**Реализация генератора комбинаторных задач ЕГЭ по информатике на языке JavaScript.**

**Цели:**

Cоздание генератора комбинаторных задач ЕГЭ на языке JavaScript.

**Задачи:**

1. Изучить основы языка разметки HTML и каскадных таблиц CSS.
2. Ознакомиться с языком программирования JavaScript.
3. Отобрать ряд комбинаторных задач и проанализировать их решение.
4. Разработать программный код для генерации текстов задач и ответов на них.
5. Создать web-страницу, предоставляющую возможность генерировать комбинаторные задачи.

**HTML**

Для отображения сайта используют язык гипертекстовой разметки – HTML. С помощью него браузер понимает, как нужно правильно отображать страницу.

Принцип работы HTML заключается в том, что при переходе по адресу страницы браузер делает запрос по нему и получает файл в формате .html. Затем браузер преобразует полученный код.

С помощью HTML можно создать структуру любой сложности. Эта структура состоит из тегов и служебных элементов. Теги представляют собой своеобразные команды, которые преобразуют код в визуальные объекты на экране пользователя. Они бывают парными, например, <p></p>, <body></body>, а также непарные - <br>, <hr>. Служебные элементы применяются для подключения сторонних скриптов, метаданных и стилей.

Ниже представлены основные теги, которые образуют базовую структуру документа.

В начале любого HTML документа используется конструкция DOCTYPE. Его главная задача – указать браузеру, какой стандарт HTML используется в данном документе. Пример применения тега:

**<!DOCTYPE html>**

Далее идет парный тег <html></html>, который содержит всю информацию HTML документа. С помощью этого тега браузер понимает, где начинается и заканчивается контент. В теге используется важный атрибут lang, который указывает на каком языке представлен контент на странице. Пример применения тега:

**<html lang=”ru”>…</html>**

После этого следует парный тег <head></head>. В нем хранится служебная информация. Из содержимого этого тега браузер может узнать метаинформацию: название страницы, её описание, подключенные стили и другое. Метаинформация часто нужна для продвижения сайта. Все теги, указанные в этом теге, не отображаются на странице. Пример применения тега:

**<head>…</head>**

**sdfs>**

Последний из обязательных тегов – парный тег <body></body>. В этом теге размещена вся информация, которая будет выведена на странице. Пример применения тега:

**<body>…</body>**

**sdfs>**

Но для создания полноценной web-страницы HTML не хватает. Поэтому часто используют CSS, для задания стилей, и JavaScript для изменения содержимого страницы и добавить реакцию на действия пользователя.

**CSS**

CSS – это язык таблицы стилей. Он позволяет применять стили выборочно к элементам в HTML. В современной web-разработке почти каждый сайт использует HTML вместе с CSS. Основная цель разработки CSS – разделение содержимого файла HTML от описания внешнего вида этой страницы. Благодаря этому можно:

* Работать над HTML и CSS отдельно.
* Создавать адаптивные версии сайтов.
* Уменьшить время загрузки сайта, так как стили не будут загружаться каждый раз при попадании на один и тот же сайт.
* Создать более понятную структуру, которую проще модернизировать.

CSS имеет весьма простую и понятную структуру. Каждое правило, задаваемое в CSS, можно разделить на две основные части – селектор и блок объявлений. Селектор указывает у какого элемента нужно изменить стиль. Он стоит до знака «{». Блок объявлений располагается после фигурных скобок и говорит о том, как нужно изменить указанный элемент. Все объявления выглядят, как сочетание свойства CSS и значения. Пример синтаксиса:

**body { background: gray; }**

**sdfs>**

Но в CSS можно обращаться не только к определенном элементу, но и к нескольким одновременно. К примеру, с помощью селектора «\*» можно обратиться ко всем элементам, а с помощью селектора «.class» можно обратиться к к определенному классу.

По мимо этого, в CSS есть медиазапросы, которые позволяют создавать различные стили под разные разрешения экранов. Пример использования:

**@media (max-widht: 768px) { селектор {блок объявлений} }**

**sdfs>**

При использывании такого примера, селектор будет применен только для страниц у которых ширина меньше 768 пикселей.

Использование CSS – это мощное средство, которое позволяет упростить процесс стилизации сайта, улучшить его оптимизацию.

**JavaScript**

Для добавления обратной реакции сайта на действия пользователя, интерактивных объектов часто применяют JavaSctipt. Это интерпритируемый язык высокого уровня. Код интерпритируемого языка выполняется построчно интерпритатором. Часто JavaScript применяют для создание сценариев и скриптов web-страницы. Скрипты представляют из себя набор инструкций, которые выполняются при наступлении того или иного события. Код JavaScript интерпритируется браузером во время загрузки страницы.

JavaScript способен исполнять различные операции, такие как:

* Математические вычисления.
* Обработка и изменение HTML.
* Взаимодействие с пользователем.
* Добавление эффектов и анимаций.

Возможности JavaScript не останавливаются на добавлении интерактивности в web-страницу. Сегодня JavaScript можно использовать повсеместно: создавать серверные приложения, web-приложения, мобильные приложения, программное обеспечение. Для этого достаточно имееть один из движков JavaScript. Например: V8, SpiderMonkey, Chakra и многие другие.

Движки работают подобным образом:

* Движок читает код скрипта.
* Происходит преобразование скрипта в машинный код.
* В итоге запускается машинный код, который вычислительная машина может обрабатывать с большой скоростью.

Важно отметить, что движок применяет оптимизации на каждом шаге. Благодаря этому скрипты работают с большой скоростью.

JavaScript является важный звеном в создании web-страницы. Он позволяет добавить большое количество функционала. Самое главное, что JavaScript поддерживается почти всеми современными браузерами.