Your Existence

Proofreeded buy

Jamon Holmgren


Who am I?

@GantLaborde

Public Speaker

Writer

Coder


KuvatON.com

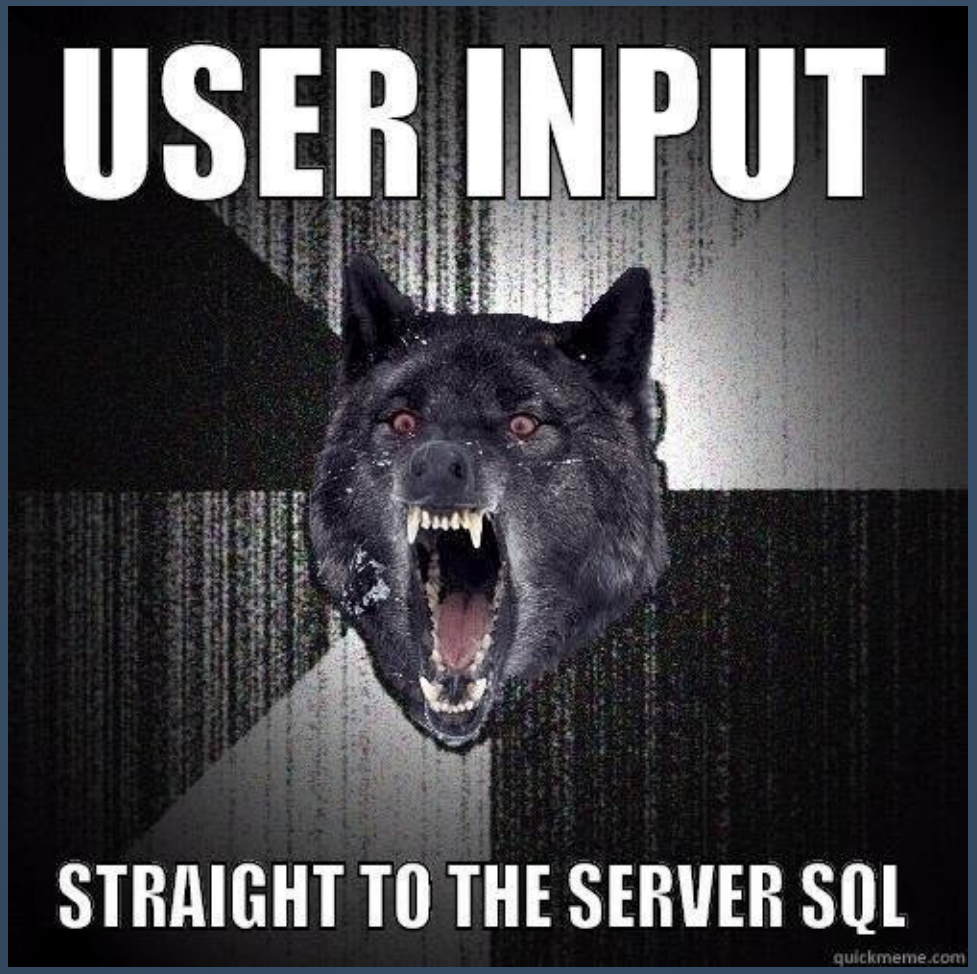
Tit e Text

infinitered


HI GUYZ
YOU WANT TO TALK ABOUT VALIDATION?
imgflip.com

Trusting User Input

Crazy or Lazy?



Start with ideal API

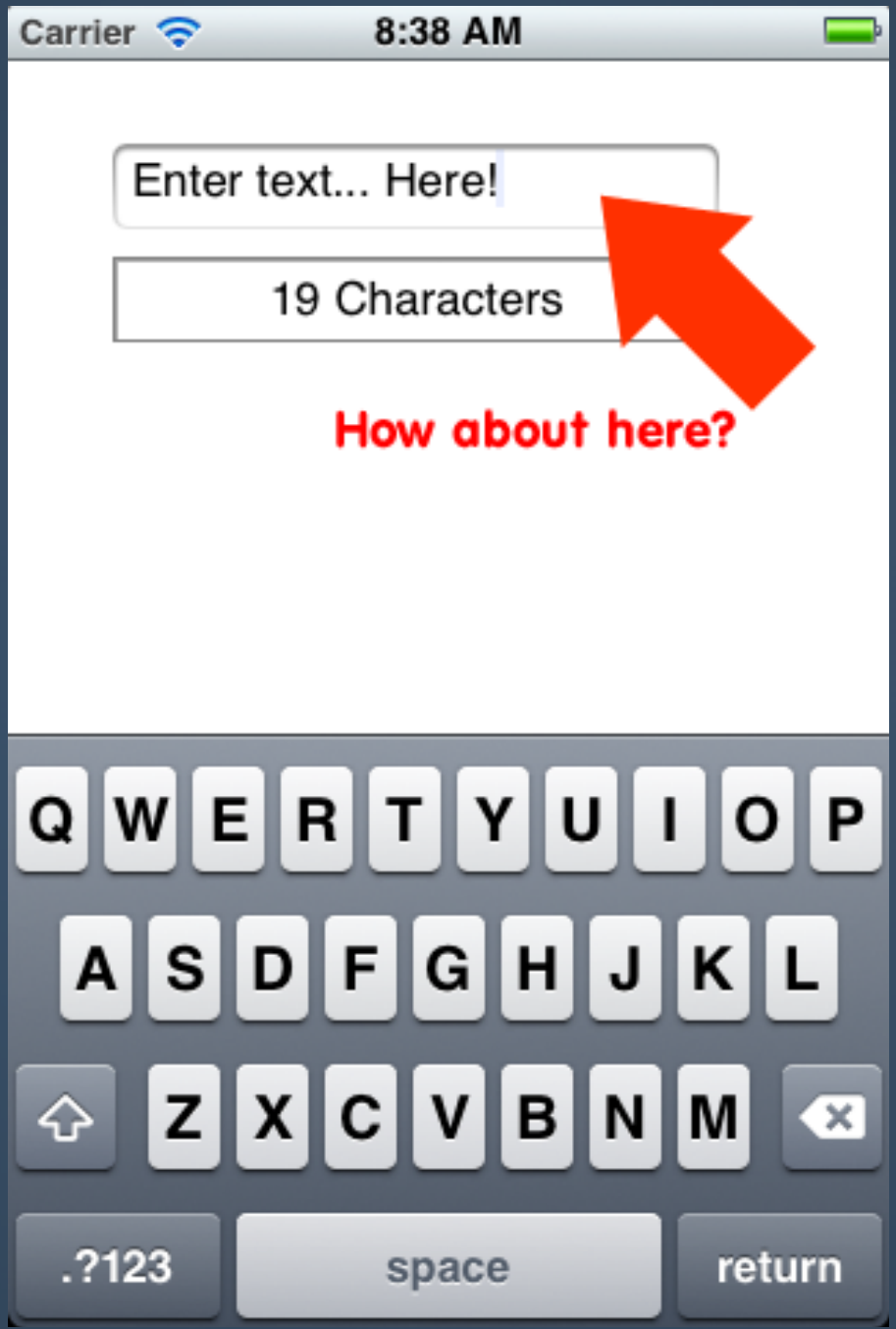
Have smart people help!

Ideal Validation?

```
# We can validate information
valid?(98.6, :number)
#=> true
```

Get user input **THEN** validate it in some place.

Stop them at the Source


Carrier
8:38 AM
Enter text... Here!
19 Characters
How about here?
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123
space
return

Ideal Validation?

take 2

```
# We can validate user input at the source
@username_textfield.validates(:number).on(:invalid) do
  # Alert the user!
end
```


WHY NOT
BOTH?
memegenerator.net


RMQ
RubyMotionQuery

Validation as a Utility

1/5

```
# Handle common validation needs
rmq.validation.valid?('test@test.com', :email) # returns true
rmq.validation.valid?('5045558989', :usphone) # returns true
rmq.validation.valid?('90210', :uszip) # returns false
```

Validation as a Utility

- :email
- :url
- :dateiso
- :number
- :digits
- :ipv4
- :time
- :uszip
- :ukzip
- :usphone
- :strong_password
- :has_upper
- :has_lower
- :presence
- :length

Validation as a Utility

3/5

```
# Make it Flexable
rmq.validation.valid?('test', :length, exact_length: 4) #true
rmq.validation.valid?('test', :length, min_length: 4) #true
rmq.validation.valid?('test', :length, max_length: 4) #true
rmq.validation.valid?('test', :length, min_length: 2, max_length: 7) #true
rmq.validation.valid?('test', :length, exact_length: 2..7) #true
rmq.validation.valid?(' test ', :length, max_length: 5, strip: true) #true
```

Validation as a Utility

4/5

```
# Roll your own!
rmq.validation.valid?('nacho', :custom, regex: Regexp.new('nachos?')) #true
```

Validation as a Utility

5/5

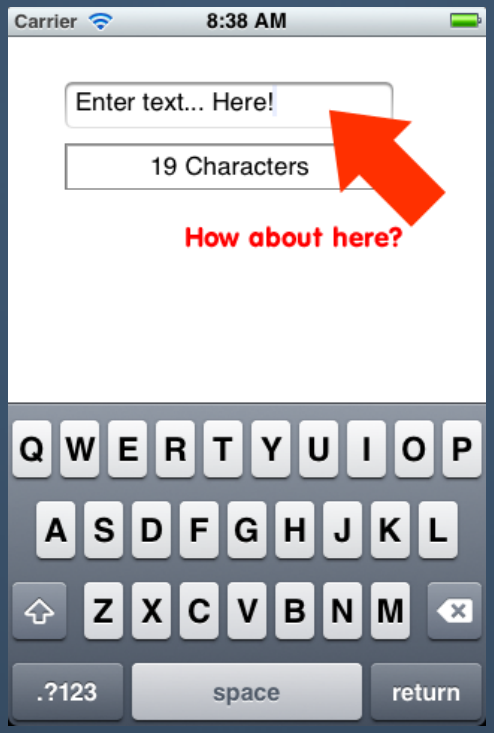
Things that validation has to think about?

- `allow_blank`
- `white_list`

Friday

Validation as a Rule

Built on the Utility



Validation as a Rule

```
# Add a rule to a textfield
rmq(@user_email).validates(:email)

# Add rule ass we append fields
rmq.append(UITextField, :weight).validates(:digits)
rmq.append(UITextField, :name).validates(:presence)

# Check that input
rmq(@user_email).valid?

# Check every applied rule
rmq.all.valid?
```

Who's Bad?

```
# Useless at first
rmq(some_selection_criteria).invalid
rmq(some_selection_criteria).valid

# moment of judgement
rmq.all.valid?

# All invalid views as an array
rmq.all.invalid
# All valid views as an array
rmq.all.valid
```

Who's Bad?

```
# I react depending on my own state
rmq.append(UITextField).validates(:digits).on(:valid) do
 #valid
 puts "WOWZERZ field is valid <3 <3 <3"

end.on(:invalid) do |invalid|
 #invalid
 puts "NOOOOOOOOOOOO!!!1!1"

end
```

Easy Come - Easy Go

```
# Bye bye Guy
rmq(some_selection).clear_validations!

# Later alligator
rmq.all.clear_validations!
```


Broken but beautiful
Informative Errors

We can be friendly

validation_errors

```
# Returns array of error messages for every invalid item in selection
rmq.all.validation_errors
# E.G ["Validation Error - input requires valid email."]
```

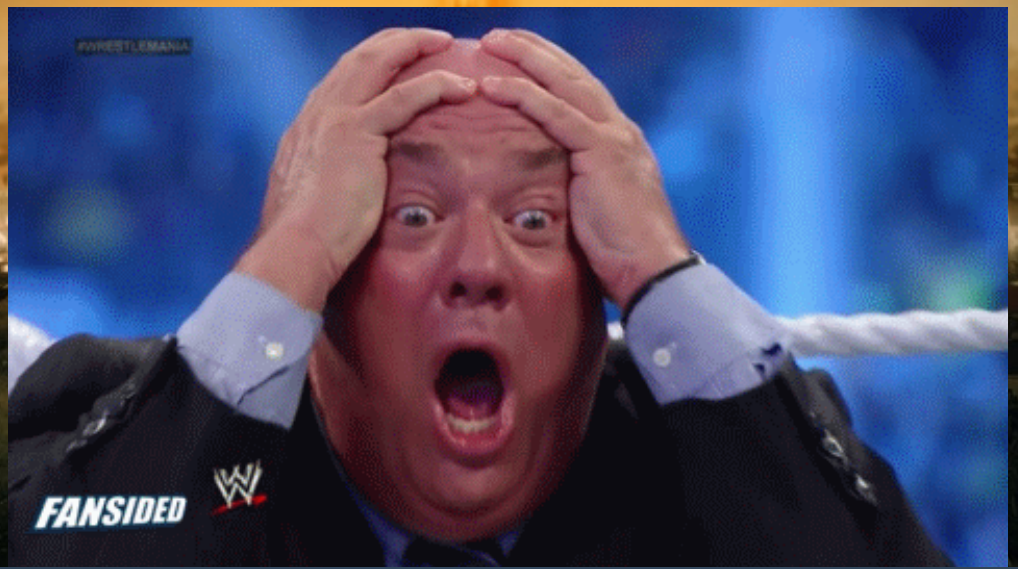
We can be friendly

custom error messages

```
  # do this in your stylesheet
  def user(st)
    #custom validation images
    st.validation_errors = {
      email: "Please check your user email and try again"
    }
  end

# overrides "Validation Error - input requires valid email."
```

Custom Validations with Selection Rules!?!



Custom Validation Rules

the easy way - simple regex

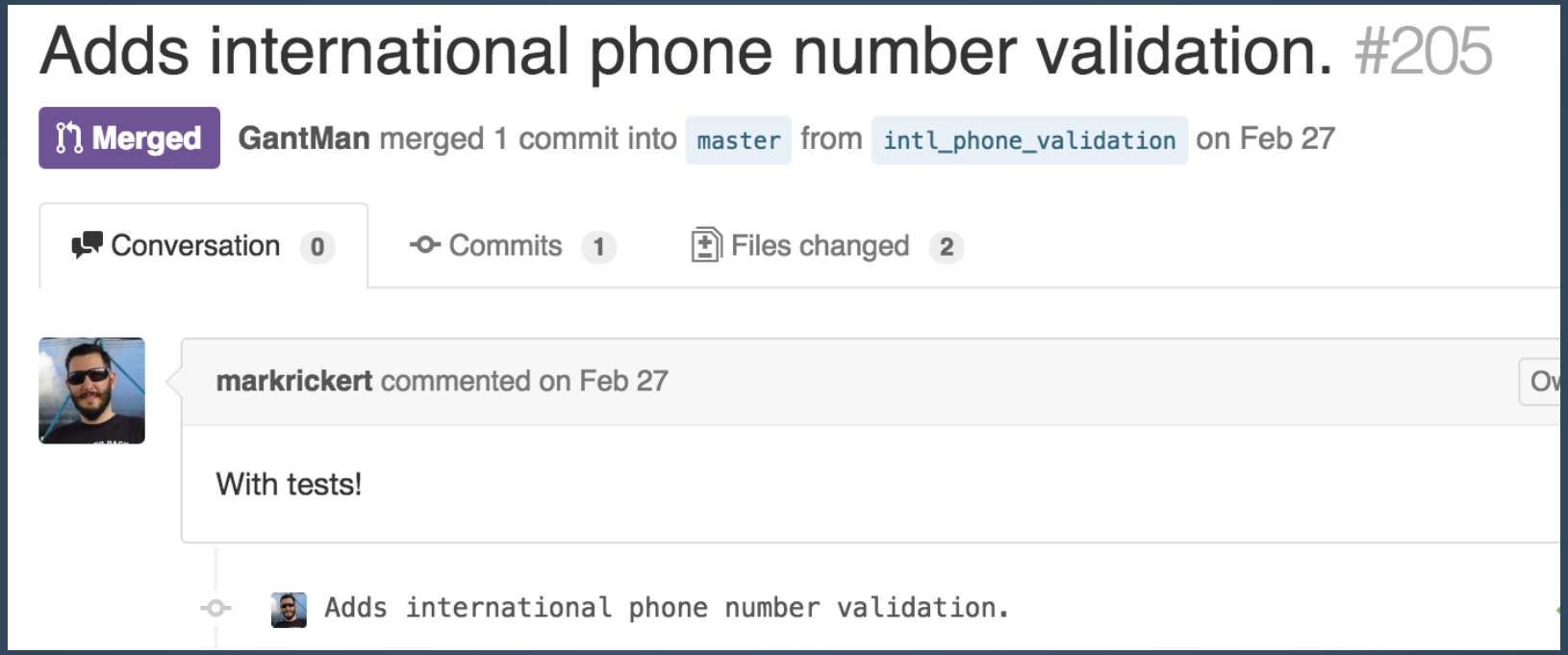
```
append(UITextField).validates(:custom, regex: /^test$/)
```

Custom Validation Rules

```
rmq.validation.add_validator(:first_letter) do |value, opts|
    value.start_with?(opts[:letter])
end
```

Now use it!

```
some_field = append(UITextField).validates(:first_letter, letter: 'x')
some_field.data("test")
some_field.valid? # => false
some_field.data("xenophobia")
some_field.valid? # => true
```



Mark Rickert (aka the "Bad Boy of RubyMotion")

Hangin' with BroBama

The Future of Validation


Thanks!