`String`

is not a sufficient type

(how using your type system can help you make better software)

About Me

→ Chris Dzombak

→ iOS Frameworks Team

→ dzombak@nytimes.com

→ @dzombak on Slack

→ @**c**dzombak on Twitter

Relax.

Let’s talk about `String`.

→ user input

→ human language words

→ output for the UI

→ SQL statements

→ keypaths for Cocoa KVO/KVC

→ …

→ …

→ XPath queries for XML

→ shell commands

→ HTML

→ regular expressions

→ and more!

`String` **is totally insufficient to represent all these different things**

Documentation

- **Compensate For Weak Type Information** as needed to clarify a parameter's **role**.

  ▼ COLLAPSE

  Especially when a parameter type is `NSObject`, `Any`, `AnyObject`, or a fundamental type such `Int` or `String`, type information and context at the point of use may not fully convey intent. In this example, the declaration may be clear, but the use site is vague:

```
func add(observer: NSObject, for keyPath: String)

grid.add(self, for: graphics) // vague
```

  To restore clarity, **precede each weakly-typed parameter with a noun describing its role**:

```
func addObserver(_ observer: NSObject, forKeyPath path: String)
grid.addObserver(self, forKeyPath: graphics) // clear
```

SQL Injection

```
SELECT * FROM items
WHERE owner = 'hacker'
AND itemname = 'name';

DELETE FROM items;
--'
```

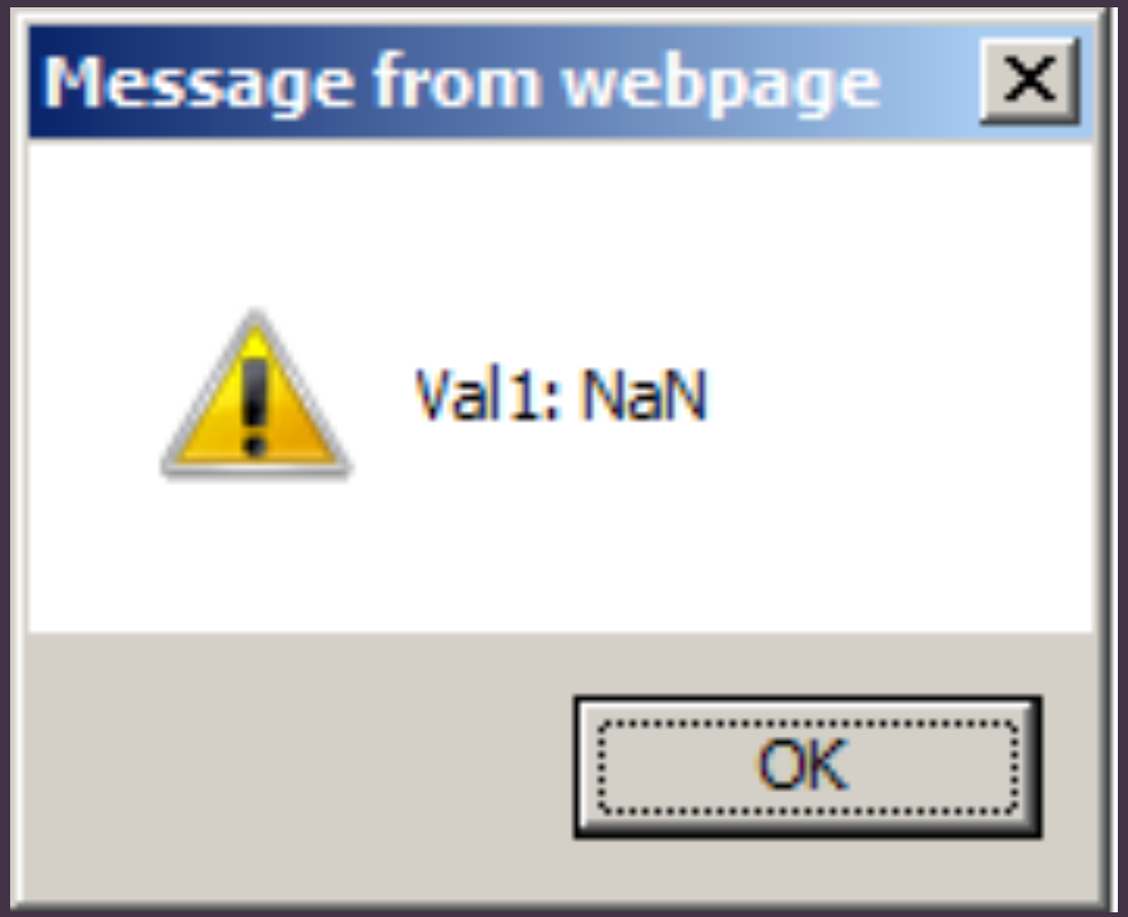
Shell Injection

```
run("gpg" , "--trust-model always -o
    \"#{File.expand_path(dst.path)}\" -e -r \"#{@recipient}\"
    \"#{File .expand_path(src .path)}\"")
```

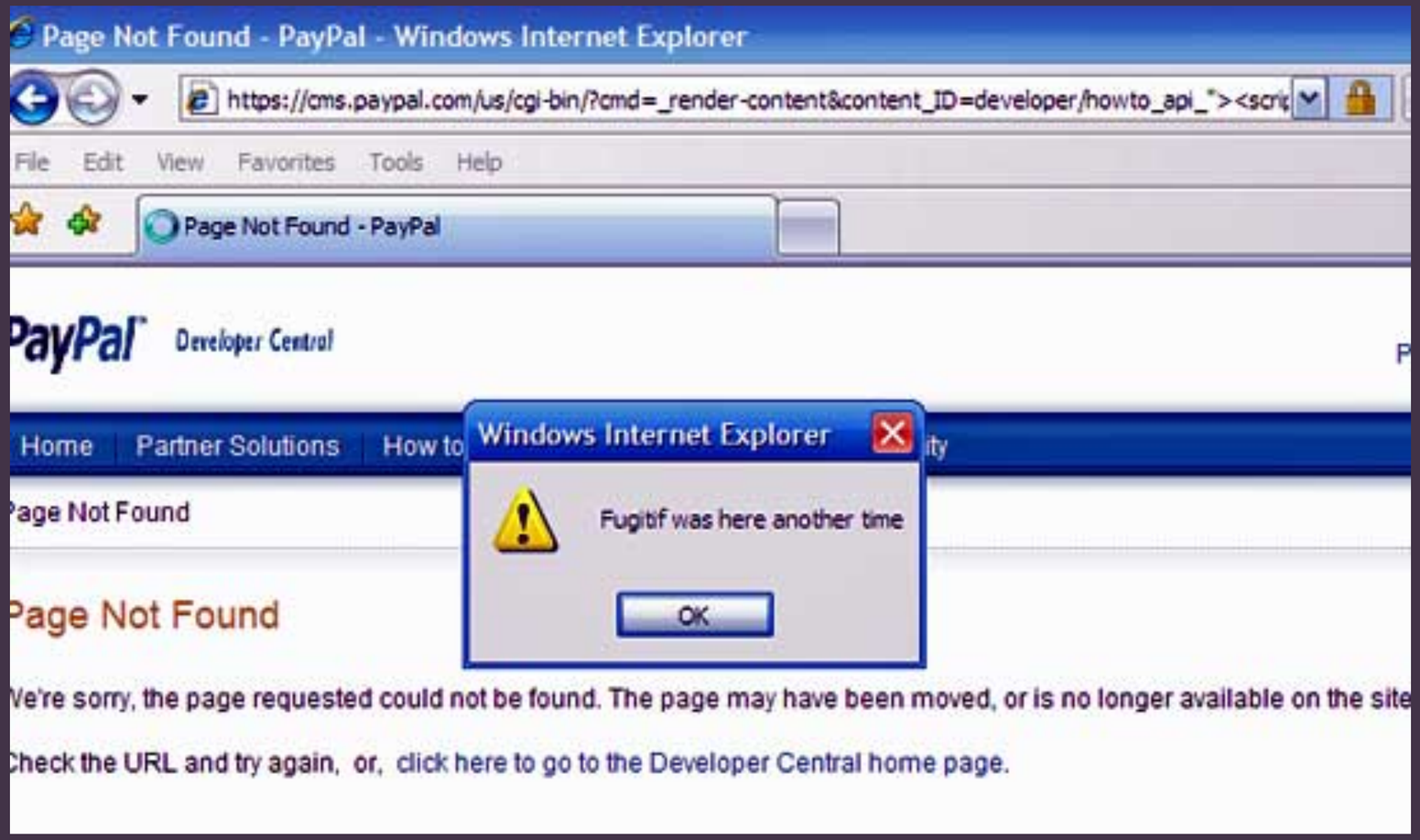
Code sample from the 2010 DC online voting pilot[1].

[1] https://jhalderm.com/pub/papers/dcvoting-fc12.pdf
https://freedom-to-tinker.com/blog/jhalderm/hacking-dc-internet-voting-pilot

Messy UI failure modes



XSS


Page Not Found - PayPal - Windows Internet Explorer
https://cms.paypal.com/us/cgi-bin/?cmd=_render-content&content_ID=developer/howto_api_"><scri
File Edit View Favorites Tools Help
Page Not Found - PayPal
PayPal Developer Central
Home Partner Solutions How to
Windows Internet Explorer
Fugitif was here another time
OK
Page Not Found
Page Not Found
We're sorry, the page requested could not be found. The page may have been moved, or is no longer available on the site
Check the URL and try again, or, click here to go to the Developer Central home page.

What if we used different `String` types for...

→ user input

→ HTML output

→ SQL statements

→ shell commands

→ etc...

```
let username = inputUsername()

let sql: SQLString = "SELECT FROM `users` WHERE `name` = '" + username + "';"
database.execute(sql)
```

```
let username = inputUsername()

let sql: SQLString = "SELECT FROM `users` WHERE `name` = '" + username + "';"
> error:                                                      ^^^
> error: cannot combine `username` of type UserInputString with type SQLString
```

I’m not crazy.

43,560

43,560

43,560 ft²

Units in math are just like your type system.

Units in math are just like your type system.

$$43{,}560\ \text{ft}^2\left(\frac{12^2\ \text{in}^2}{\text{ft}^2}\right)\left(\frac{2.54^2\ \text{cm}^2}{\text{in}^2}\right)\left(\frac{\text{m}^2}{100^2\ \text{cm}^2}\right) = 4046.9\ \text{m}^2$$

Exactly like in high school physics, we should attach meangingful information to our strings.

Exactly like in high school physics, we should attach meangingful information to our strings.

Then, our compiler or runtime will make many common mistakes impossible.

What would it take to really do this?

Standard library

→ `String`

→ `UserInputString`

→ `PathString` and `URLString`

→ (or maybe filesystem paths and URLs should be represented by separate, more capable objects entirely)

Native UI library

→ `UserFacingString`

→ A function accepting `Strings`, numbers, null references; filtering them; and outputting a “sane” string

Web templating library

→ `HTMLEscapedString`

→ A function accepting other `Strings` and escaping them for output

Database library

→ `SQLStatement`, `SQLEscapedString`

→ `SQLStatement` may only be constructed from programmer-controlled origins

→ Only `SQLEscapedString` may be combined into `SQLStatement`, in predefined safe ways

→ A function accepting `Strings` and escaping them for SQL

Cocoa KVC/KVO

→ `KeyPath`

→ `addObserver:forKeyPath:…` et al. accept `KeyPath`

→ Build `KeyPath` from string literals, with validation

→ Build `KeyPath` from runtime reflection

→ `KeyPath` instances can only be manipulated in constrained, valid ways

This sounds hard 

This sounds hard 🤔

$$43{,}560\ \text{ft}^2\left(\frac{12^2\ \text{in}^2}{\text{ft}^2}\right)\left(\frac{2.54^2\ \text{cm}^2}{\text{in}^2}\right)\left(\frac{\text{m}^2}{100^2\ \text{cm}^2}\right) = 4046.9\ \text{m}^2$$

Using plain old `String` everywhere in your program is like a professional physicist foregoing units in their calculations.

This is work we're doing already.

This is work we're doing already.

Except when we forget.

This is work we're doing already.

→ `ESAPI.encoder()`

→ `mysql_real_escape_string`
(yes, I know it's deprecated)

→ Escaping shell metacharacters[2]

→ `label.text = name.length ? name : ""`

[2] http://stackoverflow.com/a/20053121

Implementation?

Implementation?

¯_(ツ)_/¯

Conclusions

Type systems exist and we should let them help us. ✅

Conclusions

A few new types would help eliminate whole classes of vulnerabilities and other bugs. ✅

Conclusions

This wouldn’t be hard or annoying; this is work we’re already doing. ✅

Discussion