# Parentheses on the Web

## Using Common Lisp for Web Programming

Alpheus Madsen

OpenWest Conference
Friday, May 9th, 2014

# Outline

# Why Common Lisp?

Why are we even thinking about using Common Lisp on the web? Isn't Lisp that weird, quirky, crufty, language with all those parentheses?

# Why Common Lisp?

Why are we even thinking about using Common Lisp on the web? Isn't Lisp that weird, quirky, crufty, language with all those parentheses?

Yes, Common Lisp is weird, quirky, and crufty... but along with that weirdness, comes *power*. Lisp has a certain simplicity to it that allows a "Lispnik" to do interesting, even complicated things–often things that would be difficult or impossible to do in other languages. Because of this, Common Lisp can be surprisingly cutting-edge as well.

# Why Common Lisp?

## Pavel Penev, *Lisp Web Tales*

As for how practical this language is for web development, It's as practical as you make it. Lisp is the perfect language for the gray areas where were we still haven't quite figured out how to do things. I believe the web is one such area, and experimentation and playful exploration of ideas is vital. This is what Lisp was designed for, not for the web specifically, but for what it is, a new playground where flexibility and creativity have room to grow.

# Outline

# Initial Goals

My Initial Goals

- To become more familiar with libraries available for Common Lisp web programming.
- To develop a basic login for a website.
- To extend that website in some sort of interesting way...
- To see if I could get ParenScript and Weblocks to play nicely with Angular.js, CSS3 and HTML5.

# Outline

# ParenScript

ParenScript.

- Compiles to native JavaScript *without* needing external JavaScript libraries.
- As a Common Lisp library, ParenScript has access to "native" macros and other Common Lisp features.
- The Common Lisp Object System (CLOS) is available to ParenScript as a separate JavaScript library.

# ParenScript

A Sample Using ParenScript.

```
(define-easy-handler
      (example :uri "/example.js") ()
  (setf (content-type*) "text/javascript")
  (ps
    (defun greeting-callback ()
      (alert "Hello World"))))
```

# ParenScript

Caveats.

- Unfortunately, there is a certain "impedence mismatch" between ParenScript and Common Lisp, as a result of ParenScript being tightly tied to JavaScript. For example, "+" is a function in Common Lisp, but not JavaScript, so mapcar doesn't work as expected.
- Another disadvantage: Browser debugging requires manual tracing from generated JavaScript code to ParenScript. (This is a common problem with langauges that compile to JavaScript.)

# Outline

# Weblocks

Slava Akhmechet, *A User Interface Definition Language in Common Lisp*

I like HTML in the same way I like PDF - as a document serialization format that does a reasonable job and that I never want to modify by hand. This is one of the main reasons I started Weblocks framework - I never wanted to write a line of HTML again. Ironically, I ended up writing a lot of HTML and learning more about its quirks, accessibility issues, and CSS hooks than I ever wanted to, but I finally ended up with a high level user interface definition language embedded into Common Lisp. Finally, HTML is out of my life, for good. Being lazy does pay off.

# Weblocks

Weblocks uses “widgets” and “views” to display data.

- A widget can be a function, or a string, or an object (I think. . . )
- A widget keeps track of its own state and updates itself.
- Composite widgets can contain other widgets (which can contain other widgets); each widget is rendered as ordered.

# Weblocks

An example of a CLOS class definition:

```
(defclass employee ()
  ((first-name :reader employee-first-name)
   (last-name :accessor employee-last-name
              :type string)
   (contract :accessor employee-contract
             :type (member :full-time :part-time
                           :consultant :intern))
   (age :accessor employee-age
        :type (or null integer))))
```

# Weblocks

Weblocks uses “views”, with “inheritance” and “composition”to render widgets.

- Each widget can have multiple, customized views.
- “Slots” can be added or hidden, as needed.
- CSS classes and IDs are automatically generated.
- Although a goal is to avoid HTML completely, it’s available for particularly unusual needs. (Once defined, a view can be re-used.)

# Weblocks

An example of a Weblocks view:

```
(defview employee-form
        (:type form :default-method :post)
  first-name
  (last-name :requiredp t
             :label "Family Name")
  (contract :present-as (radio :choices
                               '(:full-time :part-time
                                 :consultant :intern))
            :parse-as keyword)
  (age :present-as (input :max-length 3)
       :parse-as integer))
```

# Weblocks

Weblocks uses “continuations” to keep track of a user’s state.

- Automatically handles browser navigation .
- Uses AJAX to update each widget that automatically degrades when JavaScript is turned off.
- Gracefully handles sequential control workflow.

# Weblocks

An example of sequential control flow:

```
(with-flow (composite-widgets (root-composite))
  (if (and (yield agreement-1)
           (yield agreement-2)
           (yield agreement-3))
      (show-protected-content)
      (show-error)))
```

# Outline

# Revised Goals

Revised Goals (for Future Reference)

- To develop a website login using Weblocks.
- To see what can be done with CSS3 and HTML5, and see what role JavaScript might play. (Angular.js seems redundant at this point...)
- Random Idea: See if it's possible to use a Common Lisp GUI toolkit, or even an NCurses toolkit, to create non-HTML views.

# Questions?

Any Questions?

# References

[1] *Parenscript Tutorial*, `http://common-lisp.net/project/parenscript/tutorial.html`.

[2] Slava Akhmechet, *A User Interface Definition Language in Common Lisp*, `http://www.defmacro.org/ramblings/ui-dsl.html`.

[3] _____, *Continuations-Based Web Applications in Common Lisp With Weblocks*, `http://www.defmacro.org/ramblings/continuations-web.html`.

[4] *Weblocks User Manual*, `http://trac.common-lisp.net/cl-weblocks/wiki/UserManual`.

[5] Pavel Penev, *Lisp Web Tales*.