

Kurs Front-End Developer

AJAX

Asynchronous Javascript And Xml

AJAX

AJAX

Asynchronous Javascript And Xml

Ajax – to kombinacja użycia technologii, która pozwala pobrać za pomocą JavaScript z serwera dane i umieścić je na stronie, bez ponownego przeładowania tej strony – czyli asynchronicznie.

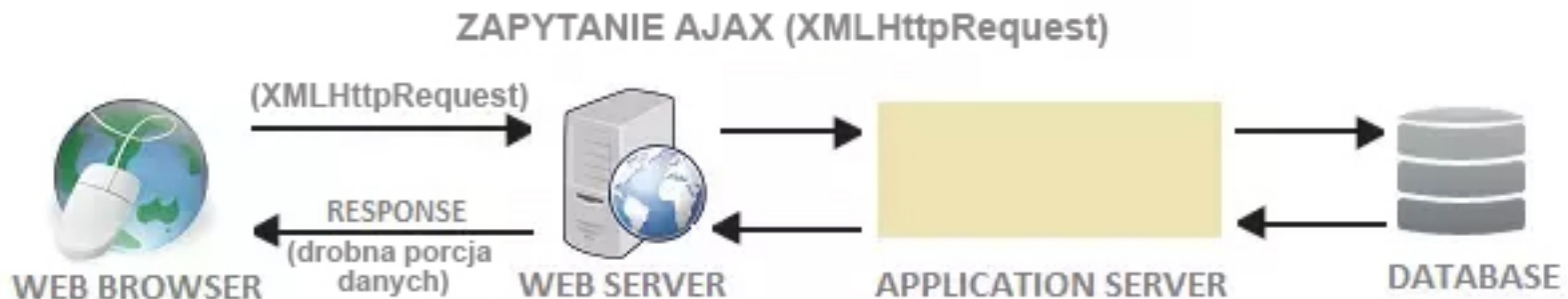
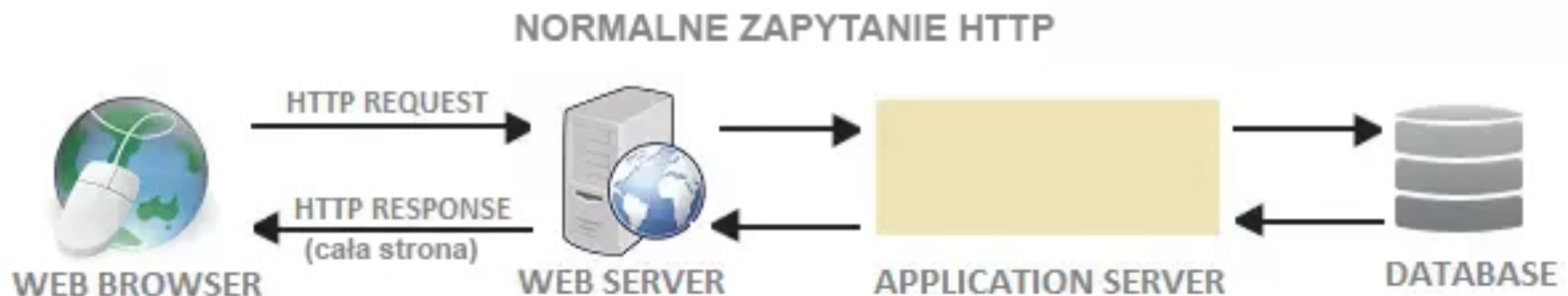
TECHNOLOGIE w AJAX

Technologie składające się na **AJAX**:

- JavaScript (obiekt XMLHttpRequest)
- HTML
- CSS
- XML
- JSON

PROTOKÓŁ HTTP i XMLHttpRequest

Jak to działa?



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 każda przeglądarka pobiera zawartość całej strony internetowej z serwera i ją wyświetla – czyli **przeładowuje całą stronę**.

Korzystając z **AJAX** możemy zmienić tylko część strony, robiąc to bez przeładowania całej strony. Robimy to **asynchronicznie** od normalnego cyklu pobierania stron, więc użytkownik może dalej korzystać ze strony w tym czasie, a strona się nie przeładuje.

Obiekt XMLHttpRequest (I-****)

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

```
// stwórz obiekt XMLHttpRequest()
// więcej informacji nt. samego obiektu:
// https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
// po utworzeniu obiektu, warto go zobaczyć w konsoli
var httpReq = new XMLHttpRequest();

// otwórz połączenie
// https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open
httpReq.open("GET", "https://jsonplaceholder.typicode.com/posts/1");
```

Obiekt XMLHttpRequest (I-****)

```
// jeżeli status połączenia został zmieniony -> httpReq.readyState
httpReq.onreadystatechange = function() {

    // 4 = dokument został w pełni przesłany i jest gotowy do użycia
    // https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState
    if (httpReq.readyState == 4 ) {

        // sprawdź kod statusu połączenia
        // https://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP
        if (httpReq.status == 200) {
            // obsługa odebranych danych
        }

        //czyścimy obiekt, dla zwolnienia pamięci
        httpReq = null;
    }
};
xml.send(); //Na koncu musimy dane wysłać do serwera i poczekać na odpowiedź 😊
```

Obiekt XMLHttpRequest (I-****)

Metody obiektu XMLHttpRequest()

- **open(method, 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 – w 99% przypadków jest to wartość true. Czasami wymagana jest autoryzacja, więc należy podać użytkownika i hasło. Ze względów bezpieczeństwa, możesz otwierać połączenie tylko ze swoją domeną.
- **send("content");** - wysyła żądanie do serwera

Dokładna dokumentacja i opis obiektu XMLHttpRequest znajduje się pod adresem: <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

Obiekt XMLHttpRequest (I-****)

Właściwości obiektu XMLHttpRequest():

- **onreadystatechange** – dopinamy funkcję, która obsłuży 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. 200 – gdy wszystko jest OK i np. 404 - gdy strona nie istnieje, itp)
- **statusText** - zwraca status połączenia w formie tekstowej - np 404 zwróci: Not Found

ODPOWIEDŹ `responseText` vs `responseXML` (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 (często jest tam zapisany obiekt `JSON`), a drugi format zawiera prawidłowy `dokument XML`, który jest przerobiony na model DOM. Czyli można się po nim poruszać za pomocą znanych nam metod.

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

Pobieranie bezpośrednio obiektów JSON

```
$.getJSON(  
    "http://echo.jsontest.com/userId/I08/userName/AkademiaI08/userURL/akademiaI08.pl",  
    function (data) {  
        // wyświetl w konsoli  
        console.log(data);  
    }  
);
```

Dokumentacja i dokładny opis funkcji w obiekcie jQuery - [jQuery.getJSON\(\)](http://api.jquery.com/jquery.getjson/)
<http://api.jquery.com/jquery.getjson/>

AJAX za pomocą jQuery - \$.ajax() (I-***)

AJAX za pomocą jQuery

```
$.ajax({  
    url: "http://echo.jsontest.com/userId/I08/userName/AkademiaI08/userURL/akademiaI08.pl",  
  
    dataType: "json",  
  
    success: function (resultJSON) {  
        console.log(resultJSON);  
    },  
  
    onerror: function (msg) {  
        console.log(msg);  
    }  
});
```

Dokumentacja i dokładny opis funkcji w obiekcie jQuery - [jQuery.ajax\(\)](http://api.jquery.com/jquery.ajax/)
<http://api.jquery.com/jquery.ajax/>

ZALETY I WADY Ajax

ZALETY:

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

WADY:

- Działania na stronie kodujemy w Javascript – przeważnie na stronie jest więcej kodu JS, a więc i utrzymanie strony jest bardziej kosztowne
- Domyślnie brak adresów URL (przydatne w SEO)

WARSZTATY – JAVASCRIPT BUTTON CLICK

Stwórz przycisk “Pobierz dane”. Pod zdarzenie **onclick** tego przycisku podepnij funkcję **pobierzDane(event)**.

Funkcja **pobierzDane(event)** pobiera z serwera obiekt JSON (URL = <http://echo.jsontest.com/userId/I08/userName/AkademiaI08/userURL/akademiaI08.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 obiekt JSON (URL = <http://echo.jsontest.com/userId/I08/userName/AkademiaI08/userURL/akademiaI08.pl>). Dane zawarte w pobranym obiekcie wstaw do strony

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

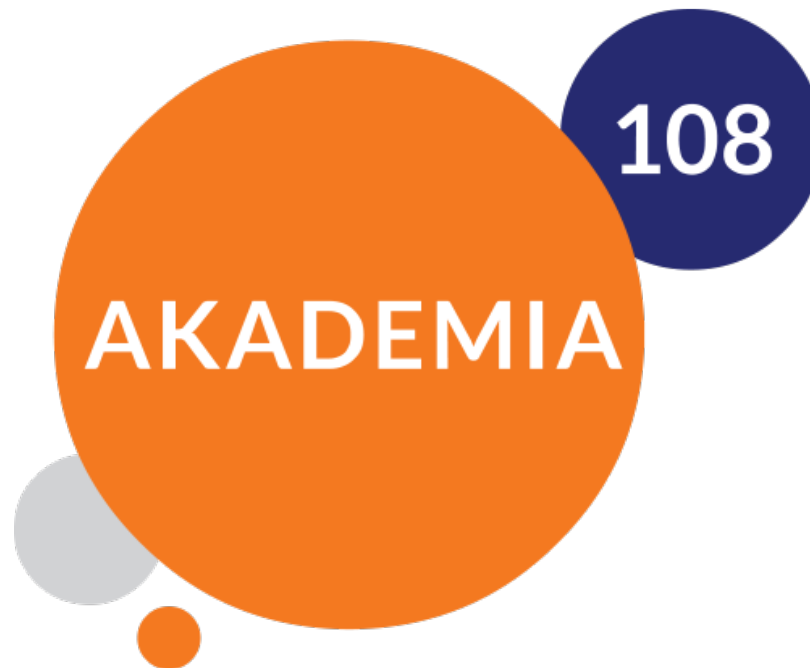
WARSZTATY – INFINITE SCROLL

Stwórz stronę internetową 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 przewiniliśmy 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>