

KAPITOLA 3

Přístup k dokumentu

Kdyby JavaScript neměl po ruce nějaký dokument, neměl by žádnou možnost, jak se předvést. Právě HTML dokument tvoří ono hmatatelné rozhraní, jehož prostřednictvím se JavaScript dostává ke svým uživatelům.

Tento vztah znamená, že je životně důležité, aby mohl JavaScript přistupovat k jakékoliv části dokumentu, manipulovat s ní a případně ji i vytvořit. Pro tyto potřeby vytvořilo konsorcium W3C tzv. objektový model dokumentu (DOM, Document Object Model), což je systém, jehož prostřednictvím mohou skripty ovlivňovat dokument. Tento systém umožňuje JavaScriptu nejenom měnit strukturu dokumentu, ale též přistupovat ke stylům dokumentu a měnit jeho vzhled.

Chcete-li převzít kontrolu nad vašimi rozhraními, musíte nejprve zvládnout DOM.

Objektový model dokumentu – mapování HTML

Když se do prohlížeče stáhne nějaký dokument HTML, musí prohlížeč to, co je v podstatě jediným dlouhatánským řetězcem, převést na webovou stránku. Aby to mohl udělat, musí rozhodnout, které části jsou odstavce, jaké části jsou záhlaví, které části jsou text atd. Aby neboží programátoři JavaScriptu nemuseli dělat totéž, prohlížeč uloží interpretaci kódu HTML jako strukturu objektů JavaScriptu, které se říká objektový model dokumentu (Document Object Model, neboli DOM).

V tomto modelu se každý prvek z HTML dokumentu stane objektem. Totéž se stane se všemi atributy a texty. JavaScript pak může ke všem těmto objektům přistupovat nezávisle, pomocí zabudovaných funkcí, které umožňují za pochodu snadno najít a změnit to, co potřebujete.

V důsledku toho, jak se HTML píše – jako hierarchie vnořovaných prvků vyznačených otevíracími a uzavíracími značkou – DOM sice vytvoří pro každý prvek jiný objekt, nicméně propojí každý objekt prvku s prvkem, který ho obklopuje (tzv. rodičovský prvek). Tím se mezi prvky vytvoří explicitní vztah rodič-potomek, což poskytuje vizualizaci DOM v podobě stromové struktury.

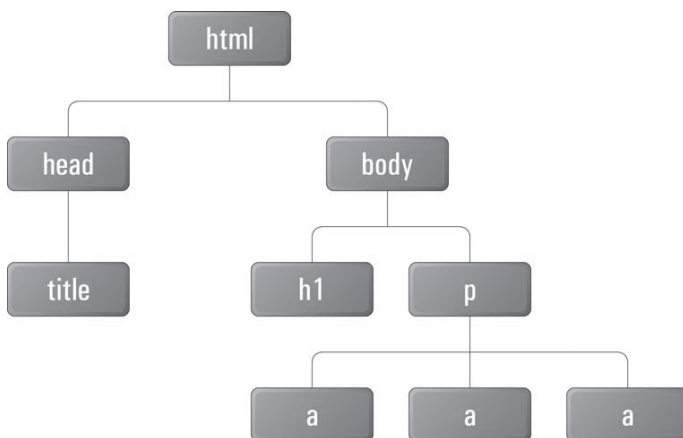
Mějme například tento kód HTML:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
  <head>
    <title>DOMinance JavaScriptu</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body>
    <h1>
      DOMinance JavaScriptu
    </h1>
    <p>
      Potřebujete-li pomoci se svým JavaScriptem, možná byste si měli
      přečíst nějaké články na stránkách <a href="http://www.danwebb.net/"
      rel="external">Dan Webb</a>,
      <a href="http://www.quirksmode.org/" rel="external">PPK</a>
      a <a href="http://adactio.com/" rel="external">Jeremy Keith</a>.
    </p>
  </body>
</html>

```

Nad prvky namapovanými v DOM se nejsnadněji rozumuje tehdy, pokud je chápete jako stromovou strukturu, která je znázorněna na obrázku 3.1.

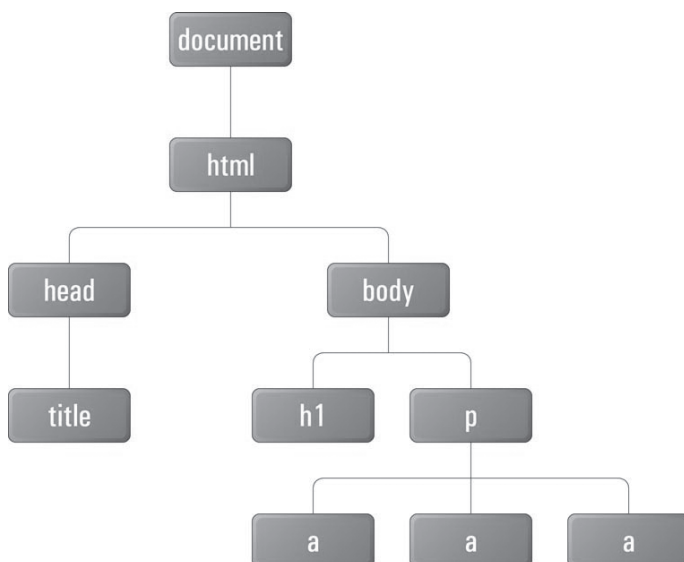


Obrázek 3.1. Každý prvek ve stránce HTML má v DOM propojení na svého rodiče.

Pokud se má pro nějaký HTML dokument vytvořit DOM, reprezentuje se každý prvek v dokumentu tím, co je obecně známo jako uzel (node). Pozice konkrétního uzlu ve stromu DOM je určena jeho rodičovským uzlem a dceřinými uzly (uzly přímých potomků).

Uzly prvků se rozpoznávají podle svých názvů prvků (head, body, h1 atd.), nicméně ty samozřejmě nemusí být jedinečné. Pokud nedodáte nějakou identifikační charakteristiku – jako je například atribut `id` – bude jeden uzel odstavce vypadat stejně jako ostatní.

Existuje jeden speciální uzel, který je v dokumentu obsažen vždy, bez ohledu na to, jaký má dokument vlastně obsah. Sídli vždy na vrcholu stromu a říká se mu uzel dokumentu (document node). Pokud toto vezmeme v potaz, získáme přesnější reprezentaci DOM (viz obrázek 3.2).



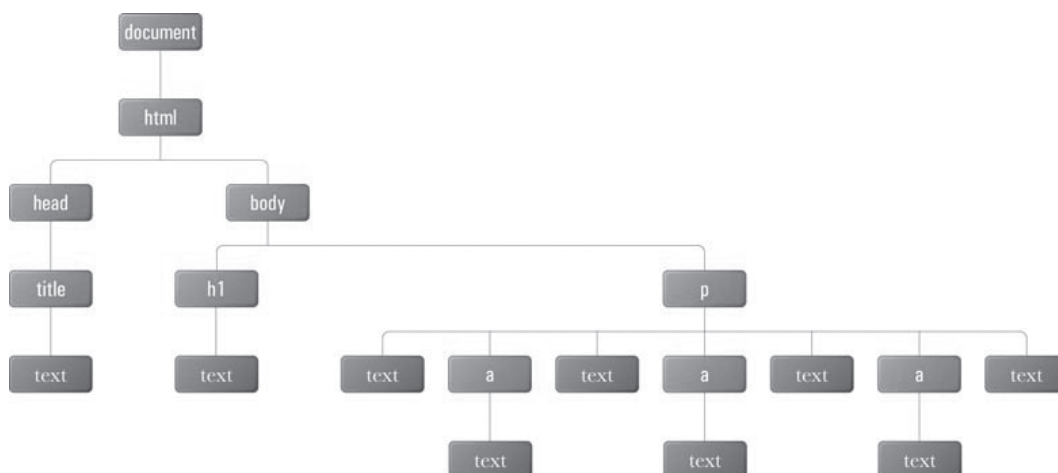
Obrázek 3.2. Strom DOM, včetně uzlu dokumentu.

Jedním typem uzlů jsou uzly prvků (element nodes), tj. uzly, které reprezentují prvky HTML. Definiují většinu struktury DOM, ovšem skutečný obsah dokumentu je obsažen ve dvou dalších typech uzlů – v textových uzlech a v uzlech atributů.

Textové uzly

Všechno z kódu HTML, co není obsaženo ve špičatých závorkách, se bude v DOM interpretovat jako textový uzel (text node). Z hlediska struktury se s textovými uzly zachází téměř stejně jako s uzly prvků – sídlí v téže stromové struktuře a dá se k nim dostat stejně jako k uzlům prvků. Nemohou mít ovšem potomky.

Pokud se ještě jednou zamyslíme nad příkladem HTML uvedeným výše a přidáme do naší vizualizace DOM textové uzly, bude strom mnohem rozsáhlejší, jak ilustruje obrázek 3.3.



Obrázek 3.3. Kompletní strom DOM, včetně textových uzlů.

Přestože textové uzly vypadají velmi podobně jako ostatní, každý z nich má svou vlastní hodnotu, v níž je uložený skutečný text, který je reprezentován tímto uzlem. Takže hodnota textového uzlu uvnitř prvku `title` je v našem konkrétním případě "DOMinance JavaScriptu."

Poznámka – textové uzly mohou produkovat i prázdné znaky

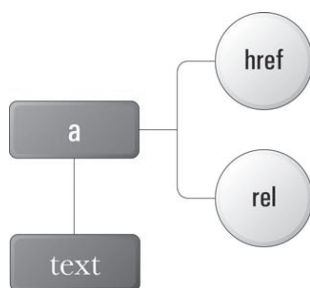
Kromě viditelných znaků obsahují textové uzly též neviditelné znaky, jako jsou např. znaky pro nový řádek nebo zarážku tabulátoru. Jestliže kód odsazujete, aby byl čitelnější (což my v této knize činíme), každý z těchto znaků pro konec řádku nebo zarážku, jimiž oddělujeme značky nebo text, bude zařazen do nějakého textového uzlu.

To znamená, že můžete mít nějaké dodatečné textové uzly mezi sousedícími prvky, nebo nějaké prázdné znaky navíc na začátku nebo na konci textového uzlu. Prohlížeče zpracovávají tyto uzly prázdných znaků (*whitespace nodes*) odlišně, a právě tato variabilita při analýze DOM je důvodem, proč musíte být pečliví a opatrní, pokud se chcete v DOM spoléhat na počet uzlů (nebo na jejich pořadí).

Uzly atributů

Protože uzly prvků a textové uzly zahrnují HTML značky a text, jediná další informace, kterou je ještě potřeba brát v úvahu pro DOM, jsou atributy. Ačkoliv na první pohled vypadají atributy jako části prvku (a jistým způsobem také jsou), ve skutečnosti zabírají svůj vlastní typ uzlů, který se příznačně nazývá uzly atributů (*attribute nodes*).

Každý ze tří hypertextových odkazů (tedy prvků `a`) se v naší ukázce DOM dá vizualizovat pomocí uzlů atributů znázorněných na obrázku 3.4.



Obrázek 3.4. Atributy `href` a `rel`, které jsou reprezentovány v DOM jako uzly atributů.

Uzly atributů jsou vždy připojeny k uzlu prvku, ale nezapadají do struktury DOM tak, jako uzly prvků a textové uzly – nezapočítávají se do dceřiných uzlů prvku, k němuž jsou připojeny. Kvůli tomu se pro práci s uzly atributů používají jiné funkce – probereme je v kapitole později.

Jak jste viděli z výše prezentovaných obrázků, DOM se velmi rychle stane složitým (a to i tehdy, když se jedná o hodně jednoduchý dokument), takže budeme potřebovat nějaké pořádné nástroje, abychom mohli identifikovat části, které potřebujeme, a uměli s nimi dobře manipulovat. Na tuto problematiku se podíváme hned teď.

Přístup k uzlům, s nimiž chcete něco dělat

Když jsme se dozvěděli, jak je DOM strukturovaný, dostali jsme jeden dobrý nápad. A to konkrétně ten, že by bylo vhodné uspořádat věci, k nimž chceme přistupovat. Každý uzel – ať už je to uzel prvku, textový uzel nebo uzel atributu – obsahuje informace, na jejichž základě ho můžeme identifikovat, nicméně by asi nebylo to pravé ořechové, kdybychom pokaždé museli projít všechny uzly v dokumentu, abychom našli ty, s nimiž chceme pracovat.

S prvkem se prostřednictvím DOM manipuluje podobným způsobem, jako se aplikují styly CSS na konkrétní prvek. Obě tyto úlohy jsou založeny na této všeobecné předloze:

1. Specifikovat prvek, nebo skupinu prvků, které chcete ovlivnit.
2. Specifikovat efekt, který na ně chcete aplikovat.

Přestože jsou způsoby, jimiž v obou technologiích manipulujeme s prvky, velmi odlišné, procesy, jimiž nacházíme prvky, se kterými chceme pracovat, jsou naopak pozoruhodně podobné.

Vyhledání prvku podle ID

Nejpřímější cesta k prvku vede přes jeho atribut `id`. Jedná se o nepovinný HTML atribut, který se dá přidat k jakémukoliv prvku na stránce, ovšem s tím omezením, že každé ID, jež použijete, musí být v rámci daného dokumentu jedinečné.

```
<p id="jedinecnyPrvek">
...
</p>
```

Zamýšlíte-li hledat prvky podle ID, uvědomte si, že tato snaha je založena na někdy nesnadno splnitelném předpokladu: že prvek, který chcete najít, má přiřazeno příslušné ID. Někdy bude takový předpoklad znamenat, že prvně budete muset strávit se svým kódem HTML nějakou tu chvilku, abyste se přesvědčili (nebo zajistili), že požadovaný prvek má skutečně přiřazeno nějaké ID. V některých případech se ID přirozeně objeví v HTML kódu jako součást sémantické struktury dokumentu. Pokud prvek má dvě ID, pomocí JavaScriptu jej naleznete až neobyčejně snadno.

Pokud se chcete na konkrétní prvek odkázat v CSS, použijte selektor ID začínající na #:

```
#jedinecnyPrvek { 1
color: blue; 2
}
```

Pokud to máme přeložit, tento kód říká následující.

1. Najít prvek s ID `jedinecnyPrvek`.
2. Nastavit jeho barvu na modrou.

CSS je hodně kompaktní jazyk. JavaScript až tak ne. Takže v JavaScriptu se na prvek s nějakým ID odkazujete prostřednictvím metody `getElementById`, která je dostupná z uzlu `document`. Jako argument přebírá řetězec, přičemž najde prvek, který obsahuje tento řetězec v podobě ID. Já osobně chápu metodu `getElementById` jako takového ostřelovače, který umí v daném okamžiku zasáhnout pouze jeden jediný prvek. Jinak řečeno, tato metoda je velmi úzce zaměřená. Představte si například, že náš dokument obsahuje následující kód HTML:

```
<h1>
  Sniper (1993)
</h1>
<p>
  V tomto mistrovském filmu hraje
  <a id="berenger" href="/name/nm0000297/">Tom Berenger</a>
  vojáka USA, který operuje v panamské džungli.
</p>
```

Pomocí následujícího kódu můžeme získat odkaz na prvek HTML s identifikátorem `berenger` (a to bez ohledu na to, o jaký typ prvku se vlastně jedná):

```
var target = document.getElementById("berenger");
```

Proměnná `target` se nyní odkazuje na uzel DOM prvku `a`, který obklopuje jméno Tom Berenger. Dále předpokládejme, že toto ID jsme přesunuli k nějakému jinému prvku:

```
<h1 id="berenger">
```

```
    Sniper (1993)
</h1>
<p>
    V tomto mistrovském filmu hraje
    <a href="/name/nm0000297/">Tom Berenger</a>
    vojáka USA, který operuje v panamské džungli.
</p>
```

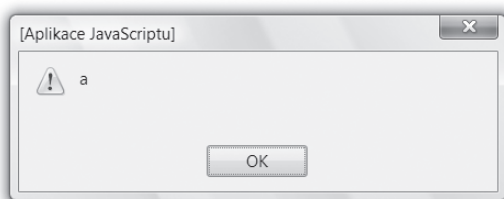
Pokud bychom nyní vykonali stejný kód JavaScriptu, odkazovala by se proměnná `target` na jiný prvek – v tomto případě na prvek nadpisu `h1`.

Jakmile máte k dispozici odkaz (referenci) na uzel prvku, můžete o něm získávat informace, nebo modifikovat jeho obsah prostřednictvím mnoha nativních vlastností a metod. Při práci s touto knihou prozkoumáte mnohé z těchto metod a vlastností.

Pokud se chcete pokusit získat nějaké informace o prvku, který jste právě našli, využijte jednu nebo několik nativních vlastností uzlu prvku. Jednou z těchto vlastností je `nodeName`, která sděluje přesný název značky uzlu, na který jste se odkázali. Chcete-li zobrazit název značky, která byla zachycena metodou `getElementById`, použijte tento kód:

```
var target = document.getElementById("berenger");
alert(target.nodeName);
```

Následně se objeví okno, v němž se vypíše název značky, jak to vidíte na obrázku 3.5.



Obrázek 3.5. Zobrazení názvu značky prostřednictvím vlastnosti `nodeName`.

Pokud neexistuje prvek s konkrétním ID, které jste použili pro hledání, `getElementById` nevrátí odkaz na uzel – vrátí hodnotu `null`. To je speciální hodnota, která obvykle indikuje, že něco chybí. V podstatě vyjadřuje nepřítomnost objektu tam, kde by za normálních okolností měl být.

Pokud si nejste jisti, zdali dokument obsahuje prvek s konkrétním ID, který hledáte, nejbezpečnější je zkontrolovat, zdali metoda `getElementById` skutečně vrátila objekt uzlu, protože když provádíte operace nad hodnotou `null`, většina operací skončí tím, že program oznámí chybu a přestane se vykonávat. Tato kontrola se snadno udělá pomocí podmínkového příkazu, který ověří, zdali odkaz vrácený z `getElementById` není náhodou `null`:

```
var target = document.getElementById("berenger");
if (target != null)
{
```

```
    alert(target.nodeName);  
}
```

Vyhledávání prvků podle názvu značky

Lokalizovat prvky podle jejich ID je vynikající technika, chcete-li modifikovat v daném čase pouze jediný prvek. Chcete-li však vyhledat celou skupinu prvků, je tou pravou metodou `getElementsByTagName`. Jejím ekvivalentem v CSS je selektor typu prvku:

```
li {  
  color: blue;  
}
```

Na rozdíl od metody `getElementById` se metoda `getElementsByTagName` může vykonat jako metoda jakéhokoli uzlu prvku – nejčastěji se ovšem volá s uzlem `document`.

Mějme následující dokument:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">  
  <head>  
    <title>Lokátor názvu značky</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
  </head>  
  <body>  
    <p>  
      V těle tohoto dokumentu jsou tři různé typy prvků:  
    </p>  
    <ul>  
      <li>  
        odstavec  
      </li>  
      <li>  
        neseřazený seznam  
      </li>  
      <li>  
        položka  
      </li>  
    </ul>  
  </body>  
</html>
```

Všechny tyto položky seznamu získáme tímto jediným řádkem JavaScriptu:


```
var listItems = document.getElementsByTagName("li");
```

Když vykonáváte tento kód, říkáte tím programu, aby prohledal všechny následníky uzlu `document`, získal všechny uzly, které mají název značky `li`, a přiřadil tuto skupinu do proměnné `listItems`. `listItems` bude obsahovat kolekci uzlů, které se říká seznam uzlů (node list). Seznam uzlů je objekt JavaScriptu, který obsahuje seznam objektů uzlů v pořadí podle zdroje. V příkladu, který jste právě viděli, mají všechny uzly seznamu uzlů název značky `li`.

Seznamy uzlů se v mnohém chovají podobně jako pole, s nimiž jsme se zabývali ve druhé kapitole, nicméně postrádají některé užitečné metody, jimiž jsou vybavena pole. Obvykle však s nimi můžete zacházet stejně. Protože `getElementsByTagName` vždy vrátí seznam uzlů ve zdrojovém pořadí, víme, že druhý uzel v seznamu bude skutečně druhým uzlem ve zdrojovém kódu HTML, takže kdybyste se na něj chtěli odkázat, použijete index 1 (vzpomeňte si, že první index v poli je 0):

```
var listItems = document.getElementsByTagName("li");  
var secondItem = listItems[1];
```

`secondItem` se nyní bude odkazovat na položku seznamu obsahující text "neseřazený seznam". Seznamy uzlů dále poskytují již známou vlastnost `length`, takže velmi snadno se dá získat počet uzlů v kolekci – stačí se odkázat na jeho `length`:

```
var listItems = document.getElementsByTagName("li");  
var numItems = listItems.length;
```

Pokud toto aplikujeme na náš dokument obsahující tři prvky seznamu, bude mít `numItems` hodnotu 3. Skutečnost, že na seznam uzlů se odkazuje podobě jako na pole, znamená, že se budou snadno sestavovat cykly, které procházejí všemi uzly seznamu, a jež s každým z nich provedou stejné akce. Chceme-li zkontrolovat, zdali `getElementsByTagName` opravdu vrátila pouze prvky se stejným názvem značky, můžeme si názvy značek všech uzlů vypsat cyklem `for`:

```
var listItems = document.getElementsByTagName("li");  
for (var i = 0; i < listItems.length; i++)  
{  
    alert(listItems[i].nodeName);  
}
```

Na rozdíl od `getElementById` vrací metoda `getElementsByTagName` seznam uzlů i tehdy, když dodaný název značky nemá žádný z prvků. Pak bude mít `length` tohoto seznamu uzlů hodnotu 0. To znamená, že je bezpečné používat příkazy, které kontrolují `length` seznamu uzlů, jako je tomu v cyklu výše. Není ovšem bezpečné se přímo odkazovat na nějaký index, pokud jste předtím nezkontrolovali `length`, abyste se přesvědčili, zdali zadaný index bude platný. Cyklování seznamem uzlů s jeho vlastností `length` jako podmínkou cyklu se obvykle považuje za nejlepší způsob, jak řešit úlohy tohoto druhu.