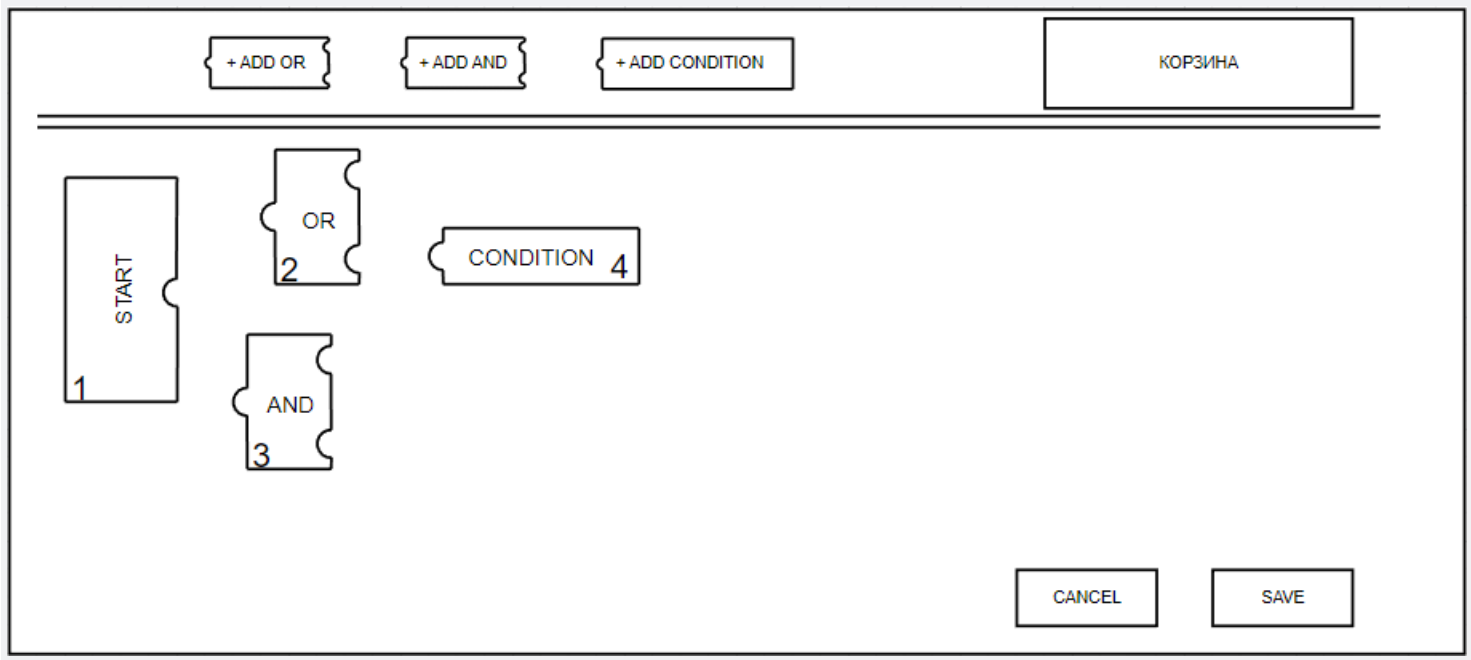
ТЗ визуальный конструктор

Нужен "визуальный конструктор условий" (далее **"Конструктор"**) для сайта: это должно быть "удобное LEGO", в котором юзер может drag-n-drop таскать элементы и стыковать их, следуя заданным правилам. На выходе у юзера должен получиться "граф" с деревом условий .
Второй уровень конструктора – настройка "листиков" – итоговых элементов дерева.
Конструктор должен обеспечивать настройку логического дерева принятия решения – на основании итоговых "условий" и их сочетаний через булёвые (логических) операции: "И" \ "ИЛИ".
Каждый "листик" (далее **"условие"**) - элемент, настраиваемый индивидуально, подчиняющийся отдельному набору правил и условий.
Конструктор валидируется на соответствие правилам и сохраняется в 2х аспектах: визуально и логически.

По сути – нужно перенести текущее решение в приятный графический вид. В текущей реализации это всё выглядит так:

Интерфейс
Должен работать на десктопе и мобиле и иметь масштабирование (зум). состав:

- поле редактора – на нём происходит все творчество
- панель инструментов - оттуда drag-drop тянутся элементы
- кнопки "сохранить" и "отмена" - соответственно.



Поведение
При построении дерева:

- МОЖНО оставлять элементы висеть в воздухе. они просто будут "не у дел"
- элементы должны визуально "вставать в паз" при наведении, sticky connection
- при невозможность корректно вставить 1 элемент в другой, должно быть явное визуально сообщение (красная подсветка, "отторжение", "отталкивание")
- можно "вынимать" элементы из слота
- при перемещении элемента, в котором закреплены дочерние, они таскаются вместе с родительским
- Элементы удаляются при перетаскивании в "корзину"

Поведение при настройке условия:

- Условие не является законченным, пока все необходимые его поля не заполнены
- до этих пор у него будет подсветка красным, и если актуально - "рваный" правый край
- при расширении элемента Условие за счет выбора значений по дереву элементов увеличение графического элемента Условие должно происходить с эстетической отрисовкой (не резким появлением следующего элемента)

Элементы

"Корень"
Начальная "точка", к которой крепится всё остальное. К нему можно прицепить элементы "УСЛОВИЕ", "И", "ИЛИ". должен быть чем-то заполнен обязательно.

"Логика ИЛИ"; "Логика И"
Это 2 элемента, создающих ветвление в дереве. К ним можно прицепить 1+ элемент "условие", "и", "или". По умолчанию у элемента 2 посадочных слота. При заполнении всех слотов, добавляется еще один дополнительный. Можно держать незаполненный слот.

"Условие"
Конечный элемент конструкции, к нему ничего не прицепляется, но он настраивается.

Настройка "условия"
Текущая реализация выглядит так:
видео как с этим работать
Несмотря на то, что к "условию" нельзя ничего "прицепить", оно само по себе является сложно-составным элементом. Само "условие" можно собрать из фрагментов, определённых настроечным json-ом, далее **flex-json.** flex-json содержит в себе рекурсивное дерево фрагментов и их спецификацию, следуя которому, можно "собрать условие".
вот он:
strategies_config (2).json

Если у фрагмента есть дочерние элементы, то они появляются справа от текущего заполненного фрагмента при исполнении условий отображения.
flex-json состоит из следующих "фрагментов":

- дропдаун – селектор, на выбор 1 из нескольких, обязателен к выбору.
- инпут - инт \ флоат, числовой короче. имеет мин-макс-шаг, округление
- чекбокс/switch - да\нет
- попап – фрагмент, вызывающий появление модалки с дочерними элементами этого попапа.

Flex-json-фрагменты могут иметь дочерние фрагменты и условия их появления: таким образом, следующий фрагмент на выбор будет рекурсивно зависеть от предыдущего.
Значения фрагментов проверяются на соответствие спецификации

Проверка дерева
Проверка разделяет все элементы в поле конструктора на те, которые будут иметь значение при дальнейшей работе и те, которые остаются "висеть без дела".
Проверка должна выполняться "на лету", и полностью корректные элементы должны подсвечиваться (обводиться) зелёным. Элементы, не прошедшие проверку (в т.ч. "висящие"), подсвечиваются красным \ желтым \ серым. Правила сочетаний элементов:

- нет "висящих в воздухе" элементов, они связаны в корректную древовидную структуру

- "корень" заполнен, имеет дочерний элемент
- все "ветви" кончаются элементом "условие"

Сохранение
Данные сохраняются в виде рекурсивного json, и в них должны быть в явном виде указаны:

- id элемента
- тип эелмента
- координаты для отображения
- статус валидации, активности
- дочерние элементы

"условия" сохраняются как отдельные json-структуры, а дерево (листик) содержит "адрес" с ID от json нужного условия. это плоский словарь полей.

Последовательность работы конструктора

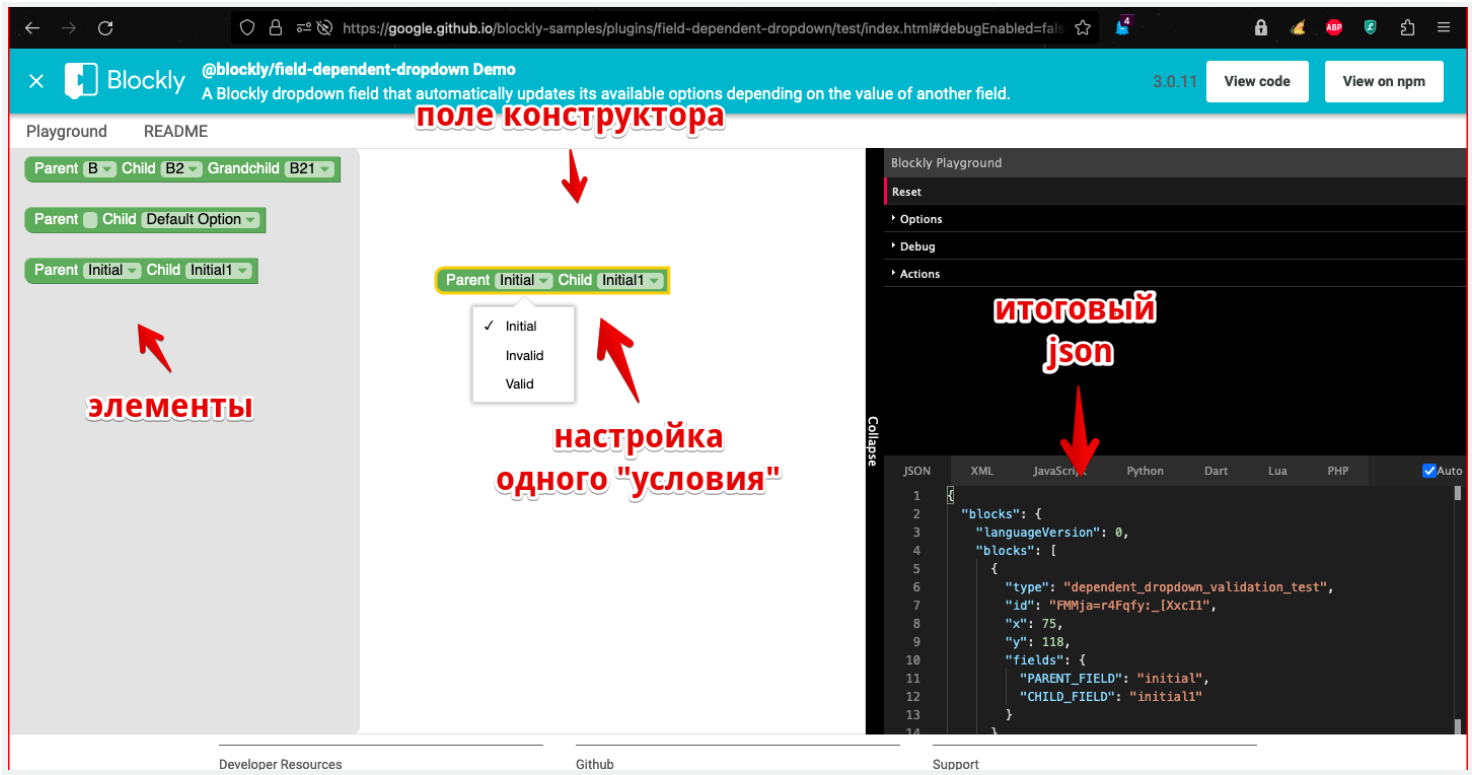
1. принять данные в виде переменной через props
2. отобразить их
3. юзер вносит изменения
4. изменения проверяются на "правила сочетаний элементов", требуется корректность.
5. изменения передаём в функцию из props
6. отобразить визуально обновлённую json-структуру

Технический стек
Конструктор исполнен в виде React-компонента на Typescript, принимающего props
На вход подаются:

- props данные для отображения (json)
- коллбек-функция

Результат своей работы (json) конструктор возвращает в коллбек-функцию

Мы вдохновлялись: https://google.github.io/blockly-samples/


https://google.github.io/blockly-samples/plugins/field-dependent-dropdown/test/index.html#debugEnabled=fals
Blockly
@blockly/field-dependent-dropdown Demo
A Blockly dropdown field that automatically updates its available options depending on the value of another field.
3.0.11
View code
View on npm
Playground
README
поле конструктора
Parent B Child B2 Grandchild B21
Parent Child Default Option
Parent Initial Child Initial1
элементы
Parent Initial Child Initial1
Initial
Invalid
Valid
настройка
одного "условия"
Blockly Playground
Reset
Options
Debug
Actions
итоговый
json
Collapse
JSON
XML
Python
Dart
Lua
PHP
Auto
{
"blocks": {
"languageVersion": 0,
"blocks": [
{
"type": "dependent_dropdown_validation_test",
"id": "FMMja=r4Fqfy:_[XxcI1",
"x": 75,
"y": 118,
"fields": {
"PARENT_FIELD": "initial",
"CHILD_FIELD": "initial1"
}
Developer Resources
Github
Support