Julia Code: Optimized

René Donner
code@donner.at

What can we do to make our code faster?

"Julia is fast"

Well, it **allows** to get C-like speed

But you can also write very **slow** code

Be aware what you ask the machine to do

Overview

Don't optimize too early - **profile** first

Know your **memory layout**

Type stability

Compiler hints

External **tools**

Profiling

Profiling

@time

Runtime, allocations, garbage collection

```julia
function f()
    for x in 1:100
        rand(100,1000)
    end
end
@time f()  # compiles code
@time f()
```

```
0.060411 seconds
(39.85 k allocations: 78.011 MB, 12.62% gc time)
0.025687 seconds
(404 allocations: 76.305 MB, 21.90% gc time)
```

Profiling

@time

```
function f()
    @time r = zeros(2,1)
    for i in 1:3
        @time r = r + rand(2,1)
    end
end
f(); f()
```

```
0.000001 seconds (1 allocation: 80 bytes)
0.000001 seconds (4 allocations: 224 bytes)
0.000000 seconds (4 allocations: 224 bytes)
0.000000 seconds (4 allocations: 224 bytes)
```

Profiling

@time

```
function f()
    r = zeros(2,1)
    for i in 1:3
        r = r + rand(2,1)
    end
    end
f(); @time f()
```

```
0.000002 seconds (17 allocations: 912 bytes)
```

```
function f()
    r = zeros(2,1)
    for i in 1:3
        for j in 1:2
            r[j] += rand()
        end
    end
end
f(); @time f()
```

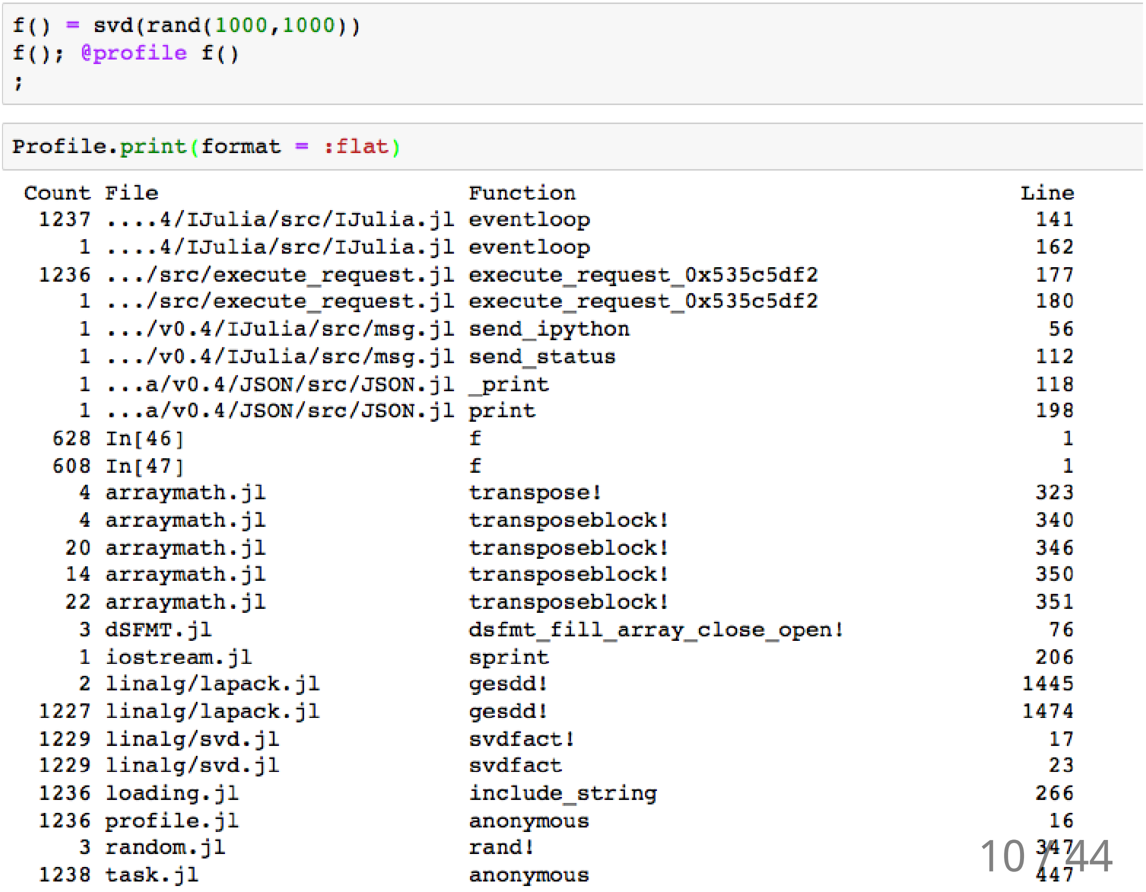
```
0.000001 seconds (5 allocations: 240 bytes)
```

Profiling

@time

@profile

```
@profile f()
Profile.print()

Profile.clear()
@profile g()
Profile.print(format = :flat)
```



```
f() = svd(rand(1000,1000))
f(); @profile f()
;
```

```
Profile.print(format = :flat)
```

```
 Count File                         Function                               Line
  1237 ....4/IJulia/src/IJulia.jl eventloop                               141
     1 ....4/IJulia/src/IJulia.jl eventloop                               162
  1236 .../src/execute_request.jl execute_request_0x535c5df2              177
     1 .../src/execute_request.jl execute_request_0x535c5df2              180
     1 .../v0.4/IJulia/src/msg.jl send_ipython                             56
     1 .../v0.4/IJulia/src/msg.jl send_status                             112
     1 ...a/v0.4/JSON/src/JSON.jl _print                                  118
     1 ...a/v0.4/JSON/src/JSON.jl print                                   198
   628 In[46]                     f                                         1
   608 In[47]                     f                                         1
     4 arraymath.jl               transpose!                              323
     4 arraymath.jl               transposeblock!                         340
    20 arraymath.jl               transposeblock!                         346
    14 arraymath.jl               transposeblock!                         350
    22 arraymath.jl               transposeblock!                         351
     3 dSFMT.jl                   dsfmt_fill_array_close_open!             76
     1 iostream.jl                sprint                                  206
     2 linalg/lapack.jl           gesdd!                                 1445
  1227 linalg/lapack.jl           gesdd!                                 1474
  1229 linalg/svd.jl              svdfact!                                 17
  1229 linalg/svd.jl              svdfact                                  23
  1236 loading.jl                 include_string                          266
  1236 profile.jl                 anonymous                                16
     3 random.jl                  rand!                                   347
  1238 task.jl                    anonymous                               447
```

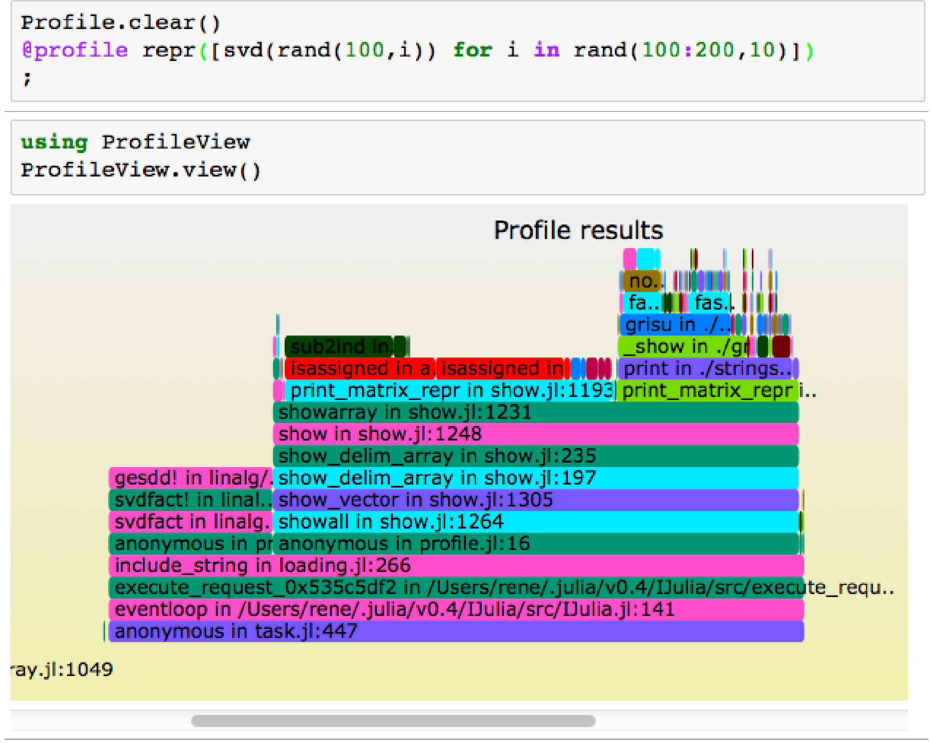
Profiling

```
@profile f()

using ProfileView
ProfileView.view()
```

@time

@profile

ProfileView



```
Profile.clear()
@profile repr([svd(rand(100,i)) for i in rand(100:200,10)])
;
```

```
using ProfileView
ProfileView.view()
```

Profile results

no..
fa..
fas..
grisu in ./..
sub2ind in.
_show in ./gr
isassigned in a.
isassigned in
print in ./strings..
print_matrix_repr in show.jl:1193
print_matrix_repr i..
showarray in show.jl:1231
show in show.jl:1248
show_delim_array in show.jl:235
gesdd! in linalg/.
show_delim_array in show.jl:197
svdfact! in linal..
show_vector in show.jl:1305
svdfact in linalg.
showall in show.jl:1264
anonymous in pr
anonymous in profile.jl:16
include_string in loading.jl:266
execute_request_0x535c5df2 in /Users/rene/.julia/v0.4/IJulia/src/execute_requ..
eventloop in /Users/rene/.julia/v0.4/IJulia/src/IJulia.jl:141
anonymous in task.jl:447
ray.jl:1049

Profiling

@time

@profile

ProfileView

Memory allocations

```
julia --track-allocation=user myscript.jl
```

```
mycode()
Profile.clear_malloc_data()
mycode()
```

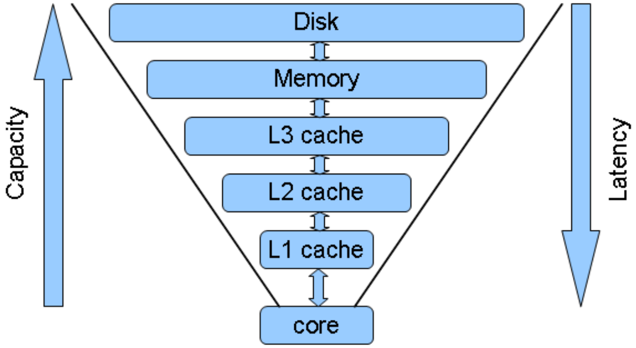
Results found in `.mem` files

`http://docs.julialang.org/en/release-0.4/manual/profile/#memory-allocation-analysis`

Memory Layout

Memory

Cache hierarchy



`http://www.1024cores.net/home/parallel-computing/cache-oblivious-algorithms`

Memory

Row major: First index changes infrequently

Column major: First index changes most often

==> memory layout

Memory

Cache hierarchy

Row major

```julia
function f(a)
    r = zero(eltype(a))
    for m in 1:size(a,1)
        for n in 1:size(a,2)
            r += a[m,n]
        end
    end
    r
end

function g(a)
    r = zero(eltype(a))
    for n in 1:size(a,2)
        for m in 1:size(a,1)
            r += a[m,n]
        end
    end
    r
end

a = rand(1000,100000)

f(a); @time f(a)
g(a); @time g(a)
```

Memory

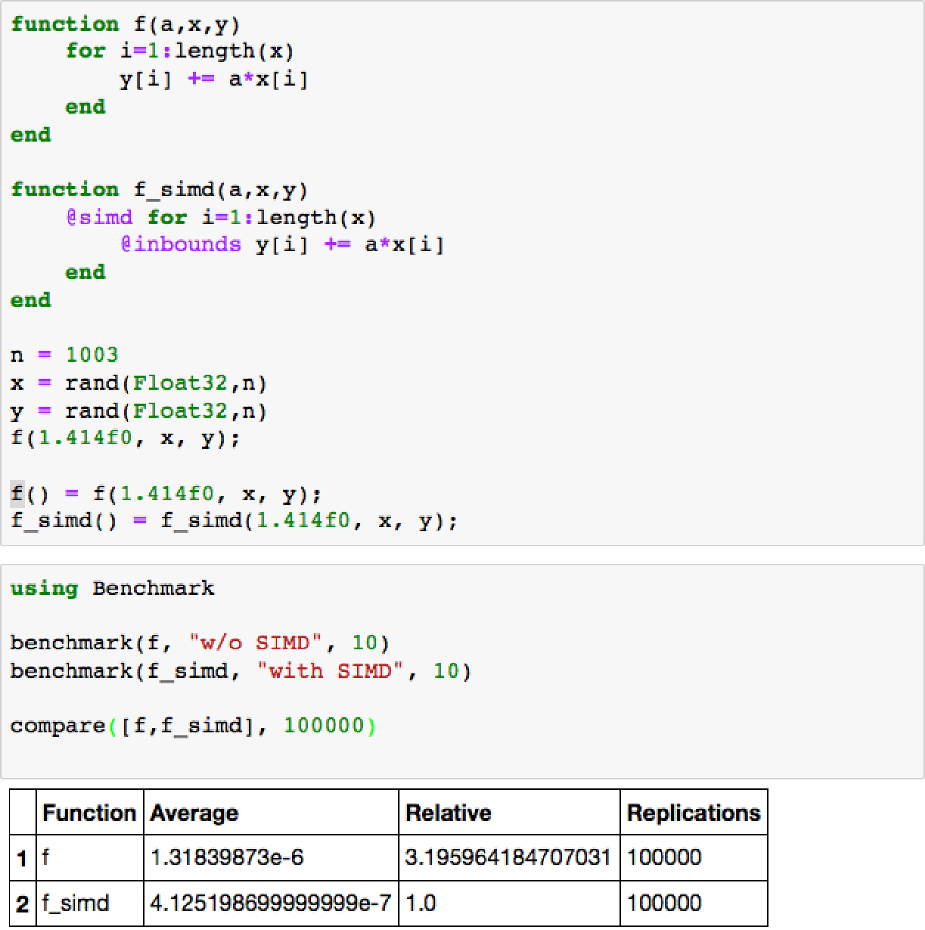
Cache hierarchy

Row major

```julia
function f(a)
    ...
end

function g(a)
    ...
end

a = rand(1000,100000)

f(a); @time f(a)
g(a); @time g(a)
```

```
0.964728 seconds (5 allocations: 176 bytes)
0.090653 seconds (5 allocations: 176 bytes)
```

- Cache aware algorithms
- Cache oblivious algorithms



```
function f(a,x,y)
    for i=1:length(x)
        y[i] += a*x[i]
    end
end

function f_simd(a,x,y)
    @simd for i=1:length(x)
        @inbounds y[i] += a*x[i]
    end
end

n = 1003
x = rand(Float32,n)
y = rand(Float32,n)
f(1.414f0, x, y);

f() = f(1.414f0, x, y);
f_simd() = f_simd(1.414f0, x, y);
```

```
using Benchmark

benchmark(f, "w/o SIMD", 10)
benchmark(f_simd, "with SIMD", 10)

compare([f,f_simd], 100000)
```

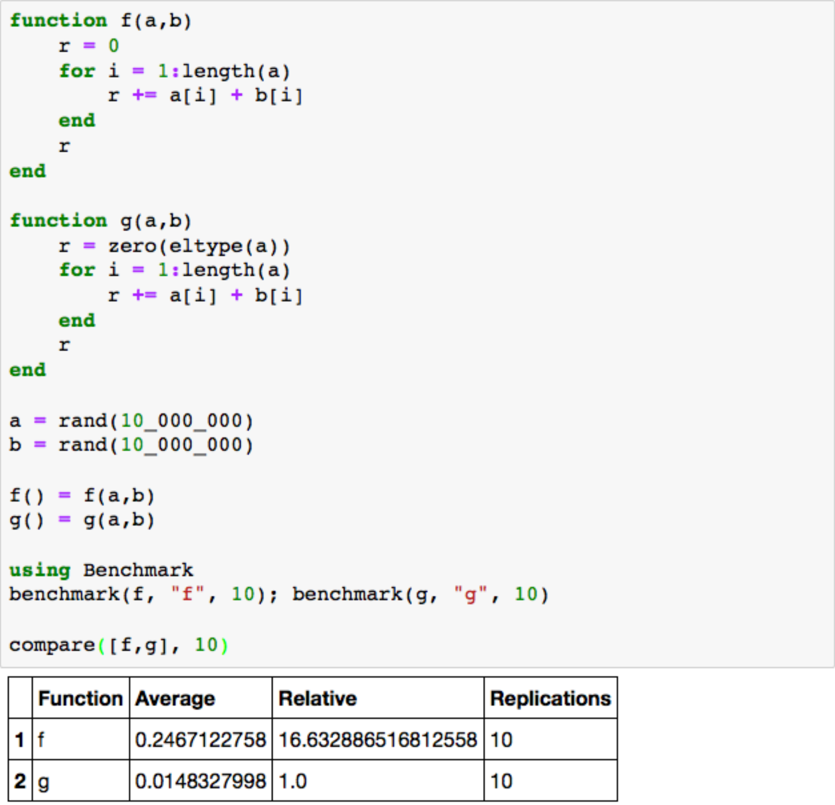
	Function	Average	Relative	Replications
1	f	1.31839873e-6	3.195964184707031	100000
2	f_simd	4.125198699999999e-7	1.0	100000

Type stability

Type stability

Typed vs Any

- Every variable has a type
- Concrete:
    - `Float64`
    - `Int`
- Boxed:
    - `Any`
    - `Union{Int, Float64}`



```
function f(a,b)
    r = 0
    for i = 1:length(a)
        r += a[i] + b[i]
    end
    r
end

function g(a,b)
    r = zero(eltype(a))
    for i = 1:length(a)
        r += a[i] + b[i]
    end
    r
end

a = rand(10_000_000)
b = rand(10_000_000)

f() = f(a,b)
g() = g(a,b)

using Benchmark
benchmark(f, "f", 10); benchmark(g, "g", 10)

compare([f,g], 10)
```

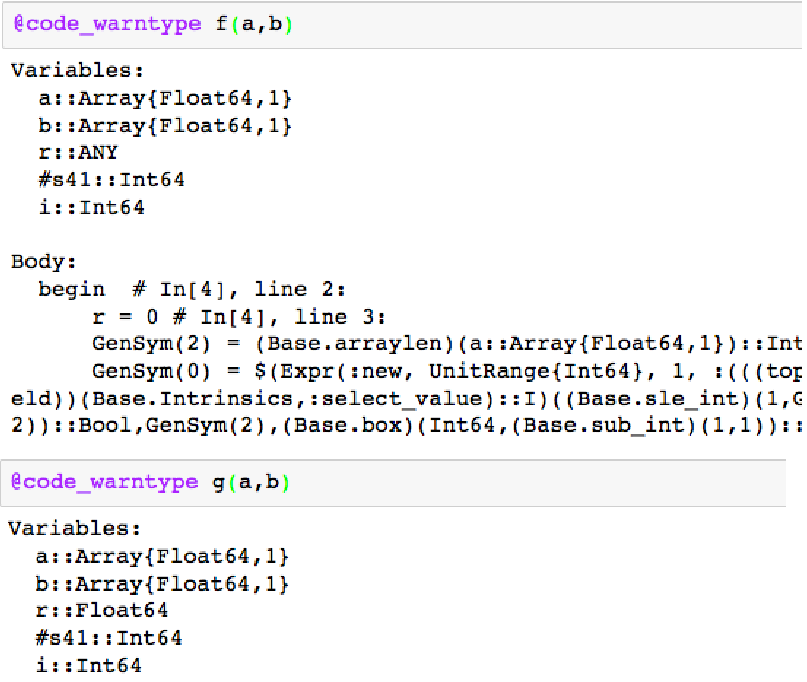
	Function	Average	Relative	Replications
1	f	0.2467122758	16.632886516812558	10
2	g	0.0148327998	1.0	10

Type stability

Typed vs Any

Effect

@code_warntype



```
@code_warntype f(a,b)

Variables:
  a::Array{Float64,1}
  b::Array{Float64,1}
  r::ANY
  #s41::Int64
  i::Int64

Body:
  begin  # In[4], line 2:
      r = 0 # In[4], line 3:
      GenSym(2) = (Base.arraylen)(a::Array{Float64,1})::Int
      GenSym(0) = $(Expr(:new, UnitRange{Int64}, 1, :(((top
eld))(Base.Intrinsics,:select_value)::I)((Base.sle_int)(1,G
2))::Bool,GenSym(2),(Base.box)(Int64,(Base.sub_int)(1,1))::
```

```
@code_warntype g(a,b)

Variables:
  a::Array{Float64,1}
  b::Array{Float64,1}
  r::Float64
  #s41::Int64
  i::Int64
```

Compiler hints

Compiler hints

- `@simd`, `@inbouds`
- `@inline`
- `@fastmath`

External tools

ParallelAccelerator

- From Intel, just released
- Converts Julia to C internally
- Uses OpenMP for parallelization

ParallelAccelerator

```
Unary functions
    -, +, acos, acosh, angle, asin, asinh, atan, atanh,
    cbrt, cis, cos, cosh, exp10, exp2, exp, expm1, lgamma,
    log10, log1p, log2, log, sin, sinh, sqrt, tan, tanh,
    abs, copy, erf

Binary functions
    -, +, .+, .-, .*, ./, .\, .%, .>, .<, .<=, .>=, .==,
    .<<, .>>, .^, div, mod, rem, &, |, $, min, max

Parallel Comprehensions

Stencils
    eg. image filtering
```

Julia Code: Optimized

René Donner
code@donner.at

Following: Documentation / examples for gistdeck / reveal.jl

How does it work?

- Markdown

- Inside HTML

A simple HTML document is needed for hosting the styles, Markdown and the generated slides themselves:

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
/* Slideshow styles */
</style>
</head>
<body>
* <textarea id="source">
<!-- Slideshow Markdown -->
</textarea>
* <script type="text/javascript" src="remark.js
</script>
<script type="text/javascript">
* var slideshow = remark.create();
</script>
</body>
</html>
```

You may download remark to

Of course, Markdown can only go so far.

Markdown extensions

To help out with slide layout and formatting, a few Markdown extensions have been included:

- Slide properties, for naming, styling and templating slides
- Content classes, for styling specific content
- Syntax highlighting, supporting a range of languages

Markdown extensions

- Slide properties

Initial lines containing key-value pairs are extracted as slide properties:

```
name: agenda
class: middle, center
# Agenda
The name of this slide is {{ name }}.
```

Slide properties serve multiple purposes:

- Naming and styling slides using properties `name` and `class`
- Using slides as templates using properties `template` and `layout`
- Expansion of `{{ property }}`

Markdown extensions

- Slide properties

- Content classes

Any occurences of one or more dotted CSS class names followed by square brackets are replaced with the contents of the brackets with the specified classes applied:

```
.footnote[.red.bold[*] Important footnote]
```

Resulting HTML extract:

```
<span class="footnote">
<span class="red bold">*</span> Important footn
</span>
```

Markdown extensions

- Slide properties

- Content classes

Code blocks can be syntax highlighted by specifying a language from the set of supported languages. Using GFM fenced code blocks you can easily specify highlighting language

A number of highlighting styles are available, including several well-known themes from different editors and IDEs.

~~~
```javascript
function add(a, b){
return a + b
~~~

~~~
``````ruby
def add(a, b)
a + b
~~~

Markdown extensions

$$\int_\Omega \nabla v \cdot \nabla u - \lambda v u \, dV = \int_\Omega v f \, dV$$

- Slide properties

-

Content classes

Presenter mode

To help out with giving presentations, a presenter mode comprising the following features is provided:

- Display of slide notes for the current slide, to help you remember key points
- Display of upcoming slide, to let you know what's coming
- Cloning of slideshow for viewing on extended display

Presenter mode

- Inline notes

Just like three dashes separate slides, three question marks separate slide content from slide notes:

```
Slide 1 content
*???
Slide 1 notes
---
Slide 2 content
*???
Slide 2 notes
```

Slide notes are also treated as Markdown, and will be converted in the same manner slide content is. Pressing **P** will toggle presenter mode.

Presenter mode

- Inline notes

- Cloned view

Presenter mode of course makes no sense to the audience. Creating a cloned view of your slideshow lets you:

- Move the cloned view to the extended display visible to the audience
- Put the original slideshow in presenter mode
- Navigate as usual, and the cloned view will automatically keep up with the original Pressing **C** will open a cloned view of the current slideshow in a new browser window.

It's time to get started!

Getting started

Getting up and running is done in only a few steps:

1. Go to your gist and write some markdown.
2. put a "deck" right after "gist" in url like this:

```
          deck
            v
https://gist.github.com/jcouyang/8acfc555a718d62b77b2
            |
            V
https://gistdeck.github.com/jcouyang/8acfc555a718d62b77b2
```

3. enjoy the slide remark generated. For more information on using remark, please check out the wiki pages.

That's all folks (for now)!

Slideshow created using remark and host on gistdeck.