



Intro to SPA framework

Modified from a presentation by

Jussi Pohjolainen

Rise of the Responsive Single Page App



[Image: http://johnpolacek.github.io/scrolldeck.js/decks/responsive/](http://johnpolacek.github.io/scrolldeck.js/decks/responsive/)

Responsive

- Unified across experiences
- Can be embedded as mobile app
- Better deployment and & maintenance
- Mobile users need to get access to everything

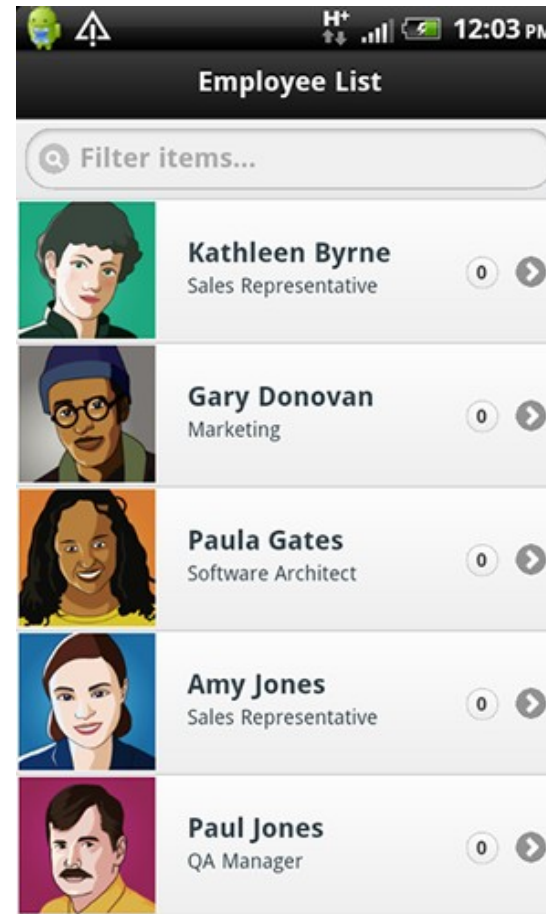


Image: <http://coenraets.org/blog/wp-content/uploads/2011/10/directory11.png>

Single--page Applications (SPA)

- Web app that fits on a **single web page**
 - Fluid UX, like desktop app
 - Examples like Gmail, Google maps
- Html page contains **mini--views** (HTML Fragments) that can be loaded in the background
- **No reloading** of the page,
- Requires handling of **browser history, navigation and bookmarks**

JavaScript

- SPAs are implemented using **JavaScript** and **HTML**

Challenges in SPA

- **DOM Manipulation**
 - How to manipulate the view efficiently?
- **History**
 - What happens when pressing back button?
- **Routing**
 - Readable URLs?
- **Data Binding**
 - How bind data from model to view?
- **View Loading**
 - How to load the view?
- Lot of coding! You could **use a framework instead ...**

Single-page Application

Single page apps typically have

- “application like” interaction
- dynamic data loading from the server-side API
- fluid transitions between page states
- more JavaScript than actual HTML

They typically do not have

- support for crawlers (not for sites relying on search traffic)
- support for legacy browsers (IE7 or older, dumbphone browsers)

SPAs Are Good For ...

- “App-like user experience”
- Binding to your own (or 3rd party) RESTful API
- Replacement for Flash or Java in your web pages
- Hybrid (native) HTML5 applications
- Mobile version of your web site













*The SPA sweet spot is likely not on web sites,
but on content-rich cross-platform mobile apps*

PJAX

Pjax is a technique that allows you to progressively enhance normal links on a page so that clicks result in the linked content being loaded via Ajax and the URL being updated using HTML5 pushState, avoiding a full page load. In browsers that don't support pushState or that have JavaScript disabled, link clicks will result in a normal full page load. The Pjax Utility makes it easy to add this functionality to existing pages.

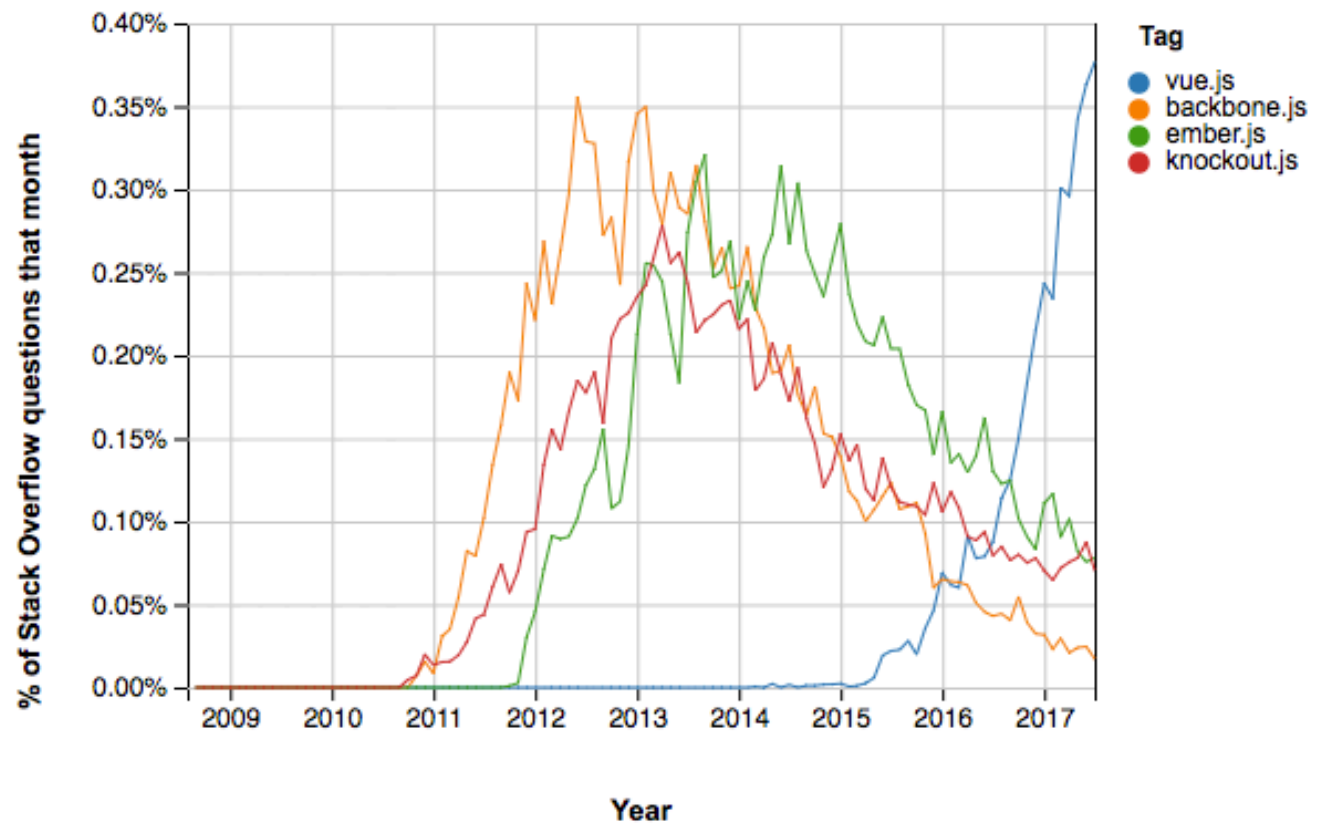
<http://yuilibrary.com/yui/docs/pjax/>

SPAs and Other Web App Architectures

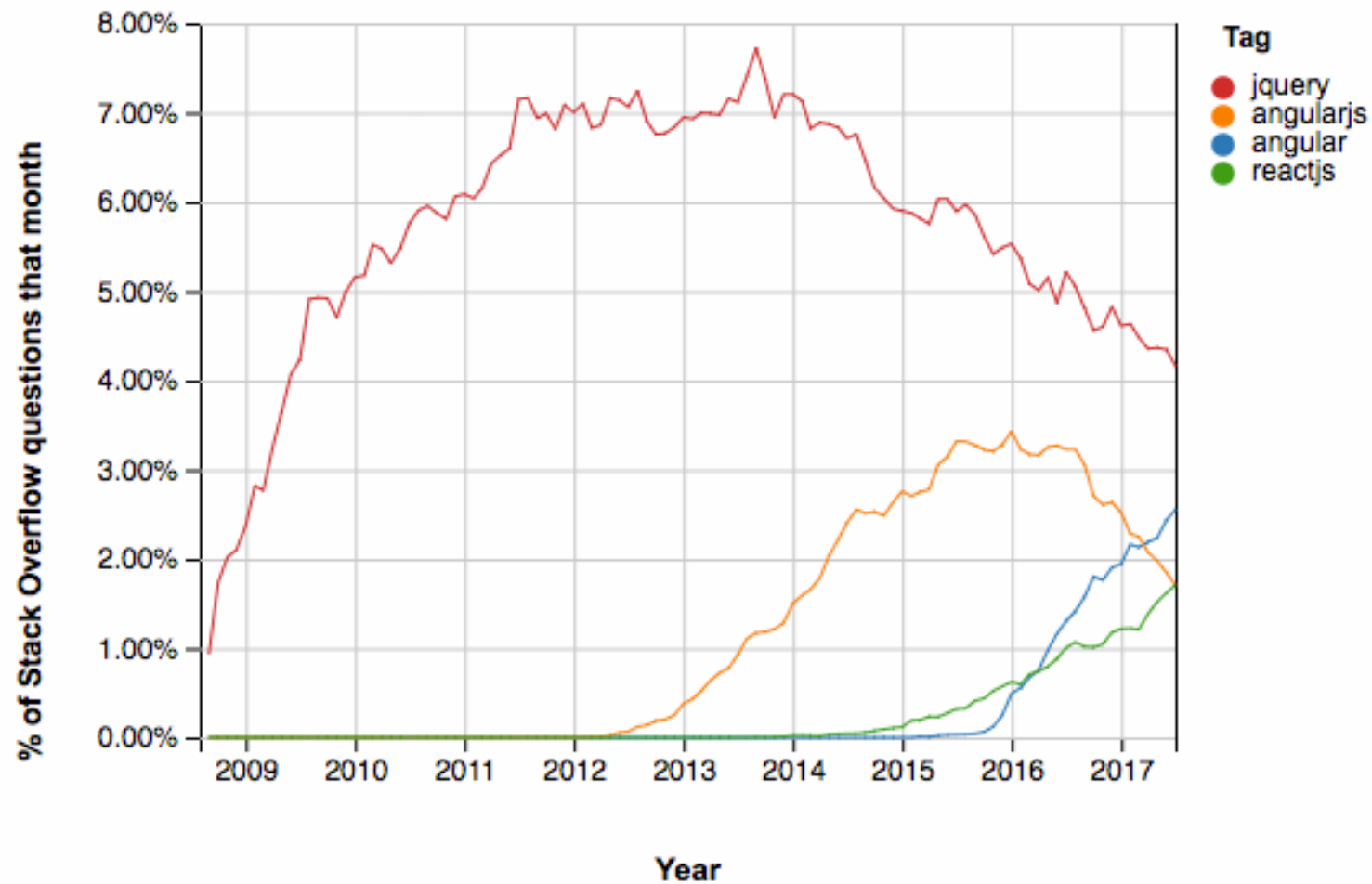
	Server-side	Server-side + AJAX	PJAX	SPA
What	Server round-trip on every app state change	Render initial page on server, state changes on the client	Render initial page on server, state changes on server, inject into DOM on client-side	Serve static page skeleton from server; render every change on client-side
How	UI code on server; links & form posting	UI code on both ends; AJAX calls, ugly server API	UI code on server, client to inject HTTP, server API if you like	UI code on client, server API
Ease of development				
UX & responsiveness				
Robots & old browsers				
Who's using it?	Amazon, Wikipedia; banks, media sites etc.	Facebook?; widgets, search	Twitter, Basecamp, GitHub	Google+, Gmail, FT; mobile sites, startups etc.

Lifecycle of new JS frameworks

There appears to be a **quick ascent**, as the framework gains popularity and then a slightly less quick but **steady decline** as developers adopt newer technologies. These lifecycles only last **a couple of years**.



Jquery, Angular JS, Angular, React



ANGULAR_JS

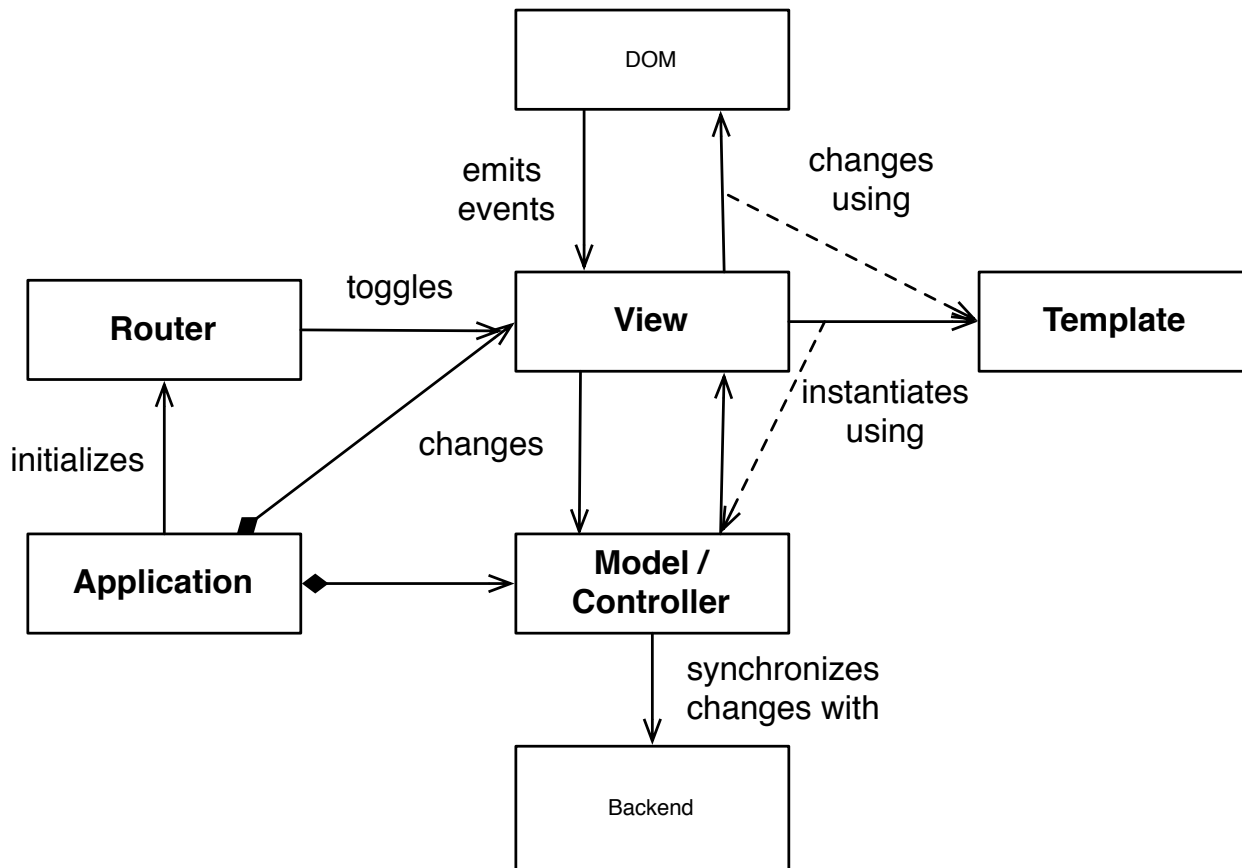
Angular JS

- **Single Page App Framework** for JavaScript
- Implements client-side **MVC** pattern
 - Separation of presentation from business logic and presentation state
- **No direct DOM** manipulation, less code
- Support for all major browsers
- Supported by Google
- Large and fast growing community

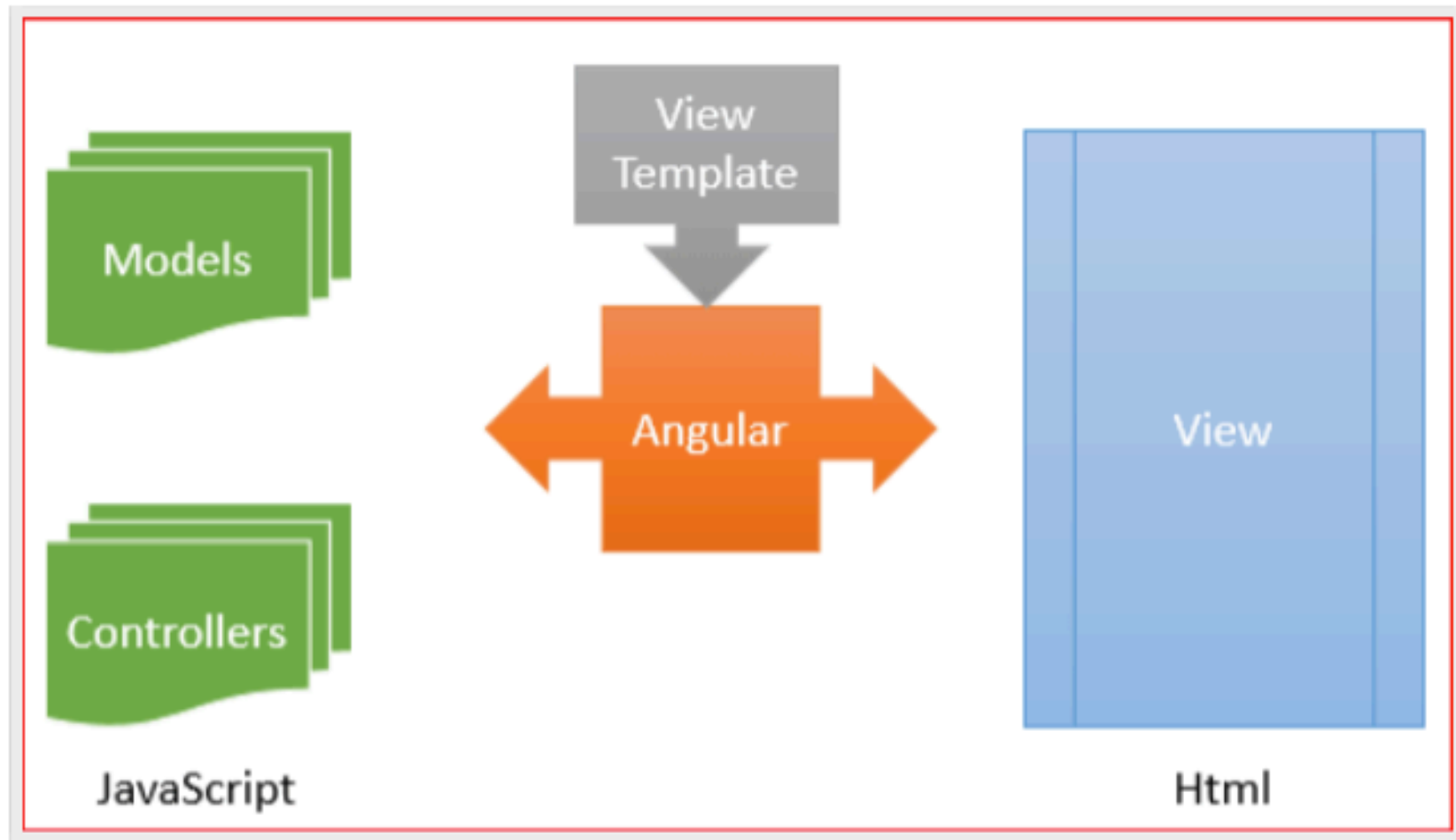
AngularJS – Main Concepts

- Templates
- Directives
- Expressions
- Data binding
- Scope
- Controllers
- Modules
- Filters
- Services
- Routing

Anatomy of a Backbone SPA

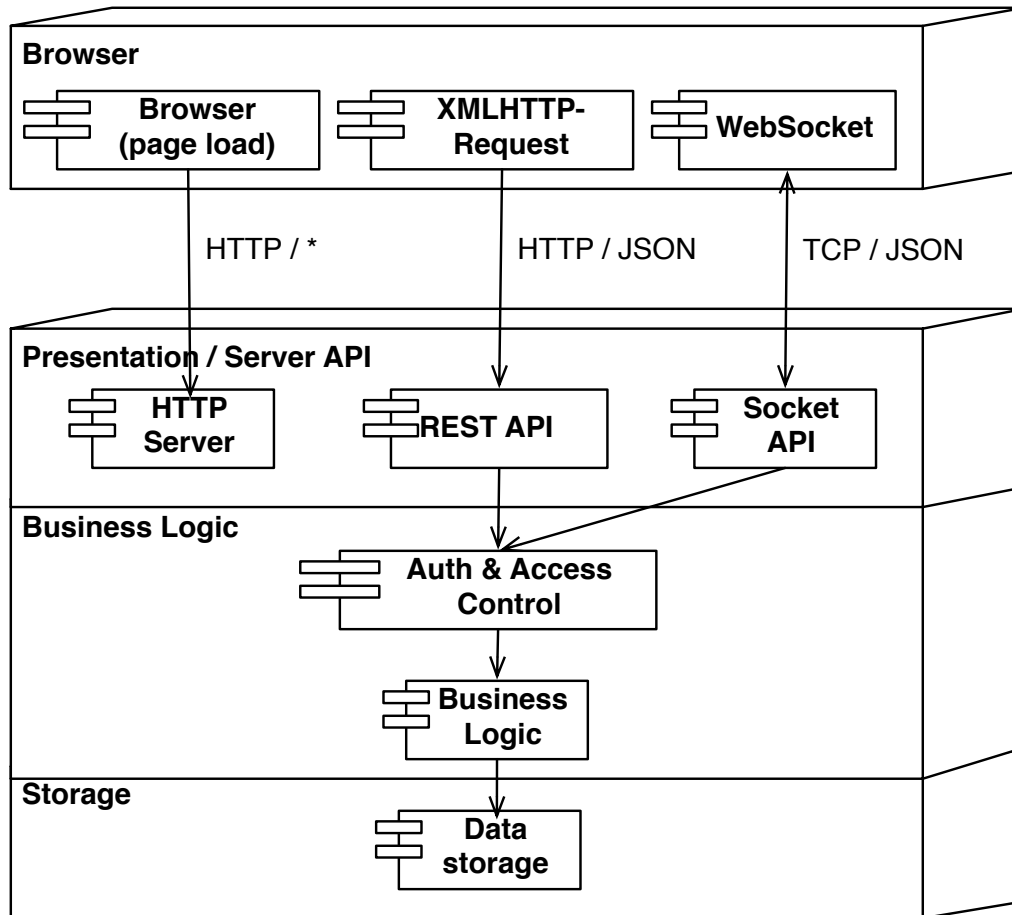


- Application as a 'singleton' reference holder
- Router handles the navigation and toggles between views
- Models synchronize with Server API
- Bulk of the code in views
- All HTML in templates



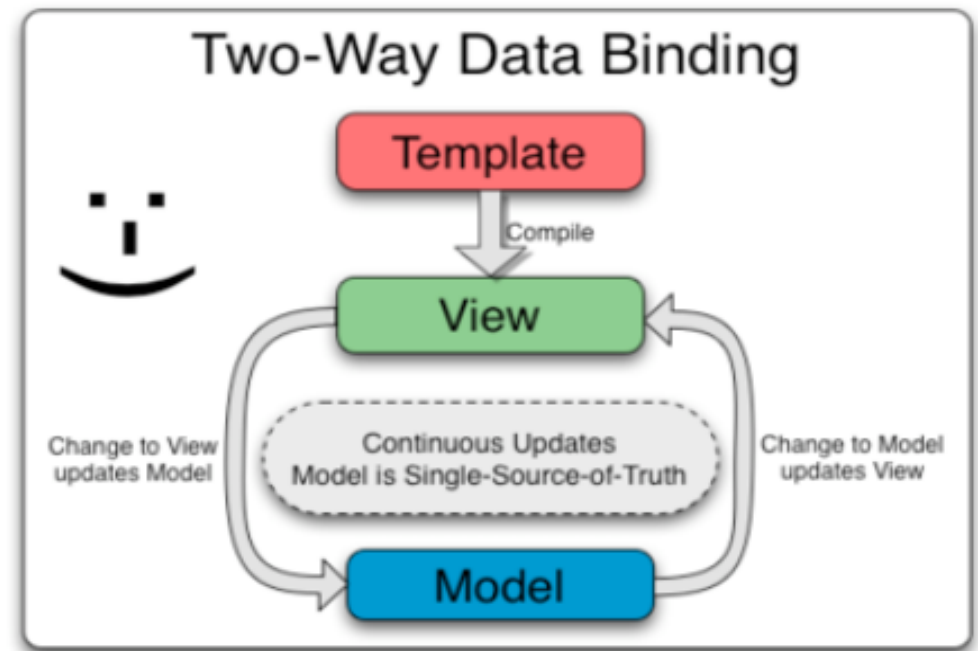
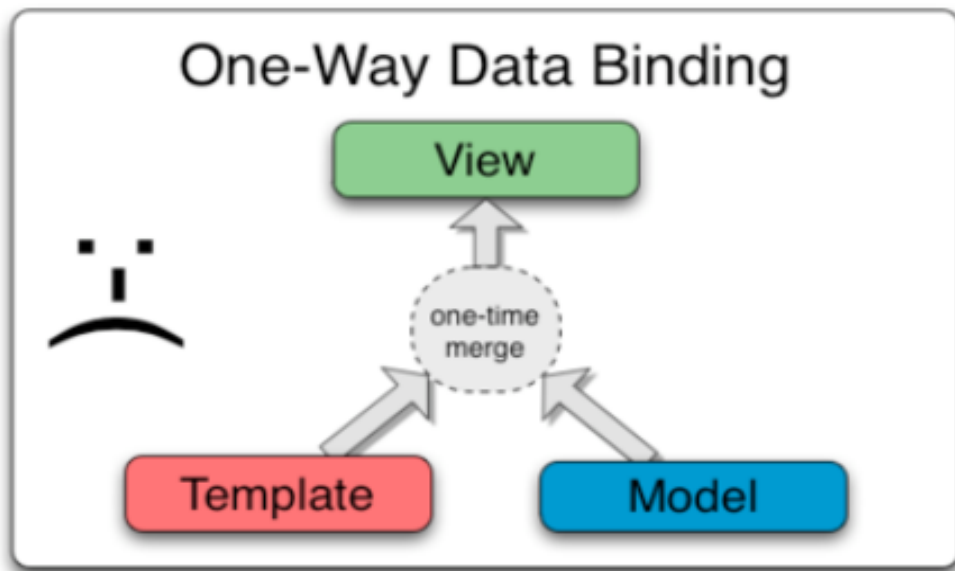
From Gary Arora

SPA Client-Server Communication

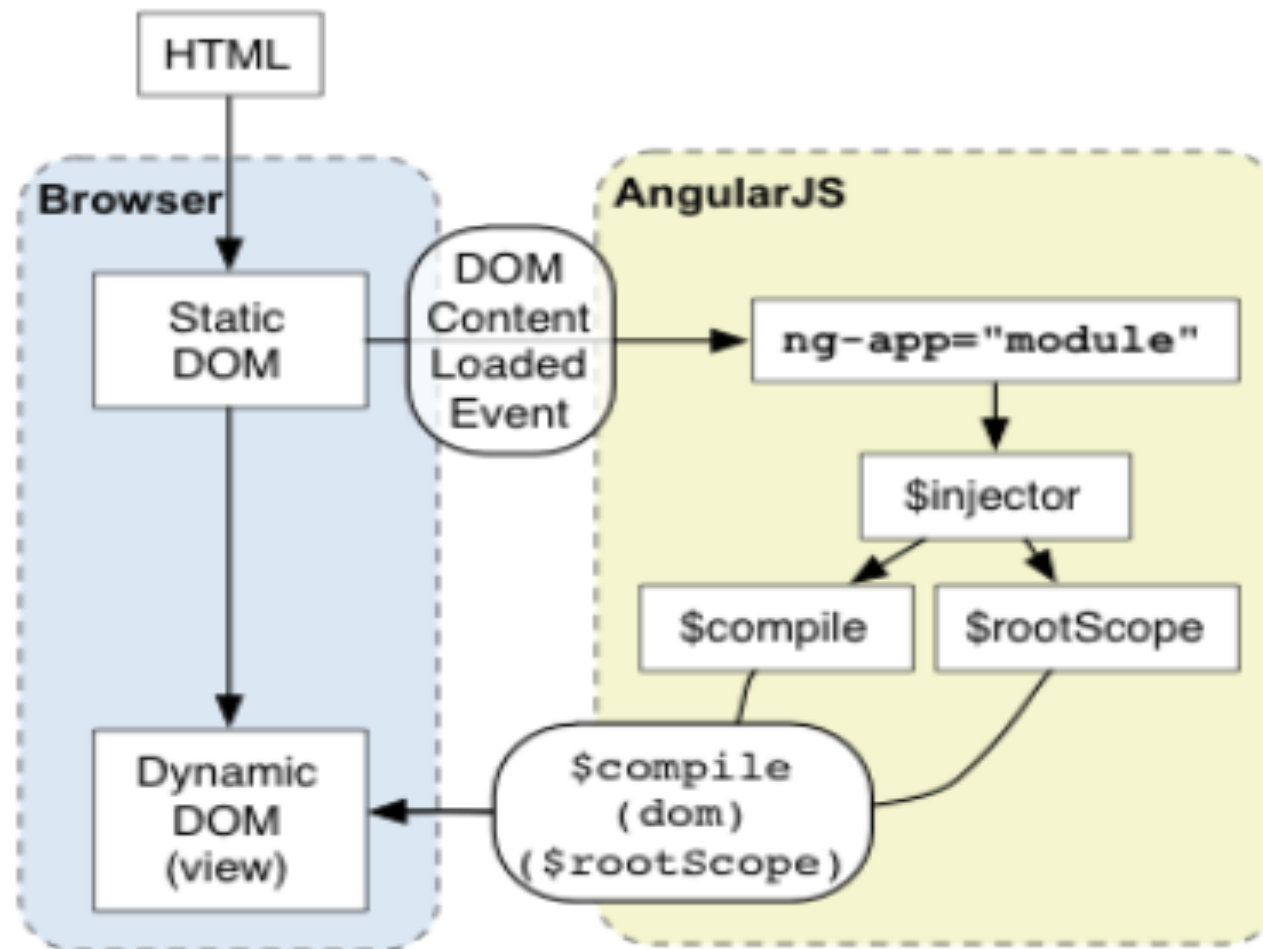


- HTML and all the assets are loaded in first request
- Additional data is fetched over XMLHttpRequest
- If you want to go real-time, WebSockets (socket.io) can help you
- When it gets slow, cluster the backend behind a caching reverse proxy like [Varnish](https://varnish.org/)

HOW IT WORKS?



HOW IT WORKS?



GETTING STARTED WITH ANGULAR_JS

Basic Concepts

- **1) Templates**
 - HTML with additional markup, directives, expressions, filters ...
- **2) Directives**
 - Extend HTML using `ng-app`, `ng-bind`, `ng-model`
- **3) Filters**
 - Filter the output: `filter`, `orderBy`, `uppercase`
- **4) Data Binding**
 - Bind model to view using expressions `{{ }}`

Name:

pippo

First Example – Template

Template

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title</title>
    <meta charset="UTF-8" />
    <style media="screen"></style>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angu
lar.min.js"></script>
  </head>
  <body>
    <div ng-app>
      <!-- store the value of input field into a variable name -->
      <p>Name: <input type="text" ng-model="name"></p>
      <!-- display the variable name inside (innerHTML) of p -->
      <p ng-bind="name"></p>
    </div>
  </body>
</html>
```

2) Directives

- **Directives** apply special behavior to attributes or elements in HTML
 - Attach behaviour, transform the DOM
- Some directives
 - **ng-app**
 - Initializes the app
 - **ng-model**
 - Stores/updates the value of the input field into a variable
 - **ng-bind**
 - Replace the text content of the specified HTML with the value of given expression

About Naming

- AngularJS HTML Compiler supports multiple formats
 - `ng-bind`
 - Recommended Format
 - `data-ng-bind`
 - Recommended Format to support HTML validation
 - `ng_bind`, `ng:bind`, `x-ng-bind`
 - Legacy, don't use

Lots of Built in Directives

- `ngApp`
- `ngClick`
- `ngController`
- `ngModel`
- `ngRepeat`
- `ngSubmit`
- `ngDb1Click`
- `ngMouseEnter`
- `ngMouseMove`
- `ngMouseLeave`
- `ngKeyDown`
- `ngForm`

2) Expressions

- Angular **expressions** are JavaScript--like code snippets that are usually placed in bindings
 - `{{ expression }}`.
- Valid Expressions
 - `{{ 1 + 2 }}`
 - `{{ a + b }}`
 - `{{ items[index] }}`
- Control flow (loops, if) are not supported!
- You can use **filters** to format or filter data

3) Filter

- With **filter**, you can **format or filter** the output
- **Formatting**
 - currency, number, date, lowercase, uppercase
- **Filtering**
 - filter, limitTo
- **Other**
 - orderBy, json

API Reference

<https://docs.angularjs.org/api/ng/filter/filter>

The screenshot shows the AngularJS API reference page for the 'filter' module. The browser address bar shows the URL <https://docs.angularjs.org/api/ng/filter/filter>. The page has a dark header with the AngularJS logo and navigation links: Home, Learn, Develop, and Discuss. A search bar is also present. Below the header, a breadcrumb trail reads: v1.3.0-build.3422 (snapshot) / API Reference / ng / filter components in ng / filter. The main content area is divided into two columns. The left column contains a sidebar with a list of filter names: filter, currency, date, filter (highlighted), json, limitTo, lowercase, number, orderBy, and uppercase. Below this list are sections for 'auto', 'service' (with \$injector and \$provide), 'ngAnimate' (with \$animateProvider), 'service' (with \$animate), 'ngAria' (with \$ariaProvider), and 'service' (with \$aria). The right column contains the main documentation for the 'filter' module. It starts with the title 'filter' and a subtitle '- filter in module ng'. There are two buttons: 'View Source' and 'Improve this Doc'. The description states: 'Selects a subset of items from array and returns it as a new array.' Below this is a 'Usage' section with two subsections: 'In HTML Template Binding' showing the syntax `{{ filter_expression | filter : expression : comparator }}`, and 'In JavaScript' showing the function call `$filter('filter')(array, expression, comparator)`. The 'Arguments' section contains a table with three columns: Param, Type, and Details.

Param	Type	Details
array	Array	The source array.
expression	string function()	The predicate to be used for selecting items from array . Can be one of: