



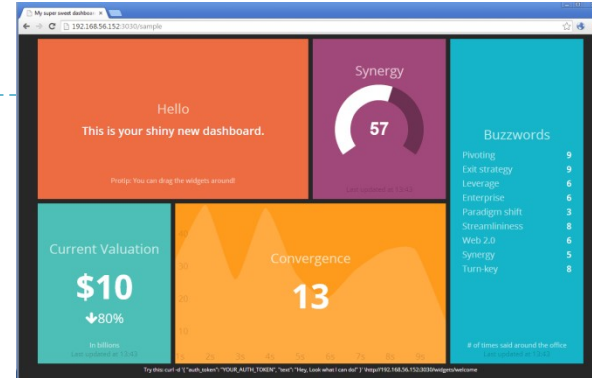
Learn Dashing Widget in 90 minutes



Larry cai <larry.caiyu@gmail.com>

Agenda

- ▶ Introduction
- ▶ Exercise 1: Install Dashing
- ▶ Exercise 2: Add one widget
- ▶ Exercise 3: Update the view
- ▶ Exercise 4: Control the data
- ▶ Exercise 5: Update the data with color (coffeescript)
- ▶ Exercise 6: Pull the data (jobs)
- ▶ Reference

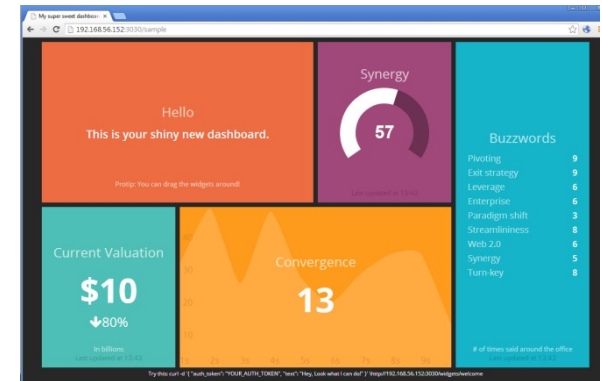


Dashing 🎀



Introduction

- ▶ Dashing is a Sinatra based framework that lets you build beautiful dashboards.
- ▶ **Key features:**
 - ▶ Use premade **widgets**, or fully create your own with scss, html, and coffeescript.
 - ▶ Widgets harness the power of data bindings to keep things DRY and simple. Powered by [batman.js](#).
 - ▶ Use the **API to push data** to your dashboards, or make use of a simple ruby DSL for **fetching data**.
 - ▶ Drag & Drop interface for re-arranging your widgets.

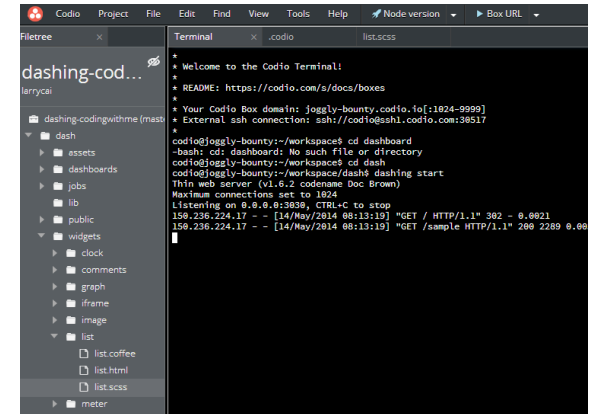


Dashing 

Source <http://shopify.github.io/dashing>

Environment

- ▶ Ruby environment (Ruby 1.9.x+, Node.js 0.10+)
- ▶ Ubuntu 14.04
 - ▶ `$ sudo apt-get install ruby, ruby-dev, gem`
 - ▶ `$ sudo apt-get install bundler, g++, make,`
 - ▶ `$ sudo apt-get install nodejs`
 - ▶ `$ sudo gem install dashing`
- ▶ Windows (use virtualbox with Ubuntu VM)
 - ▶ <http://virtualboxes.org/>
 - ▶ <http://virtualboxes.org/images/ubuntu-server/>
- ▶ Or use online service for exercise
<http://c9.io> , <http://codio.com>



Environment using c9.io

- ▶ <https://c9.io/>
 - ▶ New Ruby workspace
- ▶ In console:
`gem install dashing`

Create a new workspace

Workspace name

dashing

Description

Make a short description of your workspace

Hosted workspace

Clone workspace

Remote SSH workspace

☐ Private

This is a workspace for your eyes only


☐ Public


This will create a workspace for everybody to see


Clone from Git or Mercurial URL (optional)


e.g. ajaxorg/ace or git@github.com:ajaxorg/ace.git


Choose a template



HTML5


Node.js


PHP, Apache & ...


Python


Django


Ruby

Exercise 1: Install Dashing

▶ Create new dashboard

- ▶ `$ dashing new dashboard`
- ▶ `$ cd dashboard`

▶ Remove twitter (need both)

- ▶ Comment out twitter in Gemfile
- ▶ Delete twitter.rb in jobs (dir)

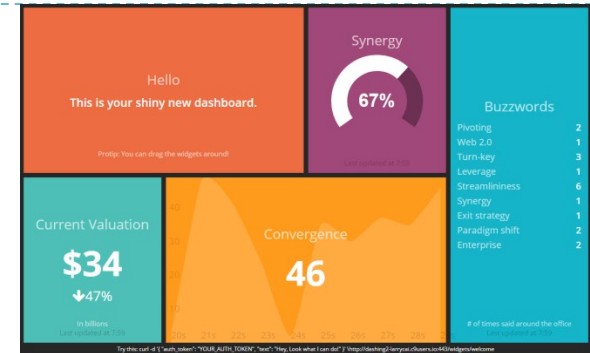
▶ Start it

- ▶ `$ bundle`
- ▶ `$ dashing start -p 8080` # default is 3030

▶ Point your browser to <http://localhost:8080>

▶ Or Run panel (if c9.io)

1. Run the project with the "Run Project" button in the menu bar on top of the IDE.
2. Preview your new app by clicking on the URL that appears in the Run panel below (<https://dashing2.larrycai.c9users.io/>).

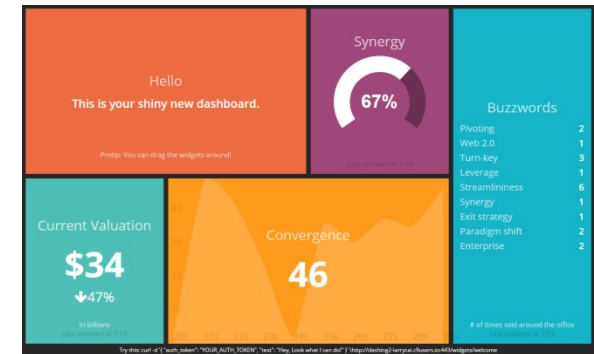


Widget

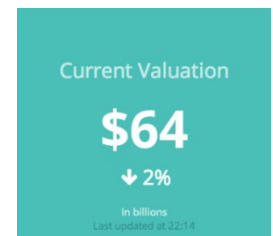
- ▶ Dashboard : dashboard/sample.erb (ruby template)

```
<li data-row="1" data-col="1" data-size="1" data-size="1">  
  <div data-id="valuation" data-view="Number" data-title="Current Valuation" data-moreinfo="In billions" data-prefix="$"></div>  
</li>
```

- ▶ **Dashboard** is created with **widgets**
 - ▶ Pre-made 50+ widgets (**market place**)
 - ▶ Create own widget using css/html
- ▶ **Widget** (set of related files)
 - ▶ widget\number\number.html – View (html)
 - ▶ widget\number\number.scss – Style (css)
 - ▶ widget\number\number.coffee – Data (coffeescript)



Dashboard



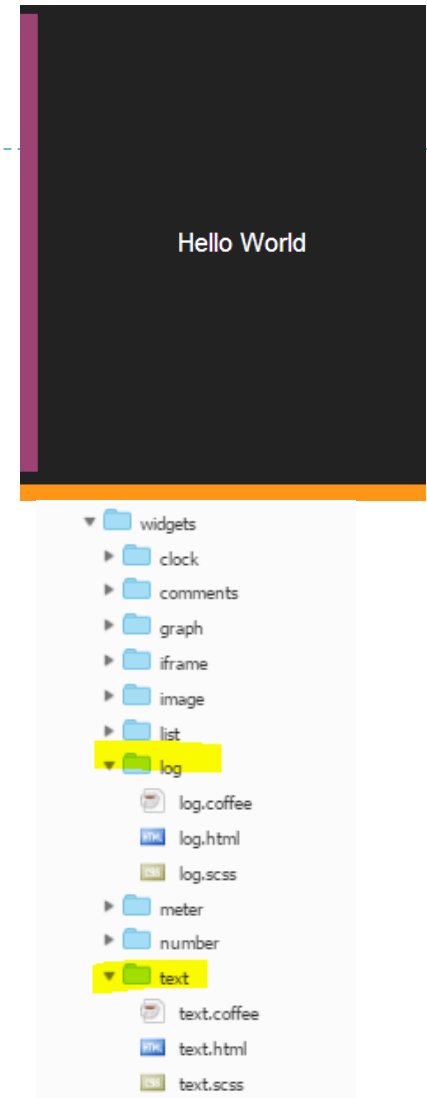
Widget

Exercise 2: Add new widget

- ▶ Generate `log` widget
 - ▶ \$ dashing generate widget log
- ▶ Add into Dashboard
 - ▶ # update sample.erb

```
<li data-row="1" data-col="1" data-size="2" data-size-y="1">  
  <div data-id="log" data-view="Log" data-title="Monitor logs" data-value="Hello World"></div>  
</li>
```

- ▶ Check log.html
 - ▶ Change data-view from Log to Text
 - ▶ View can be used in other widget “instance” / “object”
- !! Log shall be capitalized for data-view !!!

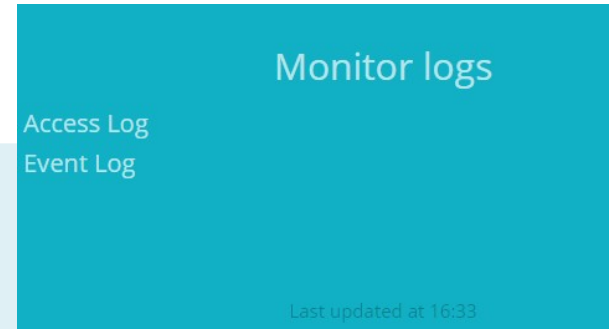


<https://gist.github.com/larrycai/79cf4c63927957a37eba/a909ff38b411eff8cf714c29973a7dc26d9f309e>

Exercise 3: Update the view

► Update log.html

```
# log.html
<h1 class="title" data-bind="title"></h1>
<ul class="list-nostyle">
  <li>
    <span class="label">Access Log</span>
    <span class="value" data-bind="access">0</span>
  </li>
  <li>
    <span class="label">Event Log</span>
    <span class="value" data-bind="event2">0</span>
  </li>
</ul>
<p class="updated-at" data-bind="updatedAtMessage"></p>
```




► Update log.scss

- Copy the view using list widget css (list.scss -> log.scss)
- Change inside .widget-list -> .widget-log

Data in Dashing

- ▶ Each widget has the data-id (like `log`), which provides the URL access point to update the data



```
<li data-row="1" data-col="1" data-size="2" data-sizey="1">  
  <div data-id="log" data-view="Log" data-title="Monitor logs" data-value="Hello World"></div>  
</li>
```

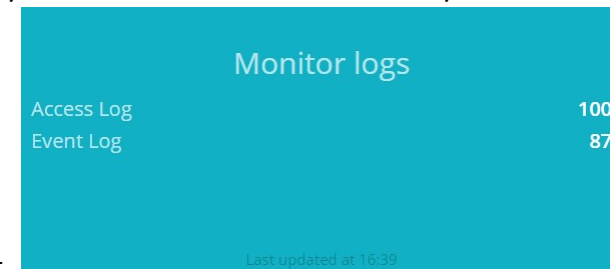
- ▶ Push data to dashing using REST API (see bottom)
 - ▶ `$ curl -d '{ "auth_token": "YOUR_AUTH_TOKEN", "title": "Current Hour Rate" }' http://<your url>/widgets/log`

Exercise 4: Control the Data

- ▶ Update the data using curl for welcome message
 - ▶ Find in the bottom of your dashboard (remove 443, \ front of http)

```
Try this: curl -d '{ "auth_token": "YOUR_AUTH_TOKEN", "text": "Hey, Look what I can do!" }' http://dashing-larrycai.c9users.io:443/widgets/welcome
```

- ▶ Update the data using curl for log
 - ▶ `$ curl -d '{ "auth_token": "YOUR_AUTH_TOKEN", "access": "100", "event2": "87" }' http://<url>/widgets/log`
- ▶ Update the data using curl for log
 - ▶ # Save the request into log.json
 - ▶ `$ curl -d @log.json http://<url>/widgets/log`
- ▶ Write your own script from anywhere!! (homework)



Data in Dashing (more)

- ▶ Push data from anywhere using any language besides curl
 - ▶ i.e. Python in jenkins job
- ▶ Push data in dashing jobs (ruby)
 - ▶ `jobs/sample.rb`

```
3 current_valuation = 0
4
5 SCHEDULER.every '2s' do
6   last_valuation = current_valuation
7   current_valuation = rand(100)
8
9   send_event('valuation', { current: current_valuation, last: last_valuation })
10 end
```



CoffeeScript

- ▶ CoffeeScript is like javascript to control views when data comes (like color) (log.coffeescript)

```
class Dashing.Number extends Dashing.Widget
  @accessor 'current', Dashing.AnimatedValue

  @accessor 'difference', ->
    if @get('last')
      last = parseInt(@get('last'))
      current = parseInt(@get('current'))
      if last != 0
        diff = Math.abs(Math.round((current - last) / 100))
        "#{diff}%"
      else
        ""

  @accessor 'arrow', ->
    if @get('last')
      if parseInt(@get('current')) > parseInt(@get('last')) then 'icon-arrow-up'
      else 'icon-arrow-down'

  onData: (data) ->
    if data.status
      # clear existing "status-*" classes
      $(@get('node')).attr 'class', (i,c) ->
        c.replace /\bstatus-\S+/g, ''
      # add new class
      $(@get('node')).addClass "status-#{data.status}"
```

```
<h1 class="title" data-bind="title"></h1>
<h2 class="value" data-bind="current"></h2>

<p class="change-rate">
  <i data-bind-class="arrow"></i><span data-bind="difference"></span>
</p>

<p class="more-info" data-bind="moreinfo"></p>
<p class="updated-at" data-bind="updatedAtMessage"></p>
```

(Skipped)

Exercise 5: Update the color with data

- ▶ `<error>/<warning>` if there is error, it is red, if there is warning, it is yellow. Otherwise green
- ▶ Use css class to change the color

```
# log.coffee
ready: ->
  # This is fired when the widget is done being rendered

onData: (data) ->
  # Fired when you receive data
  for key,value of data
    break if key in ["id","updatedAt"]
    id = $(@node).find("##{key}")
    console.log(id)
    [error,warning] = value.split("/")
    if error != "0"
      id.attr("class","value-error")
```

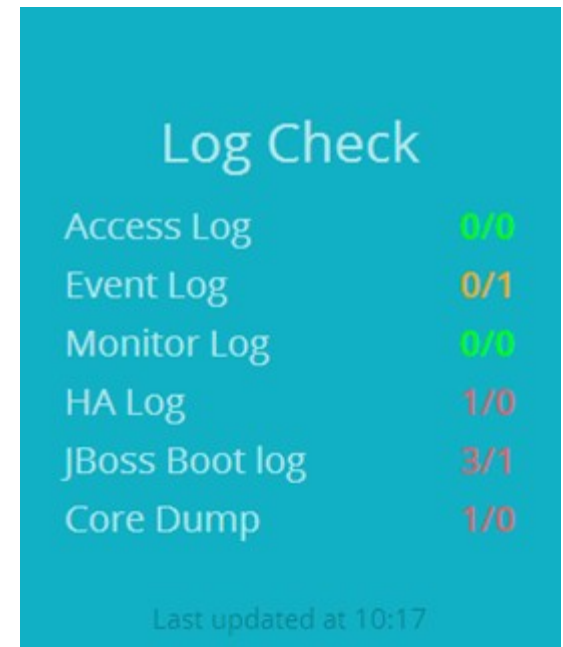
Log Check	
Access Log	0/0
Event Log	0/1
Monitor Log	0/0
HA Log	1/0
JBoss Boot log	3/1
Core Dump	1/0
Last updated at 10:17	

- ▶ Use chrome developer to debug ..

(Skipped)

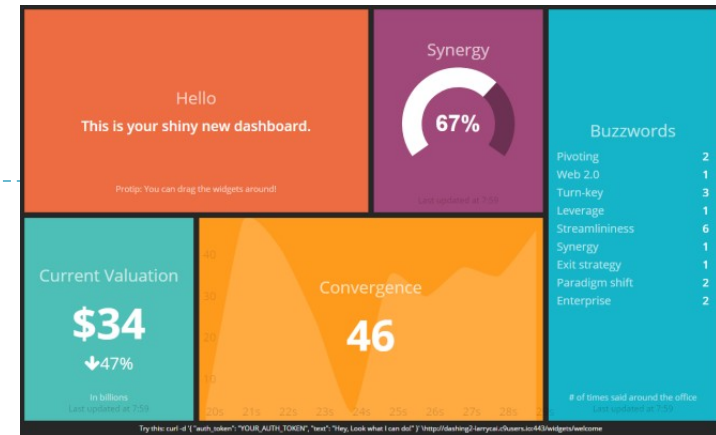
Exercise 6: Pull data in jobs

- ▶ Write log job to simulate fetching the log and send event to widget
- ▶ `$ dashing generate job log`
- ▶ Update the data in every 5 second with random data
- ▶ Check `sample.rb`



Summary

- ▶ Dashing is a Radiator framework
- ▶ Dashboard consists of Widgets
- ▶ Widget:
 - ▶ Connect with widget view (html/css) (may share one) coffeescript is used to update view when data comes
 - ▶ Connect with widget id (expose as HTTP/REST API)
- ▶ Update Dashboards can be done
 - ▶ HTTP/REST API using any language
 - ▶ Or internal ruby based jobs (crontab) inside Dashing



Reference

- ▶ <http://shopify.github.io/dashing/>
- ▶ <https://github.com/Shopify/dashing/wiki/Additional-Widgets>
- ▶ <http://dashing.challengepost.com/submissions>
- ▶ <https://www.andreydevyatkin.com/archives/getting-started-with-dashing/>
- ▶ <http://www.alexecollins.com/content/team-dashboard-dashing/>
- ▶ Log widget gist:
<https://gist.github.com/larrycai/79cf4c63927957a37eba>