

Urbit for Developers

Assembly 2021

~lagrev-nocfep • @sigilante • Neal Davis

“If you find your life tangled up with somebody else’s life for no very logical reasons,” writes Bokonon, “that person may be a member of your *karass*. ”

—Kurt Vonnegut, *Cat’s Cradle*

OKAY!



you are:

- done
- ready to move on
- bored

NOPE!



you are:

- stuck
- in need of help
- lost

PollEv. com/nealdavis717

```
# Python
def fibonacci(n):
    i = 0
    p = 0
    q = 1
    r = []
    while (i < n):
        i = i + 1
        po = p
        p = q
        q = po + q
        r.append(q)
    r.reverse()
    return r
```

```
# Python
def fibonacci(n):
    i = 0
    p = 0
    q = 1
    r = []
    while (i < n):
        i = i + 1
        po = p
        p = q
        q = po + q
        r.append(q)
    r.reverse()
    return r
```

:: “Aural” Hoon Pseudocode

```
gate n=uint {
    let [i=0 p=0 q=1 \
          r=[list uint]]
    do {
        if (i == n) return r
    } loop {
        i ← i+1
        p ← q      q ← (p + q)
        r ← [q] + r
    }
    match-type r
    apply flop
}
```

```
# Actual Hoon
|= n=@ud
%- flop
=+ [i=0 p=0 q=1 \
     r=*list @ud]
|- ^+ r
?: =[i n] r
%=$
    i +(i)
    p q
    q [add p q]
    r [q r]
==
```

```
:: “Aural” Hoon Pseudocode
gate n:uint {
    let [i=0 p=0 q=1 \
         r=[list uint]]
    do {
        if [i == n] return r
    } loop {
        i ← i+1
        p ← q      q ← (p + q)
        r ← [q] + r
    }
    match-type r
    apply flop
}
```

Fakezod Setup

- 1 Navigate to a new directory /home/username/urbit
- 2 Download the pill, or pre-packaged Urbit OS configuration:
`wget https://bootstrap.urbit.org/urbit-v1.6.pill`
- 3 Start a new fakezod in this directory with
`urbit -F zod -B urbit-v1.6.pill`
- 4 Mount the filesystem: | mount %home
- 5 Press Ctrl+D to exit the Urbit session.
- 6 Copy the fakezod to a local backup: `cp -r zod zod-ist-tot`

Fakezod Development

1 Save the file as urbit/gen/fib.hoon

2 Tell the fakezod to update from Unix:

 | commit %home

3 Run the generator in the Dojo:

 +fib 15

:: Original

```
| = n=@ud
%- flop
=+ [i=0 p=0 q=1 r=*list @ud]
|- ^+ r
?: = [i n] r
%=
$ i +[i]
p q
q [add p q]
r [q r]
==
```

:: Expanded runes & children →

```
| =
n=@ud
%- flop
=+
[i=0 p=0 q=1 r=*list @ud]
|- ^+
r
?: = [i n]
r
%=
$ i +[i]
p q
q [add p q]
r [q r]
==
```

Exercise

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

Exercise

Try the following auras:

`@ud`0x1001.1111

`@p`0b1111.0000.1111.0000

`@ub`0x1001.1111

`@ta`'hello'

`@ux`0x1001.1111

`@ud`'hello'

`@p`0x1001.1111

`@ux`'hello'

`@ud`.1

`@uc`'hello'

`@ux`.1

`@sb`'hello'

`@ub`.1

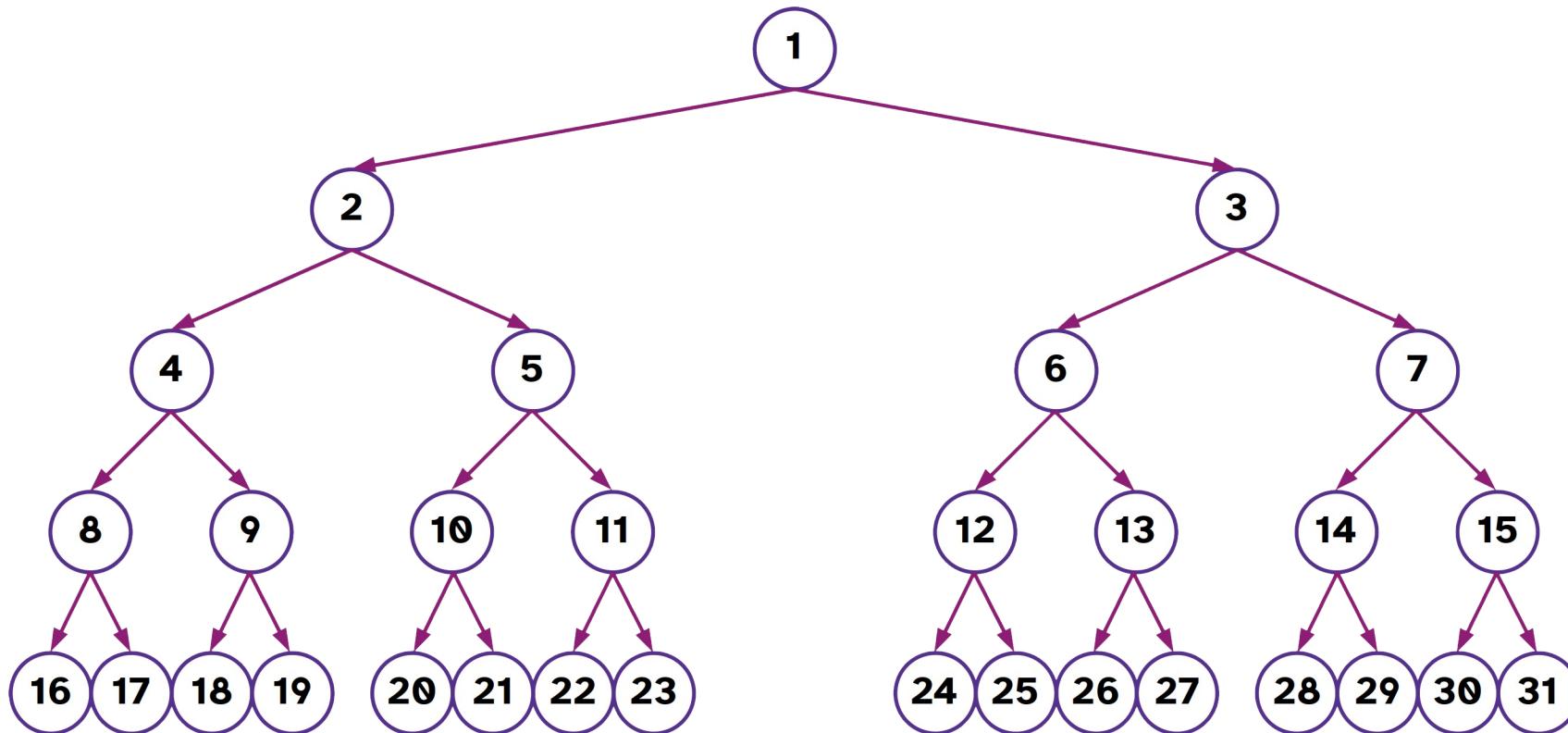
`@rs`'hello'

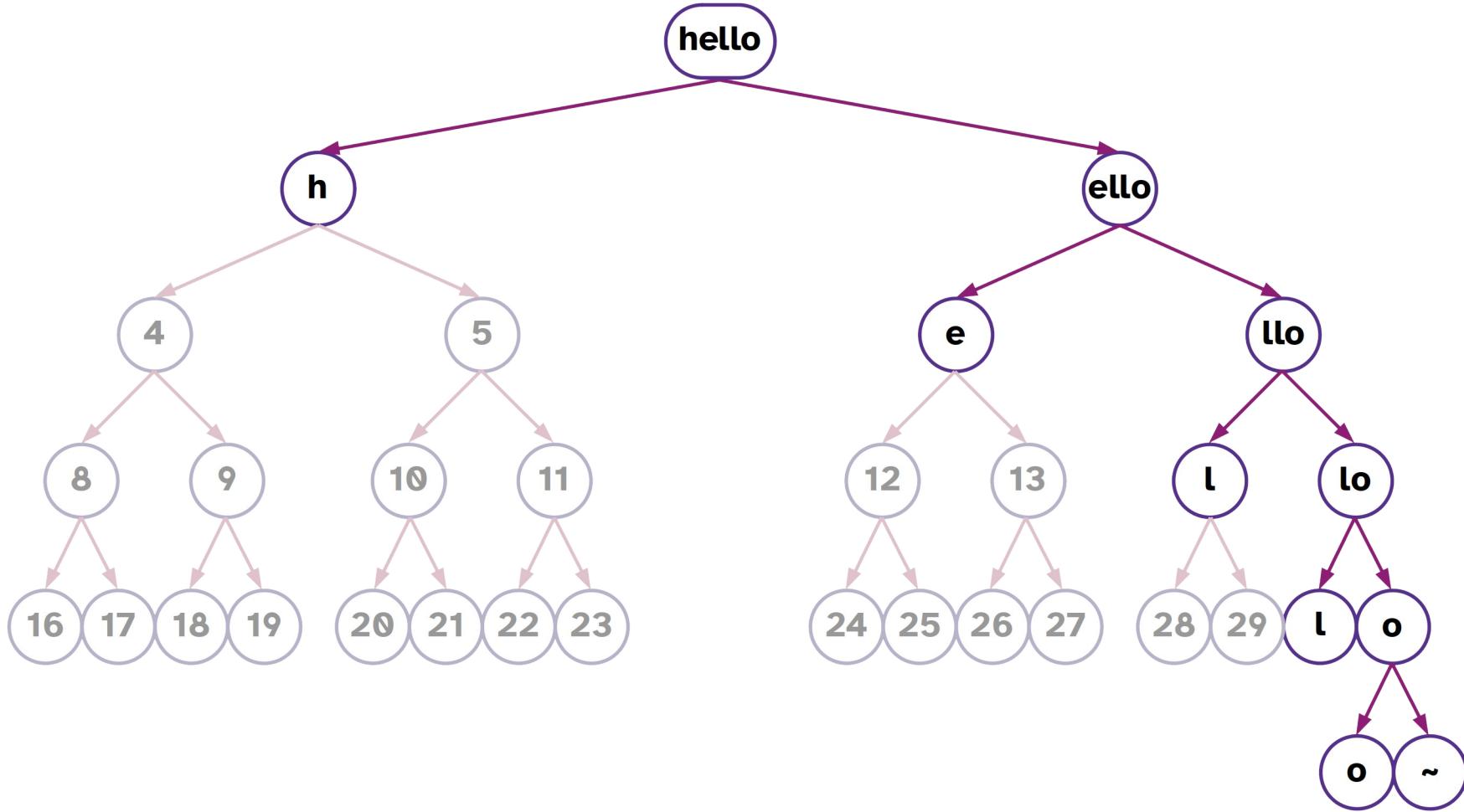
`@sb`.1



Arms
do things.

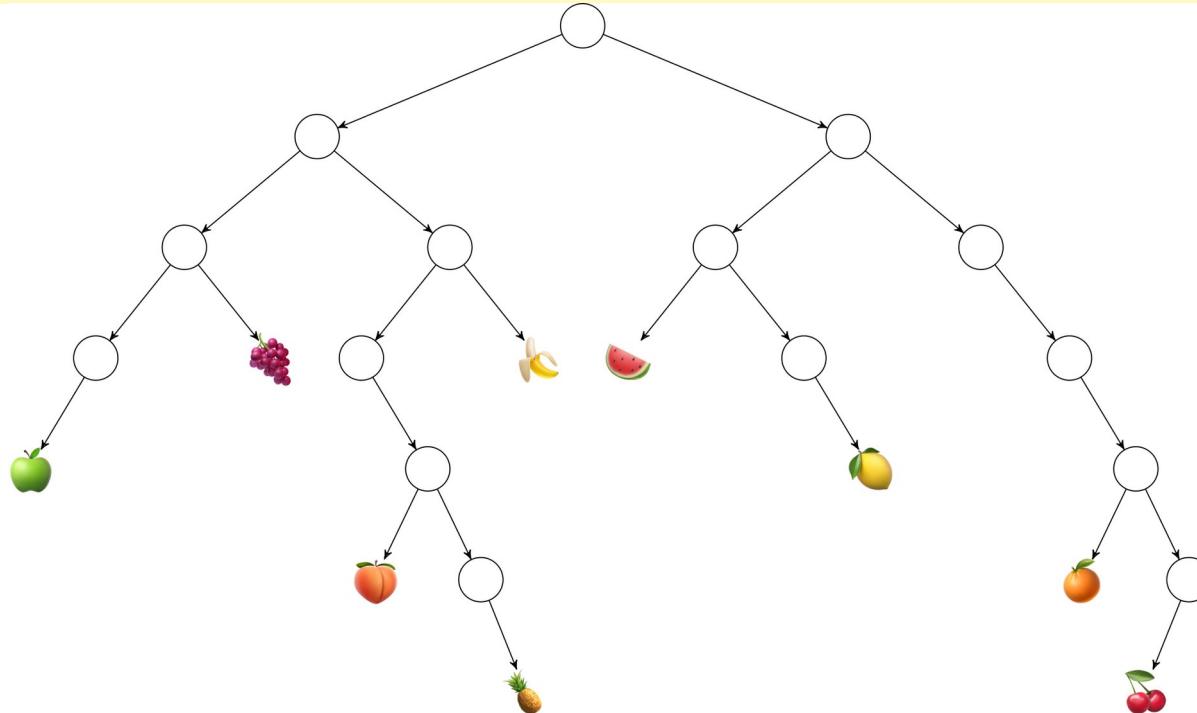
Legs
hold
data.





Exercise

Produce a cell representation of the fruit tree.
Use `~` as a placeholder for an empty node.



Exercise

Implement the Sudan function in Hoon.

$$F_0(x, y) = x + y$$

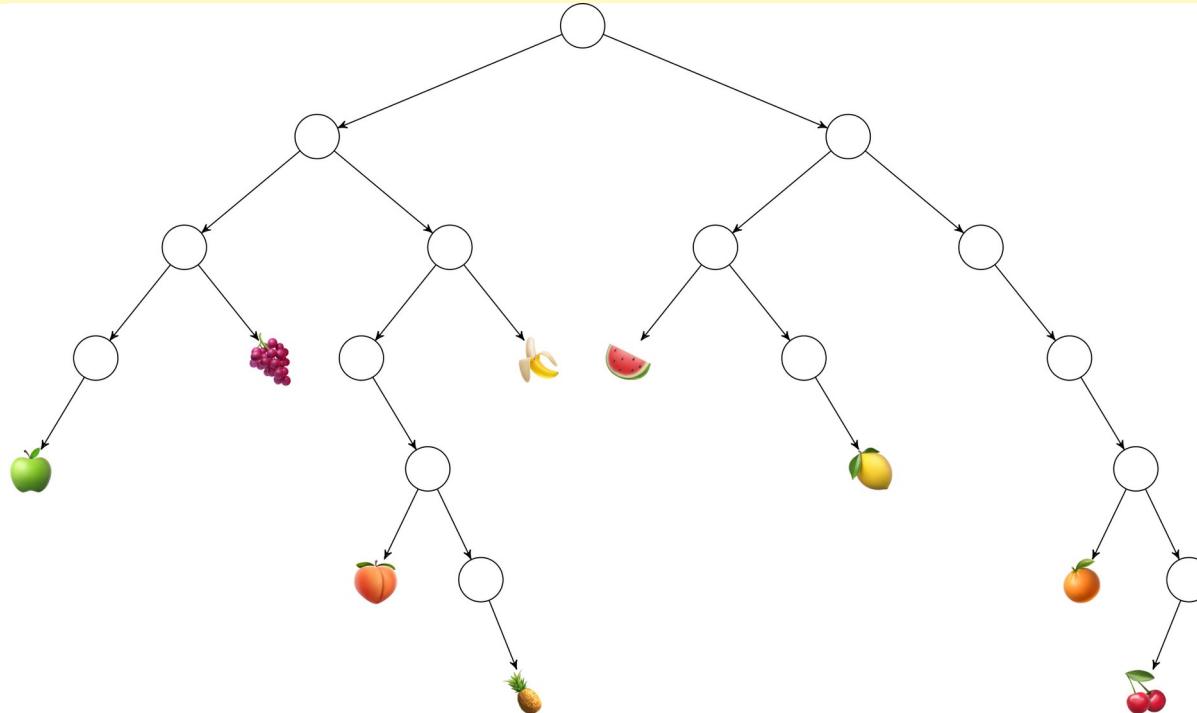
$$F_{n+1}(x, 0) = x \quad \text{if } n \geq 0$$

$$F_{n+1}(x, y + 1) = F_n(F_{n+1}(x, y), F_{n+1}(x, y) + y + 1) \quad \text{if } n \geq 0$$

(Check Wikipedia “Sudan Function” for trial cases.)

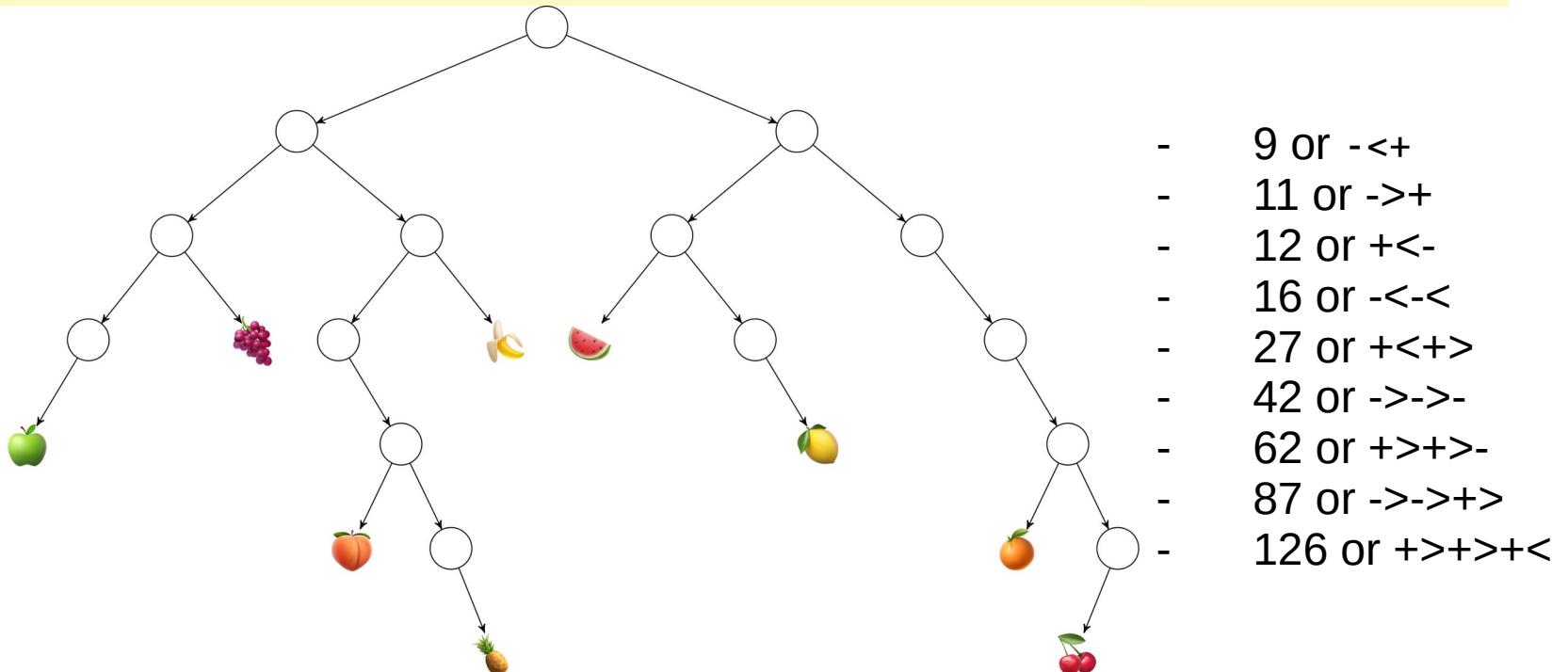
Exercise

Produce the numeric and lark-notated equivalent addresses for each of the fruit nodes in the binary fruit tree.



Solution

Produce the numeric and lark-notated equivalent addresses for each of the fruit nodes in the binary fruit tree.



Exercise

Write a `%say` generator which simulates scoring a simple dice throw of n six-sided dice. That is, it should return the sum of n dice. If no number is specified, then only one die roll should be returned.

Exercise

Here is a %say generator that returns a list of n probabilities between 0-100%.

```
: -  %say
|=  [[* eny=@uv *] [n=@ud ~] ~]
: -  %noun
=/  values `list @ud`~
=/  count 0
=/  rng ~{. og eny}
|-  ^- [list @ud]
?:  =[count n] values
=^  r  rng [rads:rng 100]          :: preserve state change
${count +[count], values [weld values ~[(add r 1)]]}
```

Exercise

The Sieve of Eratosthenes is a classic (if relatively inefficient) way to produce a list of prime numbers. Save this as a file `/gen/primes.hoon`, sync it, and run it as `+primes 100`

(Be careful not to use too large a number—use Ctrl+C to interrupt evaluation!)

Exercise

Scan through the generators in /gen and see which libraries are used and how they are imported.

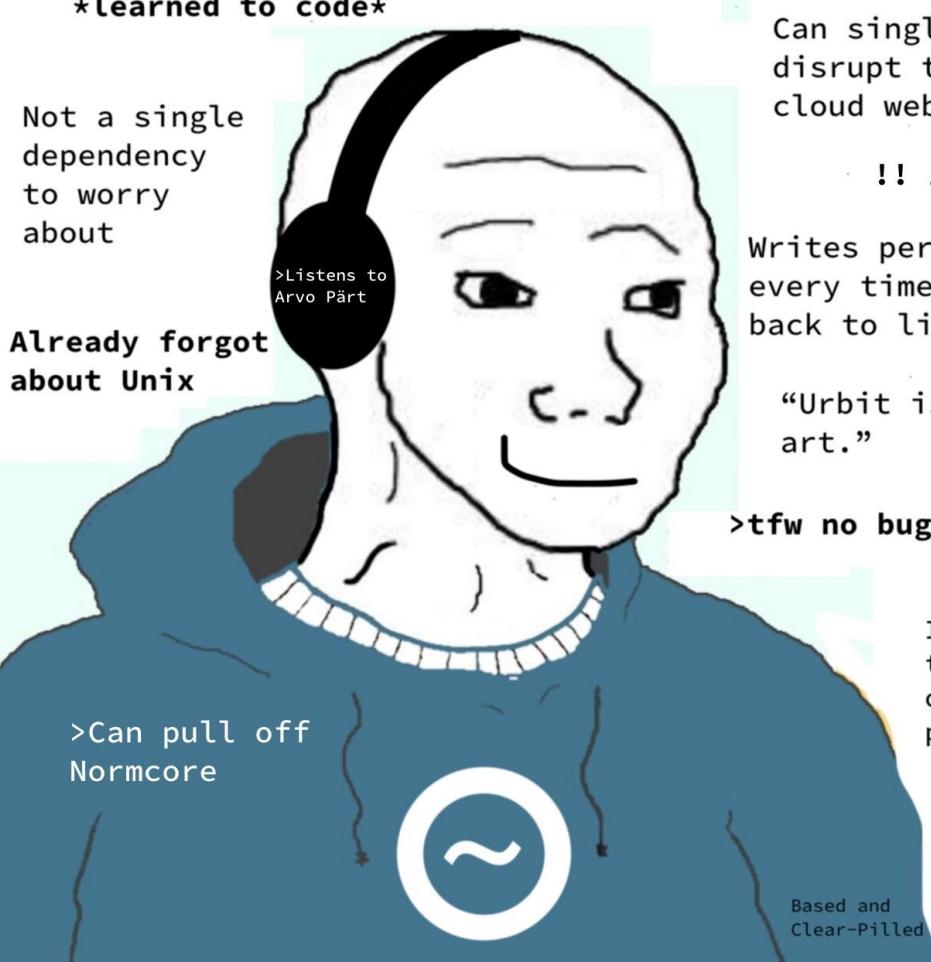
Exercise

Examine several source code files. Enumerate some principles of good Hoon style that you infer from these.

```
{  
  "name": "John",  
  "age": 30,  
  "car": null  
}
```

```
[ ~  
  [%o p={  
    [p=' car'  
     q=~  
   ]  
   [p=' name'  
     q=[%s p=' John' ]  
   ]  
   [p=' age'  
     q=[%n p=~. 30 ]  
   ]  
 }]  
 ]
```

~the.6-week.hooner



LAWFUL SOCIAL NEUTRAL REBEL CHAOTIC

GOOD

MORAL

NEUTRAL

IMPURE

EVIL

C H O O S E
Y O U R
F I G H T E R

LAWFUL SOCIAL NEUTRAL REBEL CHAOTIC

LAWFUL SOCIAL NEUTRAL REBEL CHAOTIC

GOOD

MORAL

NEUTRAL

IMPURE

EVIL



SQL



Excel



Assembly



Perl



MATLAB

C H O O S E
Y O U R
F I G H T E R

LAWFUL SOCIAL NEUTRAL REBEL CHAOTIC

LAWFUL

SOCIAL

NEUTRAL

REBEL

CHAOTIC

GOOD

MORAL

NEUTRAL

IMPURE

EVIL

C H O O S E
Y O U R
F I G H T E R



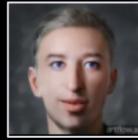
SQL



Excel



Assembly



Perl



MATLAB



C



COBOL



Bash



Visual Basic



PHP

LAWFUL

SOCIAL

NEUTRAL

REBEL

CHAOTIC

LAWFUL

SOCIAL

NEUTRAL

REBEL

CHAOTIC

GOOD

MORAL

NEUTRAL

IMPURE

EVIL

C H O O S E
Y O U R
F I G H T E R



SQL



Excel



Assembly



Perl



MATLAB



C



COBOL



Bash



Visual Basic



PHP



Java



C++



Javascript



Brainfuck



HTML/CSS

LAWFUL

SOCIAL

NEUTRAL

REBEL

CHAOTIC

LAWFUL

SOCIAL

NEUTRAL

REBEL

CHAOTIC

GOOD

MORAL

NEUTRAL

IMPURE

EVIL



Haskell



Python



Ruby



Lua



Fortran



SQL



Excel



Assembly



Perl



MATLAB



C



COBOL



Bash



Visual Basic



PHP



Java



C++



Javascript



Brainfuck



HTML/CSS

LAWFUL

SOCIAL

NEUTRAL

REBEL

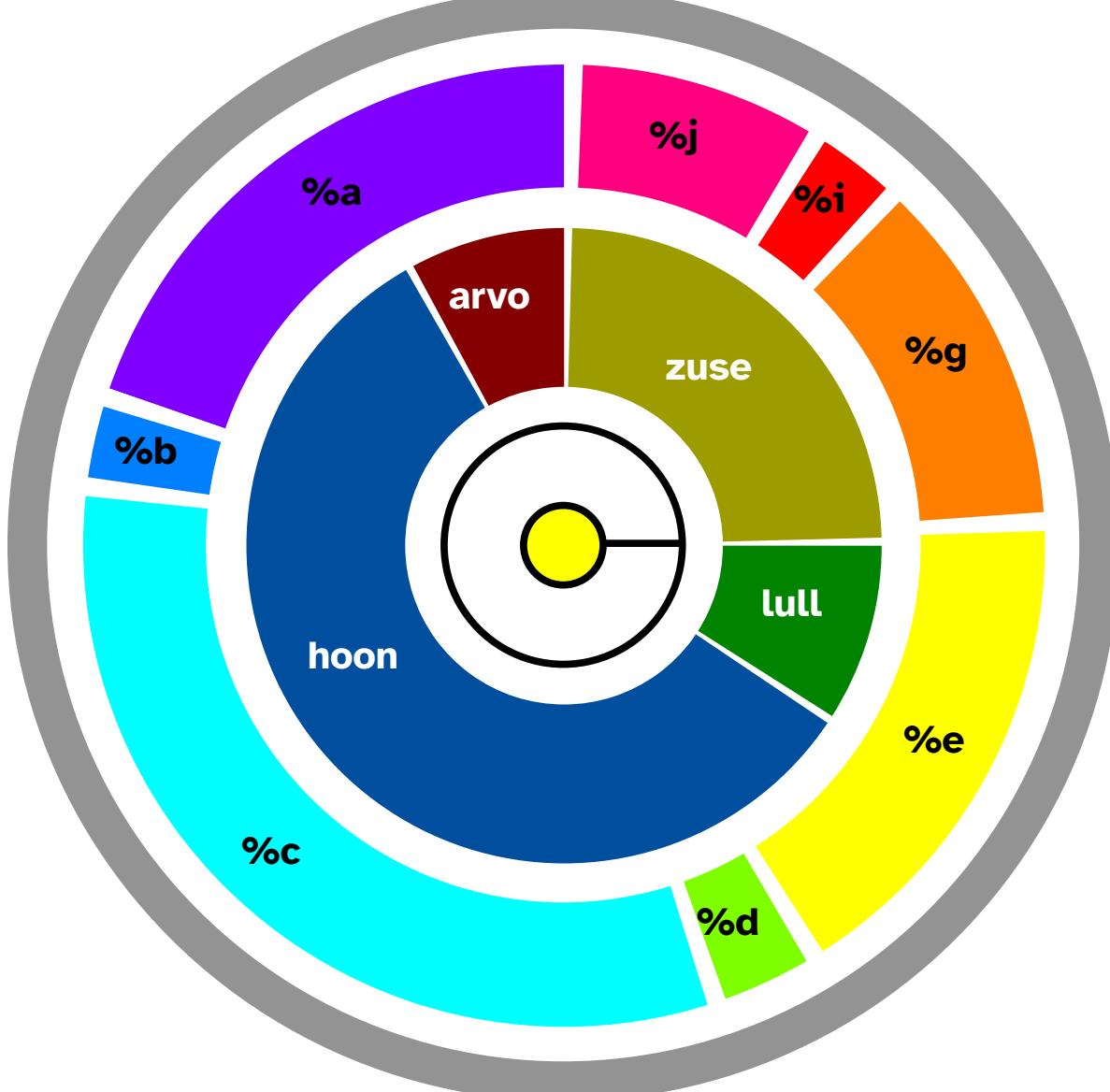
CHAOTIC

	LAWFUL	SOCIAL	NEUTRAL	REBEL	CHAOTIC	
GOOD						
MORAL						
NEUTRAL						
IMPURE						
EVIL						
	LAWFUL	SOCIAL	NEUTRAL	REBEL	CHAOTIC	

	LAWFUL	SOCIAL	NEUTRAL	REBEL	CHAOTIC	
GOOD						Hoon
MORAL						C H O O S E Y O U R F I G H T E R
NEUTRAL						
IMPURE						
EVIL						
	LAWFUL	SOCIAL	NEUTRAL	REBEL	CHAOTIC	

PollEv.com/nealdavis717





```
+$ href :: where a tile links
$% [%glob base=term =glob-location] :: location of client-side data
    [%site =path] :: location of server-rendered frontend
== ::

:: ::

+$ url   cord :: URL type
:: ::

+$ glob-location :: how to retrieve glob (client-side)
$% [%http =url] :: HTTP source, if applicable
    [%ames =ship] :: %ames source, if applicable
== ::

:: ::

+$ version :: version of app (not Kelvin version)
[major=@ud minor=@ud patch=@ud]
:: ::

+$ docket ::

$: %1 :: Docket protocol tag
    title=@t :: text on home screen
    info=@t :: long-form description
    color=@ux :: app tile color
    =href :: link to client bundle
    image=[unit url] :: app tile background
    =version :: version of app (not Kelvin version)
    website=url :: URL to open on click
    license=cord :: software release license
== ::
```

- %alpha basic demo of Gall as MWE
- %bravo add local operations
- %charlie add peer-to-peer operations
- %delta write complete basic app

Exercise

- Examine /sur/dns.hoon (basic) or /sur/bitcoin.hoon (advanced).
- Examine /mar/json.hoon
(and discuss with your *karass*)

|_ =bowl: gall
++ on-init
++ on-save
++ on-load
++ on-arvo
++ on-peek
++ on-poke
++ on-watch
++ on-leave
++ on-agent
++ on-fail
--

|_ =bowl: gall
++ on-init
++ on-save
++ on-load
++ on-arvo
++ on-peek
++ on-poke
++ on-watch
++ on-leave
++ on-agent
++ on-fail
--

| _ =bowl: gall

++ on-init

++ on-save

++ on-load

++ on-arvo

++ on-peek

++ on-poke

++ on-watch

++ on-leave

++ on-agent

++ on-fail

- -

|_ =bowl: gall

++ on-init

++ on-save

++ on-load

++ on-arvo

++ on-peek

++ on-poke

++ on-watch

++ on-leave

++ on-agent

++ on-fail

- -

Permissions Model

- A *store* is a local database.
- A *hook* is a permissions broker for the database.
- A *view* is a data aggregator which parses objects for external clients such as Landscape.



©Columbia/TriStar

**Please Stand by!
We'll Be Back in a
Jingleheimer Jiffy**

