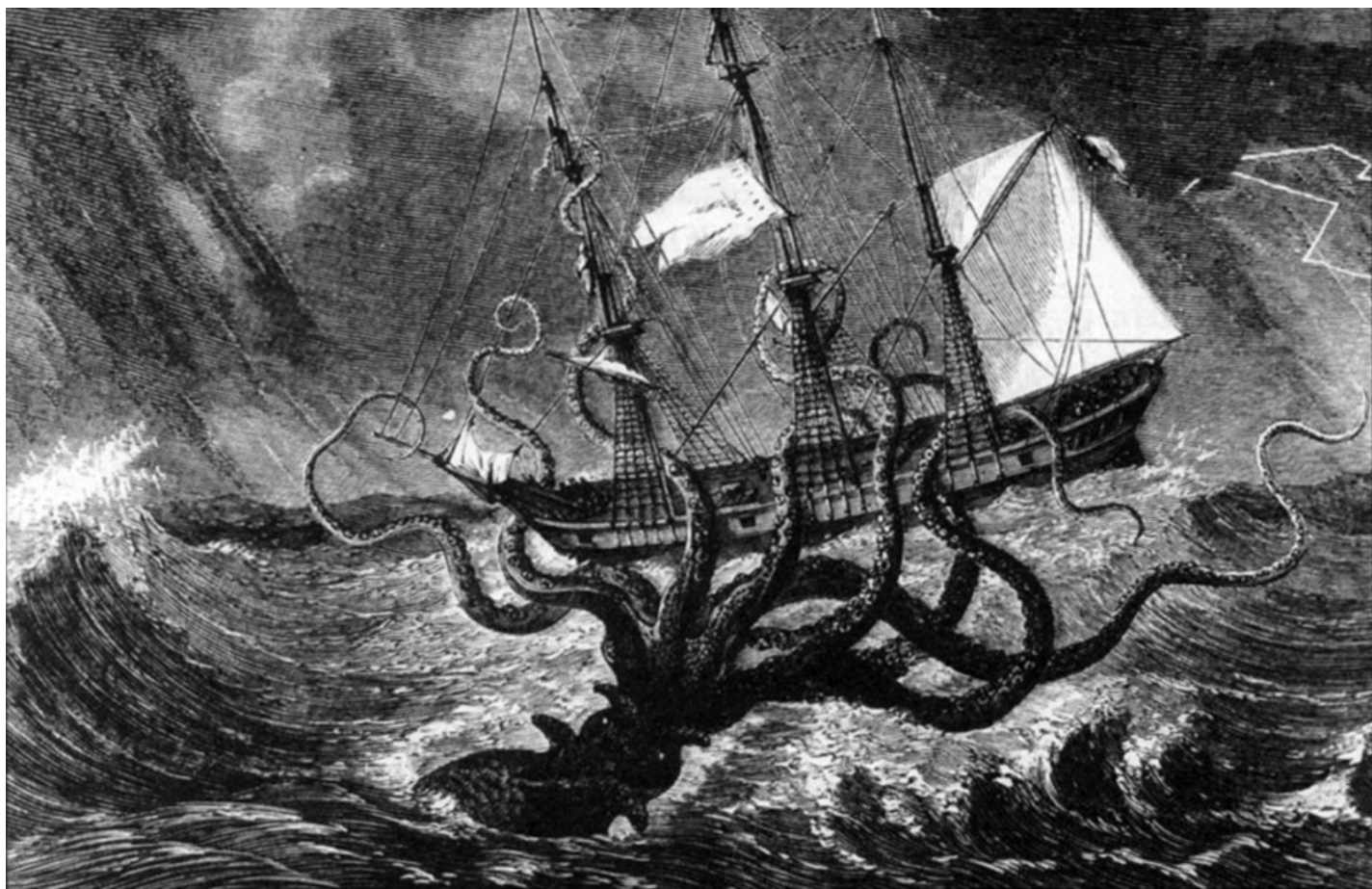


REACTJS*, JAVASCRIPT*, JAVA*, IT-СТАНДАРТЫ*, ANGULAR*

Все новое — это хорошо забытое старое

ИЗ ПЕСОЧНИЦЫ

archik 10 апреля 2017 в 11:23 👁 16k



Mary Evans Picture Library/Alamy—Ellis, R. 1994. Monsters of the Sea. Robert Hale Ltd.

Вереница фреймворков и библиотек по очереди восседающих на троне трендов JavaScript мира это уже не новость. Разработчики из других областей даже подшучивают над нами на этот счет.

Вот и мне, в процессе работы, пришлось попрыгать по различным

библиотекам и фреймворкам — qooxdoo, jQuery, Ext JS, Backbone.js, Knockout.js, Ember.js, Angular, React.

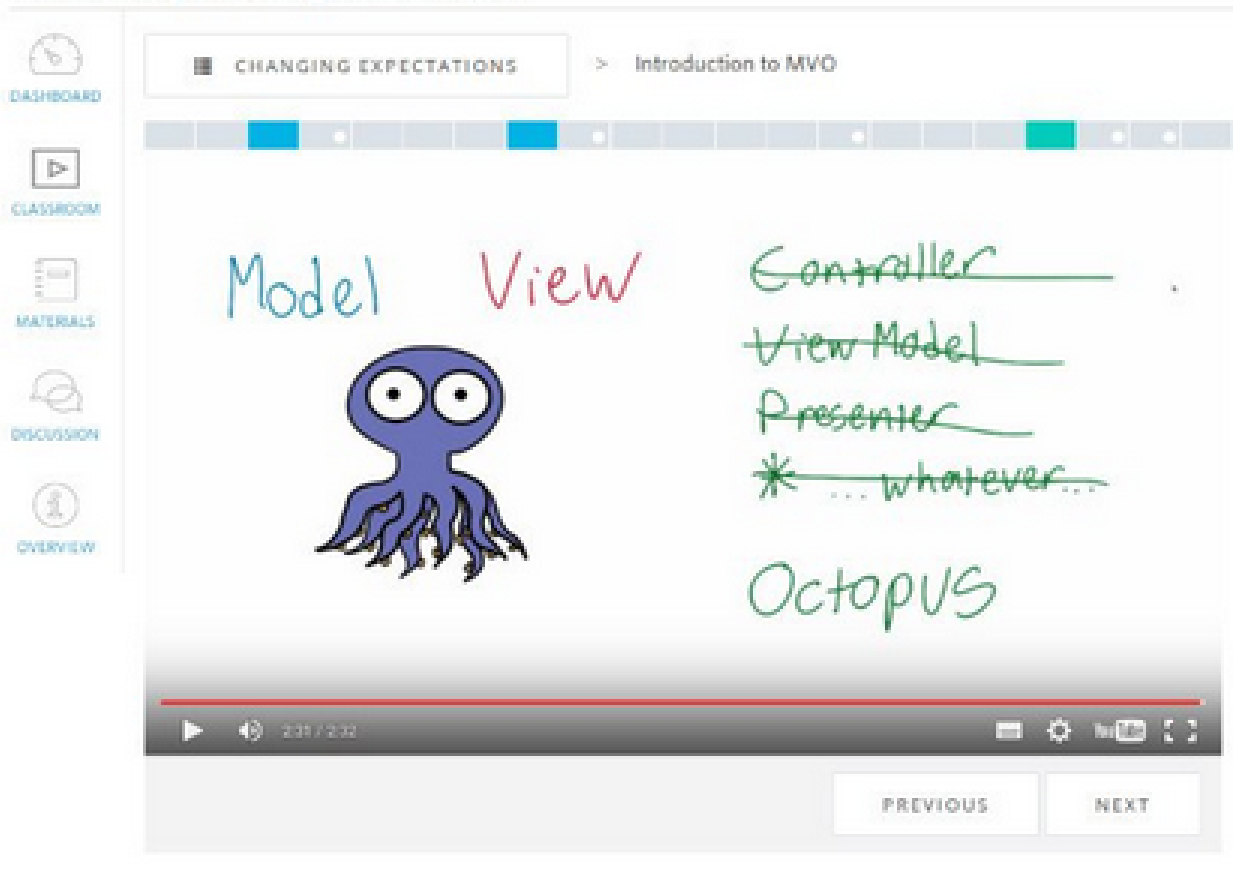
Не всегда выбор того или иного фреймворка был добровольный, модель outsource и outstaffing накладывает определенные ограничения на мою работу. Я думаю люди из этой же области поймут меня.

Последнее на чем я остановился это **React** от Facebook. Не буду скрывать, коллеги подтвердят, я с большой неохотой переходил на эту библиотеку, переход означал выход из зоны комфорта, а это то что мы так не любим. Пугало много новых слов, в частности «новое» переосмысление архитектурного стиля в лице Flux и Redux, «новые» термины — Actions, Store, Dispatcher и т.д.

Согласитесь, со всеми предыдущими «молотками и гвоздями» словарь сводился обычно к чему-то вроде **Model-View-*** (Controller || Presenter || Adapter || ViewModel || Whatever)

Мое понимание этого словаря выглядело точь-в-точь как на этой картинке:

JavaScript Design Patterns



Это картинка из курса “JavaScript Design Patterns” одной довольно популярной MOOC-платформы Udacity

Мне понравилась эта картинка. Так проще было воспринимать мир, населенный кучей клонов и различных интерпретаций MVC.

Есть View — это html, css и все что с ними. Есть Model — это JSON-данные и то место, откуда мы их вытянули. И есть Кракен / осьминог / Многоножка / Многоручка — некое мифическое существо объединяющее данные и представления.

MVO — вот ответ на все вопросы.

Classroom - Udacity - Mozilla Firefox

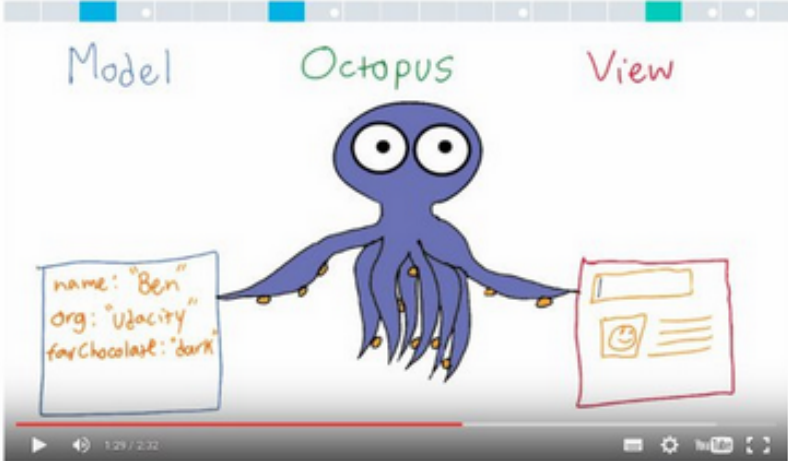
Мой д... M101J... three.js / ... three.js - ... Статус... Slook... Входя... KNCit... media... Dashb... http... 1288 Angular G... Googl... Home

https://www.udacity.com/course/viewer#/c-ud989/l-3417188540/m-3374098566

Полиск

CHANGING EXPECTATIONS > Introduction to MVO

Model Octopus View



name: "Ben"
org: "Udacity"
favChocolate: "dark"

1:29 / 2:32

PREVIOUS NEXT

Instructor Notes
No additional notes for this section

Downloadables
LESSON

Get Help
[Discussion Topics](#)

Recommended based on your courses

U

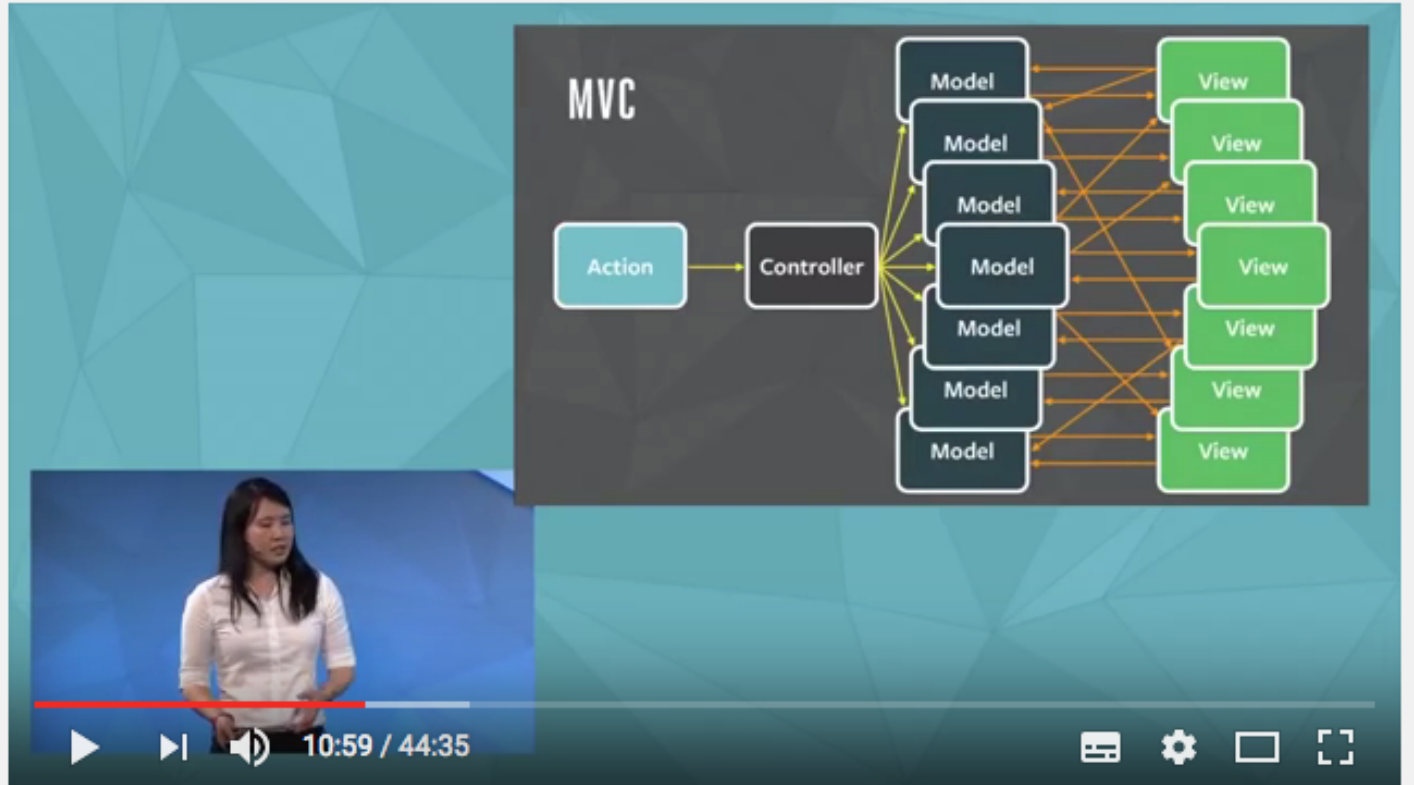
Front-end Web Developer
NANODEGREE PROGRAM

- 1:1 coaching
- Detailed code reviews
- Personalized career support

Nanodegree programs confer industry built-and-recognized credentials, and Udacity pays half your tuition when you graduate in 12 months.

START TODAY!

И тут внезапно приходит Facebook и говорит MVC это плохо, с НИМ можно вконец запутаться и потом не выбраться из этой сети зависимостей:

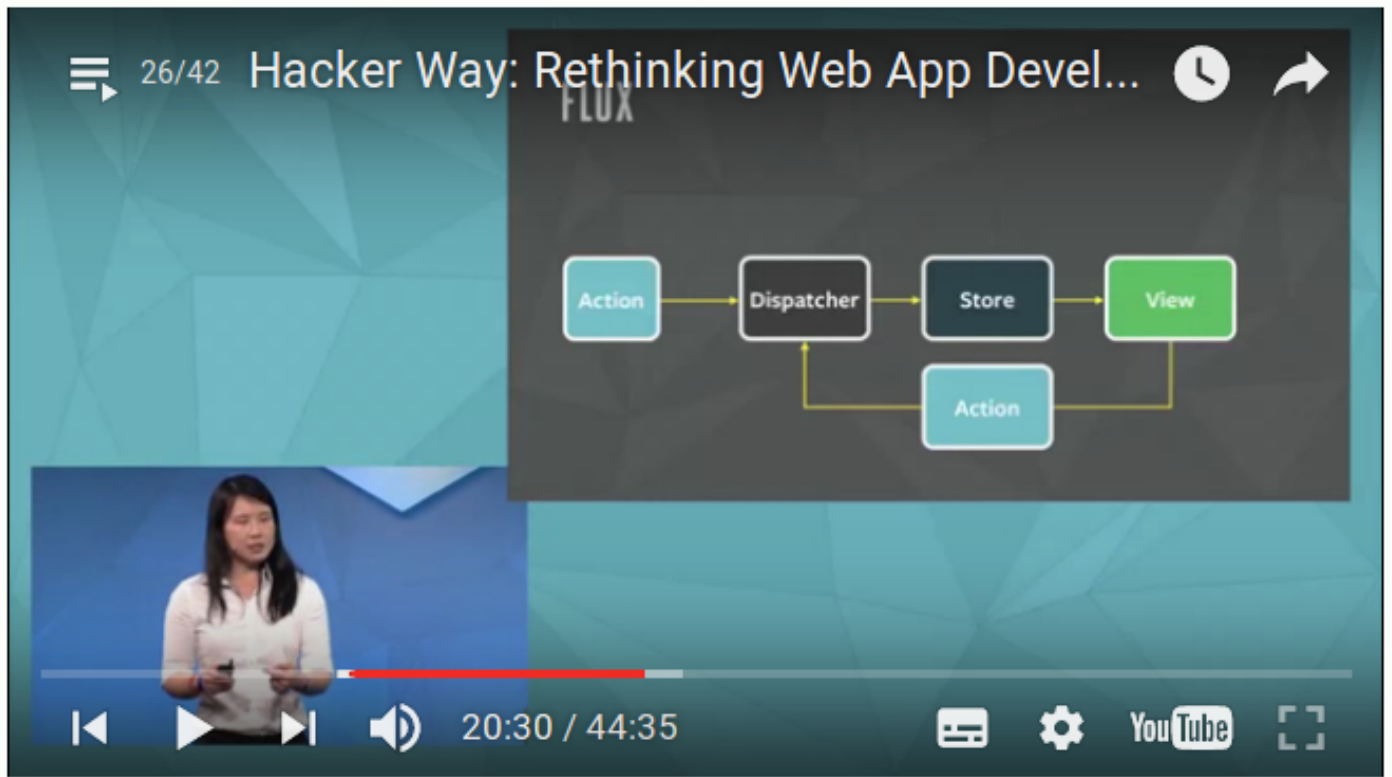


The image shows a video player interface. The main content area displays a diagram titled "MVC" (Model-View-Controller). The diagram illustrates the MVC pattern with an "Action" box on the left, a "Controller" box in the center, a vertical stack of six "Model" boxes in the middle, and a vertical stack of six "View" boxes on the right. Arrows indicate the flow of data: a yellow arrow points from "Action" to "Controller"; yellow arrows point from "Controller" to each "Model" box; and orange arrows point from each "Model" box to each "View" box. Below the diagram, a video player shows a woman speaking. The player controls include a play button, a progress bar at 10:59 / 44:35, and icons for settings, full screen, and a list.

Hacker Way: Rethinking Web App Development at Facebook

 Facebook Developers

А потом говорит, что есть серебряная пуля — **Flux** и **Unidirectional Dataflow**:



Дальше приходит кто-то еще и говорит Flux это сложно, Redux — вот что вам надо.

Ты склонен верить, ведь это Facebook, эти ребята знают толк, постепенно ты поддаешься течению.

Теперь все решается этим архитектурным стилем и библиотекой. Надо front-end? Ок, берем React, Redux и еще немного багажа в виде thunk, saga, storybook, themr, flow и дело в шляпе.

Нет, вы не подумайте, React — это прекрасная библиотека. Я в восторге от **Virtual DOM** технологии, она и вправду решает очень важную задачу. Привязка к реактивной парадигме программирования это тоже то, что меня радует. Но вот “новый” архитектурный стиль...

Наша отрасль стремительно развивается — это хорошо. В 2017

году программистов в сотни раз больше, чем в 1980 — это тоже хорошо. Появляется множество классных инструментов призванных облегчить жизнь прикладного программиста — это тоже определенно хорошо. Но и думать мы, прикладные программисты, начинаем меньше. По крайней мере мне так кажется. За 5 лет работы профессионального программиста я написал кучу посредственного и запутанного кода.

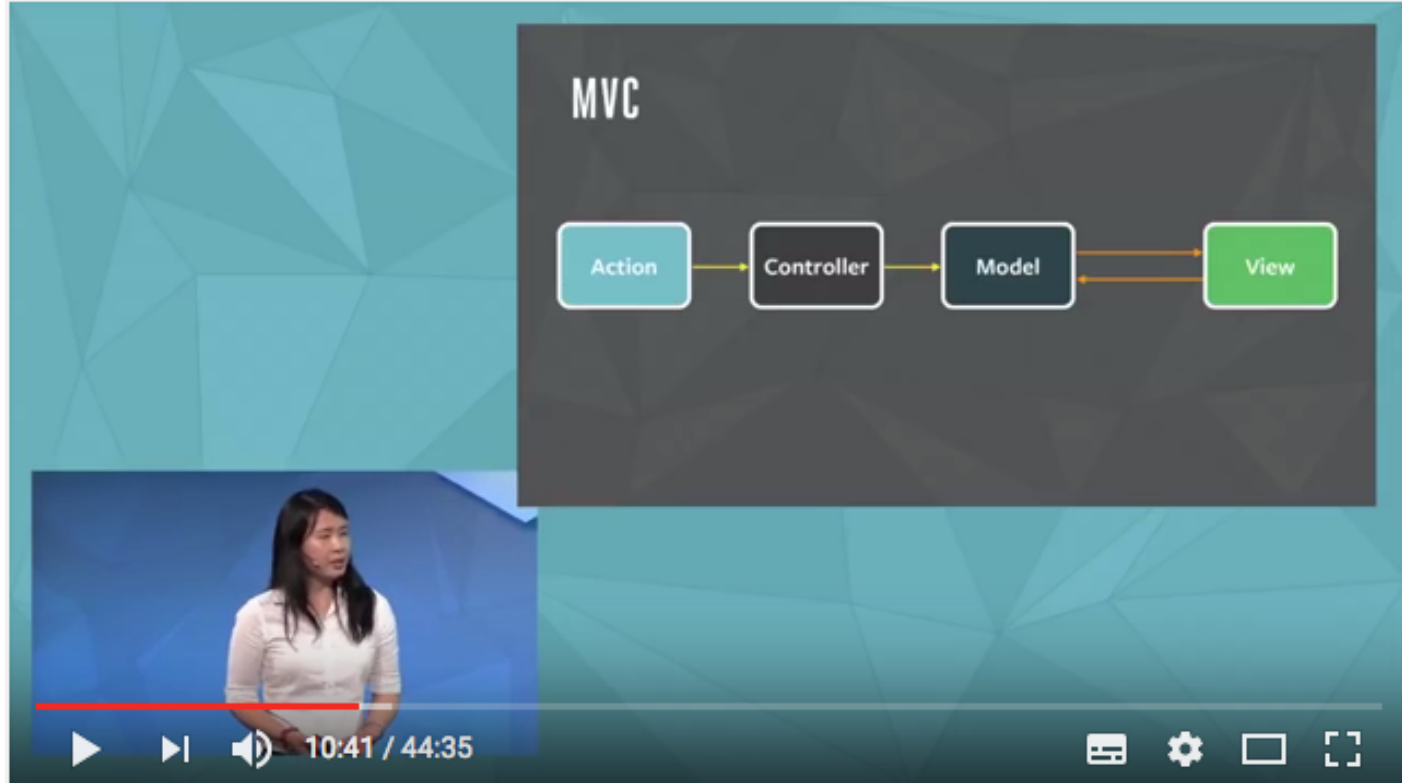
На минуту я решил остановиться. И перед тем как идти дальше, так же как и Facebook переосмыслить разработку web-приложений. Разобраться, почему Unidirectional Dataflow это то, что решает проблему этой паутины, почему до Facebook никто этого не делал и чем так плох MVC со своими стрелками во все стороны.

И начал я в обратном порядке.

MVC

Начал я с того, что пошел посмотреть, а какое же определение этому архитектурному стилю дают другие топовые компании-разработчики. Ведь чаще всего мы так и делаем, идем посмотреть к кому-то компетентному, к кому-то у кого есть вес.

С интерпретацией от Facebook мы познакомились:



Hacker Way: Rethinking Web App Development at Facebook

Facebook Developers

Нижe интерпретация от Microsoft:

- **Model.** The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
- **View.** The view manages the display of information.
- **Controller.** The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

Figure 1 depicts the structural relationship between the three objects.

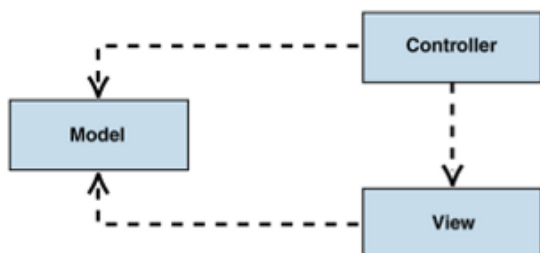


Figure 1: MVC class structure

Model и View не связаны в обратную сторону, хотя из описания ясно, что все таки модель должна ответить на запрос состояния от представления.

Т.е. в общих чертах интерпретация та же, что и у Facebook.

Идем в Apple, у этих ребят целых две схемки (традиционная и версия Cocoa API)

Figure 7-1 Traditional version of MVC as a compound pattern

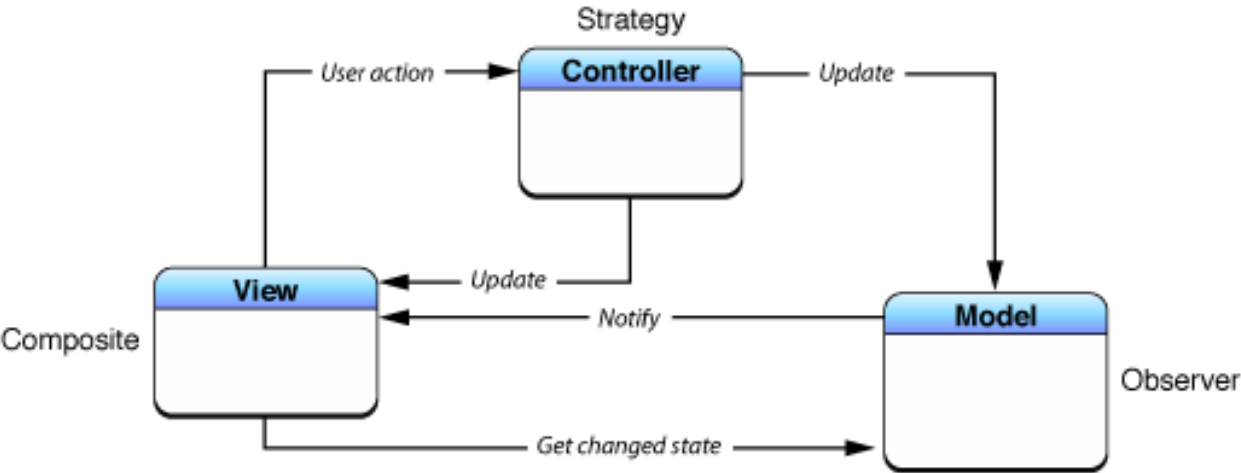
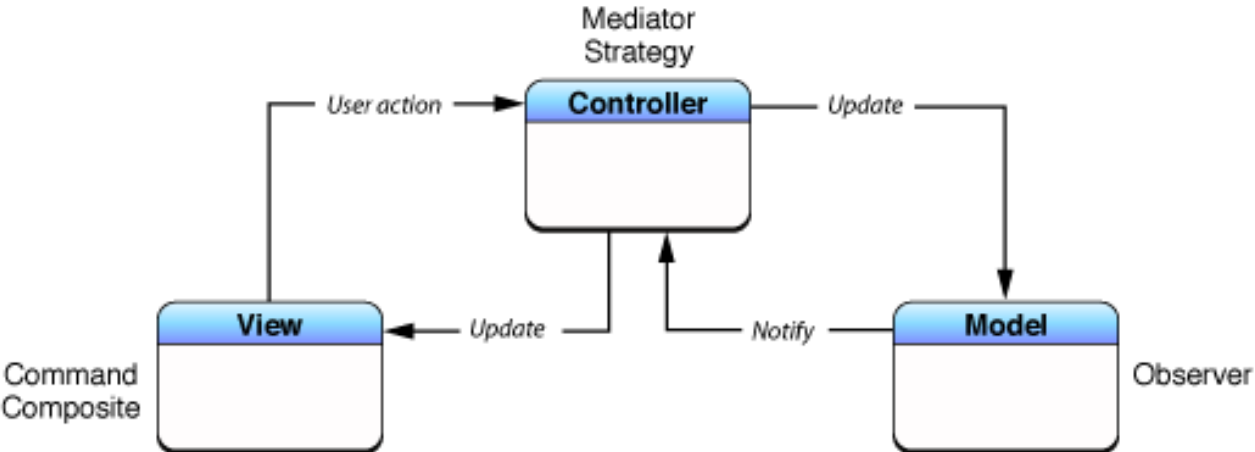


Figure 7-2 Cocoa version of MVC as a compound design pattern

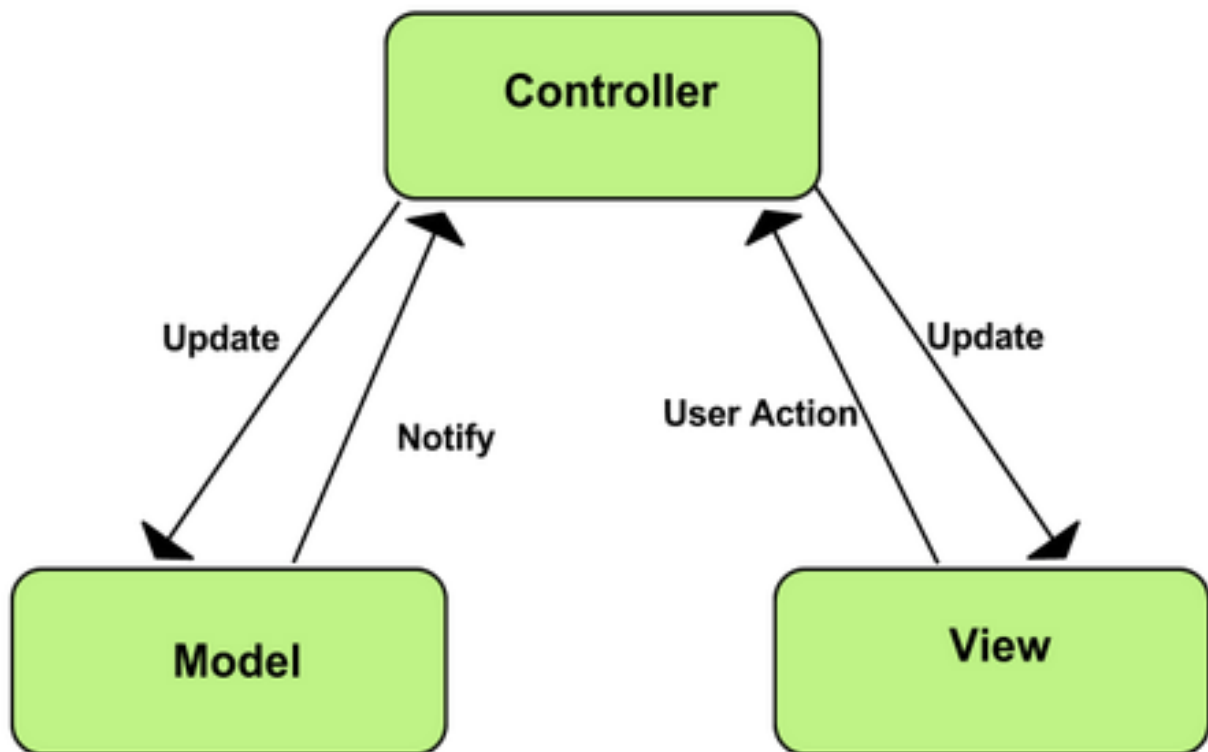


Хм, интересно. Первый вариант выглядит запутанно. Тут, как и у предыдущих интерпретаций, модель связана с представлением в обе стороны. А вот связь с контроллером уже не односторонняя. Действие пользователя проходит через представление в контроллер.

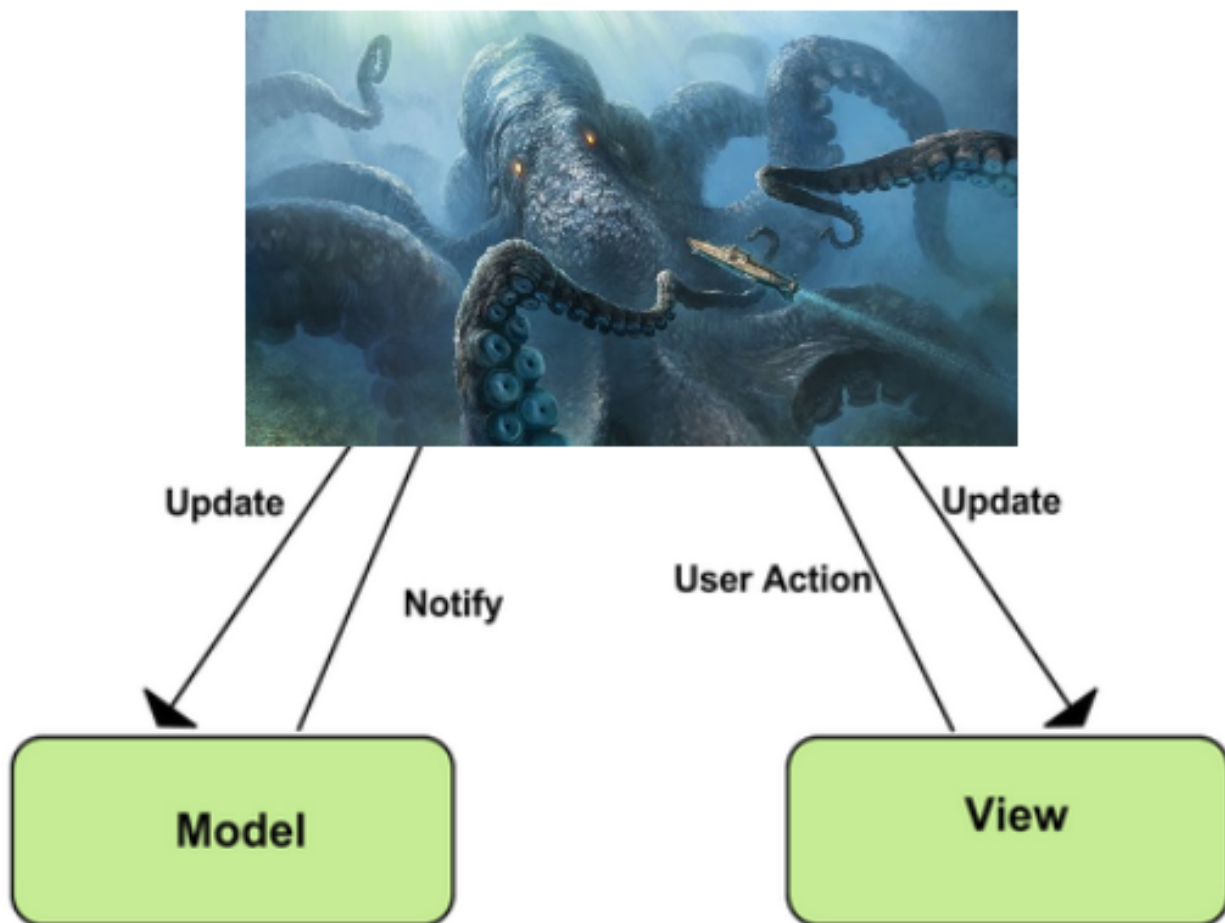
Во втором варианте модель и представление уже не связаны напрямую. Взаимодействие идет через контроллер и это напоминает картинку с осьминогом.

А вот и Google

MVC is composed of three components:



Ну вы поняли, все тот же кракен.

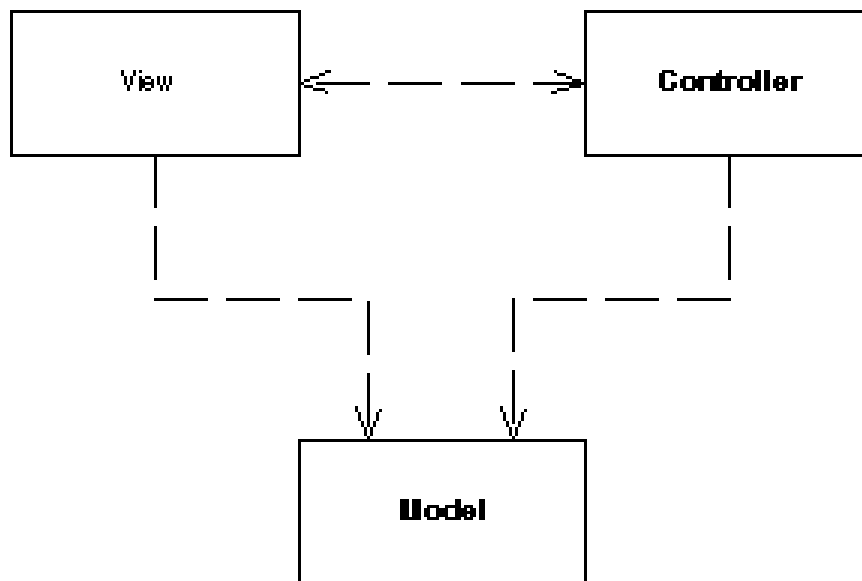


Ниже схемка из блога Мартина Фаулера (к слову, у него есть очень интересный [цикл статей об архитектурах GUI](#), где он подробно описывает его интерпретацию MVC):

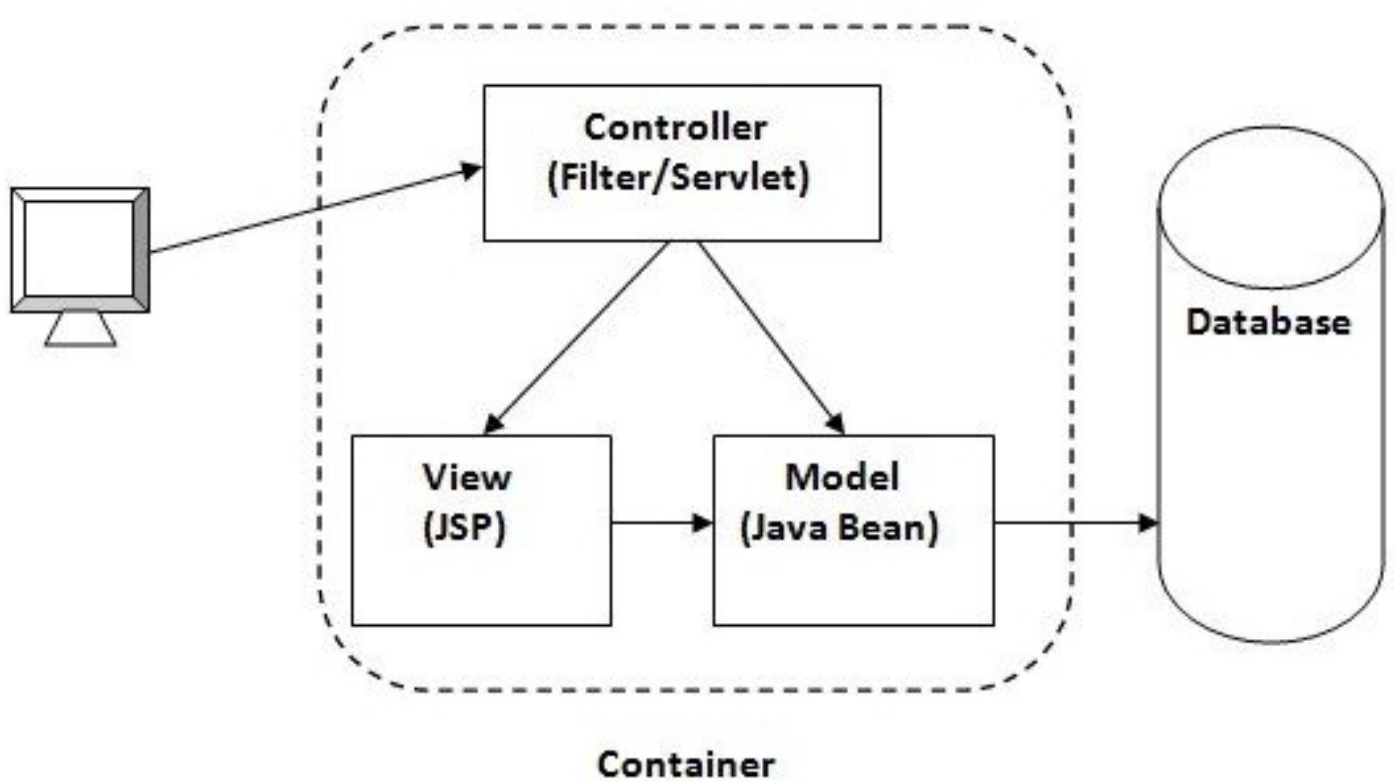
Model View Controller

Splits user interface interaction into three distinct roles.

330



Ниже интерпретация MVC из соседнего мира, такого как Java. Я не понаслышке знаком с JSP. Темное прошлое так сказать.



Вы еще не запутались? Помните как в сериале X-files? Истина где-то рядом.

Ясно одно, все варианты похожи, все они о чем-то одном и том же, но догмы в трактовке Model-View-Controller нет. Почему так?

Видимо потому что первоначальная формулировка не была столь строгой. А может была?

Часто мы очень поверхностно судим о какой-то технологии.

Принимаем на веру чужие интерпретации, вместо того, чтобы открыть первоисточник и сформировать свое собственное мнение.

Что же для MVC является первоисточником?

1970-80е

Стоило наверное начать статью с того же вопроса, который

дядюшка Боб задает на своих семинарах.

Вы знаете кто это?



Это **Трюгве Реенскауг**, профессор университета Осло, тот самый

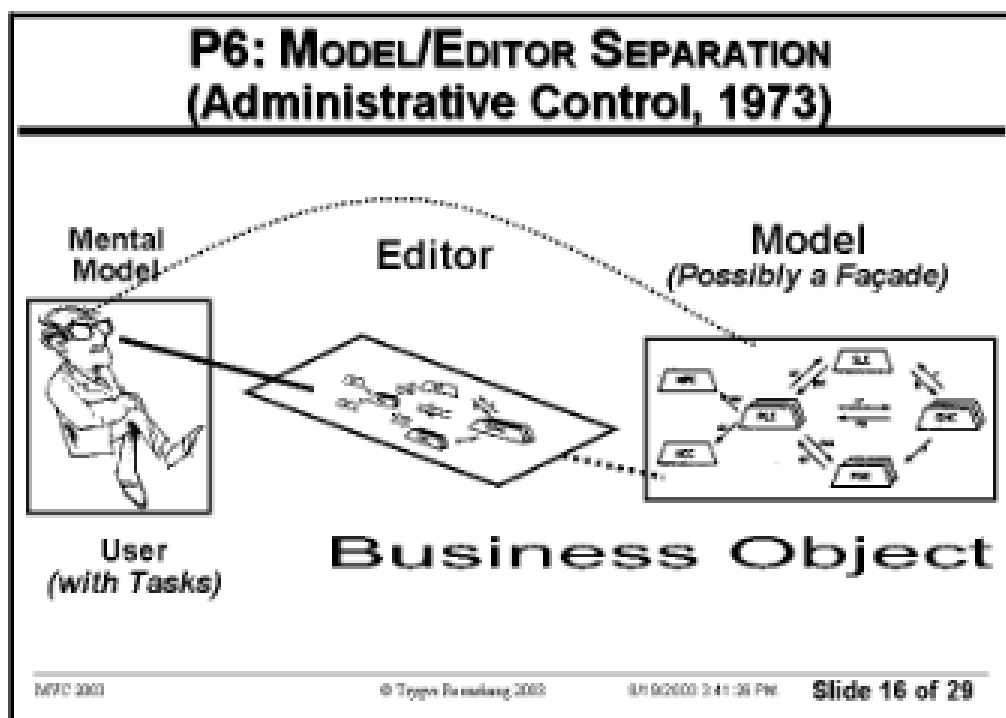
создатель MVC, человек который впервые описал этот архитектурный стиль. Это произошло в далеком 1978, во время его посещения Learning Research Group в Xerox PARC и был заложен фундамент этой концепции, как основа для проекта Dynabook, аналога современного планшета.

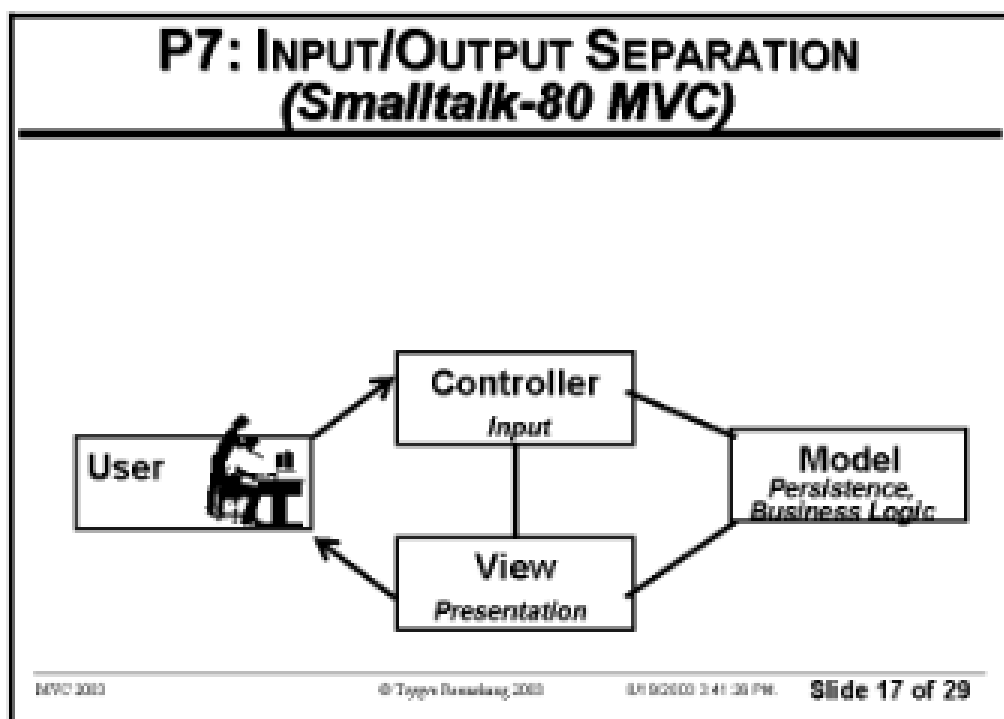
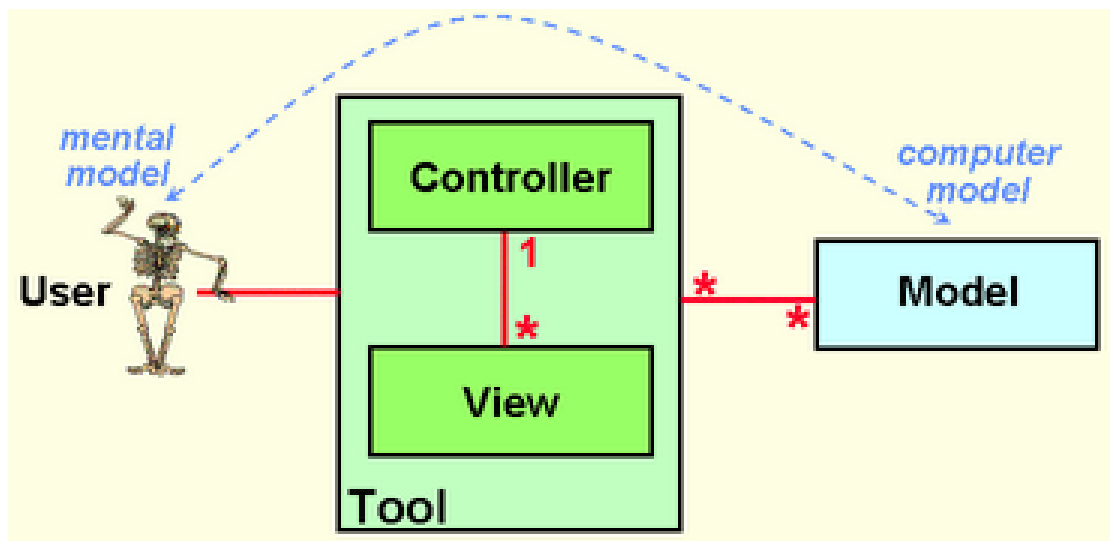
Ему приписывают следующую цитату:

MVC был задуман как главное решение пользовательской проблемы контроля больших и запутанных наборов данных. Труднейшей частью было хорошие имена для различных архитектурных компонентов. (прим. Википедия)

Вот как-то в разрез с картинками от Facebook идет это заявление.

Мне было интересно, как же сам создатель изображает этот архитектурный стиль. И вот пару интересных схем той эпохи:





Интересные картинки, не правда ли? Первое что мне на них очень нравится — это появление пользователя. Мне кажется это важно для любой схемы описывающей архитектуру клиентского программного обеспечения.

Второй интересный момент — это объединяющая область для

контроллера и представления в ранних схемах. Трюгве определяет это как **Editor**, а позже **Tool**.

В его первой заметке об этой концепции были определены четыре термина: Модель, Представление, Контроллер и Редактор.

Редактор является эфемерным компонентом, который представление создает по требованию, как интерфейс между представлением и устройствами ввода, такими как мышь и клавиатура.

После того, как Трюгве покинул Xerox PARC, Джим Альтофф и другие реализовали и внедрили первую версию MVC для библиотеки классов Smalltalk-80; Профессор не был вовлечен в эту работу.

Со слов профессора Джим Альтофф трактовал термин “Контроллер” несколько иначе.

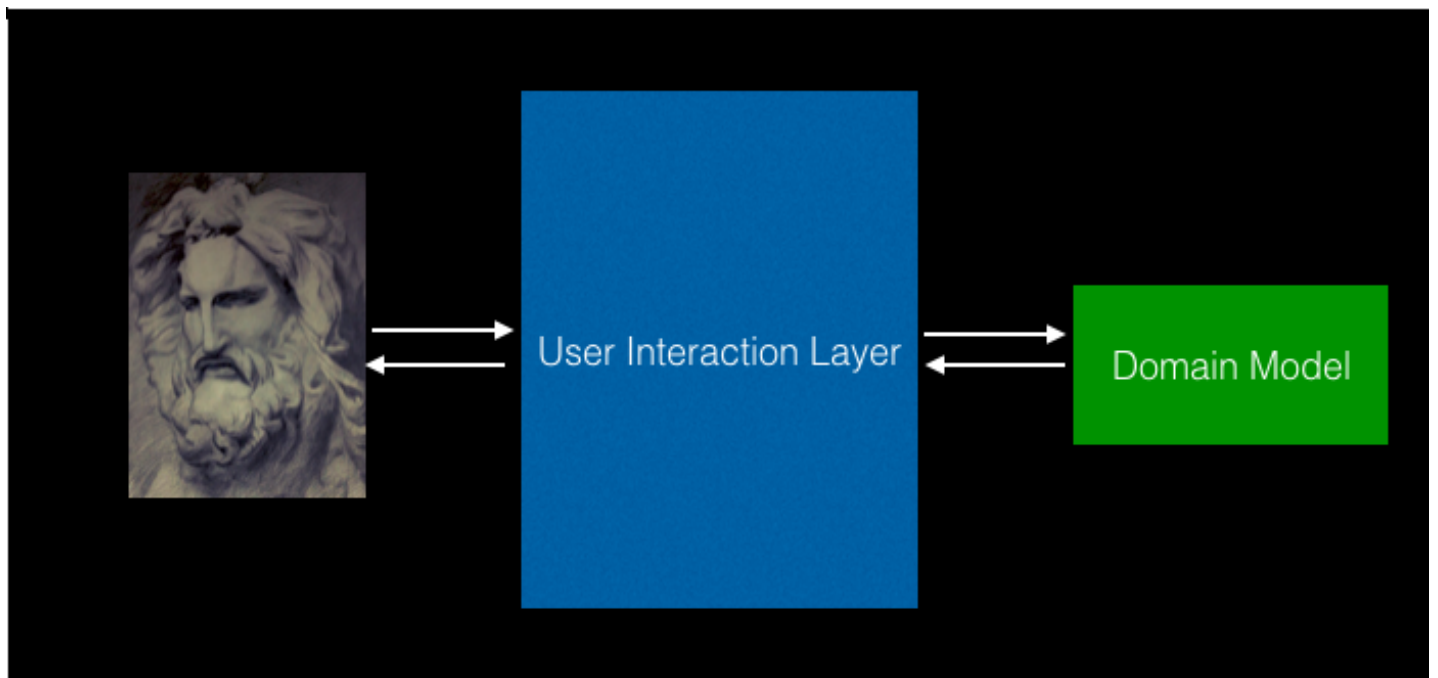
Важным аспектом оригинальной MVC было то, что его контроллер отвечал за создание и координацию своих подчиненных представлений.

В более поздних заметках представление принимает и обрабатывает пользовательский ввод, относящийся к самому себе. Контроллер принимает и обрабатывает входные данные, относящиеся к сборке Controller / View в целом, который теперь называется **Tool**.

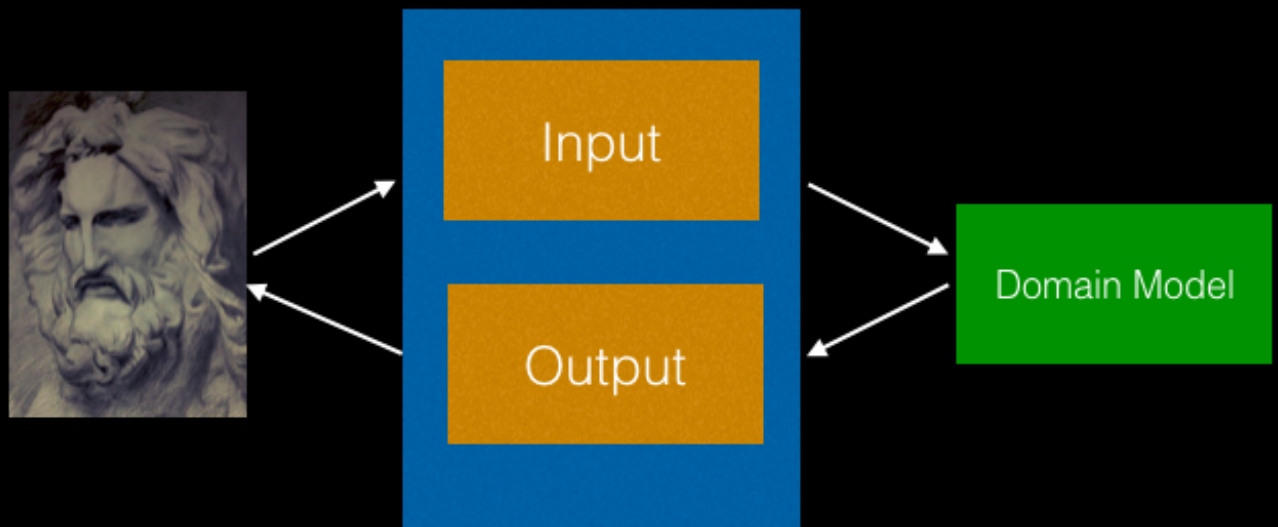
К слову, Мартин Фаулер называет этот тандем **Presentation Layer**.

В своей работе Трюгве делает уклон на то, что основные игроки любой клиентской системы это конечный пользователь (**Mental Model**) и данные предметной области (**Domain Model / Data**). И главная цель MVC это преодоление разрыва между ментальной моделью пользователя и цифровой моделью, которая существует в компьютере.

Это концепция актуальна и по сей день, невзирая на то, что, то была эпоха SmallTalk и понятия web-приложений еще не было как такового. Взаимодействие пользователя и данных — в этом суть любой программы. Между этими объектами существует слой, посредством которого они общаются друг с другом.

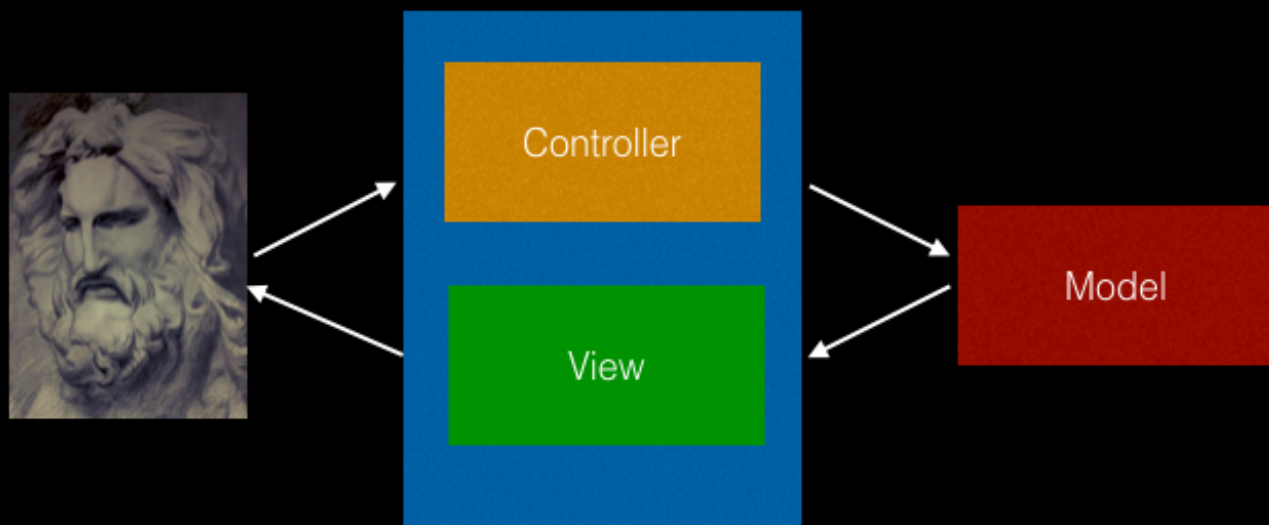


В основе взаимодействия программы и человека всегда лежат входной и выходной потоки данных. Обобщенная схема могла бы выглядеть так:



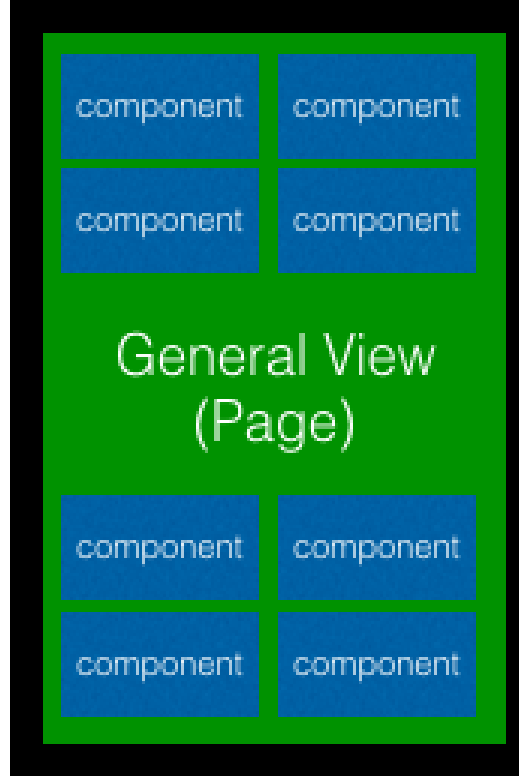
Таким образом я интерпретирую контроллер как точку входа данных, а представление как точку выхода. При чем точка входа это совокупность периферийных устройств и элементов управления представления. При этом неявно представление и контроллер связаны, но напрямую не общаются. Представление, как бы делегирует обязанность по отправке сообщения о пользовательском действии контроллеру.

Входной поток способен изменить состояние предметной модели, модель же в свою очередь должна оповестить пользователя о своих изменениях, используя для этого точку выхода данных.

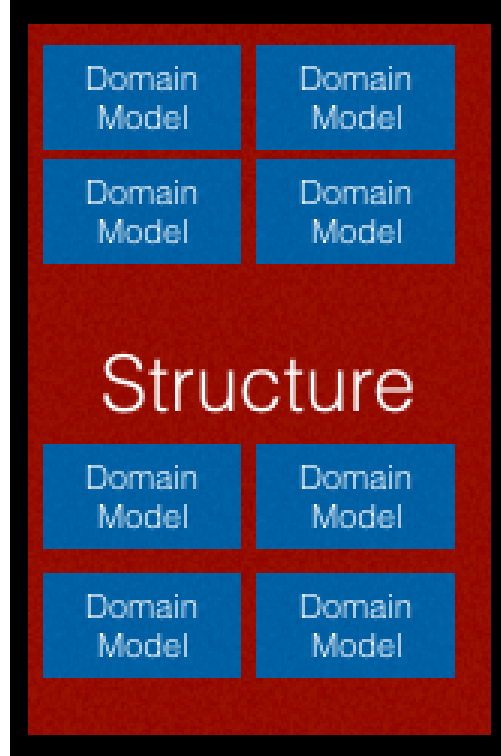


На одной из схем Трюгве связь контроллера и представления один-ко-многим. Почему? Я думаю потому что в то время любая кнопка или стрелочка на пользовательском интерфейсе являлась представлением. Изменилось ли что-то с того времени? Я думаю нет.

В интерпретации “кракен” под представлением часто понимают всю страницу. На деле это всегда дерево компонентов. Каждый компонент вполне может выступать в роли представления:



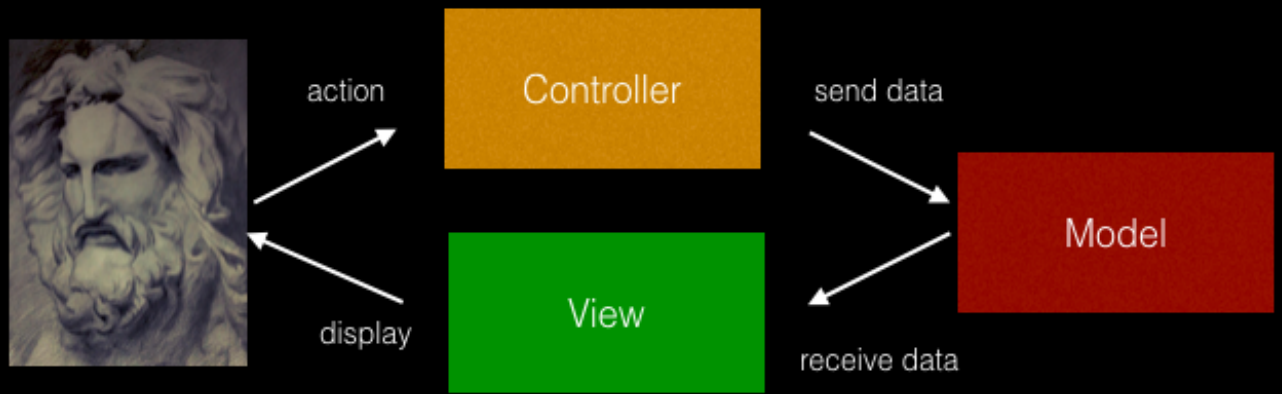
Связь слоя взаимодействия с пользователем и моделью многие-ко-многим. Это актуально и для нынешних программ, одна страница может отображать данные и характеристики различных предметных моделей, модель же в свою очередь может представлять из себя структуру данных:



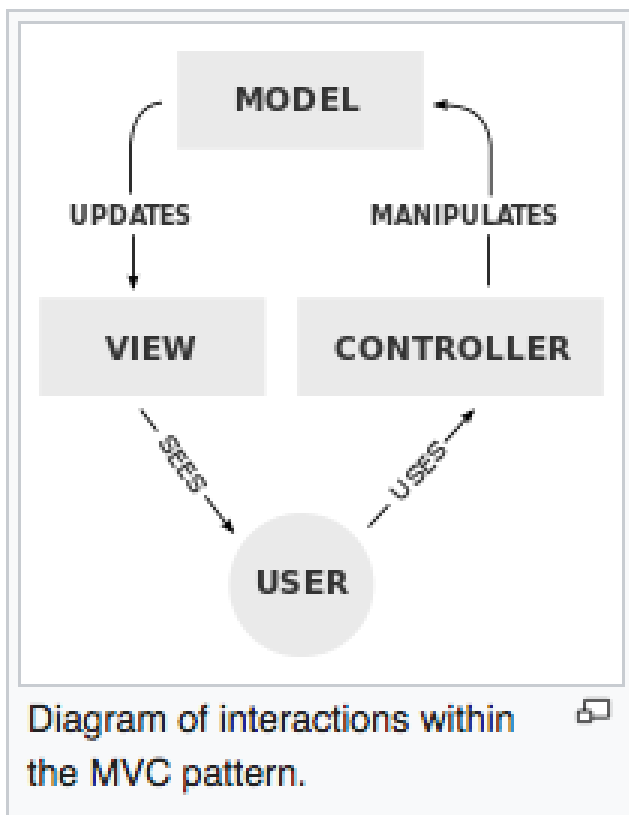
Попробуем сформулировать обобщенное описание MVC:

- Модель (**Model**) предоставляет данные и реагирует на команды контроллера, изменяя свое состояние;
- Представление (**View**) отвечает за отображение данных модели пользователю, реагируя на изменения модели;
- Контроллер (**Controller**) интерпретирует действия пользователя, оповещая модель о необходимости изменений

Выходит какой-то прям **Unidirectional Dataflow**:



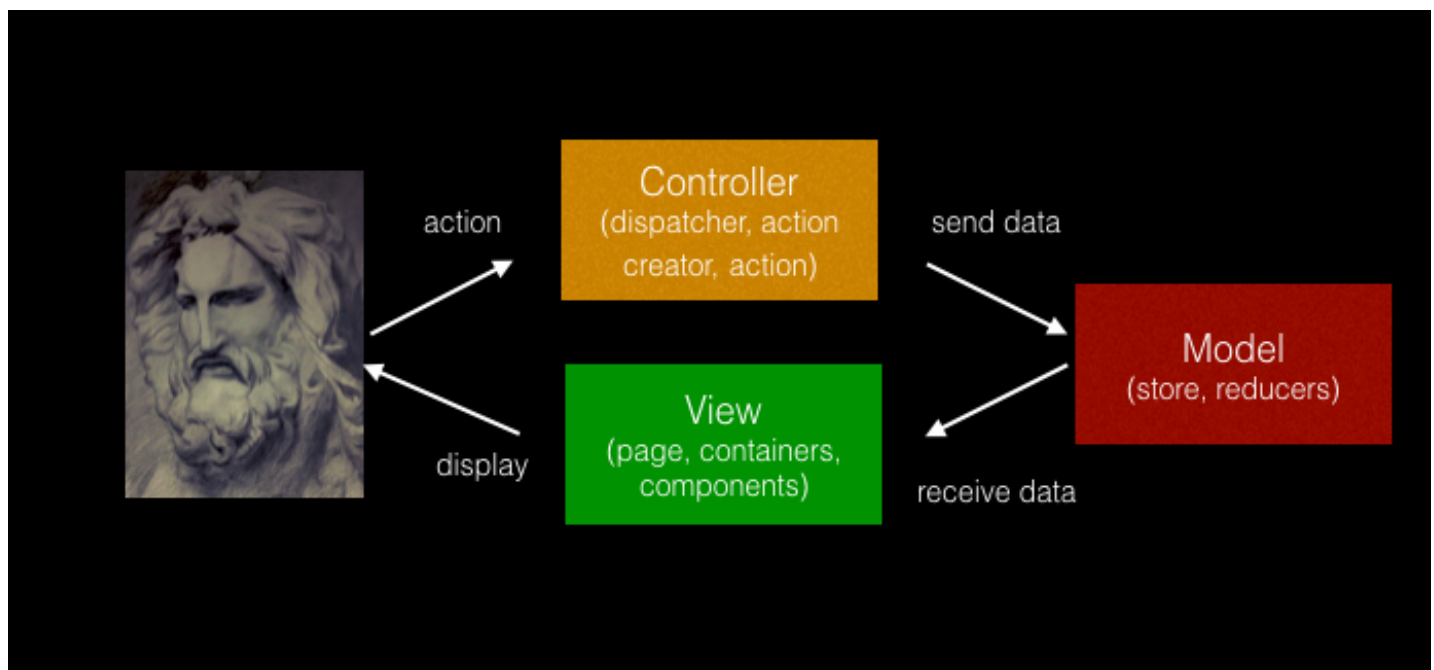
Изображение из Википедии



А не тоже ли это самое, что предлагает Facebook и Co?

- Хранилище (**Store**) — хранилище данных;

- Диспетчер (**Dispatcher**) — направляет действие пользователя хранилищу;
- Представление (**View**) — дерево React-компонентов;
- Действие (**Action**) — действие пользователя, планарный объект;
- Создатель Действия (**Action Creator**) — порождает действие (объект);
- Редьюсер (**Reducer**) — меняет состояние хранилища данных;



Выходит революции не произошло. Ничего не изменилось. Все по-прежнему через призму MVC. Я наблюдаю лишь очередную интерпретацию этой концепции и подмену понятий.

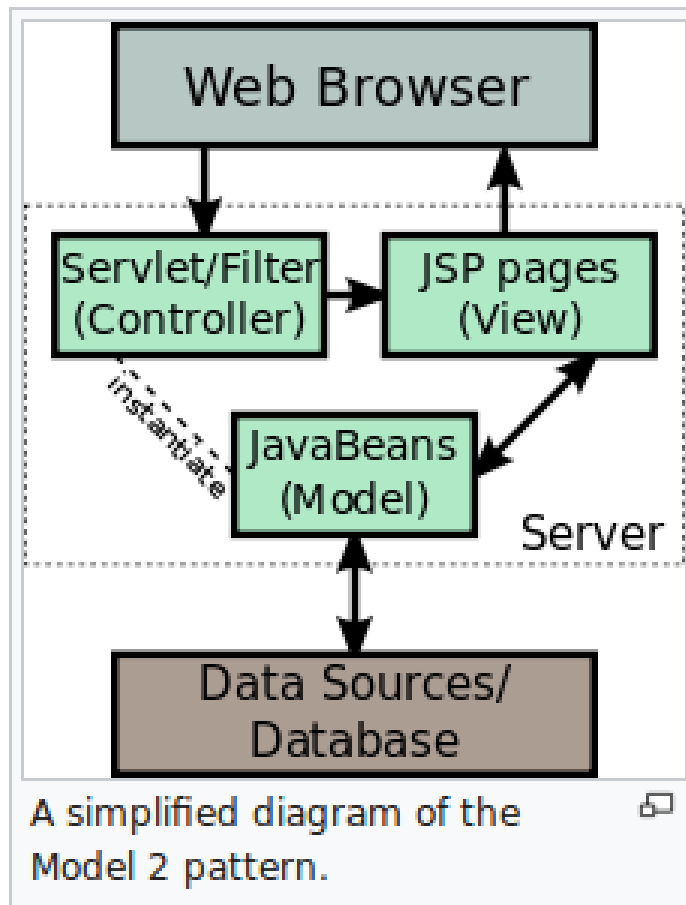
А о каких же проблемах тогда говорили представители Facebook?

Начать наверное надо с того, что с течением времени трактовка концепции о разделении обязанностей системы свелась к реализации конкретных шаблонов.

Взгляните на схемы Apple. Они описывают слои системы конкретными шаблонами: Слушатель, Стратегия, Композиция. Эта ошибка тянется со времен появления “**Банды четырех**”.

Эволюция консольного программного обеспечения в серверное и клиентское тоже добавила свой отпечаток на первоначальных определениях.

Абстракции превратились во что-то конкретное:



А что же до проблем, то проблемы все те же и озвучены уже давно. Они и для Flux/Redux будут такие же. И для любой другой концепции и фреймворка. По началу у тебя есть чистая мысль и

ты начинаешь с чего-то простого. Но с течением времени, когда ты перестаешь поддерживать чистоту мысли в своем коде, он превращается в страшного кракена макаронного монстра. Проект начинает вязнуть в этой смоляной яме.

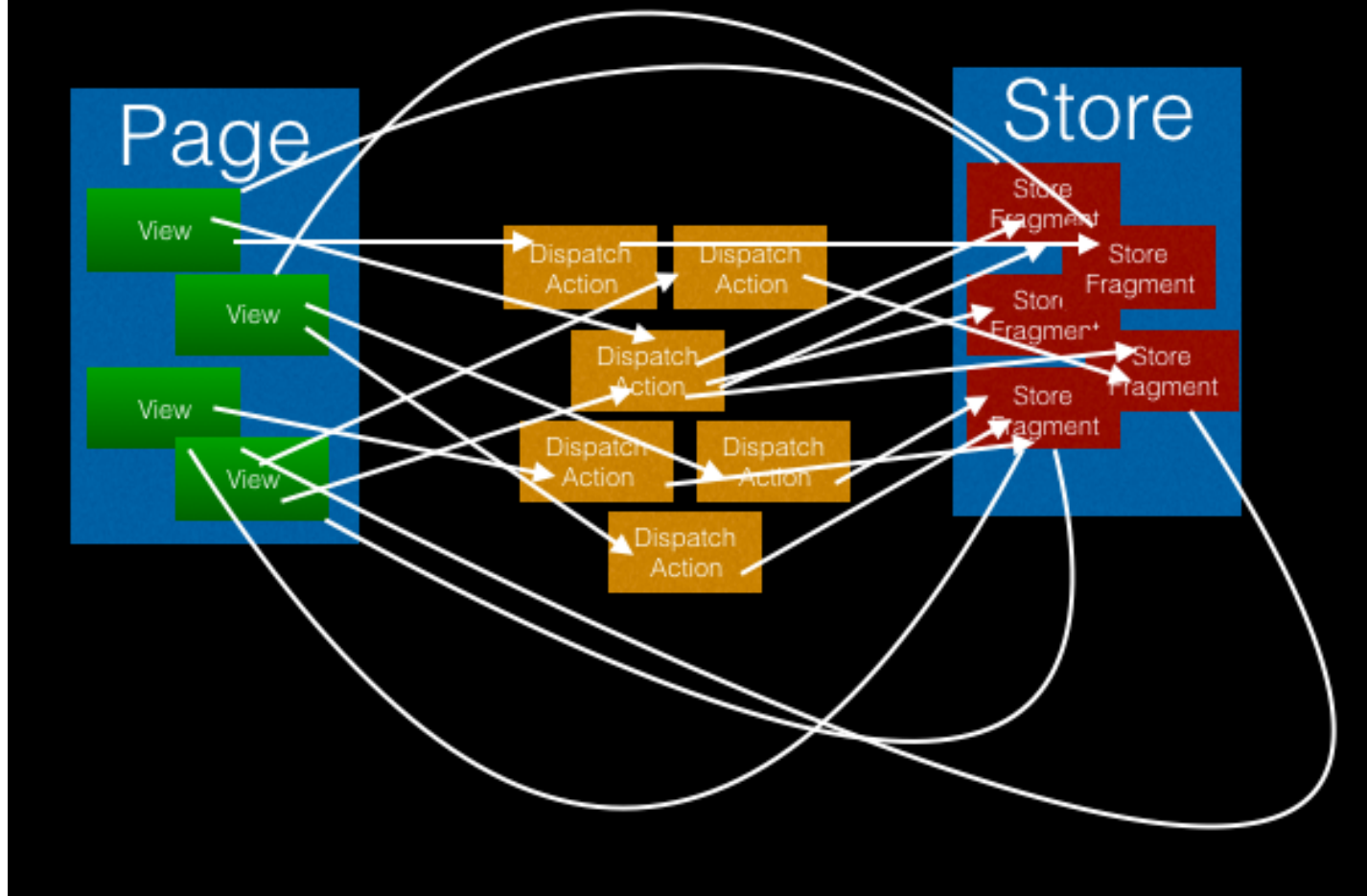
Ruby Midwest 2011 - Keynote: Architecture the Lost Years

HOW MVC GOES WRONG AS A WEB ARCHITECTURE.

The diagram illustrates a web architecture where the MVC pattern has become overly complex. On the left, a box labeled 'MODELS' contains eight green circles, each labeled 'B.O.'. In the center, a box labeled 'CONTROLLERS' contains one blue rectangle. On the right, a box labeled 'VIEWS' contains one blue rectangle. Arrows show a dense web of dependencies: every 'B.O.' in the 'MODELS' box is connected to every 'CONTROLLER' and every 'VIEW'. Additionally, the 'CONTROLLERS' box is connected to the 'VIEWS' box. On the far right, a vertical line represents 'THE WEB', with arrows pointing from the 'CONTROLLERS' and 'VIEWS' boxes towards it. The overall impression is one of a tangled, unmanageable web of dependencies.

32:32 / 1:06:38

YouTube

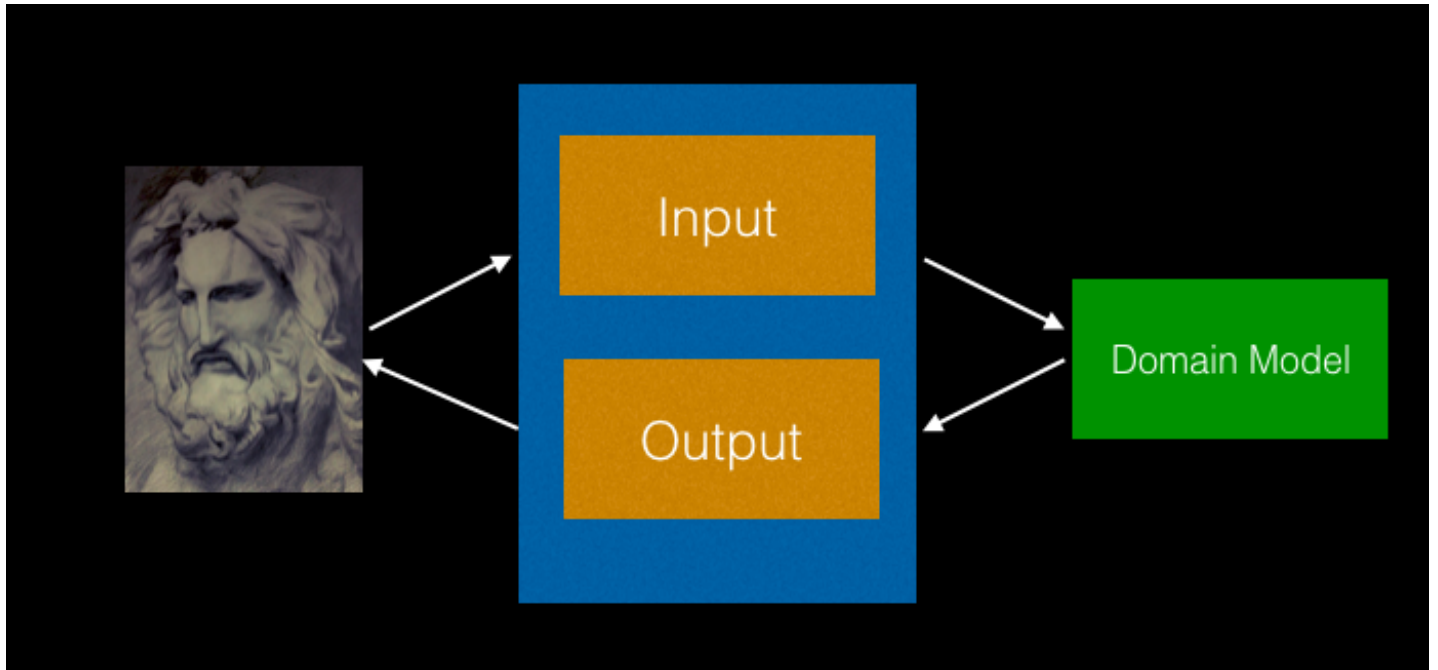


Помогут ли мне Redux и Flux выстроить архитектуру моего проекта чище и понятнее? Сохранят ли они эту чистую мысль? Есть ли вообще инструменты, которые не позволят скатиться в клубок этих стрелок?

Как инструменты и очередные интерпретации они имеют право на жизнь. Но на текущем проекте я начал потихоньку выпиливать Redux. Посмотрим, что из этого выйдет. Получится ли мне сохранить эту чистую мысль или все в очередной раз опять превратится в болото.

Выводы

Пока не произойдут кардинальные изменения в области Человеко-компьютерного взаимодействия, вся концепция будет всегда базироваться на этом:



Все будет проходить через эту точку зрения. И проблема, этих сотен стрелок в разные стороны, будет жить с этой концепцией дальше.

Просто кому-то удастся написать чисто, а кто-то все таки вязнет в своем же болоте.

И грани этой концепции стираются в исходном коде этого болота. Знаю по опыту, ибо сам превратил кучу проектов в эту трясиину.

Вам, наверное, хочется задавать вопрос: *“Ну круто дядька, и че дальше?”*

Ничего, относитесь к этому тексту как к моему Brain Dump.

Сейчас я рад, что все таки погрузился в эти термины и это «переосмысление» разработки клиентского слоя web-приложения. По крайней мере для себя я по-новому начал интерпретировать MVC. Более абстрактно трактовать эти термины.

Так же я рад что созрел опубликовать свои мысли. Возможно через какое-то время мое мнение изменится и будет интересно вернуться к этому тексту и посмеяться.

P. S. Все выше написанное это мое скромное субъективное мнение и моя интерпретация. А какая она у вас? Буду рад увидеть другие точки зрения artur.basak.devingrodno@gmail.com

Продолжение статьи

-> [«Загадка Кракена» — коротко о истории MVC и других,](#)
-> [«Охота на Кракена» — Flux сложно, Redux не катит, упрощаем все до Artux](#)

Ссылки

- [MVC Wiki](#)
- [Trygver MVC](#)
- [Flux](#)
- [Apple MVC \(1\)](#)
- [Apple MVC \(2\)](#)
- [Google MVC](#)
- [Uncle Bob](#)
- [Martin Fowler MVC](#)



+50



Поделиться:



Сохранить:



Комментарии (40)

Похожие публикации

Адаптивные Split View Controller и Popover в iOS 9 (Swift). Часть 2

WildGreyPlus • 10 марта 2016 в 19:26

0

Адаптивные Split View Controller и Popover в iOS 9 (Swift). Часть 1

WildGreyPlus • 10 марта 2016 в 17:14

1

Реализация Model-View-Presenter в Qt

skb7 • 8 ноября 2010 в 01:36

17

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на русском языке

alexkuku • вчера в 16:12

14

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада

lahmatiy • вчера в 13:05

4

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе

ПЕРЕВОД

Smileek • вчера в 10:32

25

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации

ПЕРЕВОД

ru_vds • вчера в 12:04

2

Как адаптировать игру на Unity под iPhone X к апрелю

P1CACHU • вчера в 16:13

0

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

33

Обзор рынка моноколес 2018

lozga • вчера в 06:58

70

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

alizar • вчера в 15:14

40

Стивен Хокинг и его работа — что дал ученый человечеству?

marks • вчера в 14:46

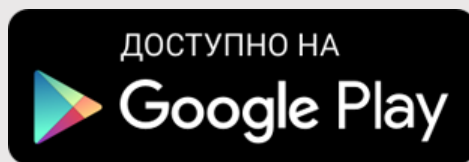
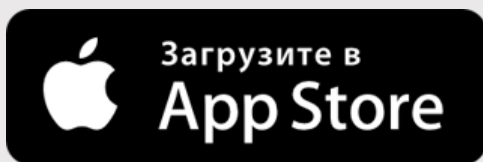
8

Sunlike — светодиодный свет нового поколения

AlexeyNadezhin • вчера в 20:32

17

Мобильное приложение



Полная версия

2006 – 2018 © TM