

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

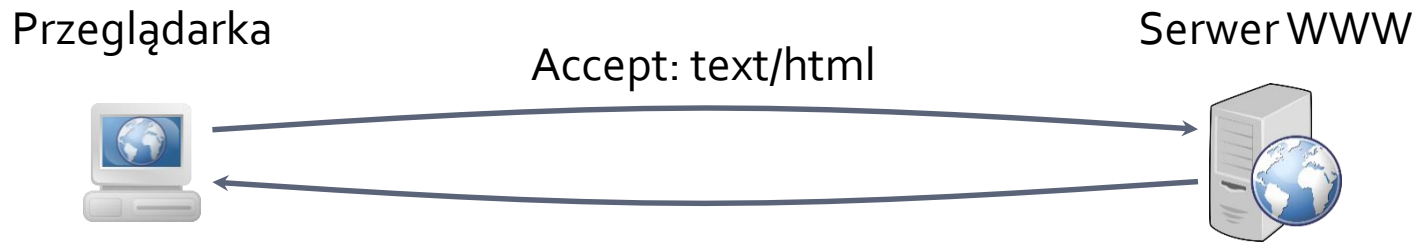
SPA and AJAX

Spis treści

- Wprowadzenie
- SPA
- Zalety i wady
- XMLHttpRequest
- AJAX w praktyce
- AJAX + jQuery
- SOP, CORS i JSONP
- Literatura

Wprowadzenie

- Jak działa klasyczna aplikacja WWW?

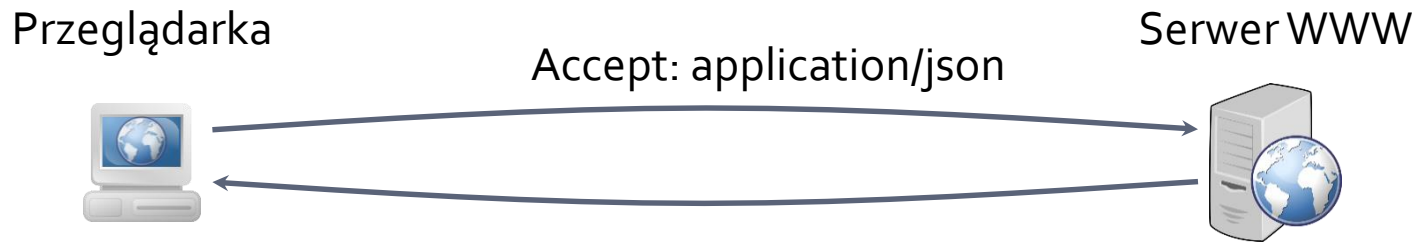


1. Prezentacja statycznego HTML-a
2. Elementy dynamiczne w celu poprawienia atrakcyjności wizualnej

1. Cała logika aplikacji
2. Tworzona jest strona HTML
3. Stan aplikacji na serwerze

SPA

■ SPA, czyli Single Page Application



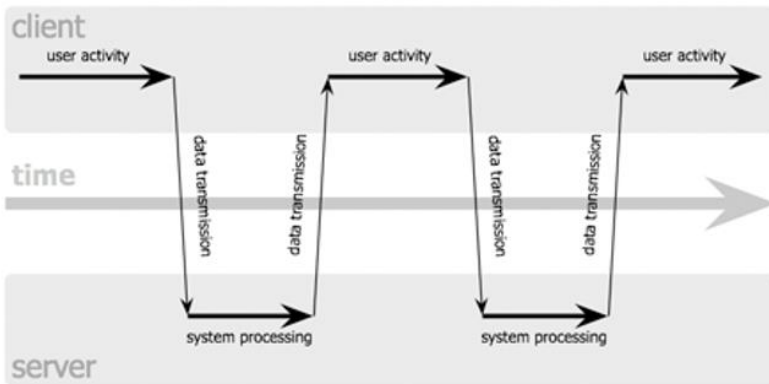
1. Logika aplikacji
2. Stan aplikacji na kliencie
3. Po pobraniu kodu aplikacji, pobierane z serwera są tylko dane

1. Hostowany jest kod aplikacji
2. Udostępniane są dane
3. Nie jest tworzony HTML

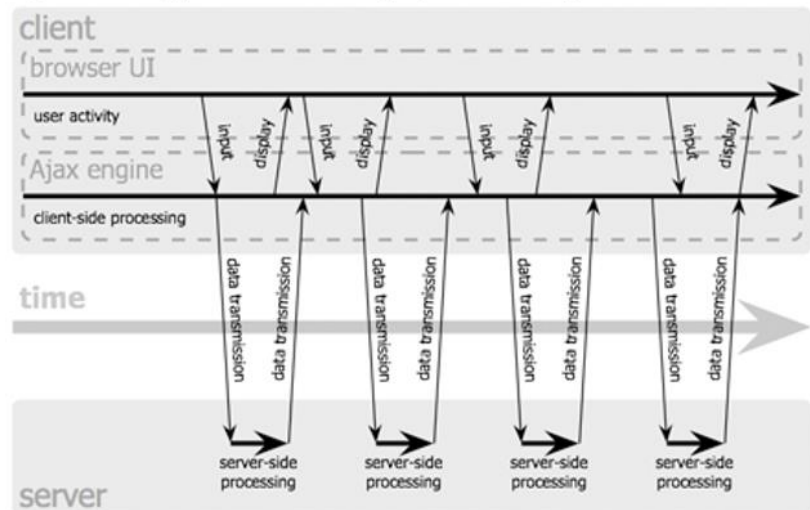
SPA vs. server-side application

- Cykl życia aplikacji

classic web application model (synchronous)



Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

SPA vs. server-side application



AJAX

- Rozwiązanie pozwalające budować SPA
- Pierwszy raz pojawia się w artykule Jesse James Garreta w roku 2005
<http://adaptivepath.org/ideas/ajax-new-approach-web-applications/>
 - ... chociaż technologie są od 1998 r.
- Z czego się składa? Cytując autora artykułu:
 - standards-based presentation using XHTML and CSS;
 - dynamic display and interaction using the Document Object Model;
 - data interchange and manipulation using XML and XSLT;
 - asynchronous data retrieval using XMLHttpRequest;
 - and JavaScript binding everything together.
- Wyróżnijmy jeden element:
 - Nagłówek *X-Requested-With: XMLHttpRequest*
 - Ciekawe rozważania:
<https://stackoverflow.com/questions/17478731/whats-the-point-of-the-x-requested-with-header>

Zalety

- Aplikacje są bardziej interaktywne
 - Szybsza aktualizacja treści na stronie
 - Mniejsze zużycie pasma
- Daje możliwość tworzenia bardziej rozbudowanych interfejsów użytkownika
- Wykorzystuje istniejące technologie

Wady

- Domyślnie aplikacja widoczna pod tylko jednym adresem:
 - Nie działa przycisk wstecz, który wg raportu Jacoba Nielsena jest pod drugą pod względem użyteczności funkcją nawigacyjną
 - Stan aplikacji jest reprezentowany przez adres URL
 - przez co nie można go zapisać np. do zakładek
 - uniemożliwia to reklamę „pantoflową” jak też zwykłe przesyłanie linków znajomym
 - Można to obejść, ale wymaga dodatkowej pracy
- Silniki wyszukiwarek mogą mieć problemy z poprawnym indeksowaniem stron
- Trudniej debugować i testować

XMLHttpRequest

- Metody obiektu XMLHttpRequest
 - abort() – przerywa żądanie
 - getResponseHeader(klucz) – pobiera wartość pola nagłówka http
 - open(metoda, uri, [async, [nazwa_użytkownika, [hasło]]) – określa parametry żądania:
 - metoda – GET lub POST
 - uri – adres żądania
 - async – czy asynchronicznie (domyślnie true)
 - użytkownik, hasło – możemy podać, jeśli dostęp do zasobu wymaga uwierzytelnienia
 - send(treść_żądania) – wysyła żądanie http
 - jeśli metoda jest POST, wprowadzamy treść żądania, jeśli GET, treść żądania powinna być null
 - jeśli async jest true, po wykonaniu send() sterowanie natychmiast wraca do skryptu (nie czeka na odpowiedź)
 - setRequestHeader(klucz, wartość) – dodaje pole nagłówka do żądania HTTP, metoda powinna być wywołana po open() i przed send()

XMLHttpRequest

- Właściwości obiektu XMLHttpRequest
 - onreadystatechange – funkcja wywoływana w momencie pojawienia się zdarzenia ReadyStateChange
 - co oznacza zmianę właściwości readyState
 - readyState – stan bieżącego żądania (tylko do odczytu)
 - status – kod stanu żądania HTTP (tylko do odczytu)
 - responseText
 - tylko do odczytu
 - tekst odpowiedzi dostępny po udanym obsłużeniu odpowiedzi, przy czym readyState musi co najmniej 3 (pełny tekst dostępny przy 4)
 - responseXML
 - tylko do odczytu
 - odpowiedź w postaci XML-a
 - dostępny przy stanie readyState=4

XMLHttpRequest

- Kody odpowiedzi
readyState może przyjmować wartości:
 - 0 – niezainicjalizowane
 - 1 – ładuje
 - 2 – załadowane
 - 3 – interaktywne (odpowieź przekazana częściowo)
 - 4 – zakończone
- UWAGA
 - fakt przejścia readyState do 4 nie oznacza poprawnego obsłużenia żądania a jedynie zakończenie obsługi żądania
 - trzeba jeszcze sprawdzić status czy jest 200

Przykłady

- Przykłady dla podejścia natywnego:
 - time.html, time.php
 - mathexpressions.html, mathexpressions.php
 - mathexpressions.html2, mathexpressions2.php
 - replacehtml.html,
replacehtml1.php, replacehtml2.php

AJAX w praktyce

- Jak korzystać z AJAX-a
 - natywne podejście bardzo kłopotliwe
 - biblioteki:
Prototype , jQuery. Ext, Script.aculo.us.
MooTools, Yahoo! UI Library. Dojo Toolkit,
AJAX.OOP. picoAjax
 - frameworki:
Echo , Google Web Toolkit (Java), ASP.NET (.NET),
Symfony, Xajax (PHP), Pyjamas (Python)

AJAX + jQuery

- Bardzo prosto i przejrzystość
- Wybrane metody
 - `jQuery.ajax()` – podstawowa metoda
 - `.load()` – wczytanie dokumentu
 - `jQuery.get()` – j.w., dane dołączone metodą GET
 - `jQuery.post()` – j.w. dane dołączone metodą POST
 - `.ajaxStart()` – podpięcie zdarzenia wysłania żądania
 - `.ajaxComplete()` – podpięcie zdarzenia zakończenia obsługi żądania

Przykłady

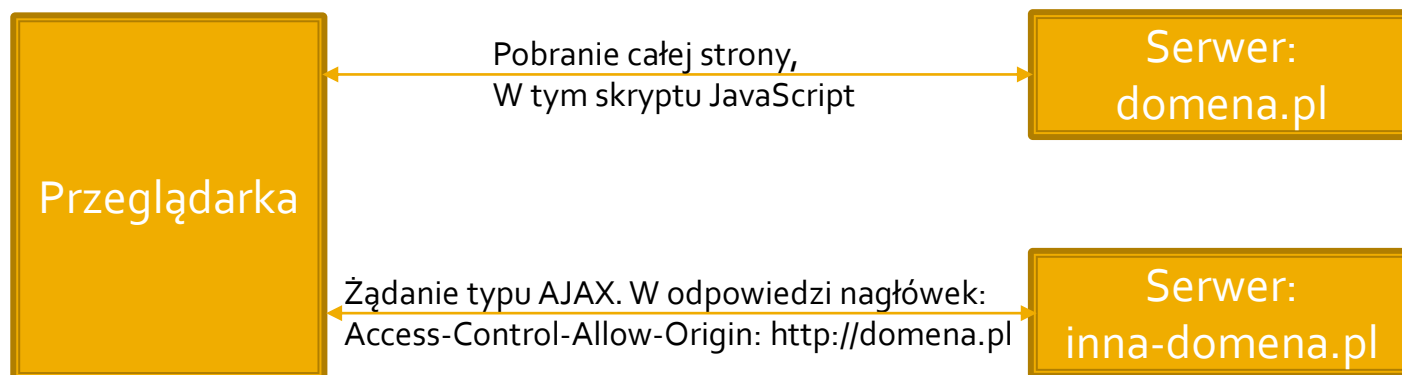
- jquery1.html
- jquery2.html

Same-Origin Policy

- SOP = Same-Origin Policy
- Na danej stronie można wykonywać żądania AJAX tylko w obrębie tej samej „origin”
- Origin, czyli
 - Protokół, host i numer portu
- Łamiąc tę regułę otrzymujemy
 - XMLHttpRequest cannot load http://external.service/. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://my.app' is therefore not allowed access.
- Dwa typowe sposoby rozwiązania „problemu”:
 - CORS i JSONP

SOP & CORS

- CORS = Cross-Origin Resource Sharing



- Dobre wyjaśnienie:

- <https://zinoui.com/blog/cross-domain-ajax-request>
- <http://enable-cors.org/>

SOP & JSONP

- JSONP = JSON with Padding
- Dozwolona jest konstrukcja
 - `<script type="text/javascript" src="http://www.example.com/path/getdata"></script>`
- A zatem zwracamy, przykładowo
 - `functionCall({"imie":"Jan", "Nazwisko":"K."})`
- Klient obsługuje `functionCall` pobierając oczekiwane dane
- Dobre wyjaśnienie
 - <http://www.ajax-cross-origin.com/how.html>
 - <https://jvaneyck.wordpress.com/2014/01/07/cross-domain-requests-in-javascript/>

Przykłady

- CORS
- JSONP

Literatura

- AJAX w wikipedii
 - <http://pl.wikipedia.org/wiki/AJAX>
- AJAX w W3Schools
 - http://www.w3schools.com/xml/ajax_intro.asp
- Krótkie wprowadzenie AJAX + jQuery
 - http://www.w3schools.com/jquery/jquery_ajax_intro.asp
- Przegląd metod do AJAX-a w jQuery
 - http://www.w3schools.com/jquery/jquery_ref_ajax.asp
- Dokumentacja jQuery + AJAX
 - <http://api.jquery.com/category/ajax/>