Mike Hahn – hahnbytes@gmail.com

Sharebiteach

Sharebiteach is an edutainment software development kit used to create web-based/Android games and lessons (called apps). All apps are either single or dual-user and are written in a Python-like language called Sharebitalk. Screen layouts are structured using a markup language called Sharebitags. Web server functionality is provided by a web container called Jetty.

Business Model

Sharebiteach links tutors with those needing instruction, and receives 15 percent of the fees charged to students by the tutors (10 percent goes to Sharebiteach and 5 percent goes to the app-writers). The app-writer revenue is distributed in proportion to the user-minute counts for each app. The user-minute count of a given app/user is incremented whenever a user is logged into that app at any time in a given minute (a day has 1440 minutes).

Monospace Mode

In monospace mode, all body text rendered to the browsers of end-users in every Sharebiteach app is in a mono-spaced, typewriter-style font. Every character takes up 2 square cells: an upper cell and a lower cell. Superscripts and subscripts are handled by employing a vertical offset of one square cell. Header text is also mono-spaced, and each character takes up 2 oversized square cells.

Additional Formatting

The grid of characters can be subdivided into panels, which can themselves be subdivided into more panels, and so on. Any panel can contain zero or more text boxes, which may overlap each other. Vertical grid lines each take up one square cell per row of square cells. Horizontal grid lines are displayed in the same pixel row as underscore characters. Any row of square cells containing a horizontal grid line which is 2 pixels wide is taller by exactly one pixel. The following bracket characters: ( ) [ ] { } can be oriented vertically or horizontally, taking up a single column or row of at least 2 square cells, respectively. Widgets such as check boxes, radio buttons, and combo box arrows take up 4 square cells (2 by 2). Images, animations, and diagrams are contained in canvas objects, which can appear anywhere panels can appear.

Rich-Text Mode

In rich-text mode, a given header or paragraph of body text can consist of a single variable-width font. Paragraphs have before/after spacing, left/right indent, and line spacing (single, double, 1.5, etc.). Panels have margins on all 4 sides. Beginner app writers start off with monospace mode, and then advance to rich-text mode. In both rich-text and monospace modes, text is rendered to the HTML5 canvas object. Some features like form fields and submit buttons use normal HTML.

Grading of Content

Users can assign a letter grade (on a 4.0 scale) to the apps/tutors they use: A = excellent, B = good, C = average, D = poor, F = fail. Users are warned prominently if a given app/tutor received any F grades in the past year. If a given user has given out more than one F grade to multiple apps/tutors in the past 12 months, then those grades are flagged as possibly unreliable.

Console Mode

User enters command-line in a text input field. Sharebiteach displays zero or more lines of output. When input from the user is needed, another text input field is displayed. Entering a single space in the text input field causes a page refresh, in which all lines of output between the current and previous text input fields are redisplayed at the top of the current web page. Entering a single space when nothing else is displayed on the current web page causes Sharebiteach to cease execution.

Events

- Keyboard: key up, key down
- Mouse: mouse up, mouse down
- Clock Tick: 0.5 Hz
- Character paint: triggered by keyboard/mouse/clock event

Object Primitives

- Bitmaps
- Vectors: pen color (on/off), multiple fill colors, single pen-width
- Shape list:
    - line, rectangle, rounded rectangle, ellipse, pie
    - each shape has one vector
    - pie has begin/end angle (float)
- Label
- Character: bitmap, shape list, and/or label

Containers

- Cell
- Canvas
- Grid
- Text box:
    - found in cell/canvas/grid
    - contains Sharebitags code

Level Editor Commands

- Arrow keys:
    - up level, down level, next, previous
- Home/End: first, last
- Click: select char. or cell
- Enter: toggle color
- Space: black/gray pen color (on keydown)
- Esc: normal mode
- Ctrl+Enter: border mode
- F1: cycle menu/alpha/off

Levels

- Top (top level):
  - show red border
  - step through list of top-level cells:
    - first, previous, next, last
    - each cell fills screen
- Parent:
  - show red border
- Cell:
  - show green border
  - move to sibling:
    - first, previous, next, last
    - move char. in previous cell
- Character:
  - show blue border
  - step through list of sub-characters:
    - first, previous, next, last
- Pen:
  - Space: black/gray pen color (on keydown)
  - Enter: toggle color
  - Next/Previous: pen widths
- Shape (bottom level):
  - step through shape list:
    - first, previous, next, last
  - Space: black/gray pen color (on keydown)
  - Enter: toggle color
  - Delete
  - Insert: new shape
  - A: select all
  - M: move, 2 clicks
  - R: resize, 2 clicks
  - B: to back
  - F: to front
  - C: copy
  - V: paste
  - P: polyline
  - Click on label: edit
  - Click on bitmap:
    - select pixel
    - Enter: toggle color of all similar pixels
- Board modification:
  - Shift+Arrow: select multiple cells
  - Delete: merge cells
  - Insert: split cell (many allowed)
  - Shift+Insert: split cell (same way as siblings, many allowed)
  - Shift+Click: select border
    - Click: move border
  - Space on multi-selected: make same size
  - Space again: form grid (many allowed)

- Border mode:
    - Enter: toggle color
    - Next/Previous: pen widths
    - Toggle border:
        - L: left
        - R: right
        - T: top
        - B: bottom
        - M: middle
        - O: outside

About Me

I am Mike Hahn, the founder of Sharebiteach.com. I was previously employed at Brooklyn Computer Systems as a Delphi Programmer and a Technical Writer (I worked there between 1996 and 2013). At the end of 2014 I quit my job as a volunteer tutor at Fred Victor on Tuesday afternoons, where for 5 years I taught math, computers, and literacy. I'm now a volunteer computer tutor at West Neighbourhood House. My hobbies are reading quora.com questions/answers and the news at cbc.ca. About twice a year I get together with my sister Cathy who lives in Victoria. She comes here or I go out there usually in the summer. Prior to starting Sharebiteach I used to lie on the couch a lot, not being very active. Now I'm busy most of the time. I visit my brother Dave once a month or so and I also visit my friends Main and Steph once or twice a month.

Contact Info

Mike Hahn
Founder, Sharebiteach.com
515-2495 Dundas St. West
Toronto, ON M6P 1X4

Country: Canada
Phone: 416-533-4417
Email: hahnbytes (AT) gmail (DOT) com
Web: www.hahnbytes.com

Lesson Player

- Default background color: white
- Default color of content text: black
- Color of tutor text: gray
- Color of student text: blue
- Page Up/Down: scroll a page at a time
- Ctrl+Home/End: top/bottom
- F6: toggle user
- Click: move text cursor, display red mouse pointer on the other user's screen
- Arrow key: move text cursor
- Letter, digit, symbol, space: insert char.
- Insert: toggle insert/overwrite mode
- Underscore: toggle underline of current char. and move right
- Vertical slash: toggle vertical grid line at cursor and move down
- Ctrl+vertical slash: insert vertical slash char.
- Home/End: beginning/end of line
- Enter: append blank line, cursor backs up to line up with beginning of word to left of original cursor position on previous non-blank line
- Backspace (on blank line): cursor backs up to line up with beginning of previous word on previous non-blank line (unindent one word at a time)
- Delete: delete/clear highlighted block (depends on insert/overwrite mode)
- Shift+Arrow: highlight multi-line block if up/down and nothing already highlighted, else highlight rectangular block
- F1: show/hide menu (Ctrl+Letters)
- Ctrl+(Z, X, C, V, Y, B, I, U, J, O, N, F, R, H, G, K, Q, S, P, T, F1): undo, cut, copy, paste, redo, bold, italics, underline, justify, overwrite toggle, notes/content view toggle, find, search/replace, heading, graphic, hyperlink, special char., shade/borders, panel/text box, table, help screen
- Ctrl+J: justify, use arrow keys: Up = cancel, Down = center
- Ctrl+Up/Down Arrow: superscript/subscript
- Ctrl+Left/Right Arrow: move left/right word at a time
- Parenthesis/square bracket/brace bracket: insert oversize open/close bracket if:
  - a narrow block of text is highlighted (one square cell by at least 3 square cells)
- Else if a wider block of text is highlighted (at least 3 cells tall and 2 cells wide):
  - enclose highlighted block in oversize open/close brackets
- Quiz mode:
  - Similar to content view
  - All lines containing content text (usually black text) are read-only
  - User presses Enter to insert a blank line
  - User presses Enter to split line containing user text (blue) into 2 separate lines
- Play Content:
  - Navigation buttons in a row at top left (content view)
  - Start Icon: Access file commands:
    - Open, Close, New, Save, Save As, Print, Exit
  - [| <] : Beginning
  - [> |] : End
  - [<<] : Previous Slide
  - [>>] : Next Slide
  - [^] : Beginning of Current Slide
  - [>] : Play
  - [||] : Pause
  - Space : Single-Step
  - [<] : Reverse Single-Step

- Chat Window:
    - Enter: transmit chat entry
    - Up/Down Arrow: display previous/next chat entry of current user
    - Shift+Up/Down Arrow: display previous/next chat entry of different user
    - Left/Right Arrow: used to edit chat entry (highlight if used with Shift)
    - Ctrl+(Z, X, C, V, Y): undo, cut, copy, paste, redo
- Mac commands (optionally disabled on non-Macs):
    - Ctrl+O: same as Insert
    - Shift+Ctrl+E: same as Home
    - Ctrl+E: same as End
    - Delete: same as Backspace

Implementation Steps

1. Learn to use GitHub with Java project
2. Create Jabbler: multi-player Scrabble (single-user)
    1. Write Jabbler pseudo-code - ***done!***
    2. Register Jabbler project with GitHub - ***done!***
    3. Implement Jabbler command-line version - ***done!***
    4. Add robot player(s) - ***done!***
    5. Two human players use mouse/keyboard
3. Implement Sharebitalk Assembler
4. Implement SharebiTalk Runtime Environment (STRE)
5. Implement Sharebitalk Compiler
6. Implement Console mode
7. Implement Team Monitor
8. Cold email Toronto/Vancouver-based GitHub members who use Java
9. Steph and Dr. Soni to approach U of T Computer Science dept.
10. Implement Monospace mode
11. Implement Rich-Text mode
12. Implement Level Editor
13. Make Level Editor dual-user
14. Make Jabbler dual-user
15. Implement Lesson Player (monospace)
16. Make Lesson Player dual-user
17. Make Lesson Player support rich-text
18. Add Sharebitalk scripting to Level Editor
19. Add Sharebitalk scripting to Lesson Player
20. Recruit Java Programmer on the autism spectrum using Specialisterne
    1. Must know some front-end web programming
    2. Knowledge of back-end Java web programming or Android an asset
21. Mike to develop Sharebiteach SDK
    1. Code editors: Sharebitalk and Sharebitags (Eclipse plug-in)
    2. Syntax highlighting, unlimited undo/redo
    3. Graphical Sharebitags editor
22. Java Programmer to develop website
    1. Tutor/app rating
    2. Tutor search
    3. Tutor scheduling
    4. Work with Mike to develop Android app functionality
23. Go live
24. Purchase Google AdWords advertising
25. Expand Sharebitalk libraries: widgets, graphics, games, math, etc.
26. Enable Sharebitalk to call user-created Java libraries
27. Add multi-user support (>2 users)
28. Finish implementing Android app functionality

Our Team

Those who know Java and wish to participate in Sharebiteach development are more than welcome to read up on our project and determine which aspects of Sharebiteach development are of interest to them. Sharebiteach is mostly open source but includes a closed source component, therefore our team members receive equity (profit sharing) but otherwise work on a volunteer basis. You would work from home and are free to work lots of hours or very few hours, whatever is most convenient for you. Not all team members receive the same amount of equity. Higher performing team members receive more equity than team members who contribute to a lesser extent.

All team members fall into 3 categories: partners, associates, and entry-level members (ELMs). Partners make substantial contributions to the project, ELMs make only trivial contributions, and associates are somewhere in between. All partners, including Mike, the founder, receive the same amount of equity. All associates receive an equal share of equity which adds up to the amount of equity received by 1.2 partners. The amount of equity received by each associate is capped at 40 percent of the amount of equity received by each partner. Similarly, all ELMs receive an equal share of equity which adds up to the amount of equity received by 1.2 associates. The amount of equity received by each ELM is capped at 40 percent of the amount of equity received by each associate. All team members are required to download the Team Monitor application. Partners and associates must sign an employment agreement before they are allowed to access the private repositories where the source code of the Sharebiteach.com website is maintained.

Team Monitor

The Team Monitor application is written in Java. The user selects from 7 open source projects: Sharebitalk, Sharebitags, Monospace Mode, Rich-Text Mode, Sharebitalk Editor, Sharebitags Editor, WYSIWYG Editor (as well as closed source projects pertaining to the website). The user indicates which source files have been added/modified/deleted; the nature of the work: testing, debugging, adding new code, writing design specs; descriptions of work performed; and the approximate amount of time spent on each task. The user uploads Team Monitor data on a weekly basis.

Sponsor Organization

Mike's friend Steph is a student at U of T, and Mike's psychiatrist, Dr. Soni, teaches at U of T. Mike was a computer science student at U of T for one semester in his second year. After the Sharebitalk programming language is working, it is hoped that both Steph and Dr. Soni will approach the U of T Computer Science department. It is then hoped that U of T will eventually become a sponsor of Sharebiteach. Steph may or may not agree to work as the Tutor Coordinator after she graduates, supervising the tutors. She would be paid a competitive wage and would receive partner-level equity, as the Tutor Coordinator or working in some other capacity (possibly software development). If she declines to become the Tutor Coordinator and does not wish to work for Sharebiteach, she would still receive associate-level equity, provided that she agrees to approach U of T and that institution becomes a sponsor of Sharebiteach.

Sharebitags Format

Simple Tag	`{<taghdr>}`
Container Tag	`{<taghdr> │ <body>}`
Null Tag	`{│ <body>}`
List Tag	`{<taghdr> <list>}`
Body List Tag	`{<taghdr> <bodlst>}`
<bodlst>	`[ │ <body>]*`
<list>	`[<col>]*`
<col>	`│ [<fldval>]* │ <body>`
<taghdr>	`<tagname> [<fldval>]*`
<fldval>	`fldname=value;` `fldname;`
<body>	body text
Comment	{% ... %}
Escape Char.	backslash (\)
Repetition:	`[ XYZ ]*`

- *XYZ* repeats zero or more times

Tag Names:
- table, row, box, grid, borders
- canvas, select, quote, rt, mono
- super, sub, text, pre, br, hr, img, a, ch, p
- input, radio, checkbox, meta, sbtk
- styles, include, h1..h5, b, i, u, ol, ul, jy

Mono Mode:
- body color fcolor (c/f)
- panel c/f width height borders-tag
- borders
  - left right top bottom inner outer none wleft wright wtop wbottom winner wouter
  - left=80ff44 right=0FF0000
  - wleft=0..2 wtop=0..2 (pixels)
- table c/f width height left top borders-tag
  - w/h/l/t (cell squares, not pixels)
- row c/f height borders-tag
  - c/f width colspan rowspan borders-tag
- box c/f width height left top borders-tag
- grid c/f width height left top rowcount colcount rowheight colwidth borders-tag
- gridrow c/f height borders-tag
  - c/f width borders-tag
- canvas c/f width height left top borders-tag
  - w/h/l/t (pixels)

Fields (old):
- width=50/0.5 (pixels/ratio)
- pad=50/0.5
- x, y = 50/0.5
- height = n (pixels)
- topb=1 (pixels)
- bottomb, leftb, rightb, midb = 1
- color=FF00FF (rgb)
- fcolor=00FF00 (text)
- bcolor=000000 (borders)
- colspan, rowspan = n
- just="L/C/R"
- b, i, u (bold, italics, underline)
- coldefs
- rows, cols = n (grid size)
- id="mynode", id="mytag"
- onclick="mypage.sbtz"

Tags RT/Mono:
- current and child panels use Rich-Text mode, not monospace
- mono overrides rt

Mono Mode (cont'd):
- a
  - href="mypage.sbtz"
  - href="#label"
  - href="somepage.sbtz#label"
  - name="label"
  - onclick="(myfunc a b c)"
- img src width height left top alt borders-tag
  - src="myicon.jpeg"
- ch name/unicode
  - Sigma
  - epsilon
  - nbsp
  - x03c7
- just=0..2 (left/center/right)
- valign=0..2 (top/center/bottom)
- jy h=0..2 v=0..2
- select
  - pt expr (int or bool)
  - 1st, 2nd, 3rd, ...
- sbtk: Sharebitalk code

Scalability

All Sharebitalk data is stored in 256-byte pagelets or 4K array pages. All pages (except array pages) have 16 pagelets. All data of a given user is stored in the same 4 GB file on disk, and that file may contain data for more than one user. As many 4 GB files as can fit on one server may exist, per server. Server resources are finite and apps with high demand for those resources are subject to being throttled, limiting performance by an order of magnitude or more.

Any one server can have up to 4 GB of RAM devoted to Sharebitalk data encompassing one or more users. To resolve a 32-bit data address for a given user, the first byte indexes the user root table of up to 256 addresses. Each address in the user root table points to a user block table of 256 addresses. The second byte in the data address indexes the user block table. The indexed address points to a 64K block. The first nybble of the third byte in the data address points to the 4K page in the block. The second nybble of the third byte in the data address points to the 256-byte pagelet in the block. The fourth byte in the data address indexes the final data location within the pagelet. For array pages the least-significant 12 bits in the data address indexes a particular array element contained in that page.

Every 64K block in RAM contains a list of 16 page headers, and each page header is of size 8 bytes. This list replaces pagelet 0 of page 0 (page 0 is never an array page). The page header contains 2 bits: swapped-out and modified. If the page is swapped out, then the rest of the page header contains 20 bits pointing to the corresponding page in the 4 GB file on disk. If the page is not swapped out, then the rest of the page header contains 2 partial data addresses, each of size 20 bits. These partial data addresses point to the next and previous pages in RAM (whether or not the corresponding page is part of the free-page list). Whenever a page in RAM is accessed (read from or written to), it is moved to the head of this doubly linked list. Whenever a page in RAM needs to be swapped out, it is selected from the tail of the doubly linked list.

Sharebitalk

Sharebitalk is a Python dialect in which all operators precede their operands, and parentheses are used for all grouping (except string literals, which as in Python are delimited with single or double quotes). Sharebitalk code can be embedded in a Sharebitags text file. Sharebitags is similar to HTML, except open tags begin with a brace bracket and a keyword, and the closing tag is simply a close brace bracket. Any text enclosed in a tag is preceded by a vertical slash (|). File extensions include .SBTK (source code), .SBTA (assembler code), .SBTC (compiled code), and .SBTZ (Sharebitags).

Keyboard Aid

This optional feature enables hyphens, open parentheses, and close parentheses to be entered by typing semicolons, commas, and periods, respectively. When enabled, keyboard aid can be temporarily suppressed by using the Ctrl key in conjunction with typing semicolons, commas, and periods (no character substitution takes place). By convention, hyphens are used to separate words in multi-word identifiers, but semicolons are easier to type than hyphens. Similarly, commas and periods are easier to type than parentheses. When entering Sharebitags code, vertical slashes, open braces, and close braces are entered by typing forward slashes, commas, and periods, respectively. Typing semicolon converts previous hyphen to a semicolon, and previous semicolon to a hyphen (use the Ctrl key to override this behaviour). Typing semicolon after close parenthesis simply inserts semicolon.

Special Characters

`()`	grouping
`-_`	used in identifiers
`;`	end of stmt.
`:`	dot operator
`"'`	string delimiters
`\`	escape char.
`#`	comment
`{* *}`	block comment
`{ }`	Sharebitags code
`\*}`	treated as Sharebitags code not block comment

Differences from Python

- Parentheses, not whitespace
- Integration with Sharebitags
- Operators come before their operands
- Single, not multiple inheritance
- Adds interfaces ("scool" defs.)
- Drops iterators and generators
- Adds lambdas
- Adds quote and list-compile functions, treating code as data

Grammar Notation

- Non-terminal symbol: <symbol name>
- Optional text in brackets: [*text*]
- Repeats zero or more times: [*text*]…
- Repeats one or more times: <symbol name>…
- Pipe separates alternatives: *opt1* **|** *opt2*
- Comments in *italics*
- Advanced features flagged as **

Compiler and Assembler

The Sharebitalk Compiler translates source code into compiled code, and optionally into assembler code. Assembler code is an intermediate language which is much simpler than source code, although both source code and assembler code are in the form of text files. During Sharebitalk development, the developer uses the Sharebitalk Assembler, which converts assembler code (hand-written by the developer) into compiled code. This is a necessary step enabling the SharebiTalk Runtime Environment (STRE) to be tested prior to the development of the Sharebitalk Compiler.

Sharebitalk Grammar

White space occurs between tokens (parentheses need no adjacent white space, also any semicolon/hyphen before a close parenthesis may be omitted):

```
<source file>:
•   [<line-comment>] [<vars>] [ do <block>] [<dec
    def>]… [<class>]… [ do <block>]

<import stmt>:
    import <module>
    import ( <module>… )
    from <rel module> import <mod list>
    from <rel module> import all

<module>:
    <name>
    ( <name> as <name> )
    ( : <name>… [ as <name>] )

<mod list>:
    <id as>
    ( <id as>… )

<id as>:
    <mod id>
    ( <mod id> as <name> )

<mod id>:
    <mod name>
    <class name>
    <func name>
    <var name>

<rel module>:
    ( <colon list> [<name>]... )
?   <name>

<class>:
•   ( <cls typ> <name> [<base class>] [<does>]
    [<vars>] do <dec def>… )
•   ( scool <name> [<does>] do [<const decl>]...
    [<dec hdr>]... )

<cls typ>:
    class
    abclass

<does>:
    does ( <scool name>... )

<scool name>:
<base class>:
    <name>
    ( : <name><name>… )

<const decl>:
    const <name><const expr> ;

<dec hdr>:
    [<dec>]… <def hdr>

<dec def>:
    [<dec>]… <def>

<dec>:
    @ <call expr>

<def hdr>:
    def <var> ( [<var>]… )

<def>:
•   def <var> ( [<var>]… ) [<vars>] do <block>

<vars>:
    var ( <var>... )

<var>:
    ( <class name><id> )
    <id>

<block>:
    ( [<stmt-semi>]… [<stmt>] )

<stmt-semi>:
    <stmt><endchar>
    <sbtz tag>

<endchar>:
    ;

<jump stmt>:
    <continue stmt>
    <break stmt>
    <return stmt>

<pjump stmt>:
    return <expr>
    ** <raise stmt>

<raise stmt>:
    raise [<expr> [ from <expr>] ]
```

<stmt>:
    <open stmt>
    <closed stmt>

<open stmt>:
    <if stmt>
    <while stmt>
    <for stmt>
    ** <try stmt>
    <pjump stmt>
    <pcall stmt>
    <asst stmt>
    <del stmt>
    <import stmt>

<closed stmt>:
    <jump stmt>
    <call stmt>
    <print stmt>
    <sbtz tag>

<sbtz tag>:
    { **...** }

<call expr>:
- (<name> [<expr>]…)
- (**:** <obj expr> [<colon expr>]…
  (<method name> [<expr>]…))
- (call <expr>…)

<call stmt>:
- (<name> [<expr>]…)

<pcall stmt>:
- **:** <obj expr> [<colon expr>]…
  (<method name> [<expr>]…)
- call <expr>…

<colon expr>:
    <name>
    (<name> [<expr>]…)

<asst stmt>:
    <asst op><name><expr>
    <asst op><target expr><expr>

<asst op>:
    set | addset | minusset | mpyset | divset |
    idivset | modset | shlset | shrset |
    andset | orset | xorset

<target expr>:
    (**:** <name> [<colon expr>]… <name>)
    (slice <arr><expr> [<expr>])
    (slice <arr><expr> all)

<arr>:   *// string or array*
    <name>
    <expr>

<obj expr>:
    <name>
    <call stmt>

<if stmt>:
- if <expr> do <block> [elif <expr> do <block>]… [else <block>]

<while stmt>:
    while <expr> do <block>

<for stmt>:
    for <name> in <expr> do <block>

<try stmt>:
- try <block> <except clause>… [else <block>]
  [finally <block>]
- try <block> finally <block>

<except clause>:
    except <name> [as <name>] <block>

<return stmt>:
    return

<break stmt>:
    break

<continue stmt>:
    continue

<del stmt>:
    del <expr>

<paren stmt>:
    (<open stmt>)
    <closed stmt>

<qblock>:
    (quote [<paren stmt>]**...**)

```
<expr>:
      <keyword const>
      <literal>
      <name>
      ( <unary op><expr> )
      ( <bin op><expr><expr> )
      ( <multi op><expr><expr>… )
      ( quest <expr><expr><expr> )
      <lambda>
      ( quote <expr>... )
      <array expr>
      <dict expr>
      <bitarray expr>
      <string expr>
      <bytezero expr>
      <bytes expr>
      <target expr>
      <obj expr>
      <cast>

<unary op>:
      minus     negate
      notbits   bitwise not
      not

<bin op>:
      <arith op>
      <comparison op>
      <shift op>
      <bitwise op>
      <boolean op>

<arith op>:
      div | idiv | mod | mpy | add | minus

<comparison op>:
      ge | le | gt | lt | eq | ne | is | in

<shift op>:
      shl | shr

<bitwise op>:
      andbits | orbits | xorbits

<boolean op>:
      and | or | xor

<multi op>:
      mpy | add | or | and |
      strdo     % operator
      strcat    + operator

<const expr>:
      <literal>
      <keyword const>

<literal>:
      <num lit>
      <string lit>
      <bytes lit>

<array expr>:
      ( array [<expr>]… )

<bitarray expr>:
      ( bitarray <expr> )

<dict expr>:
      ( dict [<pair>]… )

<pair>:
      ( <name><expr> )
      ( <literal><expr> )

<cast>:
      ( cast <type><expr> )

<print stmt>:     // built-in func
      ( print [<expr>]… )
      ( echo [<expr>]… )

<lambda>:
      ( lambda ( [<id>]... ) <expr> )
      ( lambda ( [<id>]... ) do <block> )
      ( lambda ( [<id>]... ) do <qblock> )
      // must pass qblock thru compile func
```

No white space allowed between tokens, for rest of Sharebitalk Grammar (text omitted for brevity).