Durandal

Single Page Application Framework

vijay.shivu@gmail.com

What is durandal.js ?

- Framework of libraries and frameworks
- Takes advantage of well-known frameworks
- Simple objective... To make SPA or SPI

Where to use Durandal ?

- Web app that fits on a single page.
- It provides fluent UX by loading all necessary data on page load and then fetch additional data progressively
  - Means single server load
  - Multiple client side pages (or screens)
  - Not business, not security

Why SPA or SPI ?

- Reach
  - Devices, platforms, browsers
- Rich user experience
  - Fluent pages through client-side navigation
- Reduced round tripping

Features of SPA

- Deep client-side linking
- Load what's needed on page-load
- Progressively download when required
- Easy state maintenance
  - For a web app, traditional server side does not make sense anymore
  - Think of it as an app more then a web site
  - Example: phone application which fetches screens instead of data from an API

Getting Durandal

- Nuget
  - `Install-Package durandal`
- Bower
  - `bower install durandal`
- Mimosa
  - `mimosa skel:new durandal`
- Raw downloads:
  - durandaljs.com
  - github.com/bluespire/durandal
  - Visual Studio gallery: visualstudiogallery.msdn.microsoft.com

Tools of SPA in Durandal

REQUIRE.JS

AMD (Module loading)

Data binding

Knockout.

DURANDAL

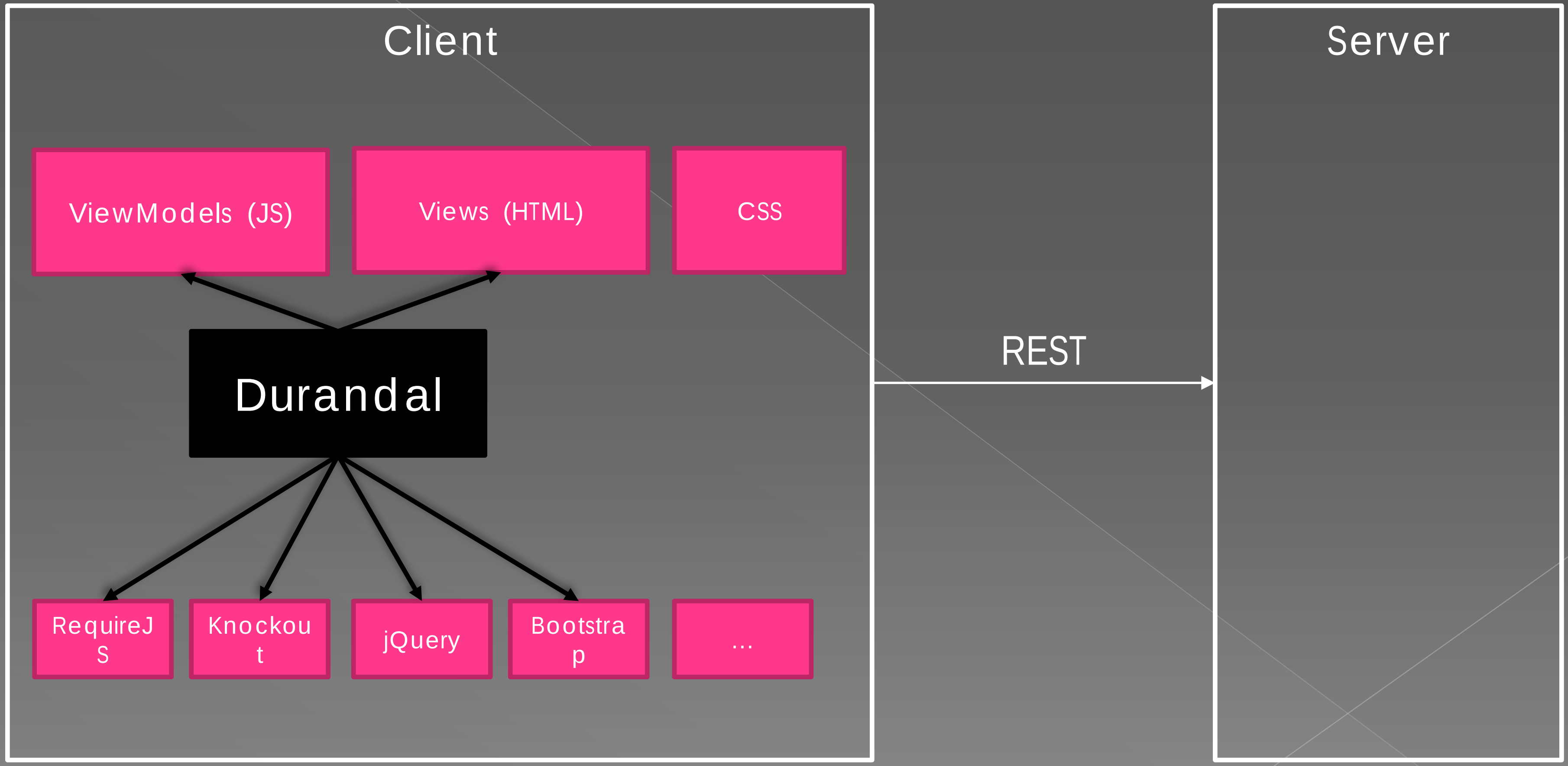
DOM manipulation

jQuery write less, do more.

UI (optional)

Bootstrap

Durandal Architecture



Keywords of Durandal

- Modularization
- Routing
- Binding
- Composition
- Lifecycle & Promises

Keywords of Durandal

- Modularization
  - Fixing JS “global”
- Routing
  - Deep linking
  - Backward navigation
- Binding
  - Solving DOM manipulation
- Composition
  - Object & view composition, user controls
- Lifecycle & Promises
  - Like an app asynchronous hooks

Built on module pattern

```
define(['jquery', 'knockout'],
    function ($, ko) {

    var loaddata = function() {
        $.ajax( ...);
    };

    var name = "myname";

    return {
        activate: loaddata,
        name: name
    };
});
```

Dependencies

Private

Public interface

AMD wrap

Binding with knockout

- Three binding types
  - Simple properties
  - ObservableArrays
  - Computed

Routing

- Client-side routing
- Deep linking
- URL parameters
- Route configuration

Composition

- Object composition
  - RequireJS and Module loading
- Visual composition
  - Durandal feature
  - Compose views + viewmodels inside other views

Lifecycle & promises

- Every page has “hooks” we can use to control behavior
- Lifecycle:
  - Deactivation
  - Activation
  - Binding
  - Composition

Lifecycle & promises

- Deactivation
  - canDeactivate()
  - deactivate()
- Activation
  - canActivate()
  - activate()
- Binding
  - binding()
  - bindingComplete()
- Composition
  - attached()
  - compositionComplete()
  - detached