**Тестовое задание (junior)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Тестовое задание для Junior разработчика**

## JavaScript:

### **Задание 1:**

При загрузке страницы на JavaScript создать три новых блока и добавить их на страницу. В каждый блок добавить новый элемент параграфа с текстом “popup”. Создать событие каждому блоку: при клике менять его цвет.

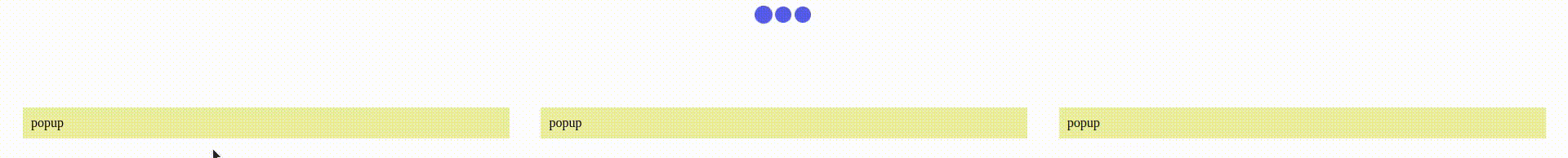
### **Задание 2:**

Приведенный ниже код должен из полученного объекта сформировать URL с GET параметрами. Задача - исправить ошибки в коде.

1. ***function*** *setGetParams(newGetparams) {*
2. *let newString*
3. ***for****(key* ***in*** *newGetParams)*
4. *newString += (newString ? '&' : '') + key + '=' + newGetParams[i]*
5. ***return*** *window.location.href + '?' + newString*
6. *}*

## CSS:

### **Задание 3:**

Создать похожую страницу и сделать похожие анимации. Не использовать flex и grid.  
Пример реализации:

## PHP:

### **Задание 4:**

Создать класс, наследуемый от **ParentClass** и реализующий интерфейс **InterfaceOne**. Класс **ParentClass** и \*\*\*\*интерфейс **InterfaceOne** создавать не требуется.

В созданном классе реализовать **шаблон “одиночка”**

### **Задание 5:**

Оптимизировать код:

1. *global $items;*
2. *var\_export(test($items));*
3. ***function*** *test($items) {*
4. *$sites = [];*
5. *$result = [];*
6. ***foreach****($items* ***as*** *$item) {*
7. *preg\_match('/(site1\.ru)/', $item->site, $matches);*
8. ***if*** *($matches) {*
9. *$sites[count($sites)] = 'site1.ru';*
10. *}* ***else*** *{*
11. *$sites[count($sites)] = 'site2.ru';*
12. *}*
13. ***if****($result < count($sites)) {*
14. *$result[count($result)] = count($sites);*
15. *}*
16. *}*
17. *testTwo($result);*
18. ***return*** *$sites;*
19. *}*
20. ***function*** *testTwo($arr) {*
21. *$newArr = [];*
22. ***foreach****($arr* ***as*** *$itemRes) {*
23. ***if****($itemRes > 0) $newArr = $itemRes;*
24. *}*
25. *}*

### 

### **Задание 6:**

Исправить ошибки в коде и оптимизировать его:

1. ***global*** *$item;*
2. *getMinPrice();*
3. ***function*** *getMinPrice($item) {*
4. ***return*** *formatprice(*
5. *usort($item->prices,* ***function****($a, $b) {*
6. ***return*** *$a->price - $b->price*
7. *})[0]->price*
8. *)*
9. *}*
10. ***function*** *formatPrice($price) {*
11. ***return*** *preg\_replace('/\B(?=(\d{3})+(?!\d))/', ',',$price)*
12. *}*

### **Задание 7:**

Создать метод, принимающий целые числа **$minPrice**, **$maxPrice** и массив **$items** вида:

1. *$items = [*
2. *0 => objectItem,*
3. *…*
4. *n => objectItem,*
5. *]*

где **objectItem** бывает двух видов:

с одной ценой:

1. *(object)[*
2. *'title' => string,*
3. *'price' => int,*
4. *'size' => string*
5. *]*

с несколькими ценами:

1. *(object)[*
2. *'title' => string,*
3. *'prices' => (****array****)[*
4. *0 => (object)[*
5. *'price' => int,*
6. *'size' => string*
7. *]*
8. *…*
9. *n => (object)[*
10. *'price' => int,*
11. *'size' => string*
12. *]*
13. *]*
14. *]*

Метод должен уметь обрабатывать оба вида.

В процессе обработки метод должен отфильтровать из принятого массива **$items** элементы, у которых **цена** больше $**minPrice** и меньше **$maxPrice** (если цен несколько: **первая цена** больше **$minPrice** и **последняя цена** меньше **$maxPrice**).

Вернуть массив вида:

1. *(****array****) [*
2. *key => (****array****) [*
3. *0 => objectItem,*
4. *…*
5. *n => objectItem,*
6. *],*
7. *…*
8. *key => (****array****) [*
9. *0 => objectItem,*
10. *…*
11. *n => objectItem,*
12. *],*
13. *]*

где **key** это поле **price** объекта **objectItem** (если цен несколько, то брать **price** из первого элемента поля **prices**).   
Возвращаемый массив должен быть отсортирован в порядке возрастания.

## MySQL:

Имеются 3 таблицы:

Таблица пользователей users:

| **Id** | **object\_id** | **email** | **password** |
| --- | --- | --- | --- |
| 1 | 2 | name@example.com | d7afde3e7059cd0a0fe09eec4b0008cd |
| 2 | 4 | name@example.ru | a93ef3a7b083e2d631f6f4d1c9053549 |
| 3 | 5 | name@example.org | ec0405c5aef93e771cd80e0db180b88b |
| 4 | 6 | name@example.ca | 1fdda3f0c5555b8b3057369ba3c58215 |
| 5 | 7 | name@example.ua | 512445d4706c301fb1bebae8ff647754 |
| 6 | 8 | name@example.so | 0962e1aed115aeec2ecc398ba3df40b9 |

Таблица objects:

| Id | status\_id | name | last\_login | created |
| --- | --- | --- | --- | --- |
| 1 | 0 | Иван | 2022-01-01 | 2021-01-01 |
| 2 | 0 | Петр | 2022-03-13 | 2022-02-01 |
| 3 | 1 | Кристина | 2022-02-09 | 2022-02-09 |
| 4 | 0 | Андрей | 2010-10-12 | 2007-10-10 |
| 5 | 1 | Федор | 2018-08-08 | 2007-03-01 |
| 6 | 1 | Виктор | 2022-08-25 | 2022-02-12 |
| 7 | 1 | Анастасия | 2022-08-25 | 2022-02-18 |
| 8 | 0 | Владимир | 2022-08-24 | 2022-02-23 |

Таблица status:

| Id | status |
| --- | --- |
| 0 | Оффлайн |
| 1 | Онлайн |

### **Задание 8:**

Составить запрос, который выводит пользователей, статус которых - онлайн и дата создания которых позже 2020-01-01. Результат должен иметь id пользователя, его имя, email, пароль, статус, дату последнего посещения и дату создания.

### **Задание 9:**

Составить запрос, который поменяет у оффлайн пользователей статус на онлайн и дату последнего посещения на 2022-08-25.