We have seen filters in JavaScript like in a list comprehension, also like functions toUpperCase(). It seems like vue used to have default filters, but now users have to create their own filters. So how to custom filters in vuejs? The idea is very alike, we do not change the input value in the data, however we are actually changing how the data displays in the browser.

Very similar to how we create components and directives. Here I created a filter globally in main as below:

Vue.filter(‘name of filter’,function(value){

return value.toUpperCase();

});

Vue.filter(‘cutParagraph’,function(value){

return value.slice(0,80);

});

After we created the filter, we want to use it. The syntax is very much alike to our command line.

For example, if we want to apply cutParagraph to our body content to our blog. We can simply write:

{{blog.body | cutParagraph}}

But is filter always a good idea to use? The answer is no. If we are dealing with a large search array, the better option is to use computed property or methods instead. The promoted one is computed since methods will recall the function every time it executes during re-rendering, yet using computed property, it will only be used when any changes has been made. The reason is because computed property is relied on cache.

The way to declare it is like:

computed: {

searchBlog:function(){

return this.blogs.filter((blog)=>{

return boolean;

})

}

};

How to register filter locally in a specific component?

filters: {

'upper':function(value){

return value.toUpperCase();

}

Or

upper(value){

return value.toUpperCase();

}