Содержание

1 Формат JSON контента HTML формы

1.1 Ключ/значение тега

Каждый тег можно описать его названием, атрибутами, содержимым контентом и указать дочерние объекты. В JSON каждый тег имеет следующие ключи:

name

Для создания тега потребуется его имя, которое будет всегда единственным значением первого ключа.

attributes

В созданный тег добавляем ключи атрибутов (их может быть неограниченное количество, поэтому значения ключа представляют собой массив). Логические атрибуты также предусмотрены модулем.

values

К созданным атрибутам в том же цикле добавляем их значения, так же являются массивом значений.

content

Для упрощения задачи текстовый объект между тегами прописывается в ключе контента тега. <h1> **Контент** </h1>

children

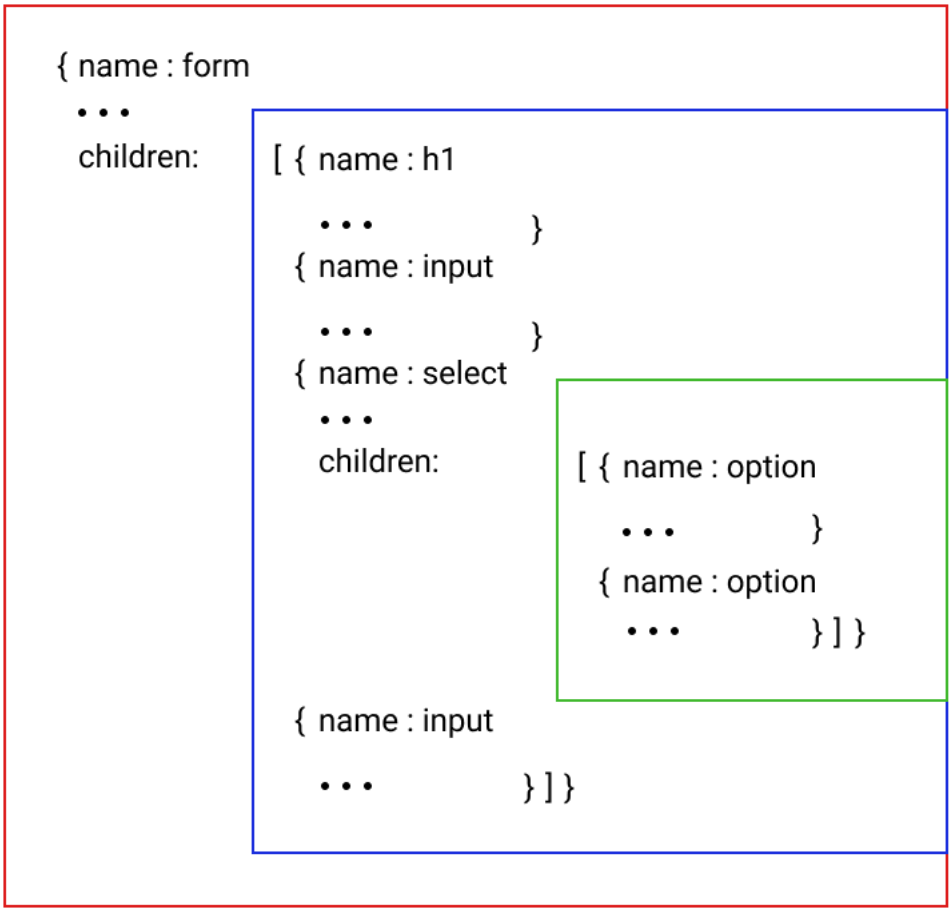
Если у тега есть дочерние объекты, то они указываются массивами в ключе «дети». Дочерние объекты так же содержат все вышеперечисленные ключи.

Все ключи создаются у тега, если они в нем используются.

Такая структура позволяет создавать следующие объекты формы: Input + label (поле записи), textarea, radio, checkbox, slider, file, datapicker и другие.

1.2 Дерево тегов

Так выглядит дерево тегов в JSON:



Принцип вложенных элементов позволяет создавать списки и выпадающие списки (ul/select), а также таблицы.

2 Работа модуля на JavaScript

2.1 Работа модуля

2.1.1 Создание тега по ключам/значениям

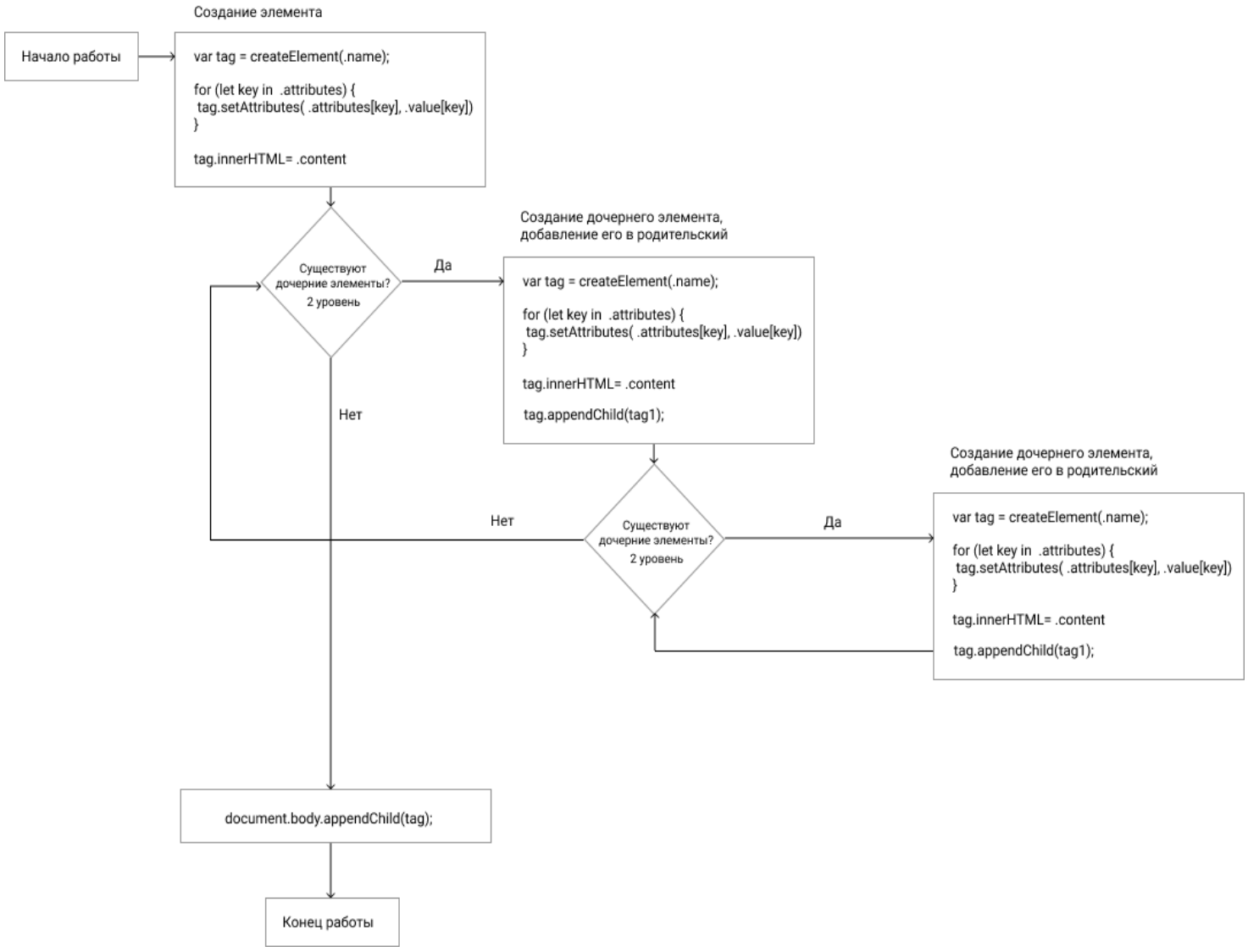
1. Создаем элемент в говорящей переменной по значению ключа .name
   var tag1 = createElement(.name);
2. Задаем ему атрибуты, так как их может быть много – создаем цикл for. Учитываем то, что атрибутов может не быть, поэтому проверяем их наличие.
   for (let key in .attributes) {
   tag1.setAttributes(.attributes[key], .value[key])
   }
   Задаем атрибутам их значения по ключу value. В коде учтено, что бывают и логические атрибуты.
3. Вкладываем контент в тег, проверяем его на наличие.
   tag1.innerHTML= .content;
4. Помещаем элемент в родительский элемент. Родители могут задаваться во внешнем цикле.
   tag.appendChild(tag1);
5. Помещаем в html документ внутрь тега body.
   document.body.appendChild(tag);

Таким образом, эта основной алгоритм создания элементов, который в теории может быть вызываемой на протяжении создания дерева функцией.

2.1.2 Алгоритм работы модуля

1. После получения данных json (преобразование их в объект) создаем верхний узел (здесь это всегда форма). Далее п. 2.1.1.
2. Запускаем цикл для создания тегов для всех дочерних элементов формы. В теле цикла п. 2.1.1, при каждом запуске помещаем объект в form, a form в body.

3. После помещения второго уровня в форму, запускается цикл 3его уровня, который в свою очередь запускается при создании элемента его родителя, и каждый новый элемент по очереди помещается в тег родителя, последний уже находится в форме.
4. В конце верхнего цикла все элементы помещаются в документ.



2.2 Использование модуля

1. В index.html задаем тип скрипта модульным.
2. Импортируем нужную функцию из файла.
3. Вызываем её, в параметрах указываем путь к своему файлу JSON.
4. Запускаем index.html – готовая верстка по данным json файла.

3 CSS

Не успела его сделать. В теории можно добавлять его через атрибуты и дочерние элементы. Общий дизайн для всех элементов формы сделать заранее.