Оглавление

Задача.

1. Создать Angular 4 проект

2. Реализовать описанный функционал

3. К ответному письму приложить проект в виде ссылки на репозиторий или облако

Реактив

Интерфейс:

- реактивный input ввода email с встроенной в Angular валидацией

Задачи:

1. Подсвечивать границу input в зависимости от результатов валидации

2. При вводе email печатать содержимое input в консоль с задержкой 300мс

Структуры данных

Интерфейс:

- дерево [PrimeNg](https://www.primefaces.org/primeng/#/tree)

Задачи:

1. Создать angular-сервис получения данных, описанных во [вложениях](attachment) 1 и 2

2. Подключить дерево PrimeNg и отобразить данные дерева 1

3. Лейблы листьев дерева, значения которых отличаются от соответствующих у дерева 2, сделать красным текстом

4. Сформировать массив из отличных листьев согласно указанному ниже формату, вывести его в консоль

Формат массива:

```
[
        { tree1: 'labelFromTree1', tree2: 'labelFromTree2'},
        { ... } ];
```

Описание решения.

Скриншот UI.

ReactiveFormComponent

Email (example@domain.com)

abc@

Form value: { "emailFormControl": "abc@" }

Email value after 500ms: abc@

Form status: "INVALID"

TreeCompareComponent

Структура 1

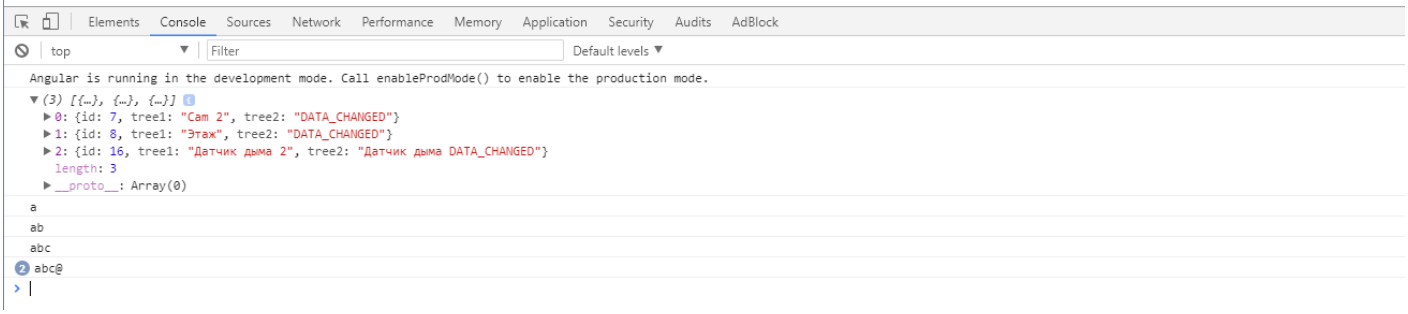
- System
  - Core
    - Камеры
      - Офис 1
        - Общий вид
        - Оборудование
          - Cam 1 (Сервер)
          - **Cam 2**
        - **Этаж**
        - Лестница
      - Офис 2
        - Общий вид
  - Core 2
    - Датчики
      - Офис 1
        - Датчик дыма 1
        - **Датчик дыма 2**
      - Офис 2
        - Датчик дыма 1

Структура 2

- System
  - Core
    - Камеры
      - Офис 1
        - Общий вид
        - Оборудование
          - Cam 1 (Сервер)
          - DATA_CHANGED
        - DATA_CHANGED
        - Лестница
      - Офис 2
        - Общий вид
  - Core 2
    - Датчики
      - Офис 1
        - Датчик дыма 1
        - Датчик дыма DATA_CHANG
      - Офис 2
        - Датчик дыма 1

Массив отличий

```
[
        {
                "tree1": "Cam 2",
                "tree2": "DATA_CHANGED"
        },
        {
                "tree1": "Этаж",
                "tree2": "DATA_CHANGED"
        },
        {
                "tree1": "Датчик дыма 2",
                "tree2": "Датчик дыма DATA_CHANGED"
        }
]
```



Решение задачи 1 (реактивная форма и контрол).

Добавляю, при помощи FormBuilder, контрол на форму, привязываю его к html-элементу.

Добавляю два валидатора (Validators.compose([Validators.required, Validators.email])), добавляю выражение, которое подсвечивает раму формы при её невалидности:

```
(formGroup.invalid && (formGroup.dirty || formGroup.touched)
```

Сам контрол подсвечивается автоматически полосой снизу, т.к. используется элемент ввода от PrimeNG (вернее, его стили, сам элемент – обычный input).

Добавляю обработчик события keyup, имитирую задержку через Promise.all в 500 миллисекунд, т.к. 300 сложнее заметить.
По истечении задержки вывожу текст из контрола в консоль и на форму.

Решение задачи 2 (дерево).

Я думаю, для сравнения таких данных (значений датчика) их лучше всего снабжать идентификатором датчика, перед процессом сравнения.

Так как в наших тестовых данных идентификатора нет, то я добавляю его, первым шагом.

В компоненте Tree от PrimeNG предусмотрено для этого необязательное поле "data".

В нашем случае обе структуры одинаковы по количеству полей и могут отличаться только значением одного поля.

Поэтому за идентификатор поля (датчика) я принимаю просто его порядковый номер.

После асинхронного извлечения обоих деревьев от сервиса-заглушки, они обрабатываются и отображаются (я отображаю оба дерева, для удобного восприятия).

Первым этапом я рекурсивно добавляю идентификаторы узлам обоих деревьев, далее преобразую их в плоские коллекции и сохраняю результат сравнения.

После чего присваиваю стиль-подсветку нужным узлам первого дерева и вывожу результат сравнения в заданном формате на экран и в консоль.