Using Regular Expressions in the D2L Quizzing Tool

Version 1.1

305 King Street West, Suite 200
Kitchener, Ontario
N2G 1B9 Canada

www.Desire2Learn.com

Please send feedback on Desire2Learn documentation to:
Docs@Desire2Learn.com.

Table of Contents

1.0 Introduction - Using Regular Expressions in D2L

Regular expressions are used in the D2L Learning platform Quizzing tool. Using regular expressions gives an instructor the ability to evaluate various combinations of student inputs to questions.

1.1 Using Regular Expressions

Currently, you can only use regular expressions in Quizzing, in the Short Answer (SA) and Fill in the Blank (FIB) question types. To use the Regular Expression you need to check the box that says “Regular Expression”, below the answer box.

1.2 How to enter regular expression syntax

From the Quiz Question Library, create a new SA or FIB question.

1.2.1 Select Question Type

Select Short Answer Question or Fill in the Blanks from the drop down menu. Click **Go**.

1.2.2 Choose a Title

Optionally, enter a title, so you can easily look it up in the future.

1.2.3 Points/Difficulty

Assign the Points and Difficulty.

1.2.4 Question Text

Enter the question text. To refer to variables that you are using, enclose them in {curly} braces. *In this example, use the text shown below:*

Text
What animal meows?

1.2.5 Box Size

Input Box allows you to choose how large or small you want the area to be for the answer. A larger box might tell the

Rows – Options range from 2-6

Columns – Options range from 20, 40 or 60

Size used in the sample question:

Row size = 2, Column size = 20

1.2.6 Add Answer

Check the checkbox that says to use Regular Expression

The answer used in the sample question:

Answer
[C|c]at

A list of regular expression Syntax with examples is given in Section 2.0 of this document.

1.2.7 Optional Areas

- Add an image to the question
- Question Hint
- Question Feedback

1.2.8 Save and Preview

Click **Save** (or click **Preview**, to see the question before it is saved)

2.0 Regular Expression Syntax

A regular expression is a pattern of text that consists of ordinary characters (for example, letters a through z) and special characters, known as metacharacters. The pattern describes one or more strings to match when searching a body of text. The regular expression serves as a template for matching a character pattern to the string being searched.

The following table contains the complete list of metacharacters and their behavior in the context of regular expressions:

Character	Description	Example
\	Marks the next character as either a special character, a literal, a backreference, or an octal escape. The sequence '\\' matches "\" and "\(" matches "(".	'**n**' matches the character "**n**". '**\n**' matches **a newline character**.
^	Matches the position at the beginning of the input string. If the RegExp object's Multiline property is set, ^ also matches the position following '\n' or '\r'.	**^cat** matches any string that begins with **cat**
$	Matches the position at the end of the input string. If the RegExp object's Multiline property is set, $ also matches the position preceding '\n' or '\r'.	**cat$** matches any string that ends with **cat**
*	Matches the preceding character or subexpression zero or more times. * is equivalent to {0,}	**be*** matches **b** or **be** or **beeeeeeeeee** **zo*** matches "**z**" and "**zoo**".
+	Matches the preceding character or subexpression one or more times. + is equivalent to {1,}.	**be+** matches **be** or **bee** but not **b** '**zo+**' matches "**zo**" and "**zoo**", but not "**z**".
?	Matches the preceding character or subexpression zero or one time. ? is equivalent to {0,1}	**colou?r** matches **color** or **colour** but not **colouur** "**do(es)?**" matches the "**do**" in "**do**" or "**does**".

Character	Description	Example
?	When this character immediately follows any of the other quantifiers (*, +, ?, {n}, {n,}, {n,m}), the matching pattern is non-greedy. A non-greedy pattern matches as little of the searched string as possible, whereas the default greedy pattern matches as much of the searched string as possible.	In the string "**oooo**", '**o+?**' matches a single "**o**", while '**o+**' matches all '**o**'s.
()	Parentheses. Creates a substring or item that metacharacters can be applied to.	**a(bee)?t** matches at or **abeet** but not **abet**
{n}	n is a nonnegative integer. Matches exactly n times. For example, 'o{2}' does not match the 'o' in "Bob," but matches the two o's in "food".	**[0-9]{3}** matches any three digits
{n,}	n is a nonnegative integer. Matches at least n times. For example, 'o{2,}' does not match the "o" in "Bob" and matches all the o's in "foooood". 'o{1,}' is equivalent to 'o+'. 'o{0,}' is equivalent to 'o*'.	**[0-9]{3,}** matches any three or more digits
{n,m}	m and n are nonnegative integers, where n <= m. Matches at least n and at most m times. For example, "o{1,3}" matches the first three o's in "fooooood". 'o{0,1}' is equivalent to 'o?'. Note that you cannot put a space between the comma and the numbers.	**[0-9]{3,5}** matches any three, four, or five digits
.	Matches any single character except "\n". To match any character including the '\n', use a pattern such as '[\s\S]'.	**cat.** matches **catT** and **cat2** but not **catty**
(*pattern*)	Matches pattern and captures the match. The captured match can be retrieved from the resulting Matches collection, using the SubMatches collection in VBScript or the $0$9 properties in JScript. To match parentheses characters ( ), use '\(' or '\)'.	
(?:*pattern*)	Matches pattern but does not capture the match, that is, it is a non-	**'industr(?:y|ies)** is a more economical

Character	Description	Example
	capturing match that is not stored for possible later use. This is useful for combining parts of a pattern with the "or" character (|).	expression than **'industry|industries'.**
(?=*pattern*)	Positive lookahead matches the search string at any point where a string matching pattern begins. This is a non-capturing match, that is, the match is not captured for possible later use. Lookaheads do not consume characters, that is, after a match occurs, the search for the next match begins immediately following the last match, not after the characters that comprised the lookahead.	'**Windows (?=95|98|NT|2000)**' matches "**Windows**" in "**Windows 2000**" but not "**Windows**" in "**Windows 3.1**".
(?!*pattern*)	Negative lookahead matches the search string at any point where a string not matching pattern begins. This is a non-capturing match, that is, the match is not captured for possible later use. Lookaheads do not consume characters, that is, after a match occurs, the search for the next match begins immediately following the last match, not after the characters that comprised the lookahead.	'**Windows (?!95|98|NT|2000)**' matches "**Windows**" in "**Windows 3.1**" but does not match "**Windows**" in "**Windows 2000**".
x|y	Matches either x or y. For example, 'z|food' matches "z" or "food". '(z|f)ood' matches "zood" or "food".	**July (first|1st|1)** will match **July 1st** but not **July 2**
[xyz]	A character set. Matches any one of the enclosed characters. For example, '[abc]' matches the 'a' in "plain".	**gr[ae]y** matches **gray** or **grey**
[^xyz]	A negative character set. Matches any character not enclosed. For example, '[^abc]' matches the 'p' in "plain".	**1[^02]** matches **13** but not **10** or **12**
[a-z]	A range of characters. Matches any character in the specified range. For example, '[a-z]' matches any lowercase alphabetic character in the range 'a' through 'z'.	**[1-9]** matches any single digit **EXCEPT 0**
[^a-z]	A negative range characters. Matches any character not in the specified	'**[^a-z]**' matches any character not in the

Character	Description	Example
	range.	range '**a**' through '**z**'
\b	Matches a word boundary, that is, the position between a word and a space.	'**er\b**' matches the **'er' in "never"** but not the **'er' in "verb".**
\B	Matches a nonword boundary.	'**er\B**' matches the **'er' in "verb"** but not the **'er' in "never"**.
\cx	Matches the control character indicated by x. The value of x must be in the range of A-Z or a-z. If not, c is assumed to be a literal 'c' character.	**\cM** matches a **Control-M** or **carriage return character**.
\d	Matches a digit character	Equivalent to [0-9]
\D	Matches a nondigit character	Equivalent to [^0-9]
\f	Matches a form-feed character.	Equivalent to \x0c and \cL
\n	Matches a newline character.	Equivalent to \x0a and \cJ
\r	Matches a carriage return character.	Equivalent to \x0d and \cM
\s	Matches any whitespace character including space, tab, form-feed, etc.	Equivalent to [\f\n\r\t\v]
\S	Matches any non-white space character.	Equivalent to [^ \f\n\r\t\v]
\t	Matches a tab character.	Equivalent to \x09 and \cI
\v	Matches a vertical tab character.	Equivalent to \x0b and \cK
\w	Matches any word character including underscore.	Equivalent to '[A-Za-z0-9_]'
\W	Matches any nonword character.	Equivalent to '[^A-Za-z0-9_]'
\xn	Matches n, where n is a hexadecimal escape value. Hexadecimal escape values must be exactly two digits long. Allows ASCII codes to be used in regular expressions.	'**\x41**' matches "**A**". '**\x041**' is equivalent to '**\x04**' & "**1**"
\num	Matches num, where num is a positive integer. A reference back to captured	'**(.)\1**' matches two consecutive identical

Character	Description	Example
	matches.	characters
\n	Identifies either an octal escape value or a backreference. If \n is preceded by at least n captured subexpressions, n is a backreference. Otherwise, n is an octal escape value if n is an octal digit (0-7).	"**\11**" and "**\011**" both match a tab character. "**\0011**" is the equivalent of "**\001**" & "**1**".
\nm	Identifies either an octal escape value or a backreference. If \nm is preceded by at least nm captured subexpressions, nm is a backreference. If \nm is preceded by at least n captures, n is a backreference followed by literal m. If neither of the preceding conditions exists, \nm matches octal escape value nm when n and m are octal digits (0-7).	
\nml	Matches octal escape value nml when n is an octal digit (0-3) and m and l are octal digits (0-7).	
\un	Matches n, where n is a Unicode character expressed as four hexadecimal digits. For example, \u00A9 matches the copyright symbol (©).	

3.0 Sample uses of Regular Expressions in Quizzing

Question Type	Question Text	Answers
Fill in the Blanks (FIB)	A _____ wags his tail. He eats dog _____ twice a day.	Blank #1 = [D|d]og Blank #2 = [F|f]ood
Short Answer (SA)	What is the word that describes red, blue, green, yellow, pink, etc.?	colou?r*
Fill in the Blanks (FIB)	You are always told as a kid to chew your food_____ times.	(twenty-four|24|twenty four|twenty four)