1. **Принцип "DRY".**

Ошибки, которые были мной выявлены:

1. Повторяющийся код в стиле «CSS».

На лендинге макета сайта много повторяющихся элементов и при верстке макета я во многих местах переписывал один и тот же код. В исправленном варианте я постарался вынести как можно больше общих деталей в общие классы (в коде «CSS» файла я постарался вынести их отдельно для наглядности) чтобы свести к минимуму копирайтинг кода.

1. Трудно читаемый код из-за множества пересекающихся между собой CSS свойств (шрифтов, размеров, отступов и т.д.).

Для исправления этого я не только выделил общие свойства в общие классы, но и поменял структуру наименования переменных. (применил структуру «БЭМ» для «html» и «CSS»), а также постарался записать стили всех блоков и элементов отдельно друг от друга, используя классы.

1. **Принцип "KISS"**

Ошибки, которые были мной выявлены:

1. Непонятный и запутанный «HTML» и «CSS» код в котором трудно разобраться. В частности, ошибки в именовании элементов, такие как:
   1. Неправильное наименование элементов, блоков и т.д.
   2. Использование большого количества несемантических тегов.
   3. Пропускалось название классов или id-селекторов некоторых элементов, что привело к созданию очень запутанных цепочек связей стилей без конкретных имен, таких как:

nav ul li {….};

. header > p {…};

. section,. button,. content-main {…} и т.п.

Для исправления этого, была переработана HTML и CSS верстка по методологии БЭМ. Постарался написать стили без цепочек вложенностей. Сменил названия классов на более понятные и точные.

1. Принцип “YAGNI”

Ошибки, которые были мной выявлены:

Как таковых лишних частей кода для удаления я не обнаружил, все делалось согласно макету и доп. функционал не реализовывался. Однако был переработан подход к верстке некоторых элементов, поэтому как в HTML, так и в CSS также было удалено некоторое количество лишнего кода, а также, они были объединены в общие классы что сократило объем и сделало код более компактным и удобным в понимании его алгоритма работы.

1. Принцип SOLID

Single Responsibility Principle

Ошибки, которые были мной выявлены:

1. Из-за беспорядочного наименования классов элементов, все было перемешано в кучу и было трудно понять, где у элемента зона действия, т.к. он мог использоваться с различными местами и с разным функционалом.

Согласно принципу, с помощью БЭМ постарался реализовать у классов подобие зоны ответственности, т.е. чтобы каждый класс отвечал за свой элемент и функционал, четко разграничивая их, что помогло сделать код более понятным и удобным для изучения.

Проверка кода на семантику и валидность

После написания, код был проверен на семантику и валидность верстки на сайтах «https://jigsaw.w3.org/css-validator/validator» и «https://validator.w3.org/nu/#file»

Ниже представлены скрины результатов.

Используемые линтеры

1. Prettier – code formatter v 9.10.3
2. Beautify css/sass/scss/less v.2.3

