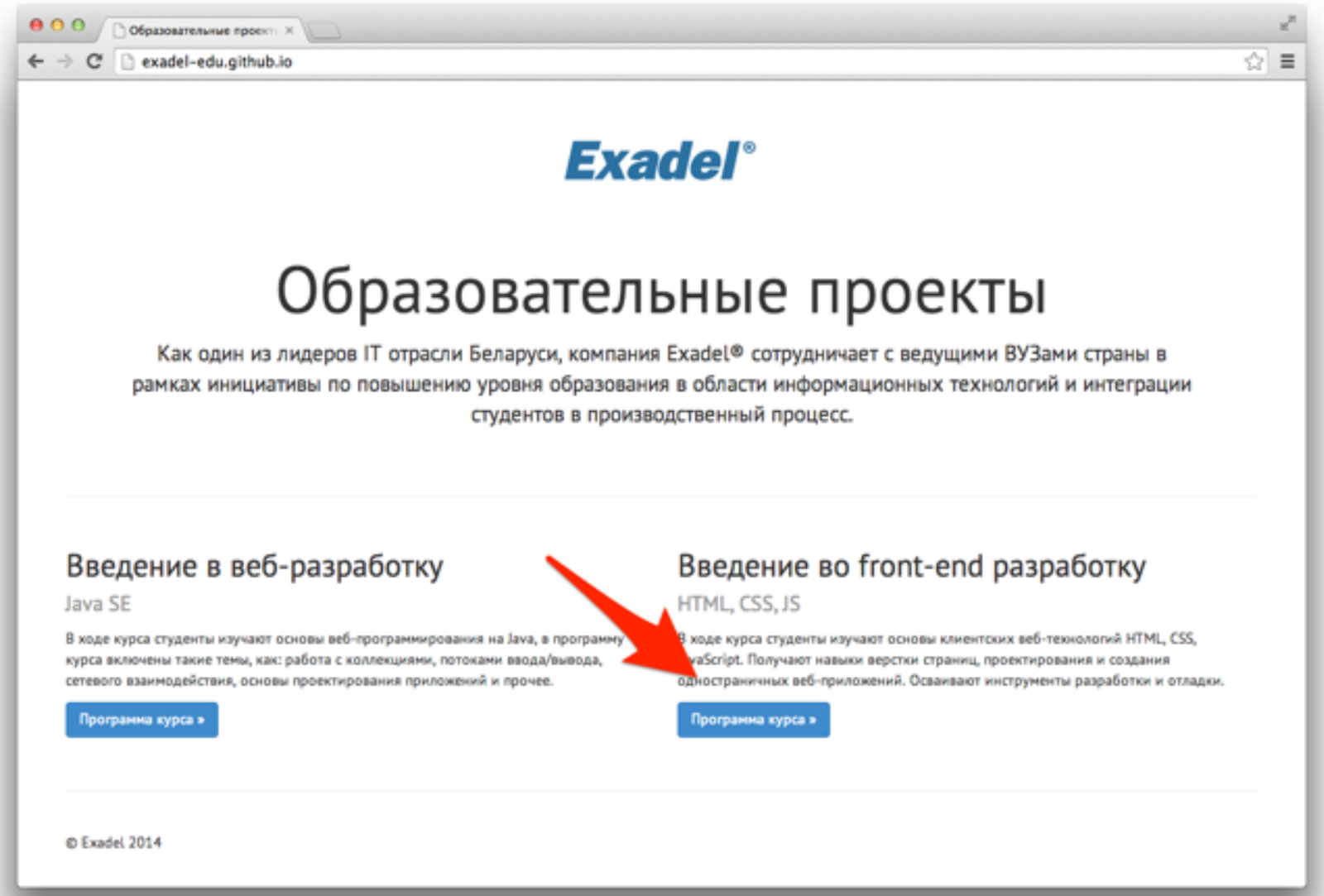
Front-end

dive-in

AJAX

Exadel®

Где раздобыть учебные материалы?



http://exadel-edu.github.io

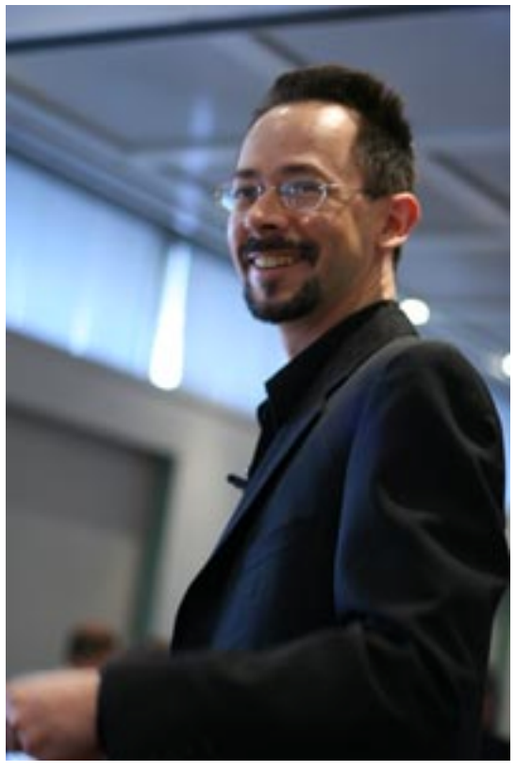
Exadel®

Что такое AJAX?

Asynchronous Javascript and XML

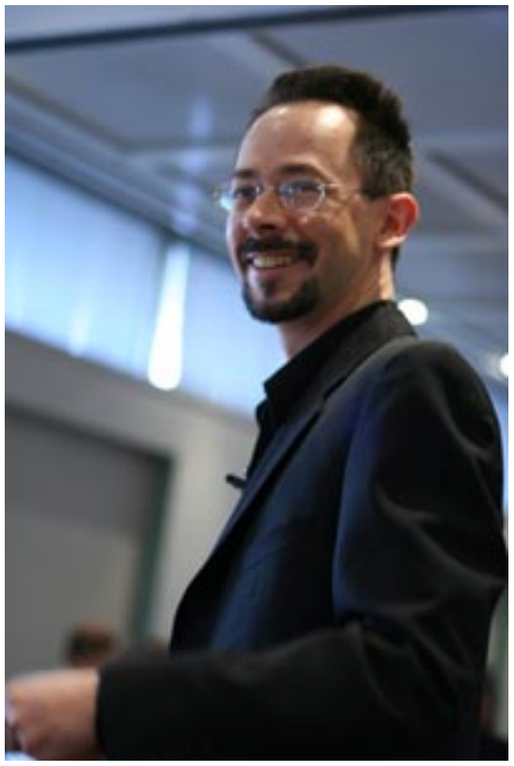
Подход к созданию веб-приложений, подразумевающий их работу без перезагрузки страницы.

Кто придумал?



Jesse James Garrett

Интересный факт



Jesse James Garrett

Jesse James Woodson

И зачем это нужно?

- снижение нагрузки на сервер
- уменьшение времени загрузки сраницы
- экономия трафика
- возможность создавать полноценные приложение в вебе

  RIA, Single Page Web Applications

А где можно использовать?

- голосовалки
- покупалки
- лайкалки
- корзины
- статистика
- пр.

А какие есть проблемы?

- история посещений
- добавление закладок

  History API

- индексация роботами

  !#

- работает только по http(s)

Что такое COMET?

Server Push

Как отправить запрос?

Основные транспорты

- <img/>
- <iframe/>
- <script/>
- XMLHttpRequest

транспорт <img/>

Как использовать

- создаем <img/>
- в src атрибут указываем целевой URL, в него же и пакуем данные для отправки
- в ответ приходит картинка

транспорт <iframe/>

Как использовать

- создаем <iframe/>
- в src атрибут указываем целевой URL, в него же и пакуем данные для отправки
- сервер возвращает ответ как HTML документ

Same-Origin policy

транспорт <script/>

Как использовать

- создаем <script/>
- в src атрибут указываем целевой URL, в него же и пакуем данные для отправки
- сервер возвращает ответ как фрагмент JS кода

транспорт XHR

Самый удобный из доступных транспортов

- кроссплатформенный, удобный API
- различные методы отправки данных
- работа с разными типами данных
- контроль заголовков
- кроссдоменные запросы*

XHR | отправка запросов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();
    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         "text/plain;charset=UTF-8");

    xhr.send(msg);
}
```

XHR | отправка запросов

```
function sendRequest(msg){
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         "text/plain;charset=UTF-8");

    xhr.send(msg);
}
```

XHR | отправка запросов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();
    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         "text/plain;charset=UTF-8");

    xhr.send(msg);
}
```

XHR | отправка запросов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();
    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         "text/plain;charset=UTF-8");

    xhr.send(msg);
}
```

Нельзя изменить

Accept-Charset	Content-Transfer-Encoding	TE
Accept-Encoding	Date	Trailer
Connection	Expect	Transfer-Encoding
Content-Length	Host	Upgrade
Cookie	Keep-Alive	User-Agent
Cookie2	Referer	Via

XHR | отправка запросов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();
    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         "text/plain;charset=UTF-8");

    xhr.send(msg);
}
```

XHR | обработка ответов

Структура ответа

- статусная строка
- заголовки
- тело ответа

XHR | обработка ответов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();

    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         “text/plain;charset=UTF-8");

    xhr.onreadystatechange = function(){
      if(xhr.readyState === 4 && xhr.status === 200){
        alert(xhr.responseText);
      }
    }

    xhr.send(msg);
}
```

XHR | состояния объекта

Константа	Числовой эквивалент	Значение
UNSENT	0	.open() не был вызван
OPENED	1	.open() был вызван
HEADERS_RECIEVED	2	получены заголовки ответа
LOADING	3	идет процесс получения ответа
DONE	4	ответ полностью получен

XHR | обработка ответов

```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();

    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         “text/plain;charset=UTF-8");

    xhr.onreadystatechange = function(){

      // XMLHttpRequest.DONE
      if(xhr.readyState === 4 && xhr.status === 200){
        alert(xhr.responseText);
      }
    }

    xhr.send(msg);
}
```

XHR | обработка ответов

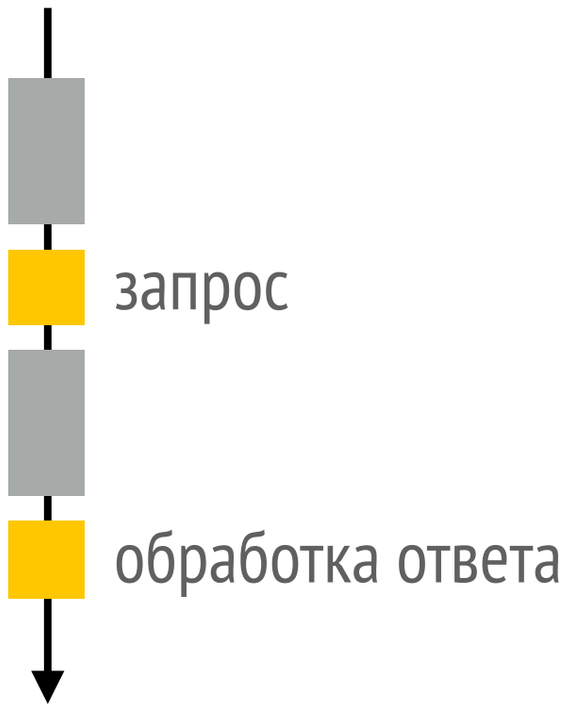
```
function sendRequest(msg){

    var xhr = new XMLHttpRequest();

    xhr.open("POST", "someData.json");
    xhr.setRequestHeader("Content-Type",
                         “text/plain;charset=UTF-8");

    xhr.onreadystatechange = function(){

      // XMLHttpRequest.DONE
      if(xhr.readyState === 4 && xhr.status === 200){
        alert(xhr.responseText);
      }
    }

    xhr.send(msg);
}
```

Какой еще статусный код подойдет?

XHR | sync/async requests

Sync

Async



XHR | sync/async requests

```
function sendSyncRequest(){

    var xhr = new XMLHttpRequest();
    xhr.open("GET", “http://test.dev“, false);
    // …
    xhr.send(null);
}
```

XHR | различные типы данных в запросе

URL-encoded

```
entity=students&grade=4&course=font-end
```

```
function sendRequest(msg, url, callback){

  var xhr = new XMLHttpRequest();

  xhr.open("POST", url);
  xhr.setRequestHeader("Content-Type",
                       "application/x-www-form-urlencoded");

  // …

  xhr.send(msg);
}
```

XHR | различные типы данных в запросе

JSON

```
{“entity”:”students”,”grade”:”4”,“course”:”front-end”}
```

```
function sendRequest(msg, url, callback){

  var xhr = new XMLHttpRequest();

  xhr.open("POST", url);
  xhr.setRequestHeader("Content-Type",
                       "application/json");

  // …

  xhr.send(msg);
}
```

XHR | различные типы данных в запросе

XML

```
<query>
  <find entity=”students”
        grade=”4”
        course=”front-end” />
</query>
```

```
function sendRequest(msg, url, callback){

  var xhr = new XMLHttpRequest();

  xhr.open("POST", url);
  xhr.setRequestHeader("Content-Type",
                       "application/xml");

  // …

  xhr.send(msg);
}
```

XHR | различные типы данных в ответе

```
function sendRequest(msg, url, callback){

  var xhr = new XMLHttpRequest();

  xhr.open("POST", url);
  xhr.setRequestHeader("Content-Type",
                       "text/plain;charset=UTF-8");

  xhr.onreadystatechange = function(){

    var type = xhr.getResponseHeader("Content-Type");

    if(type.indexOf("xml") !== -1){
      callback(xhr.responseXML);
    }
    else if(type === "application/json"){
      callback(JSON.parse(xhr.responseText))
    }
    else{
      callback(xhr.responseText);
    }
  }

  xhr.send(msg);
}
```

XHR | cross-origin request

С помощью XHR отправлять кросс-доменные запросы можно только в том случае, если сервер на который был отправлен запрос вернет в ответе заголовок

Access-Control-Allow-Origin

со значением домена, с которого был инициирован запрос.

JSONP

Суть метода

- создаем тэг <script>
- в URL, значение атрибута src, добавляем параметр callback, в который помещается имя функции, которая будет вызвана на клиенте, после получения ответа
- сервер оформляет ответ в виде вызова этой самой функции и передает нужные данные в нее, как аргументы
- как только содрежимое тэга <scirpt> загружено на страницу – оно сразу выполняется, следовательно вызывается наша функция

JSONP

```
function sendJSONPRequest(URL, callback){

    var cbNum  = "cb" + sendJSONPRequest.counter++,
        cbName = "jsonpCallback" + cbNum,
        script = document.createElement("script");

    if(URL.indexOf("?") === -1)
        URL += ("?callback=" + cbName);
    else
        URL += ("&callback=" + cbName);

    jsonpCallback[cbNum] = function(responseData){

        try{
            callback(responseData);
        }
        finally{
            delete jsonpCallback[cbNum];
            script.parentNode.removeChild(script);
        }
    }

    script.src = URL;
    document.body.appendChild(script);
}
```

JSONP

```
function sendJSONPRequest(URL, callback){

    var cbNum  = "cb" + sendJSONPRequest.counter++,
        cbName = "jsonpCallback" + cbNum,
        script = document.createElement("script");

    if(URL.indexOf("?") === -1)
        URL += ("?callback=" + cbName);
    else
        URL += ("&callback=" + cbName);

    jsonpCallback[cbNum] = function(responseData){

        try{
            callback(responseData);
        }
        finally{
            delete jsonpCallback[cbNum];
            script.parentNode.removeChild(script);
        }
    }

    script.src = URL;
    document.body.appendChild(script);
}
```

JSONP

```
function sendJSONPRequest(URL, callback){

    var cbNum  = "cb" + sendJSONPRequest.counter++,
        cbName = "jsonpCallback" + cbNum,
        script = document.createElement("script");

    if(URL.indexOf("?") === -1)
        URL += ("?callback=" + cbName);
    else
        URL += ("&callback=" + cbName);

    jsonpCallback[cbNum] = function(responseData){

        try{
            callback(responseData);
        }
        finally{
            delete jsonpCallback[cbNum];
            script.parentNode.removeChild(script);
        }
    }

    script.src = URL;
    document.body.appendChild(script);
}
```

JSONP

```
function sendJSONPRequest(URL, callback){

    var cbNum  = "cb" + sendJSONPRequest.counter++,
        cbName = "jsonpCallback" + cbNum,
        script = document.createElement("script");

    if(URL.indexOf("?") === -1)
        URL += ("?callback=" + cbName);
    else
        URL += ("&callback=" + cbName);

    jsonpCallback[cbNum] = function(responseData){

        try{
            callback(responseData);
        }
        finally{
            delete jsonpCallback[cbNum];
            script.parentNode.removeChild(script);
        }
    }

    script.src = URL;
    document.body.appendChild(script);
}
```

JSONP

```
function sendJSONPRequest(URL, callback){

    var cbNum  = "cb" + sendJSONPRequest.counter++,
        cbName = "jsonpCallback" + cbNum,
        script = document.createElement("script");

    if(URL.indexOf("?") === -1)
        URL += ("?callback=" + cbName);
    else
        URL += ("&callback=" + cbName);

    jsonpCallback[cbNum] = function(responseData){

        try{
            callback(responseData);
        }
        finally{
            delete jsonpCallback[cbNum];
            script.parentNode.removeChild(script);
        }
    }

    script.src = URL;
    document.body.appendChild(script);
}
```

The End

учебные материалы и практическое задание

http://exadel-edu.github.io

Exadel®