Control structures reference

Every parameter of if, elseif and else, switch and case (match, in, key and the default case), foreach (key, sequence, offset, max, reverse), for, while and do, and the loop helpers delimiter (with modulo), break...

Source file: doc/eng-us/help/template-language/control-structures-reference.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/template-language/control-structures-reference.md

This page lists every parameter of the branching and looping functions, as implemented in `lib/eztemplate/classes/` (`eztemplateiffunction.php`, `eztemplateswitchfunction.php`, `eztemplateforeachfunction.php`, `eztemplateforfunction.php`, `eztemplatewhilefunction.php`, `eztemplatedofunction.php` and the shared loop helper `eztemplateloop.php`). The special loop grammar is parsed in `eztemplatemultipassparser.php`. For gentler introductions see the designer pages linked at the end.

Contents

1. if, elseif, else
2. What counts as true
3. switch and case
4. foreach
5. break, continue, skip
6. delimiter and modulo
7. for
8. while
9. do
10. Related pages

if, elseif, else

```
{if <condition>} ... [{elseif <condition>} ...] [{else} ...] {/if}
```

The condition is one expression; combine tests with `and()`, `or()` and `not()`. `elseif` and `else` are child tags of `if` (its `function_attributes`).

Template

```
{foreach array( 0, 1, 7 ) as $n}
{if $n|eq( 0 )}none{elseif $n|eq( 1 )}one{else}{$n} items{/if}
{/foreach}
```

Output:

```
noneone7 items
```

What counts as true

Conditions use PHP truthiness: `0`, `'0'`, `''`, an empty array, `false()` and null are false. The values tested here are `0`, `1`, `''`, `'x'`, `'0'`, `array()`, `array( 0 )`, `false()`, `true()`:

Template

```
{foreach array( 0, 1, '', 'x', '0', array(), array( 0 ), false(), true() ) as $v}[{if $v}T{else}F
{/if}]{/foreach}
```

Output:

```
[F][T][F][T][F][F][T][F][T]
```

switch and case

```
{switch match=<value>}
  {case match=<value>} ... {/case}
  {case in=<array> [key=<array of attribute names>]} ... {/case}
  {case} ... {/case}        (the default)
{/switch}
```

Parameter	Of	Meaning
match	switch	the value to compare; required
match	case	this case is taken when it equals the switch value (==)
in	case	this case is taken when the switch value is in the array (in_array)
key	case with in	compare the switch value with this attribute of each array element instead
none	case	the default case

The first matching case wins. A second case with the same `match` value is skipped with a warning.

Template

```
{foreach array( 1, 3, 'other' ) as $v}
{switch match=$v}{case match=1}one{/case}{case in=array( 2, 3 )}two or three{/case}{case}default
for {$v}{/case}{/switch}
{/foreach}
```

Output:

```
onetwo or threedefault for other
```

With `key`, the code passes the value to `eZTemplate::variableAttribute()`, which walks a **list** of attribute names. A plain string is not turned into a list there, so only the array form finds the element; this is what a development installation does today:

Template

```
{def $people = array( hash( 'id', 7, 'name', 'Ada' ), hash( 'id', 9, 'name', 'Lin' ) )}
key as a string: {switch match=9}{case in=$people key='id'}found{/case}{case}not found{/case}{/sw
itch}
key as an array: {switch match=9}{case in=$people key=array( 'id' )}found{/case}{case}not found{/
case}{/switch}
```

Output:

```
key as a string: not foundkey as an array: found
```

When nothing matches and there is no default case, the switch prints nothing and warns:

Template

```
[{switch match=5}{case match=1}one{/case}{/switch}]
```

Output:

```
[]
```

Logged by the template engine:

```
WARNING switch: No case match and no default case
```

foreach

```
{foreach <array> as [$key =>] $item
         [sequence <array> as $seq] [offset <n>] [max <n>] [reverse]}
  ...
{/foreach}
```

Parameter	Meaning
`<array> as $item`	the array and the variable for each value; required
`$key =>`	a variable for each key
`sequence <array> as $seq`	cycles through the array, one value per iteration
`offset <n>`	start at position n; out of range is reset with a warning
`max <n>`	at most n iterations; a negative value is ignored with a warning
`reverse`	iterate from the end; a bare word, no value

`offset` and `max` count positions, not keys; with `reverse` the offset counts from the end. The key, item and sequence variables are ordinary variables of the file, not of a namespace.

Template

```
{foreach array( 'a', 'b', 'c', 'd', 'e' ) as $i => $v sequence array( 'x', 'y' ) as $s offset 1 m
ax 3}[{$i}:{$v}:{$s}]{delimiter}|{/delimiter}{/foreach}
{foreach array( 1, 2, 3, 4 ) as $v reverse}{$v}{/foreach}
{foreach hash( 'one', 1, 'two', 2 ) as $key => $value}{$key}={$value} {/foreach}
```

Output:

```
[1:b:x]|[2:c:y]|[3:d:x]4321one=1 two=2
```

An empty array runs the body zero times, without a warning:

Template

```
[{foreach array() as $v}{$v}{/foreach}] empty array: no output
```

Output:

```
[] empty array: no output
```

After the loop, do not rely on the loop variables. An interpreted template removes a variable the loop created, a compiled one keeps it with its last value; a variable that existed before the loop is overwritten in both modes:

Template

```
{foreach array( 1, 2 ) as $v}{/foreach}
[{if is_set( $v )}v still set{else}v removed after the loop{/if}]
{def $item = 'before'}
{foreach array( 1, 2 ) as $item}{/foreach}
[{$item}]
```

Output:

```
[v removed after the loop][2]
```

Output with the template compiler (compiled into a scratch folder):

```
[v still set][2]
```

With template compiling enabled, as on a live site, the compiled result is the one visitors get.

`foreach` does not loop over the characters of a string or over a number range; use `for` for ranges, and see Sections versus foreach for what `section` does with strings.

break, continue, skip

All four loops (`foreach`, `for`, `while`, `do`) accept these child tags, also inside an `if` within the loop (`eZTemplateLoop::processChildren()`):

Tag	Ends	Next iteration	Delimiter after this iteration	Sequence advances
{break}	the loop	no	no	no
{continue}	this iteration	yes	**yes**	no
{skip}	this iteration	yes	no	no

In this loop 2 is continued (its delimiter still prints, hence the double dash), 4 is skipped (no delimiter), and 6 breaks:

Template

```
{foreach array( 1, 2, 3, 4, 5, 6 ) as $v}{if eq( $v, 2 )}{continue}{/if}{if eq( $v, 4 )}{skip}{/i
f}{if eq( $v, 6 )}{break}{/if}{$v}{delimiter}-{/delimiter}{/foreach}
```

Output:

```
1--3-5-
```

delimiter and modulo

`{delimiter}...{/delimiter}` is printed between iterations. It must be a direct child of the loop. With `modulo=<n>` it is only printed when the iteration index divides by n:

Template

```
{foreach array( 1, 2, 3, 4, 5, 6 ) as $v}{$v}{delimiter modulo=2} / {/delimiter}{/foreach}
```

Output:

```
12 / 34 / 56
```

for

```
{for <first> to <last> as $i [sequence <array> as $seq]} ... {/for}
```

Counts from first to last inclusive, upward or downward. The loop helpers apply; in the third loop the delimiter after 3 is printed before the `{break}` of the next iteration runs:

Template

```
{for 1 to 5 as $i sequence array( 'o', 'e' ) as $s}{$i}{$s} {/for}
{for 5 to 1 as $i}{$i}{/for}
{for 0 to 10 as $i}{if $i|gt( 3 )}{break}{/if}{$i}{delimiter},{/delimiter}{/for}
```

Output:

```
1o 2e 3o 4e 5o 543210,1,2,3,
```

while

```
{while <condition> [sequence <array> as $seq]} ... {/while}
```

The condition is evaluated before every iteration. Change the tested variable inside the body with `{set}`, or the loop does not end:

Template

```
{def $n = 0}
{while $n|lt( 3 ) sequence array( 'a', 'b' ) as $s}[{$n}{$s}]{set $n = $n|inc}{/while}
```

Output:

```
[0a][1b][2a]
```

do

```
{do} ... {/do while <condition> [sequence <array> as $seq]}
```

The condition is written in the end tag and tested after each iteration, so the body runs at least once even when the condition is false from the start:

Template

```
{def $n = 5}
{do}[{$n}]{set $n = $n|inc}{/do while $n|lt( 3 )}
```

Output:

```
[5]
```

Related pages

- Previous: Functions versus operators
- Next: Sections versus foreach
- If and conditions, Switch and case
- Foreach loops, For loops, While and do loops
- foreach, switch, generated from the code
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.