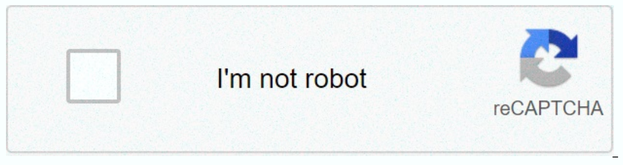

Continue

Pass string array from android to php

In this tutorial, I will make a list of common PHP array features, with examples of usage and best practices. Every PHP developer needs to know how to use them and how to combine array features to make the code readable and short. There's also a presentation with given code samples so you can download it from the related links and show it to your colleagues to build a stronger team. Basically there are two different ways to create arrays. One is to use matrix() to set the items as key value pairs. The second method is to put all items inside []. There are two important points that you should remember when creating associated arrays with key pairs. First, the key must always be unique. If you try to use the same key multiple times in an array, PHP will ignore all other key value pairs except the last one. Second, if a key is created as floats, bools, and valid string representations of integers, then it will be cast into integers. Here are a few examples of creating arrays in PHP: $first = array(10, Apple, 20, -18, Monkey); print_r($first) /* Array ([0] => 10 [1] => Apple [2] => 20 [3] => -18 [4] => Monkey) */ $second = [10, Apple, 20, -18, Monkey]; print_r($second) /* Array ([0] => 10 [1] => Apple [2] => 20 [3] => -18 [4] => Monkey) */ $third = [10, 5 => Apple, 2 => 20, -18, Monkey]; print_r($third) /* Array ([0] => 10 [5] => Apple [2] => 20 [6] => -18 [7] => Monkey) */ As you can see, using either matrix() or [] when creating arrays is equivalent. The shorthand notation has been available from PHP 5.4. You also do not need to enter a key for each array value. Once left out, php puts the key to a more than the largest specified integer key. All automatically assigned keys will be greater than or equal to 0. Work with keys and values Let's start with the basic functions that work with array keys and values. One of them is array_combine(), which creates an array using a key matrix and another for its values: $keys = ['cloud', 'grass', 'orange']] $values = ['blue', 'green', 'orange']; $array = array_combine($keys, $values) print_r($array) Array // (// [sky] => blue // [grass] => green // [orange] => orange //) You should know that the function array_values() returns an indexed array of values, array_keys() returns a number of keys in a given array, and array_flip() exchanges keys with values: print_r(array_keys($array)); ['sky', 'grass', 'orange', print_r(array_values($array)) ['blue', 'green', 'orange', print_r(array_flip,$array)) Array // (// [blue] => sky // [green] => grass // [orange] => orange //) You can check whether an array contains a specific value and get the first corresponding key using the array_search() function. You can also use If you just want to know if an array contains a specific item and is not interested in its location. Location. using array_key_exists() function when you want to check if the array is using a given key. $values = [Apples, Bananas, Mango, 100, 200]; if (in_array(100, $values)) { echo '100 is one of the values'. } // 100 is one of the values if (in_array(200, $values) !== false) { echo '200 is not one of the values'; } // 200 is not one of the values $values = [Apples, Bananas, Mango, 100, 200, 100]; echo array_search(100, $values) 3 echo array_search(100, $values, true); 5 $values = [Apples => 100, Bananas => 10, Mango => 45]; if (array_key_exists(Apples, $values)) { echo 'We have apples.'; } // We have apples. As the example above shows, be sure to use strict type control if you don't want unexpected results. If you want to look up multiple items in an array, it's usually faster to check if it contains a specific value by first reversing the array with array_flip(), and then using array_key_exists(). Make your code shorter The function() is displayed, which is not really a function, but a language construct, designed to assign variables in a short way. For example, here is a basic example of using the list function(): // define array $array = ['a', 'b', 'c']; without list() $a = $array[0]; $b = $array[1]; $c = $array[2]; list(s) list($a, $b, $c) = $array This construction works perfectly with functions such as preg_slit() or explode() . You can also skip some parameters if you don't need them to be defined: $string = 'hello|wild|world'; list($hello, $world) = explode('|', $string). echo($hello, $world) hello, world Also list() can be used with foreach, which makes this construction even better: $arrays = [[1, 2], [3, 4], [5, 6]]; foreach ($arrays as list($a, $b)) { $c = $a + $b; echo($c . ', '); 3, 7, 11, } With the extract function() you can export an associative matrix to variables. For each element of an array, a variable is created with the name of a key and value as a value of the item: $array = ['clothing' => 't-shirt', 'size' => 'medium', 'color' => 'blue',]; extract($array) echo($clothes $size $color) t-shirt medium blue Be aware that extract() is not safe if you are working with user data (as results of requests), so it is better to use this feature with the flags EXTR_IF_EXISTS and EXTR_PREFIX_ALL. The opposite of the previous function is the compact() function, which makes an associative array from variables: $clothes = 't-shirt'; $size = 'medium'; $color = 'blue'; $array = compact('clothing', 'size', 'color'); print_r($array) Array // (// [clothing] => t-shirt // [size] => medium // [color] => blue //) Filtering features There is a great feature for array filtering, and it is called array_filter(). Pass array as the first param and an anonymous function as the second param. Return sand in a callback function if you want to this element of the matrix and false if you do not: $numbers = [20, -3, 50, -99, 55]; $positive = array_filter($numbers, function($number) { return $number > 0; }); print_r($positive) [0 => 20, 2 => 50, 4 => 55] There is a way to filter not only the values. You can use ARRAY_FILTER_USE_KEY or ARRAY_FILTER_USE_BOTH as a third parameter to transfer the key or both the value and key to the callback function. You can also call array_filter() without callback to remove all empty values: $numbers = [-1, 0, 1]; $not_empty = array_filter($numbers) print_r($not_empty) [0 => -1, 2 => 1] You can only retrieve unique values from an array by using the array_unique() function. Note that the function retains the keys to the first unique elements: $array = [1, 1, 1, 1, 2, 2, 2, 3, 4, 5, 5]; $uniques = array_unique($array) print_r($uniques) Matrix // (// [0] => 1 // [4] => 2 // [7] => 3 // [8] => 4 // [9] => 5 //) With array_column(), you can obtain a list of values from a multidimensional system a response from a SQL database or an import from a CSV file. Just pass an array and column name: $array = [['id' = > 1, 'title' = > 'tree'], ['id' => 2, 'title' = > 'sun'], ['id' = > 3, 'title' = > 'cloud'],]; $ids = array_column($array, 'id') print_r($ids) [1, 2, 3] Starting from PHP 7, array_column() becomes even more powerful because it is now allowed to work with a number of objects. Then working with a number of models just got easier: $cinemas = Cinema::find ()->all(); $cinema_ids = array_column($cinemas, 'id'); php7 forever! When you go through the systems using array_map() you can apply a callback to all elements of an array. You can pass a function name or an anonymous function to get a new array based on the given array: $cities = ['Berlin', 'KIEV', 'Amsterdam', 'Riga']] $aliases = array_map('strtolower', $cities) print_r($aliases) ['berlin', 'kyiv, 'amsterdam', 'riga'] $numbers = [1, -2, 3, -4, 5]; $squares = array_map(function($number) { return $number ** 2; }, $numbers) print_r($squares) [1, 4, 9, 16, 25] There is a myth that there is no way to transfer values and keys in an array to a callback, but we can bust it: $model = ['id' = > 7, 'name'=>'James']; $callback = function($key, $value) { return $key is $value; }; }; $res = array_map($callback, array_keys($model), $model) print_r($res) Array // (// [0] => id is 7 // [1] => name is James //) But it looks dirty. It is better to use array_walk () instead. This feature looks like array_map(), but it works differently. First of all, an array is transferred using a reference so that array_walk() does not create a new array, but changes a given array. So, as a source matrix, you can transfer the array value by using a reference in a callback. Array keys can be passed easily: $fruits = ['banana' = 'yellow', 'apple' = 'green', 'orange' = 'orange', 'orange',]orange',];]; function(&$value, $key) { $value = $key is $value; }); print_r($fruits) Array // (// [banana] => banana is yellow // [apple] => apple is green // [orange] => orange is orange //) Connecting Arrays The best way to merge two or more arrays in PHP is to use array_merge() function. Arrays elements are merged and values with the same string keys are overwritten with the last value: $array 1 = ['a' => 'a', 'b' => 'b', 'c' => 'c']] $array 2 = ['a' => 'A', 'b' => 'B', 'D' => 'D']; $merge = array_merge($array 1, $array 2) print_r($merge) Matrix // (// [a] => A // [b] => B // [c] => c // [D] => D //) To remove array values from another matrix (or matrix), use array_diff(). To retrieve values that are in given arrays, array_intersect(). The next examples show how it works: $array 1 = [1, 2, 3, 4]; $array 2 = [3, 4, 5, 6]; $diff = array_diff($array 1, $array 2) print_r($diff) [0 => 1, 1 => 2] $intersect = array_intersect($array 1, $array 2) print_r($intersect) [2 => 3, 3 => 4] Use the mathematics with array values using array_sum() to obtain a sum of array values, array_product() to multiply them or create your own formula with array_reduce(): $numbers = [1, 2, 3, 4, 5]; echo(array_sum($numbers)) 15 echo(array_product($numbers)), 120 echo(array_reduce($numbers, function($carry, $item) { return $carry ? $carry / $item : 1; })); 0.0083 = 1/2/3/4/5 To count all the values in an array, array_count_values(). It will give all unique values for a given matrix as keys and a count of these values as a value: $things = ['apple', 'apple', 'banana', 'tree', 'tree', 'tree']] $values = array_count_values($things) print_r($values) Matrix // (// [apple] => 2 // [banana] => 1 // [tree] => 3 //) Generating arrays For generating an array of a given size and the same value, use array_fill(): $bind = array_fill(0, 5, '?'); print_r($bind) // ['?', '?', '?', '?', '?', '?'] To generate an array of keys and values $letters, such as print_r($letters) ['a', 'b', ..., 'z'] $hours = range(0, 23) print_r($hours) [0, 1, 2, ..., 23] To get part of an array – for example, only the first three elements – use array_slice(): $numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; $top = array_slice($numbers, 0, 3) print_r($top) [1, 2, 3] If you ever want to generate an associative array with different keys and the same value assigned to each key, you can simply use the array_fill_keys function() . $keys = [Apples, Bananas, Mango]; $fruit_count = array_fill_keys($keys, 100); print_r($fruit_count) /* Array ([Apples] = > 100 [Bananas] = > 100 [Mango] = > 100) */ Sorting Arrays It is good to remember that each sorting function in PHP works with arrays of a reference and returns about success or false on failure. There is a basic sorting function sorting function and it sorts values in ascending order without preserving keys. The sorting function can be preceded by the following letters: sorting the conserving keys k, sorting by keys r, sorting in reverse/descending order u, sorting with a user function You can see the combinations of these letters in the following table: Combine matrix functions like a boss The real magic begins when you start combining matrix functions. Here you can trim and remove empty values in just a single line of code with array_filter() and array_map(): $values = ['say', 'bye', ' ', ' to', ' spaces ', ' ']]; $words = array_filter(array_map(trim$values)) print_r($words) ['say', 'bye', 'two', 'spaces'] To create an ID for a title card from a number of models, we can use a combination of array_combine() and array_column(): $models = [$model 1, $model 2, $model 3]; $id_to_title = array_combine(array_column($models, 'id'), array_column($models, 'title')) To get the top three values in a matrix, we can use array_count_values(), scar variety() and array_slice(): $letters = ['a', 'a', 'a', 'a', 'b', 'b', 'c', 'd', 'd', 'd', 'd', 'd', 'd', 'd']; $values = array_count_values($letters) get the key to count array scarsort ($values); sort the descending conserving $top = array_slice($values, 0, 3) get top 3 print_r ($top); Matrix // (// [d] => 5 // [a] => 4 // [b] => 2 //) It is easy to use array_sum() and array_map() to calculate the sum of the order in a few rows: $order = [['product_id' => 1, 'price' => 99, 'count' => 1], ['product_id' => 2, 'price' => 50, 'count' => 2], ['product_id' => 2, 'price' => 17, 'count' => 3],]; $sum = array_sum(array_map(function($product_row) { return $product_row['price'] * $product_row['count']; }, $order)); print_r($sum) 250 Conclusion As you can see, knowledge of the main array features can make your code much shorter and more readable. Of course php has many more array features, and even the given features have many variations to use with extra parameters and flags, but I think that in this tutorial we have covered the basics that every PHP developer should know. Learn PHP With a free Online Course If you want to learn PHP, check out our free online course on PHP fundamentals! In this course, you will learn the basics of PHP programming. You start with the basics, learn how PHP works and write simple PHP loops and functions. Then you will build up to coding classes for simple object-oriented programming (OOP). Along the way, learn all the key skills for writing apps for the Web: you'll have the opportunity to practice responding to GET and POST requests, analyze JSON, authenticate users, and use a MySQL database. Database.