KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Rozproszone technologie WebServices

Prezentacja jest współfinansowana przez Unię Europejską w ramach Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

Politechnika Łódzka

Politechnika Łódzka, ul. Żeromskiego 116, 90-924 Łódź, tel. (042) 631 28 83
www.kapitalludzki.p.lodz.pl

SAX, DOM i AJAX

Bartłomiej Świercz

Katedra Mikroelektroniki i Technik Informatycznych

Łódź, 11 kwietnia 2010

Wstęp

DOM – Document Object Model zapewnia:

- Zbiór obiektów reprezentujących dokumenty XML i HTML.

Wstęp

DOM – Document Object Model zapewnia:

- **Zbiór obiektów** reprezentujących dokumenty XML i HTML.
- Model łączenia obiektów.

Wstęp

DOM – Document Object Model zapewnia:

- **Zbiór obiektów** reprezentujących dokumenty XML i HTML.
- **Model łączenia** obiektów.
- Standardowy interfejs umożliwiający dostęp i manipulację obiektami.

Wstęp

DOM – Document Object Model zapewnia:

- **Zbiór obiektów** reprezentujących dokumenty XML i HTML.
- **Model łączenia** obiektów.
- Standardowy **interfejs** umożliwiający dostęp i manipulację obiektami.

Początkowo firmy implementowały własny model dostępu do obiektów HTML w przeglądarkach. Wyjściem było zdefiniowanie standardowego modelu, który został nazwany **W3CDOM**. DOM jako API jest niezależne od platformy i języka programowania.

Poziomy DOM

Istnieje kilka poziomów modelu DOM:

- DOM Level 0 (nieoficjalny),
- DOM Level 1,
- DOM Level 2,
- DOM Level 3,

Poziomy DOM

Istnieje kilka poziomów modelu DOM:

DOM Level 0 (nieoficjalny), Model DOM z przeglądarki Netscape 3.0, skopiowany przez Microsfot i zaimplementowany we wszystkich przeglądarkach internetowych, umożliwiając dostęp tylko do pól formularzy. Nie jest standardem W3C.

DOM Level 1,

DOM Level 2,

DOM Level 3,

Poziomy DOM

Istnieje kilka poziomów modelu DOM:

DOM Level 0 (nieoficjalny),

DOM Level 1, Poziom 2 umożliwia dostęp do treści dokumentu poprzez tworzenie, modyfikowanie oraz dołączanie elementów i atrybutów.

DOM Level 2,

DOM Level 3,

Poziomy DOM

Istnieje kilka poziomów modelu DOM:

DOM Level 0 (nieoficjalny),

DOM Level 1,

DOM Level 2, Poziom dodatkowo wprowadza możliwość obsługi zdarzeń i przestrzeni nazw.

DOM Level 3,

Poziomy DOM

Istnieje kilka poziomów modelu DOM:

DOM Level 0 (nieoficjalny),

DOM Level 1,

DOM Level 2,

DOM Level 3, Na poziom trzeci składają się elementy:

- DOM Level 3 Core
- DOM Level 3 Load and Save
- DOM Level 3 XPath
- DOM Level 3 Views and Formatting
- DOM Level 3 Requirements
- DOM Level 3 Validation

Wymagania implementacyjne DOM

Poniżej opisane są ogólne wymagania odnośnie DOM stawiane przez W3C:

- Model obiektu jest językowo obojętny.
- Jądro DOM powinno być w stanie przetwarzać dokumenty XML, HTML i CSS.
- Model obiektu może być użyty do wczytywania i zapisywania dokumentu.
- Konwencja nazewnictwa musi być jednakowa na wszystkich poziomach DOM.
- Model obiektu nie powinien narażać aplikacji użytkownika na błędy związane z bezpieczeństwem, walidacją i prywatnością.

Reakcja na błędy

Standard W3CDOM określa sposób w jaki błędy mają być przekazywane przez model obiektu:

- Informacje o błędach są przekazywane za pomocą wartości zwracanej.
- Wyjątki są generowane jedynie wtedy, kiedy wystąpi błąd (warunek) nieodwracalny.
- DOM powinien dostarczać opis do błędów.
- DOM można zapytać o stan błędu.

DOM a języki programowania

Dla większości języków istnieją biblioteki dostarczające obsługę modelu DOM dla dokumentów typu XML, jednak standard W3C definiuje API interfejsu DOM jedynie dla języków Java i JavaScript. Najbardziej zaawansowaną biblioteką jest Apache Xerces i MS XML.

Na wykładzie przedstawiona zostanie biblioteka DOM (org.w3c.dom) dla języka Java.

Otwieranie pliku i parsowanie dokumentu XML

```
File file = new File( "wypozyczalnia.xml" );
Document doc = null;
try {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    doc = db.parse( file );
} catch ( java.io.IOException e ) {
    System.out.println( "Can't open the file" );
} catch ( Exception e ) {
    System.out.print( "Can't parse the file." );
}
```

Utworzenie uchwytu do elementu głównego

```
Element root = doc.getDocumentElement();
System.out.println( "The root element: "
                + root.getNodeName() + "\n" );
```

Przetwarzanie elementów pośrednich

```
NodeList children = root.getChildNodes();
System.out.print( "There are "+children.getLength()+" child elements:\n" );

for( Node child = root.getFirstChild(); child != null;
     child = child.getNextSibling() )
{
    System.out.println( "Element: " + child.getNodeName() );
}
```

Wady i zalety DOM

Wada: dokument musi być w całości załadowany do pamięci.

Zaleta: swobodny dostęp do elementów dokumentu, możliwość jego modyfikacji i zapisu.

Wstęp

SAX – Simple API for XML

SAX jest szeregowym interfejsem opartym o zdarzenia. Dostarcza mechanizm odczytywania danych z dokumentów XML. SAX został zaprojektowany przez członków listy pocztowej xml-dev bez formalnego wsparcia instytucji takich jak W3C. Początkowym liderem projektu był David Megginson, który pracował nad implementacją SAX dla języka Java. Obecnie stroną domową projektu SAX jest:

`http://www.saxproject.org`

SAX - odczytywanie dokumentu XML

SAX traktuje dokument XML jako strumień danych, który jest odczytywany sukcesywnie w czasie parsowania. Oznacza to, że nie da się przetworzyć ponownie wcześniej doczytanego elementu, bez wznowienia odczytywania całego dokumentu. API SAX oparte jest o mechanizm zdarzeń, które są generowane podczas parsowania elementu. Zdarzenie wywołuje zarejestrowaną funkcję zwrotną (ang. callback).

Funkcje zwrotne - przypomnienie

Język C

```
int add (int x, int y)
{
        return x + y;
}


void sum (int a, int b, int (*my_add) (int, int))
{
        . . . .
        my_add (a, b);
        . . . .
}
```

SAX kontra DOM

API oparte o drzewa (DOM) Algorytm mapowania dokumentu XML na drzewo elementów przechowywane w całości w pamięci.

API oparte o zdarzenia (SAX) Algorytm parsowania dokumentów XML generujący zdarzenia po napotkaniu elementów dokumentu. Aplikacja musi dostarczyć funkcje obsługi zdarzeń podobnie jak aplikacja GUI dostarcza funkcje obsługi zdarzeń elementów graficznego interfejsu użytkownika.

Wady i zalety obydwu rozwiązań . . .

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Rozpoczynamy przetwarzanie ...

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 1

Nazwa zdarzenia: Początek dokumentu

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 2

Nazwa zdarzenia: Początek elementu: “dvd”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 3

Nazwa zdarzenia: Początek elementu: “tytul”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 4

Nazwa zdarzenia: Tekst: “Rambo I”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 5

Nazwa zdarzenia: Koniec elementu: “tytul”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 6

Nazwa zdarzenia: Początek elementu: “aktor”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 7

Nazwa zdarzenia: Tekst: “Sylvester Stallone”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 8

Nazwa zdarzenia: Koniec elementu: “aktor”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 9

Nazwa zdarzenia: Koniec elementu: “dvd”

Przetwarzanie dokumentów oparte o zdarzenia

Plik XML:

```
<?xml version=‘‘1.0’’?>
<dvd>
        <tytul>Rambo I</tytul>
        <aktor>Sylvester Stallone</aktor>
</dvd>
```

Zdarzenie generowane podczas przetwarzania:

Zdarzenie numer: 10

Nazwa zdarzenia: Koniec dokumentu

Interfejs SAX

Interfejs SAX składa się z czterech głównych obiektów:

ContentHandler Metody tego obiektu wywoływane są podczas generowania zdarzeń. Jest to główny obiekt (interfejs) modułu SAX.

DTDHandler Wywoływany jest do przetwarzania zdarzeń związanych z obsługą DTD.

EntityResolver Obiekt odpowiada za obsługę zewnętrznych jednostek dokumentu.

ErrorHandler Obiekt służy do generowania informacji o błędach w parsowaniu dokumentu XML.

Nowy świat aplikacji internetowych . . .

System/przeglądarka Dzięki rozwojowi standardów internetowych coraz łatwiej jest pisać aplikacje których działanie nie zależy od systemu operacyjnego i użytej przeglądarki internetowej.

SOA – Service-Oriented Architecture Najtrafniej opisał to Albert Einstein (czyżby przewidział istnienie serwisów?):

„Things should be made as simple as possible, but no simpler."

A poczciwy unix'owiec potwierdził:

„Keep it simple, stupid!"

SPA – Single Page Application Technologia która zapewnia nam dynamiczny interfejs użytkownika.

Single Page Application

Aplikacja typu Single Page Application jest programem internetowym uruchomionym wewnątrz przeglądarki WWW. SPA jest kombinacją HTML (XHTML), JavaScript i styli CSS. Aplikacja w odróżnieniu od tradycyjnej strony WWW działa w obrębie jednej fizycznej strony wygenerowanej przez przeglądarkę. SPA modyfikuje stronę poprzez interfejs DOM.

Obecnie połączenie technologi SPA z zewnętrznymi serwisami nazywa się AJAX.

AJAX

AJAX – Asynchronous JavaScript and XML

Jest to technika tworzenia interaktywnych aplikacji WWW używając kombinacji technologii:

- HTML (XHTML) i CSS – warstwa prezentacji.
- Interfejs DOM i język JavaScript – to jest człon „JavaScript and XML”.
- Obiekt XMLHttpRequest – sprawca wystąpienia słowa „Asynchronous”.

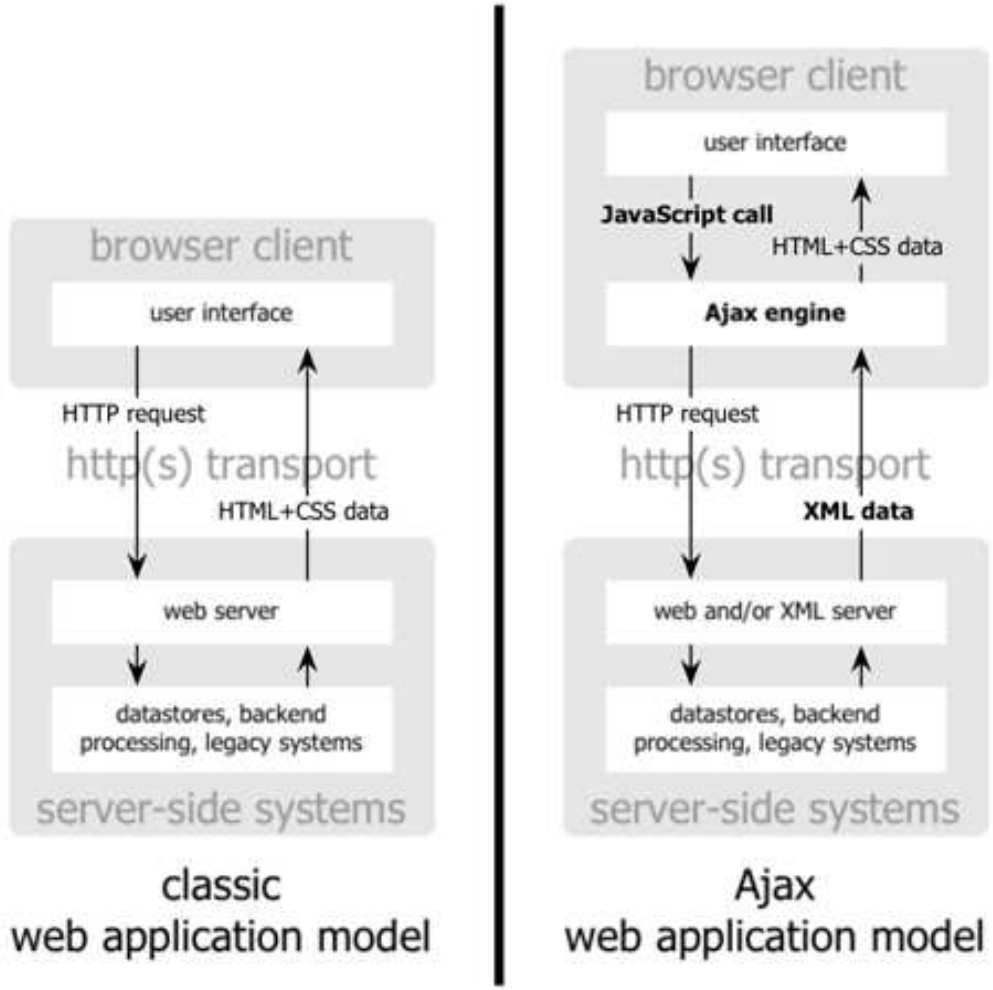
Aplikacja AJAX sprawia wrażenie, że działa w całości na maszynie użytkownika ponieważ dane są pobierane częściowo i w razie potrzeby, a nie jak to miało miało miejsce w przypadku tradycyjnych aplikacji WWW, gdzie były odświeżane całe strony.

AJAX . . .

Jest wiele kontrowersji wokół nazwy, lecz trafnie całe zamieszanie podsumował Paul Graham:

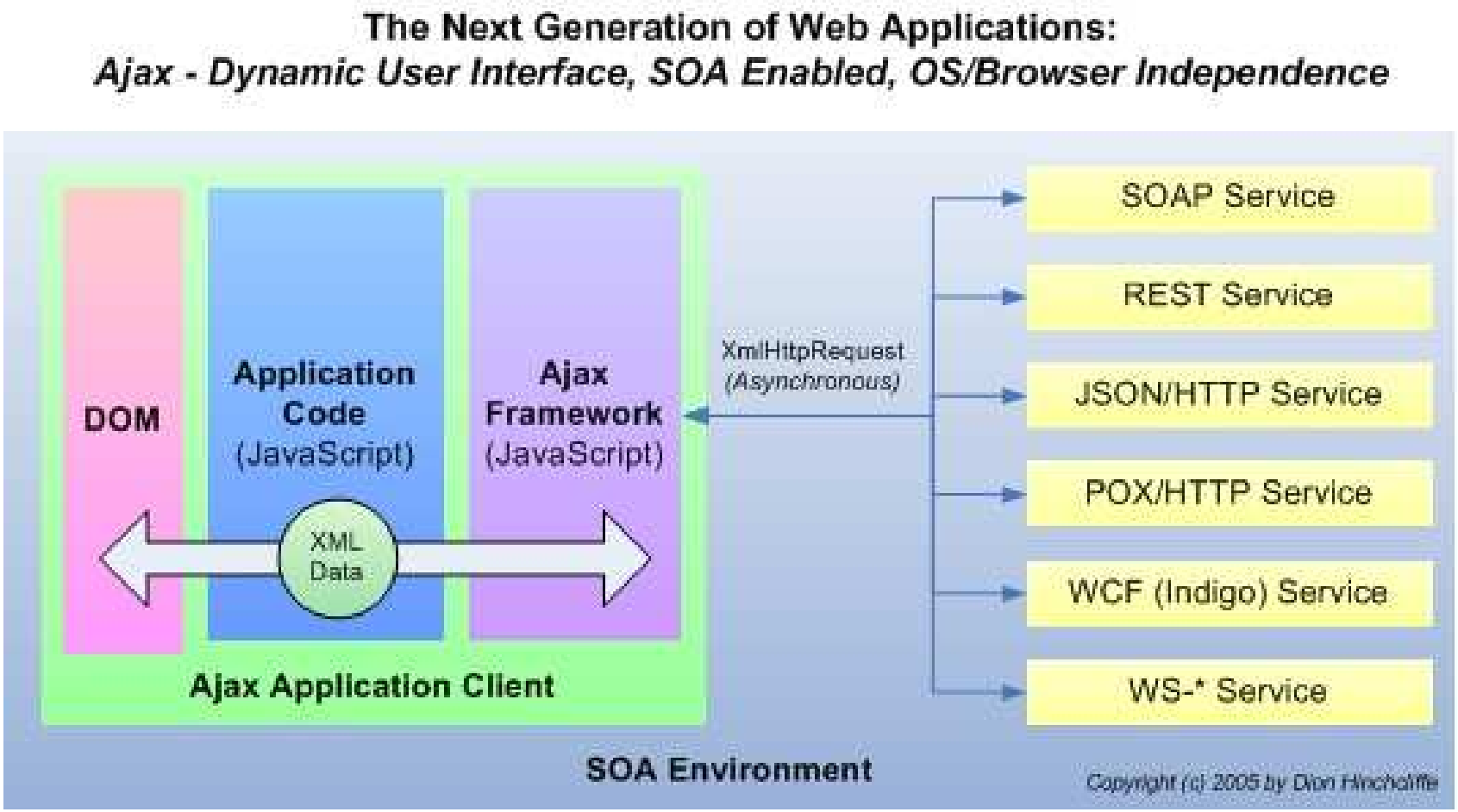
> *Basically, what "Ajax" means is "Javascript now works." And that in turn means that web-based applications can now be made to work much more like desktop ones.*

AJAX a tradycyjny model WWW



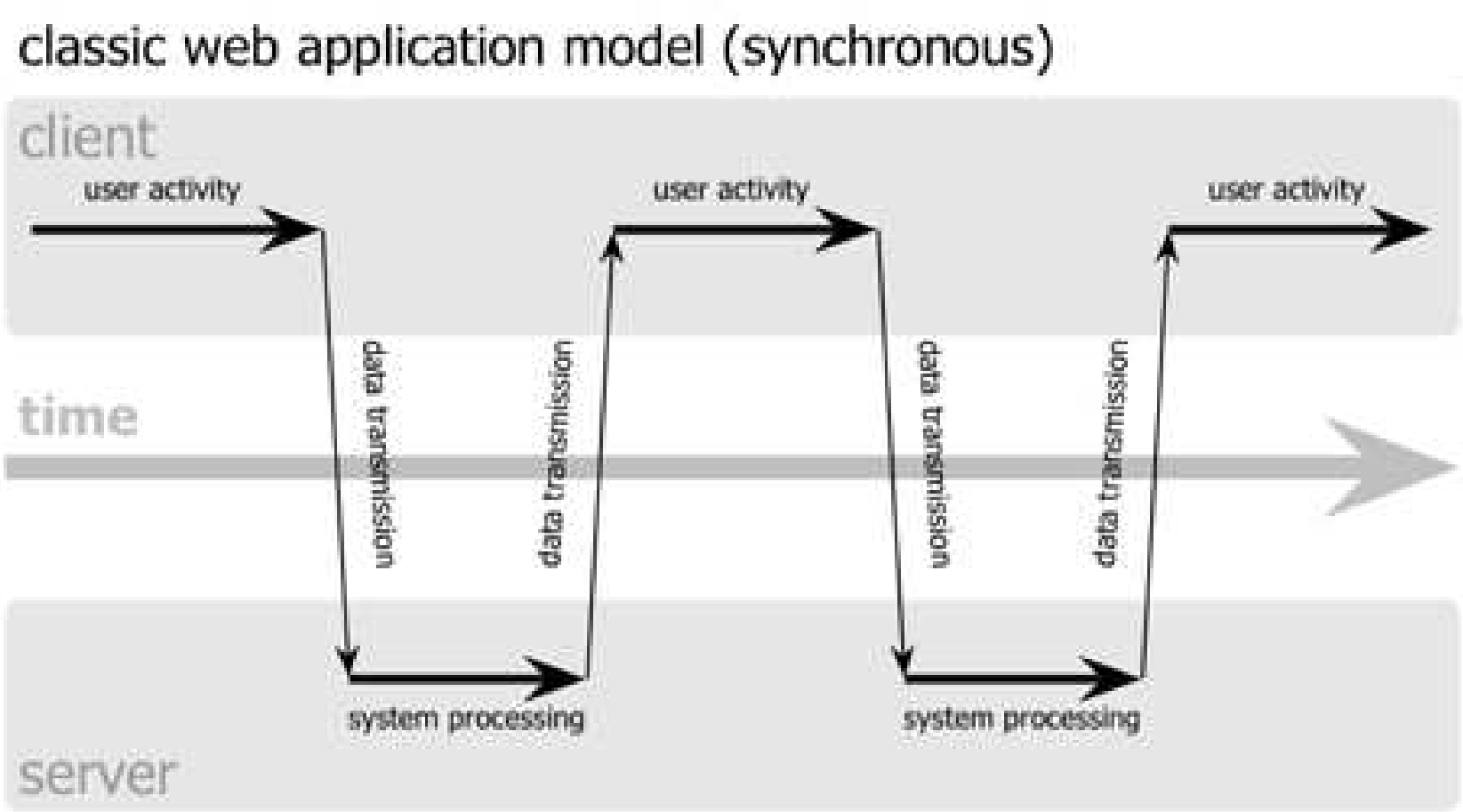
Rysunek: Jesse James Garrett „Ajax: A New Approach to Web Applications"

AJAX a model SOA



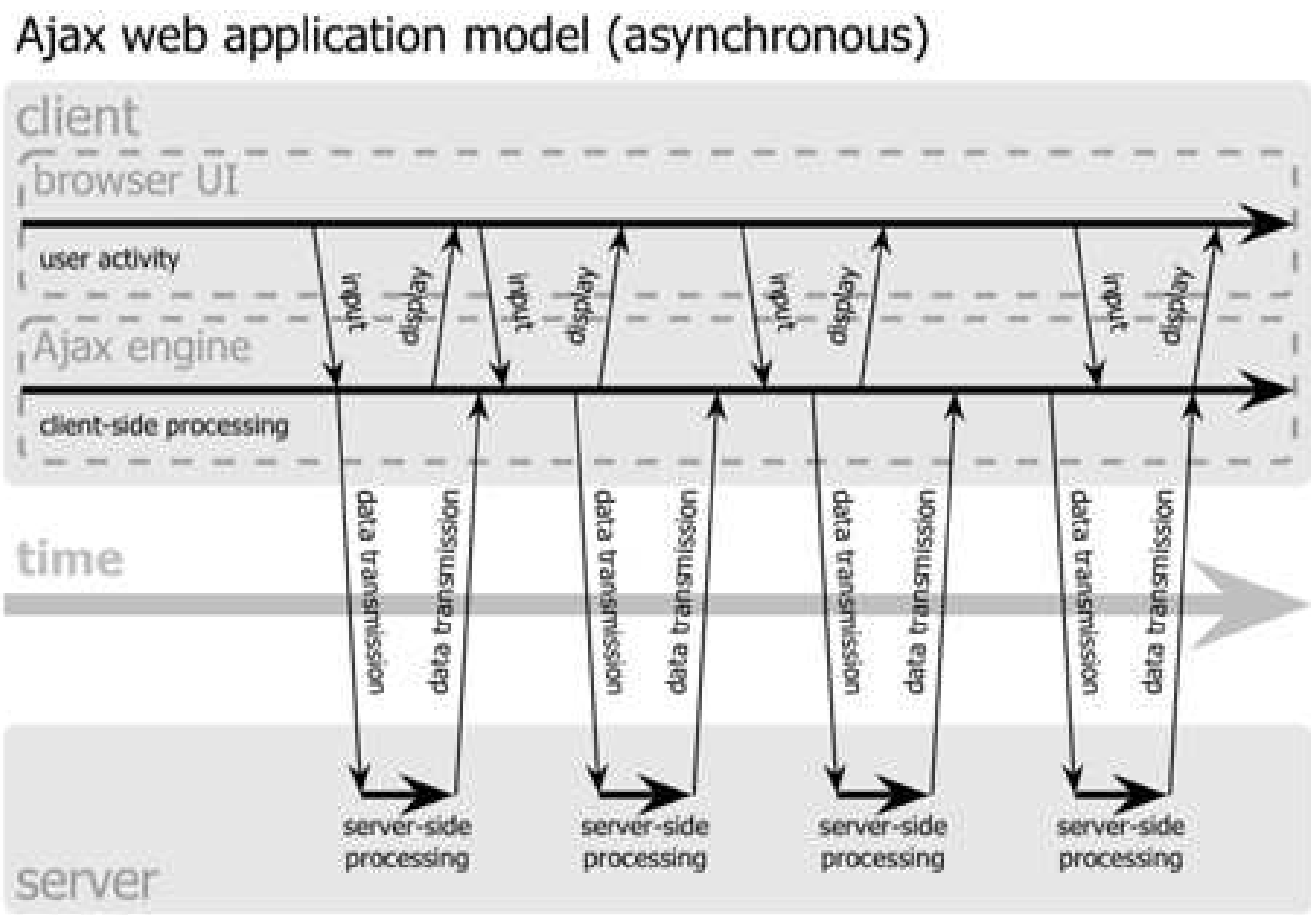
Rysunek: Dion Hinchcliffe's Blog - Musings and Ruminations on Building Great Systems

Tradycyjna synchroniczna komunikacja



Rysunek: Artykuł Jesse James Garrett „Ajax: A New Approach to Web Applications”

AJAX i asynchroniczna komunikacja



Rysunek: Artykuł Jesse James Garrett „Ajax: A New Approach to Web Applications”

Obiekt XMLHttpRequest

Firma Microsoft jako pierwsza zaimplementowała obiekt XMLHttpRequest w przeglądarce Internet Explorer 5 jako obiekt ActivX. Następnie programiści projektu Mozilla zaimplementowali natywną wersję XMLHttpRequest (zgodną programowo z obiektem ActivX Microsoftu) w przeglądarce Mozilla 1.0. Obiekt XMLHttpRequest zaimplementowany został również w przeglądarkach Opera i Safari.

Podobna funkcjonalność została zapewniona przez propozycję organizacji W3C: *Document Object Model (DOM) Level 3 Load and Save Specification*. Jednak wraz ze wzrostem liczby przeglądarek wspierających XMLHttpRequest, obiekt ten stał się de facto standardem w komunikacji asynchronicznej.

Tworzenie obiektu

```
if (window.XMLHttpRequest)
{
  http = new XMLHttpRequest();
}
else if (window.ActiveXObject)
{
  http = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Metody obiektu

abort(): Zatrzymuje aktualne rządanie.

getAllResponseHeaders(): Zwraca kompletny nagłówek (zbiór etykiet i wartości) jako string.

getResponseHeader(*"etykieta"*): Zwraca wartość pojedyńczego nagłówka (jako string), którego etykieta została podana jako parametr.

open(*"metoda"*, *"URL"*, *asyn*): Określenie docelowego adresu URL i metody planowanego rządania.

send(*dokument*): Wysłanie rządania.

setRequestHeader(*"etykieta"*, *"wartość"*): Określenie elementów nagłówka, który zostanie wysłany razem z żądaniem.

Własności obiektu

onreadystatechange Uchwyt dla funkcji wywoływanej przy każdej zmianie stanu.

readyState Stan obiektu (integer):

- 0 = uninitialized
- 1 = loading
- 2 = loaded
- 3 = interactive
- 4 = complete

responseText Pole przechowuje dane pobrane z serwera.

responseXML Dane kompatybilne z DOM zwrócone z serwera.

status Numeryczny kod zwrócony przez serwer (np. 404, 200).

statusText Opis kodu zwróconego przez serwer.

Obsługa zdarzeń obiektu XMLHttpRequest

```
http.onreadystatechange = function()
{
  if (http.readyState == 4)
  {
    // wszystko w porządku, możemy przetwarzać dane
  }
  else
  {
    // dane ciągle nie są gotowe
  }
};
```

Obsługa zdarzeń obiektu XMLHttpRequest

```
if (http.status == 200)
{
  // super!
}
else
{
  // problem z obsługą rządania
  // np. bład 404
};
```

Wysłanie zapytania

```
http.open ('GET', 'http://code.org/file.php', true);
http.send(null);
```

Praca z danymi XML

Przy pomocy obiektu XMLHttpRequest możemy pobrać dane w postaci XML i operować na nich za pomocą interfejsu DOM:

```
http.overrideMimeType ('text/xml');
...
// wykonanie rządania danych
...
var xmldoc = http.responseXML;
var dvd = xmldoc.getElementsByTagName('dvd').item(0);
```

Pięta Achillesowa AJAX

1. JavaScript – musi być włączona obsługa skryptów.
2. Niepełna implementacja standardów W3C przez producentów przeglądarek internetowych.
3. Nieoczekiwane zachowanie typowych elementów przeglądarki takich jak np. przycisk cofnij.

Przykłady aplikacji AJAX

GMail `http://gmail.com`

Google Groups `http://groups.google.com/`

Google Maps `http://maps.google.com/`

Własna strona Google `http://www.google.com/ig`

FCKeditor `http://www.fckeditor.net/demo/`

JS/UIX Terminal `http://www.masswerk.at/jsuix/`

Cheetah `http://cheetah.gnu.org.ua/`

Coś więcej?

AHAh 

Coś więcej?

AHA 

AHAH: Asychronous HTML and HTTP

AHAH jest prostą techniką pobierania danych za pomocą JavaScript. Technika ta polega na wykorzystaniu obiektu XMLHttpRequest do pobierania całych stron XHTML lub ich fragmentów i bezpośredniego wklejania ich na bieżącej stronie przeglądarki.

I na dokładkę . . .

AFLAX – Asynchronous Flash and XML

Technologia polegająca na połączeniu AJAX i Flash w celu uzyskania bardziej dynamicznych aplikacji WWW.

Koniec teorii!

Napiszmy coś wykorzystując technologie AJAX!

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Rozproszone technologie WebServices

Prezentacja jest współfinansowana przez Unię Europejską w ramach Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

Politechnika Łódzka

Politechnika Łódzka, ul. Żeromskiego 116, 90-924 Łódź, tel. (042) 631 28 83
www.kapitalludzki.p.lodz.pl