Kurs Front-End Developer

AJAX

Asynchronous Javascript And Xml

AJAX

AJAX – Asynchronous Javascript And Xml

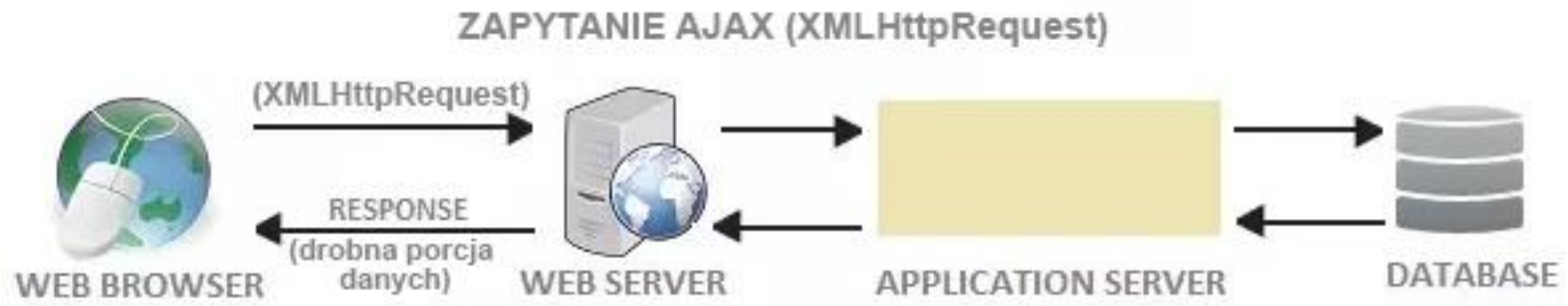
– to kombinacja użycia technologii, która pozwala pobrać za pomocą JavaScript z serwera dane i umieścić je na stronie, bez ponownego przeładowania tej strony – czyli asynchronicznie.

Technologie składające się na AJAX:

- JavaScript (objekt XMLHttpResponse)
- 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.

Fetch API (I-***)

Najbardziej przejrzystą metodą pobierania danych w sposób asychnroniczy jest wykorzystanie wbudowanej w JavaScript funkcji `fetch()`

```
// Pobierz dane z adresu URL - funkcja fetch(zwraca obiekt Promise - Obietnica)
// Jeśli Promise zostanie rozwiązany to fetch() zwróci obiekt odpowiedzi (response)
// Zwracamy obiekt odpowiedzi przetworzony na obiekt JSON
// Od tego momemtu możemy korzystać z tych danych

fetch('https://jsonplaceholder.typicode.com/posts/1')
.then(response =>response.json())
.then(response=>{
    console.log(response);
});
```

Obiekt XMLHttpRequest (2-***)

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 (2-***)

```
// jeżeli status polaczenia zostal 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) {
                // obsluga odebranych danych
        }

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

Obiekt XMLHttpRequest (2-***)

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 (2-***)

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 responeText vs responeXML (2-***)

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 (3-***)

Pobieranie bezpośrednio obiektów JSON

```
$.getJSON(
        "https://jsonplaceholder.typicode.com/users/1",
        function (data) {

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

Dokumentacja i dokładny opis funkcji w obiekcie jQuery - `jQuery.getJSON()`
http://api.jquery.com/jquery.getjson/

AJAX za pomocą jQuery - $.ajax() (3-***)

AJAX za pomocą jQuery

```
$.ajax({
    url: "https://jsonplaceholder.typicode.com/users/1",

    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/

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”. Ustaw nasłuch zdarzenia `click` na tym przycisku. Po kliknięciu wywołaj funkcję `getData`.

Funkcja getData(event) pobiera z serwera objekt JSON (URL = https://jsonplaceholder.typicode.com/users/1).

Dane – id, nazwę użytkownika, adres url – dodaj do strony.

Napisz kod za pomocą czystego JavaScript i użyj do funkcji `fetch()`.

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 = https://jsonplaceholder.typicode.com/users/1).

Dane – id, nazwę użytkownika, adres url – dodaj do strony.

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 funkcji `fetch()`.

Akademia 108

https://akademia108.pl