Array-Operatoren

Die 24 Operatoren von eZTemplateArrayOperator (array, hash, append, prepend, merge, contains, compare, extract, extract_left, extract_right, begins_with, ends_with, implode, explode, repeat, reverse, insert, remove…

Source file: `doc/ger-de/hilfe/template-operatoren/array-operatoren.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/template-operators/array-operators.md

Arrays und Hashes (assoziative Arrays) werden mit Operatoren aufgebaut und verändert: Die Template-Sprache kennt keine Literal-Syntax wie `[1, 2, 3]`. Alle 24 Operatoren dieser Seite werden von `eZTemplateArrayOperator` (`lib/eztemplate/classes/eztemplatearrayoperator.php`) über `lib/eztemplate/classes/eztemplateautoload.php` registriert.

Die meisten arbeiten **auf Arrays und auf Zeichenketten**: `modify()` sieht sich zuerst die Eingabe an. Ist die gepipte Eingabe ein Array (oder der erste Parameter ein Array), nimmt es den Array-Zweig; ist die Eingabe eine Zeichenkette, nimmt es den Zeichenketten-Zweig, der mit den Multibyte-Funktionen `mb_*` auf Zeichen arbeitet. Die Abschnitte sagen, welche Zweige ein Operator hat. Kein Operator verändert die Variable, auf die er angewendet wird: Speichern Sie das Ergebnis mit `{set $list = $list|append( 4 )}`.

Die Beispiele wurden auf einer Entwicklungsinstallation vom Template-Interpreter und noch einmal vom Template-Compiler gerendert (das Kompilieren von Templates ist dort eingeschaltet). Wo die beiden voneinander abwichen, sagt es der Abschnitt.

Inhalt

1. Aufbauen: array und hash
2. Hinzufügen: append, prepend und merge
3. Prüfen: contains, compare, begins_with und ends_with
4. Ausschneiden: extract, extract_left und extract_right
5. Verbinden und aufteilen: implode und explode
6. Umformen: repeat, reverse und unique
7. Bearbeiten: insert, remove und replace
8. array_sum
9. Veraltete Namen: array_append, array_prepend und array_merge
10. Ein ausgearbeitetes Beispiel: Namen in einer Schleife sammeln
11. Verwandte Seiten

Aufbauen: array und hash

array

Gibt eine neue Liste aller seiner Parameter in ihrer Reihenfolge zurück. Nimmt keine Eingabe.

Parameter	Typ	Erforderlich	Standardwert
beliebig viele Werte	mixed	nein	keiner (ein leeres Array)

Generierte Referenz: array.

hash

Gibt ein assoziatives Array aus Parameterpaaren zurück: Schlüssel, Wert, Schlüssel, Wert. Ein Schlüssel muss eine Zeichenkette oder eine Zahl sein; jeder andere Schlüsseltyp protokolliert "Unknown hash key type", und das Paar wird übersprungen. Ein Schlüssel am Ende ohne Wert wird ignoriert.

Parameter	Typ	Erforderlich	Standardwert
Paare aus Schlüssel und Wert	Zeichenkette oder Zahl, mixed	nein	keiner (ein leeres Array)

Generierte Referenz: hash.

Template

```
{def $list = array( 'red', 'green', 'blue' )
     $map  = hash( 'name', 'Fit & Healthy', 'node_id', 89 )}
array: {$list|implode( ', ' )} ({$list|count} items)
hash: {foreach $map as $key => $value}{$key}={$value|wash} {/foreach}
```

Ausgabe:

```
array: red, green, blue (3 items)
hash: name=Fit &amp;amp; Healthy node_id=89
```

Hinzufügen: append, prepend und merge

append

- Array: Gibt das Eingabe-Array zurück, an dessen Ende jeder Parameter angefügt ist. Ohne Eingabe ist der erste Parameter das Array, und die übrigen werden angefügt (`append( $list, 9 )`). Ohne ein Element zum Anfügen protokolliert es "Requires at least one item!".
- Zeichenkette: Gibt die Eingabe zurück, an die jeder Parameter angehängt ist.

Parameter	Typ	Erforderlich	Standardwert
Elemente	mixed	ja, mindestens eines	keiner

Generierte Referenz: append.

prepend

- Array: wie `append`, aber die Elemente kommen in der angegebenen Reihenfolge an den Anfang.
- Zeichenkette: Gibt den benannten Parameter `prepend_string` gefolgt von der Eingabe

zurück. Nur der erste Parameter wird verwendet.

Parameter	Typ	Erforderlich	Standardwert
`prepend_string` (Eingabe als Zeichenkette) oder Elemente (Eingabe als Array)	mixed	nein	`false`

Generierte Referenz: prepend.

Template

```
{def $list = array( 2, 3 )}
$list|append( 4, 5 ) = {$list|append( 4, 5 )|implode( ',' )}
$list|prepend( 0, 1 ) = {$list|prepend( 0, 1 )|implode( ',' )}
append( $list, 9 ) = {append( $list, 9 )|implode( ',' )}
'Hello'|append( ', ', 'world' ) = {'Hello'|append( ', ', 'world' )}
'world'|prepend( 'Hello ' ) = {'world'|prepend( 'Hello ' )}
```

Ausgabe:

```
$list|append( 4, 5 ) = 2,3,4,5
$list|prepend( 0, 1 ) = 0,1,2,3
append( $list, 9 ) = 2,3,9
'Hello'|append( ', ', 'world' ) = Hello, world
'world'|prepend( 'Hello ' ) = Hello world
```

Verwenden Sie die Pipe-Form. Kompiliert gab die Aufrufform `append( $list, 9 )` auf einer Entwicklungsinstallation `Array,Array` statt `2,3,9` aus; der kompilierte Code (`mergeTrans()`, dem die Eingabe immer als Parameter übergeben wird) behandelt den ersten Parameter nicht als Ziel-Array. Alle anderen Zeilen waren identisch.

Ausgabe mit dem Template-Compiler (in einen Arbeitsordner kompiliert):

```
$list|append( 4, 5 ) = 2,3,4,5
$list|prepend( 0, 1 ) = 0,1,2,3
append( $list, 9 ) = Array,Array
'Hello'|append( ', ', 'world' ) = Hello, world
'world'|prepend( 'Hello ' ) = Hello world
```

merge

Nur Arrays: PHP `array_merge()` der Eingabe (oder eines leeren Arrays, wenn es keine gibt) mit jedem Parameter, der nicht `null` ist. Numerische Schlüssel werden neu nummeriert, Zeichenkettenschlüssel späterer Arrays überschreiben frühere; das Zusammenführen von Hashes ist also der Weg, einen Schlüssel zu ändern. Bei einer Zeichenkette als Eingabe protokolliert es "merge works only with arrays.".

Parameter	Typ	Erforderlich	Standardwert
Arrays	array	ja, mindestens eines	keiner

Generierte Referenz: merge.

Template

```
{def $a = array( 1, 2 )
     $b = array( 3 )
     $h = hash( 'color', 'red', 'size', 'L' )}
$a|merge( $b, array( 4 ) ) = {$a|merge( $b, array( 4 ) )|implode( ',' )}
merge( $a, $b ) = {merge( $a, $b )|implode( ',' )}
$h|merge( hash( 'size', 'XL' ) ) = {foreach $h|merge( hash( 'size', 'XL' ) ) as $k => $v}{$k}={$v
} {/foreach}
```

Ausgabe:

```
$a|merge( $b, array( 4 ) ) = 1,2,3,4
merge( $a, $b ) = 1,2,3
$h|merge( hash( 'size', 'XL' ) ) = color=red size=XL
```

Prüfen: contains, compare, begins_with und ends_with

Alle vier geben `true` oder `false` zurück. In den Beispielen macht `choose( 'false', 'true' )` aus dem Wahrheitswert ein Wort.

contains

- Array: PHP `in_array( match, input )` (loser Vergleich).
- Zeichenkette: ob die Eingabe den Text `match` enthält (`mb_strpos`).

Verwenden Sie es mit der Pipe: Das Array oder die Zeichenkette muss die Eingabe sein.

Parameter	Typ	Erforderlich	Standardwert
`match`	string	ja	`false`

Generierte Referenz: contains.

compare

- Array: wahr, wenn die Eingabe und das Array `compare` dieselben Werte enthalten, in beliebiger Reihenfolge (zwei Aufrufe von `array_diff()`; Schlüssel und Duplikate werden nicht verglichen).
- Zeichenkette: wahr, wenn `strcmp()` die Zeichenketten für identisch hält.

Parameter	Typ	Erforderlich	Standardwert
`compare`	mixed	ja	`false`

Generierte Referenz: compare.

Template

```
{def $list = array( 'news', 'blog', 'event' )}
$list|contains( 'blog' ) = {$list|contains( 'blog' )|choose( 'false', 'true' )}
$list|contains( 'shop' ) = {$list|contains( 'shop' )|choose( 'false', 'true' )}
'Fit & Healthy'|contains( 'Healthy' ) = {'Fit & Healthy'|contains( 'Healthy' )|choose( 'false', '
true' )}
$list|compare( array( 'event', 'news', 'blog' ) ) = {$list|compare( array( 'event', 'news', 'blog
' ) )|choose( 'false', 'true' )}
'abc'|compare( 'abc' ) = {'abc'|compare( 'abc' )|choose( 'false', 'true' )}
```

Ausgabe:

```
$list|contains( 'blog' ) = true
$list|contains( 'shop' ) = false
'Fit & Healthy'|contains( 'Healthy' ) = true
$list|compare( array( 'event', 'news', 'blog' ) ) = true
'abc'|compare( 'abc' ) = true
```

begins_with

- Array: wahr, wenn die ersten Elemente der Eingabe der Reihe nach den Parametern gleichen (`begins_with( 1, 2 )`).
- Zeichenkette: wahr, wenn die Eingabe mit dem Text `match` beginnt.

Parameter	Typ	Erforderlich	Standardwert
`match` (Zeichenkette) oder die zu prüfenden Elemente (Array)	mixed	ja	`false`

Generierte Referenz: begins_with.

ends_with

- Array: wahr, wenn die letzten Elemente der Eingabe in dieser Reihenfolge den Parametern gleichen.
- Zeichenkette: wahr, wenn die Eingabe mit dem Text `match` endet (`mb_strrpos`).

Parameter	Typ	Erforderlich	Standardwert
`match` (Zeichenkette) oder die zu prüfenden Elemente (Array)	mixed	ja	`false`

Generierte Referenz: ends_with.

Template

```
{def $list = array( 1, 2, 3, 4 )}
$list|begins_with( 1, 2 ) = {$list|begins_with( 1, 2 )|choose( 'false', 'true' )}
$list|ends_with( 3, 4 ) = {$list|ends_with( 3, 4 )|choose( 'false', 'true' )}
$list|ends_with( 2 ) = {$list|ends_with( 2 )|choose( 'false', 'true' )}
'recipes/pasta'|begins_with( 'recipes/' ) = {'recipes/pasta'|begins_with( 'recipes/' )|choose( 'f
alse', 'true' )}
'image.jpg'|ends_with( '.jpg' ) = {'image.jpg'|ends_with( '.jpg' )|choose( 'false', 'true' )}
```

Ausgabe:

```
$list|begins_with( 1, 2 ) = true
$list|ends_with( 3, 4 ) = true
$list|ends_with( 2 ) = false
'recipes/pasta'|begins_with( 'recipes/' ) = true
'image.jpg'|ends_with( '.jpg' ) = true
```

Ausschneiden: extract, extract_left und extract_right

extract

- Array: PHP `array_slice( input, extract_start, extract_length )`; ohne Länge bis zum Ende.
- Zeichenkette: `mb_substr()` mit denselben Parametern.

Parameter	Typ	Erforderlich	Standardwert
extract_start	integer	ja	false
extract_length	integer	nein	false (bis zum Ende)

Generierte Referenz: extract.

extract_left

Die ersten `length` Elemente oder Zeichen.

Parameter	Typ	Erforderlich	Standardwert
length	integer	ja	false

Generierte Referenz: extract_left.

extract_right

Die letzten `length` Elemente oder Zeichen.

Parameter	Typ	Erforderlich	Standardwert
length	integer	ja	false

Generierte Referenz: extract_right.

Template

```
{def $list = array( 'a', 'b', 'c', 'd', 'e' )}
$list|extract( 1, 3 ) = {$list|extract( 1, 3 )|implode( '' )}
$list|extract( 2 ) = {$list|extract( 2 )|implode( '' )}
$list|extract_left( 2 ) = {$list|extract_left( 2 )|implode( '' )}
$list|extract_right( 2 ) = {$list|extract_right( 2 )|implode( '' )}
'Exponential'|extract( 0, 3 ) = {'Exponential'|extract( 0, 3 )}
'Exponential'|extract_left( 4 ) = {'Exponential'|extract_left( 4 )}
'Exponential'|extract_right( 4 ) = {'Exponential'|extract_right( 4 )}
```

Ausgabe:

```
$list|extract( 1, 3 ) = bcd
$list|extract( 2 ) = cde
$list|extract_left( 2 ) = ab
$list|extract_right( 2 ) = de
'Exponential'|extract( 0, 3 ) = Exp
'Exponential'|extract_left( 4 ) = Expo
'Exponential'|extract_right( 4 ) = tial
```

Verbinden und aufteilen: implode und explode

implode

Nur Arrays: PHP `implode( separator, input )`, eine Zeichenkette. Bei einer Zeichenkette als Eingabe protokolliert es "implode only works with arrays".

Parameter	Typ	Erforderlich	Standardwert
separator	string	ja	false

Generierte Referenz: implode.

explode

Die beiden Zweige tun Verschiedenes:

- Array: teilt das Array an der Position `explode_first` in zwei Teile und gibt `array( erster Teil, Rest )` zurück.
- Zeichenkette: PHP `explode( explode_first, input )`, eine Liste der Stücke zwischen den Trennzeichen.

Parameter	Typ	Erforderlich	Standardwert
explode_first	mixed (Position oder Trennzeichen)	ja	false

Generierte Referenz: explode.

Template

```
{def $list = array( 'a', 'b', 'c', 'd' )
     $parts = $list|explode( 1 )}
$list|implode( ' / ' ) = {$list|implode( ' / ' )}
$list|explode( 1 ) = [{$parts[0]|implode( ',' )}] [{$parts[1]|implode( ',' )}]
'red,green,blue'|explode( ',' ) = {'red,green,blue'|explode( ',' )|implode( ' | ' )}
```

Ausgabe:

```
$list|implode( ' / ' ) = a / b / c / d
$list|explode( 1 ) = [a] [b,c,d]
'red,green,blue'|explode( ',' ) = red | green | blue
```

Umformen: repeat, reverse und unique

repeat

- Array: die Elemente `repeat_times`-mal hintereinander wiederholt.
- Zeichenkette: PHP `str_repeat()`.

Parameter	Typ	Erforderlich	Standardwert
`repeat_times`	integer	nein	1

Generierte Referenz: repeat.

reverse

Array: `array_reverse()`. Zeichenkette: `strrev()`, das Bytes umdreht und daher Multibyte-Zeichen zerstört. Keine Parameter. Generierte Referenz: reverse.

unique

Nur Arrays: `array_unique()`; die Schlüssel der ersten Vorkommen bleiben erhalten. Bei einer Zeichenkette protokolliert es "unique works only with arrays.". Keine Parameter. Generierte Referenz: unique.

Template

```
array( 1, 2 )|repeat( 3 ) = {array( 1, 2 )|repeat( 3 )|implode( ',' )}
'-'|repeat( 10 ) = {'-'|repeat( 10 )}
array( 1, 2, 3 )|reverse = {array( 1, 2, 3 )|reverse|implode( ',' )}
'abc'|reverse = {'abc'|reverse}
array( 'a', 'b', 'a', 'c', 'b' )|unique = {array( 'a', 'b', 'a', 'c', 'b' )|unique|implode( ',' )
}
```

Ausgabe:

```
array( 1, 2 )|repeat( 3 ) = 1,2,1,2,1,2
'-'|repeat( 10 ) = ----------
array( 1, 2, 3 )|reverse = 3,2,1
'abc'|reverse = cba
array( 'a', 'b', 'a', 'c', 'b' )|unique = a,b,c
```

Bearbeiten: insert, remove und replace

insert

- Array: fügt jeden Parameter nach dem ersten an der Position `insert_position` ein.
- Zeichenkette: fügt `insert_string` an der Position `insert_position` ein.

Parameter	Typ	Erforderlich	Standardwert
`insert_position`	integer	ja	`false`
`insert_string` (und bei Arrays weitere Elemente)	mixed	ja	`false`

Generierte Referenz: insert.

remove

Entfernt `length` Elemente (oder Zeichen) ab `offset`.

Parameter	Typ	Erforderlich	Standardwert
`offset`	integer	ja	`false`
`length`	integer	nein	1

Generierte Referenz: remove.

replace

Entfernt `length` Elemente (oder Zeichen) bei `offset` und setzt jeden Parameter nach dem zweiten an ihre Stelle (bei einer Zeichenkette diese Parameter aneinandergehängt).

Parameter	Typ	Erforderlich	Standardwert
`offset`	integer	ja	`false`
`length`	integer	nein	`false`
Ersatzelemente	mixed	nein	keiner

Generierte Referenz: replace.

Template

```
{def $list = array( 'a', 'b', 'c', 'd' )}
$list|insert( 2, 'X', 'Y' ) = {$list|insert( 2, 'X', 'Y' )|implode( ',' )}
$list|remove( 1 ) = {$list|remove( 1 )|implode( ',' )}
$list|remove( 1, 2 ) = {$list|remove( 1, 2 )|implode( ',' )}
$list|replace( 1, 2, 'X' ) = {$list|replace( 1, 2, 'X' )|implode( ',' )}
'Exponential'|insert( 3, '-' ) = {'Exponential'|insert( 3, '-' )}
'Exponential'|remove( 0, 3 ) = {'Exponential'|remove( 0, 3 )}
'Exponential'|replace( 0, 3, 'IM' ) = {'Exponential'|replace( 0, 3, 'IM' )}
```

Ausgabe:

```
$list|insert( 2, 'X', 'Y' ) = a,b,X,Y,c,d
$list|remove( 1 ) = a,c,d
$list|remove( 1, 2 ) = a,d
$list|replace( 1, 2, 'X' ) = a,X,d
'Exponential'|insert( 3, '-' ) = Exp-onential
'Exponential'|remove( 0, 3 ) = onential
'Exponential'|replace( 0, 3, 'IM' ) = IMonential
```

array_sum

array_sum

PHP `array_sum()` des gepipten Arrays. Jede andere Eingabe protokolliert "Unknown input type, can only work with arrays". Keine Parameter. Generierte Referenz: array_sum.

Template

```
array( 1, 2, 3.5 )|array_sum = {array( 1, 2, 3.5 )|array_sum}
```

Ausgabe:

```
array( 1, 2, 3.5 )|array_sum = 6.5
```

Veraltete Namen: array_append, array_prepend und array_merge

`lib/eztemplate/classes/eztemplateautoload.php` kennzeichnet `array_prepend`, `array_append` und `array_merge` als "DEPRECATED/OBSOLETE", ersetzt durch `prepend`, `append` und `merge`. Auf Arrays tun sie noch dasselbe wie die neuen Namen:

Template

```
array( 1 )|array_append( 2 ) = {array( 1 )|array_append( 2 )|implode( ',' )}
array( 2 )|array_prepend( 1 ) = {array( 2 )|array_prepend( 1 )|implode( ',' )}
array( 1 )|array_merge( array( 2 ) ) = {array( 1 )|array_merge( array( 2 ) )|implode( ',' )}
```

Ausgabe:

```
array( 1 )|array_append( 2 ) = 1,2
array( 2 )|array_prepend( 1 ) = 1,2
array( 1 )|array_merge( array( 2 ) ) = 1,2
```

Bei einer Zeichenkette als Eingabe kennt der Zeichenketten-Zweig nur die neuen Namen; die alten protokollieren daher "Unknown operatorname" und geben die Eingabe unverändert zurück:

Template

```
'abc'|array_append( 'd' ) = {'abc'|array_append( 'd' )}
```

Ausgabe:

```
'abc'|array_append( 'd' ) = abc
```

Von der Template-Engine protokolliert:

```
WARNING array_append: Unknown operatorname: array_append
```

Das kompilierte Template ist hier nachsichtiger: Kompiliert gab dieselbe Zeile auf einer Entwicklungsinstallation `abcd` aus. Verlassen Sie sich auf keines von beiden; verwenden Sie `append`.

Ausgabe mit dem Template-Compiler (in einen Arbeitsordner kompiliert):

```
'abc'|array_append( 'd' ) = abcd
```

Ein ausgearbeitetes Beispiel: Namen in einer Schleife sammeln

Eine Liste in einer Schleife mit `set` und `append` aufbauen und dann zusammenfügen, an der Kategorie Recipes einer Entwicklungsinstallation (Knoten 144):

Template

```
{def $names = array()}
{foreach fetch( 'content', 'list', hash( 'parent_node_id', 144, 'limit', 3, 'sort_by', array( 'na
me', true() ) ) ) as $child}
{set $names = $names|append( $child.name )}
{/foreach}
Recipes (first three by name): {$names|implode( ', ' )|wash}
```

Ausgabe:

```
Recipes (first three by name): Almond French Toast, Avocado Kale Pesto Grilled Cheese, Gluten-Fre
e Shrimp Pad Thai with Tofu
```

Verwandte Seiten

- Array-Operatoren für Designer: die alltäglichen Verwendungen
- Arithmetische Operatoren: `count` zählt Arrays
- Foreach-Schleifen
- Alle Template-Operatoren (generiert)
- Freie Dokumentation: Diese Seite darf unter der GNU FDL 1.3 frei kopiert, übersetzt und verbessert werden.

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

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.