



КАК СТАТЬ АВТОРОМ

Выгорание в IT

Раскрываем новогодние секреты Хабра



Andriell

16 сен 2019 в 20:18

Полное руководство по Flexbox



11 мин



539K

Веб-дизайн*, CSS*, HTML*

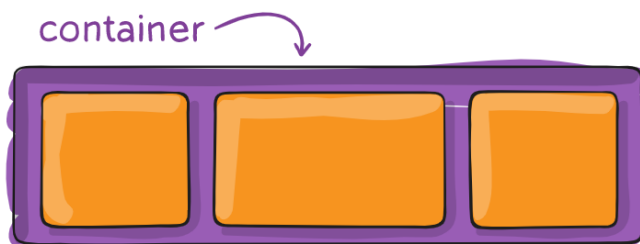
Перевод

Автор оригинала: css-tricks.com

Полное руководство по CSS flexbox. Это полное руководство объясняет все о flexbox, сосредотачиваясь на всех возможных свойствах для родительского элемента (контейнер flex) и дочерних элементов (элементы flex). Оно также включает в себя историю, демонстрации, шаблоны и таблицу поддержки браузеров.

[▶ Background](#)[▶ Основы и терминология](#)

Свойства для Родителя (flex контейнер)



display

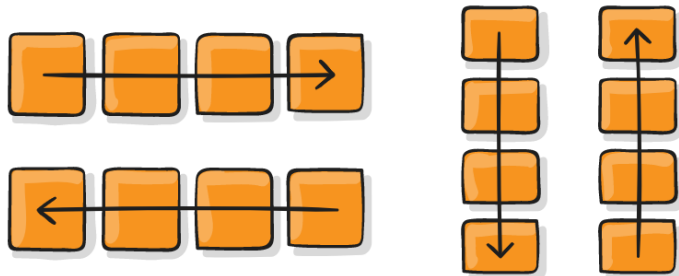
Определяет flex контейнер; inline или block в зависимости от заданного значения. Включает flex контекст для всех потомков первого уровня.

```
.container {  
  display: flex; /* or inline-flex */  
}
```

Имейте в виду:

Обратите внимание, что CSS-столбцы `columns` не влияют на flex контейнер.

flex-direction

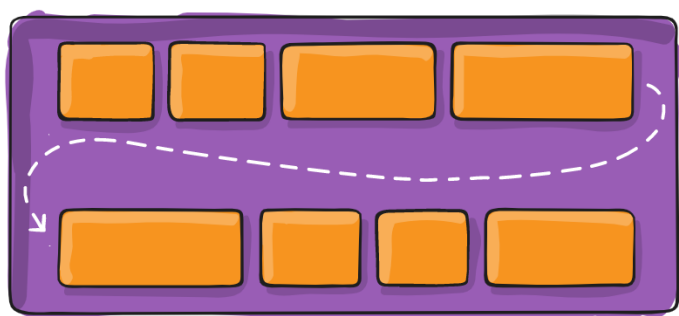


Устанавливает основную ось, таким образом определяя направление flex элементов, помещаемых в flex контейнер. Flexbox – это (помимо дополнительной упаковки) концепция однонаправленного макета. Думайте о flex элементах, как о первичных раскладках в горизонтальных рядах или вертикальных столбцах.

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

- **row** (по умолчанию): слева направо в **ltr**; справа налево в **rtl**
- **row-reverse** справа налево **ltr**; слева направо в **rtl**
- **column**: так же, как и **row** но сверху вниз
- **column-reverse**: то же самое, **row-reverse** но снизу вверх

flex-wrap



По умолчанию гибкие элементы будут пытаться уместиться на одной строке. Вы можете изменить это и позволить элементам переходить на новую строку по мере необходимости с

ПОМОЩЬЮ ЭТОГО СВОЙСТВА.

```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

- **nowrap** (по умолчанию): все flex элементы будут в одной строке
- **wrap**: flex-элементы будут перенесены на несколько строк сверху вниз.
- **wrap-reverse**: flex-элементы будут перенесены на несколько строк снизу вверх.

Посмотреть визуальные демоверсии поведения flex-wrap можно [здесь](#).

flex-flow (Применяется к: родительскому элементу flex-контейнера)

Это сокращение для **flex-direction** и **flex-wrap** свойств, которые вместе определяют основные и поперечные оси flex контейнера. Значением по умолчанию является **row nowrap**.

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

justify-content

flex-start



flex-end



center



space-between



+11

408



5



space-evenly



Это свойство определяет выравнивание вдоль главной оси. Оно помогает распределить дополнительный остаток свободного пространства, когда-либо все flex элементы в строке негибкие, либо гибкие, но достигли своего максимального размера. Это также обеспечивает некоторый контроль над выравниванием элементов, когда они переполняют линию.

```
.container {  
  justify-content: flex-start | flex-end | center | space-between | space-around |  
}
```

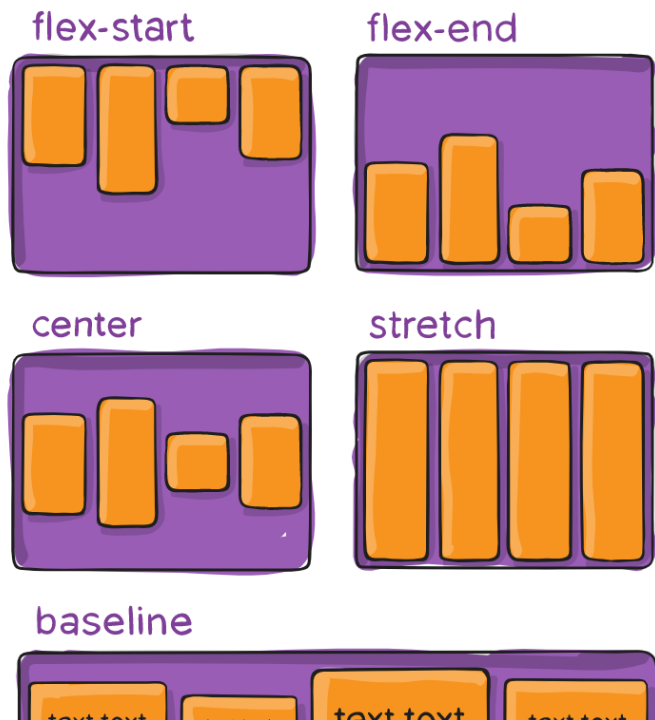
- **flex-start** (по умолчанию): элементы сдвинуты в начало flex-direction направления.
- **flex-end**: элементы сдвинуты ближе к концу flex направления.
- **start**: элементы сдвинуты к началу writing-mode направления.
- **end**: элементы сдвинуты в конце writing-mode направления.

- **left**: элементы сдвинуты по направлению к левому краю контейнера, если это не имеет смысла **flex-direction**, тогда он ведет себя как **start**.
- **right**: элементы сдвинуты по направлению к правому краю контейнера, если это не имеет смысла **flex-direction**, тогда он ведет себя как **start**.
- **center**: элементы центрированы вдоль линии
- **space-between**: элементы равномерно распределены по линии; первый элемент находится в начале строки, последний элемент в конце строки
- **space-around**: элементы равномерно распределены по линии с одинаковым пространством вокруг них. Обратите внимание, что визуально пространства не равны, так как все элементы имеют одинаковое пространство с обеих сторон. Первый элемент будет иметь одну единицу пространства напротив края контейнера, но две единицы пространства между следующим элементом, потому что у следующего элемента есть свой собственный интервал, который применяется.
- **space-evenly**: элементы распределяются таким образом, чтобы расстояние между любыми двумя элементами (и расстояние до краев) было одинаковым.

Обратите внимание, что поддержка браузером этих значений имеет свои нюансы. Например, **space-between** никогда не получал поддержку Edge, а **start** / **end** / **left** / **right** еще нет в Chrome. В MDN есть подробные графики. Самые безопасные значения это **flex-start**, **flex-end** и **center**.

Есть также два дополнительных ключевых слова, которые вы можете связать с этими значениями: **safe** и **unsafe**. Использование **safe** гарантирует, что как бы вы ни занимались этим типом позиционирования, вы не сможете расположить элемент таким образом, чтобы он отображался за пределами экрана (например, сверху) так, чтобы содержимое тоже не могло быть прокручено (это называется «потеря данных»).

align-items



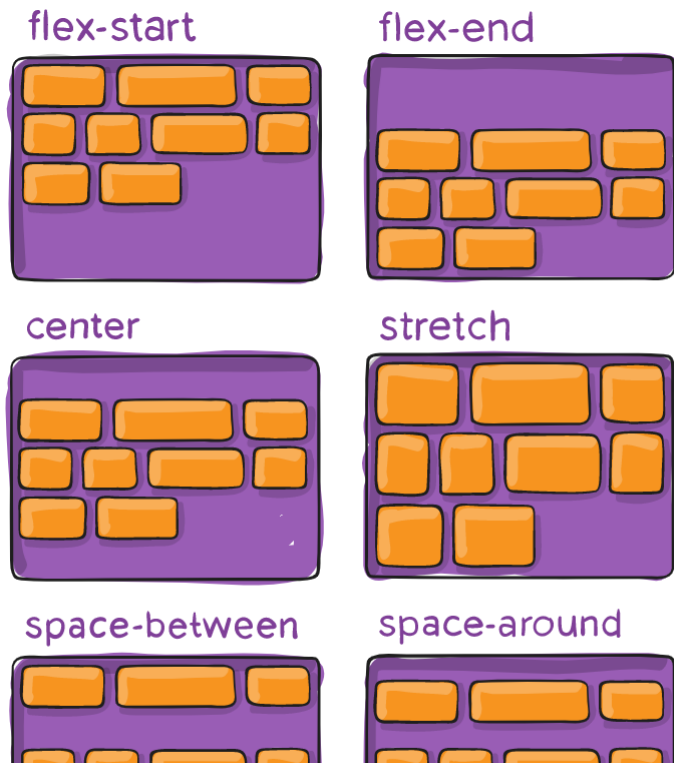
Это свойство определяет поведение по умолчанию того, как flex элементы располагаются вдоль поперечной оси на текущей линии. Думайте об этом как о justify-content версии для поперечной оси (перпендикулярной главной оси).

```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline | first baseline
}
```

- **stretch** (по умолчанию): растягивать, чтобы заполнить контейнер (все еще соблюдаются min-width / max-width)
- **flex-start** / **start** / **self-start**: элементы размещаются в начале поперечной оси. Разница между ними невелика и заключается в соблюдении **flex-direction** правил или **writing-mode** правил.
- **flex-end** / **end** / **self-end**: элементы располагаются в конце поперечной оси. Разница опять-таки тонкая и заключается в соблюдении **flex-direction** или **writing-mode** правил.
- **center**: элементы центрированы по поперечной оси
- **baseline**: элементы выровнены, по их базовой линии

safe и **unsafe** ключевые слова модификаторов могут быть использованы в сочетании со всеми из этих ключевых слов (хотя это поддерживается не всеми браузерами), это помогает предотвратить выравнивание элементов таким образом, что содержание становится недоступным.

align-content



Это свойство выравнивает линии в пределах flex контейнера, когда есть дополнительное пространство на поперечной оси, подобно тому, как **justify-content** выравнивает отдельные элементы в пределах главной оси.

Примечание: это свойство не действует, когда есть только одна строка flex элементов.

```
.container {  
  align-content: flex-start | flex-end | center | space-between | space-around | s;  
}
```

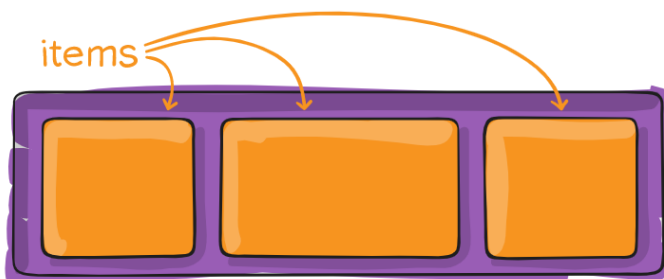
- **flex-start** / **start**: элементы, сдвинуты в начало контейнера. Более поддерживаемый **flex-start** использует, **flex-direction** в то время как **start** использует **writing-**

mode направление.

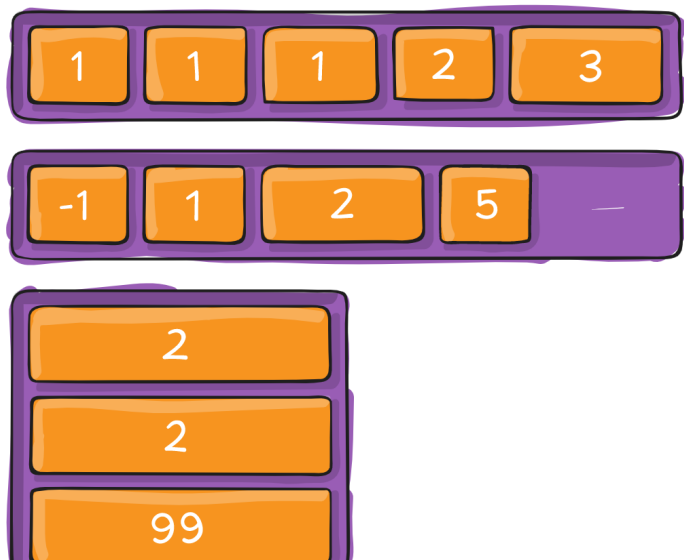
- **flex-end** / **end**: элементы, сдвинуты в конец контейнера. Более поддерживаемый **flex-end** использует **flex-direction** в то время как **end** использует **writing-mode** направление.
- **center**: элементы выровнены по центру в контейнере
- **space-between**: элементы равномерно распределены; первая строка находится в начале контейнера, а последняя — в конце
- **space-around**: элементы равномерно распределены с равным пространством вокруг каждой строки
- **space-evenly**: элементы распределены равномерно, вокруг них одинаковое пространство
- **stretch** (по умолчанию): линии растягиваются, чтобы занять оставшееся пространство

safe и **unsafe** ключевые слова модификаторов могут быть использованы в сочетании со всеми из этих ключевых слов (хотя это поддерживается не всеми браузерами), это помогает предотвратить выравнивание элементов таким образом, что содержание становится недоступным.

Свойства для первых дочерних элементов(flex элементы)



order



По умолчанию flex элементы располагаются в исходном порядке. Однако свойство **order** управляет порядком их появления в контейнере flex.

```
.item {  
  order: <integer>; /* default is 0 */  
}
```

flex-grow



Это свойство определяет способность flex элемента растягиваться в случае необходимости. Оно принимает значение от нуля, которое служит пропорцией. Это свойство, какое количество доступного пространства внутри гибкого контейнера должен занимать элемент.

Если для всех элементов **flex-grow** установлено значение 1, оставшееся пространство в контейнере будет равномерно распределено между всеми дочерними элементами. Если один из дочерних элементов имеет значение 2, этот элемент займет в два раза больше места, чем остальные (или попытается, по крайней мере).

```
.item {  
  flex-grow: <number>; /* default 0 */  
}
```

Отрицательные числа не поддерживаются.

flex-shrink

Это свойство определяет способность гибкого элемента сжиматься при необходимости.

```
.item {  
  flex-shrink: <number>; /* default 1 */  
}
```

Отрицательные числа не поддерживаются.

flex-basis

Это свойство определяет размер элемента по умолчанию перед распределением оставшегося пространства. Это может быть длина (например, 20%, 5rem и т.д.) Или ключевое слово. Ключевое слово **auto** означает «смотри на мое width или height свойство». Ключевое слово **content** означает «размер на основе содержимого элемента» – это ключевое слово все еще не очень хорошо поддерживается, так что трудно проверить что для него используется **max-content**, **min-content** или **fit-content**.

```
.item {  
  flex-basis: <length> | auto; /* default auto */  
}
```

Если установлено значение **0**, дополнительное пространство вокруг содержимого не учитывается. Если установлено значение **auto**, дополнительное пространство распределяется в зависимости от его **flex-grow** значения.

▸ [Смотрите этот рисунок.](#)

flex

Это сокращение для использования **flex-grow**, **flex-shrink** и **flex-basis** вместе. Второй и

третьи параметры (**flex-shrink** и **flex-basis**) являются необязательными. По умолчанию это **0 1 auto**.

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]  
}
```

Рекомендуется использовать это сокращенное свойство, а не устанавливать отдельные свойства. Это сокращение разумно устанавливает другие значения.

align-self

flex-start

Это свойство позволяет переопределить выравнивание по умолчанию (или указанное с помощью **align-items**) для отдельных элементов flex.

Пожалуйста, смотрите **align-items** свойство, чтобы понять доступные значения.

```
.item {  
  align-self: auto | flex-start | flex-end | center | baseline | stretch;  
}
```

Обратите внимание что свойства **float**, **clear** и **vertical-align** не влияют на flex элементы.

Примеры

Давайте начнем с очень простого примера, решающего почти ежедневную проблему: идеальное центрирование. Самое простое решение для этой задачи – это использовать flexbox.

```
.parent {  
  display: flex;
```

```
height: 300px; /* Или что угодно */
}

.child {
width: 100px; /* Или что угодно */
height: 100px; /* Или что угодно */
margin: auto; /* Магия! */
}
```

Так происходит благодаря тому, что свойство вертикального выравнивания `margin` установленное в **auto** во flex контейнере, поглощает дополнительное пространство. Таким образом, установка `margin` в **auto** сделает объект идеально отцентрированным по обеим осям.

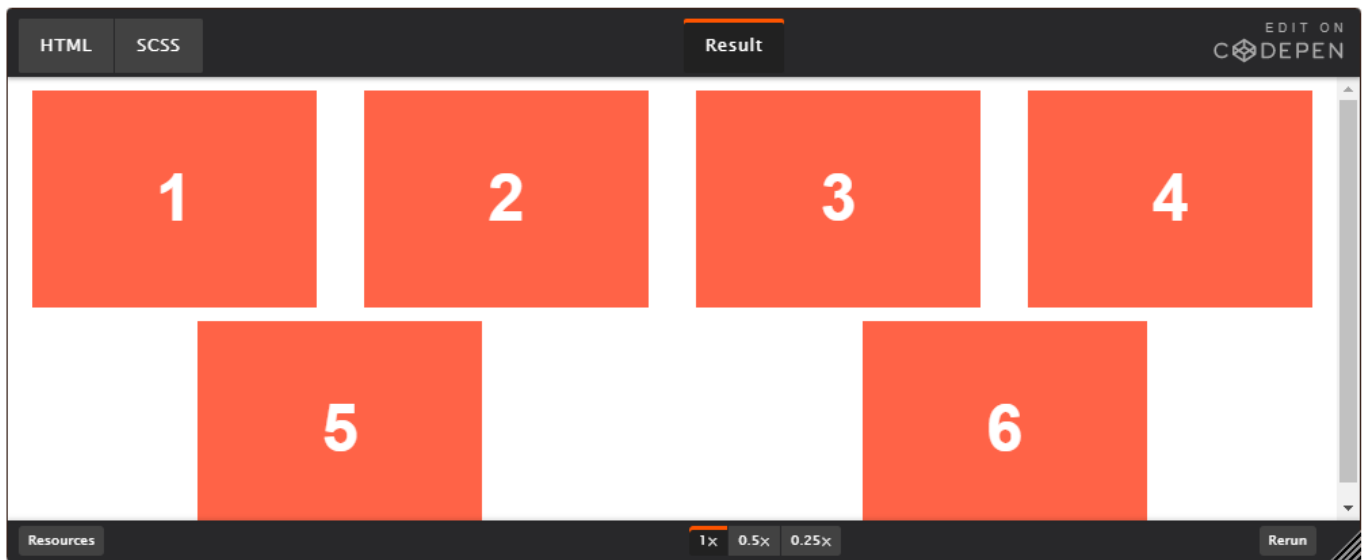
Теперь давайте используем еще несколько свойств. Рассмотрим список из 6 элементов, все с фиксированными размерами, но могут быть и авторазмеры. Мы хотим, чтобы они были равномерно распределены по горизонтальной оси, чтобы при изменении размера браузера все масштабировалось хорошо и без медиа запросов.

```
.flex-container {
/* Сначала мы создаем flex контекст */
display: flex;

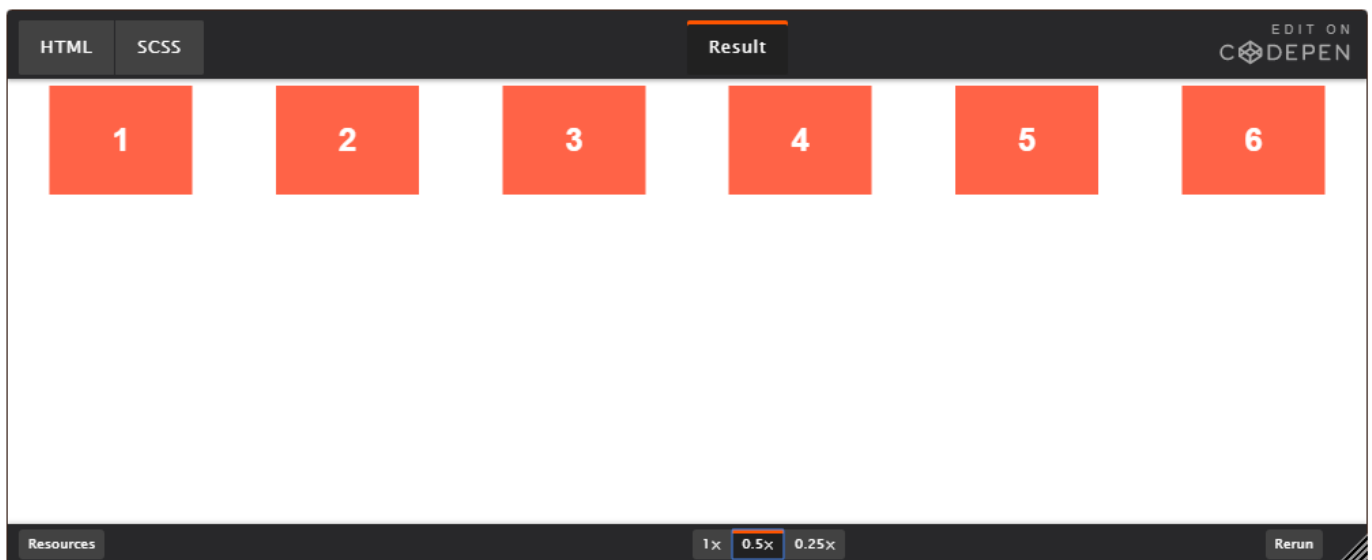
/* Затем мы определяем flex-direction и разрешаем элементам переходить на новые строки
* Запомните, что это то же самое что и:
* flex-direction: row;
* flex-wrap: wrap;
*/
flex-flow: row wrap;

/* Затем мы определяем, как распределяется оставшееся пространство */
justify-content: space-around;
}
```

Готово. Все остальное — это просто стайлинг.



Если изменить разрешение экрана или масштаб, то будет так:



Давайте попробуем что-нибудь еще. Представьте, что у нас есть выровненные по правому краю элементы навигации в верхней части нашего веб-сайта, но мы хотим, чтобы они были выровнены по ширине на экранах среднего размера и располагались в один столбец на небольших устройствах. Это достаточно просто.

```
/* Большие экраны */
.navigation {
  display: flex;
  flex-flow: row wrap;
  /* Это выравнивает элементы по конечной части линии на главной оси */
  justify-content: flex-end;
}

/* Средние экраны */
@media all and (max-width: 800px) {
```

```
.navigation {  
  /* На экранах среднего размера мы центрируем элементы, равномерно распределяя пустое пространство  
  justify-content: space-around;  
}  
  
/* Маленькие экраны */  
@media all and (max-width: 500px) {  
  .navigation {  
    /* На маленьких экранах мы больше не используем направление строки, а используем столбец  
    flex-direction: column;  
  }  
}
```

▸ [Большие экраны](#)

▸ [Средние экраны](#)

▸ [Маленькие экраны](#)

Давайте попробуем что-то еще лучше, играя с гибкостью flex элементов! Как насчет 3-колоночного макета в полную высоту страницы с хедером и футером. И не зависит от исходного порядка элементов.

```
.wrapper {  
  display: flex;  
  flex-flow: row wrap;  
}  
  
/* Мы говорим, что все элементы имеют ширину 100%, через flex-base */  
.wrapper > * {  
  flex: 1 100%;  
}  
  
/* Мы используем исходный порядок для первого мобильного варианта  
* 1. header  
* 2. article  
* 3. aside 1  
* 4. aside 2  
* 5. footer
```

```
*/

/* Средние экраны */
@media all and (min-width: 600px) {
  /* Мы говорим обеим боковым панелям встать в одну строку */
  .aside { flex: 1 auto; }
}

/* Большие экраны */
@media all and (min-width: 800px) {
  /* Мы инвертируем порядок первой боковой панели и основной и говорим главному элементу, что */
  /* */
  .main { flex: 2 0px; }
  .aside-1 { order: 1; }
  .main { order: 2; }
  .aside-2 { order: 3; }
  .footer { order: 4; }
}
```

▸ [Большие экраны](#)

▸ [Средние экраны](#)

▸ [Маленькие экраны](#)

▸ [Префикс для Flexbox](#)

▸ [Другие источники](#)









▸ [Ошибки](#)

Поддержка в браузерах

Разбита по «версии» flexbox:

- (new) означает недавний синтаксис из спецификации (например **display: flex;**)

- (tweener) означает странный неофициальный синтаксис с 2011 года (например **display: flexbox;**)
- (old) означает старый синтаксис с 2009 года (например **display: box;**)

							
Chrome	Safari	Firefox	Opera	IE	Edge	Android	iOS
20- (old) 21+ (new)	3.1+ (old) 6.1+ (new)	2-21 (old) 22+ (new)	12.1+ (new)	10 (tweener) 11+ (new)	17+ (new)	2.1+ (old) 4.4+ (new)	3.2+ (old) 7.1+ (new)

Blackberry Browser 10+ поддерживает новый синтаксис.

Для получения дополнительной информации о том, как смешивать синтаксисы, чтобы получить лучшую поддержку браузера, пожалуйста, обратитесь к этой статье (CSS-хитрости) или этой статье (DevOpera).

Теги: css, html, flexbox

Хабы: Веб-дизайн, CSS, HTML



6

0

Карма Рейтинг

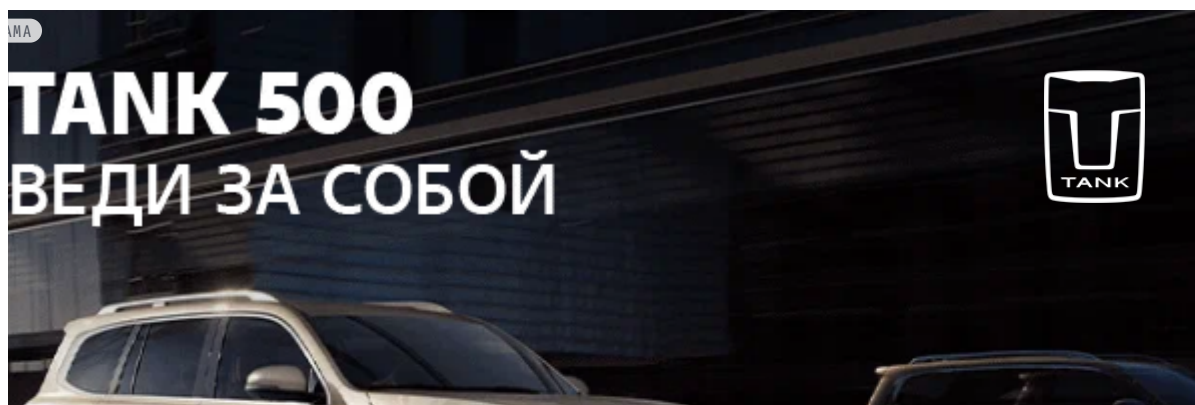
Andriell @Andriell

Программист

Подписаться



Реклама

 Комментарии 5

Публикации

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ

petr97

18 часов назад

Еще один год из жизни ReactOS

 6 мин  6.2K

Ретроспектива

 +71 17 26**shiru8bit**

17 часов назад

Новый год и Atari 2600

 Средний  16 мин  1.8K

Кейс

 +52 14 3**Erwinma1**

21 час назад

Главные мемы 2023 года: атомный советпанк, русы с ящерами, барбенгеймер и гусь-матерщинник

 Простой  7 мин  5.3K

Обзор

 +47 23 2**Rouse**

20 часов назад

Process Memory Map

 3 мин  3.3K

Рoadмэп

 +44 37 11

Rundik

10 часов назад

Традиционный новогодний Хабрачат-2024

 1 мин  980 +20 2 0**CyberexTech**

23 часа назад

Электроника для самых маленьких: или еще один UV излучатель для активации фотополимера

 Средний  4 мин  2.6K[Кейс](#) +20 25 13**PatientZero**

23 часа назад

Быстрый парсинг 8-битных целых чисел

 Простой  6 мин  4.8K[Обзор](#)[Перевод](#) +18 41 14**SkyZion**

2 часа назад

Белгород. Telegram-bot для поиска укрытий

 13 мин  1.9K[Кейс](#) +17 3 2**Exosphere**

10 часов назад

Поздравление-загадка от Хабра

 1 мин  1.3K +14 1 0



SkywardFire

6 часов назад

Статистика по Linux за 2023



Простой



3 мин



3.5K

[Дайджест](#)

+13



4



5

Отсчитываем дни до Нового года и дарим подарки с адвент-календарём

[Турбо](#)[Показать еще](#)

МИНУТОЧКУ ВНИМАНИЯ

[Турбо](#)

Подарки победителям охоты за секретами

[Промо](#)

Чёрная пятница станет вечной: промокодус поймал скидки

[Интересно](#)

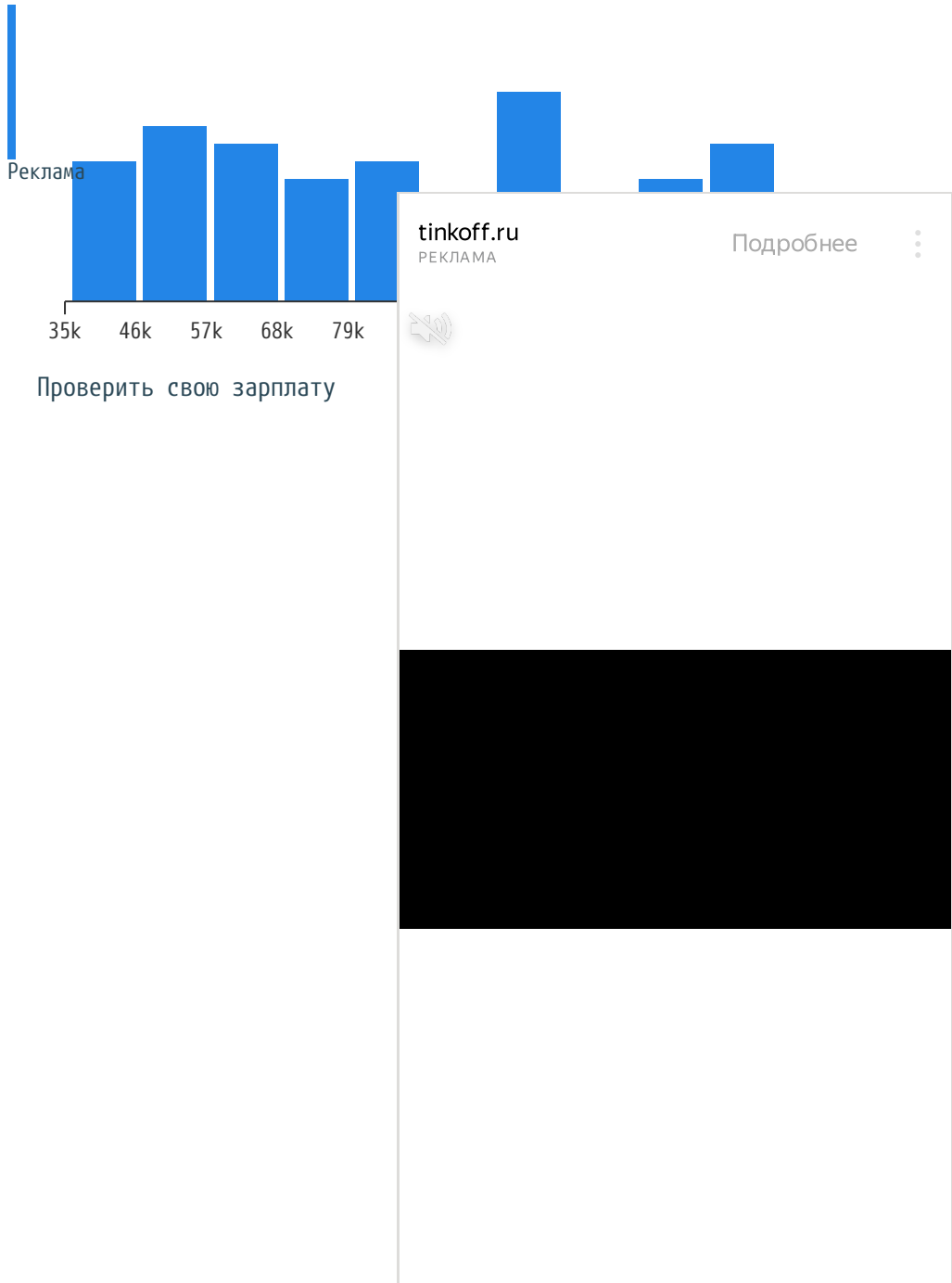
Глупым вопросам и ошибкам — быть! IT-менторство на ХК

СРЕДНЯЯ ЗАРПЛАТА В IT

93 051

₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 34 864 анкет, за 2-ое пол. 2023 года. Проверьте «в рынке» ли ваша зарплата или нет!



Ваш аккаунт

- Профиль
- Трекер
- Диалоги
- Настройки
- ППА

Разделы

- Статьи
- Новости
- Хабы
- Компании
- Авторы
- Песочница

Информация

- Устройство сайта
- Для авторов
- Для компаний
- Документы
- Соглашение
- Конфиденциальность

Услуги

- Корпоративный блог
- Медийная реклама
- Нативные проекты
- Образовательные программы
- Стартапам

[Настройка языка](#)[Техническая поддержка](#)

© 2006–2023, Habr

ЧИТАЮТ СЕЙЧАС

Статистика по Linux за 2023

 3.5K  5

Белгород. Telegram-bot для поиска укрытий

 1.9K  2

Я изучал иностранный язык 3 месяца с Duolingo. Почему это не самый эффективный инструмент для изучения языка

 28K  87

Тебе нужна своя стратегия

 4.9K  22

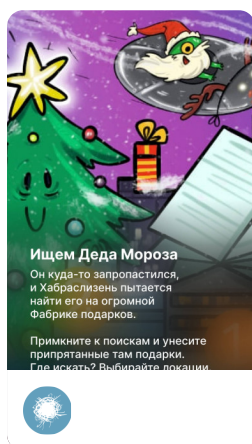
Правильная работа с базой данных на Python

 12K  19

Отсчитываем дни до Нового года и дарим подарки с адвент-календарём

Турбо

ИСТОРИИ



Ищем Деда Мороза



Слово новичкам

Как Дедушке Морозу
выбрать стекВ Новый Год с
Яндекс 360Годнота от
компаний

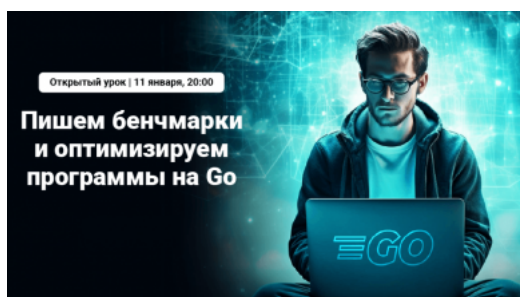
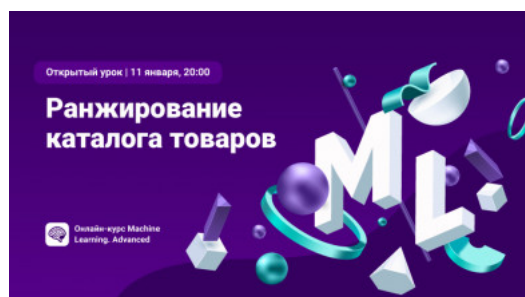
РАБОТА

Веб дизайнер

29 вакансий

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ


77 TECH TALKS
VLADIMIR


12 января / 2024


МИТАП #1

Встречи и разговоры про нашу
проблему в IT. Онлайн и офлайн

Открытый урок «Ранжирование каталога товаров»


 11 января


 20:00


 Онлайн

Подробнее в календаре

Открытый вебинар «Пишем бенчмарки и оптимизируем программы на Go»


 11 января


 20:00


 Онлайн

Подробнее в календаре

Vladimir Tech

 12 января

 18:00 – 21:00

 Онлайн

Подробнее в календаре

Реклама

