**Инструкция для верстальщиков**

 **Содержание:**-- Установка библиотеки
-- Обновление библиотеки
-- Правила работы с файлами шаблонов-- Индивидуальные случаи (дополнить вёрстку)

**Установка библиотеки**

1. Скачать zip файл со стенда (https://templates-meo.vercel.app/)
ссылка на стенде справа --[Скачать библиотеку utcf (zip)](https://templates-meo.vercel.app/_utcf/utcf.zip) --
2. Распаковать и положить папку на cdn, для подключения сборщиков тренажёров всех курсов к одной папке, чтобы исключить дубликаты библиотеки.

**Обновление библиотеки**

1. Обновление библиотеки (после внесения изменений, исправления ошибок или добавления новых тренажёров) происходит путём повторения пункта 1 и 2 – **Установка библиотеки**, т.е. заменой папки на новую, скаченную.

**Правила работы с файлами для шаблонов.**

1. Подключаем в ХТМЛ файл общие стили для тренажеров (путь правится в зависимости от того, где размещена папка utcf)

<link rel="stylesheet" ref="../utcf/common\_files/common\_\_styles.css"><!-- общие стили тренажеров-->

1. Подключаем стили для тренажеров, в зависимости от того, где находится сам тренажер (пути к файлам корректируются, как в п1.):
* если в правом поле, то <link rel="stylesheet" href="../utcf/individual\_files/individual\_styles/individual\_set\_styles\_modal.css"><!-- общие стили тренажеров-->
* если не в основном окне (левом, большом поле), то <link rel="stylesheet" href="../utcf/individual\_files/individual\_styles/individual\_set\_styles\_standart.css"><!-- общие стили тренажеров-->
1. Подключаем на страницу скрипты для каждого используемого тренажера ( пути корректируются аналогично п.1)

<script defer src="js-data/templates/right\_3\_2/dnd\_withSlider.js" type='module'></script>

1. Сам скрипт тренажера со входящими данными размещается в папке js-data
2. в файле с тренажером производятся следующие действия:
3. в самом верху идет подключение js-скрипта, которых отвечает за работу тренажера



Путь к нему нужно скорректировать, в зависимости от того, где расположена папка utcf

1. Далее идет блок для вставки в хтмл-файл базовой верстки тренажера



Вставляется вся разметка, что расположена между /\* \*/ (лишних элементов в ней нет, что-то корректировать в ней не нужно, кроме атрибута id, где должен быть указан уникальный в этом хтмл-документе id тренажера)

Если тренажер подразумевает вставку svg-картинки, то вместо свг-примера вставляется нужная svg, но к ней добавляются нужные классы, которые описаны в примере.


1. Далее идет сам блок, который отвечает за контент внутри тренажера. Он располагается:
(() => { объекты для заполнения })();.

Если данный вид тренажера встречается несколько раз на данной хтмл-странице, то создавать несколько js-файлов не нужно, просто ниже копируется и вставляется такой же блок

(() => { объекты для заполнения })();.

* Первым пунктом вставляется указанный в хтмл id тренажера 
* Далее в зависимости от типа тренажера заполняются остальные поля.
* Если встречается такая структура, то это значит, что это массив с несколькими объектами (элементами на странице).

ВАЖНО: самые первые скобочки [] и общее название после const не копируются, дублируются только внутренности



Чаще всего их больше чем 1, поэтому если этих элементов много, что внутри [] нужно несколько раз скопировать структуру 1-го элемента

Пример:



Внутри поля заполняются в зависимости от описания к полям, что указаны выше данного блока

* Если встречается такая запись

, а нужно больше элементов, то они указываются внутри через запятую



* Для такого типа данных



Добавление дополнительных полей также через копирование



* При такой сложной записи, можно копировать как внутренние [], так и {}



Пример: 

* в самом низу идет подключение самой функции, отвечающей за работу тренажера, в ней ничего не нужно менять



* **Индивидуальные случаи (дополнить вёрстку)**
* Тренажер writeAnswer\_1:

ХТМЛ-верстка заполняется верстальщиком в зависимости от задания. Для инпутов и иконок с плеером предусмотрены определенные классы:

у инпута который участвует в получении ответа, обязательно должен быть класс "writeAnswer\_1\_task\_input", по нему собирается правильность ответа

если input предусматривает числа, то type="number", если текст то type="text"

чтобы ограничить выбор отрицательных значений дописывается min='0'

пример: для текста

 <input

              class="writeAnswer\_1\_task\_input writeAnswer\_1\_task\_input\_txt"

              type="text"

              id="task1\_1"

              size="100"

            />.

Пример для числа:

 <input class="writeAnswer\_1\_task\_input" min="0" type="number" />

 Звуки:

Каждому значку звука и соответствующему ему тегу audio присваиваиваются уникальные для этой пары data-атрибут audio-data и id соответственно, по которому и будет происходить воспроизведение звуков

у тега audio обязательно должен быть класс "writeAnswer\_1\_task\_audio", по нему собираются звуки

иконка звука верстается так, внутри есть тег аудио, поля заполняются согласно правилам выше

приамер:

 <div

              class="buttonPlayPausePlayPause\_wrap buttonPlayPause--play"

              audio-data="1writeAnswer\_1\_task-1"

            >

              <div

                class="buttonPlayPause\_\_shape buttonPlayPause\_\_shape--one"

              ></div>

              <div

                class="buttonPlayPause\_\_shape buttonPlayPause\_\_shape--two"

              ></div>

              <audio

                id="1writeAnswer\_1\_task-1"

                class="writeAnswer\_1\_task\_audio"

                src="sound/writeAnswer\_1/sound\_7\_task7\_1.mp3"

                audio-data="1writeAnswer\_1\_task-1"

              >

                Your browser does not support the

                <code>audio</code> element.

              </audio>

            </div>

Если требуется выстроить элемены звук и текст с инпутом в одну строку, надо обернуть их в <div class=”writeAnswer\_1\_div” ></div>

Пример:



* Тренажер writeAnswer\_2:

ХТМЛ-верстка заполняется верстальщиком в зависимости от задания. Для селектов и иконок с плеером предусмотрены определенные классы:

у select который участвует в получении ответа, обязательно должен быть класс "writeAnswer\_2\_task\_select", по нему собирается правильность ответа пример:



 у картинки обязательно должен быть класс writeAnswer\_2\_task\_image, по нему происходит увеличение картинки пример:



у тега audio обязательно должен быть класс "writeAnswer\_2\_task\_audio", по нему собираются звуки

 id у каждого тега audio должно совпадать с атрибутом audio-data у кнопки звука

 иконка звука верстается так, внутри есть тег аудио, поля заполняются согласно правилам выше:



класс writeAnswer\_2\_task\_line - это одна строка с ответами, нужна еще одна сторока? делай еще такой див пример:



если текст используется то для вытягивания в одну строку добавляем к writeAnswer\_2\_task\_answer\_item класс writeAnswer\_2\_task\_answer\_item\_text как в примере:

