CSS-правила

Итак, у нас есть заготовка сайта, точнее его каркасная верстка. Нам необходимо приступить к визуальному оформлению HTML и для этого нужно подробнее узнать CSS.

CSS — это язык для оформления структурированных документов, например, HTML- документов. Синтаксис CSS незамысловат: это плоский список CSS-правил. CSS-правило состоит из селектора и перечня свойств и их значений:

```
селектор {
  свойство1: значение1;
  свойство2: значение2;
}
```

Комментарии в CSS выглядят как символы /* и */. Вот пример CSS-правила

```
header {
  padding-top: 60px; /* отступ сверху */
  background-color: #283618; /* установка цвета фона через HEX-палитру */
}
```

Для начала нам необходимо подключить CSS к HTML-файлу. Для этого существует 3 способа:

1. **Inline CSS**

Вся атрибутка записывается в атрибут style. Свойства и значения, прописанные таким образом, применятся точечно к одному элементу:

```
<div style="width: 50%;"></div>
```

Обычно использование этого способа считается плохой практикой. Но иногда в виде исключения бывает удобнее воспользоваться встраиванием стилей в атрибут style, чем писать отдельные CSS-правила.

2. **Global CSS**

Также считается плохой практикой. Атрибутика будет храниться в самом документе .html. При использовании глобальных стилей свойства CSS описываются и располагаются в заголовке веб-страницы. В данном случае получается хранить стили в одном месте прямо на той же странице с помощью контейнера <style>:

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Глобальные стили</title>
    <style>
      h1 {
```

```
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
    }
   </style>
 </head>
 <body>
   <h1>Hello, world!</h1>
 </body>
</html>
```

3. Связанные стили

Самое правильное и общественно признанное использование. При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением css, а для связывания документа с этим файлом применяется тег <link>. Данный тег помещается в контейнер <head>:

```
<link rel="stylesheet" href="style.css">
```

Приступим к оформлению. Оформлять будем наши предыдущие заготовки из практической работы #1 и #2.

Задание

1. Откройте папку с index.html, созданную на прошлых занятиях.
2. Создайте новый текстовый документ style, расширение смените на .css
3. Откройте его и добавьте в него следующее:

```
*{
  padding: 10px;
  border: 1px solid green;
}
```

4. В файл index.html внутри тега <head> добавьте строк подключения файла стилей:

```
<link rel="stylesheet" href="style.css">
```

5. Откройте веб-страницу в браузере (либо обновите ее)

Селекторы

Селектор находится в начале CSS-правила, до фигурных скобок, и определяет к каким HTML-элементам применятся свойства и значения из правила. Вспомните пример:

```
header {
  padding-top: 60px; /* отступ сверху */
  background-color: #283618; /* установка цвета фона через HEX-палитру */
}
```

Строка header — это селектор. Она говорит браузеру применить список свойств (отступ и фоновый цвет) ко всем тегам header на странице.

Простейшие (и самые популярные) селекторы — это селекторы по тегам и по классам. Селекторы по тегам содержат имя тега без символов < и > и применяются ко всем подходящим тегам. Селекторы по классам начинаются с точки, за которой идёт имя класса, и применяются ко всем тегам с подходящим атрибутом class. Например:

```
h1 {
  color: red; /* выберет все заголовки 1 уровня и цвет шрифта изменит на красный */
}
.info {
color: blue; /* выберет элементы с классом info, цвет шрифта сменит на синий */
}
```

Для задания элементу HTML класса необходимо в файле index.html добавлять тегам атрибут class и присваивать ему необходимое значение:

```
<h1 class=”my-title”>Сайт ученика группы БЛА_БЛА_БЛА</h1>
```

Если у CSS-правил отличаются только селекторы, а свойства и значения одинаковые, то их можно сгруппировать через запятую. Например:

```
h1, .danger {
  color: red;
}

/* То же самое, что и */

h1 {
  color: red;
}
.danger {
  color: red;
}
```

А сейчас попробуйте использовать несколько разных типов селекторов. Подставляйте в CSS-правило селектор и смотрите, что будет меняться.

Задание

1. Очистите CSS-файл, пока не заболели глаза
2. Используя селекторы по тегам и классам задайте рамку 6 элементам страницы, по три на каждый. Установка рамки выполняется строкой:

border: 1px solid green;

Первое значение здесь - толщина в пикселях
Второе - тип линии (может быть так же dashed, dotted, double и т.д.)
Третье - цвет линии (используйте свои познания в английском, либо гуглите HEX, RGB палитру)

Свойства и значения

Список свойств и значений находится внутри фигурных скобок CSS-правила. Пары «свойство-значение» отделяются друг от друга точкой с запятой, а свойство от значения отделяется двоеточием. Свойство определяет, какую характеристику внешнего вида мы хотим изменить, а значение — как именно. Ещё раз вспомните пример:

```
header {
  padding-top: 60px; /* отступ сверху */
  background-color: #283618; /* установка цвета фона через HEX-палитру */
}
```

В примере свойству padding-top задаётся значение 60px, а свойству background-color значение #283618. В итоге, это CSS-правило задаёт всем элементам с тегом header отступ сверху и фоновый цвет. С помощью CSS можно задавать параметры отображения любого тега: ширину и высоту, отступы, цвет и размер шрифта, фон и так далее. Каждый раз, когда мы добавляем новое свойство или изменяем его значение, мы меняем что-то на странице. Давайте попробуем добавить несколько новых свойств и посмотрим, что изменится на странице.

Задание

1. Добавим тегу footer класс - например “page-footer”
2. Перейдем в файл стилей и обратимся к нему. Запись начинается с точки, дальше имя класса
3. Изменим такие атрибуты, как:

a. padding - отвечает за внутренние отступы элемента. Задайте значение 20px

b. background-color - цвет фона. Задайте что-нибудь потемнее. Можно использовать название цвета на английском (red, green, т.п.), HEX-палитру (#283618), RGB/RGBa цвет (rgb(221, 161, 94)). Выбор цвета по палитре можно производить с помощью Paint

4. Зададим цвет шрифта всего документа с помощью свойства color со значением white, либо #fff. Изменять его следует с помощью селектора по тегу body
5. Сохраните изменения, обновите страницу в браузере

Наследование

На прошлом шаге мы задали белый цвет текста для body, а он применился и для остальных элементов с текстом. Почему это произошло? Всё дело в наследовании. Наследование в CSS — это механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам.

Стили, присвоенные одному элементу, наследуются всеми потомками (вложенными элементами), но только в том случае, если они где-то явно не переопределены. Например, размер шрифта и его цвет достаточно применить к body, чтобы большинство элементов внутри имели те же свойства. Рассмотрим пример наследования:

```
body {
  font-size: 14px;
}
nav {
  font-size: 18px;
}
```

Размер шрифта у всего текста на странице, кроме текста внутри навигации, станет равен 14px. У nav есть своё объявленное значение размера шрифта (18px), и оно будет использоваться вместо наследуемого от body значения (14px). А ещё 18px станет новым наследуемым значением для потомков nav. Если на странице из примера будут заголовки, то их размер тоже будет отличаться от 14px.

Наследуемые свойства

На самом деле не все свойства в CSS наследуются. К наследуемым относятся в основном свойства, определяющие параметры отображения текста:

font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction и другие.

Также к наследуемым свойствам относятся list-style, cursor, visibility, border-collapse и некоторые другие. Но они используются значительно реже. Наследуемые свойства можно и нужно задавать через предков, следуя структуре документа.

Например, параметры текста зачастую не меняются в пределах крупных блоков страницы: меню, основного содержания, информационных панелей. Поэтому общие параметры текста (цвет, размер, гарнитура) обычно указывают в стилях этих крупных блоков. Давайте проверим работу ещё пары наследуемых свойств, а заодно добавим больше «воздуха» в тексты блога и сделаем их более удобными для чтения.

Задание

1. Для body измените значение свойства font-size на 16px
2. Затем значение свойства line-height на 26px и посмотрите, что изменения отразились на всех элементах.

Ненаследуемые свойства

В предыдущем задании мы перечислили основные наследуемые свойства. Но не все свойства наследуются. Основные ненаследуемые свойства — это параметры позиционирования, размеров, отступов, фона, рамок:

background, border, padding, margin, width, height, position и другие.

Не наследуются они из соображений здравого смысла. Например, если для какого-либо блока установлен внутренний отступ, автоматически выставлять такой же отступ каждому вложенному элементу нет никакой надобности. Эти параметры чаще всего уникальны для каждого отдельного блока.

На первом этапе мы вводили стиль:

```
*{
  padding: 10px;
  border: 1px solid green;
}
```

Таким образом мы увидели рамки каждого элемента (селектор * - выделение всех элементов). Теперь возьмём это же свойство border и убедимся, что оно не наследуется.

Задание

1. Добавьте свойство border для элемента main со значением 5px solid #2d508f
2. Сохраните, обновите страницу. Проверьте что свойство изменено только у конкретного элемента, следовательно оно не наследуется на его дочерние элементы

Составные свойства

В CSS есть обычные свойства, управляющие одним параметром отображения, и есть составные свойства, управляющие одновременно несколькими параметрами.

Например, свойство font-size — обычное, оно управляет только размером шрифта. А свойство font — составное, оно задаёт сразу шесть параметров: размер и название шрифта, высоту строки и некоторые другие. Браузер всегда «расшифровывает» составные свойства в обычные. Например, такое составное свойство:

```
font: 16px/26px "Arial", sans-serif;
```

Браузер «расшифрует» в такой набор обычных свойств и их значений:

```
font-size: 16px; /* было задано в font */
line-height: 26px; /* было задано в font */
font-family: "Arial", sans-serif; /* было задано в font */
font-weight: normal; /* не было задано в font */
font-style: normal; /* не было задано в font */
font-variant: normal; /* не было задано в font */
```

Если значение обычного свойства не было задано в составном, то браузер при «расшифровке» использует исходное значение этого свойства. В примере значение 16px для font-size взято из font, а для font-weight использовано исходное значение — normal.

Составное свойство всегда задаёт значения всем своим компонентам. Для не заданных явно компонентов используются исходные значения. Поэтому составные свойства нужно использовать с осторожностью. Например, если забыть описать высоту строки:

```
font: 16px "Arial", sans-serif;
```

То для line-height браузер возьмёт исходное значение, и внешний вид текста может оказаться плохим.

Пора добавить странице блога красивый фон. Используем для этого составное свойство background.

Задание

1. Зайдите в Paint и создайте изображение размером, например 800x600. Произвольный рисунок
2. Сохраните его в папку с проектом, название файла сделайте bg.png
3. Для body задайте составное свойство background со значением #ffffff url("bg.png") no-repeat top center

Типы значений: абсолютные и относительные

Единицы измерения в CSS делятся на абсолютные и относительные. Абсолютные единицы измерения привязаны к настоящим физическим размерам и связаны между собой жёсткими пропорциями. Примеры абсолютных единиц измерения:

```
font-size: 1cm; /* 1 сантиметр */
font-size: 10mm; /* 10 миллиметров */
font-size: 38px; /* 38 пикселей */
```

Пиксели, px, используют чаще всего, остальные абсолютные единицы почти не применяют.

Относительные единицы измерения описывают значения, которые зависят от других значений. Например, ширина элемента в процентах зависит от ширины родительского элемента, а ширина элемента в em зависит от размера шрифта самого элемента.

К относительным единицам относятся:

em, rem, vh, vw, проценты и некоторые другие.

Каждая из таких единиц решает свой круг задач. Например, проценты используют для «резиновой» вёрстки, а em применяют в вёрстке государственных сайтов с особыми дополнительными требованиями к масштабированию текста.

Задание

1. Смените размер h1 на 1em
2. Теперь на 2em

Стили по умолчанию

Некоторым элементам можно не задавать никаких стилей, но у них всё равно будет какое-то оформление. Например, списки «без стилей» выглядят так:

- первый пункт списка,
- второй пункт списка,
- третий пункт списка.

У списка есть отступы и маркеры, но откуда они берутся? Потому что список такой «сам по себе»? Нет! Параметры оформления тегов описываются только в CSS, и наши списки — не исключение. Значит где-то есть стили, в которых спискам заданы отступы и маркеры? Да! И эти стили хранятся внутри браузера, это браузерные стили по умолчанию.

Например - при изменении размера шрифта body, данный шрифт не будет наследоваться заголовками. Это происходит из-за того, что размер шрифта заголовков явно задан внутри браузерных стилей по умолчанию (и наследуемое от body значение им не нужно).

Задание

1. Приведите свой список описаний dl/dt/dd в следующий вид:

```
<dl class="skills">
        <dt>HTML</dt>
        <dd>60%</dd>
        <dt>CSS</dt>
        <dd>20%</dd>
        <dt>JS</dt>
        <dd>10%</dd>
 </dl>
```

Каскадирование

Вот мы и подобрались к одному из самых важных механизмов CSS — каскадности. Именно он скрывается за первой C в аббревиатуре CSS (Cascading Style Sheets).

Когда браузер отрисовывает страницу, он должен определить итоговый вид каждого HTML-элемента. Для этого он собирает все CSS-правила, которые относятся к каждому элементу, ведь на элемент могут влиять сразу несколько CSS-правил. Например:

```
<p class="beloved-color">Зелёный - мой любимый цвет</p>
```

На этот элемент могут одновременно влиять CSS-правила по тегу p и по классу beloved-color из наших стилей, да ещё и CSS-правило по тегу p из браузерных стилей.

После того, как все правила для элемента собраны, браузер комбинирует все свойства из этих правил и применяет их к элементу. Если в наших стилях есть такой код:

```
p {
  font-size: 14px;
}
.beloved-color {
  color: green;
}
```

То у нашего абзаца про цвет будет такой итоговый набор свойств и значений:

```
font-size: 14px; /* из правила для p в наших стилях */
color: green; /* из правила для .beloved-color в наших стилях */
margin: 1em 0; /* из правила для p в браузерных стилях */
```

Этот механизм комбинирования стилей из разных источников в итоговый набор свойств и значений для каждого тега и называется каскадностью.
Давайте продолжим оформление списка навыков, а заодно применим каскад.

Задание

1. Добавим правило для тега dd, вложенного в класс .skills:
   ```
   .skills dd {
     background-color: #e8e8e8;
   }
   ```
2. Затем в это правило добавьте составное свойство margin со значением 0
3. Ниже свойства margin добавьте свойство margin-bottom со значением 10px, переопределив нижний отступ

Конфликт свойств

На один элемент могут действовать несколько CSS-правил. Если в этих правилах есть одинаковые свойства с разными значениями, то возникает конфликт. Например:

```
ul {
  list-style: disc;  /* браузерные стили */

}
.blog-navigation ul {
  list-style: none; /* наши стили */
}
```

На один и тот же список в нашем блоге действуют стили с разными значениями свойства list-style. Это и есть конфликт, ведь у свойства может быть только одно значение.

Браузеру нужно как-то решать, какими будут итоговые значения конфликтующих свойств. Конфликт разрешается максимум за три шага. Если на текущем шаге определиться не удалось, то выполняется следующий шаг. Вот эти шаги:

1. Сравниваются приоритеты стилевых файлов, в которых находятся конфликтующие свойства. Например, авторские (то есть наши) стили приоритетнее браузерных.
2. Сравнивается специфичность селекторов у CSS-правил с конфликтующими свойствами. Например, селектор по классу более специфичен, чем селектор по тегу.
3. Побеждает то свойство, которое находится ниже в коде.

Вспомните margin: 0; для .skills dd на прошлом шаге. Свойство из наших стилей (более приоритетных) вступило в конфликт со свойством из браузерных стилей и победило, обнулив отступы.

Другой конфликт возник в правиле .skills dd при появлении там margin-bottom:

```
margin: 0;
margin-bottom: 10px;
```

Каскад работает и внутри CSS-правил, поэтому в конфликт вступили «обычный» margin-bottom и аналогичный компонент составного свойства. «Обычное» свойство победило, так как находится ниже в коде:

```
margin-left: 0; /* из составного свойства */
margin-top: 0; /* из составного свойства */
margin-right: 0; /* из составного свойства */

margin-bottom: 0; /* из составного свойства */
margin-bottom: 10px;
```

А сейчас давайте превратим проценты освоения навыков в шкалы прогресса. Для этого придётся добавить в разметку дополнительные декоративные обёртки, и лучше всего для этого подходит уже знакомый тег .

Задание

1. В каждом теге dd оберните содержимое в с классом skills-level.
2. В свою очередь классу .skills-level задайте следующее оформление:
    a. font-size: 12px; /* размер шрифта*/
    b. text-align: center; /*выравнивание текста*/
    c. color: #ffffff; /*цвет текста*/
    d. background-color: #4470c4; /*цвет фона*/

Множественные классы

Разберём небольшой пример. Допустим, у вас на сайте есть разные типы уведомлений:

```
<div class="alert">Прост сообщение.</div>
<div class="alert alert-error">Ошибка!</div>
```

У них все стили кроме фона одинаковые. Фон обычных сообщений серый, фон ошибок красный. Стили для этих уведомлений можно организовать так:

```
.alert {
  background-color: lightgrey; /* свойства для рамок, отступов и так далее */
}
.alert-error {
  background-color: red;
}
```

Класс alert с общими стилями есть у всех сообщений. Класс alert-error с частными стилями есть только у ошибок. Помните: несколько классов в атрибуте class задаются через пробел. Почему код лучше организовать именно так? Мы уже знаем, как работает каскад в CSS, поэтому ответить будет легко.

Использование нескольких классов в разметке помогает явно указать, какие стили будут смешиваться с помощью каскада. И, конечно, эти стили удобнее всего разместить в коде рядом друг с другом.

Использование однотипных селекторов (например, по классам) позволяет использовать самый простой механизм разрешения конфликтов — по порядку в коде. Размещаем частные CSS-правила после общего, и всё что нужно точно переопределится.

А теперь уже с полным пониманием происходящего давайте улучшим блок навыков: сделаем шкалу с высоким процентом зелёной. Для этого в разметку добавим ещё один класс, а в стили — CSS-правило для этого класса, причём после общего CSS-правила.

Задание

1. В первом теге dd диву с классом skills-level добавьте второй класс skills-level-ok
2. В стилях создайте правило для селектора .skills-level, в нем цвет фона измените на #47bb52
3. Чтобы окончательно привести наши полосы скилов в подобающий вид произведем применение стилей инлайновым методом, описанным в первом пункте. Для этого в файле index.html зададим:
    a. Первому диву с классом skills-level атрибут style со значением width: 60%;
    b. второму диву — со значением width: 20%;
    c. а третьему диву — со значением width: 10%;

Конспект: основы CSS

CSS-правила

CSS — это язык для оформления структурированных документов, например, HTML- документов. Синтаксис — это плоский список CSS-правил. CSS-правило состоит из селектора и перечня свойств и их значений:

```
селектор {
  свойство: значение;
  свойство: значение;
}
```

Для комментариев в CSS используются символы /* и */.

Селекторы

Селектор находится в начале CSS-правила, до фигурных скобок, и определяет, к каким HTML-элементам применятся свойства и значения из правила.

```
header {
  padding-top: 60px;
}
```

Простейшие (и самые популярные) селекторы — это селекторы по тегам и по классам. Селекторы по тегам содержат имя тега без символов < и > и применяются ко всем подходящим тегам. Селекторы по классам начинаются с точки, за которой идёт имя класса, и применяются ко всем тегам с подходящим атрибутом class.

```
h1 { color: red; }
.info { color: blue; }
```

На странице может быть несколько списков, и стили применятся ко всем спискам, даже к тем, которые вы менять не хотели. Чтобы избежать таких ситуаций, лучше не использовать селекторы по тегам или использовать их как можно реже.

Если у CSS-правил отличаются только селекторы, а свойства и значения одинаковые, то их можно сгруппировать через запятую.

Также можно комбинировать любые типы селекторов через пробел. Такие селекторы называются вложенными или контекстными и читаются справа налево. Например:

```
nav a {…}
  .menu ul {…}
  .post .title {…}
```

Свойства и значения

Список свойств и значений находится внутри фигурных скобок CSS-правила. Свойство определяет, какую характеристику внешнего вида мы хотим изменить, а значение — как именно.

```
header {
  padding-top: 60px; /* отступ сверху */
```

}

Каждый раз, когда мы добавляем новое свойство или изменяем его значение, мы меняем что-то на странице.

Наследование

Наследование в CSS — это механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам. Стили, присвоенные одному элементу, наследуются всеми потомками (вложенными элементами), но только в том случае, если они где-то явно не переопределены.

Составные свойства

В CSS есть обычные свойства, управляющие одним параметром отображения, и есть составные свойства, управляющие одновременно несколькими параметрами. Например, свойство font. Оно задаёт сразу шесть параметров: размер и название шрифта, высоту строки и некоторые другие.

font: 16px/26px "Arial", sans-serif;

Если значение обычного свойства не было задано в составном, то браузер при «расшифровке» использует исходное значение этого свойства.

Типы значений: абсолютные и относительные

Абсолютные единицы измерения привязаны к настоящим физическим размерам и связаны между собой жёсткими пропорциями. Пиксели, px, используют чаще всего, остальные абсолютные единицы почти не применяют. Примеры абсолютных единиц измерения:

font-size: 1cm;

font-size: 10mm;

font-size: 38px;

Относительные единицы измерения описывают значения, которые зависят от других значений. Например, ширина элемента в процентах зависит от ширины родительского элемента, а ширина элемента в em зависит от размера шрифта самого элемента. К относительным единицам относятся em, rem, vh, vw и некоторые другие, ну и, конечно же, проценты.

Стили по умолчанию

Некоторым элементам можно не задавать никаких стилей, но у них всё равно будет какое-то оформление. Например, у списка <ul> есть отступы и маркеры. Такие стили называются стилями по умолчанию и задаются внутри браузерных стилей изначально. Их можно переопределить или сбросить, задав другие значения свойств элементу.

Каскадирование

Когда браузер отрисовывает страницу, он должен определить итоговый вид каждого HTML-элемента. Для этого он собирает все CSS-правила, которые относятся к каждому элементу, ведь на элемент могут влиять сразу несколько CSS-правил. Механизм комбинирования стилей из разных источников в итоговый набор свойств и значений для каждого тега называется каскадностью. Например, есть такой элемент в разметке:

Зелёный - мой любимый цвет

Заданные стили:

.beloved-color { color: green; }

Браузерные стили:

margin: 1em 0;

Итоговые стили:

color: green;

margin: 1em 0;

Конфликт свойств

На один элемент могут действовать несколько CSS-правил. Если в этих правилах есть одинаковые свойства с разными значениями, то возникает конфликт. Например:

ul { list-style: disc; }

.blog-navigation ul { list-style: none; }

Браузеру нужно как-то решать, какими будут итоговые значения конфликтующих свойств. Конфликт разрешается максимум за три шага. Если на текущем шаге определиться не удалось, то выполняется следующий шаг. Вот эти шаги:

1. Сравниваются приоритеты стилевых файлов, в которых находятся конфликтующие свойства. Например, авторские (то есть наши) стили приоритетнее браузерных.
2. Сравнивается специфичность селекторов у CSS-правил с конфликтующими свойствами. Например, селектор по классу более специфичен, чем селектор по тегу.
3. Побеждает то свойство, которое находится ниже в коде.

Каскад работает и внутри CSS-правил.

Встраивание и подключение внешних стилей

Внешние стили подключаются через тег <link>

<link rel="stylesheet" href="style.css">

Встраивание стилей в тег <style>. Его обычно размещают внутри <head>:

```
<head>
  <style>
    CSS-код
  </style>
</head>
```

Такой способ используется для оптимизации загрузки страницы, ведь в таком случае браузер не будет отправлять дополнительных запросов на сервер.

Встраивание в атрибут style:

Свойства и значения, прописанные таким образом, применятся точечно к одному элементу.

Обычно использование этого способа считается плохой практикой. Но иногда в виде исключения бывает удобнее воспользоваться встраиванием стилей в атрибут style, чем писать отдельные CSS-правила. Например, когда нужно управлять стилями именно из разметки, и создавать отдельные классы при этом будет излишне.

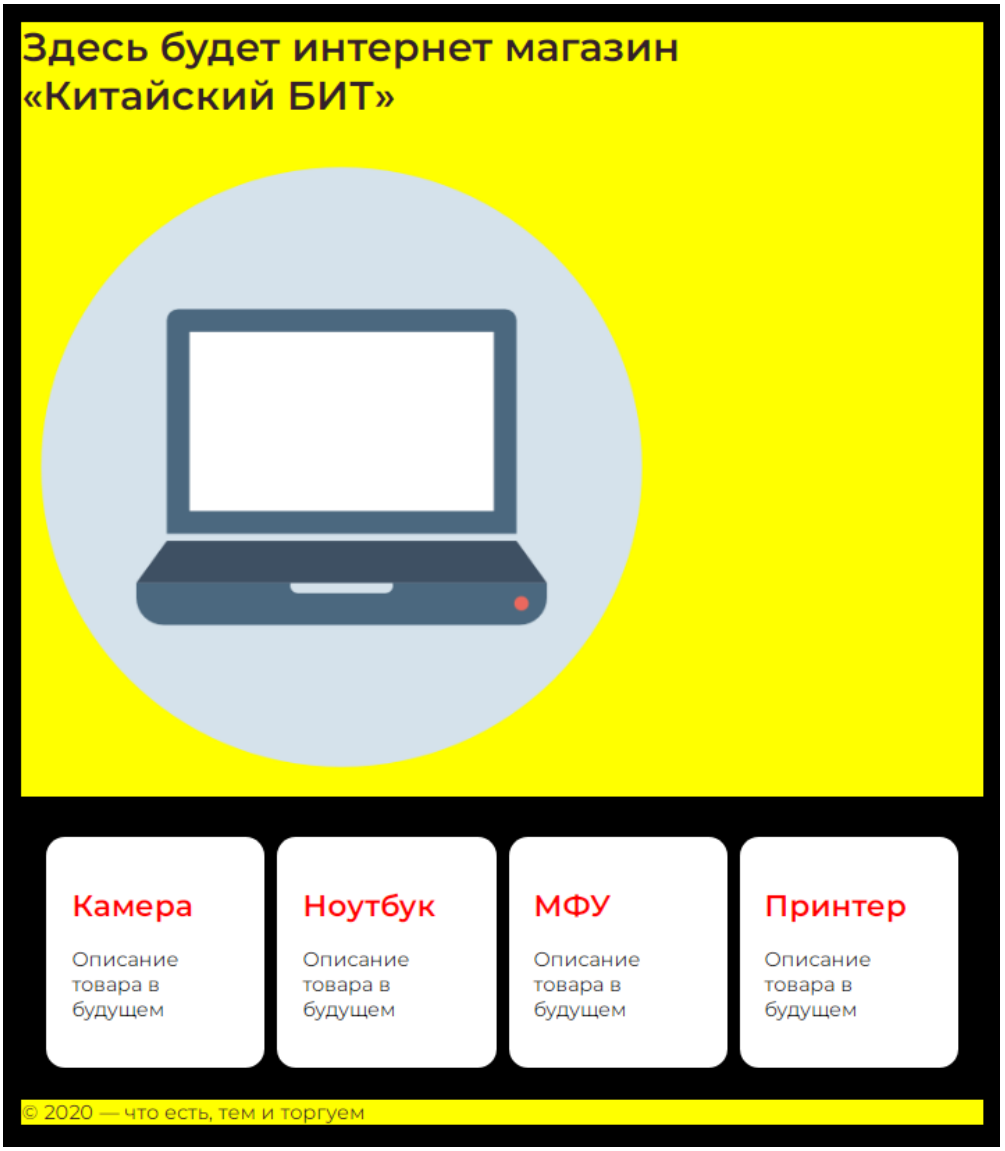
Испытание: прототип интернет-магазина.

В этом испытании вам нужно оформить разметку. Иначально папка содержит 3 файла:

- index.html - здесь хранится вся необходимая разметка
- main-style.css - основные стили страницы, которые нам нет необходимости менять
- logo.png - картинка, использованная на странице

Нам необходимо внести следующие изменения:

1. Необходимо создать файл style.css и подключить его
2. добавить атрибут width нашему изображению. Очень похоже на инлайн стиль, но это именно атрибут тега img. Значение 150
3. изменить цвет фона для шапки и подвала с вырвиглаз-желтого на #8aea92
4. сменить цвет заголовков в карточках будущих товаров с красного на #80ada0
5. добавить дополнительные классы каждой из карточек(уникальный), а затем задать им разные цвета шрифтов и фона через этот класс
6. Итого из этого:



Должно получиться примерно это:

Здесь будет интернет магазин «Китайский БИТ»

Камера

Описание товара в будущем

Ноутбук

Описание товара в будущем

МФУ

Описание товара в будущем

Принтер

Описание товара в будущем