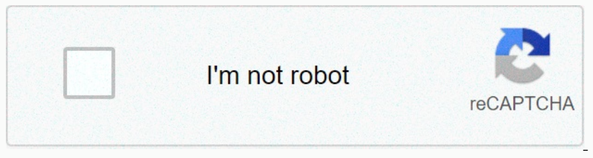

Continue

Shape massive manual

In fact, the array in PHP is an ordered visualization that matches the value and the key. This type is optimized in several ways, so you can use it as an array, a list (vector), a hash table (which is a map implementation), a dictionary, a collection, a stack, a queue, and maybe something else. Because the array can be a different PHP array, you can also create multidimensional trees and arrays. Explaining these data structures is beyond this reference guide, but you'll find at least one example for each. For more information, see the relevant literature on this extensive topic. The array can be created using an array language construct. As parameters, it accepts any number of pairs separated by key > value commas (key > value). array (key > value, key2 > value2, key3 > value3, ...) Comma after the last array element is optional and can be omitted. This is usually done for single-line arrays, i.e. array (1, 2) is preferable to array (1, 2,). Multiline arrays on the other side typically use the final comma because it makes it easier to add new elements to the end of the array. Note: There is a short array syntax that replaces array() with q. An example #1 a simple array of <?php$> array bar > foo/ Using short array syntax,> bar, bar > foo?> key can be int or string value can be of any type. the integer (int) (excluding cases where the number is preceded by the q sign) will be converted to type int. For example, a key with 8 will actually be saved with a value of 8. On the other hand, 08 will not be converted because it is not a correct decimal whole. Floating numbers -to-be will also be converted to type int, which means that the fractional part will be discarded. For example, an 8.7 key will actually be saved with a value of 8. The bool class is also converted to the int style. For example, a true key will be saved with a value of 1 and a fake key will be saved with 0. The null type will be converted to an empty line. For example, a null key will actually be saved with the value . Arrays and objects cannot be used as keys. This will be used to generate a warning: type of illegal scrolling. If multiple elements of the array ad use the same key, only the parent ad will be used and all others will be rewritten. An example #2 An example of type conversion and rewriting < elements?php$array q array (1 > a, 1 > b, 1.5 q > c, true > dvar_dump ($array);; > The result of this example is: array(1) > string (1) d - Since all the keys in the previous example are converted to 1, the value is rewritten for each new element and only the last d remains. Arrays in PHP can contain int and string keys at the same time, because PHP does not distinguish between indexed and associative arrays. Example Mixed keys of type int and string <?php$array (foo > bar, bar > foo, 100 > -100, -100 > 100var_dump ($array);? > The result of this example: array (4) qfoo > string(3) bar > string (3) foo (100) > int (-100) > int (100) key option is optional. If not specified, PHP will use the previous highest int key value, increased by 1. An example #4 indexed arrays without a <?php$array q array (foo, bar, hallo, world);var_dump ($array);; > The result of this example: array (4) > string (3) foo > bar (3) bar (> string(5) hallo > string (5) world world - It is possible to specify the key only for some elements and omit for others: Example #5 Keys for some elements of <?php$q array (a, b, 6 > c dvar_dump ($array);? > The result of this example is: array (4) > string (1) to > string (1) b > string(1) c > string(1) d when you see that the last value d was assigned to key 7. This was because the biggest value of the entire type key before that was 6. Access to array elements can be done with array syntax. An example #6 Access to array elements <?php$array q array (foo > bar 42 > 24, multi > array (dimensional > array (array > foo var_dump ($array'foo);var_dump ($array);var_dump ($array)multidimensionaldimensionaldimensionalarray);massive); > The result of this example: string(3) bar int (24) string (3) foo Note: Both brackets and brackets can be used interchangeably to access the elements of the array (that is, $array and $array{42} are equivalent). An example #7 rename the getArray of <?phpfunction() and the return array (1, 2, $$secondElement 3 secondElement) > Note: Trying to access the array key is the same as trying to access any other uncertain variables: a data-level error will be generated E_NOTICE the result will be null. Note: An array that unifies a scalar value that is not a string will give null. No error message is issued until PHP 7.4.0. Starting with PHP 7.4.0, an error is issued E_NOTICE; PhP 8.0.0 is a very E_WARNING. An existing array can be modified by explicitly setting values in it. In addition, the key can be lowered, resulting in an empty pair of square brackets. $arr [key] - value; $arr[] - value; key can be int or string/ value can be any value of any type If the array $arr not already exists, it will be created. So this is another way to identify the array of arrays. However, this method is not recommended because if the $arr variable already contains some value (such as the string value of the query variable), this value will remain in place and can actually mean access to the symbol on the line. It is best to start the variable by explicitly assigning a value. Note: Starting with PHP 7.1.0, using the empty operator index on the line will result in a fatal error. Previously, in this case, the line was silently converted to an array. To change a particular value, simply assign a new value to the item using its key. If you want to delete a key/value pair, you must use unset(). Note: As mentioned above, if the key is not specified, the maximum of the (int) indexes, and the new key will be this maximum value (as last resort 0) plus 1. If there are no int indexes yet, the key will be 0 (zero). Note that the maximum key value does not necessarily exist in the array at this time. It could have existed simply in the array for a while, as it was last reindexed. Here's the thing: There are plenty of useful features to work with arrays. See the Features for Arrays section. Note: Unset() allows you to delete array keys. Note that the array will NOT be reindexed. If you really want to delete and change the behavior, you can reindex the array using array_values(). The foreach control structure exists specifically for arrays. Makes it easy to walk through the womb. Always comment out a string literal in the associative array index. For example, type $foo' bar, not $foo. But why? Often in older scripts you can find the following syntax: It is wrong, although it works. The reason is that this code contains an untyped constant (bar) instead of a line ('bar' - pay attention to quotation marks). This works because PHP automatically converts a bare line (not a line of quotation marks that does not match any of the known language characters) into a line with the meaning of that bare line. For example, if a constant is not defined with the name bar, PHP will replace the bar with a bar and use it. Attention This backup option to handle an uncertain constant because the blank line is declared obsolete with PHP 7.2.0 and generates a E_WARNING. Previously, a file-level error was issued E_NOTICE. Note: This does not mean that you must always en enclosed in quotation marks. There is no need to cite constants or variables, as this will prevent PHP from processing them. The result of this example: Check 0: Notice: Undefined Index: undefined index: $i in/path/to/script.html on line 9 Bad: Good: 1 Notice: Undefined index: $i in /path/to/script.html on line 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i in /path/to/script.html on line 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i in /path/to/script.html on line 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i path/to/script.html in/path/to/script.html $i in /path/to/script.html on line 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i in /path/to/script.html on line 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i in /path/to/script.html online 11 Bad: Good: 1 Check 1: Notice: Undefined index: $i in <2> <9> online 9 Bad: 2 Notice: indefinite index: $i in/path/to/script.html on line 11 Bad: Well: 2 additional examples demonstrating this fact: If you translate error_reporting into error display mode E_NOTICE (for example, as E_ALL), you will immediately see these errors. By default, you error_reporting installed to display them. As indicated in the syntax section, there must be an expression within the brackets ('y's q. This means that you can write like this: This is an example of using a value return function as an array index. PHP is also known for constants: Note that E_ERROR is as true to an identifier as the bar in the first example. But the last example is essentially equivalent to this record: because E_ERROR corresponds to 1, etc. For example, you can no longer use empty and default words in this way because they are reserved keywords. Note: Repeat, within the chain enclosed in double quotation marks, it is correct not to encirify the indexes of the array with quotation marks, so $foo is the correct record. For more information, see the examples above, as well as the line processing section for variables. For any of the int, float, string, bool, and resource types, converting the value to an array becomes an array with an element (with index 0) as the rock value with which it started. In other words, (array) $scalarValue is exactly the same as array ($scalarValue). If you convert an object to an array, you receive the properties of the object (variable member) as elements. The keys will be the names of the member variables, with some notable exceptions: the inte celsial properties will no longer be available; The class name will be added to the closed private fields in the front. a 'z' symbol will be added to the protected fields of the classroom in front. These values added on both sides also have zero bytes. This can cause some unexpected behavior: The above code will display 2 keys with the name 'AA', although one of them actually has the name "0A'0A'. If you convert to a null array, you get an empty array. The array type in PHP is very flexible, here are some examples: Example #8 Using Array() An example #9 Collection <?php$colors q array ('red', 'blue', 'green', 'yellow'$colors as $color) q echo Do you like $color?? > The result of this example: Do you like red? Do you like blue? Do you like green? Do you like yellow? Changing the array values directly is possible by transmitting them by clicking on the link. An example #10 Change an element in the <?phpforeach cycle ($colors as $color) is $color strtoupper ($color$color); / Is it necessary to prevent subsequent inputs $color the last element of the q/print_r array ($colors);; > The result of this example: Array (> RED (> RED) (> BLUE q2) > GREEN (> YELLOW) The following example creates an array that begins with a unit. An example #11 Index from a unit of <?php$firstquarter q array (1 q > 'January', 'February', 'March');p rint'r ($firstquarter);? > The result of this example: Array (> 'January' (> 'February' (> 'March') An example #12 Fill array <?php//fill all elements of the opendir $handle directory ('')); While (false !) ($file q readdir ($handle > $handle $fileclosedir $files) You can calculate the number of elements in the array using the count(#14 > $files ;p $files < #13). Обратите внимание, что при присваивании массива всегда происходит копирование значения. Чтобы скопировать массив по ссылке, вам нужно использовать оператор ссылки. thomas tulinsky , I think your first main example is unnecessarily confusing, very confusing for novices:$array - array(foo > bar, bar > foo,);D ebe be eliminated. For newbies:An array index can be any string value, even a value that is also a value in the array. The value of array[foo] is bar. The value of array[bar] is fooThe following expressions are both true:$array[foo] a bar$array[bar] á foo jeff splat codedread splot com , note that if you are using strings as indexes in the $_POST array, these periods are transformed into underscores:<html><body><?php ? printf(post _post print_r:? ><form method-post action-<?php echo $_SERVER['PHP_SELF']; ?>> <input type-hidden name-Windows3.1 value-Sux> <input type-submit value-Click></form></body></html>Once you click the button, the page displays the following:POST: Array ([Windows3_1] > Sux) free in gmail dot com - Since PHP 7.1, the string will not be converted to array automatically. The codes below will fail:$a-array();$a['a'];$a['a']['b']; Warning: Illegal string offset 'b'//Warning: Cannot assign an empty string to a string offsetBe must switch to the following: $a-array();$a['a']-array(); Declare that it is a first$a['a']['b']" array; chris in ocportal dot com - Note that array value cubes are safe for references, even through serialization. <?php $x'initial' ;' $test'array('A'?>&$x,'B'">&$x); $test-unserialize(serialize($test)); $test['A']'changed'; echo $test['B']; ?> This can be useful in some cases, for example, saving RAM within complex structures. ken underscore yap atsign email dot com If you convert a NULL value into an array, you get an empty array. This turns out to be a useful property. Suppose you have a lookup function that returns an array of values in success or NULL if nothing is found. <?php $values?search(...);? > Now you want to combine the array with another array. What do we do if $values is NULL? No problem: <?php $combined?array_merge((array)$values, $other);?> Voila. Yesterday php'er --- quote ---Note:Both brackets and braces can be used interchangeably to access array elements--- quote end ---To less for php 5.4 and 5.6; if the function returns an array, the brackets do not work directly by accessing the result of the function, for example. WillReturnArray(){1} . This will give syntax error, unexpected ' in.... Personally I use only square brackets, I wait to access a single char chain. Habits... anisgazig in gmail dot com //array keys are always y los valores de tipo de datos y de matriz son todos los tipos de datos//tipo de conversión y sobrescritura (tipo de datos de clave de > matriz)//---$arr - array(1->a,//int(1)33>8080> d,//int(80)0->e,//int(0)Hellow->f,//string(6)Hellow10Hellow->h,//string(8)10Hellow1.5->j,/int(1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.5.5.1.5.6.6.1.1.1.1.1.>j5)1.5->k,//string(3)1.50.0->l,/int(0)false->m,/int(false)true->n,//int(true)true>o,//string(4)truetruefalse->p,//string(5)falsenull->q,//string(0)NULL->r,//string(0) note null y NULL son los mismos NULL ,,, > /compruebe el nombre del tipo de datos de keyforeach ($arr como $key > $value) .var_dump($key);echo
;} NOte :array and the object data type in the keys are Illegal ofset....... ia [AT] zoznam [DOT] sk - As for the previous comment, consider the fact that the reference to the last value of the array remains stored in $value after the input: <?phpforeach (?$arr?as?$key?> &$value $value 1$value at 159;? >Now the last item in $arr has a value of '159'. If we delete the comment on the unset() line, everything works as expected ($arr has all the values of '1'). Bad results can also appear in nested foreach loops (the same reason as the previous one). So, either unset $value after each foreach or better to use the longer form: <?phpforeach ('$arr's' as' $key?> $value $arr[$key] to 1?> lars-phpcomments at ukmix dot net ? Used to create arrays like this in Perl?@array, A.Z); It seems that we need the range() function in PHP: <?php$array ? array_merge(array('all'), ? range('a','?'z'));? ??> It is not necessary array_merge if it is only a range: <?php$array ? range('a', ?z');? ?> caifara aaaat im dooaat be [Editor's Note: You can achieve what you're looking for by referencing $single, instead of copying it by value to your foreach statement. See for details.] I don't know if this is known or not, but he ate some of my time and maybe he won't eat his time now... I tried to add something to a multidimensional array, but that didn't work at first, look at the code below to see what I mean: array(a á > 0, b - > 1); $a 2 - array(aa -> 00, bb > 11 $together - array($a 1, $a 2 $together as $single) to $single[c] to 3 print_r($together); foreach($together as $key > $value) - $together[$key][c] - 3 print_r ($together ?>fj700 in gmail dot com / Before php 5.4$array á array(1,2,3);// from php 5.4 , short syntax$array a [1,2,3];// I recommend using the short syntax if you have the php version > 5.4 tiss áus in building key is also an expression. This works well: $a matrix (1 x >0, 1 + 1 to >1, $k >2, $x.'4'->3); Anonymous containers for (array), returns the array with the normalization keys (without (without to_array_recursive($valueis_object($value)) á return (array) $value$class á get_class($value); $arr []; foreact ((array) $value such as $key > $val), $key , str_replace([-0*-0, -0-$class-0], '', $key); $arr[$key] $val is_object? to_array_recursive($val) : valarrto_array($value$arr á (array) $value; if (! is_object($value)) - return arrclass á get_class($value); $keys of str_replace([-0*-0, -0-$class-0], '', array_keys($arr)); return array_combine($keys, $arr?>Demo:<?phpclass TestTM protected $var 1; protected $var 2; $TestVar to 3__construct($isParent ? true$isParent) ? $this->var2 - new self(! $isParent$obj á new Test();var_dump(array) $obj, to_array_recursive($obj));? > Denise Ignatova when creating arrays, if we have an element with the same value as another element in the same array, we would expect PHP instead of creating a new zval container to increase the refcount and point the duplicate symbol to the same zval. This is true except for the integer value type. Example:$arr ? ['baby' ?> 'Bob', 'age' ?> 23, 'too' ?> 23];xdebug_debug_zval('arr');Output:arr:(refcount-2, is_ref-0)array (size-3) 'baby' ?> (refcount-1, is_ref-0)string 'Bob' (length-3) 'age' -> (refcount-0, is_ref-0)int 23 'too' -> (refcount-0, is_ref-0)int 23but :$arr - ['baby' ?> 'Bob', 'age' ?> , 'too' ?> '23'];xdebug_debug_zval('arr');will produce:arr:(refcount-2, is_ref-0)array (size-3) 'baby' ?> (refcount-1, is_ref-0)string 'Bob' (length-3) 'age' -> (refcount-0, is_ref-0)int 23 'too' ?> (refcount-1, is_ref-0)string '23' (length-2)or :$arr-['baby' ?> 'Bob', 'age' ?> [1.2], 'too' ?> [1.2]];xdebug_debug_zval('arr'); :arr:(refcount-2, is_ref-0)array (size-3) 'baby' -> (refcount-1 , is_ref 0)string 'Bob' (length 3) 'age' > (refcount-2, matrix is_ref-0 (size 2) 0 to > (refcount-0, is_ref-0)int 1 1 1 > (refcount-0, is_ref-0)int 2 'too much' > (refcount-2, matrix is_ref-0 (size 2) 0 to > (refcount-0, is_ref-0)int 1 1 a > (refcount-0 , is_ref-00)int 2 note dot php dot lorriman in spamgourmet dot org - There is another type of matrix (php> 5.3.0) produced by $array - new SplFixedArray(5); Standard arrays, as documented here, are wonderfully flexible and, because of the underlying hash table, extremely fast for certain types of lookup operation. Suppose a large array with string key is $arr ['string1">$data 1, 'string2'->$data 2 etc....] when getting the keyed data with $data-$arr['string1'];p hp not *no* you have to search through the array comparing each key string to the given key ('string1') one by one, which could take a long time with a large array. Instead, the hash table means that php takes the given key and calculates the memory location of the keyed data from it, and then instantly retrieves the data. Wonderful! And so fast. And there's no need to know anything about hash tables, as everything is hidden. However, there is a lot of overhead in that. It uses a lot of memory, because hash tables tend to (also nearly double on a 64-bit server), and must be significantly slower for integer key arrays than old (non-hashtable) integer key arrays. To see more in SplFixedArray : a standard php array (hashtabled), if you search entirely then the integer itself denotes the memory location of the data, without hash table calculation on the required integer key. This is much faster. It is also faster to create the array compared to the complex operations required for hash tables. And it uses much less memory, as there is no hashtable data structure. This is actually an optimization decision, but in some cases large arrays with integer key can significantly reduce server memory and increase performance (including avoiding costly memory deallocations of hashtable arrays when exiting the script). ivegner in yandex dot ru - Note that objects in classes that extend the ArrayObject SPL class are treated as arrays, and not as objects when converting to array.<?phpclass ArrayObjectExtended extends ArrayObject, private $private , 'private'; $hello to 'world'$object - new ArrayObjectExtended();$array á (array) $objectvar_export($array);? > Walter Tross is true that assigning fixes always involves copying value, but copying is a lazy copy. This means that the data in the two variables takes up the same memory as long as no array element changes.E.g., if you have to pass an array to a function that only needs to read it, there is no advantage in passing it by reference. gfr7oo at the point of gmail com <?phpfunction getArray() - return array(1, 2, 3$secondElement - getArray() [1$tmp á getArray();$secondElement ? $tmp[1list(, $secondElement) ? getArray();? >Here in the list(, $secondElement) - getArray(); and list($secondElement) - getArray(); is the same thing you can avoid the comma to avoid confusion. Confusion.