Не всегда можно решить проблемы, сделав элементы резиновыми. Иногда приходится прописывать стили для устройства с конкретным типом или характеристиками.

Директива ***@media*** позволяет задавать разные стили для разных параметров экрана разных устройств. *CSS* переменные не поддерживаются в качестве значений медиазапросов.

Применять медиазапросы можно в разных местах.

*// в html*

*<link rel="stylesheet" media="(resolution >= 2x)" href="retina.css">*

*<style media="(resolution >= 2x)"></style>*

*// в директиве @import*

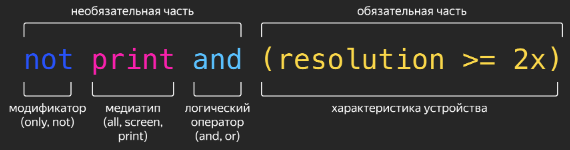
*import "retina.css" (resolution >= 2x);*

*// в css*

*@media (resolution >= 2x) {*

*/\* Стили для ретины \*/*

*}*



Медиазапрос может содержать медиатип, модификатор, характеристику устройства и логический оператор.

**Медиатип** указывают, чтобы применить стиль на определенном устройстве:

* ***all*** – на любом устройстве (по умолчанию)
* ***screen*** – на устройстве с экранами
* ***print*** – на печатающих устройствах и документах в режиме предварительного просмотра

*@media screen { // только на screen устройствах*

*body {*

*font-family: 'Open Sans', sans-serif;*

*}*

*}*

**Модификаторы** бывают двух типов:

* ***not*** – инвертирует условие
* ***only*** – говорит старым браузерам игнорировать медиазапрос.

*@media not screen { // на всех устройствах, кроме screen*

*body {*

*background-color: red;*

*}*

*}*

**Логические операторы** ***and*** и ***or*** позволяют комбинировать условия медиазапросов.

*@media (min-width: 768px) and (orientation: landscape) { // можно заменить вложенными запросами*

*body {*

*background-color: #000;*

*}*

*}*

*@media (min-width: 768px) or (orientation: landscape) { // можно заменить запятой*

*body {*

*background-color: #000;*

*}*

*}*

**Характеристики устройства**:

* ***width*** и ***height*** – ширина/высота окна просмотра
* ***max-width*** и ***min-width*** – максимальная/минимальная ширина окна просмотра
* ***max-height*** и ***min-height*** – максимальная/минимальная высота окна просмотра
* ***orientation –*** ориентация окна просмотра может быть альбомной ***landscape*** и портретной ***portrait***
* ***resolution*** – разрешение устройства (плотность пикселей в формате ***dppx***, ***dpi***, ***dpcm***)
* ***max- resolution*** и ***min- resolution*** – максимальное/минимальное разрешение устройства

*@media not screen, (max-width: 768px) { //* отключить фоновое видео для слабых устройств

*.page\_\_video,*

*.page::before {*

*display: none;*

*}*

*}*

*@media (orientation: landscape) { // при альбомной ориентации будет белый фон*

*body {*

*background-color: #ffffff;*

*}*

*}*

*@media (resolution: 2x) { // лучшее качество для высокого разрешения*

*.photo {*

*background-image: url(./images/photo\_2x.jpg);*

*}*

*}*

Для свойств связанных с характеристиками устройств можно использовать синтаксис диапазонов, основанный на математических операторах сравнения: <, >, <=, >=, =.

*// стили блока при ширине окна просмотра >= 1200px и <= 1400px*

*@media (min-width: 1200px) and (max-width: 1400px) { // запись через логические операторы*

*.block { }*

*}*

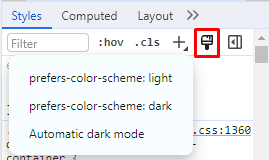
*@media (1200px <= width < 1400px) { // запись через операторы*

*.block { }*

*}*

Медиазапросы ***prefers-\**** определяют пользовательские предпочтения.

Медиазапрос ***prefers-color-scheme*** задает собственный вид страницы в зависимости от предпочитаемой цветовой схемы пользователя. На текущий момент поддерживаются значения ***light*** и ***dark***. Для тестирования цветовой схемы её можно задать в *DevTools* браузера через кисть.



Рассмотрим пример медиазапроса, позволяющего интерфейсу реагировать на предпочитаемую тему пользователя.

*@media (prefers-color-scheme: dark) {*

*body {*

*--main-text-color: #ffffff;*

*--main-background-color: #152028;*

*}*

*}*

*@media (prefers-color-scheme: light) {*

*body {*

*--main-text-color: #152028;*

*--main-background-color: #ffffff;*

*}*

*}*

Свойство ***color-scheme*** автоматически подбирает значения дефолтных стилей браузера, которые не были переопределены. Возможные значения:

* ***normal*** – элемент не имеет тем, поэтому используется цветовая тема браузера по умолчанию
* ***dark*** – есть поддержка темной темы
* ***light*** – есть поддержка светлой темы
* ***only*** – запрещает переопределять цветовую схему. Используется вместе с ***light*** или ***dark***

Если указаны несколько значений, то первым идет то, которое рекомендует разработчик.

Применять это свойство можно в разных местах.

*// в html*

*<meta name="color-scheme" content="dark light" />*

*// в css на корневом уровне*

*:root {*

*color-scheme: dark;*

*}*

*// в css на уровне элемента*

*.block {*

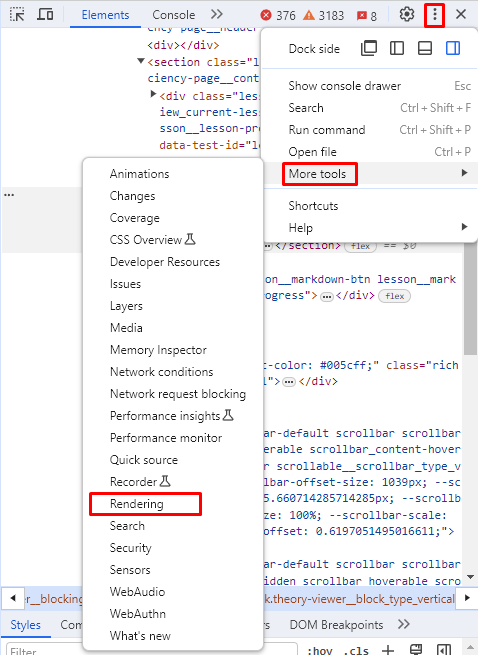
*color-scheme: normal;*

*}*

Медиазапрос ***prefers-reduced-motion*** уменьшает количество анимаций. Он позволяет устранить резкий переход (мигание) цвета. Может принимать одно из значений:

* ***no-preference*** – без предпочтений (по умолчанию)
* ***reduce*** – интерфейс с минимальной анимацией

Для тестирования цветовой схемы её можно задать в *DevTools* браузера *More tools -> Rendering*.



*@media (prefers-reduced-motion) {*

*button {*

*transition: .5s; // меняем плавность перехода*

*}*

*button:hover {*

*background-color: tomato;*

*color: white;*

*}*

*}*

Рассмотрим еще один пример.

*.button {*

*/\* стили кнопки \*/*

*}*

*@media (prefers-reduced-motion: no-preference) { // показывать анимации если юзер не запретил*

*.button {*

*// имя анимации, длительность, плавность, количество повторений, момент применения стилей*

*animation: shaking .5s linear infinite both;*

*}*

*}*

*@keyframes shaking { // определяем ключевые кадры и задаем имя анимации*

*0% { transform: translateX(0) }*

*25% { transform: translateX(5px) }*

*50% { transform: translateX(-5px) }*

*75% { transform: translateX(5px) }*

*100% { transform: translateX(0) }*

*}*

Медиазапрос ***prefers-contrast*** изменяет стилизацию в зависимости от желаемого уровня контрастности. Может принимать одно из значений:

* ***no-preference*** – без предпочтений (по умолчанию)
* ***more*** – интерфейс с высоким уровнем контрастности
* ***less*** – интерфейс с низким уровнем контрастности
* ***custom*** – режим принудительных цветов

Для тестирования контрастности её можно задать в *DevTools* браузера *More tools -> Rendering*.

*@media (prefers-contrast: more) { // меняем цвета для высокой контрастности*

*.button {*

*background-color: black;*

*}*

*.button:hover {*

*background-color: white;*

*color: black;*

*border: 2px solid black;*

*}*

*}*

Медиазапрос ***hover*** определяет, позволяет ли устройство наводить указатель на элементы. Может принимать одно из значений:

* ***none*** – не поддерживает
* ***hover*** – поддерживает

*@media (hover: hover) { // медиазапрос только для устройств с поддержкой наведения*

*.button:hover {*

*box-shadow: inset 0px 0px 0 100px rgb(0 0 0 / 15%);*

*cursor: pointer;*

*}*

*}*

Медиазапрос ***pointer*** определяет наличие и точность указывающего устройства. Может принимать одно из значений:

* ***none*** – не является указывающим устройством
* ***course*** – грубая точность (сенсорный экран, джойстик и т.п.)
* ***fine*** – хорошая точность (мышь, стилус)

*@media (pointer: course) { // медиазапрос для грубой точности*

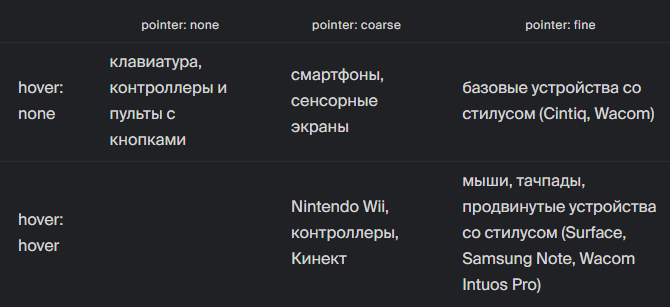
*.popup\_\_close {*

*inline-size: 44px;*

*block-size: 44px;*

*}*

*}*



Директива ***@container*** дает понять дочерним элементам, где они находятся. При запросе размеров можно использовать как физической модели, так и логической:

* *block-size = height*
* *max-block-size/min-block-size = max-height/min-height*
* *inline-size = width*
* *max-inline-size/min-inline-size = max-width/min-width*

Для корректной работы директивы нужно объявить родительский компонент стилизуемого элемента как контейнер с помощью свойства ***container-type***. При этом дочерний элемент перестает влиять на элементы за пределами своего контейнера. Может принимать одно из значений:

* ***normal*** – создает контейнер, который не позволяет запрашивать размеры, но разрешает запрос стилей (значение по умолчанию)
* ***inline-size*** – создает контейнер, который позволяет запрашивать размер по строчному направлению оси
* ***size*** – создает контейнер, который позволяет запрашивать размер по любому направлению оси

*// включаем контейнер*

*.block {*

*container-type: size;*

*}*

*// задаем поведение флекс-колонки при ширине блока не более 850px*

*@container (inline-size <= 850px) {*

*.card {*

*flex-direction: column;*

*}*

*}*

Единицы измерений контейнеров:

* ***сqw*** – 1% от ширины контейнера
* ***сqh*** – 1% от высоты контейнера
* ***сqi*** – 1% от *inline-size* контейнера
* ***сqb*** – 1% от *block-size* контейнера
* ***cqmin*** – меньшее из *cqi* и *cqb*
* ***cqmax*** – большее из *cqi* и *cqb*

Для запроса стилей используется функция ***style()***. Например, можно изменять цвет карточки в зависимости от фона контейнера:

*.sidebar {*

*container-type: inline-size;*

*--bg-color: #ffc802;*

*background-color: var(--bg-color);*

*}*

*@container style(--bg-color: #ffc802) {*

*.card\_\_content {*

*background-color: var(--bg-color);*

*}*

*}*

Запросы размеров и стилей можно комбинировать с помощью логических операторов.

Контейнерам можно присваивать имена с помощью свойства ***container-name***, а затем обращаться к ним по имени.

*.cards\_\_item {*

*container-type: inline-size;*

*container-name: cards-item;*

*background-color: #000;*

*}*

*.cards {*

*container: cards / inline-size; // сокращенная запись двух свойств*

*}*

*@container cards-item (inline-size >= 300px) {*

*.button {*

*background-color: #fff;*

*}*

*}*

**При резиновой верстке** элементы растягиваются: пропорционально масштабируются под размеры окна, но не меняют свое положение и внешний вид. Размеры задаются в относительные единицы измерения *%*, *vw*, *vh*.

**При адаптивной верстке** внешний вид сайта меняется, а элементы перестраиваются в зависимости от размера браузера. Пишутся *CSS*-правила под разные условия (размеры окна).

**При отзывчивой верстке** совмещается резиновая и адаптивная верстка. Благодаря резиновой верстке переход из одного состояния в другое происходит плавно, а благодаря адаптивной верстке в определенных точках перестраивается внешний вид.

При любом из подходов для сайта верстают несколько состояний: для телефона, планшета, ноутбука, десктопа. Используются подходы ***desktop first*** и ***mobile first***.

**Брейкпоинт** – это точка, в которой меняется интерфейс. Чаще всего для задания брейкпоинта используют пиксели, но также можно использовать относительные единицы *em* и *rem*.

При верстке ***desktop first*** нужно написать стили для самого большого разрешения, а затем уточнить их для меньших разрешений. В медиазапросах это выглядит следующим образом:

*.block {*

*/\* Базовые стили блока \*/*

*}*

*@media (max-width: 1400px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 1400px \*/*

*}*

*}*

*@media (max-width: 1200px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 1200px \*/*

*}*

*}*

*@media (max-width: 992px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 992px \*/*

*}*

*}*

*@media (max-width: 768px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 768px \*/*

*}*

*}*

*@media (max-width: 576px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 576px \*/*

*}*

*}*

Из минусов, можно столкнуться с частым сбросом стилей. Например, карточки для десктопа должны выстраиваться в линию по три штуки, а на мобильном располагаться в колонку друг над другом. Верстка ***desktop first*** может выглядеть так:

*.cards {*

*display: flex;*

*flex-wrap: wrap;*

*padding: 25px;*

*}*

*@media (max-width: 768px) {*

*.cards {*

*display: block;*

*}*

*}*

При верстке ***mobile first*** нужно написать стили для самого маленького разрешения, а затем уточнить их для больших разрешений. В медиазапросах это выглядит следующим образом:

*.block {*

*/\* Базовые стили блока \*/*

*}*

*@media (max-width: 576px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 576px \*/*

*}*

*}*

*@media (max-width: 768px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 768px \*/*

*}*

*}*

*@media (max-width: 992px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 992px \*/*

*}*

*}*

*@media (max-width: 1200px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 1200px \*/*

*}*

*}*

*@media (max-width: 1400px) {*

*.block {*

*/\* Новые стили при ширине окна просмотра <= 1400px \*/*

*}*

*}*

Из минусов, можно столкнуться с отсутствием некоторых элементов или отступов для мобильного. Вернемся к примеру с карточками при верстке ***mobile first***:

*.cards {*

*padding: 25px;*

*}*

*@media (min-width: 768px) {*

*.cards {*

*display: flex;*

*flex-wrap: wrap;*

*}*

*}*

Третий подход называется **верстка диапазонами**. Его используют очень редко из-за обилия кода. Он предполагает верстку каждого элемента под каждый диапазон.

*.about-me\_\_text {*

*/\* Базовые стили блока \*/*

*color: #000000;*

*margin-top: 20px;*

*margin-bottom: 0;*

*}*

*@media (width <= 768px) {*

*.about-me\_\_text {*

*/\* Стили блока при ширине окна просмотра <= 768px \*/*

*font-size: 11px;*

*line-height: 16px;*

*}*

*}*

*@media (768px < width < 1200px) {*

*.about-me\_\_text {*

*/\* Стили блока при ширине окна просмотра >= 768 и <= 1200px \*/*

*font-size: 12px;*

*line-height: 18px;*

*}*

*}*

*@media (width >= 1200px) {*

*.about-me\_\_text {*

*/\* Стили блока при ширине окна просмотра >= 1200px \*/*

*font-size: 14px;*

*line-height: 22px;*

*}*

*}*