

**Rails up your Backbone**



# Backbone.js

Frontend Javascript library

Models and views

Lightweight

**My one complaint:**  
**A couple fiddly bits**

**Step one**

# CoffeeScript



[theoatmeal.com](http://theoatmeal.com)

```
class window.Star extends Backbone.Moc
```

```
  defaults:  
    ships: 1  
    industry: 0  
    eta: null
```

```
  projected_ships: ->  
    @get("ships") +  
      (@get("eta") * @get("industry")) / 1  
  extra_ship_probability: ->  
    Math.round 100 * (@projected_ships  
      Math.floor @projected_ships())
```

**Step two**



# Relational models



[PaulUithol/Backbone-relational](#)

```
class window.Battle
  extends Backbone.RelationalModel

  relations: [{
    type: Backbone.HasOne
    key: "defender"
    relatedModel: "Star"
  }]

  defaults:
    defender:
      ships: null
      ws: 1

  defender_ships_remaining: ->
    @get("defender").get("ships") - 10
```

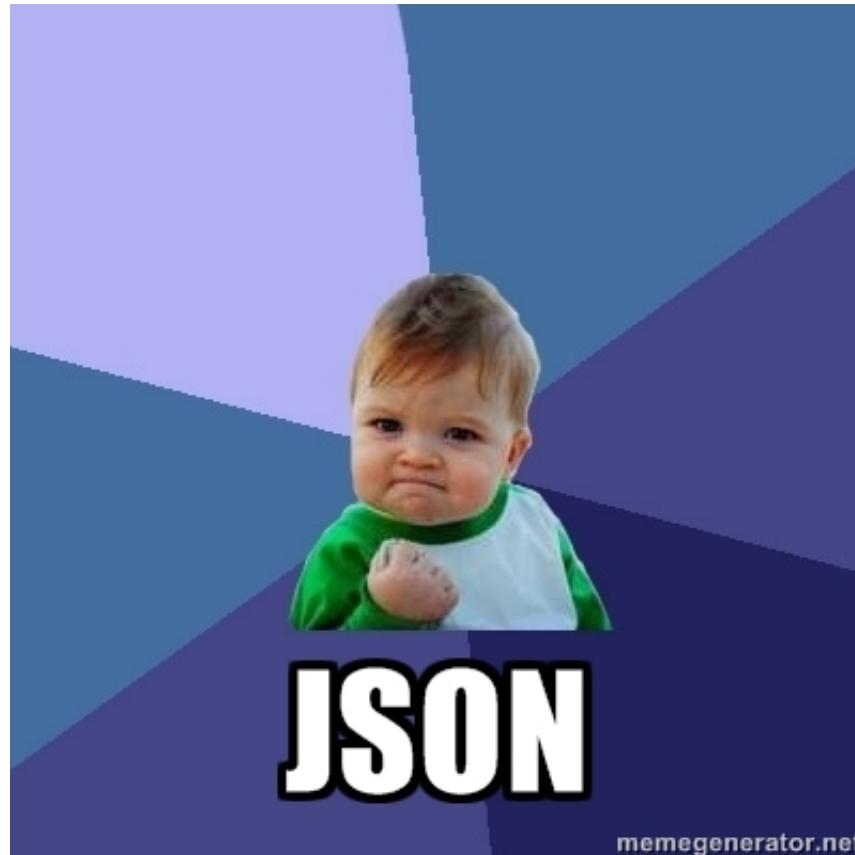
**Step three**

# jquery.formparams

[v3.javascriptmvc.com](http://v3.javascriptmvc.com)

[jupiterjs.com](http://jupiterjs.com)

```
# $('#myform').formParams()  
{  
  turn: 5  
  defender : {  
    ships: 25  
    ws: 18  
  }  
}
```



**Bringin' it all together**





```
class window.BattleView
  extends Backbone.View

  el: "form"

  events: {
    "keyup input": "updateModel"
  }

  updateModel: ->
    this.model.set(
      $("form").formParams()["battle"]
    )
```

**Sound familiar?**

```
class BattleController
  < ApplicationController

  def update
    @battle = Battle.find params[:id]
    @battle.update attributes(
      params[:battle]
    )
  end
end
```



[github.com/iangreenleaf](https://github.com/iangreenleaf)

[@iangreenleaf](https://twitter.com/iangreenleaf)

**Gems I've made recently**

# activerecord\_enum

[github.com/iangreenleaf/activerecord\\_enum](https://github.com/iangreenleaf/activerecord_enum)

# Meet MySQL ENUM

```
$ mysql> desc gemstones;
```

Field	Type	Null	Default
size	enum('small','large')	YES	NULL

```
$ mysql> insert into gemstones values ('large');
```

```
Query OK, 1 row affected (0.00 sec)
```



**Non-standard SQL?!?!**

**Well, yes**

# Not the only option

- A string column with an index
- `validates_inclusion_of`
- [nofxx/symbolize](#)

# noFX/symbolize

```
class Gemstone
  symbolize :size, \
    :in => [:small, :large]
end
```

```
@gemstone.size
# => :small
```

# Why use ENUM?

- Performance(...)
- DB structure
- Other codebases
- Legacy support

```
# schema.rb
create table "gemstones" do |t|
  t.column "size", "enum('small','large')"
end
```

**So far, so good**

\$ rake something

**But wait...**



```
diff --git a/db/schema.rb b/db/schema.  
index 0ea17e4..d4a2cf9 100644  
--- a/db/schema.rb  
+++ b/db/schema.rb  
@@ -53,16 +53,16 @@  
- t.column "size", "enum('small','large'  
+ t.string "size", :limit => 0
```

**Arrrrrrrggggh**

# Solution

Tell ActiveRecord about enums

```
# schema.rb
create_table "gemstones" do |t|
  t.enum "size", :limit => ["small", "large"]
end
```

# **Works with other database adapters, too**

So please put the weapons down

# Does

- Rails 3.0.x
- mysql2, sqlite
- ENUM and SET types

# Should add

- Rails 3.1.x
- Other DB adapters

# active\_set

[github.com/iangreenleaf/active\\_set](https://github.com/iangreenleaf/active_set)



**Meet MySQL SET**

```
$ mysql> desc balloons;
```

Field	Type	Null	Key	Default
gas	set('helium','hydrogen')	YES		NULL

```
$ mysql> insert into balloons values ('helium');
```

```
Query OK, 1 row affected (0.00 sec)
```

```
$ mysql> insert into balloons values ('helium,hydrogen');
```

```
Query OK, 1 row affected (0.00 sec)
```

```
$ mysql> select count(*) from foo where \
    find_in_set('hydrogen',gas);
```

count(*)
1

```
1 row in set (0.00 sec)
```

**Non-standard SQL?!?!?**

**Well, yes**

**...I was on a roll**

```
class Balloon < ActiveRecord::Base
  acts_as_set :gasses, ["helium", "hydroge
end
```

```
Balloon.create :gasses=>["helium", "hydrog
```

# Remember

Use with `active_record_enum` to keep your schema clean

# Does

- Not much

# What should I add?

- Probably validations
- Support `find_in_set(?)`
- Does anyone even use these?



# Bonus slide

```
$ mysql> desc my_table;
```

Field	Type	Null	Key	Default
special	set('yes','no')	YES		NULL

# tap\_out

[github.com/iangreenleaf/tap\\_out](https://github.com/iangreenleaf/tap_out)

# Problem

I have a project with Ruby + PHP code.

The phpunit tests never get run.

# Solution

Run phpunit tests from rake.

# Extra mile

Hook directly into Test::Unit

#### DiscountCodeTest

test_can_create_a_code	PASS
test_can_delete_codes	PASS
test_can_filter_codes_by_school_id	PASS
test_code_gets_generated_automatically	PASS
test_codes_must_be_unique	PASS
test_creates_charge_if_no_adjustment_exists_for_student	PASS
test_expiration_is_required	PASS
test_no_adjustment_is_created_is_some_exist_for_a_student	PASS
test_once_set_code_does_not_change	PASS
test_return_all_if_school_id_is_blank	PASS
test_student_is_required	PASS
test_valid_scope	PASS
test_values_are_set_for_category	PASS

#### DueDateTest

test_on_or_after_scope	PASS
test_on_or_before_scope	PASS

#### DbConnectionTest

testGetSingleton	PASS
testReturnsAmcoDbStatements	PASS
testQuery	PASS
testFetchTypes	PASS
testExec	PASS
testPrepareExecute	PASS
testFetchAll	PASS
testFetchDefaultParams	PASS
testFetchColumn	PASS
testOrderedPreparedStatement	PASS
testNamedPreparedStatement	PASS
testArrayFromQuery	PASS
testSimpleArrayFromQuery	PASS
testSingleValueFromQuery	PASS
testTransactionRollback	PASS
testTransactionCommit	PASS
testTransactionInProgress	PASS
testQueryResultIterator	PASS
testBadQuery with data set #0	PASS
testBadQuery with data set #1	PASS
testBadQuery with data set #2	PASS
testBadQuery with data set #3	PASS

# Test Anything Protocol

```
$ phpunit --tap .
TAP version 13
ok 1 - ValidationTest::testBadPhoneParams with data set #0
ok 2 - ValidationTest::testBadPhoneParams with data set #1
ok 3 - DbConnectionTest::testGetSingleton
not ok 4 - Failure: DbConnectionTest::testTransactionRollb
    ---
    message: 'Failed asserting that <boolean:false> is not e
    severity: fail
    ...
ok 5 - DbConnectionTest::testTransactionCommit
1..5
```

```
require 'test_helper'  
require 'test/phpunit'
```

```
class PHPTest < ActiveSupport::TestCase  
  extend Test::PHPUnit  
  phpunit "test/phpunit/", \  
    :configuration => 'test/phpunit.xml'  
end
```



# **What else should it do?**

- Perl (probably easy)
- Other languages

# **crux**

*[github.com/iangreenleaf/crux](https://github.com/iangreenleaf/crux)*

# **Greasemonkey**

- Firefox extension
- Execute your own Javascript on matched pages

```
// ==UserScript==  
// @name Github autocomplete  
// @namespace iangreenleaf.com  
// @description Autocompletion for @-n  
// @include https://github.com/*  
// @require http://.../1.6.4/jquery.mi  
// ==/UserScript==
```

```
$( ".comment-form" ).keyup(function(e) {  
    // ...  
});
```

**Greasemonkey scripts work on  
Chrome**

# **Except a few things**

Like @require.

# Chrome extensions

Turns out packaging them isn't that hard.

# manifest.json

```
{  
  "name": "My Extension",  
  "version": "2.1",  
  "description": "Sample plugin.",  
  "permissions": ["https://*.google.com/"],  
  "content_scripts":  
    [{ "js": ["jquery.js", "myscript.js"] }]  
}
```



**I am so lazy**

# **Let's package it automatically**

Use the Greasemonkey metadata and a couple assumptions.

# To Do

- Most of the actual work
- Host from GitHub
- Auto-updating
- Maybe export to [dotjs](#) as well
- CoffeeScript?

# **Ultra-minimalist gem bootstrapping**

**Let's make a gem!**

***A little intimidating...?***

**Nahhhhh**

# Hoe, Jeweler

- Gem management gems
- Comprehensive

# But...

- Kinda pushy
- I have to require the jeweler gem inside my gem
- Kinda... complicated



**Suddenly we're back to  
intimidated**

**There are some other options out  
there**

But nothing that gets me particularly excited

**There's another way**

**The tool is coming...**

**...from INSIDE YOUR  
COMPUTER**

The best way to manage your  
application's dependencies

# Bundler



```
$ bundle gem buzzfizz
  create  buzzfizz/Gemfile
  create  buzzfizz/Rakefile
  create  buzzfizz/.gitignore
  create  buzzfizz/buzzfizz.gemspec
  create  buzzfizz/lib/buzzfizz.rb
  create  buzzfizz/lib/buzzfizz/version.rb
Initializing git repo in /home/youngian/code/buzzfizz
```

```
# Rakefile
```

```
require 'bundler'
```

```
Bundler::GemHelper.install_tasks
```



```
# .gitignore
```

```
*.gem
```

```
.bundle
```

```
Gemfile.lock
```

```
pkg/*
```

```
# lib/buzzfizz/version.rb
```

```
module Buzzfizz  
  VERSION = "0.0.1"  
end
```

```
# lib/buzzfizz.rb
```

```
module Buzzfizz  
  # Your code goes here...  
end
```

*# Gemfile*

source ["http://rubygems.org"](http://rubygems.org)

*# Specify your gem's dependencies*  
*# in buzzfizz.gemspec*  
gemspec

```
# buzzfizz.gemspec
# -*- encoding: utf-8 -*-
$.push File.expand_path("../lib", __FILE__)
require "buzzfizz/version"
```

```
Gem::Specification.new do |s|
  s.name = "buzzfizz"
  s.version = Buzzfizz::VERSION
  s.platform = Gem::Platform::RUBY
  s.authors = ["TODO: Write your name"]
  s.email = ["TODO: Write your email address"]
  s.homepage = ""
  s.summary = %q{TODO: Write a gem summary}
  s.description = %q{TODO: Write a gem description}
```

```
s.rubyforge_project = "buzzfizz"
```

```
s.files = `git ls-files`.split("\n")  
s.test_files = `git ls-files -- {tes  
s.executables = `git ls-files -- bir  
s.require_paths = ["lib"]
```

```
end
```

**We'll probably want to add a  
couple things**

```
s.add_dependency "activerecord", "~> 3.0.9"
s.add_development_dependency "mocha"
```



**Let's use it**

```
$ rake -T
rake build          # Build buzzfizz-0.0.2.gem into the
                    # pkg directory
rake db:prepare     # Prepare the databases.
rake default        # Default: run all unit tests.
rake install        # Build and install buzzfizz-0.0.2.gem in
                    # system gems
rake release        # Create tag v0.0.2 and build and
                    # push buzzfizz-0.0.2.gem to Rubygems
rake spec           # Run the test suite.
rake spec:all       # Run the test suite for all DBs.
```

## sign up

You can now sign in with your RubyForge Account too!

[Already have an account?](#)

Email address

Handle

Password

# Authorize your machine

```
$ gem push  
Email:  
Password:
```

```
$ rake release  
buzzfizz 0.0.2 built to pkg/buzzfizz-0.0.2.gem  
Tagged v0.0.3  
Pushed git commits and tags  
Pushed buzzfizz 0.0.2 to rubygems.org
```

**RubyGems.org**  
your community gem host

[all gems](#)





[sign in](#)

[sign up](#)

# activerecord\_enum 0.0.3

Adds the ENUM data type natively to ActiveRecord.

**INSTALL** > `gem install  
activerecord_enum`

 Download  Documentation  Subscribe  Stats

## Authors

Ian Young

198

total downloads

3

for this version

## Owners



# **Congratulations!**

You're a gem author!



# Thanks

[github.com/iangreenleaf](https://github.com/iangreenleaf)

[@iangreenleaf](#)