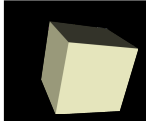
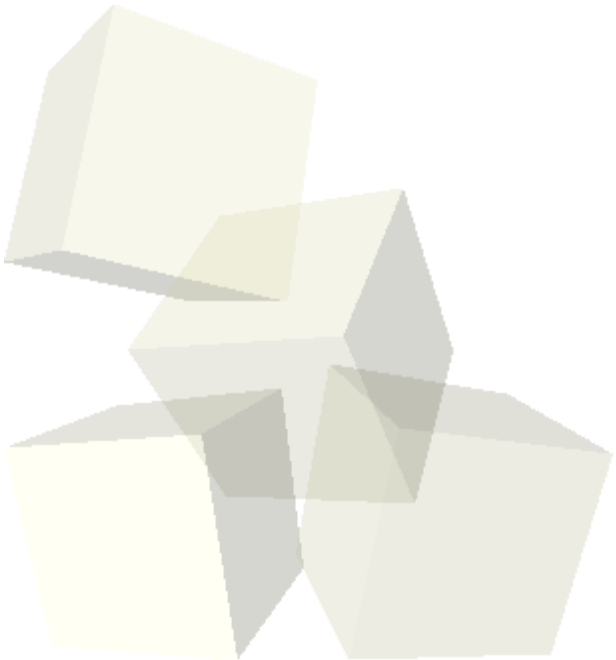


AJAX

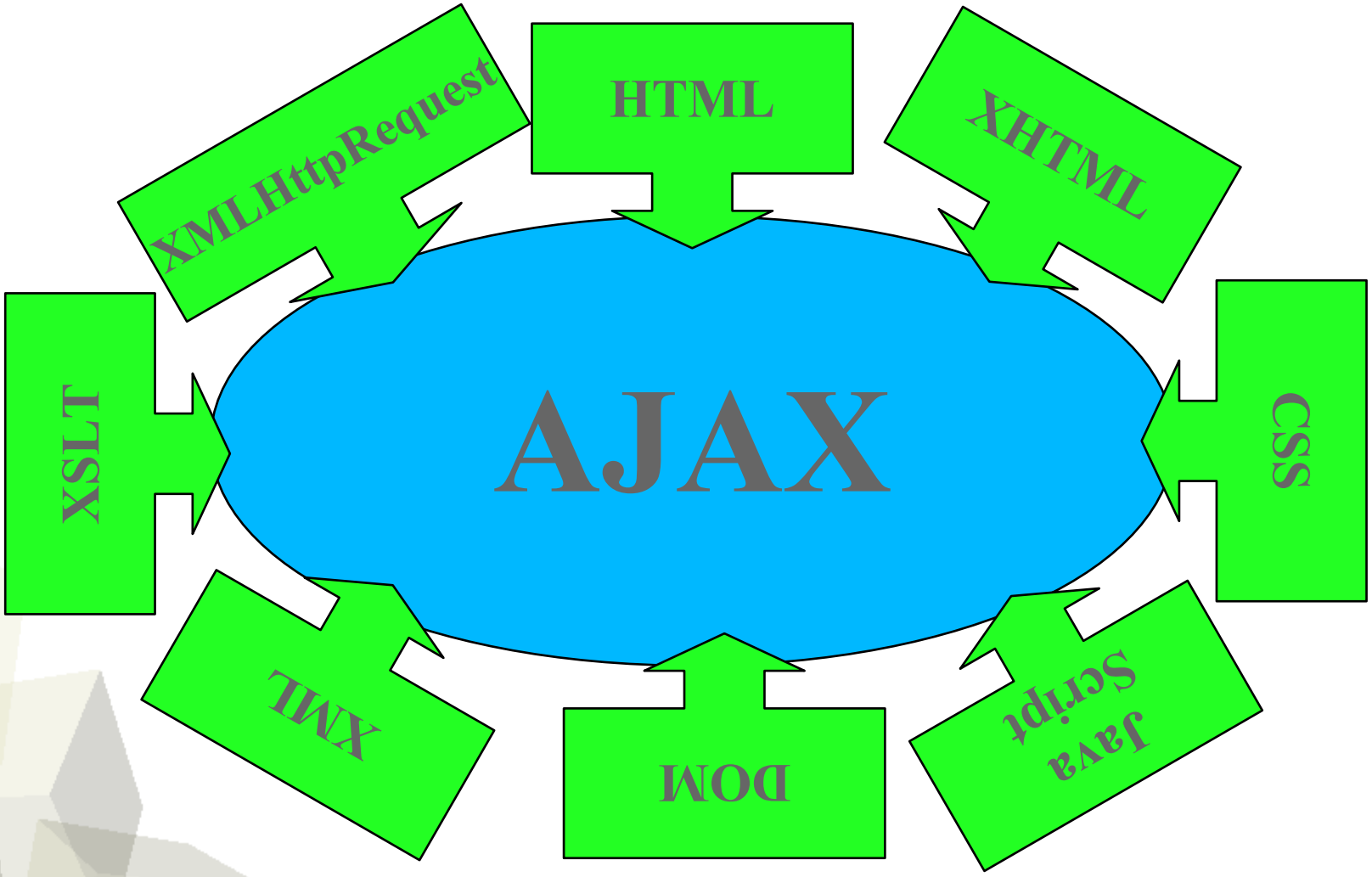


WSTĘP

AJAX (Asynchronous JavaScript and XML, Asynchroniczny JavaScript i XML) – technika tworzenia aplikacji internetowych, w której interakcja użytkownika z serwerem odbywa się bez przeładowywania całego dokumentu.

AJAX sam w sobie nie jest nową technologią; jest zbiorem istniejących technologii, zastosowanych w taki sposób, który daje nową jakość.

Technologie tworzące AJAX



Aplikacje desktopowe a webowe (bez AJAX)

Aplikacja desktopowa

- **Program odpowiada intuicyjnie i szybko** - np. walidacja może być preprowadzana na bieżąco
- **Użytkownik dostaje istotne odpowiedzi zwrotne**
  - Komórka zmienia kolor, gdy ją modyfikujemy
  - Ikony się podświetlają gdy najedziemy na nie myszką
- **Wszystko dzieją się w sposób naturalny**
  - Nie trzeba klikać na żaden przycisk, czy też link, żeby uruchmić jakiś wyzwalacz (trigger), np. automatyczne formatowanie danych w komórce po zakończeniu edycji

Aplikacja webowa

- **Sekwencja interkacji *"Kliknij, poczekaj, odśwież"***
  - Każde wysłanie danych do serwera wymusza odświeżenie całej strony
- **Synchroniczny model komunikacji *"żądanie/odpowiedź"***
  - Użytkownik musi czekać na odpowiedź zanim wykona kolejną operację
- **Nawigacja oparta o strony**
  - Logika nawigowania stronami jest określona przez serwera

AJAX = DHTML + zdolność asynchronicznej komunikacji uzyskiwanej dzięki XMLHttpRequest

- Zalety
  - Wiele potencjalnych możliwości zastosowania - dynamiczny rozwój *cloud computing* oraz *Saas* (software as a service) wróży niemal świetlaną przyszłość tej technologii
  - Realna staje się technologia RIA (Rich Internet Application- oferująca bogaty, dynamiczny, jednoekranowy interfejs)
  - Pojawiają się coraz lepsze frameworki i narzędzia
  - Nie wymaga instalacji żadnych plugin-ów ani dodatkowego kodu

- Wady
  - Nie działa w starszych przeglądarkach
  - JavaScript jest raczej trudny w konserwacji i debugowaniu

Gdzie można użyć AJAX?

- Na bieżąco przeprowadzana walidacja danych po stronie serwera
  - Identyfikatory, kody pocztowe, adresy e-mail itd.
  - Usuwa dwukrotną walidację danych: po stronie klienta
- Automatyczne dopełnianie (auto-completion)
- Wyszukiwarki – prezentacja dotychczas wyszukanych obiektów z automatyczna aktualizacją; użytkownik nie musi czekać na wynik końcowy
- Dynamiczne pobieranie treści – automatyczne pobieranie maili w webowych klientach pocztowych, aktualizacja statusu na stronie (np. Twitter, GaduGadu)

Gdzie można użyć AJAX?

- Galeria zdjęć – prezentacja miniaturek; zdjęcia w pełnym wymiarze prezentowane po wybraniu danej miniaturki przez użytkownika
- Wyświetlanie szczegółów obiektów - podpowiedzi (tootips), wyświetlenie szerszego opisu
- Relacje (np. sportowe) na żywo.
- Budowa zaawansowanych interfejsów użytkownika – struktury drzewiaste z możliwością rozwijania/zwijania, menu, paski postepu
- Podpowiedzi w systemach wyszukiwania
- Czat

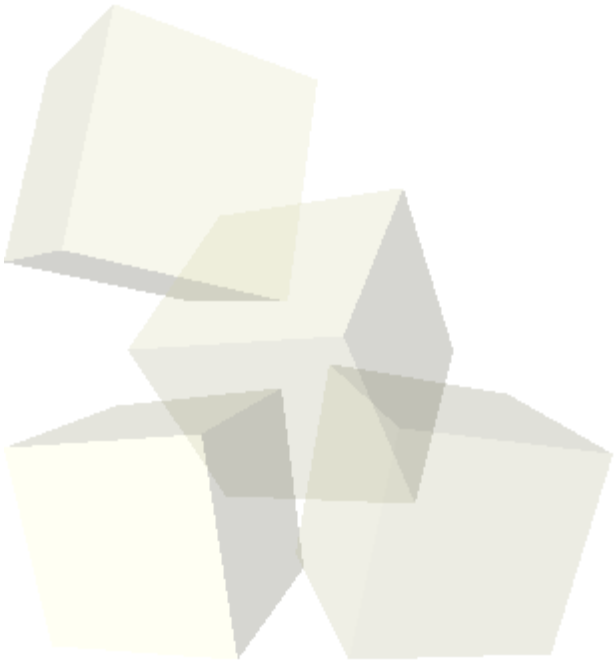
Przykłady aplikacji z AJAX

Google Maps

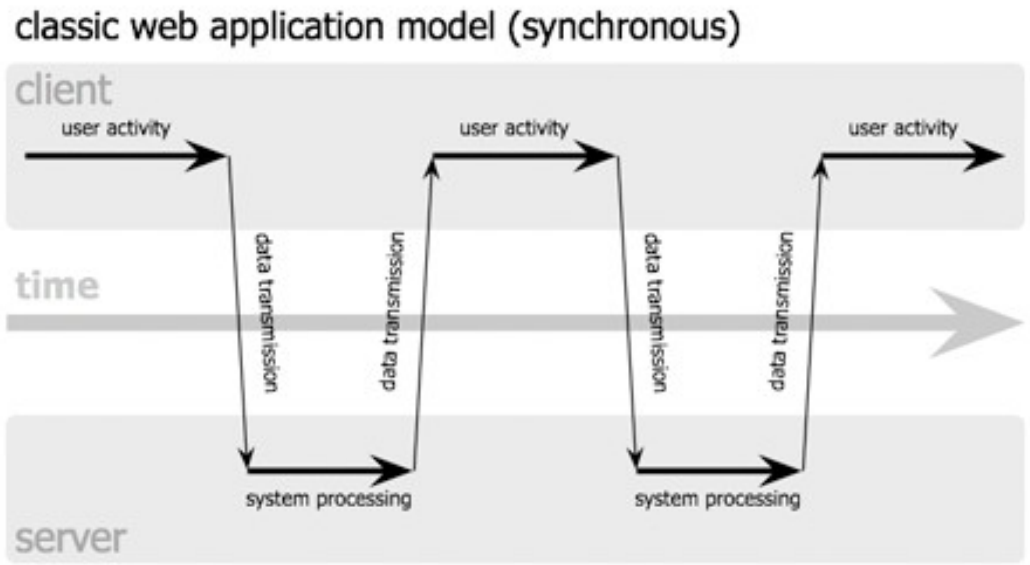
- Użytkownik może przeciągać (przemieszczać) całą mapę za pomocą myszki
  - Nie musi klikać na żadne przyciski lub inne komponenty; akcja, która wymusza ściąganie odpowiednich fragmentów mapy jest wykonywana na podstawie ruchów myszą
- Ściąganie fragmentów mapy odbywa się asynchronicznie w tle
- Pozostałe elementy strony nie są zmieniane; kontekst jest zachowywany

Przeglądarki z XMLHttpRequest

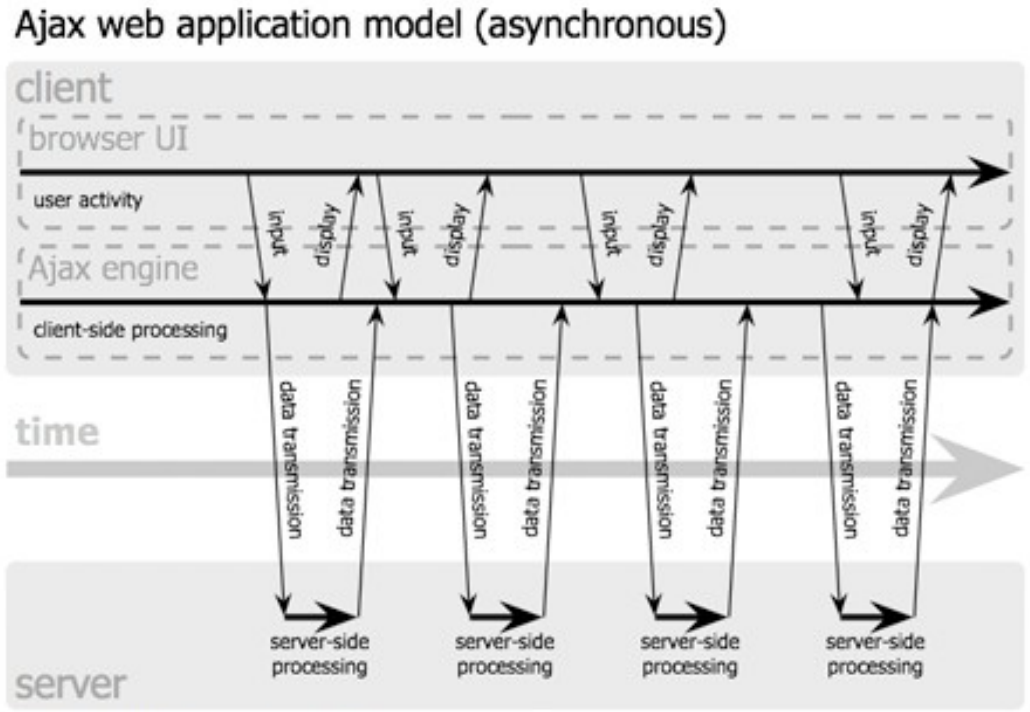
- Mozilla Firefox >= 1.0
- Netscape version >=7.1
- Apple Safari >= 1.2
- Microsoft Internet Explorer >= 5
- Konqueror
- Opera >= 7.6



Model klasyczny a AJAX

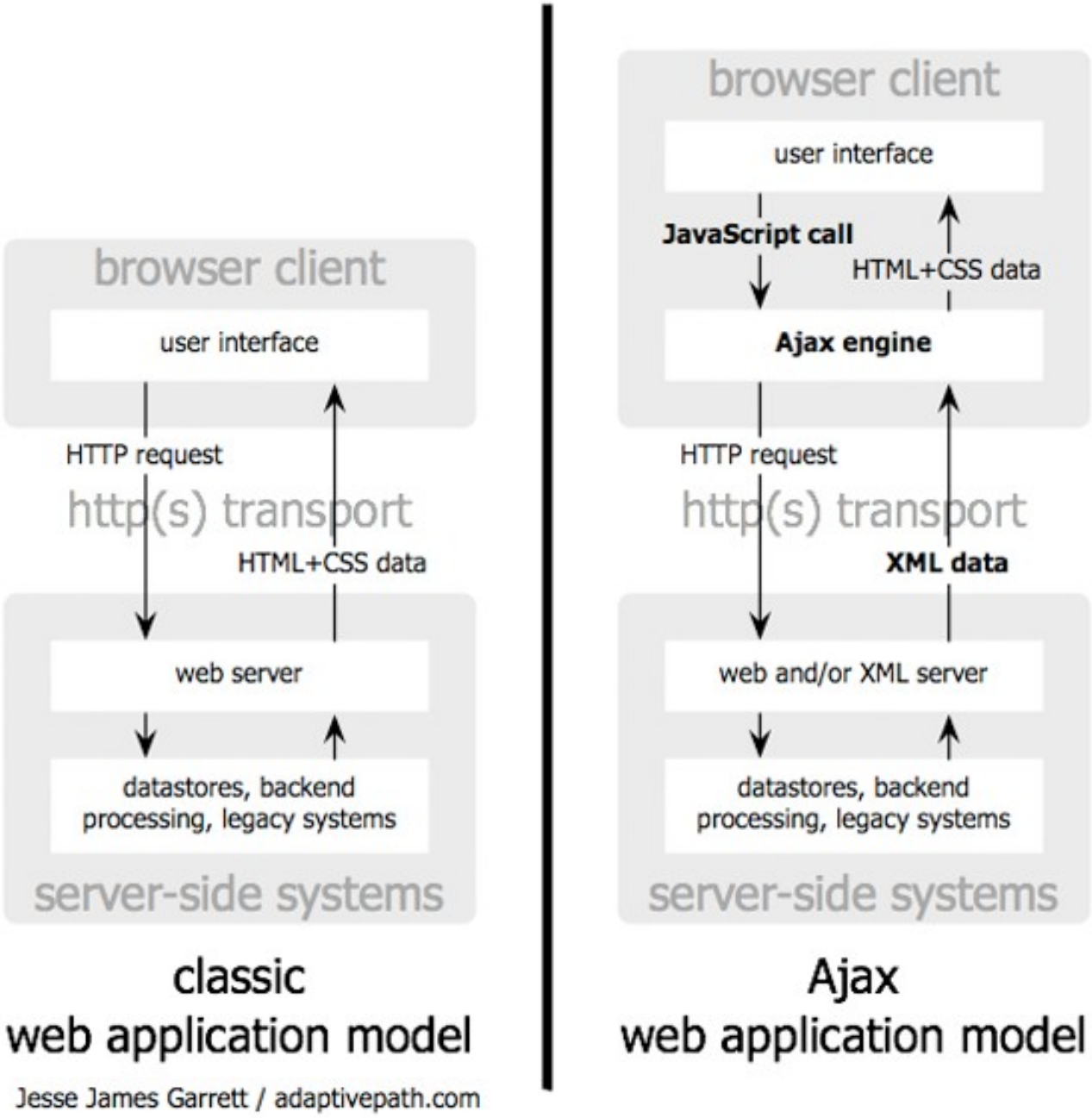


Operacja użytkownika jest przerywana do czasu, aż dane zostan pobrane



Operacje użytkownika nie są przerywane podczas pobierania danych

Model klasyczny a AJAX



XMLHttpRequest

- Obiekt JavaScript
- Zaadoptowany przez najpopularniejsze przeglądarki internetowe
  - Mozilla, Firefox, Safari, Opera, IE
- Komunikacja z serwerem poprzez standardowy protokół HTTP (GET/POST)
- Działa w tle zapewniając asynchroniczną komunikację z serwerem, nie przerywając przy tym interakcji z użytkownikiem.

Tworzenie obiektu XMLHttpRequest

- Najnowsze przeglądarki

```
variable=new XMLHttpRequest();
```

- IE5 oraz IE6

```
variable=new ActiveXObject("Microsoft.XMLHTTP");
```

```
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

Wysyłanie żądania do serwera

- **open(method, url, async)** – definicja żądania
  - method: GET lub POST
  - url: ścieżka do pliku na serwerze
  - async: true (asynchornicznie) lub false (synchronicznie). Nie rekomenduje się używania “false”, chyba, że do bardzo szybkich żądań.

- **send(string)** – wysłanie żądania
  - string: parametry (tylko w przypadku POST)

  W przypadku POST, nagłówek definiowany za pomocą funkcji: setRequestHeader(header,value)

- Wynik zwracany jest jako zwykły tekst albo dokument XML:
  - XMLHttpRequest.responseText
  - XMLHttpRequest.responseXML

XMLHttpRequest – przykłady wywołań

- Bez parametrów

```
xmlhttp.open("GET","demo_get.asp",true);
xmlhttp.send();
```

```
xmlhttp.open("POST","demo_post.asp",true);
xmlhttp.send();
```

- Z parametrami

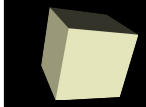
```
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();
```

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-
urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

Zdarzenie onreadystatechange

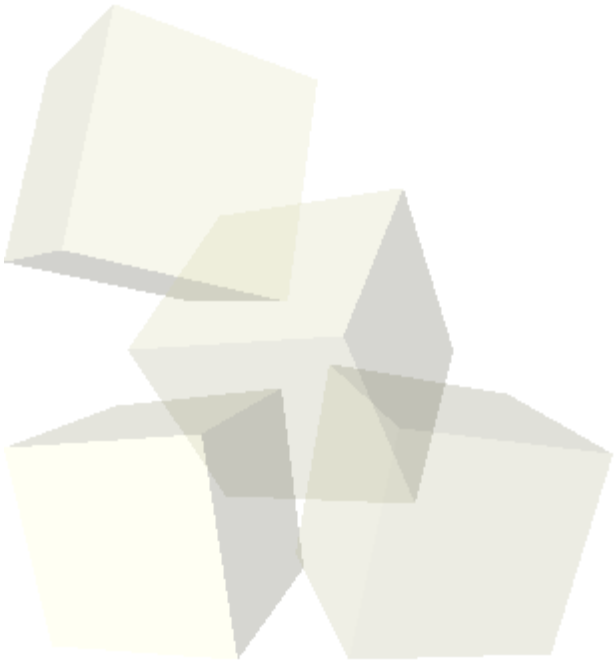
- Gdy wysyłamy żądanie do serwera, w oparciu o uzyskaną odpowiedź chcemy wykonać pewne operacje.
- Zdarzenie *onreadystatechange* jest wykonywane za każdym razem, gdy zmieni się *readyState*.
- Własność *readyState* przechowuje status XMLHttpRequest.
  - 0: żądanie nie zainicjalizowane
  - 1: nawiązano połączenie z serwerem
  - 2: żądanie wysłane
  - 3: przetwarzanie żądania
  - 4: żądanie zakończone a odpowiedź jest gotowa
- *Status*
  - *200: "OK"*
  - *404: Page not found*

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
```

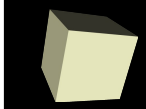


responseXML

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
  {
  txt=txt + x[i].childNodes[0].nodeValue + "<br />";
  }
document.getElementById("myDiv").innerHTML=txt;
```



```
<html>  <head>
<script type="text/javascript">
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head> <body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</body> </html>
```



KONIEC