3 ways to make wrong code look wrong-er

The Perl and Raku Conference | 2022 – Houston, TX

Talk by: codesections <§> Daniel Sockwell
daniel@codesections.com

Introductions

- About me: (codesections)
  - Languages: JavaScript, lisp, Rust
  - Acronyms: RSC, TPF, JD
- About you:
  - At least some Raku experience

Outline

- Joel Spolsky’s post
- His solution
- A Raku alternative
- Another way to do it

The Problem

↪www.joelonsoftware.com/2005/05/11/making-wrong-code-look-wrong/

- Web application
- Accept user input
- Avoid XSS

Rejecting 2 NON solutions

HTML-encode all input

```
my $name = html-encode prompt 'User name?';
#  input: 'I <3 Raku'
#  $name = 'I &lt;3 Raku'
$database.username.store: $name;
# SQL db has HTML escapes in it
#   Oops!
```

Rejecting 2 NON solutions

HTML-encode all output

```
my $name = prompt 'User name?';
my $linebreak = '<br>'
#   ...
render html-escape $linebreak ~ $name;
# prints literal '<br>', not HTML break element
#   Oops!
```

Joel’s Solution

Hungarian Notation

Apps Hungarian Example

```
my $us-name = prompt 'User name?';

$database.username.store: $us-name;

my $s-name = html-escape $us-name;

my $s-linebreak = '<br>';

render $s-linebreak ~ $s-name;
# prints '<br>I &lt;3 Raku'
# ✔ yep!
```

Apps Hungarian Lesson

Wrong code
should <u>look</u> wrong
in <u>**isolation**</u>

Apps Hungarian Drawbacks

Wait … what's with the country code in `$us-name`*?*

A Raku solution

~~Sigils!~~

Twigils!

Raku twigil example

```
my $↯name = prompt 'User name?';

$database.username.store: $↯name;

my $⟡name = html-escape $↯name;

my $⟡linebreak = '<br>';

render $⟡linebreak ~ $⟡name;
# prints '<br>I &lt;3 Raku'
# ✔ yep!
```

Wait, custom twigils?

- Raku variables
  - start with a “letter”
- Rakudo letters
  - Unicode L (Ll+Lu+Lo+…)

Acme::Custom::Sigils

```
say '©'.uniname; # COPYRIGHT SIGN
say '©'.uniprop; # So

say $© = 'nope';
#    Throws:
#    ===SORRY!=== Error while compiling:
#    Name must begin with alphabetic character
```

Acme::Custom::Sigils

```
say 'ϟ'.uniname; # OSAGE CAPITAL LETTER WA
say 'ϟ'.uniprop; # Ll
my $ϟ = 'value'; # ✔

say 'ⲥ'.uniname; # OPTIC SMALL LETTER OLD COPTIC AI
say 'ⲥ'.uniprop; # Ll
my $ⲥ = 'value'; # ✔
```

Custom Twigils

- Pros: - like Hungarian
  - more concise
  - clearly type-ish
  - twigles = Raku-ish
- Cons: - a hack?
  - potentially cryptic

TIMTOWTDI

Current:

- Type-ish
- Code looks wrong to **us**

Goal:

- Real types
- Code looks wrong to **Raku**

The API we want:

```
role HtmlStr { ... }
class SafeStr   does HtmlStr { ... }
class UnsafeStr does HtmlStr { ... }

my SafeStr() $linebreak = '<br>';
render $linebreak; # should work

my UnsafeStr() $name = prompt 'User name?';
render html-encode $name; # should work
render $name;             # should NOT work
```

Implementation:

```
class SafeStr {...}
role HtmlStr {
    has $!str is built handles <Str gist raku>;

    method COERCE(Str $str) { self.new: :$str }

    method encode {
        # omitted
        SafeStr($!str) }
}
class UnsafeStr does HtmlStr {}
class SafeStr   does HtmlStr {}
```

Implementation, pt 2:

```
sub html-encode(UnsafeStr $_) { .encode }

sub render(SafeStr $html) { say $html }

# Concatenating SafeStrs is safe
multi infix:<~>(SafeStr $lhs, SafeStr $rhs) {
    SafeStr($lhs.Str ~ $rhs.Str)
}
```

Using the code

```
my UnsafeStr() $name = prompt 'User name?';
$database.username.store: $name;

my SafeStr() $linebreak = '<br>';
my SafeStr $user = html-encode $name;

render $linebreak ~ $user; # ✔ '<br>I &lt;3 Raku'
render $linebreak ~ $name; # ✔ THROWS
```

Full code (reference)

```
class SafeStr {...}
role HtmlStr  {
    has $!str is built handles <Str gist raku>;

    method COERCE(Str $str) { self.new: :$str }

    method encode { # omitted body
                    SafeStr($!str) }
}
class UnsafeStr does HtmlStr {}
class SafeStr   does HtmlStr {}

sub html-encode(UnsafeStr $_) { .encode }
sub render(SafeStr $html) { say $html }

multi infix:<~>(SafeStr $lhs, SafeStr $rhs) { SafeStr($lhs.Str ~ $rhs.Str) }

my UnsafeStr() $name = prompt 'User name?';
my SafeStr() $linebreak = '<br>';
my SafeStr $user = html-encode $name;

render $linebreak ~ $user; # ✔ '<br>I &lt;3 Raku'
render $linebreak ~ $name; # ✔ THROWS
```

Real types

- Pros: - compiler enforced
  - self-documenting
- Cons: - more setup
  - runtime cost?

Three ways, concluded

- Which of the 3 wins?
- Wrong question!

3 ways to make wrong code look wrong-er

The Perl and Raku Conference | 2022 – Houston, TX

Talk by: codesections <§> Daniel Sockwell
daniel@codesections.com