# Rallybound Liquid Basics

## Liquid syntax

Like traditional programming languages, Liquid has a syntax, interacts with variables, and includes constructs such as output and logic. Due to its readable syntax, Liquid constructs are easy to recognize, and can be distinguished from HTML by two sets of delimiters: the double curly brace delimiters {{ }}, which denote output, and the curly brace percentage delimiters {% %}, which denote logic and control flow.

There are three main features of Liquid code:

- Objects
- Tags
- Filters

## Objects

Rallybound Liquid objects output pieces of data from a Rallybound campaign. When adding an object to an email or webpage, objects are wrapped in double curly brace delimiters {{ }}, and look like this:

| **Input** |
| --- |
| {{ Campaign.Name }} |

In the above example, Campaign is the object, and Name is an attribute of that object. Each object has a list of associated attributes.

The {{ Campaign.Name }} Liquid object can be found in the Placeholders popup in the edit Auto Responders interface and can be applied to emails and/or text-editable areas on your site. When the code in the email or webpage is compiled and rendered, the output of the Liquid object will be the campaign name. For example, the result might be:

| **Output** |
| --- |
| 5K Run |

This object can be used in different campaigns, and each campaign will output different data depending on that campaign's name. This is very useful when creating standard emails across campaigns, and when replicating or relaunching a campaign — instead of using the actual campaign name, use the Campaign.Name object and it will automatically output the correct campaign name.

To learn more about the different Liquid objects that can be used in emails and webpages, see the Rallybound Liquid objects page.

## Tags

Liquid tags are used to create logic and control flow for templates. The curly brace percentage delimiters {% %} and the text that they surround do not produce any visible output when the email or webpage is rendered. This lets you assign variables and create conditions or loops without showing any of the Liquid logic on the page.

For example, you can use Liquid tags to display different content in a registration confirmation email depending on whether or not there was a registration fee:

| **Input** |
| --- |
| {% if Payment.Amount %}<br>You will be receiving your t-shirt in the mail soon!<br>{% else %}<br>Thank you for your registration!<br>{% endif %} |

If there was a registration fee, then the output will be:

| **Output** |
| --- |
| You will be receiving your t-shirt in the mail soon! |

If there was no registration fee, then the output will be:

| Output |
| --- |
| Thank you for your registration! |

To learn more about the different Liquid tags that can be used in emails and webpages, see the Rallybound Liquid tags page. The above example uses if and else Liquid tags, which are control flow tags.

## Filters

Liquid filters are used to modify the output of numbers, strings, objects, and variables. They are placed within an output tag {{ }}, and are denoted by a pipe character |.

One example is the money number filter:

| Input |
| --- |
| {{ 50 \| money }} |

The filter modifies the number by displaying it as a dollar amount. The output will be:

| Output |
| --- |
| $50.00 |

Multiple filters can be used on one output, and they are applied from left to right:

| Input |
| --- |
| {{ 50 \| minus: 10 \| money }} |

The number is first calculated, and then the dollar sign is prefixed. The output will be:

| Output |
| --- |
| $40.00 |

You can use Liquid filters to make a wide variety of useful data manipulations. To learn more about the different Liquid filters that can be used in emails and webpages, see the Rallybound Liquid filters page.

### Please click through to learn more about Liquid basics:

- Operators
- Truthy and Falsy
- Types
- Whitespace Control