obrázek nemůže zobrazit (př. zakázáno)
ALIGN=bottom	- spodní hrana obr. je zarovnána na spodní řádek
ALIGN=top	- horní hrana obr. je zarovnána na horní řádek
ALIGN=middle	- střed obr. je zarovnan podle textu
ALIGN=left	- obr. je zarovnan na levý okraj, text obtéká
ALIGN=right	- obr. je zarovnan na pravý okraj, text obtéká
WIDTH=80	- změna šířky obr. v pixelech nebo %
HEIGHT=120	- změna výšky obr. v pixelech nebo %
BORDER=5	- definuje šířku rámečku okolo obrázku

Obrázek jako odkaz - kombinace odkazu a vložení obrázku

10. Rámy (frames)

Rozdělují okno prohlížeče na několik částí. V každé části se může zobrazit jiná stránka (adresa).

1. Rozložení rámu

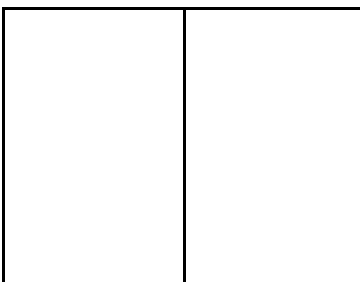
Struktura html dokumentu s rámy je odlišná od běžného dokumentu. Obsahuje definici **<FRAMESET>**. Atributy:

ROWS a COLS - určují na počet řádků a sloupců rozdělení obrazovky. Velikost se zadává v pixelech nebo v procentech z celkové výšky (šířky).

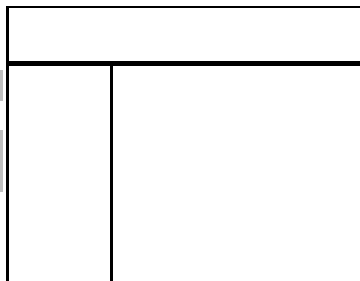
```
<FRAMESET ROWS="50%, 50%">  
<FRAME SRC="prvni.html">  
<FRAME SRC="druha.html">  
</FRAMESET>
```



```
<FRAMESET COLS="50%, 50%">  
<FRAME SRC="prvni.html">  
<FRAME SRC="druha.html">  
</FRAMESET>
```



```
<FRAMESET ROWS="20%, 80%">  
<FRAME SRC="prvni.html">  
<FRAMESET COLS="30%, 70%">  
<FRAME SRC="druha.html">  
<FRAME SRC="treti.html">  
</FRAMESET>  
</FRAMESET>
```



2. Obsah rámu

<FRAME> - určuje, které html dokumenty se mají v rámech zobrazovat, a to pomocí atributu **SRC**.

Atributy tagu <FRAME> :

NORESIZE - atribut, který se postará o to, aby nešla velikost rámu měnit

SCROLLING=NO/YES/AUTO - pokud nastavíme na hodnotu NO, nepůjde se po obsahu rámu pohybovat

FRAMEBORDER=0 – velikost rámečku (0 = úplné zrušení)

MARGINWIDTH=x - vzdálenost obsahu rámu od pravého a levého okraje

MARGINHEIGHT=x - vzdálenost obsahu od horního a dolního okraje

NAME - zadávání jména rámu, použití: odkaz v jednom rámu se zobrazuje v jiném rámu (atribut

TARGET="jméno rámu", u odkazu)

Plovoucí rámy (IFRAME)

- objekt v podobě rámu kdekoliv v HTML dokumentu, do kterého se načítá libovolná adresa (html soubor)
- je definován kdekoliv v části BODY
- jeho velikost se určuje v pixelech, podobně jako obrázek

př. <IFRAME SRC=soubor.html ALIGN="left" HEIGHT=640 WIDTH=480>

Atributy pro tag IFRAME:

SRC	- zdroj pro obsah okna (adresa internetu, soubor.html)
ALIGN	- zarovnání
BORDER, BORDERCOLOR	- rámeček (šířka, barva)
HEIGHT, WIDTH	- velikost (výška, šířka) v pixelech (rozlišení)
NORESIZE="noresize" (resize)	- možnost měnit velikost tažením okraje
NAME	- pojmenování pro funkci TARGET v odkazech

další: SCROLLING, TITLE, MARGINHEIGHT, MARGINWEIGHT ...

Tagy a <div>

Někdy je potřeba zformátovat kus textu, který není vymezen žádným konkrétním tagem. Proto se tam vloží nový tag. Zahrnuje-li formátovaná oblast více odstavců, použije se párový tag <div>, v rámci jednoho odstavce se používá , protože <div> by to roztrhal do více odstavců. Například:

```
<body>
... <!--normální odstavce -->
<div style="color: maroon">
... <!-- mnoho různých odstavců, všechny budou hnědé -->
</div>
...<!-- a už je to zase normál -->
```

<p>Normální text a text kurzívou a zase normální text.</p>
se zobrazí takto:

Normální text a *text kurzívou* a zase normální text.

- div je element blokový - **před sebou a za sebou udělá konec řádku**
- span je element řádkový - může se vyskytovat **v jednom řádku**

Otázky k testu: Tvorba WWW:

1. B:Vyjmenuj prohlížeče WWW?
2. A:Charakterizuj sekci HEAD a jeho nastavení?
3. B:Tagy a atributy pro tvorbu tabulky?
4. A:Tagy a atributy pro formátování textu?
5. B:Rámy (frameset)?
6. A:Vyjmenuj a rozděl programy pro tvorbu WWW?
7. B:Charakterizuj sekci BODY a jeho nastavení?
8. A:Tagy a atributy pro tvorbu seznamů?
9. B:Tagy a atributy pro odkazy?
- 10.A:Vkládání obrázků, tagy, nastavení?

PRÁCE SE STYLY CSS (Cascading Style Sheets)

- způsoby (metody) pro grafickou úpravu HTML stránek
- používá tag `<style>` a obecný atribut "style", kterému se přiřazuje nějaká definice.

Možnosti CSS:

- přesné nastavení požadovaného písma (nadpisů), a odstavců - formát, odsazení, řádkování
- zvýrazňování odkazů, tvorba různých grafických odrážek
- předefinovat grafický význam běžných tagů (například všechno, co je kurzívou, udělat i tučně)
- nastavit pozadí čehokoliv, stránky, tabulky ale třeba i odstavce
- přidat k čemukoli rolovací lišty, oříznout, orámovat, nastavit okraje

Použití CSS

1. **Přímý styl** - přímo v textu zdroje u formátovaného elementu pomocí atributu `style="..."`.
Nejméně časté použití.

`<p style="color: red">Tento odstavec bude červený.</p>`

2. **Pomocí "stylopisu"** (angl. "stylesheet") v **hlavičce stránky**. Je v něm obecně napsáno, co má být jak zformátováno, například že nadpisy mají být zelené. Do stránky se stylopis píše mezi tagy `<style>` a `</style>`.

```
<style>
p {color: red}
</style>
```

3. **Použitím externího stylopisu** - to je **soubor *.css**, na který se stránka odkazuje tagem `<link>`. V souboru je umístěný stylopis. Hlavní výhoda je v tom, že na jeden takový soubor se dá nalinkovat mnoho stránek, takže pak všechny vypadají podobně.

Vytvoří se soubor, který se pojmenuje třeba `styly.css`. V něm bude pouze tento text:

```
p {color: red}
```

Do hlavičky html dokumentu, který chci stylem ovlivnit, musím napsat odkaz na tento soubor:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

Zápis stylopisu - syntaxe

CSS nejsou součástí HTML, a tak se zapisují zcela jiným způsobem.

Je nutné si všimnout, kde se používají uvozovky, dvojtečky, složené závorky, středníky a čárky.

Příklad správného zápisu:

```
h2 {color: green; background-color: yellow}
```

Zápis neovlivňuje: mezery a konce řádků, velikost písmen

Chyba zápisu způsobí pravděpodobně nefunkčnost.

Hodnoty, které prohlížeč nezná, ignoruje.

Příklady:

Příklad přímého stylu	<pre><p style="color: red;">text červeného odstavce</p></pre>
Příklad stylopisu	<pre><style> p {color: red} body {background-color: yellow;} </style></pre>
Příklad jednoduchého zápisu vlastností	<pre>color: red</pre>
a složitějšího zápisu vlastností	<pre>font-family: Arial, Arial CE, sans-serif; color: red;</pre>

Tvorba vlastních tagů

a) třídy - class

Text se bude formátovat podle definice ve stylopisu deklarovaného v hlavičce dokumentu popř. v externím souboru. Deklarace třídy (class) začíná tečkou př. `.podtitul`. Ta vyjadřuje, že deklarace se nebude týkat html tagu, ale třídy. Tříd je možno nadeklarovat jakékoliv množství.

Atribut class (třída) se může použít u libovolného elementu (tagu).

Stylopis:

```
<style>
.podtitul {text-align: center; font-weight: bold; text-decoration: overline}
/* zarovnání na střed, tučné písmo a nadtržení*/
</style>
```

zápis v těle dokumentu:

```
<p class="podtitul">Text podtitulu</p>
```

Element se stejnou class se v dokumentu může vyskytovat mnohokrát (na rozdíl od ID - identifikátorů).

b) Identifikátory

Pro jednoznačný popis nějakého elementu (zejména pro potřeby [skriptů](#)) existuje univerzální atribut **ID**. I jemu se může ve stylopisu přiřadit nějaká deklarace, ale na rozdíl od třídy nezačíná tečkou, ale **dvojkřížkem #**.

V těle dokumentu by se element s jedním identifikátorem měl vyskytovat jenom jednou.

```
#podtitul { text-align: center; font-weight: bold; text-decoration: overline}
```

a v těle by se odstavci přiřadila identifikace atributem **id**:

```
<p id="podtitul">Text podtitulu</p>
```

Identifikátor id se z hlediska CSS chová stejně jako třída class.

Více tříd pro jeden prvek

Pokud chceme, aby prvek stránky přebíral formátování dvou tříd, pak je stačí uvést obě v atributu class a oddělit je mezerou. Ve většině případů je lepší spojit deklaraci do jedné třídy, ale vyskytují se stránky, kde ostatní prvky chcete „oclassovat“ odděleně.

Styl může například vypadat takto:

```
<style>
.zlutepozadi {background-color: yellow;}
.vlevo {float: left; width: 150px;}
</style>
```

Prvek dostane obě třídy:

```
<div class="zlutepozadi vlevo">
```

obsah prvku

```
</div>
```

Složené deklarace

Hromadná deklarace

Styloписy umožňují nadeklarovat vlastnosti **pro více elementů najednou**. Například deklarace

```
H1, H2, H3 {color: green}
```

obarví všechny nadpisy první, druhé i třetí úrovně na zeleno. Hromadnou deklaraci používám, pokud zadávám mnoho stejných vlastností pro mnoho elementů. Důležitá je **čárka** mezi selektory! Kdyby tam nebyla, šlo by o něco jiného, totiž o *kontextový selektor*.

Kontextová deklarace

Vysvětlím příkladem:

```
H3 A {font-style: italic}
```

Tato deklarace by udělala kurzívou všechny **odkazy uvnitř nadpisů třetí úrovně** (elementy **A** uvnitř elementu **H3**).

```
<h3>Obyčejný text nadpisu s <a>odkazem kurzívou</a></h3>
```

```
<p>Obyčejný odstavec s <a>obyčejným odkazem</a></p>
```

Odkazy v obyčejných odstavcích to nijak neovlivní, stejně tak obyčejný text trojkového nadpisu. Zápisy selektorů kontextového zápisu jsou odděleny pouze mezerou.

Pseudoelementy

Ve specifikaci CSS1 se vyskytují pseudoelementy `:first-line` a `:first-letter`. Znamenají první řádek a první písmeno. Například zápis:

```
p:first-line {color: blue}
```

```
p:first-letter {font-size: 200%}
```

by měl způsobit, že odstavce budou mít první řádek modrý a první písmeno dvakrát větší.

Z prohlížečů to podporují Mozilla 5, Internet Explorer 5.5 (nikoliv IE 5.0) a IE 6 (myslím ale, že Internet Explorer to umí jenom u tagu `<p>`).

Shrnující příklad

Ve stylopisu (v hlavičce či v externím souboru) se mohou vyskytovat i takovéhle věci:

```
<style>
#prvniodstavec {text-indent: 20 px}
A:visited {color: teal}
A:link {color: navy}
a:hover {color: red}
.velke A {font-weight: bold}
.zalozka {font-style: oblique}
.zalozka A:visited {color: navy ! important}
H1, H2 {color: #33ff66; font-variant: small-caps}
H2 {font-size: 18pt}
</style>
```

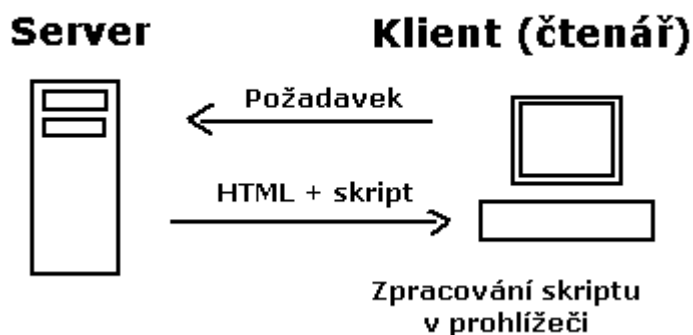
Doufám, že význam je zřejmý. Všechno to funguje. Důležité je v příkladu snad jen to, že se může jeden element deklarovat vícekrát a že kontextová deklarace se může kombinovat s třídami a pseudotřídami.

ÚVOD DO JAVASCRIPTU

JavaScript - programovací jazyk, který se používá v internetových stránkách. Zapisuje se přímo do HTML kódu mezi tagy: `<script></script>` nebo do externího souboru.

JavaScript je klientský skript. To znamená, že se program odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván. (Protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta jdou už jen výsledky.)

Klientský skript



Existují i jiné jazyky klientských skriptů, například VBScript. Jsou ale tak málo používané, že když se dnes mluví o "skriptech", myslí se tím JavaScripty.

JavaScript není Java

JavaScript je často zaměňován s Javou. Java je samostatný programovací jazyk. Má s JavaScriptem pouze podobnou syntaxi.

Příklad použití:

```
<body>  
Toto je normální text stránky.<br>  
<script>  
document.write("A toto napsal JavaScript");  
</script>  
</body>
```

Způsoby zápisu Javascriptu

1. Pomocí tagu <script> normálně do dokumentu

```
<script>  
  alert('Hlavu vzhůru, bude hůř!');  
</script>
```

Skript se může objevit kdekoliv v HTML kódu - jak v hlavičce, tak v těle dokumentu. Prohlížeč pak skript zpracovává okamžitě, jakmile na něj narazí.

2. Tagem <script> s odkazem na externí soubor a potom ho do stránky načítat

Příklad: Do souboru `externi_skript.js` uložím toto:

```
document.write("Toto napsal externí Java Script");
```

Do stránky, do které pak chci skript vložit, vložím tento kus HTML kódu – odkaz na externí soubor:

```
<script src="externi_skript.js"></script>
```

3. In-line (řádkový) zápis jako atribut tagu bez použití tagu <script>

Nevyužívá tag `<script>`, ale zapisuje se jako atribut jiného tagu. Většinou reaguje na nějakou uživatelskou událost.

Příklad:

```
<a href="http://www.meuslany.cz" onmouseover="alert('Na úřadě je myš!!!!')">Městský úřad Slaný</a>
```

Atribut `onmouseover` - znamená "při přejetí myší" - při přejetí myší se spouští **ten skript**, co je zapsán jako hodnota atributu.

Všimněte si, že v in-line zápisu je nutno používat apostrofy místo uvozovek (protože uvozovky by zakončily zápis skriptu).

V naprosté většině složitějších skriptů se tedy způsoby kombinují, protože je to efektivní.

Nejčastější způsob kombinace:

- externím skriptem jsou definovány funkce
- normálním zápisem (pomocí `<script>`) jsou inicializovány proměnné a startovní funkce
- in-line skripty volají funkce podle událostí v závislosti na reakcích uživatele

Proměnné

- Názvem proměnné může být libovolné slovo (s výjimkou několika klíčových slov jazyka). Záleží na velikosti písmen. V JavaScriptu je dobré proměnné deklarovat. Na deklarování slouží klíčové slovo `var` následované výpisem použitých proměnných. Ale nemusí se to dělat.

Proměnná použitá v příkladu má **jméno** `x`. Do proměnné `x` je načtena hodnota "**Obsah proměnné**", která je dalším příkazem `document.write` zapsána do proudu dokumentu.

Všimněte si, že v příkazu `document.write(x)` už není to `x` obaleno uvozovkami, protože se jedná o proměnnou. Kdyby tam ty uvozovky byly, tak to vypíše `x` namísto *Obsah proměnné*.

```
document.write(x); /* vypíše Obsah proměnné */  
document.write("x"); /* vypíše x */
```

Hlášky

Alert

Kdyby se JavaScriptem dalo jenom zapisovat do dokumentu, byla by to nuda. Veselým efektem je příkaz `alert`, který zobrazí upozorňovací okénko s textem.

```
<script>  
alert("To ses ale lek!");  
</script>
```

Prompt

Dialogové okno vyžadující vstup uživatele je volán příkazem `prompt`. Umožňuje do proměnné vložit hodnotu zadanou uživatelem. Zapisuje se `proměnná = prompt(hláška , inicializační hodnota)`. Příklad:

```
<body>  
<script>
```

```
x = prompt("Zadej svoje jméno", "");  
document.write("Tvoje jméno tedy je ");  
document.write(x);
```

```
</script>  
</body>
```

Můžete si [příklad zobrazit](#). Do proměnné `x` se načte hodnota pomocí **příkazu** `prompt()`. Ten má dva parametry (zadávané v závorce). Text výzvy a inicializační hodnotu. Inicializační hodnota je v tomto příkladu prázdný řetězec `""`. Skript čeká na vstup uživatele. Po zadání údajů skript běží dál a proměnná se normálně zpracovává (například je vypsaná).

Tolik úvodní příklady. Možná už tušíte, že je JavaScript docela použitelný jazyk. V dalším textu se pokusím vysvětlit další různé způsoby, kterými se skript může začlenit do stránky.

Confirm

Potvrzující dialog. Možnosti jsou ano / ne. Návrátová hodnota je `true` nebo `false`.

```
<script>  
pokracovat = confirm("chcete pokračovat?");  
// nyní mám v proměnné pokracovat uloženo true nebo false  
if(pokracovat) document.write("tak tedy pokračujeme");  
</script>
```

Poznámky

Pokud potřebujete v nějakém vypisovaném textu zalomit řádek, dělá se to pomocí sekvence `\n`.
`alert("první řádek \n druhý řádek");`

Události JavaScriptu

Přehled uživatelských událostí

Události okna a dokumentu:			
Událost	Popis	Lze použít na	Nefunguje v
onLoad	při úplném načtení stránky	body, img	IE3
onUnload	při opouštění stránky	body	IE3
onResize	při změně velikosti okna	body	IE3, NN3
onScroll	při skrolování, posouvání obsahu	body, textarea, cokoli s overflow	IE3

Události myši			
Událost	Popis	Lze použít na	Nefunguje v
onClick	při kliknutí na prvek nebo při přednastavené akci	všechny elementy, v 4. verzích prohlížečů ale jenom některé	
onDbClick	při dvojkliku na prvku		IE3, NN3
onMouseOver	při najetí myši na prvek		IE3
onMouseOut	při odjetí z prvku		IE3
onMouseMove	při pohybu myši nad prvkem		IE3, NN3
onMouseDown	při stisknutí tlačítka nad prvkem		IE3, NN3
onMouseUp	při uvolnění tlačítka nad prvkem		IE3, NN3

Události klávesnice			
Událost	Popis	Lze použít na	Nefunguje v
onKeyPress	při stisknutí klapky ve chvíli, kdy je element aktivní	asi cokoliv	IE3, NN3
onKeyDown	při stlačení klapky ve chvíli, kdy je element aktivní		IE3, NN3
onKeyUp	při uvolnění klapky ve chvíli, kdy je element aktivní		IE3, NN3

[Test onKeyPress](#)

Události formuláře a formulářových polí			
Událost	Popis	Lze použít na	Nefunguje v
onSubmit	těsně před odesláním formuláře, příklad	form	IE3

onReset	při vynulování formuláře tlačítkem reset	form	IE3
onFocus	při aktivaci políčka (okna)	input, select, textarea, window	
onBlur	při deaktivaci políčka (okna)	input, select, textarea, (window nepotvrzeno)	
onChange	při změně hodnoty políčka	input, select, textarea	
onSelect	při vybrání textu myší	body (výběr textu), textarea, input	výběr textu nefunguje v IE3 a NN*

Další události			
Událost	Popis	Lze použít na	Podpora
onAbort	při přerušení načítání	img, body	IE3

onBack - při stisknutí tlačítka "ZPĚT" (funguje pouze v historickém Netscape 4, takže vlastně nefunguje nikde)

Další události pro Internet Explorer 4 a vyšší (možná fungují v Mozille, ale v Netsapu určitě ne):

onCopy - při kopírování do schránky "Ctrl+C"

onCut - při vyjmutí do schránky "Ctrl+X"

onForward - při stisknutí tlačítka "VPŘED"

onHelp - při volání nápovědy, např "F1"

onMouseDown - při pohybu myši se stisknutým tlačítkem

onMouseWheel - při rolování kolečkem (nefunguje v IE 5.0)

onMove - při pohybu okna prohlížeče (nepodařilo se nasimulovat)

onPaste - při vkládání ze schránky "Ctrl+V"

Události okna a dokumentu

Událost	Popis	Lze použít na	Nefunguje v
onLoad	při úplném načtení stránky se všemi objekty	body, img	IE3
onUnload	při opouštění stránky	body	IE3
onResize	při změně velikosti okna	body	IE3, NN3
onScroll	při skrolování, posouvání obsahu	body, textarea, cokoli s overflow	IE3

OnLoad

Událost onLoad nastává ve chvíli, kdy je objekt načten. Praktický význam má pro dokument (tag <body>) nebo pro obrázek (). Podle mých zkušeností nelze vázat na jiné tagy (možná ještě <object>, nevím). Příklad vyhození hlášky ve chvíli načtení dokumentu:

```
<body onload="alert('Dokument je načten')">
```

Událost `onLoad` se spouští až po té, co jsou načteny i všechny vložené objekty, např. obrázky. (Nevím, jestli událost čeká i na načtení obsahů do `<iframe>`, ale asi ano.) V některých návodech se uvádí, že se `onload` spouští ve chvíli, kdy se načte poslední znak zdroje. Není to tak, čeká se i na obrázky.

Velmi praktické je použít událost `onload` také na obrázek. Dejme tomu, že chci nějakou akci spustit až ve chvíli, kdy je obrázek načten. To se hodí, pokud ta akce pracuje s tím obrázkem. Kdyby byla totiž funkce volána jinak, mohla by se spustit ve chvíli, kdy obrázek ještě není ze serveru načtený. Příklad:

```

```

Příklad předpokládá dřívější deklaraci té funkce `animuj()`, ale o to v tuto chvíli nejde. Jde o to, že se pracuje s už načteným obrázkem.

Animované gify spouštějí `onload` chybně při každém zobrazování prvního snímku.

Jako synonymum k události `onload` umí Internet Explorer od 4. verze použít také `onReadyStateChange`. Praktický rozdíl v tom není, snad jen že `onLoad` je událostí okna, kdežto `onReadyStateChange` je událostí dokumentu.

OnUnload

Událost se spouští těsně před opuštěním dokumentu. To může být přechod na jinou stránku nebo zavření okna prohlížeče. Událost se váže na objekt `window` a zapisuje se do tagu `body`:

```
<body onunload="window.alert('Nashledanou!')">
```

V tomto příkladu stránka před zavřením vyhodí hlášku s textem `Nashledanou!`. Daly by se dělat i jiné kousky, například se leckde vidí hláška `"Opravdu chcete odejít z této super stránky?"` Na základě odpovědi se dá i zavírání skriptem zrušit. Vlastnosti `onUnload` na vyhazování hlášek nebo jiných záłudností doporučuji nepoužívat. Je to jako takový křik malého fracka `"mamí nesmíš jít pryč!"`, prostě to považuji za nedůstojné.

Je ale dobré vědět, že vlastnost `onUnload` existuje a v případě potřeby na ní navěsit skript, který nebude uživatele obtěžovat.

OnResize

Událost `onResize` nastane vždy, když se změní velikost okna prohlížeče. Využívá se zejména na stránkách s dynamickým obsahem, které mají elementy vkládány stylem na přesnou pozici. Když se změní velikost okna, je potřeba pozice přepočítat.

`onresize` se dá navázat i na jiné objekty v dokumentu, které mají definovanou šířku. (Pouze v Internet Exploreru, v Mozille mi to nefunguje.) Např.

```
<div onclick="this.style.width = '50px'" onresize="alert('Měním velikost')">Klikni si!</div>
```

Klikni si!

[Jiný příklad.](#)

V některých případech změny velikosti okna se ze záhadných důvodů spouští událost `onresize` několikrát po sobě. Také se spouští ve chvíli, kdy se nějakým skriptem změní šířka dokumentu. Na to je třeba myslet a nevázat na `onresize` akce, které mění velikost dokumentu, protože by mohlo dojít k zacyklení.

OnScroll

Událost `onScroll` nastává obecně při rolování. Dá se použít u všeho, co má rolovací lišty. Nejčastěji se váže na tag `<body>`, protože to je z hlediska skriptování nejpoužitelnější. Pak se dá například přepočítávat pozice elementů, které mají být viditelné, i když se se stránkou posouvá.

[Příklad objektu zafixovaného na stránce.](#)

[Test události onscroll.](#) Událost onscroll neprobublává.

Události myši

Událost	Popis	Lze použít na	Nefunguje v
onClick	při kliknutí na prvek nebo při přednastavené akci	všechny elementy, v 4. verzích prohlížečů ale jenom některé	
ondblclick	při dvojkliku na prvku		IE3, NN3
onmouseover	při najetí myši na prvek		IE3
onmouseout	při odjetí z prvku		IE3
onmousemove	při pohybu myši nad prvkem		IE3, NN3
onmousedown	při stisknutí tlačítka nad prvkem		IE3, NN3
onmouseup	při uvolnění tlačítka nad prvkem		IE3, NN3

OnClick

OnClick se spouští ve dvou případech:

1. jestliže se na element kliklo myši
2. nebo pokud dojde k předdefinované akci elementu.

Předdefinovaná akce elementu nastává zejména při práci s tabulátorem a Entrem -- ťukáním tabulátoru se aktivují odkazy a formulářová políčka. Když se pak dá Enter, odkaz proklikne i bez myši. I v tom případě je ale volána událost onclick.

OnClick se na stránkách bohatě používá.

Při kliknutí pravým nebo prostředním tlačítkem se událost onclick nevyvolá. Tam je potřeba využít události onmousedown nebo onmouseup. Samotné události pak nejsou schopné rozlišit, zda se kliklo **levým nebo pravým tlačítkem**. Události pravého tlačítka se musejí odchytávat z události onmousedown a následnou podmínkou. Informace o tom, které tlačítko bylo stisknuto, se uložená ve vlastnosti event.button. Hodnoty event.button jsou 1 pro levé tlačítko, 2 pro pravé, 4 pro prostřední a případně se to sčítá. Hodnota 3 tedy odpovídá stisknutí pravého a levého tlačítka najednou, nulová hodnota event.button znamená, že myš nikdo neutiskuje. Příklad:

```
<img onmousedown="if(event.button == 2) alert('bylo stisknuto pravé myšičko')">
```

Po kliknutí na obrázek () pravým tlačítkem se objeví hláška. Při kliknutí levým se nic dít nebude, ačkoli proběhla událost onmousedown i onclick.

Zrušení kliknutí

Dost častý problém u odkazů je ten, že chci, aby po kliknutí spouštěly nějaký skript a aby nenutily prohlížeč přejít na jinou stránku. Kdyby totiž prohlížeč přešel na jinou stránku, tak je ten provedený skript většinou na nic. Zrušení navigace se provede pomocí klauzule `return false`, čímž jakoby odkazu řeknu, že nebyl prokliknut:

```
<a href="někam_úplně_jinam" onclick="něcoUdělej(); return false">Odkaz</a>
```

OnDbClick

Dvojklik nastává, pokud se na elementu klikne dvakrát rychle za sebou. Není to moc použitelná věc, protože lidé nejsou na internetu zvyklí klikat dvakrát. Rychlost dvojkliku záleží na nastavení systému.

OnMouseOver, onMouseOut, onMouseMove

Tyto tři vlastnosti jsou taková trojčata. Událost onMouseOver se spouští ve chvíli, kdy myš najede na element. onMouseOut se spouští, když myš element opustí. Mezitím probíhá událost onMouseMove s výjimkou situace, kdy se myš úplně zastaví.

onmouseover a onMouseOut se většinou používají ve dvojkombinaci. Něco se stane, když myš na prvek najede, jiná akce (zpravidla inverzní) se děje, když myš prvek opustí. Klasickým příkladem je záměna obrázků po najetí myši:

```

```

(Nyní pomímám, že by příklad nefungoval ve starších prohlížečích, to ale není způsobeno událostmi, leč použitím objektu this.)

Událost onMouseMove se používá dost zřídka. Není totiž moc šikovná a při uměleckém zápisu dokáže navíc pěkně zavařit procesor (např. SMS brána Oskara v roce 2002), protože se spouští opakovaně ve velmi krátkých intervalech.

OnMouseDown, onMouseUp

Zřídka používané události zachycují stisknutí a uvolnění tlačítka myši, nikoliv samotný klik. Ke kliknutí (onclick) dojde jen v tom případě, že na jednom elementu nastane po sobě onMouseDown a onMouseUp. Pokud tedy někde myšičku stisknu, odjedu na jiný element a tam myš pustím, k události onclick na elementech nedojde. (OnClick ale zachytí pravděpodobně pouze objekt document, protože myš se stiskla i uvolnila na něm.)

Podstatná výhoda těchto dvou událostí je v přesném načasování skriptu a také v tom, že narozdíl od onclick reagují i na stisknutí pravého nebo prostředního tlačítka.

Příklady: [Identifikace pravého tlačítka](#), [Pofiderní zakázání pravého tlačítka](#)

OnContextMenu

Událost se volá při vyvolání kontextové nabídky, nejčastěji kliknutím pravého tlačítka. Nefunguje v Opeře. Nejjednodušší zakázání pravého tlačítka myši:

```
<body oncontextmenu="return false">
```

A zcela mimochodem: nezakazujte pravé tlačítko, je to prasárna.

Události klávesnice

Událost	Popis	Lze použít na	Nefunguje v
onKeyPress	při stisknutí klapky ve chvíli, kdy je element aktivní	asi cokoliv	IE3, NN3
onKeyDown	při stlačení klapky ve chvíli, kdy je element aktivní		IE3, NN3
onKeyUp	při uvolnění klapky ve chvíli, kdy je element aktivní		IE3, NN3

Události klávesnice se dají použít ve chvíli, kdy je aktivní nějaký element, který by měl na stisk kláves reagovat. Naprosto nejčastěji jsou to formulářové prvky (ale teoreticky i odkazy, ty si všimají Entru).

[Test onKeyPress](#)

OnKeyDown reaguje na stlačení, onKeyUp na uvolnění klávesy. Dohromady to dává událost onKeyPress, která se také pro klávesové události používá v naprosté většině.