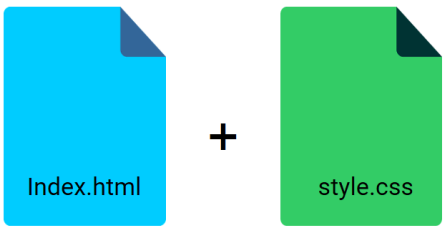
Подключение CSS
Создаем CSS-файл и подключаем его к HTML-файлу

Все HTML-файлы имеют расширение .html. Например:

- index.html
- contacts.html
- page-2.html

Точно также для CSS-стилей создаётся отдельный файл, только с расширением .css. Например, style.css.

На картинке ниже показан пример, как можно подключить файл style.css к файлу index.html.




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

Как видите, в теге <head> нам необходимо написать следующее:

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

- <link> - это специальный тег, который используется для подключения CSS-стилей.
- rel = "stylesheet" - это аттрибут rel со значением "stylesheet", что значит таблица стилей. То есть таким образом мы говорим браузеру, что мы хотим подлючить таблицу стилей.
- href = "style.css" - в аттрибуте href мы прописали путь к файлу style.css.

Шаг 1: Создадим 2 файла: index.html и style.css.

Структура файла index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Название страницы</title>
</head>
<body>
  <p>Утром надо быть особенно осторожным. Одно неловкое движение - и ты снова спишь.</p>
```

```
    <p>Вот так всегда. Запланируешь 20 дел. Сделаешь 30. И ни одного из 20 запланированных.</p>
</body>
</html>
```

Структура файла style.css

```
p {
    font-family: Georgia;
    font-size: 18px;
    color: green;
}
```

Шаг 2: Подключаем файл style.css к файлу index.html.
Для этого в файле index.html добавьте тег <link> с необходимыми аттрибутами:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Название страницы</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <p> Утром надо быть особенно осторожным. Одно неловкое движение - и ты снова спишь. </p>
    <p>Вот так всегда. Запланируешь 20 дел. Сделаешь 30. И ни одного из 20 запланнированных. </p>
</body>

</html>
```

Шаг 3: Открываем файл index.html в браузере. В браузере Вы увидите 2 текста параграфов, написанные:

- шрифтом: Georgia
- размером шрифта: 18px
- цветом: зеленым

Виды шрифтов

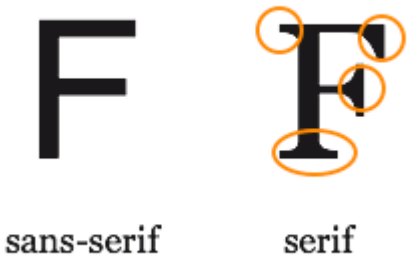
Шрифты - важная часть вида веб-страницы. Существует множество различных, не похожих друг на друга шрифтов.

Тем не менее, все шрифты можно разделить на 4 группы:

- с засечками (**serif**)
- без засечек (**sans-serif**)
- рукописные (**handwritten**)
- моноширинные (**monospace**)

Группа 1 - с засечками (serif)

Давайте посмотрим на следующую картинку:



Слева мы видим шрифт без засечек, справа - с засечками.

Шрифты с засечками по-английски называются "serif" (англ. "serif" - "засечка", "выступ", "штрих"). К таким шрифтам относятся, например:

1. **Times New Roman**
2. **Georgia**
3. **Cambria**
4. **Verdena**

Шрифты из данной группы обычно **используются для длинных текстов**, поскольку считается, что их легче воспринимать благодаря засечкам.

Группа 2 - шрифты без засечек (sans-serif)

Шрифты без засечек ('"sans-serif", от фр. "sans" - "без", и англ. "serif" - "засечка").

К шрифтам без засечек относятся, например:

1. **Arial**
2. **Helvetica**
3. **Tahoma**

Шрифты без засечек **часто используются для заголовков**.

Стоит также сказать, что шрифты данной группы считаются менее читабельными, чем шрифты с засечками - хотя по этому поводу нет единого мнения.

Группа 3 - рукописные шрифты (handwritten)
Вот Вам пример рукописного шрифта:

Рукописные шрифты обычно используются **для декоративных элементов** на страницах.
Группа 4 - моноширинные шрифты
Моноширинные шрифты - это шрифты, в которых **ширина всех букв одинаковая.**
Например:

```
ЭТО МОНОШИРИННЫЙ ТЕКСТ
```

К моноширным шрифтам относятся, например:

1. **Courier New**
2. **Lucida Console**

Моноширными шрифтами **часто пишется программный код.**
Например, если Вы пользуетесь текстовым редактором Sublime Text, тогда знайте, что когда Вы набираете код в Sublime, Вы его набираете моноширным шрифтом

Зачем front-end разработчику знать типы шрифтов?
Действительно - зачем? Разве мы как разработчики не должны просто подключить шрифт, который указан в задании?

Все дело в том, что может так получиться, что человек зашел к Вам на сайт, а при разработке сайта использовали такой шрифт, который по тем или иным причинам не установлен у него на компьютере. И все, беда... Чтобы перестраховаться от таких ситуаций **необходимо использовать такое понятие как fonts fallback или просто "фоллбэк".**
Пример fallback:

```
p{font-family:"Times New Roman", Georgia, serif;}
```

То есть этой строчкой мы указали, что:

- Все параграфы должны отображаться шрифтом Times New Roman.

- Если же вдруг у пользователя нет такого шрифта на компьютере, тогда все тексты параграфов необходимо отображать шрифтом Georgia.
- Если же и этот шрифт не найдется на компьютере пользователя, тогда отображать любым стандартным шрифтом, относящимся к группе шрифтов с засечками (serif).

text-transform

Можно заметить, что часто заголовки пишутся прописными буквами. А в английском языке в названиях (например, книг, сериалов и пр.) первые буквы всех слов прописные:

- **G**eneral **E**lectric
- **G**ame of **T**hrones
- **M**icrosoft **O**ffice

Или, например, у Вас может возникнуть необходимость отобразить часть текста прописными буквами, а часть большими буквами.

Для примера возьмем параграф из Википедии:
Слово *компьютер* является производным от английских слов to compute, computer, которые переводятся как «вычислять», «вычислитель» (английское слово, в свою очередь, происходит от латинского computāre — «вычислять»). **ПЕРВОНАЧАЛЬНО В АНГЛИЙСКОМ ЯЗЫКЕ ЭТО СЛОВО ОЗНАЧАЛО ЧЕЛОВЕКА, ПРОИЗВОДЯЩЕГО АРИФМЕТИЧЕСКИЕ ВЫЧИСЛЕНИЯ С ПРИВЛЕЧЕНИЕМ ИЛИ БЕЗ ПРИВЛЕЧЕНИЯ МЕХАНИЧЕСКИХ УСТРОЙСТВ.** В дальнейшем его значение было перенесено на сами машины, однако современные компьютеры выполняют множество задач, не связанных напрямую с математикой.
Именно для того, чтобы поменять регистр текста (из прописных в заглавные, из заглавных в прописные) и используется свойство **text-transform**.
У text-transform есть 5 значений:

- **text-tranform: capitalize** (Первая Буква Каждого Слова Будет Большой)
- **text-tranform: lowercase** (все буквы будут маленькими)
- **text-transform: uppercase** (ВСЕ БУКВЫ БУДУТ БОЛЬШИМИ)
- **text-transform: none** (отменяет предыдущие форматирование)
- **text-tranform: inherit** (наследует форматирование родителя)

Рассмотрим как это работает на практике. Представим, что у нас есть следующая HTML-страница:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset = "utf-8">
  <title>My Page</title>
  <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <p>A paragraph with some TEXT</p>
</body>
</html>
```

Как видите, стили у нас подключены отдельным файлом style.css

Итак, пока никаких CSS-стилей мы не задали, текст в теге <p> будет выглядеть так:

A paragraph with some TEXT

ПРИМЕР 1

Чтоб **сделать весь текст в теге <p> заглавными буквам**и, напишите в файле style.css следующее:

```
p {
    text-transform: uppercase;
}
```

Получим:

A PARAGRAPH WITH SOME TEXT

ПРИМЕР 2

Чтоб сделать **все буквы прописными**, напишите следующее:

```
p {
    text-transform: lowercase;
}
```

Получим:

a paragraph with some text

ПРИМЕР 3

Чтобы сделать **все первые буквы заглавными**, напишите:

```
p {
    text-transform: capitalize;
}
```

Получим:

A Paragraph With Some TEXT

Обратите внимание: слово, написанное полностью заглавными буквами, не изменилось.

ПРИМЕР 4

Теперь протестируем text-transorm: none.

Но в таком виде его тестировать не интересно. Давайте представим, что у нас еще есть **h1**:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset = "utf-8">
   <title>My Page</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <h1>Heading</h1>
   <p>A paragraph with some TEXT</p>
</body>
</html>
```

Поменяем и файл со стилями. Давайте зададим **text-transform: uppercase** для всего **body**:

```
body {
   text-transform: uppercase;
}
```

В итоге получим:

Но что если мы не хотим, чтобы текст параграфа **<p>** не менялся? Пишем:

```
body {
   text-transform: uppercase;
}

p {
   text-transform: none;
}
```

Получаем:

ПРИМЕР 5

А теперь представим, что у нас есть несколько параграфов:

```
<!DOCTYPE html>
<html>
```

```
<head>
   <meta charset = "utf-8">
   <title>My Page</title>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <p>A paragraph with some TEXT</p>
   <p>A paragraph with some TEXT</p>
   <p>A paragraph with some TEXT</p>
</body>
</html>
```

Допустим, для родительского тега body у нас задан один стиль, а для самих параграфов - другой:

```
body {
   text-transform: uppercase;
}

p {
   text-transform: none;
}
```

В итоге сейчас это выглядит так:

A PARAGRAPH WITH SOME TEXT

A PARAGRAPH WITH SOME TEXT

A PARAGRAPH WITH SOME TEXT

Но что если мы хотим, чтобы один из параграфов имел стиль родителя? Например, мы хотим его выделить таким образом?

Давайте сделаем следующее - допишем ему **text-transform: inherit**:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset = "utf-8">
   <title>My Page</title>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <p>A paragraph with some TEXT</p>
   <p style="text-transform: inherit;">A paragraph with some TEXT</p>
   <p>A paragraph with some TEXT</p>
```

```
</body>
</html>
```

Получим:

A paragraph with some TEXT

A PARAGRAPH WITH SOME TEXT

A paragraph with some TEXT

Цвета в CSS

Оформляя веб-страницу, нам часто приходится задавать цвет - цвет текста, цвет фона, и т.д. В CSS это можно делать не одним, а сразу несколькими способами.

Давайте разберем **самые распространенные** способы:
Способ 1: Указать название цвета: **blue**, **yellow**, etc.
color: red;

Способ 2: через RGB:
color: rgb(0, 0, 0);

Способ 3: с помощью HEX:
color: #4286f4;

Способ 1: Название цвета
В CSS есть около 147 цветов, которые имеют свои названия:

- red
- orange
- green
- blue
- tomato"("томатный красный")
- grey
- и т.д.

Пример 1
Создадим несколько блоков и разукрасим их в стандартные цвета:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
        <title>My Page</title>
    </head>
    <body>
        <p style="color: red" >Red</p>
        <p style="color: green" >Green</p>
        <p style="color: blue" >Blue</p>
```

```
        <p style="color: aliceblue" >AliceBlue</p>
        <p style="color: antiquewhite" >AntiqueWhite</p>
        <p style="color: aqua" >Aqua</p>
        <p style="color: blanchedalmond" >BlanchedAlmond</div>
        <p style="color: blueviolet" >BlueViolet</p>
        <p style="color: burlywood" >BurlyWood</p>
        <p style="color: cadetblue" >CadetBlue</p>
    </body>
</html>
```

Получим:

Почему тогда не пользоваться этим способом всегда?

- Дело в том, что цветов и их оттенков миллионы, а в данном способе около **147 цветов**.
- Кроме того, Вы вряд ли запомните все эти цвета. Например, **AliceBlue**, **DarkMagenta**, **HoneyDew** - все это трудно держать в голове.

А вот с помощью других способов задания цветов - RGB, RGBA и HEX - можно получить любой цвет и оттенок. Давайте их рассмотрим ниже.

Способ 2: RGB

RGB - это аббревиатура **R**ed **G**reen **B**lue. Переводится как "красный, зеленый, синий". Все дело в том, что каждый цвет можно представить как сочетание трех базовых цветов, красного, синего и зеленого:

Так, например:

- фиолетовый - это красный с синим
- желтый получается если соединить красный с зеленым

Итак, из чего состоит запись цвета в RGB? Она состоит из слова rgb, после чего записывается пара круглых скобочек, в которых через запятую записаны 3 числа, которые соответствуют цветам red, green, blue.

- Red: от **0** до **255**
- Green: от **0** до **255**
- Blue: от **0** до **255**

То есть Red (красного цвета) может быть от 0 до 255. Green (зеленого) и Blue (голубого) тоже может быть от 0 до 255. Чем больше число, тем больше соответствующего оттенка в финальном цвете. Различные сочетания Red, Green и Blue дают нам все возможные оттенки цветов, какие только вообще существую в мире. Именно так и работает RGB.

Черный цвет:

```
color: rgb(0, 0, 0);
```

говорит о том, что мы получим цвет, в котором **0** красного, **0** зеленого и **0** синего. То есть это **черный**.

Белый цвет:

```
color: rgb(255, 255, 255);
```

Как видите, все цвета у нас на максимуме (все по 255), поэтому мы получим **белый** цвет.

Красный цвет:

```
color: rgb(255, 0, 0); /* ярко-красный */
color: rgb(0, 255, 0); /* ярко-зеленый */
color: rgb(0, 0, 255); /* ярко-синий */
```

Если красный на максимум (255), а остальные по нулям, получаем **красный**.

Способ 3 - HEX

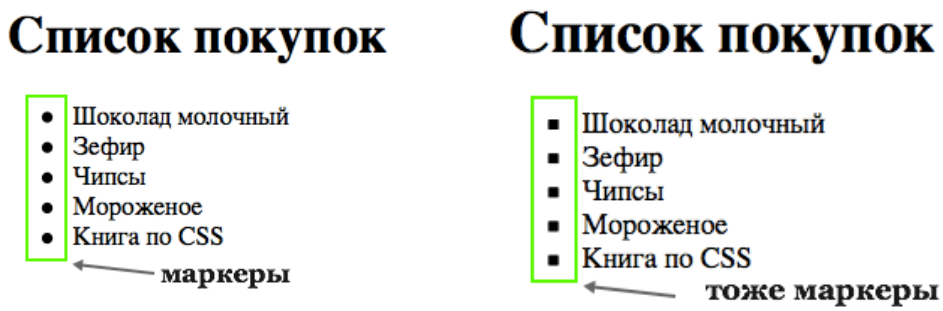
Последний способ, который мы рассмотрим - это **HEX** (обычно говорят "хекс"). Откуда название?

HEX - от англ. hexadécimal - "шестнадцатеричный". На самом деле, **HEX** - это тот же самый **RGB** (т.е. сколько в цвете красного, синего и зеленого), только записанный в другой форме - в форме шестнадцатеричного числа. Что это такое мы здесь подробно рассказывать не будем - просто знайте, что **HEX** - это тот же **RGB** только в другом формате.
HEX записывается одним числом и начинается со значка "#" (хэш):

```
color: #000000; /*черный цвет*/
color: #FFFFFF; /*белый цвет*/
color: #FFFFBF; /*лимонный цвет*/
```

Стилизация маркеров списков

Слева от пунктов списка, есть так называемые **маркеры**. В первом пример - в виде черных кружочков. Во втором - в виде черных квадратиков.



Так вот, с помощью **3-х свойств** можно менять внешний вид маркеров. Эти свойства:

- list-style-type - задает вид маркера (кружочек, квадратик, римские числа и т.д.)
- list-style-position - задает положение маркера относительно текста списка
- list-style-image -задает какое-то изображение в виде маркера

Ниже мы рассмотрим каждое свойство по порядку.

Свойство list-style-type

Как мы уже говорили, **list-style-type** используется для того, чтобы менять вид маркера. Это свойство может иметь такие значения:

Для ненумерованных списков (<ul>):

list-style-type: circle - "пустые кружочки". Например:

- Hello World!

list-style-type: disc - "зарисованные кружочки". Например:

- Hello World!

list-style-type: square - квадратный маркер. Например:

- Hello World!

Для нумерованных списков (<ol>):

decimal - "стандартная" нумерация (1, 2, 3 ...)
upper-alpha - большие латинские буквы (A, B, C ...)

lower-alpha - маленькие латинские буквы (a, b, c ...)
upper-roman - римская нумерация (I, II, III, IV, V...).
lower-roman - римская нумерация (i, ii, iii, iv, v...).
Например, попробуйте создать **ненумерованный** список и в CSS прописать следующее:

```
ul{
   list-style-type: square;
}
```

Или создайте **нумерованный** список и в CSS пропишите следующее:

```
ol{
   list-style-type: upper-roman;
}
```

Мы показали только основные возможные значения свойства **list-style-type.** На самом деле их намного больше.

Свойство list-style-position

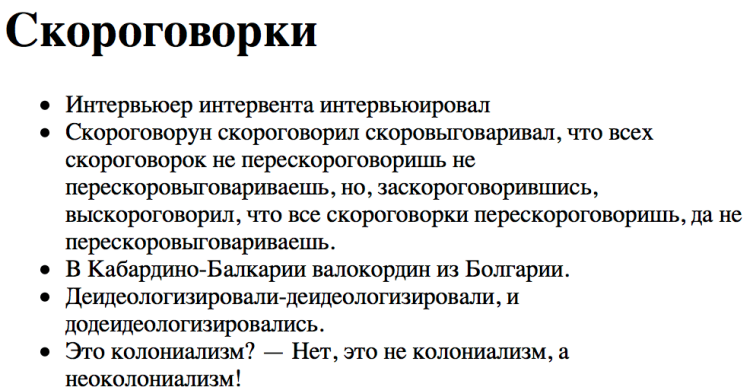
list-style-position можно перевести как "стиль списка: расположение".
Возможные значения:

list-style-position: outside

list-style-position: inside

Пример 1:
Если в CSS пропишем для списка:

```
ul{
   list-style-position: outside;
}
```



Как видите, маркеры как бы идут по внешней стороне списка.

Пример 2:
Если в CSS пропишем для списка:

```
ul{
   list-style-position:inside;
}
```

Скороговорки

- Интервьюер интервента интервьюировал
- Скороговорун скороговорил скоровыговаривал, что всех скороговорок не перескороговоришь не перескоровыговариваешь, но, заскороговорившись, выскороговорил, что все скороговорки перескороговоришь, да не перескоровыговариваешь.
- В Кабардино-Балкарии валокордин из Болгарии.
- Деидеологизировали-деидеологизировали, и додеидеологизировались.
- Это колониализм? — Нет, это не колониализм, а неоколониализм!

Как видите, маркеры как будто "утоплены" в тексте.

Свойство list-style-image

А что если нам не подходит ни один из стандартных типов маркеров? Мы можем задать свой - с помощью **list-style-image**. Например, представим, что мы хотим вот такой маркер:

Если мы вставим это изображение в оригинальном размере, оно займет пол страницы. Нам ведь это не надо?

Поэтому мы скачаем его и уменьшим его размер до нужного нам. Давайте теперь сделаем так, чтобы CSS подставил наше изображение в качестве маркера! Пропишем:

```
ul{
        list-style-image: url(image.png);
}
```

Получим:

Скороговорки

- Интервьюер интервента интервьюировал
- Скороговорун скороговорил скоровыговаривал, что всех скороговорок не перескороговоришь не перескоровыговариваешь, но, заскороговорившись, выскороговорил, что все скороговорки перескороговоришь, да не перескоровыговариваешь.
- В Кабардино-Балкарии валокордин из Болгарии.
- Деидеологизировали-деидеологизировали, и додеидеологизировались.
- Это колониализм? — Нет, это не колониализм, а неоколониализм!

Тема 1.1. Структура HTML-документа. Теги и их атрибуты. Форматирование текста на HTML-странице. Семантическая разметка HTML-документа

1. HTML-редакторы.

HTML-редактор – это программа, в которой пишут «основание» для сайтов. Технически эту роль может выполнять любой текстовый редактор, даже «Блокнот». Но лучше доверить эту задачу приложению, специально созданному для работы с кодом.

Обычно в таких программах есть подсветка синтаксиса (отдельных элементов разметки), автоматическая проверка на ошибки и опечатки, да и в целом интерфейс спроектирован таким образом, чтобы в нем было удобнее работать именно с сайтами.

HTML-редакторы не так функциональны, как полноценные среды разработки, но они подходят как для новичков, так и для профессионалов, не желающих заниматься верстской страниц в громоздких приложениях.

Иногда такие программы называют HTML-компиляторами, хотя по факту в них никакой код не компилируется. HTML – это не язык программирования, а язык разметки, не требующий компиляции как таковой. Тем не менее пользователи иногда так говорят, и я не буду им противиться.

Типы редакторов

Есть два основных типа:

Текстовый. Классический вариант для тех, кто знаком с HTML-разметкой, знает теги, CSS-классы, умеет работать с контентом внутри страницы и понимает, как его оформлять. Эти навыки необходимы, так как в редакторе разработчик должен работать вручную, прописывая свойства с помощью текстовых параметров.

Визуальный. Решение для тех, кто не понимает, что представляет собой HTML и как оформлять страницы с помощью текста. Обычно содержит в себе блоки, напоминающие конструктор. Перемещая их, пользователь может «собрать» полноценный сайт, не написав и строчки кода.

Пример HTML таблицы с заголовком th

Volkswagen AG	Daimler AG	BMW Group
Audi	Mercedes-Benz	BMW
Skoda	Mercedes-AMG	Mini
Lamborghini	Smart	Rolls-Royce

Задачи для решения

На цвета

Для решения задач данного блока вам понадобятся следующие CSS свойства: color.

1. Сделайте все абзацы **<p>** красного цвета.

2. Сделайте все **<h1>** зеленого цвета.

3. Сделайте все **<h2>** голубого цвета.

4. Сделайте все **<h3>** оранжевого цвета.

На style

Для решения задач данного блока вам понадобятся следующие HTML атрибуты: style.

5. Сделайте первый на странице абзац **<p>** зеленого цвета.

6. Сделайте второй на странице абзац **<p>** красного цвета.

На ширину и высоту

Для решения задач данного блока вам понадобятся следующие CSS свойства: width, height.

7. Сделайте все абзацы **<h2>** шириной 300px.

8. Сделайте все таблицы **<table>** шириной 400px, высотой 200px.

На выравнивание

Для решения задач данного блока вам понадобятся следующие CSS свойства: text-align.

9. Поставьте все **<h1>** по центру.

10. Поставьте все **<h2>** по правому краю.

11. Сделайте так, чтобы текст в абзацах **<p>** был выровнен одновременно и по правому и по левому краю.

12. Сделайте так, чтобы во втором абзаце **<p>** текст был выровнен по центру.

13. Поставьте все **<th>** по левому краю.

14. Поставьте все **<td>** по центру.

На жирность

Для решения задач данного блока вам понадобятся следующие CSS свойства: font-weight.

15. Сделайте все **<td>** жирным.

16. Сделайте **<h1>** нежирным.

17. Сделайте одновременно **<th>**, **<h1>** и **<h2>** нежирным.

На курсив

Для решения задач данного блока вам понадобятся следующие CSS свойства: font-style.

18. Сделайте все `<h2>` курсивом.
19. Сделайте все абзацы `<p>` курсивом, а первый абзац - нет.

На размер шрифта

Для решения задач данного блока вам понадобятся следующие CSS свойства: font-size.

20. Сделайте все `<h2>` 20px.
21. Сделайте все абзацы `<p>` 15px.

На семейство

Для решения задач данного блока вам понадобятся следующие CSS свойства: font-family.

23. Сделайте для абзацев `<p>` шрифт Arial.
24. Сделайте для `<h2>` шрифт Times New Roman.
25. Сделайте для `<h3>` любой шрифт без засечек.

На межстрочный интервал

Для решения задач данного блока вам понадобятся следующие CSS свойства: line-height.

26. Сделайте межстрочный интервал для абзацев `<p>` в 30px.

На свойство-сокращение font

Для решения задач данного блока вам понадобятся следующие CSS свойства: font.

27. Закомментируйте все стили для абзацев.
28. Для `<p>` сделайте шрифт Arial, 16 пикселей, курсив, жирный, межстрочный интервал в 30px.
29. Для `<h1>` сделайте следующий шрифт: нежирный, 20 пикселей, Verdana.

На красную строку

Для решения задач данного блока вам понадобятся следующие CSS свойства: text-indent.

30. Сделайте красную строку в абзацах 30px.
31. Для второго абзаца `<p>` уберите красную строку.

На вертикальное выравнивание

Для решения задач данного блока вам понадобятся следующие CSS свойства: vertical-align.

32. Поставьте текст в таблице `<table>` по верхнему краю по вертикали.
33. Поставьте текст в `<th>` по центру по вертикали.

Повторите страницы по образцу

Задание 1. Повторите страницу по данному по образцу:

Что такое CMS

CMS - «система управления контентом» (движок) – написанная PHP-программистами основа для сайта, с помощью которой вы сможете управлять сайтом (добавлять контент, менять пункты меню и т.п.) не зная HTML и CSS.

Однако, для того чтобы сделать сайт с помощью CMS *потребуются услуги* и программиста, и дизайнера, и верстальщика. И капиталовложения.

Какие бывают cms

Бывают различные системы управления контентом: для интернет-магазинов, для блогов, для форумов и т.д.

Примеры cms

Примеры популярных CMS: Joomla, WordPress (для блогов), PhpBB (для форумов).

CMS-ки бывают *платные* и *бесплатные*.

Задание 2. Повторите страницу по данному по образцу:

Что нужно знать, чтобы делать сайты

1. **HTML**
2. ***CSS***
3. PHP
4. SQL
5. JavaScript
6. jQuery
7. Flash
8. SEO

PHP и JavaScript

Языки программирования **PHP** и **JavaScript** позволяют сделать сайт динамичным, то есть реагирующим на действия пользователя. Например, можно сделать красивую выпадающую менюшку или слайдер

Виды скриптов

Для этого пишутся скрипты (англ. ***script*** - «сценарий») - программы, позволяющие реагировать на действия пользователя. Скрипты бывают двух видов:

- *те, которые выполняются на сервере, а результат их выполнения приходит в браузер к пользователю уже в готовом виде. Это скрипты, написанные на языке **PHP**. На нем пишутся **CMS-ки** – системы управления контентом.*
- *те, которые выполняются прямо в браузере пользователя. Это скрипты, написанные на языке **JavaScript**. Они чаще всего используются для, того чтобы сделать страницу более удобной и красивой.*

Задание 3. Создайте html- и css- файлы, результат которых показан на рисунке.