Министерство образования и науки Челябинской области

Муниципальное автономное общеобразовательное учреждение

"Миасская средняя общеобразовательная школа № 16"

Индивидуальный проект

"Реализация генератора комбинаторных задач ЕГЭ по информатике на языке JavaScript"

Автор проекта: Велюго Кирилл Олегович,

Ученик 11В класса МАОУ "МСОШ№16"

Наставник проекта: Горяинов Сергей Игоревич,

Учитель информатики

Миасс

2021

Для успешной сдачи ЕГЭ по информатике необходимо решать множество задач, для отработки тех или иных навыков. Но не все сборники для подготовки к ЕГЭ имеют необходимое количество заданий, либо не имеют ответов к ним. Поэтому создание генератора, который сможет создавать большое количество задач, способно решить эти проблемы.

Генератор должен позволять пользователю выбрать нужные типы и необходимое количество этих задач. После чего он должен вывести на экран пользователя сгенерированные условия и ответы к ним (опционально).

Данный генератор имеет востребованность среди учащихся, которые заинтересованы в улучшении навыка решения комбинаторных задач, которые встречаются в ЕГЭ по информатике, выявлении слабых сторон при решении и уменьшении времени, затрачиваемого на решение.

Ввиду того, что комбинаторные задачи зачастую вызывают сложности при решении, так как имеют множество тонкостей, о которых сложно узнать, были выбраны именно они, как задания, которые будет создавать генератор.

Процесс создания генератора потребовал решения множества задач. Первая из задач - выбор платформы на которой будет реализована задумка. Для этого были выбраны веб-средства, поскольку благодаря им генератором смогут воспользоваться все пользователи, которые имеют доступ к интернету.

Следующая задача, заключалась в том, что нужно было продумать, как будет выглядеть веб-приложение, как оно будет реализовано. Для этого были изучены основы графического редактора Figma, который позволил создать макет веб-страницы для его последующей верстки.

Далее нужно было найти и проанализировать информацию о комбинаторных задачах: поиск и отбор задач, которые будут в последующем будут генерироваться, выявление алгоритмов решения этих задач, структурировать полученную информацию. Все это было выполнено благодаря материалам с сайта Константина Юрьевича Полякова.

После этого нужно было выбрать и изучить средства, позволяющие создать веб-приложение. Для этих целей были выбраны такие средства: язык разметки HTML, каскадная таблица стилей CSS, язык программирования JS.

HTML – язык разметки, который позволяет понимать браузеру, как правильно отображать блоки на веб-странице. При загрузке веб-страницы браузер интерпретирует код HTML и выводит на экран пользователя контент. HTML имеет блочную структуру. С помощью HTML на веб-сайте получилось реализовать подобную структуру: шапка веб-страницы, блок главного экрана, блок выбора типов задач, блок сгенерированных задач.

Пример сайта без использования CSS (Рисунок 1):

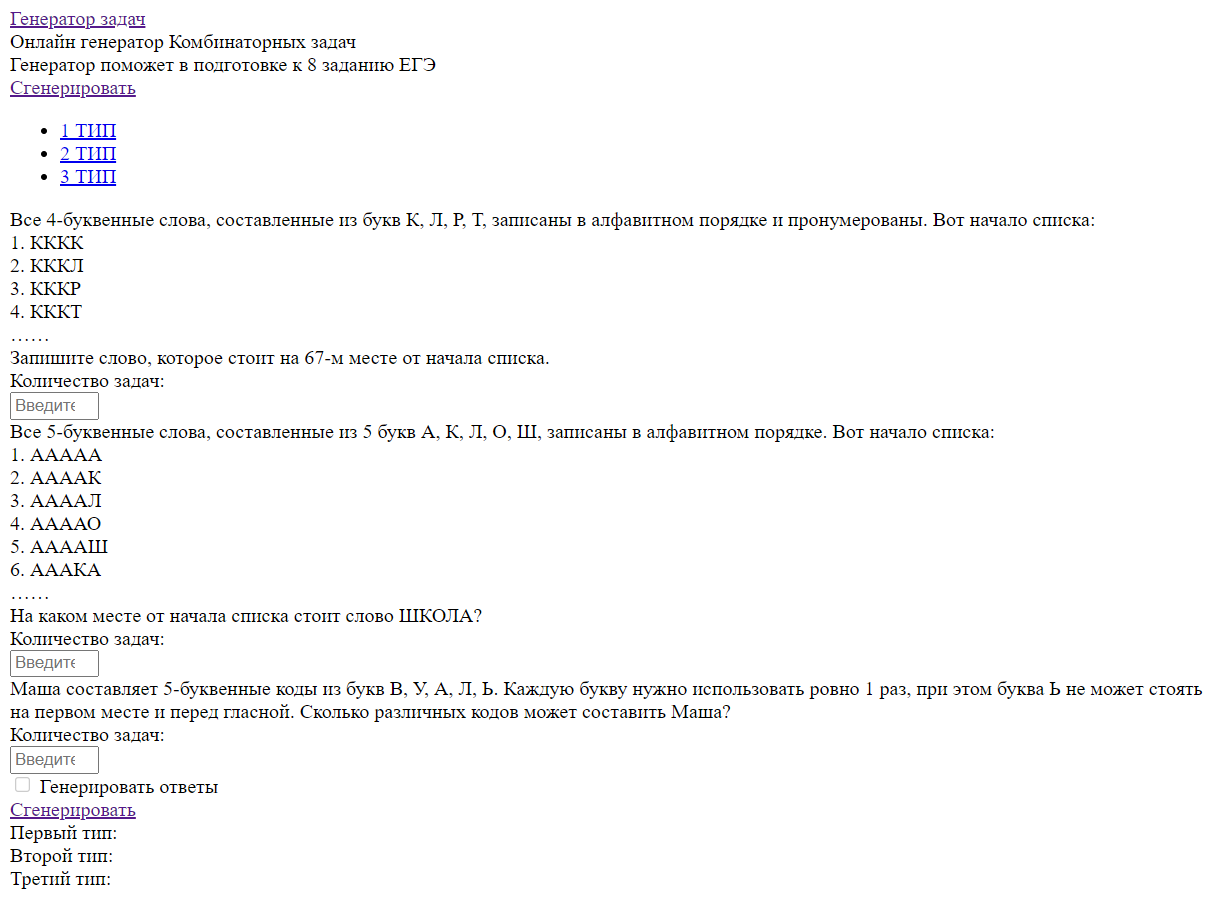
**

Рисунок 1. Сайт без использования css

Но только с помощью HTML сложно создать полноценную веб-страницу, так как у нее будет отсутствовать стилизация. Поэтому на веб-странице используется каскадная таблица стилей CSS. В современной веб-разработке почти каждый сайт использует связку HTML и CSS. CSS позволяет указать браузеру какой элемент нужно стилизовать. Для создания генератора задач потребовалось задать стили каждому блоку из HTML разметки.

Также веб-сайт должен быть адаптивным, то есть нормально функционировать на устройствах с разными размерами экранов. Для упрощения адаптации веб-сайта был использован фреймворк (средство, которое объединяет в себе различные технологии, позволяющее упростить процесс разработки продукта) Materialize, основанный на популярном направлении, которое было разработано Google, - material design. Фреймворк предоставляет 12 колоночную разметку, которая позволяет удобно располагать элементы веб-страницы относительно друг друга, в зависимости от ширины экрана на котором просматривается содержимое страницы.

Чтобы добавить обратную реакцию на действия пользователя и логику потребовалось использовать язык программирования JavaScript. Это интерпретируемый язык высокого уровня. Код интерпретируемого языка выполняется построчно интерпретатором. Часто JavaScript применяют для создание сценариев и скриптов веб-страницы. Скрипты представляют из себя набор инструкций, которые выполняются при наступлении того или иного события. Код JavaScript интерпретируется браузером во время загрузки страницы.

На веб-странице JavaScript используется для решения многих проблем. Одна из них – это переходы между блоками. Например, при нажатии кнопки «сгенерировать» блок главного экрана сменяется блоком выбора типов задач. С помощью фреймворка Materialize на веб-страницу удалось добавить вкладки для каждого типа задач. Это позволило структурировать содержимое веб-страницы и сделать более понятный интерфейс.

Другая проблема, решенная с помощью JavaSctipt, - генерация заданий, которые будут выведены пользователю на экран. Ниже приведен пример кода (Рисунок 2), который реализует генерацию выводимых задач:

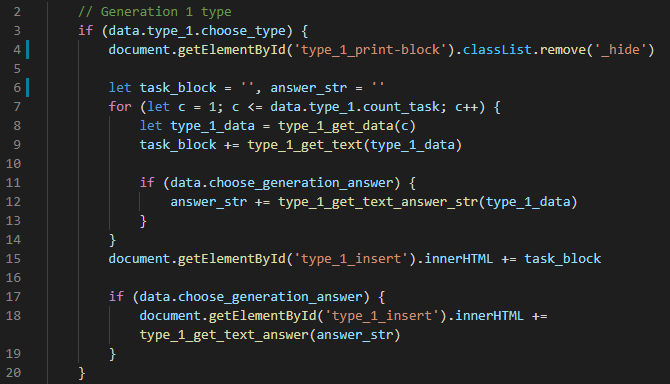


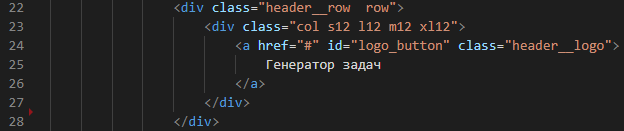
Рисунок . Пример кода, реализующий генерацию первого типа задач

В коде (Рисунок 2) выполняется следующее: если необходим генерировать задачи (строка 3), то сперва скрывается блок выбора задач (строка 4), затем происходит цикл (строки 7-20), который выполняется столько раз, сколько задач необходимо вывести. В этом цикле сначала вызывается функция (строка 8), которая возвращает объект данных, в котором содержатся необходимая информация для создания текстов, которые создаются с помощью функции (строка 9). После чего при необходимости выполняется генерация ответов к задачам (строки 11-13). Когда готовы все необходимые данные код выводит их на экран пользователя (строки 15-19).

В итоге выполнения всех поставленных подзадач и задач получилось реализовать полностью функционирующий генератор комбинаторных задач, который предоставляет пользователю выбор типов интересующих его заданий и вывод множества вариантов этих заданий с ответами.

Создание продукта проектной деятельности позволило достичь таких положительных эффектов: возможножность формирования интереса у обучающихся к решению, повышение процента решаемости, получение базовых и углубленных знаний в области решения комбинаторных задач.

В процессе реализации генератора комбинаторных задач возникло множество сложностей. Одной из сложностью было создание веб-сайта, который можно будет запустить на любом типе устройства, то есть добавление адаптивности на веб-страницу. Для решения этой проблемы была использована 12-колончатая разметка, проедоставляемая фреймворком Materialize. Ниже приведен пример (Рисунок 2), в котором она используется. Для нормальной работы контект, необходимый адаптировать, нужно поместить в блок с классом «row» (строки 22, 28), который оборачивает блок к классом «col» (строки 23, 27). В классе у блока с «col» указывается сколько колонок будет выделено для контента. Рассмотрим «s12», это означает, что для контента на маленьких экранов (small) будет выделено 12 колонок.



Другая сложность заключалась в том, что полученные алгоритмы решения задач нужно было реализовать с помощью JavaScript, для получения ответов к заданиям. Основная трудность при написании алгоритмов решения была в написании множества функций, которые генерировали ответы и выводили их в конце каждого блока с уже сгенерированными задачами. Для правильного функционирования этих функций нужно генерировать последовательности слов, состоящих из случайно выбранных букв, что также вызвало трудности в реализации.

Помимо этого другая сложность была в том, что код, который был написан в процессе реализации генератора, стал занимать очень многа места, что усложнило понимание и ориентирование в нем. Для этого были приняты решения: вынос отдельных частей кода в функции, резделение кода на несколько файлов, которые позволили улучшить ориентирование кода и понимание его.

Основным ресурсом необходимым для выполнения проекта было время. Ниже представлена таблица (Таблица 1), в которой указано сколько времени занял каждый этап работы.

|  |  |
| --- | --- |
| **Этапы** | **Примерно затраченное время** |
| Создание плана действий, выбор платформы, продумывание идеи | Около 2-ух недель |
| Анализ ифнормации о комбинаторных задачах | Около 1-го месяца |
| Изучение HTML, CSS, JS | Около 3-х месяцев |
| Реализация веб-страницы | Около 4-х месяцев |
| Написание теоретической части | Около 1-го месяца |

Таблица

**Используемые интернет ресурсы:**

1. <https://kpolyakov.spb.ru/> - Преподавание, наука и жизнь: сайт Константина Полякова.
2. <https://developer.mozilla.org/ru/> - MDN WebDocs