

# Introduction to Backbone.js

{ Jim Cowart }

(Where the temp is a  
balmy 40 degrees) -->



# Who Am I?

(well - without the existential dilemmas involved)

- Jim Cowart (a.k.a - @ifandelse)
- Chief Architect at appendTo
- You might not care about this:  
<http://freshbrewedcode.com/jimcowart>
- But you might be interested in this:  
<http://github.com/ifandelse>



If you see this, be warned, my opinions has gotten loose & is probably causing a mess somewhere...

# Who I am Not...

(Hah - take \*that\* Heraclitus)

**I HEARD YOU WROTE  
A WEB APP...**

**...AND YOU DIDN'T USE  
BACKBONE.JS!**



# Who I am Not...

(Hah - take \*that\* Heraclitus)



SO...AFTER HE SAID HE  
DIDN'T USE BACKBONE

I LIT A MATCH

# Why are we here?

(Good grief - what's up with the existential dilemmas? 3 out of 4 slides already!)

- You're tired of poorly abstracted, spaghetti-laden JavaScript
- You're new to Backbone.js
- You're curious how Backbone.js could help

*“When working on a web application that involves a lot of JavaScript, one of the first things you learn is to **stop tying your data to the DOM**. It's all too easy to create JavaScript applications that end up as **tangled piles of jQuery selectors and callbacks**, all trying frantically to keep data in sync between the HTML UI, your JavaScript logic, and the database on your server. For rich client-side applications, a more structured approach is often helpful.”*

*-- <http://documentcloud.github.com/backbone/#introduction>*

# Web Apps, meet your Chiropractor:

## Backbone.js

- Dependencies:
  - underscore
  - jQuery/Zepto (if you use backbone.sync)
- \*Lots\* of active contribution
  - over 10k watchers
  - over 1400 forks
- Ample Documentation
  - annotated source
  - blogs/screencasts

# What's in a Web App?

DOM Scripting

Model-View-Binding

HTML Templating

Modularization & Dependency Management

OOP & Functional Programming

Building & Packaging

API Communication

Unit Testing

Application Structure

Lints

Routing & History

Documentation



# How does Backbone help?

DOM Scripting

HTML Templating

OOP & Functional  
Programming

API Communication

Application Structure

Routing & History

Model-View-Binding

Modularization & Dependency  
Management

Building & Packaging

Unit Testing

Lints

Documentation\*

- Backbone directly provides something
- Backbone might provide hooks or limited facilities
- At least backbone-related items will be well documented
- You're on your own

# Caveat Emptor

- Intentionally does not try to do 'all the things'
- Opinions focus on targeted concerns - no app-level organizational patterns
- Non-trivial apps will benefit greatly from helper frameworks + your own extensions



# Primary Abstractions

Backbone.Model

Backbone.Events

Backbone.Collection


Backbone.Router

Backbone.View

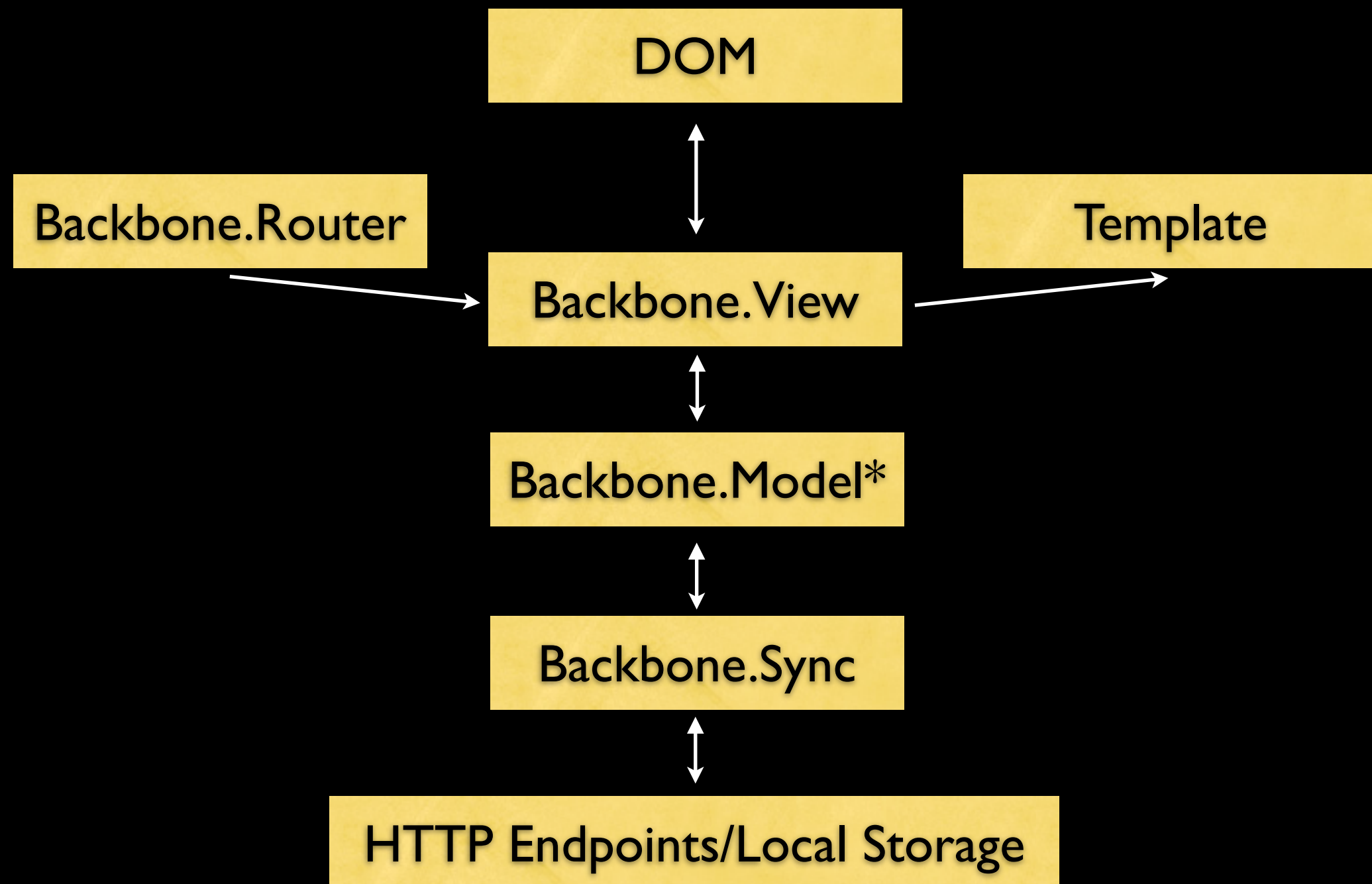
Backbone.History

Backbone.Sync

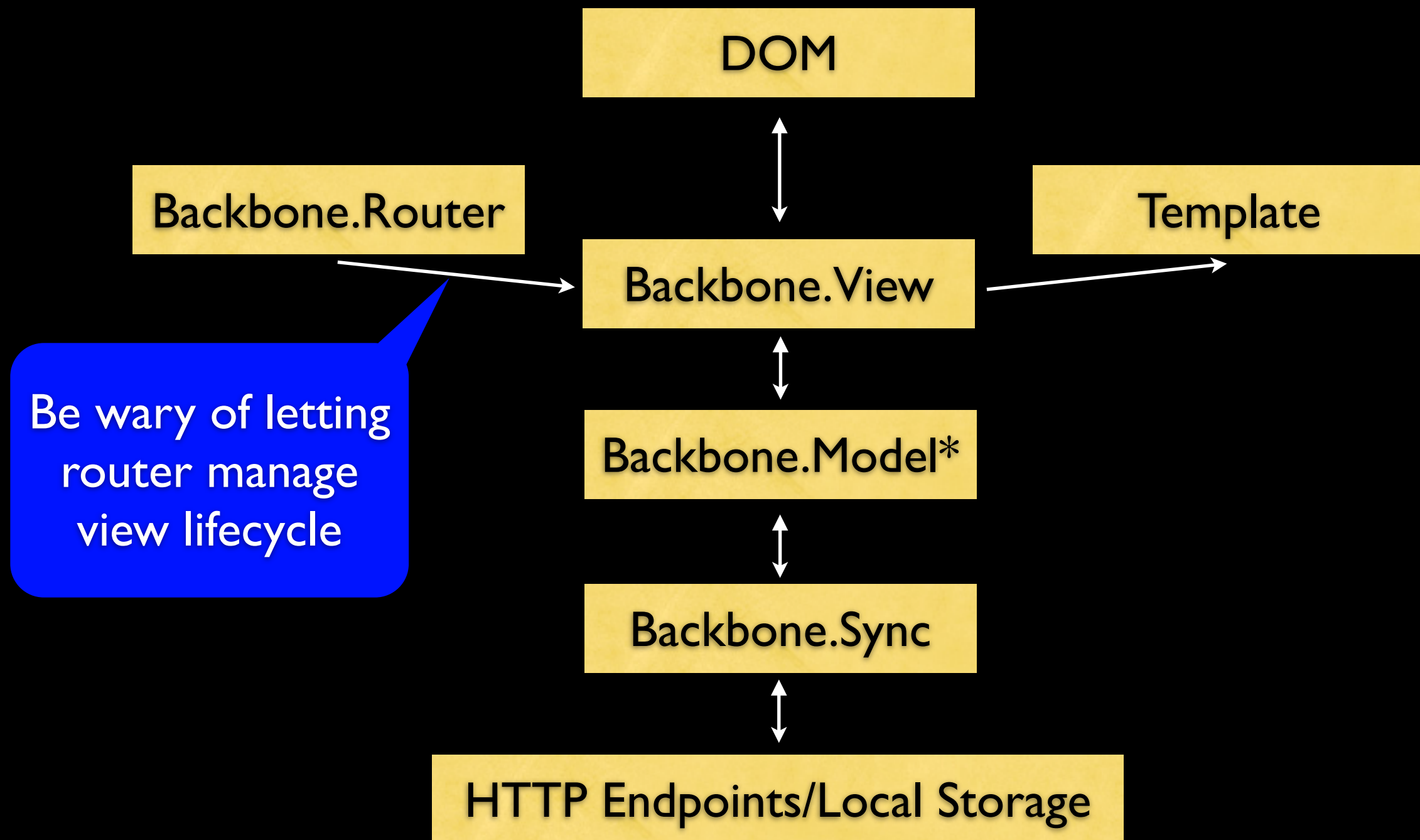
# Benefits

- Promotes good separation of concerns
- You retain control over what matters 
- Opinionated “enough” - but very extensible
- Plays well with jQuery, require.js and more

# The Obligatory Diagram

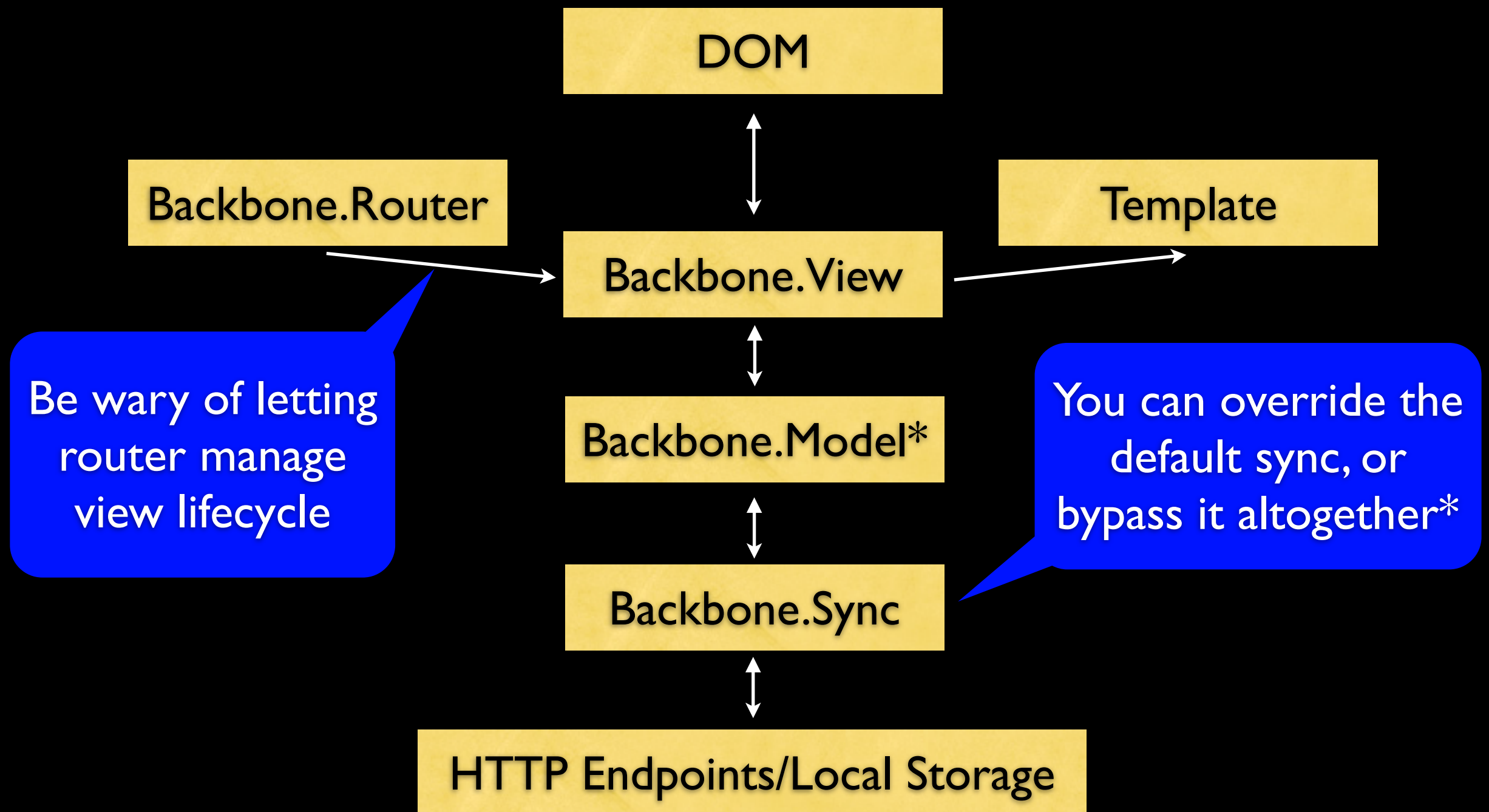


# The Obligatory Diagram





# The Obligatory Diagram



# Backbone.Events

- Provides Observer-Pattern based pub/sub
  - `on("Event", callback [, context])`
  - `off("Event" [, callback [, context] ])`
  - `trigger("Event", [ arg1, arg2....])`
- Used extensively by other Backbone objects

# Backbone.View

- Acts as a logical view or presenter
- Can be used with any templating library
- Keeps DOM scripting encapsulated and focused
- Nice event delegation conventions

{Code}

# Backbone.Model

- Effectively wraps a 'plain' model object
- Triggers events (changes, etc.)
- Handles 'CRUD' operations (via sync)
- Provides hook for validation

{Code}

# Backbone.Collection

- Effectively wraps an array of Backbone models
- Triggers events (add, remove, reset)
- Can fetch a collection (via sync)
- Provides rich functional API (via underscore)

# Backbone.Router

- Listens for hash/url changes and invokes handlers
- Supports HTML5 History, falls back to hash changes
- Uses Backbone.history instance under the hood.



# Time for {code} spelunking



**WARNING:** Code smells ahead....

# The Good

- Encapsulated views === addictive!
- Event delegation conventions (<3)



- Model-View eventing, a great start\*
- Route matching + param support
- Built-in backbone.sync is nearly painless\*

# The Gotchas

- Heavy/complicated model-view eventing gets verbose
- Backbone.sync overrides can be tricky/break idioms
- No higher level View Management
- Model validation validates *entire* model, not just attributes being set

# Helper Frameworks

- Backbone.Marionette

<https://github.com/derickbailey/backbone.marionette>

- Chaplin

<https://github.com/moviepilot/chaplin>

- Thorax

<https://github.com/walmartlabs/thorax>

- Layoutmanager

<https://github.com/tbranyen/backbone.layoutmanager>

- Aura

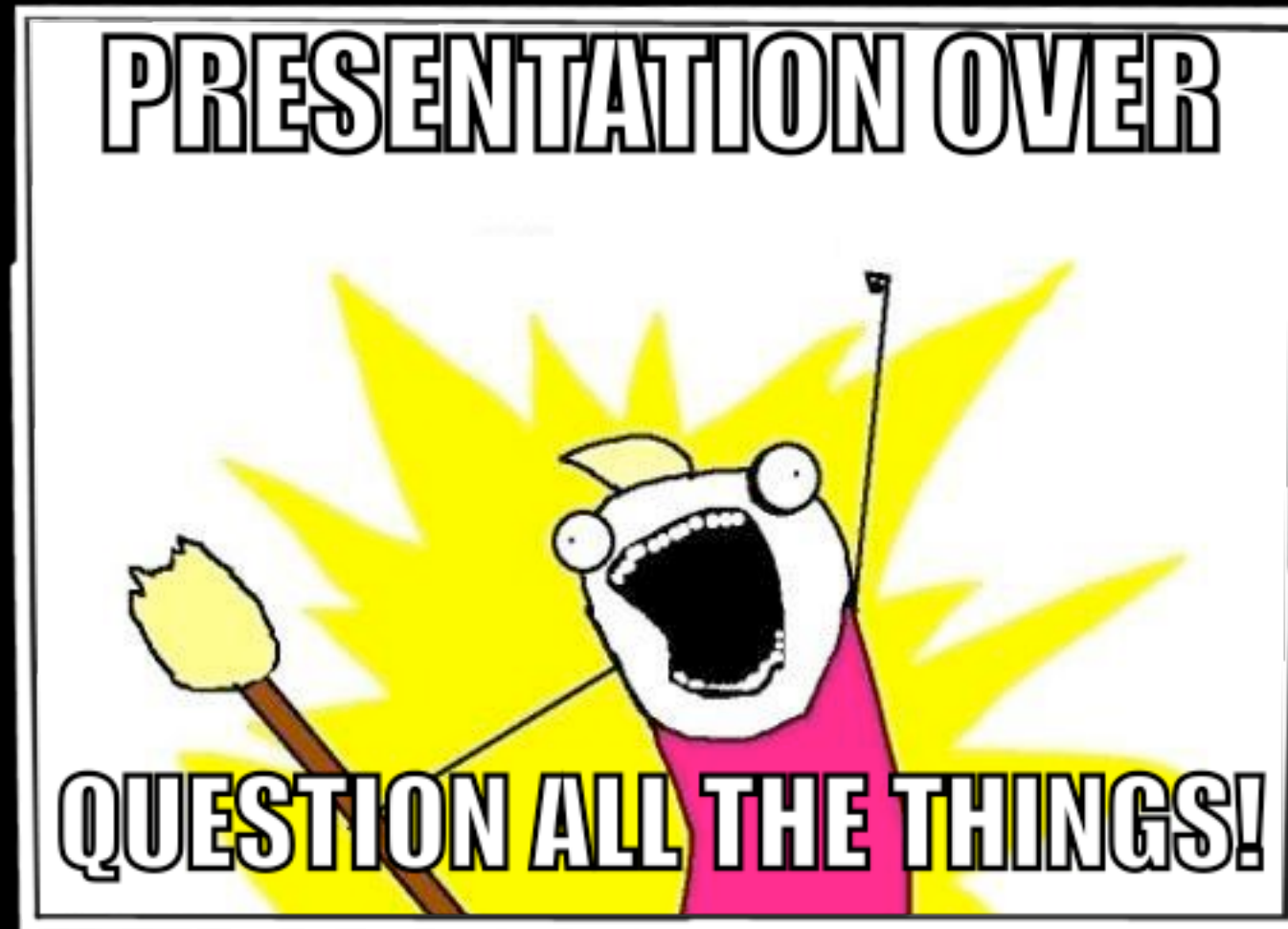
<https://github.com/addyosmani/backbone-aura>

- {Yours here} (Ship it!)

# Resources

- <http://documentcloud.github.com/backbone>
  - check out the annotated source!
  - Lots of example applications to review
  - <https://github.com/documentcloud/backbone/wiki>
- <http://backbonetraining.net/resources>
- <http://backbonefu.com>
- <http://backbonetutorials.com/>
- <http://www.quora.com/What-are-some-good-resources-for-Backbone-js>

Code/Slides for this presentation -  
<http://bit.ly/backbone-intro>



Q & A