

# Оглавление

<b>3</b>	<b>Шрифты</b>	<b>2</b>
3.1	«Анатомия» шрифта . . . . .	2
3.2	Размер шрифта . . . . .	4
3.2.1	Ключевые слова . . . . .	5
3.2.2	Единицы измерения длины . . . . .	5
3.2.3	Высота строки . . . . .	7
3.2.4	Расстояние между буквами и расстояние между словами . . . . .	8
3.3	Начертание . . . . .	9
3.3.1	Капитель . . . . .	9
3.3.2	Характеристики рисунка: наклон, насыщенность и плотность . . . . .	9
3.4	Семейства шрифтов . . . . .	12
3.5	... . . . .	12
3.6	Этапы работы браузера по работе с шрифтами . . . . .	17
3.7	Свойство font . . . . .	17
<b>4</b>	<b>Текст</b>	<b>19</b>
4.1	Текст . . . . .	19
4.2	Многоколоночность . . . . .	21
4.3	Переполнение текста . . . . .	23
4.4	Перенос текста . . . . .	24
4.4.1	Переносы внутри слов . . . . .	24
4.4.2	Мягкие переносы . . . . .	25
4.5	Псевдоэлементы для работы с текстом . . . . .	29
4.5.1	Псевдоэлемент первой строки . . . . .	29
4.5.2	Псевдоэлемент первой буквы . . . . .	30

# Глава 3

## Шрифты

### 3.1. «Анатомия» шрифта

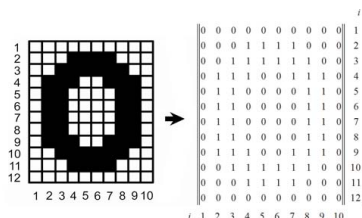
**Шрифт** — графический рисунок начертаний букв и знаков, составляющих единую стилистическую и композиционную систему, набор символов определенного размера и рисунка.

**Компьютерный шрифт** (файл шрифта) — это файл, содержащий в себе набор символов и соответствующих им кодов. Символы могут быть различными по назначению:

Аа	07	™↓	\$#!
языковые знаки	цифровые	графические	специальные

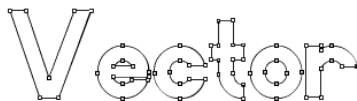
Шрифт, как и любая другая технология, не стоит на месте. Самым главным различием между старыми и новыми шрифтами является характеристика их вывода. Бывают:

**Растровые шрифты** Представляют собой набор растровых изображений каждого символа.



Были распространены в эпоху матричных принтеров и экранов с очень маленьким разрешением.

**Векторные шрифты** Представляют собой набор описаний символов с помощью математических формул.



Как и любое векторное изображение, векторные шрифты можно масштабировать без потери качества.

**Гарнитура** — типографский термин, объединяющий набор шрифтов, которые отличаются по **размеру, начертанию, наличию или отсутствию засечек на концах линий, пропорциям символов**, по соотношению размера высоты прописных и строчных знаков, величине верхних и нижних выносных элементов, **плотности**, то есть близких по характеру и отличительным знакам рисунка.

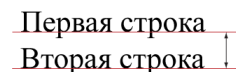
Прежде всего следует рассмотреть несколько понятий из типографики.



Базовая линия



Высота знаков



Интерлиньяж

**Базовая линия** — воображаемая прямая линия, проходящая по нижнему краю прямых знаков без учёта свисаний и нижних выносных элементов.

**Высота прописных знаков** — расстояние от базовой линии до верхней линии прописных, то есть высота прописных букв без учета свисаний.

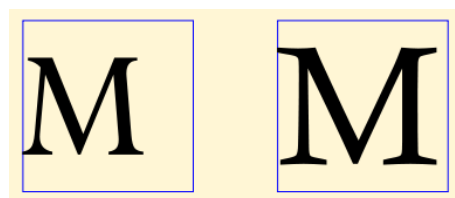
**Высота строчных знаков** — расстояние от базовой линии до верхней линии строчных, то есть высота строчных букв без свисаний и выносных элементов.

**Интерлиньяж (межстрочный интервал)** — расстояние между базовыми линиями строк. В вебе данный термин чаще всего называют высотой строки.

**Кегельная площадка** — верхняя прямоугольная часть ножки литеры, на которой расположено выпуклое (печатающее) изображение знака. В цифровом шрифте кегельная площадка важна при проектировании шрифта и представляет собой прямоугольник, в который вписывается изображение знака.



Два разных шрифта находятся в одной и той же кегельной площадке.



Сравнение площадок шрифтов Perpetua и Calisto

## 3.2. Размер шрифта

Размер шрифта определяется как высота от базовой линии до верхней границы кегельной площадки. Размер шрифта элемента определяется с помощью свойства `font-size`. Размер шрифта можно установить с помощью ключевых слов или единиц измерения длины.

### 3.2.1. Ключевые слова

Первый набор — ключевые слова, задающие абсолютные значения: **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**. Поскольку результат зависит от настроек браузера пользователя, использовать их не рекомендуется.

CSS	xx-small	x-small	small	medium	large	x-large	xx-large	
HTML	1		2	3	4	5	6	7

По сути такие ключевые слова являются прямыми наследниками `<font size="X" ></font>` с небольшим сдвигом.

Второй набор ключевых слов задают относительные значения: **smaller** и **larger**. Они изменяют размер шрифта относительно размера шрифта родителя примерно на 20%.

```
1      mark {
2          font-size: larger;
3      }
4
5      strong {
6          font-size: smaller;
7      }
```

Из-за непредсказуемости использовать их не рекомендуется.

### 3.2.2. Единицы измерения длины

Для задания размера шрифта можно использовать единицы измерения длины.

**px** Пиксель

```
1      p {
2          font-size: 10px;
3      }
```

Поскольку задание динамического размера требуется нечасто, является самой популярной единицей измерения.

**em, проценты** Размер относительно размера шрифта родительского элемента.

```

1      div {
2          font-size: .6em; /* то же, что и 60% */
3          color: #0b0;
4      }

```

Различия между em и % могут появиться только в зависимости от настроек браузера.

**rem (root em)** Размер относительно размера шрифта корневого элемента (html).

```

1      html {
2          font-size: 16px;
3      }
4
5      h2 {
6          font-size: 60px;
7      }
8
9      div {
10         font-size: 2rem;
11     }

```

Типографские единицы измерения практически не используются в вебе:

**pc** — типографская пика,

**pt** — типографский пункт,

**ch** — ширина символа «0»,

**ex** — высота символа «x».

Абсолютные единицы из реального мира могут использоваться для верстки печатных версий сайтов:

**cm** — сантиметр ( $1cm = 96px/2.54$ )

**mm** — миллиметр ( $1mm = 1cm/10$ )

**q** — четверть миллиметра ( $1q = 1cm/40$ )

**in** — дюйм ( $1in = 96px$ )

Слишком непредсказуемы при верстке в вебе, как ни странно.

Единицы измерения относительно viewport'a (vh, vw, vmin, vmax) в принципе не предназначены для работы со шрифтами.

### 3.2.3. Высота строки

Высота строки устанавливается свойством `line-height` двумя типами значений: единицами измерения длины или множителем.

```
1      body { font-size: 20px; }
2
3      .line-height-mult { line-height: 1.5; }
4      .line-height-em   { line-height: 1.5em; }
5      .line-height-pixel { line-height: 30px; }
```

Результат будет следующий:

```
1      <div class="line-height-mult">
2          <!-- Высота\ строки\ множителем 1.5 x 20 = 30 -->
3          <p>...</p>
4      </div>
5
6      <div class="line-height-em">
7          <!-- Высота\ строки\ в ем 1.5 x 20 = 30 -->
8          <p>...</p>
9      </div>
10
11     <div class="line-height-pixel">
12         <!-- Высота\ в\ пикселях 30 = 30 -->
13         <p>...</p>
14     </div>
```

Разницы, как видно, никакой. Но при изменении размера шрифта у параграфов:

```
1      body { font-size: 20px; }
2
3      .line-height-mult { line-height: 1.5; }
4      .line-height-em   { line-height: 1.5em; }
5      .line-height-pixel { line-height: 30px; }
6
7      p { font-size: 30px; }
```

получается результат, который отличается от предыдущего:

```
1      <div class="line-height-mult">
2          <!-- Высота\ строки\ множителем 1.5 * 30 = 45 -->
```

```

3     <p>...</p>
4 </div>
5
6 <div class="line-height-em">
7     <!-- Высота\ строки\ в ем 1.5 x 20 = 30 -->
8     <p>...</p>
9 </div>
10
11 <div class="line-height-pixel">
12     <!-- Высота\ в\ пикселях 30 = 30 -->
13     <p>...</p>
14 </div>

```

Множитель использует размер шрифта элемента, к которому применено свойство, а не родительского элемента, как в случае с `em`. Рекомендуется использовать именно множитель в виду его большей очевидности и гибкости.

### 3.2.4. Расстояние между буквами и расстояние между словами

Для изменения расстояния между буквами предусмотрено свойство `letter-spacing` со значением в любой единице измерения длины кроме процентов.

```

1     .zero {
2         letter-spacing: normal;
3     }
4
5     .first {
6         letter-spacing: -2.5px;
7     }
8
9     .second {
10        letter-spacing: 1em;
11    }

```

Аналогичным свойством является свойство `word-spacing`, которое устанавливает расстояние между словами. Значение может быть в любой единице измерения длины кроме процентов.

```

1     .zero {
2         word-spacing: normal;

```



```

3      }
4
5      .first {
6          word-spacing: -10px;
7      }
8
9      .second {
10         word-spacing: 1em;
11     }

```

## 3.3. Начертание

### 3.3.1. Капитель

Капитель — начертание, в котором строчные знаки выглядят как уменьшенные прописные. Устанавливается с помощью свойства `font-variant`, у которого есть два значения: `normal` и `small-caps`.

```

1  .h3 {
2      font-variant: normal;
3  }

```

```

1  .h3 {
2      font-variant:
3      ↪ small-caps;

```

Данный прием часто использовался в газетах и часто используется в заголовках сайтов.


### 3.3.2. Характеристики рисунка: наклон, насыщенность и плотность

Наклоном гарнитуры шрифта можно управлять с помощью свойства `font-style`. Он может принимать три значения:

```

1  font-style:
   ↳ normal;      /*
   ↳ a */
2  font-style:
   ↳ oblique;     /*
   ↳ б */
3  font-style:
   ↳ italic;      /*
   ↳ в */

```



а — светлые; б — полужирные;  
в — жирные

Свойство `font-weight` позволяет задавать насыщенность:

```

1  font-weight:
   ↳ normal      /*
   ↳ обычный (а)
   ↳ */
2  font-weight:
   ↳ bold        /*
   ↳ жирный
   ↳ (в) */
3  font-weight:
   ↳ light       /*
   ↳ тонкий
   ↳ */

```



а — прямые; б — наклонные;  
в — курсивные

Также можно установить значение числом от 100 до 900 с шагом 100:

```

1  font-weight: 100    /* thin      */
2  font-weight: 200    /* extra light */
3  font-weight: 300    /* light      */
4  font-weight: 400    /* normal     */
5  font-weight: 500    /* medium     */
6  font-weight: 600    /* semi-bold (б) */
7  font-weight: 700    /* bold       */
8  font-weight: 800    /* extra bold  */
9  font-weight: 900    /* black      */

```

Также можно указать относительное значение. Относительные значения зависят от настроек браузера, поэтому их не рекомендуется использовать.

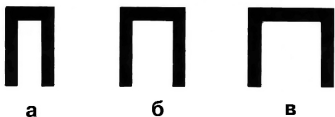
```

1  font-weight: lighter /* тоньше шрифта родителя на ~300 */
2  font-weight: bolder  /* жирнее шрифта родителя на ~300 */

```

Для того, чтобы управлять плотностью гарнитуры шрифта, в CSS есть свойство `font-stretch`:

```
1      font-stretch:
      ↳ ultra-condensed;
2      font-stretch:
      ↳ extra-condensed;
3      font-stretch:
      ↳ condensed;
4      font-stretch:
      ↳ semi-condens
5      font-stretch:
      ↳ normal;
6      font-stretch:
      ↳ semi-expanded;
7      font-stretch:
      ↳ expanded;
8      font-stretch:
      ↳ extra-expanded;
9      font-stretch:
      ↳ ultra-expanded;
```



а — узкие; б — нормальные;  
в — широкие

Данное свойство не поддерживается в Safari, а также совсем недавно появилось в Chrome. Поэтому, если нужен, например, узкий шрифт, то его нужно подключать отдельно.

```
1      .block {
2          font-family: 'My Font Condensed'
3      }
```

Если необходимо подключить широкий шрифт:

```
1      .block {
2          font-family: 'My Font Expanded'
3      }
```

Есть нюанс работы с начертаниями: в зависимости от подключенных к странице шрифтов, результат может быть не тем, какой ожидается.

Яндекс  
Яндекс

наклонный шрифт

Яндекс  
Яндекс

жирный шрифт

На первой строчке гарнитуры шрифтов были специально подготовлены дизайнерами, а на второй — сгенерированные браузером, так как у него нет нужной гарнитуры. Браузер попытается «наклонить»/«насытить» обычное начертание, используя для этого упрощенные алгоритмы.

### 3.4. Семейства шрифтов

Классификация шрифтов (по семействам) в веб следующая:

**serif** — шрифты с засечками (антиква)

**sans-serif** — шрифты без засечек (гротески)

**cursive** — рукописные шрифты (каллиграфические)

**monospace** — моноширинные шрифты (непропорциональные)

**fantasy** — декоративные шрифты (акцидентные)

В скобках указаны названия этих шрифтов в типографике.

### 3.5. ...

Существует ряд шрифтов, которые есть практически в каждой ОС, таким образом их использование считается наиболее безопасным. Самые известные из них:

Без засечек:	Arial
С засечками:	Times New Roman
Моноширинный:	Courier New

Проблем с подключением этих шрифтов к странице не должны возникать. Чтобы использовать нужный шрифт, нужно указать свойство `font-family` с названием нужного шрифта:

```
1      body {
2          font-family: 'Arial';
3      }
```

Можно писать название шрифта без кавычек:

```
1      body {
2          font-family: Arial;
3      }
```

Также можно писать несколько шрифтов через запятую:

```
1      body {
2          font-family: Arial, Helvetica, Calibri;
3      }
```

Если браузер не найдет первый шрифт, он попробует следующий. В ситуации, когда все шрифты не были найдены, браузер установит шрифт по умолчанию. В 99% случаев – Times New Roman:

```
1      body {
2          font-family: Arial, Helvetica, Calibri, /* Times New Roman
3              ↳ */;
4      }
```

Но Times New Roman – шрифт с засечками, а нам нужно, чтобы вместо него установился доступный шрифт без засечек. Чтобы указать браузеру, что нужно установить шрифт по умолчанию нужного семейства, можно в качестве значений использовать:

```
1      font-family: serif;          /* с засечками */
2      font-family: sans-serif;     /* без засечек */
3      font-family: cursive;        /* рукописные */
4      font-family: monospace;      /* моноширинные */
5      font-family: fantasy;        /* декоративные */
```

## Нестандартные шрифты

Всегда есть возможность подключить свой шрифт на страницу с помощью ат-правила `@font-face`. В нем указывается свойство `font-family`, в котором пишется название шрифта как оно будет использоваться в коде, а также свойство `src`, в котором указывается путь до файла шрифта.

```
1      @font-face {
2          font-family: myCustomFont;
3          src: url(myCustomFont.woff);
4      }
```

Формат `woff` — один из самых распространенных форматов файлов шрифтов. После этого можно написать:

```
1      body {
2          font-family: myCustomFont, sans-serif;
3      }
```

Название семейства здесь добавлено для безопасности как запасное решение. Если есть вероятность, что какой-либо шрифт уже установлен на пользовательский компьютер, можно указать его через `local()`:

```
1      @font-face {
2          font-family: myCustomFont;
3          src: local('Native Custom Font')
4              url(myCustomFont.woff);
5      }
```

А можно указать диапазон используемых в шрифте символов с помощью `unicode-range`:

```
1      @font-face {
2          font-family: myCustomFont;
3          unicode-range: U+000-5FF; /* только латинские символы */
4          src: local('Native Custom Font')
5              url(myCustomFont.woff);
6      }
```

Но есть несколько нюансов:

- Пользователи, особенно мобильных устройств, очень редко имеют установленные шрифты.

- Свойство `unicode-range` очень плохо поддерживается.

Подключенный шрифт жирного начертания и, например, курсивного начертания – два разных файла, и подключать их нужно отдельно.

```
1      @font-face {
2          font-family: Sooo Good Font Bold; /* жирный */
3          src: url(soGoodFontBold.woff);
4      }
5
6      @font-face {
7          font-family: Sooo Good Font Italic; /* курсивный */
8          src: url(soGoodFontItalic.woff);
9      }
```

В таком случае приходится запоминать название шрифта вместе с начертанием.

```
1      header {
2          font-family: Sooo Good Font Bold, sans-serif; /* жирный */
3      }
4
5      footer {
6          font-family: Sooo Good Font Italic, sans-serif; /* курсивный
7              ↳ */
8      }
```

Но можно привязывать определенный файл к определенному начертанию с помощью свойств `font-weight` и `font-style`, указанных в `@font-face`:

```
1      @font-face {
2          font-family: Sooo Good Font;
3          src: url(soGoodFont.woff);
4          font-style: normal; /* обычный */
5      }
6
7      @font-face {
8          font-family: Sooo Good Font;
9          src: url(soGoodFontItalic.woff);
10         font-style: italic; /* курсивный */
11     }
```

```

1  body {
2      font-family: Sooo Good Font; /* обычный */
3  }
4
5  footer {
6      /* font-family унаследуется от body */
7      font-style: italic; /* курсивный */
8  }

```

Раньше, чтобы подключить шрифт на страницу, приходилось использовать аж 4 формата: .svg, .ttf, .woff и .eot:

```

1  @font-face {
2      font-family: myFont;
3      src: url('myFont.eot');
4      src: url('myFont.eot?#iefix') format('embedded-opentype'),
5           url('myFont.woff')      format('woff'),
6           url('myFont.ttf')       format('truetype'),
7           url('myFont.svg')       format('svg');
8
9      font-weight: normal;
10     font-style: normal;
11 }

```

Сейчас всё куда лучше, но Android 4.3 — не понимают формат .woff, и для них еще нужно подключить .ttf:

```

1  @font-face {
2      font-family: myFont;
3      src: url(myFont.woff) format(woff),
4           url(myFont.ttf)  format(truetype);
5  }

```

Для генерации шрифтов в разных форматах существуют сервисы-генераторы, которые помогают собрать весь пакет нужных форматов в несколько кликов. Они так же помогают значительно уменьшить вес благодаря выбору только нужных символов шрифта или диапазонов, например, только кириллических и латинских символов. Самый такой сервис популярный — [Font Squirell](#). Почитать подробнее про различия сервисов можно [тут](#).

А еще существуют сервисы, которые предоставляют шрифты. Самый популярный — [Google Fonts](#).



## 3.6. Этапы работы браузера по работе с шрифтами

Шрифты – файлы, а файлы нужно загружать. Более того, когда шрифт загружен на страницу, он не применяется сразу.

Условные этапы работы браузера по работе со шрифтами:

1. Браузер получает и парсит CSS
2. Встречает в CSS @font-face, но НЕ скачивает файл шрифта
3. Парсит дальше и встречается указание шрифта в font-family
4. Начинает загрузку шрифта
5. По окончании загрузки парсит шрифт и применяет к странице

Пока шрифт не загружен, пользователь видит так называемый FOIT (Flash of invisible text). Существуют также FOUT, FOIT, FOFT ([про них можно прочесть на CSS-Tricks](#)). Веб не стоит на месте, и сейчас обсуждается свойство font-display, которое позволяет контролировать поведение шрифта, подключенного в @font-face, но не примененного на странице.

Основное состоит в том, что загрузка файлов шрифта и его отображение на странице – дорогая операция. Не вдаваясь в подробности, с данной проблемой можно сделать следующее:

- Запись шрифта напрямую в CSS в формате base64
- Исключение ненужных символов
- Архивация gzip
- Кэширование с помощью настроек сервера
- Использование формата WOFF 2.0
- Поднятие @font-face и его использования в font-family выше в коде
- и другие.

Свежая отличная [статья о шрифтах и их оптимизации](#).

## 3.7. Свойство font

Свойство font — универсальное свойство, которое позволяет контролировать каждое свойство, связанное с шрифтами. Записывается оно строго в следующем порядке:

```
1      font: variant style weight size / line-height family;
```

При этом обязательными из этих свойств являются size и family.

```
1      body {
2          font: 14px;                /* не будет работать */
3          font: italic Arial;        /* тоже не будет работать */
4
5          font: 14px Arial, sans-serif;
6          font: 14px / 1.2 Arial, sans-serif;
7          font: italic bold 14px Times New Roman;
8          font: small-caps italic 700 14px / 20px Arial, sans-serif;
9      }
```

# Глава 4

## Текст

### 4.1. Текст

Текст – зафиксированная на каком-либо материальном носителе человеческая мысль; в общем плане связная и полная последовательность символов.

Выравнивание текста осуществляется с помощью свойства `text-align`:

```
1  .uno    { text-align: left;      } /* По левому краю */
2  .dos    { text-align: start;     } /* По краю начала текста */
3  .tres   { text-align: right;     } /* По правому краю */
4  .quatro { text-align: end;       } /* По краю, противоположенному
   ↪ началу текста */
5  .cinco  { text-align: center;    } /* По центру */
6  .seis   { text-align: justify;   } /* По всей ширине */
```

Выравнивание по всей ширине работает, если больше одной строки текста.

«Украшение» текста устанавливается с помощью свойства `text-decoration`, которое имеет три законных значения `overline` (надчеркивание), `line-through` (зачеркивание) и `underline` (подчеркивание). Подчеркивание чаще всего используется для ссылок. Стоит иметь в виду, что `text-decoration` не наследуется дочерними элементами, и им нельзя отменить «украшение», которое применено к родителю.

```
1  <p style="text-decoration: overline;">
2      Текст надчеркнут.
3      <span style="text-decoration: none;">Текст по прежнему
   ↪ надчеркнут</span>.
4  </p>
```

Свойство `text-transform` позволяет менять регистр букв с помощью трех значений: `capitalize`, `lowercase` и `uppercase`.

Для задания величины отступа используется свойство `text-indent`. Допустимо использовать любые единицы измерения длины и их отрицательные значения.

```
1      .eins {
2          text-indent: 5%;
3      }
4
5      .zwei {
6          text-indent: -1em;
7      }
```

Цвет текста задается с помощью свойства `color`.

```
1      h1 {
2          color: #f00;
3      }
```

Принято использовать `hex` для сплошных цветов и `rgba()` для полупрозрачных. Почитать буквально обо всём, что связано с цветами, можно [тут](#).

Но есть несколько нюансов:

- Свойство `color` устанавливает так называемый «цвет переднего плана» (в противовес `background-color` – цвет фона), поэтому он имеет влияние на многие свойства.

```
1      p {
2          color: #0f0;           /* зеленый цвет текста */
3          border: 5px solid;     /* рамка будет зеленой */
4          box-shadow: 0 0 50px;  /* и тень элемента */
5          text-shadow: 4px 4px 10px; /* и тень текста */
6      }
```

- Свойства, которые не наследуют цвет от `color` автоматически, могут сделать это при помощи ключевого слова `currentColor`.

Тень текста указывается с помощью свойства `text-shadow`:

```
1      text-shadow: offset-x offset-y blur-radius color
```

Обязательными являются только сдвиг по `x` (`offset-x`) и сдвиг по `y` (`offset-y`). По умолчанию тень такого же цвета, как и текст.

Например, создадим тень со сдвигом в 4 пикселя вправо вниз:

```

1      h1 {
2          text-shadow: 4px 4px;
3      }

```

Закрасим её красным цветом:

```

1      h1 {
2          text-shadow: 4px 4px #f00;
3      }

```

Добавим размытие в 10 пикселей:

```

1      h1 {
2          text-shadow: 4px 4px 10px #f00;
3      }

```

Укажем еще одну тень со сдвигом влево вверх зеленого цвета:

```

1      h1 {
2          text-shadow:  4px  4px 10px #f00,
3                      -4px -4px 10px #0f0;
4      }

```

## 4.2. Многоколоночность

Многоколоночность — прием при верстке газет, который позволяет большие и широкие блоки текста разделить на несколько колонок для того, чтобы было удобнее читать.

Предположим дан текст:

```

1      <div>
2          В европейских языках чтение текста происходит слева направо,
3          в то время как есть языки, где текст читается справа
4          налево. При смешении в одном документе разных
5          по написанию символов в системе юникод,
6          их направление определяется браузером из характеристик
7          и содержимого текста.
8      </div>

```

Нужное количество колонок можно указать с помощью column-count:

```

1      div {
2          column-count: 2;
3      }

```

Расстояние между колонками можно задать, используя column-gap:

```

1      div {
2          column-count: 2;
3          column-gap: 300px;
4      }

```

С помощью column-rule можно добавить разделитель между колонками, задав ширину, тип и размер разделителя:

```

1      div {
2          column-count: 2;
3          column-rule: 2px solid #0f0;
4      }

```

С помощью column-width можно указать ширину колонок:

```

1      div {
2          column-width: 200px;
3          column-rule: 2px solid #fff;
4      }

```

Следует обратить внимание, что в данном случае не указано количество колонок, а только их ширина.

Добавить сноску на всю ширину статьи можно с помощью column-span со значением all:

```

1      <div>
2          В европейских языках чтение текста происходит слева направо,
3          в то время как есть языки, где текст читается справа
4          <h3>СНОСКА ПРЯМ ТУТА, НЕСКРОМНО</h3>
5          налево. При смешении в одном документе разных
6          по написанию символов в системе юникод,
7          их направление определяется браузером из характеристик
8          и содержимого текста.
9      </div>

1      div {
2          column-count: 2;

```

```

3         column-rule: 2px solid #fff;
4     }
5
6     h3 {
7         column-span: all;
8         font-style: italic;
9         margin: 10px;
10    }

```

Стоит заметить, что данные свойства поддерживаются далеко не во всех браузерах, а если поддерживаются, то указываются с префиксами. Чуть более подробно о колонках можно почитать [здесь](#).

### 4.3. Переполнение текста

Пусть дана следующая таблица.

Name	Surname	Points
Cloud	Strife	13
John	Cena	37
Akakiy	Stanislav	666
Sokolovskiy	Ruslan	2

В предпоследней строке не влезает в ячейку фамилия, а в последней — имя. В данном случае переполнение контента следует скрыть с помощью `overflow: hidden`.

Name	Surname	Points
Cloud	Strife	13
John	Cena	37
Akakiy	Stanislav	666
Sokolovskiy	Ruslan	2

Как видно, буквы обрезались посередине.

Чтобы сделать скрывание переполнения чуть более изящным, можно воспользоваться свойством переполнения текста `text-overflow`, имеющее два значения: `clip` (значение по умолчанию) и `ellipsis`. После применения значения `ellipsis`:

Name	Surname	Points
Cloud	Strife	77
Vladimir	Putin	147
Akakiy	Stanisl...	666
Sokolo...	Ruslan	2

Хочется отметить, что `ellipsis` не с таблицами работает несколько иначе. Об этом можно прочитать в дополнительных материалах.

## 4.4. Перенос текста

Перенос текста осуществляется в HTML с помощью `<br>`:

1

### 4.4.1. Переносы внутри слов

Но больший интерес представляют переносы внутри слов. Они избавляют от «лесенок» при выравнивании по левому краю, сводят к минимуму разброс ширины пробелов при выравнивании по всей ширине.

Чтобы использовать автопереносы, нужно указать свойство `lang` для параграфа, а также установить автопереносы с помощью свойства `hyphens`.

```
1      <p lang="ru">
2          Далеко-далеко за словесными горами в стране
3          гласных и согласных букв живут рыбные тексты.
4          Вдали от всех живут они в буквенных домах на берегу
5          Семантика большого языкового океана.
6      </p>
```

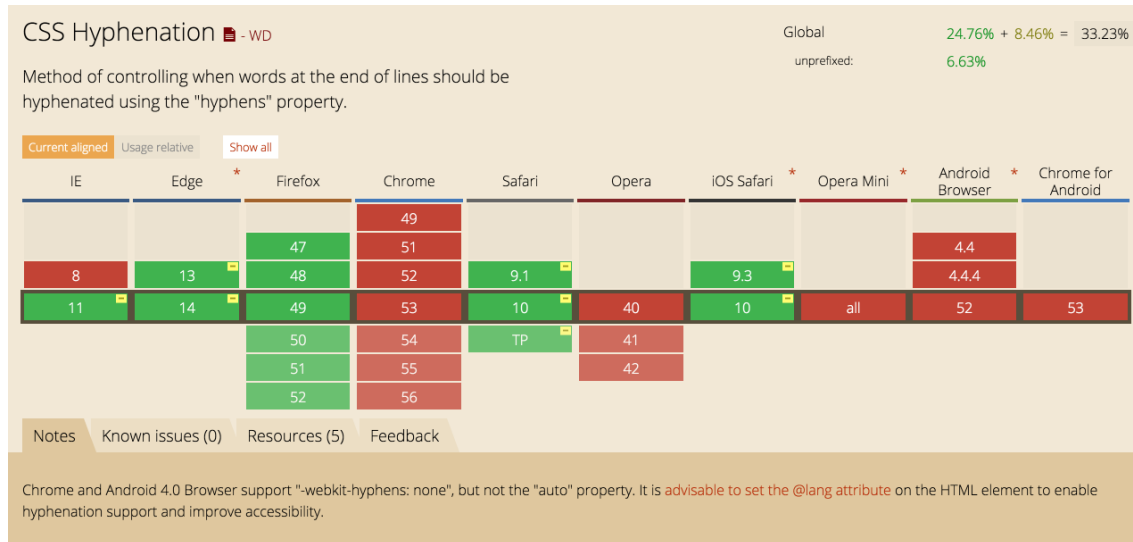


```

1      p {
2          hyphens: auto;
3      }

```

На самом деле далеко не каждый браузер поддерживает эту особенность.



## 4.4.2. Мягкие переносы

Мягкие переносы указываются с помощью `&shy;` и работают при `hyphens: manual`.

До:

```

<p>
    Далеко–далеко за словесными горами в стране гласных и согласных живут рыбные тексты.
    Вдали от всех живут они в буквенных домах на берегу Семантика большого языкового океана.
</p>

```

После:

```

<p>
    Да&shy;ле&shy;ко–да&shy;ле&shy;ко за сло&shy;вес&shy;ны&shy;ми го&shy;ра&shy;ми
    в стра&shy;не глас&shy;ных и со&shy;глас&shy;ных жи&shy;вут ры&shy;бные тексты.
    В&shy;дали от всех жи&shy;вут они в бук&shy;вен&shy;ных до&shy;мах на бе&shy;ре&shy;гу
    Се&shy;ман&shy;ти&shy;ка боль&shy;шо&shy;го язы&shy;ко&shy;во&shy;го оке&shy;ана.
</p>

```

### Пример сервиса, расставляющего переносы.

Переносы без дефиса устанавливаются с помощью `<wbr>`:

```
1      <p>
2          Да<wbr>ле<wbr>ко-да<wbr>ле<wbr>ко за
           ↳ сло<wbr>вес<wbr>ны<wbr>ми
3      го<wbr>ра<wbr>ми в стра<wbr>не глас<wbr>ных и со<wbr>
           ↳ глас<wbr>ных букв
4      жи<wbr>вут рыб<wbr>ные тексты. Вда<wbr>ли от всех жи<wbr>вут
           ↳ они в
5      бук<wbr> вен<wbr>ных до<wbr>мах на бе<wbr>ре<wbr>гу
           ↳ Се<wbr>ман<wbr>ти<wbr>ка
6      боль<wbr>шо<wbr>го язы<wbr>ко<wbr>во<wbr>го оке<wbr>ана.
7      </p>
```

Если слово разрывать можно в любом месте, то это можно указать, используя свойство `word-break` со значением `break-all` (есть еще значение `keep-all`, но работает оно только для китайского, корейского и японского языков).

```
1      .normal {
2          word-break: normal;
3      }
4
5      .breakall {
6          word-break: break-all;
7      }
```

Существуют ситуации, когда слово или словосочетание не нужно разрывать ни при каких обстоятельствах. Простейший пример:

```
1      Тут нужно что-то написать, чтобы текст сдвинул `mark` в неудобное
           ↳ положение.
2      Правило <mark>hyphens: auto;</mark> использует примерно никто,
3      ибо пока оно не работает нормально
```

Перенос нужно запретить:

```
1      mark {
2          white-space: nowrap;
3      }
```

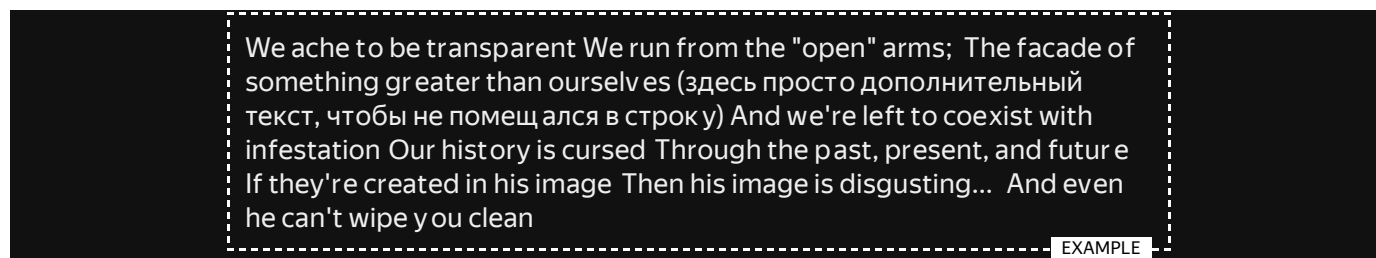
И `white-space` умеет еще много чего. Следующая верстка:

```

1      <div>
2          We ache to be transparent
3          We run from the "open" arms;
4          The facade of something greater than ourselves
           ↳ <span>(здесь просто дополнительный текст, чтобы
           ↳ не помещался в строку)</span>
5          And we're left to coexist with infestation
6
7      Our history is cursed
8
9          Through the past, present, and future
10         If they're created in his image
11         Then his image is disgusting...
12         And even he can't wipe you clean
13     </div>

```

ВЫГЛЯДИТ ВОТ ТАК.



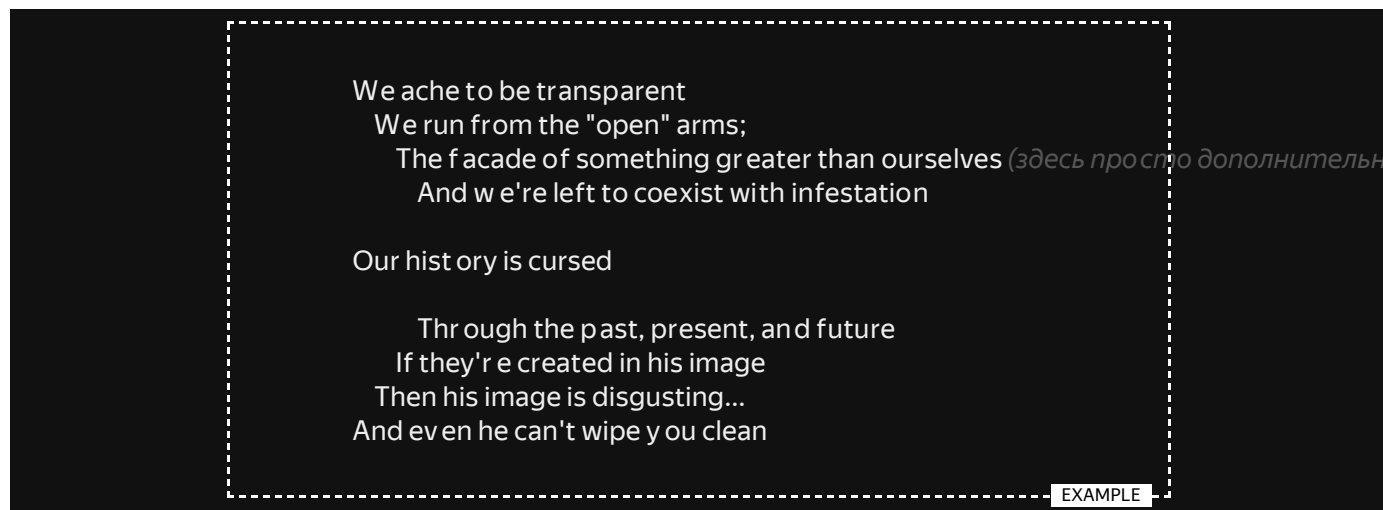
Если же указать

```

1      p {
2          white-space: pre;
3      }

```

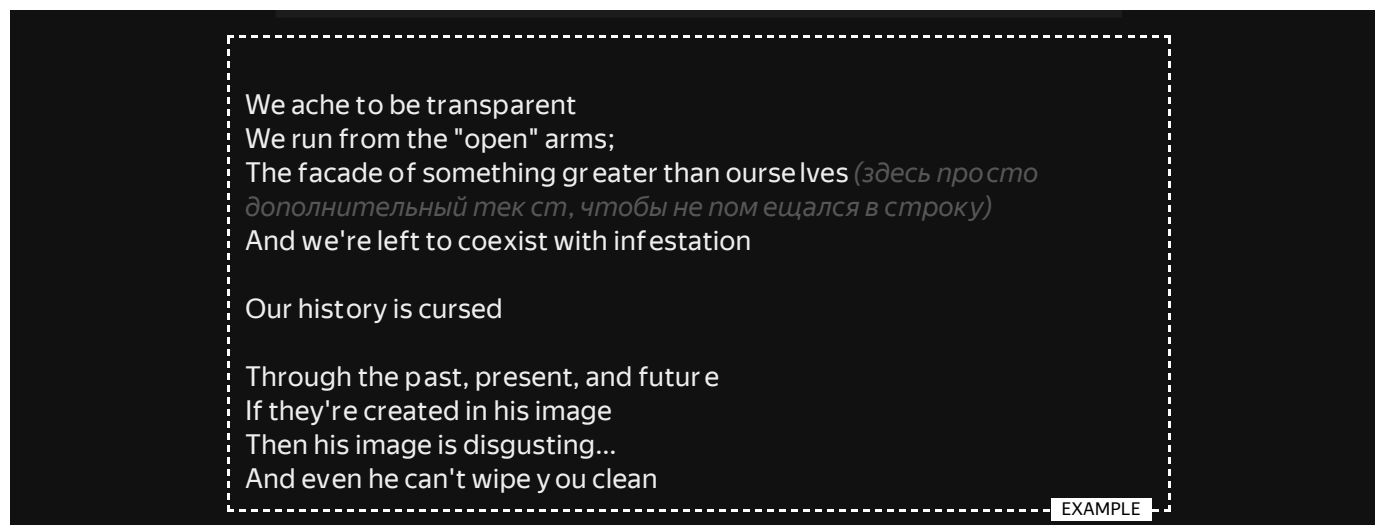
то результат будет следующий:



Если установить значение `pre-line`,

```
1  p {  
2    white-space: pre-line;  
3  }
```

то все переносы будут сохранены, а все пробелы будут схлопнуты до одного:



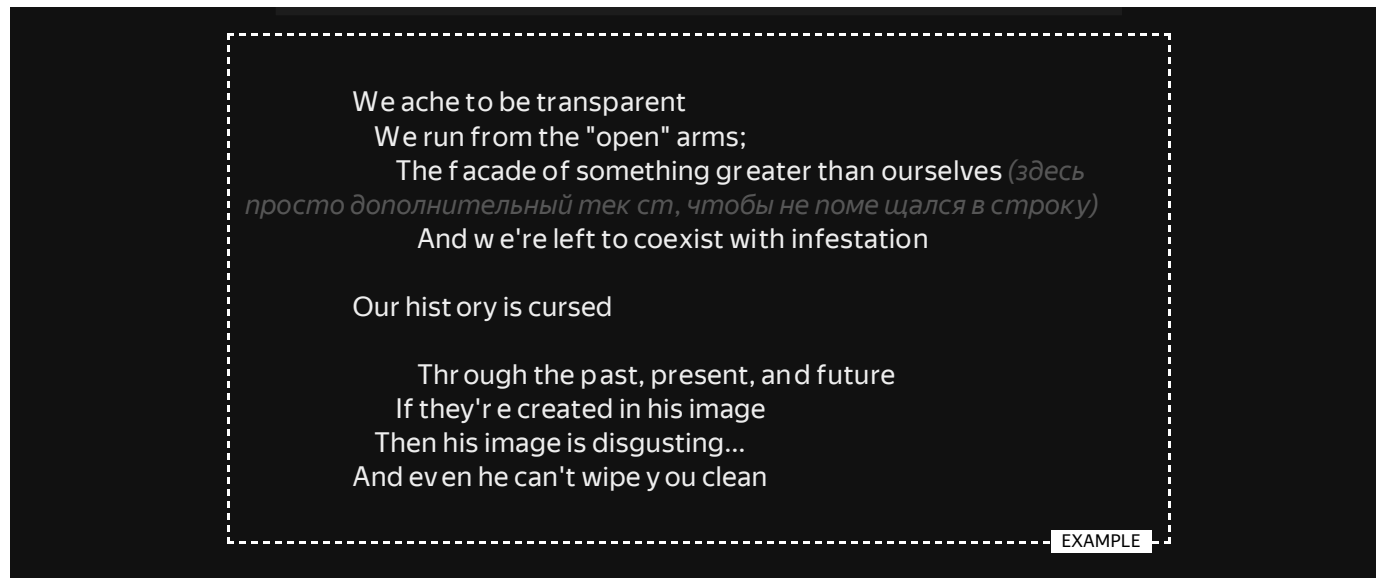
Если указать значение `pre-wrap`,

```

1      p {
2          white-space: pre-wrap;
3      }

```

получается немного другой результат: сохранены переносы и пробелы, но если текст не помещается в строку, то он переносится и начинается с самого края.



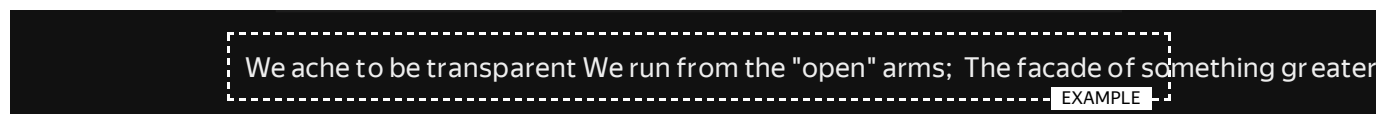
Если установлено значение nowrap,

```

1      p {
2          white-space: nowrap;
3      }

```

никакие переносы и множественные пробелы не сохраняются:



## 4.5. Псевдоэлементы для работы с текстом

### 4.5.1. Псевдоэлемент первой строки

Псевдоэлемент первой строки `::first-line` позволяет оформить первую строку. Примерный список того, что можно сделать:

- font
- word/letter-spacing
- text
- color
- background
- vertical-align

Например:

```

1      p::first-line {
2          font: 25px Courier New;
3          letter-spacing: 6px;
4      }

1      <p>
2          Текст в первой строке будет выглядеть как китайско-русская
3          ↳ инструкция
4          купленного на таганчике барахла. И не спрашивайте откуда
5          я это знаю и почему вообще показываю как этот шрифт
6          ↳ имитировать.
7      </p>

```

### 4.5.2. Псевдоэлемент первой буквы

Псевдоэлемент первой буквы `::first-letter` позволяет оформить первую букву. Список того, что можно сделать, пополнился:

- margin
- padding
- border

```

1      p::first-letter {
2          font: italic 50px Times New Roman;
3          margin: 15px;
4          padding: 10px;
5          color: #f00;

```

```

6         background: #0ff;
7         border: 2px dashed #0f0;
8     }

1     <p>
2         Хотелся сделать наиболее вырвыглазный вариант, отвратный прям
           ↪ максимально и бесповоротно
3     </p>
4     <p>
5         А еще лучше - в два параграфа, чтобы вдвойне вырвыглазнее
6     </p>

```