Цель домашнего задания

Научиться создавать две вспомогательные функции, которые пригодятся для разработки сайта:

* функция для обрезки строки на заданную параметром длину;
* функция для сортировки массива по определённому полю этого массива.

Что входит в задание

1. Задание на создание вспомогательной функции для обрезки строки.
2. Задание на создание вспомогательной функции для сортировки массива по определённому полю.

Задание 1. Создаём вспомогательную функцию cutString

Что нужно сделать

Создайте вспомогательную функцию для обрезки строки:

function cutString($line, $length = 12, $appends = '...'): string  
  
{  
  
}

Эта функция принимает три параметра:

* $line — исходная строка;
* $length — максимальная длина строки, по достижении которой происходит её обрезание;
* $appends — символы, которые следует добавить вместо отрезанной части строки.

Функция должна обрезать строку, если она длиннее, чем параметр $length, а вместо отрезанной части добавить $appends и вернуть полученную строку. Если же строка короче, чем $length, то функция ничего не должна с ней делать — она просто возвращает исходную строку.

Функция должна быть создана в отдельном файле и подключена в приложенном файле 001\_strings.php.

Обработайте этой функцией все строки в приложенном массиве и сформируйте новый массив из полученных строк. Обрежьте все строки длиннее четырнадцати символов, в конце добавьте три точки. Полученный массив выведите с помощью функции var\_dump();

Советы и рекомендации

* Старайтесь избегать работы со входящими параметрами по ссылкам в функциях — это создаёт скрытые места изменения исходных значений и не позволяет создавать чистые функции.

// пример  
$a = 3;  
  
function plusOne(&$value)  
{  
    return ++$value;  
}  
  
echo plusOne($a); // выведен 4  
echo $a; // теперь тоже выведет 4

* Функция cutString задумана как чистая функция, то есть результат её работы зависит только от входных параметров. Эта функция не подключает никакие файлы и не меняет внешние переменные.
* Если вы хотите работать со строками, содержащими кириллицу, используйте функции для работы с многобайтными строками mb\_\*.

Задание 2. Создаём вспомогательную функцию arraySort

Что нужно сделать

Создайте вспомогательную функцию для сортировки многомерного ассоциативного массива arraySort по определенному ключу:

function arraySort(array $array, $key = 'sort', $sort = SORT\_ASC): array  
{  
}

Эта функция принимает три параметра:

* $array — входной массив для сортировки;
* $key — ключ элементов этого массива, по значениям которых будет осуществлена сортировка;
* $sort — направление сортировки по возрастанию или убыванию (константы: SORT\_ASC/SORT\_DESC).

Эта функция должна отсортировать элементы переданного массива, по указанному ключу в указанном направлении и вернуть отсортированный массив.

Функция должна быть создана в отдельном файле и подключена в приложенном файле **002\_array\_to\_sort.php**.

Отсортируйте массив пунктов меню на сайте, который уже заведён в приложенном файле, по возрастанию значения поля sort элементов этого массива. Выведите результирующий массив с помощью функции var\_dump();

Советы и рекомендации

* Для сортировки массивов особым образом используйте функцию usort.
* Не используйте работу с данными по ссылке в функции, хотя usort и использует такой подход. Ваша функция не должна так делать.
* Функция **arraySort** задумана как чистая функция, то есть результат её работы зависит только от входных параметров. Эта функция не подключает никакие файлы и не меняет внешние переменные.

Что оценивается

* Функция содержит только три указанных параметра.
* Функция является чистой.
* Функция выполняет требуемую логику и возвращает корректный результат работы.
* Стили оформления psr-1 и psr-12 соблюдены.

Как отправить задание на проверку

Каждую функцию реализуйте в отдельных файлах, примените функции в соответствующих файлах *001\_strings.php, 002\_array\_to\_sort.php*. Задание сдаётся в виде архива, содержащего оба файла.