李艳增学习笔记

- HOME
- ARCHIVES
- RSS
- ABOUT ME
-

第八周总结(9/12-9/18)

Published on: September 18, 2016

本周是全栈工程师培训班的最后一周。

本周的重要收获是 xdite 老师和笑来老师的结业讲话。

xdite 老师的结业讲话主要包括三个方面:

回顾了所教的五个重要原则

1. landing page (挖掘价值)
2. user story 与项目管理(管理优先权)
3. 技术实践（肌肉学习法）
4. 协作(如何与同事协作)
5. on boarding（传达价值）

毕业后我希望我们继续做的事

1. 学习时把错误的笔记抄下来
2. 每周写周记
3. After Action Review
4. 学习学习再学习

毕业赠言

1. 不要去陷入“争论”
2. 不要抢先去“追技术”
3. 而是要抢先去“解决问题”
4. 然后再去思索可以怎么样“不去解决问题”
5. 全栈思维很重要，如何思索全局更重要

技术推荐

1. http://confreaks.com
2. http://railsconf.com
3. http://rubyweekly.com

笑来老师的讲话主要包括两个方面：

给你所做的事赋予一层又一层的意义

1. 把努力这两个字从脑子里摘掉
2. 不是努力，而是干脆停不下来
3. 努力从来都是平庸者的专利
4. 未来的世界谁的笔杆子厉害谁就有洪亮的声音
5. 想你所做的事的意义，往死了想
6. 买比特币不是冒险，不买比特币才是冒险（备注：这不是投资建议）。

不停地深入思考作品的完整性

1. 每天花三十分钟去想未来的世界是什么样的
2. 去做未来的决定，而不是盯着眼前，这也是实现财富自由的关键
3. 一切功夫都在行外，锁的钥匙不在锁孔里，要到别的地方去找
4. 浓缩书是非原创式写作
5. Github 是全世界最牛逼的社交网站

第七周总结(9/5-9/11)

Published on: September 11, 2016
本周的团队项目进行了收尾和上线的工作。欢迎各位访问：人才火箭

本周遇到的大坑是修改密码功能，已有专文描述。参见：修改密码功能进阶

本周学到的好用的工具 :chrome 浏览器插件 :octotree; 博客数据监测工具 :Google Analytics

本周学到的重要的概念是拼图理论。学习是拼图，而不是登山。你并不需要从计算机的基础理论知识开始学起，也并不需要把某个编程语言的语法规则全部掌握了，才开始去编程实战，这样，是把学习当成了登山，认为必须把基础打牢固了才可以进行下一步学习，这是错误的认知。学习不是登山而是拼图。既然是拼图，就是说你可以这里拼一块，那里拼一块，假以时日，你是可以把整个图给拼起来的。当然，更好的办法应该是先把整个图的边框给拼起来，这样，你就知道图的边界在那里了，然后，再往中间填东西。学习编程以外的技能，也可以套用这个理论。

编程初学者如何避免从入门到放弃？
Published on: September 10, 2016
软件正在吞噬整个世界。

霍师傅曾说，在中国，大概唯一增长率可以比得上北上广房产的，就是程序员的薪水了。你可能也听过那个“所有都准备好了，只差一个程序员就可以开工了”的段子，然而这并不好笑，有大量的真实项目，是真的只差一个程序员。

那么，零基础要如何学习编程呢？

或许你会到网络上搜索各种免费视频教学下载到自己的硬盘，购买各种从入门到精通的大部头书籍，到知乎上看各路人马对初学者的各种建议，甚至到国内外各大公开课平台付费学习，然而，在学习了一段时间之后，你终于成功的实现了从入门到放弃。相信我，你并不孤独。

那么，原因何在呢？是因为你没有学习学习再学习（如果你不熟悉这七个字，稍微解释一下，这不是重要的事情说三遍，而是学习了学习的方法论再学习，第二个学习是名词）。你没有掌握有效的学习方法论就开始学习，是事倍功半的。

下面，我将分享我在全栈工程师课堂上学到的三个学习方法论，分别是：拼图理论，成就感和重复训练。

1. 拼图理论。学习是拼图，而不是登山。你并不需要从计算机的基础理论知识开始学起，也并不需要把某个编程语言的语法规则全部掌握了，才开始去编程实战，这样，是把学习当成了登山，认为必须把基础打牢固了才可以进行下一步学习，这是错误的认知。学习不是登山而是拼图。既然是拼图，就是说你可以这里拼一块，那里拼一块，假以时日，你是可以把整个图给拼起来的。当然，更好的办法应该是先把整个图的边框给拼起来，这样，你就知道图的边界在那里了，然后，再往中间填东西。学习编程以外的技能，也可以套用这个理论。

2. 成就感。指的是你在学习的过程中要不断得到正向的反馈，这样让你能够有很强的成就感，甚至觉得自己原来是天才，而不是被挫败感所湮灭，从而轻易的完成从入门到放弃。初学者如何获得正向反馈，在编程学习上，应该与教练的教程设计有直接的关系，付大钱去上的课程，和你在网络上搜索到的资源，有天壤之别。

3. 重复训练。大量的重复训练可以让你形成肌肉记忆。xdite 老师说，初学编程最好最快的方式是“跟着打”“不懂的地方背起来”，然后，你就会突然间“懂”很多东西了，解题越来越快，越来越知道自己在干什么，直到最后可以独立开发项目。刚开始，我也是不信的，但还是照做了，结果是令人震惊的。

最重要的是，这些概念和方法论并不是编程学习里所独有的，而是可以实践到学习其他技能中去的。聪明的人只不过是善于举一反三。

9/9 修改密码功能进阶

Published on: September 9, 2016

1.touch config/locales/devise.zh-CN.yml,加入以下内容

```
zh-CN:
  activerecord:
attributes:
  user:
    current_password: "Password"
errors:
  models:
    user:
      attributes:
        email:
          blank: "邮箱不能为空"
          taken: "该邮箱已注册过"
        current_password:
          invalid: "原始密码错误"
          blank: "请输入原始密码"
          too_short: "原始密码错误"
        password:
          too_short: "新密码太短，请至少使用 8 个字符"
          blank: "密码不能为空"
        password_confirmation:
          confirmation: "新密码两次输入不一致"
```

```
devise:
confirmations:
  confirmed: "您的帐号已经确认，您现在已登录。"
  send_instructions: "几分钟后，您将收到确认帐号的电子邮件。"
  send_paranoid_instructions: "如果您的邮箱存在于我们的数据库中，您将收到一封确认账号的邮件。"
failure:
  already_authenticated: "您已经登录。"
  inactive: "您还没有激活帐户。"
  invalid: "邮箱或密码错误。"
  locked: "您的帐号已被锁定。"
  last_attempt: "您还有最后一次尝试机会，再次失败您的账号将被锁定。"
  not_found_in_database: "邮箱或密码错误。"
  timeout: "您已登录超时，请重新登录。"
  unauthenticated: "继续操作前请注册或者登录。"
  unconfirmed: "继续操作前请先确认您的帐号。"
mailer:
  confirmation_instructions:
    subject: "确认信息"
  reset_password_instructions:
    subject: "重置密码信息"
  unlock_instructions:
    subject: "解锁信息"
  password_change:
    subject: '密码已重置'
omniauth_callbacks:
  failure: "因为%{reason}，所以您无法从%{kind}获得授权。"
  success: "成功地从%{kind}获得授权。"
passwords:
  no_token: "这是密码重置页面，未重置邮件不得访问此页面。如果您是通过重置邮件而来的，请确保您访问的 URL 是完整的。"
  send_instructions: "几分钟后，您将收到重置密码的电子邮件。"
  send_paranoid_instructions: "如果您的邮箱存在于我们的数据库中，您将收到一封找回密码的邮件。"
  updated: "您的密码已修改成功，您现在已登录。"
  updated_not_active: "您的密码已修改成功。"
registrations:
  destroyed: "再见！您的帐户已成功注销。我们希望很快可以再见到您。"
  signed_up: "欢迎来到人才火箭，您已成功注册新账号。"
  signed_up_but_inactive: "您已注册，但尚未激活账号。"
  signed_up_but_locked: "您已注册，但账号被锁定了。"
  signed_up_but_unconfirmed: "一封带有确认链接的邮件已经发送至您的邮箱，请检查邮箱（包括垃圾邮箱），并点击该链接激活您的账号。"
```

```
  update_needs_confirmation: "信息更新成功，但我们需要验证您的新电子邮件地址，请检查邮箱（包括垃圾邮箱），并点击该链接激活您的账号。"
  updated: "帐号资料更新成功。"
sessions:
  signed_in: "登录成功."
  signed_out: "退出成功."
  already_signed_out: "已经退出成功."
unlocks:
  send_instructions: "几分钟后，您将收到一封解锁帐号的邮件。"
  send_paranoid_instructions: "如果您的邮箱存在于我们的数据库中，您将收到一封解锁账号的邮件。"
  unlocked: "您的帐号已成功解锁，您现在已登录。"
 errors:
messages:
  already_confirmed: "已经确认，请重新登录。"
  confirmation_period_expired: "请在%{period}内确认注册，请重新注册。"
  expired: "邮件确认已过期，请重新注册。"
  not_found: "没有找到。"
  not_locked: "未锁定。"
  not_saved:
    one: "因为 1 个错误导致此%{resource}保存失败:"
    other: "因为%{count}个错误导致此%{resource}保存失败:"
```

2.touch config/locaels/simple_form.zh-CN.yml，填入以下内容

```
zh-CN:
simple_form:
"yes": 'Yes'
"no": 'No'
required:
  text: 'required'
  mark: '*'
  # You can uncomment the line below if you need to overwrite the whole required html.
  # When using html, text and mark won't be used.
  # html: '<abbr title="required">*</abbr>'
error_notification:
  default_message: "请检查以下错误:"
# Examples
# labels:
#   defaults:
#     password: 'Password'
#   user:
#     new:
#       email: 'E-mail to sign in.'
#     edit:
```

```
#            email: 'E-mail.'
# hints:
#   defaults:
#     username: 'User name to sign in.'
#     password: 'No special characters, please.'
# include_blanks:
#   defaults:
#     age: 'Rather not say'
# prompts:
#   defaults:
#     age: 'Select your age'
```

3.修改 config/application.rb

```
+       config.i18n.default_locale = :"zh-CN"
```

注意：zh-CN 一定要加引号，此处是坑，请绕行。另外，修改 config 内容需重启服务器。

4.修改 app/controllers/devise/users/registration_controller.rb,增加以下内容

```
def update
self.resource = resource_class.to_adapter.get!(send(:"current_#{resource_name}").to_key)
prev_unconfirmed_email = resource.unconfirmed_email if resource.respond_to?(:unconfirmed_email)

resource_updated = update_resource(resource, account_update_params)

yield resource if block_given?

if resource_updated
  if account_update_params[:password].blank?
    flash[:alert] = "请输入新密码"
    redirect_to change_password_account_user_path(resource)
    return
  else
    if is_flashing_format?
      flash_key =
        update_needs_confirmation?(resource, prev_unconfirmed_email) ?
                  :update_needs_confirmation : :updated
      set_flash_message :notice, flash_key
    end
    bypass_sign_in resource, scope: resource_name
    respond_with resource, location: after_update_path_for(resource)
  end
else
  clean_up_passwords resource
  errors = resource.errors.messages
  puts "#{resource.errors.messages}"
```

```
    if errors[:current_password].present?
      flash[:alert] = errors[:current_password].first
      redirect_to change_password_account_user_path(resource)
      return
    elsif errors[:password].present?
      flash[:alert] = errors[:password].first
      redirect_to change_password_account_user_path(resource)
      return
    elsif errors[:password_confirmation].present?
      flash[:alert] = errors[:password_confirmation].first
      redirect_to change_password_account_user_path(resource)
      return
    end

    redirect_to change_password_account_user_path(resource)
end
end
```

5.如何不把整个 devise 的 controller 拉出来的情况下达到上述效果

新建一个 registration 的 controller，继承 devise 的 registrationscontroller

class RegistrationsController < Devise::RegistrationsController

仍然填入上面 update 的内容。注意：new 和 create action 中的 super 要打开。

6.修改路径 routes.rb

```
devise_for :users, controllers: { registrations: "registrations",
change_password: "registrations" }
```

9/8return 的用法

Published on: September 8, 2016

return 的用法

```
if @user.user_name.blank?
flash[:alert] = "请输入用户名"
render :edit
return
end
```

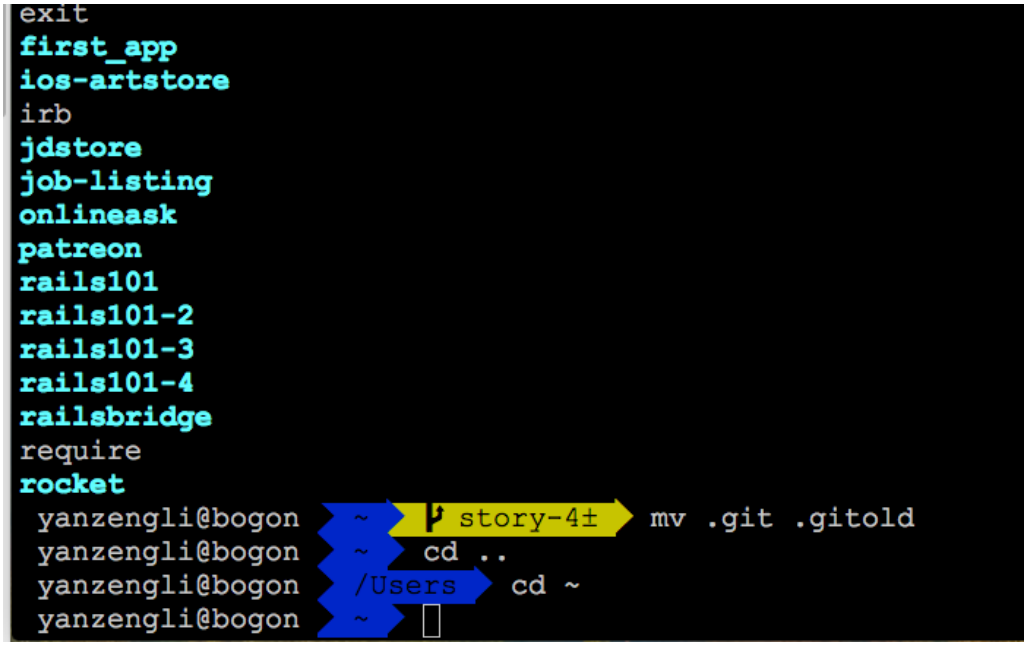
trix 编辑框出现滚轮写法

```
.formatted_content {
background-color: #ffffff;
min-height: 200px;
// width: 600px;
margin-bottom: 10px;
max-height: 300px;
overflow-y: scroll;
resize: none;
}
```

9/7 错题记录

Published on: September 7, 2016

移除旧的根目录命令



用 Heroku 上的插件 SendGrid 可以实现在线发送验证邮件的功能。

解决页面不能完全展示问题

把左右两边的 div 给包起来

标签的 target 属性规定在何处打开链接文档。

Preface

9/5 编辑器排版

Published on: September 5, 2016

筹款目标为正数的写法：

<%= f.input :fund_goal, label: "筹款目标", input_html:{min:1, max:1_000_000} %>

上传网站图标的正确写法：

<link rel="shortcut icon" href="/images/rocket2.ico">

在 CSS 的规则中，中文和英文的断行规则不一样

word-wrap：Allow long words to be able to break and wrap onto the next line:

```
p.test {
    word-wrap: break-word;
}
```

第六周总结(8/29-9/4)

Published on: September 4, 2016

本周在项目中完成的任务主要有：

1. 把 Devise 的修改密码功能和项目网页布局无缝接合
2. 把 tinymce-rails 运用到项目中
3. 把在线客服系统运用到项目中

4. 通过学习 html 和 css 完成了“关于我们”的页面排版和布局。

本周团队完成的项目任务主要有：

1. 优化整个项目发布流程，把原来分散的功能整合到一起，让发布者用起来流畅自如
2. 优化 Landing Page 页面
3. 优化 Project show 页面
4. 增加 How it works 页面，让用户了解网站运作流程

本周领悟到的最重要的一句话：

成功的真相从来不是努力而是做对了事情。

第六周遇到的大坑(8/29-9/4)

Published on: September 4, 2016

一定要防止泄露 aws 的秘钥

编辑 .gitignore 文件并不是立即生效的，要看情况。如果你 ignore 的文件没有被追踪（untracked），比如新建的文件，则添加后（git add .gitignore 并 commit 后）可以生效。但如果是已经追踪的文件，需要通过清除缓存来解决

直接编辑 .gitignore 是不能忽略文件的，参考这里（泄露了 aws 密钥的惨痛教训）

http://stackoverflow.com/questions/11451535/gitignore-not-working

git rm -r --cached .

git add .

git commit -m "fixed untracked files"

本周遇到的另一个大坑是没有很好的保护注意力。

典型案例是周四晚上本来想要实践一下郑老师所讲的 5 倍速度法看罗辑思维视频，可以十分钟之内看完一期，结果访问了某个因众所周知的原因而无法访问的网站后，注意力就被外媒所报道的北戴河会议、十九大人事布局和巴拿马文件等所吸走，一个小时后才收回注意力，罗辑思维的视频也没有看成，好吧，没有抵挡住抢夺稀缺资源的欲望，访问了不该访问的网站。

知道了保护注意力的重要性，还是会掉到坑里。郑老师总结的保护注意力的方法论：放弃抢夺稀缺资源的欲望和不害怕失去。

第六周学到的概念(8/29-9/4)

Published on: September 4, 2016

本周学到的重要概念是：Growth Product + Growth Marketing

Growth hack 和传统行销的区别是：传统行销侧重于围绕品牌进行，Growth hack 侧重于 Growth Marketing 和 Growth Product

Growth Marketing 是以成长为导向的行销，让人在商品没有名气时，也能信任你。主要包括：

1. Landing Page
2. A/b Testing
3. Content Markting

4. … … … 各项 Acquistion 技术

Growth Product 是以成长为导向的产品开发，让新进来的使用者感觉这个东西很有价值。找到自己独一无二的价值，修正打磨模式，找到正确的客户。

1. New User Experience(Onboarding)
2. Customer Support
3. NPS
4. Referral/Viral

那么，如何踏出成长的第一步？

1. 找到产品成长模型
    - 你的客户怎样自然变多的
    - 你的客户怎样介绍你的
    - Acquisition(曝光点) -> Activation(核心功能) -> Referral(转介)
2. 找到产品的成长系数
    - Activation(核心功能) -> Referral(转介)
    - 每 100 个用户会产生 30 个新用户
    - 1+0.3+0.09+0.027+…=1.42
3. 只调整黄金通道上的功能
    - 假设 100 个用户可以生出 80 个新用户
    - 成长系数是 1/(1-0.8)=5
    - 假设你调整黄金通道上的某个功能，做一点 A/B testing，效果上升 8%
    - 1/(1-0.88)=8.3
    - 8.3/5=1.66
    - 8% => 166%

接下来，如何对产品功能排定优先权

1. 把产品功能分组
    - 功能组
        - 不会形成 loop 的功能
        - 只针对旧用户
        - 线性 Channel 上的漏斗相关功能
        - 跟销售漏斗离超远的功能
    - 成长组
        - Landing Page
        - On boarding
        - Ah-ha moment
        - Referral
    - 产品好业绩不一定上涨，这是不同两件事
    - 思考你做的东西是在功能组还是成长组
2. 把使用者分群
    - Core user(核心用户)
    - Active user(一般日常用户)
    - Casual user(很偶尔才用一下的用户)
        - 让使用者养成习惯
        - 黏上产品

- Uniques(已经到你家门口了，但还没有应用)

3. 算 Project 的价值
   - ICE score
     - impact(影响)
     - confidence(实作成功的信心)
     - ease(难度-能不能做出来)
   - 加入开发成本因素
   - 右边值得做，左边先放一边
   - slam dunks
     - 成长系数上升>20%
     - 针对已存在的 loop 做简单的加强
     - 原本有做，但没做好的功能拿出来做
     - 例子：改善 Landing Page 文案
     - 例子：把注册页面优化
   - Quickies
     - Growth Multiplier 上升 10%
     - 成本比较低但 Growth Score 没那么好的改善
     - 比如说 Slam Dunks 改了三四次后就会变成 Quickies
   - Moonshots
     - 早期不要同时去三个平台（知乎、微博、微信）推广

最后的总结

- Landing Page + Signup Flow
- Onboarding
- Referral Program

8/30 用户生成随机默认头像

Published on: September 2, 2016

1.把你选好的若干张默认头像上传到 public/images/

2.把你的图片命名为 user1/user2/user3/user.../usern

3.修改 app/uploaders/headimage_uploader.rb

```
def default_url
rand_num = rand(4) + 1
"/images/user#{rand_num}.png"
end
```

8/29 用户修改密码

Published on: August 29, 2016

原来用的方法是直接用一个链接连到 devise 的用户注册页面修改，路径是 edit_user_registration_path

这种方法的缺点是让用户觉得该网站的画风陡变，有很强的违和感。学习过老师修改的第二组的代码后，总结如下：

1.修改 account 的 controller,自定义一个修改密码的方法

```
def edit_password
@user = current_user
end
```

2.修改 application_controller.rb

```
+   helper_method :resource, :resource_name, :devise_mapping
+
+   def resource_name
+      :user
+    end
+
+   def resource
+      @resource ||= User.new
+   end
+
+   def devise_mapping
+      @devise_mapping ||= Devise.mappings[:user]
+   end
```

3.修改 app/views/account/users/edit_password.html.erb

```
+<%= simple_form_for(resource, as: resource_name, url: registration_path(resource_name), html: { method: :put }) do |f| %>
+   <%= f.error_notification %>
+
+   <div class="form-inputs">
+
+      <%= f.input :password, autocomplete: "off", hint: "leave it blank if you don't want to change it", required: false %>
+      <%= f.input :password_confirmation, required: false %>
+      <%= f.input :current_password, hint: "we need your current password to confirm your changes", required: true %>
+   </div>
+
+   <div class="form-actions">
+      <%= f.button :submit, "Update" %>
+   </div>
+<% end %>
```

4.修改路径为 edit_password_account_user_path

5.修改 config/routes.rb

6.修改密码后路径如何返回用户中心而不是首页

7.修改 app/controllers/devise/users/registrations_controller.rb

```
+   protected
+   def after_update_path_for(resource_name)
+      account_users_path
```

+ end

8.修改 config/routes.rb

+ devise_for :users, controllers: { registrations: "devise/users/registrations" }

[订阅了八个得到专栏是一种怎样的体验？]

Published on: August 28, 2016

【提示】本文完全是主观体验，毫无客观可言。

得到 App 开创了一个全新的品类。就像可口可乐开创了可乐这个品类，王老吉开创了凉茶这个品类，洛可可开创了降温杯这个品类，阿芙开创了精油这个品类一样，得到在帮人节省时间的知识服务领域开创了一个全新的品类，可能以后这个品类都直接叫得到了。作为罗辑思维的一名铁杆会员，看着罗辑思维一步步走到今天，内心其实是激动的。最近两个月在北京参加培训，每天从朗园门口走过，尤其是晚上，抬头望一眼六楼和七楼的灯光，内心其实是温暖的。

李翔商业内参

李翔商业内参是得到推出的第一个付费订阅专栏，也是截止到目前为止订阅人数最多的一个专栏。可能是由于自己身份的原因，不是企业家、投资人和创业者，每天的内容只能算是了解商业世界变化的一个渠道吧，很少有让人眼前一亮的内容出现。其中印象比较深的一期是《小米在想什么》，介绍了刘德在混沌大学分享的他对小米的思考方式、方法论、布局和未来思考，回答了很多关于小米的疑问。

雪枫音乐会

虽然订阅了这个专栏，但由于本人没有音乐细胞，完全不懂音乐，认真听过的节目并不多。最近找到了一个场景，就是在写作的时候放几首曲子出来听一听。

前哨王煜全

王煜全是海银资本创始合伙人，是极少数把中国的钱拿到美国去投资高科技创新企业的投资人。对于我来说，该专栏的主要作用是长见识和开脑洞。学习到的主要概念有：积木式创新（idea，合适的团队，合适的创始人，足够的资源，启动资金，风险投资人），双长制（经验丰富的 CEO+年轻的首席科学家），拜杜法案（释放了小公司进行科研成果转化的活力）。作者最近在做行业扫描，涉及到的概念有：石墨烯，3D 打印，区块链，虚拟现实，自动驾驶，无人机，裸眼 3D，基因治疗，超级高铁，家用机器人等等。每周六介绍一期关于红酒的知识，印象比较深的是新世界和旧世界的划分。每周日介绍一期关于自我认知的知识。我只想问一句：王煜全，你怎么知道辣么多？

和菜头槽边往事

阅读和菜头博客的时间已经超过十年了，还记得第一次打开槽边往事的时候看到的那句话：口水白白流淌，板砖为谁乱放？这样美丽而忧伤的胖子，腿毛飘飘，站在山岗上。然而千回百转，这么多年过去了，在得到再次重逢。和老师在得到的定位是：带你探索更多的人生可能性。专栏开播以来，印象比较深的事情有，《放下你的瓶啤》里推荐精酿啤酒罗斯福系列和智美系列；《小河在场》里花一万块钱买了河北邯郸一名民谣歌手小河的一首《无爱》，歌词里有一句是“爱如白雪不自留”；《罗振宇的窄腿裤》建议罗振宇改变着装；回答读者的提问，物质和灵魂的自由哪个更重要？当然是物质的自由更重要，正因为我们做不到物质上的自由，所以我们才宽慰自己说，实在不行了，灵魂上的自由也是可以的；以及最近的云南特产松茸。想起柴静问绿妖为什么要跟周云蓬在一起，绿妖说“王小波小说里写过，一个母亲对女儿说，人生多长啊，要找个有趣的人一起过”。柴静问“就因为他有趣吗？”绿妖轻笑：“有趣多难啊”。

李笑来通往财富自由之路

相信新生大学的会员绝大部分都订阅了笑来老师的专栏。该专栏的出现，让别的专栏黯然失色。订阅量的急剧攀升，也是最好的说明，目前仅次于李翔商业内参，而推出的时间比李翔晚了整整两个月。可以预见，笑来老师专栏的订阅量会很快超过李翔商业内参，成为得到第一个突破 100000+订阅量的专栏。如果能看到得到的后台数据，我相信通往财富自由之路是作为礼品赠送最多的一个专栏。

万维钢精英日课

如果你只订阅了通往财富自由之路，想再订一个专栏，我推荐万维钢老师的精英日课。万维钢老师多年前就以网名“同人于野”写作“学而时嘻之”这个博客，更新频率不高，但每一篇都是精品。2014 年万老师出版了《万万没想到》之后，我才知道原来万维钢就是同人于野。

卓老板聊科技

卓老板聊科技是目前订阅量最少的一个专栏。专栏作者卓克虽然被称为卓老板，但年龄应该是目前的所有专栏作者中最小的，出道时间还不到两年，最近刚刚结婚。我去年就开始在喜马拉雅上听卓老板聊科技了，当时的节目定位是理工科版的罗辑思维。印象比较深的节目有：佩雷尔曼与庞家来猜想系列，特斯拉传，费马大定理，欧拉公式，批判中医系列，转基因系列，冬虫夏草来收税等等。转到得到开收费专栏之后，感觉卓老板有点用力过猛，最近讲到的黎曼猜想，博弈论，量子卫星等，还没有学习。

樊登速读

这个专栏我并没有订阅，原因是这样的：在樊登老师来得到开专栏之前，我已经加入了樊登读书会，并且会费已经缴到了 2018 年 5 月份，樊登速读栏目的内容是从樊登读书会已经发布过的内容中筛选出来的，不同点可能就是得到加入了樊老师回答读者提问的环节。樊老师每年讲 50 本书，如果有一本对你有用，就值回票价了。

博雅小学堂

细心的读者应该会注意到，最近得到把这个专栏给下架了，原因不详。自己推测可能是得到为了更加聚焦自己的用户群体吧，毕竟博雅小学堂是给小朋友讲故事的，而得到要做的是帮助用户节省时间的知识服务。顺便说一句，根据脱不花最近在混沌研习社的演讲，罗辑思维有三个订阅栏目在策划中，分别是每天讲一首唐诗，每天讲一幅绘画和讲解国外顶尖杂志的封面文章，我最期待讲封面文章的栏目。

备注：罗辑思维铁杆会员订阅各专栏打九折。

第五周总结(8/22-8/28)

Published on: August 27, 2016

岁月如梭。

第五周结束了。最大的感慨是自己的进步速度变慢了。这一周学到的新东西特别的少，不进步就难受斯基。

自己主要完成了用户头像上传功能和用户身份实名认证功能。复习了图片上传的流程和有限状态机的应用。

并不是没有可以学习的新内容，而是有很多很多。

本周最大的收获是在听笑来老师讲写作课时学到的：如何反复梳理自己的价值观。

以前知道价值观很重要，也知道应该不断的锤炼自己的价值观，可是并没有好的方法论。

笑来老师这次直接给出了方法论：

每隔半小时到一小时要问一遍自己如下的五个问题：

1.这事儿对吗？

2.什么更重要？
3.什么最重要？
4.什么事绝对不能做？
5.什么事一定要做到底？
有了方法论，就要践行了。不要再“懂了辣么多道理，依然过不好这一生。”
但愿岁月如歌。

第五周学到的概念(8/22-8/28)

Published on: August 27, 2016

本周学到的概念有：user onboarding(用户引导)
客户为什么离开？

- 30% 没有感受到价值
- 30% 不知道如何使用
- 10% 进来决定产品烂

Onboarding = Key of Retention
Onboarding 三要素

- Remove Friction (消除疑虑与挫折)
- Deliver Immediate Value (马上带来好处)
- Reward Desired Behaviours（根据行为给予小惠）

养成习惯三要素

- Remove Friction (消除疑虑与挫折)
- Deliver Immediate Value (马上带来好处)
- Reward Desired Behaviours（根据行为给予小惠）

Onboarding 三要素和养成习惯三要素是一模一样的
喜欢一间店

- 不是因为折扣
- 不是因为名气
- 不是因为好处
- 而是【习惯】

如何养成习惯

- 事先知道会发生什么事情
- 良好的体验（中间尽量去除挫折）
- 超乎预期的服务，希望重复

八个问题

- 在 start date 前，客户会寄信来问你什么问题？
- 在 start date 当天，客户忘记做什么会让使用者体验搞砸？（最常客诉的点）
- 他们最常做了什么【正确的事】达到很好的体验？
- 他们最常做了什么【错误的事】结果收到很糟的体验？
- 东西卖出后你如何检验他们【做了正确的事】或【做了错误的事】？
- 他们如何联络你修正问题？
- 你怎么做事后补偿的 instruction?(有 FAQ/说明书/部落格/客服专线)

- 你希望他们如何事后帮你行销？

第五周遇到的大坑（8/22-8/28）

Published on: August 27, 2016

本周遇到的大坑是在上传图片的过程中，在 uploaders/image_uploaders.rb 中 一定要写上：include CarrierWave::MiniMagick，否则后果很严重。

8/24 修改用户头像显示

Published on: August 24, 2016

1.对于用户默认头像的显示问题。在导航栏显示默认头像的时候，如需调整头像的大小，需要到.scss 里定义一个 style,例如：

```
.nav-img {
height :30px;
width :30px;
}
```

然后在需要引用的地方写 class="nav-img“ 就可以了。

2.学习 Bootstrap Panels

A panel in bootstrap is a bordered box with some padding around its content.

8/23 用户身份认证

Published on: August 24, 2016

应用有限状态机

1.安装 AASM:acts_as_state_machine; gem"aasm"

2.加入栏位：rails g migration add_aasm_state_to_xxx

3.add_column :xxx, :aasm_state, :string

4.add_index :xxx, :aasm_state

5.rake db:migrate

6.设计架构：在 model 里写入 include AASM 注意：两种状态都可指向同一种状态时，可加 []

7.状态通知标签写入 helper

8.建立连接 has_many

9.修改 controller

8/22 用户上传头像

Published on: August 22, 2016

1.1 个 project 包含多个 plan，plan 如果要引用 project_id，需要 rails g migration add_project_id_to_plan

2.rake db:rollback 的用法

3.定义图片的大小可以在 uploaders/image_uploaders.rb 中通过

```
version :xxx do
process resize_to_fill: [xx, xx]
end
```

的方式来进行。

关键点：定义图像大小后要重新上传。

4.在 uploaders/image_uploaders.rb 一定要写上：include CarrierWave::MiniMagick，否则后果很严重！

5.图像显示为圆形的方法：
<img src="xx.jpg" class="img-circle"/ >
6.Tab 键在 Terminal 中可以自动填充
7.修改用户密码的路径是：edit_user_registration_path
8.上传图片的步骤：
- 安装 gem "carrierwave"
- bunle install
- 重开 server
- rails g migration add_xxx_to_xxx
- rake db:migrate
- rails g uploader xxx
- 修改 model
- mount_uploader :xxx, XxxUploader
- 修改 views
- 修改白名单
9.定制图片尺寸步骤
- 安装 gem "mini_magick"
- bundle install
- 重开 server
- 在 uploaders/image_uploaders.rb 一定要写上：include CarrierWave::MiniMagick，否则后果很严重！

第四周总结(8/15-8/21)

Published on: August 21, 2016

课程已过半，喜忧各参半。一些迷茫，学着学着好像想通了，却又生出了很多之前都没有想过的更大的迷惘。
前三周是单打独斗阶段，本周开始做团队项目，学习团队合作。
重新梳理了一下如何开始一个新项目：

- rails new xxx
- cd xxx
- git init
- git add .
- git status
- git commit -m”xxx"
- gem “bootstrap-sass"
- mkdir app/views/common
- modify app/views/layouts/application.html.erb
- rails g controller welcome
- touch app/helper/flashes_help.rb
- gem “devise"
- rails g devise:install
- rails g devise user

- rake db:migrate
- gem “simple_form"
- rails g simple_form:install --bootstrap
- rails g model xxx title:string description:text
- rake db:migrate

第四周学到的概念(8/15-8/21)

Published on: August 21, 2016
本周学到的重要概念是：刻意练习。

其实这个概念从六年前读完《outliers》那本书的时候就知道了，怎样刻意练习一万小时，万维钢老师当时总结了四点：
1. 只在“学习区”学习
2. 大量重复训练
3. 持续获得有效反馈
4. 精神高度集中
觉得很有道理，就记在了笔记里。

2014 年韩寒导演的电影《后会无期》上映后，开始流行一句话“听过很多道理，依然过不好这一生”，搞得自己也很困惑，是呀，我也懂很多道理呀，为什么生活依然一地鸡毛呢？直到有一天看连岳的公众号，有读者在留言区提出这个问题，连岳的回答一针见血，说:“因为你根本就不想实践这些道理。”对自己也是当头棒喝。后来，笑来老师开公众号“学习学习再学习”，写《七年就是一辈子》，里面的一篇文章题目就是《最根本的学习与创作：践行》，这篇文章里提到：想到了，就去做，做到了，才算践行，做不到，没用。

绕了一大圈回来想说的是，虽然自己六年前就知道了这个概念，并且知道了践行这个概念的方法论，却没有去践行，根本没什么用。

最近万维钢老师在得到开专栏精英日课，自己当然在第一天就订阅了。罗辑思维为了推广万老师的专栏，专门做了两期节目的“广告”，也是够拼的。第二期的题目是“怎样成为一个高手”，自己的收获除了“复习”刻意练习这个概念外，是老罗说的另外一句话：“和知识的不断互动才是学习”。老罗提到一个场景，在看微信文章的时候，没时间看完或准备以后再看的就收藏到印象笔记或有道云笔记，但是在处理这些收藏的时候，一定要写自己的感想收获或评论等，哪怕只写一个字也要写，因为通过这种互动才能把这篇文章讲的内容和自己的知识体系对接起来，才算学习。听到这里才明白，自己收藏了那么多文章，却依然不能精进，原来如此。好在什么时候开始都不算晚。践行吧！

顺便贴上关于践行，自己写过的另外一段话：
笑来老师说过阅读速度是一个伪概念，理解速度才是真概念。现在看来，理解速度也是一个伪概念，认知迭代才是真概念（可进一步推论，认知迭代也是一个伪概念，践行才是真概念）。能带来迭代感的概念有三个特征：精深你对某一事物的认知，拓宽你对某一事物的认知，颠覆你对某一事物的认知。围绕认知迭代这个概念来构建方法论：读书，观影，读公众号文章，

学习知识类社群的分享（罗辑思维，新生大学，樊登读书会，混沌研习社，卓老板聊科技，晓松奇谈，吴晓波频道，观复都嘟等）时都可以用认知迭代这个概念来检验收获，并且如果有认知迭代发生一定要及时写下来，至少写成一篇短文，并且践行。

[第四周遇到的大坑（8/15-8/21）]

Published on: August 19, 2016

本周遇到的大坑与富文本编辑器 trim 有关, 安装 trim 之后, 输入的文字在显示出来的时候, 总会在两边带 标签，例如 输入单词 Hello World!，会显示为 Hello World! ，解决方法是在 views 的 index 页面的该项的显示名称前加 simple_format 或用别的 style 显示方法。

trix is easy to drop into Rails with the asset pipeline.

In your Gemfile you need to add the trix gem.

gem 'trix'

bundle install and restart your server to make the files available through the pipeline.

Import Trix styles in app/assets/stylesheets/application.css:

*= require trix

Require Trix Javascript magic in app/assets/javascripts/application.js:

//= require trix

Finally, any place where you would like to use the Trix editor in your forms, just use the trix_editor helper:

f.trix_editor :body

Or if you are using the formtastic gem:

f.input :body, as: :trix_editor

[8/18 错误记录]

Published on: August 18, 2016

1.大括号里写内容要两边加空格

```
-    validates :price, numericality: {greater_than: 0}
+    validates :price, numericality: { greater_than: 0 }
```

2.新建 develop 分支，保持 master 分支干净

3.修改写法 is_hidden

```
-    scope :published, -> { where(:is_hidden => false)}
+    scope :published, -> { where(is_hidden: false) }
```

4.代码之间不要留空行

5.在 html 中单双引号无区别，在 ruby 中有区别

6.代码写法要规范

resources :users do

```
    member do
-       post :promote
-       post :demote
+         post :promote
+         post :demote
    end
end
```

7.rake = ruby make

8.counter_cache

[8/17 continue growth rocket]

Published on: August 17, 2016

1.添加网页的背景图片 :先把图片上传到本地, 路径 :项目名称（如: jdstore）/public/images, 然后在 app/views/layouts/application.html.erb 添加一行代码 <body style="background-image: url(/images/Body_bg.png);">, Done!

2.@users = User.all 与 @user = current_user 的区别

3.devise 自动生成的登录和注册页在哪里？
在 terminal 中切换到项目目录下，执行 rails g devise:views，然后在 app/views 目录下就会生成 devise 目录，devise 的页面就保存在这里。

8/16 开始项目 GrowthRocket

Published on: August 16, 2016

1. 在写管理用户的功能的时候出现错误：
<%= link_to("Admin Users", admin_users_path) %>
在这行代码中不需要写 method: :post，如果写会报错 unknown "create" action.
2. spring stop 在什么情况下使用，spring.rb 的代码为：

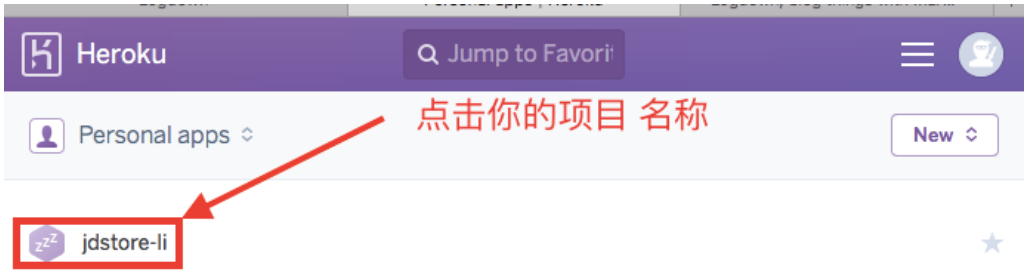
```
%w(
.ruby-version
.rbenv-vars
tmp/restart.txt
tmp/caching-dev.txt
).each { |path| Spring.watch(path) }
```

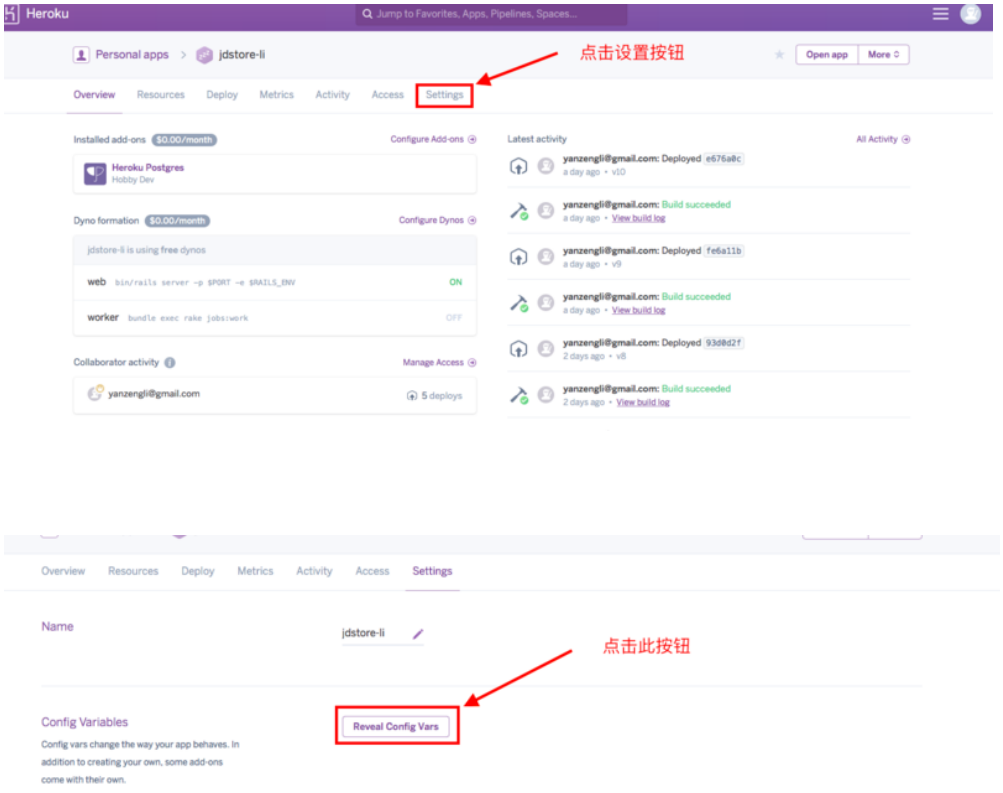
[8/15 建立 landingpage]

Published on: August 15, 2016

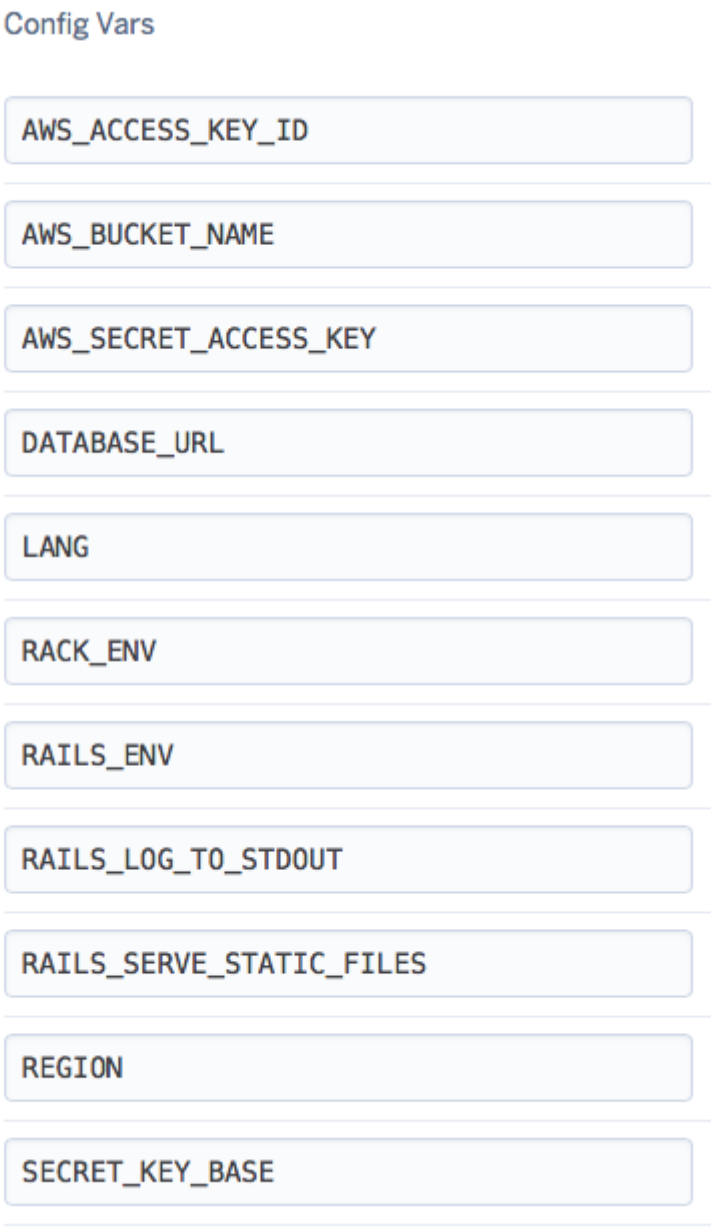
第四周开始重新分组做项目。
经过产品用户调查后，做出的 Landing Page 如下：
[用洪荒之力，让你成为下一个李笑来]
顺序一定要清楚：
1. Why，对应问题：當你用了，做了，學會 XXXX，完成了什麼事後。你最想要達成什麼目標？<== 這個 XXXX 是 學員所提供的服務
2. How，对应问题：當你在做 OOO 時，你最討厭的事情是什麼？ <== 這個 OOO 是痛點

環境
3. What，对应问题：當你學會、買到、知道 XXXX 後，你最想要做什麼事？ <== 這個 XXXX 是 學員所提供的服務
[在 Heroku 如何管理密码]
Published on: August 13, 2016
登录 heroku 之后





就可以看到相关配置信息了



delete_all 与 destroy_all 的区别与用法

Published on: August 13, 2016

QUESTION: I am looking for the best approach to delete records from a table. For instance, I have a user whose user ID is across many tables. I want to delete this user and every record that has his ID in all tables.

u = User.find_by_name('JohnBoy')

u.usage_indexes.destroy_all

u.sources.destroy_all

u.user_stats.destroy_all

u.delete

This works and removes all references of the user from all tables, but I heard that destroy_all was very process heavy, so I tried delete_all. It only removes the user from his own user table

and the id from all the other tables are made null, but leaves the records intact in them. Can someone share what the correct process is for performing a task like this?
I see that destroy_all calls the destroy function on all associated objects but I just want to confirm the correct approach.
Answer：You are right. If you want to delete the User and all associated objects -> destroy_all
However, if you just want to delete the User without suppressing all associated objects -> delete_all
According to this post : Rails :dependent => :destroy VS :dependent => :delete_all
destroy / destroy_all: The associated objects are destroyed alongside this object by calling their destroy method
delete / delete_all: All associated objects are destroyed immediately without calling their :destroy method
Rails 为什么设计两个方法呢？什么情况下该用 delete_all, 而什么情况下该用 destroy_all 呢？这样设计很明显是有用途的，delete_all 是一条 sql 语句删除，如果删除的数据量太多的话，锁表的时间就会很长，会影响对表的使用，如果删除出现问题，回滚起来代价也比较大，但是少了很多网络 IO，网络开销就比较小。后者是先查出所有的数据之后再一条一条的按照主键删除，几乎不会锁表，但是网络开销就比较大，同样的数据量的话，delete_all 删除的速度更快些。
根据自己的数据量以及使用的场景选择合适的方法就好，一般情况下问题都不大。

第三周总结（8/8-8/14）

Published on: August 13, 2016

本周的技术学习主要是围绕【购物车】进行。
周一：练习建立购物车的 action，显示购物商品明细，计算总价。
周二：建立结账页，用支付宝或微信结账，寄送邮件通知。
周三：部署 JDstore 到 Heroku。
周四：消费者可以取消订单并退货。
关于周四的内容还需要再学习消化一下。

课程进行到现在，最大的收获应该是学习方法上的，因为这是可以举一反三的东西。用一个字来概括就是“用”，在知道了最少必要知识之后就去实战，然后在实战中再不断补充需要的知识，狠有成效。

本周感悟：思维深度的三个层次：能想明白，能说明白，能写明白。

产品用户调查

Published on: August 13, 2016
1.你對「誰」提供「什麼服務」？（可以是產品、諮詢服務、課程）
对【区块链技术爱好者】提供【付费订阅内容：BlockChain】
2.請寫下你這個產品的主要 3 - 5 個 feature。(你自己認為可以幫忙「上述對象」完成什麼)
A. 帮订阅者节省时间。区块链技术发展迅猛，资讯繁多，即使是专业从业人员也没有足够的时间关注所有信息。

B. 帮订阅者积累知识。讲解区块链技术的前世今生及关注未来发展。
C. 和订阅者一起思考未来。在众多的区块链投资机会中寻找到适合自己的。
3.How it works。(請寫下至少步驟，可以是背後運作，或者是參與後需要客戶配合的步驟)
A. 创作内容每日更新，积累影响力或寻找有影响力的人合伙
B. 开发产品并进行推广
4.進行對目標客戶問卷調查 （至少找 3-5 個人做問卷調查，不可以自己幻想答案）
A. 當你想了解区块链相关知识時，你最討厭的事情是什麼？
a 找不到靠谱的、简洁的知识来源
b 无关的广告营销
c 听着高大上，但是不知道区块链是啥，干啥用的？
B. 當你知道区块链频道後，你最想要做什麼事？
a 先看看免费内容，再决定是否订阅
b 看看
c 翻看看 看能不能让我迅速明白区块链相关知识。
C. 當你订阅区块链频道後，你最想要達成什麼目標？
a 知道还未被大众接受的有价值的区块链资产并进行投资
b 每一个区块链资产的详细信息以及购买地址
c 通俗易懂的解释，以及可能对我有哪方面的帮助。

第三周学到的概念（8/8-8/14）

Published on: August 13, 2016
本周学到的重要概念有：

1. 贝叶斯定理，告知我们如何利用新证据修改已有的看法。
2. 可证伪性，指科学结论必须容许逻辑上的反例存在。如所有的天鹅都是白色的，就属于可证伪。上帝是存在的就属于不可证伪。
3. GrowthHack。技术创业型团队，通过数据分析和量化指标，来推广产品时所使用的一种市场营销技术。说穿了就是一种揉和了【技术开发】、【产品运营】、【市场行销】的新时代成长手段。并不是所有产品都适合成长骇客，只有快速达到 PMF 的产品才适合成长骇客。成长骇客的重要原则只有两句话：增强信心，降低疑虑。

第三周遇到的大坑（8/8-8/14）

Published on: August 13, 2016
本周遇到的大坑出现在把 jdstore 部署到 heroku 后，本地功能一切正常，但是 heroku 上的支付功能有问题。

根据 heroku logs 的信息提示，原因是 is_paid 这个字段没有上传到 heroku 上面去。之所以没有传上去，是因为本地数据库里通过 rails g migration add_is_paid_to_order 生成的条目消失了（可能有人动我电脑了 O(∩∩)O~），*而本地功能并不受影响。*

解决方案：
首先用 rails d migration 把原来的栏位删除，再新增，但是无效，rake db 无法成功。

其次想直接在 heroku 上面增加栏位，用了 heroku run rails g migration AddIs_paidToOrder is_paid:boolean 的命令，结果也是无效。

*向助教求助后解决，方法是直接在 order 的另外一个栏位 payment method 里面直接把 is*paid 加进去，没有做 rake db ，因为本地的功能是正常的。然后把 heroku 上面的数据库清空，重新 push，搞定！

8/11 部署 jdstore 到 heroku

Published on: August 11, 2016

Objective

關於今天的課程，你記得什麼?完成了什麼?

完成了部署 jdstore 到 heroku，并且可以上传图片。

Reflective

你要如何形容今天的情緒？

上午开始部署 jdstore 到 heroku，主要由于网络问题，导致不能成功，前后折腾四个小时才部署完成，效率太低。

今天的低点：把本地项目上传到 heroku，上传了一个多小时，终于还是失败了...(⊙o⊙)…

Interpretive

今天學到了什麼?

1. heroku 管理后台的命令 heroku console

2. jdstore 在 push 到 heroku 的时候出错不要慌张，要去看错误提示信息

3. 上传到 heroku 时半路停止，就 ctrl+c 取消，到 heroku 上把该 app 给删除了，重新上传却还是传到该 app，向大神求助后解决，cd.git,ls,atom config,找到该 app 删除，就可以重新上传了。

4. duplicateColumn:column "image" of relation "products" already exists.出现这个错误后，是表格重复了，删除重复的表格之后就可以了。

5.figaro 密码管理功能出错，学习了大神的 figaro 配置，但是没理解 cp config/application.yml config/application.yml.example 这句话的意思，只是写了这个命令，却没有到 application.yml 修改配置文件加入配置信息，囧...

6.把配置信息同步到 heroku,执行 figaro heroku:set -e production

今天一個重要的領悟是什麼?

贝叶斯定理能够告知我们如何利用新证据修改已有的看法。

Decisional

有哪些工作需要明天繼續努力?

做“取消订单”功能。

8/10 建立历史订单

Published on: August 10, 2016

ORID 是一個思考框架，利用 ORID 整理今天的思緒。

Objective

關於今天的課程，你記得什麼?完成了什麼?

建立历史订单，对特定的订单进行支付，密码管理，上传到 Heroku。学习 GrowthHack。

Reflective

你要如何形容今天的情緒？

很正常。提交了两个 pull request。

今天的高峰是什麼?
修改了半天路径，一下看到了所有的历史订单。订阅万维刚“得到”专栏。
今天的低點是什麼?
db 出现问题。
Interpretive
我們今天學到了什麼?
1. 变量要用全名
2. 安装 annotate
3. migration 不能手动删除，会出问题。
4. rails s -p 3001 可以打开第二个端口
今天一個重要的領悟是什麼?
万物皆是 CRUD。
Decisional
我們會如何用一句話形容今天的工作
有哪些工作需要明天繼續努力?
发邮件，Heroku.

8/9 建立结账页

Published on: August 10, 2016
ORID 是一個思考框架，利用 ORID 整理今天的思緒。
Objective
關於今天的課程，你記得什麼?完成了什麼?
第二天写购物车功能，学会了如何建立结账页，完成了 story-4。
Reflective
你要如何形容今天的情緒？今天的高峰是什麼?今天的低點是什麼?
今天感到有点慌张，觉得有点赶不上教学的进度（实际上是按照进度在学习的，并没有拉下）。只有当天的日记没有写完，现在是 8/10 在写 8/9 日记。是因为老猫的推荐，昨天晚上去重看了罗辑思维吴伯凡讲人类的历史就是一部通信史的节目，据说里面吴老师提到了比特币的相关内容。结果看了第二遍发现关于比特币吴老师只提了一句话，还是我注意力不集中没听到别的？顺便提一句：在自己的视野范围内，最近在公开场合正面提到比特币的名人至少有三位：马未都，王煜全，吴伯凡。负面提到比特币的名人：罗振宇，懒投资的张磊，长投网的小熊，米课的华超。持负面观点的这几位除华超外都是大概两年前的观点，他们持负面观点的理由与普通人并无本质上的差异，在币圈根本不值一驳，不知道这几位大佬最近的观点有没有改变？
Interpretive
我們今天學到了什麼?
今天一個重要的領悟是什麼?
当天的作业要当天完成，不要负债。
把注意力集中在技术学习上，不要关注太多暂时无关的事情。
Decisional
我們會如何用一句話形容今天的工作
有哪些工作需要明天繼續努力?
做历史订单功能，付款功能，上传到 Heroku 功能。

8/8 学写购物车

Published on: August 8, 2016
学到的知识点主要有：
1. 完成了购物车的基本功能
2. 一键清空购物车
3. 更改购物车内购买数量
4. 没有库存的商品不能购买
5. 购买数量不能大于库存
6. 加深了对 controller 和 view 的关系的理解
Xdite 老师推荐了相关学习网站：
1. confreaks.com
2. railscast.com
3. gorails.com
4. ruby5.com
5. greenruby.com
6. rubyweekly.com
7. rubyinside.com

听课笔记：如何构建商业模式

Published on: August 6, 2016
下午在风利社区收听了瞰见创业营姚老师讲如何构建商业模式的分享。
由于直播效果问题和自己没有集中注意力收听导致并没有听明白，但是听明白一点姚老师在投资上有一套自己的体系，非常牛逼。
我想再收听一次（愿意付费），可是没有机会了。把后半段听到的一些观点记录如下：
1.不看好小米公司的发展。小米公司的 A 轮投资人在公司未进行 IPO 的情况下已经退出，公司的估值从 450 亿美元已经掉到 200 亿美元（不确定是否听清），主要问题是小米的手机是小米公司的核心，但是小米手机的核心竞争力并不强，在这一点没有得到有效改善的情况下，去做小米的生态链是没有用的，别的东西做再好也没用，小米手机如果垮掉，别的生态链上的产品都活不下来。小米的阶段性成功是因为两件事情，第一，踩准了时间点，正好是从功能机向智能机转换的阶段，当时国内没有强的竞争对手，第二，改变了销售渠道，首次把手机的销售渠道搬到了线上。看好华为手机的发展。
2.更不看好锤子手机的发展。如果说小米手机没有壁垒的话，锤子手机就更没有了。认为罗老师应该去做脱口秀，别人学都学不会的技能，他天生骄傲。
3.看好阿里巴巴的发展，马云是战略第一牛人，能够看到五年之后的事。从阿里巴巴到淘宝天猫到蚂蚁金服到阿里云，战略能力无人能及。在战略上，腾讯与阿里差的太远，或者说马化腾与马云差的太远，更不要提李彦宏。腾讯的微信并不是规划出来的一个产品，而是一个比较偶然的事情。微信的数据并不在腾讯云上面。张小龙经常不到深圳去开会。腾讯抓到了一手好牌但是不会打。
4.跟谁学的战略方向错误。A 轮融到了 5000 万美金，但由于战略方向错误，导致钱已经快烧光了，快撑不下去了。
5.看好分答的发展。
6.以上只是姚老师利用他的投资框架和逻辑分析时所举的例子，并不是信口开河。但遗憾的

是，理论没听懂，只记住了一些例子还不一定准确。
7.最重要的是要学习思维模式，不断锤炼自己的价值观，才能做出自己的判断（不管是正确还是错误）。比如说罗振宇就非常看好锤子科技的发展，讲了一大堆的理由，自己听得头头是道 ;现在姚老师又讲了一大堆理由非常不看好锤子科技的发展,也很有道理,自己就懵了。投资是最考验一个人的独立思考能力的活动。如何提高独立思考能力？

第二周总结（8/1-8/7）

Published on: August 6, 2016
第二周结束了，借用笑来老师的话说是“一如既往地快”。
本周在技术学习上的收获：
- 周一周二在练习招聘网站，学习到的新内容是用 carrierwave 上传简历。
- 周三开始练习克隆京东商店。Xdite 老师这次没有像原来一样提供完整答案供我们“照抄”，而是在前面学习的基础上自由发挥。学习到的新内容主要是上传图片及之后的页面排版。
- 开始敢动手去改一些代码了。
本周在鸡汤学习上的收获：
- 笑来老师讲见识决定一切。但问题的关键是你不知道你不知道呀，这个怎么解？Xdite 老师提供了一种方法是去做高一级的决定，听从牛人的指点。像本次的工程师培训班，听说有几个同学都是借钱来上的，这个就属于高一级的决定了。那么再回想一下，自己做过哪些高一级的决定呢？
- 分享的重要性。分享其实也是一个反直觉的概念。一般人会直觉认为我辛辛苦苦学会的东西分享给你了,你就占便宜了,我吃亏了。可事情的真相是你分享出去的东西才是你的东西，你分享出去了，你才留下了。从原理上来讲：首先你分享的过程就是你把学会的东西在大脑里重新整理盘点的过程，越理越顺；其次可以促使你去学习更新的内容，成为进步型人格而不是表现型人格。

第二周学到的工具和概念(8/1-8/7)

Published on: August 6, 2016
工具
1. feedly，一个免费的 RSS 订阅软件，打开页面后，按 shift+?可以看到快捷键。
2. sizeup，窗口管理软件，尤其适用于双屏幕。
3. 一篇文章。

高效 MacBook 工作环境配置

4.多态，一个有情怀的工具，该软件的 slogan 是英国哲学家罗素的名言“参差多态，乃幸福本源”，很明显也是该软件名称的出处(我本人有一段时间也特别喜欢这句话，想起武汉著名的咖啡馆-参差咖啡)，和本软件实现的功能 cross the GreatWall 联系起来，你懂的。。。O(∩_∩)O
概念
1. 理解了`col-md/lg/sm/ex-number`的含义
2. 把两个`div`并排放置的方法：设置左边`div`样式为`float="left"`
3. 理解了`navbar-default`和`navbar-inverse`的区别
4. erb\rb\html\css 不同的语言用不同的注释方法

5. 定义图片居中的方法：`class="row text-center"`
6.语句：`<%= image_tag product.image.thumb %>`

第二周遇到的大坑（8/1-8/7）

Published on: August 6, 2016
本周遇到的大坑都与图片上传有关，第一个大坑在 8/3 日记中已经记录，本篇记录第二个大坑，也是出现在上传图片的过程中。
具体过程是这样：在 admin 后台点击上传图片按钮，添加图片后在前端不能显示出来，只能显示图片的标题，不知何处出错。前提是原来上传的图片可以正常显示，新添加的图片无法显示，原来上传的图片可以正常删除。
向本组同学请教后，得知原因如下：原来上传的图片是通过非 admin 上传的，但是在后来的修改中把非 admin 上传的功能给取消了，取消后没有再上传过图片。现在需要上传图片，只能通过 admin，但是 admin 的添加新商品页面没有上传图片的栏位，于是我到 views/admin/products/new.html.erb 页面添加一行代码<%= f.input :image %>，这样 admin 的添加新商品页面出现了上传图片的栏位，但是提交图片后不能在前端显示。
问题出在 image 没有添加到 controller/admin 的 product_params 的 permit 中。

8/4 黑客松经验

Published on: August 4, 2016
今天郑老师介绍了她参加 FacebookHackathon 的经验。
首先让同学们抢答什么是好专案和什么是烂专案，由于加分和奖品（笑来老师签名书）的吸引，各组同学回答踊跃，课堂气氛高涨。
其次让每位同学排列管理专案时五件事情的轻重缓急，本题我也答对了，给本组增加 5000 分。XD
五件事情的正确排列是：
1. 先确立要做出一个什么样的产品
2. 盘点资源（人员，时间。。。）
3. 找出风险部分
4. Must have
5. Shoule have
第三，提出一个问题，在整个 hackathon 的过程中，最重要的一步是什么，在一片嘈杂声中二组的一位同学抢答成功，正确答案是 presentation.
第四，再提出一个问题，本次 Hackathon 中，总共做了几个功能，我在心里默想的应该是挺多的，可能在 3-10 个，因为这样才牛逼呀，谁知正确答案是 1 个。这两个问题的答案对我来说都是“情理之中，意料之外”呀。。。
第五，开始讲 hackathon 的具体过程，其中很重要的是在中午休息时给对手心里压力，O(∩_∩)O~。。。
第六，讲了别的团队是如何失败的，大牛太多，意见无法达成一致是很重要的失败原因

8/3 练习京东商店

Published on: August 4, 2016

今天开始练习京东商店，郑老师这次没有给答案，但是可以参考前面练习过的招聘网站来进

行实做。今天的练习中，主要出现了三个问题，都出现在上传图片的过程中，上传图片也是本次练习的新增加内容。

问题一

在新增 image_Uploader 的过程中，由于教程上面的指导中多打了一个 Uploader,导致在程序运行的时候出现问题，并且经过各种查找及搜索都没有能够解决，后来向大神求助，一句话就解决了问题。

问题二

第二个问题应该可以候选本周遇到的最大的坑。在上传图片的过程中，给 image 新增加了一个 attachment 的属性（是参考给 resume 新增加了一个 attachment 的属性），结果是图片显示上传成功了，结果在前端却显示不出来。正确的做法是直接给 product 增加一个 image 属性，把图片直接上传到 image 就可以了。更深层次的原因是对整个 mvc 机制的运行理解还不太深刻，彼此之间的关系还不太明晰，参考招聘网站没有进行灵活修改就容易出错。

问题三

对于 html/css 的学习和理解还需进一步深入。

8/2 继续练习招聘网站

Published on: August 2, 2016

今天继续练习招聘网站，做完了第二遍练习，可以按照教程上的步骤跟下来，但是不看教程的话基本还是两眼一抹黑。

郑老师今天讲了个人成长秘技，如果能够早几年按照成长秘技来做，人生应该会不一样。

1.找到正确的工作->练功时间暴增

2.找到自己的天命->努力方向正确

3.主动扛起前进责任->学到许多组合技

4.时常主动分享->360 度补血跳级（分享一定要真诚）

5.回答别人的问题->得到百倍灵感

controller 里面的 render 的用法

Published on: August 2, 2016

1.render :text

render 纯文字。若是要把复杂的 code 包在 js 里面的时候，可能会出现单双引号打架之类的问题，所以用 render 的方式吐出 string 也许是一种方法

render "嗨，自己的文字自己 render"

2.render :template

render 的中文意思是【渲染】，所以 render template 就是【渲染】指定的模板，render 的特性是不跑 controller action,直接将该 action 下预设的模板传出来，我们也可以自己指定要哪个特定的模板。

render views/foo/bar.html.erb

render"foo/bar"

render 同一个 controller 下的 action 的 view，用 symbol 和 string 都是一样的：

render :foobar

render "foobar"

3.render :layout

Rails 预设的 layout 是 app/view/layouts/application.html.erb 这个档案。

但有时候我们会希望预设的版型不一样，比方说我们的 admin 页面 head 内不希望加上 GA

和一些有的没的追踪 script。
这时候我们就可以建立一个新的 layout 版型 app/view/layouts/admin.html.erb
只要在 controller 中指定使用 admin layout 即可：
class AdminsController < ApplicationController
layout"admin"
end
view 里面的 render 的用法
Published on: August 2, 2016
1. 在 view 里面的 render80%的情况表示调用 partial
`<%= render "item" %>`等价于`<%= render :partial => "item" %>`

2. 在 view 里面的 render15%的情况表示调用 collection partial
正常的 partial 只会将内容 render 出来一次，collection partial 则会自动 count 我们给予的物件，并 rendercount 的次数，可以省去在 partial 内再写 block 或 helper 的繁琐，让 partial 更简洁。

`<%=render :collection => @jobs %>`展开就是

<% @jobs.each do |job| %>
<%= render :partial => "job", :locals => { :job => job } %>
<% end %>

3. 在 view 里面的 render5%的情况表示调用 layout
scope 的用法
Published on: August 2, 2016
scope 的作用就是将时常使用或是复杂的 ORM 语法组合成懒人包，这样下次要用的时候只要把懒人包拿出来就可以了，举例说明：
class Topic < ActiveRecord::Base
scope :recent, -> { order("created_at DESC")}
end
上面这段 code 我们定义了 recent 这个 scope，以后我们只要下 recent 这个指令就等于下 order("created_at DESC)是一样的。

使用情景
- 当有过于复杂的资料查询
- 当有重复使用的资料查询

使用方式

没带参数的方式
class Post < ActiveRecord::Base
scope :published, -> { where(published:true) }
end

带有参数的方式
class Post < ActiveRecord::Base

```
    scope :created_before, ->(time){where(created_at <?", time)}
end
```

可以串接在一起，顺序没有影响

```
class Event < ActiveRecord::Base
    scope :published, -> {where(published: true)}
  scope :created_before, ->(time){where(created_at <?", time)}
end
Event.published.created_before(time.now)
```

参考资料

- http://guides.rubyonrails.org/active_record_querying.html

8/1 应聘者提交简历

Published on: August 2, 2016

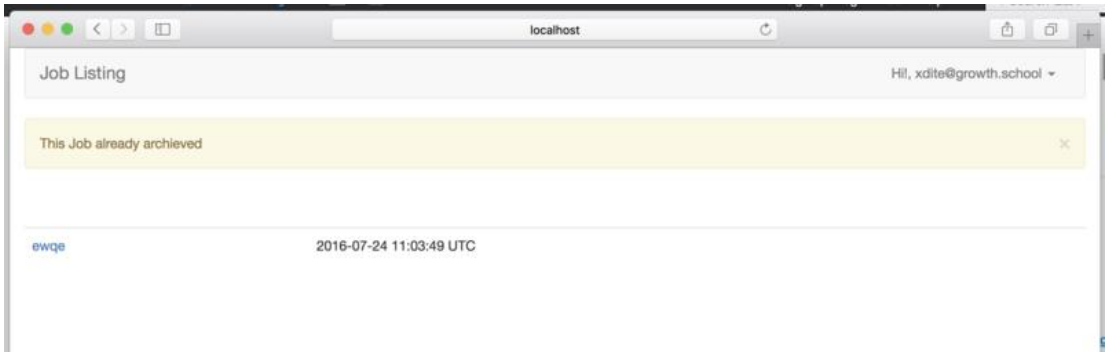
目标： 1. 应征者可以在列表看到按照【最高薪资条件排序】的职位
2. 应征者可以在列表看到按照【张贴日期排序】的职位
3. 应征者可以提交自己的简历
4. Admin 可以看到哪些使用者提交了自己的简历
步骤： 1. 先将薪资上限、薪资下限也加入首页职缺列表
2. 加上下拉选单
3. 在 JobsController 里面制造判断式，决定要送出何种资料
4. 重构重复的程式码，建立 publishedscope
5. 建立 recentscope
6. 建立投递简历的链接
7. 产生简历 resume 的 model
8. 将 Resume 与 user/job 挂起来
9. 建立简历表单
10.加入简历上传功能（按照 carrierwave）
11.重开 rails server
12.新增 attachment 栏位在 resume 这个栏位上
13.挂上 AttachmentUploader 到 Resume 上
14.修改 resume_params
15.使用.gitignore 屏蔽使用者产生的档案
16.在 Admin/jobs 列表加入薪资上限、下限、收到多少份简历
17.可以看到职缺里面有谁投过简历
18.设计简历列表
领悟： 郑老师讲两个重要的人生感悟：1.看说明书 2.最可怕的是你不知道你不知道什么东西，这样就陷入了一个死循环，所 以要听从高人的指点去做高一级的决定。
郑老师推荐一本书《师父》。

7/31 有关 flash 中的 warning

Published on: July 31, 2016

错误描述：在写招聘网站的限制已经【被隐藏的工作】，不可以被看到一步中，无法看到下

图：



在第一遍练习中就出现过这个错误，当时问题已经解决，就是把路径中的 admin 直接删除就可以达到效果，非 admin 无法看到被隐藏的工作。但是第二遍练习中又出现了此问题，把路径中的 admin 直接删除却无法看到图上的效果，而是直接跳到了 localhost:3000 这个页面。

继续排查原因，发现没有被隐藏的条目在路径中删除 admin 是不会跳到 localhost:3000 这个页面的，而被隐藏的条目在路径中删除 admin 后会跳到 localhost:3000 页面。并且在 show 这个 action 中，

```
if @job.is_hidden
   flash[:warning] = "This Job already archieved"
   redirect_to root_path
end
```

删除如上代码后被隐藏的条目也不会跳到 localhost:3000 页面。

这样，说明问题出在上面这段代码中，经过仔细排查发现：

```
module FlashesHelper
FLASH_CLASS = { alert: "danger", notice: "success", warning:"warning"}.freeze
def flash_class(key)
FLASH_CLASS.fetch key.to_sym, key
end
def user_facing_flashes
flash.to_hash.slice"alert", "notice","warning"
end
end
```

在将 bootstrap 的 js 提示套件 bootstrap/alert[挂]进专案时，在如上一段代码中单词 warning 出现拼写错误，导致了点击被隐藏工作，直接跳到 localhost:3000 页面，而不出现如图所示的警告。

[7/30 错误记录]

Published on: July 30, 2016

1. class="caret"；类 caret 显示下拉功能
2. 在 Atom 中，ctrl+shift+k 删除当前行；command+shift+d 复制当前行
3. `params.require(:job).permit(:title, :description)`其中`job`不加 s
4. `if !current_user.admin?`其中感叹号忘记写两次。
5. `|job|`忘记写后面的`|`
6. 在 terminal 中，输入命令出错，提示 dquote,解决办法 ctrl+c,终止当前项目运行

7. 在管理员登录情况下，一切正常；在非管理员登录情况下，增加职位和修改职位都无法提交成功，原因在非管理员的
controller 中忘记加上白名单栏位，只在管理员 controller 中增加了。
8. 括号和逗号都要用英文输入法，中文是错的且难以识别出错误。
9. link_to 后面加括号时不能有空格。
10. 点击”登出“按钮没反应，没有实际的”登出“功能。原因：在 application.js 中挂 bootstrap/dropdown 时出现拼写
错误

第一周总结（7/15-7/29）

Published on: July 29, 2016

从 7/15 日做作业开始算起，到现在整整两周的时间了。应该说进步还是挺大的，从刚开始什么都不懂的一脸懵逼到现在略微懂一点点了。
关于技术方面的进步基本都记录在了每天的错误日志里。
关于学习方法学到的最重要的两条，写错误日记，看别的同学的错误日记。坚持这两条，可以达到郑老师所说的 Bittorrents 式的成长。

7/29 小组讨论会

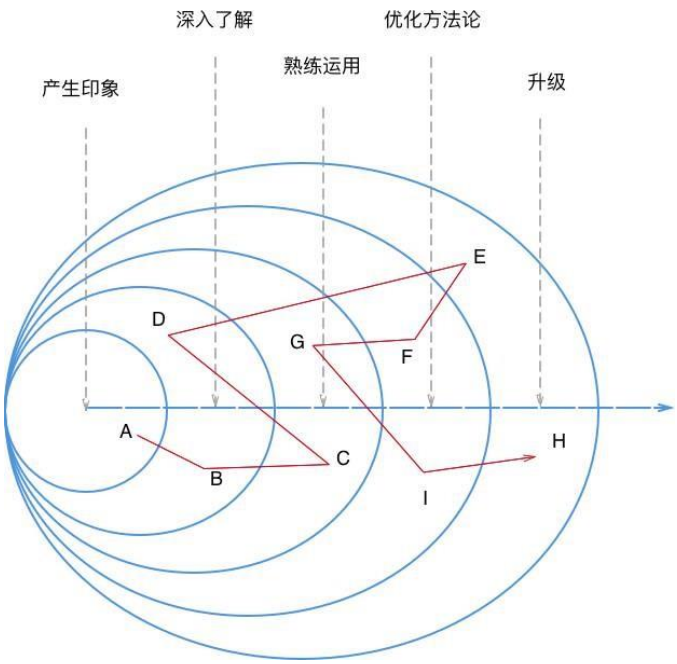
Published on: July 29, 2016

自学了两个 Markdown 技巧：
1.空四个空格可以显示代码
2.加反引号可以在段落中显示代码
召开了第一次小组讨论会：
1.知道了 rake db:migration 的情况下，在编辑器里面改代码是没有用的，需要重新生成一个 migration.这是一个高质量的坑。以后需要注意。
2.本组学霸介绍了很多可以提高效率的插件和快捷键，虽然暂时还用不上，但知道有这回事。
3.可以在 timeline 中学习别的同学的博客，从别人的错误中更好的提高自己。
4.rails 错误退出如何处理？
lsof -i tcp:3000 然後找到 PID 之後用 kill -9 <PID>(指读取到的正在运行的 PID)

第一周学到的最棒的概念--折回

Published on: July 29, 2016

通过本周的学习，忆起笑来老师讲过的一个重要的概念，叫“Folding Back”，“折回”。
Pirie & Kieren 在 1994 年提出一个数学理解成长模型，用来解释人们在学习数学的时候分别走过的若干个阶段及其路径。也就是说，从入门到精通这个过程，并不是一路走过去的，而是在不同的阶段不得不“折回”。折回的原因是为了更深刻地理解、为了去除部分错误的理解、为了加强某些必要方面的理解，有时甚至是为了重新认识。



第二个重要的概念是注意力。
源于笑来老师的讲话，人生中最宝贵的是不可再生的资源，金钱不是最重要的，因为金钱是可以再生的，时间比金钱重要，因为时间是不可再生的，比时间更重要的是你的注意力，因为每个人的时间都是固定的，你把时间花在了什么地方就特别重要，珍惜自己的注意力。

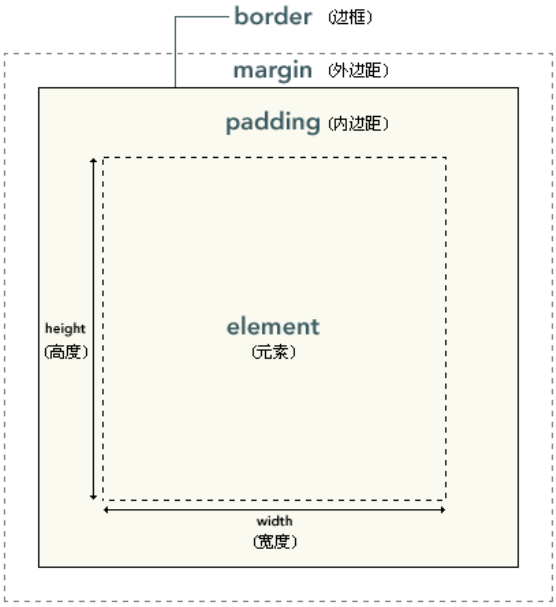
第一周遇过的最大的坑

Published on: July 29, 2016

回顾了本周的学习过程，遇到的坑主要有：
1.拼写错误，包括忘记加-s，单词中字母顺序写反，少写一个字母，-与_弄混淆，忘记冒号，括号，引号，感叹号等。
2.随时保存。写完一段代码要 command+s。全部保存的快捷键是 option+command+s。
3.没有使用 Timeline，可以看到别的同学的作业和博客，从中可以学习到很多的有价值的内容。

CSS 自学作业

Published on: July 29, 2016
1.margin 与 padding 的差异
margin 指的是外边距，围绕在元素边框的空白区域。
padding 指的是内边距，在边框和内容区之间。
2.什么是 box model
CSS 框模型规定了元素框处理元素内容、内边距、边框和外边距的方式。



W3School.com.cn

3.为什么要使用 em 而非 px 来定义字的大小？
font-size 值可以是绝对值或相对值。
绝对值
- 将文本设置为指定的大小
- 不允许用户在所有的浏览器中改变文本大小（不利于可用性）
- 绝对大小在确定了输出的物理尺寸时很有用
相对大小
- 相对于周围的元素来设置大小
- 允许用户在浏览器改变文本大小
用 px 定义的文本大小，在 firefox,Chrome,and Safari 中，可以进行重新调整，但是在 Internet Explorer 中不行。
如果要避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels.
浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。
可以使用公式将像素转换为 em：pixels/16=em.
4.h1 {margin : 10px 0px 15px 5px;}
margin-top: 10px
margin-right: 0px
margin-left: 5px
margin-bottom: 15px

7/28 修改招聘网站

Published on: July 29, 2016

今天学习到的主要内容有：
1.加入 sidebar
2.限制已隐藏职位不能存取

3.将职位状态换成图示
4.设计【隐藏】【重新显示】的按钮
在做第二步限制已隐藏职位时出现了问题，情况是：
自己的代码写好了，但是不能出现“the job already archieved”样的黄色警告，经过向助教请教后，得到解决，并不是自己的代码出问题了，而是自己的访问路径出现了问题，在 admin 的路径下访问时不会有这样的黄色警告的，因为 admin 是有权限的，把访问路径中的 admin 删除，变成路人模式，就可以出现黄色警告了。
还有两个小问题：
1.在定义 pulish 和 hide 按钮的时候：
def publish
@job = Job.find(params[:id])
@job.publish!
redirect_to :back
end
没注意到@job.publish 后面的感叹号，导致程序不能运行。
2.在 routes.rb 里面家 publish 和 hide 的时候，没注意到这两个词的首字母都是小写，导致程序不能正常运行。也可能是 atom 编辑器自动生成了首字母大写，而自己没有发现调整过来。

HTML 自学作业

Published on: July 28, 2016
1.解释 div/span 的不同
div 元素是块级元素，它是可用于组合其他 HTML 元素的容器。
div 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。
div 元素的另一个常见用途是文档布局。
span 是行内元素。

<h2>London</h2>
London is the capital city of England. It is the most populous city in the United Kingdom , with a metropolitan area of over 13 million inhabitants.

<h1>My Important Heading</h1>
2.解释 class/id 的不同
class 和 id 都是标签的属性。class 是规定元素的类名；id 规定元素的唯一 id。
3.解释 p/br 的不同
是段落标签，是块级元素。

是空的 HTML 元素，在希望不产生一个新段落的情况下进行换行，需使用
标签。推荐使用
而不是
4.如何使用 table 排版
<table>是表格标签，每个表格均有若干行（由<tr>标签定义），每行被分割为若干单元格（由<td>标签定义）。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。如果不定义边框属性，表格将不显示边框。

表格的表头使用<th>标签进行定义。

7/27 写招聘网站

Published on: July 27, 2016

今天学习到的内容主要有：
1.controller 里面的 method 叫 action,外面的叫 method.
2.devise 自动生成的登录和注册页在哪里？
在 terminal 中切换到项目目录下，执行 rails g devise:views，然后在 app/views 目录下就会生成 devise 目录，devise 的页面就保存在这里。
3.如何查看通过 rails g model xxx 创建出来的表的结构？
在 Gemfile 中添加 gem "annotate"，然后在 terminal 中切换到项目目录下，然后执行 bundle install 进行安装，安装好之后执行 annotate。然后打开 model 文件，比如 app/model/user.rb， 就可以看到在文件最上方已经生成好了 users 表的表结构信息。
4.如何修改 last update 的时间为北京时间？
在 application.rb 中添加以下两条配置:
config.time_zone = 'Beijing'
config.active_record.default_timezone = :local
今天的重要领悟：混乱是学习的常态，学习是拼图。
7/26 日记
Published on: July 26, 2016
今天的课程所学到的内容主要有：
1.一些常用的命令包括 :cd;ls -al;pwd;rails s;rails g;rails console,分别代表的含义是转换目录；引出目录下的文件；确认现在所在的目录；打开本地服务器；创建新目录；后门。
2..erb 是一种理解 ruby 在做什么，既可写 html 又可写 ruby 的语法。
3.git 常用命令：git init;git add README;git status;git add .;git commit
4.<%= @post.title%>和<% @post.title %>的区别，前者是印出，后者是执行。
遇到了一个问题用 Rails s 竟然打不开本地服务器，原因还没明白。
7/25 日记
Published on: July 25, 2016
笑来老师来课堂上讲话，要点如下：
1.关于全栈工程师的定义：并不是说要让你学会所有的相关技能，而是说要学到刚好够用。在工程师的工作中，编码只占 20%的比例。举了 knewone 的例子，自带流量。举了道哥的网站 just-dice 的例子说明在金融的世界中信用是最重要的。
2.最好的学习方式就是花时间让别人弄明白。
3.学习的一个重要技巧就是学习你所学的东西的历史。
4.大众创业是对的，很多人反对只是害怕别人成功。
5.人生三大坑：莫名其妙凑热闹，心急火燎随大流，操碎了别人的心肝。
6.只读英文文档。
7.时间比金钱重要，比时间更重要的是你的注意力。
关于今天的课程，学到了两点，user story 和 Fork。完成了 pull request。
明天继续学习 html 和 css。
出现两个【登出】按钮

Published on: July 22, 2016

在制作“My Group”按钮的过程中，出现了如图所示的错误，“My Group”按钮没有出来，却出来了两个【登出】按钮，经过排查原因如下：

```
<ul class="dropdown-menu">
<li> <%= link_to("My Groups", account_groups_path) %></li>
<li class="divider"> </li>
<li> <%= link_to("登出", destroy_user_session_path, method: :delete) %> </li>
</ul>
</li>
```

在以上这段代码中，少写了最后一个，但是纵观全段代码：

```
<nav class="navbar navbar-default" role="navigation">
<div class="container-fluid">
<!-- Brand and toggle get grouped for better mobile display -->
<div class="navbar-header">
<a class="navbar-brand" href="/">Rails 101</a>
</div>
<!-- Collect the nav links, forms, and other content for toggling -->
<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
<ul class="nav navbar-nav navbar-right">
<% if !current_user %>
<li><%= link_to("注册", new_user_registration_path) %> </li>
<li><%= link_to("登入",new_user_session_path) %></li>
<% else %>
<li class="dropdown">
<a href="#" class="dropdown-toggle" data-toggle="dropdown">Hi!,
<%= current_user.email %>
<b class="caret"></b>
</a>
<ul class="dropdown-menu">
<li> <%= link_to("My Groups", account_groups_path) %></li>
<li class="divider"> </li>
<li> <%= link_to("登出", destroy_user_session_path, method: :delete) %> </li>
</ul>
</li>
<ul class="dropdown-menu">
```

 <%= link_to("登出", destroy_user_session_path, method: :delete) %>

<% end %>

<!-- /.navbar-collapse -->

<!-- /.container-fluid-->
</nav>
自己少些的这个
好像是多余的，因为并没有相对应的<li>.具体原因还需再核实。

[实际操作【加入退出】群组]
Published on: July 22, 2016

按照教材的要求全部编程之后，出现如图所示的问题，就是并没有出现加入群组的按钮，经过排查，问题如下：

<% if current_user && current_user.is_member_of?(@group) %>
<label class="label label-success">群组成员</label>
<%= link_to("Quit Group", quit_group_path(@group), method: :post, class: "btn btn-default")
%>
<% else %>
<label class="label label-warning">不是群组成员</label>
<%= link_to("Join Group", join_group_path(@group), method: :post, class: "btn btn-default")
%>
<% end %>

在以上这段代码中，少输入了一行：
<%= link_to("Join Group", join_group_path(@group), method: :post, class: "btn btn-default")
%>

修改之后：

分页功能出错

Published on: July 22, 2016

在运行分页功能时出现错误，经过排查发现原因如下：

```
<tbody>
<% @posts.each do |post| %>
<tr>
  <td><%= post.content %></td>
  <td><%= post.user.email %></td>
  <td><%= post.created_at %></td>
</tr>
<% end %>
```

在这段代码中的<% @posts.each do |post| %>中的 posts 写成了 post.

未出现【提交】按钮

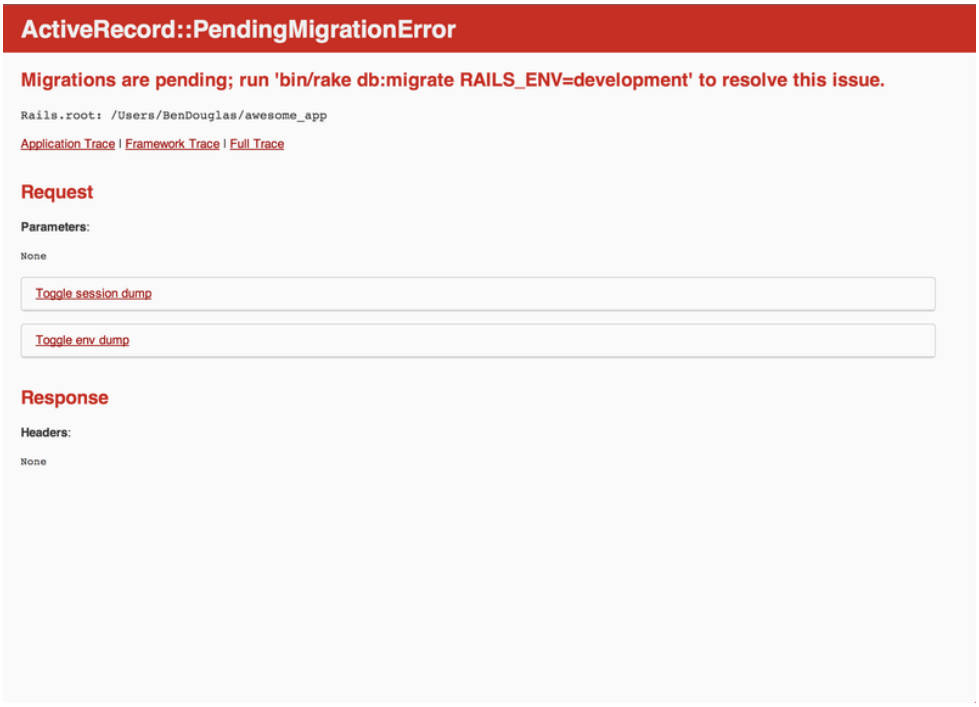
Published on: July 22, 2016

在“发表实际文章”一节中，程序运行时出现了如图所示的问题，一切正常，但是没有提交按钮，无法提交内容，经过检查，发现是漏掉了一段代码如下：

<%= f.submit "Submit", disable_with: "Submitting...", class: "btn btn-primary" %>

ActiveRecord::PendingMigrationError

Published on: July 22, 2016



在程序运行过程中，出现了如图所示的错误。

经过查找获得解决方法如下：运行 rake db:reset,然后运行 rake db:migrate

删除一个分支的命令

Published on: July 21, 2016

在写作业的过程中创建了很多的分支，如何能够删除一个分支呢？

1. git checkout 分支名称，首先用这个命令转到别的分支，因为一般总是想删除现在所在的这个分支。
2. git branch -D 分支名称，用这个命令就可以删除你不需要的分支了。

group and groups

Published on: July 21, 2016

在“限制「标题为空」的文章，不能被送出”一节中，程序运行时出现了问题，经查找原因如

下：

```
<tbody>
  <% @groups.each do |group| %>
  <tr>
    <td>#</td>
    <td><%= link_to(group.title, group_path(group)) %></td>
    <td><%= group.description %></td>
    <td>
      <%= link_to("Edit", edit_group_path(group), class: "btn btn-sm btn-default") %>
      <%= link_to("Delete", group_path(group), class: "btn btn-sm btn-default", method: :delete,data: { confirm: "Are you sure?" }) %>
    </td>
  </tr>
  <% end %>
</tbody>
```

以上为正确的代码，自己的错误出现在<% @groups.each do |group| %>中，在写 groups 的时候忘了加-s，导致程序不能运行。

[两个冒号连在一起]

Published on: July 21, 2016

在练习建立讨论群的架构时，其中建立 index action 以及 index 的 view 中的一段代码

```
<%= link_to("Delete", group_path(group), class: "btn btn-sm btn-default", method: :delete
, data: { confirm: "Are you sure?" } )%>
```

在抄写的时候漏掉了 delete 前面的冒号，只写了 method 后面的那个而没有写 delete 前面的那个冒号，在程序运行的时候就出现了问题，经过排查发现是少写了一个冒号，冒号所代表的含义由于时间关系并没有彻底搞明白，留待以后解决。

[注意拼写问题]

Published on: July 21, 2016

写完了第二遍的作业，用了超过 10 个小时的时间，整个过程还算比较顺利，主要是出现了一些拼写上的错误，导致在运行的时候会出现各种各样的问题。主要有:

simple_form 写成了 simple-form

format 写成了 fromat

group 写成了 grop

还有会漏掉冒号，半个括号，半个分号等

学会了一个 heroku 删除 app 的命令：heroku apps:destroy --app app 名字

[增加一个【扣分】按钮]

Published on: July 17, 2016

自己独立完成了“增加一个【扣分】按钮”这个作业，还是有点成就感的，把中间所犯的错误记录如下：

步骤一：

参照加分项目的编辑：

```
def upvote
@topic = Topic.find(params[:id])
```

@topic.votes.create
redirect_to(topics_path)
end
自己复制如下：
def downvote
@topic = Topic.find(params[:id])
@topic.votes.create
redirect_to(topics_path)
end
步骤二：
经过 rake routes 检验，最终在 post"upvote"下面增加了一行 post"downvote"显示成功。
步骤三：
在<td><%= button_to '+1', upvote_topic_path(topic), method: :post %></td>下面增加了一句：
<td><%= button_to '-1', downvote_topic_path(topic), method: :post %></td>
运行错误，检查编辑，找不出问题在哪。
1：在 atom 里，把 def downvote 移到了 def upvote 上面，运行成功了，但是当点击“-1”这个按钮时，投票总数的实际结果是“+1”，就是说不管点“+1”还是点“—1”，实际结果都是“+1”，在这种情况下继续寻找原因。
2：把<td><%= button_to '+1', upvote_topic_path(topic), method: :post %></td>这个语句里的 post 修改为 destroy，结果不行，修改为 delete 结果还是不行。
3：从头复习了一遍教程内容，看到一句话：
删除一笔 topic 的投票记录：
my_topic.votes.first.destroy
就这样把 def downvote 里面的@topic.votes.create 修改成@topic.votes.first.destroy 成功了。
Hello World
Published on: July 15, 2016
Hi, This a demo post of Logdown.
Logdown use Markdown as main syntax, you can find more example by reading thisdocument on Wikipedia
Logdown also support drag & drop image uploading. The picture syntax is like this:

Bloging with code snippet:
inline code
Plain Code
puts "Hello World!"
Code with Language
puts "Hello World!"
Code with Title
hello_world.rb

puts "Hello World!"

MathJax Example

Mathjax

$$x = \_b \pm b2 \_4ac \_\_\_ \_\_\_\_\_ \sqrt{2a}$$

Inline Mathjax

The answser is $a2+b2=c2$

Table Example

Tables	Are	Cool
col 1	Hello	$1600
col 2	Hello	$12
col 3	Hello	$1

delete_all 与 destroy_all 的区别与用法

Published on: August 13, 2016

QUESTION: I am looking for the best approach to delete records from a table. For instance, I have a user whose user ID is across many tables. I want to delete this user and every record that has his ID in all tables.

u = User.find_by_name('JohnBoy')

u.usage_indexes.destroy_all

u.sources.destroy_all

u.user_stats.destroy_all

u.delete

This works and removes all references of the user from all tables, but I heard that destroy_all was very process heavy, so I tried delete_all. It only removes the user from his own user table and the id from all the other tables are made null, but leaves the records intact in them. Can someone share what the correct process is for performing a task like this?

I see that destroy_all calls the destroy function on all associated objects but I just want to confirm the correct approach.

Answer：You are right. If you want to delete the User and all associated objects -> destroy_all

However, if you just want to delete the User without suppressing all associated objects -> delete_all

According to this post : Rails :dependent => :destroy VS :dependent => :delete_all

destroy / destroy_all: The associated objects are destroyed alongside this object by calling their destroy method
delete / delete_all: All associated objects are destroyed immediately without calling their :destroy method
Rails 为什么设计两个方法呢？什么情况下该用 delete_all, 而什么情况下该用 destroy_all 呢？
这样设计很明显是有用途的，delete_all 是一条 sql 语句删除，如果删除的数据量太多的话，锁表的时间就会很长，会影响对表的使用，如果删除出现问题，回滚起来代价也比较大，但是少了很多网络 IO，网络开销就比较小。后者是先查出所有的数据之后再一条一条的按照主键删除，几乎不会锁表，但是网络开销就比较大，同样的数据量的话，delete_all 删除的速度更快些。
根据自己的数据量以及使用的场景选择合适的方法就好，一般情况下问题都不大。

Published on: August 28, 2016