# What is the difference between the remap, noremap, nnoremap and vnoremap mapping commands in vim?



What is the difference between the remap, noremap, nnoremap and vnoremap mapping commands in vim?

vim mapping command

edited Mar 5 '14 at 14:35

TRiG
3,897 1 28 60

asked Sep 23 '10 at 7:13

Chetan
14.6k 15 66 113

## 4 Answers

`remap` is an **option** that makes mappings work recursively. By default it is on and I'd recommend you leave it that way. The rest are **mapping commands**, described below:

`:map` and `:noremap` are **recursive** and **non-recursive** versions of the various mapping commands. What that means is that if you do:

```
:map j gg
:map Q j
:noremap W j
```

`j` will be mapped to `gg`. `Q` will *also* be mapped to `gg`, because `j` will be expanded for the recursive mapping. `W` will be mapped to `j` (and not to `gg`) because `j` will not be expanded for the non-recursive mapping.

Now remember that Vim is a **modal editor**. It has a **normal** mode, **visual** mode and other modes.

For each of these sets of mappings, there is a mapping that works in normal, visual, select and operator modes (`:map` and `:noremap`), one that works in normal mode (`:nmap` and `:nnoremap`), one in visual mode (`:vmap` and `:vnoremap`) and so on.

For more guidance on this, see:

```
:help :map
:help :noremap
:help recursive_mapping
:help :map-modes
```

edited Oct 5 '14 at 7:14

Sparhawk
199 1 13

answered Sep 23 '10 at 7:24

DrAl
33.7k 6 55 76

2 Thanks for your answer! Also, when is recursive used, and when is non-recursive used? – Chetan Sep 29 '10 at 22:35

4 @Chetan: It depends what you want to achieve. I tend to use non-recursive more often, but if you've defined a relatively complicated mapping using non-recursive and what another mapping that does everything the first mapping does and more, it can be easier to use a recursive mapping that includes the original one rather than retyping the whole of the non-recursive one again (particularly if you then need to tweak the original one). – DrAl Sep 30 '10 at 7:02

37 A well written answer, I wish ViM documentation on these was this clear! :-) – Ashwin Nov 24 '10 at 3:19

5 I had assumes `noremap` to be some opposite of `map`. I mean something which removes a mapping. Thanks for the answer. It clarified me – xeek Mar 14 '12 at 9:07

2 +1 for providing the appropriate help subjects. – naught101 Mar 19 '12 at 1:01



I think the vim documentation should've explained the meaning behind the naming of these commands. Just telling you what they do doesn't help you remember the names.

`map` is the "root" of all recursive mapping commands. The root form applies to "normal", "visual+select", and "operator-pending" modes.

`noremap` is the "root" of all non-recursive mapping commands. The root form applies to the same modes as `map`.

(Note that there are also the `!` modes like `map!` that apply to insert & command-line.)

See below for what "recursive" means in this context.

Prepending a mode letter like `n` modify the modes the mapping works in. It can choose a subset of the list of applicable modes (e.g. only "visual"), or choose other modes that `map` wouldn't apply to (e.g. "insert").

Use `help map-modes` will show you a few tables that explain how to control which modes the mapping applies to.

Mode letters:

- `n` : normal only
- `v` : visual and select
- `o` : operator-pending
- `x` : visual only
- `s` : select only
- `i` : insert
- `c` : command-line
- `l` : insert, command-line, regexp-search (and others. Collectively called "Lang-Arg" pseudo-mode)

"**Recursive**" means that the mapping is expanded to a result, then the result is expanded to another result, and so on.

The expansion stops when one of these is true:

1. the result is no longer mapped to anything else.
2. a non-recursive mapping has been applied (i.e. the "noremap" [or one of its ilk] is the final expansion).

At that point, vim's default "meaning" of the final result is applied/executed.

"**Non-recursive**" means the mapping is only expanded once, and that result is applied/executed.

Example:

```
nmap K H
nnoremap H G
nnoremap G gg
```

The above causes `K` to expand to `H`, then `H` to expand to `G` and stop. It stops because of the `nnoremap`, which expands and stops immediately. The meaning of `G` will be executed (i.e. "jump to last line"). At most one non-recursive mapping will ever be applied in an expansion chain (it would be the last expansion to happen).

The mapping of `G` to `gg` only applies if you press `G`, but not if you press `K`. This would be true regardless of whether `G` was mapped recursively or not, since it's line 2 that caused the expansion of `K` to stop.

edited Oct 14 '14 at 17:25 answered Jul 26 '12 at 19:00


thirtythreeforty
**2,310** 8 31


Kelvin
**7,240** 25 41

1 One thing: `map` only applies to normal, visual, select, and operator-pending modes, not to all modes. – Sort of a beginner Sep 20 '14 at 3:36

@Sortofabeginner yes, you're correct. Updated. – Kelvin Sep 22 '14 at 17:02

Excellent! +1 for listing the mode letters. – user3751385 Apr 2 at 8:42

Caution, `vnoremap` and `vmap` work in Visual AND Select mode. To have a mapping only in Visual mode, use `xmap` and `xnoremap` .

edited Aug 1 '11 at 3:11

ib.
**15.8k** 5 31 50

answered Sep 24 '10 at 13:58

Benoit
**38.7k** 10 100 161

I think this should be a comment. – lindhe Dec 5 '14 at 13:52

Note: `map` **does not map to all modes**.

Quoting the `help map-modes` tables

- `:map` does `nvo` == normal + (visual + select) + operator pending
- `:map!` does `ic` == insert + command

So to map to all modes you need both `:map` and `:map!` .

edited Nov 21 '14 at 9:13

answered Jan 22 '14 at 12:03

Ciro Santilli 六四事件 法轮
**9,631** 4 57 48

This should be a comment. – lindhe Dec 5 '14 at 13:53