Validation with FormEncode

By Karl Shouler

A little guide..

What in the world are validators?

Why should I be using them?

from formencode import validators

What in the world are validators?

Validators check for syntactical correctness.

(shameless lift from wikipedia)

Where might I have some data that would benefit from a little validation?

FORMS!

email address
phone number
name
password
etc.

Why should I use validators?
(a quick study)

HP LaserJet P2035 Printer

Serial Number: * 1234567890

» How do I find my product serial number?

Purchase date: * December 01 2011

Purchase location: Select One

Personal Information

First Name: * a3gh

Last Name: * heymom

E-mail Address: * wutchadoin@123@456.fancy

Address:

City: d6s78ajjjjjj;

State: Kansas

Zip Code: aaaaa –

Phone Number: (345) fhj – j111

Future dates cannot be entered.

Invalid format for email address.

Invalid zipcode format has been entered.

...so I tended to their complaints.

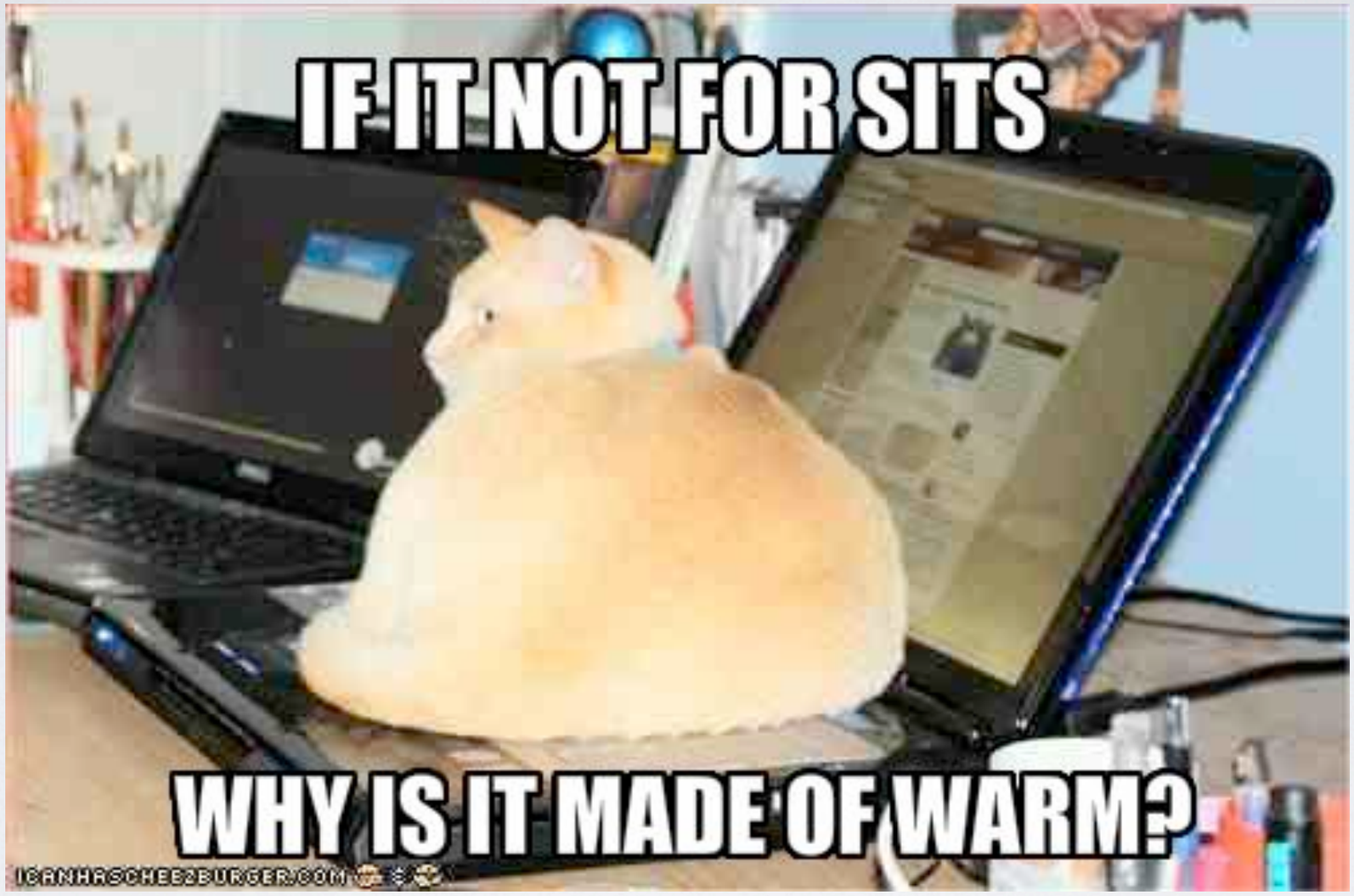
```
# date in the past
# properly formatted email with a bogus domain
# zipcode 11111 (which doesn't exist)
```

Registration success!

a3gh, thank you for registering your product! We will e-mail you a co
We recommend that you save the e-mail, and use it to create or access y

Product name:	HP LaserJet P2035 Printer
Model name:	CE461A
Serial number:	1234567890

Don’t trust user input!



formencode.org

A lot of validators
Internal vs. External data
to_python()

```
>>> import formencode
>>> from formencode import validators
```

Basic Validators

```
### Bool
>>> from formencode import validators as v
>>> v.Bool.to_python(0)
False
>>> v.Bool.to_python(True)
True
>>> v.Bool.to_python(None)
False
>>> v.Bool.to_python('')
False
```

```
### Strings
>>> v.String().to_python('I\'d be crazy not to follow')
"I'd be crazy not to follow"
>>> v.String(max=10).to_python('Follow where you lead')
Traceback (most recent call last):
  ...
formencode.api.Invalid: Enter a value less than 10 characters long
>>> v.String(max=10).to_python('Your eyes')
'Your eyes'
```

Basic Validators

```
### Dates
>>> d = v.DateValidator(earliest_date=datetime(2009, 4, 22))
>>> d.to_python(datetime(2010, 7, 11))
datetime.datetime(2010, 7, 11, 0, 0)
>>> d.to_python(datetime(2005, 7, 11))
Traceback (most recent call last):
  ...
formencode.api.Invalid: Date must be after Wednesday, 22 April 2009
```

```
### Email addresses
>>> v.Email().to_python('karl@monetate.com')
'karl@monetate.com'
>>> v.Email().to_python('karlmonetate')
Traceback (most recent call last):
  ...
formencode.api.Invalid: An email address must contain a single @
```

Other interesting validators...

```
# RequireIfMissing()
# CreditCardValidator()
# IPAddress()
# URL()
```

Simple Form Schema

```
class UserForm(formencode.schema.Schema):
    """
    Validates some form elements.
    """
    first_name = validators.String(not_empty=True)
    last_name = validators.String(not_empty=True)
    email = validators.Email()
    username = validators.String(not_empty=True)
```

```
### Validate user info fields
validators.UserForm.to_python(
    {
    'first_name': 'karl',
    'last_name': 'shouler',
    'username': 'kmano8',
    'email': 'karl@monetate.com'
    })
```

Simple Password Schema

```
class PasswordForm(formencode.schema.Schema):
    """
    Validates password fields
    """
    new_password = SecurePassword(not_empty=False)
    new_password_again = validators.String(not_empty=False)
    chained_validators = [validators.FieldsMatch(
        'new_password', 'new_password_again')]
```

```
class SecurePassword(validators.FancyValidator):

    def validate_python(self, value, state):
        if len(value) < 8:
            raise formencode.Invalid('Your password must be at least 8 characters',
                value, state)
        non_letters = self.LETTER_REGEX.sub('', value)
        if len(non_letters) < self.REQ_NON_LETTER:
            raise formencode.Invalid(
                self.message('Your password requires at least 1 non-alpha character',
                value, state)
```

Validators...

Straightforward to implement
Yield tasty results

karlshouler.com

karl@monetate.com

@kmano8