4.2. Медиавыражения

~ 4 минуты

Медиавыражения

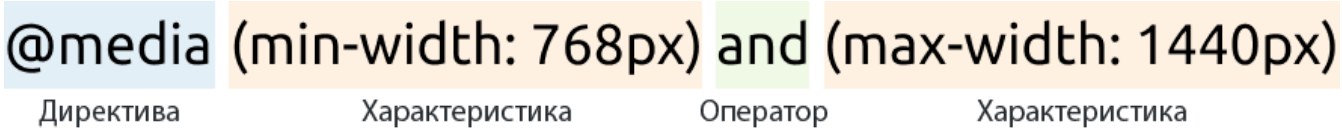
Медиавыражения — это способ определить в стилях характеристики браузера или предпочтения пользователя, а затем применить стили или запустить другой код на основе этих параметров. Наиболее распространенными медиавыражениями являются те, которые следят за размером области просмотра и применяют пользовательские стили к разрешению экрана. Именно они и породили идею адаптивного дизайна.

```
@media (min-width: 768px){
  .block {
    /* стили для разрешения от 768px */
  }
}
```

Помимо ширины экрана, есть много других вещей, на которые мы можем ориентироваться. Это может быть разрешение экрана, ориентация устройства, настройки операционной системы и многое другое.

Общий синтаксис

Давайте рассмотрим анатомию медиавыражения, а после пройдёмся подробнее по каждой его части.



Чтобы медиавыражение заработало, нужно добавить директиву `@media`.

Директивы начинаются с символа `@`. С некоторыми из них вы уже знакомы, например, с `@font-face` и `@keyframes`.

Характеристики

Современные устройства имеют множество характеристик. Рассмотрим самые полезные из них.

`resolution`

Определяет плотность экрана. Используется с префиксом `min-`, чтобы определить любые экраны с повышенной плотностью выше указанной.

```
@media (min-resolution: 2dppx) {
    .block {
        background-image: url("image@2x.png");
    }
}
```

Это медиавыражение срабатывает для экранов с повышенной плотностью 2 и больше.

`width`

Медиавыражение с `width` используется крайне редко, так как оно срабатывает только для размера вьюпорта, который равен указанной в выражении ширине. Например, такое медиавыражение сработает только для размера вьюпорта 1024px по ширине:

```
@media (width: 1024px){
  .block {

  }
}
```

Чаще всего вы можете встретить медиавыражение с параметром `min-width` и `max-width`.

```
@media (min-width: 1024px){
  .block {

  }
}

@media (max-width: 1024px){
  .block {
```

```
  }
}
```

CSS-правила начинают работать от указанного размера:

— `min-width` — от указанного значения и выше, включая значение.

— `max-width` — от указанного значения и ниже, включая значение.

`height`

Как и `width`, `height` никогда не используют без префиксов, так как интерфейс будет меняться только на конкретном значении высоты.

Значения `min-height` и `max-height` используются в тех случаях, когда адаптируют плотность или разрежённость интерфейса для экранов разной высоты — например, для интерфейса почты.

Синтаксис работы точно такой же, как и с шириной.

```
@media (min-height: 1024px){
  .block {

  }
}

@media (max-height: 1024px){
  .block {

  }
}
```

Операторы

Как и многие языки программирования, медиавыражения поддерживают логические операторы, поэтому мы можем комбинировать выражения. Условиями выступают операторы, с помощью которых мы можем объединять функции или давать несколько факторов срабатывания изменения интерфейса.

`and`

`and` — оператор «и», который объединяет выражения так, что они должны выполняться одновременно. Если какое-либо выражение не срабатывает, то медиавыражение не применяется.

Мы можем указать условие в определённом промежутке экрана:

```
@media screen (min-width: 320px) and (max-width: 768px) {
  .block {

  }
}
```

`.block` изменится, если размер вьюпорта от 320px до 768px.

or

`or` — оператор «или», который делает часть выражения необязательными. Из всех указанных выражений может подойти только одно, и этого будет достаточно, чтобы медиавыражение сработало.

Вместо оператора `or` мы также можем разделить функции запятыми:

```
@media screen (min-resolution: 2dppx), (min-width: 2560px) {
  .block {

  }
}
```

Медиавыражение сработает, если плотность экрана не меньше двух или ширина окна не меньше 2560px.

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

- [ ] Изучите регламент
- [ ] Прочитайте FAQ
- [ ] Добавьте свой Гитхаб
- [ ] Выберите наставника
- [ ] Создайте проект

Мой наставник

Выбрать наставника

Работа с наставником

У вас осталось **10** из 10 консультаций.

История

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Услуги

Работа наставником

Для учителей

Стать автором

Информация

Об Академии

О центре карьеры

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

Sk

Участник