Poznámka

Nadradeným elementom sa hovorí rodič alebo predchodca a vnoreným elementom sa hovorí potomok, ale tiež dieťa alebo následník (len o úroveň nižšie). Vzniká tak priame príbuzenstvo a vzťah rodič - potomok. Elementy na rovnakej úrovni predstavujú súrodencov, pričom všetky majú jedného rodiča a zároveň sú potomkami rodiča. Potomkom môže byť aj element, ktorý je vnorený o viac ako jednu úroveň v rodičovi. Ak je vnorený len o dve úrovne, ide o potomka priameho potomka rodiča.

Rozvojom DOM sa zaoberá organizácia W3C (World Wide Web Consortium), ktorá zároveň definovala odporúčania pre DOM (**http://www.w3.org/DOM/**). Aktivity na vývoji DOM sú v súčasnosti ukončené, pričom práca je riadená rôznymi pracovnými skupinami konzorcia W3C, medzi ktoré patria skupiny HTML, SVG a WebAPI.

W3C ukončilo prácu na DOM v roku 2004 a to dokončením odporúčaní pre DOM level 3. Niektoré pracovné skupiny W3C kontinuálne pracujú na rozvíjaní DOM. Objektový model dokumentu sa delí na tri úrovne s označením Level 1, Level 2 a Level 3.

Organizácia W3C vytvorila niekoľko odporúčaní, ktoré sú uvedené v nasledujúcich dokumentoch:[4]

- Document Object Model Level 1
- Document Object Model Level 2 Core
- Document Object Model Level 2 Views
- Document Object Model Level 2 Events
- Document Object Model Level 2 Style
- Document Object Model Level 2 Traversal and Range
- Document Object Model Level 2 HTML
- Document Object Model Level 3 Core
- Document Object Model Level 3 Load and Save
- Document Object Model Level 3 Validation

Popri tom pracovné skupiny pripravili k týmto odporúčaniam rôzne pripomienky, ktoré W3C zapracovalo do nasledujúcich dokumentov.

- Document Object Model Level 1 (Second Edition)
- Document Object Model Level 3 XPath
- Document Object Model Requirements
- Document Object Model Level 3 Views and Formatting
- Document Object Model Level 3 Events
- Document Object Model Level 3 Abstract Schemas

[4] **http://www.w3.org/DOM/Activity**
http://www.w3.org/DOM/DOMTR

JavaScript v dokumente XHTML

Dokument XHTML vrátane jazykov CSS a JavaScript sa zobrazí hneď po načítaní a spracovaní internetovým prehliadačom. Dokument XHTML sa spracováva zhora nadol. Po prejdení na posledný riadok dokumentu sa už nič nevykoná. Ak sú vykonané dáke zmeny v dokumente alebo v externých súboroch, dokument alebo súbory je nutné so zmenami uložiť a opätovne vykonať načítanie dokumentu XHTML. Pre opätovné načítanie dokumentu XHTML môže byť použitá funkcia refresh (v internetových prehliadačoch zvyčajne kláves F5).

Jazyk JavaScript môže byť do dokumentu XHTML zapísaný viacerými spôsobmi. Prvý spôsob je umiestnenie programového kódu v JavaScripte do hlavičky dokumentu (element `<head>`). Druhý spôsob je umiestnenie programového kódu v JavaScripte do externého súboru. Tretí spôsob je umiestnenie JavaScriptu do tela dokumentu XHTML, medzi zápis `<body> … </body>`. Ďalšou možnosťou je kombinácia umiestnenia JavaScriptu do tela aj do hlavičky alebo umiestnenie JavaScriptu do atribútu elementu najčastejšie udalosti.[12]

Jazyk JavaScript je možné do dokumentu XHTML implementovať prostredníctvom elementu `<script>` všade tam, kde to jazyk XHTML dovoľuje.[13]

Príklad zápisu elementu `<script>` do dokumentu XHTML.

```
<script type="text/javascript">
//<![CDATA[

Programový kód v jazyku JavaScript

//]]>
</script>
```

Sekcia CDATA (`//<![CDATA[ //]]>`) umožňuje v elemente `<script>` používať jazyk JavaScript. Okrem priameho zápisu jazyka JavaScript do dokumentu XHTML je možné použiť aj zápis, ktorý k dokumentu pripojí externý súbor. Súbory s jazykom JavaScript majú najčastejšie príponu `*.js`. Názov súboru sa zapisuje ako hodnota atribútu `src`. Pripojenie súboru s jazykom JavaScript je možné vykonať v hlavičke dokumentu XHTML (zápis do sekcie Head) alebo v tele dokumentu XHTML (zápis do sekcie XHTML).

Príklad pripojenia externého súboru s názvom `jscript.js` k dokumentu XHTML.

```
<script type="text/javascript" src="jscript.js"></script>
```

[12] Odporúča sa používať platné zápisy jazyka XHTML. Iné spôsoby umiestnenia jazyka JavaScript ako hodnoty atribútu elementu nemusia byť internetovými prehliadačmi podporované.

[13] Viac informácii o elemente `<script>` sa nachádza v druhom vydaní publikácie s názvom *Príručka programátora - Prehľadný sprievodca jazykom XHTML 1.1*.

DOM A CSS

Vlastnosti a hodnoty jazyka CSS je možné z dokumentov XHTML získať prostredníctvom metód `getComputedStyle()` a `currentStyle()`. Vlastnosti jazyka CSS sa v DOM zapisujú bez pomlčky. Slovo nasledujúce za pomlčkou začína veľkým písmenom (zápis typu camel-casing).

Príklad č.10. *Výpis podporovaných vlastností prvého elementu `<div>` s kontrolou podpory metód `getComputedStyle()` a `currentStyle()`.*

```
var element = document.getElementsByTagName("div")[0];
if (typeof element.currentStyle != "undefined") var hodnota =
element.currentStyle;
else var hodnota = window.getComputedStyle(element, null);
for (var x in hodnota) document.write(x + "<br/>");
```

Nastavenie hodnôt vlastností jazyka CSS je možné vykonať prostredníctvom vlastnosti `style`. Vlastnosť nastavuje v elemente hodnotu atribútu `style`.

```
element.style.vlastnost = hodnota;
```

Ak atribút `style` s príslušnou vlastnosťou nie je v elemente definovaný, nasledujúci formát zápisu bude nefunkčný.

```
var hodnota = element.style.vlastnost;
```

Príklad zápisu vlastnosti `style`

```
document.getElementsByTagName("div")[0].style.color = "red";
```

Získanie hodnôt vlastností jazyka CSS je možné aj prostredníctvom metódy `getComputedStyle()` a vlastnosti `getPropertyValue`.

Formát zápisu

```
window.getComputedStyle(element, pseudoelement)
```

Argument `pseudoelement` môže mať hodnotu `null`. Metódu `getComputedStyle()` je možné aplikovať na objekt `document` alebo `window`.

Formáty zápisu

```
document.defaultView.getComputedStyle(element, null)
```

```
document.defaultView.getComputedStyle(element, null).getPropertyValue(vlastnost)
```

OBJEKTY DOM

Objektový model dokumentu špecifikuje ako sú dokumenty XHTML a XML zastúpené objektmi. Pri práci s objektmi sa používa objektovo orientované programovanie. Objekty DOM rovnako ako v jazyku JavaScript majú svoje metódy (funkcie) a vlastnosti (premenné/parametre s hodnotami). Hodnoty vlastností môžu byť určené len pre čítanie alebo pre zápis aj čítanie. Typ hodnoty vlastnosti je závislý od príslušného objektu. Metódy ako funkcie sú k objektom priraďované prostredníctvom operátora `.` (bodka). Na rozdiel od skriptovacieho jazyka JavaScript, názvy objektov DOM obsahujú malé písmená.

Formát zápisu

```
objekt.metoda();
objekt.vlastnost;
```

V nasledujúcej časti sú uvedené vybrané metódy a vlastnosti DOM Level 2 objektov `window` a `document`. Príklady v tejto publikácii sú uvádzané pre skriptovací jazyk JavaScript. DOM je možné aplikovať aj prostredníctvom iných jazykov, napr. PHP, ASP alebo Java, pričom princíp manipulácie s objektmi je rovnaký.

Objekt window

Objekt `window` je súčasťou DOM ako API (application programming interface - aplikačné programové rozhranie) pre dokumenty XHTML a XML. Je rodičovským prvkom ďalších objektov `document`, `frames`, `history`, `location`, `navigator`, `screen` a `forms`. Keďže rámy už nie sú v jazyku XHTML podporované, význam vlastností a metód objektu `frames` nie je v tejto publikácii uvedený. Ak dokument obsahuje rámy, každý rám má svoj vlastný objekt `window`. Rámy sú uložené v kolekcii `frames`. Každý rám obsahuje vlastnosť `name` s jeho názvom.

Pre prácu s dokumentmi XHTML je objektom hosťovského prostredia objekt `window`. Objekt `window` zastupuje objekt ECMAScriptu `Global`. Objekt `window` pracuje s oknom internetového prehliadača a objekt `document` pracuje so stránkou resp. dokumentom v objekte `window` v okne internetového prehliadača. Objekty `window` a `document` sú používané v kombinácii s ďalšími metódami a vlastnosťami. Priamy potomkovia objektu `window` nemusia mať zápis `window.metoda()`, stačí použiť len zápis `metoda()`.

METÓDA ALERT()

Metóda `alert()` zobrazí dialógové okno s upozornením a tlačidlom `OK`, ktorým sa okno zatvorí. Okno je možné zatvoriť aj ikonou krížik pre zatvorenie okna. Firefox vyžaduje v metóde uviesť aspoň jeden argument.

METÓDA PROMPT()

Metóda `prompt()` zobrazí dialógové okno s možnosťou vloženia reťazca alebo zrušenia okna. Metóda po stlačení tlačidla `OK` vráti vložený reťazec. Pri stlačení tlačidla `Cancel` vráti hodnotu `null`.

Formát zápisu

```
prompt('text_v_okne', 'text_v_poli');
```

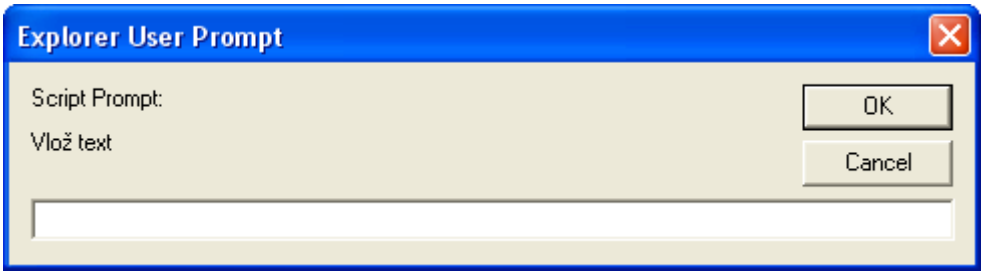
Význam argumentov

- `text_v_okne`, definuje fixný text v okne.
- `text_v_poli`, definuje text v poli okna. Preddefinovaná hodnota môže byť zmenená.

Príklad č.21. *Použitie metódy* `prompt()`.

```
var text = prompt('Vlož text','');
document.write("Do promptu bol vložený text: " + text + "<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*



METÓDA SHOWMODALDIALOG()

Metóda `showModalDialog()` zobrazí modálne okno s dokumentom, definovanými parametrami a vlastnosťami modálneho okna.

Formát zápisu

```
window.showModalDialog(uri[, parametre][, moznostiOkna]);
```

Význam argumentov

- `uri`, definuje URI dokumentu.
- `parametre`, argument obsahuje hodnoty, ku ktorým je možné pristupovať prostredníctvom vlastnosti `dialogArguments`.

VLASTNOSŤ TOP

Vlastnosť `top` vráti odkaz na najvrchnejšie okno v hierarchii okien.

Formát zápisu

```
window.top
```

VLASTNOSTI INNERWIDTH A INNERHEIGHT

Vlastnosti `innerWidth` a `innerHeight` vrátia veľkosť využiteľnej oblasti okna. Vlastnosť `innerWidth` vráti šírku okna a vlastnosť `innerHeight` výšku okna.

Formáty zápisu

```
window.innerWidth
window.innerHeight
```

VLASTNOSTI SCREENLEFT A SCREENTOP

Vlastnosť `screenLeft` vráti pozíciu ľavého okraja okna internetového prehliadača na obrazovke používateľa. Vlastnosť `screenTop` vráti pozíciu horného okraja okna internetového prehliadača na obrazovke používateľa. Okno prehliadača nie je pozícia internetového prehliadača ale priestor, kde sa zobrazuje dokument. Čím viac riadkov tvoria nástroje v internetovom prehliadači, tým väčšia bude hodnota vlastnosti `screenTop`. Vlastnosti nefungujú vo Firefoxe.

Formáty zápisu

```
window.screenLeft
window.screenTop
```

VLASTNOSTI SCREENX A SCREENY

Vlastnosti `screenX` a `screenY` vrátia súradnice umiestnenia okna internetového prehliadača na obrazovke. Vlastnosť `screenX` vráti x-ovú súradnicu a `screenY` y-ovú súradnicu umiestnenia internetového prehliadača. Vlastnosť nie je podporovaná všetkými internetovými prehliadačmi. Ide o alternatívu vlastností `screenLeft` a `screenTop`.

Formáty zápisu

```
window.screenX
window.screenY
```

VLASTNOSŤ BGCOLOR

Vlastnosť `bgColor` vráti alebo nastaví farbu pozadia dokumentu. Farba môže byť zadaná prostredníctvom kľúčových slov alebo hexadecimálnym zápisom.

Formáty zápisu

```
document.bgColor
```

Príklad č.45. *Zmena pozadia dokumentu.*

```
document.bgColor = "green";
var farba = document.bgColor;
document.write("Pôvodná farba pozadia dokumentu je " + farba + ".");
alert("Zmena farby pozadia.");
document.bgColor = "#aaccdd";
```

VLASTNOSŤ BODY

Vlastnosť `body` vracia odkaz na element `<body>` aktuálneho dokumentu. V prípade dokumentu s rámami vracia uzol `frameset`.

Formát zápisu

```
document.body
```

Príklad č.46. *Použitie vlastnosti `body` v kombinácii s vlastnosťou `style`, ktorá prostredníctvom vlastnosti jazyka CSS zmení farbu pozadia dokumentu.*

```
document.body.style.backgroundColor = "#aaccff";
```

VLASTNOSŤ CONTENTTYPE

Vlastnosť `contentType` vráti typ MIME dokumentu (napríklad `text/html`). Typ `MIME` je prevzatý z hlavičky `http` a nie z elementu `<meta>`.

Formát zápisu

```
document.contentType
```

VLASTNOSŤ DEFAULTCHARSET

Vlastnosť `defaultCharset` vráti regionálne nastavenie znakovej sady.

Formát zápisu

```
document.title
```

VLASTNOSŤ URL

Vlastnosť vráti URL dokumentu.

Formát zápisu

```
document.URL
```

VLASTNOSTI HEIGHT A WIDTH

Vlastnosti `height` a `width` určujú výšku a šírku dokumentu. Fungujú len vo Firefoxe. Chrome vráti veľkosť okna.

Formáty zápisu

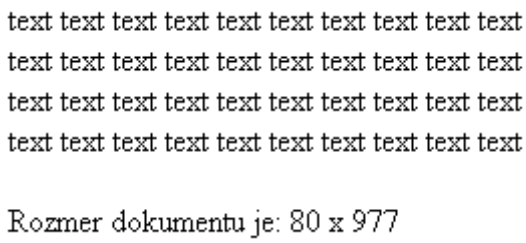
```
document.height
```

```
document.width
```

Príklad č.54. *Vygenerovanie textu a zistenie veľkosti dokumentu.*

```
function text(vypis) {
  for (var i = 2; i < 42; i++)  {
      document.write(vypis);
      if (i % 10 == 1) document.write("<br/>");
  }                     }
text('text ');
document.write("<br/>Rozmer dokumentu je: " + document.height + " x " +
document.width + "<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači Firefox.*



```
<legend>Dotazník</legend>
</fieldset>
</form>
```

Zápis do sekcie JavaScript

```
var element = document.getElementById('vloz');
var obsah = document.createTextNode("Zadaj meno: ");
element.appendChild(obsah);
var novy_element = document.createElement("input");
novy_element.setAttribute("type", "text");
element.appendChild(novy_element);
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Príklad č.74. *Pridanie zaškrtávacieho poľa do formulára.*

Zápis do sekcie XHTML

```
<form  method="post" action="spracuj.php" style = "width : 300px">
<fieldset id="vloz">
<legend>Vyber položku</legend>
</fieldset>
</form>
```

Zápis do sekcie JavaScript

```
var element = document.getElementById('vloz');
var obsah = document.createTextNode("Auto ");
element.appendChild(obsah);
var novy_element = document.createElement("input");
novy_element.setAttribute("type", "checkbox");
element.appendChild(novy_element);
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

VLASTNOSŤ CANCELABLE

Vlastnosť `cancelable` zistí, či je udalosť možné ukončiť. Ak je udalosť možné ukončiť, vráti hodnotu `true`, inak vráti `false`.

Formát zápisu

```
var hodnota = event.cancelable;
```

VLASTNOSŤ CURRENTTARGET

Vlastnosť `currentTarget` vráti odkaz na aktuálny objekt, ktorý udalosť spracováva. Aktuálny objekt nemusí byť cieľový objekt.

Formát zápisu

```
event.currentTarget
```

Príklad č.91. *Použitie vlastnosti `currentTarget` na všetky elementy `<div>`.*

```
var eDiv = document.getElementsByTagName('div');
function skri(e){
  e.currentTarget.style.visibility = "hidden";
  }
function odkri(e){
  e.currentTarget.style.visibility = "visible";
  }
for(var i = 0; i < eDiv.length; i++){
  eDiv[i].addEventListener('click', skri, false);
  }
for(var i = 0; i < eDiv.length; i++){
  eDiv[i].addEventListener('mouseout', odkri, false);
  }
```

Príklad č.92. *Použitie vlastnosti `currentTarget` na prvý element `<div>`..*

```
var element = document.getElementsByTagName('div')[0];
function zobraz(e) {
  alert("Udalosť teraz spracováva element: " + e.currentTarget.nodeName + "\n
  Cieľom udalosti je element: " + e.target.nodeName);
  }
element.addEventListener('click', zobraz, false);
```