AJAX

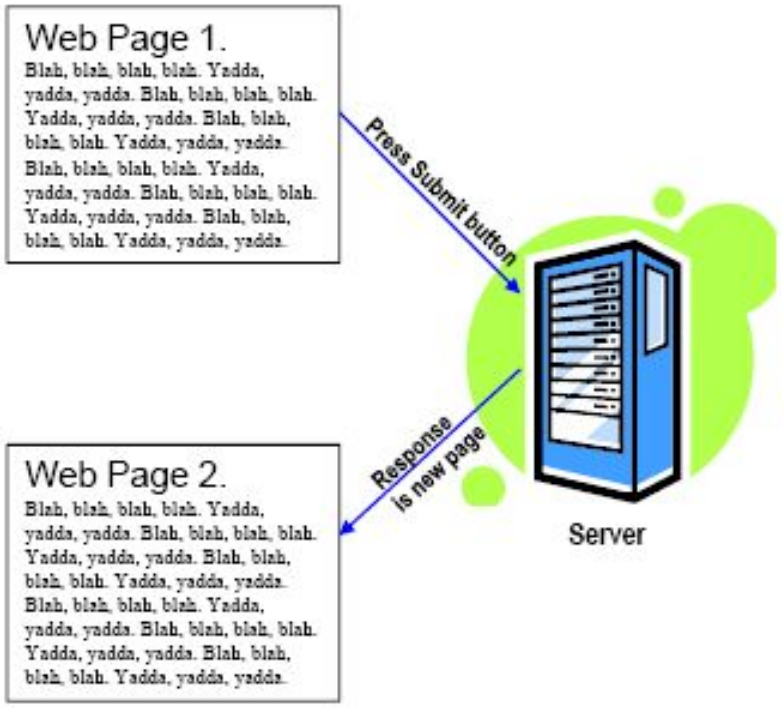
Asynchronous Javascript and XML

Для чего нужен AJAX

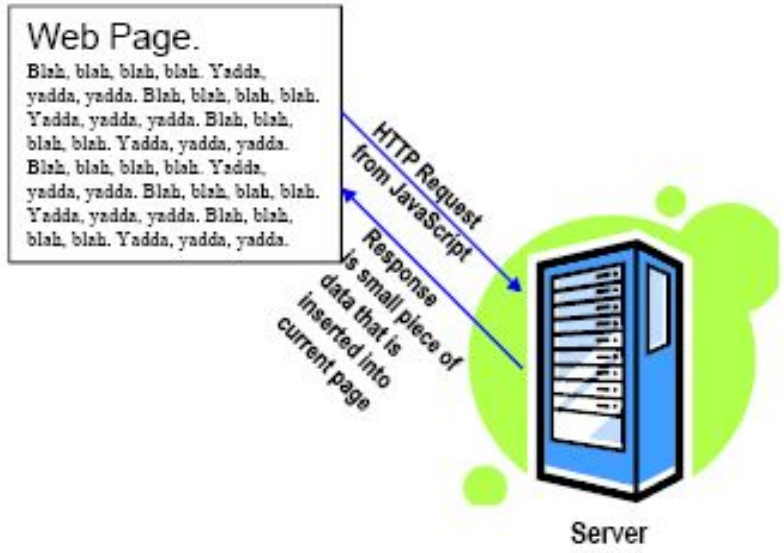
- Недостаточная функциональность HTTP и HTML
  - Не интерактивен
  - Нет частичных обновлений
- Альтернативы
  - Java Applets
    - Нет универсальной поддержки
    - Нет взаимодействия с HTML
  - стек технологий Flash в виде ActionScript 3, Adobe Flex и Flash Remoting составляет технологическую основу RIA (Rich Internet Applications) активно продвигаемыхMacromedia (теперь часть Adobe)
  - Новые и еще не имеющие широкой поддержки
    - Microsoft Silverlite
    - JavaFX
    - Adobe AIR

Традиционные веб-приложения и AJAX

- **Traditional Web Apps: Infrequent Large Updates**



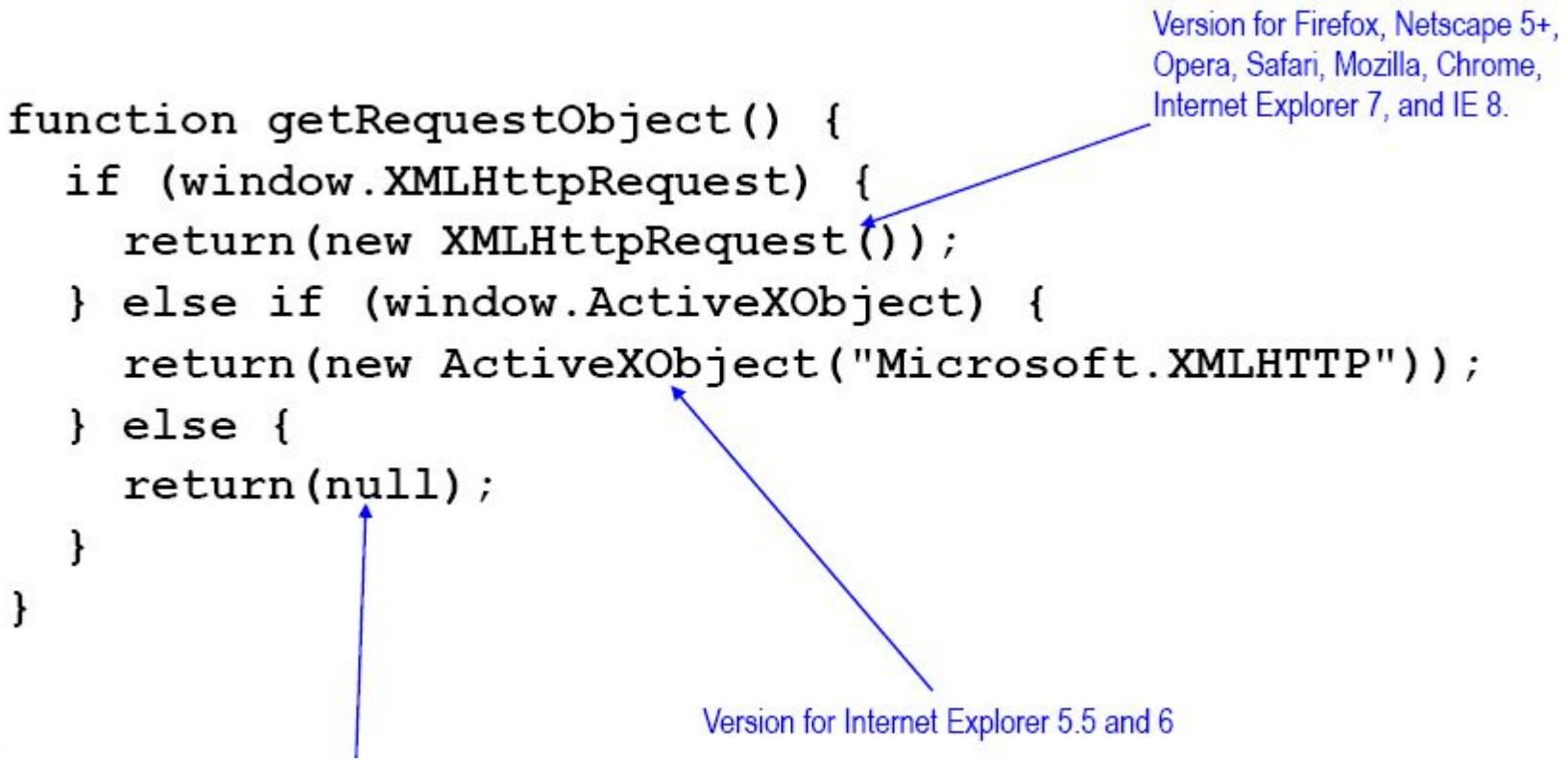
- **Ajax Apps: Frequent Small Updates**



Основной процесс AJAX

- JavaScript
  - Определение объекта для генерации HTTP-запросов
  - Инициирование запроса
    - Получить объекта запроса
    - Определить анонимный обработчик ответа
      - И использовать его в качестве атрибута onreadystatechange запроса
    - Инициировать GET или POST запрос
    - Отправить данные
  - Обработка ответа
    - Обработать readyState = 4 и HTTP status = 200
    - Извлечь и обработать текст ответа с помощью responseText или responseXML
    - Что-то сделать с результатом
- HTML
  - Загрузить JavaScript
  - Определить элемент управления для инициирования запроса
  - Определить ID для элементов ввода и вывода ответа

Определение объекта запроса



Инициирование запроса

```
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request) };
  request.open("GET", "message-data.html", true);
  request.send(null);
}
```

Code to call when server responds

URL of server-side resource. Must be on same server that page was loaded from.

POST data
(always null for GET requests)

Don't wait for response
(Send request asynchronously)

Обработка ответа

```
function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

4 means response from server is complete (handler gets invoked multiple times – ignore the first ones)

Text of server response

Особенности функций JavaScript

- На JavaScript можно передавать функции как аргументы
  - function doSomethingWithResponse() { code }
  - request.onreadystatechange = doSomethingWithResponse;
- Возможны анонимные функции
  - var request = getRequestObject();
  - request.onreadystatechange =
  - function() { code-that-uses-request-variable };
- В Java есть анонимные классы, но нет анонимных функций
- В C и C++ нет анонимных функций
- Анонимные функции (т.н. closures) широко используются в технологиях Lisp, Ruby, Scheme, C#, Python, Visual Basic, ML, PHP (as of 5.3), Clojure, Go и др.

Функции в качестве аргументов

```
function square(x) { return(x * x); }
function triple(x) { return(x * 3); }
function doOperation(f, x) { return(f(x)); }

doOperation(square, 5); → 25

doOperation(triple, 10); → 30

var functions = [square, triple];

functions[0](10);  → 100

functions[1](20);  → 60
```

Анонимные функции

```
function square(x) { return(x * x); }

square(10);  → 100
(function(x) { return(x * x); })(10);  → 100

function makeMultiplier(n) {
  return(function(x) { return(x * n); });
}

var factor = 5;
var f = makeMultiplier(factor);

f(3);  → 15
factor = 500;
f(3);  → 15
```

Некорректный подход (с использованием глобальной переменной Request)

```
var request;

function getRequestObject() { ... }

function sendRequest() {
  request = getRequestObject();
  request.onreadystatechange = handleResponse;
  request.open("GET", "...", true);
  request.send(null);
}

function handleResponse() {
  if (request.readyState == 4) {
    alert(request.responseText);
}
```

Некорректный подход (с использованием глобальной переменной Request)

- Сценарий
  - 2 кнопки, 2 обработчика: function1 и function2
  - function1 получает данные с сервера в среднем 5 с
  - function2 – 1 с
  - Пользователь нажимает кнопку 1 затем – кнопку 2 с интервалом около секунды
- Проблема
  - function1 при обращении к request.responseText получит текст ответа для function2
- Решение
  - Использование анонимной функции с локальной копией объекта запроса

Правильный подход

```
function getRequestObject() { ... }

function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "...", true);
  request.send(null);
}

function handleResponse(request) {
  ...
}
```

JavaScript код show-message.js

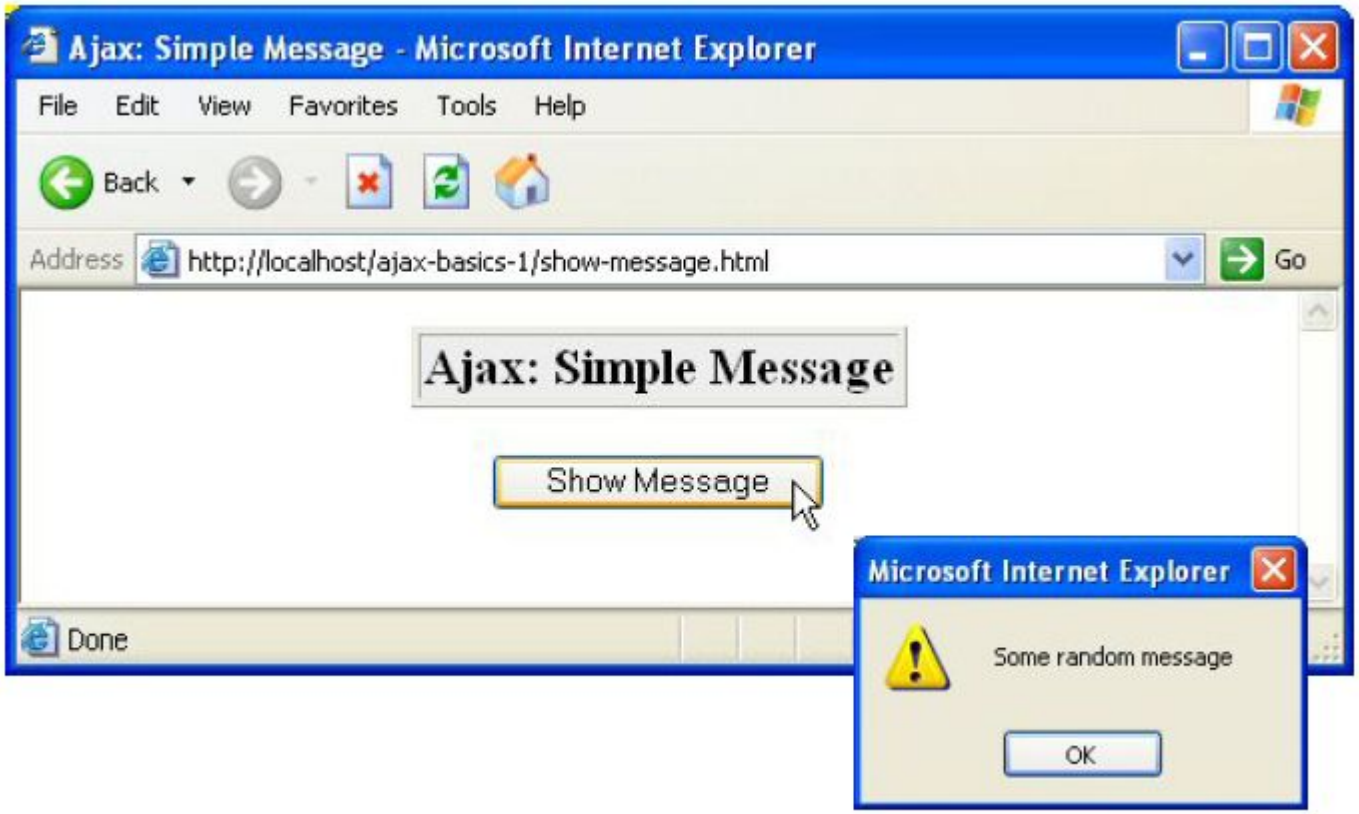
```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}

function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "message-data.html", true);
  request.send(null);
}

function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

HTML код show-message.html

```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Simple Message</title>
<script src="show-message.js"
        type="text/javascript"></script>
</head>
<body>
<center>
<table border="1" bgcolor="gray">
  <tr><th><big>Ajax: Simple Message</big></th></tr>
</table>
<p/>
<input type="button" value="Show Message"
       onclick="sendRequest()"/>
</center></body></html>
```

message-data.html

- Some random message
- Результат:



Взаимодействие сервлетом и динамическое изменение страницы

```
package coreservlets;
import ...

public class ShowTime extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    Date currentTime = new Date();
    String message =
      String.format("It is now %tr on %tD.",
                    currentTime, currentTime);
    out.print(message);
  }
}
```

Инициирование запроса

```
function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

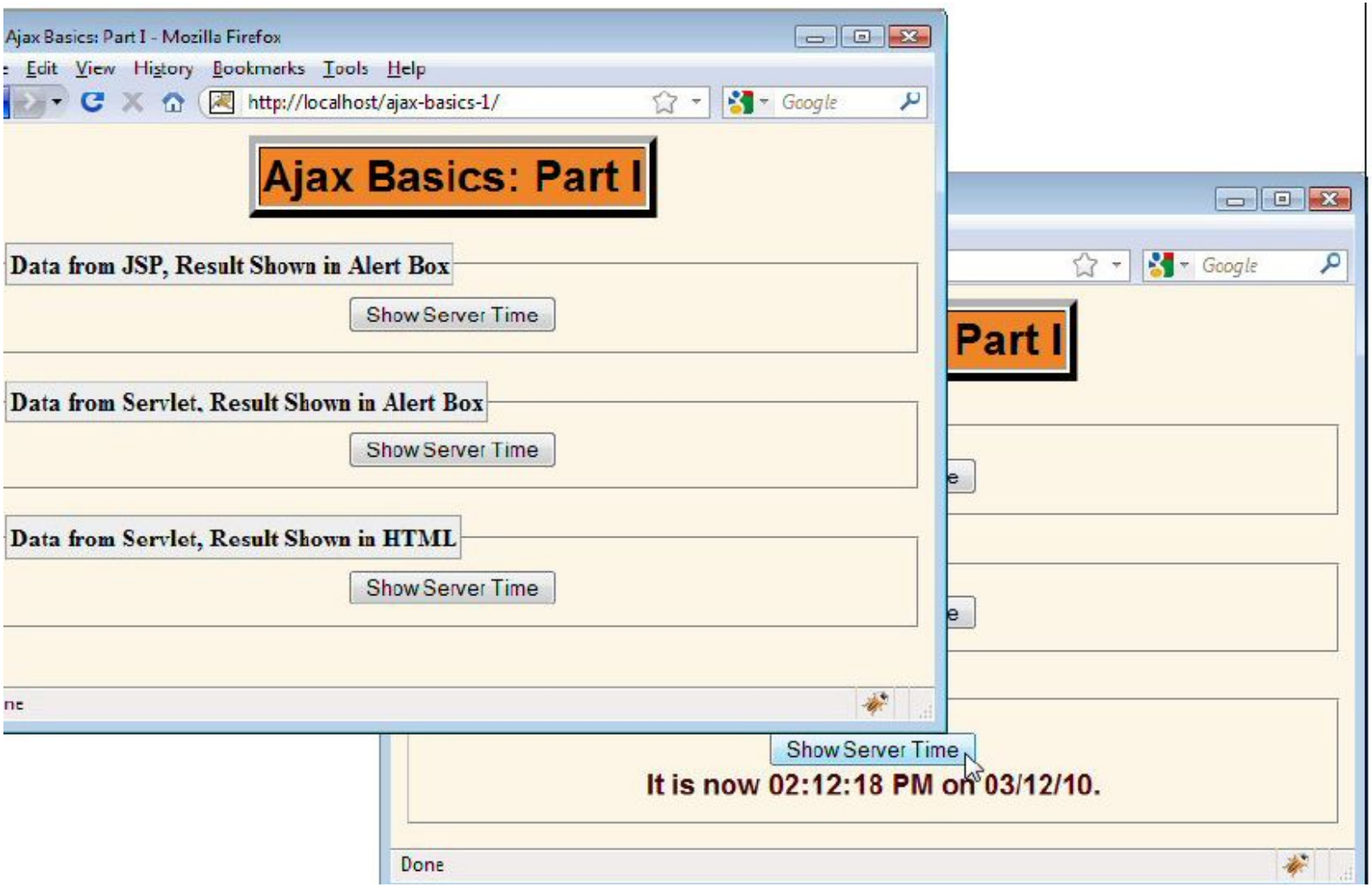
Обработка ответа

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    document.getElementById(resultRegion).innerHTML =
      request.responseText;
  }
}
```

HTML-код

```
...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-basics.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>Data from Servlet, Result Shown in HTML</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxResult("show-time","timeResult1")'/>
  <div id="timeResult1" class="ajaxResult"></div>
</fieldset>
...
```

Результат


Ajax Basics: Part I - Mozilla Firefox
Edit View History Bookmarks Tools Help
http://localhost/ajax-basics-1/
Google
Ajax Basics: Part I
Data from JSP, Result Shown in Alert Box
Show Server Time
Data from Servlet, Result Shown in Alert Box
Show Server Time
Data from Servlet, Result Shown in HTML
Show Server Time
Part I
Show Server Time
It is now 02:12:18 PM on 03/12/10.
Done