

HTML, CSS

# ОСНОВЫ ПОЗИЦИОНИРОВАНИЯ, РАБОТА С Flexbox



# На этом уроке

1. Узнаем, как можно скрыть элемент в HTML с помощью CSS.
2. Узнаем, как можно из строчного элемента сделать блочный, и наоборот.
3. Научимся расставлять элементы благодаря популярной технологии Flexbox.
4. Рассмотрим несколько способов расставления элементов.

## Оглавление

### [Свойство display](#)

[Скрыть блок](#)

[Сделать блочным](#)

[Сделать строчным](#)

[Строчно-блочный элемент](#)

[Табличный элемент](#)

[Flexbox](#)

[Расположение элементов по горизонтали — justify-content](#)

[Расположение элементов по вертикали — align-items](#)

[Расположение элементов по вертикали — align-content](#)

[Вертикальное позиционирование flex-элементов — flex-direction](#)

[Управление многострочностью. перенос элементов на новую строку](#)

[Порядок отображения flex-элементов в блоке родителя — order](#)

[Выравнивание по вертикальной оси дочерних элементов — align-self](#)

[Гибкие размеры блоков](#)

[Как быстрее всего изучить Flexbox](#)

[Позиционирование блоков](#)

[Абсолютное позиционирование](#)

[Относительное позиционирование](#)

[Комбинирование relative и absolute](#)

[Фиксированное позиционирование](#)

[Z-index](#)

[Практическое задание](#)

[Дополнительные материалы](#)

[Используемые источники](#)

# Свойство display

При помощи CSS можно изменить тип элемента, то есть блочный тег сделать строчным, а строчный — блочным. Для этого существует CSS-свойство `display`. Рассмотрим пример и для элементов `<div>` зададим значение свойства `display: inline;`, а для `<span>` — значение `display: block;`.

```
div, span {
  border: 1px solid #000;
  width: 200px;
  height: 100px;
}

div {
  display: inline;
}

span {
  display: block;
}
```

В этом случае получается, что элементы поменялись местами, `<div>` стал строчным элементом, и ему теперь невозможно задать ни ширину, ни высоту, а `<span>` стал блочным, и ему теперь можно задать ширину и высоту.

Значения свойства `display`:

- `none` — скрыть;
- `block` — блочный;
- `inline` — строчный;
- `inline-block` — строчно-блочный;
- `table-cell` — ячейка таблицы;
- `flex` — гибкий.

## Скрыть блок

**Скрыть блок** (`display: none;`) помогает спрятать элемент, чтобы в дальнейшем его можно было показать при наведении или клике на соседний блок. Хороший пример — выпадающее меню. Изначально его не видно, но при наведении на элементы меню мы можем увидеть выпадающий блок.

### [Пример](#)

## Сделать блочным

**Блочный элемент** (`display: block;`) создаёт разрыв строки перед тегом и после него. Он образует прямоугольную область, занимающую всю ширину веб-страницы или блока-родителя, если для него не задано значение `width`. Высота этого блока определяется его содержимым: в блоке может быть очень много контента, или он может отсутствовать вовсе — такой блок будет 0 высоты. Блочные элементы могут содержать внутри себя элементы любого типа. Нельзя размещать блочные элементы внутри строчных, за исключением элемента `<img>`. Для блочных элементов можно задавать отступы.

### Резюме

1. Блочные элементы отображаются на веб-странице в виде прямоугольника.
2. Занимают всю доступную ширину.
3. Высота определяется содержимым.
4. Начинаются с новой строки.
5. Допускается вкладывать один блочный элемент внутрь другого.
6. Запрещено добавлять внутрь строчных элементов блочные.

**Примеры блочных элементов:**

- `<div>;`
- `<form>;`
- `<h1>;`
- `<p>;`
- `<table>;`
- `<ul>;`
- ...

## Сделать строчным

**Строчные элементы** (`display: inline;`) не создают блоки, они отображаются на одной строке с содержимым рядом стоящих тегов. Это потомки блочных элементов. Строчные элементы игнорируют верхние и нижние отступы, но, если для элемента задан фон, он будет распространяться на верхний и нижний внутренний отступ, заходя на соседние строки текста. Простыми словами, эти теги придуманы для стилизации части текста. Если на странице выделен произвольным цветом какой-то текст, эту особенность можно создать с помощью `inline`-тегов.

Ширина и высота строчного элемента зависят только от его содержания, задать размеры с помощью CSS нельзя. Можно увеличить расстояние между соседними элементами по горизонтали с помощью горизонтальных полей и отступов.

## Резюме

1. Строчные элементы используются для изменения вида текста и логического выделения.
2. Являются частью строки.
3. Ширина равна содержимому плюс значения отступов.
4. Внутри строчных элементов допустимо помещать текст или другие строчные элементы.
5. Свойства, связанные с размерами, неприменимы.
6. Элементы, идущие подряд, не переносятся на другую строку, располагаются на одной строке.

### Примеры строчных элементов:

- `<span>`;
- `<a>`;
- `<i>`;
- `<strong>`;
- `<u>`;
- ...

## Строчно-блочный элемент

Есть ещё одна группа элементов, которые браузер обрабатывает как **строчно-блочные** (`display: inline-block`;). Это встроенные элементы, но для них можно задавать поля, отступы, ширину и высоту.

Ещё в начале 2000-х годов `inline-block` использовался для расставления элементов по горизонтали. Сейчас этот способ устарел, поэтому, если вам необходимо расставить элементы горизонтально, применяйте более современный способ `flexbox`, о котором вы узнаете чуть позже. Это свойство используется только в ситуации, когда строчному элементу необходимо задать параметры ширины и высоты. Хороший пример — ссылка, которая является строчной, но достаточно часто используется как кнопка на сайте.

Особенности `inline-block`:

1. Можно задавать свойства `width`, `height`.
2. Является частью строки.
3. Размер устанавливается по содержимому, если не заданы значения ширины и высоты.
4. Элемент всегда прямоугольный.

## Табличный элемент

**Табличный элемент** `table-cell` (ячейка таблицы). Это свойство устарело и в современной вёрстке не используется, так как таблицы не адаптивны, следовательно, их негде применять.

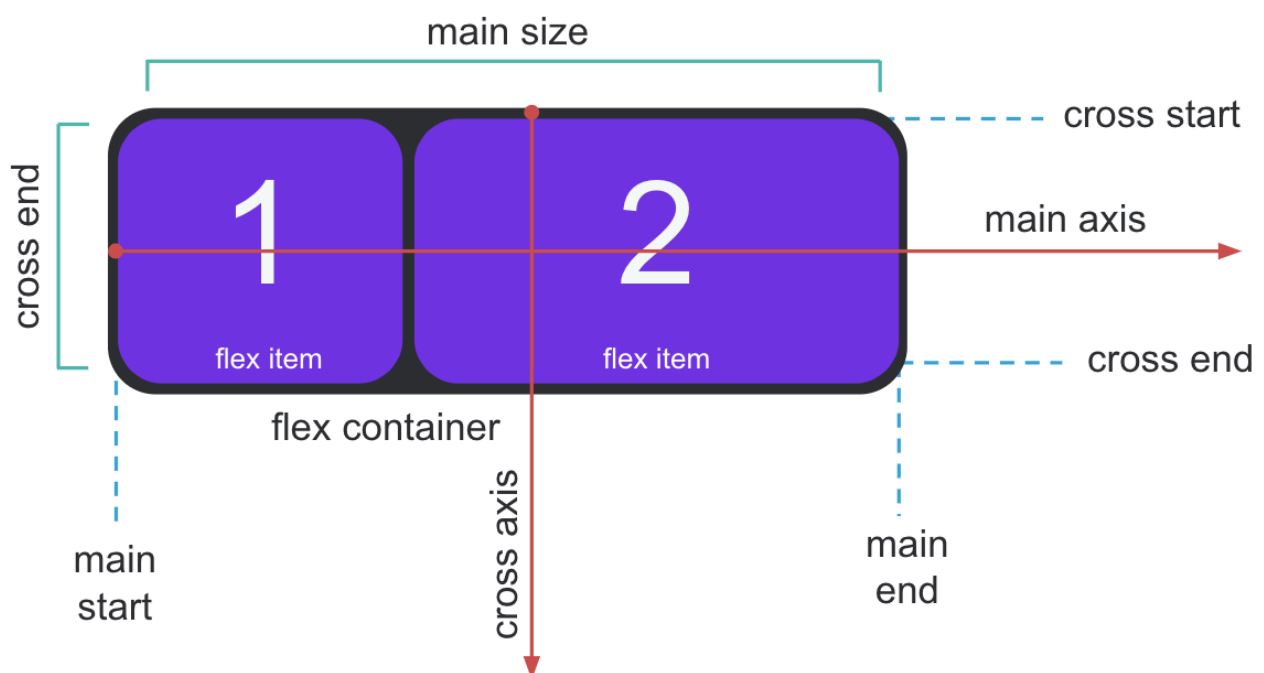
## Flexbox

**Гибкий элемент** `display: flex` — наделение контейнера способностью изменять ширину или высоту для поддержки всех видов дисплеев и разных разрешений экранов. Flexbox — это целый модуль, а не просто единичное свойство, он объединяет в себе множество свойств. Некоторые из них должны применяться к контейнеру — родительскому элементу, так называемому flex-контейнеру. Другие применяются к дочерним элементам или flex-элементам.

Особенности flexbox:

1. Элементы могут сжиматься и растягиваться, занимая необходимое пространство.
2. Элементы могут автоматически выстраиваться в несколько строк.
3. Возможно выравнивание не только по вертикали, но и по горизонтали.
4. Есть возможность формирования блоков справа налево.

Если обычный layout основывается на направлениях потоков блочных и inline-элементов, то flex-layout основывается на направлениях flex-потока. Ознакомьтесь со схемой из спецификации, разъясняющей основную идею flex-layout'ов.



В основном элементы будут распределяться либо вдоль главной оси (от `main-start` до `main-end`), либо вдоль поперечной оси (от `cross-start` до `cross-end`).

1. `main axis` — главная ось, вдоль которой располагаются flex-элементы. Обратите внимание: она не обязательно должна быть горизонтальной, всё зависит от свойства `justify-content` (см. ниже). Основная задача flexbox — расставить элементы горизонтально, в итоге мы выставляем элементы по `main axis`.
2. `main-start` | `main-end` — flex-элементы размещаются в контейнере от позиции левой части блока (`main-start`) до позиции правой части блока (`main-end`).
3. `main size` — ширина или высота flex-элемента в зависимости от выбранной основной величины. Простыми словами, это ширина или высота блока контейнера.
4. `cross axis` — поперечная ось, перпендикулярная главной. Её направление зависит от направления главной оси. Проще всего запомнить, что это ось Y. Подобные обозначения добавлялись в геометрии.
5. `cross-start` | `cross-end` — flex-строки, заполняются элементами и размещаются в контейнере от позиции верхней части блока родителя (`cross-start`) и до позиции нижней части блока родителя (`cross-end`).
6. `cross size` — ширина или высота flex-элемента в зависимости от выбранной размерности. Это свойство совпадает с `width` или `height` элемента. Чаще всего это значение высоты контейнера, зависит от заданных параметров `height` или высоты контентной части.

Flex-контейнер не блочный элемент, поэтому для дочерних элементов не работают такие CSS-свойства, как `float`, `clear`, `vertical-align`.

Flex-контейнер устанавливает форматирование для содержимого, которое обуславливает следующие особенности:

1. Для flex-элементов блокируется их значение свойства `display`. Значения `display: inline-block;` и `display: table-cell;` вычисляются в `display: block;`.
2. Пустое пространство между элементами исчезает: оно будет наследовать их, например параметры шрифта, от flex-контейнера.
3. Абсолютно позиционированный flex-элемент может находиться в блоке flex родителя, но не участвует в расставлении элементов. Простыми словами, в flex-блоке может размещаться абсолютный блок, он будет размещаться поверх остальных блоков, которые вы расставили горизонтально.
4. Внешние отступы не схлопываются.
5. Процентные значения отступов вычисляются от внутреннего размера содержащего их блока.
6. `margin: auto;` — это свойство не работало у блочных элементов, а во flex-элементах оно помогает центрировать элементы не только горизонтально, но и вертикально.

7. Автоматический минимальный размер flex-элементов. Если вы не задали параметры ширины для дочерних блоков, по умолчанию значение их будет равно 0 и дальше зависит только от контентной части.

## Расположение элементов по горизонтали — justify-content

Элементы в контейнере поддаются выравниванию вдоль главной оси при помощи свойства `justify-content`. Это свойство принимает целых пять вариантов значений:

1. `flex-start` (`default`): гибкие элементы выравниваются по началу главной оси.
2. `flex-end`: элементы выравниваются по концу главной оси.
3. `center`: элементы выравниваются по центру главной оси.
4. `space-between`: элементы занимают всю доступную ширину в контейнере, крайние элементы вплотную прижимаются к краям контейнера, а свободное пространство равномерно распределяется между элементами. Проще всего данную часть запомнить как расстояние между блоками, оно сохраняется одинаковым, а элементы распределяются горизонтально, чтобы сохранить эти параметры.
5. `space-around`: гибкие элементы выравниваются таким образом, что свободное пространство равномерно распределяется между элементами. Но стоит отметить, что пространство между краем контейнера и крайними элементами будет в два раза меньше, чем между элементами в середине ряда. Простыми словами, отступы слева и справа одного блока равняются отступам слева и справа любого другого блока.

## Расположение элементов по вертикали — align-items

Мы также имеем возможность выравнивания элементов по вертикальной оси (Y). Применяв свойство `align-items`, которое принимает пять разных значений, можно расставить элементы где угодно по вертикали. Это свойство позволяет выравнивать элементы в строке относительно друг друга.

1. `flex-start`: все элементы прижимаются к верхней части родительского элемента.
2. `flex-end`: элементы прижимаются к нижней части родительского элемента.
3. `center`: центрирование элементов по вертикали.
4. `baseline`: элементы выравниваются по базовой линии текста. Простыми словами, если у вас есть блок в 3 строки, а во втором всего одна строка, он будет их располагать идеально по центру, второй блок будет вровень со второй строчкой из первого блока.
5. `stretch` (`default`): элементы растягиваются, полностью заполняя высоту блока родителя.



## Расположение элементов по вертикали — align-content

Ещё одно свойство, похожее на предыдущее, — `align-content`. Оно отвечает за выравнивание целых строк относительно гибкого контейнера. Свойство не будет давать эффекта, если гибкие элементы занимают одну строку.

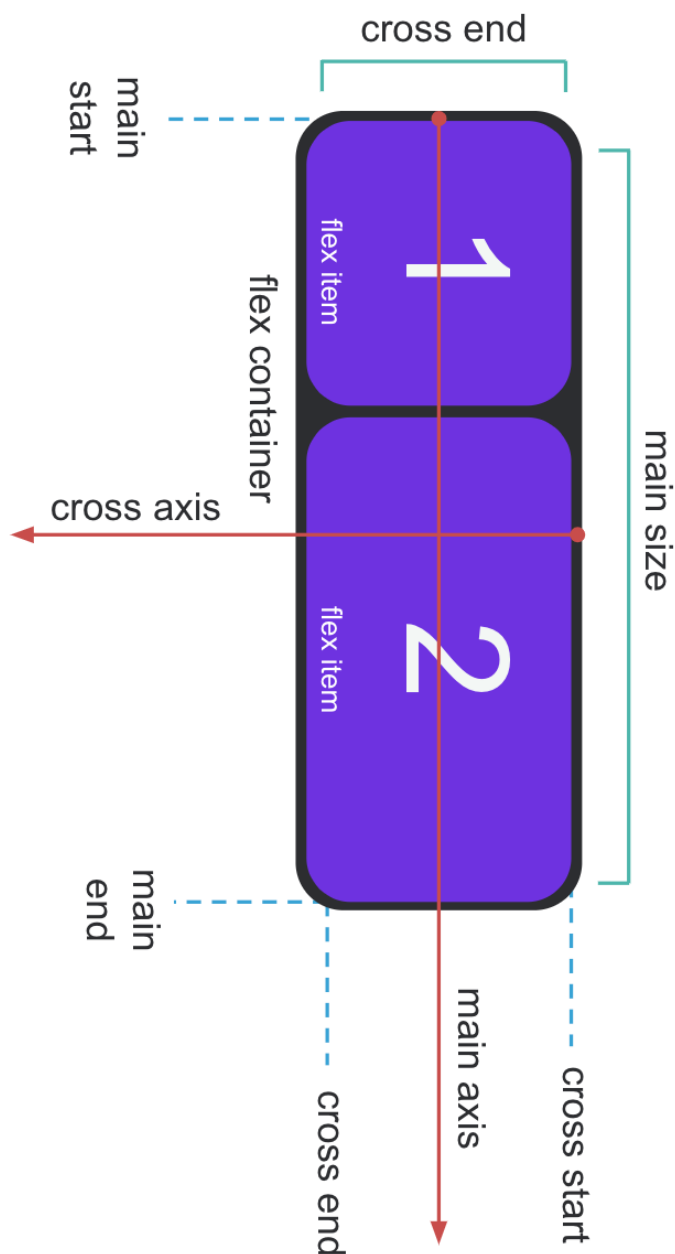
`align-content` принимает шесть разных значений:

1. `flex-start`: все линии прижимаются к началу вертикальной оси.
2. `flex-end`: все линии прижимаются к концу вертикальной оси.
3. `center`: flex-элементы центрируются относительно flex-контейнера.
4. `space-between`: линии распределяются от верхнего края до нижнего, оставляя свободное пространство между строками, крайние же строки прижимаются к краям контейнера.
5. `space-around`: равномерное распределение блоков по контейнеру.
6. `stretch (default)`: линии растягиваются, занимая всё доступное пространство блока родителя.

## Вертикальное позиционирование flex-элементов — flex-direction

Не всегда нужно, чтобы блоки располагались горизонтально, поэтому мы можем повернуть главную ось (X) и сделать её вертикальной.

**Внимание!** При расставлении блоков по вертикали всегда есть мысль, что элементы выстраиваются вертикально, но оси остаются. На самом деле main- и cross-ось меняются местами. Детское сравнение: мы крутим блок родителя по часовой стрелке на 90 градусов.



1. `row`: значение по умолчанию, слева направо (в rtl справа налево). Flex-элементы выкладываются в строку. Начало (`main-start`) и конец (`main-end`) направления главной оси соответствуют началу (`inline-start`) и концу (`inline-end`) оси строки (`inline-axis`).
2. `row-reverse`: направление справа налево (в rtl слева направо). Flex-элементы выкладываются в строку относительно правого края контейнера (в rtl — левого).
3. `column`: направление сверху вниз. Flex-элементы выкладываются в колонку. На изображении выше как раз представлен этот вариант.
4. `column-reverse`: колонка с элементами в обратном порядке, снизу вверх.

## Управление многострочностью, перенос элементов на новую строку

Свойство определяет, будет ли flex-контейнер однострочным или многострочным, а также задаёт направление поперечной оси, определяющее направление укладки новых линий flex-контейнера.

Если вам нужно, чтобы элементы перепрыгивали на новую строку, так как они не помещаются горизонтально, вам необходимо задать значение `flex-wrap: wrap;`.

По умолчанию flex-элементы укладываются в одну строку вдоль главной оси. Если дочерние элементы не помещаются в блоке родителя по горизонтали, они начнут перестраиваться на новую строку.

**Внимание!** Если ваш сайт будет адаптивным (а мы больше чем уверены, что он у вас адаптивен) вам необходимо задать `flex-wrap: wrap;` сразу же после добавления `display: flex;`. Только в ситуации, если вы не хотите, чтобы блоки подстраивались друг под друга, необходимо использовать `flex-wrap: nowrap;`.

Также существует сокращённая форма записи `flex-direction` и `flex-wrap`. Это свойство называется `flex-flow`, тут вы можете написать `flex-flow: column nowrap`. На данном этапе мы не рекомендуем использовать сокращённую форму, так как для запоминания лучше привыкнуть к отдельным значениям, а в дальнейшем уже можно будет применять сокращения.

## Порядок отображения flex-элементов в блоке родителя — order

Изначально все flex-элементы имеют номер блока, и он равен 0 (`order: 0`). Если вы хотите расставить блоки в определённой последовательности и не изменять HTML-структуру, используйте свойство `order`. Вы можете задать любому блоку произвольный номер: например, у вас есть 3 блока (1, 2, 3), вы последнему задали `order: 2`, первому задали `order: 3`; а второму — `order: 1`; в итоге ваши блоки расставляются как 2, 3, 1. Детское сравнение: на физкультуре 10 учеников, вы им выдали номера от 1 до 10. Неважно, как ученики стояли изначально, после раздачи цифр они должны выстроиться по номерам. Так же и свойство `order` позволяет переопределить последовательность расставленных элементов.

## Выравнивание по вертикальной оси дочерних элементов — align-self

Это свойство отвечает за выравнивание отдельно взятого flex-элемента по высоте контейнера. Переопределяет выравнивание, заданное `align-items`.

`align-self` повторяет значения `align-item`:

1. `flex-start`: все элементы прижимаются к верхней части родительского элемента.
2. `flex-end`: элементы прижимаются к нижней части родительского элемента.
3. `center`: центрирование элементов по вертикали.
4. `baseline`: элементы выравниваются по базовой линии текста. Простыми словами, если у вас есть блок в 3 строки, а во втором всего одна строка, он будет их располагать идеально по центру, второй блок будет вровень со второй строчкой из первого блока.
5. `stretch (default)`: элементы растягиваются, полностью заполняя высоту блока родителя.

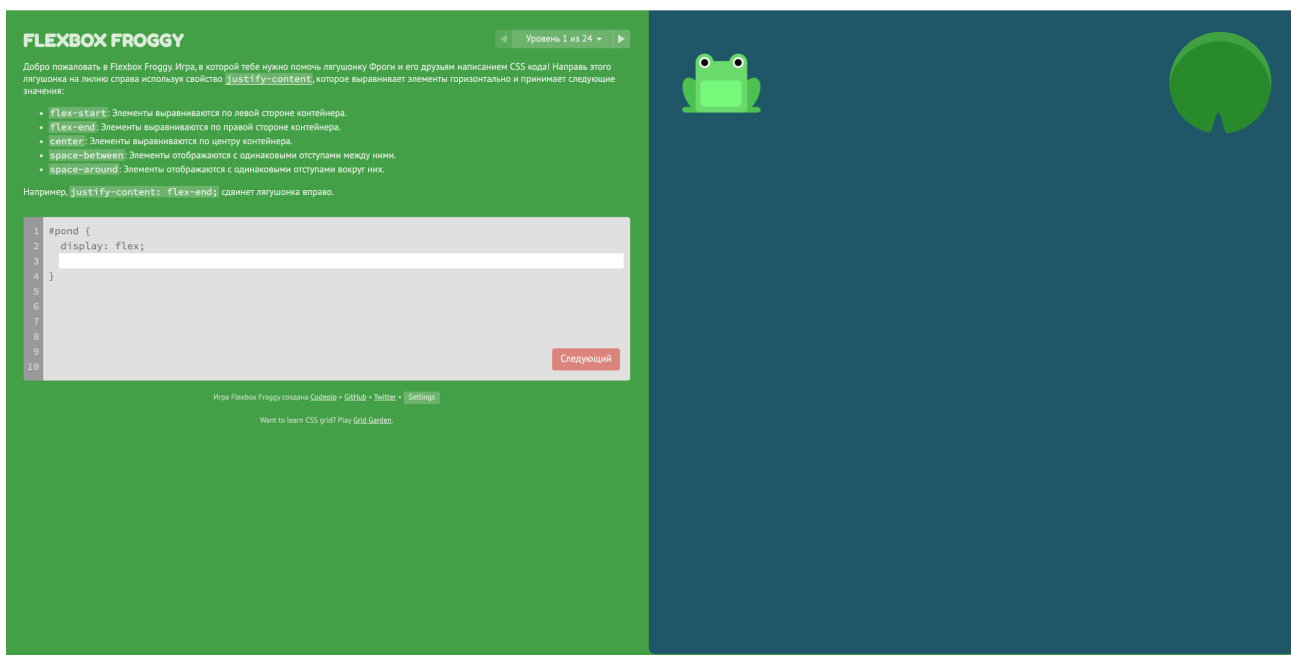
**Внимание!** Если вы хотите добавить позиционирование отдельного элемента из flex-контейнера, вам необходимо задать класс к данному элементу. Не пугайтесь, если классов будет несколько. Например: `<div class="item item_center">`, и далее задавать стили для `item_center`.

## Гибкие размеры блоков

Эту тему мы будем рассматривать на седьмом уроке, а пока достаточно материала, представленного в видео. Не нужно стремиться выполнить сразу два действия: сделать десктопную и мобильную версии продукта. Вам необходимо создать десктопную версию проекта и уже после этого добавить гибкость вашему сайту, которая в дальнейшем и приведёт к адаптивности вашей вёрстки.

## Как быстрее всего изучить Flexbox

При выборе вариантов позиционирования блоков начинающие веб-разработчики находят слишком много разной информации и тратят слишком много времени на изучение всех нюансов позиционирования, но на начальном этапе необходимо максимально быстро освоить позиционирование элементов. Для этого используйте онлайн-тренажеры, которые помогут быстро запомнить достаточно сложные технологии, плюс в них вы сразу сможете потренировать навыки на практике. Онлайн-тренажеров много, один из самых популярных — [Flexbox Froggy](#). На начальном этапе рекомендуем пройти эту игру несколько раз, чтобы лучше отточить навыки позиционирования.



## Позиционирование блоков

Идея, лежащая в основе позиционирования, довольно проста. Оно позволяет точно определить, где появятся блоки относительно другого элемента или окна браузера. Как только вы добавляете блочные элементы на странице, они располагаются друг под другом, так как это особенность блочных элементов — занимать 100% ширины и начинаться с новой строки. Изначально им присвоено значение `position: static;`

Свойство `position` вместе со значениями `top`, `right`, `bottom` и `left` отображает элемент с нарушением обычного порядка, смещая его на заданное расстояние. При позиционировании элементов можно использовать как положительные, так и отрицательные значения. В итоге получается несколько видов позиционирования.

## Абсолютное позиционирование

При абсолютном позиционировании блок полностью исчезает из потока выставленных HTML элементов и его положение задаётся относительно левого верхнего угла браузера. Задать этот тип можно через значение `absolute` свойства `position`. Координаты указываются относительно краёв окна браузера, называемого видимой областью.

**Внимание!** Если вы задали блоку свойство `position: absolute`, он будет отсчитываться от левого верхнего угла браузера и, следовательно, у всех пользователей будет разный внешний вид сайта. Простыми словами, использовать данное свойство без `relative` бессмысленно.

Несколько особенностей абсолютного позиционирования:

1. Ширина и высота блока зависят от контента, поэтому, если вы хотите, чтобы элемент занимал всё доступное пространство, необходимо выставить `width: 100%;`.
2. У блока нет определённого позиционирования, если у него нет свойств `right`, `left`, `top` и `bottom`.
3. Свойства `left` и `top` имеют более высокий приоритет по сравнению с `right` и `bottom`. Если `left` и `right` противоречат друг другу, значение `right` игнорируется. То же самое касается и `bottom`.
4. Если задать отрицательное значение `left`, то можно спрятать блок как бы за экран браузера. Иногда это используется для выпадающих блоков.
5. Если `left` задать значение больше ширины видимой области или указать `right` с отрицательным значением, появится горизонтальная полоса прокрутки. Подобное правило работает и с `top`, только речь пойдёт о вертикальной полосе прокрутки.
6. Одновременно указанные свойства `left` и `right` формируют ширину слоя, но только если `width` не указано. Стоит добавить свойство `width`, и значение `right` будет проигнорировано. Аналогично произойдёт и с высотой слоя, только участвуют свойства `top`, `bottom` и `height`.
7. Это свойство рекомендуется использовать только совместно со свойством `position: relative;`, которое задано родительскому элементу.









## Относительное позиционирование

`relative` (относительное позиционирование) — блок, которому задано это свойство, будет занимать ширину и высоту, где располагался ранее. Если мы зададим этому блоку смещение `left`, `top`, `right`, `bottom`, он сместится только визуально, но место, которое он занимал, будет пустым. Небольшое детское сравнение: если человек купил место в театре и куда-то отошёл, оно так и останется пустым. Неважно, где именно в данный момент находится человек.

## Комбинирование `relative` и `absolute`

Если вам нужно поставить один элемент поверх другого, используйте комбинацию `relative` и `absolute`. Для этого задайте родительскому элементу свойство `relative` — этот блок и будет точкой отсчёта для дочернего элемента со значением `absolute`.

Также эта комбинация достаточно часто применяется для позиционирования псевдоэлементов. Пример в иллюстрации:

Эко trade-in	Новинка	10% кэшбэк!	10% кэшбэк!
			
Телевизор Samsung UE32M5550AU ★ 4.7 308 отзывов	Телевизор LG 55UN73506LB ★ 4.9 120 отзывов	Телевизор Samsung QE50Q87TAU ★ 5.0 5 отзывов	Телевизор Samsung UE43TU7097U ★ 4.8 8 отзывов
<b>24 990 Р</b> 22 990 Р Эко скидка + 3 249 БР ?	<b>37 990 Р</b> 41-990 Р от 1 860 Р/мес.	<b>94 990 Р</b> от 4 651 Р/мес.	<b>28 990 Р</b> от 1 424 Р/мес.
 В корзину	 В корзину	 В корзину	 В корзину

Текст «Новинка, 10% кэшбэк» проще всего отпозиционировать с помощью комбинации `position: relative` и `absolute`.

## Фиксированное позиционирование

Фиксированное положение слоя задаётся значением `fixed` свойства `position` и по своему действию похоже на абсолютное позиционирование. В отличие от него, привязывается к указанной свойствами `left`, `top`, `right` и `bottom` точке на экране и не меняет своего положения при прокрутке веб-страницы. Ещё одно отличие от `absolute` — при выходе фиксированного слоя за пределы видимой области справа или снизу от неё не возникает полос прокрутки.

Такой тип позиционирования применяется для создания любых элементов, которые должны быть закреплены на странице и всегда видны посетителю.

## Z-index

Любые позиционированные элементы на веб-странице могут накладываться друг на друга в определённом порядке. Мы с вами рассматривали горизонтальную ось X и вертикальную ось Y. В этом варианте позиционирования появляется ось Z, перпендикулярная экрану. Каждый элемент может находиться как ниже, так и выше других объектов веб-страницы. Если вы работали с графическими программами, вы знаете, что большинство элементов на странице — это слои, которые могут быть ближе к нам или располагаться под другими, это и будет выставление элементов на

перпендикулярной оси. Их размещением по Z-оси и управляет `z-index`. Это свойство работает только для элементов, у которых значение `position` задано как `absolute`, `fixed` или `relative`.

В качестве значения используются целые числа — положительные, отрицательные и ноль. Чем больше значение, тем выше находится элемент по сравнению с теми элементами, у которых оно меньше. При равном значении `z-index` на переднем плане находится тот элемент, который в коде HTML описан ниже.

Кроме числовых значений применяется `auto` — порядок элементов в этом случае строится автоматически, исходя из их положения в коде HTML и принадлежности к родителю, поскольку дочерние элементы имеют тот же номер, что их родительский элемент. Значение `inherit` указывает, что оно наследуется у родителя.

## Практическое задание

На этом уроке мы переходим к позиционированию элементов.

1. В файле стилей первой строчкой добавить обнуление стилей `* {margin: 0; padding: 0;}`.
2. В файле стилей создать блок `container`, отвечающий за центрирование нашей страницы:

```
.container { width: 1140px; margin: 0 auto; }
```

3. В шапке сайта необходимо расположить элементы горизонтально, текст «Главная» должен располагаться слева, «Контакты» — справа.
4. Заголовку `h1` указать ширину `510px`.
5. Тексту под заголовком `h1` указать ширину `425px`.
6. В HTML-файле `class="what-we-do"` добавить класс `container`, чтобы получилось `class="what-we-do container"` для центрирования содержимого блока `what-we-do`:
  - a. Блоку карточки (заголовок `h4`: Make your business, смотри макет ниже и `p`) указать значение ширины `359px`.
  - b. Создать ещё 2 карточки товара, как представлено в [изображении макета](#). Итого получится 3 карточки.
  - c. Расположить их горизонтально и выставить равные отступы между ними.
7. В подвале сайта:
  - a. Добавить центрирование всего контента, в подвале сайта внутри блока `footer` создать блок `container` для центрирования содержимого. Должна получиться структура:

```
<div class="footer">  
<div class="container">Содержимое футера</div>  
</div>
```



- b. Блоку контента (h3 "Intersted to woek") в верхней части футера указать ширину 340px.
  - c. Тексту (p) в подвале сайта указать ширину 495px.
  - d. Список в нижней части подвала (номер телефона и email) расположить горизонтально и выставить равные отступы между ними.
8. Страница «Контакты»:
- a. Добавить центрирование всей контентной части страницы с помощью блока `container`.
  - b. В соответствии с изображением страницы «Контакты» разместить слева блок с контактами, справа — блок формы заполнения данных.
  - c. Карту сайта сделать на всю ширину `container`.

## Дополнительные материалы

- 1. [Pixactly](#).
- 2. [Онлайн-тренажёр Flexbox Froggy](#).
- 3. [Онлайн-тренажер по grid](#).

## Используемые источники

- 1. [htmlbook.ru](http://htmlbook.ru).
- 2. [html5book.ru](http://html5book.ru).
- 3. [WebReference](#).