Backbone.js in multipage apps

@iffyuva
codemancers.com

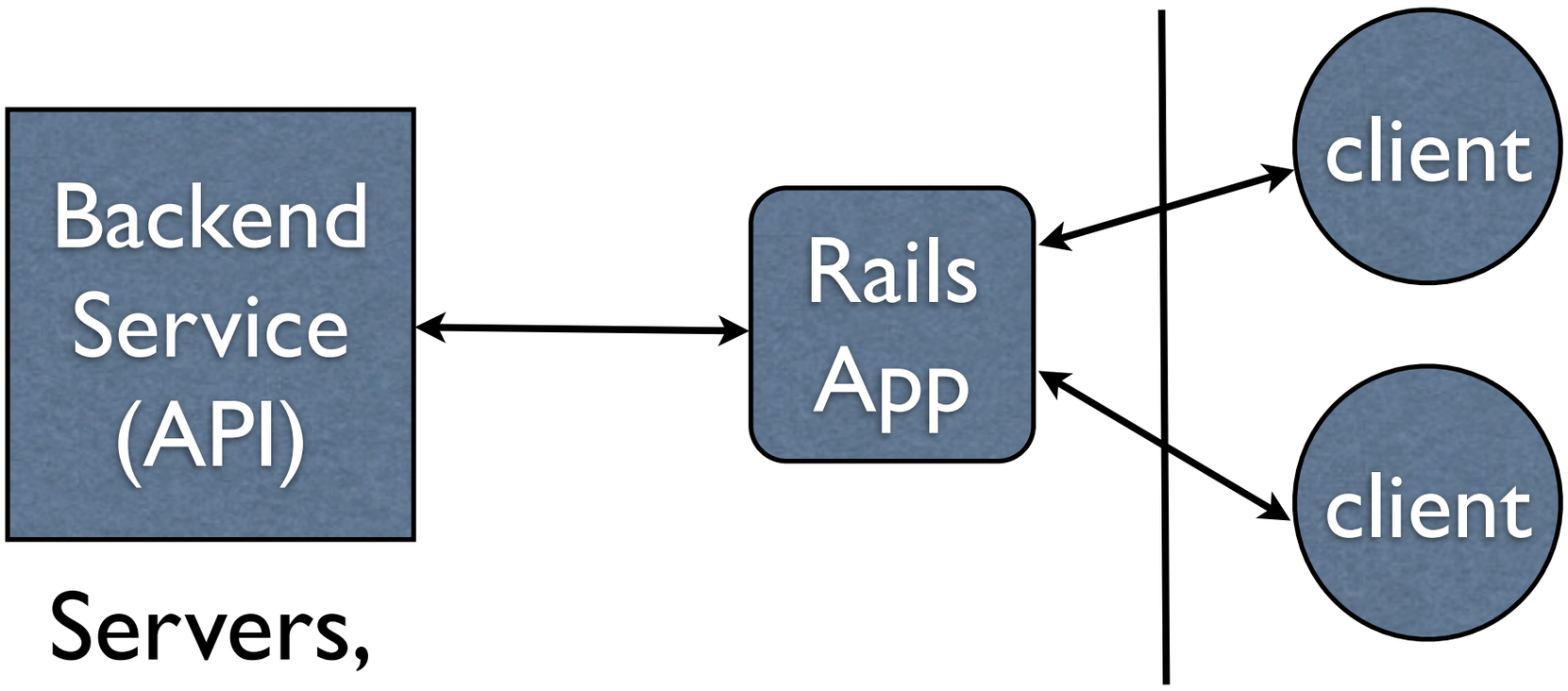
Backbone.js

- MV Library (Not a Framework)
- Single Page Apps
- Popular, Nicely Documented, ~1.5K Sloc
- underscore.js, json2, jQuery or Zepto

Backbone.js

- Models and Collections
- Views
- Events
- Routers
- History and Sync

Problem Statement



Servers,
Cloud IPs,
Firewalls,
Balancers

Problems Faced

- Filtering resources
- Costly trips to backend service
- Complex js code interacting with DOM
- Hard to maintain tests

Decisions

- Use backbone.js
- No strict memory management
- One router/master view per page

Resource Pages

- Tabular View
- Modals for create/update resource
- Pagination
- Filtering

List View

- Abstracts Table View
- Renders Collection
- Listens to Collection Events

```
class App.Views.ListView extends Backbone.View
  initialize: (options) ->
    # Here we bind collection events
    @collection.listenTo 'add',      'onAdded'
    @collection.listenTo 'remove',   'onRemoved'
    @collection.listenTo ‘paginate’, ‘onPaginate’

  addAll: () =>
    # Iterate the collection and call `@addOne`

  addOne: (model) =>
    # Render each model inside the table

renderEmpty: =>
  # Render message if the collection is empty

fetchCollection: (options)->
  # This is where the data is fetched through a collection

render: (options) =>
  # Call `@addAll` to start the rendering process
  this
```

Model View

- Renders model as a row in table
- Listens to model changes
- Edits/Destroys model

```
class App.Views.ModelView extends Backbone.View
  events: ->
    "click  .edit"    : "onEdit"
    "click  .destroy" : "onDestroy"

  initialize: ->
    # Here we bind or initialize model events
    @model.listenTo 'change', 'onChanged'

  onEdit: (e)=>
    # Handle displaying the edit form

  onDestroy: (e)=>
    # Handle deletion of a single model
```

Form View

- Renders New/Edit Forms
- No data bindings
- Pops up a modal
- ajax submission

```
class App.Views.ModalFormView extends Backbone.View
  # Since we were using twitter-bootstrap modal
  attributes:
    class: "modal hide fade"
    tabindex: "-1"
    "data-backdrop": "static"
    "data-keyboard": "false"

  events: ->
    "submit       form" : "onFormSubmit"
    "ajax:success form" : "onFormSuccess"
    "ajax:error   form" : "onFormError"

onFormSubmit: =>
  # It does client-side validation if required and then submits the
  # form using ajax.

onFormSuccess: (event, data)=>
  # Here we add the newly created resource to the collection.
  # For update, it updates the model attributes.
  # In both the cases the `ModelView` automatically renders the changes.

onFormError: (event, xhr, status, error) =>
  # Handles server-side errors
```

Paginator View

- Renders paginator view
- Handles pagination related changes
- Delegates actions to paginatable collection

```
class App.Views.Paginator extends Backbone.View
  events:
    'change #currentPage' : 'goToPage'
    'change #itemsPerPage': 'changeItemsPerPage'
    'click #previous'     : 'goToPreviousPage'
    'click #next'         : 'goToNextPage'

  minPageSize: 10

  goToPage: (e) =>
    @collection.goTo(page)

  changeItemsPerPage: (e) =>
    @collection.setPageSize(pageSize)

  goToPreviousPage: (e) =>
    @collection.previousPage()

  goToNextPage: (e) =>
    @collection.nextPage()

  render: ->
    # renders paginator
```

Paginatable Collection

- Extends Backbone.Collection
- Adds interface for pagination
- Slices models based on current page and per-page size
- Fires `paginate` event

```
class App.Collection extends Backbone.Collection
  nextPage: ->
    # calculate page and paginate
    @paginate()

  previousPage: ->
    # calculate page and paginate
    @paginate()

  goTo: (page) ->
    # calculate page and paginate
    @paginate()

  paginate: =>
    # calculate begin and end based on page and per-page
    @visibleModels = @models.slice(begin, end)
    @trigger('paginate')

  _addModel: (model) =>
    @paginate()

  _removeModel: (model) =>
    @paginate()
```

Realtime Updates

- Better UX
- Resources take time to create/destroy
- Resources have states
  - creating, created, active, deleting, deleted

Enters Faye

- pub-sub messaging system
- server side events
- backend service publishes resource events
- backbone subscribes to events and updates models and collections

Faye Connection

- Creates Faye Client
- Subscribes for all events. accountId is used for filtering
- Fires a formatted event

```
class App.Stream.Connection
  _.extend @prototype, Backbone.Events

  constructor: (options)->
    @client = new Faye.Client(url)
    @client.subscribe("#{accountId}", @onEvent)

  onEvent: (message)=>
    event = message.state + '.' + @guessResource(message.resource)
    @trigger(event, message)

  guessResource: (resource)->


jQuery ->
  App.streamConnection = new App.Stream.Connection(options)
```

Consuming Faye Events

- All collections register to faye connection
- On any event, collection checks for event nature
- If its create/delete, adds/removes resource resp.
- Otherwise, forwards event to resource

```
class App.Collection extends Backbone.Collection
  initialize: ->
    App.streamConnection?.on 'all', @onStreamEvent

  # Faye Events System Interface
  onStreamEvent: (streamEvent, message)=>
    # handles stream event, otherwise forwads it to resource
    # if present

  onStreamCreateEvent: (message)=>
    # fetches resource by id, and adds to collection

  onStreamDeleteEvent: (message)=>
    # finds resources by id, and deletes from collection
```

```
class App.Model extends Backbone.Model
  onStreamEvent: (streamEvent, message)->
    # handle event
```

Questions

Thank You