

ВЕРСТКА РЕЗИНОВОГО МАКЕТА



СЕМЕН БОЙКО



СЕМЕН БОЙКО

Front-end разработчик



simonderus@gmail.com



fb.me/simonboycko

ПЛАН ЗАНЯТИЯ

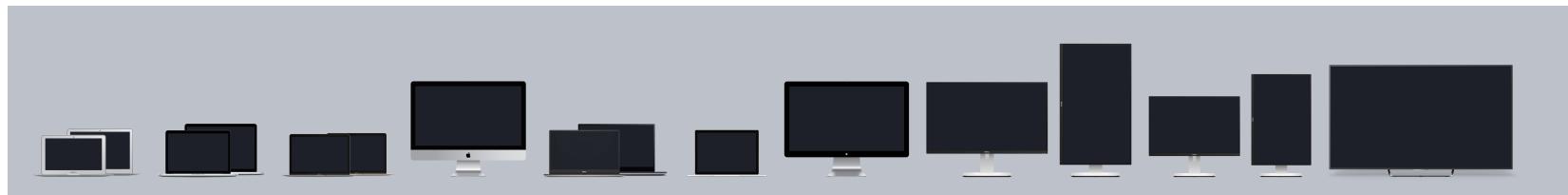
1. Верстка резинового макета
2. Проценты как единица измерения
 - Ширина в процентах
 - Высота в процентах
 - Размеры в процентах для нестатичных блоков
 - min-width max-width
 - Размер шрифта в процентах
 - margin и padding в процентах
3. Flexbox
 - Главная и дополнительная оси
 - flex-direction
 - Прогрессивное улучшение от float к flexbox

ВЕРСТКА РЕЗИНОВОГО МАКЕТА

РАЗНООБРАЗИЕ УСТРОЙСТВ

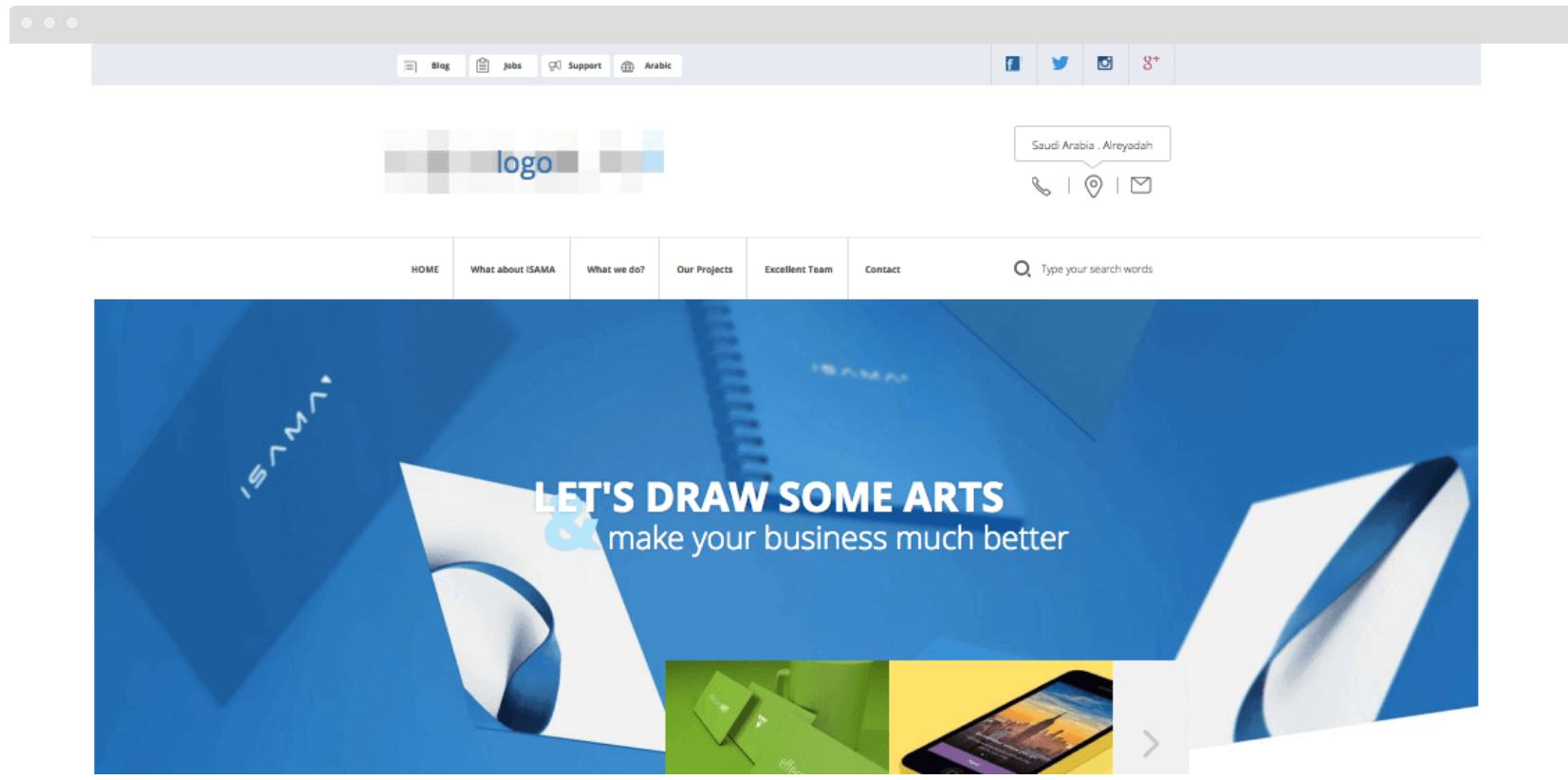
Пользователи могут просматривать веб-страницы, используя множество вариантов устройств, каждое из которых обладает своей шириной экрана и имеет свои особенности.

Поэтому важно, чтобы верстка выглядела хорошо на всех типах устройств.



ФИКСИРОВАННАЯ ВЕРСТКА

При фиксированной верстке на широкоэкраных мониторах сайт будет сиротливо располагаться посередине.



ГОРИЗОНТАЛЬНАЯ ПРОКРУТКА

Если ширина сайта будет больше, чем ширина монитора, то снизу страницы появится полоса прокрутки.



What about ISAMA

ISAMA is a marketing foundation located in Riyadh built on the talent of creation that leads to a modern concept of professionalism. At ISAMA we believe even if you are good at what you do, you have a great product or you provide an excellent service. We still can present you a little better to

Why w

Reaching c them with clients acce efficient ai solutions, w that disting



Резиновая верстка – это верстка, в которой размеры дочерних элементов подстраиваются под размеры их родительских блоков.



ЭЛЕМЕНТЫ ОСТАЮТСЯ НА МЕСТАХ

Резиновая верстка отличается от адаптивной или отзывчивой тем, что при изменении ширины окна браузера элементы будут растягиваться или сжиматься, оставаясь на своих местах.

ПРОЦЕНТЫ ВМЕСТО ПИКСЕЛЕЙ

Достигается такая «резиновость» за счет того, что вместо фиксированных размеров, заданных с помощью пикселей (px), при резиновой верстке ширина элементов задается с помощью относительных единиц, например, процентов (%), что как раз и позволяет им тянуться вслед за родительским блоком.

ПРОЦЕНТЫ КАК ЕДИНИЦА ИЗМЕРЕНИЯ

ШИРИНА В ПРОЦЕНТАХ

Чтобы сделать вёрстку резиновой, нужно позволить блокам растигиваться в ширину. Для этого задаем ширину блока в процентах, например:

`width: 100%;` или `width: 40%;`.

Благодаря этой записи, ширина блока будет рассчитываться динамически и будет напрямую зависеть от ширины родительского блока.

БЛОКИ С ФИКСИРОВАННОЙ ШИРИНОЙ

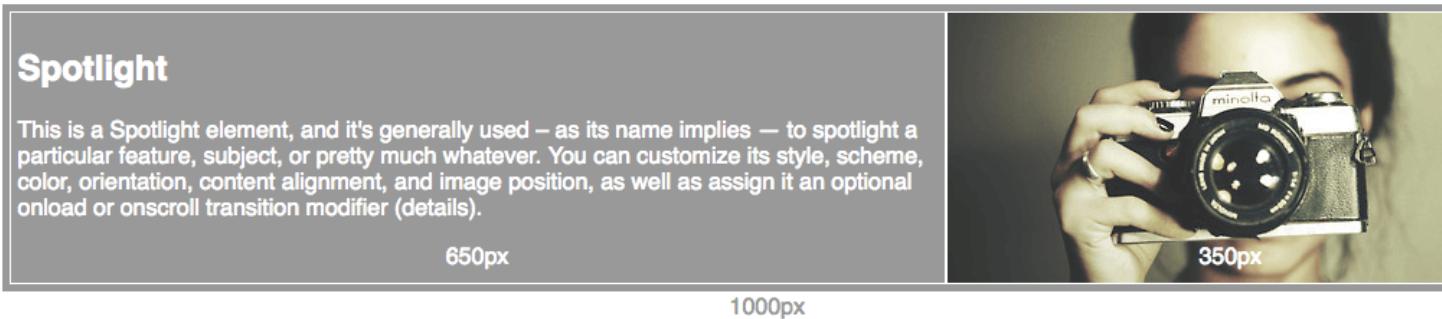
Возьмем блок с классом `.spotlight` и зададим ему ширину `1000px`.

Внутрь этого блока поместим еще два блока и зададим им свои размеры:

```
1 .spotlight {  
2     width: 1000px;  
3 }  
4  
5 .content {  
6     width: 650px;  
7 }  
8  
9 .image {  
10    width: 350px;  
11 }
```

СТАТИЧНАЯ ВЕРСТКА

Добавим остальные CSS-свойства, немного контента и в результате получим следующий элемент интерфейса:



Код примера

МЕНЯЕМ ШИРИНУ РОДИТЕЛЯ

Что произойдет, если ширина блока `.spotlight` изменится до `800px`? Ширина блоков `.content` и `.image` останется неизменной и правый блок «упадет» ниже, на следующую строку, потому что перестал помещаться рядом с левым:

Spotlight

This is a Spotlight element, and it's generally used – as its name implies — to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).



ШИРИНА В ПРОЦЕНТАХ

Если мы перепишем код следующим образом, то при ширине блока `.spotlight` равной `1000px` соотношение «650 к 350» останется прежним, а при изменении ширины `.spotlight` ширина вложенных в него блоков будет меняться пропорционально:

```
1 .image {  
2   width: 35%;  
3 }  
4  
5 .content {  
6   width: 65%;  
7 }
```

ШИРИНА РОДИТЕЛЯ В ПРОЦЕНТАХ

А если задать родителю 100% в качестве значения ширины, то получим блоки, занимающие весь экран браузера независимо от его ширины в той же самой пропорции.

```
1 | .spotlight {  
2 |   width: 100%;  
3 | }
```

РЕЗИНОВЫЙ БЛОК

Spotlight

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).



[Код примера](#)

ВЫСОТА В ПРОЦЕНТАХ

Высота блока также может быть задана в процентах, но здесь все не так просто, как с шириной.

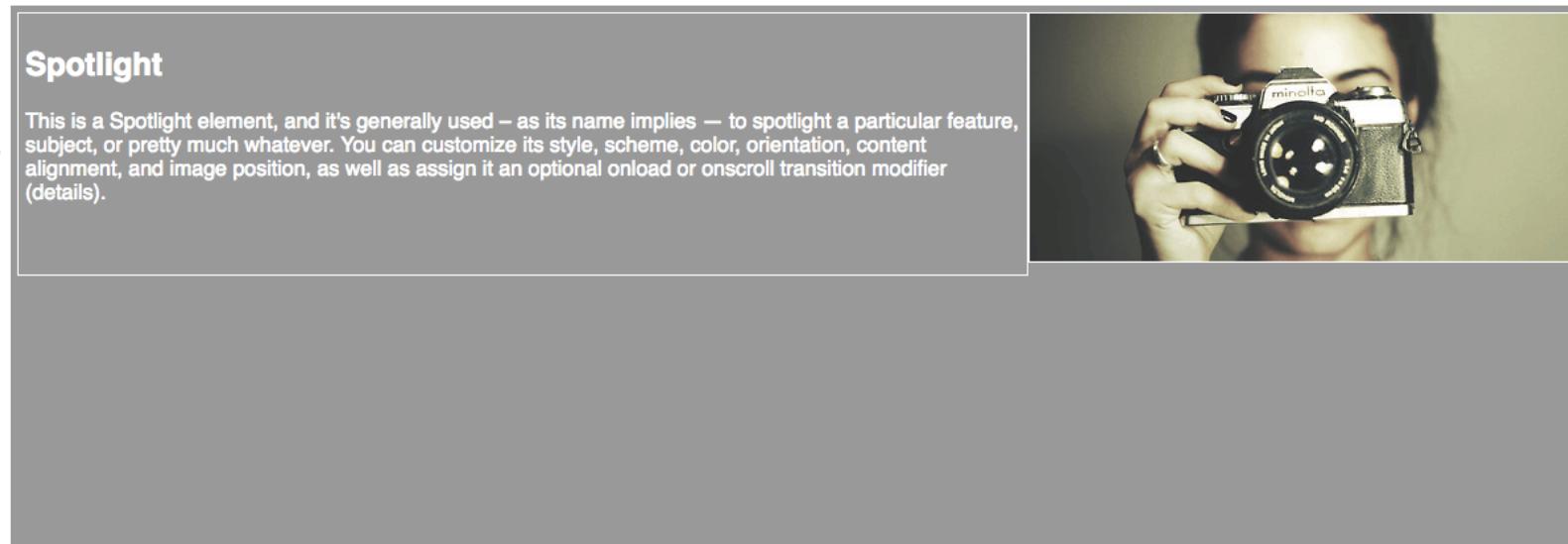
Высота любых элементов с `display: block;` имеет значение `auto` и устанавливается по высоте его содержимого.

ВЫСОТА РОДИТЕЛЯ НЕ ЗАДАНА

Если мы установим для блока `.content` из нашего примера `height: 50%`, то ничего не изменится, так как указание высоты в процентах никак не повлияет на дочерние блоки, если у родителя высота не задана и равна `auto`.

ВЫСОТА РОДИТЕЛЯ УКАЗАНА

А вот если высота блока `.spotlight` будет задана в пикселях (например, `height: 400px`), то при указании высоты `height: 50%` блоку `.content` его фактическая высота примет значение половины высоты родителя (в нашем случае это `200px`).



[Код примера](#)

НЕ УКАЗЫВАЙТЕ ФИКСИРОВАННУЮ ВЫСОТУ

Нужно отметить, что устанавливать фиксированную высоту элементов – **плохая практика**, так как мы не знаем, сколько места будет занимать контент блока, а значит, не знаем, какой высоты он получится в итоге.

ВОЗМОЖНЫЕ ПРОБЛЕМЫ

Если высота контента получится больше, чем высота блока, то такой контент окажется за пределами родительского блока `.spotlight`. Если за родительским блоком будет следовать другой – он будет наезжать поверх текста.

Spotlight

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).

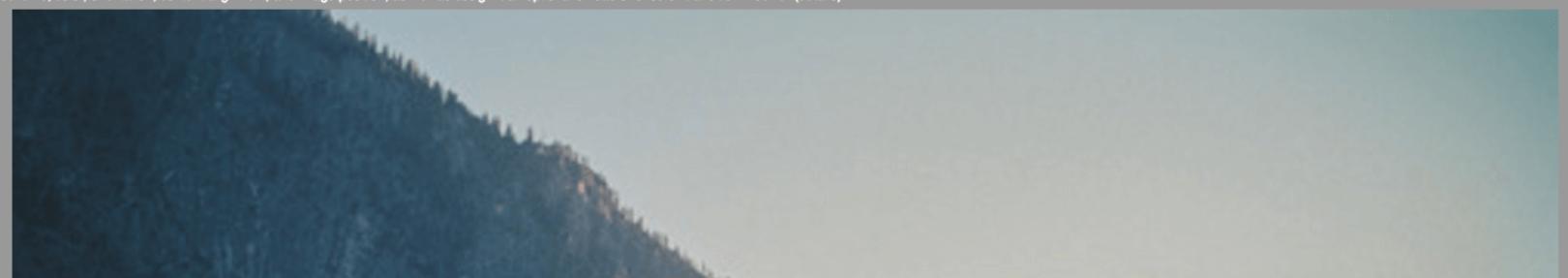
This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).

This is a Spotlight element, and it's generally used – as its name implies – to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).



ПОЗИЦИОНИРОВАННЫЕ БЛОКИ И ПРОЦЕНТЫ

Создадим несложную разметку:

```
1 <div class="wrapper">
2   <div class="parent">
3     parent
4     <div class="child">
5       child
6       </div>
7     </div>
8   </div>
```

СТИЛИ ДЛЯ ПРИМЕРА

```
1 .wrapper {  
2     position: relative;  
3     width: 500px;  
4     height: 200px;  
5     border: 1px dashed red;  
6 }  
7  
8 .parent {  
9     width: 100px;  
10    height: 100px;  
11    border: 2px solid blue;  
12    margin: 20px;  
13 }  
14  
15 .child {  
16     position: absolute;  
17     width: 100%;  
18     height: 100%;  
19     background: rgba(200, 200, 0, 0.5);  
20     top: 0;  
21     left: 0;  
22 }
```

ДОЧЕРНИЙ БЛОК БОЛЬШЕ РОДИТЕЛЯ



Код примера

Мы видим, что в данном случае блок `.child` имеет ширину и высоту большие, чем у непосредственного родителя: ведь его ширина и высота напрямую зависят от ширины и высоты блока `.wrapper`.

ОТ КОТОРОГО РОДИТЕЛЯ СЧИТАЕМ?

Дело в том, что для блоков с `position: absolute` родительским блоком – тем, от которого считается ширина, заданная в процентах – будет ближайший родительский блок с нестатическим типом позиционирования. И это не обязательно непосредственный родитель.

В нашем примере это блок с классом `.wrapper`.

ИЗ АБСОЛЮТНОГО В ФИКСИРОВАННЫЙ

Заменим для блока `.child` абсолютное позиционирование на фиксированное:

```
1 .child {  
2     position: fixed;  
3     width: 100%;  
4     height: 100%;  
5     ...  
6 }
```

РАСЧЕТ ОТ ОКНА БРАУЗЕРА



Код примера

Для блоков с `position: fixed` ширина и высота, заданные в процентах, рассчитываются от размеров окна браузера.

ЗАНИМАЕМ ВСЕ ПРОСТРАНСТВО ОКНА

Создадим заглушку для сайта, находящегося в разработке:

```
1 <div class="wrapper">
2   <div class="wrapper__container">
3     <h1 class="main-title">Mountains is awesome!</h1>
4     <p class="text">Almost as this site.</p>
5     <p class="text">But it is under construction...</p>
6   </div>
7 </div>
```

ФИКСИРУЕМ ПСЕВДОЭЛЕМЕНТ

```
1 .wrapper {  
2     position: relative;  
3 }  
4  
5 .wrapper::before,  
6 .wrapper::after {  
7     content: "";  
8     display: block;  
9     position: fixed;  
10    top: 0;  
11    left: 0;  
12    width: 100%;  
13    height: 100%;  
14 }
```

ДОБАВИМ ФОН И ГРАДИЕНТ

```
1 .wrapper::before {  
2     background-image: linear-gradient(to top,  
3         rgba(19, 21, 25, 0.5),  
4         rgba(19, 21, 25, 0.5));  
5     background-size: cover;  
6     background-position: center;  
7     background-repeat: no-repeat;  
8     z-index: 2;  
9 }  
10  
11 .wrapper::after {  
12     background-image: url("./images/bg.jpg");  
13     background-position: center;  
14     background-size: cover;  
15     background-repeat: no-repeat;  
16     z-index: 1;  
17 }
```

КРАСИВАЯ ЗАГЛУШКА НА ВЕСЬ ЭКРАН

В данном примере нам «пригодилась» возможность задать высоту в процентах для того, чтобы блок `.wrapper` занимал все доступное пространство экрана по высоте.



[Код примера](#)

ПРОЦЕНТЫ И `min-width / max-width`

Выше мы рассмотрели пример, когда мы указываем для контентной области значение `width: 100%` и получаем контейнер, заполняющий всю ширину окна браузера.

Однако такое поведение будет нам иногда мешать.

ШИРОКИЙ ЭКРАН

На очень широких экранах контент может «вытянуться» и занимать всего несколько строчек:

Spotlight

This is a Spotlight element, and it's generally used — as its name implies — to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).



ОГРАНИЧИМ РАСТЯЖЕНИЕ

Для того, чтобы содержимое читалось нормально на очень широких экранах, мы можем ограничить его растяжение, задав предельную точку, по достижению которой блок перестанет тянуться:

```
1 .spotlight {  
2   width: 100%;  
3   max-width: 1280px;  
4 }
```

Теперь, как только ширина блока станет равна `1280px`, блок тянуться перестанет.

ПОМЕСТИМ ПО ЦЕНТРУ

Но теперь при расширении окна браузера справа будет оставаться пустое место.

Поместим блок по центру:

```
1 .spotlight {  
2   width: 100%;  
3   max-width: 1280px;  
4   margin: 0 auto;  
5 }
```

БЛОК НА БОЛЬШОМ ЭКРАНЕ

Spotlight

This is a Spotlight element, and it's generally used — as its name implies — to spotlight a particular feature, subject, or pretty much whatever. You can customize its style, scheme, color, orientation, content alignment, and image position, as well as assign it an optional onload or onscroll transition modifier (details).



[Код примера](#)

УЗКИЙ ЭКРАН

На очень узком экране блок может стать узким, так что на строке не уместится больше одного слова:



ОГРАНИЧИВАЕМ МИНИМАЛЬНУЮ ШИРИНУ

Чтобы запретить блоку становиться уже подходящей нам минимальной ширины, мы можем задать ему значение `min-width`:

```
1 .spotlight {  
2   width: 100%;  
3   max-width: 1280px;  
4   margin: 0 auto;  
5   min-width: 800px;  
6 }
```

Теперь, помимо прочего, как только ширина блока станет равной `800px`, дальше наш блок сжиматься не будет.

[Код примера](#)

РАЗМЕР ШРИФТА В ПРОЦЕНТАХ

А что, если задать в процентах размер шрифта? И так тоже можно:

```
1 | .content {  
2 |   font-size: 50%;  
3 | }
```

В этом случае размер шрифта блока будет вычисляться в зависимости от размера шрифта родителя.

СТРАНИЦА С КНОПКОЙ

Создадим разметку с кнопкой:

```
1 <div class="container">
2   <p>...</p>
3   <button class="btn">Learn more</button>
4 </div>
```

МЕНЯЕМ РАЗМЕР КНОПКИ

Если нам, в соответствии с дизайном, нужно сделать кнопку с размером шрифта, равным размеру текста `14px`, мы можем добиться этого с помощью стилей:

```
1 .container {  
2   font-size: 14px;  
3 }  
4  
5 .btn {  
6   font-size: 14px;  
7 }
```

Здесь нам понадобилось задать размер шрифта кнопки отдельно, так как в соответствии со стилями браузера он по умолчанию равен `13.333px`.

ИЗМЕНЕНИЕ ТРЕБОВАНИЙ

Но теперь дизайнер хочет, чтобы размер шрифта текста на кнопке увеличился до `16px`.

Добиться этого можно, переписав код:

```
1 .container {  
2   font-size: 16px;  
3 }  
4  
5 .btn {  
6   font-size: 16px;  
7 }
```

ТРУДОЗАТРАТНО

Нам потребовалось переписать код в двух местах. А ещё при дальнейшей поддержке проекта можем забыть об этой связи и поменять размер шрифта только в одном месте, тогда связь размеров нарушится.

Есть ли какой-то способ запрограммировать зависимость размеров? Как если бы мы захотели сказать «сделай шрифт кнопки такого же размера, как шрифт текста».

СВЯЗАННЫЕ РАЗМЕРЫ

Такая возможность на самом деле есть – если использовать размер шрифта в процентах:

```
1 .container {  
2   font-size: 14px;  
3 }  
4  
5 .btn {  
6   font-size: 100%;  
7 }
```

МЕНЯЕМ СРАЗУ ВЕЗДЕ

При записи `font-size: 100%` в нашем примере размер шрифта кнопки будет равен `14px`, а если же мы поменяем размер шрифта `.container`, размер шрифта кнопки поменяется «автоматически», что очень удобно.

[Код примера](#)

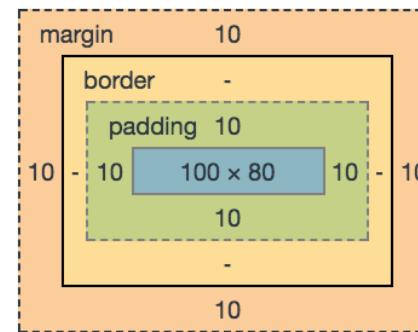
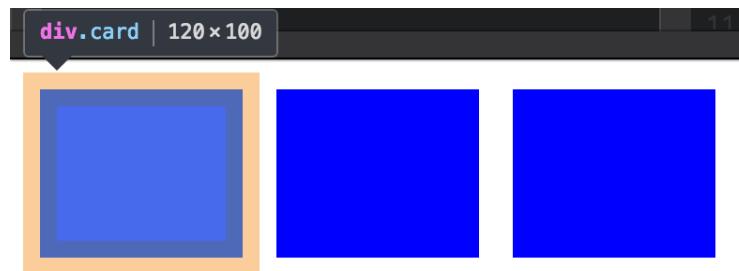
margin / padding В ПРОЦЕНТАХ

При задании в процентах внутренних и внешних отступов блока – `margin` и `padding` – их фактические значения будут рассчитаны в зависимости от ширины родительского блока:

```
1 .card-wrapper {  
2   width: 1000px;  
3 }  
4  
5 .card {  
6   width: 100px;  
7   height: 80px;  
8   margin: 1%;  
9   padding: 1%;  
10 ...  
11 }
```

ФАКТИЧЕСКОЕ ЗНАЧЕНИЕ

Значения всех четырех `padding` и всех четырех `margin` фактически получились `10px`, потому что 1% от `1000px` (ширины родительского блока `.card-wrapper`) равен `10px`.



ВЕРТИКАЛЬНЫЕ ТОЖЕ СЧИТАЕМ ОТ ШИРИНЫ

Здесь нужно особо обратить внимание на то, что вертикальные отступы – `margin-top`, `margin-bottom`, `padding-top`, `padding-bottom` – мы задали по 1% и фактически они получились равными `10px`.

Так произошло потому, что они, так же как и горизонтальные отступы, рассчитываются от ширины блока, а не от высоты, как мы могли бы подумать.

КВАДРАТНЫЕ ФОТОГРАФИИ

Эту особенность можно использовать для создания квадратных блоков с фото.

```
1 <div class="portfolio">
2   <div class="portfolio-item">
3     <div class="portfolio-item__wrapper">
4       
6     </div>
7   </div>
8 </div>
```

СТИЛИ ДЛЯ ПРИМЕРА

Для набора элементов `.portfolio-item` мы можем написать стили:

```
1 .portfolio {  
2     width: 100%;  
3 }  
4  
5 .portfolio-item {  
6     width: 25%;  
7     float: left;  
8 }  
9  
10 .portfolio-item__wrapper {  
11     overflow: hidden;  
12     height: 0;  
13     padding-bottom: 100%;  
14     position: relative;  
15 }
```

ИДЕАЛЬНЫЙ РЕЗИНОВЫЙ КВАДРАТ

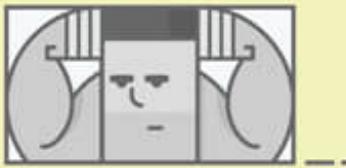
Таким образом мы получим квадратные карточки. При этом соотношение сторон (ширина равна высоте) при изменении ширины родительского блока `.portfolio` будет прежним.

[Код примера](#)

FLEXBOX



CSS flexbox (Flexible Box Layout Module) – модуль макета гибкого контейнера – представляет собой способ раскладки элементов.



flexbox

FLEXBOX ОБЛЕГЧАЕТ РАБОТУ

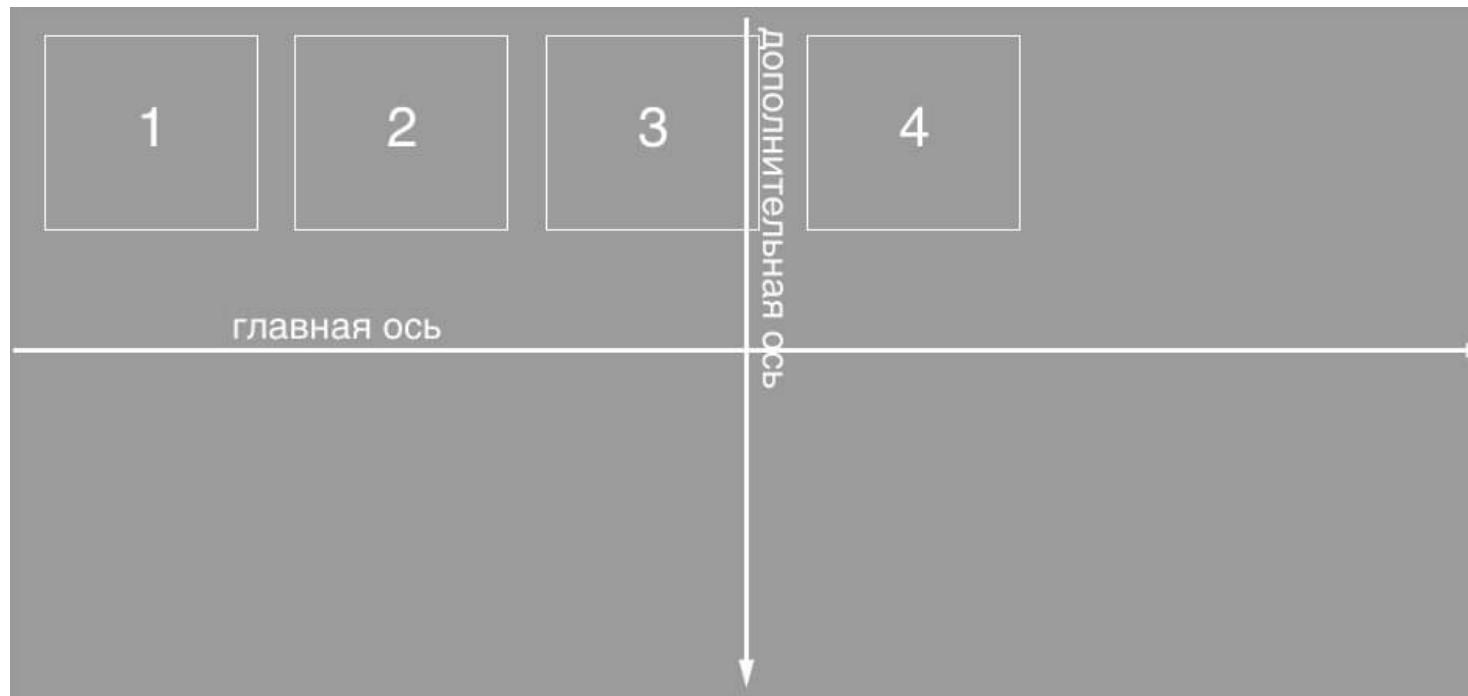
Flexbox состоит из контейнера и flex-элементов – дочерних блоков.

Flexbox позволяет решать множество задач, которые ранее не имели решения либо решались сложно и запутанно.

Все блоки с помощью `display: flex` очень легко делаются «резиновыми», при этом мы с легкостью сможем поменять направление потока элементов.

ГЛАВНАЯ И ДОПОЛНИТЕЛЬНАЯ ОСИ

Flexbox позволяет располагать блоки в двухмерном пространстве и использует для этого две оси – главную и дополнительную:



СЛЕВА НАПРАВО

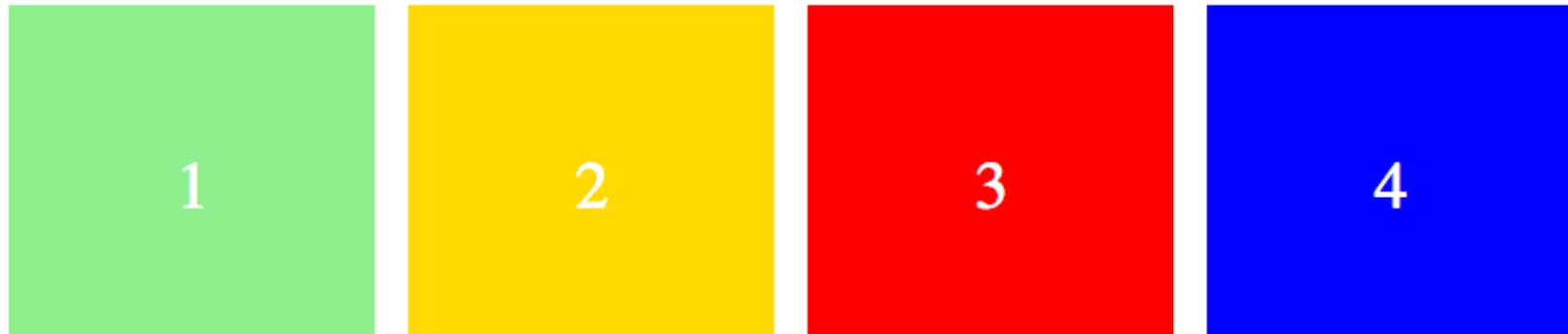
Несколько блоков и `display: flex` для родительского элемента:

```
1 <div class="elements">
2   <div class="element">1</div>
3   <div class="element">2</div>
4   <div class="element">3</div>
5   <div class="element">4</div>
6 </div>
```

```
1 .elements {
2   display: flex;
3 }
```

В СТРОКУ ПО УМОЛЧАНИЮ

По умолчанию flex-элементы выстраиваются вдоль главной оси, слева направо. Поэтому они будут располагаться в ряд по горизонтали, как только мы укажем `display: flex` родительскому блоку.



ВЫСОТА ОТ КОНТЕЙНЕРА

Нужно отметить, что по умолчанию flex-элементы имеют высоту, равную высоте контейнера. А если высота контейнера не задана, то он будет растягиваться и принимать высоту по своему контенту.

ПОДСТРОЙКА ПОД КОНТЕНТ

Если внутри контейнера несколько flex-элементов с контентом разной высоты, то высота контейнера будет совпадать по высоте с элементом наибольшей высоты:

Идейные соображения высшего порядка, а также реализация намеченных плановых заданий влечет за собой процесс внедрения и модернизации существенных финансовых и административных условий. Товарищи! постоянный количественный рост и сфера нашей активности требуют от нас анализа позиций, занимаемых участниками в отношении поставленных задач. Не следует, однако забывать, что рамки и место обучения кадров влечет за собой процесс внедрения и модернизации соответствующий условий активизации. Значимость этих проблем настолько очевидна, что консультация с широким активом позволяет выполнять важные задания по разработке модели развития.

Повседневная практика показывает, что начало повседневной работы по формированию позиции в значительной степени обуславливает создание модели развития. Таким образом начало повседневной работы по формированию позиции требуют определения и уточнения позиций, занимаемых участниками в отношении поставленных задач. Повседневная практика показывает, что начало повседневной работы по формированию позиции играет важную роль в формировании системы обучения кадров, соответствует насущным потребностям. Равным образом начало повседневной работы по формированию позиции представляет собой интересный эксперимент проверки системы обучения кадров, соответствует насущным потребностям.

С другой стороны рамки и место обучения кадров представляет собой интересный эксперимент проверки позиций, занимаемых участниками в отношении поставленных задач. Не следует, однако забывать, что постоянный количественный рост и сфера нашей активности способствует подготовки и реализации форм развития. Задача организации, в особенности же консультация с широким активом позволяет оценить значение форм развития. Товарищи! консультация с широким активом обеспечивает широкому кругу (специалистов) участие в формировании дальнейших направлений развития. Не следует, однако забывать, что начало повседневной работы по формированию позиции в значительной степени обуславливает создание модели развития. Таким образом сложившаяся структура организации играет важную роль в формировании существенных финансовых и административных условий.

Разнообразный и богатый опыт консультация с широким активом представляет собой интересный эксперимент проверки форм развития. Идейные соображения высшего порядка, а также начало повседневной работы по формированию позиции позволяет оценить значение форм развития. Товарищи! консультация с широким активом обеспечивает широкому кругу (специалистов) участие в формировании дальнейших направлений развития. Не следует, однако забывать, что начало повседневной работы по формированию позиции в значительной степени обуславливает создание модели развития. Таким образом сложившаяся структура организации играет важную роль в формировании существенных финансовых и административных условий.

КОЛОНКИ РАВНОЙ ВЫСОТЫ

Часто в практике требуется сверстать две колонки равной высоты. Раньше эта задача имела достаточно трудное решение, теперь решается просто.

Создадим учебную разметку:

```
1 <div class="wrapper">
2   <main class="content">
3     <!-- контент -->
4   </main>
5   <aside class="sidebar">
6     <!-- контент -->
7   </aside>
8 </div>
```

СТИЛИ КОЛОНК

```
1 .wrapper {  
2     display: flex;  
3 }  
4  
5 .content {  
6     width: 80%;  
7     padding: 20px;  
8     background: gold;  
9 }  
10  
11 .sidebar {  
12     width: 20%;  
13     padding: 20px;  
14     background: lightgray;  
15 }
```

БЫСТРЫЙ РЕЗУЛЬТАТ

Вот так просто мы получили две колонки одинаковой высоты:

Lorem Ipsum — это текст- "рыба", часто используемый в печати и веб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum...

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..." "Нет никого, кто любил бы боль саму по себе, кто искал бы её и кто хотел бы иметь её просто потому, что это боль..."

Код примера

flex-direction

Существует свойство `flex-direction`, которое позволяет изменить то, в каком направлении располагаются flex-элементы, переключая направление главной оси.

По умолчанию значение свойства `flex-direction` равно `row`.

МЕНЯЕМ НАПРАВЛЕНИЕ ГЛАВНОЙ ОСИ

Flex-элементы всегда располагаются в направлении главной оси. Однако, если мы указываем `flex-direction: column`, мы меняем местами главную и дополнительную оси.

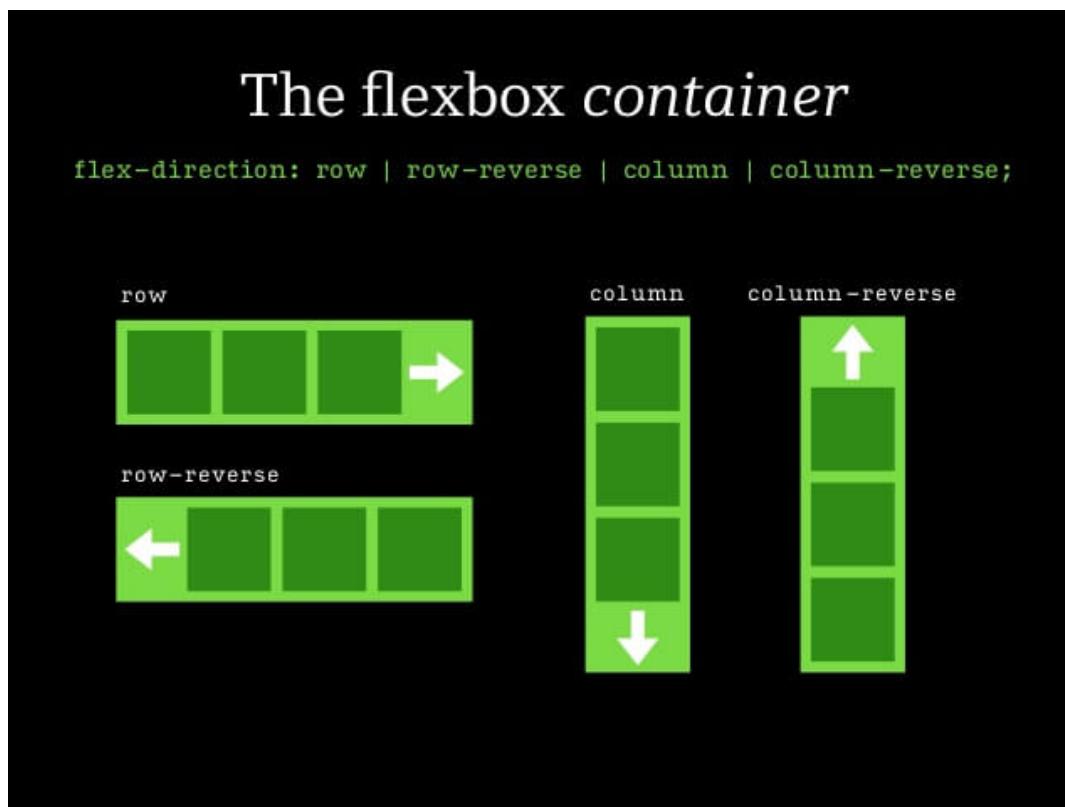
ВОЗМОЖНЫЕ ЗНАЧЕНИЯ

Помимо `column` свойство `flex-direction` может принимать три других значения:

- `row` (по умолчанию);
- `row-reverse`;
- `column-reverse`.

НАПРАВЛЕНИЯ ГЛАВНОЙ ОСИ

При значении `row-reverse` главная ось будет направлена справа налево по горизонтали (в противоположном `row` направлении), а при значении `column-reverse` – снизу вверх по вертикали (в направлении, противоположном `column`):



ПРОГРЕССИВНОЕ УЛУЧШЕНИЕ ОТ float К FLEXBOX

```
1 <section class="news">
2   <article class="news-item">
3     <h3 class="news-item__title">One</h3>
4     
6   </article>
7   <article class="news-item">
8     <h3 class="news-item__title">Two</h3>
9     
11  </article>
12  <article class="news-item">
13    <h3 class="news-item__title">Three</h3>
14    
16  </article>
17 </section>
```

БЛОКИ В СТРОКУ

Для расположения блоков в строчку используем свойство `float`:

```
1 .news {  
2     background-color: #f2f2f2;  
3 }  
4  
5 .news-item {  
6     float: left;  
7     margin: 0 10px;  
8     border: 1px solid;  
9 }  
10  
11 .news-item__title {  
12     padding: 0 10px;  
13 }
```

CLEARFIX

Высота блока, внутри которого лежат только блоки с `float`, равна нулю. В данном примере нам понадобится «хак» для того, чтобы высота блока `.news` не была равна нулю:

```
1 .news::after {  
2   content: "";  
3   display: table;  
4   clear: both;  
5 }
```

ПРЕИМУЩЕСТВА FLEXBOX

Flexbox-раскладка лишена этого недостатка. Запишем ниже стили:

```
1 | .news {  
2 |   display: flex;  
3 | }
```

[Код примера](#)

ПОДДЕРЖКА ВСЕХ БРАУЗЕРОВ

Внешний вид верстки останется прежним.

Теперь в браузерах, поддерживающих flexbox, блоки будут выстроены с помощью этого свойства. Во всех прочих будет срабатывать свойство `float`:

One



Two



Three



НЕ НУЖНО ПЕРЕОПРЕДЕЛЯТЬ СВОЙСТВА

Одна из замечательных особенностей flexbox заключается в том, что некоторые текущие свойства, примененные к блоку выше, теряют свою силу.

В нашем случае `float` не применяется к блокам, которые обернуты во flex-контейнер – и это позволяет нам использовать такой прием для прогрессивного улучшения.

ИТОГИ

РЕЗИНОВАЯ ВЕРСТКА

- Резиновая верстка – это верстка, в которой размеры дочерних элементов подстраиваются под размеры родительских блоков. Простыми словами страница «тянется».
- При изменении ширины окна элементы остаются на месте.
- Основа резиновой верстки – относительные единицы измерения. Например, проценты.

РАЗМЕРЫ В ПРОЦЕНТАХ

- Ширина, заданная в процентах, высчитывается динамически и зависит от ширины родителя.
- Высота в процентах работает, если высота родительского блока не равна `auto` (значение по умолчанию).
- Лучше избегать фиксированной высоты. Контент блока с заданной высотой может наехать на следующий за ним элемент.
- Ширина блока с абсолютным позиционированием, заданная в процентах, высчитывается от ближайшего родителя с нестатичным позиционированием.
- Ширина в процентах у блока с фиксированным позиционированием высчитывается от ширины окна браузера.

ФИШКИ РЕЗИНОВОЙ ВЕРСКИ

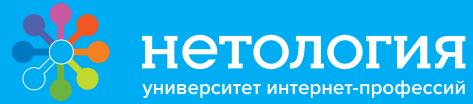
- Можно ограничить сужение элемента при помощи `min-width`.
- Растижение элемента можно ограничить при помощи `max-width`.
- Шрифт в процентах высчитывается от размера шрифта родителя.
- `margin` / `padding` в процентах всегда расчитывается от ширины элемента.

FLEXBOX

- Родительский элемент, которому задан `display: flex;`, называется flex-контейнер.
- Вложенные во flex-контейнер блоки называются flex-элементами.
- Как только родителю задан `display: flex;`, дочерние элементы тоже становятся флексами.
- Flex-элементы подстраиваются под высоту родителя, если она задана.
- Если родитель не имеет высоты, то и он и элементы подстроятся под самый высокий контент.

ОСИ

- Существует главная и дополнительная оси.
- По умолчанию flex-элементы встают в ряд в прямом порядке. Главная ось идет горизонтально.
- `flex-direction` позволяет поменять главную и дополнительные оси местами и сменить их направление.
- `row` (значение по умолчанию) выстраивает элементы в ряд. `column` – колонку.
- Добавляем слово `reverse` и меняем направление. Не слева направо, а справа налево.



Задавайте вопросы и напишите отзыв о лекции!

СЕМЕН БОЙКО



simonderus@gmail.com



fb.me/simonboycko