

Управление состоянием Vue-компонентов. Директивы условной компиляции и отрисовки списка.

Изучите директивы условной компиляции [1] и директивы отрисовки списка [2]. Обратите внимание на роль атрибута `key` при отрисовке списков посредством директивы `v-for`. Очень рекомендуем прочитать статью [3], в которой даны понятные объяснения и советы по использованию атрибута `v-for`.

Изучите способы регистрации и переиспользования SFC-компонентов [4], а также передачу реактивных состояний в эти компоненты с помощью пропсов [5].

В заданиях Вам понадобится эмиттировать события из SFC-компонента, для этого рекомендуется ознакомиться с разделом [6].

Вопросы и задания

1. Назовите минимум два отличия работы директив `v-if` и `v-show`. Предложите для каждого из следующих случаев выбор необходимой директивы и обоснуйте свой выбор:
 - карточки в игре на поиск соответствий, где при клике на карточку показывается изображение и исчезает через секунду;
 - переключение вида отображения элементов с сетки на список и наоборот;
 - показать кнопку панели администратора в зависимости от роли пользователя, выбранной при авторизации;
 - показать или скрыть Vue-компонент, хранящий некоторое заданное (отличное от дефолтного) состояние, сохранив это состояние.
2. Предположим Вам необходимо отрисовать список текстов, находящихся в массиве `texts = ["text1", "text2", ...]`, при помощи тэгов `<р>`. Однако, допустим, нам требуется вывести не каждый текст, а только те, что имеют четное число букв. Предложите два способа реализации этой задачи на Vue. Какой способ более оптимальный для очень большого неизменяемого списка и почему? Тот же вопрос

для небольшого списка. Тот же вопрос для большого и небольшого списка в случае его частых мутаций.

3. Для чего требуется атрибут `key` при работе со списком? Предоставьте фрагмент кода на Vue в качестве примера (придумайте сами), демонстрирующий проблему обновления состояния вложенных компонентов для элементов некоторого списка.
4. Продолжите работу над приложением `notes-app`. Работать с фреймворком сейчас и в последующем будем в стиле «Composition API». Добавьте состояния и обработчики событий для строки поиска (например, `searchQuery = ref(“”)`), для переключателей вида списка и т. д.
5. Создайте SFC-компонент заметки согласно дизайну [8]. Поместите его в директорию `/src/views/notes/components/note`. Добавьте все необходимые состояния, пропсы и обработчики событий. Для кнопок меню заметки (редактирование, удаление) пока можете сделать только эмиты событий из компонента. Для реализации закрытия меню взаимодействия с заметкой при клике на область мимо самого меню, может оказаться полезной статья про `click outside` [7].
6. Выведите список заметок на основную страницу (компонент `/src/views/notes.vue`). Для получения данных списка сделайте временно реф на массив объектов следующего вида (должен быть валидный JSON формат):

```
{
  id:      Number,      (уникальный в массиве)
  title:   String,      (не может быть пустой строкой)
  text:    String,      (может быть пустой строкой)
  image:   String | null, (ссылка или null)
  date:    String       (ISO - формат)
}
```

Придумайте порядка 20-30 объектов заметок с различным наполнением. Пример:

```
[
  {
    id: 0,
```


[D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%B0?node-id=0-1&p=f&t=bOSL2DfIJq0mNLAH-0](#)