## Фоны

|  |
| --- |
| [Пробуем управлять потоком](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/1) |
| [Управление потоком, шаг 2](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/2) |
| [Управление потоком, шаг 3](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/3) |
| [Создадим другой поток](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/4) |
| [Другой поток, шаг 2](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/5) |
| [Другой поток, шаг 3](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/6) |
| [Другой поток, финал](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/7) |
| [Погружение в флоаты](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/8) |
| [float и ширина](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/9) |
| [float и выпадание из потока](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/10) |
| [Флоат рядом с флоатом](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/11) |
| [Когда флоатов много, а места мало](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/12) |
| [Испытание: паззл на флоатах](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/13) |
| [Свойство clear](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/14) |
| [Борьба с выпаданием флоатов: распорки](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/15) |
| [Борьба с выпаданием флоатов: псевдораспорки](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/16) |
| [Простейшая сетка, шаг 1](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/17) |
| [Простейшая сетка, шаг 2](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/18) |
| [Простейшая сетка, финал](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/19) |
| [Сетка посложнее, шаг 1](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/20) |
| [Сетка посложнее, шаг 2](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/21) |
| [Сетка посложнее, шаг 3](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/22) |
| [Сетка посложнее, добавляем содержание](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/23) |
| [Последняя сетка, шаг 1](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/24) |
| [Последняя сетка, шаг 2](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/25) |
| [Последняя сетка завершена](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/26) |
| [Испытание: строим сетку](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/27) |
| [Погружение в inline-block](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/28) |
| [float vs inline-block](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/29) |
| [Простая сетка на inline-block](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/30) |
| [inline-block и пробелы в коде](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/31) |
| [Испытание: котогалерея на inline-block](https://npoed.htmlacademy.ru/htmlcss110/course/13/run/32) |

# Пробуем управлять потоком [1/32]

C понятием потока документа вы познакомились в разделе про позиционирование.

Поток — это порядок отображения элементов на странице. По умолчанию блочные элементы отображаются как прямоугольные области, идущие друг за другом сверху вниз, а строчные элементы располагаются сверху вниз и слева направо и при необходимости переносятся на новую строку.

Потоком можно управлять и изменять привычное поведение элементов в потоке. Например, можно заставить блочные элементы двигаться не сверху вниз, а выстраиваться в несколько колонок.

А научившись управлять потоком, вы сможете строить сетки.

В первой серии заданий мы продемонстрируем как один и тот же HTML-код можно выстроить в разные потоки. Например, вот так:



# Управление потоком, шаг 2 [2/32]

На первом шаге мы с помощью свойства float изменили поведение первого блока.

Обратите внимание, как он стал наезжать на последующие блоки. Это одна из особенностей флоатов, которую мы разберём позже.

А пока продолжим: сейчас будем изменять поведение второго блока.

# Управление потоком, шаг 3 [3/32]

Со вторым блоком разобрались. Теперь уже первые два блока наезжают на третий.

Чтобы получить нужный нам поток, осталось задать размеры третьему блоку, отцентровать его и избавиться от эффекта «наезжания».

Поможет нам в этом свойство clear.

# Создадим другой поток [4/32]

У вас получилось создать первый поток. Теперь попробуем те же самые элементы выстроить по-другому. Вот так:



# Другой поток, шаг 2 [5/32]

А пока вы пошагово создаёте второй поток, поговорим о сетках.

Сетка — это взаимное расположение крупных блоков сайта. Вспомните любой сайт, например, ВКонтакте или HTML Academy.

У ВКонтакте простая сетка, которая состоит из верхней части, или «хедера», под которой располагаются две колонки, а в самом низу расположен «футер» или нижняя часть сайта. Вот так схематично можно представить сетку ВКонтакте:

-------------------------------------

хедер

-------------------------------------

левая колонка | правая колонка

меню | основное содержание

-------------------------------------

футер

-------------------------------------

У HTML Academy на разных страницах используются разные сетки. Например, на главной странице есть двухколончатый хедер, под ним содержание страницы в три колонки, затем нижнее меню и двухколончатый футер.

Как вы понимаете, сетки могут быть самыми разными.

# Другой поток, шаг 3 [6/32]

Что нужно знать и уметь, чтобы строить сетки? Две вещи:

1. управлять размерами элементов;
2. управлять поведением элементов в потоке.

Первая часть детально разбирается в разделе «Блочная модель документа». Там вы учитесь:

* задавать размеры;
* рамки;
* внешние и внутренние отступы;
* рассчитывать полный размер элемента;
* рассчитывать занимаемую им площадь.

Управлять поведением в потоке мы будем учиться в этом разделе.

# Другой поток, финал [7/32]

Существует несколько способов управлять потоком и строить сетки:

1. флоаты;
2. инлайн-блоки;
3. табличная вёрстка;
4. флексбоксы.

Табличная вёрстка — самый простой для понимания способ построения сеток. Но он считается устаревшим и использовать его не рекомендуется. В будущем можно будет использовать для такой вёрстки не таблицы, а элементы с display:table, display:table-row и display:table-cell.

Флексбоксы — это новая и очень мощная технология для построения сеток. К сожалению, её поддержка браузерами ещё достаточно слабая.

А флоаты и инлайн-блоки мы разберём в этом разделе более подробно. Основной упор сделаем на флоаты, так как сейчас это самый распространённый способ построения сеток.

А пока что завершите создание своей второй сетки.

# Погружение в флоаты [8/32]

А теперь детально разберём свойство float, его значения, особенности поведения и способы применения.

Т. к. у этого свойства нет устоявшегося русского перевода или термина, то мы будем называть его «флоатом», а также использовать конструкции наподобие «зафлоатить».

Изначально float было предназначено для того, чтобы включать обтекание элементов текстом. Наподобие того, как в более старой версии HTML текст обтекал изображения c атрибутами align="left" или align="right".

То есть, можно сказать, что float — это свойство, включающее режим обтекания. Но, как часто бывает, судьба уготовила ему совсем другую роль.

Свойство float имеет следующие значения:

1. left — прижимает элемент к левому краю родителя, другие элементы обтекают его справа;
2. right — прижимает элемент к правому краю родителя, другие элементы обтекают его слева;
3. none — отключает режим обтекания и возвращает элементу нормальное поведение.

Обратите внимание, что зафлоатить элемент по центру нельзя.

# float и ширина [9/32]

Вы уже знаете, что по умолчанию блочные элементы растягиваются на всю доступную ширину родителя.

Если мы задаём элементу свойство float:left или float:right, то он прижимается к левому или правому краю, а также начинает ужиматься по ширине под своё содержимое. С той стороны, которая не прижата к краю родителя, появляется свободное место. Это место может быть занято другими элементами.

Зафлоаченному элементу можно явно задавать размеры и отступы.

Есть тонкость, связанная со строчными элементами. Если зафлоатить строчный элемент, то он начинает вести себя как блочный, а именно: воспринимать размеры и отступы.

# float и выпадание из потока [10/32]

Мы уже говорили про выпадание элементов из потока, когда разбирали абсолютное и фиксированное позиционирование.

Зафлоаченные элементы тоже выпадают из потока, но лишь частично:

* Блочные элементы, которые идут в коде после зафлоаченного блока, перестают его замечать. Они подтягиваются вверх и занимают его место, как будто его и нет.
* Строчные же элементы, расположенные в коде после зафлоаченного блока, начинают обтекать его со свободной стороны.

Ещё раз: для блочных элементов флоатные не существуют, но текст внутри блоков флоатные обтекает.

Такое поведение флоатов даёт интересные эффекты:

1. Эффект прохождения сквозь блоки. Проявляется, когда зафлоаченный элемент выше, чем несколько последующих за ним блоков.
2. Эффект выпадания из родителя или схлопывания родителя. Проявляется тогда, когда все дочерние блоки в родителе зафлоачены. В этом случае родитель схлопывается по высоте, как будто в нём нет содержимого, а блоки выпадают из него.

# Флоат рядом с флоатом [11/32]

Флоатные элементы становятся невидимыми для блочных элементов и видимыми для текста. А как же они взаимодействуют друг с другом?

Если вкратце то: флоатные элементы видят друг друга.

Идущие друг за другом флоаты выстраиваются в ряд, пока им хватает свободного места. Если места не хватает, то они начинают переноситься на следующую строчку. Почти как текст.

Как раз эта особенность флоатов и позволила применять их для создания сеток. Ведь в начале двухтысячных в CSS не было никакого другого способа создавать колонки и задавать им размеры.

Блочные элементы всегда располагались в разных строках. А строчные элементы, хоть и могли располагаться на одной строке, совершенно не воспринимали размеры. Позиционирование тоже не подходило для сеток, т. к. элементы выпадали из потока.

В общем, флоаты оказались как нельзя кстати для тех, кто решил уходить с табличной вёрстки.

# Когда флоатов много, а места мало [12/32]

Следует отметить, что поведение нескольких флоатов, когда им не хватает места в одной строке, является очень странным.

Когда не влезающий флоат переносится на новую строку, возможно несколько вариантов и не все из них логичны. Например, флоат может «зацепиться» за один из предшествующих флоатов и встать ниже не в самом начале строки, а за предшествующим.

В этом упражнении примеры специально подобраны так, чтобы вы увидели странности поведения флоатов при недостатке места.

Естественно, эти особенности повлияли на приёмы использования флоатов в создании раскладок (еще одно название для сеток). Но об этом позже.

# Испытание: паззл на флоатах [13/32]

В качестве первого испытания мы подготовили небольшую головоломку. Суть её такова:

Все CSS-стили готовы к использованию, а редактор CSS заблокирован.  
Но вы можете изменять HTML.  
Задача заключается в том, чтобы правильно подобрать порядок блоков.

Конечно, головоломка сделана таким образом, чтобы вы прочувствовали все тонкости поведения флоатов. Обратите внимание, что среди блоков есть не зафлоаченный блок. И это не ошибка.

Напомним об одной тонкости: для того, чтобы флоатный блок мог обтекаться обычным, он должен в коде располагаться выше обычного.

# Свойство clear [14/32]

Свойство clear запрещает обтекание элемента другими элементами. Вот его значения:

1. left — запрещено обтекание слева;
2. right — запрещено обтекание справа;
3. both — запрещено обтекание с обеих сторон;
4. none — обтекание разрешено.

Если после флоатного элемента расположен элемент с запрещённым обтеканием, то последний опускается под флоатный.

Свойство clear учит блочные элементы «видеть» зафлоаченные.

# Борьба с выпаданием флоатов: распорки [15/32]

Эффект выпадания флоатов из родителя был большой проблемой при построении сеток. Ведь сетки на флоатах обычно делаются вот так:

1. Создаём блок-контейнер для колонок.
2. В контейнер добавляем флоатные блоки-колонки.
3. Рассчитываем ширины колонок так, чтобы им хватало места в родителе.

Такой подход работает неплохо. Но проблемы начинаются тогда, когда у контейнера есть фон, отличный от фона страницы. Когда колонки выпадали, родитель схлопывался и фон пропадал.

Необходимо было найти способ борьбы с выпаданием. Тут и пригодилось свойство clear:both. Стали применять такую структуру:

<div class="container"> - блок-контейнер

<div class="column1">...</div> - колонка, флоат

<div class="column2">...</div> - колонка, флоат

<div class="clearfix"></div> - распорка с clear:both

</div>

Т. е. начали добавлять после зафлоаченных колонок пустой элемент-распорку со свойством clear:both. Этот элемент видел колонки, не давал им пройти через себя, а заодно и растягивал родительский блок по высоте.

Для таких распорок прижилось специальное название класса — clearfix.

# Борьба с выпаданием флоатов: псевдораспорки [16/32]

Хорошие верстальщики не очень любят, когда в вёрстке появляются ненужные дополнительные элементы и стараются от них избавляться.

До поры до времени с блоками-распорками приходилось мириться, т. к. без них было никак не обойтись. Но время шло и в браузерах появилась поддержка так называемых псевдоэлементов.

Псевдоэлементы позволяют с помощью CSS вставить в структуру HTML-документа узлы, которых нет в HTML коде. Т. е. можно вставить в код элемент и не писать его в HTML. Благодаря псевдоэлементам появилось решение проблемы флоатов без использования дополнительного элемента.

Назовём это решение псевдораспорками. Есть несколько его вариаций, но вот одна из распространённых. В CSS добавляется следующее правило:

.clearfix:after{

content:'';

display:table;

clear:both;

}

А затем класс clearfix добавляется к контейнеру, внутри которого лежат флоатные колонки. После этого в контейнер не нужно добавлять дополнительный элемент-распорку, т. к. распорка создаётся с помощью псевдоэлемента.

# Простейшая сетка, шаг 1 [17/32]

Мы разобрали по косточкам свойство float, познакомились с clear и научились бороться с некоторыми проблемами флоатов.

В этой части мы будем практиковаться строить сетки на флоатах.

И начнём с простейшей сетки «хедер - две колонки - футер». Одна колонка будет фиксированной ширины, а вторая будет растягиваться в зависимости от ширины контейнера.

В исходном состоянии у нас 4 блока, расположенных друг под другом. Первым делом займёмся боковой панелью или «сайдбаром»: зафлоатим её влево и зададим её фиксированную ширину.

# Простейшая сетка, финал [19/32]

Поздравляем! Вы победили выпадание. Теперь завершим сетку.

В почти готовую сетку мы добавили немного текста, чтобы посмотреть как она будет себя вести с наполнением. Сейчас видно, что синий блок контента частично залезает под сайдбар. Чтобы избавиться от этого эффекта, можно задать блоку контента маргин слева.

После этого сетка будет завершена. Кстати, мы получили ещё один интересный результат. Мы не зафлоатили блок контента, поэтому он тянется на всю ширину родительского контейнера. А фиксированный маргин слева предохраняет его от заползания под сайдбар.

Таким образом, у нас получилась сетка из двух колонок, одна из которых фиксированная, а вторая тянется. Здесь важно следующее — сайдбар в HTML коде должен располагаться до блока контента, иначе такого эффекта не получится.

# Сетка посложнее, шаг 1 [20/32]

Теперь построим более сложную сетку. В конце у нас получится вот такой результат:



Да, мы повторяем раскладку главной страницы HTML Academy: двухколончатый хедер и контент в три колонки.

На этом шаге создадим сетку хедера. Пока что работаем без содержания.

# Сетка посложнее, шаг 2 [21/32]

Обратите внимание на способ именования классов: колонки имеют класс layout-column.

Префикс layout обозначает «сетка» или «раскладка». Таким образом, мы даём понять, что классы с этим префиксом предназначены для создания сетки. Обычно для таких классов задают только флоаты, размеры и внешние отступы.

Использование значимых префиксов для именования классов — хороший приём. Используйте его в своей вёрстке.

На втором шаге мы создадим колонки содержания.

# Сетка посложнее, шаг 3 [22/32]

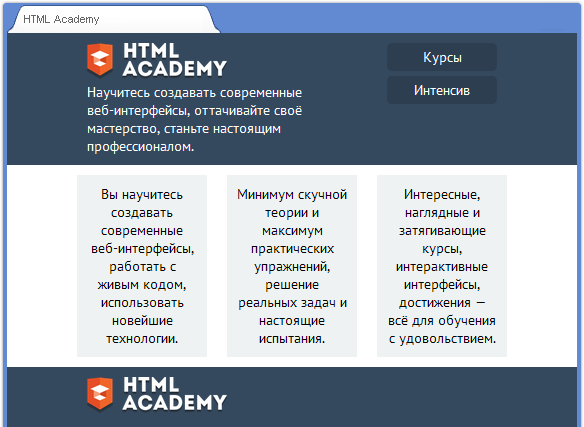
Теперь нам нужно задать отступы между колонками содержания. Сделать это просто — добавим им маргин справа.

Но есть небольшая проблема. По макету последняя колонка должна прижиматься к правому краю контейнера. Поэтому нам нужно обнулить маргин справа у последней колонки в содержании, иначе она не будет влезать в контейнер и перенесётся на следующую строку.

Чтобы выбрать последнюю колонку, вы можете использовать псевдокласс :last-child, с которым вы уже знакомились в разделе про селекторы.

# Последняя сетка, шаг 1 [24/32]

Поздравляем с созданием первой сложной сетки! Теперь давайте немного усложним её. Мы будем делать сетку, у которой фон хедера и футера растягивается на всю ширину окна браузера, вот так:



Сначала нам нужно изменить HTML-разметку. Мы избавимся от блока wrapper, который центрует всё содержание страницы.

Затем нужно будет добавить внутрь хедера, контента и футера дополнительный блок, который будет центровать их содержимое. Вот так:

<div class="header clearfix">

<div class="layout-positioner">

<!-- блоки колонок -->

</div>

</div>

Не забудьте правильно закрыть все тэги. Наша система проверки пока не может отследить эту ошибку

# Последняя сетка, шаг 2 [25/32]

Блоки с классом layout-positioner добавлены в код. Это невидимые блоки-центровщики. Теперь хедер и футер могут тянуться на всю ширину окна, а их содержимое будет отцентровано с помощью добавленных блоков.

Мы задали голубую рамку для центровщиков, чтобы было видно, как они себя ведут. Также мы добавили им псевдораспорки, поэтому можно удалить класс clearfix из HTML кода.

И еще одна мелочь: класс wrapper уже нигде не используется, поэтому он удалён из CSS. Класс clearfix мы оставили в CSS, т. к. это вспомогательный класс, который может пригодиться при вёрстке.

На этом шаге ваша задача — отцентровать центровщики.

## <https://codepen.io/paalso/pen/BaaQGor>

# Испытание: строим сетку [27/32]

В этом испытании задача очень простая:

Нужно построить схематичную сетку макета, как в образце. Вы можете писать любой HTML и любой CSS, без каких либо ограничений.

Единственная тонкость — учитывайте общую ширину макета, которая задана в body. Это сделано, чтобы упростить проверку. И не забывайте пользоваться новыми инструментами для сравнения результата и образца.

Как обычно, все размеры и отступы кратны 5, а используемые цвета перечислены в комментарии в CSS.

# Погружение в inline-block [28/32]

C блочно-строчными элементами вы уже знакомились в разделе про блочную модель документа. Напомним, что это элементы, у которых свойство display имеет значение inline-block.

Блочно-строчные элементы ведут себя двояко. Снаружи они выглядят как обычные строчные, но внутри они ведут себя как блочные.

От строчных им достались следующие черты:

* по ширине они ужимаются под своё содержимое;
* могут располагаться в одну строку;
* реагируют на вертикальное выравнивание, vertical-align;
* реагируют на горизонтальное выравнивание, text-align, заданное у родителя.

От блочных:

* им можно задавать размеры с помощью width и height;
* а также внешние и внутренние отступы и рамки, которые работают во всех направлениях и увеличивают размер элемента.

# float vs inline-block [29/32]

Блочно-строчные элементы могут выстраиваться друг за другом, а если не хватает места, то они переносятся на следующую строку.

Поведение похоже на флоатные элементы, однако, есть существенное отличие.

При переносе на следующую строку блочно-строчные ведут себя намного логичней. Высота строк у них рассчитывается по максимальному элементу с учётом отступов.

Это упражнение создано, чтобы продемонстрировать отличие в поведении флоатов и инлайн-блоков. В исходном состоянии мы видим несколько флоатных блоков, которые не смогли выстроиться в ряды, так как зацепились друг за друга.

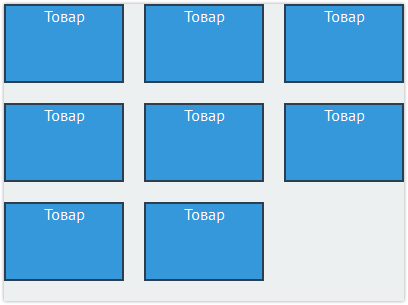
Вы измените тип блоков на блочно-строчные и сами всё увидите, насколько умнее ведут себя инлайн-блоки.

# Простая сетка на inline-block [30/32]

Блочно-строчные элементы очень часто используют для создания декоративных элементов: кнопок, плашек, блочков. Также благодаря их умному поведению с их помощью создают различные списки товаров в каталогах.

Для создания сеток страниц эти элементы используются реже. Первая причина заключается в том, что их не поддерживают старые браузеры, например, IE7 и младше. А сетка страницы — слишком критичная вещь, чтобы позволять ей ломаться даже в старых браузерах.

Есть и еще одна проблема, с которой мы столкнёмся в этом задании. Давайте попробуем построить сетку, как на картинке, когда у нас края товаров вплотную прилегают к краям контейнера:



Ширина контейнера 400px, ширина товара с рамками 120px, отступ между товарами 20px. Чтобы три товара влезали в строку «пиксель-в-пиксель», нужно у каждого третьего товара обнулить отступ справа. Используем для этого селектор .item:nth-child(3n).

# inline-block и пробелы в коде [31/32]

Мы рассчитали всё правильно, однако по три товара в строку не помещается.

Причина заключается в пробелах после тэгов в HTML-коде. Блочно-строчные ведут себя как текст, поэтому если в коде есть пробел между элементами, то он отображается и на странице. Этот пробел увеличивает отступы между товарами, не давая им поместиться в одну строку.

Бороться с пробелом после блочно-строчных можно несколькими способами:

* удалять пробелы в коде;
* обнулять размер шрифта;
* играться с маргинами после блочно-строчного.

У каждого из способов есть свои недостатки, а подробнее эти и другие способы разбираются в [нашем переводе хорошей статьи](https://htmlacademy.ru/blog/21) (есть ещё более обширная [статья](http://css-live.ru/articles/zagadochnye-otstupy-mezhdu-inlajn-blokami.html) про блочно-строчные).

Мы попробуем последние два способа.

Способ со шрифтом заключается в том, что мы задаём нулевой размер шрифта у контейнера инлайн-блоков, а самим инлайн-блокам задаём исходный размер шрифта. Способ не работает, если вы используете относительные размеры шрифта.

Способ с маргинами заключается в том, что мы уменьшаем отступ после инлайн-блока на ширину пробела, около 4px. А если нам нужно, чтобы элементы стояли вплотную друг к другу, то задаём отрицательный отступ. Проблема с этим способом заключается в том, что размер пробела может быть разным в разных шрифтах и может изменяться при изменении размера шрифта.

# Испытание: котогалерея на inline-block [32/32]

В этом испытании вам нужно доверстать галерею картин с котами с помощью блочно-строчных элементов.

HTML-код полностью готов и заблокирован. CSS-код готов частично. Мы сделали заготовки стилей, чтобы вам не нужно было верстать всё с нуля и угадывать не относящиеся к сетке мелочи.

Вам нужно доверстать рамки, задать внешние и внутренние отступы элементов галереи, а также выстроить их в указанную сетку.

Примените те приёмы работы с блочно-строчными элементами, которые разбирались в предыдущих заданиях.