VISHNU
GOPAL

# EASY COMPILERS

## WITH RUBY AND TREETOP

- Vishnu Gopal
  - Started as a Software Engineer, SlideShare Inc.
  - Human-Computer Interaction from UCLIC, London
  - *CTO, MobME Wireless*.
  - Working with Ruby for over 5 years.

# ABOUT ME

vishnugopal.com

@vishnugopal

- College startup in 2006
- NASSCOM Global Leadership Award, 2011
- VAS & core network solutions for mobile operators
- Over 40 engineers working in Ruby
  *& 2 speakers at RubyConf India*

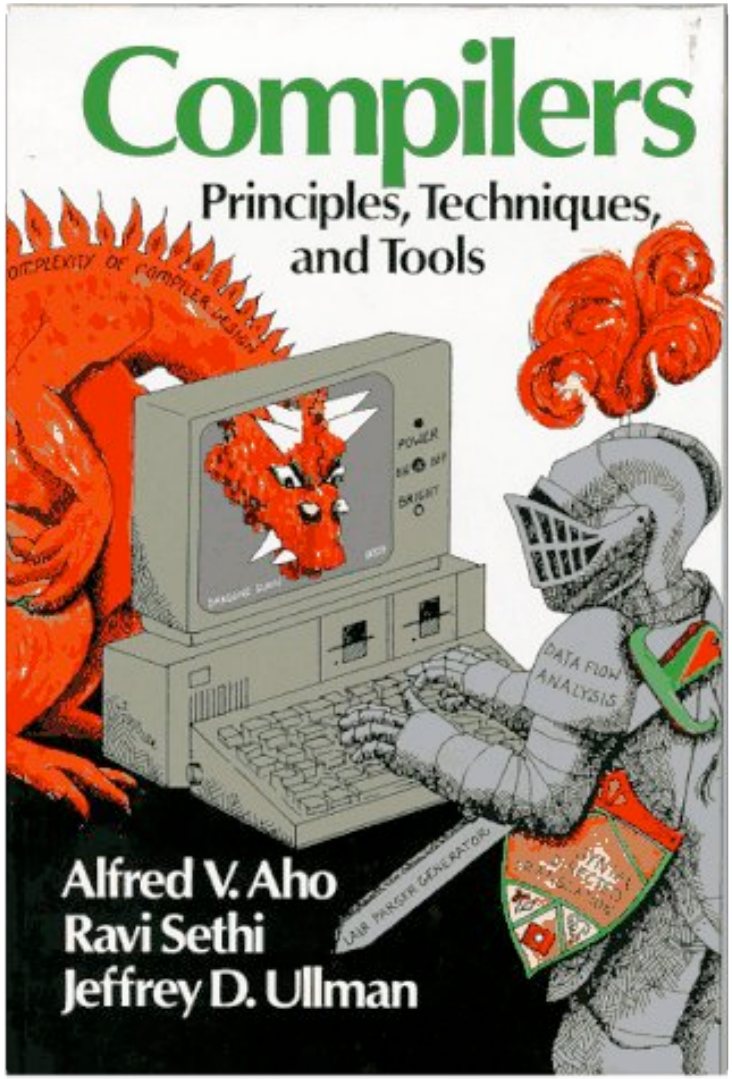
# MOBME WIRELESS

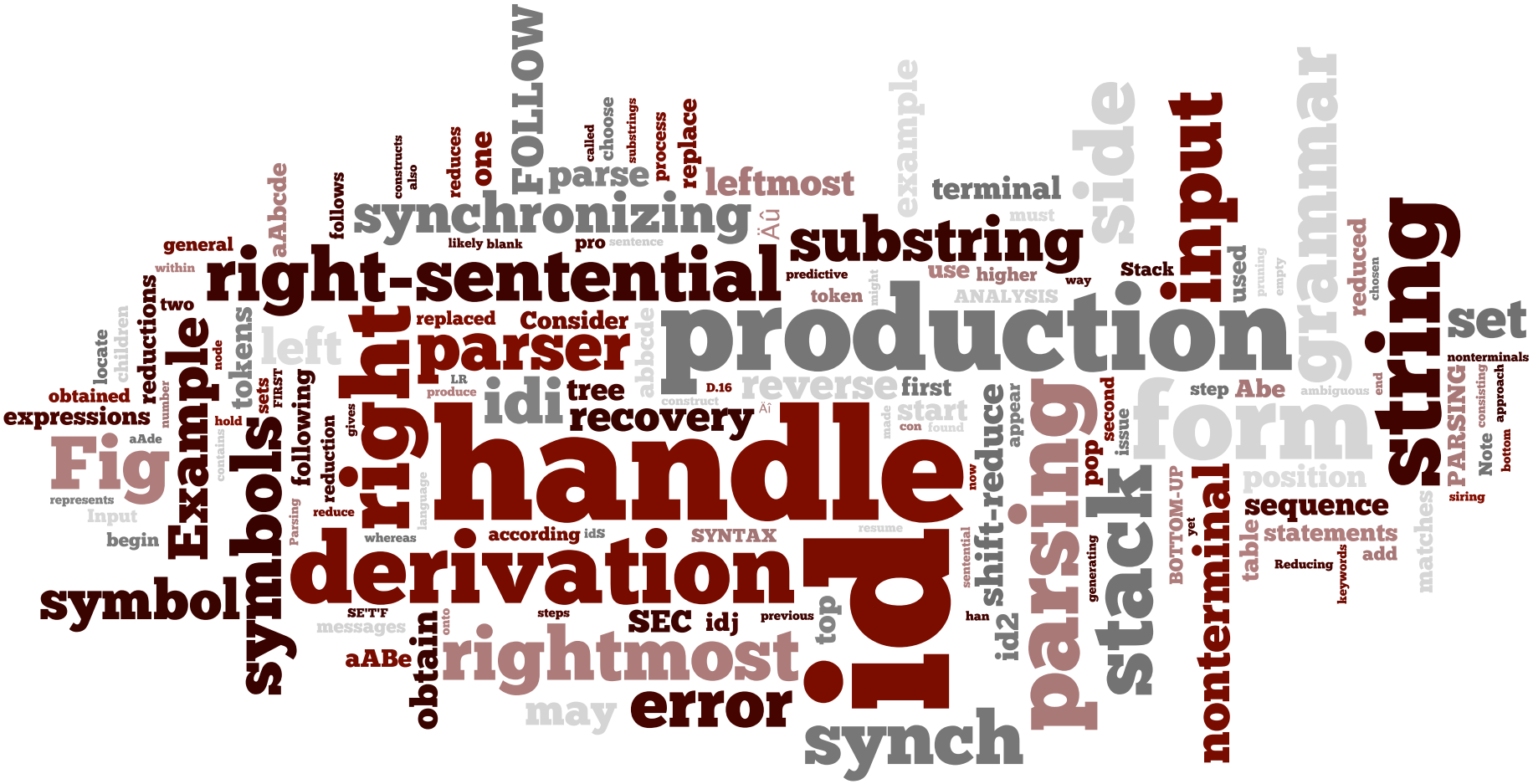
mobme.in

@mobmewireless

# DRAGON BOOK

- Aho, Sethi & Ullman
- 1009 pages of wisdom



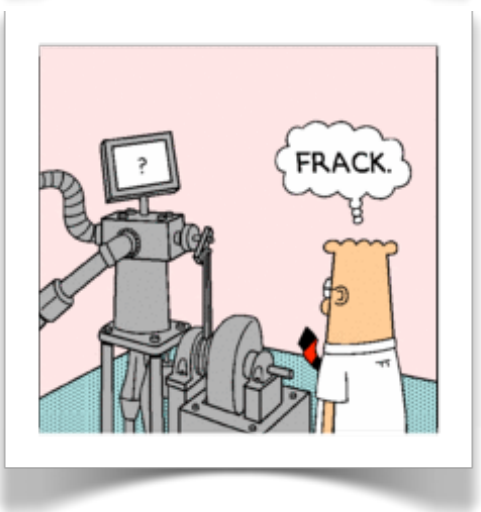



- Too complicated.
- Meant for Computer *Scientists*

- Designed for a programmer
- Not too hard to grok
- Iterative development
- Friendly
- Fun

# IN RUBY






# COMPILERS?

HOW ARE PARSERS AND COMPILERS USEFUL?

A compiler converts a high-level language into a form that computers can understand.

A parser structures a high-level language into an intermediate form that compilers can then use.

# WHAT?

```
/[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}/
```

# PARSING

```
/[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}/

/<regex>/ # I'm a regex!

[ ]       # this means I'm a set, I match anything in it!

[ ]+      # Ah, one or more!

@         # The lowly character '@'
```

# PARSING 2

- Convert this regular expression into a FSM.
- Run a block of text against this FSM and return true or false.

```
"vishnu@mobme.in"   # I match!

"32"                # And I don't!
```

# COMPILING

- When you write a regular expression, you do not worry about all of this. You do not care about how it's parsed, compiled or matched.
- A regular expression matches strings, but what if you want to do other things with your own mini language?
- A compiler construction library will help you do that.

# ABSTRACTION

- A search query language like Google's:
  Cricket -Tendulkar
- A SQL-like frontend for your favourite NoSQL backend:
  SELECT * FROM LIST redis.rubyconf
- A small DSL to match twitter entries.
  tweet.text contains "India" AND tweet.text has hyperlink

# POSSIBILITIES

- Yacc & LEX
- Bison & FLEX
- Boost Proto

# NOT MY IDEA


treetop
Bringing the simplicity of Ruby
to syntactic analysis.

# A RUBY WAY

```
tweet.text contains "India" AND tweet.text has hyperlink

NOT tweet.text contains "India"

tweet.text contains "rubyconf" OR tweet.text contains "#rubyconf"
  AND tweet.author is "vishnugopal"
```

# TREETWEET

- BNF Form: a computer-sciency way to define language keywords, syntax and order of precedence of operators.
- Or, a formal definition that you can feed to a *parser*

# BACKUS NAUR

```
tweet.text contains "#rubyconfindia" OR
tweet.text contains "rubyconf" AND tweet.text contains "india"
  AND NOT tweet.text contains "#rubyconf"
  AND tweet.author is "vishnugopal"
```

# TT BNF

```
tweet.text contains "#rubyconfindia" OR
(tweet.text contains "rubyconf" AND tweet.text contains "india")
  AND (NOT tweet.text contains "#rubyconf")
  AND tweet.author is "vishnugopal"
```

# TT BNF 2

```
tweet.text contains "#rubyconfindia" OR
tweet.text contains "rubyconf" AND tweet.text contains "india"
  AND (NOT tweet.text contains "#rubyconf"
  AND tweet.author is "vishnugopal")
```

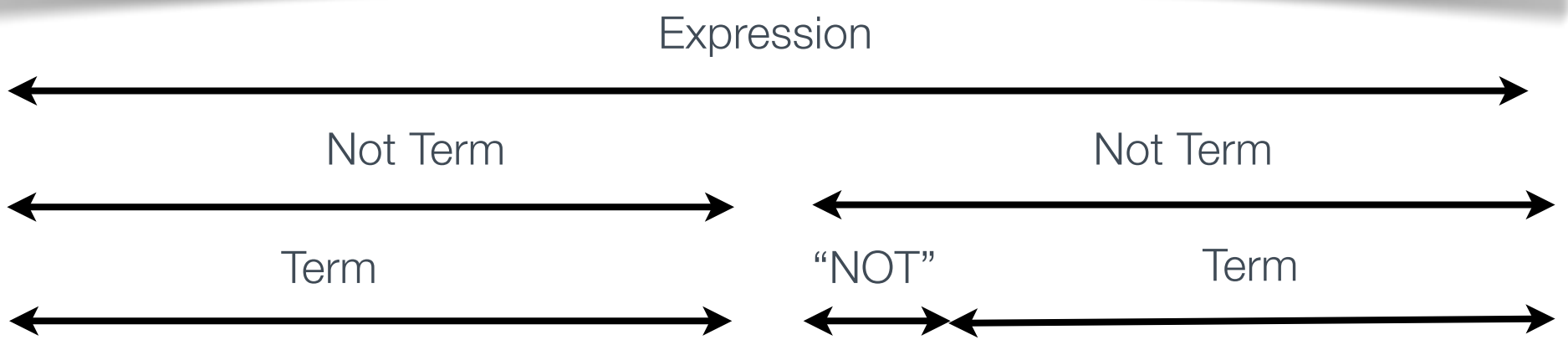
Incorrect!

# TT BNF 3

```
expression = not-term ("AND" not-term | "OR" not-term)*
not-term = "NOT" term | term
term = group | predicate
group = "(" expression ")"
predicate = target binary_operator argument
target = [a-zA-Z] [0-9a-zA-Z.-_]*
binary_operator =  "contains" | "is"
argument = '"' ('\"' | -'"')* '"'
```

# TT BNF 4

```
tweet.text contains "India" AND NOT tweet.text has hyperlink
```




target = tweet.text

binary_operator = contains

argument = “India”

# TT BNF EXAMPLE

```
expression = not-term ("AND" not-term | "OR" not-term)*

rule expression
  not_term ("AND" not_term / "OR" not_term)*
end
```

```
target = [a-zA-Z] [0-9a-zA-Z.-_]*

rule target
  [a-zA-Z] [0-9a-zA-Z\.\-_]*
end
```

```
argument = '"' ('\"' | -'"')* '"'

rule argument
  '"' ('\"' | !'"' .)* '"'
end
```

# BNF TO TREETOP

```
gem install treetop
tt treetweet.treetop -o ./treetweet_parser.rb

require 'treetop'
require 'treetweet_parser'
TreeTweetParser.parse(tweet)
```

# GET IT WORKING!

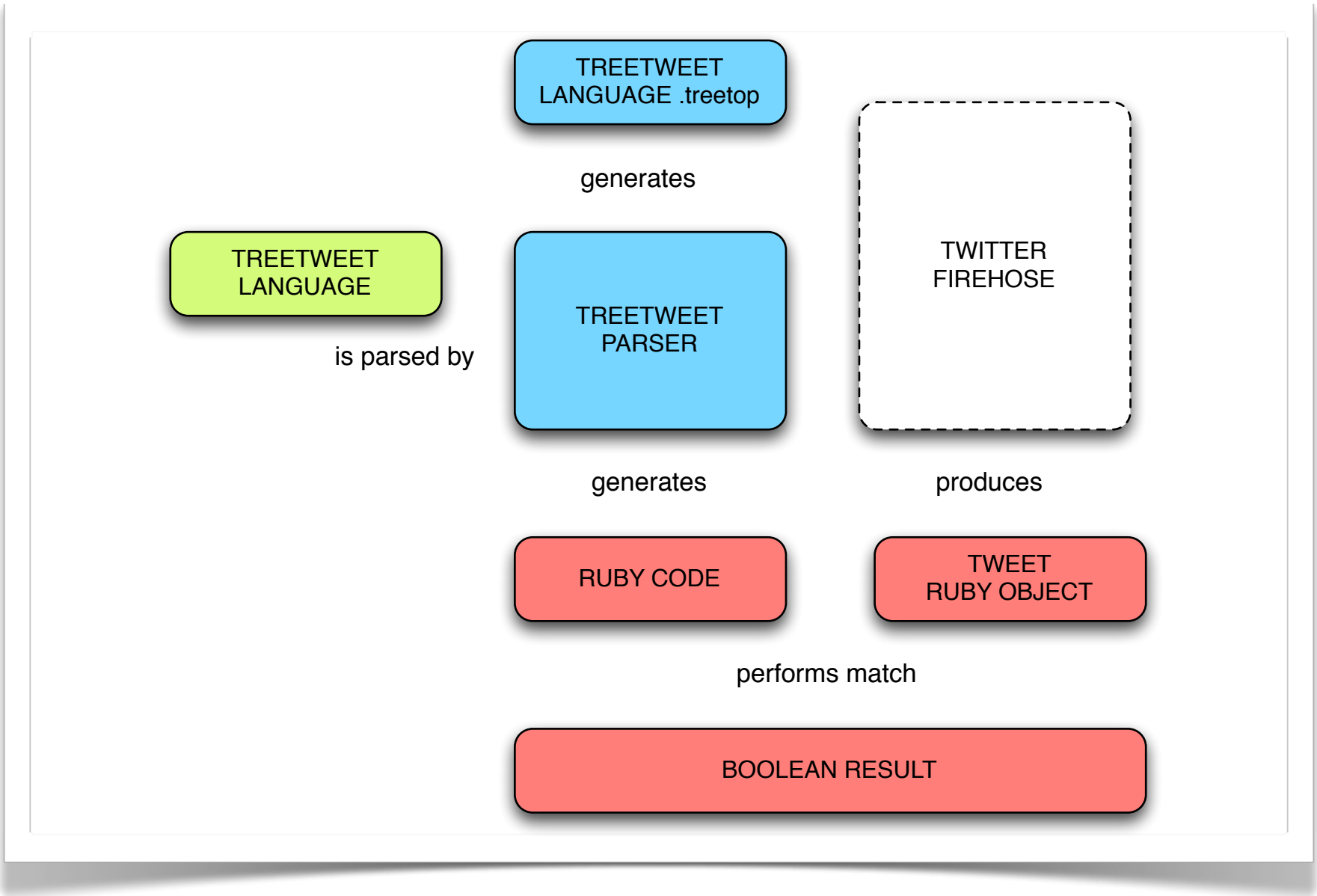
A compiler converts a high-level language into a form that computers can understand.

A parser structures a high-level language into an intermediate form that compilers can then use.

# AGAIN?

- *Syntax Oriented Programming*
- Matching rules can have Ruby methods inline or mixed in as a module.
- These methods are available to the parser.
- Implement a “compile” method for e.g. to compile parsed objects into an executable form.
- Let us compile TreeTweet into our favourite language!

# SYNTAX FIRST


TREETWEET
LANGUAGE .treetop
generates
TREETWEET
LANGUAGE
is parsed by
TREETWEET
PARSER
TWITTER
FIREHOSE
generates
produces
RUBY CODE
TWEET
RUBY OBJECT
performs match
BOOLEAN RESULT

# OVERVIEW

```
argument = '"' ('\"' | -'"')* '"'

rule argument
  ('"' ('\"' | !'"' .)* '"')
end

rule argument
  ('"' letters:('\"' | !'"' .)* '"' <ArgumentValue>)
end
```

# COMPILE 1

```ruby
module ArgumentValue
  # We are converting this to a
  # double-quoted Ruby string
  def compile
    %Q("#{letters.text_value}")
  end
end

argument.compile # this now works!
```

# COMPILE 2

```
module ContainsOperator
  # tweet_object is the tweet to be matched against
  def compile(target, argument)
    %Q(
      tweet_object[#{target.compile}].index(#{argument.compile})
    )
  end
end
```

# COMPILE 3

```
# JSON Tweet
{
  text: "Railsconf India is awesome, awesome!",
  author: "vishnugopal"
}

# To Ruby Object
tweet_object = {
  :text => "Railsconf India is awesome, awesome!",
  :author => "vishnugopal"
}
```

# COMPILE 4

```
# TreeTweet
tweet.text contains "Railsconf" AND tweet.text contains "India"

# To Ruby
( tweet_object[:text].index("Railsconf") )
&&
( tweet_object[:text].index("India") )

# To Boolean
true
```

# COMPILE 5

- What we've done is implement a cross-compiler.
- The cross-compiler compiles the TreeTweet DSL down to Ruby.
- It could however, be easily modified to compile it down to say Java, or even C if you need performance.
- It's small and concise. The treetop grammar is less than 50sloc. The compiler less than 400sloc.
- It's really easy to modify.

# RECAP

- github.com/vishnugopal
- Not released yet, soon!

# SOURCE CODE


codejam.mobme.in

# QUESTIONS?