Client Side Javascript Frameworks

wifi:??

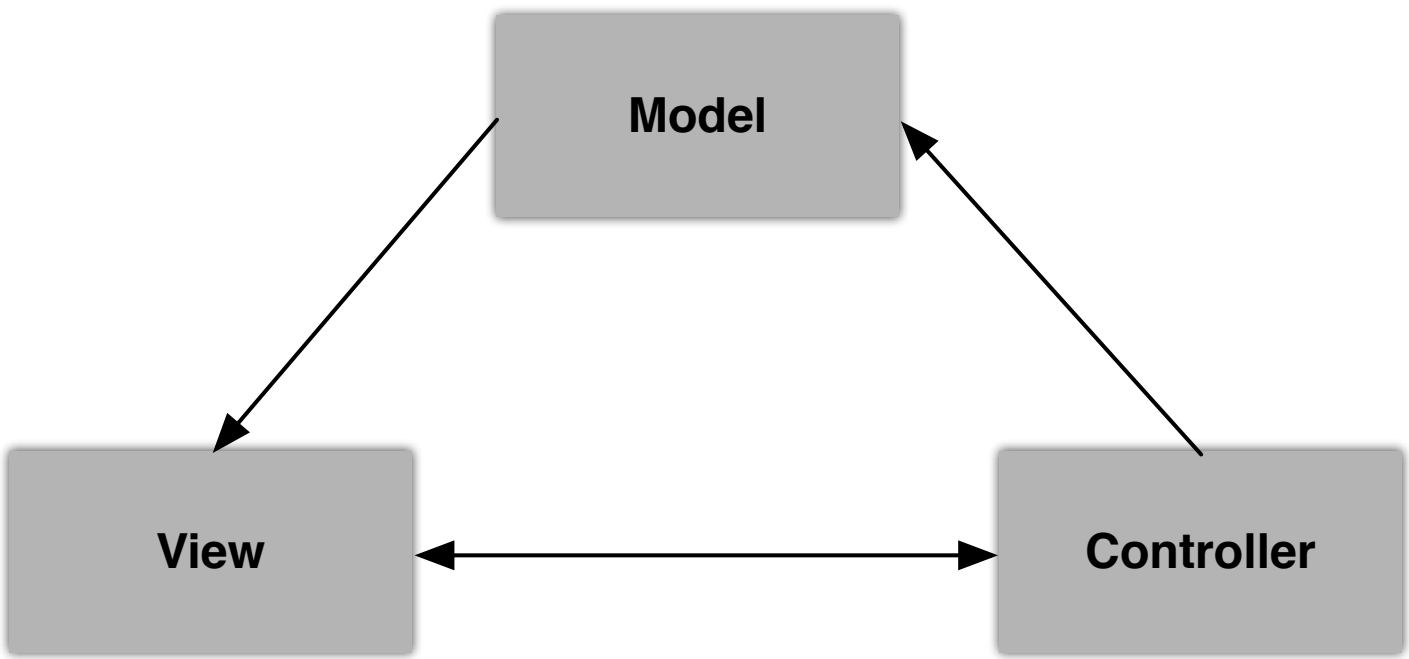
What is MVC?

A design pattern developed in the 70s which separates the representation of information from the user's interaction with it.

- Models
- Views
- Controllers

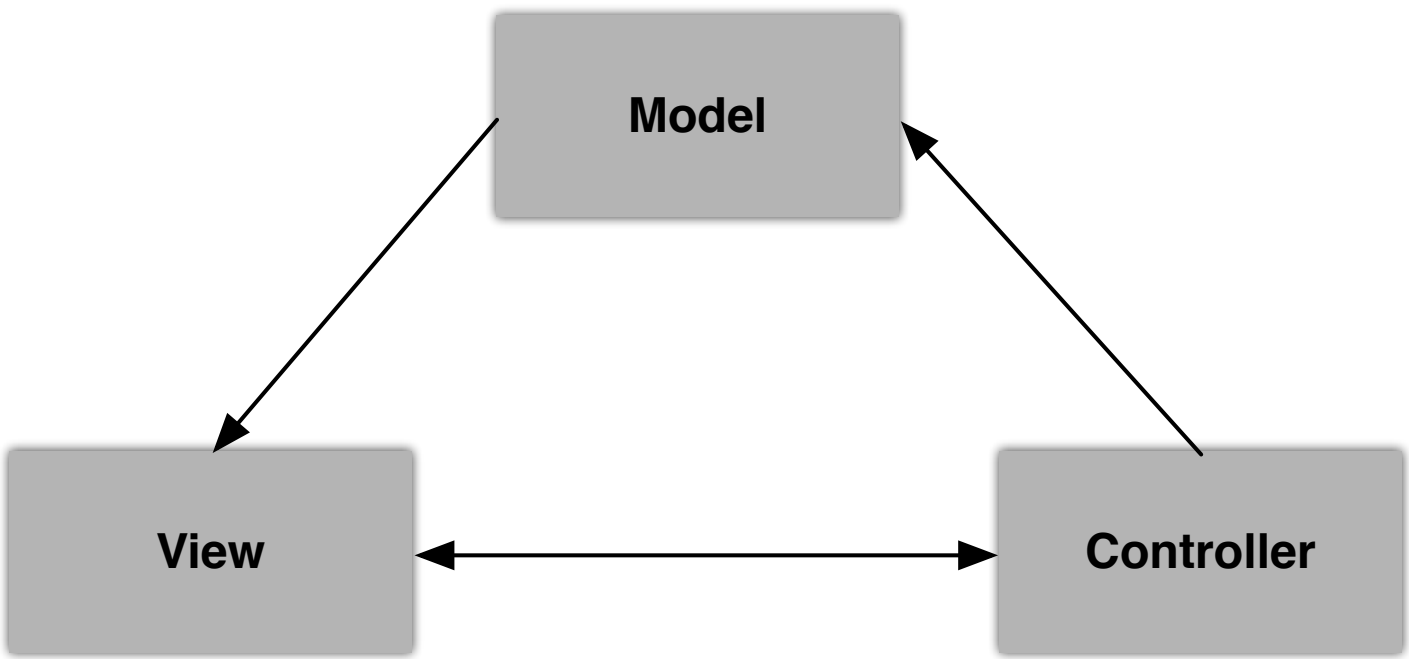
Models

- The atomized domain specific data of your application (User, photo, tweet)
- Notify observers of state/data changes



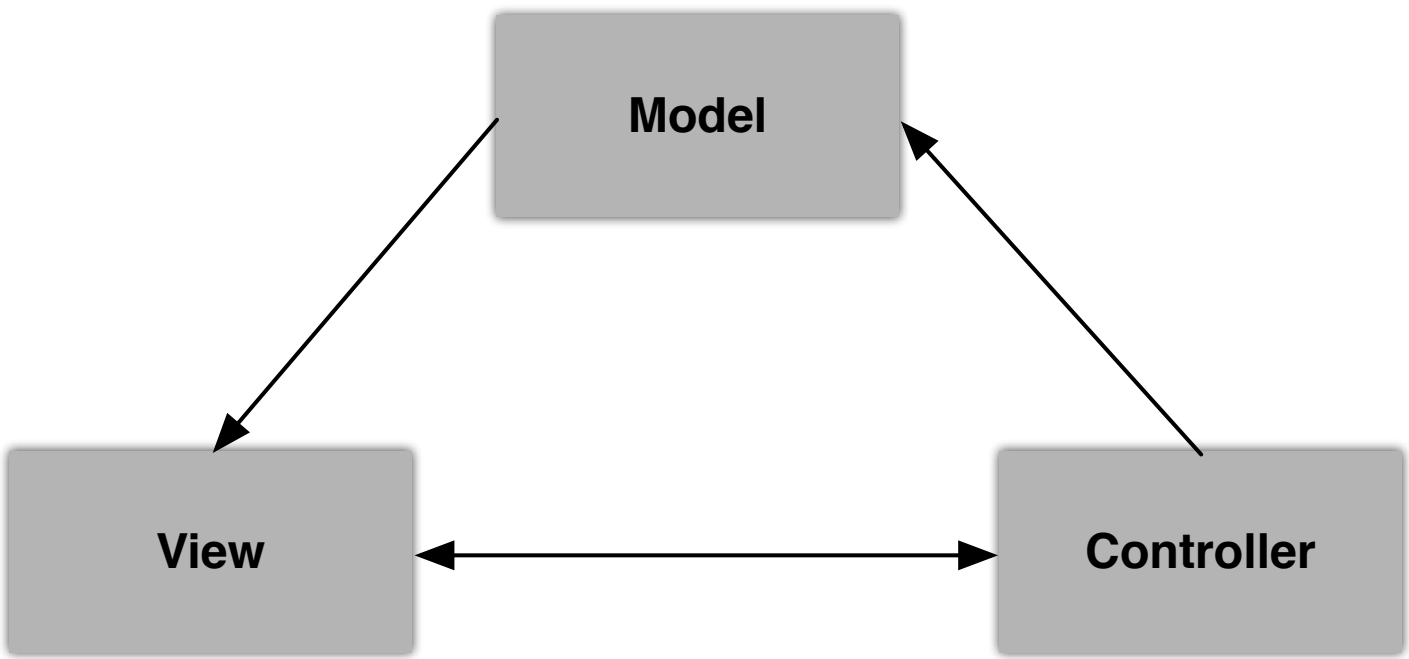
Views

- The UI/Presentation Layer
- Observe models and update UI accordingly



Controllers

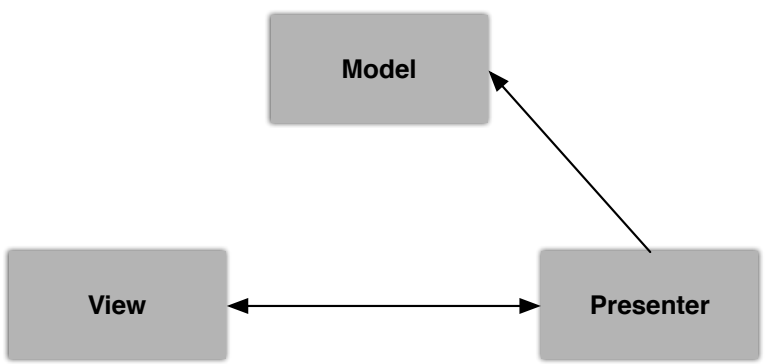
- Handles application and user events
- Manipulates the model

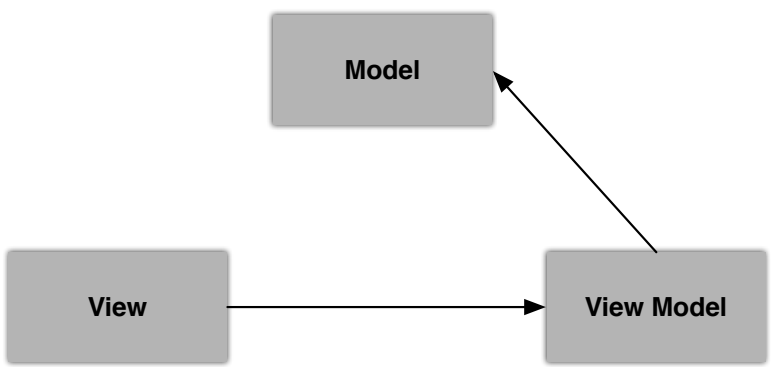


MVP & MVVM

- Model (Domain Objects)
- View (UI)
- Presenter (Application Logic)

- Model (Domain Objects)
- View (UI)
- View Model (Mediates between the Model and the View via data bindings)





http://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailmvcmvp

JavaScript MV*

- Models
- Views
- Collections
- Events
- Routers
- Controllers
- Services
- Factories
- Filters
- Providers
- Directives
- Objects

...

MV* is not (just) DOM Manipulation

- MV* is much much more than DOM querying & manipulation framework
- Some MV* have DOM frameworks baked in
- Others have dependancies and/or extensions

Benefits of MVC

- Better Code Organization & Reusability
  - Avoid Spaghetti Code
- Better for unit testing
- Help write large JS applications
- Baked in understanding of RESTful http calls

Keep Your Truth Out of the DOM

RESTful APIs

- Representable State Transfer
- Use HTTP methods
  - GET
  - POST
  - PUT
  - DELETE
- A base URI for the web API, such as `http://example.com/reources/`

http://apigee.com/about/api-best-practices/all/ebook

When to use MV*

- When building an Single Page Application that communicates with an API or backend data service, where much of the heavy lifting is done on the client
- gmail, git, dropbox

When not to use MV*

- When building and application that relies heavily on the server for view rendering, and javascript is used mostly for interactive behaviors (lightbox, sliders, galleries, etc)

Javascript MVC Crippled by choice?

- Backbone
- Ember
- Angular
- Knockout
- Batman
- Dojo
- Yui
- Spine
- + More every day

http://todomvc.com/

Some Criteria for Selecting a Framework

- What is the framework really capable of?
- Has the framework been proved in production?
- Is the framework mature?
- Is the framework flexible or opinionated?
- Have you really played with the framework?
- Does the framework have a comprehensive set of documentation?
- What is the total size of the framework?
- What dependencies does the framework have?
- Have you reviewed the community around the framework?

Backbone

- Models
- Views
- Collections
- Events
- Routers
- http://backbonejs.org/

Angular

- Modules
- Views
- Controllers
- Services
- Factories
- Filters
- Providers
- Directives

http://angularjs.org/

Ember

- Models - Objects
- Views
- Controllers
- Routers
- Templates via Handlebars

http://emberjs.com/

The Twitter app

Setting up Backbone

Backbone dependancies:

DOM query & manipulation: jquery - zepto

Templates & : Underscore, lowdash

```
<script src="../components/jquery/jquery.js"></script>
<script src="../components/underscore/underscore.js"></script>
<script src="../components/backbone/backbone.js"></script>
<script src="scripts/app.js"></script>
```

Twitter Search API

https://search.twitter.com/search.json?

```
created_at:
from_user:
from_user_id:
from_user_id_str:
from_user_name:
geo:
id:
id_str:
iso_language_code:
metadata:
profile_image_url:
profile_image_url_https:
source:
text:
```

The Model & The Collection

```
var Tweet = Backbone.Model.extend({});

var Tweets = Backbone.Collection.extend({

        model: Tweet,

        url:function(){

              return 'http://search.twitter.com/search.json?q=backbonejs';

        }

})
```

Refactoring

```
var app = app || {}

app.Tweet = Backbone.Model.extend({});

app.TweetsCollection = Backbone.Collection.extend({

        model: app.Tweet,

        url:function(){

              return 'https://search.twitter.com/search.json?callback=?&q=backbonejs';

        },

        initialize:function(){

              this.fetch();

        }

})

app.Tweets = new app.TweetsCollection;
```

Views

```
app.TweetView = Backbone.View.extend({
        el: '#tweets-list',
        initialize:function(){
              this.listenTo( app.Tweets, 'add', this.render );
        },
        render: function( tweets ){
              console.log( 'tweets => ',  tweets );
              return this
        }
})
app.Tweets = new app.TweetsCollection;
new app.TweetView();
```

Views pt2

```
app.TweetsCollection = Backbone.Collection.extend({

    ...

    parse : function( result ){

        return result.results

    }

})

app.TweetView = Backbone.View.extend({

        ...

        render: function( tweet ){

              $(this.el).append( '<div>' + tweet.get( 'text' ) + '</div>');

        }

})
```

_ Templates

```
<script type="text/template" id="tweets-template">
        <div class="tweet">
              <img src="<%= profile_image_url %>" />
              <h1><%= from_user_name %>
                 <a href="#/user/<%= from_user_id %>" class="from_user">
                    @<%= from_user %>
                 </a>
              </h1>
              <p><%= text  %></p>
        </div>
</script>
```

_ Templates

```
app.TweetView = Backbone.View.extend({

        ...

        render: function( tweet ){

              $(this.el).append( this.template( tweet.toJSON()  ));

        }

})


...

app.Tweets = new app.TweetsCollection;

new app.TweetView;
```

Multiple Views

```
app.AppView = Backbone.View.extend({

        el: '#twitter-app',

        initialize : function(){

              this.tweet_list = $( '#tweets-list' )

              this.listenTo( app.Tweets, 'add', this.addTweet );

        },

        addTweet: function( tweet ){

              var view = new app.TweetView({ model: tweet });

              this.tweet_list.append( view.render().el );

        }

})
```

Multiple Views

```
app.TweetView = Backbone.View.extend({
        template: _.template( $('#tweets-template').html() ),
        render: function( ){
              $( this.el ).append( this.template( this.model.toJSON()  ) );
              return this;
        }
})
```

Add Search

```
app.TweetsCollection = Backbone.Collection.extend({

    ...

    url : function(){

        return 'https://search.twitter.com/search.json?callback=?&q=' + this.query;

    },

    initialize : function(){//constructor

        this.query = 'backbonejs';

        this.fetch();

    }

    ...

})
```

Add Search pt 2

```
app.AppView = Backbone.View.extend({

    ...

    initialize : function(){

        ...

        this.search_field = $( '#search-field' );

        this.listenTo( app.Tweets, 'reset', this.clearTweets );

        ...

    },

    events:{

        'click #init-search' : 'initSearch'

    }
```

Add Search pt 3

```
app.AppView = Backbone.View.extend({

    ...

    clearTweets: function(){

        this.tweet_list.html('');

    },

    initSearch: function(){

        app.Tweets.reset();

        app.Tweets.query = this.search_field.val();

        app.Tweets.fetch();

    }

})
```

Routing

```
app.Router = Backbone.Router.extend({
        routes:{
              'user/:id':'userDetails',
        },
        userDetails: function( id ){
              console.log( id )
        }
})
...
app.TwitterRouter = new app.Router();
Backbone.history.start();
```

User View

```
app.UserView = Backbone.View.extend({

        template: _.template( $( '#user-template' ).html() ),

        events:{

              'click .close': 'hideMe'

        },

        render: function( ){

              $( this.el ).append( this.template( this.model.toJSON() ) );

              return this;

        },

        hideMe:function(){

              app.MainView.hideUser();

        }

})
```

User _ template

```
<script type="text/template" id="user-template">
        <div id="panel">
              <div id="user">
              <img src="<%= profile_image_url %>" />
              <p><%= from_user_name %></p>
              <span class="close">X</span>
        </div>
        </div>
</script>
```

User _ template

```
<script type="text/template" id="user-template">
        <div id="panel">
              <div id="user">
              <img src="<%= profile_image_url %>" />
              <p><%= from_user_name %></p>
              <span class="close">X</span>
        </div>
        </div>
</script>
```

Coffeescript?

- Javascript Preprocessor
  - (less to CSS or Jade to HTML)
- Additional learning curve
  - python and ruby devs will feel at home
- Potentially more difficult to debug

Coffeescript?

```
class AppView extends Backbone.View

        el: '#twitter-app'

        initialize: ->

              _.bindAll @

              @collection = Tweets

              @tweet_list = $ '#tweets-list'

              @search_field = $ '#search-field'

              @user_div = $ '#user-details'


              @listenTo @collection,'add', @addTweet

              @listenTo @collection,'reset', @clearTweets
```

Freebase API

https://developers.google.com/freebase/

Search:

https://www.googleapis.com/freebase/v1/search?filter=(all type:/film/film)&query=

Topic:

https://www.googleapis.com/freebase/v1/topic/m/

Explorer:

http://www.freebase.com/