

Основы создания адаптивного сайта

Адаптивный сайт

Адаптивная вёрстка предполагает отсутствие горизонтальной полосы прокрутки и масштабируемых областей при просмотре на любом устройстве, читабельный текст и большие области для кликабельных элементов. С помощью медиазапросов можно управлять компоновкой и расположением блоков на странице, перестраивая шаблон таким образом, чтобы он адаптировался под разные размеры экранов устройств.

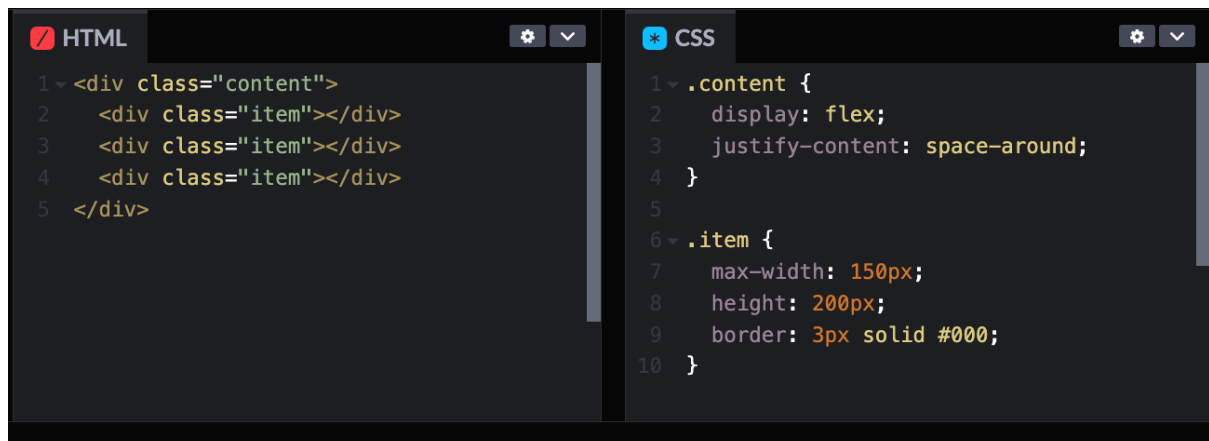
Адаптивные свойства блоков

Устанавливают максимальную ширину элемента. Значение ширины элемента будет вычисляться в зависимости от значений установленных свойств `width`, `max-width` и `min-width`.

`max-width`

Посмотрим, чем же отличается свойство `width`, которое мы использовали до этого урока, и в каких ситуациях его необходимо будет поменять на `max-width`. Если вы задали родительскому блоку значение `width`, при адаптиве у вас появится полоса прокрутки в нижней части браузера. Чтобы такого не произошло, вам нужно поменять это свойство ширины на `max-width`.

Одна из самых частых ошибок начинающих веб-разработчиков — вывод, что абсолютно все значения `width`: надо поменять на `max-width`. Это неверный вывод, так как новое значение хорошо подходит для родительского блока, но если вы его зададите дочерним элементам, когда у родителя задано свойство `display: flex`, то получится, что значение ширины дочерних элементов равно 0.



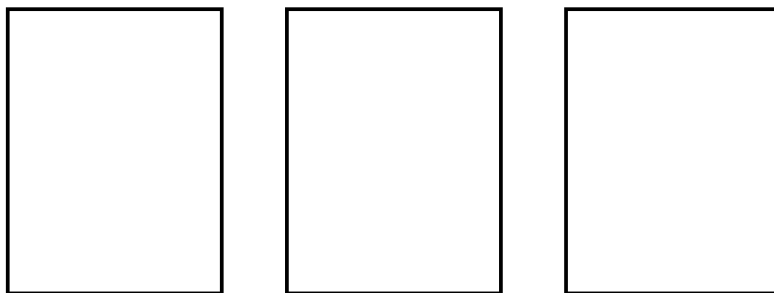
```
HTML
1 <div class="content">
2   <div class="item"></div>
3   <div class="item"></div>
4   <div class="item"></div>
5 </div>

CSS
1 .content {
2   display: flex;
3   justify-content: space-around;
4 }
5
6 .item {
7   max-width: 150px;
8   height: 200px;
9   border: 3px solid #000;
10 }
```

Это свойство в большинстве ситуаций относится к родительским блокам. Рассмотрите похожий пример:здесь значение выставлено для родительского блока content и уже работает отлично.

```
HTML
1 <div class="content">
2   <div class="item"></div>
3   <div class="item"></div>
4   <div class="item"></div>
5 </div>

CSS
1 .content {
2   display: flex;
3   flex-wrap: wrap;
4   justify-content: space-around;
5   max-width: 600px;
6   margin: 0 auto;
7 }
8
9 .item {
10  width: 150px;
11  height: 200px;
12  border: 3px solid #000;
13 }
```



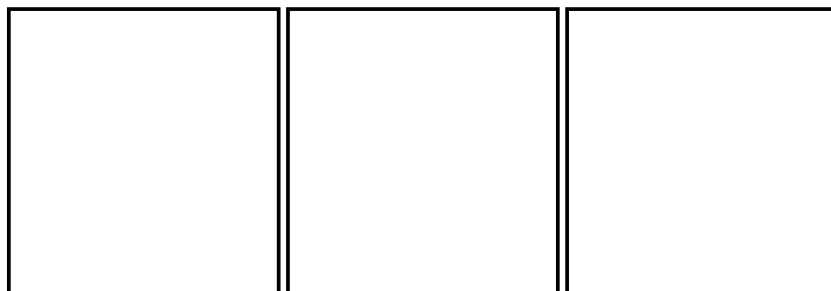
min-width

Если сравнивать два, казалось бы, похожих свойства `max` и `min width`, то можно заметить большую разницу в использовании: `max-width` встречается в большинстве проектов, а вот `min-width` встречается реже, может и вовсе отсутствовать. Свойство `min-width` используется, чтобы размеры контентной части не становились меньше, чем 320 px.

Пример: меньше чем 320px значение ширины `content` не будет, так как мы выставили `min-width: 320px` на 7 строчке `css`

```
HTML
1 <div class="content">
2   <div class="item"></div>
3   <div class="item"></div>
4   <div class="item"></div>
5 </div>

CSS
1 .content {
2   display: flex;
3   flex-wrap: wrap;
4   justify-content: space-around;
5   max-width: 600px;
6   margin: 0 auto;
7   min-width: 320px;
8 }
9
10 .item {
11   width: 190px;
12   height: 200px;
13   border: 3px solid #000;
14 }
```



Мы можем использовать это свойство как минимальный ограничитель, и даже если пользователь откроет сайт с устройства с меньшим разрешением экрана, контент будет располагаться верно, правда, с полосой прокрутки.

min-height

Это свойство очень популярно при создании адаптивного сайта, и его можно встретить практически в каждом проекте. Свойство min-height используется, если контента может стать больше или при адаптиве контент начнёт перестраиваться.

Пример. нашей задачей является выставление минимальное значение высоты для блока, но при уменьшении экрана количество контента становится больше, поэтому мы используем min-width: 320px на 7 строчке css и это и будет служить минимальным ограничителем.



**Lorem ipsum dolor sit amet, consectetur
adipiscing.**

max-height

Так же, как и в примере с min-width:, свойство max-height используется в разы реже, чем min-height:, так что его можно встретить далеко не в каждом адаптивном сайте.

Это свойство используется в ситуациях, когда мы хотим чтобы не занимал больше контента, чем у нас есть всего, допустим высота блока может быть от высоты буквы до 200px, получается что для него можно выставить max-height: 200px; которое и будет данным ограничителем.

Единицы измерения ширины и высоты устройства

Мы уже знаем такие единицы измерения как: px, %, em но как быть в ситуации, когда нам необходимо использовать высоту экрана устройства, с которого мы просматриваем контент, тут к нам на помощь приходят новые единицы измерения **vh**, **vw** (value height, value width)

Самый простой способ понять как работают новые единицы измерения, это представить что у нас есть проценты, только проценты работают от блоков где располагаются, а vh и vw это проценты от экрана.

Теперь давайте разберём каждое значение по отдельности

vh Относительная единица измерения — 1% от высоты области просмотра.

vw Относительная единица измерения — 1% от ширины области просмотра.

Эмуляция мобильных устройств

Чтобы посмотреть отображение сайта на планшете или мобильном телефоне, в отладчике браузера есть специальный функционал — эмуляция мобильных устройств.

Как посмотреть отображение сайта на планшете и мобильном телефоне

1. Открыть сайт в браузере.
2. Открыть отладчик браузера: F12, Ctrl + Shift + L, правой клавишей мыши — «Просмотр кода элемента».
3. Левый верхний угол отладчика (toggle device toolbar).
4. Вы сможете увидеть отображение вашего сайта на мобильном устройстве.

Если вы открыли мобильную версию сайта и замечаете, что сайт выглядит точно так же, как десктопная версия, значит, вы забыли добавить свойство viewport.

Viewport

Типичный сайт, оптимизированный для мобильных устройств, содержит следующий метатег:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Свойство width определяет размер окна просмотра. Он может быть установлен на определенное количество пикселей, скажем, width=600 или на специальное значение device-width, которое означает ширину экрана в пикселях CSS в масштабе 100%. Есть также соответствующие значения height и device-height, которые могут быть полезны для страниц с элементами, изменяющими размер или положение на основе высоты окна просмотра.

Сайты могут устанавливать свой viewport на определённый размер. Например, определение «width=320, initial-scale=1» может использоваться для точного размещения на маленьком дисплее телефона в портретном режиме. Это может вызвать проблемы, когда браузер не

отображает страницу большего размера. Чтобы исправить это, браузеры, если необходимо, увеличат ширину окна просмотра, чтобы заполнить экран по заданной шкале.

Для страниц, задающих начальный или максимальный масштаб, это значит, что свойство width фактически переводит в минимальную ширину viewport. Например, если ваш макет должен иметь ширину не менее 500 пикселей, вы можете использовать следующую разметку. При ширине экрана более 500 пикселей браузер будет расширять область просмотра (а не увеличивать), чтобы она соответствовала экрану.

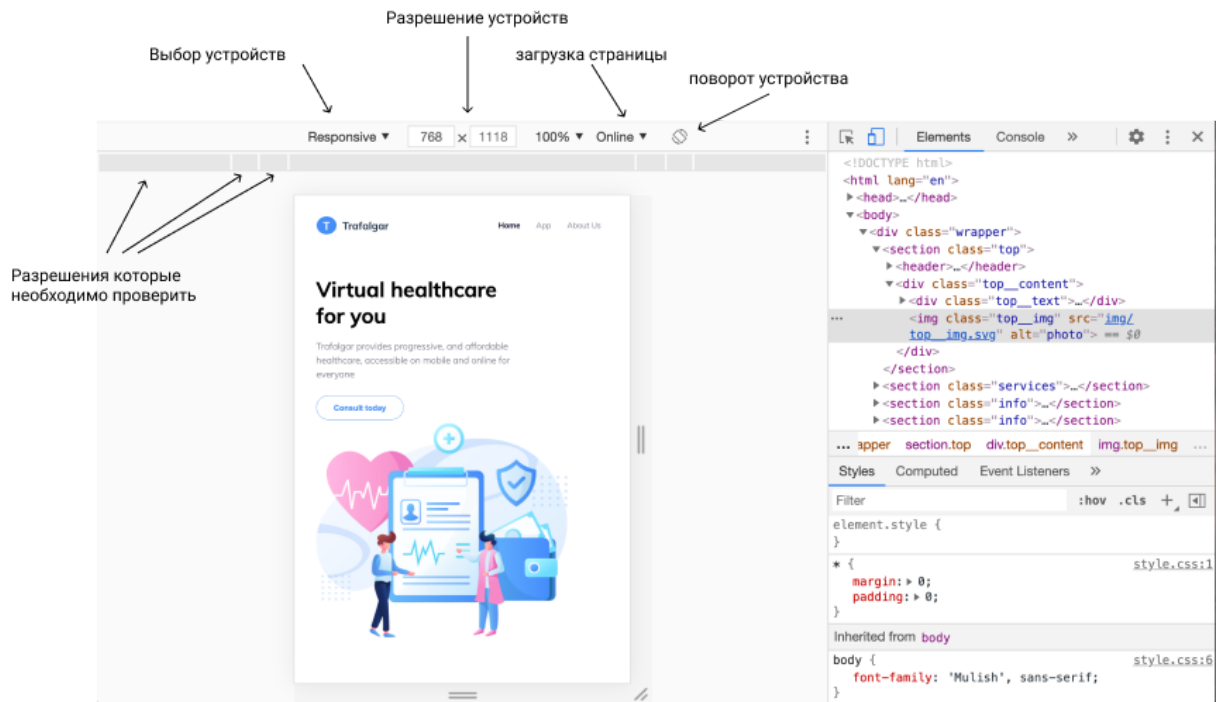
Другие доступные атрибуты — minimum-scale, maximum-scale, и user-scalable. Эти свойства влияют на начальный масштаб и ширину, а также ограничивают изменения уровня масштабирования.

Не все мобильные браузеры обрабатывают изменения ориентации таким же образом. Например, Mobile Safari часто просто увеличивает масштаб страницы при смене с вертикальной ориентации на горизонтальную, вместо того чтобы выкладывать страницу так, как если бы она была первоначально загружена в «ландшафт». Если веб-разработчики хотят, чтобы их настройки масштаба оставались неизменными при переключении ориентации на iPhone, они должны добавить значение maximum-scale, чтобы предотвратить масштабирование, которое иногда имеет нежелательный эффект, мешающий пользователям изменять масштаб:

```
<meta name="viewport" content="initial-scale=1, maximum-scale=1">
```

Toggle device toolbar

Рассмотрим функционал мобильной разработки в Google Chrome.



На иллюстрации мы можем увидеть, что у программы есть весь необходимый функционал без перегруженных возможностей. С ним должен справиться любой начинающий веб-разработчик.

Медиазапросы

Наряду с типами носителей в CSS3 включена поддержка различных технических параметров устройств, на основе которых требуется загружать те или иные стили. К примеру, можно определить смартфон с максимальным разрешением 640 пикселей и для него установить одни стилевые свойства, а для остальных устройств — другие. Также можно выявить различные характеристики вроде наличия монохромного экрана и ориентации (портретная или альбомная). Все характеристики легко комбинируются, поэтому допустимо задать стиль только для устройств в альбомной ориентации с заданным разрешением экрана.

Возможности медиазапросов не ограничиваются выявлением мобильных устройств, с их помощью можно создавать адаптивный макет. Такой макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину, число колонок, размеры изображений и текста. Медиазапросы ограничивают ширину макета, и при достижении этого значения, к примеру, за счёт уменьшения окна или при просмотре на устройстве с указанным размером, применяется другой стиль.

Синтаксис

Все запросы начинаются с правила `@media`, после чего следует условие, в котором используются типы носителей, логические операторы и медиафункции.

Плюсы

1. Чёткое отображение страниц на экране с любым разрешением.
2. Возможность просмотра группы контента на любом устройстве.
3. Отсутствие горизонтальной полосы прокрутки независимо от размера окна.

Условия для Media Queries

```
@media screen and (max-width: XXXpx) { }  
@media screen and (min-width: XXXpx) { }  
@media screen and (min-width: XXXpx) and (max-width: YYYpx) { }  
@media screen and (max-device-width: XXXpx) { }
```

С их помощью можно отслеживать разрешение экрана пользователя и отображать необходимые стили для каждого разрешения или устройства.

Пример:

```
@media screen and (min-width: 1024px) {  
  .content {  
    background-color: #ccc;  
  }  
}
```

В результате, если у пользователя экран больше или равен 1024px, задний фон для content будет серым.

Логические операторы

С помощью логических операторов можно создавать комбинированные медиазапросы, в которых будет проверяться соответствие нескольким условиям.

Оператор and

Оператор **and** связывает друг с другом разные условия:

```
@media screen and (min-width: 1024px) {  
  /* CSS-стили */  
}
```

Стили этого запроса будут применяться только для экранных устройств с шириной области просмотра не более 1024px.

```
@media (min-width: 320px) and (max-width: 768px) {  
  /* CSS-стили */  
}
```

Стили этого запроса будут применяться для всех устройств при ширине области просмотра от 320px до 768px включительно.

Оператор запятая

Оператор запятая работает по аналогии с логическим оператором **or**.

```
@media screen, projection {  
    /* CSS-стили */  
}
```

В этом случае CSS-стили, заключённые в фигурные скобки, работают только для экранных или проекционных устройств.

Оператор not

Оператор **not** позволяет сработать медиазапросу в противоположном случае. Ключевое слово **not** добавляется в начало медиазапроса и применяется ко всему запросу целиком, то есть запрос:

```
@media not all and (monochrome) {...}
```

будет эквивалентен запросу:

```
@media not (all and (monochrome)) {...}
```

Стратегии использования медиазапросов

Для создания дизайна, позволяющего лучшим образом отображать сайт на различных устройствах, используют общие стратегии медиазапросов:

1. Уменьшение количества колонок (столбцов) и постепенная отмена обтекания для мобильных устройств.
2. Использование свойства `max-width` вместо `width` при задании ширины блока-контейнера.
3. Уменьшение полей и отступов на мобильных устройствах, например нижних отступов между заголовком и текстом, левого отступа для списков и т. п.
4. Уменьшение размеров шрифтов для мобильных устройств.
5. Превращение линейных навигационных меню в раскрывающиеся.

6. Скрытие второстепенного содержимого на мобильных устройствах с помощью `display: none`.
7. Подключение фоновых изображений уменьшенных размеров.

В этом материале мы рассмотрели основы, которые помогут вам создать адаптивный сайт и решить проблемы, которые могут возникать по мере создания проекта. Также важная особенность — использование всех возможностей `flexbox`, которые мы изучали ранее.