# Generating Headless Javascript Tests for Validations


js.chi
2009/06/25
Kurt Stephens
CashNetUSA


http://kurtstephens.com/
files/headless_js_testing.pdf

# Problem

- Many Views.
- Many Abstract Data Types (ADT).
- Many Models that the same ADTs.
- Many different validations on each ADT depending on the context (localization).
- Client **and** Server-side validation required.
- Client and Server are different platforms (JS .vs. RoR).
- Traditional JS testing in a browser eats humans alive.

# Issues and Tech

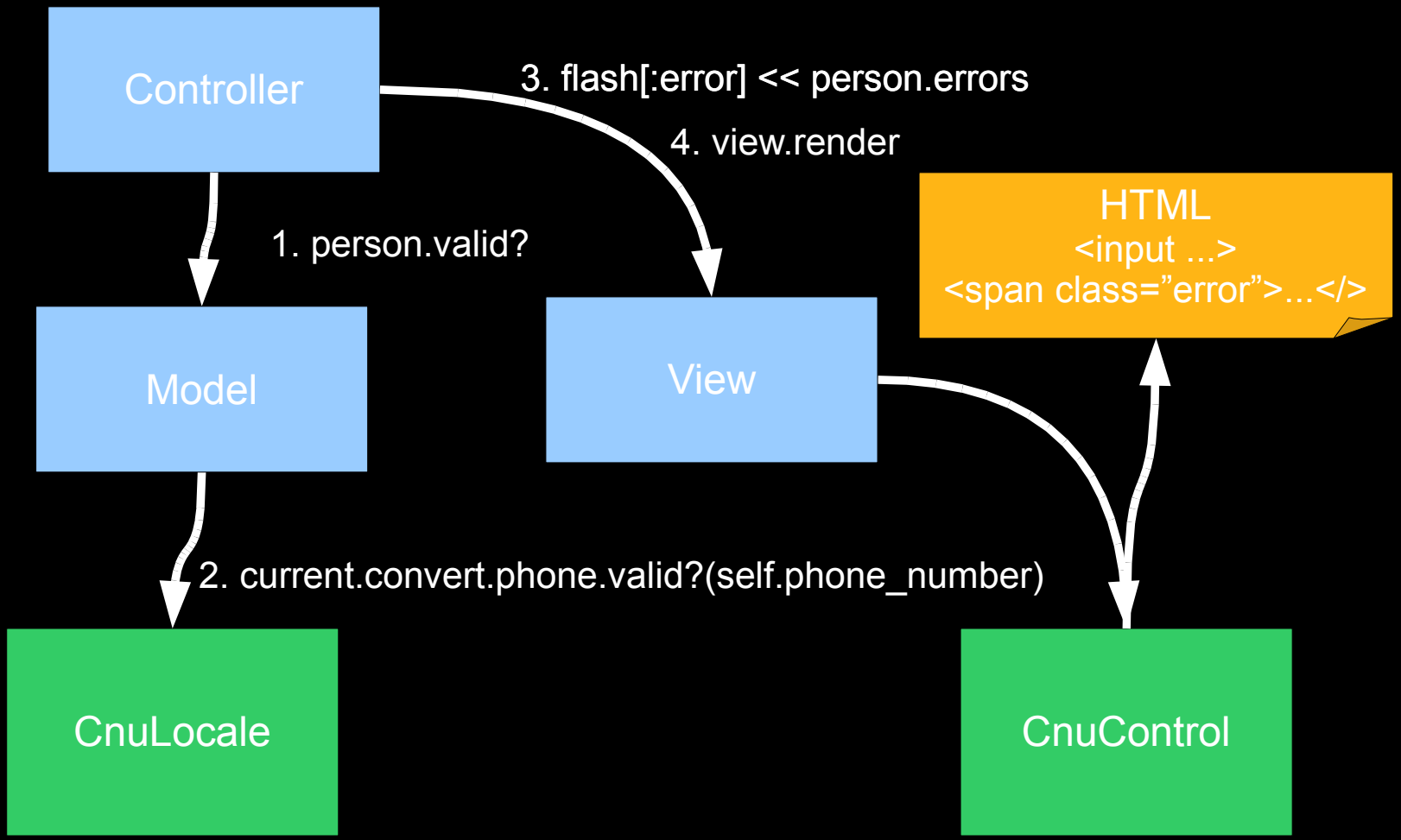
- MVC
- Ruby and Rails
- JavaScript (aka ECMAScript)
- Localization
- Client-side and Server-side validations
- Rspec
- Seamonkey Stand-alone interpreter
  - apt-get install smjs

# Overview

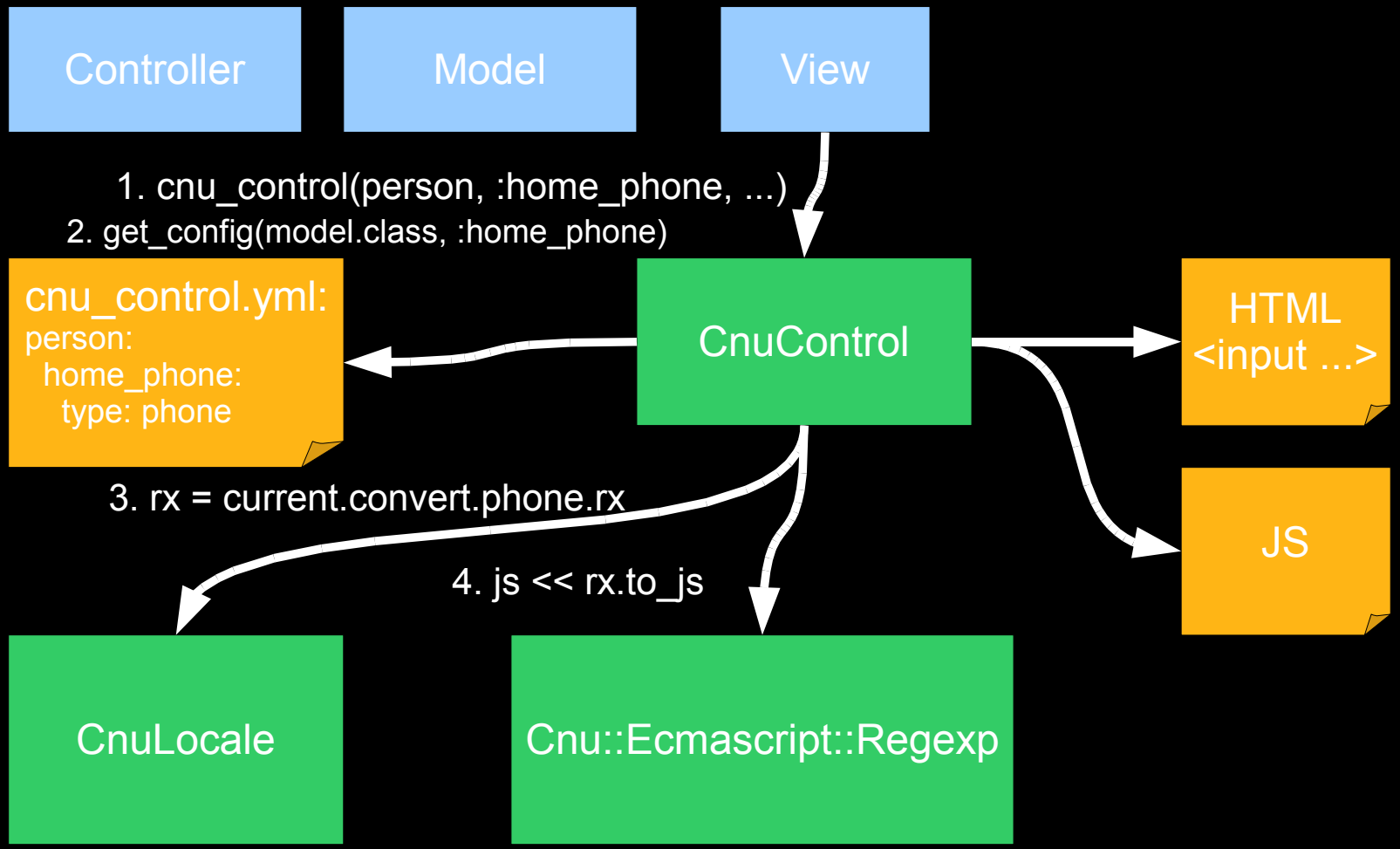
- Server- and Client-side Validations
- Widgets
- Localization
- Validations and Regular Expressions
- Translating Ruby Regexp to JS RegExp
- Test generation and Execution

# Layers

# Server-Side Validation



# Many Views beget Widgets



# Overkill?

# Localization

- CnuLocale
  - ADTs
    - Phone Number, Date, Money, Passport, etc.
  - Validations
    - Is user input valid?
    - Uses Regexp, but can be programatic.
  - Normalization
    - ”+61 02 1234 5678” => ”161212345678”
  - Presentation
    - ”161212345678” => AU ”(02) 1234 5678”
    - ”161212345678” => US ”+1 61212345678”

# This Can Happen To You!

# Australian Phone Number

```
def phone_number_au
  @@phone_number_au ||=
    begin
      sep = "[-.\\s]"
    /
    # +CC: +61 | 61 | \b
    ((?:\+|\b)\s*((?# country_code) 61)#{sep}?)?
    # m[:digits] will contain digits and formatting, without the + country code.
    ((?# digits)
    # 04 1234 5678     => 04 1234 5678
    # 0412345678       => 04 1234 5678
      ((?# area_code) 0?[234578])  #{sep}?((?# prefix) \d{4})#{sep}?((?# suffix) \d{4})
    |
    # (04) 1234 5678   => 04 1234 5678
    # (04)12345678     => 04 1234 5678
    \(((?# area_code) 0?[234578])\)#{sep}?((?# prefix) \d{4})#{sep}?((?# suffix) \d{4})
    |
    # 0412 345 678     => 04 1234 5678
    # 0412 345678      => 04 1234 5678
      ((?# area_code) 0?[234578])((?# prefix) \d{2}  #{sep} \d{2})((?# suffix) \d{1}#{sep}?\d{3})
    |
    # (0412) 345 678   => 04 1234 5678
    # (0412)345 678    => 04 1234 5678
    # (0412)345678     => 04 1234 5678
    \(((?# area_code) 0?[234578])((?# prefix) \d{2}\)#{sep}?\d{2})((?# suffix) \d{1}#{sep}?\d{3})
    |
    # 1800 123 456     => 1800 123 456
    # 1800123456       => 1800 123 456
      ((?# area_code) 1[38]00)  #{sep}?((?# prefix) \d{3})#{sep}?((?# suffix) \d{3})
    |
    # ... A BUNCH MORE HERE! ...
    # 18 1234          => 18 '' 1234
    # 181234           => 18 '' 1234
      ((?# area_code) 1[38])  #{sep}?((?# prefix) )((?# suffix) \d{4})
    )
    \b
    /x.tag!
    end
end
```

# Keeping it Managable

# Elements of Style

- Optional '+' and or country code prefix:
  - `/((?:\+|\b)\s*((?# country_code) 61)#{sep}?)?/`
- Zero-length non-word assertions:
  - `/\b((?# digits) \d+)\b/`
- Tagged Captures:
  - `/((?# area_code) 0?[234578]) #{sep}?((?# prefix) \d{4})#{sep}?((?# suffix) \d{4})/`
- Extended Syntax:
  - `/something else/x`
- Tag Transformation:
  - `(”foo1234bar” =~ /foo((?# digits) \d+)bar/.tag!)[:digits] => ”1234”`

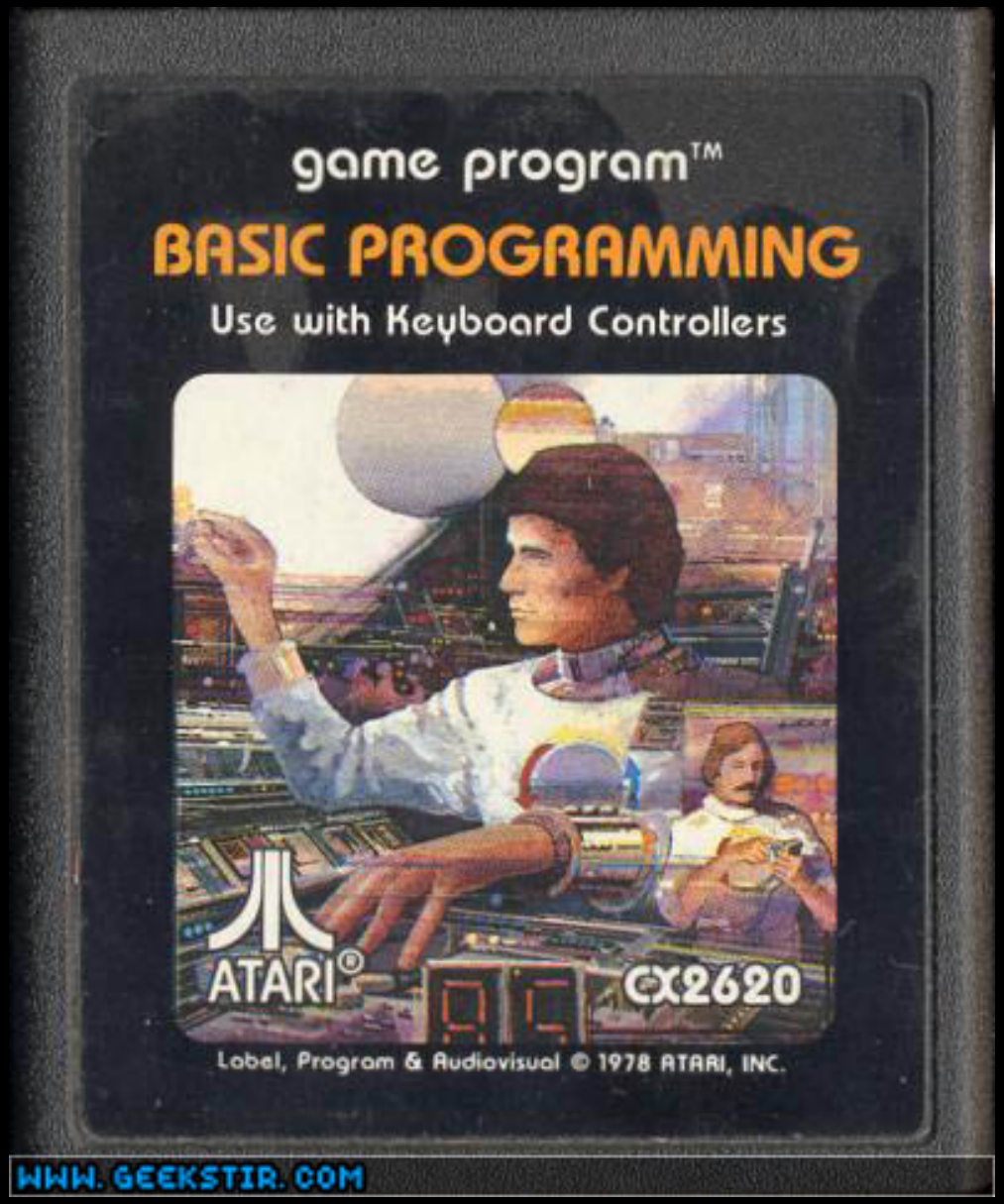
# Standards: Everyone's Got One

- Ruby Regexp
  - Handles (?# comments).
  - Use ((?# tag) captured expr) for naming captures.
  - //x enables insignificant whitespace and # comments.
- JS RegExp
  - Does not handle //x or comments.
  - \b does not match BOL and EOL.

# JS Regexp

- NO: \b also matches $ and ^
- NO: internal comments.
- NO: extended syntax prefix.
- NO: extended syntax comments (e.g.: "\n# foo bar\n")
- YES: non-capturing groups.
- YES: pattern multiplicity: (e.g.: /\d{3}/ => /\d\d\d/)
- NO: /\A/ or /\Z/
- YES: /\b/ zero-width word-boundary assertion.
- YES: /\B/ zero-width non-word-boundary assertion.
- YES: look-ahead assertions
- NO:  look-behind assertions

# Basics


game program™
BASIC PROGRAMMING
Use with Keyboard Controllers
ATARI®
CX2620
Label, Program & Audiovisual © 1978 ATARI, INC.
WWW.GEEKSTIR.COM

# Regexp in the Wild

- CnuControl
  - Generate JS and executed server-side validation for HTML widgets for ADT values.
- CnuLocale
  - Regexp with tagged captures to find relevant ADT data in user input.
  - Normalize relevant user input to ADT.
- Cnu::Ecmascript::Regexp
  - Converts Ruby Regexp objects to JS RegExp code.
  - Ruby /\bfoo\b/.to_js => JS expr "(new RegExp(\"(?:\\\\b|^|$)foo(?:\\\\b|^|$)\", \"\"))"

# My Wookie is Also My Bookie

HAN, STFU AND DRIVE!

http://www.morningstar.nildram.co.uk/A_New_Sith.html

# Testing

- Rspec test helper:
  - Tests Ruby Regexp for match/not match on a string literal.
  - Generates JS expression that is the same test on a JS Regexp generated from the Ruby Regexp.
  - Executes the test using smjs (seamonkey JS interpreter)

# Rspec

```
it "converts Regexp.phone_number_au" do
    rx = Regexp.phone_number_au
    check_rx(rx, "", false)
    check_rx(rx, "foobar", false)
    check_rx(rx, "61 (03) 1234 5678", true)
    check_rx(rx, "+61.(03).1234.5678", true)
end
```

# Generated Ruby and JS Tests

```
check_rx(rx, “foobar”, false) =>

  (rx === “foobar”).should == nil


  cnu_assert(
  "(new RegExp(\"...\", \"\")).test(\"foobar\") == false",
  (new RegExp("((?:\\+|(?:\\b|^|$))\\s*(61)[-.\\s]?)?((0?[234578])[-.\\s]?
(\\d{4})[-.\\s]?(\\d{4})|\\((0?[234578])\\)[-.\\s]?(\\d{4})[-.\\s]?
(\\d{4})|(0?[234578])(\\d{2}[-.\\s]\\d{2})(\\d{1}[-.\\s]?\\d{3})|\\((0?
[234578])(\\d{2}\\)[-.\\s]?\\d{2})(\\d{1}[-.\\s]?\\d{3})|(1[38]00)[-.\\s]?
(\\d{3})[-.\\s]?(\\d{3})|\\((1[38]00)\\)[-.\\s]?(\\d{3})[-.\\s]?(\\d{3})|
(1[38]0)[-.\\s]?()(\\d{4})|\\((1[38]0)\\)[-.\\s]?()(\\d{4})|\\((1[38])\\)
[-.\\s]?()(\\d{4})|(1[38])[-.\\s]?()(\\d{4}))(?:\\b|^|$)",
"")).test("foobar"),
  "==",
  false);
```