Тестовые задания для Лаборатории

## Задание №1

Разработать генератор форм для ввода пользовательских данных.

На вход подаются данные об элементах формы в формате JSON:

* Тип элемента;
* Внешнее оформление (стили);
* Маска ввода;
* Правила валидации значений;
* Отображаемое значение по умолчанию.

На выходе должно получиться клиентское приложение/html-страница, отображающая элементы формы, описанные во входном JSON-файле.

Можно использовать любой язык программирования.

Добавлять какую-то дополнительную клиентскую логику и оформление не требуется.

Пример Json-файла

{

"form": {

"name": "Название формы",

"postmessage": "Сообщение в случае успешного заполнения формы",

"items": [{

"type": "filler",

"attributes": {

"message": "Произвольные текст"

}

}, {

"type": "text",

"attributes": {

"name": "Имя элемента",

"placeholder": "Текст для placeholder",

"required": true,

"value": "",

"label": "Label для элемента",

"class": "css-class",

"validationRules": [{

"type": "email"

}],

"disabled": false

}

}]

}

}

Структура входных данных

Входные данные в виде JSON-массива:

* "form" – это блок формы содержащий массив элементов и сообщение пользователю по результатам взаимодействия с формой:
  + "name" – имя формы (пример: "two", "one");
  + "items" – массив элементов формы;
  + "postmessage" – сообщение в случае успешного взаимодействия.

Все элементы формы содержат атрибут "type", он определяет какой элемент описывается:

**Тип filler**

«Наполнитель» - это произвольный текст, поддерживает html разметку. Элемент содержит атрибут "message", который содержит размещаемый контент.

**Тип text**

Поле для ввода текста. Содержит атрибуты:

* "name" – имя элемента;
* "placeholder" – при проверке, значение поля не должно быть равно значению плейсхолдера.  Необязательный атрибут;
* "required" – это свойство, указывающее на обязательность заполнения элемента, принимает значения true / false;
* "validationRules" – это правила валидации. Содержит атрибуты:
  + "type" – это тип проверки. Может принимать следующие значения:
    - text – поле должно содержать любой текст;
    - tel – поле должно содержать номер телефона;
    - email – поле должно содержать e-mail адрес;
* "value" – это предустановленное значение элемента, может быть пустым;
* "label" – описание поля, произвольный текст, может быть пустым;
* "class" – css-класс элемента;
* "disabled" – активен или нет элемент, принимает значения true / false.

**Тип textarea**

Поле для ввода многострочного текста. Атрибуты типа аналогичны атрибутам типа text.

**Тип checkbox**

Поле для выбора. Атрибуты типа аналогичны атрибутам типа text, за исключением атрибута "value". Также содержит дополнительный атрибут "checked", указывающий первоначальное состояние флажка. Он принимает значение true/false.

**Тип button**

Кнопка. Содержит атрибуты:

* "class" – css-класс кнопки;
* "text" – текст кнопки.

**Тип select**

Выпадающий список. Атрибуты типа аналогичны атрибутам типа text. Кроме того, этот тип содержит дополнительный атрибут:

* "options" – список выпадающих значений, каждое значение которого содержит атрибуты:
  + "value" – значение элемента списка;
  + "text" – текст элемента списка;
  + "selected" – активен или нет элемент по умолчанию, принимает значение true / false.

**Тип radio**

Атрибуты типа аналогичны атрибутам типа text. Содержит дополнительный атрибут:

* "items" – список элементов набора опций. Каждый элемент содержит:
  + "value" – значение опции, которое будет передано на сервер;
  + "label" – текст опции, который видит пользователь;
  + "checked" – активна или нет опция, принимает значение true / false.

Задание №2. Анализ кода

Дан исходный код двух функций. Необходимо проанализировать исходный код и написать, что он выполняет. Если на ваш взгляд, исходный код можно оптимизировать или каким-то другим способом улучшить, то предложить свой вариант.

static int Func1(KeyValuePair<int, string>[] a, int low, int high, int key)

{

  int middle = low + ((high - low)/2);

  if (low == high)

    return low;

  if (key > a[middle].Key)

    return Func1(a, middle + 1, high, key);

  return Func1(a, low, middle, key);

}

static void Func2(ref KeyValuePair<int, string>[] a, int key, string value)

{

  int pos;

  KeyValuePair<int, string> keyValuePair;

  if (a.Length == 0)

  {

    Array.Resize(ref a, 1);

    keyValuePair = new KeyValuePair<int, string>(key, value);

    a[0] = keyValuePair;

    return;

  }

  if (key < a[0].Key)

    pos = 0;

  else if (key > a[a.Length - 1].Key)

    pos = a.Length;

  else

    pos = Func1(a, 0, a.Length - 1, key);

  Array.Resize(ref a, a.Length + 1);

  for (int i = a.Length - 1; i > pos; i--)

    a[i] = a[i - 1];

  keyValuePair = new KeyValuePair<int, string>(key, value);

  a[pos] = keyValuePair;

}