

Проектирование веб-приложений

От идеи до кода



Требования

Анализ

Проектирование

Разработка

Тестирование

Требования

Функциональные

- Какие задачи решает
- Какими инструментами

Источник

- ЦА
- Заказчик/бизнес
- Технологии

Техническое задание

Документ, регламентирующий требования, предполагаемый вид, устройство и процесс разработки продукта

- Термины и определения
- Назначение
- Описание механики
- Структура
- Дизайн
- Требования к технической и программной реализации
- Обязанности сторон
- Условия сдачи-приемки и т.д.



Прототип

Low
Fidelity
Prototypes



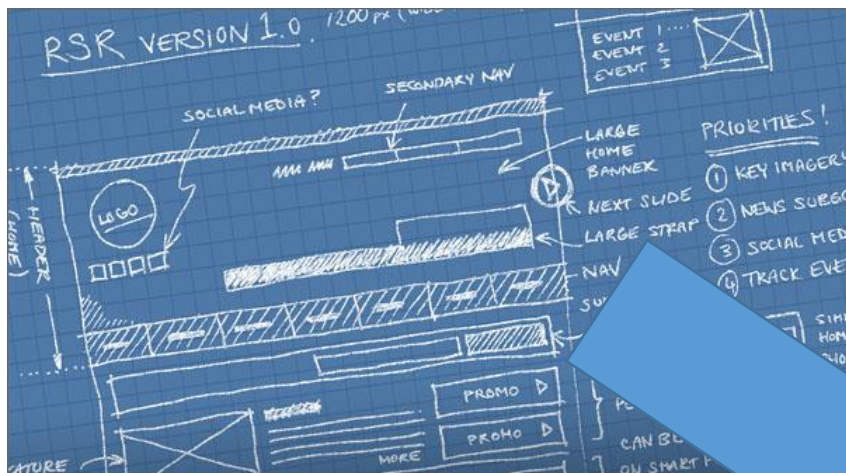
Medium



High
Fidelity
Prototypes



Проектирование архитектуры



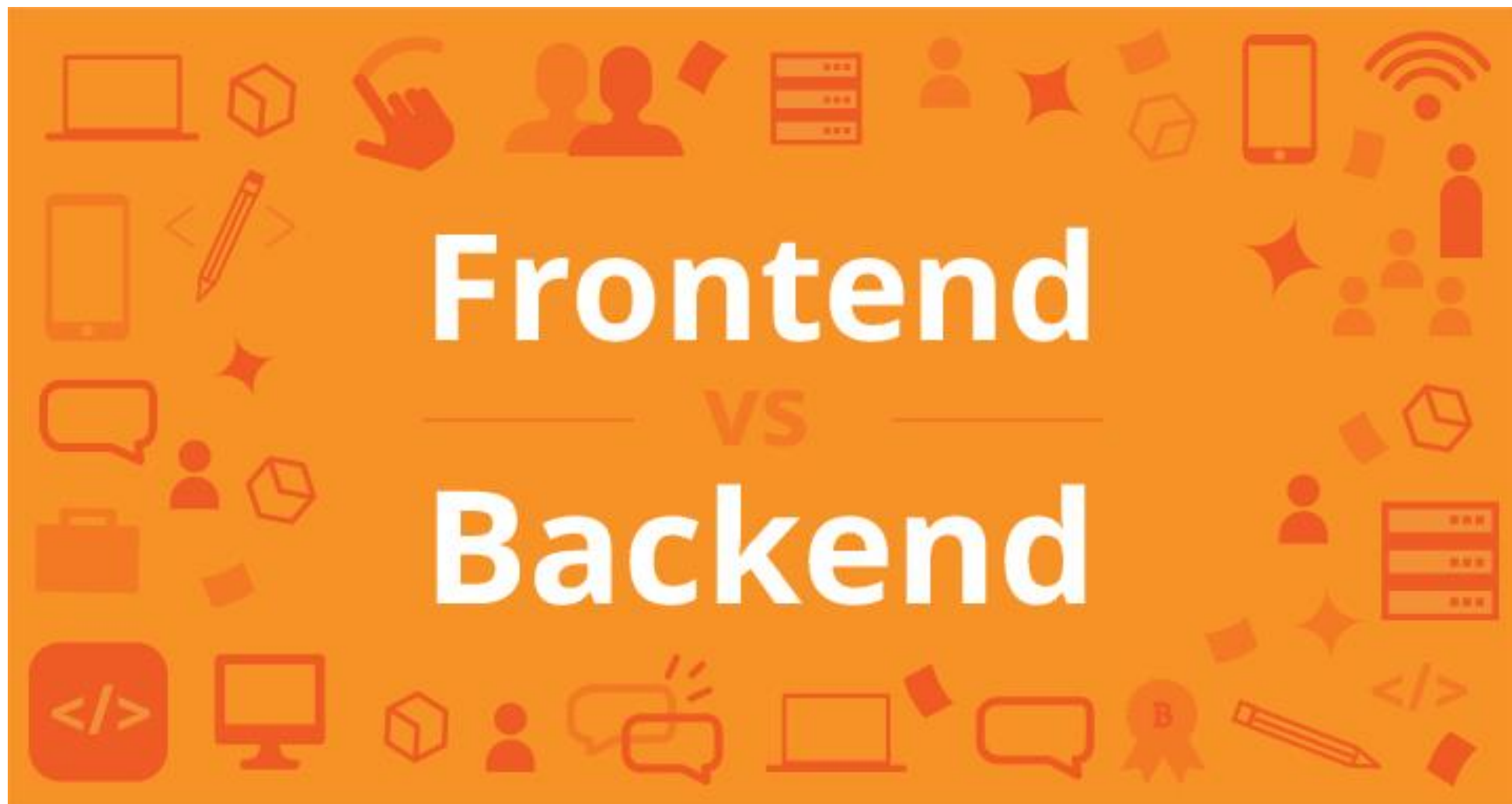
Архитектура

Архитектура программного обеспечения (англ. software architecture) — это высокоуровневая структура программной системы, дисциплина создания таких структур и документация по этим структурам.

Хорошая архитектура – это важно



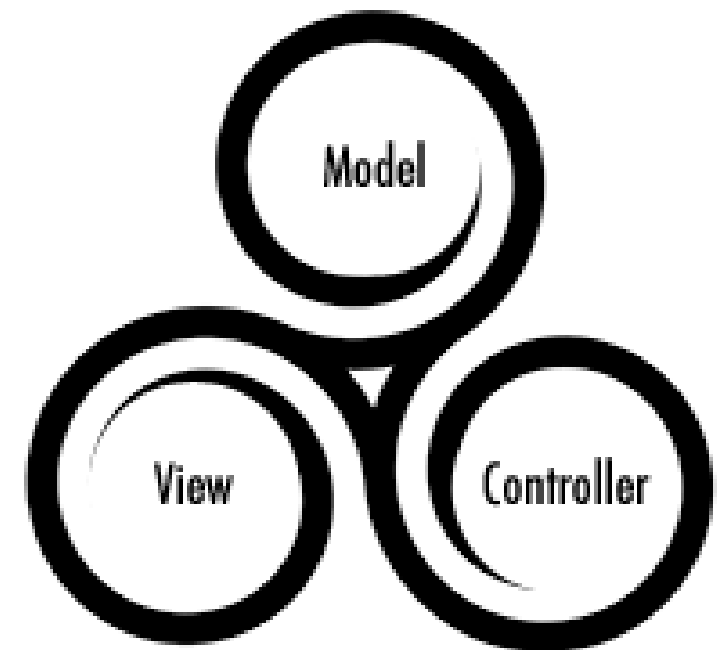
Клиент-серверная архитектура



Паттерны проектирования

Семейство «MVC-паттернов» :

- MVC
- MTV
- MPV
- MVVM
- MVPVM
-

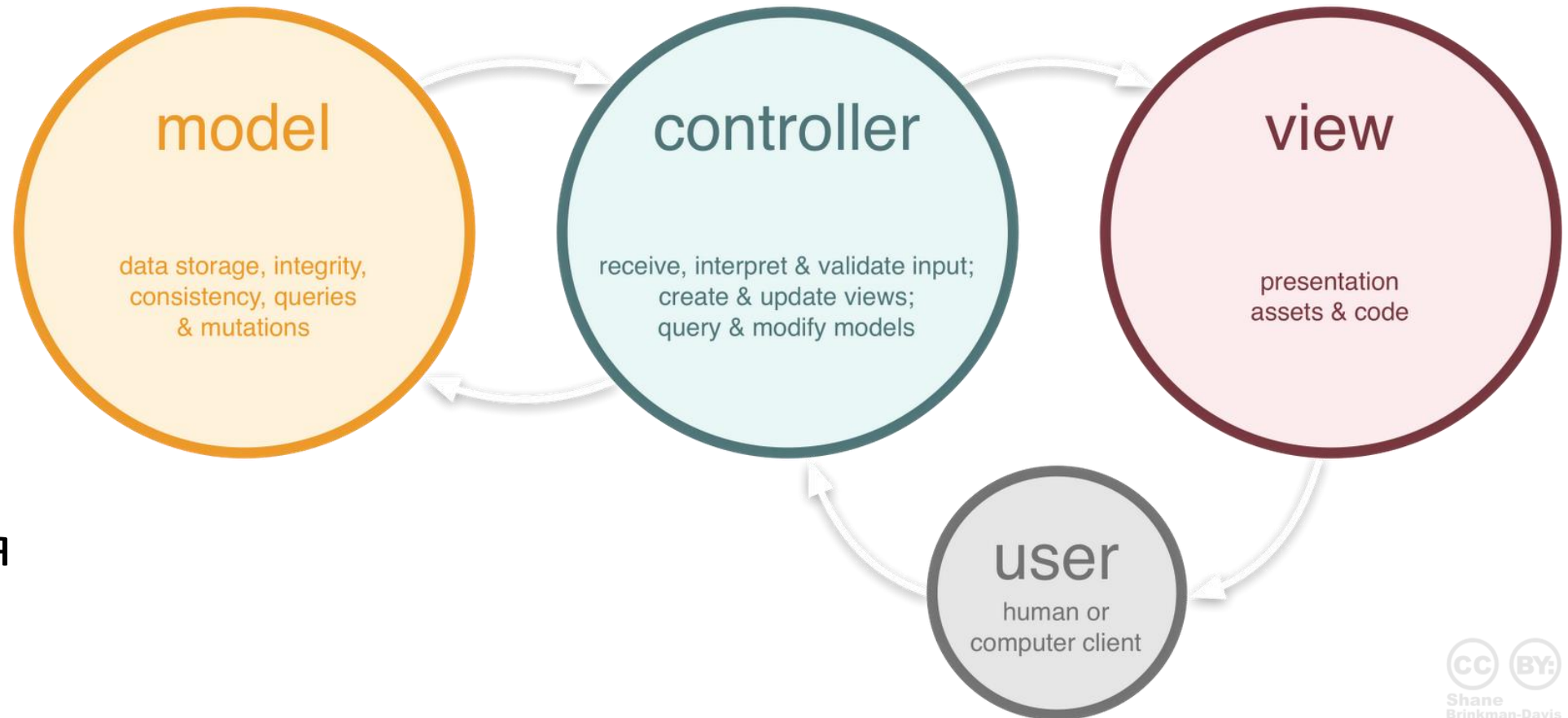


MVC

Model
View
Controller

Трюгве Рееенскауг в 1979 году, язык Smalltalk

MVC: Model-View-Controller



- Активная
- Пассивная модель

FSUC: Fat Stupid Ugly Controllers



MTV: Model-Template-View (Django-взгляд на MVC)

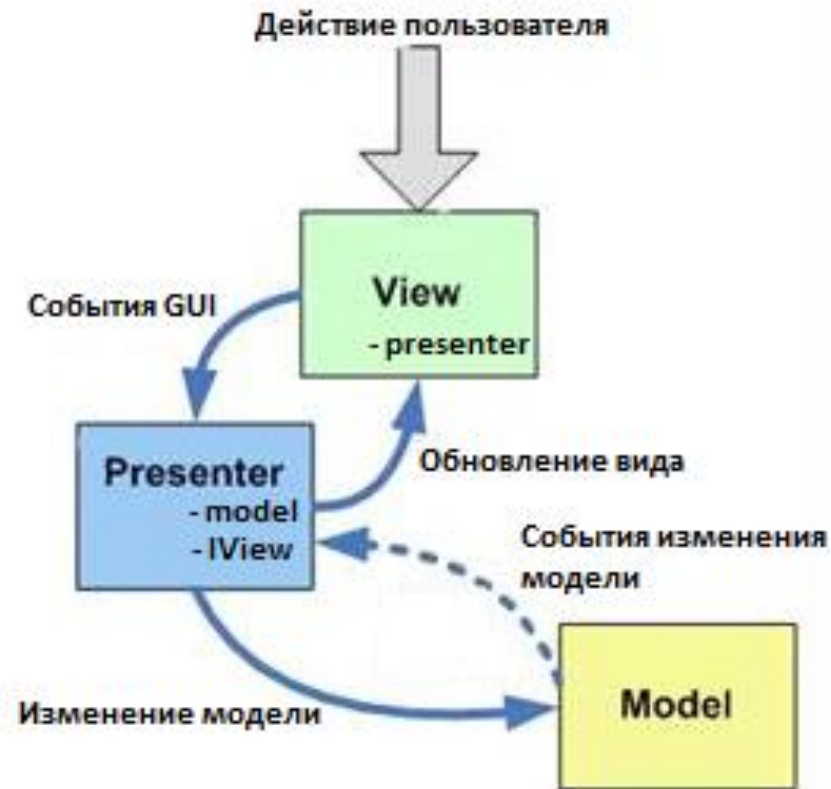
MVC vs. MTV

Model --> Model

View --> Template

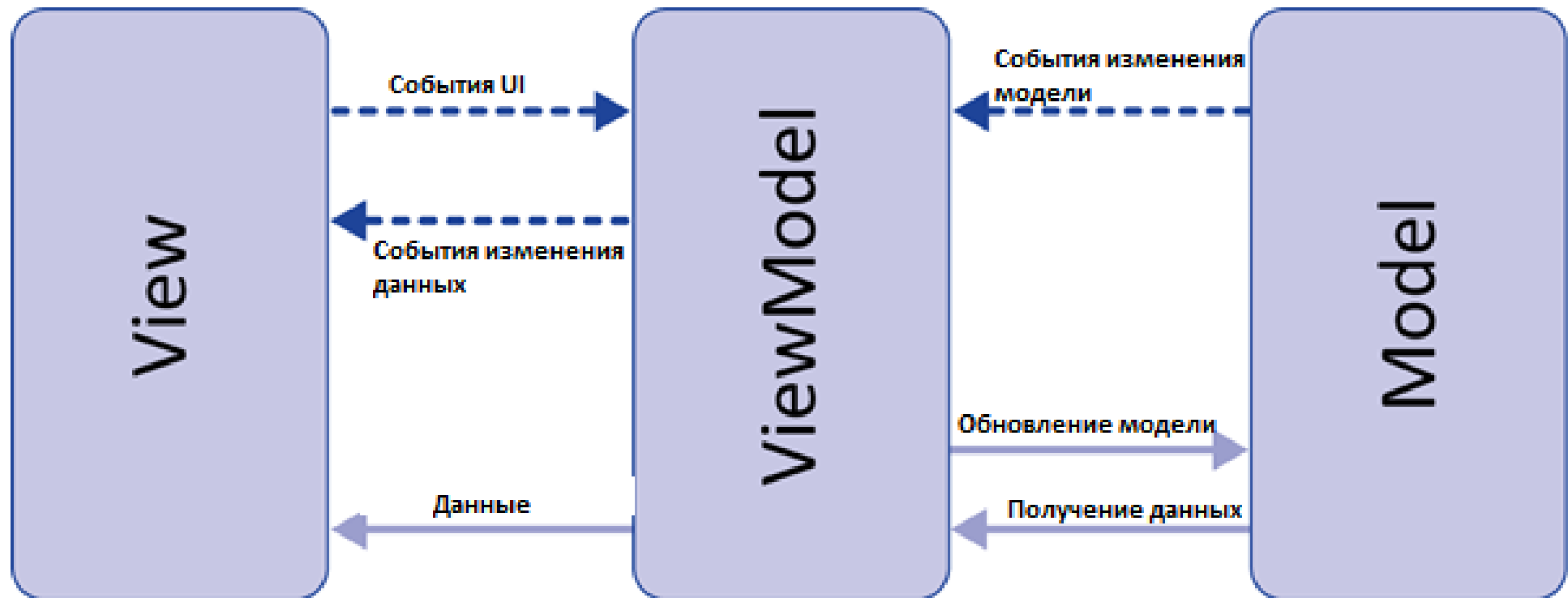
Controller --> View

MVP: Model-View-Presentation

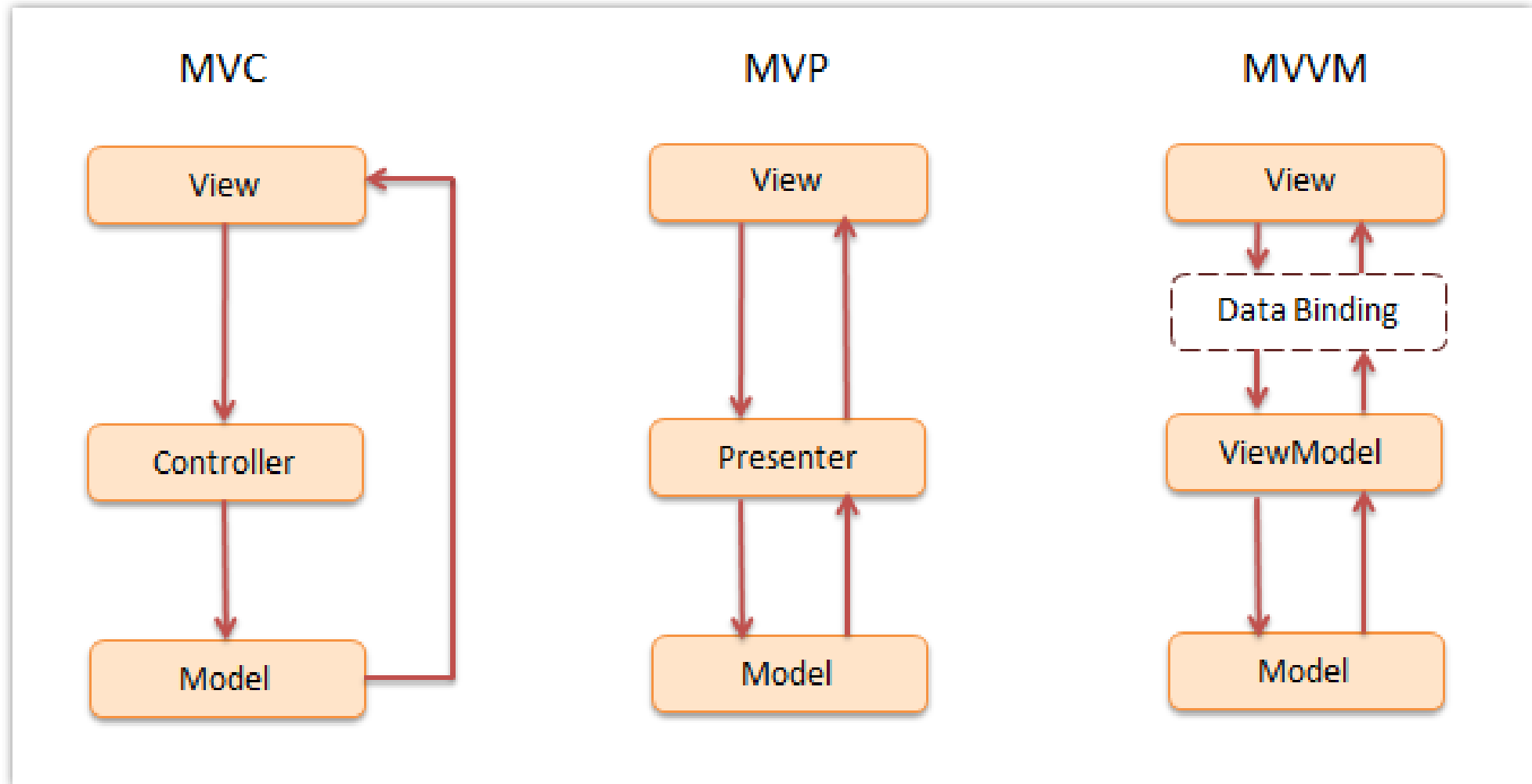


Model-View-Presenter

MVVM: Model-View-View Model

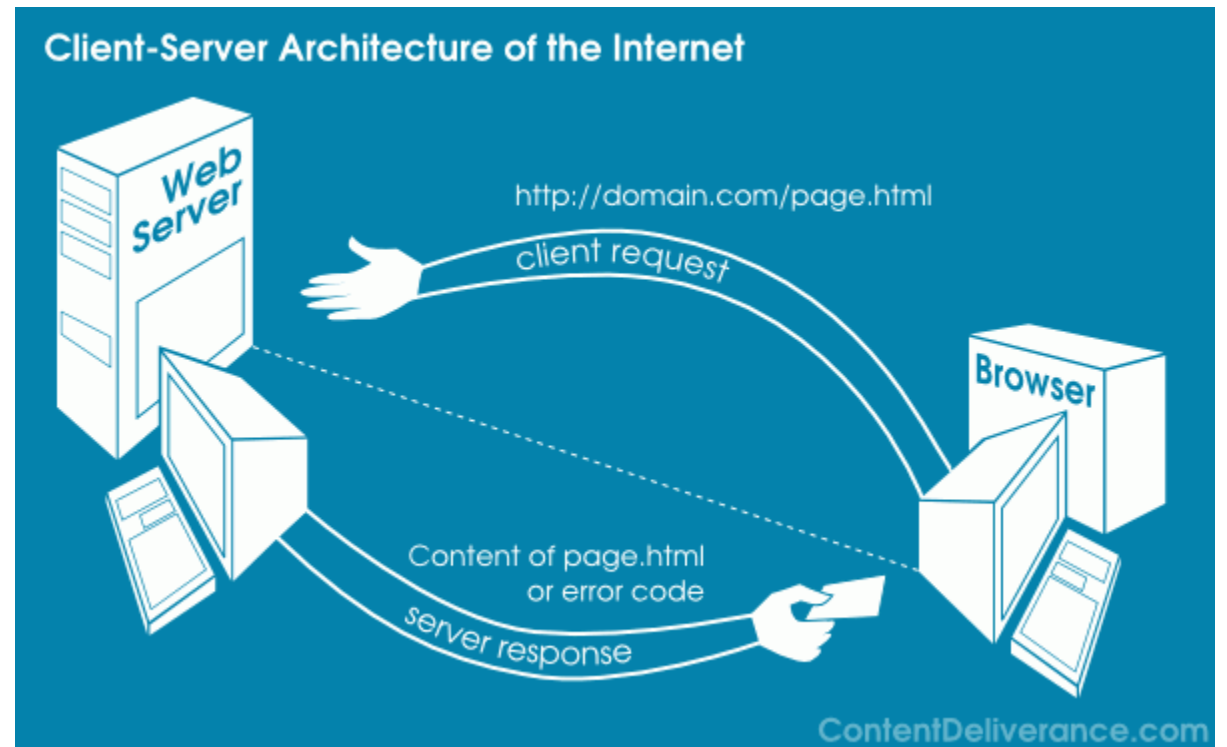


MVC vs MVP vs MVVM

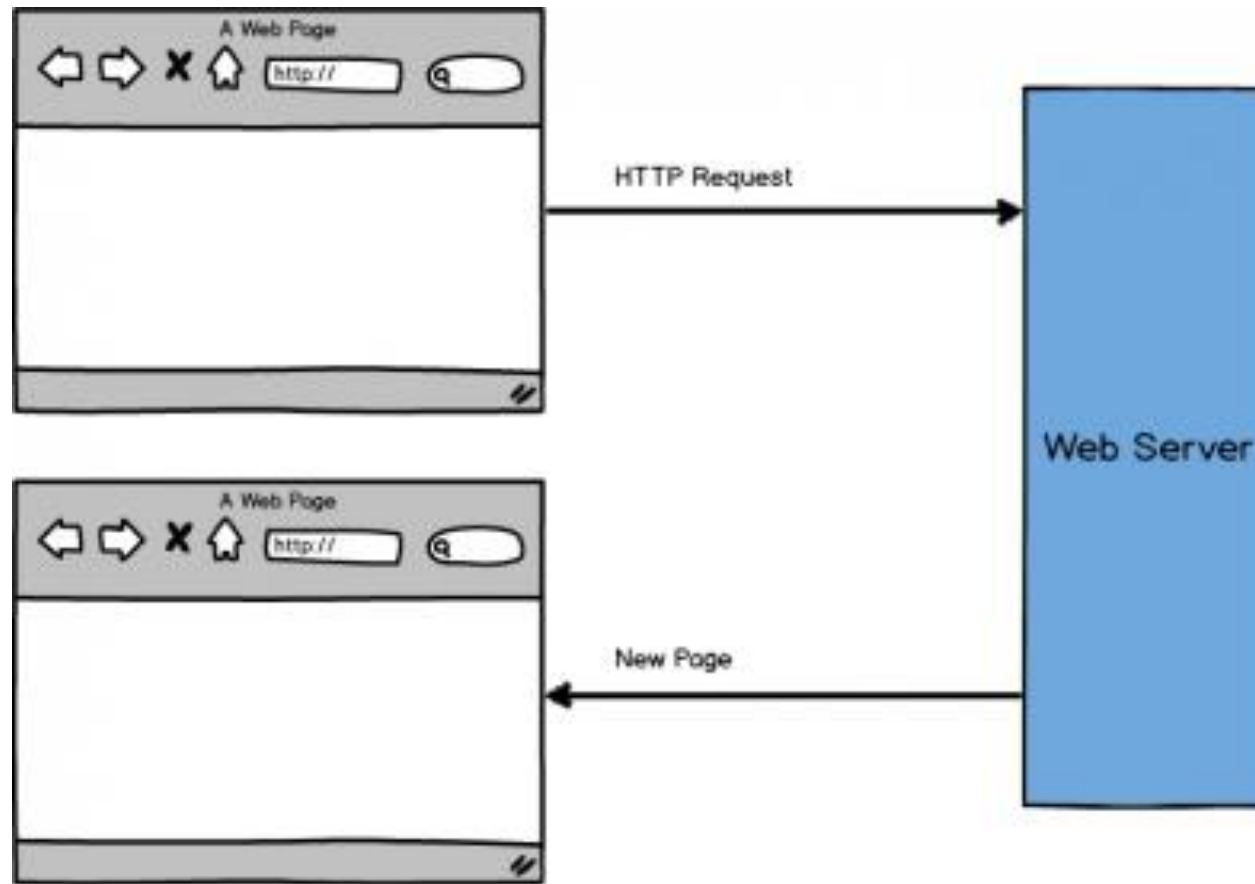


Классификации клиент-серверных архитектур в Вебе

- МРА-SPA
- Толстый-Тонкий
- Изоморфный

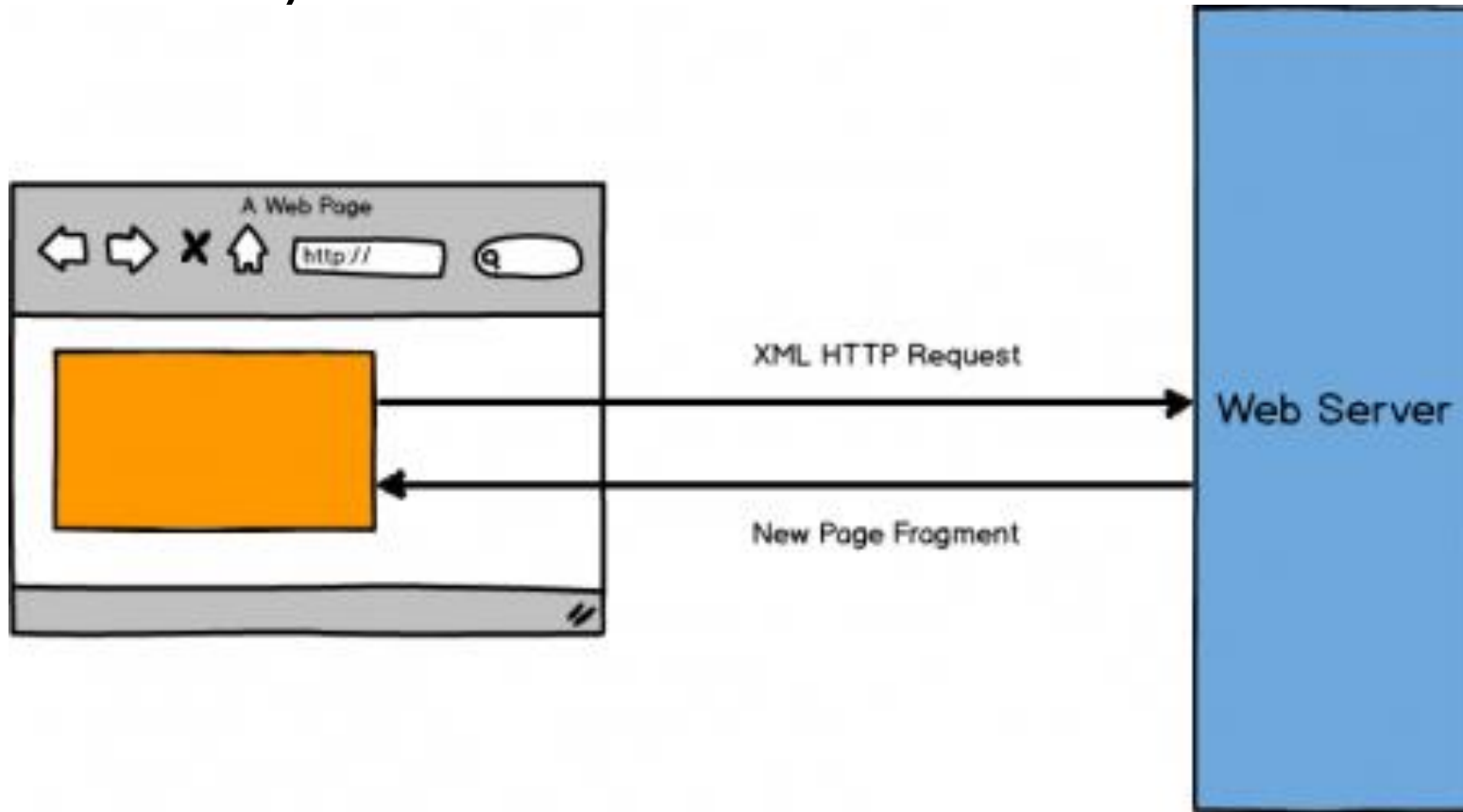


Многостраничное приложение (MPA, Multi Page Application)



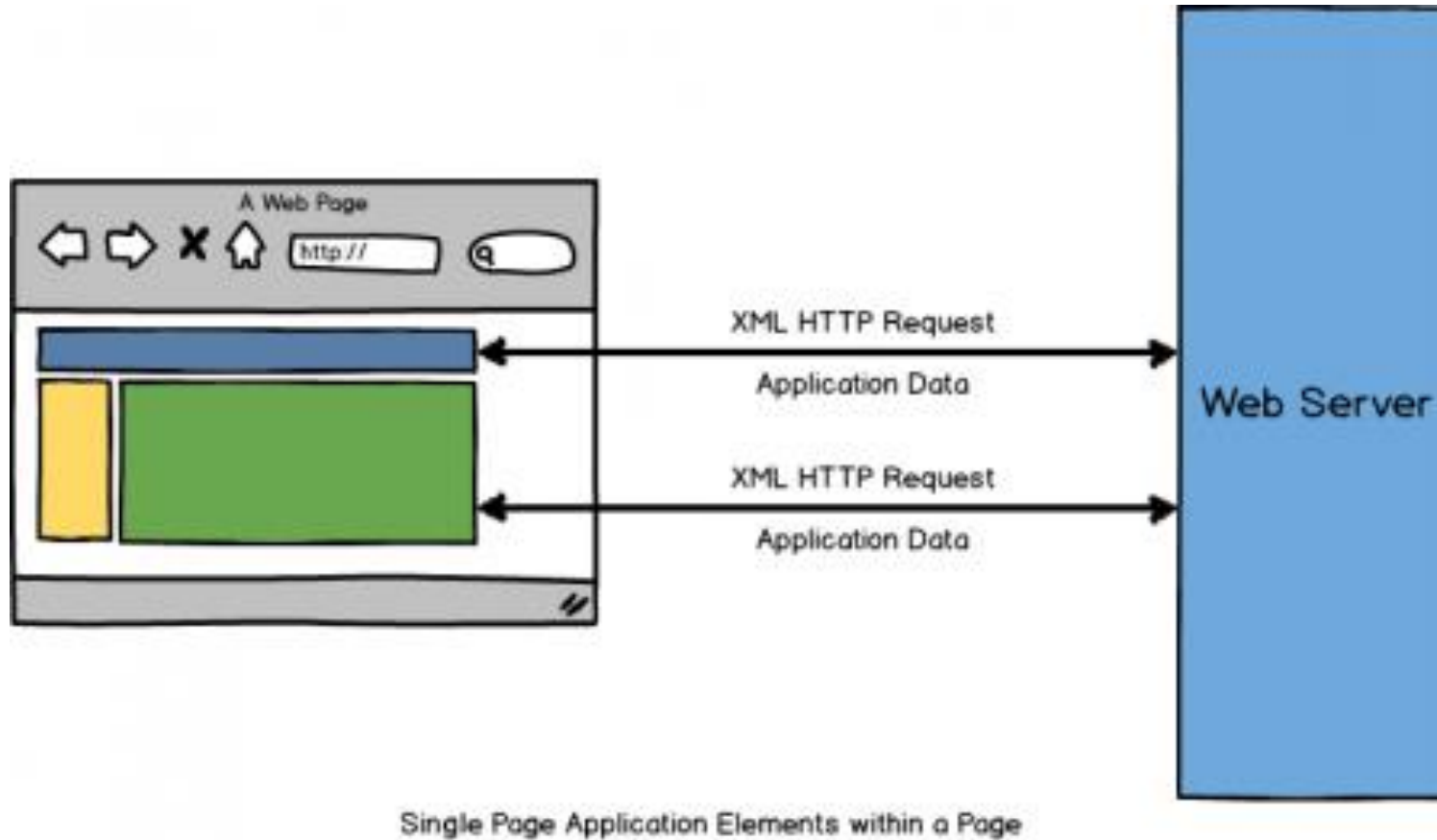
Traditional Full-Page Postback Operation

Многостраничное приложение с асинхронной загрузкой данных и частичным обновлением (MPA + AJAX)



AJAX XMLHttpRequest Partial Rendering

Одностраничное приложение (SPA, Single Page Application)



SPA = MVC + AJAX/JSON + REST

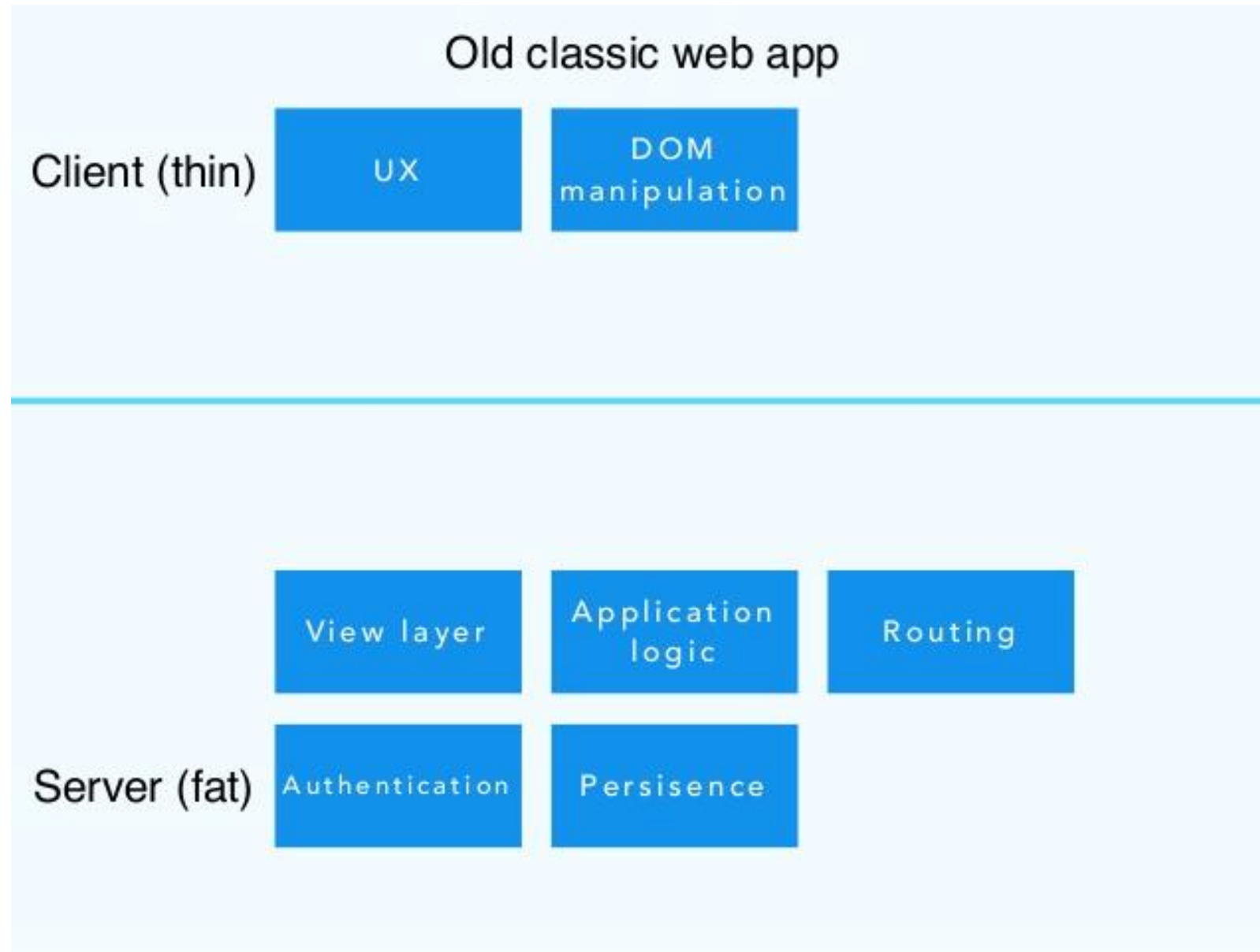
Достоинства	Недостатки
Быстрая загрузка/обновление страниц	Тяжелые фреймворки
Удобство пользователя	Интерпретируемые языки
Изолирование фронтенда и бэкенда	SEO

Толстый и тонкий

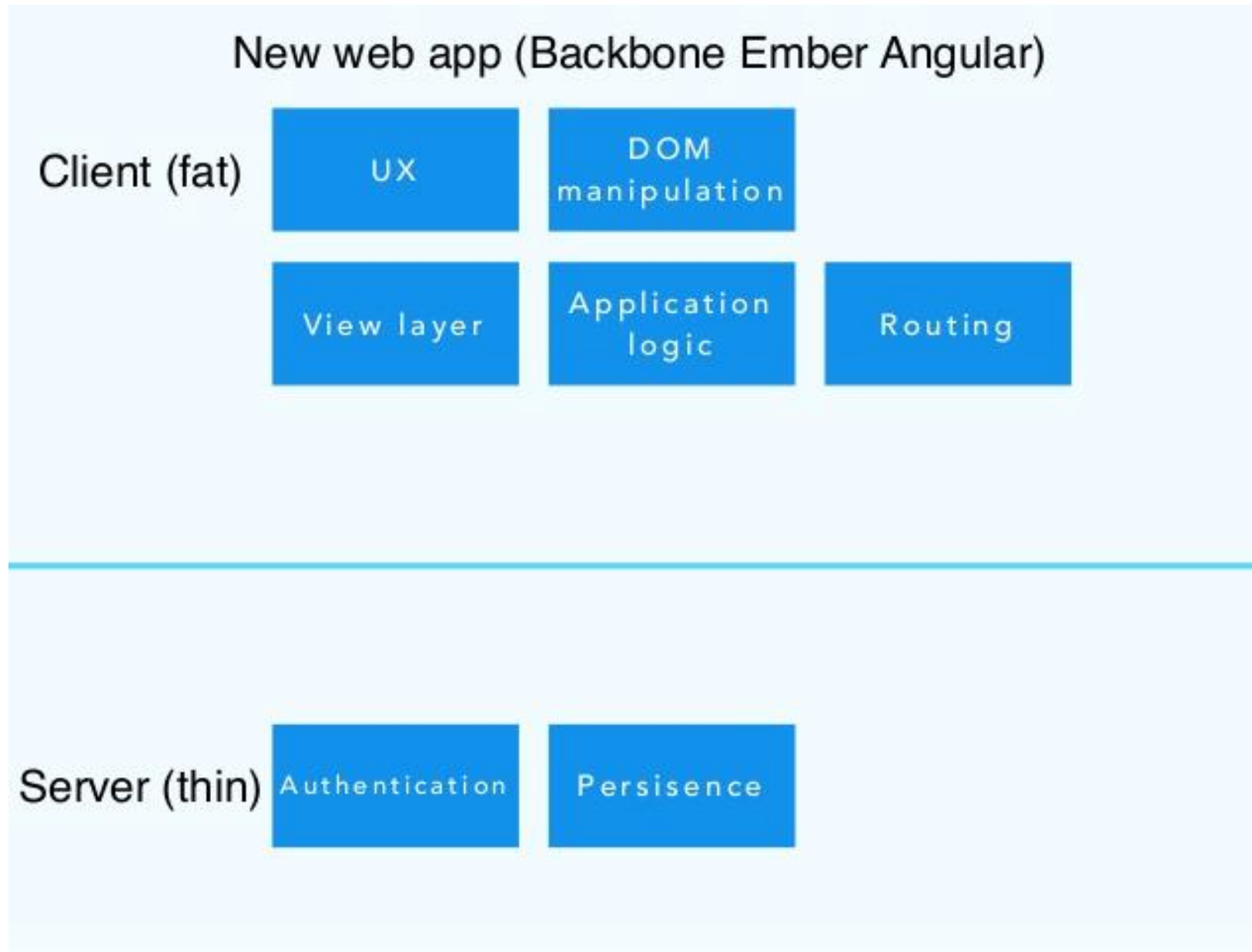


клиент

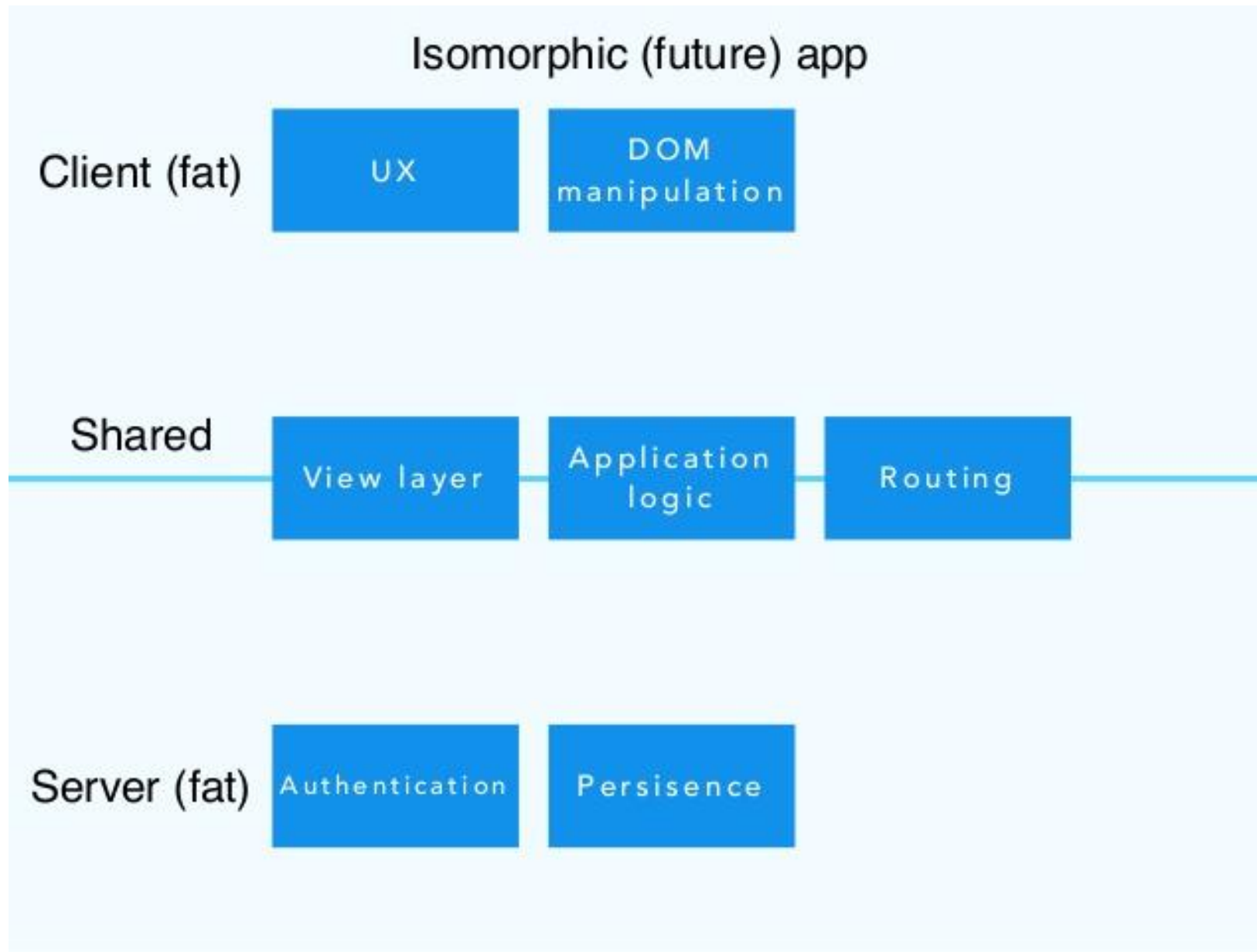
Тонкий клиент (Классический МРА)



Толстый клиент (RIA: Rich Internet Application)



Изоморфное приложение



Изоморфное приложение

- Единый код клиент-сервер
- Объединение достоинств МРА и SPA:
 - Скорость загрузки МРА
 - SEO-возможности МРА
 - Гибкость и удобство SPA
 - Тестируемость
 - Поддержка

Выбор архитектуры: от задач

Там где возможно – МРА

- Скорость разработки
- Скорость работы
- Нет проблем с SEO
- Дешево

Сайты-визитки, персональные страницы, сайты компаний

Выбор архитектуры: от задач

Там где невозможно МРА - SPA

- Гибкая архитектура
- Максимальное удобство пользователя
- Простота поддержки и модификации

Любое веб-приложение, веб-интерфейс сервисов и т.д.

Выбор архитектуры: от задач

Там где невозможно SPA – изоморфное

- Одни достоинства
- Слишком модно и современно
- Дорого
- Необходимо любить JS

Любой сайт и приложение станет потрясающим, если будет изоморфным

Технологический стек разработки



Технологические стеки

Клиентский стек

Серверный стек

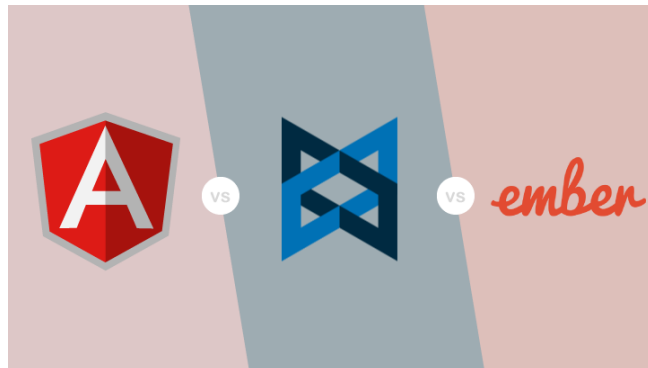
Фулл-стек

Клиентский стек

- Языки и платформы



- Библиотеки и фреймворки

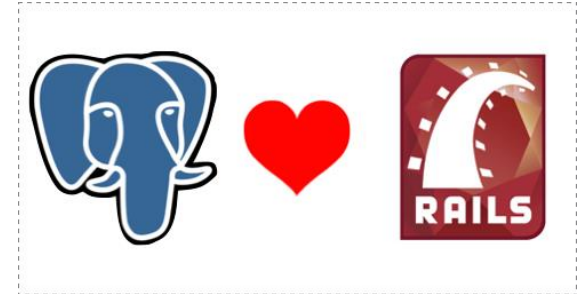


Серверный стек

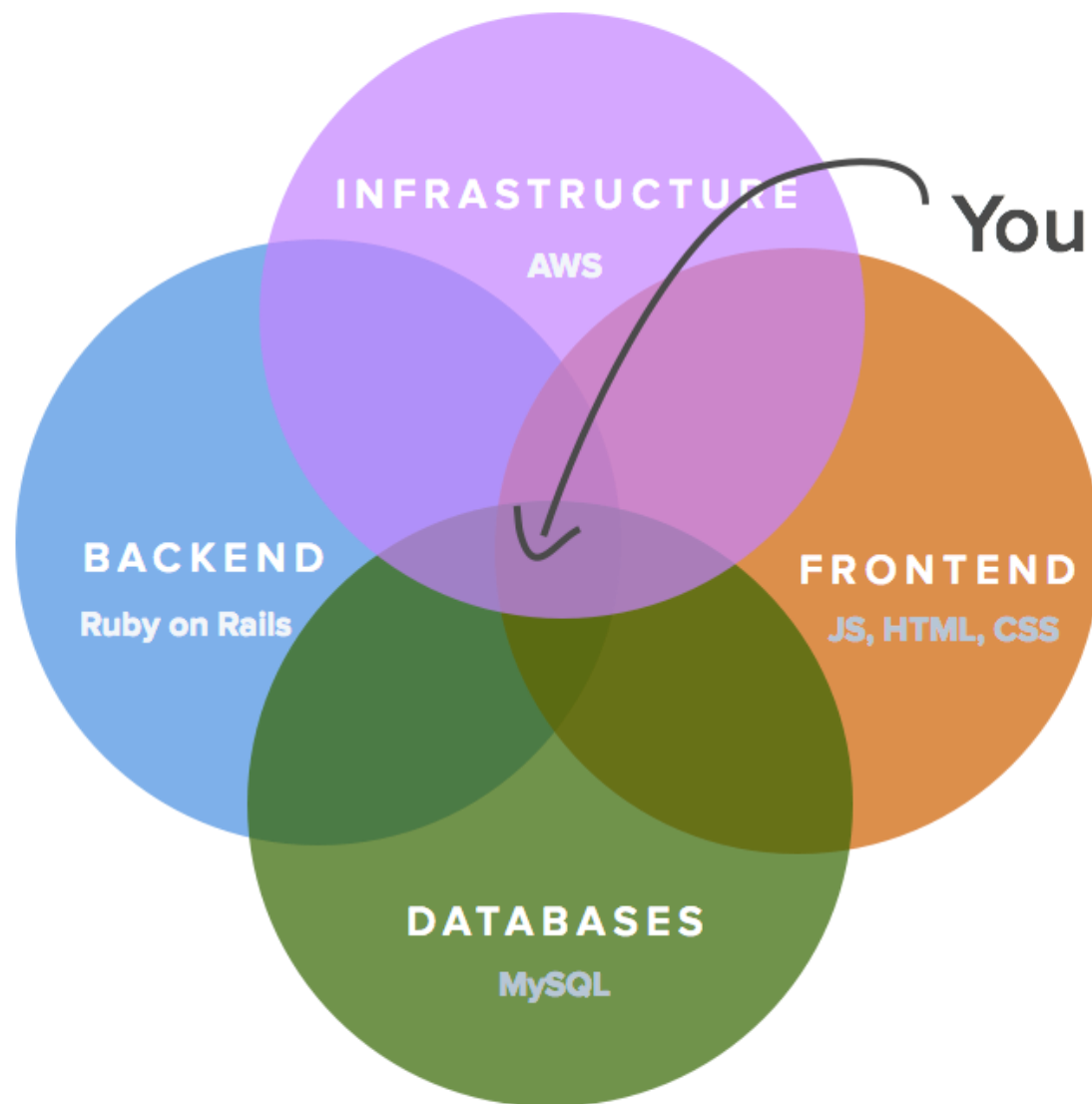
- Веб-сервер
- Сервер приложений
(языки и платформы)
- Библиотеки и фреймворки
- Сервер баз данных



ASP.NET



Фулл-стек

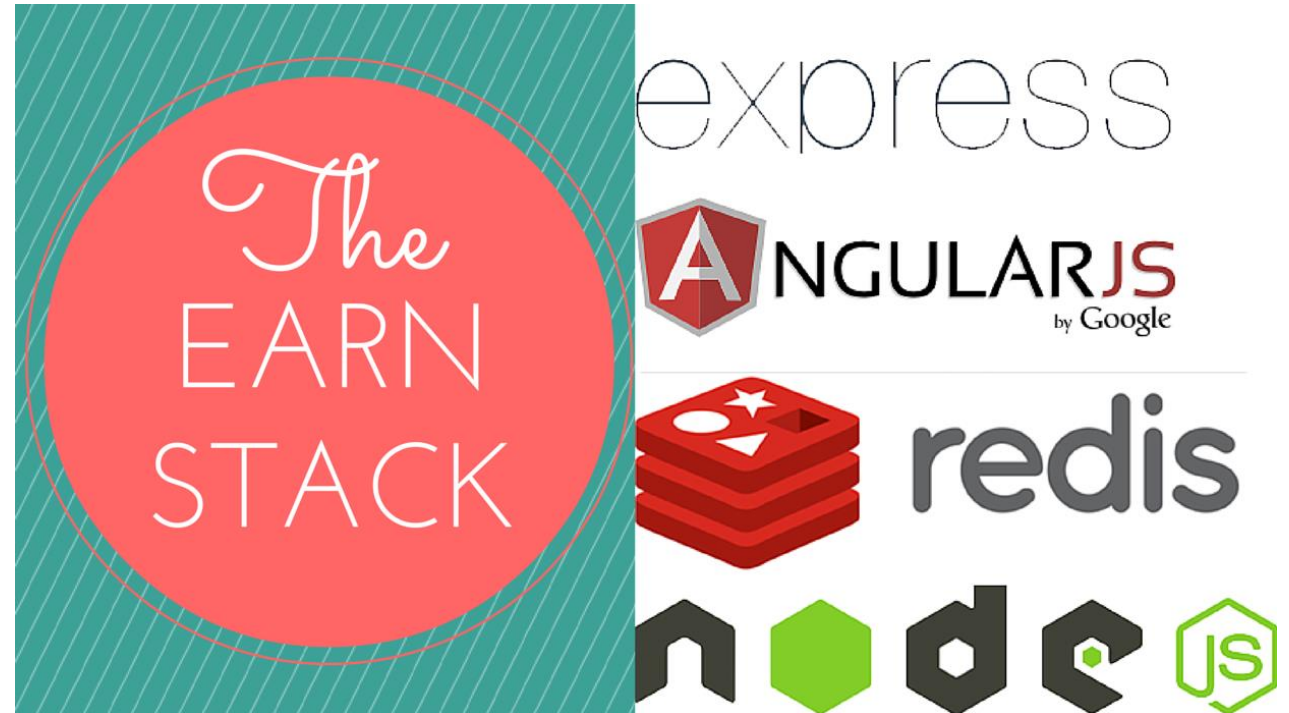
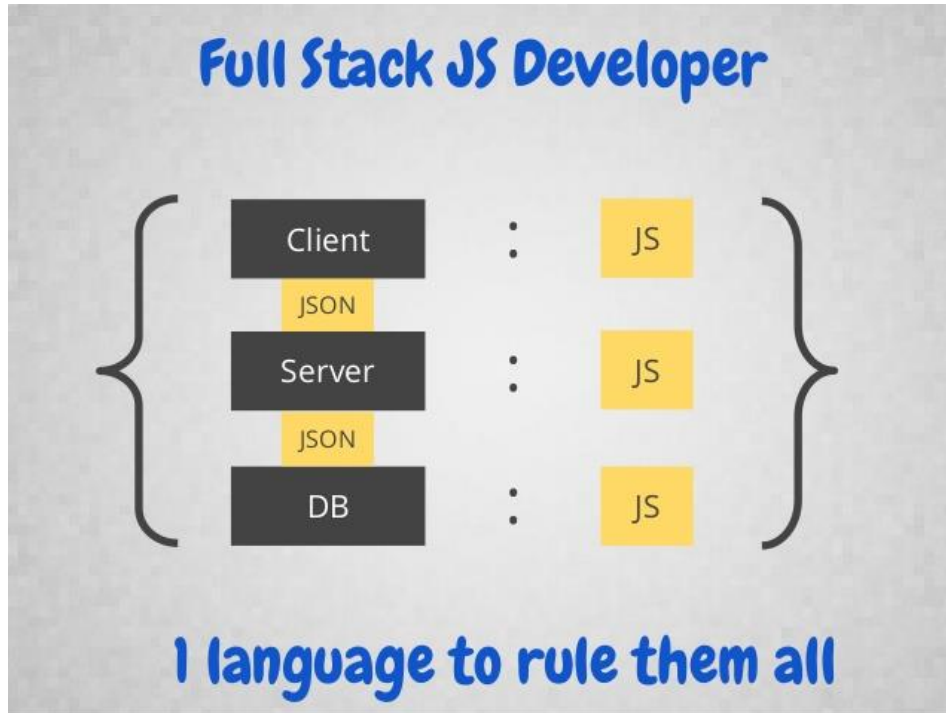


Полный стек

- ASP MVC – стек:
 - MS Sql Server
 - C# - ASP .Net MVC
 - JS+HTML+CSS
- Django – стек:
 - MySQL/PostgreSQL/...
 - Python – Django
 - JS+HTML+CSS

И т.д...

JS-FULL-Stack





ember



BACKBONE.JS

Knockout.




jQuery
write less, do more



ANGULARJS

node 

ACTIVE  JS

Выбор стека: от задачи

- В зависимости от выбранной архитектуры (MPA-SPA-...)
- От требований к скорости работы, скорости разработки, хранилищу данных, среде, ОС, языку и т.д.
- От личный предпочтений

Фреймворк! Решение всех проблем?

Фреймворк – каркас, структура

- Переиспользование
- Не надо писать велосипед
- Многослойность
- Хорошая, продуманная гибкая архитектура (по идее...)

Главное правило разработки

Использовать готовые решения **всегда**, когда только это возможно



MVC-фреймворки сервера

- Ядро, загрузчик, библиотека функций и конфигурация.
- Главная задача: реализация функционала **любого** веб-приложения.
- Любые веб-приложения – MPA/SPA

Типовая структура



Компоненты фреймворка

- Маршрутизатор
- Шаблонизатор
- Обработчик форм
- ORM
- Поддержка HTTP
- Разнообразные утилиты

MVC-фреймворки

- ASP.NET MVC
- Django
- Ruby on Rails
- Symfony
- Express
-

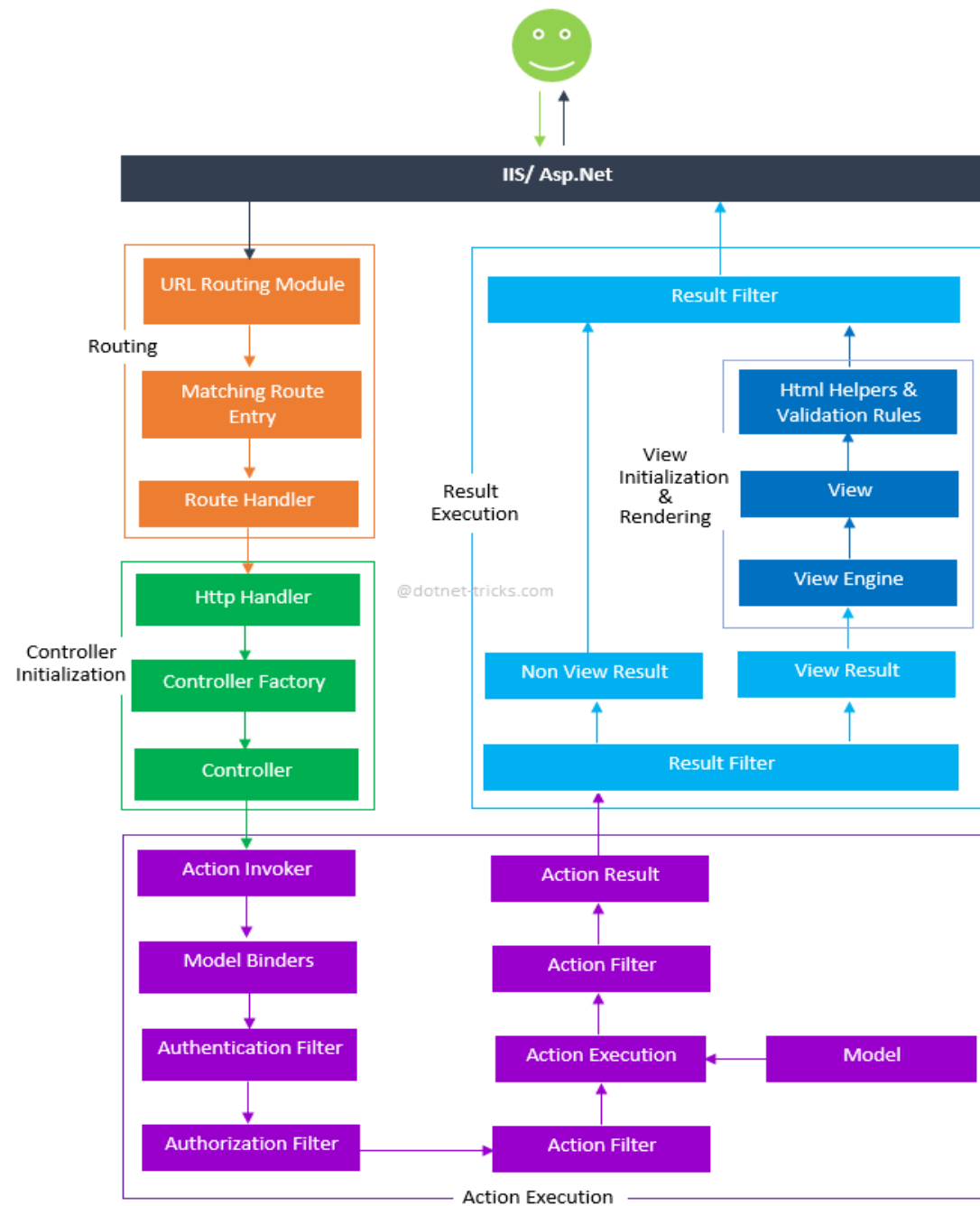
Мир C#: ASP.NET MVC

- Появился в 2009
- С 2012 – Apache License
- На данный момент – кросс-платформенный
- Последняя версия ASP .NET MVC 6 (ASP .Net Core)



Мир C#: ASP.NET MVC

- MVC
- Маршрутизация
- Шаблонизация (Razor)
- Скафолдинг (автогенерация контроллеров по модели)
- C#/VB
- ORM для любых БД
- Аутентификация
- Поддержка REST



ASP.NET MVC Pipeline

Настройка маршрутизации

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.MapMvcAttributeRoutes();

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller="Home", action="Index", id=UrlParameter.Optional }
        );
    }
}
```

Пример контроллера с маршрутизацией

```
public class BooksController : Controller
{
    [Route("books/{isbn?}")]
    public ActionResult View(string isbn)
    {
        if (!String.IsNullOrEmpty(isbn))
        {
            return View("OneBook", GetBook(isbn));
        }
        return View("AllBooks", GetBooks());
    }

    [Route("books/lang/{lang=en}")]
    public ActionResult ViewByLanguage(string lang)
    {
        return View("OneBook", GetBooksByLanguage(lang));
    }
}
```

Пример использования шаблонизатора Razor

```
@{  
    var txt = "";  
    if(DateTime.Now.Hour > 12)  
        txt = "Good Evening";  
    else  
        txt = "Good Morning";  
}  
<html>  
    <body>  
        <p>The message is @txt</p>  
    </body>  
</html>
```

Мир Python: Django

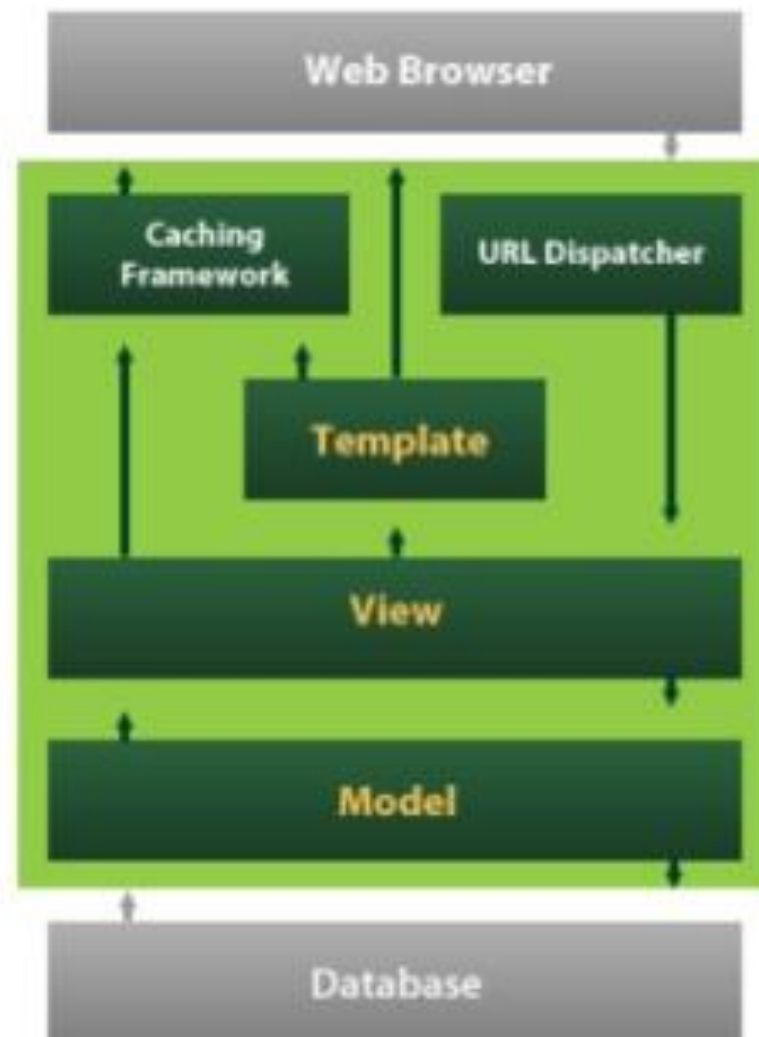
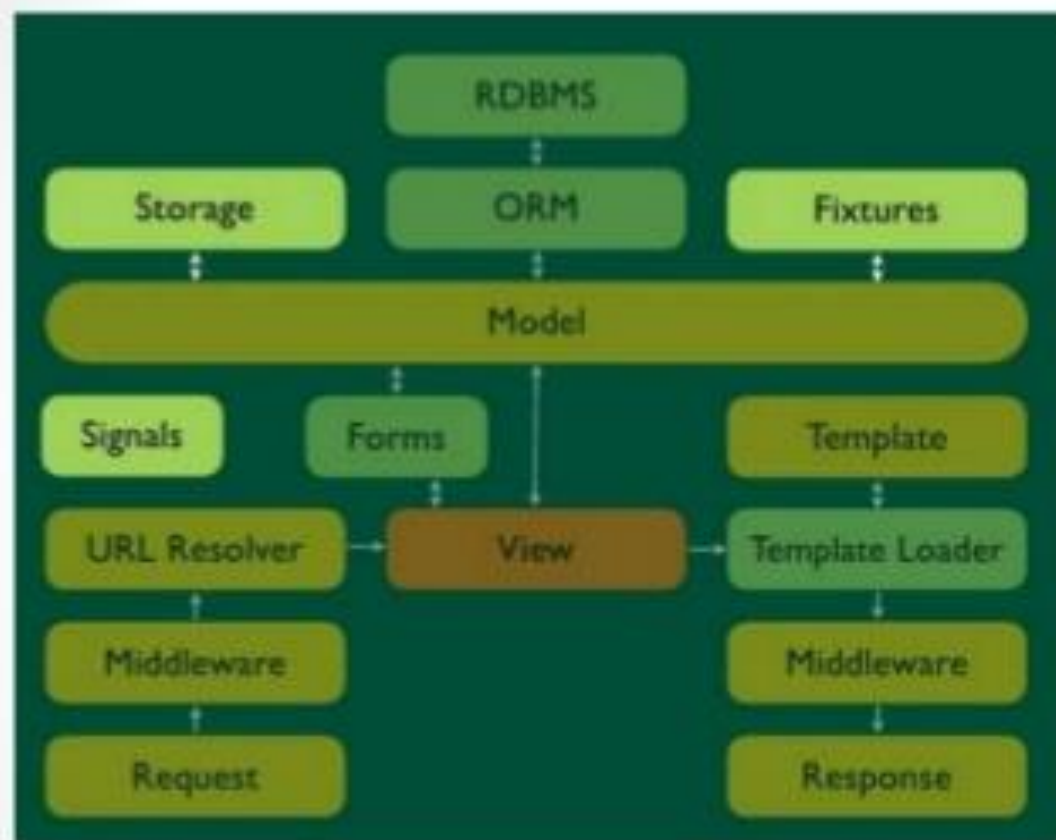
- 2003
- Кросс-платформенный
- Язык python
- Лицензия BSD
- Назван в честь музыканта Джанго Рейнхардта



Мир Python: Django

- MTV(вариант MVC, похожий на MPV)
- Маршрутизация с помощью регулярных выражений
- Шаблонизация
- Автогенерация админки
- Python
- ORM для большинства БД
- Аутентификация
- REST

Architecture



URL routing

```
from django.conf.urls import url
from . import views
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
    url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
]
```

```
def detail(request, question_id):
    return HttpResponse("Question %s." % question_id)

def results(request, question_id): response = «Results of question %s.»
    return HttpResponse(response % question_id)
```

View

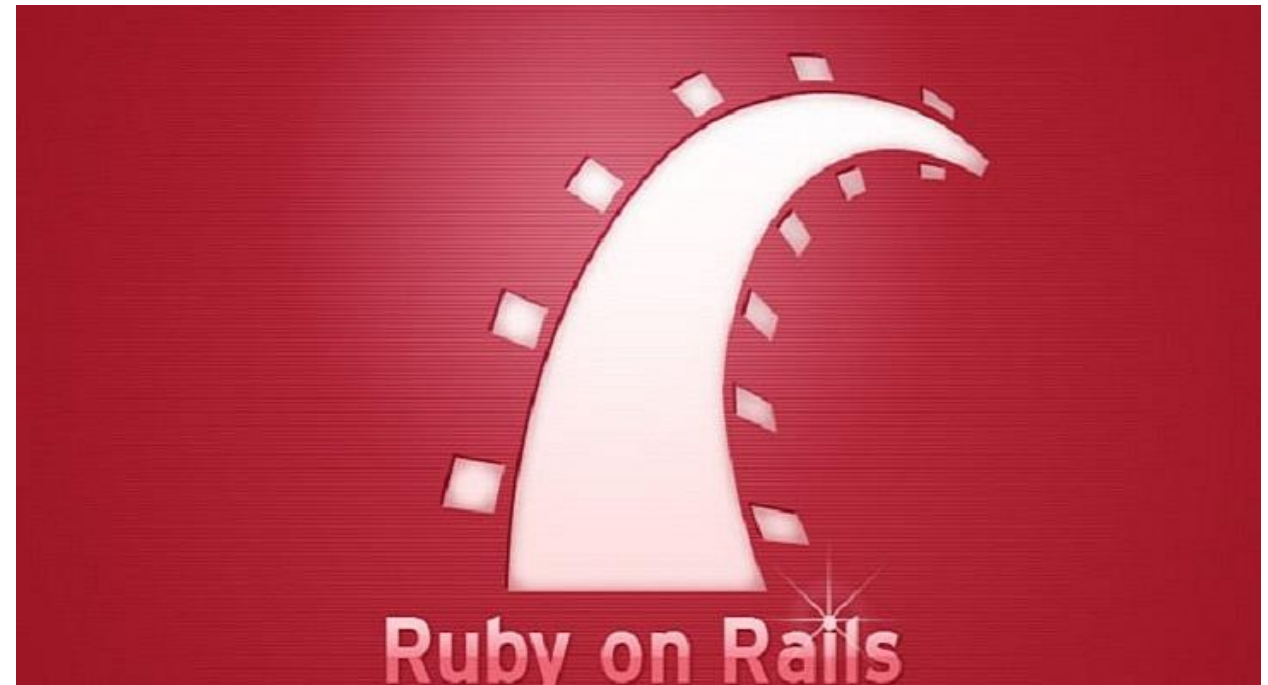
```
from django.http import HttpResponse
from django.template import RequestContext, loader
from .models import Question
def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    template = loader.get_template('polls/index.html')
    context = RequestContext(request, { 'latest_question_list': latest_question_list, })
    return HttpResponse(template.render(context))
```


Шаблонизация

```
{% if latest_question_list %}
<ul>
    {% for question in latest_question_list %}
    <li>
        <a href="/polls/{{ question.id }}/">{{ question.question_text }}</a>
    </li>
    {% endfor %}
</ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Мир Ruby: Ruby on Rails

- 2004
- Кросс-платформенный
- Язык Ruby
- Лицензия MIT

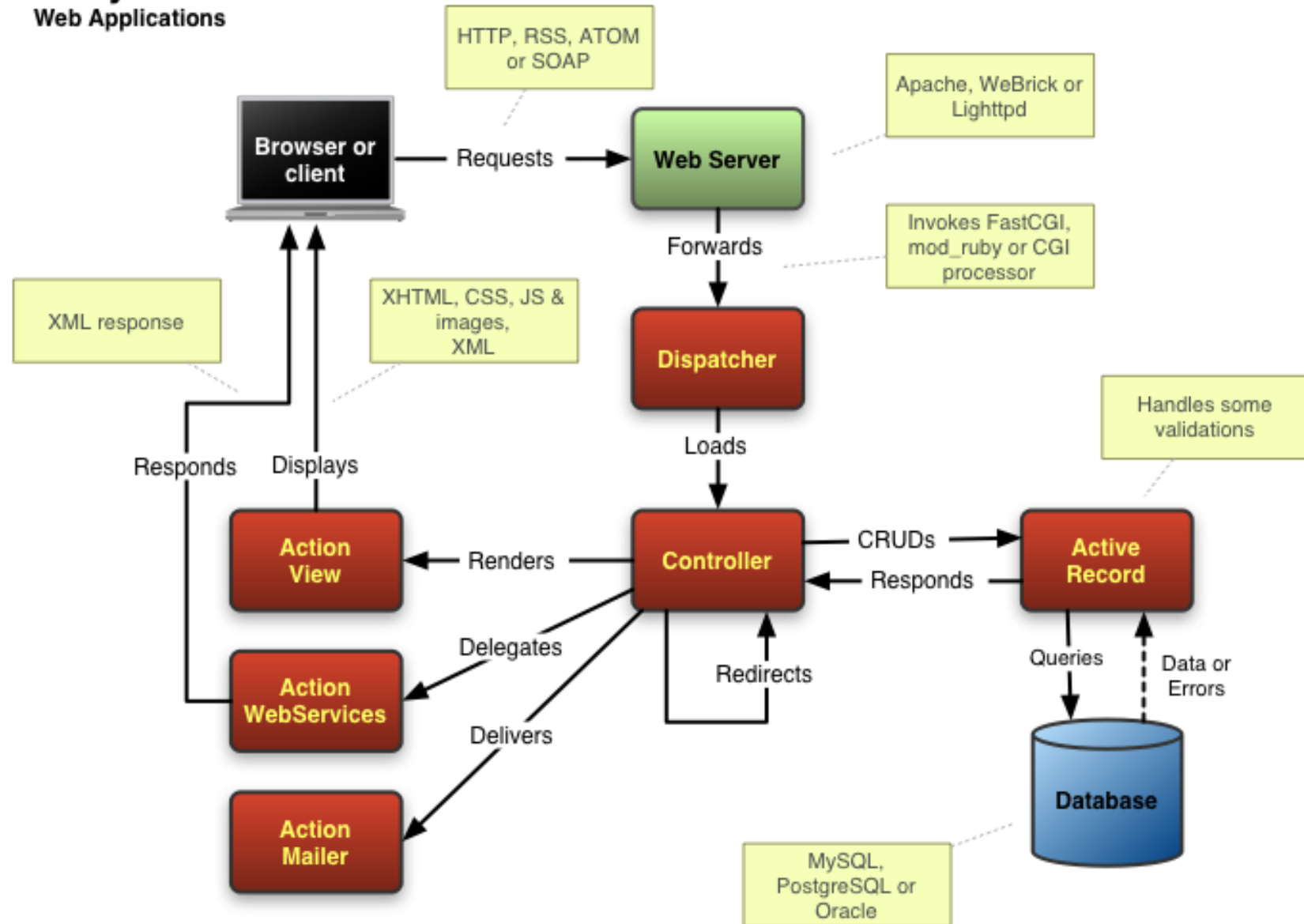


Мир Ruby: Ruby on Rails

- MVC
- Маршрутизация
- Шаблонизация
- Автогенерация админки
- Ruby
- ORM для большинства БД
- Аутентификация
- REST

Ruby on Rails

Web Applications



Пример маршрутизации

get *'/clients/:status' => 'clients#index', foo: 'bar'*

Пример контроллера

```
class ClientsController < ApplicationController

  # GET/clients?status=activated
  def index
    if params[:status] == "activated"
      @clients = Client.activated
    else
      @clients = Client.inactivated
    end
  end

  # POST/clients
  def create
    @client = Client.new(params[:client])
    if @client.save
      redirect_to @client
    else
      render "new"
    end
  end
end
```

Шаблонизация

```
<%= form_for :article do |f| %>
  <p>
    <%= f.label :title %><br>
    <%= f.text_field :title %>
  </p>

  <p>
    <%= f.label :text %><br>
    <%= f.text_area :text %>
  </p>

  <p>
    <%= f.submit %>
  </p>
<% end %>
```



PHP: минутка ненависти

PHP был разработан с целью максимально легкого обучения и использования не-программистами.

Не был спроектирован одним человеком либо небольшой группой специалистов. Расмус Лерддорф, автор PHP, принимал любые добавления в код от сторонних авторов.

«Я не знаю ни одного программиста, который бы продолжил развиваться как программист, перейдя на пхп. Даже профессиональные программисты, переключаясь на пхп, со временем начинают деградировать»

*«Фактически каждая деталь PHP в какой-то мере поломана. Язык, структура, экосистема: всё **плохо**.»*

«PHP — это комьюнити любителей. Очень мало людей, которые его создают, работают над ним или пишут код на нём, вообще представляют, что они делают»

Мир PHP

- Полное отсутствие стиля (даже в именовании)
- Не общепринятые названия и поведение функций
- Неоднозначное и неявное поведение всего
- Умалчивание критических ситуаций
- Дизайн не имеет определённой философии. Ранний PHP был вдохновлён Perl'ом; огромная std-библиотека из C; ОО-часть сделана как в C++ и Java.
- Нет пространств имен, много глобально видимых функций и классов
- PHP поощряет смешение разметки страниц и логики приложения
- Слабая типизация

MVC-PHP-фреймворки

- **Laravel**
 - **Yii**
 - **Symfony**
-
- MVC
 - ORM
 - REST
 - Шаблонизация
 - Тестирование



Мир Java: Grails и Spring

Grails:

- На основе Ruby on Rails
- Использует язык Groovy (Java + Python, Ruby, Smalltalk)

Spring:

- **Spring MVC** – часть большого фреймворка для бизнес-приложений



Мир JS: NodeJS + Express + Mongoose

- NodeJS – платформа web-приложений
- Пример MVC на NodeJS:
 - Express для маршрутизации/ контроллеров
 - Mongoose – ORM для модели (для СУБД MongoDB)
 - EJS для шаблонизации представлений



Технологии тонкого клиента

- HTML
- CSS
- JS
- библиотека jQuery

Технологии RIA-клиента

- JS + HTML + CSS
- Flash (ActionScript)
- .Net Silverlight
- Java applet
- ActiveX
- Другие сторонние плагины и расширения

Технологии RIA-клиента

- JS + HTML + CSS
- ~~Flash (ActionScript)~~
- ~~.Net Silverlight~~
- ~~Java applet~~
- ~~ActiveX~~
- ~~Другие сторонние плагины и расширения~~

Технологии RIA-клиента

- HTML/CSS-фреймворки
- JS-фреймворки

HTML/CSS фреймворки

- Библиотеки пользовательских элементов управления
- Библиотеки стилей



jQuery



- Переход по дереву DOM, включая поддержку [XPath](#) как плагина
- События
- Визуальные эффекты
- AJAX-дополнения
- JavaScript-плагины

AngularJS



MVVM-фреймворк от Google, 2009

Angular 1.0:

- сервисы, фабрики, фильтры (model)
- контроллеры (~view model)
- директивы (компоненты view)
- binding шаблонов (view) к областям видимости контроллеров (view model)
- binding реализован через dirty checking
- Маршрутизация
- Разрешение зависимостей

BackboneJS



MVP-фреймворк от создателя CoffeeScript, 2010

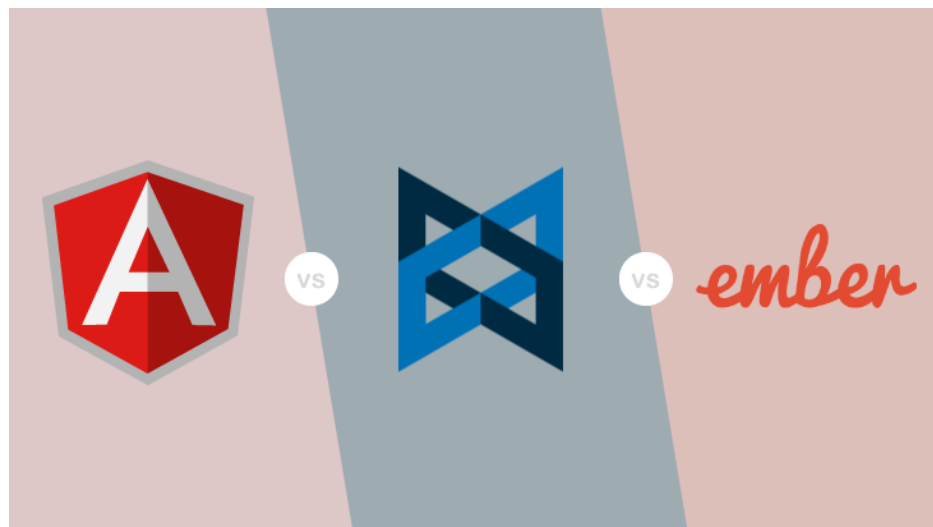
- Маршрутизация
- Контроллеры
- Модели
- Представления (вместе с логикой отображений и реакции на события)
- Нет шаблонизатора (для этого – Underscore.js)
- Нет архитектуры приложения
- Есть мнение, что это не фреймворк

EmberJS

MVC-фреймворк, 2007

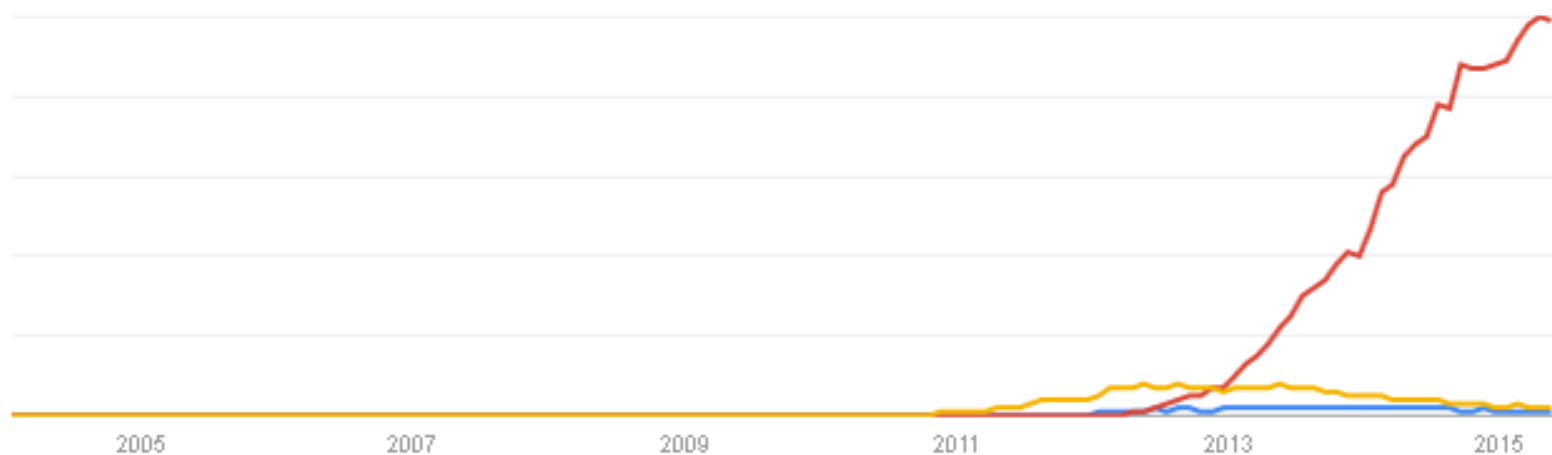
- Маршрутизация
- Модель
- Контроллеры
- Шаблонизация (HTMLBars)
- Готовый модуль работы с данными





Interest over time. Web Search. Worldwide, 2004 - present.

■ ember.js ■ angularjs ■ backbone.js



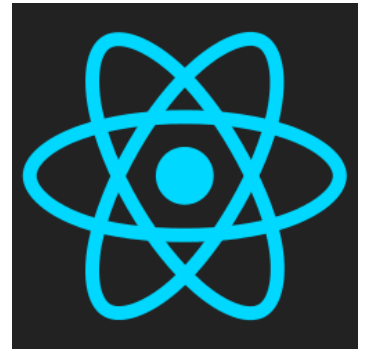
Knockout

MVVM-библиотека

- View model
- Шаблонизация
- Отслеживание зависимостей
- binding



React



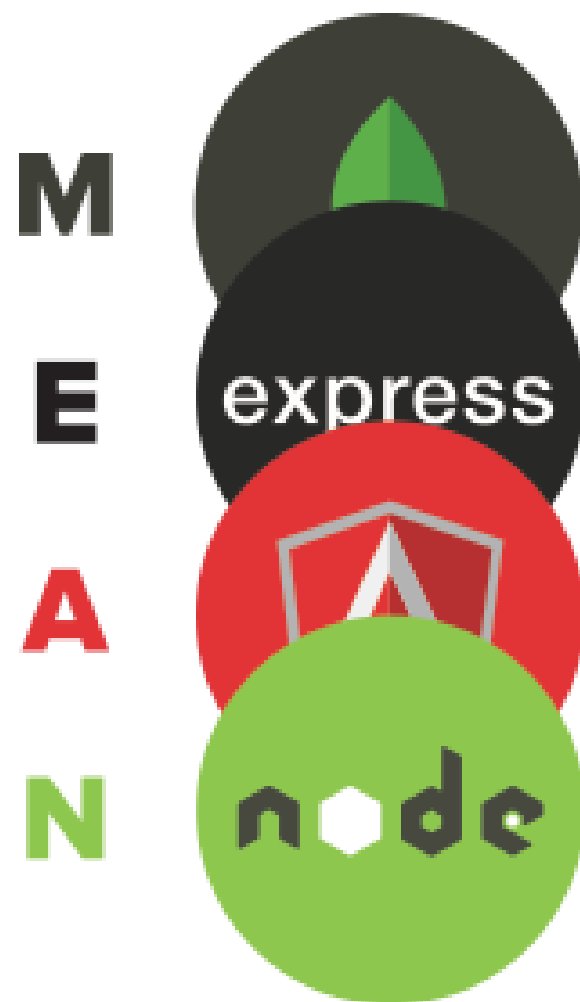
Библиотека, специализирующаяся на View для MVC – фреймворков

- Строит виртуальный DOM
- Элементы и компоненты
- Используется в изоморфных приложениях

Meteor

- Фуллстек платформа для приложений «реального» времени
- Для связи с браузерами используется Distributed Data Protocol (на AJAX или WebSocket-ax)
- Позволяет строить изоморфные приложения
 - Передаются только данные
 - БД доступна отовсюду
 - Компенсация задержек

JS-фуллстек



JS

- В 1995, Брендон Эйк написал за неделю LiveScript для управления Netscape Navigator
- С 2010-х на нем разрабатываются фулл-стек приложения (база данных-сервер-клиент)



Преимущества JS на клиенте

Это единственный язык программирования, который нативно работает во всех браузерах

Преимущества JS на сервере

- Единый язык
- Единые библиотеки
- Единый код для моделей
- Возможность построить изоморфное приложение

Преимущества JS в базе данных

- Единый формат данных в приложении (JSON)
- Единая кодовая база (хранимые процедуры на JS)
- Один язык везде (зачем нужен SQL?)