Regular Expressions for URL Targeting

Regular expressions are used widely in computer programming, allowing people to search through many lines of code for a specific piece of data or find a very specific and precise set of information that would otherwise take them many hours of searching and sifting. At Qualaroo, we use regular expressions (shortened to regex and regexes) to allow you to target your surveys to a specific set of pages, or to URLs that are more complex.

There's a lot of information about regexes on the internet, and a lot of it isn't really applicable to how you'll be using them at Qualaroo, so we've created this guide to Regexes for URL Targeting to help you get started.

We'll cover the following topics:

Regular Expression Basics

Regular Expressions use punctuation characters to tell a computer what to do when looking for something -

- start looking
- find this kind of character
- this many
- look for this particular thing and then ignore the result
- stop looking

Regexes look really complicated sometimes, and it can take a lot of work to figure out what is actually going on. That's because regular expressions were made by programmers, for computers, and aren't

designed to be user-friendly. But don't worry! Once you start using regular expressions, and the characters, they'll get easier.

Here's a list of the things you'll be using most often*:

Regex Characters

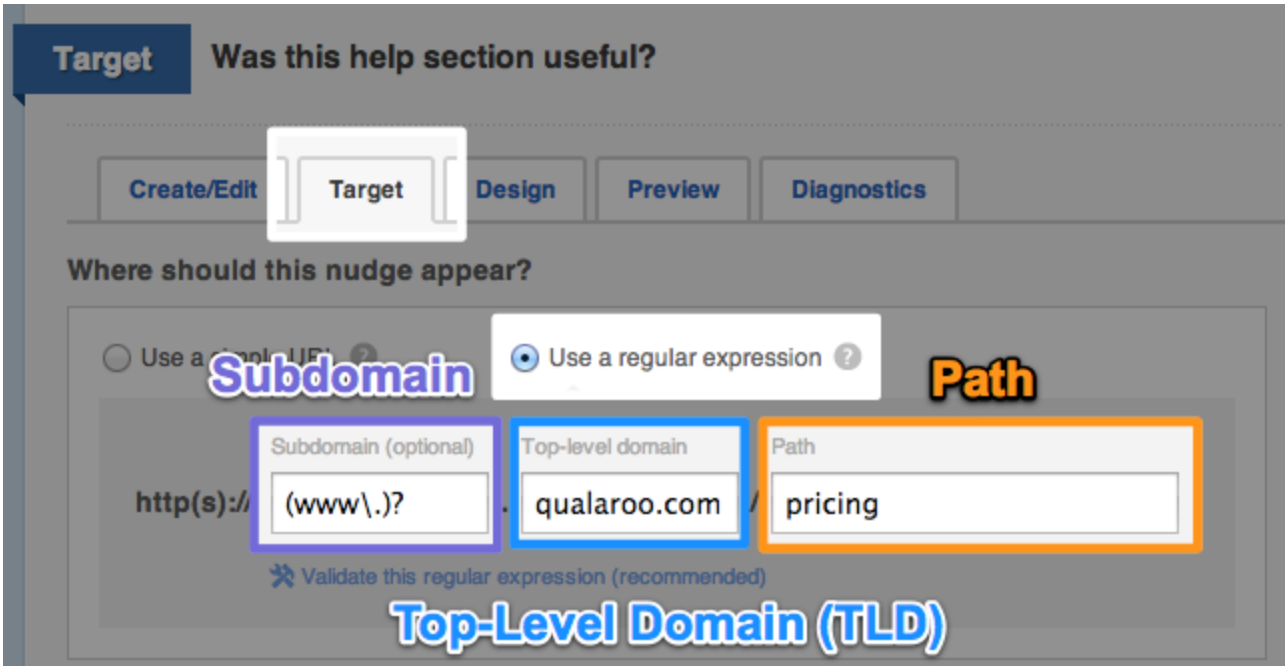
`.` – Matches any character.
`a*` – Matches zero or more of the preceding character (in this case, a)
`a+` – Matches one or more of the preceding character.
`a?` – Preceding character is optional. Matches zero or one occurrence.
`\d` – Matches any single digit
`\w` – Matches any word character (letters, numbers and underscore).
`(a|b)` - Matches a OR b
`[xyz]` – Matches any single character in the brackets: x, y, OR z.
`[^a-z]` – When inside of a character class, the ^ means NOT. Here, match anything that is NOT a lowercase letter.
`[A-Z]` – Capital A through Capital Z.
`[a-z]{2}` – Exactly 2 a-z letters. (hint: this is useful for URLs like testing-site.com/**en** or /**fr**)

(* Please see the Note on Regular Expression Characters in the More Resources section.)

Regular Expressions for URL Targeting

The Regular Expression Interface

At Qualaroo, we've made a special effort to make the regex interface as easy to use as possible. We split the URL into three parts: **Subdomain**, **Top-Level Domain (TLD)** and **Path**.



The **Subdomain** is everything between the http:// and yourwebsite.com. This can be www, development, help, blog, hotels, or any other section of your website that comes before the actual domain. You will be able to use regular expressions here.

The **TLD** is the main part of your website - time.com, wikipedia.org, stanford.edu, ox.ac.uk. Once you've entered the TLD in the appropriate regex field, Qualaroo's interface will remember it for the next survey you make. All you have to do is start typing, and it will fill in the rest.

The **Path** is everything that comes after the TLD, all the pages, folders, sections, images, search results - everything on your website. This is where you will probably be using most of your regular expressions, so you can target your surveys to specific sections, types of pages, certain keywords, and any other conditions you want.

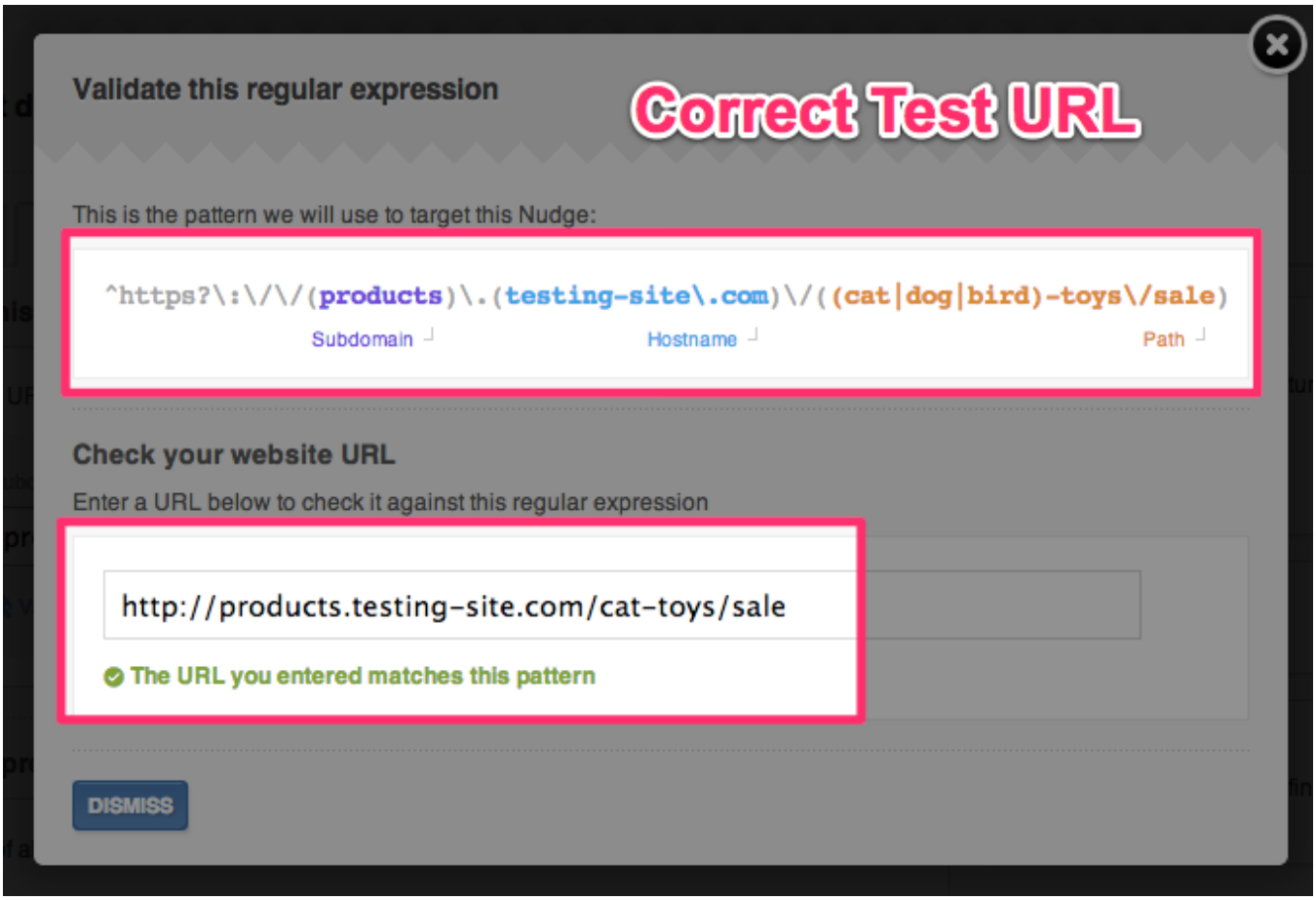
Throughout this guide, we will go over the regex characters and their meanings, examples of when to use each character, and provide examples of how you should enter your regex in the interface. These examples will look like this:

To use this example, you would add the following to your regex fields:

Regular Expressions for URL Targeting

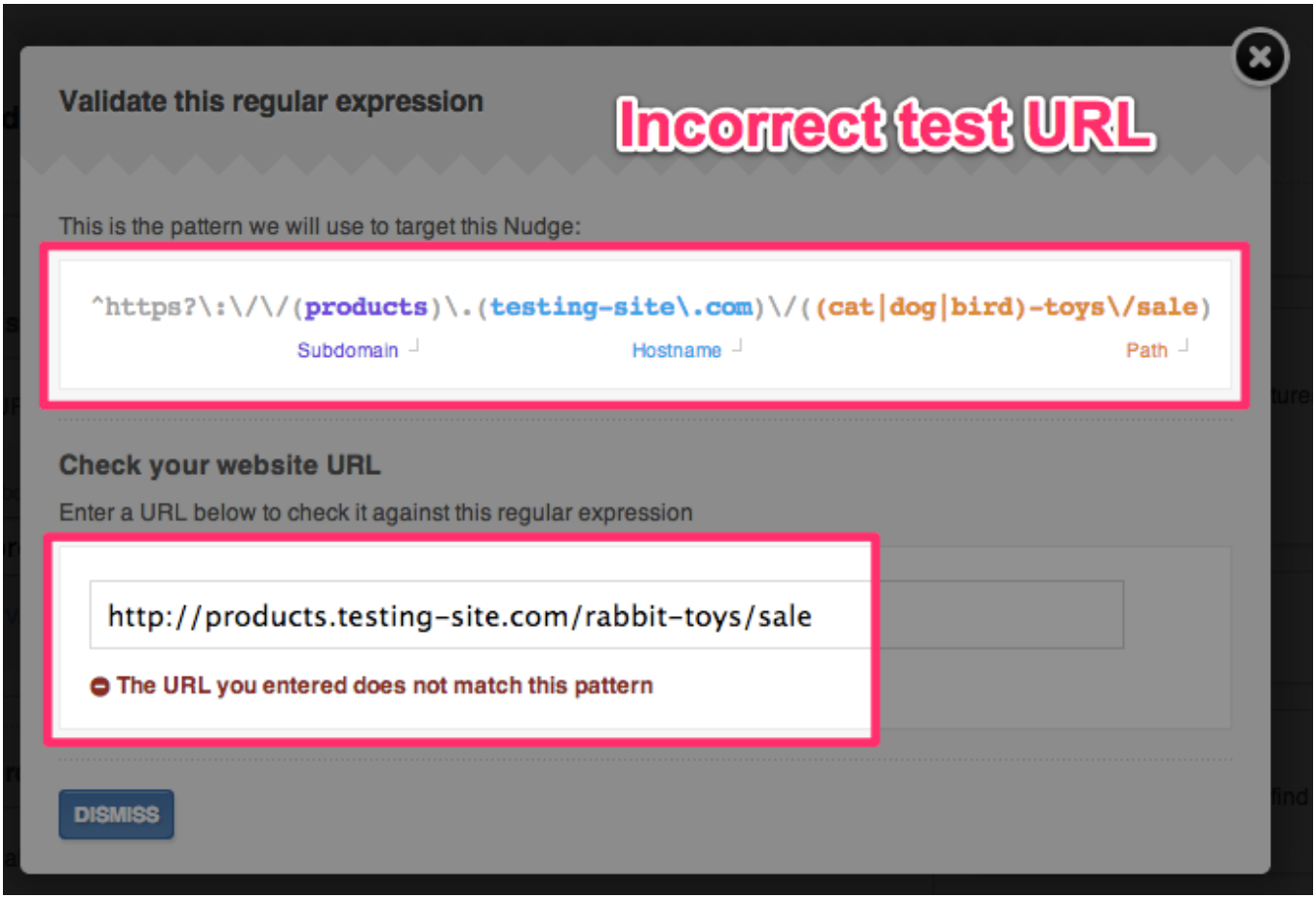
Subdomain: `(www\.)?`
TLD: `qualaroo.com`
Path: `pricing`

The colors we used in these examples were chosen to match the regex validator. This is an extra tool we created to let you validate your regex. Using this tool will allow you to make sure that the regex you created matches the pages you want it to. The validator will tell you if a URL matches or not.



You should always try to test several URLs - some that match your pattern, and you would like to target with the survey, and some that don't match, to make sure they aren't being included by mistake.

Regular Expressions for URL Targeting



Note: If you are using an older survey, it may not have the new interface. Please see the section on "Formatting URLs in the old Regex Interface" for how to format your regexes.

Backslash - Escape!

You'll also see a lot of these: \ (backslash) These have a very special use - they "escape" the character that comes afterward, and tell the computer to treat it differently. If you don't escape the special characters, they are treated as part of the regex. In a URL, you have to escape the periods:

```
helloworld\.html
```

This tells the computer that you mean helloworld -dot- html, not helloworld -any character- html.
You will also need to escape slashes (/) and you will end up with a pattern like this:

```
\/
```

Put them together, and you will get:

`helloworld\.html\/`

You can see examples of where we use escape characters in the regex validator:

In purple, Qualaroo automatically escapes the periods and slashes in between the three fields, so you don't have to worry about those. In pink, we escape the backslashes, periods and question marks that we want to include in the URL as-is.

At the end of the regex in green, we leave the period and asterisk unescaped, because we want to use them as special characters.

Question Mark - Not Required

The question mark is one of the most versatile characters in regular expressions. It is used to mean that whatever came before it is optional - there can be 0 or 1 occurrence.

You can use it for single characters:

`cats?` will match `cat` and `cats`

Or for whole words by using parentheses:

`(an)?droid` will match `android` and `droid`

Some websites are set up so that you can go to both **www.site.com** and **site.com**. If you're using a regex on a website like this, you will want to use the question mark to make sure pages on the www.site.com version show up.

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `site.com`
Path:

If your website always uses www.site.com, then just make sure to add `www` in the Subdomain field.

To use this example, you would add the following to your regex fields:
Subdomain: **`www`**
TLD: **`site.com`**
Path:

Note that you will not have to escape the period after the www, Qualaroo will take care of that for you. In the new regex interface, you no longer need to deal with formatting your regex for https. However, if you're using an older survey, you may need to know how to do this so that your regex works properly. Please see the last section for more information.

Digits and Words

Digits

We also use backslashes with `\d` and `\w` - this tells the computer you want to use the d and w as part of the regular expression, not as part of the search:

`example-file-\d\.html`

will give you

```
example-file-0.html
example-file-1.html
example-file-2.html
...
example-file-9.html
```

Remember that `\d` is only for a *single* digit, so if you wanted to have longer numbers, you would use this:

`example-file-\d+\.html`

This tells a computer to match any digit, at least one digit. So the results from this search would get you

```
example-file-0.html
example-file-1.html
```

```
example-file-2.html
...
example-file-10.html
...
example-file-2450.html
...
```

and so on.

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `website.com`
Path: `example-file-\d+\.html`

Word Characters

Say you want to target pages with human-readable names that don't use special characters. Maybe these are photo album folders, or documents that your users have created. We can do this easily by using the \w function:

`\w+\.php`

This will match any file with only word characters (a-z, A-Z and _) in the name. Here are some example files that this regex will match:

```
paris.php
Melbourne.php
McMurdo_Field_Work_Summer_2012.php
Christmas1997.php
SpencerBirthday_Age5.php
SFTrip20030206.php
```

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `website.com`
Path: `\w+\.php`

Dot-Star: Anything Goes!

One combination of characters that you will be using a lot is `.*` This means "any character" "zero or

more times" and is great for all sorts of URL pieces:

`.*\/help`

Will match

website.com/**users**/help
website.com/**photos**/help
website.com/**cats**/help
website.com/**documentation**/help
website.com/**any-P0ss1ble_characters**/help

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `website.com`
Path: `.*\/help`

Note: If you want to target every page on your website, it's a lot easier to use the Simple URL Targeting field. That way, you only need to use a * at the end of your URL, and Qualaroo will do the rest.

The Or Pipe - Smoke 'em If You Got 'em

The OR character | (also called a "pipe") is very useful when you just want your survey to show on a few specific pages, or in two or more sections. Say you want to match these pages:

blog.mycats.com/peggysue.html
blog.mycats.com/turbo.html

To use this example, you would add the following to your regex fields:
Subdomain: **`blog`**
TLD: **`mycats.com`**
Path: **`(peggysue|turbo)\.html`**

You can use this anywhere in the regex, and as many times as you like, too:

If you want to target the gallery pages of the Brazil, Chile and Argentina sections on largetravelcompany.com, using the OR pipe makes this very easy.

To use this example, you would add the following to your regex fields:
Subdomain: `(brazil|chile|argentina)`
TLD: `largetravelcompany.com`
Path: `gallery`

You can even use multiple OR characters in the same regex, to target several pages across multiple subdomains.

To use this example, you would add the following to your regex fields:
Subdomain: `(library|parks)`
TLD: `smalltown.gov`
Path: `(kids|special_events|holiday)-activities\/sign_up_form`

Using Parentheses, Brackets and Sets

Parentheses and brackets are very useful characters for grouping words and ranges of letters and numbers, and being very precise in exactly which URLs you are targeting. From before, these are the characters and their uses:

`(a|b) - Matches a OR b`
`[xyz]` – Matches any single character in the brackets: x, y, OR z.
`[a-z]{2}` – Exactly 2 a-z letters.

Parentheses

Using the parentheses and the OR pipe, you can tell your regex to target one word (sometimes called a "string") or another in your URL. This is what we used earlier:

To use this example, you would add the following to your regex fields:
Subdomain: `blog`
TLD: `mycats.com`
Path: `(peggysue|turbo)\.html`

and it targets `blog.mycats.com/peggysue.html` and `blog.mycats.com/turbo.html`

Brackets and Curly Braces

You can use the brackets to target a range of letters (like a-z or a-f) or numbers (0-9, 1-5). You can also

Regular Expressions for URL Targeting

use the curly brackets to ask for a specific number of letters or numbers, or a range that you will allow.

If you want to target a survey across several sections of your website, you can use the OR pipe, or the brackets, if they have a similar format. For example, if you want to target the following sections:

`www.international.com/`**`en`**`/products`
`www.international.com/`**`ca`**`/products`
`www.international.com/`**`uk`**`/products`
`www.international.com/`**`au`**`/products`
`www.international.com/`**`nz`**`/products`

You can use the brackets to show the letter range `[a-z]` and curly brackets to determine how many letters you will allow `{2}`.

To use this example, you would add the following to your regex fields:
Subdomain: **`www`**
TLD: **`international.com`**
Path: **`[a-z]{2}\/products`**

You can put them together to target pages that have a specific format in the URL, like 6 letters and 8 numbers.

To use this example, you would add the following to your regex fields:
Subdomain: **`(www\.)?`**
TLD: **`gifts-for-everyone.org`**
Path: **`holiday\/special_deals\/[a-z]{6}-[0-9]{8}`**

This regex will match pages like:

`www.gifts-for-everyone.org/holiday/special_deals/`**`lawnmo-45061367`**
`www.gifts-for-everyone.org/holiday/special_deals/`**`hairdr-00002239`**
`www.gifts-for-everyone.org/holiday/special_deals/`**`poster-08825041`**

It will not match the following pages:

`development.gifts-for-everyone.org/holiday/special_deals/`**`lawnmo-45061367`**
Wrong subdomain
`www.gifts-for-everyone.org/holiday/special_deals/`**`lawnmower-45061367`**
Wrong number of letters
`www.gifts-for-everyone.org/holiday/special_deals/`**`lawnmo-451367`**

Regular Expressions for URL Targeting

Wrong number of numbers

If you'd like more flexibility in the ranges of letters and numbers in the pages you want to target, the curly brackets can be used for this as well. Say you want to target the same kinds of pages as before, but there can be between 4 and 8 letters, and between 2 and 8 numbers.

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `gifts-for-everyone.org`
Path: `holiday\/special_deals\/[a-z]{4,8}-[0-9]{2,8}`

This will allow the survey to be targeted at a wider range of pages, including:

`www.gifts-for-everyone.org/holiday/special_deals/`**`bowl-27`**
`www.gifts-for-everyone.org/holiday/special_deals/`**`heatlamp-00019223`**
`www.gifts-for-everyone.org/holiday/special_deals/`**`boots-4512`**
`www.gifts-for-everyone.org/holiday/special_deals/`**`catmitte-123380`**

Multi-Digit Number Ranges

One of the very few restrictions in regular expressions is that it doesn't know how to deal with numbers greater than 9. To do ranges in the double or triple digits, you must specify the range of each digit. For example, if you want to target pages with numbers 25-50, you would have to use a few sets of numbers and ranges. Specifically, you must define 25-29, then 30-49, then 50. First we will make each range, and then put them together into a single regex.

`2[5-9]` will match 25-29
`(3|4)[0-9]` will match 30-30 or 40-49
`50` will match 50

Since we want to target any of these pages, we use the OR pipe to separate each number range.
`(2[5-9]|(3|4)[0-9]|50)`

To use this example, you would add the following to your regex fields:
Subdomain: `www`
TLD: `learning_math_is_fun.com`
Path: `chapter_(2[5-9]|(3|4)[0-9]|50)`

Regular Expressions for URL Targeting

Regular Expressions for URL Targeting

Lookahead and Lookbehind Delimiters

Sometimes you may want to target a wide range of pages, but not include some that might otherwise be caught up (like mysite.com/products/**item-###**, /products/**seasonal-###** but excluding /products/**promo-###**)

Regular expressions have this great feature called "lookahead" and "lookbehind" and these are used quite literally, to look ahead in the URL to target something, or to look behind a particular portion and make sure that another part is included. These can be used in the "positive" or "negative" sense, meaning a regex can look ahead and explicitly include (positive) or exclude (negative) a particular thing.

Negative Lookahead

One of the most useful implementations of this is the **negative lookahead**. It allows you to exclude whole sets of pages, files, subdomains, or any other part of the URL you don't want to target. In regex terms, it looks for something that is NOT followed by something else. You specify what you DON'T want to include, and put it inside of these characters **`(?!StuffYouDontWant)`**

Example 1: Excluding a Folder

I want to target my survey to all the pages on http://mysite.com/photos/, /cats/, and /documentation/ but not /users/ or any other single pages.

To use this example, you would add the following to your regex fields:
Subdomain: **`(www\.)?`**
TLD: **`mysite.com`**
Path: **`(?!users)\/.*`**

This will target any page on my site in a subfolder, EXCEPT for all pages in the /users/ section, and anything not in a subfolder. For example,

```
http://mysite.com/photos/NevadaDesert.html
http://mysite.com/photos/DeathValley.html
http://mysite.com/photos/Carrum.html

http://mysite.com/cats/PeggySue.html
http://mysite.com/cats/Turbo.html

http://mysite.com/documentation/Qualaroo.html
http://mysite.com/documentation/personalwebsite.txt
http://mysite.com/documentation/NextBigAndroidApp.php
```

```
http://mysite.com/documentation/1337Resume.html
```

And these pages do not show surveys:

```
http://mysite.com/users/admin
http://mysite.com/users/user1234
http://mysite.com/users/mom

http://mysite.com/contact
http://mysite.com/about-us
http://mysite.com/pricing/
http://blog.mysite.com/
```

Example 2: Excluding Groups of Pages

For the example used earlier in this section, say you want to target all the item pages in snacktastic.com/products/**item-###**, snacktastic.com/products/**seasonal-###** but excluding snacktastic.com/products/**promo-###**.

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `snacktastic.com`
Path: `products\/(?!promo)-\d+\/?`

The survey will show up on:

```
http://snacktastic.com/products/item-0733
http://snacktastic.com/products/item-561211
http://snacktastic.com/products/seasonal-559
http://snacktastic.com/products/seasonal-01223
```

But not

```
http://snacktastic.com/products/promo-001
http://snacktastic.com/products/promo-55776
```

You could also use the | character to get the same results.

To use this example, you would add the following to your regex fields:

Regular Expressions for URL Targeting

Subdomain: `(www\.)?`
TLD: `snacktastic.com`
Path: `products\/(item|seasonal)-\d+\/?`

There's lots of ways to get to the same answer with regular expressions. You might find yourself using one set of tools more frequently than another, and that's fine.

Positive Lookahead

A Positive Lookahead is basically the opposite of a negative lookahead - it defines a pattern that MUST appear in the URL for the page to be targeted. This is done by adding (?=) around whatever you want to require.

If you want to target any page on your site with "dragonfly" in the URL, you can do so very easily.

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `naturaljewelrydesigns.com`
Path: `.*(?=dragonfly)`

This regex will match any page with "dragonfly" **anywhere** in the URL path:

`http://www.naturaljewelrydesigns.com/`**`dragonfly`**
`http://www.naturaljewelrydesigns.com/products/rings/`**`dragonfly`**
`http://www.naturaljewelrydesigns.com/new_designs/green`**`dragonfly`**`.php`

You can also combine the positive lookahead with other regex characters:

To use this example, you would add the following to your regex fields:
Subdomain: `(www\.)?`
TLD: `naturaljewelrydesigns.com`
Path: `.*(?=dragonf(ly|lies|ire))`

A regex like this will match any page with the words `dragonfly`, `dragonflies` and `dragonfire` in the URL path.

More Resources

Note on Regular Expression Characters

It is important to point out that the regular expression features that are supported by Qualaroo depend on those supported by JavaScript. Consequently, these regular expression features are not supported by Qualaroo:

- `s` (single-line mode) and `x` (extended syntax) flag
- `\a \e \l \u \L \U \E \Q \A \Z \z \G` escape sequences
- `(?<= )` positive look-behind anchor and the `(?<! )` negative look-behind anchor
- `(?# )` comment and the other extended (?) syntaxes.

Regular Expression Testing

Regex101.com

There are a few tools out there that let you test your regular expressions and see if the URLs you want are matched. The one we use most frequently is http://regex101.com/ and it's free, provides lots of explanation for each part of the regular expression, and has a field where you can test out sample URLs to make sure they match or don't match.

At the bottom of that site is a grid of the most common regex characters. Don't be afraid to check it if you're not sure about something! There's a few more listed than what is in this guide, but their meanings aren't too difficult to understand. They also have a "complete reference" portion - most of this is targeted at programmers and while these characters can be used in your URL regular expression, most of them won't be relevant.

RegexPal.com

Another great tool is http://regexpal.com/. It is very similar to http://regex101.com/, but it allows you to test multiple URLs against your regex at the same time. Efficiency!

Regular Expressions for URL Targeting

Regular Expressions for Google Analytics

LunaMetrics

There is a fantastic ebook put out by LunaMetrics on Regular Expressions for Google Analytics. It goes over the basics of regular expressions, and how to apply them to Google Analytics. The is the URL for the PDF form of the book:
http://www.lunametrics.com/regex-book/Regular-Expressions-Google-Analytics.pdf

Lookaheads and Lookbehinds

RexEgg.com

This page goes further in depth on lookaheads and lookbehinds:
http://www.rexegg.com/regex-lookarounds.html In general, it's a really awesome regex tutorial site.

Forging Bravely into the Great Unknown

Google Search

As in most things, Google is Your Friend. However, Google is particularly difficult to use for help with regexes because it **specifically excludes punctuation** from search results. You know, things like `.*`, `(?! )`, `\/`, all that good stuff that makes regular expressions possible. A workaround is to preface every search with "regex" and use words to describe what you want to do. It's not the best solution, but until some whiz kid invents a new kind of search engine, this is what we have.

StackOverflow.com

Stack Overflow is a great resource for all things programming - you ask questions, other random programmer-type people on the internet help you out. Sometimes they aren't very friendly, sometimes they're awesome, sometimes they're awesome and nice but totally wrong! This is why you should always check a few different places to make sure they have the same answer, and always test your

regex before it goes live. A lot of people have asked a lot of questions though, and oftentimes, these threads are what will come up in your searches.

All Fun and No Play

RegexCrossword.com

If you feel like you need a little more practive before you're ready to start making these on your own, play around with http://regexcrossword.com. They have a series of crossword-style puzzles, increasing in difficulty, that will help you get comfortable with the regex characters and combinations. The puzzles are also a little nerdy, so you may end up wanting re-read *Hitchhiker's Guide To The Galaxy* by the time you're through.

Bits and Pieces

Here's some pieces of URLs in regex form that might come in handy. On the internet, copying this kind of thing isn't considered stealing - it's just more efficient! If you don't believe us, ask one of your engineers :)

`https?:\/\/(www\.)?` - Start any URL with http(s)://(www.)

`[a-zA-Z]{2}` - Exactly 2 letters (great for /EN, /fr, /No and other 2-letter language modifiers)

`.*(\.[Jj][Pp][Gg]|\.[Gg][Ii][Ff]|\.[Jj][Pp][Ee][Gg]|\.[Pp][Nn][Gg])` - Will match any .jpg, .gif, .jpeg or .png image file, capital or lower case

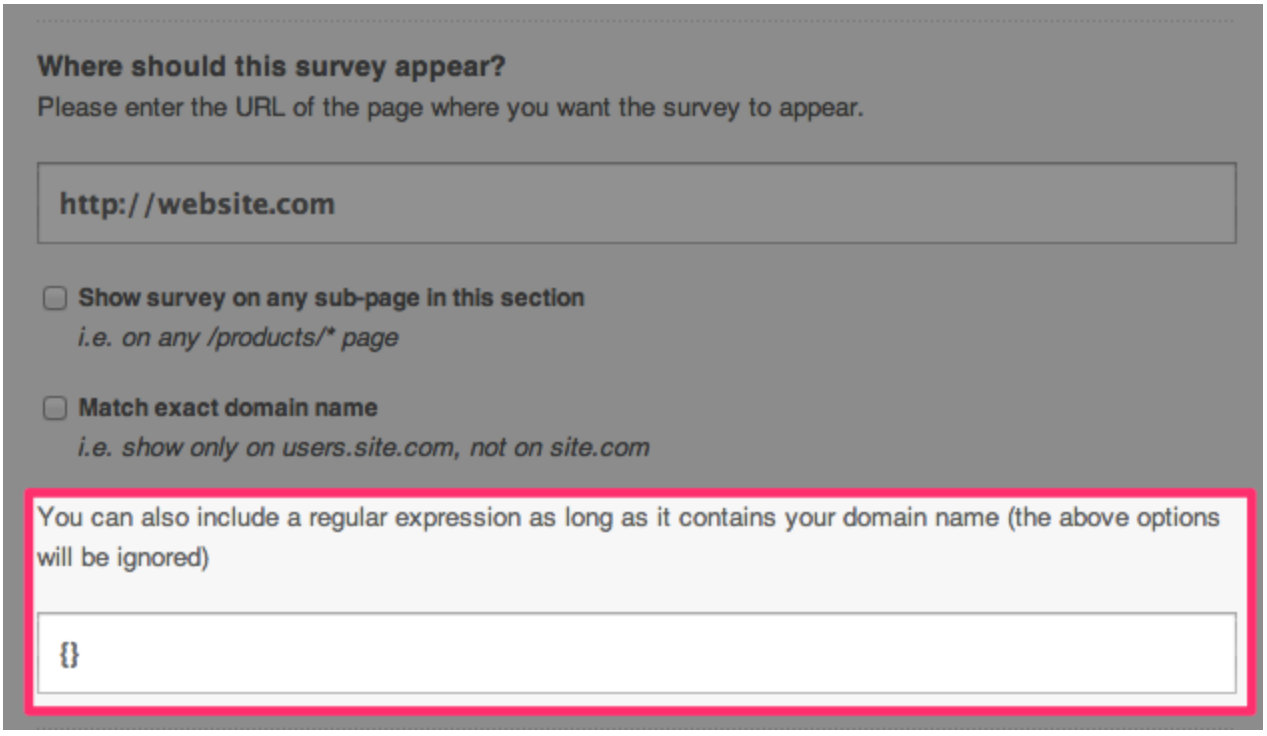
Qualaroo Support

If you get lost at any point or want help troubleshooting your regex, drop the Qualaroo Customer Success Team a line. We're here to help you out! We hope you enjoyed this guide, and feel comfortable writing your own regular expressions now. You can do it!

Formatting URL Regexes The Old Fashioned Way

This page is for formatting regexes using an older interface. It will have a single field for the regex, not the 3-field version with subdomain, tld and path.

Regular Expressions for URL Targeting



HTTPS and WWW

Since some URLs start with http, and others with https, and some of your visitors may have a browser extension that automatically turns all http URLs into https ones, it's occasionally very important to include both options within your regex. Here's how you would do that:

```
https?
```

The `?` after the "`s`" means that there can be 0 or 1 "`s`" - meaning http and https are both included. The question mark only affects the ONE character in behind it, unless you put them inside parentheses, as you'll see in just a moment. Put this together with the escaped-slash we learned earlier, and you get:

```
https?:\/\/
```

The `\/\/` pattern is the two // with escape slashes. A single slash would look like `\/`, and an optional slash would look like `\/?`

We don't need to escape the : because it's not a special character.

To include the "www." you would include the following part:

```
(www\.)?
```

Regular Expressions for URL Targeting

The parentheses around the "`www.`" mean `(everything in here)` - just like in Algebra class. We still have to escape the `.` and we add the `?` at the end to show that it's not required, that there can be 0 or 1 "`www.`"

Putting the Pieces Together

If the site we are using is http://www.website.com, our whole regex targeting it would look like:

```
https?:\/\/(www\.)?website\.com\/?
```

This will match:

```
http://website.com
https://website.com
http://www.website.com
https://www.website.com
http://website.com/
https://website.com/
http://www.website.com/
https://www.website.com/
```

Note that these URLs include all the combinations of **http**, **https**, **www** or **no www**, and the final **/**

The Beginning and The End

In this older interface, we also need to tell the regex where to stop and end. This is done by using two other special characters, `^` and `$`. The `^` marks the beginning of the regex, and the `$` marks the end. Using the example from above, this would be your regex:

```
https?:\/\/(www\.)?website\.com\/?
```

> To use this example, you would add the following to your regex field:
> **`^https?:\/\/(www\.)?website\.com\/?$`**