Mike Hahn – hahnbytes@gmail.com

Sharebitor

Sharebitor is a tool used by content publishers. The content lives in the Dropbox folders of the content publishers, and is made dynamic by apps written in an open source Python-like language called Sharebitalk (along with a markup language called Sharebitags). Content consumers search for content on the (closed source) Sharebitor.com website, and then link to the Dropbox folders of the content publishers. To view the content, the content consumer starts up the Sharebitor local web server and points her browser to localhost, after copying the content to her local drive. Sharebitalk is written in Java and works with a web container called Jetty.

Business Model

Subscribers pay $15 per year and are allowed to publish content (subscribers who are content consumers pay just $10 per year). App-writers pay no fees and receive all Google AdSense revenue when users click on ads displayed at Sharebitor.com. The ads appear on the app home screen or on those screens which include a link to the corresponding app. All basic apps are ranked according to the ratio of users who are subscribers to users who are non-subscribers. Apps in the topmost quartile (the most subscribers) receive a bonus equal to 100 percent of their AdSense revenue. Apps in the next quartile receive a bonus of 50 percent of that revenue. (Note that most Sharebitor users are non-subscribers, who pay no fees.)

Premium Apps

Premium apps (as opposed to basic apps) incur activation fees (or even additional app-specific subscription fees) and/or support in-app purchases. Only subscribers can use these apps, and pay the activation fees (and/or additional fees) to the app-writers. All in-app purchases require the input of a PIN number. If the PIN number is forgotten or needs to be changed, an email is sent to the account holder that includes a 5-digit code, which must be entered prior to resetting the PIN number. Both basic and premium apps have a corresponding subdomain, i.e. sample.sharebitor.com (so Pixtibil, the flagship app which is an image organizer, can eventually be found at pixtibil.sharebitor.com).

Cheap to Publish Content

For one low price of $15 per year, content publishers can publish as much content as they want, using as many apps as they want. Hosting is cheap, since the content lives in Dropbox folders, not on the web. For those content publishers who are coders, or who have access to coders, Sharebitor is a very inexpensive way to create dynamic web-oriented content. It's free to be an app-writer, and app-writers also receive Google AdSense revenue. Apps can be written in Sharebitalk or Java.

About the Name

Sharebitor rhymes with orbiter, except the first syllable is share instead of or. Sharebitalk and Sharebitags have a similar pronunciation to Sharebitor, except the final syllable replaces the "ter" sound with talk and tags, respectively. Sharebitor lets content publishers share their data (bits) with content consumers.

Conventional Websites

Ordinary (non-localhost) websites can be developed using Sharebitalk and Sharebitags, using 2 converters: Sharebitags-to-HTML and Sharebitalk-to-JavaScript. Server-side code is written in Sharebitalk.

Monospace Mode

In monospace mode, all body text rendered to the browsers of end-users in every Sharebitor 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.

Console Mode

User enters command-line in a text input field. Sharebitor 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 Sharebitor to cease execution.

Security

Any time Sharebitalk code (belonging to an app called "myappname") accesses the user's local drive, only relative path names are permitted. The path name (call it mypath) begins with "sharebitor/app/myappname/". The full path name begins with the path to wherever Sharebitor is installed, concatenated with mypath. Sharebitalk is the "secret sauce" of Sharebitor: users can download any and all apps without fear of malware, due to the above security feature built in to Sharebitalk. Sharebitor itself is a special app, written in Sharebitalk/tags like regular apps.

Implementation Steps

1. Project evaluation by Alastair, Sam Davis
2. Implement Sharebitalk Assembler
3. Implement SharebiTalk Runtime Environment (STRE)
4. Implement Sharebitalk Compiler
5. Implement Console mode
6. Implement Team Monitor
7. Cold email Toronto/Vancouver-based GitHub members who use Java
8. Expand previous step to US and Canadian GitHub members (optional)
9. Relocate to Vancouver (optional)
10. Begin reading book: Java for Web Applications
11. Implement Monospace mode
12. Implement Rich-Text mode
13. Implement Sharebitags-to-HTML converter
14. Implement Sharebitalk-to-JavaScript converter
15. Implement Java API: write Sharebitor apps in Java
16. Convert Pixtibil (image organizer) from Python to Java
17. Convert Pixtibil from Java to Sharebitalk
18. Pixtibil uses both Monospace and then Rich-Text modes
19. Pixtibil advanced features
20. Code editors: Sharebitalk and Sharebitags
    1. Eclipse plugins
    2. Syntax highlighting, unlimited undo/redo
21. Graphical (WYSIWYG) Sharebitags editor
22. Sharebitor website
    1. Dropbox support
    2. Search published content (indexed text)
    3. Cash flow: subscription fees, payments to app-writers/team members
23. Expand Sharebitalk libraries: widgets, graphics, games, math, etc.
24. Go live
25. Recruit beta testers
26. Purchase Google AdWords advertising
27. Support MMORPG functionality
28. Implement Android multiplayer game client

Sample Apps

Image collection organizer • blogs • e-commerce • shopping cart • catalogs • games • mmorpg • multiplayer Android games • lessons (education) • edutainment • team collaboration • clubhouses • self-publishing • *the sky's the limit!*

AdWords Phrases

1. image organizer
2. end user programming
3. new programming language
4. web programming language
5. app authoring system
6. app sdk
7. app software development kit
8. content publishing
9. dropbox addon
10. embedded web server
11. freemium app
12. make your own games
13. game development framework
14. web development framework

Our Team

Those who know Java and wish to participate in Sharebitor development are more than welcome to read up on our project and determine which aspects of Sharebitor development are of interest to them. Sharebitor 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 Sharebitor.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.

About Me

I am Mike Hahn, the founder of Sharebitor.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 math/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 Sharebitor 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	Country: Canada
Founder, Sharebitor.com	Phone: 416-533-4417
515-2495 Dundas St. West	Email: hahnbytes (AT) gmail (DOT) com
Toronto, ON M6P 1X4	Web: www.hahnbytes.com

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

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 JuppleTalk Runtime Environment (JTRE) 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).