DotLiquid Scripting Filters

05/20/2025 4:07 pm CDT

▼ Details
▼

For a full list of properties accessible via DotLiquid, please visit:
https://serviceminder.io/support/dotliquid

abs

Returns the absolute value of a number.

Input

```
{{ -17 | abs }}
```

Output

```
17
```

Input

```
{{ 4 | abs }}
```

Output

```
4
```

`abs` will also work on a string that only contains a number:

Input

```
{{ "-19.86" | abs }}
```

Output

```
19.86
```

▼

add

Adds two numbers together. Non-numbers will throw an error.

Input

```
{{ "3" | add: "8" }}
```

Output

```
11
```

▼

add_days

Adds a positive or negative whole number of days to a date string.

Input

```
{{ "2020-08-19" | add_days: -3 | date: "YYYY-mm-dd" }}
{{ "2020-08-19" | add_days: 3 | date: "YYYY-mm-dd" }}
```

Output

```
2020-08-16
2020-08-22
```

▼

append

Concatenates two strings and returns the concatenated value.

Input

```
{{ "/my/fancy/url" | append: ".html" }}
```

Output

```
/my/fancy/url.html
```

`append` can also be used with variables:

Input

```
{% assign filename = "/index.html" %}
{{ "website.com" | append: filename }}
```

Output

```
website.com/index.html
```

▼

at_least

Limits a number to a minimum value.

Input

```
{{ 4 | at_least: 5 }}
```

Output

```
5
```

Input

```
{{ 4 | at_least: 3 }}
```

Output

```
4
```

▼

at_most

Limits a number to a maximum value.

Input

```
{{ 4 | at_most: 5 }}
```

Output

```
4
```

Input

```
{{ 4 | at_most: 3 }}
```

Output

```
3
```

▼

capitalize

Makes the first character of a string capitalized.

Input

{{ "title" | capitalize }}

Output

Title capitalize only capitalizes the first character of a string, so later words are not affected:

Input

{{ "my great title" | capitalize }}

Output

My great title

ceil

Rounds the input up to the nearest whole number. Liquid tries to convert the input to a number before the filter is applied.

Input

```
{{ 1.2 | ceil }}
```

Output

```
2
```

Input

```
{{ 2.0 | ceil }}
```

Output

```
2
```

Input

```
{{ 183.357 | ceil }}
```

Output

```
184
```

Here the input value is a string:

Input

```
{{ "3.5" | ceil }}
```

Output

```
4
```

compact

Removes any `nil` values from an array.

For this example, assume `site.pages` is an array of content pages for a website, and some of these pages have an attribute called `category` that specifies their content category. If we `map` those categories to an array, some of the array items might be `nil` if any pages do not have a `category` attribute.

Input

```
{% assign site_categories = site.pages | map: "category" %}

{% for category in site_categories %}
- {{ category }}
{% endfor %}
```

Output

```
- business
- celebrities
-
- lifestyle
- sports
-
- technology
```

By using `compact` when we create our `site_categories` array, we can remove all the `nil` values in the array.

Input

```
{% assign site_categories = site.pages | map: "category" | compact %}

{% for category in site_categories %}
- {{ category }}
{% endfor %}
```

Output

```
- business
- celebrities
- lifestyle
- sports
- technology
```

▼

concat

Concatenates (joins together) multiple arrays. The resulting array contains all the items from the input arrays.

Input

```
{% assign fruits = "apples, oranges, peaches" | split: ", " %}
{% assign vegetables = "carrots, turnips, potatoes" | split: ", " %}

{% assign everything = fruits | concat: vegetables %}

{% for item in everything %}
- {{ item }}
{% endfor %}
```

Output

```
- apples
- oranges
- peaches
- carrots
- turnips
- potatoes
```

You can string together `concat` filters to join more than two arrays:

Input

```
{% assign furniture = "chairs, tables, shelves" | split: ", " %}

{% assign everything = fruits | concat: vegetables | concat: furniture %}

{% for item in everything %}
- {{ item }}
{% endfor %}
```

Output

```
- apples
- oranges
- peaches
- carrots
- turnips
- potatoes
- chairs
- tables
- shelves
```

▼

date

Converts a timestamp into another date format. The format for this syntax is the same as `strftime`. The input uses the same format as Ruby's `Time.parse`.

Input

```
{{ article.published_at | date: "%a, %b %d, %y" }}
```

Output

```
Fri, Jul 17, 15
```

Input

```
{{ article.published_at | date: "%Y" }}
```

Output

```
2015
```

`date` works on strings if they contain well-formatted dates:

Input

```
{{ "March 14, 2016" | date: "%b %d, %y" }}
```

Output

```
Mar 14, 16
```

To get the current time, pass the special word `"now"` (or `"today"`) to `date`:

Input

```
This page was last updated at {{ "now" | date: "%Y-%m-%d %H:%M" }}.
```

Output

```
This page was last updated at 2019-09-19 17:48.
```

Note that the value will be the current time of when the page was last generated from the template, not when the page is presented to a user if caching or static site generation is involved.

▼

day_of_week

Gets the numeric representation of the day of the week, 0-6 (Sunday-Saturday).

Input

```
{{ "2020-11-20" | day_of_week }}
```

Output

```
5
```

▼

display_day_of_week

Gets the name of the day of the week.

Input

```
{{ "2020-11-20" | day_of_week }}
```

Output

```
Friday
```

▼

default

Allows you to specify a fallback in case a value doesn't exist. `default` will show its value if the left side is `nil`, `false`, or empty.

In this example, `product_price` is not defined, so the default value is used.

Input

```
{{ product_price | default: 2.99 }}
```

Output

```
2.99
```

In this example, `product_price` is defined, so the default value is not used.

Input

```
{% assign product_price = 4.99 %}
{{ product_price | default: 2.99 }}
```

Output

```
4.99
```

In this example, `product_price` is empty, so the default value is used.

Input

```
{% assign product_price = "" %}
{{ product_price | default: 2.99 }}
```

Output

```
2.99
```

▼

divided_by

Divides a number by another number.

The result is rounded down to the nearest integer (that is, the floor) if the divisor is an integer.

Input

```
{{ 16 | divided_by: 4 }}
```

Output

```
4
```

Input

```
{{ 5 | divided_by: 3 }}
```

Output

```
1
```

Controlling rounding

`divided_by` produces a result of the same type as the divisor — that is, if you divide by an integer, the result will be an integer. If you divide by a float (a number with a decimal in it), the result will be a float.

For example, here the divisor is an integer:

Input

```
{{ 20 | divided_by: 7 }}
```

Output

```
2
```

Here it is a float:

Input

```
{{ 20 | divided_by: 7.0 }}
```

Output

```
2.857142857142857
```

Changing variable types

You might want to use a variable as a divisor, in which case you can't simply add `.0` to convert it to a float. In these cases, you can `assign` a version of your variable converted to a float using the `times` filter.

In this example, we're dividing by a variable that contains an integer, so we get an integer:

Input

```
{% assign my_integer = 7 %}
{{ 20 | divided_by: my_integer }}
```

Output

```
2
```

Here, we multiply the variable by `1.0` to get a float, then divide by the float instead:

Input

```
{% assign my_integer = 7 %}
{% assign my_float = my_integer | times: 1.0 %}
{{ 20 | divided_by: my_float }}
```

Output

```
2.857142857142857
```

▼

downcase

Makes each character in a string lowercase. It has no effect on strings which are already all lowercase.

Input

```
{{ "Parker Moore" | downcase }}
```

Output

```
parker moore
```

Input

```
{{ "apple" | downcase }}
```

Output

```
apple
```

▼

escape

Escapes a string by replacing characters with escape sequences (so that the string can be used in a URL, for example). It doesn't change strings that don't have anything to escape.

Input

```
{{ "Have you read 'James & the Giant Peach'?" | escape }}
```

Output

```
Have you read 'James &amp; the Giant Peach'?
```

Input

```
{{ "Tetsuro Takara" | escape }}
```

Output

```
Tetsuro Takara
```

▼

escape_once

Escapes a string without changing existing escaped entities. It doesn't change strings that don't have anything to escape.

Input

```
{{ "1 < 2 & 3" | escape_once }}
```

Output

```
1 < 2 & 3
```

Input

```
{{ "1 < 2 & 3" | escape_once }}
```

Output

```
1 < 2 & 3
```

▼

first

Returns the first item of an array.

Input

```
{{ "Ground control to Major Tom." | split: " " | first }}
```

Output

```
Ground
```

Input

```
{% assign my_array = "zebra, octopus, giraffe, tiger" | split: ", " %}

{{ my_array.first }}
```

Output

```
zebra
```

You can use `first` with dot notation when you need to use the filter inside a tag:

```
{% if my_array.first == "zebra" %}
  Here comes a zebra!
{% endif %}
```

▼

floor

Rounds the input down to the nearest whole number. Liquid tries to convert the input to a number before the filter is applied.

Input

```
{{ 1.2 | floor }}
```

Output

```
1
```

Input

```
{{ 2.0 | floor }}
```

Output

```
2
```

Input

```
{{ 183.357 | floor }}
```

Output

```
183
```

Here the input value is a string:

Input

```
{{ "3.5" | floor }}
```

Output

```
3
```

▼

join

Combines the items in an array into a single string using the argument as a separator.

Input

```
{% assign beatles = "John, Paul, George, Ringo" | split: ", " %}

{{ beatles | join: " and " }}
```

Output

```
John and Paul and George and Ringo
```

▼

last

Returns the last item of an array.

Input

```
{{ "Ground control to Major Tom." | split: " " | last }}
```

Output

```
Tom.
```

Input

```
{% assign my_array = "zebra, octopus, giraffe, tiger" | split: ", " %}

{{ my_array.last }}
```

Output

```
tiger
```

You can use `last` with dot notation when you need to use the filter inside a tag:

```
{% if my_array.last == "tiger" %}
  There goes a tiger!
{% endif %}
```

▼

lstrip

Removes all whitespace (tabs, spaces, and newlines) from the left side of a string. It does not affect spaces between words.

Input

```
{{ "         So much room for activities!         " | lstrip }}
```

Output

```
So much room for activities!
```

▼

map

Creates an array of values by extracting the values of a named property from another object.

In this example, assume the object `site.pages` contains all the metadata for a website. Using `assign` with the `map` filter creates a variable that contains only the values of the `category` properties of everything in the `site.pages` object.

Input

```
{% assign all_categories = site.pages | map: "category" %}

{% for item in all_categories %}
- {{ item }}
{% endfor %}
```

Output

```
- business
- celebrities
- lifestyle
- sports
- technology
```

▼

minus

Subtracts a number from another number.

Input

```
{{ 4 | minus: 2 }}
```

Output

```
2
```

Input

```
{{ 16 | minus: 4 }}
```

Output

```
12
```

Input

```
{{ 183.357 | minus: 12 }}
```

Output

```
171.357
```

▼

modulo

Returns the remainder of a division operation.

Input

```
{{ 3 | modulo: 2 }}
```

Output

```
1
```

Input

```
{{ 24 | modulo: 7 }}
```

Output

```
3
```

Input

```
{{ 183.357 | modulo: 12 }}
```

Output

```
3.357
```

▼

newline_to_br

Replaces every newline (`\n`) in a string with an HTML line break (`<br />`).

Input

```
{% capture string_with_newlines %}
Hello
there
{% endcapture %}

{{ string_with_newlines | newline_to_br }}
```

Output

```
Hello

there
```

▼

plus

Adds a number to another number.

Input

```
{{ 4 | plus: 2 }}
```

Output

```
6
```

Input

```
{{ 16 | plus: 4 }}
```

Output

```
20
```

Input

```
{{ 183.357 | plus: 12 }}
```

Output

```
195.357
```

▼

prepend

Adds the specified string to the beginning of another string.

Input

```
{{ "apples, oranges, and bananas" | prepend: "Some fruit: " }}
```

Output

```
Some fruit: apples, oranges, and bananas
```

`prepend` can also be used with variables:

Input

```
{% assign url = "example.com" %}
{{ "/index.html" | prepend: url }}
```

Output

```
example.com/index.html
```

▼

remove

Removes every occurrence of the specified substring from a string.

Input

```
{{ "I strained to see the train through the rain" | remove: "rain" }}
```

Output

```
I sted to see the t through the
```

▼

remove_first

Removes only the first occurrence of the specified substring from a string.

Input

```
{{ "I strained to see the train through the rain" | remove_first: "rain" }}
```

Output

```
I sted to see the train through the rain
```

▼

replace

Replaces every occurrence of the first argument in a string with the second argument.

Input

```
{{ "Take my protein pills and put my helmet on" | replace: "my", "your" }}
```

Output

```
Take your protein pills and put your helmet on
```

▼

replace_first

Replaces only the first occurrence of the first argument in a string with the second argument.

Input

```
{{ "Take my protein pills and put my helmet on" | replace_first: "my", "your" }}
```

Output

```
Take your protein pills and put my helmet on
```

▶

money

▼

number

Attempts to force numeric behavior for a string. Limited applicability.

▼

reverse

Reverses the order of the items in an array. `reverse` cannot reverse a string.

Input

```
{% assign my_array = "apples, oranges, peaches, plums" | split: ", " %}

{{ my_array | reverse | join: ", " }}
```

Output

```
plums, peaches, oranges, apples
```

Although `reverse` cannot be used directly on a string, you can split a string into an array, reverse the array, and rejoin it by chaining together filters:

Input

```
{{ "Ground control to Major Tom." | split: "" | reverse | join: "" }}
```

Output

```
.moT rojaM ot lortnoc dnuorG
```

▼

round

Rounds a number to the nearest integer or, if a number is passed as an argument, to that number of decimal places.

Input

```
{{ 1.2 | round }}
```

Output

```
1
```

Input

```
{{ 2.7 | round }}
```

Output

```
3
```

Input

```
{{ 183.357 | round: 2 }}
```

Output

```
183.36
```

▼

rstrip

Removes all whitespace (tabs, spaces, and newlines) from the right side of a string. It does not affect spaces between words.

Input

```
{{ "          So much room for activities!          " | rstrip }}
```

Output

```
          So much room for activities!
```

▼

size

Returns the number of characters in a string or the number of items in an array.

Input

```
{{ "Ground control to Major Tom." | size }}
```

Output

```
28
```

Input

```
{% assign my_array = "apples, oranges, peaches, plums" | split: ", " %}

{{ my_array.size }}
```

Output

```
4
```

You can use `size` with dot notation when you need to use the filter inside a tag:

```
{% if site.pages.size > 10 %}
  This is a big website!
{% endif %}
```

▼

short_date

Converts a date to a concise string representation.

Input

```
{{ "2020-08-15 08:15:33" | short_date }}
```

Output

```
8/15/2020
```

▼

slice

Returns a substring of 1 character beginning at the index specified by the first argument. An optional second argument specifies the length of the substring to be returned.

String indices are numbered starting from 0.

Input

```
{{ "Liquid" | slice: 0 }}
```

Output

```
L
```

Input

```
{{ "Liquid" | slice: 2 }}
```

Output

```
q
```

Input

```
{{ "Liquid" | slice: 2, 5 }}
```

Output

```
quid
```

If the first argument is a negative number, the indices are counted from the end of the string:

Input

```
{{ "Liquid" | slice: -3, 2 }}
```

Output

```
ui
```

▼

sort

Sorts items in an array in case-sensitive order.

Input

```
{% assign my_array = "zebra, octopus, giraffe, Sally Snake" | split: ", " %}

{{ my_array | sort | join: ", " }}
```

Output

```
Sally Snake, giraffe, octopus, zebra
```

An optional argument specifies which property of the array's items to use for sorting.

```
{% assign products_by_price = collection.products | sort: "price" %}
{% for product in products_by_price %}


  {{ product.title }}
{% endfor %}
```

▼

sort_natural

Sorts items in an array in case-insensitive order.

Input

```
{% assign my_array = "zebra, octopus, giraffe, Sally Snake" | split: ", " %}

{{ my_array | sort_natural | join: ", " }}
```

Output

```
giraffe, octopus, Sally Snake, zebra
```

An optional argument specifies which property of the array's items to use for sorting.

```
{% assign products_by_company = collection.products | sort_natural: "company" %}
{% for product in products_by_company %}


  {{ product.title }}
{% endfor %}
```

▼

split

Divides a string into an array using the argument as a separator. `split` is commonly used to convert comma-separated items from a string to an array.

Input

```
{% assign beatles = "John, Paul, George, Ringo" | split: ", " %}

{% for member in beatles %}
  {{ member }}
{% endfor %}
```

Output

```
John

Paul

George

Ringo
```

▼

strip

Removes all whitespace (tabs, spaces, and newlines) from both the left and right sides of a string. It does not affect spaces between words.

Input

```
{{ "     So much room for activities!     " | strip }}
```

Output

```
So much room for activities!
```

▼

strip_html

Removes any HTML tags from a string.

Input

```
{{ "Have *you* read **Ulysses**?" | strip_html }}
```

Output

```
Have you read Ulysses?
```

▼

strip_newlines

Removes any newline characters (line breaks) from a string.

Input

```
{% capture string_with_newlines %}
Hello
there
{% endcapture %}

{{ string_with_newlines | strip_newlines }}
```

Output

```
Hellothere
```

▼

times

Multiplies a number by another number.

Input

```
{{ 3 | times: 2 }}
```

Output

```
6
```

Input

```
{{ 24 | times: 7 }}
```

Output

```
168
```

Input

```
{{ 183.357 | times: 12 }}
```

Output

```
2200.284
```

▼

truncate

Shortens a string down to the number of characters passed as an argument. If the specified number of characters is less than the length of the string, an ellipsis (…) is appended to the string and is included in the character count.

Input

```
{{ "Ground control to Major Tom." | truncate: 20 }}
```

Output

```
Ground control to...
```

Custom ellipsis

`truncate` takes an optional second argument that specifies the sequence of characters to be appended to the

truncated string. By default this is an ellipsis (…), but you can specify a different sequence.

The length of the second argument counts against the number of characters specified by the first argument. For example, if you want to truncate a string to exactly 10 characters, and use a 3-character ellipsis, use **13** for the first argument of `truncate`, since the ellipsis counts as 3 characters.

Input

```
{{ "Ground control to Major Tom." | truncate: 25, ", and so on" }}
```

Output

```
Ground control, and so on
```

No ellipsis

You can truncate to the exact number of characters specified by the first argument and avoid showing trailing characters by passing a blank string as the second argument:

Input

```
{{ "Ground control to Major Tom." | truncate: 20, "" }}
```

Output

```
Ground control to Ma
```

▼

truncatewords

Shortens a string down to the number of words passed as an argument. If the specified number of words is less than the number of words in the string, an ellipsis (…) is appended to the string.

Input

```
{{ "Ground control to Major Tom." | truncatewords: 3 }}
```

Output

```
Ground control to...
```

Custom ellipsis

`truncatewords` takes an optional second argument that specifies the sequence of characters to be appended to the truncated string. By default this is an ellipsis (…), but you can specify a different sequence.

Input

```
{{ "Ground control to Major Tom." | truncatewords: 3, "--" }}
```

Output

```
Ground control to--
```

No ellipsis

You can avoid showing trailing characters by passing a blank string as the second argument:

Input

```
{{ "Ground control to Major Tom." | truncatewords: 3, "" }}
```

Output

```
Ground control to
```

▼

uniq

Removes any duplicate elements in an array.

Input

```
{% assign my_array = "ants, bugs, bees, bugs, ants" | split: ", " %}

{{ my_array | uniq | join: ", " }}
```

Output

```
ants, bugs, bees
```

▼

upcase

Makes each character in a string uppercase. It has no effect on strings which are already all uppercase.

Input

```
{{ "Parker Moore" | upcase }}
```

Output

```
PARKER MOORE
```

Input

```
{{ "APPLE" | upcase }}
```

Output

```
APPLE
```

▼

url_decode

Decodes a string that has been encoded as a URL or by url_encode.

Input

```
{{ "%27Stop%21%27+said+Fred" | url_decode }}
```

Output

```
'Stop!' said Fred
```

▼

url_encode

Converts any URL-unsafe characters in a string into percent-encoded characters.

Input

```
{{ "john@liquid.com" | url_encode }}
```

Output

```
john%40liquid.com
```

Input

```
{{ "Tetsuro Takara" | url_encode }}
```

Output

```
Tetsuro+Takara
```

▼

where

Creates an array including only the objects with a given property value, or any truthy value by default.

In this example, assume you have a list of products and you want to show your kitchen products separately. Using `where`, you can create an array containing only the products that have a `"type"` of `"kitchen"`.

Input

```
All products:
{% for product in products %}
- {{ product.title }}
{% endfor %}

{% assign kitchen_products = products | where: "type", "kitchen" %}

Kitchen products:
{% for product in kitchen_products %}
- {{ product.title }}
{% endfor %}
```

Output

```
All products:
- Vacuum
- Spatula
- Television
- Garlic press

Kitchen products:
- Spatula
- Garlic press
```

Say instead you have a list of products and you only want to show those that are available to buy. You can `where` with a property name but no target value to include all products with a truthy `"available"` value.

Input

```
All products:
{% for product in products %}
- {{ product.title }}
{% endfor %}

{% assign available_products = products | where: "available" %}

Available products:
{% for product in available_products %}
- {{ product.title }}
{% endfor %}
```

Output

```
All products:
- Coffee mug
- Limited edition sneakers
- Boring sneakers

Available products:
- Coffee mug
- Boring sneakers
```

The `where` filter can also be used to find a single object in an array when combined with the `first` filter. For example, say you want to show off the shirt in your new fall collection.

Input

```
{% assign new_shirt = products | where: "type", "shirt" | first %}

Featured product: {{ new_shirt.title }}
```

Output

```
Featured product: Hawaiian print sweater vest
```

▼

xml_encode

Substitutes XML special characters with encoded representations.

Input

```
{{ "<Cell>this might break a template if unencoded</Cell>" | xml_encode }}
```

Output

```
&lt;Cell&gt;this might break a template if unencoded&lt;/Cell&gt;
```