Mike Hahn - hahnbytes@gmail.com

Uniqrome

Uniqrome is a language used for developing Android and iOS smartphone apps. All Uniqrome users must install the free QRE app: uniQrome Runtime Environment. Uniqrome users fall into 3 categories: bronze, silver and gold. Bronze users pay no fees but their display is limited to shades of gray and a single hue. This hue can only be used for rectangles (with or without rounded corners) and/or text. Silver users pay $10/year, and gold users pay $10/month. Gold users can use premium apps having the most bandwidth (top 10 percent of GBs downloaded and uploaded). Silver users can only use premium apps in grayscale mode (shades of gray). Uniqrome for Windows/Mac/Linux is free.

Paying Out User Fees

App authors fall into 5 groups according to the no. of user tap events (up or down) generated by the app users. Each app in the bottom 30 percent receives the same amount, this amount is called the baseline. Each app in the next-to-lowest 20 percent receives twice the baseline. Each app in the middle 20 percent receives 5 times the baseline. Each app in the next 20 percent (next-to-highest) receives 10 times the baseline. Each app in the top 10 percent receives 20 times the baseline. During the first year after a given user first installs the QRE, Google or Apple collects 30 percent of the fees paid by that user. After one year that amount drops to 15 percent. Uniqrome.com collects the remaining 70 or 85 percent respectively, keeping 30 or 15 percent, passing on 40 or 70 percent to the app authors. App authors are allowed to charge in-app fees to the users, Uniqrome.com does not get a cut of those fees. Only bronze Uniqrome users see ads.

App Throttling

Users of apps with high bandwidth requirements (GBs downloaded and uploaded) may experience throttling, which involves longer waits when data is downloaded from or uploaded to the server. This is necessary to discourage app authors from making unreasonably high demands on server resources. Those app authors who have high bandwidth requirements can always purchase extra bandwidth to avoid throttling. Server-side code is converted from Uniqrome to Javascript at the time it is uploaded.

Extensible Apps

The secret sauce of Uniqrome is twofold: ads are optional and app authors have the option of making their apps extensible. The app extensions are written by end-users with knowledge of programming. They work similar to how Wordpress websites can be customized using child themes and plugins. The QRE is bundled with an app browser, which browses Uniqrome apps when the user enters the app name into the address bar. Simple apps contain nothing but static Qromitags code along with client-side Uniqrome code. Qromitags is a text markup language which works with Uniqrome.

Exit Strategy

In case Uniqrome is unprofitable, existing app authors will be given at least 30 days notice, allowing them to download their database files. Uniqrome would then become open source, and app authors can download the QRE from GitHub and bundle it with their Uniqrome and Qromitags code. Google or Apple would then provide web hosting directly to the app authors.

Qromiteach

Qromiteach.com is the home of Qromiteach: a tool used for teaching various subjects, including such STEM subjects as math and coding, and is implemented in Java. Qromiteach will be developed after Uniqrome.com is launched.

Implementation Steps

1. Develop foundation of Uniqrome code execution - ***almost done!***
2. Develop rest of Uniqrome code execution
3. Release Uniqrome as console-based compiler on GitHub
4. Implement GUI: monospaced mode
5. Release Uniqrome/GUI on GitHub
6. Write Qromitags design specs
7. Develop Qromitags
8. Integrate Uniqrome with Qromitags
9. Uniqrome/Qromitags: uniQrome Runtime Environment (QRE)
10. QRE with full GUI is closed source
11. Hire Java programmer who is on autism spectrum, as co-founder
    - Use Specialisterne, they find IT jobs for people on spectrum
12. Make pitch to DMZ tech incubator at Ryerson
13. If pitch is successful, start paying co-founder, otherwise skip to Step 18.1
14. Search for angel investor
15. Co-founder to port Uniqrome console-based compiler to Android
16. Co-founder to port QRE to Android
17. Mike to lead development of Uniqrome SDK:
    1. Develop Uniqrome code editor
    2. Expand code editor to Uniqrome SDK
    3. Develop monetizing functionality
    4. Release Uniqrome SDK
18. Upon failure of angel investor search:
    1. Co-founder is laid off
    2. Mike to develop rest of project, excluding iOS QRE
19. Otherwise:
    1. Convert Android version of QRE to Swift
    2. Hire Swift/Java programmer to develop iOS QRE
20. Develop Uniqrome-to-Javascript converter
21. Launch website
22. Launch Android QRE
23. Launch iOS QRE
24. Purchase Google AdWords advertising
25. Develop Qromitype
26. Launch Qromiteach.com
27. Develop Qromidesq
28. Implement Keyboard Aid (bells and whistles of editor)
29. Develop WYSIWYG Qromitags screen editor
30. Implement optional Qromitags-to-HTML converter

Revenue

Assume 100,000 Uniqrome users exist, including 3000 gold users, 10,000 silver users, and the rest, 87,000 bronze users. Annual gross revenue from user fees equals 3000 x 120 + 10,000 x 10 = 360,000 + 100,000 = $460,000. Assume Google/Apple receive 20%, the House receives 20%, and the app authors receive 60%. So annual net revenue equals 460,000 x 0.2 = $92,000. Since this amount is too small, assume the user count is doubled, to 200,000, and annual net revenue is $184,000. The app authors receive 60% of $920,000 = $552,000. Assume that each bronze user generates $2/year in ad revenue. Then ad revenue equals 87,000 x 2 x 2 = $348,000. So total net revenue equals 184,000 + 348,000 = $532,000.

Assume 100 app authors exist. Let b = amount received by each app author in the bottom tier. The top tier of app authors receive 10 x 20b = 200b. The 2nd tier receives 20 x 10b = 200b. The middle tier receives 20 x 5b = 100b. The 4th tier receives 20 x 2b = 40b. The bottom tier receives 30b. The sum of all tiers = 570b. Therefore b = 552,000 / 570 = $968. So the top tier of app authors receive 200b, and each app author in the top tier receives 20b = $19,368. Assume 500 app authors exist. Then b = 968 / 5 = $194, and each app author in the top tier receives $3,874.

Handwritten Uniqrome Design

The following 18 lines of text rendered in a monospaced font were originally handwritten with a ballpoint pen on a piece of notepad paper, dimensions: 4.25" x 5.5". Just before I wrote this down, I dreamed up the core idea for Uniqrome in the wee early morning hours of Monday, December 5th, 2022. I was deeply depressed all day Saturday, but it lifted on Sunday morning. Sunday afternoon I spent a lot of time reading the CBC news website as I often do.

```
-----------------------------------------------
|    bronze:  $0: mono + 1 hue                |
|    silver:  $10/yr                          |
|    gold:    $10/mo: top 10% of apps (GBs)   |
|    metric:  user tap down/up events         |
|  _                                          |
| /  Apple/Google: 30% - 15%                  |
|<   Me:           30% - 15%                  |
| \_ Apps:         40% - 70%                  |
|                                             |
|   5 sections:                               |
|   1. Top 10%   x 20 = 200                   |
|   2. Next 20%  x 10 = 200                   |
|   3.  "         x 5 = 100                   |
|   4.  "         x 2 =  40                   |
|   5. Bottom 30% x 1 = _30                   |
|                       570                   |
|   Throttle users of apps (GB-hogs):         |
|   less CPU time of server-side code         |
-----------------------------------------------
```

Qromiteach

Qromiteach.com is the home of Qromiteach: a tool used for teaching various subjects, including such STEM subjects as math and coding, and is implemented in Java. The student's laptop displays the Qromitype, a specialized whiteboard, and the tutor's smartphone displays a window: a partial copy of the student's screen. For some subjects, the student displays the Qromidesq, which is not limited to monospaced text. An always-on-top chat window (or a simultaneous phone conversation) takes care of the student's questions and the tutor's instructions, in case the tutor is non-local, otherwise Bluetooth provides connectivity. Tutors and students pay $20 and $10/year respectively to access the Qromidesq. The basic math Qromitype is free for all users.

Qromitype

The Qromitype supports math being taught, using text in monospaced mode. Most of its functionality is written in Java, but extensions used to teach STEM subjects are written in Uniqrome. The most commonly used commands are as follows:

- Use the arrow keys to move the cursor.
- Type underscore(s) to underline the numerator of a fraction.
- Use the special character command (Ctrl+K) to insert special characters such as pi, square root, sum, and integral.
- Use Tab/Shift+Tab to display/undo the next step in the math problem being solved.
- Type question mark (?) to explain the current step or to break the current step down into lower-level steps.
- Click on Help after typing question mark to access the help system.

Miscellaneous commands:

- Use asterisk and slash for multiply and divide.
- Fractions or matrices enclosed in brackets use tall brackets.
- Smart down/up arrow: press it after inserting a character moves the cursor beneath/above that character.
- Functions such as lines and parabolas can be plotted interactively on a graph.
- The default-to-upper-case setting assumes that all letters entered are upper case (use the shift key to enter a lower case letter), so Caps Lock is unnecessary.

Qromidesq:

- Display screen based on Qromitags, a text markup language
- May include panels, some containing a Qromitype

Expression Language

Mathematical expressions are encoded (internally) using the Uniqrome programming language. Each step in the math problem being solved manipulates this Uniqrome expression. Even if the user enters steps in a different order than the default ordering, the simplification logic can handle that. The user can type Tab/Shift+Tab to redo/undo her previous step, as well as to redo/undo the computer's previous step.

Computer Demos

Qromiteach can be used to teach computer skills. The student's laptop runs the practise demos featuring screenshots, cursor animation, and always-on-top yellow windows with black text. The yellow windows contain instructions to the student, and the tutor's smartphone is in sync with the student. The student can also run live demos including yellow windows, with MS Office, Chrome, or other applications running beneath the yellow windows. During the live demos, the tutor's smartphone is also in sync with the student.

Advanced Qromitype Commands

These next 2 paragraphs may be ignored, they are written in computerese. Use Shift+Arrow Key to highlight a rectangular block. Press Insert to insert a row or column of spaces before a highlighted block (insert blank line if no highlight). Press Shift+Insert/Delete to insert/delete an entire row/column when a block is highlighted. Press Enter at end of a line of text: insert blank line, back up on that line to line up with beginning of text on previous line. Press Enter on blank line to back up to line up with beginning of text on a previous line, or insert blank line if already at beginning of line. Press Ctrl+Tab to move forward to line up with beginning of first or next word on a previous line. Press Home to move to beginning of text on current line, press it again to toggle between beginning of line and beginning of text. This usage of Enter, Tab and Home is useful for editing program code with multiple indentation levels. The user doesn't have to memorize these commands: type question mark at any time to access the help system.

Superscripts

Superscripts and subscripts in monospaced mode are handled by employing a vertical offset of half a line per level of superscripting or subscripting. The caret symbol (^) is used as a superscript prefix, double-caret (^^) is used as a subscript prefix, and backslash (\) is used as an escape character (terminate super/subscript with a semicolon). Carets and double-carets cannot be mixed (exception: one level of superscript can be combined with one level of subscript).

About Us

I am Mike Hahn, the founder of Uniqrome.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, and became a volunteer math/computer tutor at West Neighbourhood House. I quit that job in mid-2019. I have a part-time job working for a perfume store. My hobbies are reading and I often go for walks. I don't read books very often, but on March 19, 2021 I started reading a biography of Steve Jobs which my brother gave me. I read the CBC news website, news/tech articles on my Flipboard app, and miscellaneous articles on my phone (same screen as my Google web page). I visit my brother once a month or more. For almost 30 years I was depressed on and off (I'm a rapid cycler), but it largely vanished after I ramped up development of my previous Aljegrid project in early March 2021.

Contact Info

Mike Hahn
Founder
Uniqrome.com
2495 Dundas St. West
Ste. 515
Toronto, ON M6P 1X4
Canada

Phone: 416-533-4417
Email: hahnbytes (AT) gmail (DOT) com
Web: treenimation.net/hahnbytes/

Uniqrome Language

Uniqrome (implemented in Java) is an open source 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, also statements are separated by semicolons). Uniqrome source files have a .UNIQ extension. Qromitags files (the sister language of Uniqrome, a text markup language) have a .QMTG extension. Uniqrome boasts an ultra-simple Lisp-like syntax unlike all other languages.

Special Characters

Core:

- ( ) grouping
- - word separator
- ; end of stmt.
- : dot operator
- " string delimiter
- \ escape char.

Operators:

- + - * / %
- = < >
- & | ^ ~ ! ?

Other:

- # comment
- {} block comment
- _ used in identifiers
- $ string prefix char.

Differences from Python

- Parentheses, not whitespace
- Operators come before their operands
- Integration with Qromitags
- Information hiding (public/private)
- Single, not multiple inheritance
- Adds interfaces ("hedron" defs.)
- Drops iterators and generators
- Adds lambdas
- Adds quote and list-compile functions, treating code as data
- Adds cons, car and cdr functionality

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. Typing space after hyphen at end of identifier converts hyphen to underscore. The close delim switch automatically inserts a closing parenthesis/brace/double quote when the open delimiter is inserted.

Qromitags

Qromitags is a simplified markup language used to replace HTML. Mock JSON files using Qromitags syntax have a .QMJS extension, and include no commas. Instead of myid: val, use [myid: val]. Instead of [1, 2, 3], use [arr: [: 1][: 2][: 3]]. Arbitrary Qromitags code can be embedded in the Uniqrome echo statement. Qromitags syntax, where asterisk (*) means occurs zero or more times, is defined as follows:

Tags:

- [tag]
- [tag (fld val)*: body]
- [tag (fld val)*| body |tag]

Body:

- text
- [(fld val)*: text]*

Uniqrome call:

- [expr: <expr>]
- [exec: <stmt>...]
- [uniq: <path>]

Uniqrome Grammar

White space occurs between tokens (parentheses and semicolons need no adjacent white space).

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*

<source file>:
- do ([<imp>]... [<def glb>] [<def>]... [<class>]...)

<imp>:
    <import stmt> **;**

<import stmt>:
    import <module>...
    from <rel module> import <mod list>
    from <rel module> import all

<module>:
    <name>
    (**:** <name><name>...)
    (as <name><name>)
    (as (**:** <name><name>...) <name>)

<mod list>:
    <id as>...

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

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

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

<cls typ>:
    class
    iclass

<hedron>:
    hedron
    ihedron

<class>:
- <cls typ><name> [<base class>] [<does>] [<vars>] [<ivars>] do (<def>…) **;**
- abclass <name> [<base class>] [<does>] [<vars>] [<ivars>] do (<anydef>…) **;**
- <hedron><name> [<does>] [<const list>] do ([<abdef>]... [<defimp>]...) **;**
- enum <name><elist> **;**
- ienum <name><elist> **;**

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

<hedron name>:
<base class>:
    <name>
    (**:** <name><name>…)

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

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

<def glb>:
    gdefun [<vars>] [<ivars>] do <block> **;**

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

<defimp>:
- defimp (<name> [<parms>]) [<vars>] [<gvars>] [<dec>] do <block> **;**

<abdef>:
    abdefun (<name> [<parms>]) [<dec>] **;**

<defun>:
    defun
    idefun

<anydef>:
    <def>
    <abdef>

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

<ivars>:
    (ivar [<id>]...)

<gvars>:
    (gvar [<id>]...)

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

<parm>:
    (<set op><id><const expr>)

<dec>:
    (decor <dec expr>...)

<block>:
    ([<stmt-semi>]…)

<stmt-semi>:
    <stmt> **;**

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

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

<stmt>:
    <if stmt>
    <while stmt>
    <for stmt>
    <switch stmt>
    <try stmt>
    <asst stmt>
    <del stmt>
    <jump stmt>
    <call stmt>
    <print stmt>
    <bool stmt>

<call expr>:
- (<name> [<arg list>])
- (**:** <colon expr>… <name>)
- (**:** <colon expr>… (<method name> [<arg list>]))
- (**::** <colon expr>… <name> else <expr>)
- (**::** <colon expr>… (<method name> [<arg list>]) else <expr>)
- (call <expr> [<arg list>])

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

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

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

<dec expr>:
    <name>
    (<name><id>...)
    (**:** <name><id>...)
    (**:** <name>... (<id>...))

<dot op>:
    dot | **:**

<dotnull op>:
    dotnull | **::**

<del stmt>:
    del <expr>

<set op>:
    set | =

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

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

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

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

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

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

<for stmt>:
- for <name> [<idx var>] in <expr> do <block>
- for (<bool stmt>**;** <bool stmt>**;** < bool stmt>)
  do <block>

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

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

<bool stmt>:
  quest [<expr>]
  ? [<expr>]
  <asst stmt>

<switch stmt>:
  switch <expr><case body> [else do <block>]

<case body>:
  [case <id> do <block>]...
  [case <dec int> do <block>]...
  [case <str lit> do <block>]...
  [case <tuple expr> do <block>]...

<return stmt>:
  return

<break stmt>:
  break

<continue stmt>:
  continue

<paren stmt>:
  (<stmt>)

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

<quest>:
  quest | ?

<inc op>:
  incint | decint | ++ | --

<expr>:
  <keyword const>
  <literal>
  <name>
  (<unary op><expr>)
  (<bin op><expr><expr>)
  (<multi op><expr><expr>…)
  (<quest><expr><expr><expr>)
  <lambda>
  (quote <expr>...)
  <cons expr>
  <tuple expr>
  <list expr>
  <dict expr>
  <venum expr>
  <string expr>
  <bytes expr>
  <target expr>
  <call expr>
  <cast>

<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>

<cons expr>:
(cons <expr><expr>)
(<crop><expr>)
<tuple expr>:
(tuple [<expr>]…)
(<literal> [<expr>]…)
( )

<list expr>:
(jist [<expr>]…)

<dict expr>:
(dict [<pair>]…)

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

<venum expr>:
(venum <enum name> [<elist>])
(venum <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>
<id>
(**:** <id><id>)
<cast>:
(cast <literal><expr>)
(cast <class name><expr>)

<print stmt>: *// built-in func*
print <expr>…
println [<expr>]…
echo <expr>…

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

No white space allowed between tokens, for rest of Uniqrome 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>:
    null
    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

<str lit>:
    " [<str item>]... "

<str item>:
    <str char>
    <escaped str char>
    <str newline>

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

<str newline>:
    \ <newline> [<white space>] "

<escaped char>:
    \\ *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*

<crop>:
    c <crmid>... r

<crmid>:
    a | d

Not implemented: string prefix and bytes data type (rest of grammar)

<str lit>:
    [$ <str prefix>] <quoted str>

<str prefix>:
    r | R

<quoted str>:
    " [<str item>]... "

<bytes lit>:
    $ <byte prefix><quoted bytes>

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

<quoted bytes>:
    " [<bytes item>]... "

<bytes item>:
    <bytes char>
    <escaped char>
    <str newline>

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