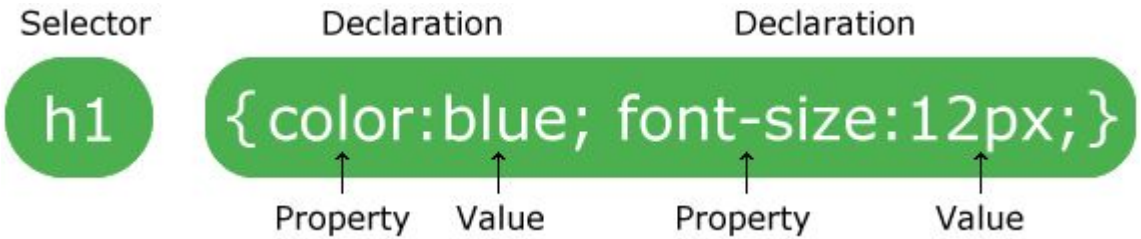
CSS 3

Основы CSS

CSS означает **C**ascading **S**tyle **S**heets (Каскадная Таблица Стилей).
CSS описывает **как HTML элементы должны отображаться на экране, бумаге или других носителях.**
CSS **экономит много работы**. Он может управлять макетом нескольких веб страниц одновременно
Определения стилей обычно сохраняются во внешних файлах .css.
С помощью внешнего файла таблицы стилей, можно изменить внешний вид всего веб сайта, изменив только один файл!
Объявление стиля состоит из двух частей: **селектора** и **объявления**. Объявление состоит из двух частей: имя свойства (например, `color`) и значение свойства (`blue`). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.
Каждое объявление содержит имя свойства CSS и значение, разделенные двоеточием.
Несколько объявлений CSS разделяются точкой с запятой, а блоки объявлений окружаются фигурными скобками.



Виды таблиц стилей

Встроенные стили

Когда мы пишем **встроенные стили**, мы пишем CSS-код в HTML-файл, непосредственно внутри тега элемента с помощью атрибута `style`:

```
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст. </p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

Внутренние стили

Внутренние стили встраиваются в раздел `<head></head>` HTML-документа и определяются внутри тега `<style></style>`. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут `style`).

Виды таблиц стилей

Внешняя таблица стилей

Внешняя таблица стилей представляет собой текстовый файл с расширением `.css`, в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью тега `<link>`, расположенного внутри раздела `<head></head>`. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов `<link>`, указав в атрибуте тега `media` назначение данной таблицы стилей. `rel="stylesheet"` указывает тип ссылки (ссылка на таблицу стилей).

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

Атрибут `type="text/css"` не является обязательным по стандарту HTML5, поэтому его можно не указывать. Если атрибут отсутствует, по умолчанию используется значение `type="text/css"`.

Виды таблиц стилей

Правило @import

Правило `@import` позволяет загружать внешние таблицы стилей. Чтобы директива `@import` работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
    @import url(mobile.css);
    p {
        font-size: 0.9em;
        color: grey;
    }
</style>
```

Правило `@import` также используется для подключения веб-шрифтов:

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,cyrillic)
```

Виды селекторов

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

Универсальный селектор

Соответствует любому HTML-элементу. Например, `* {margin: 0;}` обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: `*:after {CSS-стили}`, `*:checked {CSS-стили}`.

Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. Например, `h1 {font-family: Lobster, cursive;}` задаст общий стиль форматирования всех заголовков `h1`.

Виды селекторов

Селектор класса

Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта. Например, для создания заголовка с классом `headline` необходимо добавить атрибут `class` со значением `headline` в открывающий тег `<h1>` и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

```
<h1 class="headline">Инструкция пользования персональным компьютером</h1>
```

```
.headline {
    text-transform: uppercase;
    color: lightblue;
}
```

Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

```
<h1 class="headline post-title">Инструкция пользования персональным компьютером</h1>
```

Виды селекторов

Селектор идентификатора

Селектор идентификатора позволяет форматировать **один** конкретный элемент. Значение `id` должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

```
<div id="sidebar"></div>
```

```
#sidebar {
    width: 300px;
    float: left;
}
```

Виды селекторов

Селектор потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. Например, `ul li {text-transform: uppercase;}` — выберет все элементы `li`, являющиеся потомками всех элементов `ul`.

Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс:

- `p.first a {color: green;}`— данный стиль применится ко всем ссылкам, потомкам абзаца с классом `first`;
- `p .first a {color: green;}`— если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого тега класса `.first`, который является потомком элемента `<p>`;
- `.first a {color: green;}`— данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом `.first`.

Виды селекторов

Дочерний селектор

Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

Например, `p > strong` — выберет все элементы `strong`, являющиеся дочерними по отношению к элементу `p`.

Сестринский(соседний) селектор

Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня:

- `h1 + p` — выберет все первые абзацы, идущие непосредственно за любым тегом `<h1>`, не затрагивая остальные абзацы;
- `h1 ~ p` — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку `h1` и идущие сразу после него.

Виды селекторов

Селектор атрибута

Селекторы атрибутов выбирают элементы на основе имени атрибута или значения атрибута:

- `[атрибут]` — все элементы, содержащие указанный атрибут, `[alt]` — все элементы, для которых задан атрибут `alt`;
- `селектор[атрибут]` — элементы данного типа, содержащие указанный атрибут, `img[alt]` — только картинки, для которых задан атрибут `alt`;
- `селектор[атрибут="значение"]` — элементы данного типа, содержащие указанный атрибут с конкретным значением, `img[title="flower"]` — все картинки, название которых содержит слово `flower`;
- `селектор[атрибут~="значение"]` — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел, `p[class~="feature"]` — абзацы, имя класса которых содержит `feature`;
- `селектор[атрибут|="значение"]` — элементы, список значений атрибута которых начинается с указанного слова, `p[class|="feature"]` — абзацы, имя класса которых `feature` или начинается на `feature`;
- `селектор[атрибут^="значение"]` — элементы, значение атрибута которых начинается с указанного значения, `a[href^="http://"]` — все ссылки, начинающиеся на `http://`;
- `селектор[атрибут$="значение"]` — элементы, значение атрибута которых заканчивается указанным значением, `img[src$=".png"]` — все картинки в формате `png`;
- `селектор[атрибут*="значение"]` — элементы, значение атрибута которых содержит в любом месте указанное слово, `a[href*="book"]` — все ссылки, название которых содержит `book`.

Виды селекторов

Селектор псевдокласса

Псевдоклассы — это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Псевдоклассы характеризуют элементы со следующими свойствами:

- `:link` — не посещенная ссылка;
- `:visited` — посещенная ссылка;
- `:hover` — любой элемент, по которому проводят курсором мыши;
- `:focus` — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;
- `:active` — элемент, который был активизирован пользователем;
- `:valid` — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;
- `:invalid` — поля формы, содержимое которых не соответствует указанному типу данных;
- `:enabled` — все активные поля форм;
- `:disabled` — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;
- `:in-range` — поля формы, значения которых находятся в заданном диапазоне;
- `:out-of-range` — поля формы, значения которых не входят в установленный диапазон;
- `:lang()` — элементы с текстом на указанном языке;
- `:not(селектор)` — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — `:not([type="submit"])`;
- `:target` — элемент с символом `#`, на который ссылаются в документе;
- `:checked` — выделенные (выбранные пользователем) элементы формы.

Виды селекторов

Селектор структурных псевдоклассов

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

- `:nth-child(odd)` — нечётные дочерние элементы;
- `:nth-child(even)` — чётные дочерние элементы;
- `:nth-child(3n)` — каждый третий элемент среди дочерних;
- `:nth-child(3n+2)` — выбирает каждый третий элемент, начиная со второго дочернего элемента `(+2)`;
- `:nth-child(n+2)` — выбирает все элементы, начиная со второго;
- `:nth-child(3)` — выбирает третий дочерний элемент;
- `:nth-last-child()` — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с `:nth-child()`, но начиная с последнего, в обратную сторону;
- `:first-child` — позволяет оформить только самый первый дочерний элемент тега;
- `:last-child` — позволяет форматировать последний дочерний элемент тега;
- `:only-child` — выбирает элемент, являющийся единственным дочерним элементом;
- `:empty` — выбирает элементы, у которых нет дочерних элементов;
- `:root` — выбирает элемент, являющийся корневым в документе — элемент `html`.

Виды селекторов

Селектор структурных псевдоклассов типа

Указывают на конкретный тип дочернего тега:

- `:nth-of-type()` — выбирает элементы по аналогии с `:nth-child()`, при этом берёт во внимание только тип элемента;
- `:first-of-type` — выбирает первый дочерний элемент данного типа;
- `:last-of-type` — выбирает последний элемент данного типа;
- `:nth-last-of-type()` — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;
- `:only-of-type` — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

Виды селекторов

Селектор псевдоэлемента

Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства `content`:

- `:first-letter` — выбирает первую букву каждого абзаца, применяется только к блочным элементам;
- `:first-line` — выбирает первую строку текста элемента, применяется только к блочным элементам;
- `:before` — вставляет генерируемое содержимое перед элементом;
- `:after` — добавляет генерируемое содержимое после элемента.

Комбинация и группировка селекторов

Комбинация селекторов

Для более точного отбора элементов для форматирования можно использовать комбинации селекторов:

- `a[href][title]` — выберет все ссылки, для которых заданы атрибуты `href` и `title`;
- `img[alt*="css"]:nth-of-type(even)` — выберет все четные картинки, альтернативный текст которых содержит слово `css`.

Группировка селекторов

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:

```
h1,
h2,
p,
span {
  color: tomato;
  background: white;
}
```

Наследование и каскад

Наследование и каскад — два фундаментальных понятия в CSS, которые тесно связаны между собой.

Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как `color`, `font`, `letter-spacing`, `line-height`, `list-style`, `text-align`, `text-indent`, `text-transform`, `visibility`, `white-space` и `word-spacing`. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются. Это `background`, `border`, `display`, `float` и `clear`, `height` и `width`, `margin`, `min-max-height` и `-width`, `outline`, `overflow`, `padding`, `position`, `text-decoration`, `vertical-align` и `z-index`.

Наследование и каскад

Принудительное наследование

С помощью ключевого слова `inherit` можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

Как задаются и работают CSS-стили

Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения `inherit`).

Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.

К одному элементу могут применяться стили из разных источников. Проверить, какие стили применяются, можно в режиме разработчика браузера. Для этого над элементом нужно щёлкнуть правой кнопкой мыши и выбрать пункт «Посмотреть код» (или что-то аналогичное). В правом столбце будут перечислены все свойства, которые заданы для этого элемента или наследуются от родительского элемента, а также файлы стилей, в которых они указаны, и порядковый номер строки кода.

Наследование и каскад

При определении стиля можно использовать любую комбинацию селекторов — селектор элемента, псевдокласса элемента, класса или идентификатора элемента.

```
<div id="wrap" class="box clear"></div>
```

```
div {border: 1px solid #eee;}
```

```
#wrap {width: 500px;}
```

```
.box {float: left;}
```

```
.clear {clear: both;}
```

Наследование и каскад

Каскадирование — это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила. Существует три критерия, которые определяют порядок применения свойств — правило `!important`, специфичность и порядок, в котором подключены таблицы стилей.

Правило !important

Вес правила можно задать с помощью ключевого слова `!important`, которое добавляется сразу после значения свойства, например, `span {font-weight: bold!important;}` Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Наследование и каскад

Специфичность

Для каждого правила браузер вычисляет **специфичность селектора**, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: `0, 0, 0, 0`. Специфичность селектора определяется следующим образом:

- для `id` добавляется `0, 1, 0, 0`;
- для `class` добавляется `0, 0, 1, 0`;
- для каждого элемента и псевдоэлемента добавляется `0, 0, 0, 1`;
- для встроенного стиля, добавленного непосредственно к элементу — `1, 0, 0, 0`;
- универсальный селектор не имеет специфичности.

```
h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
em {color: silver;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1, 2*/
.sidebar {color: grey;} /*специфичность 0, 0, 1, 0*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar  {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0, 1*/
```

Наследование и каскад

В результате к элементу применятся те правила, специфичность которых больше. Например, если на элемент действуют две специфичности со значениями `0, 0, 0, 2` и `0, 1, 0, 1`, то выиграет второе правило.

Порядок подключённых таблиц

Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице. Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

CSS Комментарии

Комментарии используются для объяснения кода и могут помочь при редактировании исходного кода на более позднем этапе.

Комментарии игнорируются браузерами.

CSS комментарий начинается с `/*` и заканчивается `*/`:

```
/* Это однострочный комментарий */

p {
  color: red;  /* Установите цвет текста на красный */
}

/* Это
многострочный
комментарий */
```

CSS блочная модель

Каждый блок имеет прямоугольную **область содержимого** в центре, **поля вокруг** содержимого, **рамку** вокруг полей и **отступ за пределами** рамки. Размеры этих областей определяют свойства `padding` и его подсвойства — `padding-left`, `padding-top` и т.д., `border` и его подсвойства, `margin` и его подсвойства.

Каждый блок имеет **область содержимого**, в которой находится текст, дочерние элементы, изображение и т.п., и необязательные окружающие ее `padding`, `border` и `margin`. Размер каждой области определяется соответствующими свойствами и может быть нулевым, или, в случае `margin`, отрицательным.

Поля, рамка и отступы могут быть разбиты на верхний, правый, нижний и левый сегменты, каждый из которых независимо управляется своим соответствующим свойством.

Фон области содержимого, полей и рамки блока определяется свойствами фона. Область рамки может быть дополнительно окрашена с помощью свойства `border`. Отступы элемента всегда прозрачны, что позволяет показывать фон родительского элемента.

Так как поля и отступы элемента не являются обязательными, по умолчанию их значение равно нулю. Тем не менее, некоторые браузеры добавляют этим свойствам положительные значения по умолчанию на основе своих таблиц стилей. Очистить стили браузеров для всех элементов можно при помощи универсального селектора:

```
* {
    margin: 0;
    padding: 0;
}
```

CSS блочная модель

Отступы окружают край рамки элемента, обеспечивая расстояние между соседними блоками. Свойства отступов определяют их толщину. Применяются ко всем элементам, кроме внутренних элементов таблицы. Сокращенное свойство `margin` задает отступы для всех четырех сторон, а его подсвойства задают отступ только для соответствующей стороны.

Смежные вертикальные отступы элементов в блочной модели схлопываются.

Смежные вертикальные отступы двух или более элементов уровня блока `margin` объединяются (перекрываются). При этом ширина общего отступа равна ширине большего из исходных.

Объединение отступов выполняется только для блочных элементов в нормальном потоке документа. Если среди схлопывающихся отступов есть отрицательные значения, то браузер добавит отрицательное значение к положительному, а полученный результат и будет расстоянием между элементами. Если положительных отступов нет, то максимум абсолютных значений соседних отступов вычитается из нуля.

Отступы не схлопываются:

- Между плавающим блоком и любым другим блоком;
- У плавающих элементов и элементов со значением `overflow`, отличным от `visible`, со своими дочерними элементами в потоке;
- У абсолютно позиционированных элементов, даже с их дочерними элементами;
- У строчно-блочных элементов.

CSS блочная модель

Выпадение вертикальных отступов

Если внутри одного блока расположить другой блок и задать ему `margin-top`, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху, т.е. внутренний блок «выпадет» из родительского блока. Если у родительского элемента также был задан верхний отступ, то выберется наибольшее из значений.

Чтобы избавиться от эффекта выпадения, можно задать родительскому элементу `padding-top` или добавить `border-top: 1px solid transparent`.

Физические свойства отступов: свойства margin-top, margin-right, margin-bottom, margin-left

Свойства устанавливают верхний, правый, нижний и левый отступ блока элемента соответственно. Отрицательные значения допускаются, но могут существовать ограничения для конкретной реализации.

Свойства не наследуются.

CSS блочная модель

margin-top/margin-right/margin-bottom/margin-left

Значения:
`длина` — Размер отступа задается в единицах длины, например, `px`, `in`, `em`. Значение по умолчанию `0`.
`%` — Вычисляется относительно ширины блока контейнера. Изменяются, если изменяется ширина родительского элемента.
`auto` — Для элементов уровня строки, плавающих (`float`) значения `margin-left` или `margin-right` вычисляются в `0`. Если для элементов уровня блока задано `margin-left: auto` или `margin-right: auto` — соответствующее поле расширяется до края содержащего блока, если оба — их значения становятся равными, что горизонтально центрирует элемент относительно краев содержащего блока.
`initial` — Устанавливает значение свойства в значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

Синтаксис

```
margin-top: 20px;
margin-right: 1em;
margin-bottom: 5%;
margin-left: auto;
margin-top: inherit;
margin-right: initial;
```

CSS блочная модель

Краткая запись отступов: свойство margin

Свойство `margin` является сокращенным свойством для установки `margin-top`, `margin-right`, `margin-bottom` и `margin-left` в одном объявлении.

Если существует только одно значение, оно применяется ко всем сторонам.

Если два — верхний и нижний отступы устанавливаются на первое значение, а правый и левый — устанавливаются на второе.

Если имеется три значения — верхний отступ устанавливается на первое значение, левый и правый — на второе, а нижний — на третье.

Если есть четыре значения — они применяются сверху, справа, снизу и слева соответственно.

CSS блочная модель

Поля элемента

Область полей представляет собой пространство между краем области содержимого и рамкой элемента. Свойства полей определяют толщину их области. Применяются ко всем элементам, кроме внутренних элементов таблицы (за исключением ячеек таблицы). Сокращенное свойство `padding` задает поля для всех четырех сторон, а подсвойства устанавливают только их соответствующие стороны.

Фоны элемента по умолчанию закрашивают поля элемента и пространство под его рамкой. Это поведение можно настроить с помощью свойств background-origin и background-clip.

Физические свойства полей: свойства padding-top, padding-right, padding-bottom, padding-left

Свойства устанавливают верхнее, правое, нижнее и левое поля соответственно. Отрицательные значения недопустимы.

Свойства не наследуются.

CSS блочная модель

padding-top/padding-right/padding-bottom/padding-left

Значения:

`длина` — Поля элемента задаются при помощи единиц длины, например, `px`, `pt`, `cm`. Значение по умолчанию `0`.
`%` — Вычисляются относительно ширины родительского элемента, могут меняться при изменении ширины элемента. Поля сверху и снизу равны полям слева и справа, т.е. верхние и нижние поля тоже вычисляются относительно ширины элемента.
`initial` — Устанавливает значение свойства в значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

Синтаксис:

```
padding-top: 0.5em;
padding-right: 0;
padding-bottom: 2cm;
padding-left: 10%;
padding-top: inherit;
padding-bottom: initial;
```

CSS блочная модель

Краткая запись полей: свойство padding

Свойство `padding` является сокращенным свойством для установки `padding-top`, `padding-right`, `padding-bottom` и `padding-left` в одном объявлении. Поведение аналогично `margin`.

Рамки элемента

Рамки элемента заполняют область рамок, визуально очерчивая края блока. Свойства рамок определяют толщину области границы блока, а также ее стиль и цвет.

Блочные и строчные элементы

Выделяют две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы — **блочные** и **строчные элементы**. С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов `<area>` и `<img>`).

Модель визуального форматирования

HTML-документ организован в виде дерева элементов и текстовых узлов. Модель визуального форматирования CSS представляет собой алгоритм, который обрабатывает HTML-документ и выводит его на экран устройства.

Каждый блок в дереве представляет соответствующий элемент или псевдоэлемент, а текст (буквы, цифры, пробелы), находящийся между открывающим и закрывающим тегами, представляет содержимое текстовых узлов.

Чтобы создать дерево блоков, CSS сначала использует каскадирование и наследование, позволяющие назначить вычисленное значение для каждого css-свойства каждому элементу и текстовому узлу в исходном дереве.

Блочные и строчные элементы

Затем для каждого элемента CSS генерирует ноль или более блоков в соответствии со значением свойства `display` этого элемента. Как правило, элемент генерирует один основной блок, который представляет самого себя и содержит свое содержимое. Некоторые значение свойства `display`, например, `display: list-item;`, генерируют блок основного блока и блок дочернего маркера. Другие, например, `display: none;`, приводят к тому, что элемент и/или его потомки вообще не генерируют блоки.

Положение блоков на странице определяется следующими факторами:

- размером элемента (с учётом того, заданы они явно или нет);
- типом элемента (строчный или блочный);
- схемой позиционирования (нормальный поток, позиционированные или плавающие элементы);
- отношениями между элементами в DOM (родительский — дочерний элемент);
- внутренними размерами содержащихся изображений;
- внешней информацией (например, размеры окна браузера).

Блочные и строчные элементы

Блочные элементы и блочные контейнеры

Блочные элементы — элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства `display`, такие как `block`, `list-item` и `table` делают элементы блочными. Блочные элементы генерируют основной блок, который содержит только блок элемента. Элементы со значением `display: list-item` генерируют дополнительные блоки для маркеров, которые позиционируются относительно основного блока.

Блочные элементы могут размещаться непосредственно внутри элемента `<body>`. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя.

Блочные элементы могут размещаться непосредственно внутри элемента `<body>`. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя. Блочные элементы могут содержать как строчные, так и блочные элементы.

Элемент `<p>` относится к блочным элементам, но он не должен содержать внутри себя другой элемент `<p>`, а также любой другой блочный элемент.

Блочные и строчные элементы

Строчные элементы и строчные контейнеры

Встроенные (строчные) элементы генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства `display`, такие как `inline` и `inline-table` делают элементы строчными.

Строчные элементы могут содержать только данные и другие строчные элементы. Исключение составляет элемент `<a>`, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

Строчно-блочные элементы

Существует еще одна группа элементов, которые браузер обрабатывает как строчно-блочные `{display: inline-block;}`. Такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.

Блочные и строчные элементы

Ширина содержимого: свойство width

Свойство `width` определяет ширину содержимого блока.

Это свойство не применяется к незамещаемым строчным элементам `display: inline;`. Ширина содержимого встроенных блоков определяется шириной отображаемого содержимого внутри них. Встроенные блоки сливаются в линейные блоки. Ширина линейных блоков определяется шириной содержащего блока, но может быть уменьшена из-за наличия свойства `float`.

Отрицательные значения не допускаются. Свойство не наследуется.

Значения:
`длина` — Ширина элемента задается в единицах длины, например, `px`, `em` и т.д.
`%` — Вычисляется относительно ширины содержащего блока. Для абсолютно позиционированных элементов процент вычисляется с учетом ширины области отступов `padding` содержащего блока.
`auto` — Ширина вычисляется в зависимости от значений других свойств. Значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

Блочные и строчные элементы

Минимальная и максимальная ширина: свойства min-width и max-width

Свойства `min-width` и `max-width` позволяют ограничивать ширину содержимого до определенного диапазона. Значения не могут быть отрицательными. Для `min-width` значение по умолчанию `0`, для `max-width` — `none`.

Свойства не наследуются.

Значения:

`длина` — Задает фиксированную минимальную или максимальную используемую ширину.
`%` — Указывает процент для определения используемого значения. Процент рассчитывается относительно ширины содержащего блока.
`none` — Означает отсутствие ограничений ширины блока.
`inherit` — Наследует значение свойства от родительского элемента.

Блочные и строчные элементы

Высота содержимого: свойство height

Свойство `height` определяет высоту содержимого блока. Это свойство не применяется к незамещаемым строчным элементам. Значения длины не могут быть отрицательными.

Свойство не наследуется.

Значения:
`длина` — Высота области содержимого задается в единицах длины.
`%` — Задает высоту в процентах. Процент рассчитывается относительно высоты содержащего блока. Если высота содержащего блока не указана явно (то есть зависит от высоты содержимого) и этот элемент не является абсолютно позиционированным, значение вычисляется как `auto`. Для абсолютно позиционированных элементов процент вычисляется с учетом высоты области отступов `padding` содержащего блока.
`auto` — Высота зависит от значений других свойств. Значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

Блочные и строчные элементы

Минимальная и максимальная высота: свойства min-height и max-height

Иногда полезно ограничить высоту элементов определенным диапазоном. Свойства `min-height` и `max-height` предлагают эту функциональность.

Значения:
длина — Задает фиксированную минимальную или максимальную вычисленную высоту в единицах длины. Значения не могут быть отрицательными.
`%` — Указывает процент для определения используемого значения. Процент рассчитывается относительно высоты содержащего блока. Если высота содержащего блока не указана явно (т.е. зависит от высоты содержимого) и этот элемент не является абсолютно позиционированным, процентное значение обрабатывается как `0` для `min-height` или `none` для `max-height`.
`none` — Отсутствие ограничений высоты блока, только для `max-height`.
`inherit` — Наследует значение свойства от родительского элемента.

Блочные и строчные элементы

Расчет высоты строки: свойства line-height и vertical-align

Как описано выше, пользовательские агенты (браузеры) передают блоки встроенного уровня в вертикальный стек линейных блоков. Высота линейного блока определяется следующим образом:

- Высота каждого встроенного прямоугольника в линейном блоке вычисляется. Для замещаемых, `inline-block` и `inline-table` элементов это высота их области поля (margin box).
- Блоки уровня строки выравниваются вертикально в соответствии со значением свойства `vertical-align`. Если они выровнены по верху или по низу, они должны быть выровнены так, чтобы минимизировать высоту линейного блока.

Высота линейного блока — это расстояние между самой верхней и самой нижней частью блока. Пустые встроенные элементы генерируют пустые встроенные блоки, но эти блоки по-прежнему имеют поля, отступы, границы, высоту строки и, таким образом, влияют на эти вычисления также, как и элементы с содержимым.

В элементе уровня блока, содержимое которого состоит из элементов встроенного уровня, свойство `line-height` определяет минимальную высоту линейных блоков внутри элемента. Минимальная высота состоит из минимальной высоты над базовой линией и минимальной глубины под ней.

Для элементов уровня строки свойство `line-height` указывает высоту, которая используется при расчете высоты линейного блока.

Отрицательные значения не допустимы.

Свойство наследуется.

Блочные и строчные элементы

Значения line-height:

`normal` — Сообщает пользовательским агентам установить «разумное» значение на основе шрифта элемента. Значение по умолчанию. Когда элемент содержит текст, отображаемый более чем одним шрифтом, пользовательские агенты могут определить значение `normal` в соответствии с наибольшим размером шрифта.

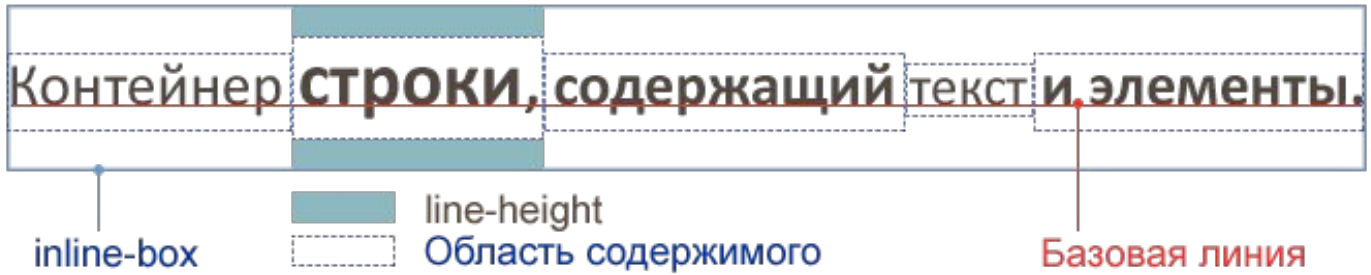
`длина` — Значение задаётся в единицах длины, создавая фиксированное значение высоты строки. Если задать значение меньше единицы, смежные строки будут находить друг на друга.

`число` — Используемое значение свойства — это число, умноженное на размер шрифта элемента.

`%` — Вычисленное значение свойства — это процент, умноженный на вычисленный размер шрифта элемента.

Синтаксис:

```
line-height: normal;
line-height: 2em;
line-height: 1.5;
line-height: 50%;
line-height: inherit;
```



Блочные и строчные элементы

Свойство `vertical-align` влияет на вертикальное позиционирование в линейном блоке элементов уровня строки: `display: inline` и `display: table-cell`. Значения этого свойства имеют другие значения в контексте таблиц.

Значения:
`baseline` — Выравнивает базовую линию элемента по базовой линии его родителя, совмещая среднюю линию элемента со средней линией родительского элемента.
`sub` — Делает элемент подстрочным (аналогично с тегом `<sub>`). Величина понижения элемента может меняться в зависимости от браузера пользователя.
`super` — Делает элемент надстрочным (аналогично с тегом `<sup>`). При этом значения `sup` и `super` не меняют размер шрифта, по умолчанию текст надстрочного и подстрочного элемента имеет такой же размер, как и текст родительского элемента.
`top` — Верхний край элемента совмещается с верхним краем самого высокого элемента в линии.
`text-top` — Верхний край элемента совмещается с верхним краем шрифта родительского элемента.
`middle`Средняя линия элемента (обычно изображения) совмещается с линией, проходящей через середину родительского элемента.
`bottom` — Нижний край элемента совмещается с нижним краем самого низкого элемента в линии.
`text-bottom` — Нижний край элемента совмещается с нижним краем шрифта родительского элемента.
`%` — Не позволяет устанавливать `middle`, вычисляется как часть `line-height` элемента, а не его родителя, т.е. если установить значение `vertical-align`, равное `50%` для элемента с `line-height` равным `20px`, то базовая линия элемента поднимется на `10px`.
`длина` — Устанавливает значение в единицах длины, перемещая элемент на заданное расстояние.

Блочные и строчные элементы

Изменение блочной модели: свойство box-sizing

Свойство `box-sizing` переключает блочную модель с фиксированных размеров длины и ширины на `content-box` и `border-box`. Это влияет на интерпретацию всех свойств, определяющих размеры, включая `flex-basis`.

Свойство не наследуется.

Значения:
`content-box` — Заданные ширина и высота (и соответствующие min/max-свойства) применяются к ширине и высоте области содержимого элемента. Поля `padding` и рамка `border` элемента располагаются за пределами указанной ширины и высоты. Значение по умолчанию.
`border-box` — Любые `padding` или `border`, заданные для элемента, размечаются и отрисовываются внутри указанных значений ширины и высоты. Ширина и высота содержимого вычисляются путем вычитания ширины границ и полей соответствующих сторон из указанных свойств ширины и высоты. Значение `auto` свойств `width` и `height` не зависит от свойства `box-sizing` и всегда устанавливает размер блока с содержимым. Сумма `padding` и `border` не должна превышать заданные значения `width` и `height`, в противном случае размер области содержимого будет равен нулю.

CSS-позиционирование

CSS рассматривает макет html-документа как дерево элементов. Уникальный элемент, у которого нет родительского элемента, называется **корневым** элементом. Модуль CSS-позиционирование описывает, как любой из элементов может быть размещен независимо от порядка документа (т.е. извлечен из «потока»).

Расположение этих блоков регулируется:

- размерами и типом элемента,
- схемой позиционирования (нормальный поток, обтекание и абсолютное позиционирование),
- отношениями между элементами в дереве документа,
- внешней информацией (например, размер области просмотра, внутренними размерами изображений и т. д.).

CSS-позиционирование

В CSS блок элемента может быть расположен в соответствии с тремя схемами позиционирования:

Нормальный поток

Нормальный поток включает блочный контекст форматирования (элементы с `display` `block`, `list-item` или `table`), строчный (встроенный) контекст форматирования (элементы с `display` `inline`, `inline-block` или `inline-table`), и относительное и «липкое» позиционирование элементов уровня блока и строки.

Обтекание

В обтекающей модели блок удаляется из нормального потока и позиционируется влево или вправо. Содержимое обтекает правую сторону элемента с `float: left` и левую сторону элемента с `float: right`.

Абсолютное позиционирование

В модели абсолютного позиционирования блок полностью удаляется из нормального потока и ему присваивается позиция относительно содержащего блока. Абсолютное позиционирование реализуется с помощью значений `position: absolute;` и `position: fixed;`.

CSS-позиционирование

Содержащий блок

Положение и размер блока(ов) элемента иногда вычисляются относительно некоторого прямоугольника, называемого **содержащим блоком** элемента (containing block). В общих словах, содержащий блок — это блок, который содержит другой элемент. В случае нормального потока корневой элемент `html` является содержащим блоком для элемента `body`, который, в свою очередь, является содержащим блоком для всех его дочерних элементов и так далее. В случае позиционирования содержащий блок полностью зависит от типа позиционирования. Содержащий блок элемента определяется следующим образом:

- Содержащий блок, в котором находится корневой элемент, представляет собой прямоугольник — так называемый **начальный содержащий блок**.
- Для некорневого элемента с `position: static;` или `position: relative;` содержащий блок формируется краем области содержимого ближайшего родительского блока уровня блока, ячейки таблицы или уровня строки.
- Содержащим блоком элемента с `position: fixed;` является окно просмотра.
- Для некорневого элемента с `position: absolute;` содержащим блоком устанавливается ближайший родительский элемент со значением `position: absolute/relative/fixed`следующим образом:
  — если предок — элемент уровня блока, содержащим блоком будет область содержимого плюс поля элемента `padding`;
  — если предок — элемент уровня строки, содержащим блоком будет область содержимого;
  — если предков нет, то содержащий блок элемента определяется как начальный содержащий блок.
- Для «липкого» блока содержащим блоком является ближайший предок с прокруткой или окно просмотра, в противном случае.

CSS-позиционирование

Выбор схемы позиционирования: свойство position

Свойство `position` определяет, какой из алгоритмов позиционирования используется для вычисления положения блока.

Свойство не наследуется.

Значения:
`static` — Блок располагается в соответствии с нормальным потоком. Свойства `top`, `right`, `bottom` и `left` не применяются. Значение по умолчанию.
`relative` — Положение блока рассчитывается в соответствии с нормальным потоком. Затем блок смещается относительно его нормального положения и во всех случаях, включая элементы таблицы, не влияет на положение любых следующих блоков. Тем не менее, такое смещение может привести к перекрытию блоков, а также к появлению полосы прокрутки в случае переполнения.
Относительно позиционированный блок сохраняет свои размеры, включая разрывы строк и пространство, первоначально зарезервированное для него.

Относительно позиционированный блок создает новый содержащий блок для абсолютно позиционированных потомков.

CSS-позиционирование

`absolute` — Положение блока (и, возможно, размер) задается с помощью свойств `top`, `right`, `bottom` и `left`. Эти свойства определяют явное смещение относительно его содержащего блока. Абсолютно позиционированные блоки полностью удаляется из нормального потока, не влияя на расположение сестринских элементов.
Отступы `margin` абсолютно позиционированных блоков не схлопываются.

Абсолютно позиционированный блок создает новый содержащий блок для дочерних элементов нормального потока и потомков с `position: absolute;`.

Содержимое абсолютно позиционированного элемента не может обтекать другие блоки. Абсолютно позиционированный блок могут скрывать содержимое другого блока (или сами могут быть скрыты), в зависимости от значения `z-index` перекрывающихся блоков.

`sticky` — Положение блока рассчитывается в соответствии с нормальным потоком. Затем блок смещается относительно своего ближайшего предка с прокруткой или окна просмотра, если ни у одного из предков нет прокрутки.
«Липкий» блок может перекрывать другие блоки, а также создавать полосы прокрутки в случае переполнения.

«Липкий» блок сохраняет свои размеры, включая разрывы строк и пространство, первоначально зарезервированное для него.

«Липкий» блок создает новый содержащий блок для абсолютно и относительно позиционированных потомков.

CSS-позиционирование

`fixed` — Фиксированное позиционирование аналогично абсолютному позиционированию, с отличием в том, что для содержащим блоком устанавливается окно просмотра. Такой блок полностью удаляется из потока документа и не имеет позиции относительно какой-либо части документа. Фиксированные блоки не перемещаются при прокрутке документа. В этом отношении они похожи на фиксированные фоновые изображения.
При печати фиксированные блоки повторяются на каждой странице, содержащим блоком для них устанавливается область страницы. Блоки с фиксированным положением, которые больше области страницы, обрезаются.

CSS-позиционирование

Смещение блока: свойства top, right, bottom, left

Элемент считается позиционированным, если свойство `position` имеет значение, отличное от `static`. Позиционированные элементы генерируют позиционированные блоки и могут быть расположены в соответствии со следующими четырьмя физическими свойствами: top, right, bottom, left

Значение:
`auto` — Влияние значения зависит от типа элемента. Значение по умолчанию.
`длина` — Смещение на фиксированном расстоянии от указанного края. Отрицательные значения допускаются.
`%` — Процентные значения вычисляются относительно высоты содержащего блока. Для «липкого» блока — относительно высоты корневого элемента. Отрицательные значения допускаются.
`initial` — Устанавливает значение свойства в значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

Положительные значения смещают элемент внутрь содержащего блока, а отрицательные — за его пределы.

CSS-позиционирование

Обтекание: свойство float

Обтекание позволяет блокам смещаться влево или вправо на текущей строке. «Плавающий блок» смещается влево или вправо до тех пор, пока его внешний край не коснется края содержащего блока или внешнего края другого плавающего блока. Если имеется линейный блок, внешняя верхняя часть плавающего блока выравнивается с верхней частью текущего линейного блока.

При использовании свойства `float` для элементов рекомендуется задавать ширину. Тем самым браузер создаст место для другого содержимого. Если для плавающего элемента недостаточно места по горизонтали, он будет смещаться вниз до тех пор, пока не уместится. При этом остальные элементы уровня блока будут его игнорировать, а элементы уровня строки будут смещаться вправо или влево, освобождая для него пространство и обтекая его.

Правила, регулирующие поведение плавающих боков, описываются свойством `float`.

Свойство не наследуется.

Значение:
`none` — Отсутствие обтекания. Значение по умолчанию.
`left` — Элемент перемещается влево, содержимое обтекает плавающий блок по правому краю.
`right` — Элемент перемещается вправо, содержимое обтекает плавающий блок по левому краю.
`inherit` — Наследует значение свойства от родительского элемента.

CSS-позиционирование

Плавающий блок принимает размеры своего содержимого с учетом внутренних отступов и рамок. Верхние и нижние отступы `margin` плавающих элементов не схлопываются.

Плавающие элементы могут использовать отрицательные отступы `margin`, чтобы перемещаться за пределы области содержимого их родительского элемента.

Свойство не оказывает влияние на элементы с `display: flex` и `display: inline-flex`. Не применяется к абсолютно позиционированным элементам.

CSS-позиционирование

Управление потоком рядом с плавающими элементами: свойство clear

Свойство `clear` указывает, какие стороны блока/блоков элемента не должны прилегать к плавающим блокам, находящемся выше в исходном документе. Свойство применяется только к неплавающим элементам уровня блока.

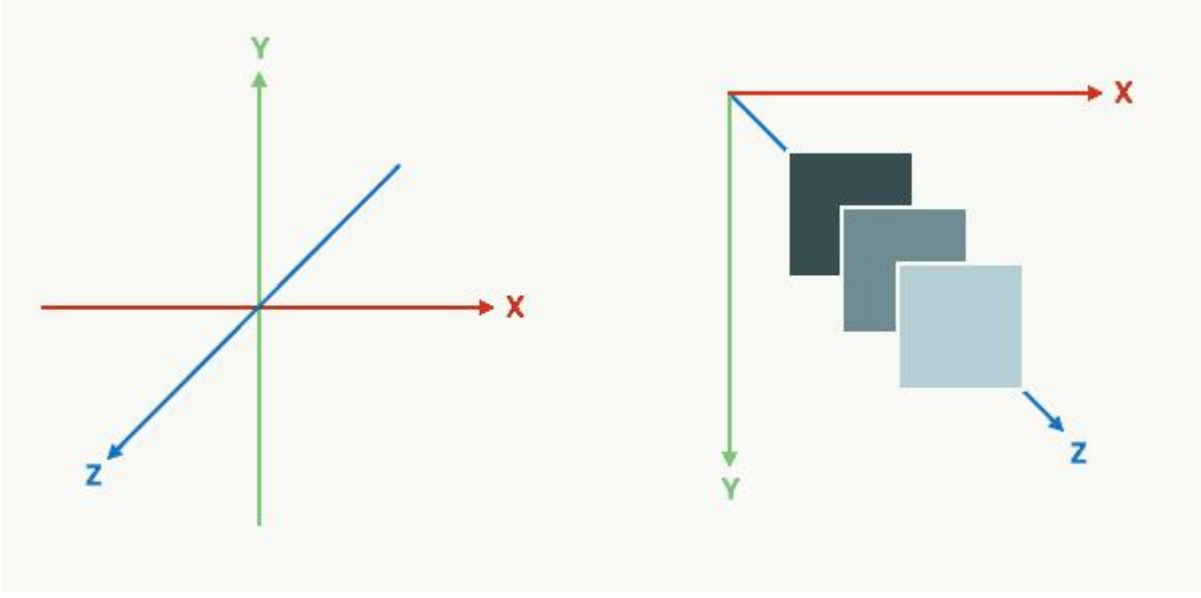
Значение:
`none` — Означает отсутствие ограничений на положение элемента относительно плавающих блоков. Значение по умолчанию.
`left` — Смещает элемент вниз относительно нижнего края любого плавающего слева элемента, находящемся выше в исходном документе.
`right` — Смещает элемент вниз относительно нижнего края любого плавающего справа элемента, находящемся выше в исходном документе.
`both` — Смещает элемент вниз относительно нижнего края любого плавающего слева и справа элемента, находящемся выше в исходном документе.
`inherit` — Наследует значение свойства от родительского элемента.

Для предотвращения отображение фона или границ под плавающими элементами используется правило `{overflow: hidden;}`.

CSS-позиционирование

Определение контекста наложения: свойство z-index

В CSS каждый блок имеет позицию в трех измерениях. В дополнение к горизонтальному и вертикальному положению, блоки выкладываются вдоль оси Z друг над другом. Положение вдоль оси Z особенно важно, когда блоки визуально накладываются друг на друга.



CSS-позиционирование

Порядок, в котором дерево документа отрисовывается на экране, описывается с помощью **контекста наложения**. Каждый блок принадлежит одному контексту наложения. Каждый блок в данном контексте наложения имеет целочисленный уровень, который является его положением на оси Z относительно других блоков в том же контексте наложения.

Блоки с более высокими уровнями всегда отображаются перед блоками с более низкими уровнями, а блоки с одинаковым уровнем располагаются снизу вверх в соответствии с порядком следования элементов в исходном документе. Блок элемента имеет ту же позицию, что и блок его родителя, если только ему не присвоен другой уровень свойством `z-index`.

Свойство `z-index` позволяет изменить порядок наложения позиционированных элементов в случае, когда они накладываются друг на друга.

CSS-позиционирование

Значение:
`auto` — Вычисляется в `0`. Если для блока задано `position: fixed;` или это корневой элемент, значение `auto` также устанавливает новый контекст наложения. Значение по умолчанию.
`целое число` — Определяет положение блока в текущем контексте наложения. Также устанавливает новый локальный контекст наложения. Можно использовать любое целое число, включая отрицательные числа. Отрицательные значения помещают элемент вглубь экрана.
`inherit` — Наследует значение свойства от родительского элемента.
`initial` — Устанавливает значение свойства в значение по умолчанию.

CSS-позиционирование

Если для элементов свойства `z-index` и `position` не заданы явно, контекст наложения равен порядку их расположения в исходном коде и браузер отображает элементы на странице в следующем порядке:

- Корневой элемент `<html>`, который содержит все элементы веб-странице.
- Блочные элементы, неплавающие и непозиционированные.
- Плавающие `float` непозиционированные элементы в порядке их расположения в исходном коде.
- Строковые непозиционированные элементы (текст, изображения).
- Позиционированные `position` элементы в порядке их следования в исходном коде. Последний из них будет расположен на переднем плане.

Свойство `z-index` создает новый контекст наложения. Оно позволяет изменить порядок наложения **позиционированных элементов**. Элементы будут отображаться на странице в следующем порядке (если для них не заданы свойства, влияющие на контекст наложения — `opacity`, `filter`, `transform`):

- Корневой элемент `<html>`, который содержит все элементы веб-странице.
- Позиционированные элементы с отрицательным значением `z-index`.
- Блочные элементы, неплавающие и непозиционированные.
- Плавающие `float` непозиционированные элементы в порядке их расположения в исходном коде.
- Строковые непозиционированные элементы (текст, изображения).
- Позиционированные элементы со значениями `z-index: 0;` и `z-index: auto;`.

CSS-рамка

CSS-рамка элемента представляет собой одну или несколько линий, окружающих содержимое элемента и его поля `padding`. Рамка задаётся с помощью краткого свойства `border`. Стиль рамки задается с помощью трех свойств: **стиль**, **цвет** и **ширина**.

Стиль рамки border-style

По умолчанию рамки всегда отрисовываются поверх фона элемента, фон распространяется до внешнего края элемента. Стиль рамки определяет ее отображение, без этого свойства рамки не будут видны вообще. Для элемента можно задавать рамку для всех сторон одновременно с помощью свойства `border-style` или для каждой стороны отдельно с помощью уточняющих свойств `border-top-style` и т.д. Не наследуется.

border-style (border-top-style, border-right-style, border-bottom-style, border-left-style)

Значения:
`none` — Значение по умолчанию, означает отсутствие рамки. Также убирает рамку элемента из группы элементов с установленным значением данного свойства.
`hidden` — Эквивалентно `none`.
`dotted` — Определяет границу штрихом
`dashed` — Определяет границу пунктиром
`solid` — Определяет границу сплошной
`double` — Определяет границу двойной
`groove` — Определяет 3D границу желобом.
`ridge` — Определяет 3D границу коньком.
`inset` — Определяет 3D границу вставкой.
`outset` — Определяет 3D границу начальной.
`{1,4}` — Одновременное перечисление четырех разных стилей для рамок элемента, только для свойства `border-style`:
`{border-style: solid dotted none dotted;}`

CSS-рамка

Цвет рамки border-color

Свойство задаёт цвет рамок всех сторон одновременно. С помощью уточняющих свойств можно установить свой цвет для рамки каждой стороны элемента. Если для рамки цвет не задан, то он будет таким же, как и цвет текста элемента. Если в элементе нет текста, то цвет рамки будет таким же, как и цвет текста родительского элемента. Не наследуется.

border-color (border-top-color, border-right-color, border-bottom-color, border-left-color)

Значения:

`transparent` — Устанавливает прозрачный цвет для рамки. При этом ширина рамки остается. Можно использовать для смены цвета рамки при наведении курсора мыши на элемент, чтобы избежать смещение элемента.
`цвет` — Цвет рамок задается при помощи значений свойства color.
`{1,4}` — Одновременное перечисление четырех разных цветов для рамок элемента, только для свойства `border-color`:
```
{border-color: #cacd58 #5faf8a #b9cea5 #aab238;}
```
`initial` — Устанавливает значение свойства в значение по умолчанию.
`inherit` — Наследует значение свойства от родительского элемента.

CSS-рамка

Ширина рамки border-width

Ширина рамки задается с помощью единиц измерения длины или ключевых слов. Если для свойства `border-style` задано значение `none`, и для рамки элемента установлена какая-то ширина, то в данном случае ширина рамки приравнивается к нулю. Не наследуется.

border-width (border-top-width, border-right-width, border-bottom-width, border-left-width)

Значения:

`thin / medium / thick` — Ключевые слова, устанавливают ширину рамки относительно друг друга. Первое значение уже, чем второе, второе — тоньше третьего. Значение по умолчанию — `medium`

`width (px, pt, cm, em, т.п)` — **{border-width: 5px;}**

`{1,4}` — **Возможность одновременного задания четырех разных ширин для рамок элемента, только для свойства `border-width`:**

```
{border-width: 5px 10px 15px 3px;}
```

`initial` — Устанавливает значение свойства в значение по умолчанию.

`inherit` — Наследует значение свойства от родительского элемента.

CSS-рамка

Задание рамки одним свойством

Свойство `border` позволяет объединить в себе следующие свойства: `border-width`, `border-style`, `border-color`, например:

```
div {
    width: 100px;
    height: 100px;
    border: 2px solid grey;
}
```

При этом заданные свойства будут применяться ко всем границам элемента одновременно. Если какое-то из значений не указано, его место займет значение по умолчанию.

CSS-рамка

Задание рамки для одной границы элемента

В случае, когда необходимо задать разный стиль границ элемента, можно воспользоваться краткой записью для соответствующей границы.

Перечисленные ниже свойства объединяют в одно объявление следующие свойства: `border-width`, `border-style` и `border-color`. Перечень свойств указывается в заданном порядке, при этом одно или два значения могут быть пропущены, в этом случае их значения примут значения по умолчанию.

Стиль верхней границы задается с помощью свойства `border-top`, нижней — `border-bottom`, левой — `border-left`, правой — `border-right`.

Синтаксис

```
p {border-top: 2px solid grey;}
```