



Pete Helgren
pete@valadd.com

Ruby On Rails on i

Value Added Software, Inc
801.581.1154

18027 Cougar Bluff
San Antonio, TX 78258





Agenda



- Primer on Ruby
- Review Prerequisites
- Install Ruby/JRuby
- Install Rails
- Install Activerecord-JDBC (DB Connectivity)
- Review Rails (quick overview)
- Build a Blog Application
- Run all of it
- Play (a bit)





Ruby Primer



- What is Ruby?
 - Object oriented language first released in 1995 that is “a scripting language that was more powerful than Perl, and more object-oriented than Python” - Yukihiro Matsumoto
 - Find out more at <http://www.ruby-lang.org/>
- What is Rails?
 - Rails is an open source Ruby framework for developing web-based, database-driven applications.
 - Stresses DRY development and convention over configuration (leverage reflection and discovery).



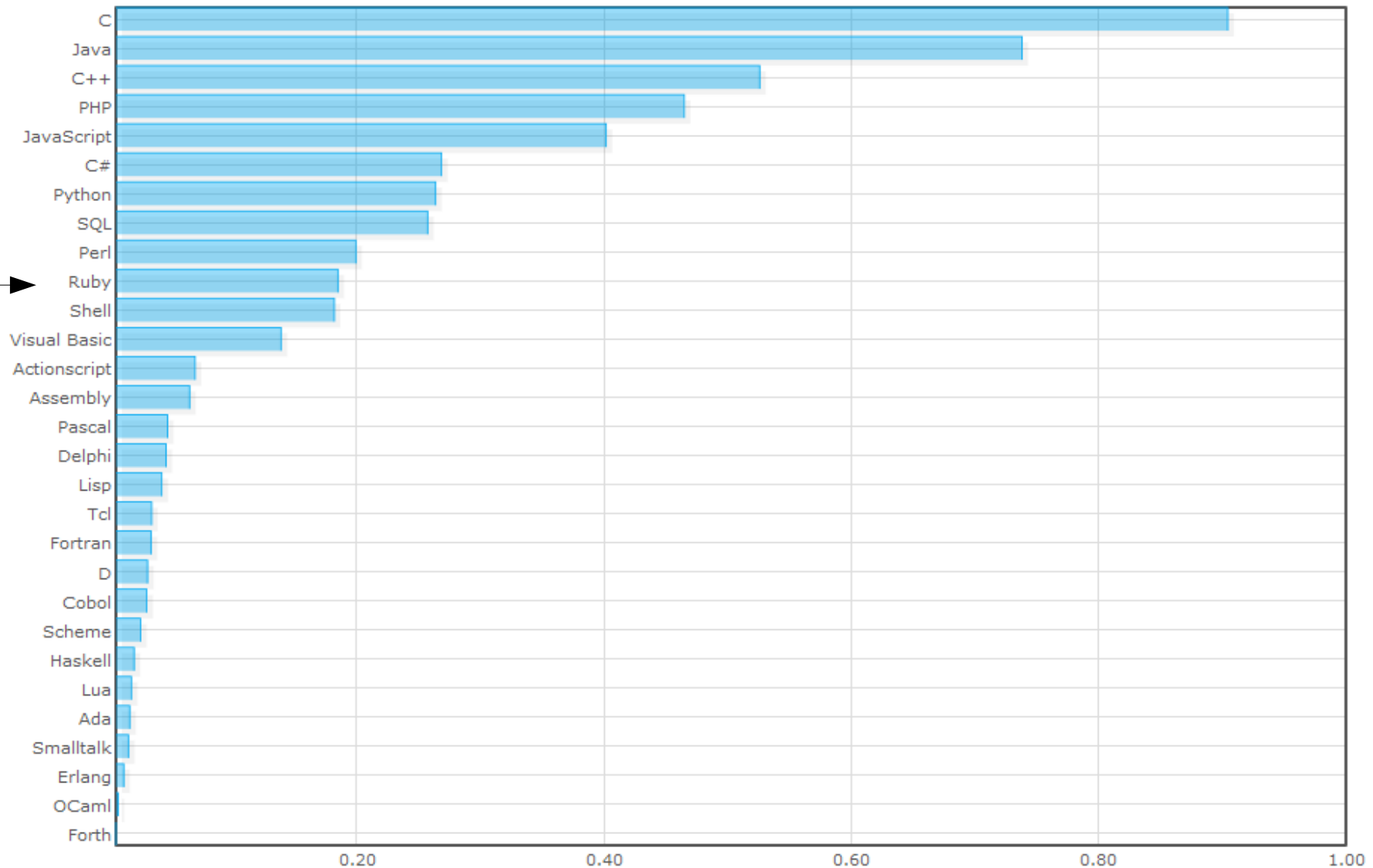


Ruby relative to other languages



2009 (from langpop.com)

This is a chart showing combined results from all data sets.

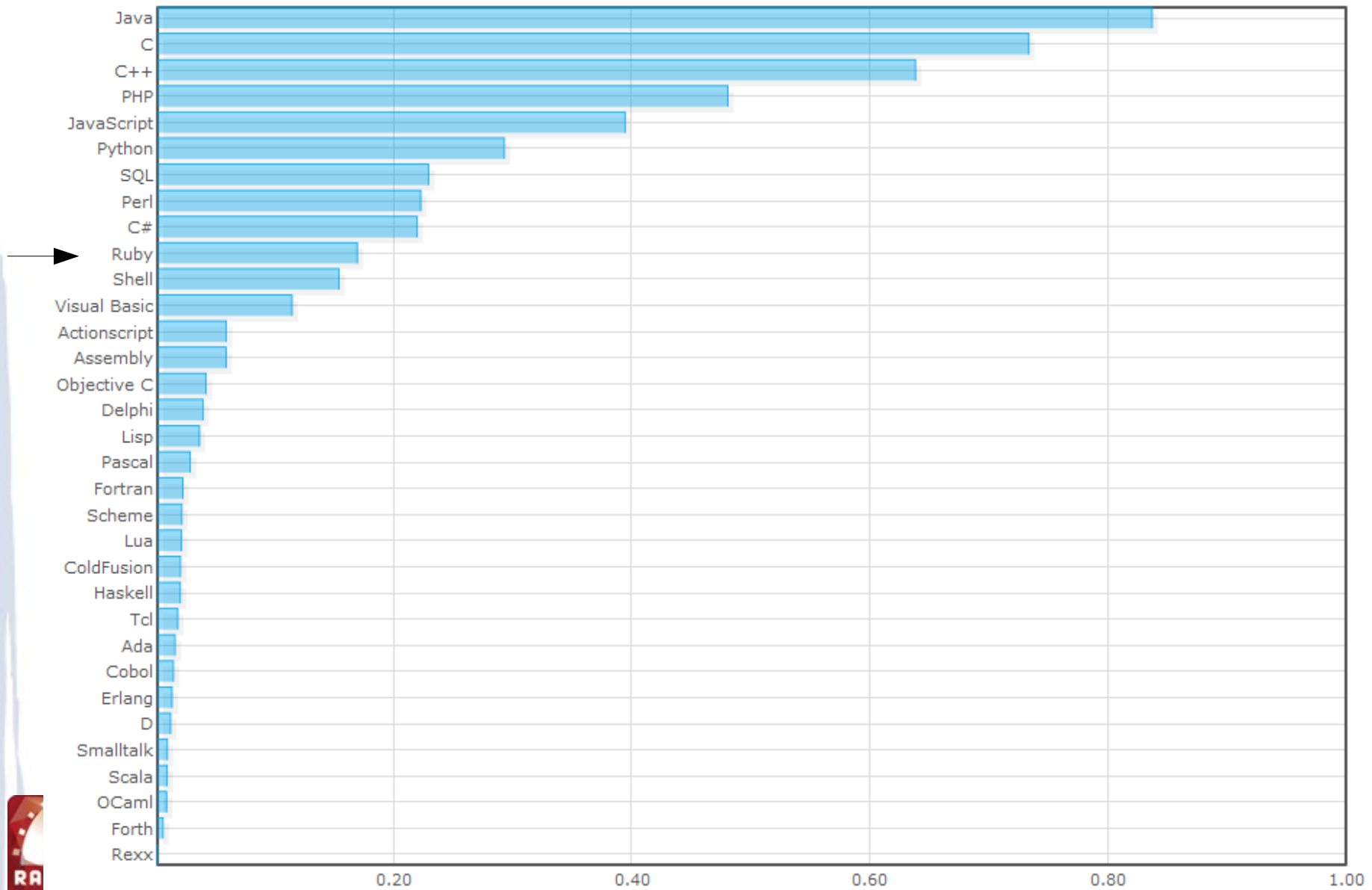




Ruby relative to other languages



2010 (from langpop.com)

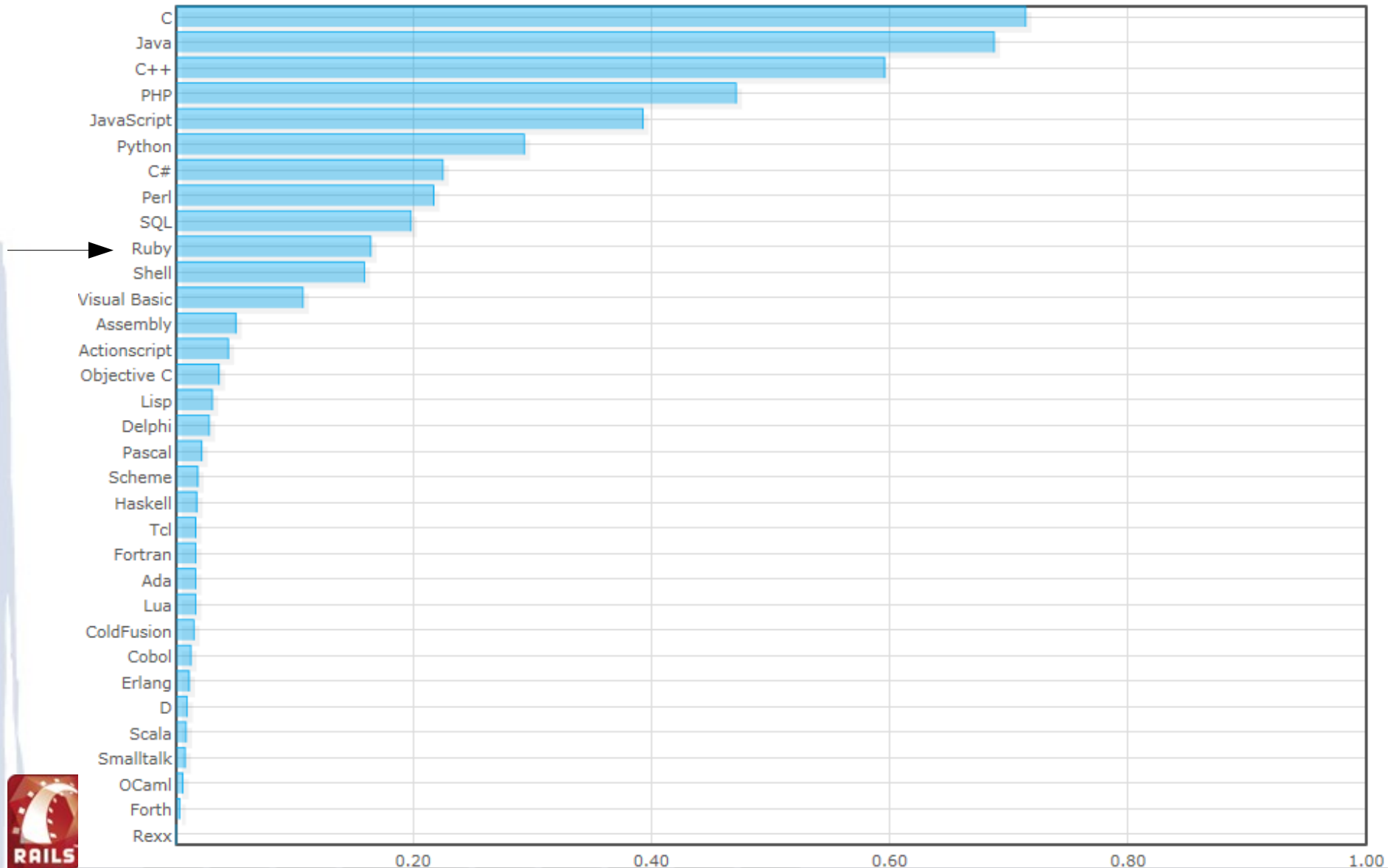




Ruby relative to other languages



2011 (from langpop.com)

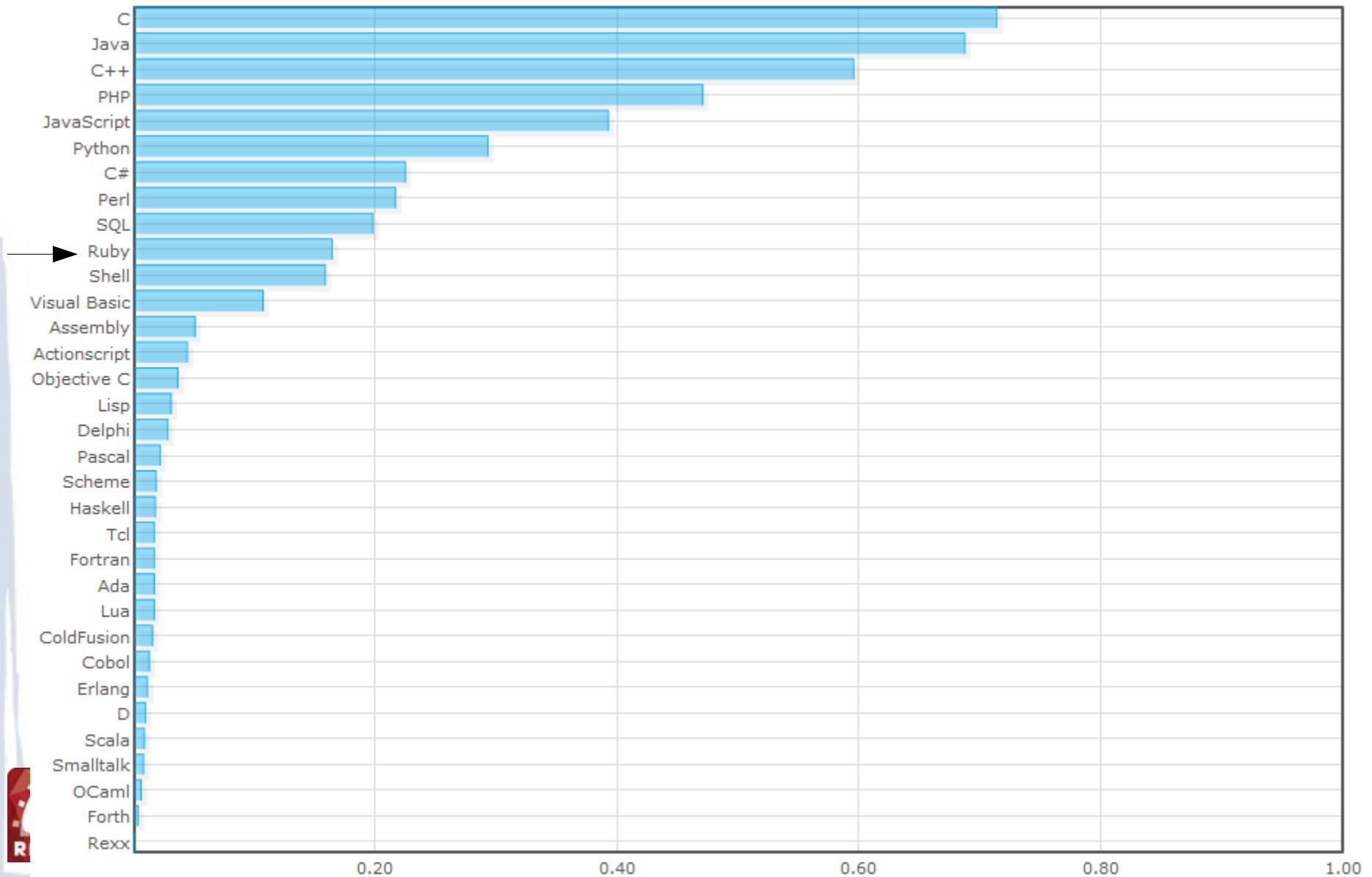




Ruby relative to other languages



2012 (from langpop.com)

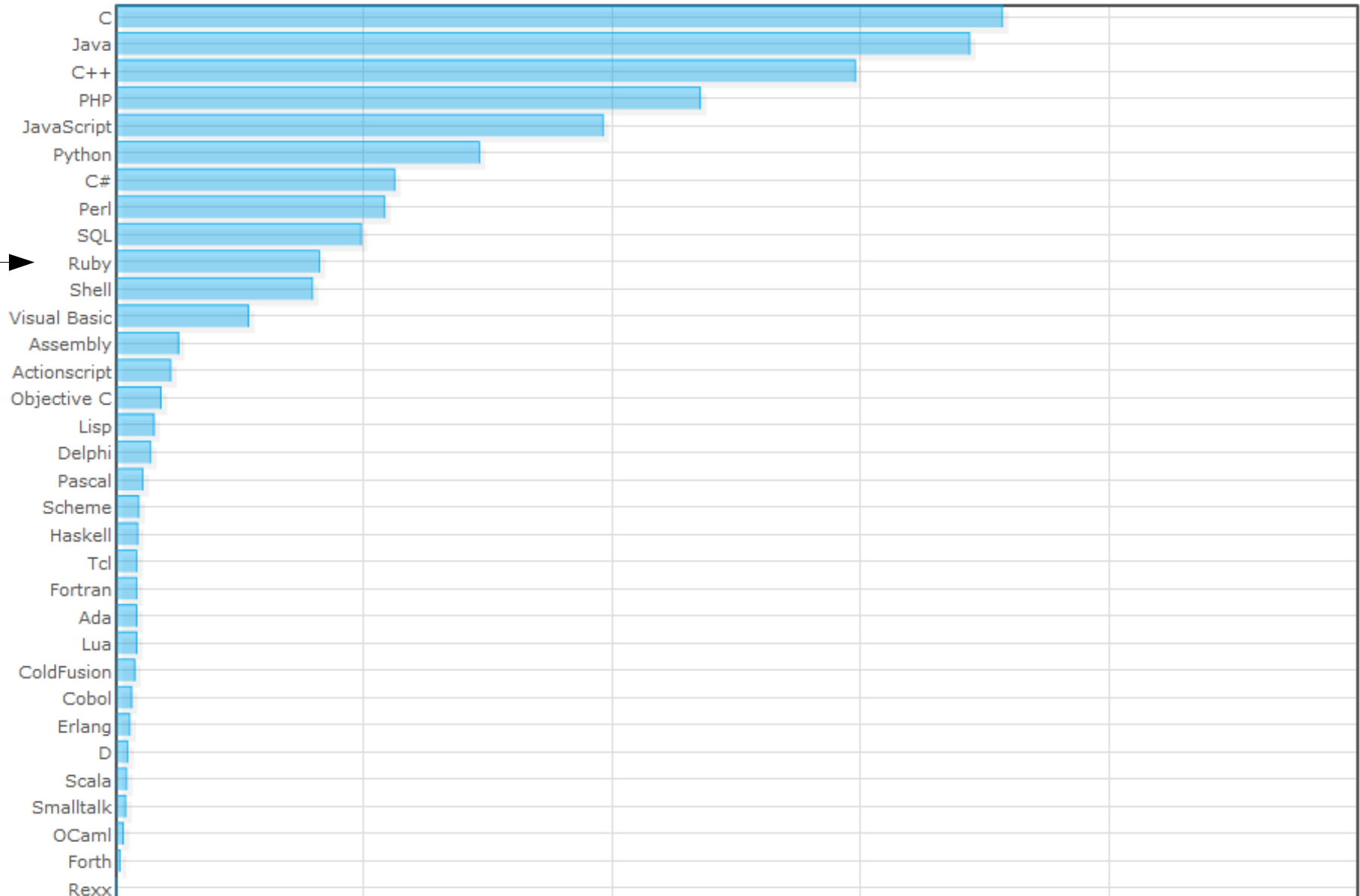




Ruby relative to other languages



2013 from langpop.com





Sites built with Rails



More sites: <http://rubyonrails.org/applications>



[Basecamp](#): The original Rails app.



[Twitter](#): Stay connected.



[Shopify](#): E-commerce made easy.



[Yellow Pages](#): Find it locally.



[Github](#): Git repo hosting.



[LivingSocial](#): Online local commerce.



[Sortfolio](#): Find a web designer.



[Hulu](#): Stream TV & Movies.



[Groupon](#): Daily deals.





Why use Ruby/Rails ?



- Reasons are similar to PHP
 - May have skills available (and more readily available)
 - Tons of code and complete applications
 - Many online books and tutorials
 - Can run natively on i (in a couple of different ways)
- Quick development of web sites





Possible Gotchas



- Learning curve
 - Ruby is WAY different from anything you have seen as an RPG programmer.
- Some installation issues on i (JRUBY)
 - J9 JVM and native extensions
- Native access to IBM i APIs
 - “Legacy” field naming issues
- PowerRuby (NEW!) takes care of many of these.





JRuby Installation on i (easy)



- Unzip the jruby-bin-1.6.7.2.zip (latest) into the IFS (I put mine in the /usr/local folder)
- Add the following hack to the jruby script file:

```
# Hacks to make this friendlier to IBM i  
JRUBY_BASE=/usr/local/jruby-1.6.7.2  
JRUBY_HOME=/usr/local/jruby-1.6.7.2
```

```
JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/  
jdk60/32bit
```



The JAVA_HOME should point to at least Java 1.6 (32 bit is fastest on i)



Check the install



Type QSH to start Qshell (PASE) which is where all of the action takes place with JRuby and Rails.

Then type the `jruby -v` command to check the installed version level

```
QSH Command Entry

$
> jruby -v
jruby 1.7.12 (1.9.3p392) 2014-04-15 643e292 on IBM J9 VM jvmap3260sr14-201307
05_01 [OS/400-PowerPC]
$

===> _____
```

We are good to go!





PowerRuby Install (NEW!)



- PowerRuby is in beta so you'll have to contact them directly for an install but once done... (just a series of RSTLICPGM from a SAVF)

```
/QOpenSys/usr/bin/-sh

$
> ruby -v
ruby 1.9.3p545 (2014-02-24 revision 45159) [powerpc-aix6.1.0.0]
$

===> _____
```



Looks a lot like the JRuby command prompt



Install Rails



Type 'gem install rails' (without the quotes)

```
> gem install rails
Successfully installed rails-3.2.3
1 gem installed
Installing ri documentation for rails-3.2.3...
Installing RDoc documentation for rails-3.2.3...
$
```

```
===>
```

```
F3=Exit    F6=Print  F9=Retrieve F12=Disconnect
F13=Clear  F17=Top   F18=Bottom F21=CL command entry
```

ri and RDoc are documentation for the gems





Build the application framework

Rails is an application framework. It not only can execute and run the web application but it can also generate the framework needed to build the application. The rails command does it all.

```
$  
> rails new comblog  
  [1m [32m      create [0m  
  [1m [32m      create [0m  README.rdoc  
  [1m [32m      create [0m  Rakefile  
  [1m [32m      create [0m  config.ru  
  [1m [32m      create [0m  .gitignore  
  [1m [32m      create [0m  Gemfile  
  [1m [32m      create [0m  app  
  [1m [32m      create [0m  app/assets/images/rails.png  
  
====>
```

What isn't shown above is a long list of other skeletal framework resources that rails will create for the application. Rails will also run the bundler gem to make sure all gems required by the application are installed.





Install Rails dependencies



Bundler is a gem tool that installs dependencies for Ruby applications. Gems are applications and libraries that have “helper” programs for Ruby applications.

The rails 'new' command will create the skeletal framework and run bundler which should give you all you need but occasionally you may add other features that require additional gems so running 'bundle install' will take care of the details for you.





Verify installation



```
Using thor (0.14.6)
Using railties (3.2.3)
Using coffee-rails (3.2.2)
Using jquery-rails (2.0.2)
Using jruby-openssl (0.7.6.1)
Using bundler (1.1.3)
Using rails (3.2.3)
Using sass (3.1.16)
Using sass-rails (3.2.5)
Using therubyrhino (1.73.3)
Using uglifier (1.2.4)
  [32mYour bundle is complete! Use `bundle show [gemname]` to see where a bund
led gem is installed. [0m
$

==>
_
_
_
_

F3=Exit   F6=Print F9=Retrieve F12=Disconnect
F13=Clear F17=Top  F18=Bottom F21=CL command entry
```

When bundler completes, you should see that last line indicating that rails is ready to go.





Start the web application



Change your directory to the application folder (created by rails new) and then start the rails application. Rails has a minimalist web server called WEBrick that is great for testing apps but generally shouldn't be used in production. The start command is 'rails s' but I added the -p switch to start the server on a different port (3080 in this case)

```
$
> cd comblog
$
> rails s -p 3080
=> Booting WEBrick
=> Rails 3.2.3 application starting in development on http://0.0.0.0:3080
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2012-05-03 17:56:50] INFO WEBrick 1.3.1
[2012-05-03 17:56:50] INFO ruby 1.8.7 (2012-05-01) [java]
[2012-05-03 17:56:50] INFO WEBrick::HTTPServer#start: pid=493231462 port=3080
0
```

====> _____

F3=Exit F6=Print F9=Retrieve F12=Disconnect
F13=Clear F17=Top F18=Bottom F21=CL command entry



Security



You may receive a security warning when you start your rails app. For testing this is OK because you will probably only run the rails app locally.

You need to fix this before you go into production. Just follow the instructions on the warning (Google will help)





Display the welcome page



It's difficult to see in this screen shot but the app is running on my IBM i (10.0.10.205 on my internal network) at port 3080.

10.0.10.205:3080

Google

Welcome aboard

You're riding Ruby on Rails!

[About your application's environment](#)

Getting started


Here's how to get rolling:

1. Use `rails generate` to create your models and controllers
To see all available options, run it without parameters.
2. Set up a default route and remove `public/index.html`
Routes are set up in `config/routes.rb`.
3. Create your database
Run `rake db:create` to create your database. If you're not using SQLite (the default), edit `config/database.yml` with your username and password.

Browse the documentation

- [Rails Guides](#)
- [Rails API](#)
- [Ruby core](#)
- [Ruby standard library](#)

Find: EACCES Next Previous Highlight all Match case





Check on the environment



Welcome aboard

You're riding Ruby on Rails!

About your application's environment

Ruby version	1.8.7 (java)
RubyGems version	1.8.24
Rack version	1.4
Rails version	3.2.3
JavaScript Runtime	therubyrhino (Rhino)
Active Record version	3.2.3
Action Pack version	3.2.3
Active Resource version	3.2.3
Action Mailer version	3.2.3
Active Support version	3.2.3
ActionDispatch::Static	
Rack::Lock	





Adding to the framework



- Probably will want a new welcome page so lets start with something simple: A new index page:
rails generate controller home index
- After the resources have been generated, delete the index.html.erb file in the public folder
- Hack the index.html.erb file in the app/views/home folder to have the following:
- `<h1>Hello, Rails on IBM I! </h1>`





Adding to the framework



- Add the following to the routes.rb (in the config folder) file by uncommenting the line:

```
#...
```

```
# You can have the root of your site routed  
with "root"
```

```
# just remember to delete public/index.html.
```

```
root :to => "home#index"
```

- This sets your home page to the index.html.erb file in the home folder.





Adding to the framework

(building the blog application)



- Generate a resource using the rails 'generate' command (rails g for short). The generate command creates all of the framework for the resource.

```
rails g scaffold post title:string body:text
```

```
$  
> rails generate scaffold post title:string body:text  
   [1m [37m      invoke [0m  active_record  
   [1m [32m      create [0m    db/migrate/20120503225955_create_posts.rb
```

```
===> _
```

```
F3=Exit   F6=Print F9=Retrieve F12=Disconnect  
F13=Clear F17=Top  F18=Bottom F21=CL command entry
```





Building on the framework



- The 'generate scaffold' command gave us the following resources in our project:
 - Migration
 - Model
 - Controller
 - Helper
- We'll be working with these resources to create our “blog” application.





Migrations



- Rails works with many databases – sqlite is used by default. Since we are using JRuby, we will need to make sure that the activerecord-jdbc-adapter gem is installed. This allows us to use any database resources that has a JDBC driver available. This is the best solution for using DB2 and MySQL on IBM i.
- You will need to make a change to the database.yml file.





Migrations (cont)



- Migrations allow us to track database changes from the very beginning of application development. We can then roll back or roll forward to/from any database schema modifications we have made

Very cool!





Editing files



- Ruby files are just text files you can edit with any text editor.
- I use an IDE called RubyMine. There are many others.





database.yml hack



```
# SQLite version 3.x
#   gem 'activerecord-jdbcsqlite3-adapter'
#
# Configure Using Gemfile
# gem 'activerecord-jdbcsqlite3-adapter'
#
}development:
  username: jruby
  password: demo
  adapter: jdbc
  driver: org.gjt.mm.mysql.Driver
  url: jdbc:mysql://localhost:3306/rails_demo?autoReconnect=true

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
}# Do not set this db to the same as development or production.
}test:
  adapter: sqlite3
  database: db/test.sqlite3
}
}production:
  adapter: sqlite3
  database: db/production.sqlite3
```





Run the migration



```
$
rake db:migrate
==
-- create_table(:posts)
   -> 0.1210s
   -> 0 rows
== CreatePosts: migrated (0.1220s) =====
==
```

From here on out whenever I make a change to the schema; Adding a table, changing the columns of any table I run migrate and it will back up and change the schema and track those schema changes





A couple of additional changes



- The original application 'index' in the public folder and the application index in the layout folder (views) need to be deleted





Restart the Application (new scaffold in place)



- The scaffold basically builds a CRUD application based on the resource template. This can be as complex or as simple as you want but for a quick and dirty app, you can get pretty far in a short amount of time





Write some code



- Start your rails app and leave it running. We'll edit the files directly and rails will update them in real time (its a scripting language, remember!).





Standard Rails Components



- Convention over configuration! These are the “standard” Rails web methods:
 - index – list of posts
 - new – show the new posts
 - create – create a new post entry
 - show – show an existing post
 - edit – show an existing post for editing
 - update – change an existing post
 - destroy – delete and existing post





The “magic” Scaffold!



- You don't **have** to use the full scaffold on a Rails application. You can generate just sections like controllers or models if you want.
- A quick demo.....





Controller



- Much of the heavy lifting occurs here:
 - index
 - show
 - new
 - edit
 - create
 - update
 - destroy

Let's check out some code....





Views



- Most of this is HTML mixed with Ruby (as little as possible). Very similar to PHP and JSP's.
- Partials are a great way to leverage DRY development by assigning sections of repetitive views to a file.





Adding links to pages



- Lets add a link to our posts in the blog by adding this line to our home page:
 - `<%= link_to "My IBM i Blog", posts_path %>`





Thanks! Questions?



Pete Helgren
Value Added Software, Inc.
pete@valadd.com

<http://www.roroni.com/>

