



## Quick functional UI sketches with Lua templates and mermaid.js

Alexander Gladysh  
@agladysh

FOSDEM 2017

# Talk plan

1. The Case
2. Approaches to design
3. Enter the Mermaid
4. Lugram Templates
5. Conclusion
6. Questions?

# About me

- Programmer background
- Mainly doing management work now
- In löve with Lua since 2005

# The Case

- A huge professional enterprise application
- being converted from 20-year-old windows app
- to a modern SPA web-app.

# The product is huge

Sufficient expertise can only be found on a team level:

- Technology experts don't have product-level vision
- PO and PM don't draw professionally (and lack deep insight on the tech)
- Designer does not have the professional-level technology expertise

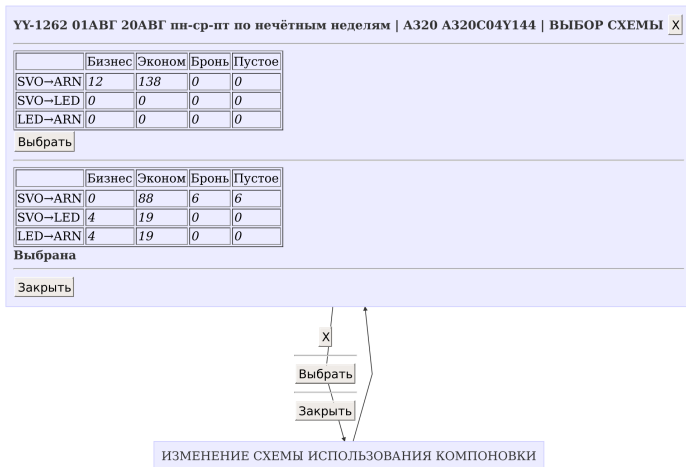
# The UI design and development process

For each "screen" in the application:

- Concept
- Functional sketches and (sometimes) interactive studies
- Design sketches
- Layout implementation
- Business logic implementation

# A functional sketch

What is on the screen, how does it WORK?



# A design sketch

How does it LOOK?

★ EQUIP ПЛ-1264 01.JAN

ПЛ-1264 Изменение схемы использования компоновки

SVO → LED → ARN  
01 JAN

CY9 / CY9-100-95  
Кресел 60. Бизнес 8. Эконом 70.

20m ago

×

Выберите схему использования компоновки:

	SVO → ARN	SVO → LED	LED → ARN
Бизнес	15	2	2
Эконом	43	8	8
Бронь	6	3	2
Пустое	6	1	0

	SVO → ARN	SVO → LED	LED → ARN
Бизнес	15	2	2
Эконом	43	8	8
Бронь	6	3	2
Пустое	6	1	0

	SVO → ARN	SVO → LED	LED → ARN
Бизнес	15	2	2
Эконом	43	8	8
Бронь	6	3	2
Пустое	6	1	0

	SVO → ARN	SVO → LED	LED → ARN
Бизнес	15	2	2
Эконом	43	8	8
Бронь	2	3	2
Пустое	10	1	0

	SVO → ARN	SVO → LED	LED → ARN
Бизнес	15	2	2
Эконом	43	8	8
Бронь	2	3	2
Пустое	10	1	0



# Goals

- I need a diagram of the flow between app screens
- I need functional sketches of the screens themselves
- Basically it doesn't matter how I make them as long as they are easy to make and change and there are some facilities for reuse.

# What tools to use?

- Photoshop (Krita, Gimp...)
- InkScape
- Google Documents
- Visio
- Balsamiq
- Sketch
- ...

I'm a programmer, I'm better with structured text than with images.

I work fastest with the keyboard, not having to touch the mouse.

# Enter the Mermaid

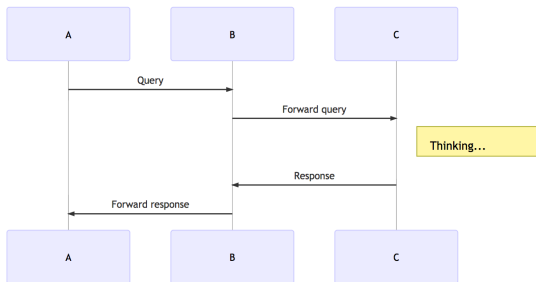
<http://bit.ly/mermaid-editor>

Enter mermaid diagram syntax here

```
sequenceDiagram
A->> B: Query
B->> C: Forward query
Note right of C: Thinking...
C->> B: Response
B->> A: Forward response
```

This webapp uses the [mermaid](#) library to generate graphs and sequence diagrams. Syntax for sequence diagrams can be found [here](#)

Generated diagram



[LINK TO VIEW](#)

[LINK TO EDIT](#)

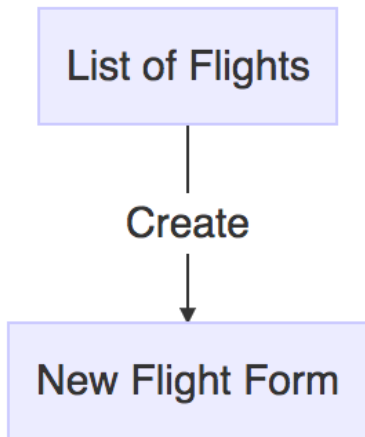
[DOWNLOAD SVG](#)

# Screen Flow Diagram

graph TD

```
list[List of Flights]  
new[New Flight Form]
```

```
list-->|Create|new
```

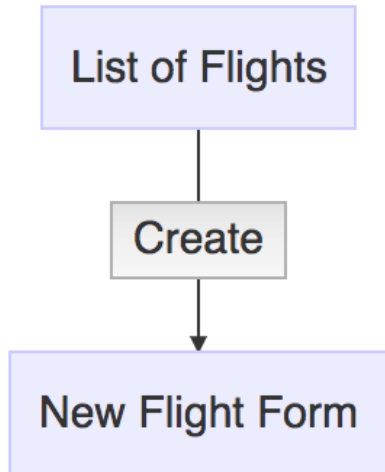


## Screen Flow Diagram (II)

graph TD

```
list[List of Flights]  
new[New Flight Form]
```

```
list-->|" "  
<button>Create</button>  
"|new
```

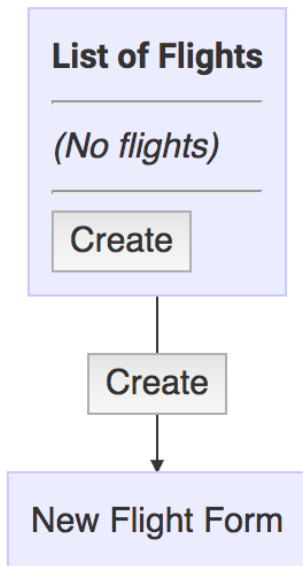


# Screen Prototypes

graph TD

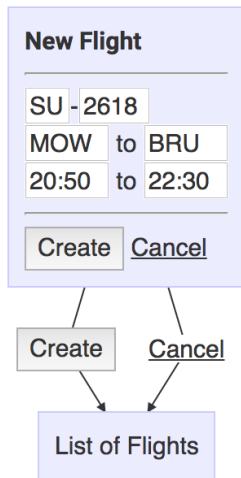
```
list["<b>List of Flights</b>  
<hr>  
<i>(No flights)</i><hr>  
<button>Create</button>"]
```

```
new["New Flight Form"]  
list-->|"  
<button>Create</button>  
"|new
```

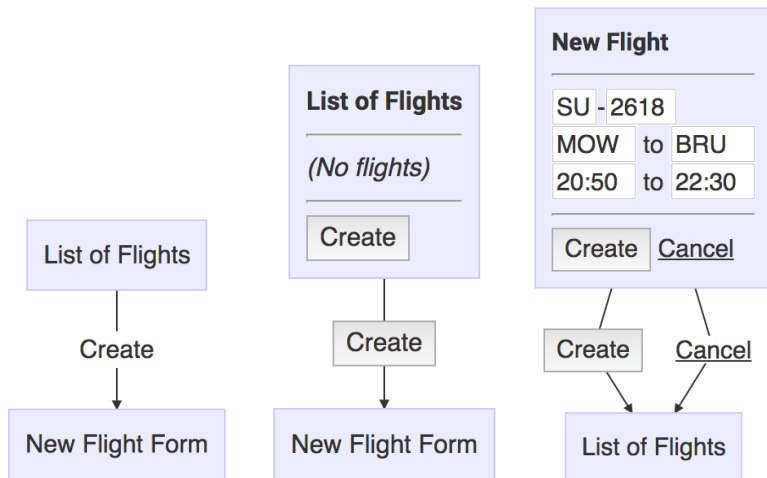


## Screen Prototypes (II)

```
graph TD
list["List of Flights"]
new["<b>New Flight</b><hr>
<input value='SU' size='2'>-
<input value='2618' size='4'><br>
<input value='MOW' size='5'> to
<input value='BRU' size='5'><br>
<input value='20:50' size='5'> to
<input value='22:30' size='5'>
<hr><button>Create</button>
<u>Cancel</u>"]
new-->|
<button>Create</button>"|list
new-->|"<u>Cancel</u>"|list
```



# Basic Documentation Structure





## It is hard to do HTML without templates: list

```
<b>${title List of Flights}</b><hr>
```

```
<i>(No flights)</i><hr>
```

```
${link new <button>Create</button>}
```

# It is hard to do HTML without templates: new

```
<b>${title New Flight}</b><hr>
```

```
<input value='SU' size='2'>-  
<input value='2618' size='4'><br>
```

```
<input value='MOW' size='4'> to  
<input value='BRU' size='4'><br>
```

```
<input value='20:50' size='5'> to  
<input value='10:30' size='5'><hr>
```

```
${link list <button>Create</button>  
${link list <button>Cancel</button>
```

# Basic helpers

```
${title New Flight}
```

```
$title <text:*>
```

```
${link list <button>Create</button>}
```

```
$link <screen:word> <body:*>
```

## Basic helpers: pass-through definitions

```
${define title {'*', 'text'}} [[${text}]]
```

```
${title New Flight} --> New Flight
```

```
${define link  
  {'word', 'target'}, {'*', 'body'}}  
  [[${body}]]
```

```
${link list <button>Create</button>  
--> <button>Create</button>
```

```
$define <symbol:word> <arguments:table> <code:*>
```

# define

```
define = function(context, str)
    local symbol, str = eat.word(str)
    local args, str = eat.table(str)
    local code, str = eat['*'](str)
    args, code = lua_value(args), lua_value(code)
    if type(code) == 'string' then code =
        function(ctx) return ctx:replace(code) end
    end
    context._ROOT[symbol] = function(parent, str)
        local ctx = { }
        for i = 1, #args do
            ctx[args[i][1]], str = eat[args[i][2]](str)
        end
        return code(parent:push(ctx))
    end
end
end
```

## helpers: definitions using Lua

```
function(title, 'text') function(context)
    local text = context:replace(context.text)
    context._ROOT._SCREENS[text] = text
    return text
end}
```

```
function(link {'word', 'target'}, {'*', 'body'})
function(context)
    local target = context:replace(context.text)
    context._ROOT._LINKS[target] = context._MODULE
    context:include(target) -- Ignoring result
    return context:replace(context.body)
end}
```

# include

```
include = function(context, template)
  return context:push(
    { _MODULE = template }
  ):replace(
    assert(io.open(filename)):read("*a")
  )
end
```

# Diagram styles

Depending on how you define \$title and \$link, you get several kinds of diagram from the same set of templates:

- Outline diagram (titles and arrows only, "screen flow")
- Closeup diagram (screen content and flow from this screen to others)
- Printable diagram (screen content only)



# More Useful helpers

```

${define # {{'*', 'comment'}}} [[]]}
${define --[HR]-----...----- {} [[<hr>]]}

${define expr {{'*', 'code'}}} function(context)
  return assert(loadstring('return '
    .. context:replace(context.code)))()
end}

```

## with helper

```

${define with {'table', 'more_context'},
  {'*', 'body'}} function(context)
  return context:replace(
    context:push(context.more_context),
    context.body)
end}

```

```

${define form {} [[
  ${when editable
    <input value='MOW'> to <input value='BRU'>}
  ${unless editable MOW to BRU}
  ]]}

```

```

${with {editable = true} ${form}}

```

## with helper (II)

```
${define histogram {'word', 'a'},  
  {'word', 'b'}, {'word', 'c'}} [[  
  ${with { w = '${expr ${a} + ${b} + {c}}' }  
    <div style='width:${w}px' class='red'>  
      <div style='width:${a}px'  
        class='green'>${a}</div>  
      <div style='width:${b}px'  
        class='blue'>${b}</div>  
    </div>  
  ]]]}  
  
${histogram 1 2 3}
```

# Some statistics

- Two days to implement core, about 250 LOC.
- After six months of non-fulltime usage, core grew to about 330 LOC, mostly additional diagnostics.
- About 60 sketches finalized (5KLOC of templates), more to come.

# Was it worth it?

Yes.

- I've got a low-cost lightweight flexible framework for design that does not chafe in wrong places.
- Its output, while not ideal, is reasonably understandable by all members of the team.
- Also: much fun implementing yet another template engine.

# Why not X?

- The cost is so low, the adaptation of existing tool to my requirements (or just learning the proper ropes) would probably cost about the same.
- But if you know a good candidate that fits here, please do chime in.

# Problems

- Error diagnostics and debugging for templates. Almost non-existent. Lots of low-hanging fruit there.
- Debugging of the HTML output render. IE6-hard, rather difficult to improve. Keep HTML simple.
- Expressive power of the language could be improved. No need so far.

Questions?

@agladysh

agladysh@gmail.com

<https://github.com/tais-aero/lugram>