# Google Analytics Training

Implementing Google Analytics
Using Regular Expressions

## Regular Expressions (RegEx)

Regular Expressions are patterns used to match text:

- They can contain characters and metacharacters
- They are used in GA to match or capture portions of a data field
- Within Google Analytics, Regular Expressions are most useful in defining filters and tracking goals

Filters in Google Analytics use Regular Expressions to match data and perform an action when a match is achieved

•e.g. You would use a Regular Expression to exclude a range of IP addresses

A regular expression is a set of characters and metacharacters that are used to match text in a specified pattern.

You can use regular expressions to configure flexible goals and powerful filters.

For example, if you want to create a filter that filters out a range of IP addresses, you'll need to enter a string that describes the range of the IP addresses that you want excluded from your traffic.

Let's start off by looking at each metacharacter.

Metacharacters are characters that have special meanings in regular expressions.

## Dot .

. Match any single character

`Act ., Scene 3` would match "Act 1, Scene 3" and "Act 2, Scene 3"

The operative word here is *single*.
The regex would *not* match "Act 10, Scene 3"

What will?

```
Act .., Scene 3
```

To make your regex more flexible, you could use a quantifier like the + sign:

`Act .+, Scene 3`

But we'll talk about repetition a bit later.

Use the dot as a wildcard to match any single character.

The operative word here is "single", as the regex would NOT match Act 10, Scene 3. The dot only allows one character, and the number ten contains two characters -- a 1 and a 0.

How would you write a regular expression that would match "Act 10, Scene 3"?

You could use two dots.

To make your regex more flexible, and match EITHER "Act 1, Scene 3" or "Act 10, Scene 3", you could use a quantifier like the + sign.

But we'll talk about repetition a bit later in this module.

## Backslash \

\ Escape the special meaning of metacharacters

`U.S. Holiday` would match "UPS. Holiday" "U.Sb Holiday" and "U3Sg Holiday"

Remember that the dot is a special character that matches with any single character, so if you want to treat a dot like a regular dot, you have to escape it with the backslash.

`U\.S\. Holiday` matches *only* with "U.S. Holiday"

Will `192.168.1.1` match *only* this IP address?

No. The dots are wildcards, so this regex would match many strings like "192.168.151", "1921168.1.1", and others. To match only this one IP address, you need to escape the dots with backslashes:

`192\.168\.1\.1`

Backslashes allow you to use special characters, such as the dot, as though they were literal characters.

Enter the backslash immediately before each metacharacter you would like to escape.

"U.S. Holiday" written this way with periods after the U and the S would match a number of unintended strings, including UPS. Holiday, U.Sb Holiday, and U3Sg Holiday.

Remember that the dot is a special character that matches with any single character, so if you want to treat a dot like a regular dot, you have to escape it with the backslash.

You'll use backslashes a lot, because dots are used so frequently in precisely the strings you are trying to match, like URLs and IP addresses.

For example, if you are creating a filter to exclude an IP address, remember to escape the dots.

## Character Sets and Ranges [ ]

[ ] Match one item in this character set

\- Use a hyphen inside a character set to specify a range

`[uU]\.[sS]\. Holiday` matches "u.s. Holiday" and "U.S. Holiday"

Why *won't* it match "US Holiday" or "u.s. holiday"?
The regex requires periods after the U and the S, and the H to be capitalized.

`[uU]\.?[sS]\.? [hH]oliday` will match all of the above

You can also use a hyphen - to specify a range of characters.

`[0-9]` matches 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

You can negate matches using a caret ^ after the opening square bracket [

`[^0-9]` does *not* match 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Use square brackets to enclose all of the characters you want as match possibilities. So, in the slide, you're trying to match the string U.S. Holiday, regardless of whether the U and the S are capitalized.

However, the expression won't match U.S. Holiday unless periods are used after both the U and the S. The expression also requires that the H is capitalized.

There is a regex you can write to match all of these variations. The question mark used here is another "quantifier", like the '+' sign mentioned earlier.

Again, we'll talk about repetition in the next slide.

You can either individually list all the characters you want to match, as we did in the first example, or you can specify a range.

Use a hyphen inside a character set to specify a range. So instead of typing square bracket 0 1 2 3 4 5 6 7 8 9, you can type square bracket 0 dash 9.

And, you can negate a match using a caret after the opening square bracket.

Typing square bracket caret zero dash nine will exclude all numbers from matching.

Note that later in this module, you will see the caret used a different way—as an anchor.

The use of the caret shown here is specific to character sets, and the negating behaviour occurs only when the caret is used after the opening square bracket in a character set.

## Quantifiers and Repetition ? + *

? Match zero or one of the previous item

+ Match one or more of the previous item

* Match zero or more of the previous item

`31?` matches 3, 31
`31+` matches 31, 311, 3111, 31111
`31*` matches 3, 31, 311, 3111, 3111

Specify repetition using `{minimum, maximum}`
`31{2}` matches 311. Does *not* match 3, 31, or 3111.
`31{1,3}` matches 31, 311, 3111. Does *not* match 3, or 31111.

Recall that a dot `.` matches any single character.
What would you use to match a wildcard of indeterminate length?

`.*` This will match any string of any size

Now let's talk about using quantifiers to indicate repetition.

In earlier examples, we've used the plus sign and the question mark.

The question mark requires either zero or one of the preceding character. In the expression "3-1-?" , the preceding character is a 1. So, both 3 and 3-1 would match.

The plus sign requires at least one of the preceding character. So, "3-1-+" wouldn't match just a 3. It would match 3-1, 3-1-1, and so on.

The asterisk requires zero or more of the preceding character. In the expression, "3-1-*", the preceding character is a 1. So it would match 3, 3-1 -, 3-1-1, and so forth.

You can also SPECIFY repetition using a minimum and maximum number inside curly brackets.

Recall that a dot matches any single character. What would you use to match a wildcard of indeterminate length?

Dot star will match a string of any size. Dot star is an easy way to say "match anything," and is commonly used in Google Analytics goals and filters.

## Grouping ( )

( ) Group and remember contents as an item

| Either/Or

`(U\.S\.|US|u\.s\.|us) Holiday` matches
"U.S. Holiday", "US Holiday", "u.s. Holiday", and "us Holiday"

Will it match "U.S Holiday"?

No, because it isn't one of the options we listed

`(u\.?s\.?|U\.?S\.?)` will match all of the above

It is handy to use the parentheses and the pipe symbol (also known as the OR symbol) together.

Basically, you can just list the strings you want to match, separating each string with a pipe symbol -- and enclosing the whole list in parentheses.

Here, we've listed four variations of "US" that we'll accept as a match for US Holiday.

If it's not in the list, it won't get matched. That's why "US Holiday" won't get matched if one of the periods is missing.

In our list, we've accounted for both periods missing, but not for just one period missing.

Using question marks, the second regex in the slide will match all of the above.

## Anchors ^ $

^ Start of a string

$ End of a string

^US matches "US Holiday", but *not* "Next Monday is a US Holiday"

Holiday$ matches "US Holiday", but *not* "US Holiday Schedule"

Anchors can be useful when specifying an IP address:

192\.168\.1\.1$ matches "192.168.1.1" but not "192.168.1.15"

^72\.168\.1\.1 matches "72.168.1.1" but not "172.168.1.1"

The caret signals the beginning of an expression. In order to match, the string must BEGIN with what the regex specifies..

The dollar sign says, if there are any more characters after the END of this string, then it's not a match.

So, caret US means start with US. US Holiday matches, but "Next Monday is a US Holiday" does not match.

Holiday$ means end with Holiday. US Holiday still matches, but "US Holiday Schedule" does not match.

Anchors can be useful when specifying an IP address. Take a look at these examples.

## Shorthand Character Classes \d \s \w

\d Match any number (same as [0-9])

\s Match any whitespace

\w Match any letter, number, or underscore (same as [A-Za-z0-9_])

\d{1,5}\s\w* matches "345 Embarcadero"

However, just "Embarcadero" does not. Why not?

Numbers are required as part of the regular expression.

(\d{1,5}\s)?\w* will match all of the above

Note that "1600 Amphitheatre Parkway" would not match either.
You would have to use something like:
\d{1,5}\s\w*\s\w*

Some character classes are used so commonly that there is a shorthand you can use instead of writing out the ranges within square brackets.

Let's look at the example of a simplified regex that could match an addres:

Backslash d means match any one digit zero through nine.
Use curly brackets and a minimum and maximum number to specify how many digits to match.

Backslash d followed by 1 comma 5 in curly brackets means that the address must contain at least one digit, and at most five digits.

Backslash s means that the number should be followed by one space, backslash w means match any alphanumeric character and the star means include as many alphanumeric characters as you want.

"345 Embarcadero" matches, but just "Embarcadero" does not, because this regex requires the string to start with a number.

If you want to make the number optional, group the first part of the regex with parentheses--including the space--and follow it with the question mark.

Note that an address like "1600 Amphitheatre Parkway" would not match either, because the regex does not account for the space between Amphitheatre and Parkway.

The slide shows one way you could account for this.

# RegEx Review

## RegEx Review

[ ] Matches a Range of Characters
( ) Groups statements
| Either/Or

**Example:**

```
([Gg]oogle|[Yy]ahoo)
```

**Matches:**
Google
google
Yahoo
yahoo

Let's review.

In the example on the slide, we've created an expression that will match the strings Google or Yahoo, regardless of whether or not Google and Yahoo are capitalized.

## RegEx Review

( ) Groups statements
. Match Any Single Character
* Matches Zero Or More Characters
\ Escape
| Either/Or
$ End of a string

**Sample URL:**

```
.*index\.php\?dl=video/trailers/(internet|theatrical)$
```

**Matches:**
(anything)index.php?dl=video/trailers/internet
(anything)index.php?dl=video/trailers/theatrical

Here, we've created an expression that will match URLs for internet and theatrical movie trailers.

The first part of the expression indicates that the URL can begin with anything.

Then the expression specifies that the URL must end with index.php?dl=video/trailers/ and then either internet or theatrical.

The $ sign ensures that any URLs that are any longer than this won't get included in the match.

## Common Uses for Regular Expressions

There are several common uses for Regular Expressions within Google Analytics:

- Creating filters
  - Filter Internal Traffic: `205\.172\.23[2345]`
- Setting up goals
  - Making a Goal out of Several Pages
    `.*index\.php\?dl=video/trailers/(internet|theatrical)$`
- Tracking equivalent pages
  - `/downloads/casestudy/.*`
- Filtering data within the reporting interface
  - `([Gg]oogle|[Yy]ahoo)`

You'll find lots of applications for regular expressions in Google Analytics.

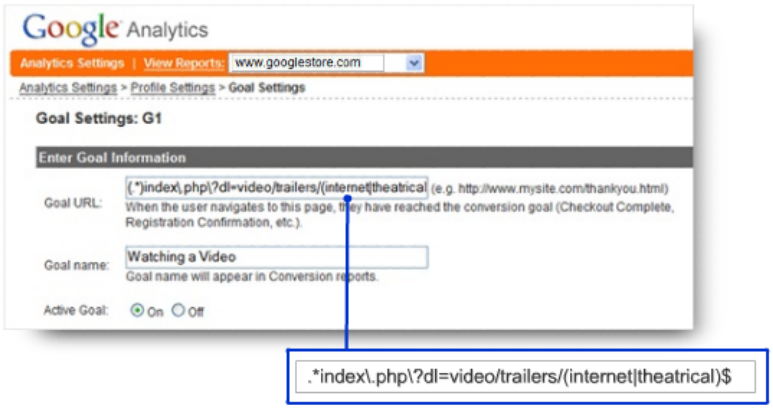
Some common examples are:
• filtering out internal traffic by specifying a set of IP addresses
• setting up a goal that needs to match multiple URLs
• tracking equivalent pages in a funnel
• and using the filter box that appears on your reports to find specific entries in a table.

## RegEx Filters


Use Regular Expressions to create filters that include or exclude certain types of traffic

Here's an example of a custom filter that uses a very simple regular expression.

# RegEx Goals

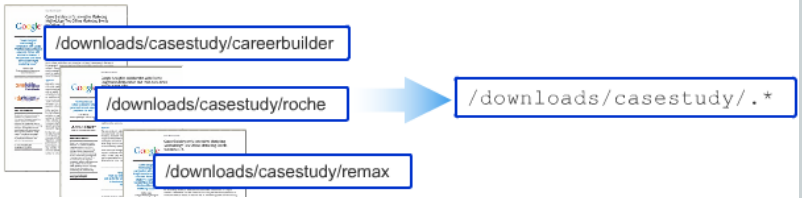



Use RegEx to set-up goals where several different types of pages constitute a goal or step in the funnel process

Here's a regular expression used to define a goal URL.

# RegEx and Tracking Equivalent Pages

- If your site has multiple pages or steps that are equivalent in value (e.g. whitepaper or case study downloads), you can use Regular Expressions to group them
- Regular Expressions allow you to create one funnel as opposed to tracking each page or action individually
- Example:

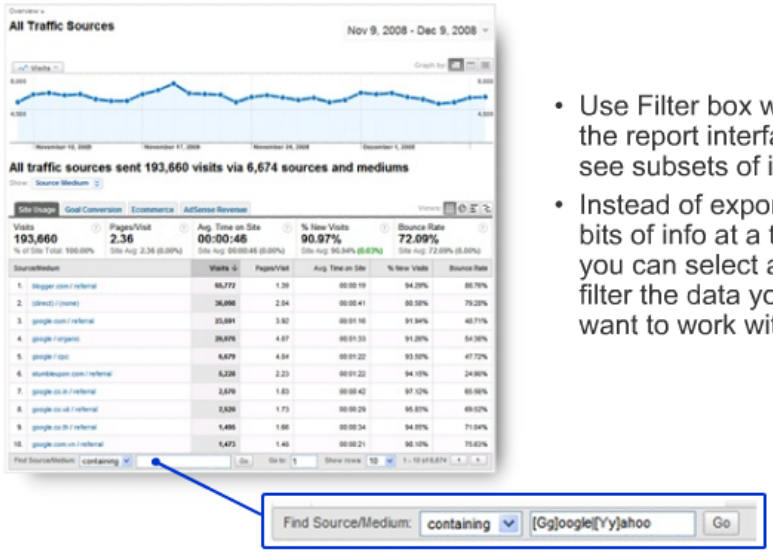



Here's how you might use regular expressions to group pages or funnel steps on your site.

Using a regular expression allows you to track them as one funnel step rather than tracking each page or action individually.

Learn how goals and funnels work in the module on goals.

# RegEx Within the Report Interface



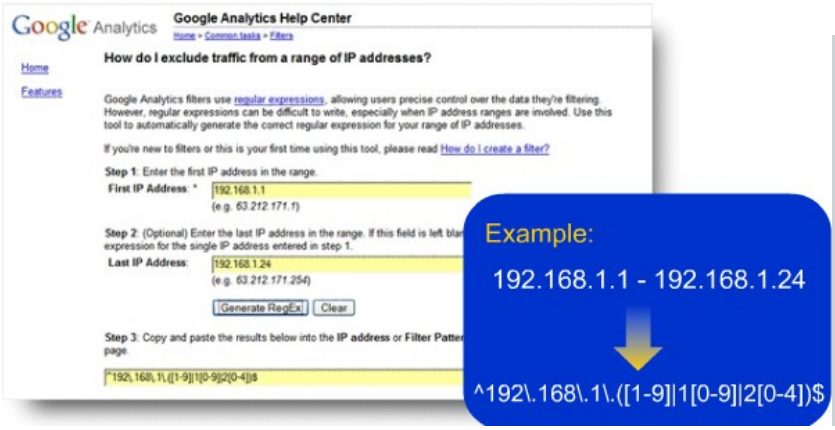

- Use Filter box within the report interface to see subsets of info
- Instead of exporting bits of info at a time, you can select and filter the data you want to work with

And, here's an example of using regular expressions within your reports.

We're using the Find box to display all the rows in the table that contain Google or Yahoo.

## RegEx Generator for IP Address Ranges

Use this tool to create regular expressions involving a range of IP Addresses:
http://www.google.com/support/googleanalytics/bin/answer.py?hl=en&answer=55572




Google Analytics provides a tool that makes it easier to generate a regular expression that matches a range of IP addresses.

It's called the Regular Expression Generator and you can find it at the URL shown in the slide.

Or, you can search for Regular Expression Generator in the Google Analytics Help Center.

## Points to Remember

- Regular Expressions are a powerful search/find tool
- Can be used with filters, goals, and within the reporting interface
- Carefully test your RegEx statements
  - Small mistakes can have a big impact
  - Even correctly made RegEx statements may have flaws
  - Test, test, test then consider having someone else test for you
- Search for "regex" to find additional resources on the web

You'll find a number of useful applications for regex as you use Google Analytics.

But, it's important that you think through all the implications of each expression that you use when you set up a filter or a goal.

It's easy to make a mistake and not get the data or the result you're looking for.

Set up a duplicate profile to test your regex statements. After enough data has been collected, check your results and make sure they're what you expect.

Remember to always maintain a backup profile that includes all your data.

There are lots of regex resources on the web. To get started, just search for regex.