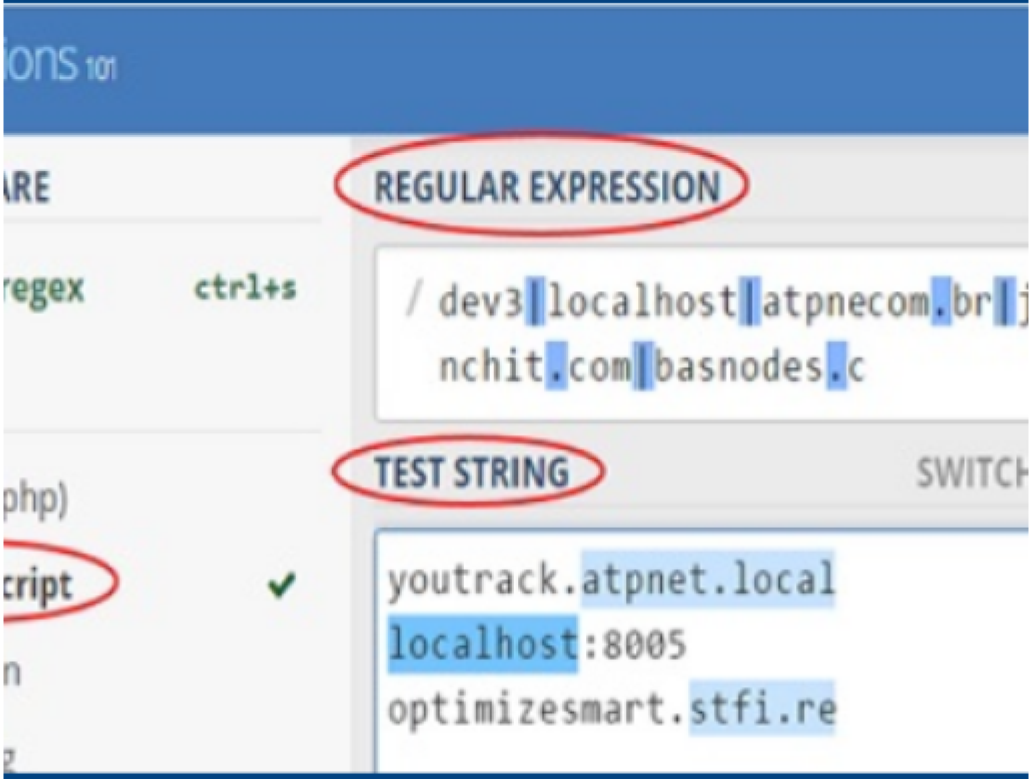
# REGULAR EXPRESSIONS CHEAT SHEET

## FOR GOOGLE ANALYTICS AND GOOGLE TAG MANAGER



```
/ dev3|localhost|atpnecom.br|j
  nchit.com|basnodes.c
```

```
youtrack.atpnet.local
localhost:8005
optimizesmart.stfi.re
```

BY HIMANSHU SHARMA

FOUNDER OF OPTIMIZESMART.COM

OptimizeSmart.com

# Regular Expressions Cheat Sheet for Google Analytics and Google Tag Manager

**Written by Himanshu Sharma, Founder of Optimize Smart**

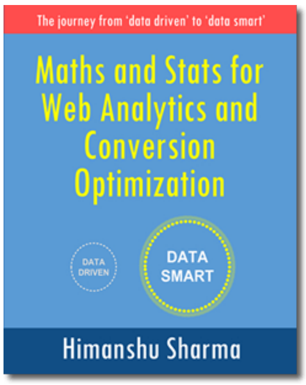
## About the author

- Founder, OptimizeSmart.com
- Over 15 years of experience in digital analytics and marketing
- Author of four best-selling books on digital analytics and conversion optimization
- Nominated for Digital Analytics Association Awards for Excellence
- Runs one of the most popular blogs in the world on digital analytics
- Consultant to countless small and big businesses over the decade

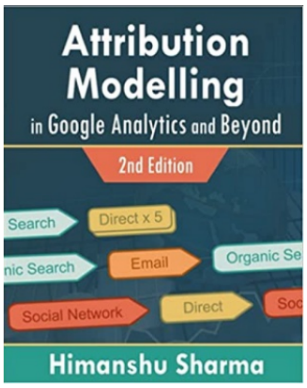
Website: [www.optimizesmart.com](www.optimizesmart.com)

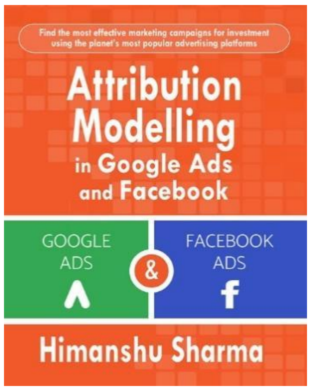
Linkedin: [https://www.linkedin.com/in/analyticsnerd](https://www.linkedin.com/in/analyticsnerd)

Facebook: [https://www.facebook.com/optimizesmart](https://www.facebook.com/optimizesmart)


The journey from 'data driven' to 'data smart'
Maths and Stats for Web Analytics and Conversion Optimization
DATA DRIVEN
DATA SMART
Himanshu Sharma

Master the Essentials of
Email Marketing Analytics
The journey from inbox placement to conversion
Himanshu Sharma




Find the most effective marketing campaigns for investment using the planet's most popular advertising platforms
Attribution Modelling
in Google Ads and Facebook
GOOGLE ADS
&
FACEBOOK ADS
Himanshu Sharma

## Following are our most downloaded ebooks for career advancement:

**#1** [Sales and ROI Accelerator (150+ pages)]

**WHAT'S INSIDE**: My step-by-step blueprint for generating record sales and ROI by leveraging analytics data.

**#2** [Set Up Your Google Analytics 4 (GA4) Account Correctly And Fast (70 pages)]

**WHAT'S INSIDE:** Learn to set up your GA4 account correctly and fast using this 62 points checklist.

**FAQ: Do you show “How” to do each item on the checklist? If so, with screenshots?**
Yes. There are links to the articles with detailed step by step instructions.

**FAQ: Does this ebook cover GTM too?**
Yes.

**#3** [Google Tag Manager Data Layers (100+ pages)]

**WHAT'S INSIDE**: My step-by-step blueprint for getting started with data layers. Get the only ebook on GTM data layers ever published. Learn the JavaScript behind it.

**#4 Learn to Read E-Commerce Reports in Google Analytics (100+ pages)**

**WHAT'S INSIDE:** My step-by-step guide to reading both standard and enhanced e-commerce reports in Google Analytics. E-commerce reports are the most valuable reports in Google Analytics.

**#5 Do you want better skills in digital analytics and marketing? If yes, then register for the free training:**

Here's what we're going to cover…

1. Why digital analytics is the key to online business success.
2. The number 1 reason why most marketers are not able to scale their advertising and maximize sales.
3. Why Google and Facebook ads don't work for most businesses & how to make them work.
4. Why you won't get any competitive advantage in the marketplace just by knowing Google Analytics.
5. The number 1 reason why conversion optimization is not working for your business.
6. How to advertise on any marketing platform for FREE with an unlimited budget.
7. How to learn and master digital analytics and conversion optimization in record time.

## Get helpful tips on a daily basis

If you are the type of person who finds it helpful to receive short tips on building your website traffic, improving conversions, fixing attribution issues and learning about analytics in general, then follow me on LinkedIn. I post a few short tips each day.

**Click here and follow me on LinkedIn**

# Introduction to Regular Expressions

Regular expressions, also called regex, are a set of characters that help you to find specific data in a data set. They are a very powerful tool to find and match data while applying a specific set of filters.

Regex is an expression that is used to check for a pattern in a string.

For example **^Colou?r$** is a regular expression that matches both the string: 'color' and 'colour'.

A regex is made up of characters and metacharacters.

**Metacharacters** are the characters that have special meaning in a regex. They are the building blocks of a regex.

For e.g. [], ^, (), {}, $, +, * etc.

## Advantages of using REGEX in Google Analytics

There are many cases where regular expressions are very useful in Google Analytics. Some such cases are:

- Setting up a goal that should match multiple-goal pages instead of one
- Setting up a funnel in which a funnel step should match multiple pages instead of one.
- Excluding traffic from an IP address range via filters
- Setting up complex custom segments like the segments which can filter out branded keywords.
- Understanding the commercial value of long-tail keywords.
- Rewriting URLs in Google Analytics reports.
- Filtering data based on complex patterns within the Google Analytics reporting interface.
- Finding referrer spam in Google Analytics.
- Blocking spam referrers through the custom advanced filter in Google Analytics.
- Creating content groups in Google Analytics
- Creating Channel grouping in Google Analytics

- Building audiences
- Tracking Site Search without Query Parameter.
- Debugging Google Analytics tracking issues.

## Flavours of Regular Expressions (aka Regex Engines)

Implementations of regex functionality using a particular type of syntax are called a **regex engine**.

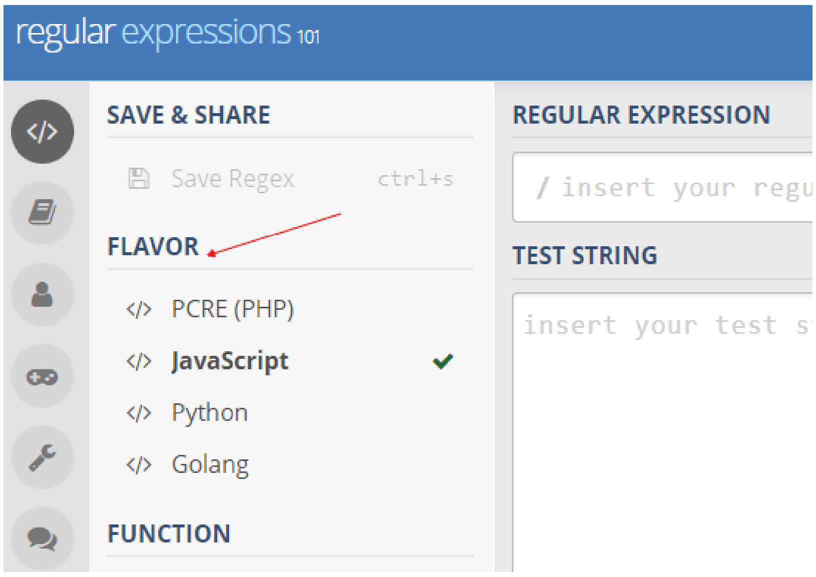
There are many types of regex engines available. The most popular among them are:

1. **PCRE (PHP)**
2. **JavaScript**
3. **Python**
4. **Golang**

Different regex engines support different types of syntax and the meaning of metacharacters may change depending upon the regex engine being used.

Thus a regular expression that is considered valid under one regex engine may not be considered valid under another regex engine.

Whenever you test a regex using a **regex tester tool**, you get the option to select the flavour under which you want to test your regular expression:



Since the **regex engine used by Google Analytics and Google Tag Manager is JavaScript**, you should always select 'JavaScript' as the flavour before testing your regular expressions for GA/GTM.

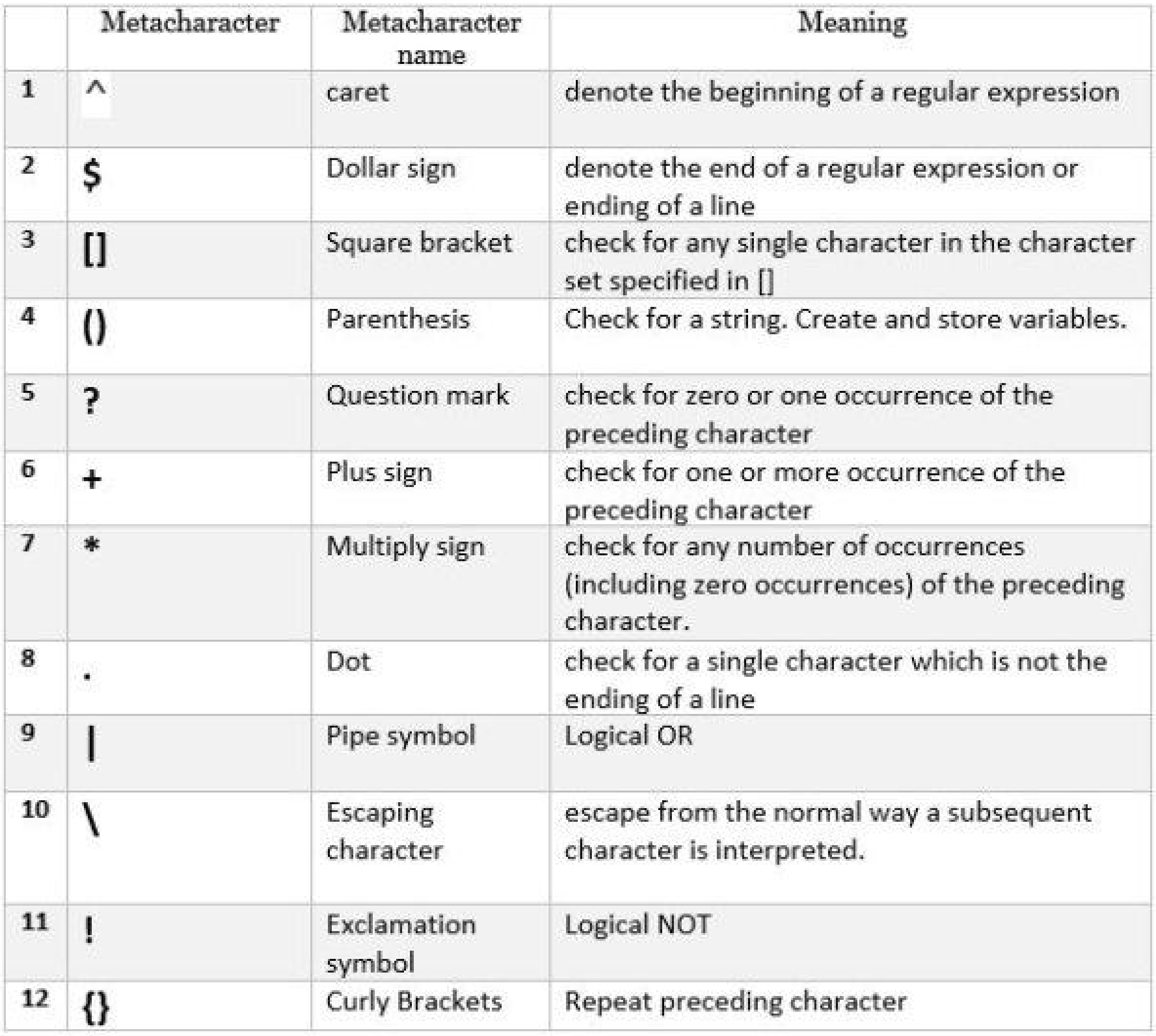
## Advantages of using REGEX in Google Tag Manager

Through regular expressions you can:

- Set up complex triggers in GTM.

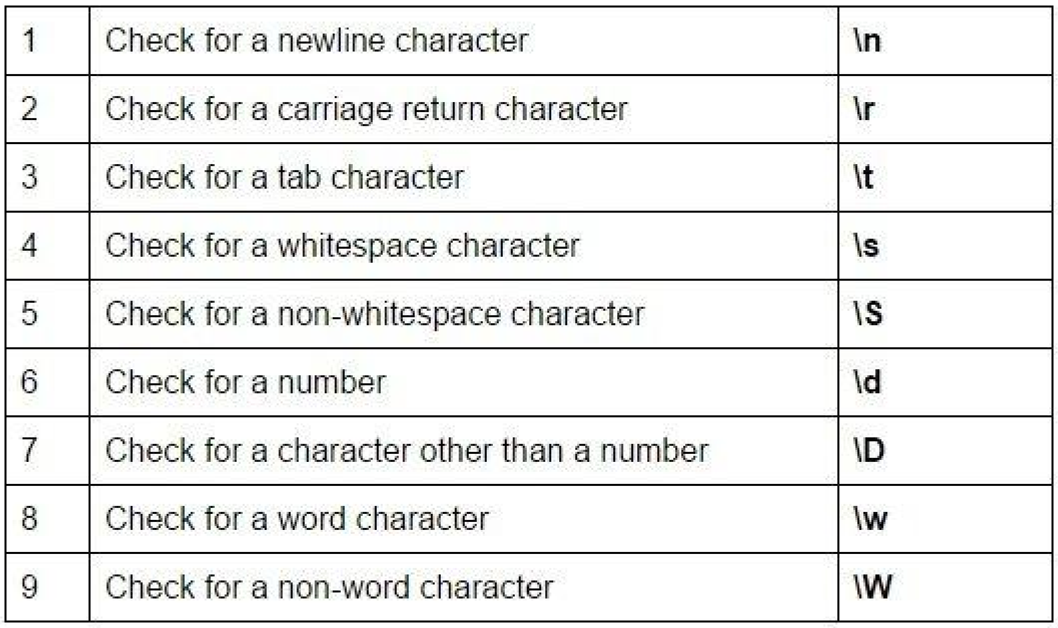
- Use the regex table variable in Google Tag Manager.
- You can use regex in custom JavaScript variables like when tracking site search without query parameter in Google Tag Manager.

## The Building Blocks of a Regular Expression



| | Metacharacter | Metacharacter name | Meaning |
|---|---|---|---|
| 1 | ^ | caret | denote the beginning of a regular expression |
| 2 | $ | Dollar sign | denote the end of a regular expression or ending of a line |
| 3 | [] | Square bracket | check for any single character in the character set specified in [] |
| 4 | () | Parenthesis | Check for a string. Create and store variables. |
| 5 | ? | Question mark | check for zero or one occurrence of the preceding character |
| 6 | + | Plus sign | check for one or more occurrence of the preceding character |
| 7 | * | Multiply sign | check for any number of occurrences (including zero occurrences) of the preceding character. |
| 8 | . | Dot | check for a single character which is not the ending of a line |
| 9 | \| | Pipe symbol | Logical OR |
| 10 | \ | Escaping character | escape from the normal way a subsequent character is interpreted. |
| 11 | ! | Exclamation symbol | Logical NOT |
| 12 | {} | Curly Brackets | Repeat preceding character |

## Other Meta Characters in Regex



| | | |
|---|---|---|
| 1 | Check for a newline character | \n |
| 2 | Check for a carriage return character | \r |
| 3 | Check for a tab character | \t |
| 4 | Check for a whitespace character | \s |
| 5 | Check for a non-whitespace character | \S |
| 6 | Check for a number | \d |
| 7 | Check for a character other than a number | \D |
| 8 | Check for a word character | \w |
| 9 | Check for a non-word character | \W |

## Escaping Character – Back Slash \

'\' is the escaping character that is used to escape from the normal way a subsequent character is interpreted.

Through escaping character, you can convert a regular character into a metacharacter or turn a metacharacter into a regular character.

For example, **Forward Slash (/)** has a special meaning in a regex. It is used to mark the beginning and end of a regular expression.

**For example:**

var a = /colou?r/;

If you want regex to treat forward slash as a forward slash and not some special character then you need to use it along with the escaping character like this: **\/**

So if you want to check for a pattern say **/shop/** in the string **/shop/collection/men/**

then your regex should be: **\/shop**

If you use the regex **/shop** then it won't match the string **/shop/collection/men/** because **/** would be treated as a special character instead of a regular forward slash.

**Another example:**

'**n'** is a regular character. But if you add escaping character before it then it would become a metacharacter: **\n** which is a new line character.

So if you want to check for a pattern say **n** in the string **abcd\n3456**

then your regex should be: **abcd\\n3456**

If you use the regex **abcd\n** then it won't match the string **abcd\n3456** because **\n** would be treated as a newline character instead of a regular character.

**Another example:**

'**?**' is a metacharacter. To make it a regular character, you need to add escaping character before: **\?**

So if you want to check for a question mark in the string **colou?r**

then your regex should be: **colou\?r**

If you use the regex **colou?r** then it would match the string **color** or **colour** and not **colou?r** as then **?** will be treated as a metacharacter.

## Caret ^

**'^'** – This is known as 'Caret' and is used to denote the beginning of a regular expression.

**^\/Colou?r =>** Check for a pattern that starts with '/Color' or '/Colour'. Example:

**/Colour**/?proid=3456/review

**/Color**-red/?proid=3456/review

**^\/Nov(ember)? =>** Check for a pattern that starts with '/Nov' or '/November'.

Example:

**/November**-sales/?proid=3456/review

**/Nov**-sales/?proid=3456/review

**^\/elearning\.html =>** Check for a pattern which starts with '/elearning.html'.
Example:

**/elearning.html**/?proid=3456/review

**^\/.*\.php =>** Check for a pattern which starts with any file with .php extension. Example:

**/elearning.php**/color/?proid=3456/review

**/games.php**/?proid=3456/

**/a1.php**/color/?proid=3456&gclid=153dwf3533

**^\/product-price\.php =>** Check for a pattern which starts with '/product-price.php'. Example:

**/product-price.php**?proid=123&cid=2142

**/product-price.php**?cid=2142&gclid=442352df

**Caret also means NOT when used after the opening square bracket.**

**[^a]** => Check for any single character other than the lowercase letter 'a'.

For example: the regex **product-[^a]** will match:

/shop/men/sales/**product-b**

/shop/men/sales/**product-c**

**[^B] = >** Check for any single character other than the uppercase letter ‘B’.

For example: the regex **product-[^B]** will match:

/shop/men/sales/**product-b**

/shop/men/sales/**product-c**

**[^1]** => Check for any single character other than the number ‘1’.

For example: the regex **proid=[^1]** will match:

/men/product-b?**proid=3**456&gclid=153dwf3533

but will not match:

/men/product-b?proid=1456&gclid=153dwf3533

**[^ab]** => Check for any single character other than the lower case letters ‘a’ and ‘b’.

For example: the regex **location=[^ab]** will match:

/shop/collection/prodID=141?**location=c**anada

but will not match:

/shop/collection/prodID=141?location=america

/shop/collection/prodID=141?location=bermuda

**[^aB] =>** Check for any single character other than the lower case letter 'a' and uppercase letter 'B'.

**[^1B]** => Check for any single character other than the number '1' and uppercase letter 'B'

**[^Dog]** => Check for any single character other than the following: uppercase letter 'D', lowercase letter 'o' and the lowercase letter 'g'.

For example: the regex **location=[^Dog]** will match:

/shop/collection/prodID=141?**location=c**anada

/shop/collection/prodID=141?**location=d**enmark

but will not match:

/shop/collection/prodID=141?location=Denver

/shop/collection/prodID=141?location=ontario

/shop/collection/prodID=141?location=greenland

**[^123b]** => Check for any single character other than the following characters: number '1', number '2', number '3' and lowercase letter 'b'.

**[^1-3]** => Check for any single character other than the following: number ‘1’, number ‘2’ and number ‘3’.

For example: the regex **prodID=[^1-3]** will match:

/shop/collection/**prodID=4**5321&cid=1313

/shop/collection/**prodID=5**321&cid=13442

but will not match:

/shop/collection/prodID=12321&cid=1313

/shop/collection/prodID=2321&cid=1313

/shop/collection/prodID=321&cid=1313

**[^0-9]** => Check for any single character other than the number.

For example: the regex **de\/[^0-9]** will match all those pages in the de/ folder whose name doesn’t start with a number:

/**de/s**chool-london

/**de/g**eneral/

but will not match:

/de/12fggtyooo

**[^a-z]** => Check for any single character which is not a lower case letter.

For example: the regex **de\/[^a-z]** will match all those pages in the de/ folder whose name doesn't start with a lower case letter:

**/de/1**london-school

**/de/?**productid=423543

but will not match:

/de/school/london

**[^A-Z]** => Check for any single character which is not an upper case letter.

## Dollar $

**'$'** – It is used to denote the end of a regular expression or end of a line. For e.g.

**Colou?r$ =>** Check for a pattern which ends with 'Color' or 'Colour'

**Nov(ember)?$ =>** Check for a pattern that ends with 'Nov' or 'November'

**elearning\.html$ =>** Check for a pattern which ends with 'elearning.html'

**\.php$ =>** Check for a pattern which ends with .php

**product-price\.php$ =>** Check for a pattern which ends with 'product-price.php'

## Square Bracket []

'[]' – This square bracket is used to check for any single character in the character set specified in []. For e.g:

**[a]** => Check for a single character which is a lowercase letter 'a'.

**[ab]** => Check for a single character which is either a lower case letter 'a' or 'b'.

**[aB]** => Check for a single character which is either a lower case letter 'a' or uppercase letter 'B'

**[1B]** => Check for a single character which is either a number '1' or an uppercase letter 'B'.

**[Dog]** => Check for a single character which can be any one of the following: uppercase letter 'D', lower case letter 'o' or the lowercase letter 'g'.

**[123b]** => Check for a single character which can be anyone of the following: number '1', number '2', number '3' or lowercase letter 'b'.

**[1-3]** => Check for a single character which can be any one number from 1, 2 and 3.

**[0-9]** => Check for a single character which is a number.

**[a-d]** => Check for a single character which can be any one of the following lower case letter: 'a', 'b', 'c' or 'd'.

**[a-z]** => Check for a single character which is a lower case letter.

**[A-Z]** => Check for a single character which is an upper case letter.

**[A-T]** => Check for a single character which can be any uppercase letter from 'A' to 'T'.

**[home.php]** => Check for a single character which can be any one of the following characters: lowercase letter 'h', lowercase letter 'o', lowercase letter 'm', lowercase letter 'e', special character '.', lower case letter 'p', lowercase letter 'h' or lowercase letter 'p'

**Note**: if you want to check for a letter regardless of its case (upper case or lower case) then use **[a-zA-Z]**

## Parenthesis ()

'**()'** – This is known as parenthesis and is used to check for a string. For e.g.

**(a)** => Check for string 'a'

**(ab)** => Check for string 'ab'

**(dog)** => Check for string 'dog'

**(dog123)** => Check for string 'dog123'

**(0-9) =>** Check for string '0-9'

**(A-Z)** => Check for string 'A-Z'

**(a-z)** => Check for string 'a-z'

**(123dog588)** => Check for string '123dog588'

**Note: ()** is also used to create and store variables. For e.g. ^ (.*) $

## Question Mark?

**'?'** is used to check for zero or one occurrence of the preceding character. For e.g.

**[a]? =>** Check for zero or one occurrence of the lowercase letter 'a'.

**[dog]? =>** Check for zero or one occurrence of the lowercase letter 'd', 'o' or 'g'.

**[^dog]? =>** Check for zero or one occurrence of a character that is not the lowercase letter 'd', 'o' or 'g'.

**[0-9]? =>** Check for zero or one occurrence of a number

**[^a-z]? =>** Check for zero or one occurrence of a character that is not a lower case letter.

**^colou?r$ =>** check for color or colour.

**^[nN]ov(ember)? 28(th)?$** => check for 'nov 28', 'november 28, Nov 28th and November 28th

**Note:** ? when used inside a regular expression makes the preceding letter or group of letters optional.

For e.g. the regular expression: **^colou?r$** matches both 'color' and 'colour'.

Similarly, the regular expression: **^[nN]ov(ember)? 28(th)?$** matches: 'nov 28', 'november 28, Nov 28th and November 28th

## Plus +

**'+'** is used to check for one or more occurrences of the preceding character. For e.g.

**[a]+ =>** Check for one or more occurrences of lowercase letter 'a'.

**[dog]+ =>** Check for one or more occurrences of letters 'd', 'o' or 'g'.

**[548]+** => Check for one or more occurrences of numbers '5', '4' or '8'.

**[o-9]+ =>** Check for one or more numbers

**[a-z]+ =>** Check for one or more lower case letters

**[^a-z]+ =>** Check for one or more characters which are not lowercase letters.

**[a-zA-z]+ =>** Check for any combination of uppercase and lowercase letters.

**[a-z0-9]+ =>** Check for any combination of lowercase letters and numbers.

**[A-Z0-9]+ =>** Check for any combination of uppercase letters and numbers.

**[^9]+ =>** Check for one or more characters which is not the number 9.

## Multiply *

'*****' is used to check for any number of occurrences (including zero occurrences) of the preceding character.

For example, **31*** would match 3, 31, 311, 3111, 31111 etc.

## Dot .

'.' is used to check for a single character (any character that can be typed via a keyboard other than a line break character (\n)).

For example the regular expression: **Action ., Scene2** would match:

- Action 1, Scene2
- Action A, Scene2
- Action 9, Scene2
- Action &, Scene2

but not

- Action 10,Scene2
- Action AB,Scene2

## Pipe |

‘|’ is the logical OR. For example:

**(His|Her)** => Check for the string ‘his’ or ‘her’.

His|Her => Check for the string ‘his’ or ‘her’.

For example, the regex **his|her** will match:

1. t**his** is his book
2. t**his** is her book
3. **his** or her
4. **her** or his

## Exclamation !

‘!’ – It is a logical NOT. But unlike ^ (caret), it is used only at the beginning of a rule or a condition. For e.g.

1. **(!abc) =>** Check for a string which is not the string ‘abc’.
2. **[!0-9] =>** Check for a single character which is not a number.
3. **[!a-z] =>** Check for a single character which is not a lower case letter.

## Curly Brackets {}

{} is used to check for 1 or more occurrences of the preceding character.

It is just like the metacharacter '+' but it provides more control over the number of occurrences of the preceding character you want to match.

For example:

**1{1}** => check for 1 occurrence of the character '1'. This regex will match 1

**1{2}** => check for 2 occurrences of the character '1'. This regex will match 11

**1{3}** =>check for 3 occurrences of the character '1'. This regex will match 111

**1{4}** => check for 4 occurrences of the character '1'. This regex will match 1111

**1{1,4}** =>check for 1 to 4 occurrences of the character '1'. This regex will match 1,11, 111, 1111

**[0-9]{2}** => check for 2 occurrences of a number or in other words, check for two digits number like 12

**[0-9]{3}** => check for 3 occurrences of a number or in other words check for three digits number like 123

**[0-9]{4}** => check for 4 digits number like 1234

**[0-9]{1,4}** => check for 1 to 4 digits number.

**[a]{1}** => check for 1 occurrence of the character 'a'. This regex will match a

**[a]{2}** => check for 2 occurrences of the character 'a'. This regex will match aa

**[a]{3}** =>check for 3 occurrences of the character 'a'. This regex will match aaa

**[a]{4}** => check for 4 occurrences of the character 'a'. This regex will match aaaa

**[a]{1,4}** =>check for 1 to 4 occurrences of the character ‘a’. This regex will match a,aa,aaa,aaaa

**[a-z]{2}** => check for 2 occurrences of a lower case letter. This regex will match aa, bb, cc etc

**[A-Z]{3}** => check for 3 occurrences of a upper case letter. This regex will match AAA, BBB, CCC etc

**[a-zA-Z]{2}** => check for 2 occurrences of a letter (doesn’t matter whether it is upper case or lower case). This regex will match aa, aA, Aa, AA etc

**[a-zA-Z]{1,4}** => check for 1 to 4 occurrences of a letter (doesn’t matter whether it is upper case or lower case). This regex will match aaaa, AAAA, aAAA, AAAa etc

**(rock){1}** => check for 1 occurrence of the string ‘rock’. This regex will match: rock

**(rock){2}** => check for 2 occurrence of the string ‘rock’. This regex will match: rockrock

**(rock){3}** => check for 3 occurrence of the string ‘rock’. This regex will match: rockrockrock

**(rock){1,4}** => check for 1 to 4 occurrence of the string ‘rock’. This regex will match: rock, rockrock, rockrockrock, rockrockrockrock

## White Spaces

To create white space in a regular expression, just use the white space. For e.g.

(Himanshu Sharma) => Check for the string 'Himanshu Sharma'

## Inverting Regex in JavaScript

Inverting a regex means inverting its meaning.

You can invert a regex in JavaScript by using positive and negative lookaheads.

> ***Use positive lookahead if you want to match something that is followed by something else.***
>
> ***Use negative lookahead if you want to match something not followed by something else.***

Positive Lookahead starts with **(?=** and ends with **)**

Negative Lookahead starts with **(?!** and ends with **)**

For example, the regex **de\/[^a-z]** will match all those pages in the de/ folder whose name doesn't start with a lower case letter:

**/de/1**london-school

**/de/?**productid=423543

but will not match:

/de/school/london

The invert of this regular expression would be: match all those pages in the de/ folder whose name starts with a lower case letter:

For example: the regex **de\/(?![^a-z])** will match:

**/de/**school/london

but will not match:

/de/1london-school

/de/?productid=423543

**Note**: JavaScript only supports lookaheads and not lookbehind. Google Analytics doesn't support either lookahead or lookbehind.

## More Regex Examples

**^(*\.html)$ =>** Check for any number of characters before .html and store them in a variable.

**^dog$ =>** Check for the string 'dog'

**^a+$ =>** Check for one or more occurrences of a lower case letter 'a'

**^(abc)+$ =>** Check for one or more occurrences of the string 'abc'.

**^[a-z]+$ =>** Check for one or more occurrences of a lower case letter.

**^(abc)*$ =>** Check for any number of occurrences of the string 'abc'.

**^a*$ =>** Check for any number of occurrences of the lower case letter 'a'

#. Find all the files which start from 'elearning' and which have the '.html' file extension

**^elearning* \.html$**

#. Find all the PHP files

**^*\.php$**

## You are doing Google Analytics all wrong. Here is why...

I have dealt with hundreds of Google Analytics accounts in my career.

I have seen a lot of issues from incorrect tracking code, selecting the wrong KPIs to analyzing data without using custom reports or advanced segments.

But do you know the biggest issue of all in Google analytics?....

**It is the "misinterpretation of analytics data".**

Many marketers make the mistake of crediting conversions to the wrong marketing channel.

And they seem to be making this mistake over and over again.

They give the credit for conversions to the last touchpoint (campaign, ad, search term...).

They can’t help themselves because they believe that the Google Analytics reports are ‘what you see is what you get’.

But they are actually ‘what you interpret is what you get’.

This has resulted in marketers making wrong business decisions and losing money.

**All the data you see in Google Analytics reports today lies to you unless you know exactly how to interpret it correctly.**

For example, let's talk about direct traffic.

All untagged or incorrectly tagged marketing campaigns from display ads to emails could be reported as direct traffic by Google.

**Whenever a referrer is not passed, the traffic is reported as direct traffic by Google.**

Mobile applications don’t send a referrer. Word/PDF documents don’t send a referrer.

‘302 redirects’ sometimes cause the referrer to be dropped. Sometimes browsers don’t pass the referrer.

During an HTTP to HTTPS redirect (or vice versa) the referrer is not passed because of security reasons.

All such traffic is reported as direct traffic by Google.

So on the surface, it may look like that most people are visiting your website directly but this is not usually the case.

But this analysis does not end here, because you are still not looking at the complete picture.

**People do not always access your website directly and then make a purchase straight away.**

They are generally exposed to multiple marketing channels ( display ads, social media, paid search, organic search, referral websites, email etc) before they access your website directly.

Before they make a purchase.

So if you are unaware of the role played by prior marketing channels, you will credit conversions to the wrong marketing channels.

Like in the present case to direct traffic.

To get this type of understanding you need to understand and implement web analytics.

But **you learn data analysis and data interpretation from web analytics and not from Google Analytics.**

The direction in which your analysis will move will determine the direction in which your marketing campaigns will move.

You get that direction from 'web analytics' and not from 'Google Analytics'.

**Web/Digital analytics is not about Google Analytics (GA) or Google Tag Manager (GTM). It is about analyzing and interpreting data, setting up goals, strategies and KPIs.**

It's about creating a strategic roadmap for your business.

That's why the knowledge of web/digital analytics is so important.

So, what I have done is put together some completely free training for you.

This training will teach you what digital analytics really is and how I have been able to leverage it to generate floods of new sales and customers.

I will also show you how you can copy what I have done to get similar results.

You can sign up for the free training here:

## Reserve My Seat Now

I hope you find it helpful.

All the best,

Himanshu