JavaScript DOM

Lili Nemec Zlatolas

Anotace

Tento kurz představuje objektový model dokumentu v jazyce JavaScript.

Cíle

Kurz poskytuje rychlý přehled o potřebných znalostech, jako jsou základy HTML a JavaScriptu. Studenti se seznámí s objektovým modelem dokumentu v jazyce JavaScript.

Klíčová slova

JavaScript, DOM, HTML

Datum vytvoření

15.4.2022

Časová dotace

12 hodin

Jazyková verze

česky

Licence

ISBN

Literatura

[1] Matt Frisbie. Professional JavaScript for Web Developers. Publishing place: John Wiley & Sons, 2019. 978-1-119-36644-7.

[2] R. Ferguson. Beginning JavaScript: The Ultimate Guide to Modern JavaScript Development. Apress, Ocean, 2019. 3rd edition.

[3] M. Haverbeke. Eloquent JavaScript - A Modern Introduction to Programming. Third Edition. No Starch Press, San Francisco, 2018.

[4] W3schools. Javascript HTML DOM. https://www.w3schools.com/js/js_htmldom.asp.

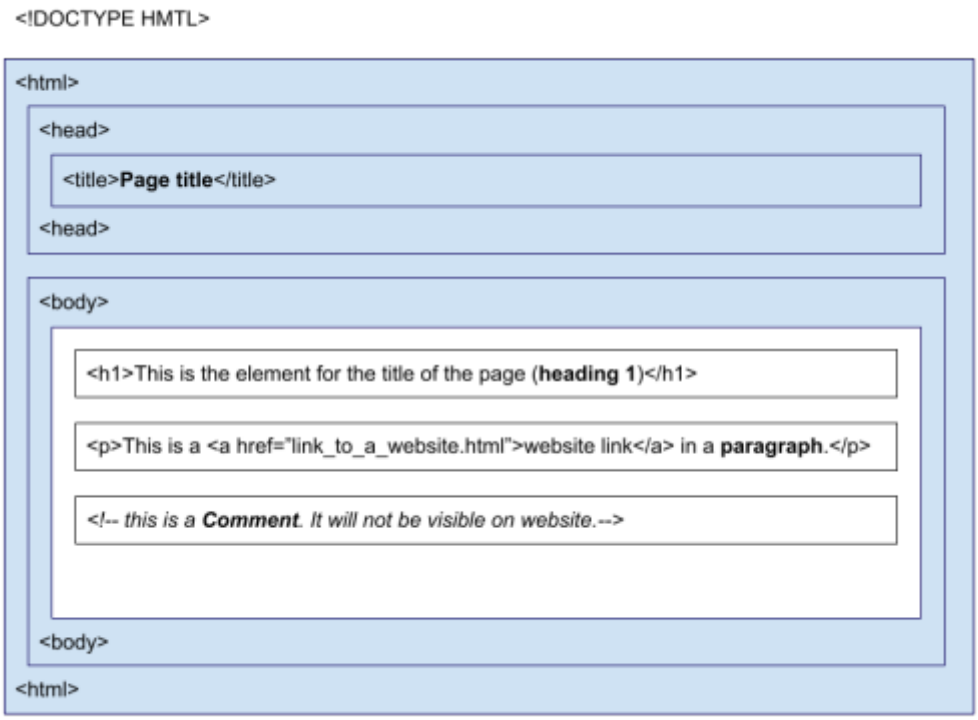
KAPITOLA 1

Základy jazyka HTML

K manipulaci s webovými stránkami pomocí objektového modelu dokumentu (Document Object Model – DOM) jsou nutné předchozí znalosti jazyka HTML a jazyka JavaScript. HTML je značkovací jazyk, který popisuje strukturu webových stránek. Jazyk HTML se skládá z prvků, které jsou opatřené značkami a posílají prohlížeči informace o tom, jak má zobrazit obsah dokumentu.

Posledním přijatým standardem je verze HTML5. HTML se skládá z elementů, které jsou definovány otvírací značkou – tagem, obsahem elementu a uzavírací značkou: <tag> obsah </tag>

Ukázka HTML dokumentu:



Obr. 1. Struktura stránky HTML a některé prvky

K úpravě dokumentů můžete použít Poznámkový blok nebo podobné programy (např. Notepad++, Visual Studio Code). Dokument HTML musí mít příponu .html.

HTML nerozlišuje velká a malá písmena, ale doporučuje se používat malá písmena ve značkách HTML.

Tabulka 1. Některé základní prvky jazyka HTML

Značka	Popis
`<! DOCTYPE html>`	Deklarace HTML5 dokumentu, která je umístěna na začátku dokumentu HTML.
`<h1>`	V HTML jsou nadpisy od <h1> do <h6>, přičemž první je ten největší.
`<p>`	Tato značka označuje odstavec.
` `	Tato značka slouží k přechodu na další řádek (odřádkování).
`<a href="https://www.websitelink.com">`	Tato značka je určena pro odkaz na webovou stránku. Používá atributy.
`<img src="image_location.jpg" alt="image_description">`	Tato značka slouží k vložení obrázku. Používá také různé atributy.
`<b>`	Tato značka slouží k zobrazení tučného písma v textu.

Některé prvky se nazývají nepárové. Příkladem nepárového prvku je
, který nemá obsah ani uzavírací značku. Pokud však použijeme pravidla XHTML, může obsahovat uzavírací značku uvnitř otvírací značky:

Značky HTML mohou mít atributy. Některé prvky nemají bez atributů žádné funkce. Atributy se uvádějí v otvíracích značkách. Jedním z často používaných atributů je style, který lze použít například v odstavci:

```
<p style="color: blue;"> Tento text bude modrý. </p>
```

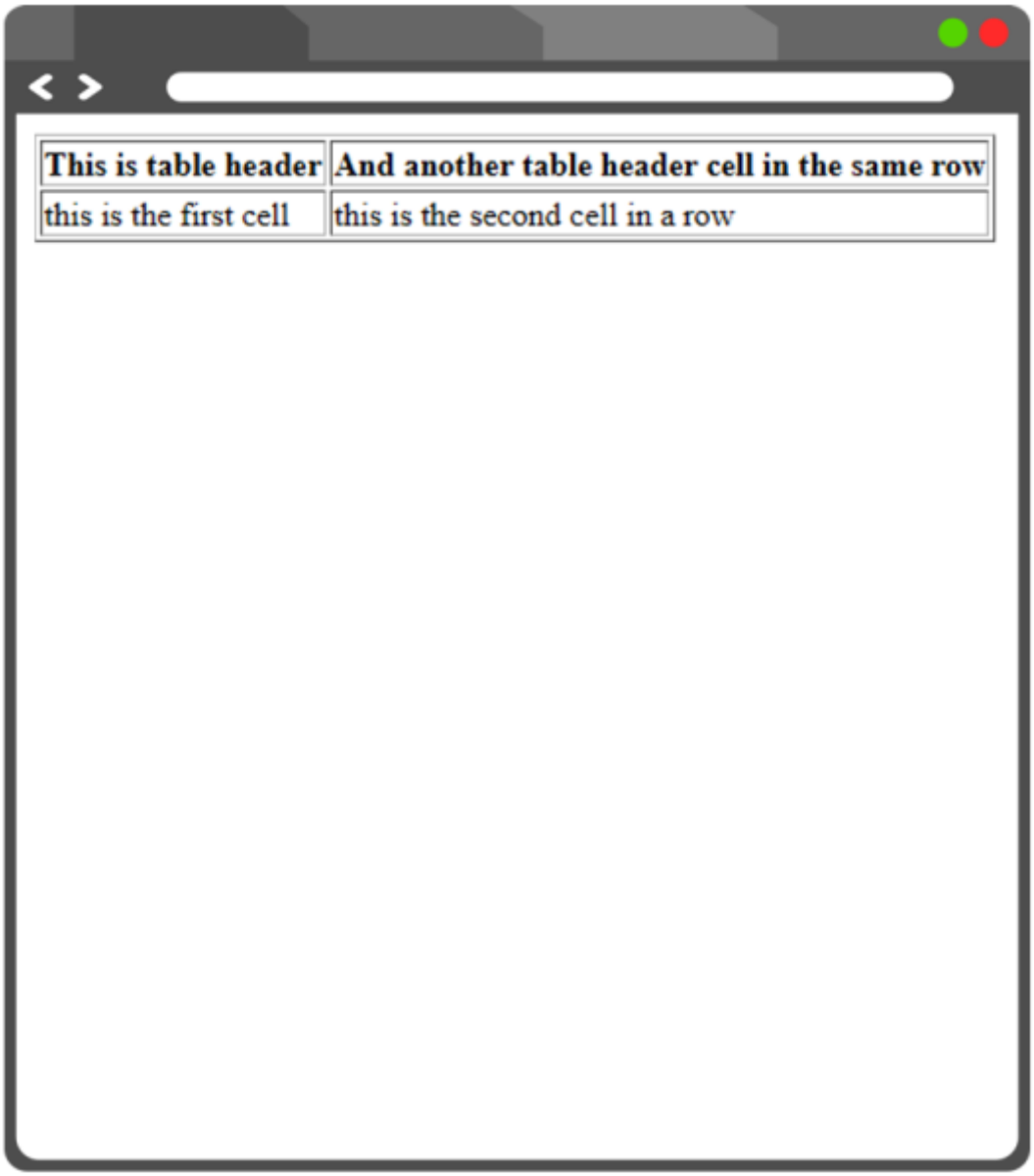
Interaktivní prvek

V HTML se často používají tabulky, které se skládají z buněk uvnitř sloupců a řádků tabulky. Jednoduchá tabulka HTML vypadá následovně:

PŘÍKLAD

```
<table border="1">
  <tr> <!-- toto je řádek tabulky, který nemá obsah -->
    <th> Toto je záhlaví tabulky </th>
    <th> Další buňka záhlaví na stejném řádku </th>
  </tr>
  <tr>
    <td> první buňka </td>
    <td> druhá buňka v řádku </td>
  </tr>
</table>
```

Interaktivní prvek



Obr. 2. Jak vypadá tabulka v prohlížeči.

Pomocí značek lze v HTML vytvářet uspořádané, nebo neuspořádané seznamy. Uspořádané seznamy (značka <ol>) jsou obvykle číslované a neuspořádané (značka <ul>) seznamy mají obvykle bodové odrážky.

Tabulka 2. Příklad seznamů v HTML

Příklad seznamu v jazyce HTML	Zobrazení v prohlížeči

   - Pes - Kočka - Ryba 	• Pes • Kočka • Ryba
   - Pes - kočka - Ryba 	1. Pes 2. Kočka 3. Ryba

Interaktivní prvek

Toto byl krátký úvod do jazyka HTML. V příští kapitole se budeme věnovat základům syntaxe jazyka JavaScript.

KAPITOLA 2

Základy jazyka JavaScript

JavaScript je programovací jazyk, který umožňuje zvýšit interaktivitu a flexibilitu webových stránek. Používá se v jazyce HTML mezi značkami <script> a </script> buď v části <body>, nebo <head>, případně v obou částech.

Kód JavaScriptu lze také umístit do souboru .html jako externí soubor s příponou .js. Například:

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

Existuje několik možností, jak zobrazit data v jazyce JavaScript. Můžeme použít:

innerHTML (např. document.getElementById(id))

- Těmto metodám se budeme věnovat v dalších kapitolách, protože se jedná o DOM.

document.write()

- Tento postup se obvykle používá k testování, protože z dokumentu odstraní veškeré existující HTML a zobrazí pouze obsah skriptu.

window.alert()

- Vytvoří okno s upozorněním pro zobrazení dat. Výraz window lze vynechat.

console.log()

- Obvykle se používá pro účely ladění a zobrazuje data v prohlížeči.

Jednotlivé příkazy jsou odděleny středníkem (;).

Bloky kódu jsou seskupeny uvnitř {složených závorek}.

Interaktivní prvek

Tabulka 3. Vyhrazená slova v jazyce JavaScript

Vyhrazené slovo	Popis
`var`	Deklaruje proměnnou
`let`	Deklaruje blokovou proměnnou, kterou lze měnit.
`const`	Deklaruje proměnnou, kterou nelze měnit.
`return`	Ukončí funkci
`if`	Začátek kódu podmínky

`for`	Začátek kódu smyčky
`function`	Deklaruje funkci

Interaktivní prvek

Syntaxe jsou pravidla JavaScriptu, podle kterých jsou programy konstruovány. Nejprve se deklarováním vytvoří proměnné a poté se v programu použijí.

Desetinná **čísla** se oddělují **tečkou**. **Texty** se zapisují do **jednoduchých** nebo **dvojitých uvozovek**. **Znaménka rovnosti** přiřazují proměnným hodnoty.

Názvy proměnných musí začínat písmeny abecedy (A-Z), znakem dolar nebo podtržítkem. **Rozlišují se velká a malá písmena.**

Ukázka definice různých proměnných:

```
let firstNumber;
var firstText;
firstNumber = 13;
var secondNumber = 17;
firstText = "This is number 13."
```

Tabulka 4. Rovnítka v JavaScriptu

Rovnítko	Popis
=	Jedná se o operátor přiřazení (např. **var a = a * 2;**) a nemá stejný význam jako algebraický znak.
==	Jedná se o operátor rovnosti, který se používá v algebře, např. v případě 7+2=9, a v JavaScriptu se používá např. jako **if (x == 2) {};**
===	Jedná se o stejnou hodnotu a také stejný datový typ.

Pomocí = a + můžete spojit nebo vypočítat hodnotu proměnné.

```
let wholeName = "Jméno" + " " + "Příjmení";
```

Interaktivní prvek

Jazyk JavaScript má 3 logické operátory:

- **&&** logické **a**
- || logické **nebo**
- **!** negace

Funkce jsou bloky kódu, které je třeba zavolat, aby se provedly. Mohou mít více parametrů a jejich kód se nachází ve složených závorkách. Jsou užitečné, když danou funkci použijeme vícekrát s různými argumenty. Příklad funkce:

PŘÍKLAD

```
function circle_area( a, b)
{
return a * a * b;  // Funkce vrací součin a, a a b. Pokud chceme
vypočítat plochu kruhu, musíme místo b zadat číslo π.
}
let area = circle_area (15, Math. PI);  // Funkce je volána s parametry
15 a π.
document. write( "Plocha kruhu je " + area + " m2.");
```

Při spuštění výše uvedené funkce se v prohlížeči zobrazí: "**Plocha kruhu je 706.8583470577034 m2.**"

Je ještě mnoho dalších věcí, které se můžeme v JavaScriptu naučit, ale toto je základ a několik nezbytných znalostí, které potřebujeme pro začátek práce s DOM v JavaScriptu.

KAPITOLA 3

Úvod do objektového modelu dokumentu (DOM)

Prohlížeč při načítání webové stránky vytvoří objektový model dokumentu (DOM). DOM je standard konsorcia W3C (World Wide Web Consortium) pro přístup k dokumentům.

Model HTML DOM je vytvořen jako strom objektů.

V animaci je uveden příklad následujícího kódu.

```
<html>
  <head>
    <title>Toto je hlavní stránka</title>
  </head>
  <body>
    <h1 align="left">Toto je nadpis</h1>
    <p style="background-color: blue">Toto je obrázek.</p>
    <img src="image.jpg" />
  </body>
</html>
```

Interaktivní prvek

Animace 1. Strom objektů v modelu HTML DOM

Strom má **uzly** pro elementy, které představují značky HTML a určují strukturu dokumentu. Standard je navržen tak, že nejprve vytvoříme uzel, pak k němu přidáme potomka a k němu atributy. Proto může být kód poměrně dlouhý.

V animaci je <html> *kořenový uzel* bez nadřazeného uzlu, ale je *nadřazený* prvnímu *podřízenému uzlu* <head> a *poslednímu podřízenému uzlu* <body>.

JavaScript může přistupovat k modelu DOM a měnit prvky a atributy HTML stránky.

V modelu DOM jsou všechny prvky HTML definovány jako **objekty**.

DOM je standard, který obsahuje informace o tom, jak se dostáváme k prvkům HTML, jak je měníme, přidáváme nebo odstraňujeme. DOM používá **metody** pro přístup k prvkům HTML a provádění akcí s nimi.

Interaktivní prvek

3.1 Navigace v DOMu – procházení stromové struktury

Uzly ve stromu uzlů mají hierarchický vztah, což znamená, že každý uzel má přesně jednoho předka, kromě prvního uzlu, který nemá žádného předka. Uzel může mít více potomků. Uzly se stejným předkem se nazývají sourozenecké uzly (siblings).

Pro navigaci mezi uzly pomocí JavaScriptu můžeme použít následující vlastnosti uzlu:

- parentNode – nadřazený uzel (předek)
- childNodes[nodenumber] – výběr uzlu ze seznamu podřízených uzlů
- firstChild – první podřízený uzel (potomek)
- lastChild – poslední podřízený uzel (potomek)
- nextSibling – následující uzel na stejné úrovni
- previousSibling – přecházející uzel na stejné úrovni

Animace 2. Příklad uzlů v dokumentu

Vlastnost uzlu *childNodes[0]* je stejná jako *firstChild*. Obsahuje objekt podobný poli s vlastností *length* (délka/počet prvků v poli), pomocí které přistupuje k podřízeným uzlům.

Interaktivní prvek

Uzly a způsoby získávání obsahu z prvků se budeme podrobněji zabývat v kapitole Uzly DOM.

KAPITOLA 4

Metody modelu DOM

Pomocí **metod** modelu DOM můžeme provádět akce s prvky HTML. **Vlastnosti** modelu DOM jsou hodnoty prvků HTML, které můžeme nastavovat nebo měnit.

PŘÍKLAD

```
<p id="example"> To se na stránce nezobrazí, protože Javascript přepíše
text uvnitř tohoto odstavce. </p>
<script>
  document.getElementById("example").innerHTML = " Tento text se zobrazí
na webové stránce.";
</script>
```

V tomto příkladu se v prohlížeči při spuštění kódu zobrazí „**This is the text that will be displayed on the website.**“.

V tomto příkladu je getElementById **metoda** a innerHTML je **vlastnost**. Velmi často se pro vyhledání elementu v dokumentu používá **id** prvku (v příkladu id="example"). V jiných případech můžeme pro přístup k určitým prvkům použít nadřazené a podřízené uzly modelu DOM.

Vlastnost innerHTML slouží k nastavení nebo vrácení obsahu HTML elementu, ve výše uvedeném příkladu se jedná o změnu textu uvnitř značky <p> s parametrem id="example".

Tabulka 5. Metody vyhledávání, změny, přidávání a odstraňování prvků HTML

Metoda	Popis
document.getElementById(id)	Vyhledání prvku podle id prvku
document.getElementsByTagName(name)	Vyhledání prvku podle názvu značky
document.getElementsByClassName(name)	Vyhledání prvku podle názvu třídy
element.setAttribute(attribute, value)	Změna hodnoty atributu prvku HTML
document.createElement(element)	Vytvoření elementu HTML
document.removeChild(element)	Odstranění podřízeného prvku HTML
document.appendChild(element)	Přidání podřízeného prvku HTML
document.replaceChild(new, old)	Nahrazení podřízeného prvku HTML

Interaktivní prvek

Video 1. Příklady získání prvku podle X

Interaktivní prvek

KAPITOLA 5

Uzly (nody) DOM

Všechny **prvky HTML, jejich atributy a texty jsou uzly**. Některé prvky obsahují i další uzly.

Navigace mezi uzly již byla popsána v kapitole Navigace v DOMu.

Zde je příklad, jak můžeme přistupovat k hodnotám uzlů.

```
<p id='paragraph'>Toto je první odstavec</p>
```

Element <p> obsahuje **textový uzel** s hodnotou "Toto je první odstavec". K hodnotě textu lze přistupovat pomocí vlastnosti innerHTML uzlu:

```
textFromParagraph = document.getElementById('p').innerHTML;
```

Totéž lze provést přístupem k hodnotě **nodeValue**:

```
textFromParagraph
= document.getElementById('p').childNodes[0].innerHTML.nodeValue;
```

Video 2. Podřízené uzly a hodnoty uzlů

Kořenové uzly mají přístup k celému dokumentu:

- document.body – Obsah dokumentu
- document.documentElement – Celý dokument

Vlastnost **nodeValue** určuje hodnotu uzlu. Pro uzly prvků je nulová, ale je užitečná pro textové uzly, kde představuje samotný text. Pro uzly atributů vrací vlastnost nodeValue hodnotu atributu.

Vlastnost **nodeName** určuje název uzlu a je určena pouze pro čtení. nodeName uzlu prvku vrací název tagu a vrací název atributu uzlu. nodeName textového uzlu je *#text* a uzlu dokumentu je *#document*.

V dalších podkapitolách budeme používat některé z následujících metod pro vytváření, odstraňování a nahrazování prvků DOM HTML (uzlů). V této tabulce se seznámíte s metodami pro vytváření prvků a textových uzlů.

Tabulka 6. Metody pro vytváření nových prvků HTML pomocí DOM

Metoda	Popis
createElement(type)	Vytvoří uzel prvku s určitým typem (např. typ "p").
createTextNode()	Vytvoří textový uzel.

V následující tabulce jsou uvedeny metody vytváření, odstraňování a nahrazování uzlů. Obvykle musíme vytvořit prvek, pak v něm vytvořit textový uzel a ten přidat do existující struktury. Příklady použití jsou uvedeny v dalších podkapitolách.

Tabulka 7. Metody vytváření, odstraňování a nahrazování uzlů

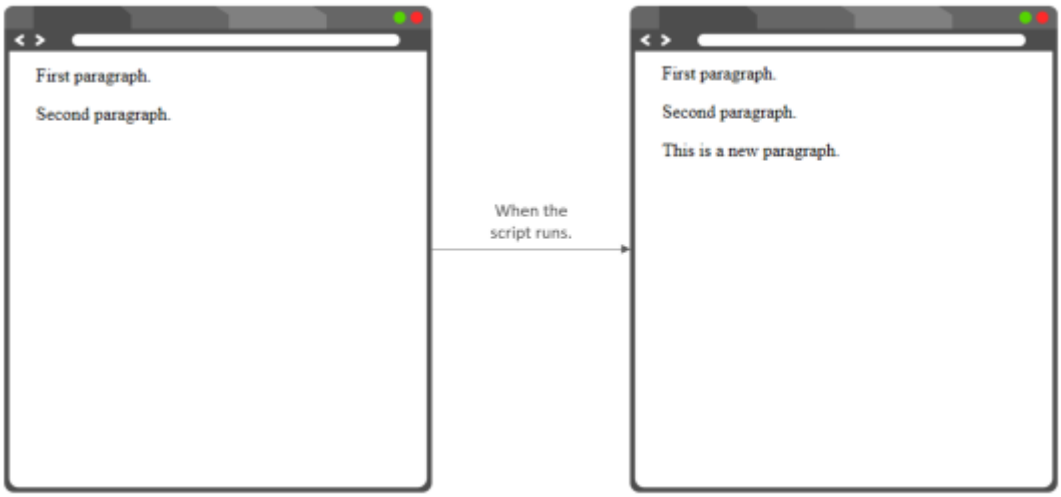
Metoda	**Popis**
appendChild(node)	Přidá k prvku nový podřízený prvek (uzel).
insertBefore(new, existing)	Vloží podřízený uzel před stávajícího podřízeného.
replaceChild(new, old)	Nahradí podřízený uzel novým uzlem.
remove()	Odebere prvek z dokumentu. Nemá žádné parametry.
removeChild(node)	Odstraní podřízený prvek.

5.1 Vytváření prvků DOM HTML (uzlů)

V této kapitole se zaměříme na to, jak můžeme přidávat, odebírat nebo nahrazovat prvky HTML DOM.

Chceme-li přidat nový prvek, musíme nejprve vytvořit uzel prvku a poté jej přidat ke stávajícímu prvku. Zde je příklad kódu, ve kterém jsme přidali nový odstavec s podřízenými uzly.

```
<div id="div01">
 <p id="p01">První odstavec.</p>
 <p id="p02">Druhý odstavec.</p>
</div>
<script>
var newParagraph = document.createElement("p");
var textForParagraph = document.createTextNode("Toto je nový odstavec");
newParagraph.appendChild(textForParagraph);
var element = document.getElementById("div01");
element.appendChild(newParagraph);
</script>
```



Obr. 3. Jak vypadá kód v prohlížeči bez skriptu a se skriptem.

Metoda **createElement(“p”)** vytvoří nový prvek <p>. Chceme-li do tohoto prvku přidat text, musíme vytvořit textový uzel pomocí metody **createTextNode**. Poté musíme textový uzel přidat k prvku <p> pomocí metody **appendChild**. Nakonec musíme nový prvek přidat k existujícímu prvku div.

Nyní použijeme stejný příklad, ale odstavec vložíme na první pozici pomocí metody **insertBefore** namísto na poslední pozici, jak jsme to udělali pomocí metody **appendChild**.

Video 3. Příklad použití metody insertBefore

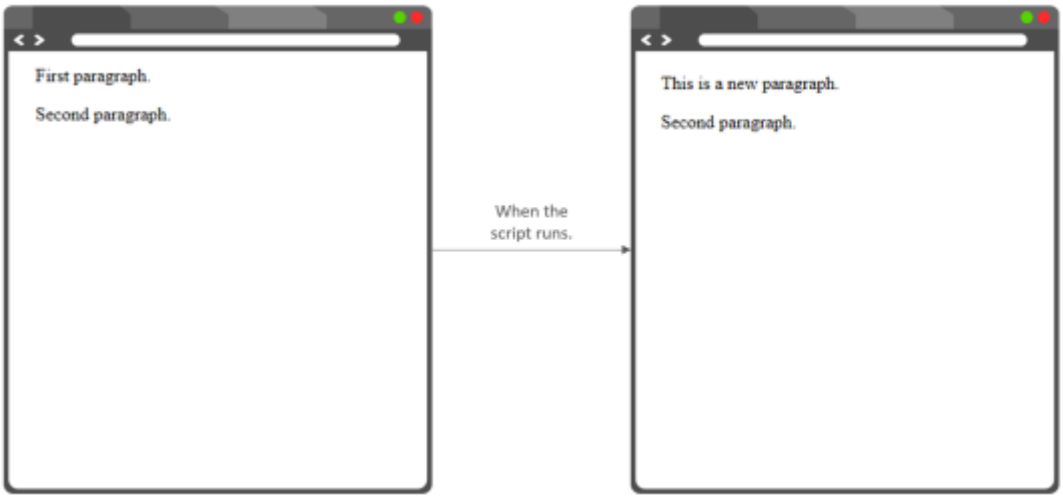
Interaktivní prvek

5.2 Nahrazování prvků DOM HTML (uzlů)

Chceme-li nahradit prvek, musíme nejprve vytvořit uzel prvku a poté jej nahradit již existujícím prvkem. Zde je příklad kódu, ve kterém jsme nahradili stávající odstavec novým odstavcem.

PŘÍKLAD

```
<div id="div01">
 <p id="p01">První odstavec.</p>
 <p id="p02">Druhý odstavec.</p>
</div>
<script>
var newParagraph = document.createElement("p");
var textForParagraph = document.createTextNode("Toto je nový odstavec");
newParagraph.appendChild(textForParagraph);
var parent = document.getElementById("div01");
var child = document.getElementById("p01");
parent.replaceChild(newParagraph,child);
</script>
```



Obr. 4. Jak vypadá kód v prohlížeči bez skriptu a se skriptem.

Interaktivní prvek

5.3 Odstranění prvků DOM HTML (uzlů)

V této kapitole si ukážeme animaci, ve které odstraníme jeden prvek pomocí metody remove() a jeden prvek odstraníme pomocí přístupu k rodiči pomocí metody removeChild(). Metoda remove() nefunguje ve starších prohlížečích, a proto někdy musíme použít metodu removeChild().

V příkazu removeChild musíme vyhledat rodiče, abychom mohli odstranit potomka.

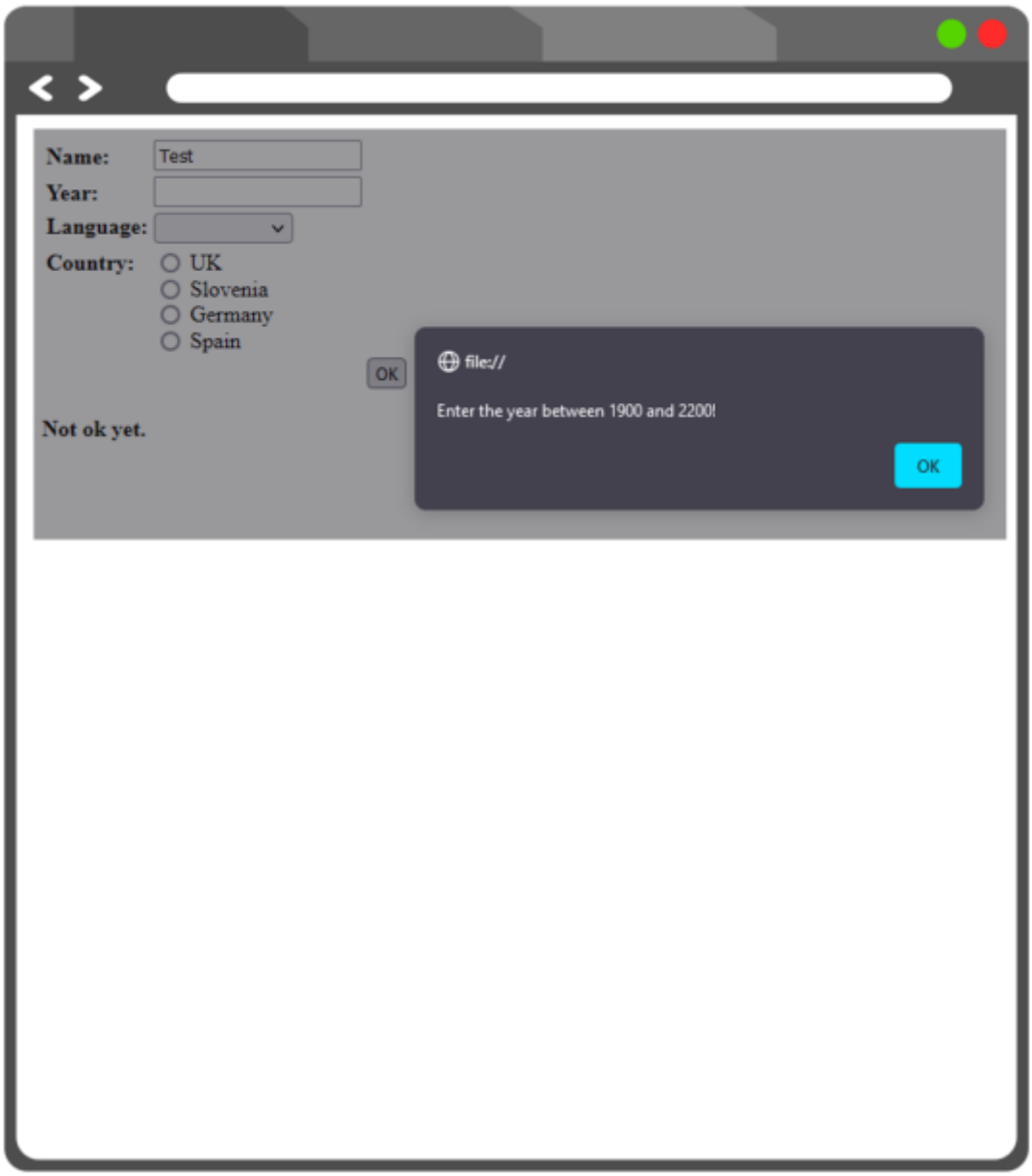
Animace 3. Dva příklady odstranění prvků HTML pomocí DOM

Interaktivní prvek

KAPITOLA 6

Ověřování formulářů s využitím DOM

Validaci formuláře HTML můžeme provádět pomocí JavaScriptu s využitím DOM. Obvykle chceme zkontrolovat, zda uživatel vyplnil správné údaje ve správném formátu. Chceme se ujistit, že uživatel zadal údaje správně. V tomto příkladu vytvoříme jednoduchý formulář a pomocí funkce DOM zkontrolujeme, zda uživatel zadal správná data.



Obr. 5. Příklad varování a formuláře, který bude uveden v následujícím kódu.

Vytváříme formulář se 4 různými možnostmi zadávání dat. Pro lepší zobrazení jsme použili tabulku.

Ve formuláři jsme použili textové pole, výběr z možností, výběr pomocí tlačítka typu radio button a tlačítko odeslat.

```
<form onsubmit="return validation( this)" action="#">
<table>
<tr>
<td><b>Jméno: </b></td>
<td><input type="text" name="name"></td>
</tr>
<tr>
<td><b>Rok: </b></td>
<td><input type="text" name="year"></td>
</tr>
<tr>
<td valign="top"><b>Jazyk: </b></td>
  <td>
  <select name="language">
<option> </option>
<option value="eng">anglicky</option>
<option value="slo">slovinsky</option>
<option value="nem">německy</option>
<option value="fra">francouzsky</option>
</select>
</td>
</tr>
<tr>
<td valign="top"><b>Země: </b></td>
<td>
<input type="radio" name="country" value="1"> Velká Británie<br/>
<input type="radio" name="country" value="2"> Slovinsko<br/>
<input type="radio" name="country" value="3"> Německo<br/>
<input type="radio" name="country" value="4"> Španělsko
  </td>
</tr>
<tr>
<td></td>
<td></td>
  <td><input type="submit" value="OK"></td>
</tr>
</table>
</form>
<script type="text/javascript">
```

Zde je vytvořena ověřovací funkce. Pokud uživatel něco nevyplní správně, zobrazí se okno s upozorněním.

```
function validation( form)
{
```

Pomocí DOM jsme získali přístup k formuláři, položce jméno a k její hodnotě. Pokud je tento údaj prázdný, zobrazí se upozornění.

```
if ( form. name. value == "")
{
alert( "Zadejte jméno! ")
return false
}
```

Pokud je zadaný rok mimo rozsah 1900 a 2200, zobrazí se upozornění. Pomocí funkce isNaN také zkontrolujeme, zda uživatel zadal číslo.

```
if ( isNaN( form. year. value) ||  form. year. value < 1900 ||  form. year. value >
2200)
{
alert( "Zadejte rok mezi 1900 a 2200! ")
return false
}
```

Dále se zobrazí upozornění, pokud uživatel v rozevíracím seznamu jazyků nevybral žádný z jazyků.

```
if ( form. language. selectedIndex <= 0)
  {
alert( "Vyberte jazyk! ")
return false
  }
```

A nakonec, pokud uživatel nevybral žádnou ze zemí pomocí přepínače, zobrazí se mu výstražné okno.

```
var i = 0
while ( i < form. country. length && ! form. country[ i]. checked)
++i

if ( i == form. country. length)
{
alert( "Vyberte zemi! ")
return false
}
return true;
}
</script>
```

Interaktivní prvek

KAPITOLA 7

Změny CSS pomocí DOM

DOM se často používá ke změně stylu prvků HTML. Syntaxe je následující:

document.getElementById(id).style.property = new style;

Návštěvník webových stránek obvykle klikne na nějaký prvek, například na tlačítko, a v dokumentu může dojít ke změně stylu CSS. Těmto událostem se také říká DOM events. Syntaxe je následující:

onclick=JavaScript

Interaktivní prvek

V následujícím příkladu se podíváme na některé vlastnosti a události, které můžeme použít.

Animace 4. Příklad změny CSS pomocí DOM v CSS

PŘÍKLAD

```
<form name="newForm">
<p id='p01'>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
```

Tlačítko níže změní barvu textu výše na zelenou.

```
<input type='button' onclick='document.getElementById("p01").style.color
= "green";' value='Change text color'/><br/><br/>
```

Toto tlačítko změní velikost písma výše uvedeného textu na 27px.

```
<input type='button'
onclick='document.getElementById("p01").style.fontSize = "27px";'
value='Change font size'/><br/><br/>
<p id='p02'>Zde bude zobrazen aktuální čas. </p>
```

Toto tlačítko zavolá funkci s date, která se nachází v části <script>, a zobrazí aktuální čas ve výše uvedeném odstavci.

```
<input type='button' onclick='date()' value='Zobraz aktuální
čas'/><br/><br/>
```

První uvedené zadávací pole představuje pole, do kterého uživatel napíše barvu. Další je zaškrtávací políčko a poslední je tlačítko, které obsahuje funkci, která změní pozadí dokumentu na

danou barvu při zaškrtnutém zaškrtávacím políčku.

```
<input type='text' id='entry' value='Zadejte barvu pozadí' size='40' />
<br/><br/>
<input type="checkbox" id="box" value="box"> Změň barvu pozadí
<input type='button' onclick='changeBackground()' value='Změnit pozadí'/>
</form>
<script>
```

Tato funkce zobrazí aktuální datum a čas.

```
function date() {
 document.getElementById("p02").innerHTML = Date();
}
```

Tato funkce změní barvu pozadí na barvu, kterou uživatel zadal do políčka, a to pouze v případě, že je zaškrtávací políčko zaškrtnuto.

```
function changeBackground() {
var checkbox = document.forms[0].box;
if (checkbox.checked) {
var body = document.getElementsByTagName("body")[0];
body.style.background = document.getElementById("entry").value;
}
}
</script>
```

Interaktivní prvek

KAPITOLA 8

Práce s tabulkami pomocí DOM

V tomto příkladu vytvoříme seznam úkolů v HTML tabulce. S tabulkami můžeme manipulovat přidáváním nebo odstraňováním řádků, záhlaví, buněk atd.

Tabulka 8. Metody pro přidávání a odstraňování různých prvků v tabulkách

Metoda	Popis
deleteRow()	Odstraní <tr> z tabulky.
insertRow()	Vytvoří prázdný prvek <tr> v tabulce.
deleteCell()	Odstraní buňku z aktuálního řádku tabulky.
insertCell()	Vytvoří buňku do aktuálního řádku tabulky.

V níže uvedeném příkladu vytvoříme seznam úkolů, do kterého můžeme přidávat a odebírat tabulky kliknutím na zaškrtávací políčko.

PŘÍKLAD

Nejprve vytvoříme dvě tlačítka pro přidání a mazání úkolu a textové pole, do kterého může uživatel napsat popis úkolu. Funkce se volají pomocí funkce onclick a nacházejí se v sekci <script>.

```
<button onclick="addTask()">Přidat nový úkol</button>
<button onclick="deleteTask()">/Úkol je hotový</button>
<input type="text" id="textbox" placeholder="Text úkolu"><br><br>
```

Jedná se o jednoduchý tabulku pouze s nadpisem, bez jakýchkoli úkolů. Ty budou přidány pomocí funkcí.

```
<table id="table" style="width: 50%">
<tr>
  <th>Zaškrtávátko</th>
  <th>Číslo řádku </th>
  <th>Úkol</th>
</tr>
</table>

<script>
```

Nejprve vytvoříme funkci pro přidávání nových úloh do tabulek. Musíme vyhledat tabulku a v tomto příkladu použijeme getElementById.

```
var count_lines=0;
function addTask() {
 var table = document.getElementById("table");
```

Dále je třeba vložit řádky odspodu nahoru pomocí metody insertRow a parametru -1 (aby úloha přešla na spodní část). Poté se do tabulky vloží tři buňky odpovídající třem sloupcům.

```
var row = table.insertRow(-1);
 var cell1 = row.insertCell(0);
 var cell2 = row.insertCell(1);
 var cell3 = row.insertCell(2);
```

Dále přidáme hodnotu 1 do proměnné count_lines pro druhý sloupec, který počítá aktuální počet přidaných úloh.

```
count_lines++;
```

Do první buňky se vloží zaškrtávací políčko. Do druhé buňky se vloží čítač řádků a do třetího pole se vloží hodnota textu z textového pole.

```
cell1.innerHTML = `<input type="checkbox" name="box" value="0">`;
 cell2.innerHTML = count_lines;
 cell3.innerHTML = document.getElementById("textbox").value;
}
```

Dále vytvoříme funkci pro mazání dokončených úloh v tabulce. Opět musíme vyhledat tabulku a v tomto příkladu použijeme getElementById.

```
function deleteTask() {
 var table = document.getElementById("table");
```

Dále jsme vytvořili proměnnou i, abychom našli řádek, kde je zaškrtnuté políčko, a poté můžeme řádek, který uživatel vybral, odstranit jako vyřízený. Použili jsme childNodes a zkontrolovali, zda je zaškrtávací políčko zaškrtnuté.

```
let i;
 for (i = table.rows.length-1; i >= 1; i--) {
if(table.rows[i].cells[0].childNodes[0].checked==true){
  table.deleteRow(i);
}
  }
}
</script>
```

Interaktivní prvek

Interaktivní prvek

6. Zobrazení to-do listu v prohlížeči

Obr. 6. Zobrazení to-do listu v prohlížeči

Interaktivní prvek

Toto byl úvodní kurz do jazyka JavaScript DOM.

KAPITOLA 9

Test

Kde se zobrazuje značka <title>?

- ◯ uvnitř zančky <body>
- ◯ uvnitř zančky <head>
- ◯ uvnitř zančky <p>
- ◯ uvnitř zančky <img>

Která značka slouží k ztučnění textu?

- ◯ <u>
- ◯ <i>
- ◯ <b>
- ◯ <q>

Jak vytvoříte komentář v jazyce HTML?

- ◯ <!-- komentář -->
- ◯ /* komentář */
- ◯ // komentář

Která značka vytváří seznam s tečkami vpředu?

- ◯
- ◯ <dl>
- ◯

- ◯

Která značka musí být v HTML, aby bylo možné spustit kód Javascriptu?

- ◯ <js>
- ◯ <src>
- ◯ <script>
- ◯ <body>

Může mít funkce v jazyce JavaScript více parametrů?

- ◯ ano
- ◯ ne

Které znaménko musíme v tomto příkladu uvést, chceme-li zjistit, zda hodnota x je 5, a typ proměnné není důležitý? if (x ??? 5) {};

- ◯ ==
- ◯ =
- ◯ ===
- ◯ ====

Jaký je první objekt v HTML DOM?

- ◯ <p>
- ◯ <html>
- ◯ document
- ◯ <head>

Chceme změnit značku <p>. Jakou metodu můžeme použít, pokud v zančky <p> není žádné id?

- ◯ setAttribute
- ◯ getElementsByClassName
- ◯ getElementById
- ◯ getElementsByTagName

Které prvky můžeme použít pro vytvoření nových prvků HTML?

- ◯ setAttribute
- ◯ createElement
- ◯ getElementById
- ◯ appendChild

Proč se používá innerHTML?

- ◯ pro přístup k obsahu dokumentu HTML
- ◯ pro přístup k obsahu zančky <html>
- ◯ k přechodu na porřízený prvek z aktuálního prvku

Co dělá metoda appendChild()?

- ◯ připojí podřízený prvek podřízenému prvku
- ◯ připojí nový prvek k rodičovskému prvku
- ◯ připojí podřízený prvek k sousednímu prvku

Co znamená childNodes[1]?

- ◯ vybere druhý prvek v poli
- ◯ vybere první prvek v poli
- ◯ vybere stejný prvek jako lastChild

Kterou metodu můžeme použít, pokud chceme přidat nový element k již existujícímu elementu na posledním místě?

- ◯ appendChild
- ◯ insertBefore
- ◯ replaceChild

Kterou možnost můžeme použít ve formuláři pro volání funkce?

- ◯ action
- ◯ onsubmit
- ◯ href

Jakou vlastnost můžeme použít pro přístup k obsahu pole ve formuláři HTML?

- ◯ isNaN
- ◯ year
- ◯ value

Čím můžeme přistupovat k číslu vybrané možnosti v rozevíracím seznamu?

- ◯ checked
- ◯ selectedIndex
- ◯ value

Čím můžeme zjistit, zda bylo pole ve formuláři vybráno?

- ◯ checked
- ◯ selectedIndex
- ◯ value

Můžeme měnit CSS pomocí DOM bez použití událostí?

- ◯ ano
- ◯ ne

Pomocí jaké vlastnosti můžeme změnit barvu textu?

- ◯ style.font-color
- ◯ style.color
- ◯ style.background-color

Pomocí jaké vlastnosti můžeme změnit barvu pozadí?

- ◯ style.background
- ◯ style.color
- ◯ style.background-color

Jak můžeme vytvořit nový <tr> v tabulce?

- ◯ insertCell()
- ◯ insertRow
- ◯ insertTableRow

Jaké jsou prvky v jazyce HTML?

- ☐ username
- ☐ začátek značky
- ☐ obsah
- ☐ konec značky

Co je to prázdný element?

- ☐ značka <>
- ☐ značka, která nemá koncovou značku
- ☐ prvek, který nemá obsah
- ☐ prvek bez atributu

Kde se může nacházet značka <script>?

- ☐ uvnitř značky <head>
- ☐ uvnitř značky <table>
- ☐ uvnitř značky <body>
- ☐ před značkou <!DOCTYPE html>

Text v jazyce JavaScript se zapisuje uvnitř:

- ☐ zpětných lomítek
- ☐ jednoduchých uvozovek
- ☐ pro začátek textu nejsou potřeba žádné znaky
- ☐ uvnitř dvojitých uvozovek

Která slova deklarují proměnnou v jazyce JavaScript?

- ☐ var
- ☐ const
- ☐ for
- ☐ let

Které uzly můžeme použít k navigaci mezi <html> a <head>?

- ☐ sibling
- ☐ firstChild
- ☐ sisterNode
- ☐ parentNode

Jaké vztahy mohou mít elementy <body> a <head>?

- ☐ previousSibling
- ☐ nextSibling
- ☐ firstChild
- ☐ parentNode

Co z těchto příkladů představuje uzel?

- ☐ HTML element
- ☐ text v HTML elementech
- ☐ atribut HTML elementu
- ☐ celý HTML element

Které metody můžeme použít k vytvoření nového odstavce s uzly?

- ☐ value()
- ☐ createTextNode()
- ☐ getElementById()
- ☐ createElement()

Které metody můžeme použít, pokud chceme odstranit element?

- ☐ removeParent()
- ☐ removeSibling()

- ☐ remove()
- ☐ removeChild()

Které metody potřebujeme, pokud chceme vytvořit nový <td> v novém <tr>?

- ☐ insertRow()
- ☐ checked()
- ☐ insertCell()
- ☐ deleteCell()