accenture technology

Strategy | Digital | Technology | Operations

UI Architecture Patterns and Frameworks Point of View

High performance. Delivered.

Sponsor

NITIN M. SAWANT

IDC - Advanced Technology And Architecture Lead

Authors

MANOHARAN RAMASAMY

Advanced Technology & Architecture(ATA), UI Architecture Capability Lead

BASKARAN VARADARAJAN

MUTHARASU GURUSAMY

Agenda

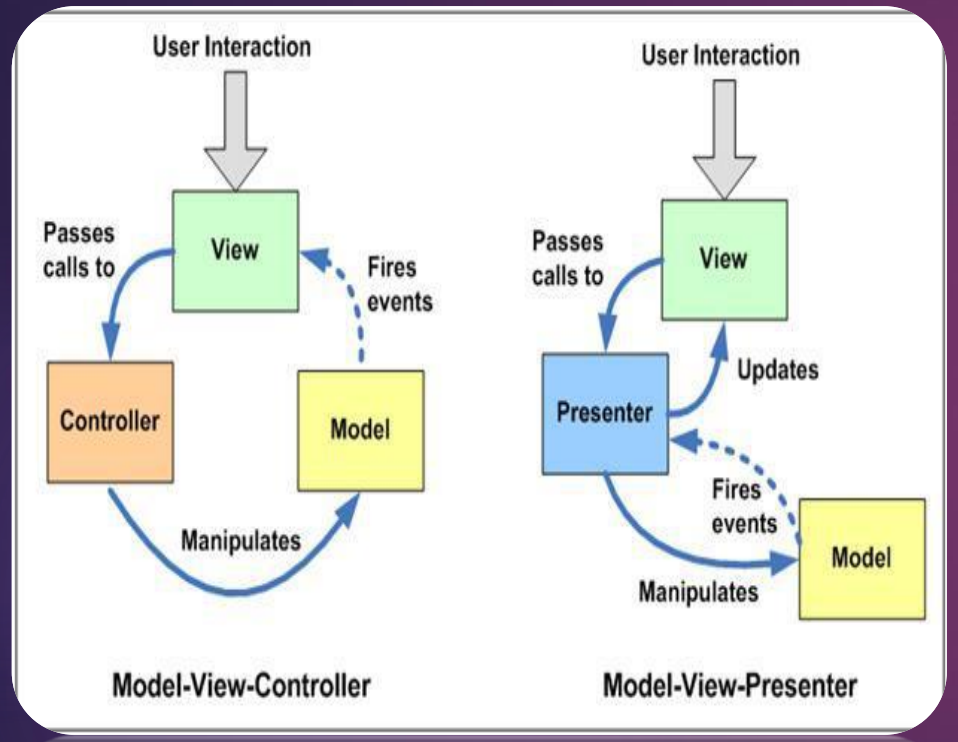
1. UI Architecture Patterns
    1.1. What is UI Pattern - Why its needed
    1.2. Types Of UI Patterns
2. UI Libraries /Frameworks
    2.1 Overview
    2.2 Patterns in support
    2.3 Usage Comparison
3. Comparison of UI Frameworks/ Libraries
4. Comparison of UI Patterns
5. UI Tools
6. References

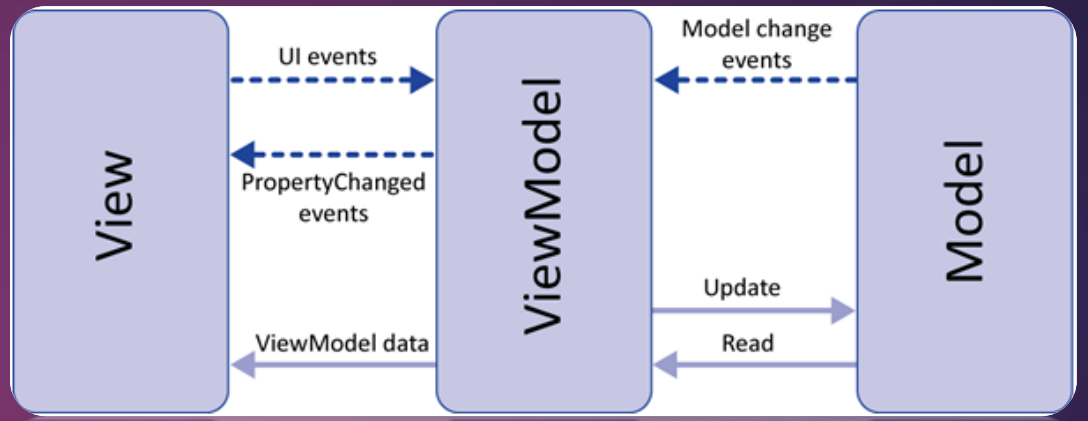
UI Architecture Patterns

What(Why) is UI Pattern

- One of the biggest problems associated with user interface is lot of cluttered code. This cluttered code is due to two primary reasons:
  - UI has complicated logic to manipulate the user interface objects.
  - UI also maintains state of the application.
- UI patterns revolve around how to remove the UI complication and make the UI more clean and manageable. Below are different variety and classifications of UI patterns as shown in the below figure.

Model-View-(C or VM or P)



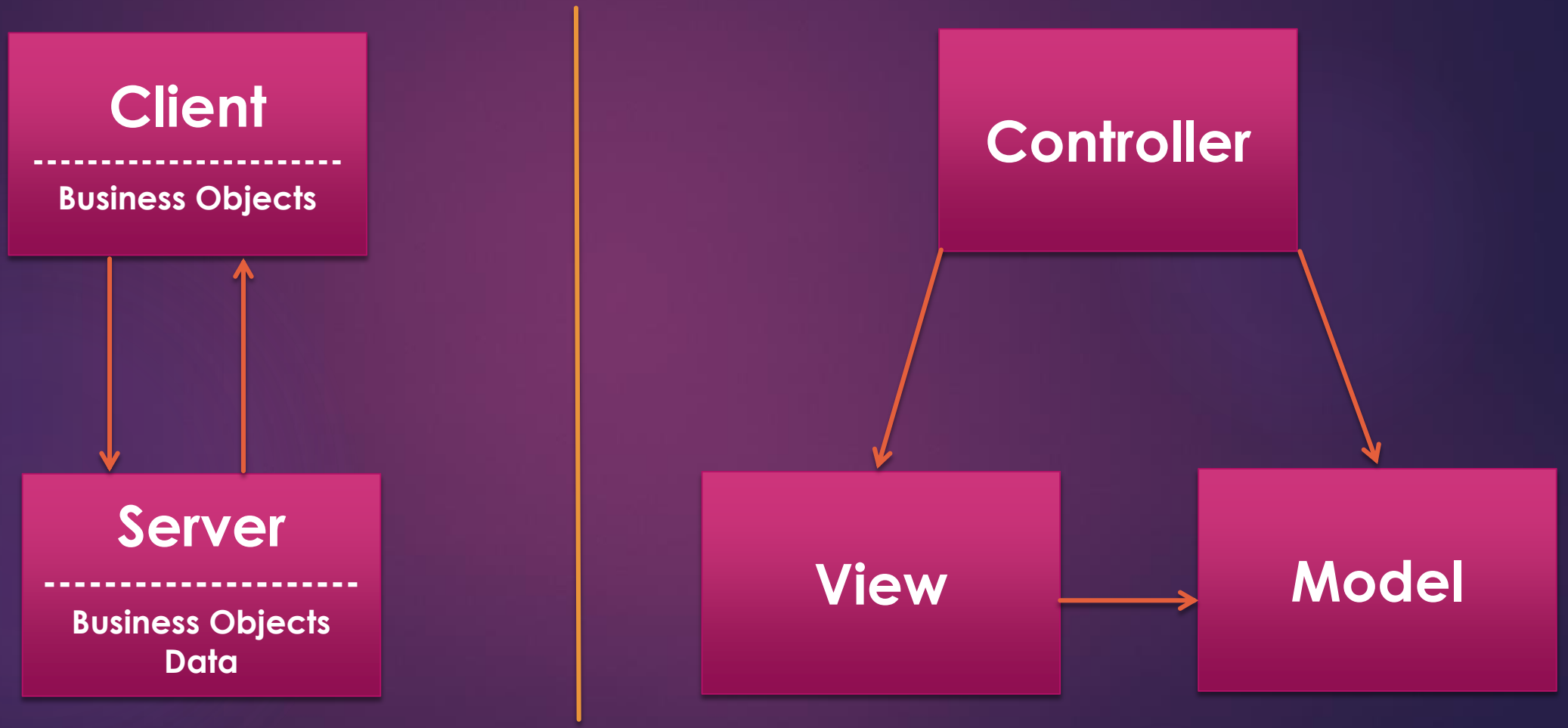


- Model-View-Controller (MVC) pattern
- Model-View-Presenter (MVP) pattern
- Model-View-View Model (MVVM) pattern

What is Model/View/(C or VM or P)

- Patterns that describe a modular approach to software development
- Modules include:
  - Model – Data
  - View – Presentation Layer
  - C or VM or P – Glue Logic
- They are based upon a "Separation of Duties"
  - Seen in many other types of system frameworks
- It is different from n-Tier development in the 90's - Distributed interNet Architecture (DNA)

Client/Server (DNA) vs MVC



Why Model-View-(C or VM or P)

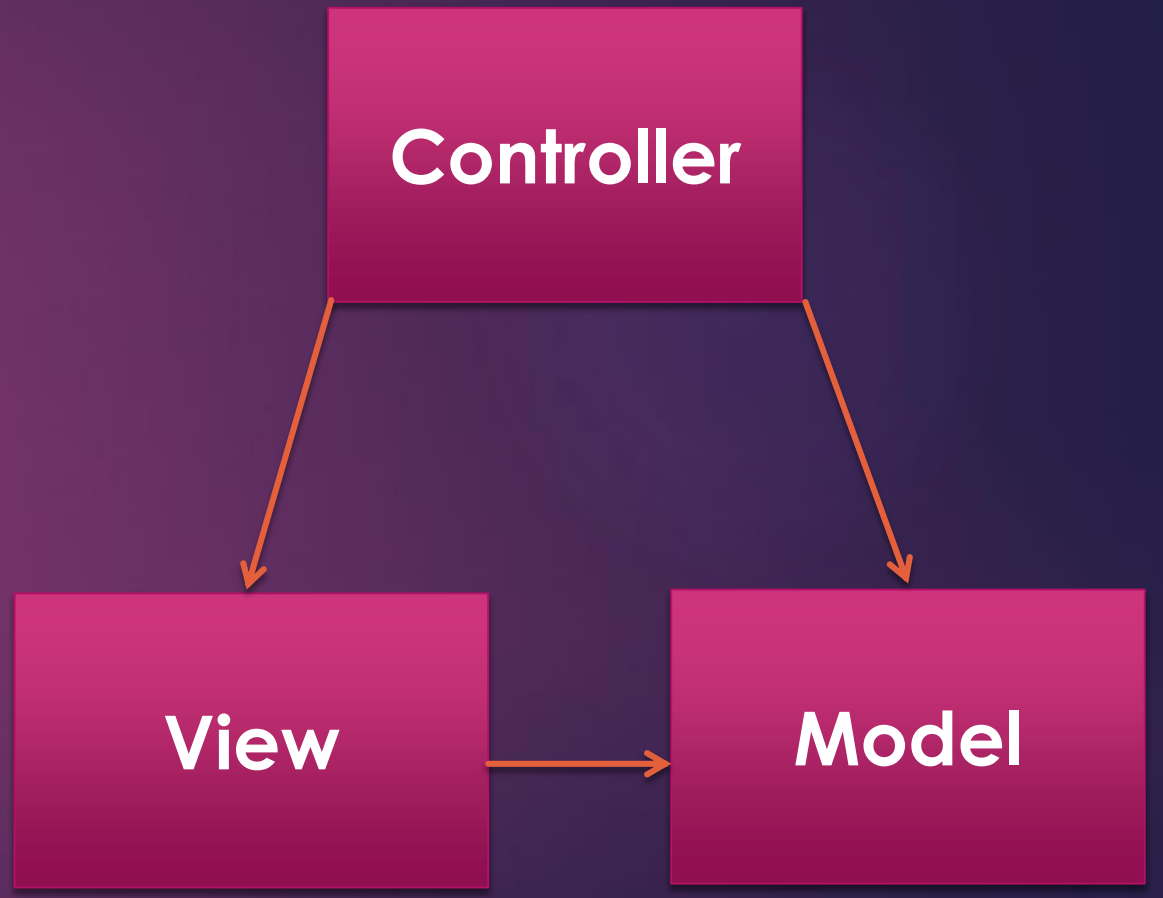
- The Patterns all have similar goals, however,

  achieve

  them in different ways
- The Patterns goals are to increase:
  - Modularity
  - Flexibility
  - Testability
  - Maintainability

Model-View-Controller (MVC)

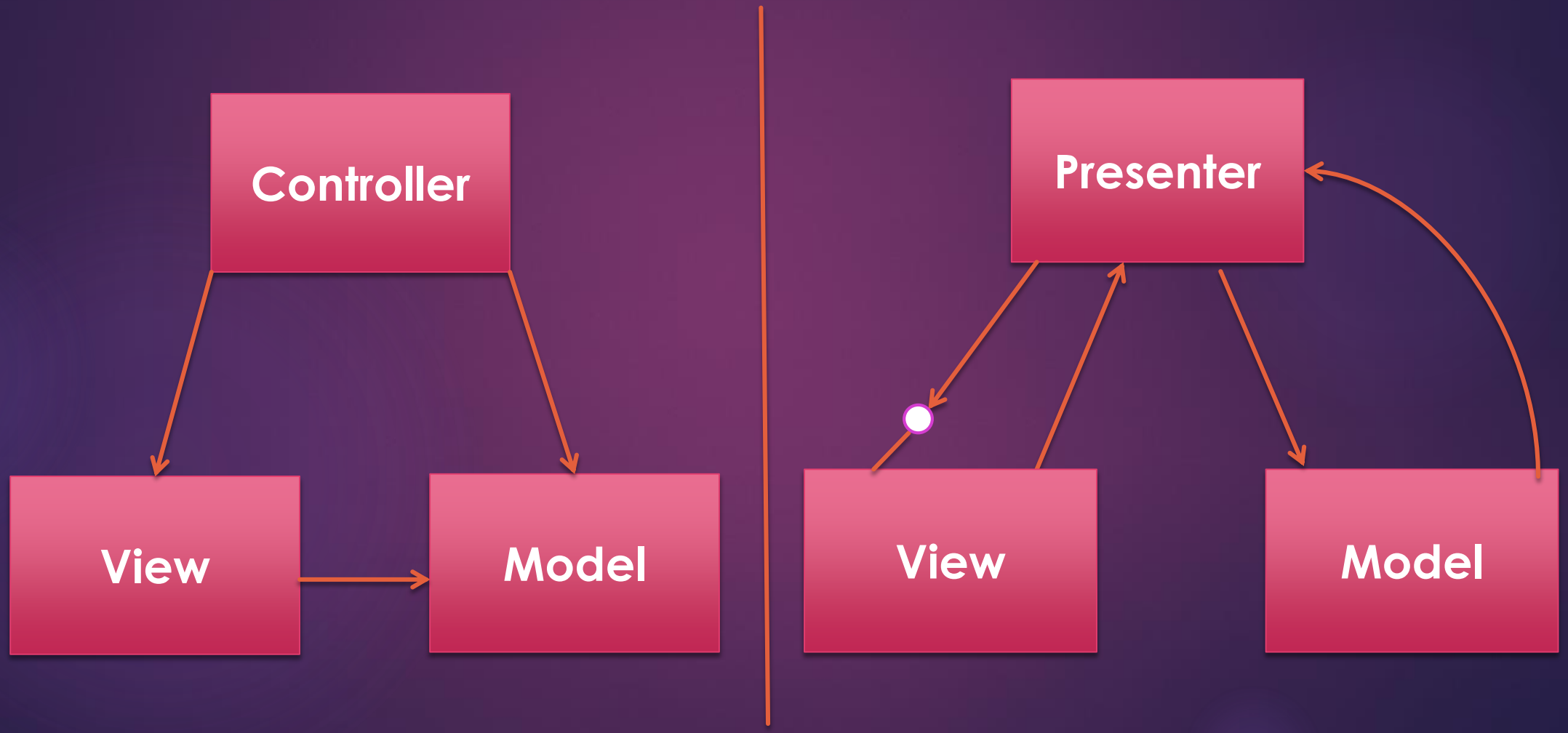
- Controller is centerpiece that decouples the Model and View
- Control flow:
  - User interaction event
  - Controller handles event and converts it to a user action the Model can understand
  - Model manages the behavior and data of the application domain
  - The View interacts with the Controller and Model to generate a user interface



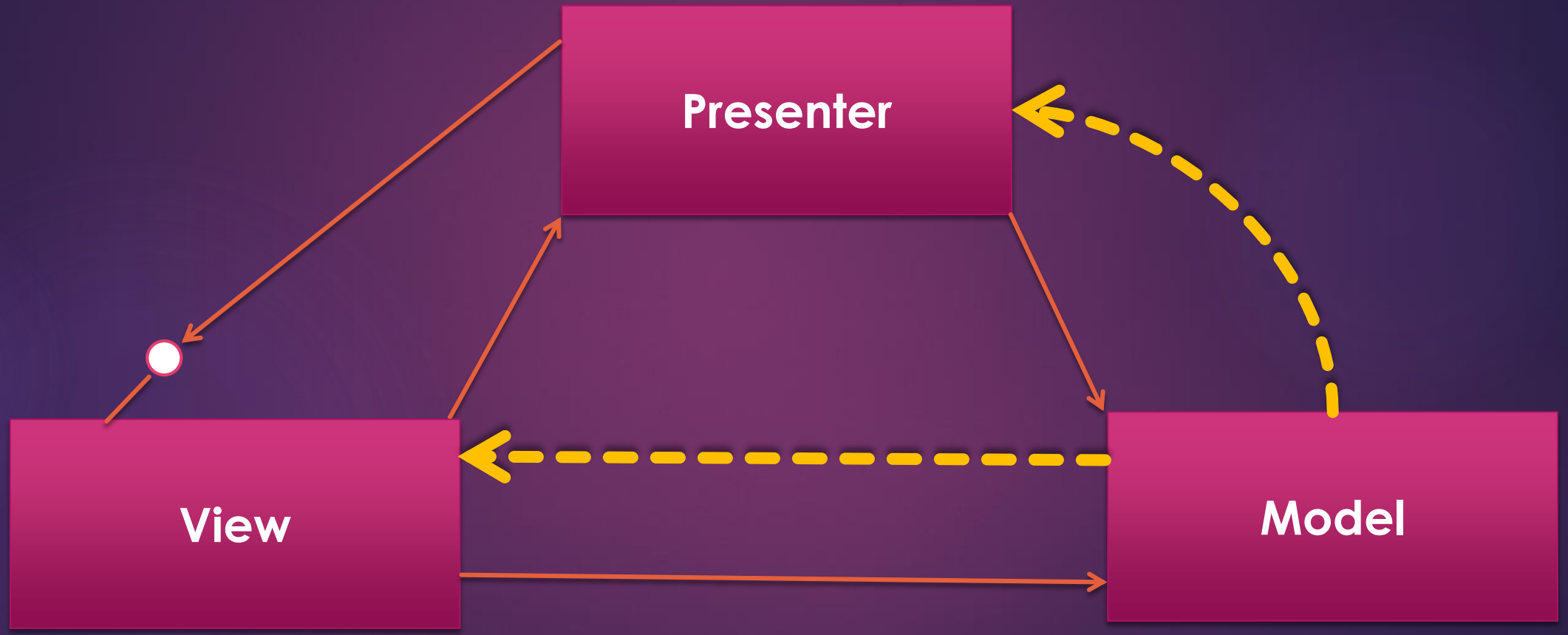
Model-View-Presenter (MVP)

- MVP is a derivative of MVC
- Two types of implementation
  - Passive View
  - Supervising Controller
- Presenter assumes the functionality of the MVC Controller
- View is responsible for handling UI events
- Model becomes strictly a Domain Model
- More User Interface centric

MVC vs MVP (Passive)



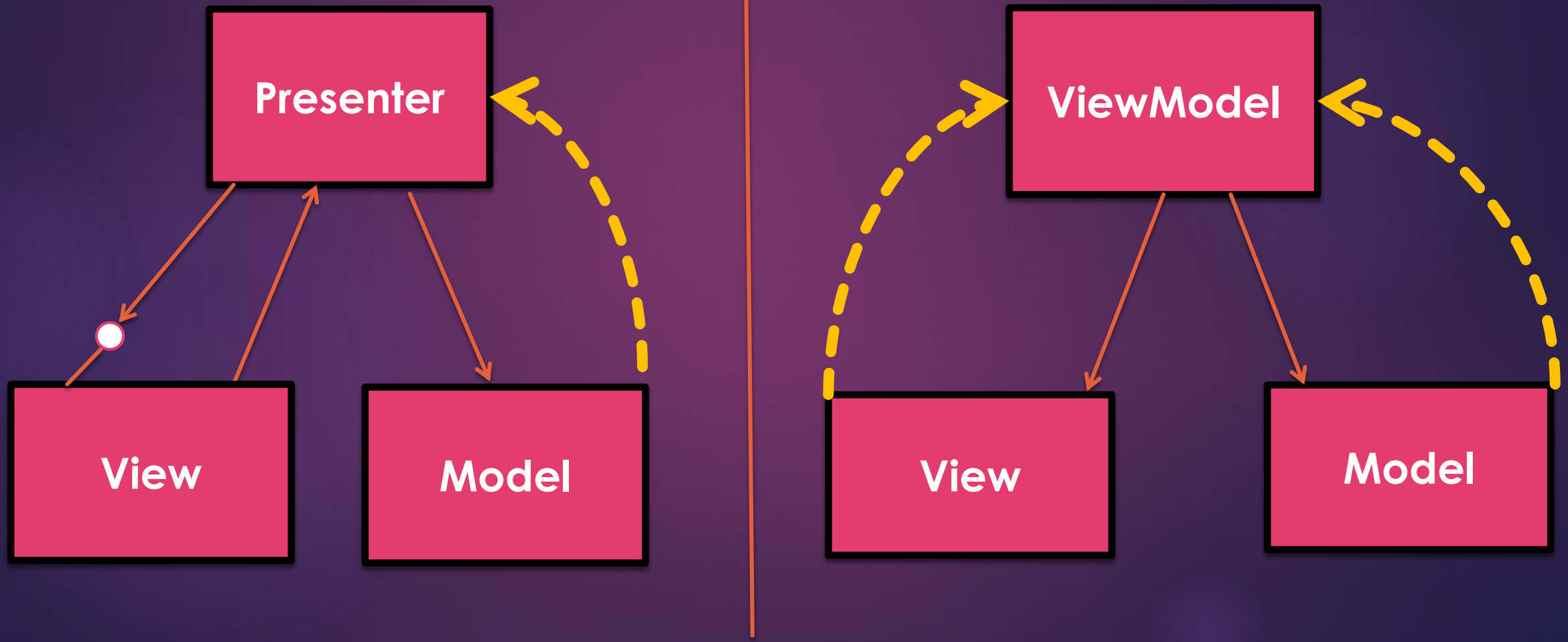
MVP – Supervising Controller Pattern



Model-View-ViewModel (MVVM)

- Largely based on MVC
- Specialization of the MVP pattern known as the Presentation Model
- Built specifically for the AngularJs.
- Model and View works just like MVC.
- ViewModel is a "Model of the View"
  - It extends the Model with Behaviors the View could use
  - Data Binding between View and Model
  - Passes commands between the View and Model

MVP (Passive) vs MVVM



Pattern Approach

MVC - Model View Controller

All input coming from user interaction, such mouse click, are directed to the Controller first.
The Controller then kick off some functionality.
A single Controller may render many different Views based on the operation being executed.
Also view doesn't have any knowledge of or reference to the Controller.
Controller interact with the Model and pass the model to the View.

MVP - Model View Presenter

MVP pattern looks very similar to MVC pattern, but has some key distinctions.
In MVP, the input is directed to the View not the Presenter.
For each View, there is a dedicated Presenter to render that View.
The View hold the reference to the Presenter.
The Presenter is also reacting to events being triggered from the View, so its aware of the View its associated with it. The Presenter updates the View with the data from the Model and vice versa. But he View is not aware of the Model

MVVM - Model View View Model

Like MVP pattern, the input is directed to the View.
View hold a reference to the ViewModel, but ViewModel is not aware of the View.
So it is possible to have one ViewModel to support many Views.
Also the Model is decoupled from the View.
ViewModel interact with the Model and feed the View with the data.

When To Use MV* Patterns

- Model-View-Controller (MVC) pattern
  - When updates need to be made to the application
  - Decoupling models and views.
  - Duplication of low-level model and controller code is eliminated across the application.
- Model-View-Presenter (MVP) pattern
  - Used most often in enterprise-level applications where it's necessary to reuse as much presentation logic as possible.
  - For applications with very complex views and a great deal of user interaction
  - When all of the complex logic should be encapsulated in a presenter, which can simplify maintenance greatly.

When To Use MV* Patterns

- Model-View-ViewModel (MVVM) pattern
  - When 2-way data-binding is needed (AngularJS).
  - Use this MVVM, where separation of the development of the GUI from the development of the business logic (back-end logic) is needed.

What To Use Where

- Model-View-Controller (MVC) pattern
  - Angular JS
  - .NET MVC3
  - Disconnected Web Based Applications
- Model-View-Presenter (MVP) pattern
  - Web Forms/SharePoint, Windows Forms
  - UI state logic already wired up (Backbone.js)
- Model-View-ViewModel (MVVM) pattern
  - Silverlight, WPF
  - Two way data-binding (AngularJS)

When Not To Be Used

- Model-View-Controller (MVC) pattern
  - Should not be used where unit testing is essential.
  - For applications with very complex views and a great deal of user interaction may find that MVC doesn't quite fit the bill here as solving this problem may mean heavily relying on multiple controllers

UI Libraries / Framework

2.1. jQuery

2.2. Angular Js

2.3. Backbone Js

2.4. Comparison of UI Libraries/Frameworks

2.5. Current state of JavaScript Frameworks

2.6. JavaScript Toolsets

jQuery

jQuery is a fast and concise JavaScript library and not a framework like AngularJs and BackboneJs.

- DOM manipulation - jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called Sizzle.
- Event handling - The jQuery offers an elegant way to capture a wide variety of events
- Animations - The jQuery comes with plenty of built-in animation effects which you can use in yourwebsites.
- Cross Browser Support - jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- Latest Technology – The jQuery supports CSS3 selectors and basic XPath syntax.

Advantages

- Ease of use for DOM traversal.
- Large library
- Apply styles to make UI more attractive
- Strong open source community. (Several jQuery plugins available)
- Great documentation and tutorials
- Simple to make AJAX calls.

Limitations

- Functionality maybe limited.
- JQuery javascript file required

Patterns Supported

- Doesn't support any specific pattern

AngularJs

- Powerful JavaScript based framework to create RICH Internet Applications(RIA).
- Provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.
- Application written in AngularJS is cross-browser compliant.
- AngularJS automatically handles JavaScript code suitable for each browser.
- Simplifies application logic by automatically updating views when application data changes, and vice versa
- Supports reusable widgets via directives
- Enforces a "no DOM manipulation" principle out of the box.

Advantages

- Powerful 2-way data binding
- "HTML is the template"
- Powerful ng-Directives.
- Specially built for Single Page Apps (SPA).
- Dependency Injection
- Good for Small → Medium → Large Apps

Limitations

- Slightly higher learning curve for developers
- Does not integrate well with Require.js (Asynchronous Module Loading)
- DOM based template (Everything loaded on startup)

Patterns Supported

- MVC
- MVVM

Backbone Js

Backbone supports MVP design pattern

- Lightweight, fast and less memory footprint
- Very flexible; it can work pretty much with any JavaScript templating framework
- Delegates DOM manipulation to separate libraries such as jQuery.
- Uses an Actor model to represent the driving behavior of the framework.

Patterns Supported

- MVP

Advantages

- Lightweight and easy to learn.
- Much faster than AngularJS.
- Strong community
- Plays well with almost any library and 3rd party APIs

Limitations

- Lack of structure out of the box
- Lack of data binding capabilities out of the box.
- Often need to write custom JavaScript to ensure that your templates(underscore js) are working exactly as they are supposed to.
- Requires additional libraries to work with any templates - jQuery (for DOM manipulation), Marionette (for nested views), etc.,

Comparison of UI Libraries/Frameworks

Overview

jQuery : Extensively simplifies using JavaScript on a website and it's lightweight as well as fast.

Angular Js - is used in toolset based on extending the HTML vocabulary for your application.

Backbone Js - Provides models with key-value binding and custom events, collections, and connects it all to your existing API over a RESTful JSON interface.

How fast running from home page link ?

jQuery: Fast

Angular Js - Very Fast.

Backbone Js – Average.

Third Party Integration

jQuery: Easy to integrate.
Angular Js - not possible. Mostly impossible..
Backbone Js - easy to integrate.

When to use

jQuery: When we need to easily manipulate the contents of a webpage. Best suited for DOM manipulation.

Angular Js - Could be very useful when working on Data Driven Applications.

Backbone Js - Could be very useful when we need to do heavy DOM manipulation.

Dependencies on other Javascript Framework

jQuery: It just like a Plug-in.

Angular Js - No dependencies.

Backbone Js - Depends on underscore js and Jquery.

Design Pattern

jQuery: NA
Angular Js - Can support the MVC and MVVM design patterns
Backbone Js - Can support MVP design pattern

Comparison of UI Libraries/Frameworks

Routing support

jQuery: Not supported.
Angular Js : Routing is very simple.
Backbone Js : Routing is very simple.

Views – stuff displayed in the screen

jQuery: NA

Angular Js : Uses Html as templating language and automatically pulls in Html templates via Ajax when needed.

Backbone Js : Simple and straight forward and easy for jquery and Dom skills developer. Handlebars can be used for templating.

Development tools

jQuery: NA
Angular Js - available.
Backbone Js - not available.

Data binding from server

jQuery : NA
Angular Js : Not using JQuery but can do with Angular's $http. For better solutions can use $resources.
Backbone Js : can use Jquery's $.ajax to power backbone. Very easy to understand..

Support for Testing Application

jQuery: Supports Unit Testing using Unit Test Runner.

Angular Js - Has fabulous test supports and Karma developed by Angular js team is popular test runner.

Backbone Js - Has no default test solutions. We can use Sinon.js and Jasmine third party solutions to test our self.

UI Toolset

Tool Type	Tool Name	jQuery	Angular Js	Backbone Js
Development Tools	Web Storm	Y	Y	Y
	AngularJsEclipse		Y	
	Sublime Text	Y		
	jsFiddle	Y		
Unit and behavioral testing	Jasmine		Y	Y
	Mocha		Y	
	Chai		Y	
	Karma	Y	Y	
	Sinon	Y	Y	Y
	Protractor		Y	

UI Toolset

Tool Type	Tool Name	jQuery	Angular Js	Backbone Js
Debugging Tools	Angular Batarang		Y	
	Ng-Inspector		Y	
	FireBug	Y	Y	Y
	Backbone Debugger	Y	Y	Y
	Backbone Eye			Y
	Marionette Inspector			Y
Functional Testing	Selenium	Y	Y	Y
	UFT	Y	Y	Y

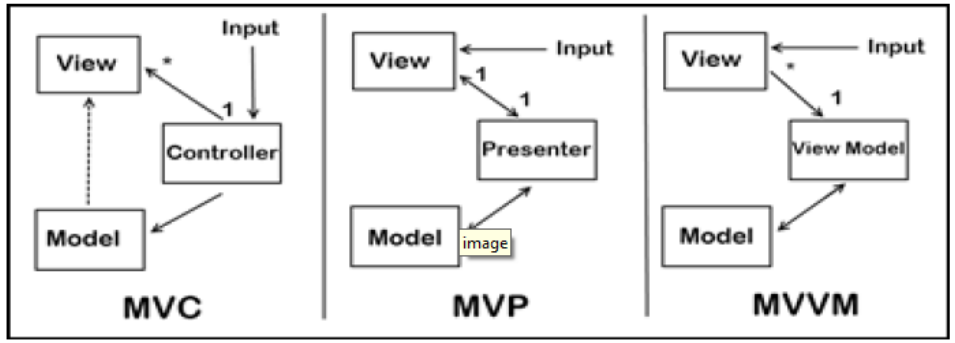
UI Toolset

Tool Type	Tool Name	jQuery	Angular Js	Backbone Js
Automated Code Quality	JsHint	Y	Y	Y
	SonarQube	Y	Y	Y
	BlanketJS			Y
Test Automation	PhantumJS	Y	Y	Y
	CasperJS		Y	
Continuous Integration	Jenkins	Y	Y	Y
Package Management	Bower		Y	Y
	NPM	Y	Y	Y

Appendix

http://trochette.github.io/Angular-Design-Patterns-Best-Practices/#/intro

Difference between MVP, MVVM, and MVC.



MVC	MVP	MVVM
The Input is directed at the controller	The Input begins with View	The input begins with model
Many-to-one relation between Controller and View	One-to-one mapping between view and its associated presenter	One-to-may mapping between various and one ViewModel
View does not have any knowledge about the controller	The View hold reference to the presenter; Presenter also reacting to the events being triggered from the View,so presenter does aware of the View	The View holds reference to the ViewModel;View Model has no information about the View.Here different technology can share viewModel (i.e) WCF and Silverlight can share same ViewModel.
Controller does pass the Model to the View,so View has knowledge about the controller	View is not Model aware	View is not Model aware

Current state of JavaScript Frameworks

- **MVC** – Frameworks strongly modeled after the Model-View-Controller pattern, with clearly identifiable components; also includes variations on the MVC concept such as MVP (Model View Presenter), MVVM (Model View ViewModel)
- **Component-based** – Strong orientation towards applications built out of standalone, reusable components that communicate with each other via events. Can be modelled after the W3C Component spec (Polymer, X-Tags) or a more liberal interpretation of components (Flight, React)

Estimated as relevant for <CLIENT> use cases and scenarios

Not relevant

Basic : Online Training

- http://www.w3schools.com/angular/
- http://www.learn-angular.org/
- https://www.codecademy.com/learn/learn-angularjs
- https://thinkster.io/a-better-way-to-learn-angularjs
- http://www.tutorialspoint.com/angularjs/
- https://www.codementor.io/learn-angularjs

Advanced : Online Training

- https://www.codeschool.com/courses/shaping-up-with-angular-js
- https://egghead.io/articles/new-to-angularjs-start-learning-here
- https://www.udemy.com/angularjs/
- https://www.udemy.com/learn-angularjs/
- https://docs.angularjs.org/tutorial/step_00
- http://www.thinkful.com/learn/angularjs-tutorial-build-a-gmail-clone/#Introduction