**ЧЕКБОКСЫ,БЛОКИ, DISPLAY-FLEX**

Чекбокс - это кнопка или специальный символ, который служит для заполнения специальных полей в анкетах, бюллетенях, опросах. Чекбокс - это ничто иное как графический элемент, используемый в различных компьютерных приложениях для обозначения таких параметров как quot;даquot; (включить), quot;нетquot; (выключить). обозначается чекбокс как пустой квадратик. Чекбокс - это инструмент интерфейса, который служит для обозначения какой-либо группы (например, для определения конкретных опций) и дальнейшей работы над ней, а также для обозначения состояния (вкл/выкл). quot;Внешностьquot; у этого флажка квадратная, но может быть, например, и круглой:Для того, чтобы активировать чекбокс, его quot;телоquot; заполняется галочкой, крестиком, либо же квадратиком или иным графическим элементом.

Блоки в HTML обозначаются <div></div>. Всему, что размещено в блоке можно придать значение в файле styles.css, когда пользуюсь внешним стилем.

В самом доке HTML значения задаю, когда пользуюсь глобальным и внутренним стилем, тогда внешний не использую. Внешний стиль объединяет HTML и css тегом <link rel="stylesheet" type="text/css" href="styles.css"> . Когда блоки выведены на экран в хроме, нажав кнопку F-12, значения и размеры блоков, а так же их размещение на странице можно регулировать в табличке, которая выйдет после нажатия F-12.

Блокам задаются и регулируются размеры и границы с помощью стилей css:

Padding;

Border;

Margin, Margin-top, Margin-bottom, Margin-left и right , Margin отрицательный( используется редко);

Width: ;

Height: ;

**Position: static;** Элемент находится в обычном состоянии, расположен на своём месте в документе. Свойства top, right, bottom, left и z-index не применяются к такому элементу. Это значение по умолчанию.

display: inline-block; - выравнивает блоки по горизонту

position: relative; - позволяет располжить блок как мне хочется добавив нужное мне расположение, например, top:100px; left:20px; и т.д.

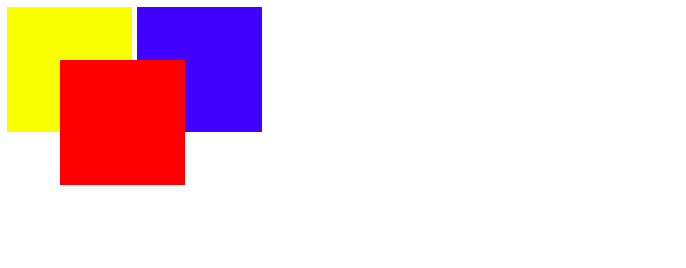
|  |  |
| --- | --- |
| **Position: relative;** | Элемент с этим значением позиционируется относительно своего нормального положения при помощи свойств top, right, bottom и left. |



|  |  |
| --- | --- |
| **Position: fixed;** | Элемент позиционируется относительно *viewport*, то есть его положение всегда остаётся неизменным при использовании прокрутки. |

Такое значение часто используется, чтобы зафиксировать верхнюю часть в начале страницы и оставить её доступной постоянно. Также в последнее время часто используют для всплывающего чата в нижней части экрана.

Элемент с position: fixed; не оставляет никакого зарезервированного пространства в том месте, где он определён в документе изначально.



|  |  |
| --- | --- |
| **Position: absolute;** | Элемент позиционируется относительно ближайшего предка (вместо позиционированного относительно окна просмотра, как fixed). Если элемент с абсолютным позиционированием не имеет позиционированных предков, он использует тело документа и перемещается вместе с прокруткой страницы. |

Родителю, относительно которого хочется спозиционировать элемент, часто добавляется свойство position: relative; без указания дополнительных свойств top, right, bottom и left. Таким образом, родительский элемент никак визуально не меняется, а необходимый элемент смещается относительно него.

|  |  |
| --- | --- |
| **Position: sticky;** | Элемент позиционируется на основе позиции прокрутки пользователя. Можно перевести как «липкое» позиционирование. |

Position: sticky; переключает элемент между *relative* и *fixed*-позиционированием. *Relative*используется, пока положение по умолчанию остаётся во*viewport.*Когда скролл проходит дальше, включается *fixed*-позиционирование.

Обратите внимание, что это свойство position: sticky; **НЕ** поддерживается ни в одной версии браузера *Internet Explorer*.

**Свойство z-index** определяет порядок (какой элемент должен быть расположен сверху или снизу других). Это свойство может иметь как положительные, так и отрицательные значения. Элемент с большим значением перекрывает элемент с меньшим значением.

Если два позиционированных элемента перекрываются без указания *z*-индекса, элемент, размещённый последним в коде *HTML*, будет показан сверху.

**Что же регулирует, с точки зрения браузера, z-index?**

У каждого элемента в *CSS* есть определённое место на экране, которое характеризуется координатами *X* и *Y*, но если бы существовали только эти две координаты, то мы бы не смогли накладывать один элемент поверх другого. Поэтому существует ещё одна координата — пространственная координата *Z*. Поэтому z-index так и называется: по сути, это номер слоя, координата по оси *Z*.

Float-left: и др.;

Opacity: ;

Position: relative;

left: ;

bottom: ;

transform: translate();- изменение позиционирования блока, не влияет на поток, используют(редко) для анимацией; не используют в 99­ процентах.

position: fixed; absolute используют, чтоб не влиять на поток, а только на один или сколько нужно блоков - квадрат двигается, когда уменьшаю и увеличиваю границы стр. , при этом добавить Position: relative; в блок родитель.

Box-sizing: border-box;

Border-radius – регулирует форму, из квадрата в круг и наоборот;

Основной секрет margin - Когда блок расположены вертикально их можно регулировать по отступу друг от друга стилями margin, горизонтально так сделать нельзя. Добавив в название блока родителя Display:flex, блоки, которые содержаться в нем, можно расположить горизонтально.

Margin отрицательный( используется редко);, когда не хотят выбивать блок из потока, но с ним нужно, что-то сделать. Эти операции проводят в нижнем левом окне таблицы. Окошко Styles.

Поток это вот -

<div class="exampe"> - блок родитель

<div class="example\_block"></div>

<div class="example\_block"></div>

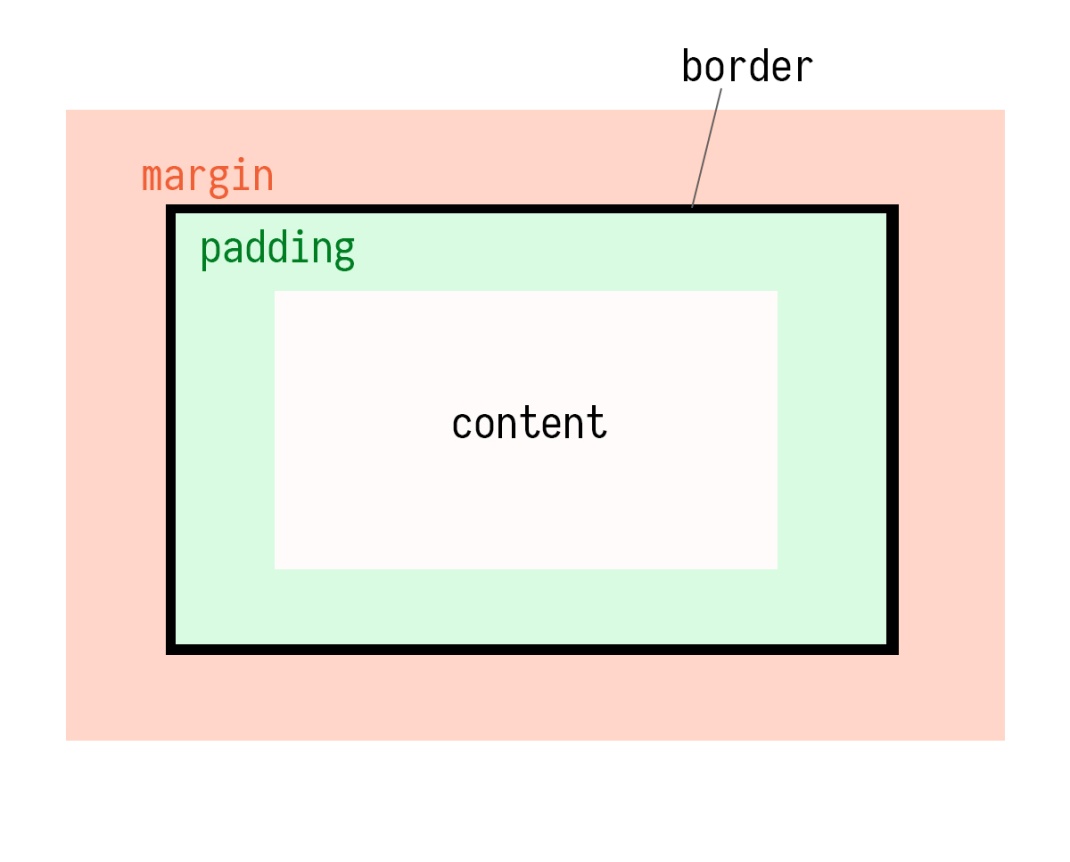
<div class="example\_block"></div>

</div> , нахождение в блоке родителе нескольких блоков. Доминирует при отрицательном раскладе нижний блок, если ему не прописать значения в окошке Styles.

Для выхода блока из потока есть несколько способов.

Прописать Float-left , opacity в параметрах блока, в табл. Styles. Блок остается, но его не видно.

Основой блока выступает его содержимое (текст, картинка, видео), ширина которого задаётся свойством width, а высота — height. Вокруг содержимого идут поля (**padding**), они создают пустое пространство от содержимого до внутреннего края границ; затем идут границы (**border**), а вокруг границ отступы (**margin**). Порядок этих свойств в блоке чётко определён и не может быть нарушен. Но при этом определение ни одного из этих свойств не является обязательным.



Свойство [padding](https://developer.mozilla.org/ru/docs/Web/CSS/padding" \t "_blank) устанавливает пустое поле между содержимым элемента и его границей. Использование этого свойства повышает читаемость текста, улучшает внешний вид страницы.

padding: верх право низ лево;

padding: 4px 8px 15px 2px;

**Три** значения свойства возможны в том случае, если размеры полей справа и слева совпадают.

padding: верх право-лево низ;

padding: 4px 4px 15px;

**Два** значения задают поля сначала по вертикали, потом по горизонтали.

padding: верх-низ право-лево;

padding: 4px 2px;

Наконец, **одно** значение задаст поля со всех сторон.

Кроме того, для задания полей с одной или двух сторон можно использовать развернутую запись свойства padding:

**padding-top** — изменит поле сверху;

**padding-right** — справа;

**padding-bottom** — снизу; **padding-left** — слева.

Свойство [border](https://developer.mozilla.org/ru/docs/Web/CSS/border" \l "%3Cbr-width%3E" \t "_blank) задаёт вокруг области контента границу определённой толщины, стиля и цвета. Эти значения необходимо последовательно указать через пробел, обязательно нужно задать только стиль.

div {

**background-color**: yellow;

**width**: **100px**;

**height**: **100px**;

**border**: **3px** **dashed** black;

}

Можно использовать отдельные свойства [border-width](https://developer.mozilla.org/ru/docs/Web/CSS/border-width" \t "_blank), [border-style](https://developer.mozilla.org/ru/docs/Web/CSS/border-style" \t "_blank), [border-color](https://developer.mozilla.org/ru/docs/Web/CSS/border-color" \t "_blank).

HTML

CSS

Результат

div {

**background-color**: yellow;

**width**: **100px**;

**height**: **100px**;

**border-width**: **3px**;

**border-style**: **double**;

**border-color**: black;

**Толщину границы**(*border-width*) можно задать в пикселях или любых других единицах измерения. Но существует также возможность использовать словесное описание. **Thin** — тонкая граница, **medium** — средней толщины, **thick** — толстая. Перевода описания в конкретные единицы измерения не существует. А выглядит каждая из них так:

HTML

CSS

Результат

**.one** {

**background-color**: white;

**width**: **100px**;

**height**: **100px**;

**border-width**: **thin**;

**border-style**: **solid**;

**border-color**: black;

}

**.two** {

**background-color**: yellow;

**width**: **100px**;

**height**: **100px**;

**border-width**: **medium**;

**border-style**: **solid**;

**border-color**: black;

}

**.three** {

**background-color**: red;

**width**: **100px**;

**height**: **100px**;

**border-width**: **thick**;

**border-style**: **solid**;

**border-color**: black;

}

div {

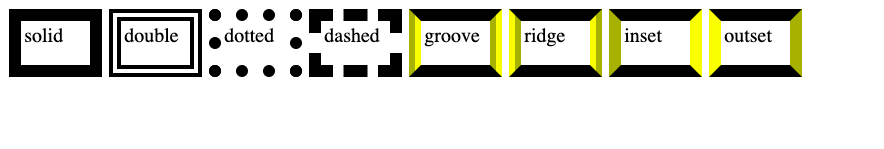
**display**: **inline**-**block**;

**margin**: **0px**;

**padding**: **30px**;

}

**Стиль границы**(*border-style*) может принимать восемь значений. У двух самых востребованных — solid и double — особенностей почти нет. Вид точек и тире в браузерах отличается, поэтому значения **dotted** и **dashed** нужно использовать с осторожностью. Для **groove, ridge, inset** и **outset** лучше задавать два цвета, так как эти значения связаны с переходом цветов.



**.one** {

**border-width**: **10px**;

**border-style**: **solid**;

**border-color**: black;

}

**.two** {

**background-color**: white;

**border-width**: **10px**;

**border-style**: **double**;

**border-color**: black;

}

**.three** {

**border-width**: **10px**;

**border-style**: **dotted**;

**border-color**: black;

}

**.four** {

**border-width**: **10px**;

**border-style**: **dashed**;

**border-color**: black;

}

**.five** {

**border-width**: **10px**;

**border-style**: **groove**;

**border-color**: black yellow;

}

**.six** {

**border-width**: **10px**;

**border-style**: **ridge**;

**border-color**: black yellow;

}

**.seven** {

**border-width**: **10px**;

**border-style**: **inset**;

**border-color**: black yellow;

}

**.eight** {

**border-width**: **10px**;

**border-style**: **outset**;

**border-color**: black yellow;

}

div {

**background-color**: white;

**width**: **50px**;

**height**: **30px**;

**display**: **inline**-**block**;

**margin**: **1px**;

**padding**: **2px**;

**vertical-align**: **top**;

}

Кроме того, как и в случае с padding, есть возможность задать границу только для одной стороны [border-top](https://developer.mozilla.org/en-US/docs/Web/CSS/border-top" \t "_blank), [border-right](https://developer.mozilla.org/en-US/docs/Web/CSS/border-right" \t "_blank), [border-bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/border-bottom" \t "_blank), [border-left](https://developer.mozilla.org/en-US/docs/Web/CSS/border-left" \t "_blank).

К отдельным свойствам границы, которые невозможно задать в сокращённом варианте, относится [border-radius](https://developer.mozilla.org/en-US/docs/Web/CSS/border-radius" \t "_blank). Оно позволяет закруглить края элемента. Это свойство часто используется для рисования в *CSS*. Вообще, к достоинствам большинства свойств, связанных с border,можно отнести возможность их анимации.

**Круг**

HTML

CSS

Результат

**.d1** {

**width**: **150px**;

**height**: **150px**;

**background**: red;

**border**-radius: **50**%;

}

Из круга делают овал и наоборот.

**Свойство margin**

Свойство [margin](https://developer.mozilla.org/en-US/docs/Web/CSS/margin" \t "_blank) устанавливает пустой отступ от внешнего края границы, или внутреннего отступа, или содержимого блока. У margin нет своего фона, поэтому это свойство принимает фон родительского элемента.

Благодаря margin можно задавать отступы от соседних блоков и элементов. По аналогии с padding значения перечисляют, **начиная с верха**, продолжают правым полем, затем — нижним, завершают — левым. То есть **по часовой стрелке**.

margin: верх право низ лево;

margin: 14px 18px 5px 12px;

**Три** значения свойства возможны в том случае, если размеры отступов справа и слева совпадают.

margin: верх право-лево низ;

margin: 4px 4px 15px;

**Два** значения задают отступам сначала по вертикали, потом по горизонтали.

margin: верх-низ право-лево;

margin: 4px 2px;

**Одно** значение margin задаст одинаковые отступы со всех сторон. Точно так же, как и у padding, возможно применение частных свойств: **margin-top**, **margin-right**, **margin-bottom**, **margin-left.**Единицами измерения margin могут быть пиксели, *em*, *rem* и так далее. Проценты берутся от ширины всего блока.

Но в отличие от padding:

* + Значение margin может быть отрицательным. То есть, допустим, margin-top: -15px; означает, что блок сдвинется вверх на 15 пикселей, margin-left: 40px; приведёт к перемещению блока вправо на 40 пикселей.
  + margin может принимать значение auto. В сочетании с шириной элемента, заданной при помощи свойства width, такое значение позволяет выровнять содержимое элемента по горизонтали.

Для некоторых тегов браузер устанавливает величину margin автоматически. Например, для списков (<ol>, <ul>, <dl>) отступы сверху и снизу по умолчанию заданы, как 1*em*. Макетам такие значения по умолчанию, как правило, не очень полезны. Поэтому хорошей практикой считается сбрасывать их перед началом работы, устанавливая margin: 0;.

Чтобы включить флексбокс, нужно задать элементу display: flex. Тогда элемент станет флекс-контейнером, а все вложенные в него элементы станут флекс-элементами.

Относитесь к флекс-контейнеру, как к прямоугольному формату, внутри которого можно расположить несколько элементов по строкам или колонкам, выровняв их по одной из сторон или распределив по формату.

ЦСС-свойства флекс-контейнера управляют направлением (flex-direction), расположением (justify-content) и переносом (flex-wrap) вложенных элементов.

**flex-direction:row;row-reverse;column;column-reverse**

**justify-content:**flex-start;flex-end;center;space-betweens;pace-around

**align-items:**flex-start;flex-end;center;baseline;stretch

Свойство **align-self** выравнивает **конкретный флекс-элемент** по дополнительной оси. Его нужно задавать не всему контейнеру, а нужному флекс-элементу.

**HTML:**

<div class="wrapper">

<div class="block">1</div>

<div class="block special">2</div>

<div class="block">3</div>

</div>

CSS:

.special {

align-self: flex-start;

}

Вы можете не только менять **направление** всех элементов, но и **порядок** любого из них с помощью свойства **order**. Свойство order задаёт элементам номер, по которому они будут выстраиваться. Если у элементов одинаковый номер, они встают друг за другом в изначальном порядке. По умолчанию у всех флекс-элементов order равен нулю (0).



<div class="wrapper">

<div class="block special-1">1</div>

<div class="block special-2">2</div>

<div class="block special-3">3</div>

<div class="block special-4">4</div>

<div class="block special-5">5</div>

<div class="block special-6">6</div>

</div>

.special-1 {

order: 1;

}

.special-2 {

order: 2;

}

.special-3 {

order: -1;

}