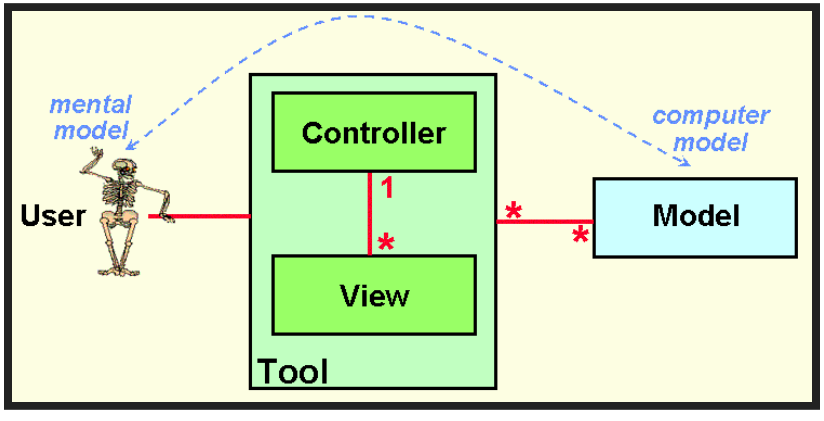
JavaScript & MVC



Model-View-Controller

- Шаблон за дизайн (design pattern)
- Създаден около 1978 и от тогава не е излизал от мода
- Решение са специфичен проблем - да се улесни мисловният модел на потребителя и да му се даде възможност да управлява по-добре информацията (separation of concerns)
- Trygve Reenskaug & Xerox PARC

Separation of Concerns

Проблемът, който MVC решава

- Model - данни
- View - изобразяване
- Controller - взаимодействие

Защо трябва да ни пука за класическото MVC?

Доброто разбиране на решението, ни помага да използваме MVC фреймуърците по-добре.

На теория :)

Модели

- Обхващат специфични за дадена област знания и данни
- Комуникират с контролера в изолация от вюто

Вюта

- Трябва да знаят за контролера
- Трябва да знаят за модела

Контролери

- Обработват input и output
- Променят състоянието на моделите
- Някои имплементации съчетават вюта и контролери

Още малко история

Какво и как се случва за да се постигнат настоящите резултати

Спагети, минимален JavaScript в уеб страниците

JavaScript не е на фокус в разработката на уеб

Rails и популяризирането на MVC в Python, PHP

Rails и Django показаха, че MVC наистина работи

Все още минимум JavaScript в браузърите

Ajax

jQuery, Dojo, Mootools

REST и разработката на все по-бърз уеб

REST е толкова лек, че лесно може да работи с JavaScript

Нужда от стабилни архитектури за правене на решения, в които не се генерира браузърен код на сървъра

Изобилие от JavaScript MVC фреймуърци

Броят им постоянно расте, а различни архитектури се конкурират

Но как се стига до тук

Интерпретация

Шаблоните за дизайн би трябвало да бъдат общ език, но всъщност всички ги интерпретират различно

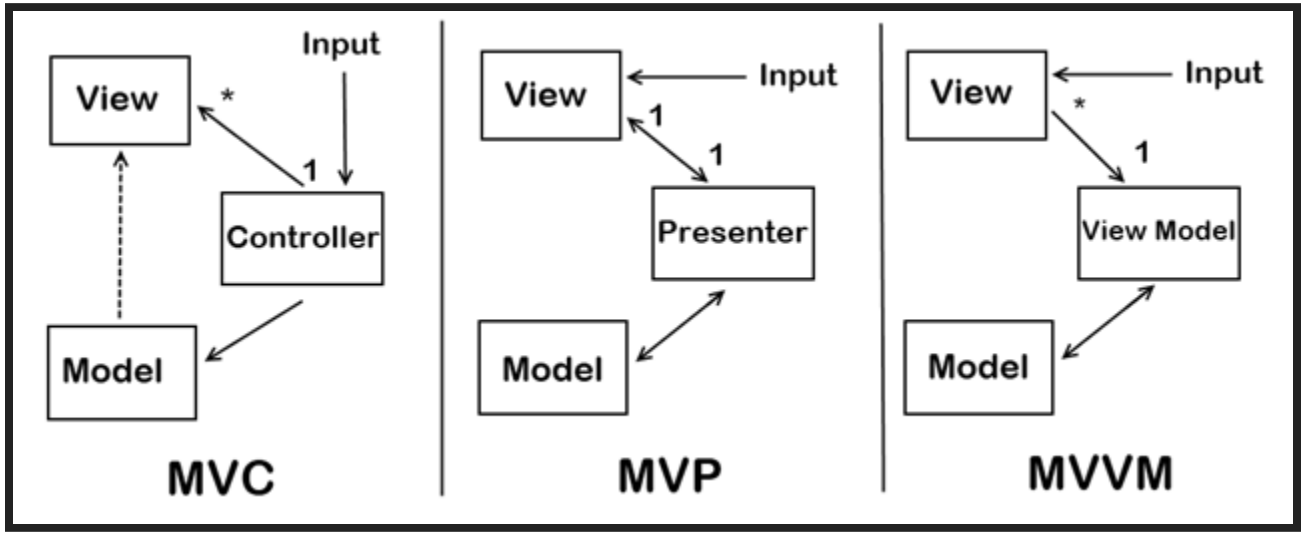
Разнообразие

Всеки има различно мнение за това какво представлява separation of concerns

Не е нужен фреймуърк за да ползвате MVC

Разновидности на MVC

- MVC
- MVP (Model-View-Presenter)
- MVVM (Model-View-ViewModel)
- MV*



Примери

- Ember.js
- Backbone.js - между MVP и MVC
- KnockoutJS - MVVM
- AngularJS

Реалност

В момента ние всички ползваме класически шаблони и ги адаптираме за работа с уеб

Понякога се получава :)

Без какво MV проектите не могат

Фреймуърците поддържат в различна степен най-важните функционалности за жизнеспособни MV* приложения

Маршрутизиране (Routing)

Маршрутизирането е важно за състоянието на приложенията

Зареждане на модули (Module Loading)

Големите приложения имат нужда от модули, а модулите от ефективно зареждане

- Asynchronous Module Definition (AMD) & RequireJS & friends
- Bundling - Browserify & friends

Rendering

Вградени - Angular, Ember

Handlebars & Underscore

React & Ractive

Internationalization

Достъпност

Nice to have

TodoMVC

Ако искате да научите повече за MVC в JavaScript