

|           |   |                      |
|-----------|---|----------------------|
| <b> </b>  | <b>bars make cores</b>  |                      |
| _         | produces a door (a core with sample)                                    |                      |
| %         | produces a core (battery and payload)                                   |                      |
| @         | produces a wet core (battery and payload)                               |                      |
| .         | produces a trap (a core with one arm)                                   |                      |
| :         | produces a gate with a custom sample                                    |                      |
| -         | produces a trap (a core with one arm) and evaluates it                  |                      |
| ^         | produces a core whose battery includes a \$ arm and computes the latter |                      |
| ~         | produces an iron gate   |                      |
| *         | produces a wet gate (a one-armed core with sample)                      |                      |
| =         | produces a dry gate (a one-armed core with sample)                      |                      |
| ?         | produces a lead trap  |                      |
| \$        | produces a mold   |                      |
| <b>\$</b> | <b>bucs form molds</b>  |                      |
| \$@       | structure that normalizes a union tagged by head atom                   |                      |
| \$:       | forms a cell type   | [a=foo b=bar c=baz]  |
| \$_       | structure that normalizes to an example                                 | _foo                 |
| \$%       | structure that recognizes a union tagged by head atom                   |                      |
| \$*       | bunt (irregular form is *)  |                      |
| \$^       | structure that normalizes a union tagged by head depth (cell)           |                      |
| \$~       | defines a custom type default value                                     |                      |
| \$-       | structure that normalizes to an example gate                            |                      |
| \$=       | structure that wraps a face around another structure                    | foo=bar              |
| \$?       | forms a type from a union of other types                                | ?(\$foo \$bar \$baz) |
| <b>%</b>  | <b>cens put the fun in function</b>                                     |                      |
| %_        | resolves a wing with changes, preserving type                           |                      |
| %.        | calls a gate, inverted  |                      |
| %^        | calls a gate with triple sample   |                      |
| %+        | calls a gate with a cell sample   |                      |
| %-        | calls a gate  | (fun arg)            |
| %:        | calls a gate with many arguments  |                      |
| %~        | evaluates an arm in a door  | ~(arm core arg)      |
| %*        | evaluates an expression, then resolves a wing with changes              |                      |
| %=        | resolves a wing with changes  | foo(x 1, y 2, z 3)   |
| <b>:</b>  | <b>cols make cells</b>  |                      |
| :_        | constructs a cell, inverted   |                      |
| :^        | constructs a cell, 4-tuple  | [a b c d]            |
| :+        | constructs a cell, 3-tuple  | [a b c]              |
| :-        | constructs a cell, 2-tuple  | [a b], a^b           |
| :~        | constructs a null-terminated list                                       | ~[a b c]             |
| :*        | constructs an n-tuple   | [a b c d e ...]      |
| ::        | marks a comment   |                      |
| <b>.</b>  | <b>dots nock</b>  |                      |
| .+        | increments an atom using Nock 4   | +(42)                |
| .*        | evaluates using Nock 2  |                      |
| .=        | tests for equality using Nock 5   | =(a b)               |
| .?        | tests for cell or atom using Nock 3                                     |                      |
| .^        | loads from namespace using Nock 12                                      |                      |

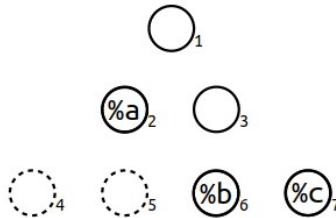
|               |   |                             |
|---------------|---|-----------------------------|
| <b>^</b>      | <b>kets cast</b>  |                             |
| <b>^ </b>     | converts a gold core to an iron core (invariant)                        |                             |
| <b>^.</b>     | typecasts on value  |                             |
| <b>^-</b>     | typecasts by explicit type label  | <code>`foo`bar</code>       |
| <b>^+</b>     | typecasts by inferred type  |                             |
| <b>^&amp;</b> | converts a core to a zinc core (covariant)                              |                             |
| <b>^~</b>     | folds constant at compile time  |                             |
| <b>^=</b>     | binds name to a value   | <code>foo=bar</code>        |
| <b>^?</b>     | converts a core to a lead core (bivariant)                              |                             |
| <b>^*</b>     | produces example type value   |                             |
| <b>^:</b>     | produces a 'factory' gate for a type                                    |                             |
| <b>~</b>      | <b>sigs hint</b>  |                             |
| <b>~ </b>     | prints in stack trace if failure  |                             |
| <b>~\$</b>    | profiler hit counter  |                             |
| <b>~_</b>     | prints in stack trace, user-formatted                                   |                             |
| <b>~%</b>     | registers jet   |                             |
| <b>~/</b>     | registers jet with registered context                                   |                             |
| <b>~&lt;</b>  | raw hint, applied to product ("backward")                               |                             |
| <b>~&gt;</b>  | raw hint, applied to computation ("forward")                            |                             |
| <b>~+</b>     | caches a computation  |                             |
| <b>~&amp;</b> | prints (used for debugging)   |                             |
| <b>~?</b>     | prints conditionally (used for debugging)                               |                             |
| <b>~=</b>     | detects duplicate   |                             |
| <b>~!</b>     | prints type if compilation failure                                      |                             |
| <b>;</b>      | <b>mics make</b>  |                             |
| <b>;;</b>     | calls a binary function as an \$n\$-ary function                        | <code>:(fun a b c d)</code> |
| <b>;&lt;</b>  | glues a pipeline together (monadic bind)                                |                             |
| <b>~</b>      | glues a pipeline together with a product-sample adapter (monadic bind)  |                             |
| <b>;;</b>     | normalizes with a mold, asserting fixpoint                              |                             |
| <b>+</b>      | ( <a href="#">Sail</a> ) makes a single XML node                        |                             |
| <b>*</b>      | ( <a href="#">Sail</a> ) makes a list of XML nodes from Hoon expression |                             |
| <b>=</b>      | ( <a href="#">Sail</a> ) makes a list of XML nodes                      |                             |
| <b>/</b>      | ( <a href="#">Sail</a> ) yields tape as XML element                     |                             |
| <b>=</b>      | <b>tises alter</b>  |                             |
| <b>= </b>     | combines default type value with the subject                            |                             |
| <b>=.</b>     | changes one leg in the subject  |                             |
| <b>=?</b>     | changes one leg in the subject conditionally                            |                             |
| <b>=^</b>     | pins the head of a pair; changes a leg with the tail                    |                             |
| <b>=:</b>     | changes multiple legs in the subject                                    |                             |
| <b>=/</b>     | combines a named noun with the subject                                  |                             |
| <b>=;</b>     | combines a named noun with the subject, inverted                        |                             |
| <b>=&lt;</b>  | composes two expressions, inverted                                      | <code>foo:bar</code>        |
| <b>=&gt;</b>  | composes two expressions  |                             |
| <b>=-</b>     | combines a new noun with the subject                                    |                             |
| <b>=*</b>     | defines an alias  |                             |
| <b>=,</b>     | exposes namespace   |                             |
| <b>=+</b>     | combines a new noun with the subject                                    |                             |
| <b>=~</b>     | composes many expressions   |                             |

|                     |  |
|---------------------|--|
| <b>? wuts test</b>  |  |
| ?                   | logical OR<br> (foo bar baz)   |
| ?:                  | branches on a boolean test   |
| ?.                  | branches on a boolean test, inverted   |
| ?<                  | negative assertion   |
| ?>                  | positive assertion   |
| ?-                  | switches against a union, no default   |
| ?^                  | branches on whether a wing of the subject is a cell  |
| ?=                  | tests pattern match  |
| ?#                  | tests pattern match  |
| ?+                  | switches against a union, with default   |
| ?&                  | logical AND<br>&(foo bar baz)  |
| ?@                  | branches on whether a wing of the subject is an atom   |
| ?~                  | branches on whether a wing of the subject is null  |
| ?!                  | logical NOT<br>!foo  |
| <b>! zaps wild</b>  |  |
| !:                  | turns on stack trace   |
| !.                  | turns off stack trace  |
| !,                  | emits AST of expression  |
| !;                  | emits the type for an expression using the type of type  |
| !>                  | wraps a noun in its type   |
| !=                  | makes the Nock formula for a Hoon expression   |
| !?                  | restricts Hoon version   |
| !!                  | crashes  |
| !<                  | lift dynamic value into static context   |
| <b>/ fases ford</b> |  |
| /\$                 | slams a gate on extra arguments  |
| /                   | takes a series of horns and produces the first one (L-to-R) that succeeds; if none succeed, produces stack traces from arguments   |
| /=                  | runs a horn (usually produced by another Ford rune), takes the result of that horn, and wraps a face around it   |
| /.                  | produces a null-terminated list from a sequence of horns, terminated by ==   |
| /,                  | acts as switch statement, picking a branch to evaluate based on whether the current path matches the path in the switch statement  |
| /&                  | pass a horn through multiple marks   |
| /_                  | <i>unfiltered</i> : takes a horn, producing new horn mapping supplied horn over list of files in current directory; <i>filtered</i> : runs a horn on each file matching aura |
| /~                  | produces a horn that evaluates a twig and places the product in the subject  |
| /:                  | takes a path and a horn, and evaluates the horn with the current path set to the supplied path   |
| /^                  | takes a mold and a horn, and casts the result of the horn to the mold  |
| /!                  | produces a mark  |
| /+                  | accepts a filename and loads that filename from the lib directory  |
| /-                  | accepts a filename and loads that filename from the sur directory  |
| //                  | parses relative path as a hoon twig, and adds the resulting twig to the subject  |
| /;                  | takes a twig and a horn; the twig should evaluate to a gate, which is then slammed with the result of the horn as its sample   |

**/#** takes a horn and produces a cell of the dependency hash of the result of the horn, and the result itself  
**/%** forwards extra arguments to enclosed renderers  
**-/=** **terminators terminate**  
**--** terminates core expression  
**==** terminates running series of Hoon expressions  
**+** **luses change**  
**+|** labels a chapter  
**+\$** produces a structure arm (type definition)  
**++** produces a (normal) arm  
**+\*** produces a type constructor arm

**syntax**

|                              |              |                 |              |
|------------------------------|--------------|-----------------|--------------|
| +1:[%a [%b %c]]              | [%a [%b %c]] | ..[%a [%b %c]]  | [%a [%b %c]] |
| +2:[%a [%b %c]] %a           |              | -.[%a [%b %c]]  | %a           |
| +3:[%a [%b %c]] [%b %c]      |              | +. [%a [%b %c]] | [%b %c]      |
| +4:[%a [%b %c]] %ride failed |              | -<:[%a [%b %c]] | %ride failed |
| +6:[%a [%b %c]] %b           |              | +<:[%a [%b %c]] | %b           |
| +7:[%a [%b %c]] %c           |              | +>:[%a [%b %c]] | %c           |



· current subject

+ +:.

- -:.

+> +>:.

^face face in outer core

..arm core in which ++arm is defined

~ 0 (nil)

%.y & yes/true

%.n | no/false

eny entropy

now current time

`a [~ a]

~[a b c] [a b c ~]

[a b c]~ [[a b c] ~]

<[1 2 3]> renders list as a tape

>[1 2 3]< renders list as a tank

?=(\$hoon %hoon) %.y

?=(\$hoon %loon) %.n

**@p notation**

|      |   |                              |
|------|---|------------------------------|
| @c   | Unicode codepoints                                  | ~~45fed.                     |
| @d   | Date  |                              |
| @da  | Date, absolute                                      | ~2020.12.25..7.15.0..1ef5    |
| @dr  | Date, relative                                      | ~d71.h19.m26.s24..9d55       |
| @n   | Nil   |                              |
| @p   | Phonemic base                                       | ~laszod-dozser-fosrum-fanbyr |
| @r   | IEEE-754 floating-point number                      |                              |
| @rh  | Floating-point number, half-precision, 16-bit       | .~~3.14                      |
| @rs  | Floating-point number, single-precision, 32-bit     | .3.141592653589793           |
| @rd  | Floating-point number, double-precision, 64-bit     | .~3.141592653589793          |
| @rq  | Floating-point number, quadruple-precision, 128-bit | .~~~3.141592653589793        |
| @s   | Integer, signed (sign bit low)                      |                              |
| @sb  | Signed binary                                       | --0b10.0000                  |
| @sd  | Signed decimal                                      | --1.000                      |
| @sv  | Signed base-32                                      | --0v201.4gvml.245kc          |
| @sw  | Signed base-64                                      | --0w2.04AfS.G8xqc            |
| @sx  | Signed hexadecimal                                  | --0x2004.90fd                |
| @t   | UTF-8 text (cord)                                   | "urbit"                      |
| @ta  | ASCII text (knot)                                   | ~.urbit                      |
| @tas | ASCII text symbol (term)                            | %urbit                       |
| @u   | Integer, unsigned                                   |                              |
| @ub  | Unsigned binary                                     | 0b10.1011                    |
| @ud  | Unsigned decimal                                    | 8.675.309                    |
| @uv  | Unsigned base-32                                    | 0v88nvd                      |
| @uw  | Unsigned base-64                                    | 0wx5~J                       |
| @ux  | Unsigned hexadecimal                                | 0x84.5fed                    |