

CodeGolf

Create something
in 1024 characters

Create something
in 140 characters

Create something

Constraints

stimulate

Creativity



~/Code/rubysauna

I

https://www.youtube.com/watch?v=Ugz_oOWOzlM&feature=youtu.be

```
(f=proc{|x,y|sleep 1;puts
"\033c",x;f[x.gsub(/\\\\
\\.,'~/\\\')].gsub(/.o./,y ??
'\o/:'|o|),!y]})[ '~/\\
\+'+'~'*12+'0~~']
```

116 characters

...but what do

they mean?

☺ cat lucas.md

Lucas Dohmen

- * Consultant @ innoQ GmbH
- * From Cologne, Germany
- * Open Source Developer
- * Podcaster
- * CoderDojo Cologne Organizer
- * Code Golfer

```
(f=proc{|x,y|sleep 1;puts
"\033c",x;f[x.gsub(/\\\\
\\.,'~/\\\')].gsub(/.o./,y ??
'\o/:'|o|),!y]})[ '~/\\
\+'+'~'*12+'0~~']
```

Recursion using a Proc

```
(f=proc{ |x,y|  
    sleep 1  
    system 'clear'  
    puts x  
    f[...]  
} )[...]
```

Calling a Proc

```
>> f = proc { |a,b| puts a }
=> #<Proc:0x007fa918b3c7c0@(irb):1>
>> f['hello']
hello
=> nil
```

Calling a Proc

```
>> f = proc { |a,b| puts a }  
=> #<Proc:0x007fa918b3c7c0@(irb):1>  
>> f['hello']  
hello  
=> nil
```

**Arguments that were
not provided are initialised as nil
(not the case for a lambda)**

Calling a Proc

JS

```
>> f = proc { |a,b| puts a }  
=> #<Proc:0x007fa918b3c7c0@(irb):1>  
>> f['hello']  
hello  
=> nil
```

**Arguments that were
not provided are initialised as nil
(not the case for a lambda)**

First time we call it...

```
'~/\\' + '~'*12 + 'o~~'
```

First time we call it...

```
'~/\\" + ' ~'*12 + '0~~'
```

```
=> ' ~/\~~~~~0~~~'
```

**On every call from now on
we will do two things**

- 1. Move the shark**
- 2. Change the arms**

Moving the shark

```
x.gsub(/\//,\.\./, '~/\\"')
```

Moving the shark

```
x.gsub(/\\\\./, '~\\\\')
```

Moving the shark

```
x.gsub(/\\\\./, '~\\\\')
```

The diagram illustrates the movement of a shark character ('~') within a regular expression pattern. The pattern consists of three backslashes (\\) followed by a dot (.) followed by another backslash (\). Two red arrows point upwards from the bottom of the slide towards the dot character. A green arrow points upwards from the bottom of the slide towards the second backslash after the dot. Below the pattern, two red double slashes (\\) are shown, indicating the starting and ending positions of the matching group.

Moving the shark

```
x.gsub(/\//,\.\./, '~/\\"')
```

Moving the shark

```
x.gsub(/\\\\.\/, '~/\\"')
```

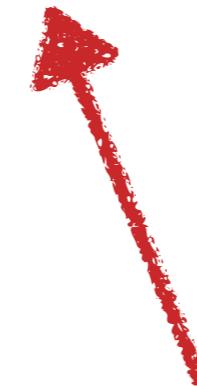


remove one character

from the right

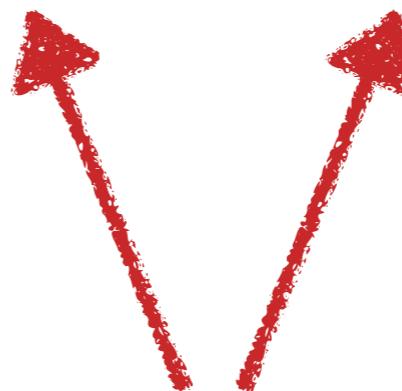
Moving the shark

```
x.gsub(/\//,\~,'~/')
```



Moving the shark

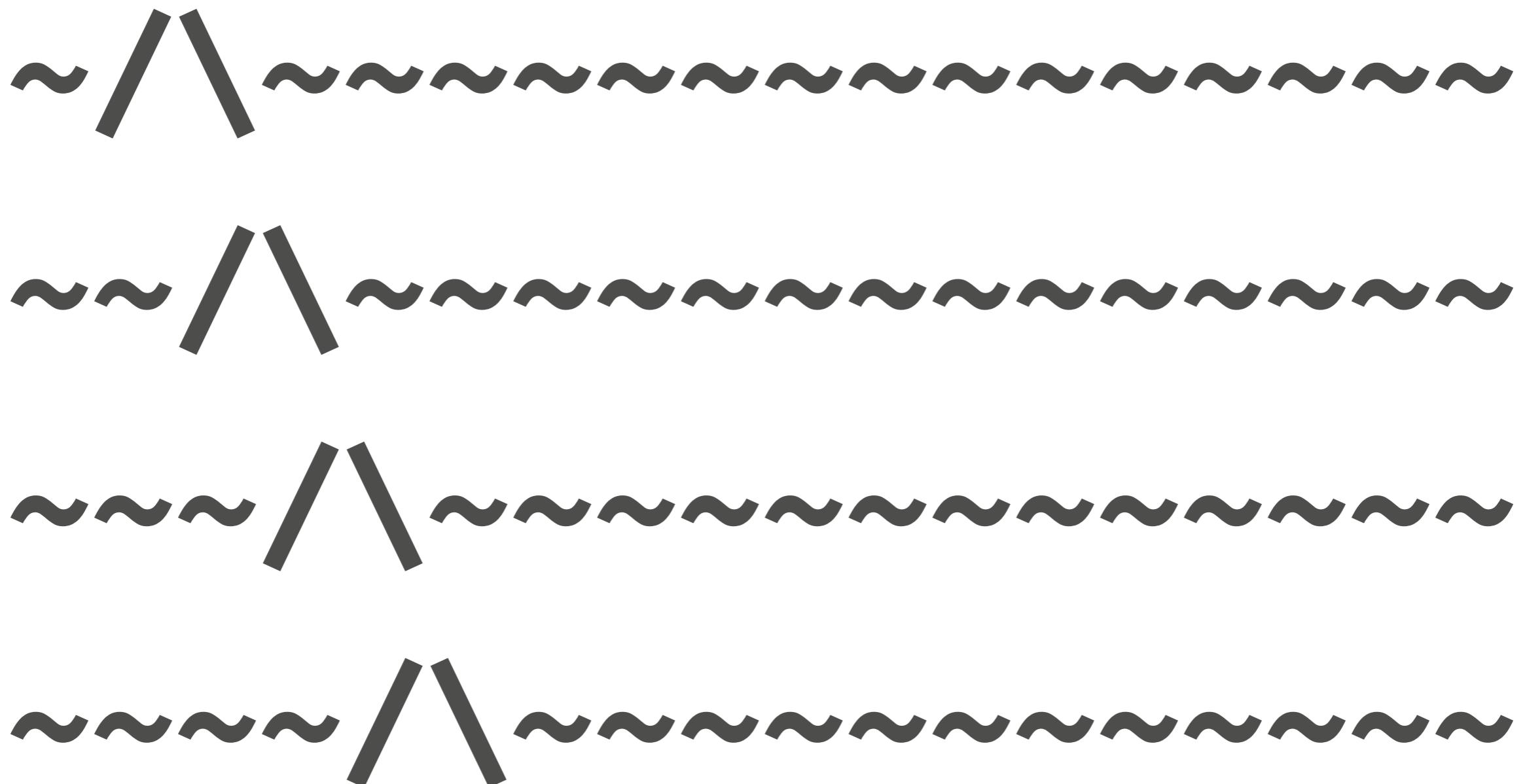
```
x.gsub(/\\\\.\/, '~/\\"')
```



add one wave

to the left

Moving the shark



Changing the arms

```
.gsub(/.o./, y ? '\o/' : '|o|')
```

Changing the arms

```
.gsub(/.o./, y ? '\o/' : '|o|')
```

matches all stages

~0~ \o/ |o|

of the animation

Changing the arms

```
.gsub(/.o./, y ? '\o/' : '|o|')
```

depending on y

we change it to

either \o/ or |o|

Changing the arms

```
(f=proc{|x,y|  
    sleep 1  
    system 'clear'  
    puts x  
    f[...,!y]  
} )[...]
```

```
(f=proc{|x,y|sleep 1;puts
"\033c",x;f[x.gsub(/\\\\
\\.,'~/\\\')].gsub(/.o./,y ??
'\o/:'|o|),!y]})[ '~/\\
\+'+'~'*12+'0~~']
```



~/Code/rubysauna

{

<https://www.youtube.com/watch?v=d68x-Rl-yqw&feature=youtu.be>

This app has one main trick:

The list is 1-indexed

If you enter a line
it will be stored as

1. a string
2. a fixnum

If the fixnum is 0

it is a todo item

otherwise

it is a check-off action

'Clean the house'.to_i #=> 0

'12'.to_i #=> 12

'0'.to_i #=> 0

```
'Clean the house'.to_i #=> 0
```

```
'12'.to_i #=> 12
```

```
'0'.to_i #=> 0
```

There is no item 0,

does not matter

```
todo_list = []

loop do
    input_as_string = gets
    input_as_int = input_as_string.to_i
    system "clear"

    if input_as_int > 0
        todo_list.delete_at(input_as_int - 1)
    else
        todo_list << input_as_string
    end

    todo_list.each_with_index do |item, index|
        puts "#{index+1}: #{item}"
    end
end
```

Ok, that's pretty

short.

Now let's golf it.

```
input_as_string=gets;input_as_int=input_as_string.to_i  
input_as_int=(input_as_string=gets).to_i
```

```
input_as_string=gets;input_as_int=gets;input_as_string.to_i  
input_as_in
```



Use
ternary if/else
instead of
if/else

```
todo_list.each_with_index do |item, index|
  puts "#{index+1}: #{item}"
end
```

```
todo_list.each_with_index do |item, index|
  puts "#{index+1}: #{item}"
end
```

Way too long...

```
todo_list.each_with_index{|item, index| puts "#{index+1}: #{item}"}
```

```
todo_list.each_with_index{|item, index| puts "#{index+1}: #{item}"}
index=0; todo_list.each{|item| puts "#{index+=1}: #{item}"}
```

```
todo_list.each_with_index{|item, index| puts "#{index+1}: #{item}"}
index=0; todo_list.each{|item| puts "#{index+=1}: #{item}"}
index=0; todo_list.map{|item| puts "#{index+=1}: #{item}"}
```

```
todo_list.each{|item| puts "#{$index+1}: #{item}"}
index=0;todo_list.map{|item| puts "#{$index+1}: #{item}"}
index=0;todo_list.map{|item| puts "#{$index+1}: #{item}"}
```

```
todo_list = []
loop do
  input_as_int=(input_as_string=gets).to_i
  system "clear"
  input_as_int>0?todo_list.delete_at(input_as_int-1):todo_list<<input_as_string
  index=0
  todo_list.map{|item|puts "#{index+=1}: #{item}"}
end
```

Replace names & save whitespace

```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
system "clear"  
i>0?t.delete_at(i-1):t<<s  
t.map{|g|puts "#{$j+=1}: #{g}"}}
```

103 characters

bonus round

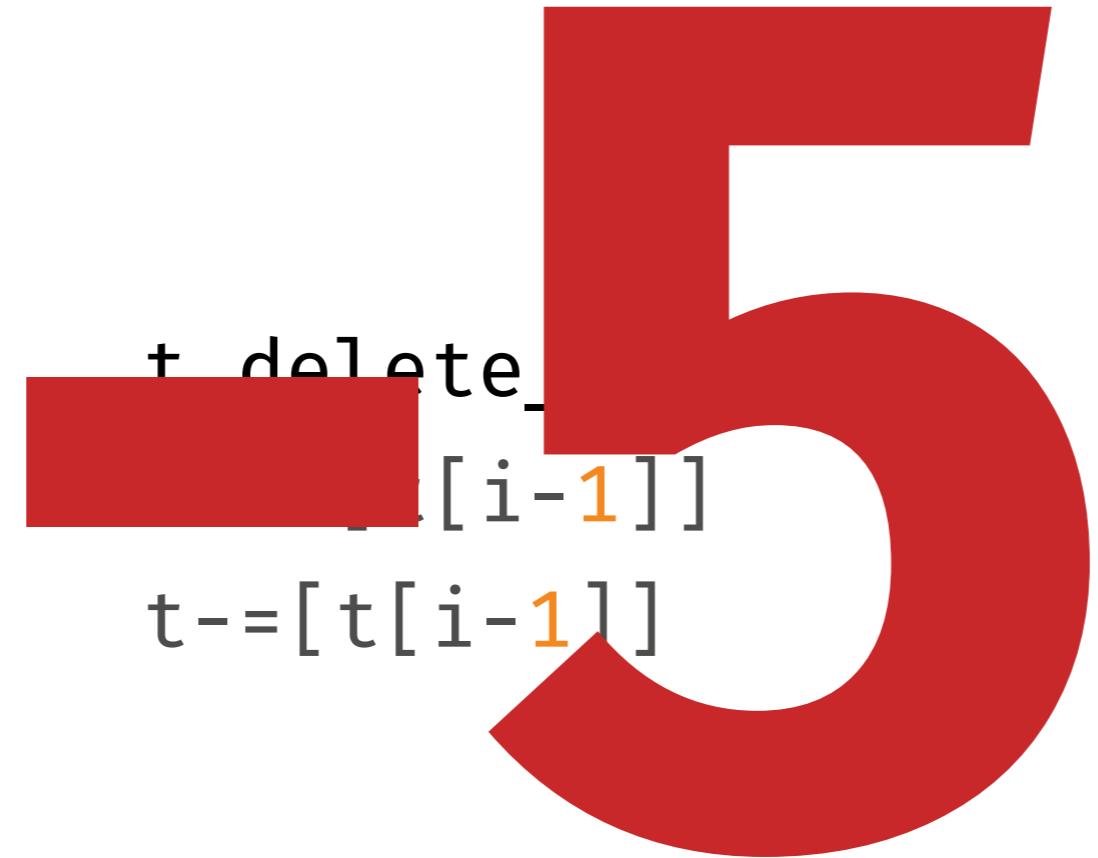
delete_at is way too long

```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
system "clear"  
i>0?t.delete_at(i-1):t<<s  
t.map{|g|puts "#{$j+=1}: #{g}"}}
```

`t.delete_at(i-1)`

`t=t-[t[i-1]]`

`t-= [t[i-1]]`



```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
system "clear"  
i>0?t-[t[i-1]]:t<<s  
t.map{|g|puts "#{$j+=1}: #{$g}"}}
```



```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
    system "clear"  
    i>0?t-=t[i-1]:t<<s  
    t.map{|g|puts "#{$j+=1}: #{$g}"}}
```

```
system "clear"  
puts "\033c"
```



puts "cle
"\\03

```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
puts "\033c"  
i>0?t-[t[i-1]]:t<<s  
t.map{|g|puts "#{j+=1}: #{g}"}}
```

```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
  puts "\033c"  
  i>0?t-[t[i-1]]:t<<s  
  t.map{|g| puts "#{$j+=1}: #{$g}"}}
```

DRY

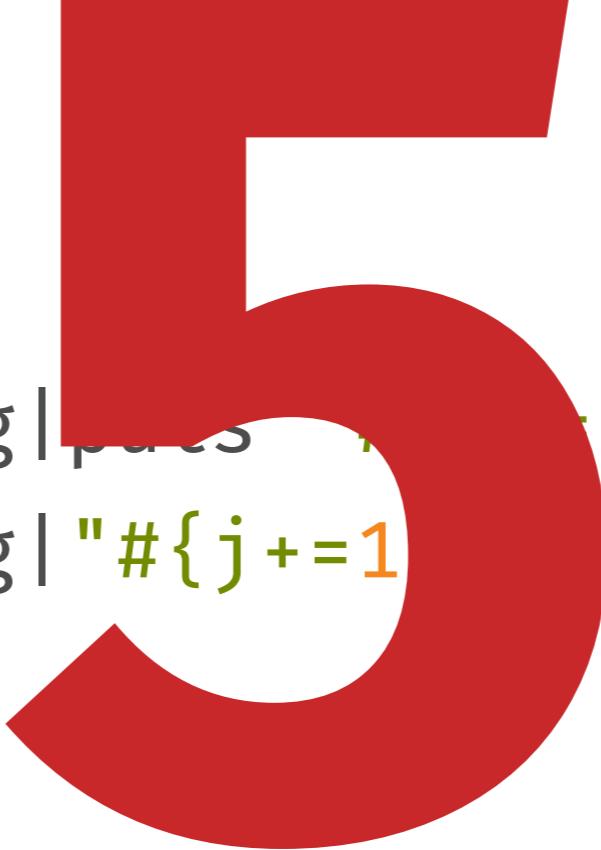
DRY



```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
puts "\033c"  
i>0?t-[t[i-1]]:t<<s  
t.map{|g|puts "#{j+=1}: #{g}"}}
```

```
t=[ ]  
loop{i,j=(s=gets).to_i,0  
i>0?t-[t[i-1]]:t<<s  
puts "\033c"  
t.map{|g|puts "#{$j+=1}: #{$g}"}}
```

```
puts "\033c"; t.map{|g| puts "#{j+=1}: #{g}"}  
puts "\033c", t.map{|g| "#{j+=1}: #{g}"}
```



```
puts "\033c" + t.map{|g| puts g + 1}: #{g}"}
puts "\033c", t.map{|g| "#{j+=1}" + g}: #{g}"}
```

```
t=[]
loop{i,j=(s=gets).to_i,0
i>0?t-[t[i-1]]:t<<s
puts "\033c",t.map{|g| "#{j+=1}: #{g}"}}
```

A close-up photograph of a man with light brown hair, wearing round-rimmed glasses and a well-groomed, dark brown mustache. He is wearing a light gray t-shirt and is looking towards the left of the frame with a neutral expression. In the upper-left corner of the image, there is a white rectangular overlay containing a large red dollar sign (\$) and a red minus sign (-) positioned below it, suggesting a cost or deduction.

\$ -

```
t=[]
loop{i,j=gets.to_i,0
i>0?t-[t[i-1]]:t<<$_
puts "\033c",t.map{|g| "#{j+=1}: #{g}"}}
```

t=[]
loop{i,j=getchar(),0
i>0?t-[t[i-1]]:t<<\$_
puts "\033c",t.map{|g|g.chr|=1}: #{g}" } }

3

```
t=[]
loop{i,j=gets.to_i,0
i>0?t-[t[i-1]]:t<<$_
puts "\033c",t.map{|g| "#{j+=1}: #{g}"}}
```

88 characters

88 characters?

Now we have
space again.

88 characters?

Now we have

space again.

A persistent todo app

in 129 characters.

```
loop{t=IO.readlines(?a)rescue[]
j=0
puts "\033c", t.map{|g| "#{$j+=1}: #{g}"}
i=gets.to_i
i>0?t-=[t[i-1]]:t<<$_
IO.write ?a,t.join}
```

```
loop{t=IO.readlines(a)rescue  
j=0  
puts " "+a.map{|g|t[j+=1]}+{g}" }  
i=gets.to_i  
i>0?t=[t[i-1]]:t<<$_  
IO.write ?a,t.join}
```



~/Code/golf-course/ruby

[]

<https://www.youtube.com/watch?v=q5kSdZ7Rb3A&feature=youtu.be>

Thank you

@moonbeamlabs