**Конспект «Оформление текста»**

**Свойство font-size**

Управляет размером шрифта. Значение свойства задаёт желаемую высоту символа шрифта. Причём единицы измерения могут быть [абсолютными или относительными](https://htmlacademy.ru/courses/307/run/8).

Самая часто используемая единица измерения размера шрифта — пиксели px:

p {

font-size: 20px;

}

Но, чтобы при изменении основного размера шрифта родителя его дочерние элементы пропорционально меняли свои размеры шрифта, есть специальная единица измерения — em.

Величина 1em — это *такой же* размер шрифта, что и у родителя. Соответственно, если нужно, чтобы шрифт дочернего элемента был всегда в 2 раза больше родительского, то надо задать значение 2em:

h1 {

font-size: 2em;

}

**Свойство line-height**

Свойство управляет высотой строки или межстрочным интервалом.

По умолчанию это свойство имеет значение normal. Оно указывает браузеру, что межстрочный интервал нужно подобрать автоматически, исходя из размера шрифта. Спецификация рекомендует устанавливать его в пределах 100-120% от размера шрифта. То есть:

p {

font-size: 10px;

line-height: normal; /\* значение будет примерно 12px \*/

}

Значение normal позволяет всем нестилизованным текстам выглядеть удобочитаемо. Однако, если есть необходимость отойти от стилизации по умолчанию, line-height можно задать фиксированное абсолютное значение в px.

p {

font-size: 16px;

line-height: 26px;

}

Если нужно задать line-height относительное значение, но не такое, как normal, то значение задаётся в процентах или в виде множителя. В таком случае браузер вычисляет значение динамически в зависимости от font-size:

p {

font-size: 10px;

line-height: 150%; /\* вычисленное значение: 10px \* 150% = 15px \*/

line-height: 2; /\* вычисленное значение: 10px \* 2 = 20px \*/

}

Относительные значения более гибкие, чем абсолютные. Но для простых сайтов фиксированных font-size и line-height будет вполне достаточно.

**Свойство font-family**

В значении этого свойства указывается список названий шрифтов, перечисленных через запятую. В начале списка располагают самый желаемый шрифт, затем менее желаемый, а в самом конце списка — общий тип шрифта. Браузер проходит по списку слева направо и использует первый найденный в системе или на сайте шрифт. Если название шрифта состоит из нескольких слов, то его нужно заключать в кавычки.

body {

font-family: "PT Sans", "Arial", sans-serif;

}

**Свойство font-weight**

Свойство задаёт насыщенность или толщину шрифта. Шрифт может быть жирнее или тоньше обычного начертания. В качестве значения можно использовать ключевое слово или число. Самые часто встречающиеся значения:

* 400 или normal — обычный шрифт, значение по умолчанию;
* 700 или bold — жирный шрифт.

Например:

h1 {

font-weight: 400; /\* то же самое что и normal \*/

}

p {

font-weight: bold; /\* то же самое что и 700 \*/

}

**Свойство text-align**

Описывает, как выравнивается текст и другие инлайновые элементы (изображения, инлайн-блоки, инлайн-таблицы и другие) внутри блока по горизонтали.

Свойство может принимать следующие значения:

1. left — выравнивание по левому краю блока, это значение по умолчанию;
2. right — по правому краю блока;
3. center — по центру блока;
4. justify — по ширине блока, при этом слова в строке будут размещаться так, чтобы занять равномерно всё пространство строки (пробелы между словами в таком случае становятся неравномерными, так как браузер «растягивает» слова в строке).

Важно помнить, что свойство text-align применяется именно к самому блоку-контейнеру, внутри которого находится текстовый контент:

HTML:

<p>

Я текст внутри абзаца

</p>

CSS:

p {

text-align: center;

}

**Свойство vertical-align**

Свойством можно выравнивать *инлайновые* элементы относительно содержащей его строки. Самый простой пример — выровнять картинку <img> по вертикали в текстовой строке.

У свойства vertical-align много значений, но самые часто используемые:

1. top — выравнивание по верхнему краю строки;
2. middle — по середине строки;
3. bottom — по нижнему краю строки;
4. baseline — по базовой линии строки (значение по умолчанию).

В отличие от text-align свойство vertical-align задаётся самому элементу, а не содержащему его контейнеру:

HTML:

<p>

<img src="picture.png" alt="Я картинка">

Я текст внутри абзаца

</p>

CSS:

img {

vertical-align: middle;

}

**Свойство color**

Цветом текста можно управлять свойством color.

Цвет может быть задан в виде ключевого слова (полный список ключевых слов приводится в [спецификации](https://www.w3.org/TR/css-color-3/#svg-color)). Например:

color: black; /\* чёрный цвет \*/

color: red; /\* красный цвет \*/

color: white; /\* белый цвет \*/

Ещё один вариант указания цвета — в виде [шестнадцатеричного значения](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D1%81%D1%82%D0%BD%D0%B0%D0%B4%D1%86%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F). В этом случае цвет формируется из *красной*, *зелёной* и *синей* составляющих, заданных в виде шестнадцатеричного числа от 00 до ff. Помимо шести, цветовой код может содержать три знака, в этом случае второй символ в цветовых составляющих дублируется первым:

color: #000000; /\* чёрный цвет \*/

color: #f00; /\* красный цвет, то же что #ff0000 \*/

color: #fff; /\* белый цвет, то же что #ffffff \*/

Если не хочется иметь дело с шестнадцатеричными значениями, можно воспользоваться специальной функцией rgb, в которой указывается цвет в более привычном десятичном виде в диапазоне от 0 до 255 также в виде трёх цветовых составляющих, перечисленных через запятую:

color: rgb(0, 0, 0) /\* чёрный, то же что #000000 \*/

color: rgb(255, 0, 0) /\* красный, то же что #ff0000 \*/

color: rgb(255, 255, 255) /\* белый, то же что #ffffff \*/

У функции rgb есть расширенная версия — rgba. В этом случае помимо указания цвета последним значением указывается степень непрозрачности цвета — alpha. Значение может быть от 0 (полностью прозрачный) до 1 (полностью непрозрачный):

color: rgba(0, 0, 0, 0.5) /\* чёрный, непрозрачный на 50% \*/

color: rgba(255, 0, 0, 0.3) /\* красный, непрозрачный на 30% \*/

color: rgba(255, 255, 255, 0.9) /\* белый, непрозрачный на 90% \*/

**Контраст цвета текста и фона**

Фоновое изображение и фоновый цвет блока всегда должен достаточно сильно контрастировать с цветом текста в этом блоке. Чем больше контраст, тем удобнее этот текст читать в разных условиях освещённости и на разных устройствах. Поэтому если вы задаёте блоку фоновое изображение, нужно обязательно дополнительно задавать подходящий фоновый цвет. В этом случае, пока изображение загружается, или в случае, если оно совсем не загрузится, текст всё равно можно будет прочитать:

p {

/\* идеальный контраст: цвет текста белый, цвет фона — чёрный \*/

background-color: #000000;

color: #ffffff;

}

span {

/\* плохой контраст: цвет текста и фона — серые \*/

background-color: #cccccc;

color: #dddddd;

}

**Свойство white-space, управление пробелами**

Браузер игнорирует множественные пробелы и переносы строк в HTML-коде. С помощью свойства white-space можно управлять пробелами и переносами строк. Свойство принимает значения:

* nowrap — схлопывает лишние пробелы и отображает весь текст одной строкой без переносов;
* pre — сохраняет пробелы и переносы как в исходном коде аналогично тегу <pre>;
* pre-wrap — работает как значение pre, но добавляет автоматические переносы, если текст не помещается в контейнер;
* normal — режим по умолчанию: лишние пробелы и переносы строк схлопываются, текст переносится, пробелы в конце строк удаляются.

**Свойство text-decoration**

Задаёт дополнительное оформление текста. Значения свойства:

1. underline — подчёркивание;
2. line-through — зачёркивание;
3. overline — надчёркивание;
4. none — убирает вышеперечисленные эффекты.

К тексту можно одновременно применить несколько эффектов, если перечислить значения через пробел:

p {

text-decoration: underline; /\* подчёркнутый текст \*/

}

span {

/\* подчёркнутый и зачёркнутый текст \*/

text-decoration: underline line-through;

}

Свойство text-decoration — [составное](https://htmlacademy.ru/courses/307/run/7). Оно раскладывается на отдельные свойства:

* text-decoration-line — вид линии: зачёркивание, подчёркивание или надчёркивание;
* text-decoration-style — стиль линии, может принимать значения:
  + solid — сплошная линия;
  + double — двойная линия;
  + dotted — точечная линия;
  + dashed — пунктирная линия;
  + wavy — волнистая линия.
* text-decoration-color — цвет линии.

**Свойство font-style**

Свойством можно задать начертание текста. Его основные значения:

1. normal — обычное начертание;
2. italic — курсивное начертание;
3. oblique — наклонное начертание.

Если задано значение italic, браузер будет пытаться найти в заданном шрифте отдельное курсивное начертание символов. В некоторых шрифтах отдельный курсив предусмотрен.

Если отдельного курсивного начертания в шрифте не предусмотрено, то браузер сделает текст наклонным, то есть сымитирует курсив. Что равноценно заданию тексту значения font-style: oblique.

**Свойство text-transform**

С его помощью можно управлять регистром символов: делать буквы строчными (маленькими) или заглавными (большими). Значения свойства:

1. lowercase — все строчные;
2. uppercase — все заглавные;
3. capitalize — каждое слово начинается с большой буквы;
4. none — отменяет изменение регистра.

**Отступы**

Важный фактор того, что текст в блоке будет удобочитаемым, это наличие свободного пространства в блоке для этого текста. Вокруг текста должно быть достаточно «воздуха», он не должен «прилипать» к краям, ему не должно быть «тесно».

За отступы в CSS отвечают два свойства: padding задаёт внутренние отступы в блоке, а margin задаёт внешние отступы.

<https://copypastecharacter.com/> Cods of Unicode

[icomoon.io/app](https://icomoon.io/app/), [fontastic.me](http://fontastic.me/), [glyphter.com](http://glyphter.com/) Icons fonts

# Логотип: выравнивание размеров

Для завершения оформления логотипа нам остаётся задать символам нужные размеры и отступы.

Давайте сделаем надпись «WARMPHOTO» плотнее, уменьшив межсимвольное расстояние.

Это делается c помощью свойства letter-spacing. Браузер по умолчанию устанавливает межсимвольное расстояние автоматически согласно определенному значению у каждого шрифта. Значение по умолчанию соответствует letter-spacing: 0.

Увеличение или уменьшение значения letter-spacing изменит расстояние между символами на заданную величину. Значение задаётся в px, em, pt или других единицах длины CSS.

# Логотип: выравнивание размеров

Для завершения оформления логотипа нам остаётся задать символам нужные размеры и отступы.

Давайте сделаем надпись «WARMPHOTO» плотнее, уменьшив межсимвольное расстояние.

Это делается c помощью свойства letter-spacing. Браузер по умолчанию устанавливает межсимвольное расстояние автоматически согласно определенному значению у каждого шрифта. Значение по умолчанию соответствует letter-spacing: 0.

Увеличение или уменьшение значения letter-spacing изменит расстояние между символами на заданную величину. Значение задаётся в px, em, pt или других единицах длины CSS.

# Свойство text-indent

Рассмотрим свойства, с помощью которых можно улучшить читабельность или украсить большие тексты.

И первым свойством будет text-indent. Оно устанавливает отступ для первой строки блока текста. С помощью этого свойства удобно делать красную строку в абзацах.

Значение свойства может принимать положительное или отрицательное значение, задаётся в px, em, pt или других единицах длины CSS. При задании значения в процентах, отступ первой строки вычисляется в зависимости от ширины блока.

# Псевдоэлемент ::first-letter

В CSS существуют особые псевдоэлементы, позволяющие стилизовать первую букву в слове или первую строку в блоке текста.

Крупная, отличная от прочих, первая буква блока текста называется буквицей. Исторически буквицей украшали главы и разделы печатных книг и рукописей. Для создания буквицы воспользуемся псевдоэлементом ::first-letter:

p::first-letter { color: red; }

Давайте создадим и оформим буквицу.

# Псевдоэлемент ::first-line

Первую строку блока текста можно стилизовать при помощи псевдоэлемента ::first-line.

p::first-line { color: green; }

Мы добавили буквице небольшой padding для выравнивания её относительно текста. Давайте завершим стилизацию буквицы и оформим первую строку первого параграфа.

# Колоночная разметка: свойство column-count

Многоколоночная разметка CSS — замечательная возможность, позволяющая разбить блок с текстом на несколько колонок. Обычно очень длинные строки сложно читать: если приходится слишком долго перемещать взгляд с конца одной строки на начало другой, можно легко потерять нужную строку. Чтобы сэкономить место на экране, но при этом сделать текст читабельным, можно разбить один сплошной блок текста на несколько колонок, как это делается в газетах.

Уже сейчас во всех популярных современных браузерах можно использовать многоколоночную разметку CSS. Давайте рассмотрим основные свойства, позволяющие создавать колонки.

Первое из них — column-count. Это свойство принимает в качестве параметра целое число и делит блок текста на заданное число колонок, равных друг другу по ширине.

# Колоночная разметка: свойство column-width

Свойство column-width задаёт минимальную желаемую ширину колонки. Если свойство column-count ещё не было задано, браузер автоматически поделит текст на такое количество колонок, чтобы они уместились во всю доступную ширину.

Значение свойства положительное и задаётся в px, em, pt или других единицах длины CSS.

Следует отметить, что если одиночная строка может включать от 45 до 75 символов, чтобы быть читабельной, то для колонок текста рекомендуется придерживаться ширины, включающей 40-50 символов.

# Колоночная разметка: свойство column-gap

Между колонками есть промежуток. Рекомендованная ширина промежутка по умолчанию равна 1em.

Но она может быть изменена при помощи свойства column-gap. Единицы измерения тоже px, em, pt и другие.

**Направление текста**

Ещё одной интересной, но малоиспользуемой возможностью работы с текстом в CSS является управление направлением текста. Обычно эта возможность применяется для корректного отображения арабского языка и иврита, в которых чтение происходит справа налево. В CSS за направление текста отвечает свойство direction, принимающее значения ltr (направление слева направо) и rtl (направление справа налево).

Однако помимо направления текста свойство direction также влияет на позицию полосы прокрутки в блоке.

Если попробовать задать для кириллицы или латиницы обратное направление текста (справа налево) direction: rtl;, мы увидим, что текст в блоке выровнялся наоборот, и полоса прокрутки блока сменила своё положение на противоположное. Но при этом сам текст не стал менять своё направление. Это произошло потому, что браузер автоматически задал тексту корректное направление, проанализировав используемые символы Unicode. Чтобы повлиять на это решение браузера существует свойство unicode-bidi, принимающее значения:

* normal — браузер самостоятельно определяет, как ему следует отображать текст на основе используемых символов Unicode;
* embed — переопределяет направление текста, располагая его согласно свойству direction (применяется, когда в блоке текст на двух разнонаправленных языках);
* bidi-override — переопределяет порядок символов в тексте согласно значению direction.

# Переполнение текста

Ещё одно интересное текстовое свойство — text-overflow. Оно позволяет определить, как будет выглядеть текст, если не поместится в контейнер. Оно принимает разные значения, но универсальными и работающими во всех современных браузерах являются clip и ellipsis.

Значение clip задано по умолчанию, и при нём текст просто обрезается по размеру контейнера, а при ellipsis — обрезается и к концу строки добавляется многоточие.

К сожалению, у этого свойства есть ограничения, которые делают его менее полезным: оно применимо только к однострочным текстам, а также к блокам, значение свойства overflow которых установлено в auto, scroll или hidden.

# Интервал между словами

И, наконец, последнее свойство, которое мы рассмотрим в этой части, — word-spacing. Оно задаёт расстояние между отдельными словами и строчными элементами. Значение может быть положительное или отрицательное и задаётся в px, em, pt или других единицах измерения CSS.

Свойство word-spacing также можно использовать для задания отступов между блочно-строчными элементами, а также изображений, ведь изображения тоже являются строчными.

# Декоративная стилизация строки, шаг 2

Теперь нам нужно изменить правила переноса, чтобы символы, не поместившиеся по ширине в контейнер, переносились на новую строку.

Для этого в CSS существует свойство word-wrap. Оно принимает значения: normal и break-word. В случае normal слова переносятся на новую строку по обычным правилам (то есть по пробелам). А при значении break-word перенос производится браузером в тех местах, где слова перестают помещаться в контейнер.

Это как раз то, что нам нужно: зададим нужный перенос слова, увеличим межсимвольное расстояние и поправим отступ.

Также нужно заметить, что свойство word-wrap — изначально проприетарное расширение Microsoft, а в текущем черновике спецификации CSS переименовано в overflow-wrap. Название word-wrap сейчас обозначено как «альтернативное» именование.