

## МЕДИА-ЗАПРОСЫ.

*Понятие медиа-запроса. Типы устройств. Логические операторы. Медиа-функции.*

---

### Понятие медиа-запроса

Стандарт CSS3 принес в нашу жизнь и верстку множество интересных вещей и возможностей. Например, **медиа-запросы**.

Макет, построенный с использованием медиа-запросов, будет подстраиваться под размер экрана или другие его характеристики. Реализуется такое поведение с помощью разных стилей для одних и тех же элементов.

Резонно возникает вопрос, а как же браузер определит какой из запросов ему выполнить и какие из стилей применить? В состав медиа-запроса мы включим инструкцию – указание на тип устройства или даже его технические параметры.

Синтаксис запроса достаточно прост.

Все они начинаются с правила **@media**, затем следует **условие**, в котором используются типы носителей, логические операторы и медиа-функции. Если их больше одного, то они разделяются между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых пишутся **стилевые правила**.

```
@media <условие> {  
    описание стилей  
}
```

### Типы устройств

На практике, список типов устройств, которые нам позволяет указать медиа-запрос, не так уж и велик:

- значение по умолчанию для всех типов устройств – **all**;
- принтеры и печатающие устройства – **print**;
- экраны – **screen**;
- речевые синтезаторы любого толка, в том числе программы для воспроизведения текста – **speech**.

Так в следующем примере, для всех блоков `div` в документе при печати будет задан цвет фона `#ffccdd`.

```
@media print {  
  div { background-color: #ffccdd; }  
}
```

Некоторые из устройств признаны устаревшими, хотя и должны быть распознаны как допустимые:

- смартфоны и планшеты – **handheld**;
- проекторы – **projection**;
- устройства для чтения слепыми людьми – **braille**;
- принтеры, использующие для печать систему Брайля – **embossed**;
- телевизоры – **tv**;
- устройства с фиксированным размером символов (терминалы, телетайпы и пр.) – **tty**.

Вместо них рекомендуется задавать соответствующие **медиа-функции**, которые будут подробно описывать устройства (о них далее в лекции).

Вроде неплохо... Необходимо вывести печатный отчет с сайта – скрываем изображения, делаем крупнее текст, добавляем контрастности.

Если в качестве типа устройства выбран телевизор – загружаем на страницу полноразмерные фото с большим разрешением.

Но ведь экран экрану рознь...

И при создании адаптивного сайта необходимо учесть и описать поведение элементов как минимум для трех диапазонов разрешений... прибавьте к этому по два варианта ориентации для мобильных и планшетов.

При этом мы должны стремиться к оптимизации кода и не раздувать искусственно его количество.

А ведь может быть так, что страница на десктопе будет выглядеть точно также, как и страница на развернутом в альбомную ориентацию планшете?

Может и мы переходим ко второму элементу условий – логическим операторам.

## Логические операторы

Логические операторы, позволят нам создавать разные сочетания устройств или их параметров. Стил будет выполнен в том случае, если запрос возвращает истину, то есть, указанные условия выполняются.

Суть логических операторов осталась неизменной со времен знакомства с математической логикой. Оператор **and** объединяет условия. Стили будут применяться, если оба условия выполнены.

В примере ниже для всех блоков **div** с классом **container** при отображении на всех цветных устройствах будет задан цвет фона #ffccdd.

```
@media all and (color) {  
  div.container { background-color: #ffccdd; }  
}
```

Кстати, внимательный читатель заметит, что никаких **color** в лекции не было. Это как раз один из самых простых примеров медиа-функций.

Следующий оператор **not** – отрицает. Кстати, **not** можно заменить для себя фразой - для всех кроме.

Изменим ширину блока для всех устройств КРОМЕ принтеров.

```
@media all and (not (print)){  
    div.container { width: 100%; }  
}
```

Также при использовании операторов можно указывать скобки, чтобы менять приоритет операций.

Далее чуть-чуть сложнее. Оператор **not** оценивается в запросе последним, поэтому ... ширина блока изменится для устройств кроме всех цветных.

```
@media not all and (color) {  
    div.container { width: 100%; }  
}
```

А еще есть оператор для старых браузеров **only**. Старые браузеры считают это ключевое слово типом носителя, но поскольку такого типа не существует, то игнорируют весь стиль целиком.

В списке нет логического оператора **or** - его роль выполняет запятая. Перечисление нескольких условий через запятую говорит о том, что если хотя бы одно условие выполняется, то стиль будет применён.

Значит правило сработает для экранов или при печати страницы, и ширина блока изменится.

```
@media screen, print {  
    div.container { width: 100%; }  
}
```

## Медиа-функции

Медиа-функции задают технические характеристики устройства, на котором отображается документ.

Если вы создаёте медиа-выражение без указания значения, вложенные стили будут использоваться до тех пор, пока значение функции не равно нулю.

Если функция не применима к устройству, на котором работает браузер, выражения, включающие эту функцию, всегда ложны.

## Height | width

**Тип носителя:** print, screen.

**Значение:** размер.

Описывает высоту и ширину отображаемой области. Это может быть окно браузера или печатная страница.

```
/*Изменим цвет текста для отображаемой области шириной 360px*/  
  
@media (width: 360px) {  
  div { color: red; }  
}
```

Большинство функций могут содержать также приставку **min-** и **max-**, которая соответствуют минимальному и максимальному значению свойства.

Например, **max-width: 400px** означает, что указанные настройки оформления будут применяться, если ширина окна браузера меньше 400 пикселей, а **min-width: 1000px**, наоборот, говорит нам, что ширина окна должна быть от 1000 пикселей и выше.

Медиа-правило ниже говорит нам, что если высота отображаемой области будет от 25rem и более, то фон блоков div станет желтым.

```
@media (min-height: 25rem) {  
  div { background: yellow; }  
}
```

## Orientation

**Тип носителя:** print, screen.

**Значение:** landscape | portrait.

Определяет, что устройство находится в альбомном режиме (ширина больше высоты) или портретном (ширина меньше высоты).

Установим разное фоновое изображение для разной ориентации устройства.

```
@media screen and (orientation: landscape) {  
  #logo { background: url(bg-l.png) no-repeat; }  
}  
@media screen and (orientation: portrait) {  
  #logo { background: url(bg-p.png) no-repeat; }  
}
```

## Color

**Тип носителя:** print, screen.

Эти стили будут применяться к любому устройству с цветным экраном:

```
@media (color) {  
  body { background: #333333; }  
}
```

Например, стили, вложенные в следующий запрос, никогда не будут использоваться, потому что ни одно речевое устройство не имеет цвета:

```
@media speech and (color) {  
  body { background: #cccccc; }  
}
```

Если хотим уточнить параметры цветопередачи, то можно указать число бит на канал цвета. Например, 3 значит, что каналы могут отображать  $2^3$  цветов каждый, что в общем составляет 512 цветов.

```
@media screen and (color:3) {
```

```
body { background: #cccccc; }  
}
```

## Color-index

Указываем количество цветов, которое поддерживает устройство. В качестве значения – целое число. Тип носителя - **print** и **screen**.

Установим стиль для экранов, отображающих не меньше 256 цветов.

```
@media all and (min-color-index: 256) {  
  div.container { background-color: #ffccdd; }  
}
```

## Grid

Тип носителя: print, screen.

Значение: 0 | 1.

Определяет, что текущее устройство - с фиксированным размером символов. Размеры букв на таком устройстве занимают одинаковую ширину и высоту и выстраиваются по заданной сетке.

Например, терминалы и телефоны, которые поддерживают только один шрифт. Для таких устройств, если вам требуется отформатировать текст, не указывайте размер в пикселях, только в em.

Итак, если устройство отображения построено на основе сетки (например, текстовый терминал), функция определяется как 1 и для всех элементов с классом **text** будет определен цвет текста как черный. Если же устройство имеет растровый экран, то цвет текста станет голубым.

```
@media (grid: 0) {  
  .text { color: #6699cc; }  
}
```

```
@media (grid: 1) {  
  .text { color: #000000; }  
}
```

## Aspect-ratio

Тип носителя: print, screen.

Значение: целое число/целое число.

Определяет, соотношение ширины и высоты отображаемой области устройства.

*/\*соотношение сторон отображаемой области 1 к 1, т.е. ширина и высота равны\*/*

```
@media (aspect-ratio: 1/1) {  
  div { background: #f9a; }  
}
```

## Device-aspect-ratio

Тип носителя: print, screen.

Значение: целое число/целое число.

Определяет, соотношение сторон экрана устройства.

*/\*стиль для экранов с соотношением сторон 16:9 и более\*/*

```
@media screen and (min-device-aspect-ratio: 16/9) {  
  div.container { width: 100%; }  
}
```

**Важно!** Функции с приставкой **device-...** не рекомендуются к использованию в спецификации Media Queries Level 4. Не рекомендуются, но и не запрещены. Поэтому в лекции их оставим, но на заметку возьмем.

Конечно, мы отразили лишь некоторые из возможных медиа-функций. Какие-то из существующих достаточно специфичны,



например проверка предпочитает ли пользователь меньше движения на странице или какая цветовая схема у него в приоритете или как часто устройство вывода может изменять внешний вид контента. Некоторые из них поддерживаются не всеми браузерами. Если вас интересует эта тема, можете продолжить ее изучение самостоятельно.

## ЗАДАНИЕ 28 МЕДИА-ЗАПРОСЫ. PRINT

В практической работе имитируем страницу с информацией о Сотрудниках для сайта.

В примере их всего трое, но вам важно не количество. Наша страница будет перестраиваться при выводе на печать.

1. На основе сетки из предыдущего занятия сверстаем страницу.

```
<div class="container">
  <div class="row">
    <div class="cols col-12">
      <h1>Имитируем страницу сайта с информацией о
        сотрудниках</h1>
      <h4>При выводе страницы на печать изображения
        скрываем, цветовое оформление изменяем.</h4>
    </div>
  </div>
  <div class="row">
    <div class="cols col-12">Отдел продаж</div>
  </div>

  <div class="row">      </div>
  <div class="row">      </div>
  <div class="row">      </div>

</div>
```

2. Отредактируем стили.

```
*{
  margin:0;
  padding:0;
  box-sizing: border-box;
}

.container {width: 80%;
  margin: 0 auto;}

.container .cols {
  float: left;
  margin: 0 0 1em;
  padding: 0 1em;
  text-align: center;}
```

```

.container .cols.col-3 {width: 25%;}
.container .cols.col-9 {width: 75%;}
.container .cols.col-12 {width: 100%;}

.container:before,
.container:after,
.row:before,
.row:after,
.clear:before,
.clear:after
{
    content: " ";
    display: table;
}

.container:after,
.row:before,
.row:after,
.clear:after
{
    clear: both;
}

```

3. В каждой "строке" внутри контейнера разместим блок с фото и блок с информацией о сотруднике. Таких блоков может быть сколь угодно много (да и информация в них может выводиться, например, из базы). Повторите представленную структуру, изменяя контент.

```

<div class="cols col-3 foto">
    
</div>
<div class="cols col-9 text">
    <p>ФИО | <span>Рогов Глеб Анатольевич</span></p>
    <p>Должность | <span>Руководитель отдела</span></p>
    <p>Образование | <span>Высшее профессиональное</span></p>
    <p>Номер телефона | <span>+7 904 104 17 06</span></p>
    <p>Стаж работы | <span>7 лет 6 мес.</span></p>
</div>

```

4. Файл со стилями сначала дополним простым оформлением фото и текста.

```

.foto img{
    max-width: 100%;
    border-radius: 5%;
}

```

```

}

.text {
  min-height: 200px;
  height: auto;
  background-color: #333333;
  color: #ffffff;
}

.text p{
  padding: 1%;
  font-size: 1.2em;
  text-indent: 1;
}

.text span{ text-decoration: underline; }

```

5. А вот после займемся нашим первым медиа запросом. Он расскажет браузеру, что, при выводе страницы на печать, мы скроем блок с фото, текст разместим на всю ширину и заодно инвертируем цвета текста и фона.

```

@media print {

  .container .cols.col-3.foto {display: none;}

  .container .cols.col-9 {width: 100%;}

  .text {
    background-color: #ffffff;
    color: #333333;
  }
}

```

Отмена