Mike Hahn – hahnbytes@gmail.com

Jophrium

Jophrium (Jovelyst + Freemium) is an open source tool used to build freemium websites, and is implemented in Java. The freemium business model means most users pay nothing (for free) and a smaller number of subscribers pay fees in order to access premium features. Javrium is a sister tool, also open source, in which freemium websites are coded in Java. The .PDF document you are now reading contains exactly the same info as the rest of this website.

- Makes use of 2 new open source web programming languages: Jovelyst and Lystagger
- Jovelyst is similar to Python (all the freemium websites are coded in Jovelyst)
- Lystagger is similar to HTML
- Assume that a given freemium website is called "mysite"
- Domain name: mysite.jophr.net
- Sample open source freemium websites:

  1. jophspace.jophr.net: virtual spaces for marginalized communities
  2. jophpic.jophr.net: organize/share image folders
  3. jophnet.jophr.net: social network website
  4. jophteach.jophr.net: links tutors with students
  5. jophboard.jophr.net: make your own 2-player board games

- Jophteach and Jophboard are written up under Jophspace
- Subscribers pay $20/year (gold) or $12.50/year (silver)
- Silver members can only join at most 2 freemium websites at once, and can only switch websites once a week
- Gold members are members of all freemium websites
- Calculation of website resources consumed:
  - Let W = (no. of image file megabytes served) x (no. of kilo-nodes created)
  - Nodes are 12-byte chunks of RAM
  - Summed up for all users of a given website in one month
- Let H = cost of web hosting
- Websites where W is less than twice the median M are free: H = 0
- H = for W between 2 and 4 times M = $10/mo.
- W between 4 and 8 times M = $15/mo.
- W between 8 and 16 times M = $25/mo.
- W between 16 and 32 times M = $40/mo.
- W greater than 32,768 times M = $300/mo.
- W greater than 100,000 times M: website speed is throttled
- Every paying website gets a rebate (part of subscription fee revenue = F) proportionate to HR where R = A / B and A is less than or equal to B
- All rebates add up to F
- A = W for non-members only and
- B = W for members only or vice versa
- Purpose of R = A / B: motivate developers to create balanced websites, where population sizes of members and non-members are as equal as possible
- Allow financial transactions between developers and end-users using credit cards/PayPal
- Don't charge transaction fees
- No freemium website is allowed to directly compete with any of the 5 sample websites
- Transaction disputes (cheating):
  - Developer accused of cheating by 3 or more unique users in 90 days or less
  - User accused of cheating by 2 or more unique developers in 180 days or less
  - User/developer gets F rating for 2 years (allowed to appeal)
- Any Lystagger web page can have a corresponding HTML web page which contains converted HTML code and a link to its Lystagger page

factor = W / M	monthly fee = H	W / M	H	W / M	H
2	$10	64	$80	2048	$200
4	15	128	100	4096	225
8	25	256	125	8192	250
16	40	512	150	16,384	275
32	60	1024	175	>32,768	300

Javrium

- Sister tool of Jophrium
- All freemium websites coded in Java, JavaScript and HTML
- Assume that a given freemium website is called "mysite"
- Domain name: mysite.jophr.net
- Sample open source freemium websites:

  1. jpic.jophr.net: organize/share image folders
  2. jteach.jophr.net: links tutors with students
  3. jboard.jophr.net: make your own 2-player board games

- Subscribers pay $20/year (gold) or $12.50/year (silver)
- Silver members can only join at most 2 freemium websites at once, and can only switch websites once a week
- Gold members are members of all freemium websites
- Calculation of website resources consumed:
  - Let W = (no. of image file megabytes served) x (no. of tokens in Java source code)
  - Comments and the following chars. ignored: ( ) { } [ ] ;
  - Constants: numeric, string, character count as one token
  - Summed up for all users of a given website in one month
- Let H = cost of web hosting
- Websites where W is less than twice the median M are free: H = 0
- H = for W between 2 and 4 times M = $10/mo.
- W between 4 and 8 times M = $15/mo.
- W between 8 and 16 times M = $25/mo.
- W between 16 and 32 times M = $40/mo.
- W greater than 32,768 times M = $300/mo.
- W greater than 100,000 times M: website speed is throttled
- Every paying website gets a rebate (part of subscription fee revenue = F) proportionate to HR where R = A / B and A is less than or equal to B
- All rebates add up to F
- A = W for non-members only and
- B = W for members only or vice versa
- Allow financial transactions between developers and end-users using credit cards/PayPal
- Don't charge transaction fees
- No freemium website is allowed to directly compete with any of the 3 sample websites
- Transaction disputes (cheating):
  - Developer accused of cheating by 3 or more unique users in 90 days or less
  - User accused of cheating by 2 or more unique developers in 180 days or less
  - User/developer gets F rating for 2 years (allowed to appeal)

Revenue and Expenses

- Let Q = no. of freemium projects
- Assume Q = 20
- Let q = no. of freemium projects who pay hosting fees
- Assume q = Q x 30 percent = 6
- Let s = no. of silver members/project
- Assume s = 20
- Divide by 2, assuming each silver member belongs to 2 projects
    - Then no. of silver members = Qs / 2 = 20 x 20 / 2 = 200
    - Assume user conversion rate = 5 percent
    - Let U = no. of users
    - Then U = 200 / 5 percent = 4000
    - F = subscription fees total = 200 x 12.5 = $2500/year
    - Let H = web hosting fees total
    - Let h = avg. hosting fee per project per month
    - Assume h = $20
    - H = 12qh = 12 x 6 x 20 = $1440/year
    - Let N = net amt. paid to each project
    - Then N = (F - H) / q
    - N = (2500 - 1440) / 6
    - N = $177/year
    - Let N = (KF - H) / q
    - Assume N = 0
    - H = KF
    - K = H / F
    - K = 1440 / 2500 = 58 percent
    - Let V = revenue
    - V = F - H
    - V = 2500 - 1440 = $1060/year
- Assume Q = 100
- Assume q = Q x 30 percent = 30
- Assume s = 40
- Divide by 2, assuming each silver member belongs to 2 projects
    - Then no. of silver members = Qs / 2 = 100 x 40 / 2 = 2000
    - Assume user conversion rate = 5 percent
    - Let U = no. of users
    - Then U = 2000 / 5 percent = 40,000
    - F = subscription fees total = 2000 x 12.5 = $25,000/year
    - Assume h = $20/month
    - H = 12qh = 12 x 30 x 20 = $7200/year
    - Let N = (KF - H) / q
    - Assume N = 0
    - K = H / F = 7200 / 25,000 = 29 percent
    - V = F - H = 25,000 - 7200 = $17,800/year
    - Let E = expenses
    - E = Google AdWords cost + web hosting
    - E = 3600 + 7500 = $11,100/year
    - Let P = profit
    - P = V - E
    - P = 17,800 - 11,100 = $6700/year

Jophpic

Jophpic.com is a tool (one of the 5 sample websites, also located at jophpic.jophr.net) which lets you organize and share your image folders. It makes use of 2 open source third-party tools: an embedded web server called Jetty, and a text search engine called Lucene. Users must first launch the Jophpic web launcher and then point their web browsers to http://localhost:6886/. All image files are stored on the user's local hard drive. Jophpic makes use of a markup language (simplified HTML) called Lystagger.

Qpic Folders

A qpic is a queue of image files contained in a folder, and qpic folders can contain other qpic folders recursively. Newly added image files go to the head of the queue in each qpic. Every qpic contains a special queue which is a subset of the main queue of images. Both the main and the special queues support image reordering commands: head, tail, move left, move right. The head of the queue is the leftmost image in the queue. Every qpic has 2 yes/no flags: the H-flag and the X-flag. H stands for hidden (not shared publicly) and the X-flag warns users that its images are such that if the user is at work or sitting at a public computer, then proceed with caution. By default, qpics having an X-flag value of yes are hidden from the user.

Business Model

Jophpic no-name users pay no fees. Jophpic members must be Jophrium members. All users can share qpics with other users by emailing links. All users can perform text-based searches, based on image captions, slide panels, and qpic names/descriptions. All users can browse non-hidden qpics of members without restrictions. All images contained in qpics of no-name users can be browsed, but any sub-qpics of a given no-name user's qpic are not displayed. All search results returned belonging to no-name users are stripped of all text: image captions, qpic names/descriptions, and user names. All of that same text information, including user names, is not displayed when browsing images in any qpic of a no-name user.

Commands

- **Enter** - down a level
- **Up Arrow** - up a level
- **Left/Right Arrow** - previous/next
  - qpic/screen/image/slide
- **Down Arrow** - toggle main/special
- **Ctrl+Down Arrow** - toggle slide mode
- **1** - first
- **0** - last
- **Shift+Up Arrow** - move to top
- **Shift+Left Arrow** - move left
- **Shift+Right Arrow** - move right
- **Shift+Down Arrow** - move to bottom
- **L** - like image: make it special
- **U** - undo L command
- **I** - insert image/qpic
- **D** - delete (slide mode)
- **Ctrl+D** - delete image/qpic
- **Ctrl+X** - cut qpic
- **Ctrl+C** - copy qpic
- **Ctrl+V** - paste qpic
- **Ctrl+N** - new qpic
- **Ctrl+R** - rename qpic
- **J** - justify (left, center, right)
- **B** - bookmark qpic/access bookmarks
- **H** - hidden qpic on/off
- **X** - X-flag qpic on/off
- **Y** - sync qpic
- **S** - search
- **Q** - quit
- **F1** - cycle: menu/help/normal
- **F11** - fill screen
- **[x]** - close menu bar

Folders Mode

Displays parent folder name followed by an indented list of folder names. Current folder is highlighted (enclosed in square brackets). Folder properties: name, description, img-flag, H-flag, X-flag. If img-flag is false, folder contains no images, only other folders. By convention, images stored in non-image folders reside in a sub-folder called "$".

Tiles Mode

Whenever the user is in Folders Mode and presses Enter when an image folder is highlighted, the current mode becomes Tiles Mode. The display is divided into 3 rows of equal height (or n rows where n > 1). Each row contains images. All portrait-mode images are of equal height but of varying widths. Every landscape-mode image is the same width as the height of the row which contains that image. All images are separated by a white, one-pixel gap (user may increase pixel count of gap, globally). Clicking on an image will display it in Image Mode. Pressing Up Arrow makes the current mode become Folders Mode.

Image Mode

Single image is displayed, expanded by the maximum amount available on the user's display. Pressing L or U modifies special flag if needed and the current mode becomes Tiles Mode. Pressing Up Arrow makes the current mode become Tiles Mode. Pressing Enter enables editing of one-line caption, and/or toggling display of image captions, and/or toggling the filtering out of images which lack captions. If this image is accompanied by a link to a video, click on play to follow that link under a new browser tab. The image-view count of the user who is the image owner is incremented, if different from the user viewing the image.

Slide Mode

Press Ctrl+Down Arrow to toggle between Tiles Mode and Slide Mode. Slide Mode displays between 1 and 3 images per slide. Each slide tries to fill the entire display. Click on an image in slide mode to enter image mode, then press D to delete the image from the slide. In slide mode, press I to insert an image. The next time the I command is used in image mode, the image is inserted into the slide. Various image arrangements (side-by-side, stacked, or some combination) are used automatically, depending on the aspect ratios of the images on the slide. New slides can only be inserted at the head of the slide list. One of the panels on a given slide (which contains 1 to 3 panels) can contain Lystagger code instead of an image.

Image Folder Sharing

Google Drive (or possibly Dropbox) will be used as the image sharing platform. Users can use Jophpic to keep track of their favorite image-sharing users and the names of their favorite image folder names shared by those users.

Bookmark/Insert Commands

The Insert command displays the select-user web page. After selecting a user, the Local mode switch is off. Further Insert commands insert images/folders into the current local folder, and allow the user to change the current local folder. The Quit command sets the Local mode switch back to on. The Bookmark command bookmarks the current folder when Local mode is off (enabling bookmark parent list selection), and accesses the bookmark tree when Local mode is on.

Searching

Popularity is used in searches: how many times an image/folder has been viewed, downloaded, or bookmarked. Searching is used to search for users and the images held by those users. Text searches involve qpic names/descriptions, image captions, and slide panels.

Image File Names

Newly added image files can have arbitrary file names. After the sync command is used, newly added image files (except for the least recently added image, which is manually appended with .1 when saved by the user) are renamed to $QNNNN.ext, where ext is a graphics file type such as PNG or JPEG and NNNN is a 4-digit number. Clock arithmetic is used, where 9999 corresponds to -1, 9998 corresponds to -2, and so on. Usually, the head of the queue is positive, and the tail of the queue is 0 or negative. No qpic can have more than 10,000 image files.

The sync command programmatically removes the .1 extension of the least recently added image. When the user goes to add the next batch of newly added images, and tries to save the same image file F that originally had the .1 extension, it will already exist in the current qpic. This will let the user know when to stop adding images. The next time the sync command is performed, image file F is renamed to have the same $QNNNN format as all of the older image files in the current qpic.

Jophnet

Jophnet.com (one of the 5 sample websites, also located at jophnet.jophr.net) is an innovative social network freemium website combined with a new markup language called Lystagger. Each user belongs to one or more nodewicks. Each nodewick corresponds to a different Wikipedia article of the same name. Users can select an existing nodewick or search Wikipedia and create a new one (based on an existing Wikipedia article). All posts are either public or private. Private posts only go out to the followers of the user who published the post. Public posts are searchable by anyone. All posts and followers are filtered by the current nodewick. So a user can have different sets of followers for each nodewick they belong to. Jophnet search capabilities are handled using Lucene.

Organizations

Each member organization belongs to one or more nodewicks. Employees, customers, and clients (collectively called "members", they must all be subscribers) of the organizations use Jophnet as social networking glue, facilitating their interactions with each other and the organizations which serve them. Many organizations will employ one or more moderators to oversee their members, dealing with trolls and making sure that Jophnet remains a safe space for the members.

Posts and Followers

Whenever a user publishes a post (which may have an attached image/link/video), it can be public (the default), or just for followers. Users can search for public posts by entering keywords, similar to a search engine, and the top 10 search results are displayed. If the user clicks on a search result, that user can optionally follow the user who published the post. Once a user follows another user, both users can send each other private messages, and any private message can be copied to an arbitrary no. of users who follow or are followed by the user sending the message. Any user is free to unfollow another user or permanently/temporarily block a following user at any time. Any follower of the user who published the post is allowed to comment on that post. Anyone, not just followers, can like/dislike a post. Any user is free to enter the email of a friend, which automatically sends an email to that friend inviting her or him to join Jophnet.

Nodewick Graph

Every time a user performs a global search (no nodewick selected), the Nodewick Graph is potentially modified. Only the top 6 search results matter (rank N = 0 to 5). The connection strength c between any 2 search results, in which the associated nodewicks are unequal, is equal to G^(M+N), where G = 0.618 = the golden mean, both M and N are the ranks, and the caret (^) means raise to a power. All values of c are summed for a given pair of unequal nodewicks. The lowest values of M and N are plugged in to get c, then the next lowest values are plugged in to get the next value of c, and so on, and all of the c values are summed. So for any given pair of nodewicks, the no. of c values equals the minimum of 2 quantities: the no. of occurrences of both the first and second nodewicks in the search results. The sum of the c values equals C. If neither nodewick was clicked on, the C value is ignored.

In order for a C value to matter, the same nodewick pair must come up in at least 2 searches performed by a given user. If so, then C is added to the global konnection strength K associated with that nodewick pair. Whenever a user wishes to display an ordered list of nodewicks related to the current nodewick, the associated global K value is used to rank the nodewicks, in descending order. Also, only nodewick pairs which come up in 2 searches made by a single user, and that is repeated for at least 3 users in the previous 12 months, are displayed in the ordered list of nodewicks.

Nodewick Selection

Users can select any one of their nodewicks at any given time, which filters posts/followers by the selected nodewick. Users can join new nodewicks or drop out of old ones at any time, and can also create new nodewicks (which must be valid Wikipedia articles). A robot is used to crawl the Wikipedia website upon creation of a new nodewick. If the nodewick name is, for example, mynodewick, then the robot visits the following URL:

- https://en.wikipedia.org/wiki/Special:WhatLinksHere/mynodewick

The robot scans that web page for the following piece of text: "No pages link to mynodewick". If that piece of text is found that usually means that mynodewick is not a valid Wikipedia article and the user is prevented from adding mynodewick to the Nodewick database table.

Business Model

Subscribers may be employees, customers, and clients of member organizations. Subscribers also have the ability to embed Lystagger code in their posts. Non-subscribers can only enter plain text, blank lines, and hypertext links using the Lystagger "a" tag. An example of that tag is as follows:

[a (link www.mysite.com): Click here]

Directory Structure

- User Name: johndoe
- Home: jophnet.com/u/johndoe/
- Mike's named posts: <home>/YYYY/MM/DD/title-of-post.lstg
- Mike's unnamed posts: <home>/YYYY/MM/DD/HH.MM.SS.lstg

Sample Organizations

Progress Place belongs to the following nodewicks: Clubhouse Model of Psychosocial Rehabilitation, Mental health, and Mental disorder. CAMH belongs to Centre for Addiction and Mental Health, Mental health, Mental disorder, Substance dependence, and Psychiatric hospital. Fred Victor belongs to Poverty reduction, Subsidized housing, and Employment counsellor. West Neighbourhood House belongs to Poverty reduction, Tutoring agency, and Elderly care. All 4 sample organizations belong to Nonprofit organization and Social work. Progress Place will be the first member organization to act as a testbed for Jophnet.

Lystagger

Lystagger is a simplified markup language used to replace HTML. Arbitrary Lystagger code can be embedded in the Jovelyst echo statement. Lystagger syntax, where asterisk (*) means repetition, is defined as follows:

- Tags:
  - [tag]
  - [tag: body]
  - [tag (fld val)*: body]
- Body:
  - text
  - [: text]*
  - [(fld val)*: text]*
- Call Jovelyst code:
  - [expr: <expr>]
  - [exec: <stmt>...]
  - [lyst: <path>]

Monospace Mode

In monospace mode, all body text rendered to the screens of end-users 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. In both rich-text and monospace modes, text is rendered to the HTML5 canvas object. Some features like form fields and submit buttons use hidden HTML.

Implementation Steps

1. Read Murach's Java Servlets and JSP book
2. Implement Jabbler: web-based HTML5 Scrabble game, user vs. robot
    - Jabbler is currently console-based Java Scrabble game
3. Write basic Lystagger design specs
4. Implement Jovelyst 0.1, console-based
    - Token parsing and building program tree has already been implemented
5. Finish Jovelyst 1.0, console-based
6. Write advanced Lystagger design specs
7. Implement LSTG-to-HTML converter
8. Implement monospace mode
9. Implement rich-text mode
10. Integrate Jovelyst with Lystagger (monospace/rich-text modes)
11. Implement LYST-to-JS converter
12. Recruit GitHub open source coders/testers
13. Approach Progress Place
14. Implement Jophspace
15. Design non-commercial (free) website
16. Launch website
17. Beta test Jophspace
18. Implement Jophrium
19. Beta test Jophrium
20. Implement Jabbler: web-based, 2-player
21. Implement monospace mode, dual user
22. Implement rich-text mode, dual user
23. Implement volunteer Jophteach
24. Beta test volunteer Jophteach
25. Implement paid Jophteach
26. Implement Jophboard
    1. Integrate Jabbler as first Java-based sample board game
    2. WYSIWYG board/piece editor
    3. Codeless prototyping system
    4. Beta test
    5. Implement Jovelyst code editor
    6. Integrate with board editor/prototyping system
27. Implement Jophpic
28. Design commercial website
29. Launch commercial website
30. Beta test Jophpic
31. Beta test paid Jophteach
32. Accept credit card payments
33. Implement Jophnet
34. Beta test Jophnet
35. Hire Java programmer with expertise in making websites scalable
36. Hire translators to internationalize Jophrium

Jophspace

Jophspace.com is a website (one of the 5 sample websites, also located at jophspace.jophr.net) which features virtual spaces for marginalized communities. Every non-profit organization whose clients belong to various marginalized communities pays Jophrium.com the sum of $1.00 per client per year, billed quarterly. The virtual spaces include forums and chat rooms, and clients can maintain blogs (using Lystagger, not just plain text). Forums can be organization-specific, or population-specific. The initial target population is for consumer/survivors, or individuals with mental health issues. Progress Place, a clubhouse for consumer/survivors, will act as a testbed for Jophspace. Clubs having fewer than 25 members can use Jophspace for free.

Jophteach

Jophteach.com is a website (one of the 5 sample websites, also located at jophteach.jophr.net) which links tutors with students. Some tutors charge their students an hourly rate, and Jophrium receives 10 percent of that revenue stream. Non-members can only make use of volunteer tutors. The tutors teach math, literacy, and coding. A web-based interactive whiteboard enables the tutor to interact with a single student. The tutor and student take turns interacting with the whiteboard. At the beginning of each turn, every move (mouse clicks and text entered during the previous turn) is replayed, and then further interaction takes place. Prefabricated lessons are prepared by volunteer curriculum-writers and displayed on the interactive whiteboard.

Jophboard

Jophboard.com is a website (one of the 5 sample websites, also located at jophboard.jophr.net) where you can play 2-player non-animated games: think board games and card games. You can also create your own Jophboard games using Jovelyst and Lystagger.

Jophboard Members

Jophboard members (who must be Jophrium members) can create home pages/bios written in Lystagger, post in forums, view games in progress, participate in tournaments, and hold player rating values for each Jophboard game they are involved with. An example of a player rating value is a chess rating, where a rating of 1700 or more would be held by a very skilled chess player. The "Outer Forum" is a special forum used only by non-members, who have read/write access to that forum.

About Me

I am Mike Hahn, the founder of Jophrium.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. A few months prior to starting my Jophrium project 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, Jophrium.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

Jovelyst

Jovelyst is a Python dialect in which all operators precede their operands, and parentheses are used for all grouping (except string literals, which are delimited with double quotes). Jovelyst code is often accompanied by Lystagger screen definition files. Lystagger is similar to HTML, except open tags begin with an open square bracket and a keyword, and the closing tag is simply a close square bracket. Any text enclosed in a tag is preceded by a colon. File extensions include .LYST (Jovelyst) and .LSTG (Lystagger).

Special Characters

- `( )` grouping
- `-` used in identifiers
- `;` end of stmt.
- `:` dot operator
- `"` string delimiter
- `\` escape char.
- `#` comment
- Extra:
  - `_` used in identifiers
  - `$` string prefix char.
  - `*` public switch
  - `{ }` block comment

Version 0.1

- No inheritance
- No interfaces
- No IDE
- No rich text

Differences from Python

- Parentheses, not whitespace
- Integration with Lystagger
- Operators come before their operands
- Information hiding (public/private)
- 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>
- Optional text in brackets: [*text*]
- Repeats zero or more times: [*text*]…
- Repeats one or more times: <symbol>…
- Pipe separates alternatives: *opt1* **|** *opt2*
- Comments in *italics*

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. 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. The close delim switch automatically inserts a closing parenthesis/double quote when the open delimiter is inserted.

Jovelyst Grammar

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

```
<source file>:
•     [<use>] [ * <vars>] [<vars>] [ do <block>]
      [<def>]… [<class>]… [ do <block>]

<use>:
      use ( <import-semi>... )

<import-semi>:
      <import-stmt> ;

<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>:
      ( : [<num>] [<name>]... )
      <name>   // ?

<class>:
•     ( [ * ] <cls typ><name> [<base class>] [<does>]
      [ * <vars>] [<vars>] <def>… )
•     ( [ * ] scool <name> [<does>] [<const list>]
      [<def hdr>]... )
•     ( [ * ] enum <name><elist> )

<cls typ>:
      class
      abclass

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

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

<const list>:
      const ( <const pair>... )

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

<def hdr>:
      ( <defun><name> ( [<parms>] ) [<dec>] )

<def>:
•     ( [ * ] <defun><name> ( [<parms>] ) [<vars>]
      [<dec>] do <block> )

<defun>:
      def
      abdef

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

<parms>:
      <parm>... [ ( * <id> ) ] [ ( ** <id> ) ]

<parm>:
      <id>
      ( tuple <id>... )
      ( <set op><id><expr> )
      ( <set op> ( tuple <id>... ) <expr> )

<dec>:
      decor <call expr>...

<block>:
      ( [<stmt-semi>]… )

<stmt-semi>:
      <stmt> ;
```

```
<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>

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

<call expr>:
•   ( <name> [<arg list>] )
•   ( : <obj expr> [<colon expr>]…
    ( <method name> [<arg list>] ))
•   ( call <expr> [<arg list>] )

<call stmt>:
    ( <name> [<arg list>] )

<pcall stmt>:
•   : <obj expr> [<colon expr>]…
    ( <method name> [<arg list>] )
•   call <expr> [<arg list>]

<colon expr>:
    <name>
    ( <name> [<arg list>] )

<arg list>:
    [<expr>]... [ ( <set op><id><expr> ) ]...

<asst stmt>:
    <asst op><target expr><expr>
    <set op> ( tuple <target expr>... ) <expr>

<asst op>:
    set | addset | minusset | mpyset | divset |
    idivset | modset |
    shlset | shrset | shruset |
    andbset | xorbset | orbset |
    andset | xorset | orset |
    = | += | -= | *= | /= |
    //= | %= |
    <<= | >>= | >>>= |
    &= | ^= | '|=' |
    &&= | ^^= | '||='

<set op>:
    set | =

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

<arr>:        // string or array/lyst
    <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>
    do <block> while <expr>

<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>] do <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>...)
    <renum expr>
    <tuple expr>
    <lyst expr>
    <dict expr>
    <bitarray expr>
    <string expr>
    <bytezero expr>
    <bytes expr>
    <target expr>
    <obj expr>
    <cast>

<quest>:
    quest | ?

<unary op>:
    minus | notbitz | 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 | shru |
    << | >> | >>>

Note: some operators delimited with single quotes for clarity (quotes omitted in source code)

<bitwise op>:
    andbitz | xorbitz | orbitz |
    & | ^ | '|'

<boolean op>:
    and | xor | or |
    && | ^^ | '||'

<multi op>:
    mpy | add | strdo | strcat |
    and | xor | andbitz | xorbitz |
    or | orbitz |
    * | + | % | + |
    && | ^^ | & | ^ |
    '||' | '|'

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

<literal>:
    <num lit>
    <str lit>
    <bytes lit>

<tuple expr>:
    (tuple <expr>…)

<lyst expr>:
    (lyst [<expr>]…)

<dict expr>:
    (dict [<pair>]…)

<bitarray expr>:
    (bitarray <enum name> [<elist>])
    (bitarray <enum name><idpair>...)

<elist>:
    <id>...
    <intpair>...
    <chpair>...

<intpair>
    // integer constant
    <int const>
    (<int const><int const>)

```
<chpair>
      // one-char. string
      <char lit>
      ( <char lit><char lit> )

<idpair>
      <idt>
      ( <id><id> )

<pair>:
      // expr1 is a string
      ( <expr1><expr2> )
      ( <str lit><expr> )

<renum expr>
      ( renumize <expr><ren id>... )
      ( renumize <expr><ren int>... )
      ( renumize <expr><ren ch>... )

<ren id>:
      ( 0 <id> )
      ( 1 <id> )
      ( 1 <id><id> )

<ren int>:
<ren ch>:
      // expr is <dec int> | <char lit>
      ( 0 <expr> )
      ( 0 <expr><expr> )
      ( 1 <expr> )
      ( 1 <expr><expr> )

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

<print stmt>:      // built-in func
      ( print <expr>… )
      ( println [<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 Jovelyst Grammar

```
<white space>:
      <white token>...

<white token>:
      <white char>
      <line-comment>
      <blk-comment>

<line-comment>:
      # [<char>]... <new-line>

<blk-comment>:
      { [<char>]... }

<white char>:
      <space> | <tab> | <new-line>

<name>:
•     [<underscore>]… <letter> [<alnum>]…
      [<hyphen-alnum>]… [<underscore>]…

<hyphen-alnum>:
      <hyphen><alnum>…

<alnum>:
      <letter>
      <digit>
```

In plain English, names begin and end with zero or more underscores. In between is a letter followed by zero or more alphanumeric characters. Names may also contain hyphens, where each hyphen is preceded and succeeded by an alphanumeric character.

```
<num lit>:
      <dec int>
      <long int>
      <oct int>
      <hex int>
      <bin int>
      <float>

<dec int>:
      [<hyphen>] 0
      [<hyphen>] <any digit except 0> [<digit>]…

<long int>:
      <dec int> L
```

```
<float>:
    <dec int><fraction> [<exponent>]
    <dec int><exponent>

<fraction>:
    <dot> [<digit>]…

<exponent>:
    <e> [<sign>] <digit>…
<e>:
    e | E

<sign>:
    + | -

<keyword const>:
    none
    true
    false

<oct int>:
    0o <octal digit>…

<hex int>:
    0x <hex digit>…
    0X <hex digit>…

<bin int>:
    0b <zero or one>…
    0B <zero or one>…

<octal digit>:
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

<hex digit>:
    <digit>
    A | B | C | D | E | F
    a | b | c | d | e | f

<string lit>:
    [ $ <str prefix>] <short long>

<str prefix>:
    r | u | R | U

<short long>:
    " [<short item>]... "
    " " " [<long item>]... " " "

<short item>:
    <short char>
    <escaped str char>

<long item>:
    <long char>
    <escaped str char>

<short char>:
    any source char. except "\", newline, or
    end quote

<long char>:
    any source char. except "\"

<bytes lit>:
    $ <byte prefix><shortb longb>

<byte prefix>:   // any case/order
    b | br

<shortb longb>:
    " [<shortb item>]... "
    " " " [<longb item>]... " " "

<shortb item>:
    <shortb char>
    <escaped char>

<longb item>:
    <longb char>
    <escaped char>

<shortb char>:
    any ASCII char. except "\", newline, or
    end quote

<longb char>:
    any ASCII char. except "\"

<escaped char>:
    \newline
        ignore "\", newline chars.
    \\  backslash
    \"  double quote
    \}  close brace
    \a  bell
    \b  backspace
    \f  formfeed
    \n  new line
    \r  carriage return
    \t  tab
    \v  vertical tab
    \ooo    octal value = ooo
    \xhh    hex value = hh

<escaped str char>:
    <escaped char>
    \N{name}    Unicode char. = name
    \uxxxx      hex value (16-bit) = xxxx
```