Оптимизация для QA

С целью упрощения e2e тестирования необходимо выполнять оптимизации frontend приложения.

data-test-id

Для интерактивных и содержащих элементов данные, подлежащие проверке, прописываем тестовые атрибуты **data-test-id**.

Стратегию именования атрибутов наследуем из БЭМ:

Имя (TestID) сущности уникально. Основная идея соглашения по именованию — вложить смысл в имена и сделать их максимально информативными для разработчика и QA.

Можно сравнить один и тот же TestID, написаный разными способами:

- `menuitemvisible`
- `menu-item-visible`
- `menuItemVisible`

Чтобы понять смысл первого имени, нужно вчитаться в каждое слово. В последних двух примерах имя явно разделяется на логические части. Но ни одно из имен пока не помогает точно определить, что `menu` — это блок, `item` — элемент, а `visible` — модификатор. Чтобы имена сущностей были однозначными и понятными, в БЭМ были разработаны правила формирования имен сущностей.

Правила формирования имен

`block-name__elem-name_mod-name_mod-val`

1. Имена записываются латиницей в нижнем регистре.

- Для разделения слов в именах используется дефис (`-`).
- Имя блока задает пространство имен для его элементов и модификаторов.
- Имя элемента отделяется от имени блока двумя подчеркиваниями (__).
- Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (_).
- Значение модификатора отделяется от имени модификатора одним подчеркиванием (_).
- Значение булевых модификаторов в имени не указывается.

Блок

Функционально независимый компонент страницы, который может быть повторно использован. Это может быть компонент или его внутренний элемент.

Особенности:

- Название блока характеризует смысл («что это?» — «меню»: `menu`, «кнопка»: `button`), а не состояние («какой, как выглядит?» — «красный»: `red`, «большой»: `big`).

Предлагается всегда давать корневому элементу компонента TestID в виде блока, который может в последствии служить пространством имен для вложенных блоков и компонентов.

```
<!-- some lorem-ipsum.component.html file -->


<div class="lorem-ipsum" data-test-id="lorem-ipsum">
        <!-- component body -->
</div>
```

Блоки могут быть вложенными друг в друга, Например

```
<!-- some lorem-ipsum.component.html file -->


<div class="lorem-ipsum" data-test-id="lorem-ipsum">

        <div data-test-id="edit-form">
                <!-- ... -->
        </div>


        <div data-test-id="linked-entities">
                <!-- ... -->
        </div>
</div>
```

Что делать блоками

- компоненты
  - допускается использование (при необходимости выделить компоненты среди таких же) в родительском шаблоне, например

```
<div *ngFor="entity of entities$ | async;">
        <lorem-ipsum data-test-id="lorem-ipsum-{{ entity.id }}"
                     [entity]="entity"
        ></lorem-upsum>
</div>
```

- крупноблочные не-компоненты, карточки, формы
- блоки и компоненты, выводимые в циклах

Элемент

Составная часть блока, которая не может использоваться в отрыве от него.

Особенности:

- Название элемента характеризует смысл («что это?» — «пункт»: `item`, «текст»: `text`), а не состояние («какой, как выглядит?» — «красный»: `red`, «большой»: `big`).
- Структура полного имени элемента соответствует схеме: `-__-`. Имя элемента отделяется от имени блока двумя подчеркиваниями (__).

```
<!-- some lorem-ipsum.component.html file -->


<div class="lorem-ipsum" data-test-id="lorem-ipsum">

        <div data-test-id="product-edit-form">
                  <input data-test-id="product-edit-form__input">

                <!-- ... -->
            <button type="submit" data-test-id="product-edit-form__submit"
></button>


        </div>
/div>
```

Вложенность

- Элементы можно вкладывать друг в друга.
- Допустима любая вложенность элементов.
- Элемент — всегда часть блока, а не другого элемента. Это означает, что в названии элементов нельзя прописывать иерархию вида `block__elem1__elem2`.

Блок может иметь вложенную структуру элементов в DOM-дереве:

```
<div data-test-id="block" -->

    <div data-test-id="block__elem1">

        <div data-test-id="block__elem2">

            <div data-test-id="block__elem3"></div>
        </div>
    </div>
</div>
```

Модификатор

!!! Необходимость модификаторов для TestID пока остается открытой.

По бэм модификатор - это сущность, определяющая внешний вид, состояние или поведение блока либо элемента.

Особенности:

- Название модификатора характеризует внешний вид («какой размер?», «какая тема?» и т. п. — «размер»: `size_s`, «тема»: `theme_islands`), состояние («чем отличается от прочих?» — «отключен»: `disabled`, «фокусированный»: `focused`) и поведение («как ведет себя?», «как взаимодействует с пользователем?» — «направление»: `directions_left-top`).
- Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (_).

Для тестовых целей пока не подтвеждена необходимость учета состояний, поэтому модификаторы можно использовать например для вложенных сущностей одного типа, например полей форм.

```
<!-- some lorem-ipsum.component.html file -->


<div class="lorem-ipsum" data-test-id="lorem-ipsum">

        <div data-test-id="product-edit-form">
                    <input data-test-id="product-edit-form__input_entity-
name">



                <input data-test-id="product-edit-form__input_entity_tech-
description">


                <textarea data-test-id="product-edit-
form__input_entity_art-description"></textarea>


                <!-- ... -->
            <button type="submit" data-test-id="product-edit-form__submit"
></button>


        </div>
/div>
```

best practices / refs

https://medium.com/better-programming/decouple-tests-with-data-attributes-c920606c5f27

https://docs.cypress.io/guides/references/best-practices.html

https://www.logigear.com/blog/test-automation/15-best-practices-for-building-an-awesome-protractor-framework/

https://medium.com/slalom-build/testing-angular-applications-with-selenium-java-4bca1c6d08b5