

An Approach to Developing on Urbit

Urbit

# **An Approach to Developing on Urbit**

**Customise this page according to your needs**

N E Davis \*

April 5, 2021

Malancandra & Sons

\* University of Illinois

## An Approach to Developing on Urbit

**Copyright ©2021 by N E Davis**

### **Colophon**

This document was typeset with the help of KOMA-Script and L<sup>A</sup>T<sub>E</sub>X using the kaobook class.

The source code of this book is available at:

<https://github.com/davis68/urbit-textbook>

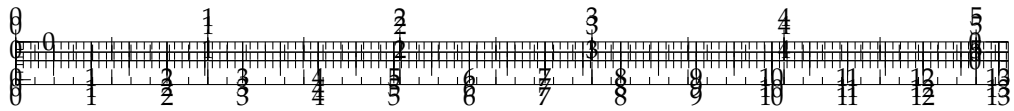
### **Publisher**

First printed in July 2022 by Malancandra & Sons

Lights All Askew in the Heavens.  
Stars Not Where They Seemed or Were Calculated to Be.  
A BOOK FOR 12 WISE MEN.  
No More in All the World Could Comprehend It.

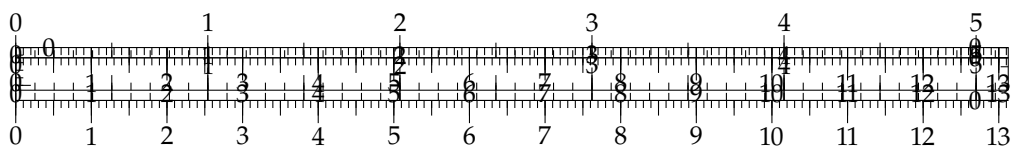
– *The New York Times*, November 19, 1919

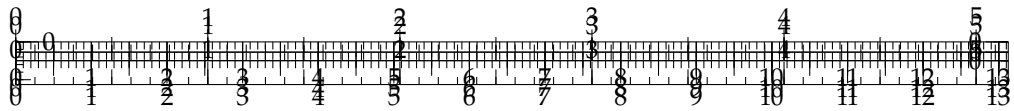




# Contents

<b>Contents</b>	<b>v</b>
<b>1 A Brief Introduction</b>	<b>1</b>
1.1 Why Urbit Matters . . . . .	1
1.2 Azimuth, the Urbit Address Space . . . . .	1
1.3 Accessing Urbit . . . . .	2
1.4 Developing for Urbit . . . . .	2
 <b>LANGUAGE ESSENTIALS</b>	 <b>4</b>
<b>2 Nock, A Combinator Language</b>	<b>5</b>
2.1 Primitive rules and the combinator calculus . . . . .	5
2.2 Compound rules . . . . .	5
2.3 Kelvin versioning . . . . .	5
 <b>3 Elements of Hoon</b>	 <b>6</b>
3.1 Reading the Runes . . . . .	6
3.2 Irregular Forms . . . . .	6
3.3 Nouns . . . . .	6
3.4 Hoon as Nock Macro . . . . .	6
3.5 Key Data Structures . . . . .	6
Cores, Gates, Doors . . . . .	6
Molds . . . . .	6
Maps, Sets, Tree . . . . .	6
3.6 Generators . . . . .	6
Naked Generators . . . . .	6
%say generators . . . . .	6
%ask generators . . . . .	6
3.7 Libraries . . . . .	6
3.8 Unit Tests . . . . .	6
3.9 Building Code . . . . .	6
 <b>4 Advanced Hoon</b>	 <b>7</b>
4.1 Cores . . . . .	7
Variadicity . . . . .	7
Genericity . . . . .	7





4.2	Molds . . . . .	7
	Polymorphism . . . . .	7
4.3	Rune Families . . . . .	7
4.4	Marks and Structures . . . . .	7
4.5	Helpful Tools . . . . .	7
4.6	Deep Dives . . . . .	7
	Text Stream Parsing . . . . .	7
	JSON Parsing . . . . .	7
	HTML/XML Parsing . . . . .	7

## **SYSTEM DEVELOPMENT 8**

### **5 The Kernel 9**

5.1	Arvo . . . . .	9
	%zuse and %lull . . . . .	9
5.2	%ames, A Network . . . . .	9
5.3	%behn, A Timer . . . . .	9
5.4	%clay, A File System . . . . .	9
	++ford, A Build System . . . . .	9
	Scrying . . . . .	9
	Marks and conversions . . . . .	9
5.5	%dill, A Terminal driver . . . . .	9
5.6	%eyre and %iris, Server and Client Vanes . . . . .	9
5.7	%jael, Secretkeeper . . . . .	9
5.8	Azimuth, Address Space Management . . . . .	9
5.9	The Hoon Parser . . . . .	9

### **6 Userspace 10**

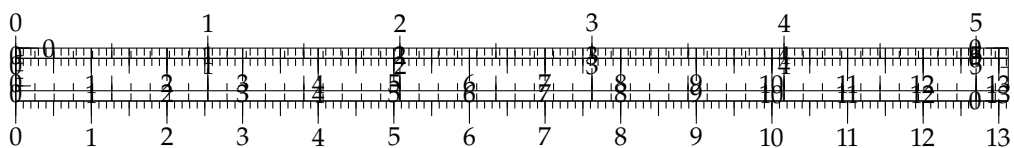
6.1	%gall . . . . .	10
	Walkthrough: Time (Clock) . . . . .	10
	Walkthrough: Chat CLI . . . . .	10
	Walkthrough: Publish . . . . .	10
	Walkthrough: %graph-store . . . . .	10
	Walkthrough: Drum and Helm . . . . .	10
	Walkthrough: Bitcoin API . . . . .	10
	Walkthrough: Bots . . . . .	10
6.2	Urbit API . . . . .	10

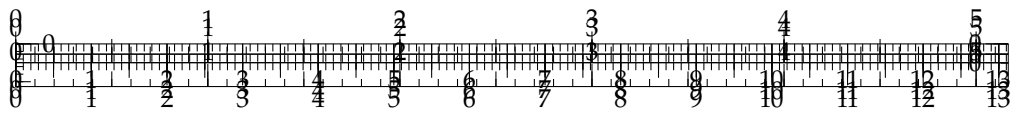
### **7 Supporting Urbit 11**

7.1	Bootng and Pills . . . . .	11
7.2	%unix Events . . . . .	11
7.3	Nock Virtual Machines . . . . .	11
	++mock . . . . .	11
	King and Serf Daemons . . . . .	11
7.4	Jetting . . . . .	11
	Jet matching and the dashboard . . . . .	11

### **8 Concluding Remarks 12**

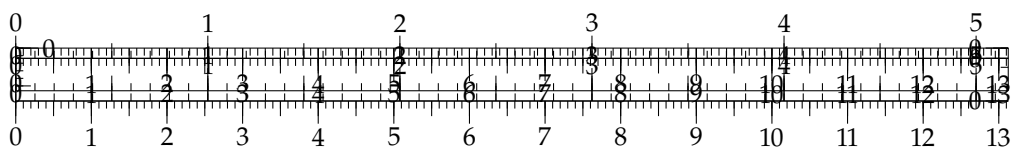
8.1	Bootng and Pills . . . . .	12
-----	----------------------------	----

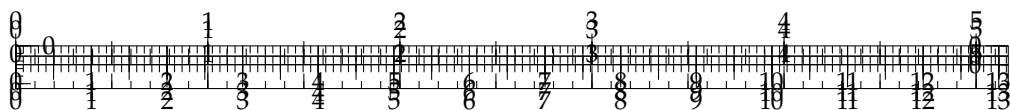




**APPENDIX** **13**

<b>A Appendices</b>	<b>14</b>
A.1 Comprehensive table of Hoon runes . . . . .	14
A.2 Hoon versions . . . . .	14
A.3 Nock versions . . . . .	14
A.4 Hoon comparison with other languages . . . . .	14
A.5 %zuse/%lull versions . . . . .	14
A.6 Textbook changelog . . . . .	14

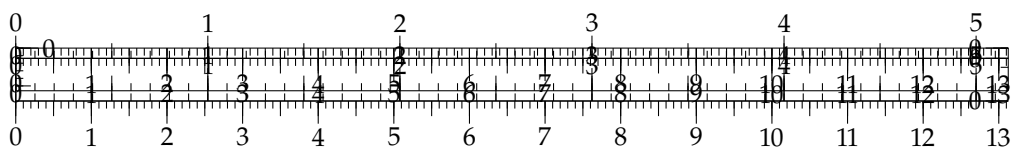




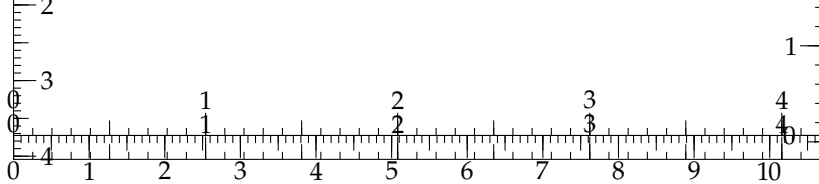
## List of Figures

1.1 The Mona Lisa . . . . .	2
-----------------------------	---

## List of Tables







# Chapter 1

## A Brief Introduction

### 1.1 Why Urbit Matters

Urbit is a network-first, compatibility-breaking

As of this writing, Urbit runs on any of several interpreters as a "hosted OS," or a

1.1 Why Urbit Matters . . . . .	1
1.2 Azimuth, the Urbit Address Space . . . . .	1
1.3 Accessing Urbit . . . . .	2
1.4 Developing for Urbit . . . . .	2

### 1.2 Azimuth, the Urbit Address Space

Many modern printed textbooks have adopted a layout with prominent margins where small figures, tables, remarks and just about everything else can be displayed. Arguably, this layout helps to organise the discussion by separating the main text from the ancillary material, which at the same time is very close to the point in the text where it is referenced.

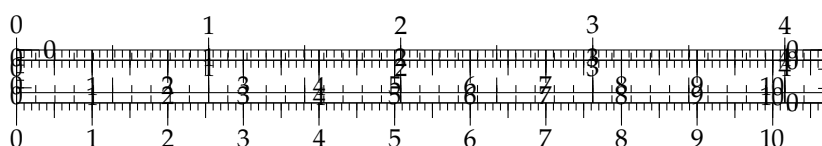
This document does not aim to be an apology of wide margins, for there are many better suited authors for this task; the purpose of all these words is just to fill the space so that the reader can see how a book written with the kaobook class looks like. Meanwhile, I shall also try to illustrate the features of the class.

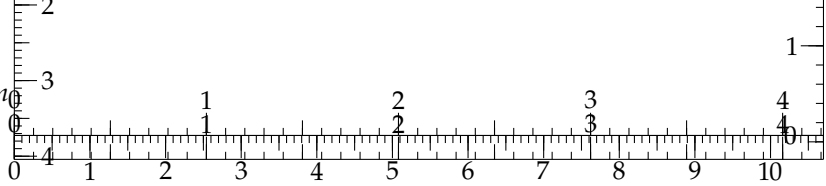
The main ideas behind kaobook come from this blog post, and actually the name of the class is dedicated to the author of the post, Ken Arroyo Ohori, which has kindly allowed me to create a class based on his thesis. Therefore, if you want to know more reasons to prefer a 1.5-column layout for your books, be sure to read his blog post.

Another source of inspiration, as you may have noticed, is the Tufte-Latex Class. The fact that the design is similar is due to the fact that it is very difficult to improve something which is already so good. However, I like to think that this class is more flexible than Tufte-Latex. For instance, I have tried to use only standard packages and to implement as little as possible from scratch;<sup>1</sup> therefore, it should be pretty easy to customise anything, provided that you read the documentation of the package that provides that feature.

In this book I shall illustrate the main features of the class and provide information about how to use and change things. Let us get started.

1: This also means that understanding and contributing to the class development is made easier. Indeed, many things still need to be improved, so if you are interested, check out the repository on github! In addition to the pronounceable `ops`, the sigil system affords a unique visual representation of each addressable point less than  $2^{32}$ .





## 1.3 Accessing Urbit

The kaobook class focuses more about the document structure than about the style. Indeed, it is a well-known  $\text{\LaTeX}$  principle that structure and style should be separated as much as possible (see also Section ?? on page ??). This means that this class will only provide commands, environments and in general, the opportunity to do things, which the user may or may not use. Actually, some stylistic matters are embedded in the class, but the user is able to customise them with ease.

The main features are the following:

**Page Layout** The text width is reduced to improve readability and make space for the margins, where any sort of elements can be displayed.

**Chapter Headings** As opposed to Tufte-Latex, we provide a variety of chapter headings among which to choose; examples will be seen in later chapters.

**Page Headers** They span the whole page, margins included, and, in twoside mode, display alternatively the chapter and the section name.<sup>2</sup>

**Matters** The commands `\frontmatter`, `\mainmatter` and `\backmatter` have been redefined in order to have automatically wide margins in the main matter, and narrow margins in the front and back matters. However, the page style can be changed at any moment, even in the middle of the document.

**Margin text** We provide commands `\sidenote` and `\marginnote` to put text in the margins.<sup>3</sup>

**Margin figs/tabs** A couple of useful environments is `marginfigure` and `marginfigure`, which, not surprisingly, allow you to put figures and tables in the margins (*cfr.* Figure 1.1).

**Margin toc** Finally, since we have wide margins, why don't add a little table of contents in them? See `\margintoc` for that.

**Hyperref** `hyperref` is loaded and by default we try to add bookmarks in a sensible way; in particular, the bookmarks levels are automatically reset at `\appendix` and `\backmatter`. Moreover, we also provide a small package to ease the hyperreferencing of other parts of the text.

**Bibliography** We want the reader to be able to know what has been cited without having to go to the end of the document every time, so citations go in the margins as well as at the end, as in Tufte-Latex. Unlike that class, however, you are free to customise the citations as you wish.

The order of the title pages, table of contents and preface can be easily changed, as in any  $\text{\LaTeX}$  document. In addition, the class is based on KOMA-Script's `scrbook`, therefore it inherits all the goodies of that.

2: This is another departure from Tufte's design.

3: Sidenotes (like this!) are numbered while marginnotes are not

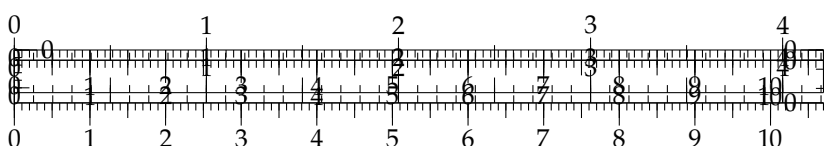


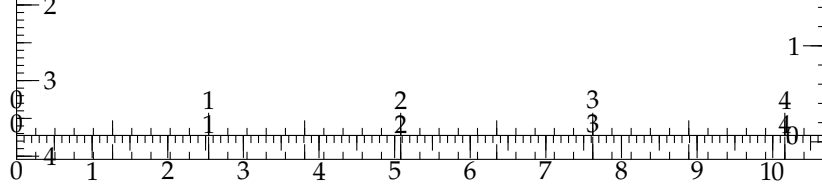
**Figure 1.1:** The Mona Lisa.  
[https://commons.wikimedia.org/wiki/File:Mona\\_Lisa,\\_by\\_Leonardo\\_da\\_Vinci,\\_from\\_C2RMF\\_retouched.jpg](https://commons.wikimedia.org/wiki/File:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg)

## 1.4 Developing for Urbit

Urbit development can be divided into three cases:

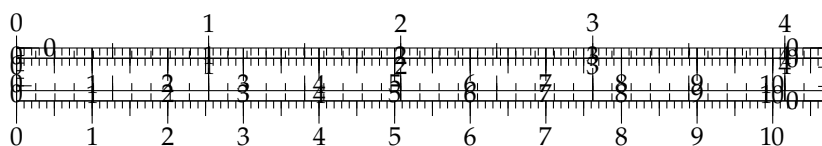
1. Kernel development
2. Userspace development, Urbit-side (`%gall` and generators)

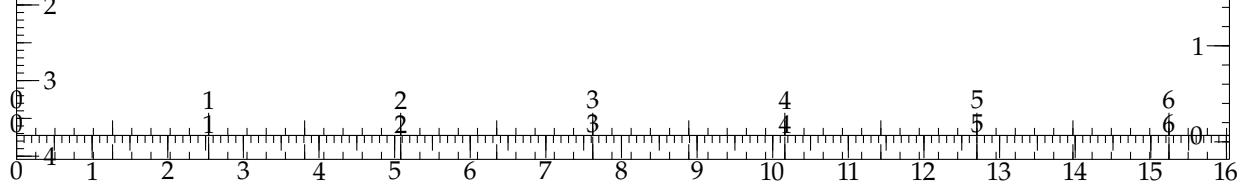




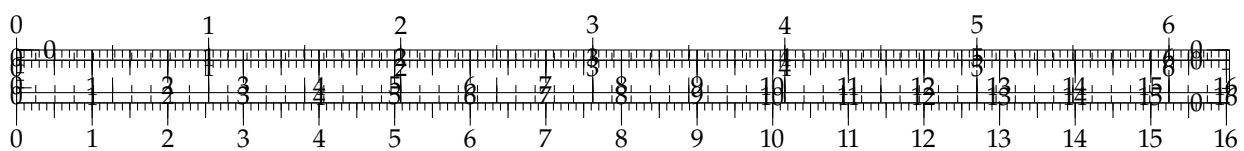
### 3. Userspace development, client-side (Urbit API)

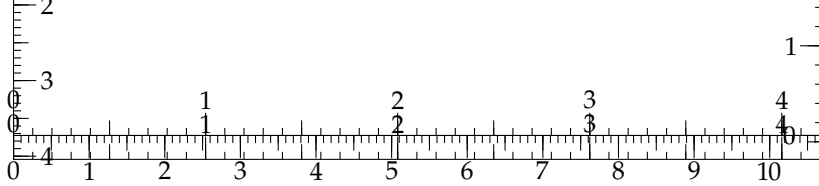
This guide focuses on getting the reader up to speed on the second development case early, then branches out.





# LANGUAGE ESSENTIALS





# Chapter 2

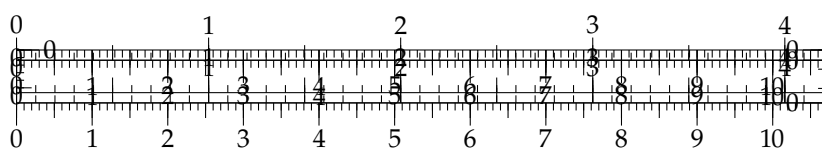
## Nock, A Combinator Language

### 2.1 Primitive rules and the combinator calculus

### 2.2 Compound rules

### 2.3 Kelvin versioning

2.1 Primitive rules and the combinator calculus . . . . .	5
2.2 Compound rules . . . . .	5
2.3 Kelvin versioning . . . . .	5





# Chapter 3

## Elements of Hoon

3.1 Reading the Runes . . . . .	6
3.2 Irregular Forms . . . . .	6
3.3 Nouns . . . . .	6
3.4 Hoon as Nock Macro . . . . .	6
3.5 Key Data Structures . . . . .	6
Cores, Gates, Doors . . . . .	6
Molds . . . . .	6
Maps, Sets, Tree . . . . .	6
3.6 Generators . . . . .	6
Naked Generators . . . . .	6
%say generators . . . . .	6
%ask generators . . . . .	6
3.7 Libraries . . . . .	6
3.8 Unit Tests . . . . .	6
3.9 Building Code . . . . .	6

### 3.1 Reading the Runes

### 3.2 Irregular Forms

### 3.3 Nouns

### 3.4 Hoon as Nock Macro

### 3.5 Key Data Structures

#### Cores, Gates, Doors

#### Molds

#### Maps, Sets, Tree

### 3.6 Generators

#### Naked Generators

#### %say generators

#### %ask generators

### 3.7 Libraries

### 3.8 Unit Tests

### 3.9 Building Code





# Chapter 4

## Advanced Hoon

### 4.1 Cores

Variadicity

Genericity

### 4.2 Molds

Polymorphism

### 4.3 Rune Families

### 4.4 Marks and Structures

### 4.5 Helpful Tools

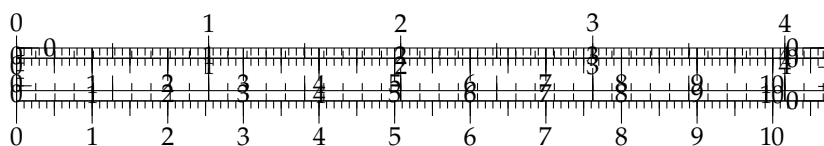
### 4.6 Deep Dives

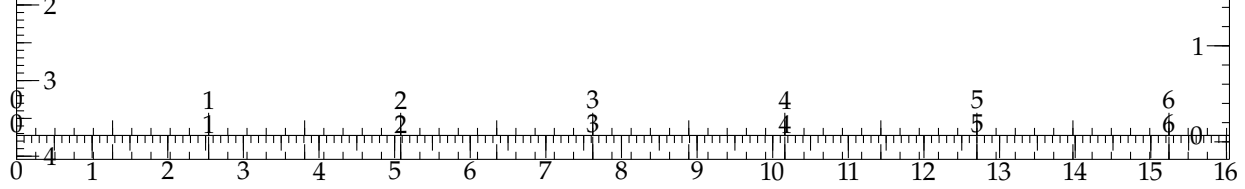
Text Stream Parsing

JSON Parsing

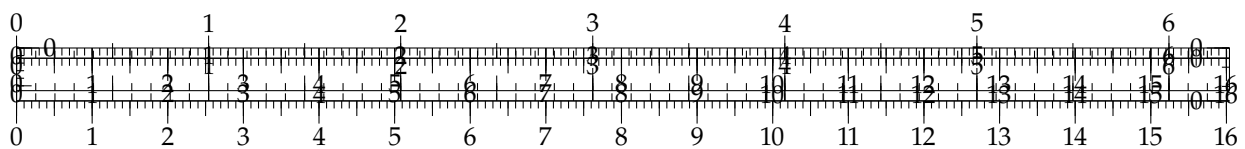
HTML/XML Parsing

4.1 Cores . . . . .	7
Variadicity . . . . .	7
Genericity . . . . .	7
4.2 Molds . . . . .	7
Polymorphism . . . . .	7
4.3 Rune Families . . . . .	7
4.4 Marks and Structures . . . . .	7
4.5 Helpful Tools . . . . .	7
4.6 Deep Dives . . . . .	7
Text Stream Parsing . . . . .	7
JSON Parsing . . . . .	7
HTML/XML Parsing . . . . .	7





# SYSTEM DEVELOPMENT







# Chapter 5

## The Kernel

### 5.1 Arvo

Arvo is essentially an event handler which can coordinate and dispatch messages between vanes as well as emit %unix events to the underlying (presumed Unix-compatible) host OS. Arvo does not carry out several tasks specific to the machine hardware, such as memory allocation, system thread management, and hardware- or firmware-level operations. These are left to the king and serf, or the daemon processes which together run Arvo. Collectively, the system-level instrumentation of Arvo is described in Chapter ??.

%zuse and %lull

### 5.2 %ames, A Network

### 5.3 %behn, A Timer

### 5.4 %clay, A File System

++ford, A Build System

Scrying

Marks and conversions

### 5.5 %dill, A Terminal driver

### 5.6 %eyre and %iris, Server and Client Vanes

### 5.7 %jael, Secretkeeper

### 5.8 Azimuth, Address Space Management

### 5.9 The Hoon Parser

5.1 Arvo . . . . .	9
%zuse and %lull . . . . .	9
5.2 %ames, A Network . . . . .	9
5.3 %behn, A Timer . . . . .	9
5.4 %clay, A File System . . . . .	9
++ford, A Build System . . . . .	9
Scrying . . . . .	9
Marks and conversions . . . . .	9
5.5 %dill, A Terminal driver . . . . .	9
5.6 %eyre and %iris, Server and Client Vanes . . . . .	9
5.7 %jael, Secretkeeper . . . . .	9
5.8 Azimuth, Address Space Management . . . . .	9
5.9 The Hoon Parser . . . . .	9





# Chapter 6

## Userspace

6.1 %gall . . . . .	10
Walkthrough: Time (Clock) . . .	10
Walkthrough: Chat CLI . . .	10
Walkthrough: Publish . . .	10
Walkthrough: %graph-store . . .	10
Walkthrough: Drum and Helm . . .	10
Walkthrough: Bitcoin API . . .	10
Walkthrough: Bots . . . . .	10
6.2 Urbit API . . . . .	10

### 6.1 %gall

Walkthrough: Time (Clock)

Walkthrough: Chat CLI

Walkthrough: Publish

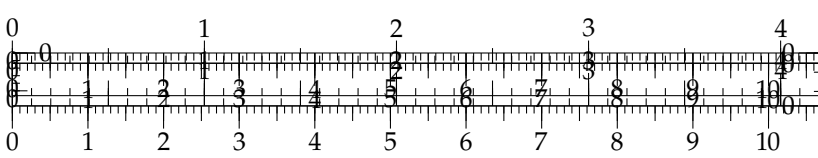
Walkthrough: %graph-store

Walkthrough: Drum and Helm

Walkthrough: Bitcoin API

Walkthrough: Bots

### 6.2 Urbit API





# Chapter 7

## Supporting Urbit

### 7.1 Booting and Pills

### 7.2 %unix Events

### 7.3 Nock Virtual Machines

**++mock**

**King and Serf Daemons**

**Vere (Reference C Implementation)**

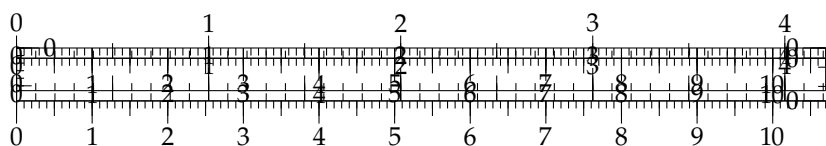
**King Haskell (Haskell Implementation)**

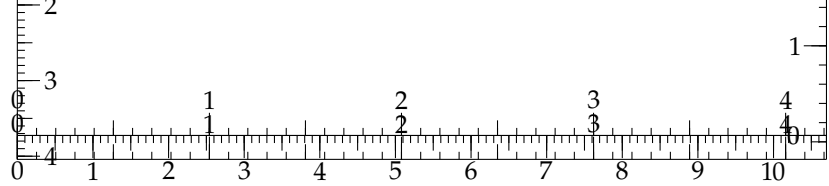
**Jaque (JVM Implementation)**

### 7.4 Jetting

**Jet matching and the dashboard**

7.1 Booting and Pills . . . . .	11
7.2 %unix Events . . . . .	11
7.3 Nock Virtual Machines . . . .	11
++mock . . . . .	11
King and Serf Daemons . . . .	11
7.4 Jetting . . . . .	11
Jet matching and the dash- board . . . . .	11



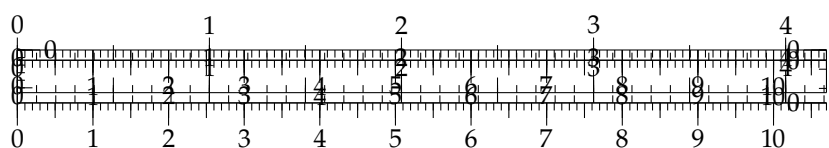


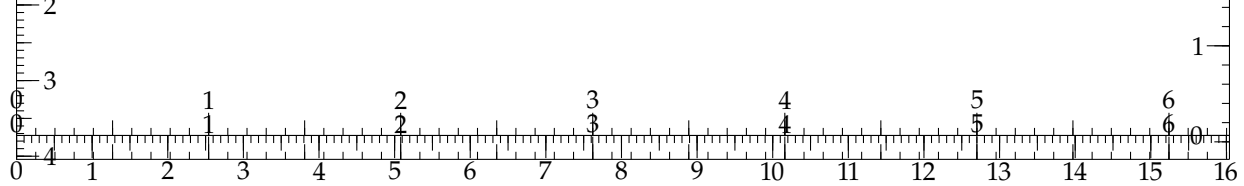
# Chapter 8

## Concluding Remarks

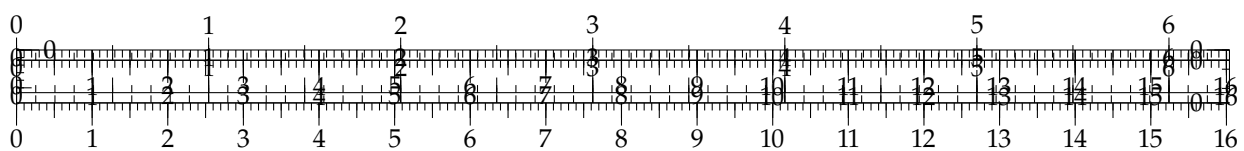
8.1 Booting and Pills . . . . . 12

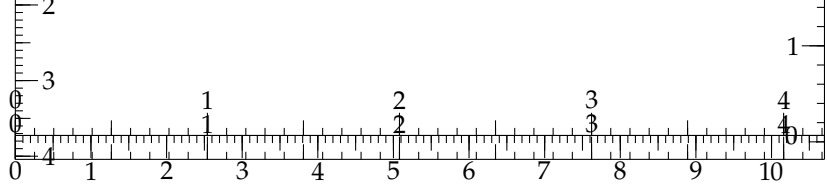
### 8.1 Booting and Pills





## APPENDIX





# Appendix **A**

## Appendices

A.1 Comprehensive table of Hoon runes . . . . .	14
A.2 Hoon versions . . . . .	14
A.3 Nock versions . . . . .	14
A.4 Hoon comparison with other languages . . . . .	14
A.5 %zuse/%lull versions . . . . .	14
A.6 Textbook changelog . . . . .	14

### A.1 Comprehensive table of Hoon runes

### A.2 Hoon versions

### A.3 Nock versions

### A.4 Hoon comparison with other languages

### A.5 %zuse/%lull versions

### A.6 Textbook changelog

