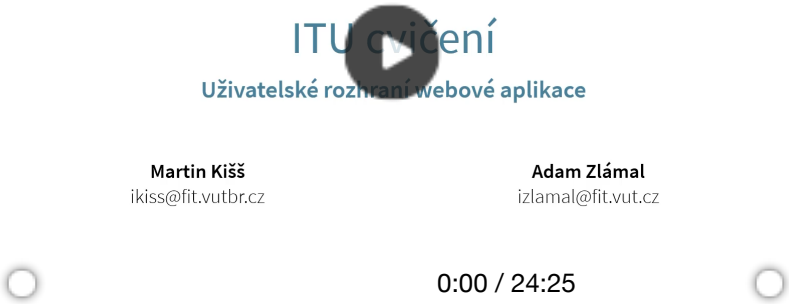
Uživatelské rozhraní webové aplikace



Obsah

Úvod

Moderní webové stránky (např. YouTube, Facebook, různé e-shopy a další) dokáží **dynamicky měnit svůj obsah** pomocí dat získaných z **externího zdroje** (nějakého serveru), **bez nutnosti stránku obnovovat**. Typicky takovou funkcionalitou může být načítání dalšího obsahu při scrollování, aktualizace vizualizovaných dat v reálném čase a další. V rámci tohoto cvičení si vyzkoušíte vytvořit webovou stránku, která bude z poskytnutého serveru **získávat nejnovější zprávy** a **nové zprávy na server odesílat**.

Zadání cvičení

Cílem tohoto cvičení je vytvořit chatovací aplikace pomocí **HTML**, **JavaScriptu** a přichystaného serveru. Je pro vás přichystána kostra aplikace, kterou budete upravovat. Výsledná aplikace by měla **v pravidelném intervalu získat ze serveru nejnovější zprávy**, zpracovat je a vypsat do příslušného okna. Zároveň by také měla umět **odesílat zprávy na server**. **To vše by mělo fungovat bez obnovování stránky, pouze pomocí zasílání požadavků na server pomocí AJAXu**.

Hodnocení

Výsledný program je hodnocen na základě jeho předvedení. V jednom okně mějte spuštěný chat, ve druhém zdrojové soubory. Do WISu odevzdávejte soubor `itu-ajax.html`. Za cvičení můžete získat až **5 bodů** za:

- 1 bod - zobrazuje se "okno" obsahující zprávy přímo ze serveru (ve formátu JSON)
- 1 bod - zobrazují se naformátované (zpracované) zprávy
- 1 bod - je možné vložit novou zprávu,
- 1 bod - v "okně" se zprávami se objevují nové zprávy - ihned po odeslání vlastní zprávy i automaticky v časových intervalech,
- 1 bod - nepřekresluje se celé okno, pouze se přidávají nové zprávy (id nových zpráv zobrazujte ve stavovém řádku).

Upozornění - ze serveru získáte vždy 20 nejnovějších zpráv. Je možné, že některou ze zpráv jste již dříve zpracovali a je tedy nutné kontrolovat, zda se má daná zpráva uživateli vypsat.

Nastavení WWW adresáře

Na serveru `eva.fit.vutbr.cz` v domovském adresáři musí být vytvořený adresář `WWW` a v něm musí být vytvořený soubor `index.html` nebo `index.php`. Všechny adresáře (i `xlogin00`) a soubory musí mít nastavené správné opravnění. Po cvičení je vhodné nastavit práva zpět, aby Váš adresář nebyl přístupný všem. Celý postup pro nastavení adresáře je následující:

```
mkdir ~/WWW/
touch ~/WWW/index.html
chmod 0755 ~/../`whoami`
chmod 0755 ~/WWW/
chmod 0644 ~/WWW/index.html
```

```
chmod 0644 ~/WWW/itu-ajax.html
```

V případě, že budete mít správně nastavený adresář, je možné k němu přistoupit z prohlížeče zadáním adresy ve tvaru `http://www.stud.fit.vutbr.cz/~xlogin00/nazev-vaseho-souboru.html`. Pro přístup k serveru `eva.fit.vutbr.cz` můžete využít několik možností:

1. SSH
   - ve Windows přes program `PUTTY`, na linuxu v terminálu `ssh login@eva.fit.vutbr.cz`
   - měli byste se dostat na server přes konzolovou aplikaci (to je vhodné především pro nastavení oprávnění k adresáři)
2. FTP klient
   - je možné použít nějaký FTP klient (např. WinSCP, aj.), po vyplnění údajů a připojení byste se měli dostat do domovského adresáře (stejný adresář, jako v případě ssh připojení)
   - zde je pak možné přímo editovat jednotlivé soubory
3. WIS
   - ve WISu je jednoduchý FTP klient (`[v horním menu] ostatní → FTP klient → eva.fit.vutbr.cz`), opět byste měli vidět svůj domovský adresář, jedná se však o nejméně pohodlnou variantu pro práci

V laboratořích na OS Windows se programy PUTTY a WinSCP nacházejí v adresáři Q:\netapp.

AJAX

AJAX je zkratkou pro Asynchronous JavaScript and XML. AJAX umožňuje JavaScriptu **na straně klienta na pozadí zasílat požadavky na server** a získat tak nová data k zobrazení, aniž by došlo ke znovunačtení zobrazované stránky. Lze tak docílit mnohem dynamičtějších webových stránek, jejichž chování může být z hlediska uživatelských rozhraní mnohem inteligentnější.

Požadavky

K zasílání požadavků využívá AJAX protokol HTTP. Tento protokol se používá pro komunikaci mezi klientem (v tomto případě prohlížečem) a serverem. Typicky spolu komunikují prohlížeč se serverem tak, že prohlížeč zašle **HTTP**

požadavek na server a ten mu základě požadaveku **zašle odpověď** (požadovaná data). Každý požadavek má atributy, které je před odesláním potřeba nastavit. Jedná se především o URL adresu, HTTP metodu a případná odesílaná data.

HTTP definuje několik metod, které lze v rámci požadavků použít. V rámci této aplkace se však omezíme pouze na metody **GET** a **POST**, které jsou nejpoužívanější. Typicky se metoda GET používá pro **získání dat ze serveru** a případné parametry se posílají zakódované v rámci URL adresy (`<URL>?parameter1=value1&parameter2=value2`).
Metoda POST se používá pro **odeslání formulářových dat na server**. Detailnější rozdíl mezi oběma metodama je popsán např. na stránkách W3Schools dole.

Důležitým atributem požadavku v rámci AJAXu je tzv. readyState. ReadyState vyjadřuje **aktuální stav požadavku** pomocí celočíselné proměnné, přičemž stav označený jako `0` znamená, že požadavek ještě nebyl inicializován, a hodnota `4` označuje stav, kdy již byla získána celá odpověď ze serveru.

Odpovědi

Na každý požadavek server odesílá odpověď. Důležitým atributem odpovědi je její status. Pomocí něj se určuje, zda se daný požadavek podařilo zpracovat, nebo došlo k chybě. Status odpovědi typicky zobrazují také prohlížeče, kdy například při chybně zadané URL dostaneme známý kód `404` (nenalezeno). Pokud vše proběhne v pořádku, používá se kód `200` (OK). Druhým důležitým atributem jsou získaná data, která je potřeba následně zpracovat.

Užitečné kódy

Vytvoření požadavku

Kód pro vytvoření požadavku je součástí kostry programu. Zde je uveden pouze pro úplnost.

```
 1.    /***
 2.     * XMLHttpRequest object
constructor (for compatibility with
various browsers)
 3.     */
 4.    function
createXmlHttpRequestObject()
 5.    {
 6.        var request;
```

```
 7.
 8.          try
 9.          {
10.              request = new
XMLHttpRequest(); // should work on
all browsers except IE6 or older
11.          }
12.          catch (e)
13.          {
14.              try
15.              {
16.                  request = new
ActiveXObject("Microsoft.XMLHttp");
// browser is IE6 or older
17.              }
18.              catch (e)
19.              {
20.                  // ignore error
21.              }
22.          }
23.
24.          if (!request)
25.          {
26.              alert("Error creating
the XMLHttpRequest object.");
27.          }
28.          else
29.          {
30.              return request;
31.          }
32.      }
```

Odeslání požadavku

Abychom mohli odeslat požadavek, je potřeba jej nejprve **vytvořit a správně inicializovat**. Inicializace se provádí pomocí metody `open`, které je nutné předat parametry: metodu požadavku (v našem případě `"GET"` nebo `"POST"`), URL adresu serveru, na kterou chceme požadavek zaslat, a hodnotu typu bool, která vyjadřuje asynchronnost požadavku (v našem případě `true`). Protože je požadavek asynchronní (po odeslání se nečeká na odpověď, ale pokračuje se ve vykonávání dalšího kódu), je potřeba zpracovat odpověď až ve chvíli, kdy je oznámeno, že již přišla celá odpověď. K tomuto účelu slouží událost (event) `onreadystatechange`. Tato událost je vyvolána vždy, když se změní hodnota atributu `readyState` (viz předchozí část). Poté, co nastane změna hodnoty, vyvolá se kód (callback), který je pro tuto událost

naprogramovaný. Jako callback lze použít buďto **ukazatel na funkci**, nebo použít takzvanou **anonymní funkci**. V našem případě je potřeba otestovat, jakou hodnotu má aktuálně atribut `readyState` a jaký je status kód odpovědi (potřebné hodnoty těchto proměnných jsou uvedeny výše). Odpověď ze serveru se potom nachází v atributu `responseText`.

Pro odeslání požadavku se používá metoda `send`. Pokud chceme odeslat nějaká data, dávají se jako parametr této metody. V případě, že žádná data neodesíláme, je možné použít hodnotu `null`, nebo parametr vůbec nezadat. Při odesílání dat je potřeba také určit typ, jakým jsou data zadána. To se dělá pomocí **hlaviček** (metoda `setRequestHeader` objektu požadavku), kdy každá hlavička má nějaký název (v tomto případě `"Content-Type"`) a hodnotu (zde `"application/x-www-form-urlencoded;"`). Celé nastavení hlavičky by pak mělo vypadat takto: `request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded;");`. Hlavičky je potřeba vyplnit ještě před samotným odesláním požadavku.

```
 1.    function send()
 2.    {
 3.        var request =
createXmlHttpRequestObject();
 4.        request.open(METHOD, URL,
true);
 5.        request.onreadystatechange
= function() // anonymous function (a
function without a name).
 6.        {
 7.            if
((request.readyState ==
READY_STATE_FINISHED) &&
(request.status == STATUS_OK)) //
process is completed and http status
is OK
 8.            {
 9.
alert(request.responseText);
10.
alert(request.responseXML);
11.            }
12.        }
13.
14.        request.send(null);
15.    }
```

Server

Server, který máte v rámci cvičení k dispozici, má URL ~~http://www.stud.fit.vutbr.cz/~xmlich02/itu-ajax/api.php~~ http://pckiss.fit.vutbr.cz/itu/api.php. Všichni budete získávat i odesílat data na tento server, tudíž budete mít jeden společný chat, který všichni uvidí a do kterého budou všichni přispívat svými zprávami.

Důležité - je potřeba, aby vaše webová stránka byla ~~ve stejné doméně jako server, tj.~~ v doméně www.stud.fit.vutbr.cz! Postup pro správné nastavení se nachází v úvodní části.

Popis API

Získání zpráv

Pro získání nejnovějších dvaceti zpráv zašlete GET požadavek na URL serveru bez jakýchkoliv parametrů. Server vám v textu odpovědi vrátí JSON, kde pro každou zprávu je uvedeno její ID, login autora, obsah zprávy a časové razítko, kdy byla zpráva odeslána.

Odeslání zprávy

Pro odeslání zprávy je potřeba zaslat POST požadavek na URL serveru s tím, že v obsahu požadavku bude vaše zpráva ve formátu `"data=<obsah_zprávy>"`. Žádné další informace serveru zasílat nemusíte (~~váš login jde zjistit z URL, ze které je požadavek posílán,~~ identifikační číslo i časové razítko zprávě server přidělí sám). Volitelně je možno zaslat informaci o jménu autora (v případě nezaslání se použije generický autor `xlogin00`) přidáním parametru `user` do odesílaných dat. Celý odesílaný řetězec by pak mohl vypadat například takto: `"user=muj_login&data=Moje zprava"` Server vám v odpovědi vrátí pouze informaci, že byla vaše zpráva uložena na server.

Základy JavaScriptu

JavaScript je hodně volný jazyk, syntaxí se podobá jazyku C.

Vývojářské nástroje

Většina nejnovějších prohlížečů umožňuje přístup k tzv. vývojářským nástrojům (např. v Google Chrome pomocí `Ctrl+Shif+I`, `F12`, nebo přes `Menu → Další nástroje → Vývojářské nástroje`). Zde je možné se podívat na HTML kód zobrazené webové stránky, do konzole a na další užitečné údaje pro ladění stránek. V případě, že v JavaScriptu dojde k chybě (proměnná neexistuje, funkce neexistuje nebo libovolná jiná výjimka), tak prohlížeč ukončí vykonávání skriptu. Výstup se v tomto případě zobrazí právě v konzoli. Pokud v JavaScriptu použijete funkci `console.log()`, do konzole se vypíše text, který ji zadáte jako parametr. Do samotné konzole potom můžete také zapisovat kód, který se ihned na stránce vykoná. Další užitečnou záložkou jsou informace o síti. Zde je možné v čase vidět všechny odesílané požadavky, které odcházejí z webové stránky. Můžete si tak zkontrolovat, jaké údaje odesíláte a jaké odpovědi získáváte.

Užitečné kódy

Vyskakovací okno pro zobrazení dat

```
1.    alert("hello world");
```

Výpis do konzole

```
1.    console.log("hello world");
```

Přísup k elementům dokumentu

```
1.    <div id="myId">myId must be
unique within the web page</div>
2.
document.getElementById("myId").inner
HTML = "<b>new element content</b>";
```

Formulářová data

```
1.    <input type="text" id="myId2"
/>
2.
alert(document.getElementById("myId2"
).value);
```

Spuštění JavaScriptu

```
1.    <a
href="javascript:do_something()">star
t</a>
2.    <form onsubmit="return
do_something()"> <input
type="submit"> </form> <!-- function
do_something() should return false as
the form is processed by JavaScript
-->
3.    <script> do_something();
</script>
```

Opakované spouštění funkce

```
1.    setTimeout(function_pointer,
```

```
3000); // calls function once after 3
seconds; a function pointer is only
the name of a function (without ())
2.    setInterval(function_pointer,
3000); // calls function periodically
after 3 seconds; a function pointer
is only the name of a function
(without ())
```

Zpracování JSONu

```
1.    var pole =
JSON.parse(request.responseText);
2.    for (var i in pole)
3.    {
4.        console.log(pole[i]);
5.    }
```

Literatura

- Jak psát web
- W3Schools - JavaScript
- W3Schools - JSON
- W3Schools - AJAX
- W3Schools - HTTP
- kolektiv autorů: AJAX a PHP - tvoříme interaktivní webové aplikace, Zoner Press, 2006.
- Asleson, R., Schutta, N. T.: AJAX - vytváříme vysoce interaktivní webové aplikace, Computer Press, 2006.