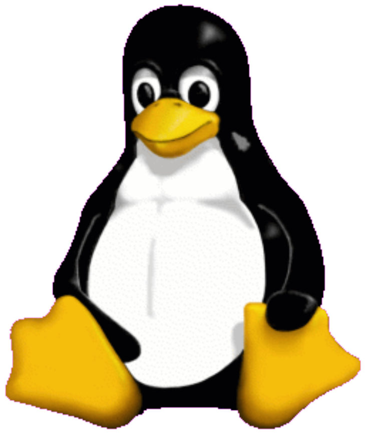
Unix103

BIOL647
Digital Biology

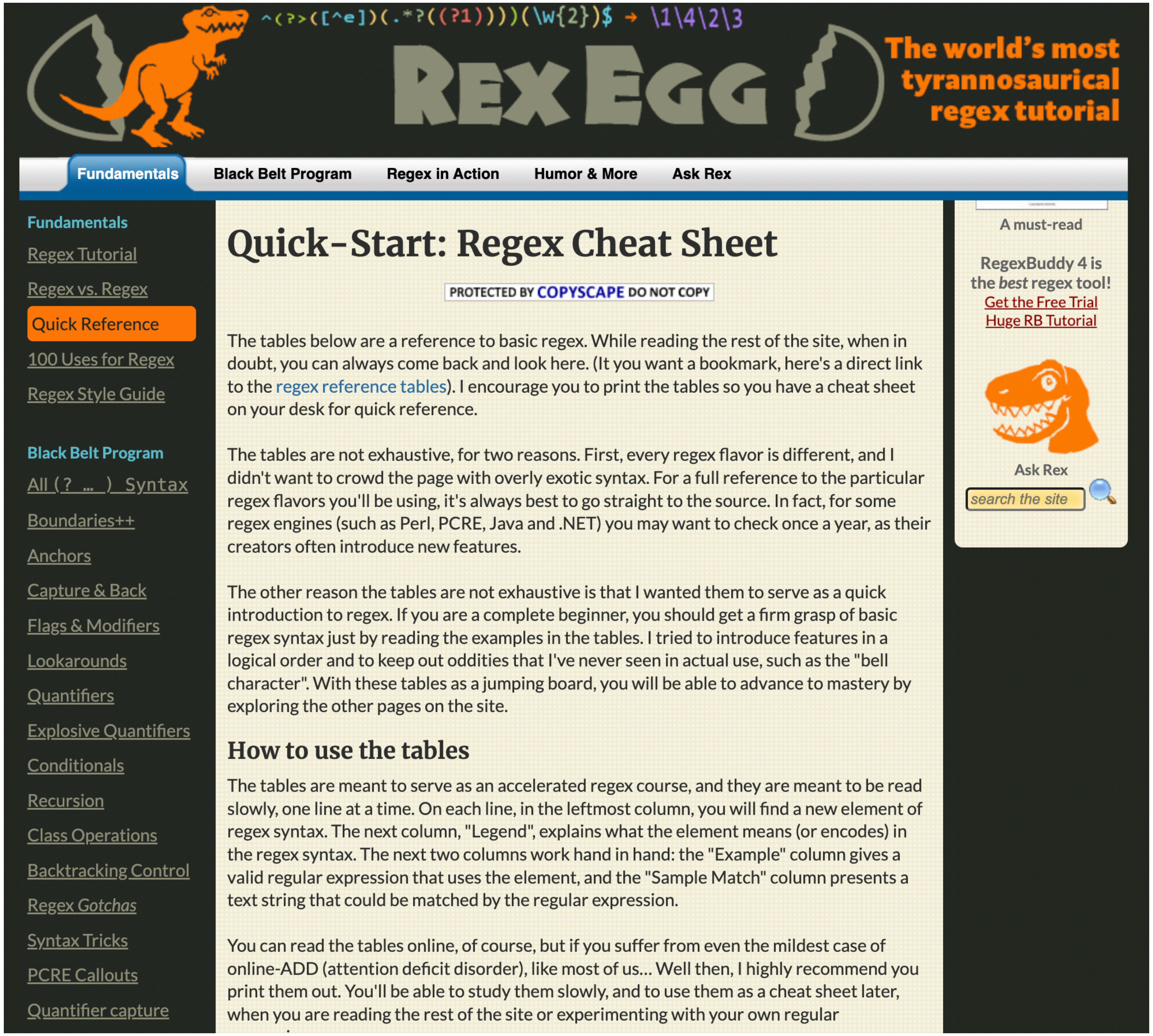
Rodolfo Aramayo



Unix103

GREP (Global Regular Expression Print) Pattern Searching

Regular Expressions Syntax



^(?>([^e])(.*?((?1))))(\w{2})$ → \1\4\2\3
REX EGG
The world's most tyrannosaurical regex tutorial
Fundamentals
Black Belt Program
Regex in Action
Humor & More
Ask Rex

Fundamentals

Regex Tutorial

Regex vs. Regex

Quick Reference

100 Uses for Regex

Regex Style Guide

Black Belt Program

All (? …) Syntax

Boundaries++

Anchors

Capture & Back

Flags & Modifiers

Lookarounds

Quantifiers

Explosive Quantifiers

Conditionals

Recursion

Class Operations

Backtracking Control

Regex *Gotchas*

Syntax Tricks

PCRE Callouts

Quantifier capture

Quick-Start: Regex Cheat Sheet

PROTECTED BY COPYSCAPE DO NOT COPY

The tables below are a reference to basic regex. While reading the rest of the site, when in doubt, you can always come back and look here. (It you want a bookmark, here's a direct link to the regex reference tables). I encourage you to print the tables so you have a cheat sheet on your desk for quick reference.

The tables are not exhaustive, for two reasons. First, every regex flavor is different, and I didn't want to crowd the page with overly exotic syntax. For a full reference to the particular regex flavors you'll be using, it's always best to go straight to the source. In fact, for some regex engines (such as Perl, PCRE, Java and .NET) you may want to check once a year, as their creators often introduce new features.

The other reason the tables are not exhaustive is that I wanted them to serve as a quick introduction to regex. If you are a complete beginner, you should get a firm grasp of basic regex syntax just by reading the examples in the tables. I tried to introduce features in a logical order and to keep out oddities that I've never seen in actual use, such as the "bell character". With these tables as a jumping board, you will be able to advance to mastery by exploring the other pages on the site.

How to use the tables

The tables are meant to serve as an accelerated regex course, and they are meant to be read slowly, one line at a time. On each line, in the leftmost column, you will find a new element of regex syntax. The next column, "Legend", explains what the element means (or encodes) in the regex syntax. The next two columns work hand in hand: the "Example" column gives a valid regular expression that uses the element, and the "Sample Match" column presents a text string that could be matched by the regular expression.

You can read the tables online, of course, but if you suffer from even the mildest case of online-ADD (attention deficit disorder), like most of us... Well then, I highly recommend you print them out. You'll be able to study them slowly, and to use them as a cheat sheet later, when you are reading the rest of the site or experimenting with your own regular

A must-read

RegexBuddy 4 is the *best* regex tool!

Get the Free Trial

Huge RB Tutorial

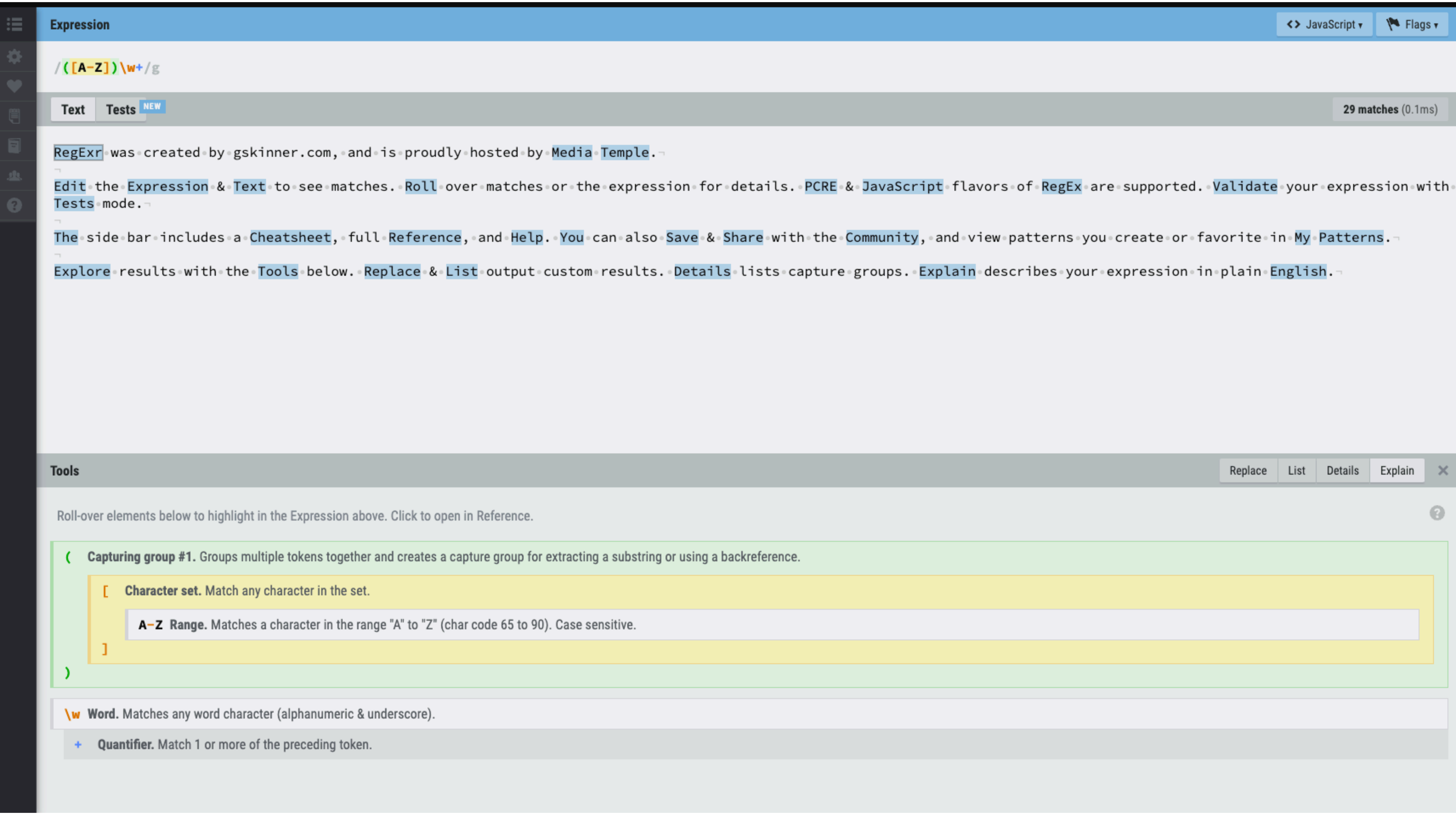
Ask Rex

search the site

Unix103

Introduction to Regular Expressions

Go to regexr.com



```
/([A-Z])\w+/g
```

29 matches (0.1ms)

RegExr was created by gskinner.com, and is proudly hosted by Media Temple.

Edit the Expression & Text to see matches. Roll over matches or the expression for details. PCRE & JavaScript flavors of RegEx are supported. Validate your expression with Tests mode.

The side bar includes a Cheatsheet, full Reference, and Help. You can also Save & Share with the Community, and view patterns you create or favorite in My Patterns.

Explore results with the Tools below. Replace & List output custom results. Details lists capture groups. Explain describes your expression in plain English.

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

(**Capturing group #1.** Groups multiple tokens together and creates a capture group for extracting a substring or using a backreference.

[**Character set.** Match any character in the set.

A-Z **Range.** Matches a character in the range "A" to "Z" (char code 65 to 90). Case sensitive.

]

)

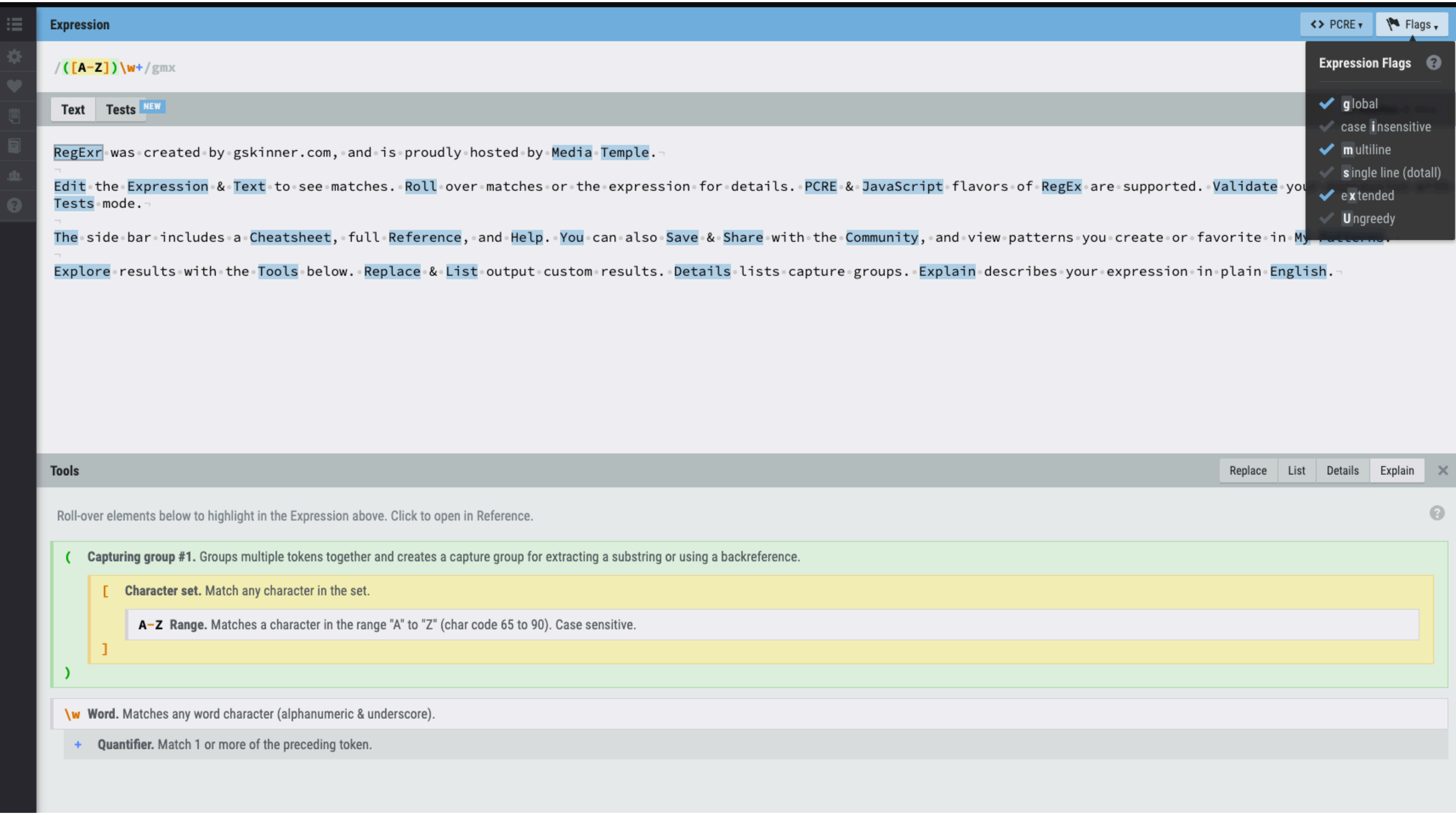
\w **Word.** Matches any word character (alphanumeric & underscore).

+ **Quantifier.** Match 1 or more of the preceding token.

Unix103

Introduction to Regular Expressions

Select PCRE and Flags: global, multiline and extended



```
/([A-Z])\w+/gmx
```

Expression Flags

- ✓ global
- case insensitive
- ✓ multiline
- single line (dotall)
- ✓ extended
- Ungreedy

```
RegExr was created by gskinner.com, and is proudly hosted by Media Temple.

Edit the Expression & Text to see matches. Roll over matches or the expression for details. PCRE & JavaScript flavors of RegEx are supported. Validate you
Tests mode.

The side bar includes a Cheatsheet, full Reference, and Help. You can also Save & Share with the Community, and view patterns you create or favorite in My

Explore results with the Tools below. Replace & List output custom results. Details lists capture groups. Explain describes your expression in plain English.
```

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

(**Capturing group #1.** Groups multiple tokens together and creates a capture group for extracting a substring or using a backreference.

[**Character set.** Match any character in the set.

A-Z **Range.** Matches a character in the range "A" to "Z" (char code 65 to 90). Case sensitive.

]

)

\w **Word.** Matches any word character (alphanumeric & underscore).

+ **Quantifier.** Match 1 or more of the preceding token.

Unix103

Introduction to Regular Expressions

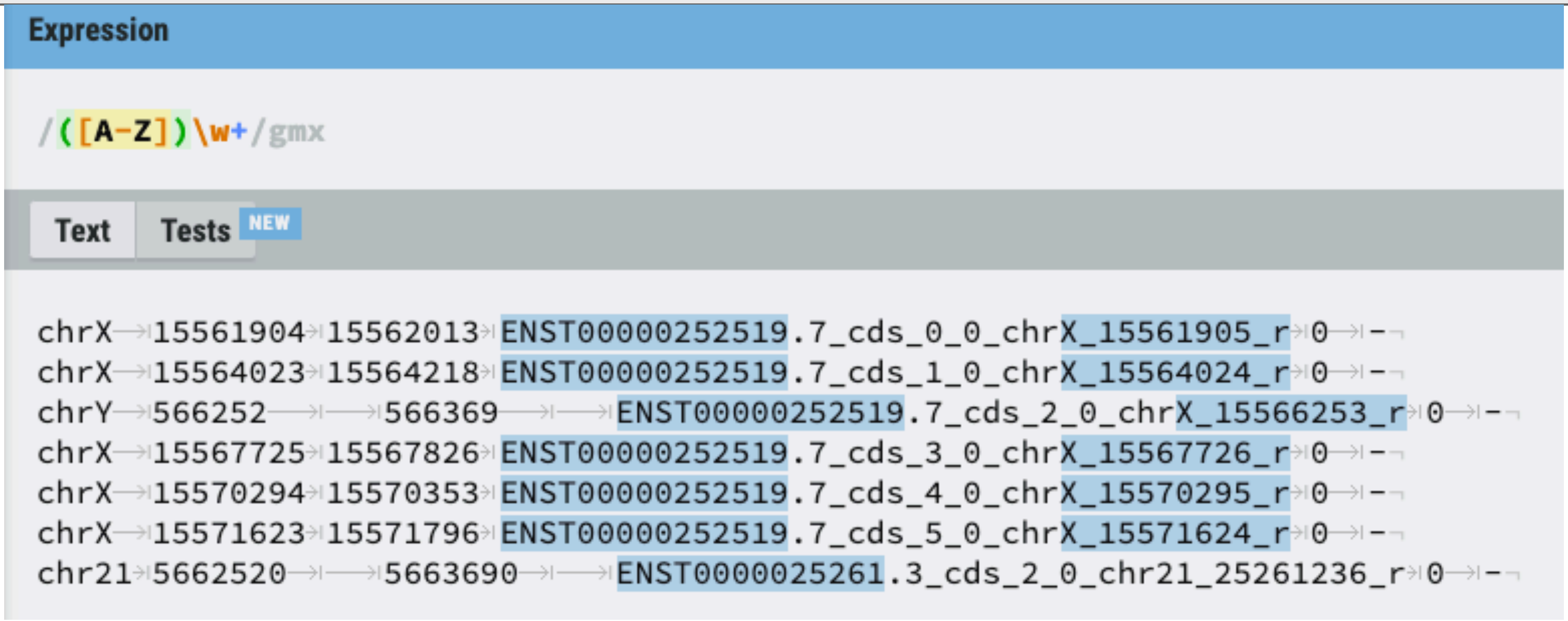
A Regular Expression is a pattern describing a certain amount of text

- The ( ) { } [ ] . * ? + ^ $ are all special characters
- \ can be used to “escape” a special character, allowing that special character

  (i.e., ( ) { } [ ] . * ? + ^ $), to be searched for

Copy and Paste the following Text Extracted from this: file. <– Link to text file

```
chrX 15561904   15562013   ENST00000252519.7_cds_0_0_chrX_15561905_r0-
chrX 15564023   15564218   ENST00000252519.7_cds_1_0_chrX_15564024_r0-
chrY 566252     566369     ENST00000252519.7_cds_2_0_chrX_15566253_r0-
chrX 15567725   15567826   ENST00000252519.7_cds_3_0_chrX_15567726_r0-
chrX 15570294   15570353   ENST00000252519.7_cds_4_0_chrX_15570295_r0-
chrX 15571623   15571796   ENST00000252519.7_cds_5_0_chrX_15571624_r0-
chr215662520    5663690    ENST0000025261.3_cds_2_0_chr21_25261236_r0-
```



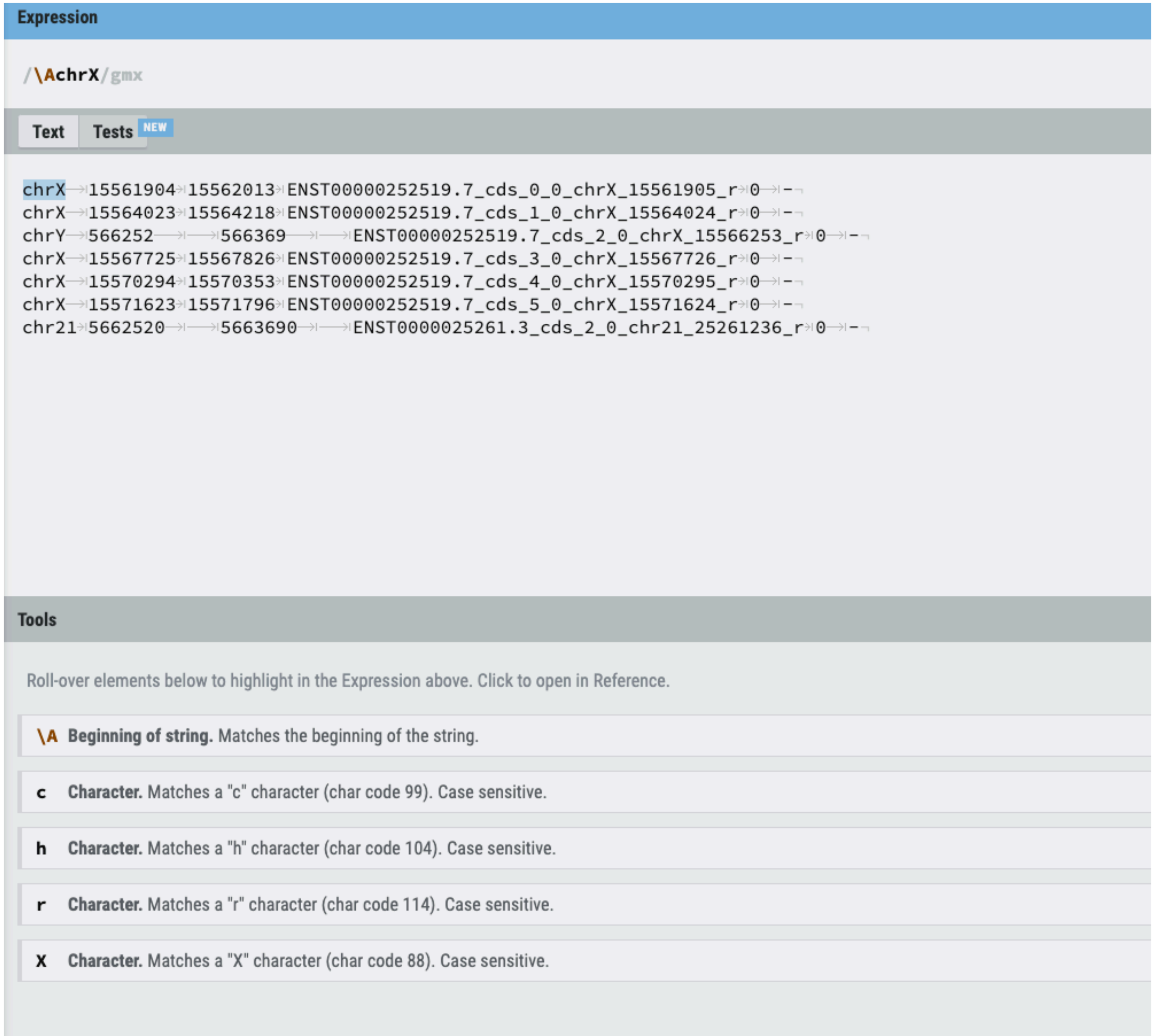
```
/([A-Z])\w+/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Unix103

Introduction to Regular Expressions

- “\A” matches the beginning of a string (but not an internal line
- Regex: \AchrX



```
/\AchrX/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\A **Beginning of string.** Matches the beginning of the string.

c **Character.** Matches a "c" character (char code 99). Case sensitive.

h **Character.** Matches a "h" character (char code 104). Case sensitive.

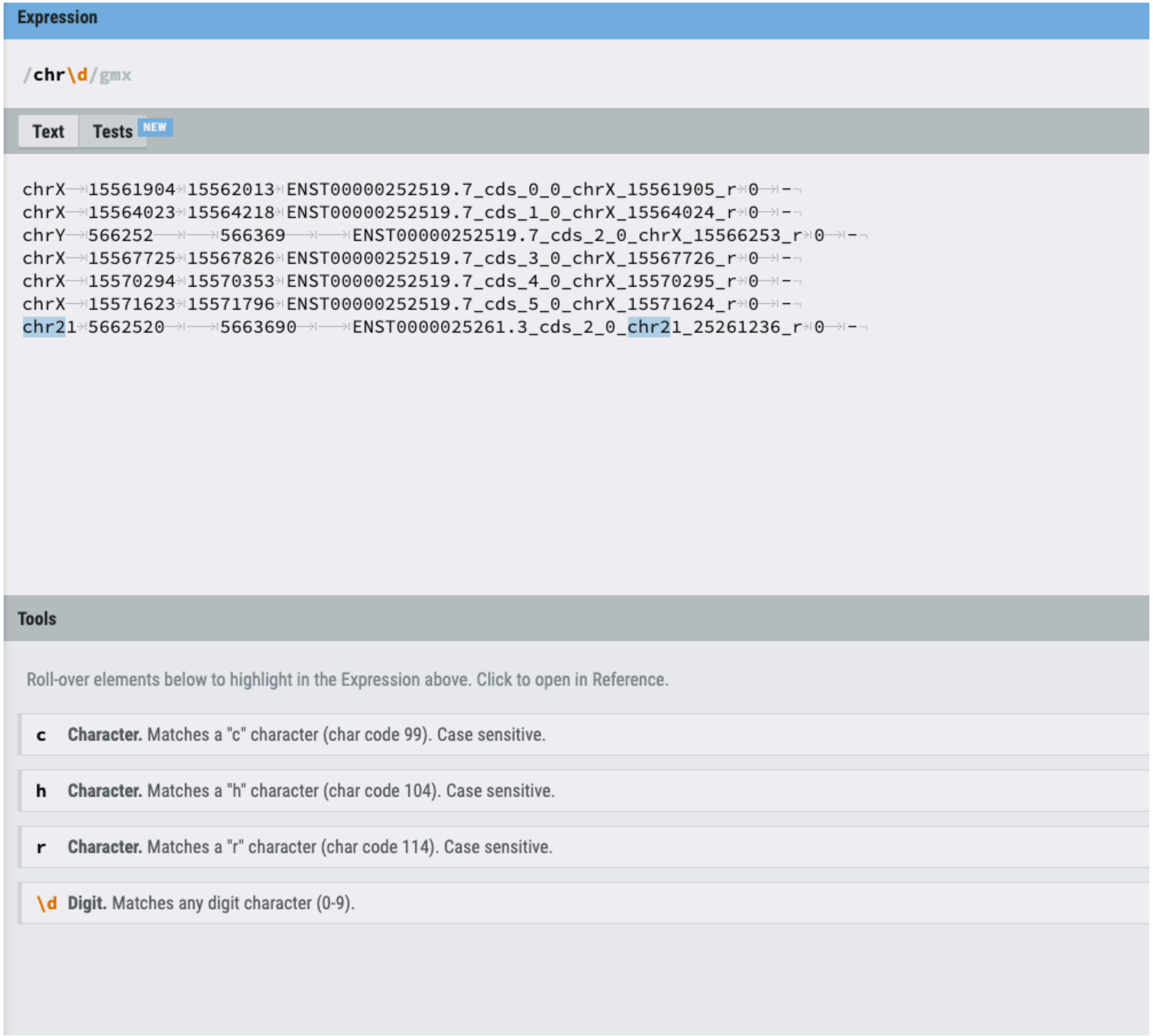
r **Character.** Matches a "r" character (char code 114). Case sensitive.

X **Character.** Matches a "X" character (char code 88). Case sensitive.

Unix103

Introduction to Regular Expressions

- “\d” matches a digit class, same as [0-9]
- Regex: chr\d



Expression

```
/chr\d/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Tools

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

c **Character.** Matches a "c" character (char code 99). Case sensitive.

h **Character.** Matches a "h" character (char code 104). Case sensitive.

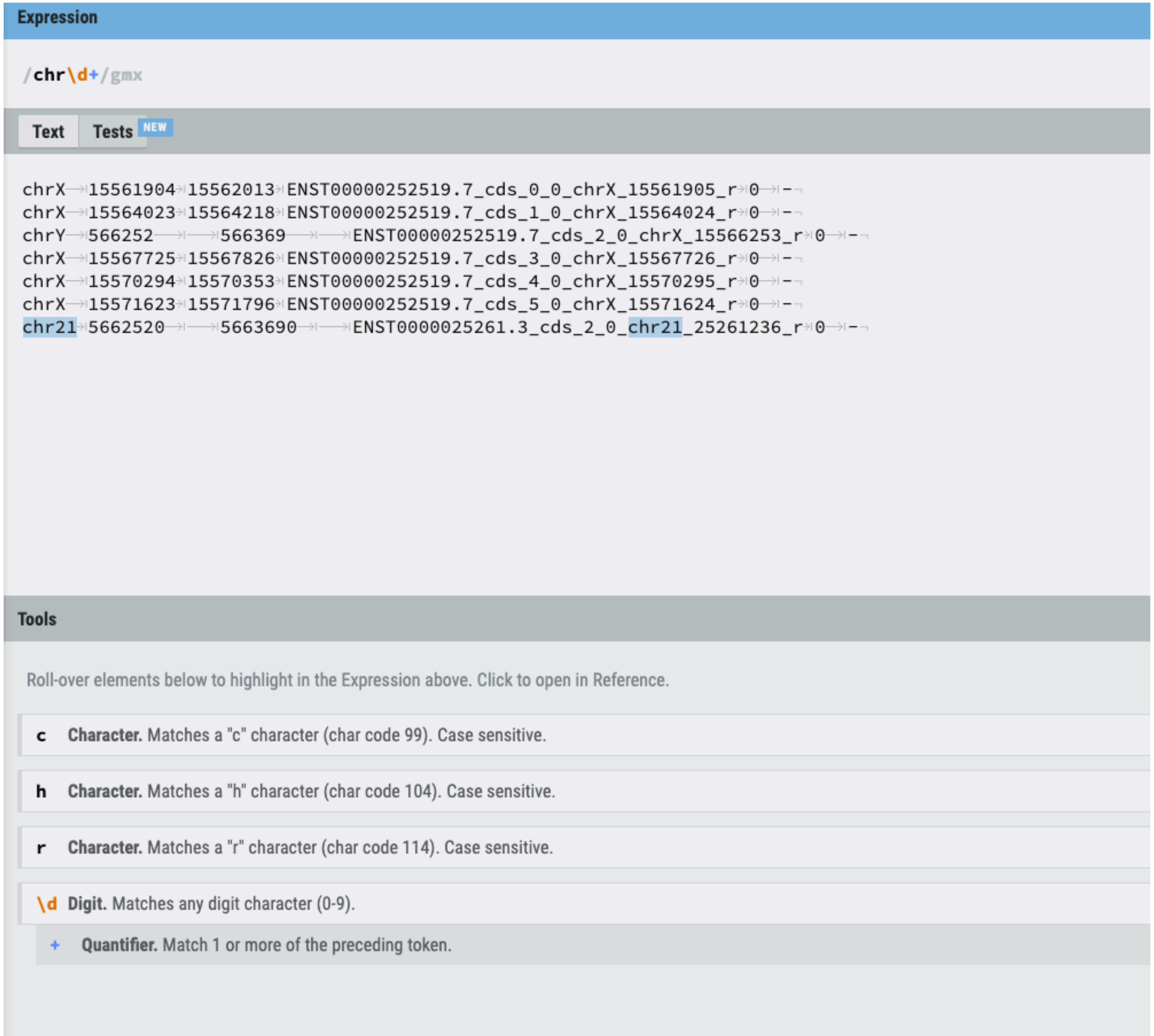
r **Character.** Matches a "r" character (char code 114). Case sensitive.

\d **Digit.** Matches any digit character (0-9).

Unix103

Introduction to Regular Expressions

- “\d” matches a digit class, same as [0-9]
- Regex: chr\d+



Expression

```
/chr\d+/gmx
```

Text Tests NEW

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Tools

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

c **Character.** Matches a "c" character (char code 99). Case sensitive.

h **Character.** Matches a "h" character (char code 104). Case sensitive.

r **Character.** Matches a "r" character (char code 114). Case sensitive.

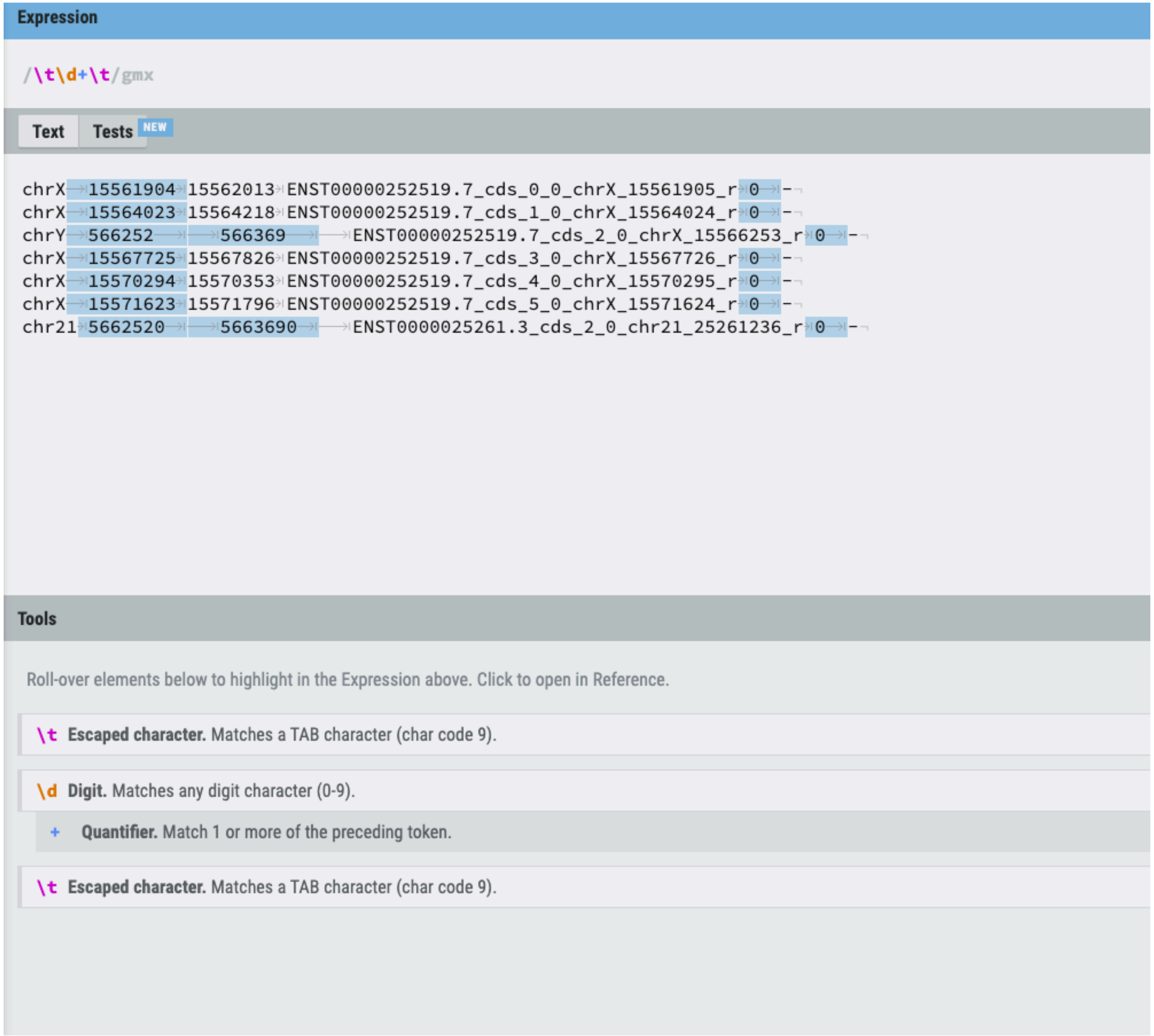
\d **Digit.** Matches any digit character (0-9).

+ **Quantifier.** Match 1 or more of the preceding token.

Unix103

Introduction to Regular Expressions

- “\d” matches a digit class, same as [0-9]
- Regex: \t\d+\t



Expression

```
/\t\d+\t/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Tools

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\t **Escaped character.** Matches a TAB character (char code 9).

\d **Digit.** Matches any digit character (0-9).

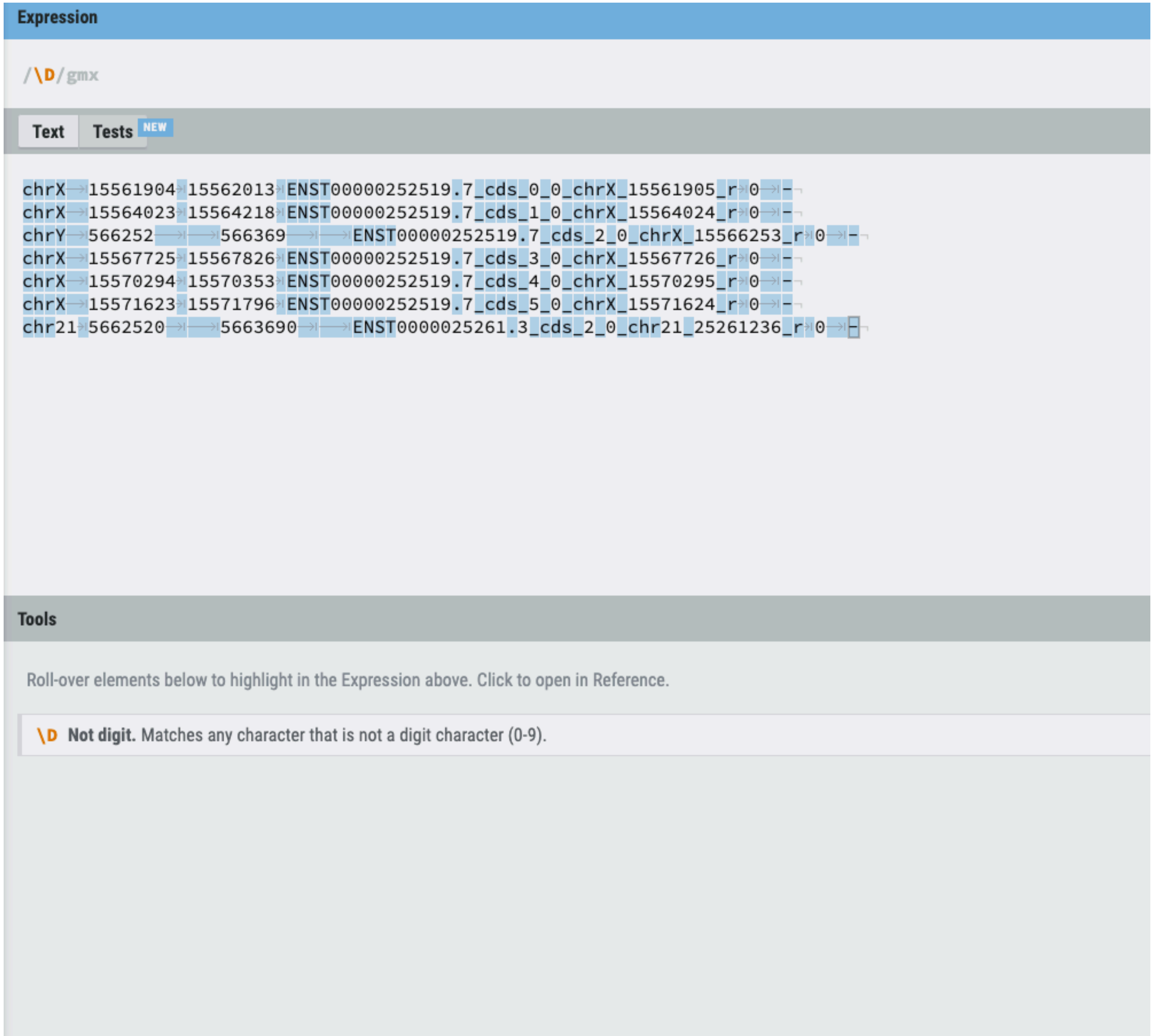
+ **Quantifier.** Match 1 or more of the preceding token.

\t **Escaped character.** Matches a TAB character (char code 9).

Unix103

Introduction to Regular Expressions

- “\D” matches a non-digit
- Regex: \D



```
/\D/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

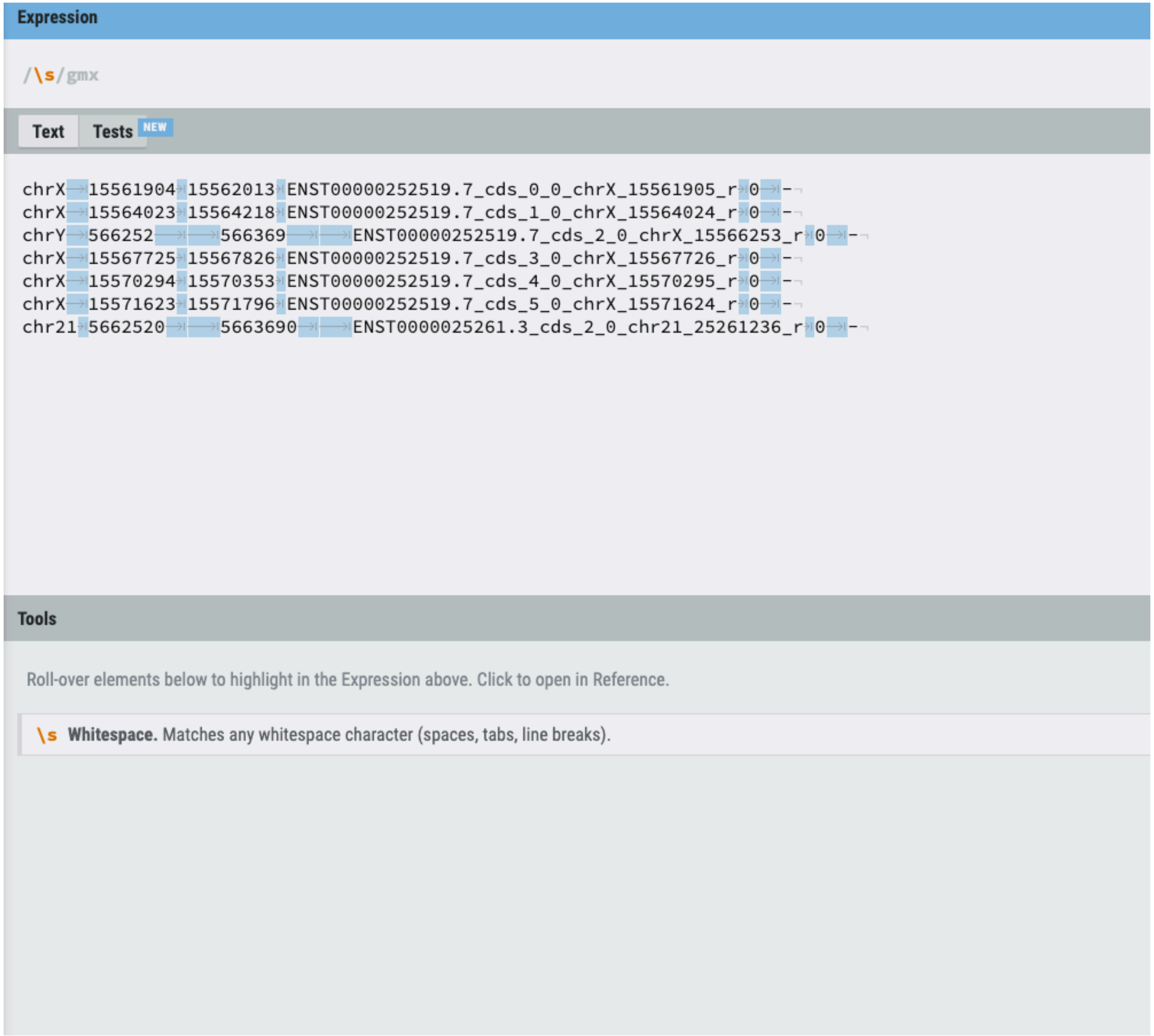
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\D **Not digit.** Matches any character that is not a digit character (0-9).

Unix103

Introduction to Regular Expressions

- “\s” matches a whitespace character
- Regex: \s



```
/\s/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

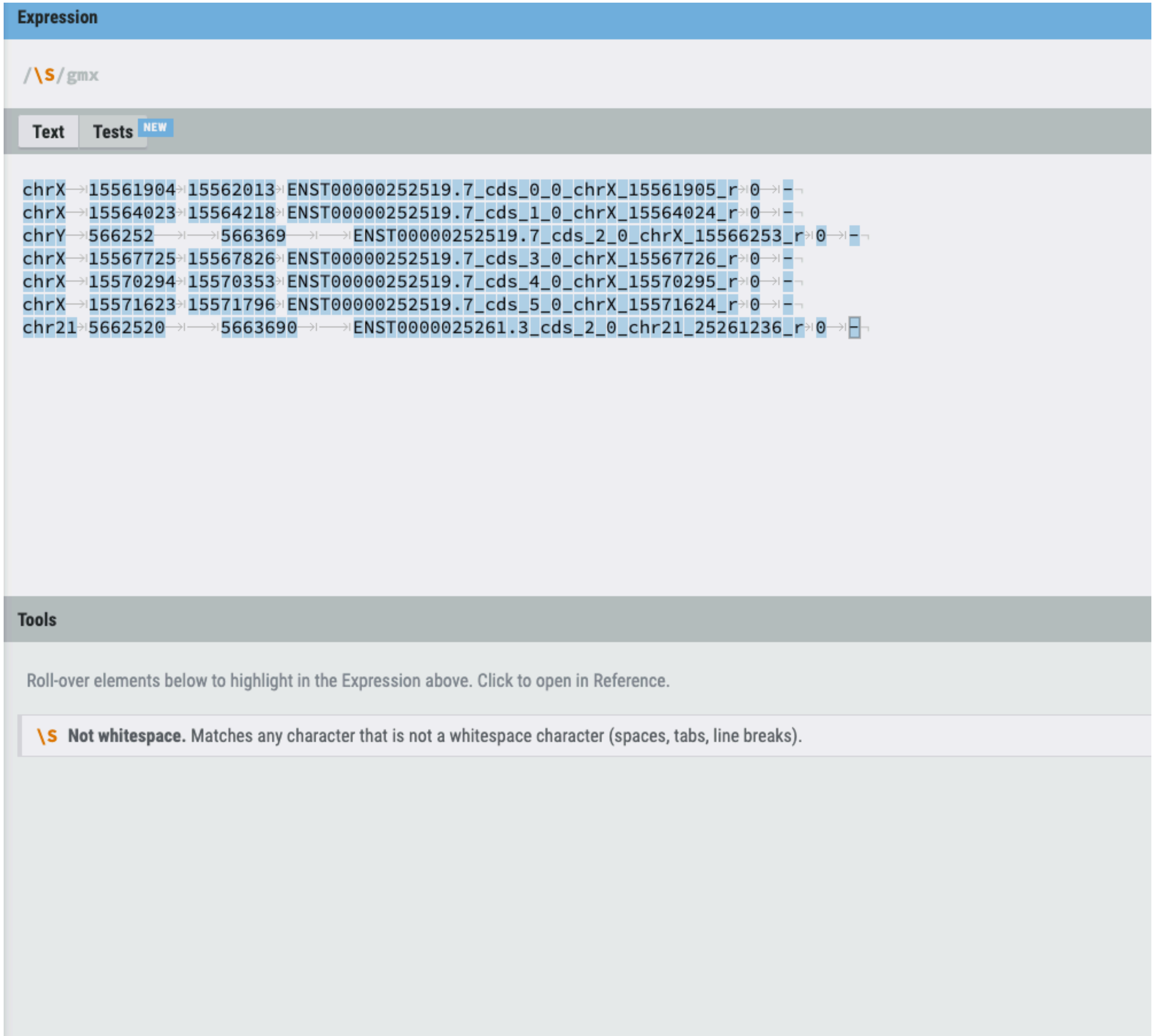
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\s **Whitespace.** Matches any whitespace character (spaces, tabs, line breaks).

Unix103

Introduction to Regular Expressions

- "\S" matches anything BUT a whitespace
- Regex: \S



```
/\S/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

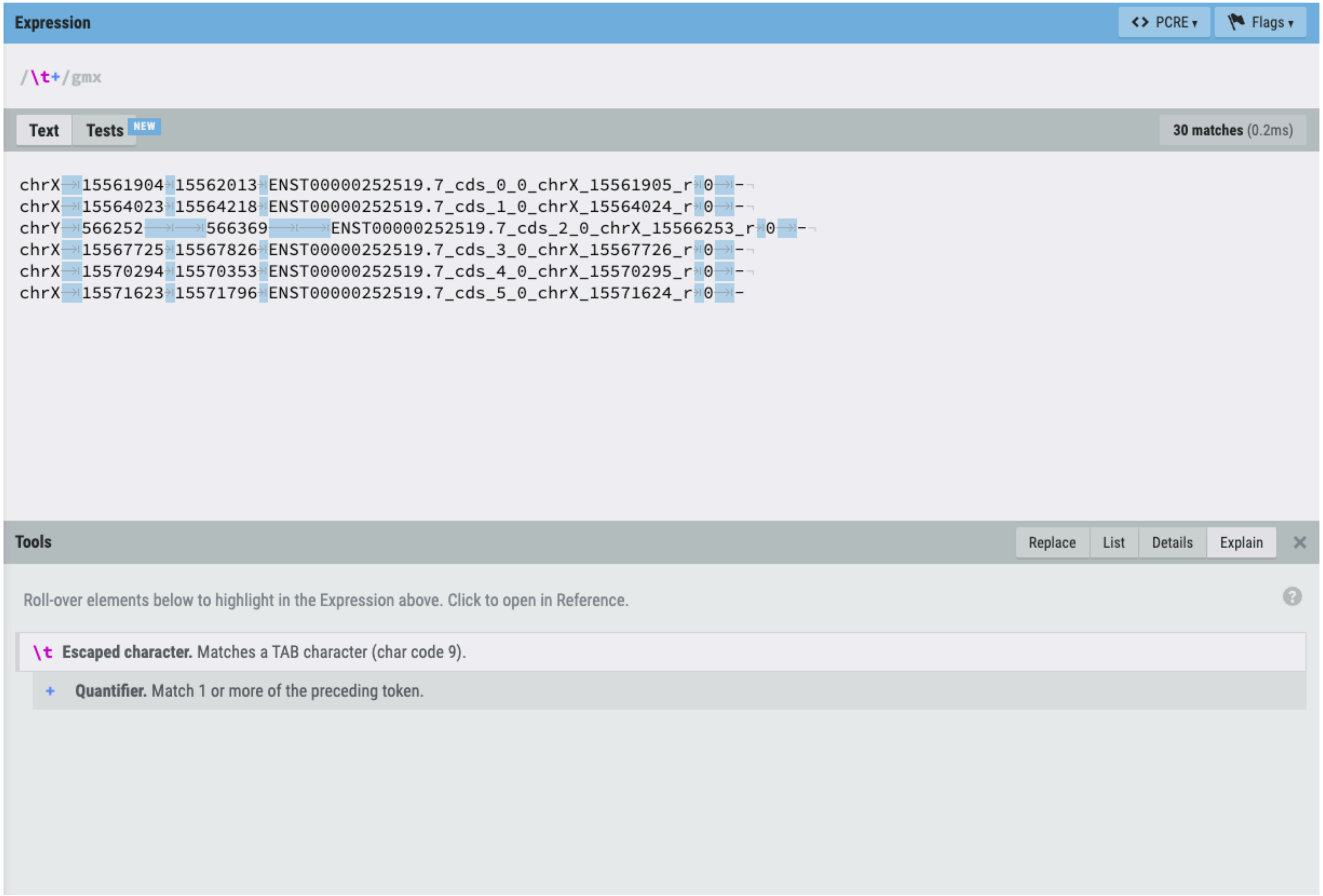
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\S **Not whitespace.** Matches any character that is not a whitespace character (spaces, tabs, line breaks).

Unix103

Introduction to Regular Expressions

- “\t” matches a tab
- Regex: \t
- Regex: \t+



```
/\t+/gmx
```

30 matches (0.2ms)

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
```

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

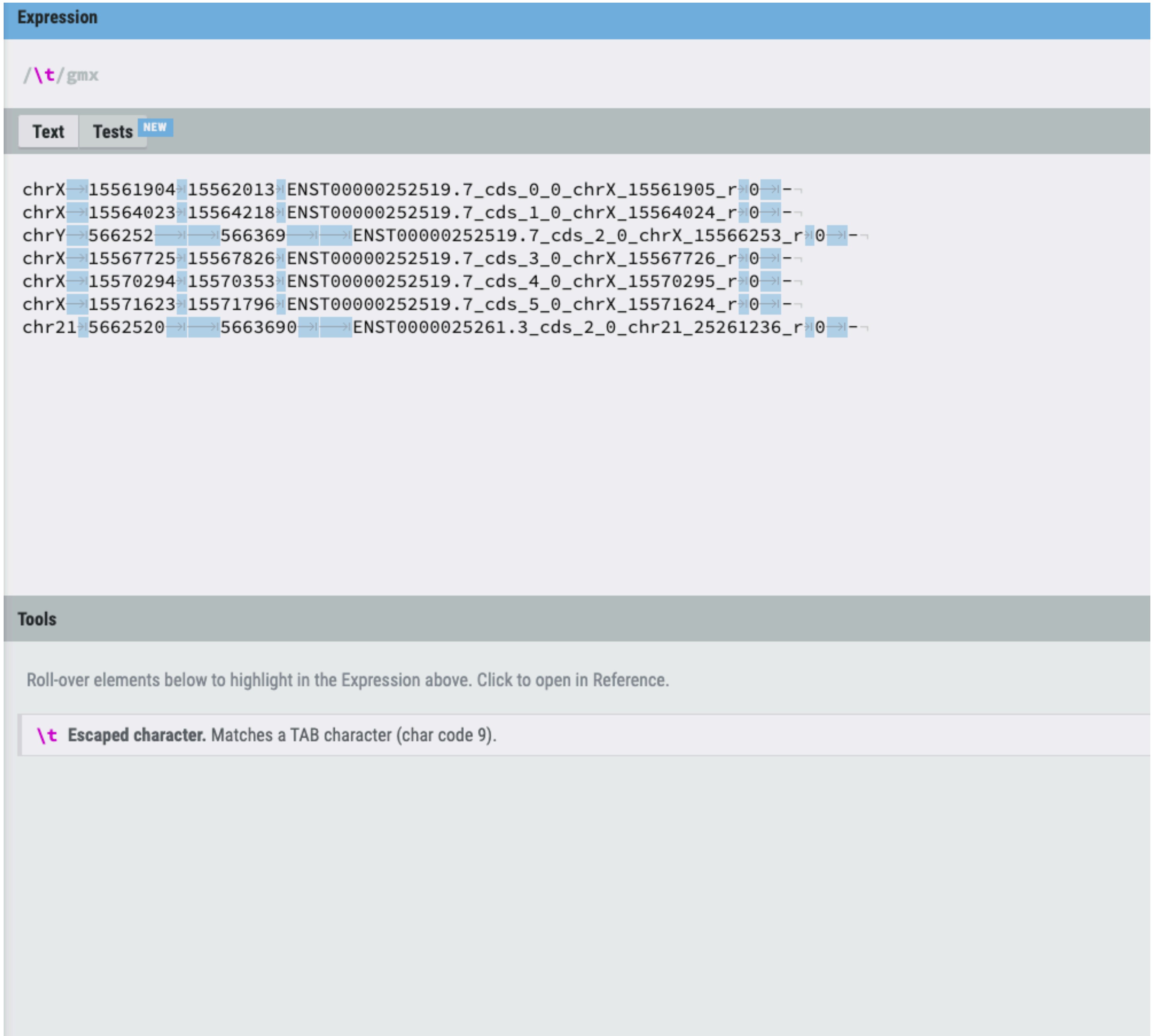
\t **Escaped character.** Matches a TAB character (char code 9).

+ **Quantifier.** Match 1 or more of the preceding token.

Unix103

Introduction to Regular Expressions

- “\t” matches a tab
- Regex: \t



```
/\t/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

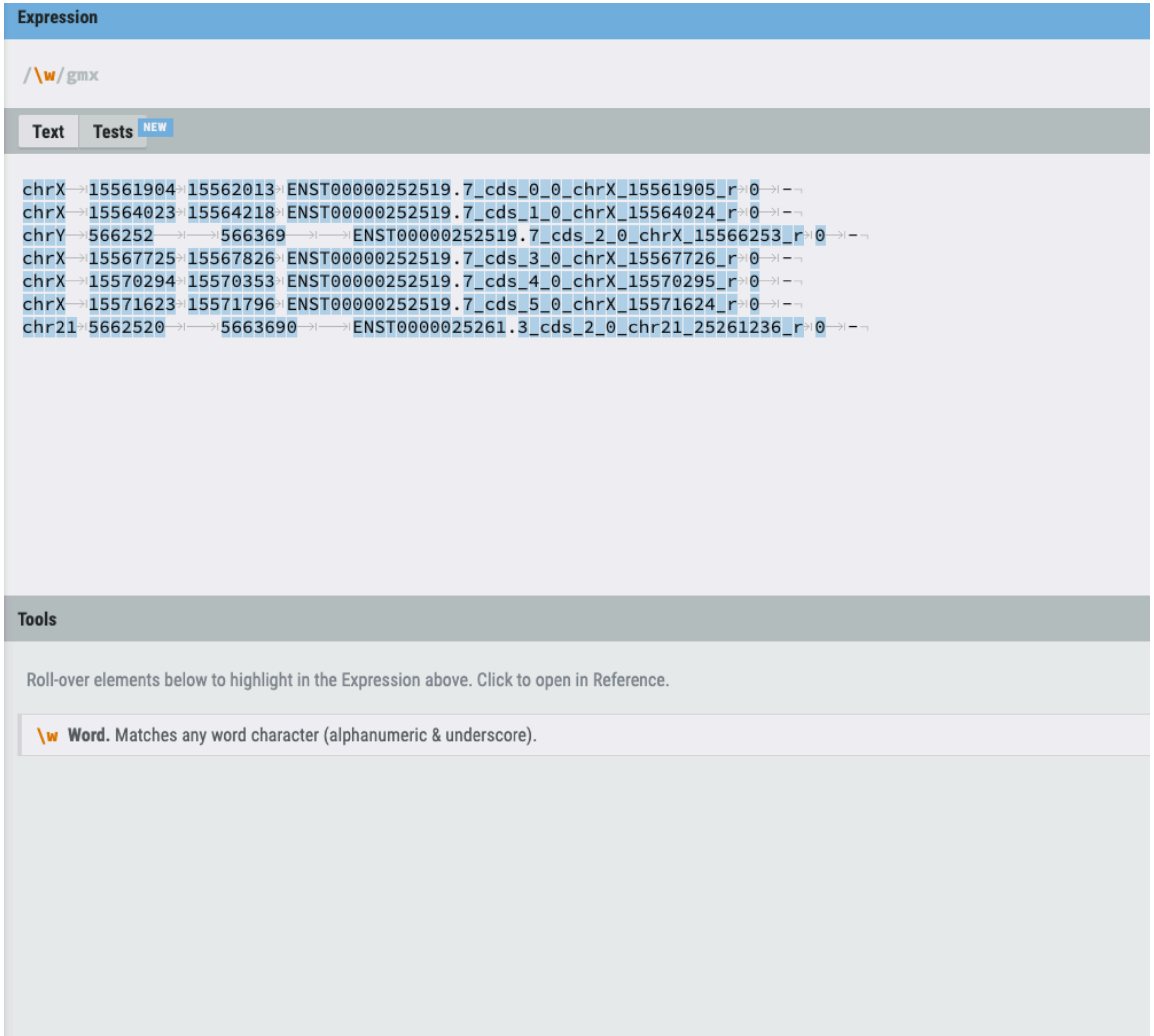
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\t **Escaped character.** Matches a TAB character (char code 9).

Unix103

Introduction to Regular Expressions

- “\w” matches an alphanumeric character
- Regex: \w



```
/\w/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

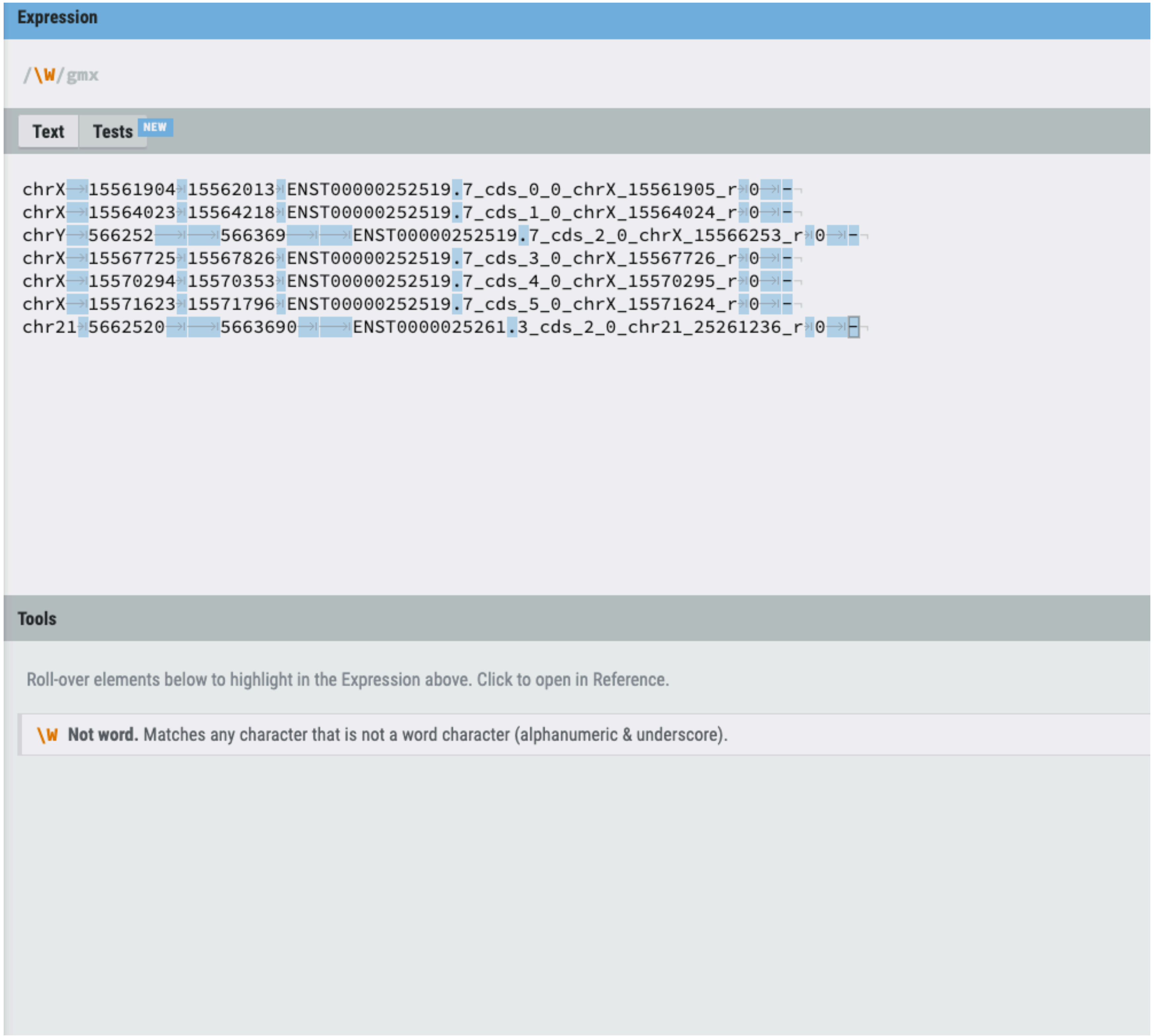
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\w **Word.** Matches any word character (alphanumeric & underscore).

Unix103

Introduction to Regular Expressions

- “\W” matches anything but an alphanumeric character
- Regex: \W



```
/\W/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

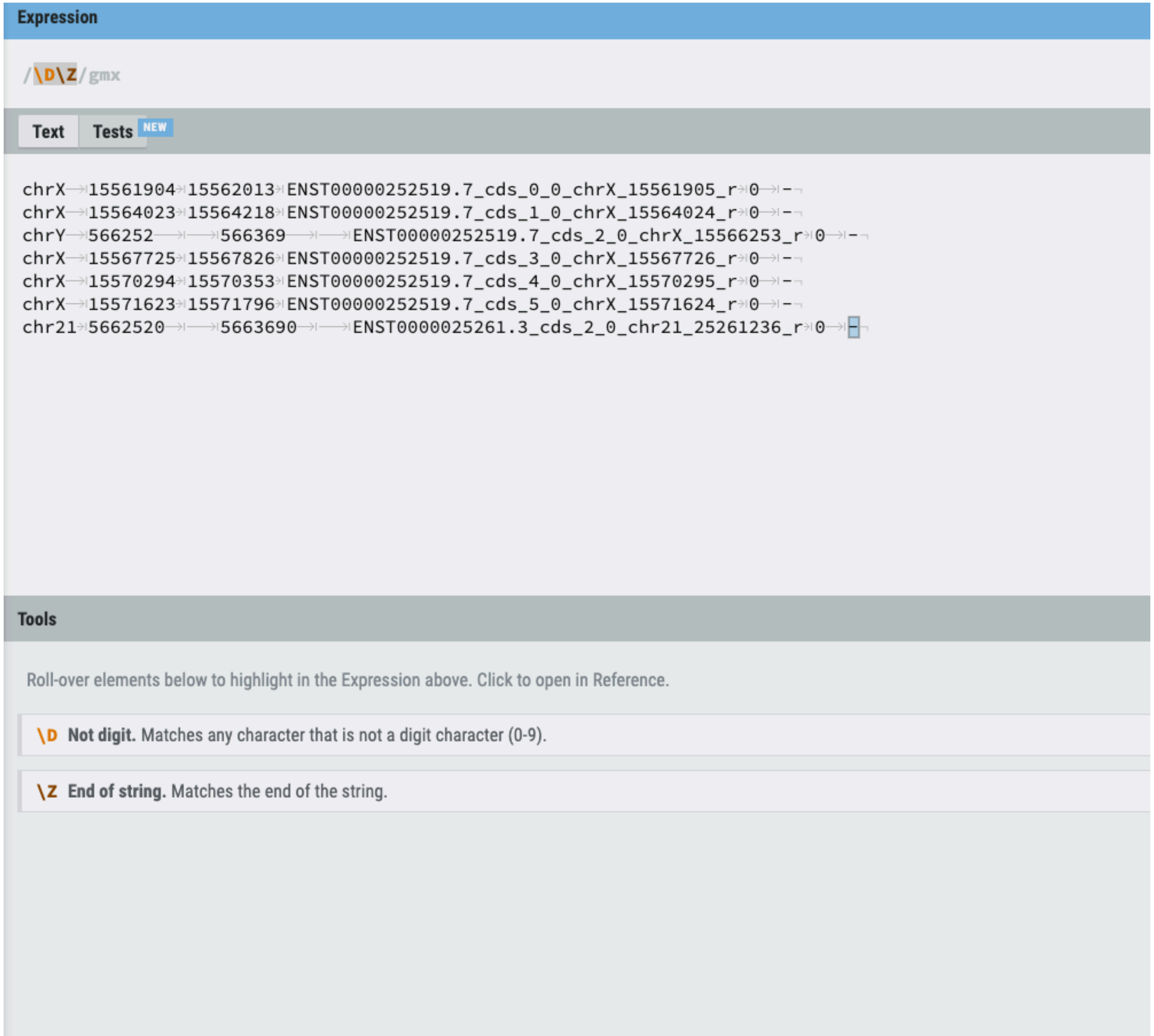
Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\W **Not word.** Matches any character that is not a word character (alphanumeric & underscore).

Unix103

Introduction to Regular Expressions

- "\Z" matches the end of a string(but not a internal line)
- Regex: \D\Z



```
/\D\Z/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Roll-over elements below to highlight in the Expression above. Click to open in Reference.

\D **Not digit.** Matches any character that is not a digit character (0-9).

\Z **End of string.** Matches the end of the string.

Unix103

Introduction to Regular Expressions

- “{ n or n, or n,m }” specifies an expected number of repetitions of the preceding pattern

- “{n}” The preceding item is matched exactly n times.

- “{n,}” The preceding item is matched n or more times.

- “{n,m}” The preceding item is matched at least n times but not more than m times.

- “[…]” creates a character class
  - Within the brackets, single characters can be placed
  - A dash (-) may be used to indicate a range such as a-z

- “.” Matches any single character except a newline

- ”*” The preceding item will be matched zero or more times

- “?” The preceding item is optional and matched at most once

- + The preceding item will be matched one or more time

- “^” has two meaning:
  - matches the beginning of a line or string
  - indicates negation in a character class

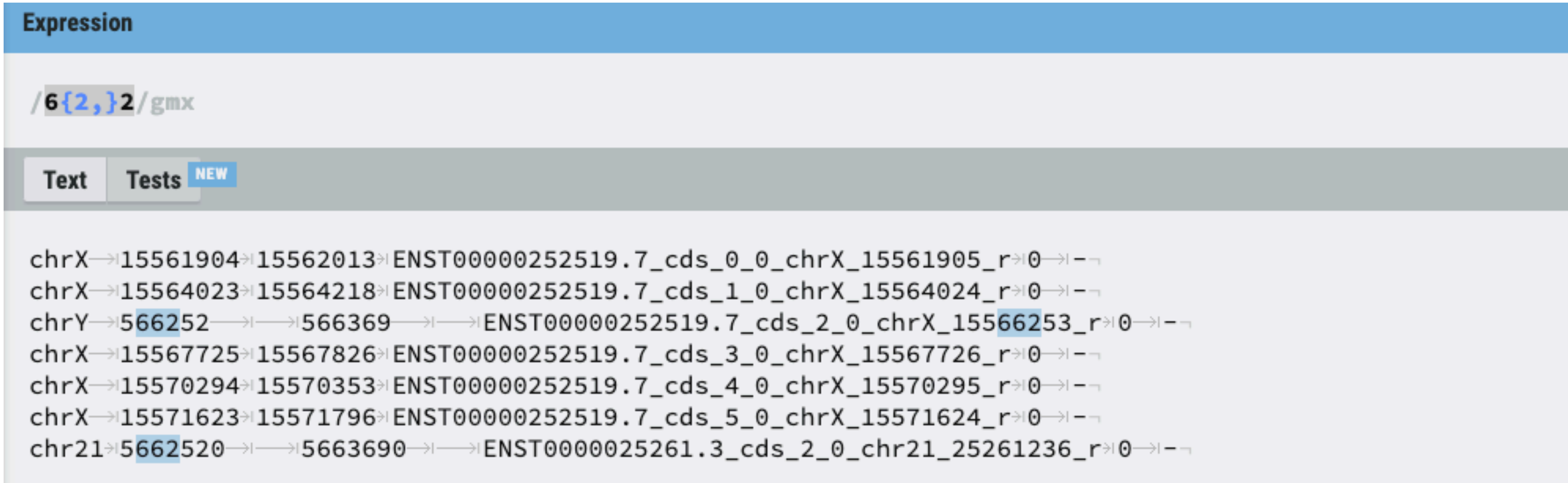
- For example, [^…] matches every character except the ones inside brackets

  - “$” matches the end of a line or string
  - “|” Separates alternate possibilities
  - “(..)” groups a particular pattern

Unix103

Introduction to Regular Expressions

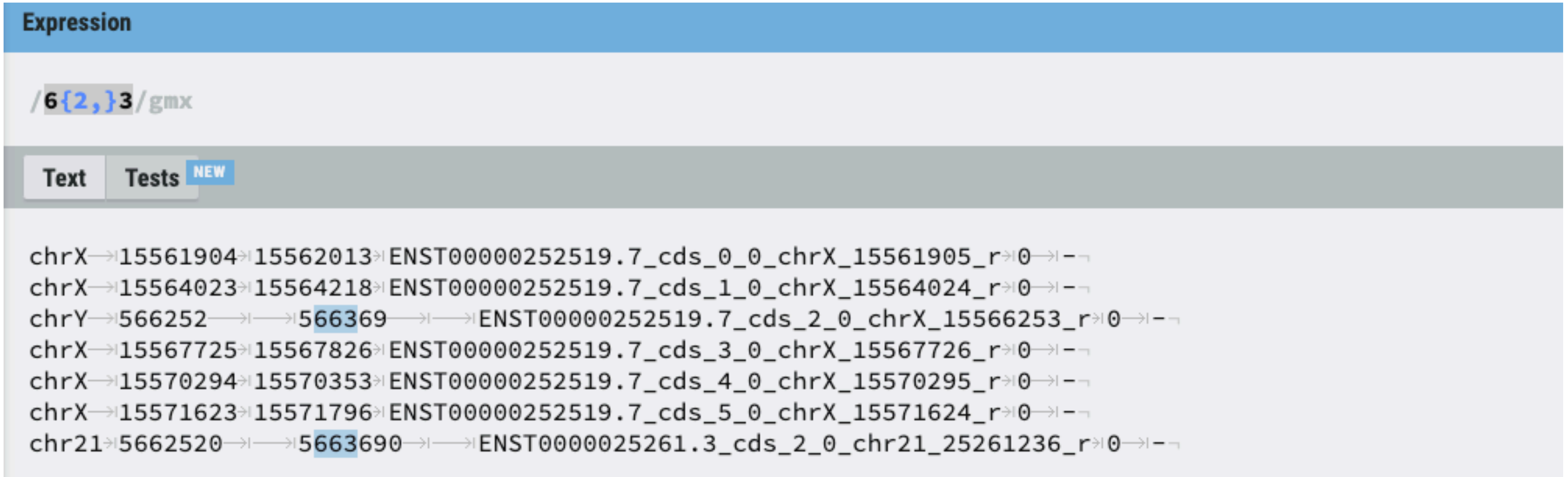
- 6{2,}2



```
/6{2,}2/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

- 6{2,}3



```
/6{2,}3/gmx
```

```
chrX	15561904	15562013	ENST00000252519.7_cds_0_0_chrX_15561905_r	0	-
chrX	15564023	15564218	ENST00000252519.7_cds_1_0_chrX_15564024_r	0	-
chrY	566252		566369		ENST00000252519.7_cds_2_0_chrX_15566253_r	0	-
chrX	15567725	15567826	ENST00000252519.7_cds_3_0_chrX_15567726_r	0	-
chrX	15570294	15570353	ENST00000252519.7_cds_4_0_chrX_15570295_r	0	-
chrX	15571623	15571796	ENST00000252519.7_cds_5_0_chrX_15571624_r	0	-
chr21	5662520		5663690		ENST0000025261.3_cds_2_0_chr21_25261236_r	0	-
```

Unix103

GREP (Global Regular Expression Print)
Where GREP Came From - Computerphile

- **Regular Expressions (regex)**
  - Describes text and text patterns
  - Do not have to contain literal text
  - Comprised of metacharacters
  - Metacharacters are processed by ‘parsing’

- **Text Searching versus Grep Searching**
  - Text searching is literal, whereas GREP searching is abstract and conditional
  - Text is finite - GREP is flexible
  - Text looks for characters (what) - GREP looks for locations (where and what)
  - History of GREP
  - Also known as Regular Expression Parser
  - Original command-line text search utility

    In sed (stream editor):

    ```
    g/re/p or global/regular expression/print
    ```

  - GREP remembers what it found and can be directed to re-use it
  - GREP searches for patterns and most text can be described as a pattern

-

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

A Phone number as an example

```
> 979-694-1234             # <- As 'mortals' see it
> \d\d\d-\d\d\d-\d\d\d\d   # <- As GREP see it
> \d+-\d+-\d+

> cd
> cd DB2022_xx
> mkdir 06_Lecture
> cd 06_Lecture
> cp -v /vol_b/zzStorage/list.txt .
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

GREPping list.txt

```
# Inspect the file
> cat list.txt
> less list.txt

# Search for the string “111”
> grep 111 list.txt or

> grep '111' list.txt or

> grep "111" list.txt or  <—Preferred

# Generally this is not recommended for performance issues…
> cat list.txt | grep "111" or and so on...
```

Unix103

GREP (Global Regular Expression Print)

Pattern Searching

```
# To add number lines to your search use the -n flag
> cat list.txt | grep -n "111"

# To count the number of hits use the -c flag
> cat list.txt | grep -c "111"

# To recursively search for a given pattern (e.g., ‘111’) inside the
files of a directory use the -R flag
> grep -R "111" .

# Flag -n = Prefix each line of output with the 1-based line number
> grep -R -n "111" .

# Flag -l = Suppress normal output; instead print the name of each
input file from which output would normally have been printed. The
scanning will stop on the first match
> grep -R -l "111" .
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

Using Positional Assertions

```
> cat list.txt | grep "111"

> cat list.txt | grep "^1"

> cat list.txt | grep "1$"

> cat list.txt | grep "12"

> cat list.txt | grep "12[34]"

> cat list.txt | grep -E "[[:digit:]]+2[34]"
```

Unix103

GREP (Global Regular Expression Print)

Pattern Searching

```
# Search for string “Banana”
> grep "Banana" list.txt

# Search for string “Apple”
> grep "Apple" list.txt

# Search for string “Banana AND Apple” <=Does not work
> grep "Banana|Apple" list.txt

# Activating extended regex (-E or -P flags)
# Search for string "Banana AND Apple" <=Does work
> grep -E "Banana|Apple" list.txt
> grep -P "Banana|Apple" list.txt

# Activating ignore case (-i flag)
> grep -E -i "Banana|Apple" list.txt

# Activating word-regex (-w flag)
> grep -E -i -w "Banana|Apple" list.txt

# Activating invert-match (-v flag)
> grep -E -i -v "Banana|Apple" list.txt
```

Unix103

GREP (Global Regular Expression Print)

Pattern Searching

Understanding Character Classes or “[ ]”

```
# "a[xyz]c" matches "axc" or "ayc" or “azc" does not match "axyzc"
> echo "axc" | grep "a[xyz]c"
> echo "ayc" | grep "a[xyz]c"
> echo "azc" | grep "a[xyz]c"

> echo "axyzc" | grep "a[xyz]c"

# "a[a-z]c" can match "abc"
> echo "abc" | grep "a[a-z]c"

# "a[a-zA-Z]c" can match "abc" or "aBc"
> echo "abc" | grep "a[a-zA-Z]c"
> echo "aBc" | grep "a[a-zA-Z]c"
> echo "ABc" | grep "a[a-zA-Z]c"

# "a[^xyz]c" does not match "axc" nor "ayc" nor "azc"
> echo "axc" | grep "a[^xyz]c"
> echo "ayc" | grep "a[^xyz]c"
> echo "axc" | grep "a[^xyz]c"

# "a[^a-z]c" does not match "abc" nor "ayc" nor "azc"
> echo "abc" | grep "a[^a-z]c"
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

The "*", Matches zero or more repeats of the previous item

```
# "ab*c" matches "abc" or "abbbbbc" or "ac" but not "axc"
> echo "abc" | grep "ab*c"

> echo "abbbbbc" | grep "ab*c"

> echo "ac" | grep "ab*c"

> echo "axc" | grep "ab*c"
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

The "( )", Allows repeats Multiple Times

```
# "ab*c" matches "abbbbc"
> echo "abbbbc" | grep "ab*c"

# "(ab)*c" matches "ababababc"
> echo "ababababc" | grep "ab*c"

> echo "ababababc" | grep "(ab)*c"

> echo "ababababc" | grep -E "(ab)*c"

> echo "ababababc" | grep -P "(ab)*c"
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

Understanding Greedy Characters

```
# The quantifiers +, *, ? and {} are "greedy"
# That is, they will always make the longest possible match possible
to a given pattern, so if your pattern is E+(one or more E’s) and
your text contains “EEEE”, the pattern matches all the E’s at once,
not just the first one

# +, Quantifier. Match one or more of the preceding token
> echo "abababababc" | grep -P "(ab)+"

# *, Quantifier. Match 0 or more of the preceding token
> echo "abababababc" | grep -P "(ab)*"

# ?, Quantifier. Match between 0 and 1 of the preceding token
> echo "abababababc" | grep -P "(ab)?"

# Dot, Matches any character except a line break
> echo "abababababc" | grep -P "(ab)."

> echo "abababababc" | grep -P "(ab){4,}"

> echo "abababababc" | grep -P "(ab){5,}"
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

Understanding Greedy Characters

```
# ^, Matches beginning of a line (unless used in a character class)
# $, Matches end of a line (unless used in a character class)
# You can combine "^" and "$" within a pattern to force a match to
constitute an entire line

> echo -e "foo" | grep -P "^foo$"

> echo "foo"

> echo "foo" | grep "^foo$"

> echo -e "foo\nfoofoo"

> echo -e "foo\nfoofoo" | grep "^foo$"
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

In Summary, Regexs are:

- Symbols representing a text pattern
- Formal language interpreted by a regular expression processor (e.g., grep)
- Used for matching, searching and replacing text
- Are NOT a programming language
- Have a set of rules
- These rules tell the computer what to do
- Most programming languages use regular expressions
- Most programmers probably used regular expressions the most, but they have no variables and you cannot add “1+”
- A regex ‘matches’ text if it describes the text
- Text ‘matches’ a regex if it is correctly described by the regex

Unix103

Introduction to SED: Stream Editor

Regular SED

```
Typical Command: sed 's/a/b/'
                      s=substitution
                      a=search string
                      b=replacement string

> echo "upstream" | sed 's/up/down/' <-Find and Replace

> echo "upstream and upward" | sed 's/up/down/'

> echo "upstream and upward" | sed 's/up/down/g'

> echo "upstream and upward" | sed 's:up:down:g'

> echo "upstream and upward" | sed 's|up|down|g'

> echo "during daytime we have sunlight" | sed ’s/day/night/'

> echo "during daytime we have sunlight" | sed -e ’s/day/night/' -e 's/sun/moon/'
```

Unix103

Introduction to SED: Stream Editor

Using Regex

To use extended Regex:

```
> echo "who needs vowels?" | sed 's/[aeiou]/_/g'



# Does not work (No 'E' Flag)
> echo "who needs vowels?" | sed 's/[aeiou]+/_/g'



# Does work ('E' Flag Activated)
> echo "who needs vowels?" | sed -E 's/[aeiou]+/_/g'
```

Unix103

Introduction to SED: Stream Editor
Using Regex

Using Backreferences:

```
# Does not work (No 'E' Flag)
> echo "daytime" | sed    's/(...)time/\1light/'


# Does work (escaping parenthesis)
> echo "daytime" | sed    ’s/\(...\)time/\1light/'


# Does work ('E' Flag Activated)
> echo "daytime" | sed –E ’s/(...)time/\1light/'

> echo "daytime" | sed –E 's/(.)time/\1light/'

> echo "daytime" | sed –E ’s/(.+)time/\1light/'

> echo "FirstName LastName" | sed –E 's/([A–Za–z]+) ([A–Za–z]+)/\2, \1/'
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

How many sequences are present in the cds file?

```
>  wget http://ftp.ensembl.org/pub/release-105/fasta/homo_sapiens/cds/Homo_sapiens.GRCh38.cds.all.fa.gz

> gunzip -k Homo_sapiens.GRCh38.cds.all.fa.gz

> wget http://ftp.ensembl.org/pub/release-105/fasta/homo_sapiens/cds/CHECKSUMS

> wget http://ftp.ensembl.org/pub/release-105/fasta/homo_sapiens/cds/README

#Verify File Integrity
> sum Homo_sapiens.GRCh38.cds.all.fa.gz
```

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

1.How could we determine how many "Ns" are in chr22?

3.How could we determine how many nucleotides are in chr22?

5.How could we determine how many adenosines are there on chr22?

Unix103

GREP (Global Regular Expression Print)
Pattern Searching

Let's create a file called patterns/hg.b38.chr22.fa in your home directory

```
> mkdir patterns

> cd patterns

> wget 'ftp://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz' -O chr22.fa.gz

> gunzip -k chr22.fa.gz

> mv chr22.fa hg.b38.chr22.fa
```

Unix103

BIOL647
Digital Biology

Rodolfo Aramayo