Designing a programming language for local reasoning and easy debugging

Whoami

- Robin Heggelund Hansen
- Consultant at Bekk
- Creator of Gren

Gren

- Is a purely functional, staticly typed programming language with ML-syntax
- It aims to be small and easy to learn, while still being performant and expressive enough for general use
- Gren targets JavaScript, for maxium portability
- Still in early development

Why?

There is no such thing as a perfect programming language ...

... but you can get pretty close within a specific domain

The sort of projects I work on

- Big corporations in public and private sectors
- Not technically advanced (rest services)
- Relatively low-traffic
- Important services that can make the news when there are issues

An interesting thing about consultants

- Don't tend to stick around
- I tend to work in codebases I have little experience with

My ideal programming language

- Make it easy to understand what code does and doesn't do, without requiring that I know the entire codebase
- Has guarantees and tooling that make it easy to pinpoint and fix problems
- Performance isn't terribly important, but it shouldn't get in my way

My ideal programming language

Enables local reasoning, and easy debugging

Local reasoning

What does that mean?

Error handling

```
canViewAccount : Request -> Account -> Bool
canViewAccount req account =
    let
        userDetails =
            decodeUserDetails req
    in
    case userDetails of
        Admin ->
            True
        User details ->
            List.member account.id details.accountIds
```

Managing side-effects

loadFromCache :: Key -> IO (Maybe Value)

Managing side-effects

module FileSystem (..)

openForRead : Permission -> String -> Task AccessError (ReadableFileHandle a)

Tradeoffs

- Code size
- But to me, that is a tradeoff worth making

Debugging

When local reasoning won't do

Step-Debuggers are useful

- Makes for easy exploration of the running application
- Makes it easier to learn how the language works
- Sometimes, reasoning fails

```
encodeHelp :: Int -> String -> String
encodeHelp num acc =
  let clamped =
       num .&. 31
     newNum =
       num `Bit.shiftR` 5
      newClamped =
        if newNum > 0
          then clamped .|. 32
          else clamped
     newAcc =
        base64Table ! newClamped : acc
   in if newNum > 0
        then encodeHelp newNum newAcc
        else List.reverse newAcc
```

```
encodeHelp :: Int -> String -> String
encodeHelp num acc =
  let clamped =
       num .&. 31
     newNum =
       num `Bit.shiftR` 5
      newClamped =
        if newNum > 0
          then clamped .|. 32
          else clamped
     newAcc =
       base64Table ! newClamped : acc
   in if newNum > ⊘ --< BREAK HERE
        then encodeHelp newNum newAcc
        else List.reverse newAcc
```

```
encodeHelp :: Int -> String -> String
encodeHelp num acc =
  let clamped =
        num .&. 31
      newNum =
        num `Bit.shiftR` 5 --< STEPS TO HERE</pre>
      newClamped =
        if newNum > 0
          then clamped .|. 32
          else clamped
      newAcc =
        base64Table ! newClamped : acc
   in if newNum > 0
        then encodeHelp newNum newAcc
        else List.reverse newAcc
```

```
encodeHelp :: Int -> String -> String
encodeHelp num acc =
  let clamped =
       num .&. 31
     newNum =
       num `Bit.shiftR` 5
      newClamped =
        if newNum > 0
          then clamped .|. 32
          else clamped
     newAcc =
        base64Table ! newClamped : acc
   in if newNum > 0
        then encodeHelp newNum newAcc
        else List.reverse newAcc
```

```
encodeHelp :: Int -> String -> String
encodeHelp num acc =
  let clamped =
       num .&. 31
     newNum =
       num `Bit.shiftR` 5
      newClamped =
        if newNum > 0
          then clamped .|. 32
          else clamped
     newAcc =
       base64Table ! newClamped : acc
   in if newNum > 0
        then encodeHelp newNum newAcc
        else List.reverse newAcc --< NOW HERE
```

Importance of stack traces

Map.!: given key is not an element in the map CallStack (from HasCallStack): error, called at libraries/containers/containers/src/Data/Map/Internal.hs:613:17 in containers-0.6.6:Data.Map.Internal

🕞 🔂 🛛 Elements Console Source	es Network Performance Memory Application Security Lighthouse	🔼 1 🏟 🗄 🗙		
Page Filesystem »	index.html ×	▶		
▼ □ top 521 521	7	 Paused on breakpoint 		
▼	9 var \$elm\$html\$Html\$Events\$on = F2(▶ Watch		
Users/robin/Workspace/gren/e 522	1 return A2(Breakpoints		
index.html 522	<pre>2 \$elm\$virtual_dom\$VirtualDom\$on, 3 avent</pre>			
522	s event, 4 \$elm\$virtual_dom\$VirtualDom\$Normal(decoder)); 5 })،	Pause on caught exceptions		
522	<pre>6 var \$elm\$html\$Html\$Events\$onClick = function (msg) {</pre>	🔻 💿 index.html		
522	7 return A2(✓ return A2(5239		
522	9 'click',	▼ Scope		
523	<pre>0 \$elm\$json\$Json\$Decode\$succeed(msg));</pre>	▼Local		
523 523 523 523 523 523 523	<pre>var \$elm\$html\$Html\$span = _VirtualDom_node('span'); var \$elm\$virtual_dom\$VirtualDom\$text = _VirtualDom_text; var \$elm\$html\$Html\$text = \$elm\$virtual_dom\$VirtualDom\$text; var \$elm\$html\$Html\$ul = _VirtualDom_node('ul'); var \$author\$project\$Main\$view = function (model) { model = 0 var smallRange = List fromArray(smallRange = {\$: '::', a: -1, b: {}}</pre>	<pre>this: undefined model: 0</pre>		
523	<pre>8 [model - 1, model, model + 1]); model = 0</pre>	▼ Call Stack		
523 524 524	<pre>9 Preturn DA2(0 \$elm\$html\$Html\$div, 1 _List_Nil,</pre>	\$author\$project\$Main\$view index.html:5239		
524	2 _List_fromArray((anonymous) index.html:3975		
524 524 524	A2(5 \$elm\$html\$Html\$div, 6 List Nil.	_Browser_makeAnimator index.html:4038		
524	7List_fromArray((anonymous) index.html:3973		
524 524 525	8 l 9 A2(0 \$elm\$html\$Html\$span,	_Platform_initialize index.html:1893		
525	1 _List_fromArray((anonymous) index.html:3957		
525	2 l 3 \$elm\$html\$Html\$Attributes\$id('count')	(anonymous) index.html:34		
525 525	4]), 5 _List_fromArray((anonymous) index.html:5295		

	Elements Console	Sources	Network	Performance	Memory	Application	Security	Lighthouse			\$: ×
Page	Filesystem »		Main ×					Þ	▶_ ♠	† †	→• v ⁄	۵
🕶 🗖 top		23	type Msg = C	licked					1 Pause	d on breal	cpoint	
▼			update : Msg -> Model -> Model						▶ Watch			
▼ ■	Users/robin/Workspace/gren/e	27	update msg model =						 Breakpoints 			
	index.html	29	Clic	Clicked ->						Pause on uncaught exceptions		
	Basics	30		Pause on caught exceptions								
	Browser			▼ Scope								
	Char		view : Model view model =	. –> Html Msg :								
	Dict	35	let	10			. 1		this:	undefine	ed	
	Html	36 37	in smal	IRange = [mo	odel – 1, mo	odel, model +	1]		model	0		
	Html.Attributes	38	Html.div						► small	Range: (3	3) [-1, 0	0, 1]
	Html_Events	1 391	I I HT	mi_01v ii					- DE CLOSULE			
		40		[Html.span					▶ Global			Window
	Json.Decode	40 41 42	1	[Html.span [Attribu	ute.id "cour	it"]	del 1		► Global	ck		Window
	Json.Decode Main	40 41 42 43	1 113	[Html.span [Attribu [Html.te , Html.buttor	ute.id "cour ext < Strin n	nt"] ng.fromInt mo	del]		 ▶ Global ▼ Call Sta ▶ \$author 	ck \$project\$l	Main\$view	Window
	Json.Decode Main Set	40 41 42 43 44	L 113	[Html.span [Attribu [Html.te , Html.button [Attribu Event of	ute.id "cour ext < Strin n ute.id "incu	nt"] ng.fromInt mo rease-count" cked	del]		 ▶ Global ▼ Call Sta ♦ \$author 	ck ŕ\$project\$l	Main\$view	Window v Main:38
	Json.Decode Main Set String	40 41 42 43 44 45 46	[[Html.span [Attribu [Html.te , Html.button [Attribu , Event.c]	ute.id "cour ext < Strin n ute.id "inco onClick Clio	nt"] ng.fromInt mo rease-count" cked	del]		 Global Call Sta \$author (anonyr) 	ck ⁄\$project\$l nous)	Main\$view I index.h	Window v Main:38 ıtml:121
	Json.Decode Main Set String Task	40 41 42 43 44 45 46 47 48		[Html.span [Attribu [Html.te , Html.button [Attribu , Event.c] [Html.te	ute.id "cour ext < Strin n ute.id "inco onClick Clio ext "Count"	nt"] ng.fromInt mo rease-count" cked]	del]		 Global Call Sta \$author (anonyr Brows 	ck ⁄\$project\$l nous) er makeA	Main\$view I index.h nimator	Window v Main:38 tml:121
	Json.Decode Main Set String Task Url	40 41 42 43 44 45 46 47 48 49	, Ht	[Html.span [Attribu [Html.te , Html.button [Attribu , Event.c] [Html.te] :ml.ul []	ute.id "cour ext < Strin n ute.id "incr onClick Clic ext "Count"	nt"] ng.fromInt mo rease-count" cked]	del]		 Global Call Sta \$author (anonyr Brows 	ck ŕ\$project\$l nous) er_makeA	Main\$view I index.h nimator index.h	Window v Main:38 .tml:121 .tml:191
	Json.Decode Main Set String Task Url VirtualDom	40 41 42 43 44 45 46 47 48 49 50 51	, Ht]	[Html.span [Attribu [Html.te , Html.buttor [Attribu , Event.c] [Html.te] :ml.ul [] (Array.map (\	ute.id "cour ext < Strin n ute.id "incr onClick Clic ext "Count" \num -> Htm	nt"] ng.fromInt mo rease-count" cked] l.li [] [Html	del] .text < Str	ring.fromInt num]) smallR	 Global Global Call Sta \$author (anonyr Brows (anonyr 	ck r\$project\$l nous) er_makeA nous)	Main\$view I index.h nimator index.h index.h	Window v Main:38 tml:121 tml:191 tml:120
	Json.Decode Main Set String Task Url VirtualDom	40 41 42 43 44 45 46 47 48 49 50 51 52	, Ht]	[Html.span [Attribu [Html.te , Html.buttor [Attribu , Event.c] [Html.te] ml.ul [] (Array.map (\	ute.id "cour ext < Strin n ute.id "incr onClick Clic ext "Count" \num -> Htm ⁻	nt"] ng.fromInt mo rease-count" cked] l.li [] [Html	del] .text < Str	ring.fromInt num]) smallR	 Global Global Call Sta \$author (anonyr Brows (anonyr 	ck f\$project\$l nous) er_makeA nous) rm_initializ	Main\$view I index.h nimator index.h index.h	Window Wain:38 Itml:121 Itml:191 Itml:120
	Json.Decode Main Set String Task Url VirtualDom	40 41 42 43 44 45 46 47 48 49 50 51 52	, Ht]	[Html.span [Attribu [Html.te , Html.buttor [Attribu , Event.c] [Html.te] ml.ul [] (Array.map (\	ute.id "cour ext < Strin n ute.id "incr onClick Clic ext "Count" \num -> Htm	nt"] ng.fromInt mo rease-count" cked] L.li [] [Html	del] .text < Str	ring.fromInt num]) smallR	 Global Global Call Sta \$author (anonyr Brows (anonyr Platfor 	ck *\$project\$l nous) er_makeA nous) rm_initializ	Main\$view I index.h nimator index.h index.h e Json.Decc	Window Wain:38 html:121 html:191 html:120 ode:619
	Json.Decode Main Set String Task Url VirtualDom	40 41 42 43 44 45 46 47 48 49 50 51 52	, Ht]	[Html.span [Attribu [Html.te , Html.buttor [Attribu , Event.c] [Html.te] ml.ul [] (Array.map (\	ute.id "cour ext < Strin n ute.id "incr onClick Clic ext "Count" \num -> Htm]	nt"] ng.fromInt mo rease-count" cked] L.li [] [Html	del] .text < Str	ring.fromInt num]) smallR	 Global Global Call Sta \$author (anonyr Brows (anonyr Platfor (anonyr 	ck *\$project\$l nous) er_makeA nous) rm_initializ	Main\$view index.h nimator index.h index.h g Json.Decc index.h	Window Main:38 Itml:121 Itml:121 Itml:120 Dde:619
	Json.Decode Main Set String Task Url VirtualDom	40 41 42 43 44 45 46 47 48 49 50 51 52	, Ht]	[Html.span [Attribu [Html.te , Html.buttor [Attribu , Event.c] [Html.te] ml.ul [] (Array.map (\	ute.id "cour ext < Strin ute.id "incr onClick Clic ext "Count" \num -> Htm	nt"] ng.fromInt mo rease-count" cked] L.li [] [Html	del] .text < Str	ring.fromInt num]) smallR	 Global Global Call Sta \$author (anonyr Brows (anonyr Platfor (anonyr (anonyr 	ck f\$project\$l nous) er_makeA nous) rm_initializ nous)	Main\$view I index.h nimator index.h index.h Json.Deco index.h index.h	Window Main:38 tml:121 tml:121 tml:120 ode:619 tml:104 html:34

Tradeoffs

- Strict evaluation makes it easier to step through the code
- Readable stack traces make it easy to locate grivious errors
- Being able to debug the actual source code complicates and slows down the compiler
- Using the target platform's primitive types makes it easier to inspect state

Do we need a new language for this?

Problems with new languages

- Learning them takes time and commitment
- People usually have limited time to learn new things
- Few are willing to bet on a language without a future
- To be successfull the language needs to be small, and have a low complexity budget.
- Also, should be portable.

Why make a new language?

- Haskell is big and complex. Could do better on local reasoning and debugging.
- Elm is great! ... but it's hard to use for backends or terminal applications. Also, debugging experience could be better.



Gren

- Small, has simple but powerful features that compose, and aims to be learnable with a low time investment
- Great for local reasoning
- Integrates well with the JS debugger
- Can use it almost everywhere

Questions?



Gren: https://gren-lang.org Mastodon: @robinheghan@snabelen.no