

Цели занятия

- подробно разобрать что такое media-запросы в CSS и для чего они используются;
- изучить синтаксис и основные возможности media-запросов;
- поверхностно познакомиться с Grid Layout;

Полезные материалы

- [Документация по media-запросам](#)
- [Статья про использование media queries](#)
- [Как верстать с помощью CSS Grid? \(для продвинутых\)](#)

Что такое media-запросы?

Медиа-запросы (media queries) – это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, это конструкции, которые позволяют определять на основании некоторых условий какие стили необходимо использовать на веб-странице, а какие нет.

Медиа-запросы предназначены для создания адаптивных дизайнов. Адаптивный дизайн отличается от других тем, что он может «приспосабливаться» (видоизменяться) в зависимости от того, какую ширину экрана имеет устройство. Или, к примеру, изменять вёрстку сайта, когда пользователь переворачивает экран телефона в горизонтальный вид.

Но при создании адаптивных веб-страниц также необходимо обратить внимание на метатег viewport, который обеспечивает корректное отображение адаптивных дизайнов сайтов на экранах устройств, имеющих высокую плотность пикселей.

Правильно установленный метатег viewport:

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

Синтаксис

Создание медиа-запроса начинается с ключевого слова `@media` после которого указывается **одно или несколько условий**. В качестве условия можно указывать **тип устройства** или **требования к определённой характеристике**. Требование к определённой характеристике записывается **в круглых скобках**.

Комбинирование нескольких условий выполняется с помощью **логических операторов**.

После составления `@media`, стили, указанные в нём, будут применяться только в том случае, когда итоговый результат вычисления условий является истинной.

Пример медиа-запроса с одним условием:

```
/* стили будут применяться, когда условие истинно */  
@media screen {  
  .button {  
    background-color: coral;  
  }  
}
```

Пример медиа-запроса с комбинированием нескольких условий:

```
@media (min-width: 992px) and (max-width: 1199.98px) {  
  ...  
}
```

Типы устройств

В @media можно указывать **определённые типы устройств**:

- all – для всех;
- print – для принтеров и в режиме предварительного просмотра страницы перед печатью;
- screen – для устройств с экранами;
- speech – для программ чтения с экрана.

Например, этот @media только для экранов:

```
@media screen { ... }
```

А здесь для экранов и принтеров:

```
@media screen, print { ... }
```

Логические операторы

Логические операторы `and`, `,` (запятая), `not` предназначены для создания сложных медиа-запросов. Это относится уже к продвинутому уровню, поэтому в рамках данного курса не будет задач, где это может пригодиться, но будет полезно для общего понимания.

and

Оператор `and` используется для объединения нескольких условий. В этом случае их результат будет истинным, когда каждое из них будет истинным.

Например, следующий @media будет применяться только при выполнении всех трёх условий (это экран, width >= 1200px и ориентация landscape):

```
@media screen and (min-width: 1200px) and (orientation: landscape) { ... }
```

, (запятая)

Применение стилей, когда необходимо лишь выполнение одного из указанных условий, достигается посредством разделения их между собой с помощью , (запятой).

В этом примере стили будут применяться к странице в двух случаях. Когда **width >= 544px** или **ориентация landscape**.

```
@media (min-width: 544px), (orientation: landscape) { ... }
```

not

Ключевое слово `not` используется для отрицания.

При использовании `not` с `and` отрицание работает для всего медиа-запроса. При этом, когда указываем `not` необходимо обязательно задавать тип устройства.

Например, применим стили только в том случае, когда **не (экран и width >= 411px)**.

```
@media not screen and (min-width: 411px) { ... }
```

Медиа-характеристики

При составлении **условия** кроме типов устройств и логических операторов можно ещё задавать **требования к определённым характеристикам**, которые должен иметь браузер, устройство вывода, или окружение.

Каждая характеристика в `@media` должна быть заключена в круглые скобки.

При этом применяться стили указанные в `@media` будут также как раньше, т.е. только в том случае, когда результат вычисления всего выражения будет являться истиной.

Чаще всего адаптивный дизайн подстраивается под ширину экрана пользователя, поэтому мы разберём именно этот вариант.

Например, нам нужно задать стили, которые будут применяться только тогда, когда ширина экрана равна или меньше чем 1024px:

```
/* Этот стиль применяется по умолчанию */  
.button {  
  background-color: red;  
}  
  
/* А этот только когда ширина <= 1024px */  
@media (max-width: 1024px) {  
  .button {  
    background-color: green;  
  }  
}
```

Ровно с таким же принципом можно задавать условия для любой другой ширины, например, можно написать правило, которое будет применяться только в определенном диапазоне ширины:

```
/* Этот стиль применяется по умолчанию */  
.button {  
  background-color: red;  
}  
  
/* А этот только когда ширина от 576px до 1200px */  
@media (min-width: 576px) and (max-width: 1199px) {  
  .button {  
    background-color: green;  
  }  
}
```

Что такое Grid Layout?

CSS Grid Layout — это двумерная сетка для размещения элементов страницы. Верстая с данной технологией, надо думать столбцами и рядами, как в таблице. По сути, Grid CSS — это набор свойств, при помощи которых можно расположить элементы в нужном порядке, в нужном размере ячейках и задать расстояния между ними.

В заранее определенной гибкой сетке макета мы можем размещать дочерние элементы. В этом плане он немного напоминает Flex Layout, с которым мы уже знакомы — мы управляем элементами контейнера и их размерами, задавая необходимые свойства самому контейнеру.

Изучение Grid довольно обширная тема и зачастую относится к более продвинутому уровню CSS, поэтому в рамках данного курса будет рассматриваться поверхностно и в домашних заданиях не будет задач, которые потребовали бы использовать Grid. Но самостоятельное изучение можно начать с некоторых вводных статей, например [этой](#) или [этой](#).