Kurs Front-End Developer

AJAX

Asynchronous Javascript And Xml

AJAX

Asynchronous Javascript And Xml

Ajax – to kombinacja działań, która pozwala pobrać z serwera dane i umieścić je na stronie, bez ponownego przeładowania tej strony. Zapytania do serwera są wysyłane Asynchronicznie do załadowanej strony.

TECHNOLOGIE w AJAX

Technologie składające się na AJAX:

- HTML
- CSS
- XML
- JSON
- XMLHttpResponse object
- Javascript

PROTOKÓŁ HTTP

Pobieranie stron internetowych z serwerów opiera się o **protokół HTTP** (dlatego początek adresu internetowego zaczyna się od **http://** lub **https://**).

Przeglądarka wysyła do serwera żądanie. Serwer je przetwarza, po czym wysyła odpowiedź poprzedzoną kodem odpowiedzi.
Niektóre kody już znamy ☺ są bardzo popularne np. **404** – strona nie istnieje
Lub **401** brak autoryzacji.

Standardowo strona internetowa aby pobrać zawartość z serwera i ją wyświetlić musi **przeładować całą stronę**.

Korzystając z **AJAX** możemy przeładować i zmienić część strony, robiąc to bez przeładowania całej strony. Robimy to **asynchronicznie**, więc użytkownik może dalej korzystać ze strony w tym czasie

Obiekt XMLHttpRequest (I-***)

Głównym zadaniem AJAX jest połączenie się z serwerem za pomocą obiektu **XMLHttpRequest**.

Starsze wersje IE (<7) używają do tego celu bibliotek ActiveX, reszta przeglądarek wykorzystują do tego celu obiekt **XMLHttpRequest**.

~~IE (<7) już prawie nie istnieje, więc nie jest to już problemem.~~

```
// stwórz obiekt XMLHttpRequest()
var xml = new XMLHttpRequest();
```

Obiekt XMLHttpRequest (I-***)

Metody obiektu XMLHttpRequest()

- **open("metoda", "url", async, "user", "password")** - otwiera połączenie do serwera. Zazwyczaj korzystamy z 3 pierwszych atrybutów - metody (GET/POST), adresu pliku na serwerze, oraz boolowskiej zmiennej określającej czy dane połączenie ma być asynchroniczne. Ze względów bezpieczeństwa, możesz otwierać połączenie tylko ze swoją domeną.
- **send("content")** - wysyła żądanie do serwera
- **abort()** - zatrzymuje żądanie
- **setRequestHeader()** - wysyła konkretny nagłówek do serwera
- **getResponseHeader("nazwa_nagłówka")** - pobiera konkretny nagłówek http
- **getAllResponseHeaders()** - pobiera wszystkie wysłane nagłówki http jako string

Obiekt XMLHttpRequest (I-***)

Właściwości obiektu XMLHttpRequest():

- **onreadystatechange** - zdarzenie odpalane w chwili zmiany stanu danego połączenia
- **readyState** - zawiera aktualny status połączenia (0: połączenie nie nawiązane, 1: połączenie nawiązane, 2: żądanie odebrane, 3: przetwarzanie, 4: dane zwrócone i gotowe do użycia)
- **responseText** - zawiera zwrócone dane w formie tekstu
- **responseXML** - zawiera zwrócone dane w formie drzewa XML (jeżeli zwrócone dane są prawidłowym dokumentem XML)
- **status** - zwraca status połączenia np 404 - gdy strona nie istnieje, itp)
- **statusText** - zwraca status połączenia w formie tekstowej - np 404 zwróci Not Found

Tworzymy połączenie pobieramy przez GET (1-***)

```
var xml = new XMLHttpRequest();

xml.open("GET", "https://jsonplaceholder.typicode.com/posts/1", true);
//jeżeli stan dokumentu został zmieniony
xml.onreadystatechange = function() {
    //4 = dokument został w pełni przesłany i jest gotowy do użycia
    if ( xml.readyState == 4 ) {
        // xml.responseXML zawiera zwrócony dokument xml
        // xml.responseText zawiera zwrócony tekst
        // (if no XML document was provided)
        //czyścimy obiekt, dla zwolnienia pamięci
        xml = null;
    }
};
xml.send();
```

Kontrola statusu połączenia HTTP (I-***)

```
if (xml.status == 404) {
    console.warn('strona nie istnieje')
}

if (
    //Każdy status z przedziału 200-300 jest ok
    ( xml.status >= 200 && xml.status < 300 ) ||
    //zwrócone dane są takie same jak w przeglądarce
    (xml.status == 304) ||
    // Safari zwraca pusty ciąg znaków jeżeli zwrócone dane nie są
zmodyfikowane - to też jest ok
    (navigator.userAgent.indexOf("Safari") >= 0 && typeof xml.status ==
"undefined"))
{ alert(xml.responseText); }
```

ODPOWIEDŹ responeText vs responeXML (I-***)

Po nawiązaniu połączenia i sprawdzeniu czy mamy poprawną odpowiedź z serwera, możemy pobrać przesłane dane i ich użyć.

Zwrócone dane mogą być w formacie zwykłego tekstu **responseText** lub w formacie XML – **responseXML**

Pierwszy format zawiera czysty tekst, a drugi format zawiera prawidłowy dokument XML, który jest przerobiony na model DOM. Czyli można się po nim poruszać za pomocą znanych nam metod.

ODPOWIEDŹ responeText vs responeXML (1-***)

```
var xml = new XMLHttpRequest();

xml.open("GET", ”http://adres.serwisu.pl", true);
xml.onreadystatechange = function() {
    if ( xml.readyState == 4 && (r.status >= 200 && xml.status < 300 ||
xml.status == 304 || navigator.userAgent.indexOf("Safari") >= 0 && typeof
r.status == "undefined")) {
        if (xml.responseText=="ok") {
            var ok = document.createTextNode('***OK***');
            document.getElementsByTagName('body')[0].appendChild(ok);
        }
        xml = null;
    }
};
xml.send();
```

AJAX za pomocą jQuery – Obiekty JSON (1-***)

Pobieranie bezpośrednio obiektów JSON

```
$.getJSON('http://echo.jsontest.com/userId/108/userName/Akademia108/
userURL/akademia108.pl', function (data) {

    // wyswietl w konsoli
    console.log(data);
});
```

AJAX za pomocą jQuery - $.ajax() (1-***)

AJAX za pomocą jQuery

```
$.ajax({
    url: "http://echo.jsontest.com/userId/108/userName/Akademia108/
userURL/akademia108.pl",
    dataType: 'json',
    success: function (resultJSON) {

        console.log(resultJSON);
    },
    onerror: function (msg) {
        console.log(msg);
    }
});
```

ZALETY I WADY Ajax

ZALETY:

- Lepsza interaktywność
- Strona jest kompaktowa (nie potrzeba wielu podstron)
- „Lekkie” zapytania do serwera (brak przeładowania)
- Zminiejszenie trasferu danych

WADY:

- Nie potrzebny jest przycisk Refresh
- Działania na stronie kodujemy w Javascript
- Brak adresów URL (przydatne w SEO)

WARSZTATY – JAVASCRIPT BUTTON CLICK

Stwórz przycisk "Pobierz dane". Pod zdarzenie **onclick** tego przycisku podepnij funkcje **pobierzDane(event)**.

Funkcja pobierzDane(event) pobiera z serwera objekt JSON (URL = http://echo.jsontest.com/userId/108/userName/Akademia108/userURL/akademia108.pl). Dane zawarte w pobranym obiekcie wstaw go do strony.

Napisz kod za pomocą czystego Javascript i użyj do tego stworzonej na zajęciach funkcji **ajax()**.

WARSZTATY – jQuery BUTTON CLICK

Stwórz przycisk "Pobierz dane". Za pomocą jQuery, gdy strona już się całkowicie załaduje, podepnij własną funkcję pod zdarzenie **click** tego przycisku.

W tej funkcji pobierz z serwera objekt JSON (URL = http://echo.jsontest.com/userId/108/userName/Akademia108/userURL/akademia108.pl). Dane zawarte w pobranym obiekcie wyświetl w konsoli.

Napisz kod za pomocą jQuery i użyj do tego funkcji **getJSON()**

WARSZTATY – INFINITE SCROLL

Stwórz stronę intenretową zawierającą listę użytkowników (tj. ich ID, IMIE i adres URL). Lista użytkowników ma być dłuższa niż wysokość okna przeglądarki, aby włączał się mechanizm scrollowania.

Następnie podepnij pod zadrzenie **onscroll** funkcję, która sprawdza, czy przewineliśmy stronę do końca.

Za każdym razem, gdy strona zostanie przescrollowana do samego dołu pobierz za pomocą AJAX listę użytkowników w formacie JSON (URL = https://jsonplaceholder.typicode.com/users). Pobranych użytkowników za każdym razem doklej u dołu strony.

Napisz kod za pomocą czystego Javascript i użyj do tego stworzonej na zajęciach funkcji **ajax()**.

Akademia 108

https://akademia108.pl