Разработка Интернет-приложений
Лекция 5: Интерактивные пользовательские интерфейсы

AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

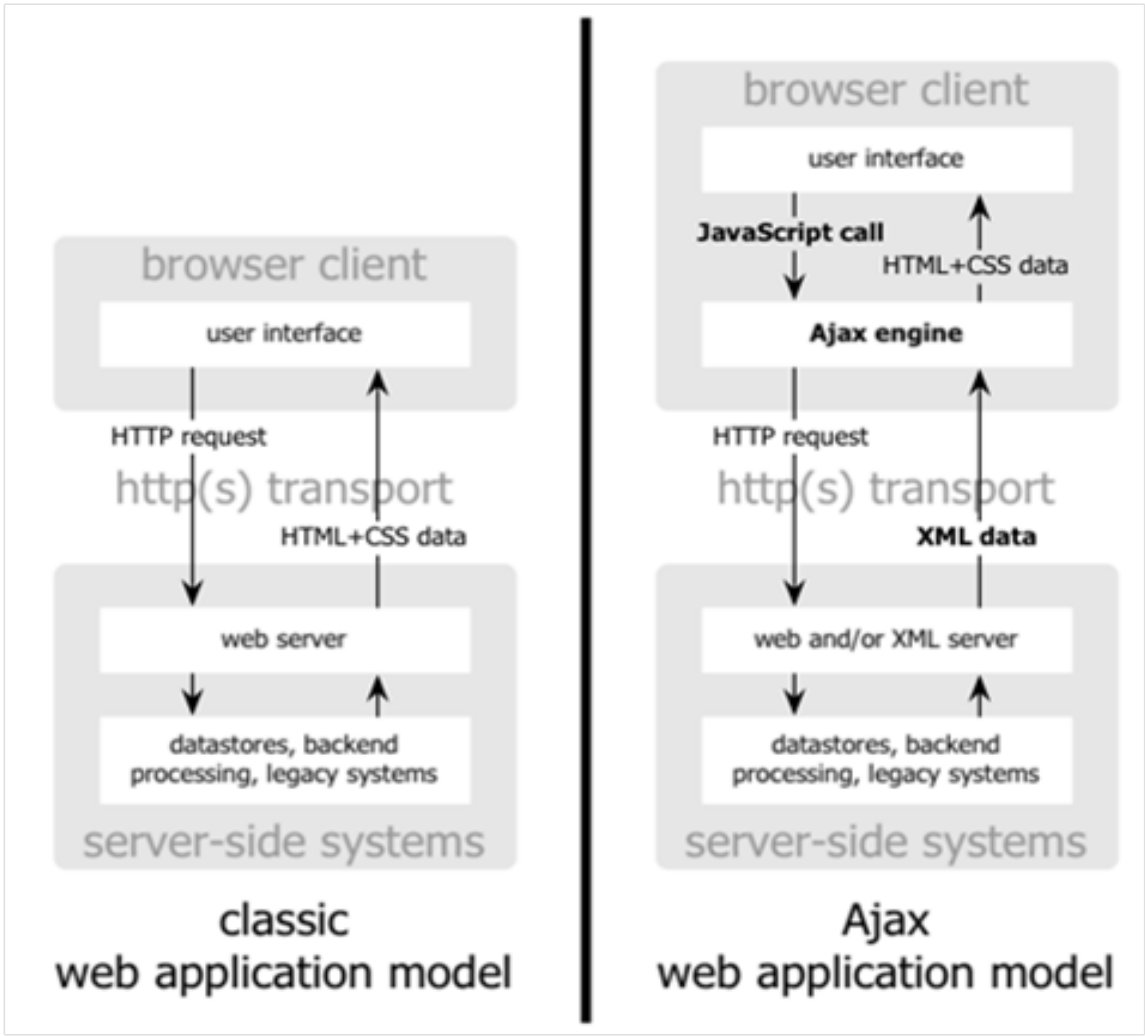
AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

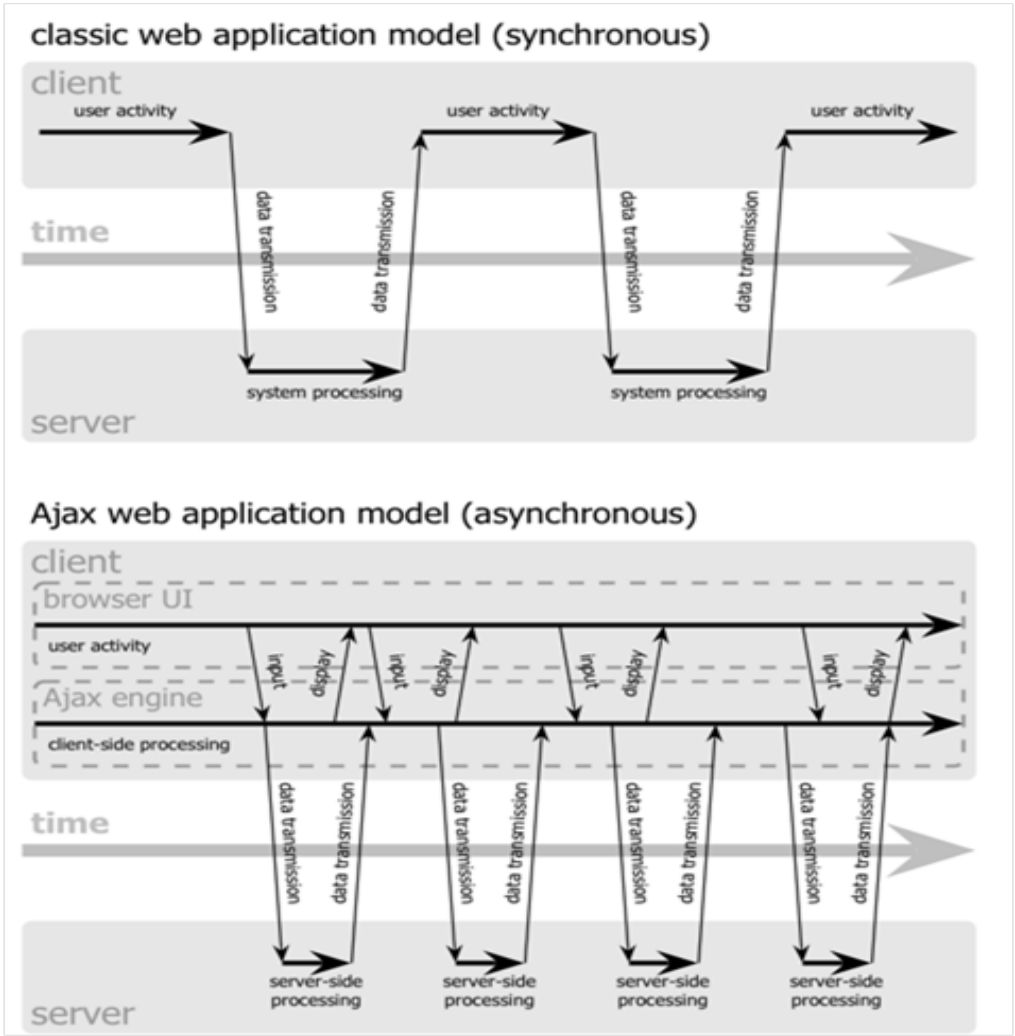
AJAX

- AJAX – Asynchronous JavaScript + XML
  - Представление
    XHTML и CSS
  - Динамика представления
    DOM
  - Формат данных
    XML и XSLT, HTML (AJAH), JSON (AJAJ)
  - Канал обмена данными
    XMLHttpRequest, <script>, iframe
  - JavaScript

Сравнение традиционной и AJAX-модели web-приложений



Сравнение традиционной и AJAX-модели web-приложений



Преимущества

- Возможности построения динамических пользовательских интерфейсов
- Экономия трафика
- Уменьшение нагрузки на сервер

Преимущества

- Возможности построения динамических пользовательских интерфейсов
- Экономия трафика
- Уменьшение нагрузки на сервер

Преимущества

- Возможности построения динамических пользовательских интерфейсов
- Экономия трафика
- Уменьшение нагрузки на сервер

Недостатки

- Полностью девальвируется идея гипертекста
- Отсутствие интеграции со стандартными инструментами браузера
- Динамически загружаемое содержимое недоступно поисковикам
- Старые методы учёта статистики сайтов становятся неактуальными

Недостатки

- Полностью девальвируется идея гипертекста
- Отсутствие интеграции со стандартными инструментами браузера
- Динамически загружаемое содержимое недоступно поисковикам
- Старые методы учёта статистики сайтов становятся неактуальными

Недостатки

- Полностью девальвируется идея гипертекста
- Отсутствие интеграции со стандартными инструментами браузера
- Динамически загружаемое содержимое недоступно поисковикам
- Старые методы учёта статистики сайтов становятся неактуальными

Недостатки

- Полностью девальвируется идея гипертекста
- Отсутствие интеграции со стандартными инструментами браузера
- Динамически загружаемое содержимое недоступно поисковикам
- Старые методы учёта статистики сайтов становятся неактуальными

Канал обмена данными

- Объекты
  - ActiveXObject("Microsoft.XMLHTTP") //IE
  - XMLHttpRequest() //остальные
- Другие
  - Через скрытые фреймы
  - Посредством автоматической генерации тегов <script src="<URL>"></script>
  - Вероятно, какие-то еще. . .

Канал обмена данными

- Объекты
  - ActiveXObject("Microsoft.XMLHTTP") //IE
  - XMLHttpRequest() //остальные
- Другие
  - Через скрытые фреймы
  - Посредством автоматической генерации тегов <**script src**="<URL>"></**script**>
  - Вероятно, какие-то еще. . .

Формат данных

- XML

```
<xml>
  <contacts>
    <person firstname="Joe" lastname="Smith" phone="555-1212" />
    <person firstname="Sam" lastname="Stevens" phone="123-4567" />
  </contacts>
</xml>
```

- JSON

```
{contacts:[
  {"firstname":"Joe", "lastname":"Smith",  "phone":"
    555-1212"},
  {"firstname":"Sam", "lastname":"Stevens", "phone":"
    123-4567"}
]}
```

- HTML

Формат данных

- XML

```
<xml>
  <contacts>
    <person firstname="Joe" lastname="Smith" phone="555-1212" />
    <person firstname="Sam" lastname="Stevens" phone="123-4567" />
  </contacts>
</xml>
```

- JSON

```
{contacts:[
  {"firstname":"Joe", "lastname":"Smith",  "phone":"
    555-1212"},
  {"firstname":"Sam", "lastname":"Stevens", "phone":"
    123-4567"}
]}
```

- HTML

Формат данных

- XML

```
<xml>
  <contacts>
    <person firstname="Joe" lastname="Smith" phone="555-1212" />
    <person firstname="Sam" lastname="Stevens" phone="123-4567" />
  </contacts>
</xml>
```

- JSON

```
{contacts:[
  {"firstname":"Joe", "lastname":"Smith",  "phone":"
    555-1212"},
  {"firstname":"Sam", "lastname":"Stevens", "phone":"
    123-4567"}
]}
```

- HTML

JSON

JSON - JavaScript Object Notation

www.json.org

```
//список студентов + список предметов
{students: [
    {name: "Иванов", course: 1, group: 2 },
    {name: "Петрова", course: 3, group: 1 }
 ],
 subjects: [
    {name: "История", course: 1, semestr: 1 },
    {name: "ЯиСП", course: 1, semestr: 2 }
 ]}
```

Лекция 4: Клиентская часть
ООП, создание объекта

- Классический вариант

```
var obj=new Object();
obj.prop1=5;          // свойство создается во время жизни
    объекта
obj.func1=function() { // метод создается во время жизни
    объекта
  alert(this.prop1);
}
obj.func1();
```

- JSON-синтаксис (эквивалентно)

```
var obj2={
  prop1: 5,
  func1: function() {
alert(this.prop1);
  }
```

Асинхронность

```
var oXml = new XMLHttpRequest();
oXml.open("GET", "getData.php", true);
oXml.onreadystatechange = processingFunction;
oXml.send();

function processingFunction(){
  if(oXml.readyState!=4) return; //запрос не выполнен

  var xmlDoc = oXml.responseXML;
  //Результаты обрабатываются здесь.
}
```

Архитектура

- Толстый клиент
  Дублирование моделей (клиент и сервер)
  - Верификация данных
  - Синхронизация моделей
  - Различные языки
- Реализация vs. Генерация клиентского кода

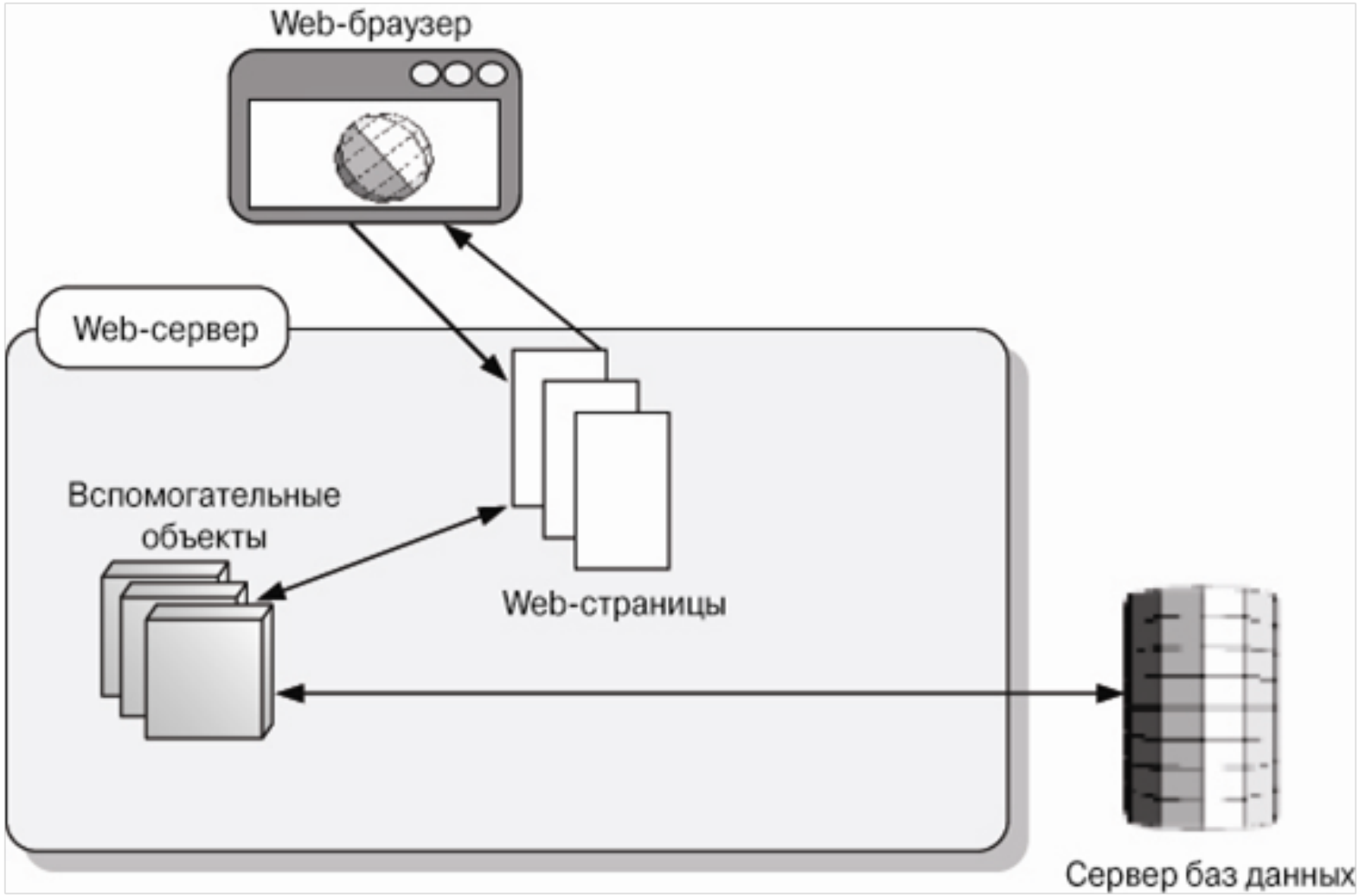
Архитектура

- Толстый клиент
  Дублирование моделей (клиент и сервер)
  - Верификация данных
  - Синхронизация моделей
  - Различные языки
- Реализация vs. Генерация клиентского кода

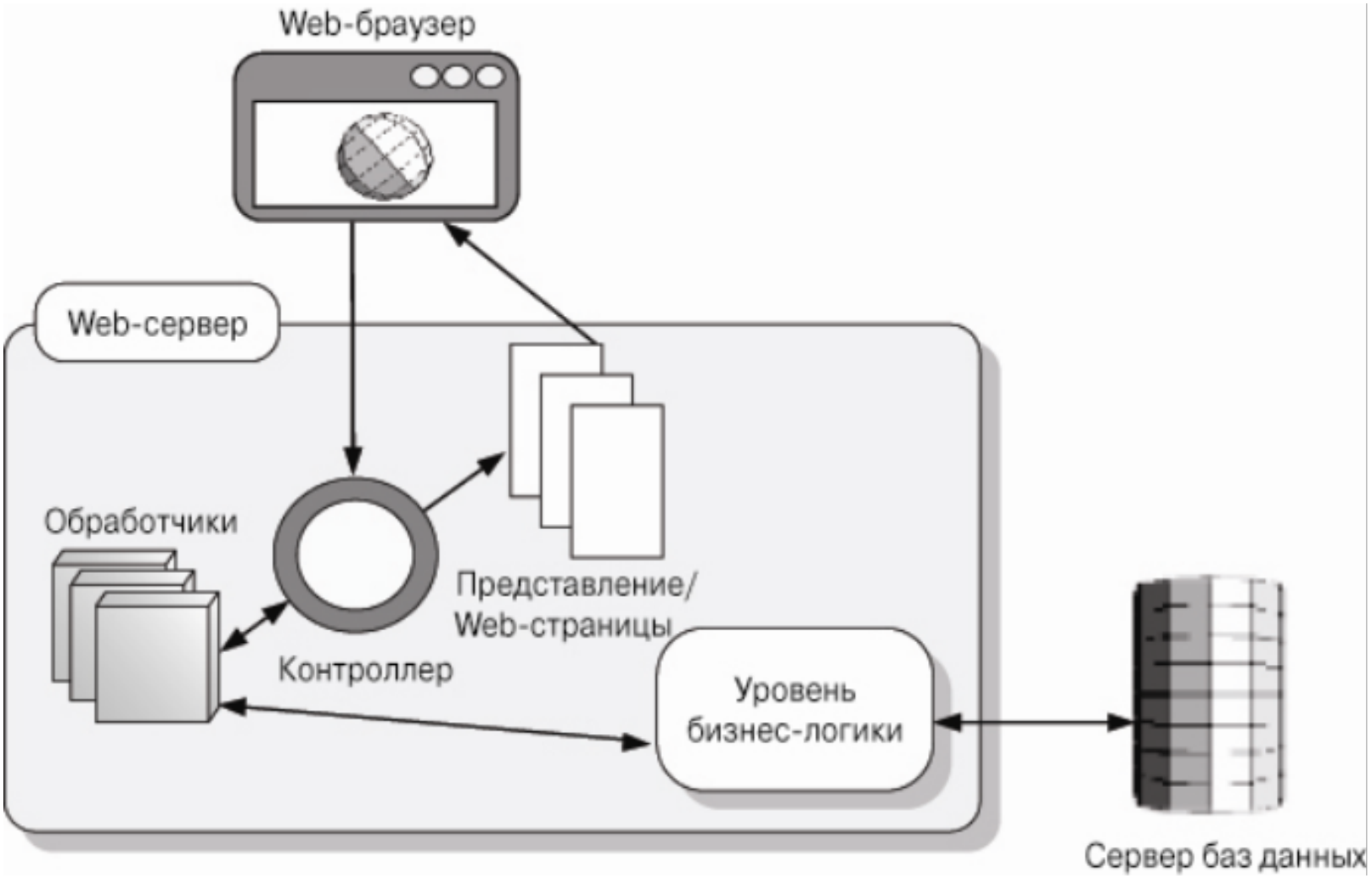
Архитектура

- Толстый клиент
  Дублирование моделей (клиент и сервер)
  - Верификация данных
  - Синхронизация моделей
  - Различные языки
- Реализация vs. Генерация клиентского кода

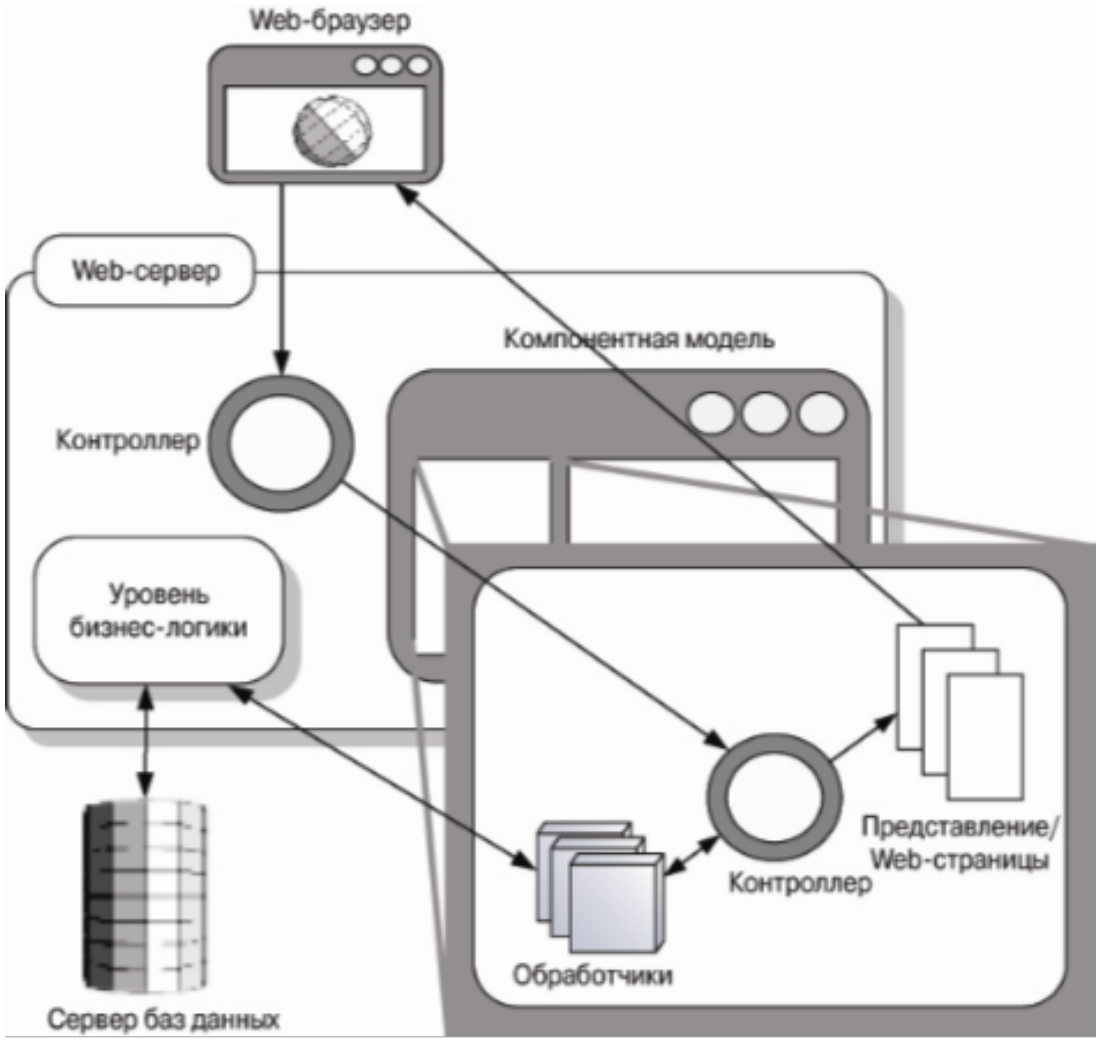
Самая простая архитектура — это отсутствие таковой



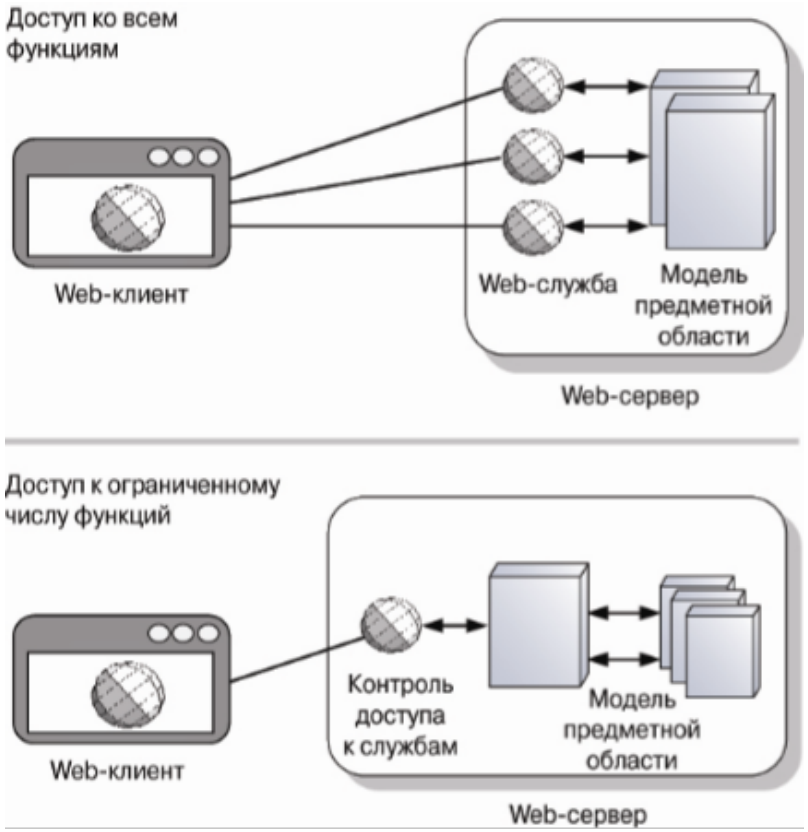
Model2


Web-браузер
Web-сервер
Обработчики
Представление/
Web-страницы
Контроллер
Уровень
бизнес-логики
Сервер баз данных

Компонентная архитектура

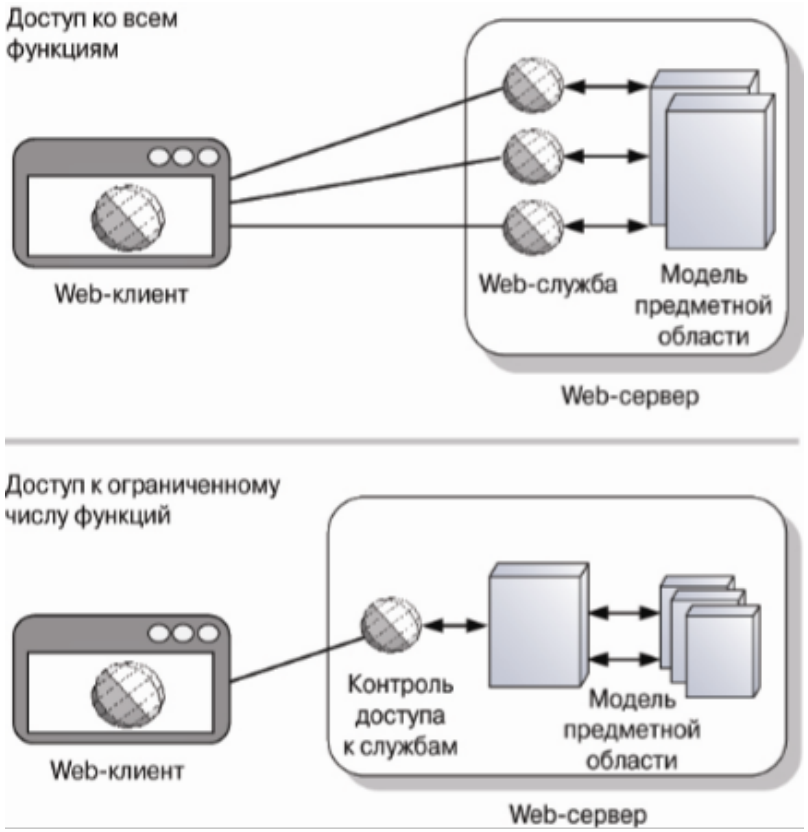


Архитектуры, ориентированные на использование Web-служб



RESTful

Архитектуры, ориентированные на использование Web-служб



RESTful

Передача данных серверу

- GET: URL ресурсов
- POST: HTML-форма
- XMLHttpRequest
- Другие методы HTTP:

```
HEAD, PUT, PATCH, DELETE, LINK, UNLINK
  (?: OPTIONS, TRACE)
```

Передача данных серверу

- GET: URL ресурсов
- POST: HTML-форма
- XMLHttpRequest
- Другие методы HTTP:

```
HEAD, PUT, PATCH, DELETE, LINK, UNLINK
  (?: OPTIONS, TRACE)
```

Передача данных серверу

- GET: URL ресурсов
- POST: HTML-форма
- XMLHttpRequest
- Другие методы HTTP:

```
HEAD, PUT, PATCH, DELETE, LINK, UNLINK
  (?: OPTIONS, TRACE)
```

Передача данных серверу

- GET: URL ресурсов
- POST: HTML-форма
- XMLHttpRequest
- Другие методы HTTP:

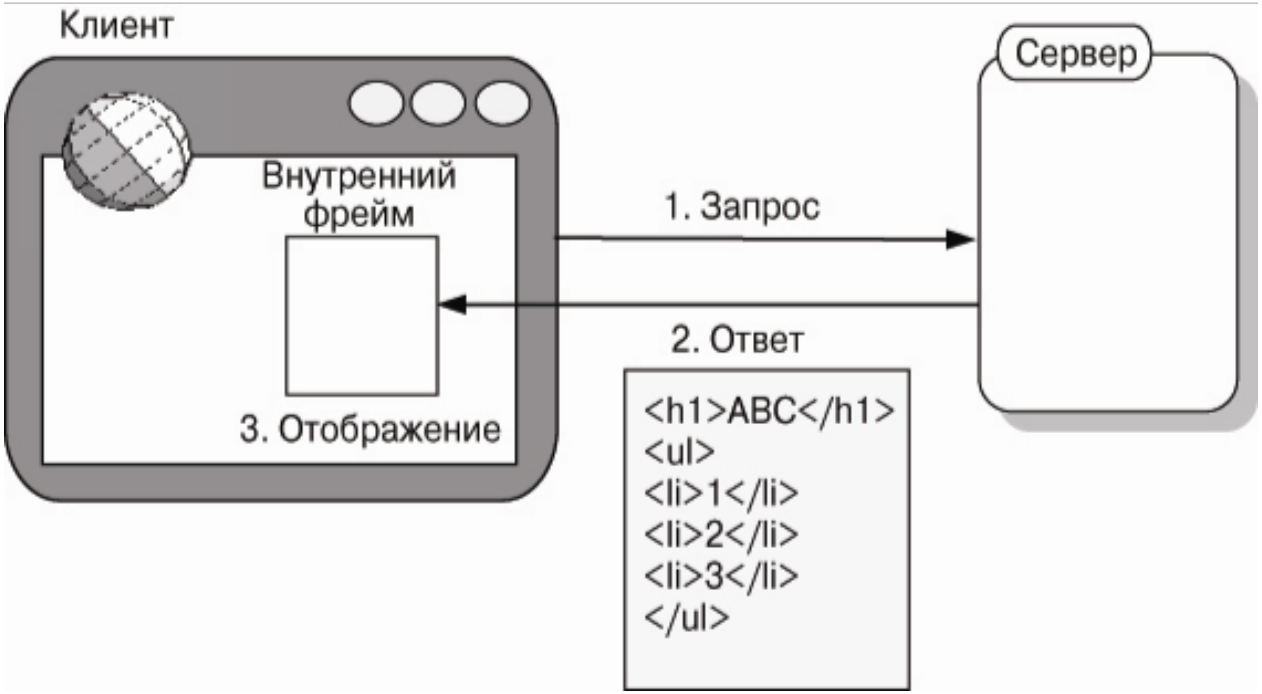
```
HEAD, PUT, PATCH, DELETE, LINK, UNLINK
 (?: OPTIONS, TRACE)
```

Обмен данными

- Взаимодействие, затрагивающее только клиентскую программу

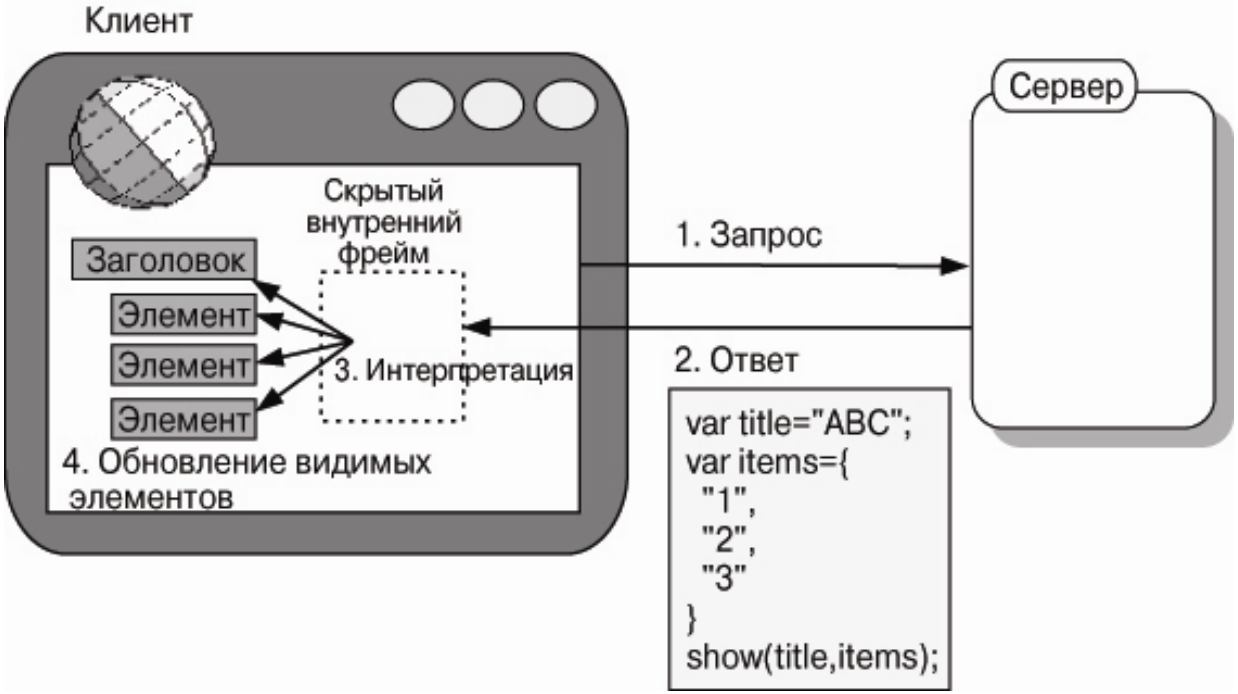
Обмен данными

- Взаимодействие, затрагивающее только клиентскую программу
- Взаимодействие, ориентированное на содержимое *innerHTML*



Обмен данными

- Взаимодействие, затрагивающее только клиентскую программу
- Взаимодействие, ориентированное на содержимое innerHTML
- Взаимодействие, ориентированное на сценарий



Обмен данными

- Взаимодействие, затрагивающее только клиентскую программу
- Взаимодействие, ориентированное на содержимое innerHTML
- Взаимодействие, ориентированное на сценарий
- Взаимодействие, ориентированное на данные
  XSLT