

АДАПТИВНАЯ ТИПОГРАФИКА



МИХАИЛ ЛАРЧЕНКО / SYTAC B.V.



МИХАИЛ ЛАРЧЕНКО

Tech Lead в Sytac B.V.



larchanka@me.com



[+31621967276](tel:+31621967276)

ПЛАН ЗАНЯТИЯ

1. Медиафункции `aspect-ratio` и `orientation`
2. Особенности HTML-элементов
3. Единицы измерения `rem`
4. Итоги

МЕДИАФУНКЦИИ

aspect-ratio И

orientation

РАЗНЫЕ СТИЛИ НА ВСЕ СЛУЧАИ

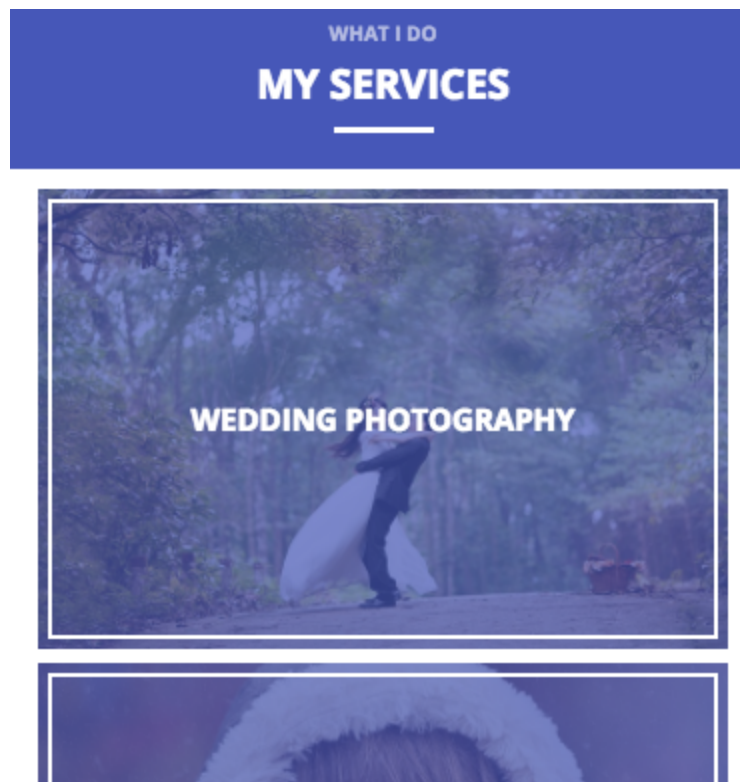
Дизайн страницы для мобильного телефона в вертикальной ориентации не всегда подходит для случая, когда пользователь повернул устройство.

Поэтому часто при создании адаптивной верстки возникает необходимость подключать разные стили исходя из пропорций экрана.



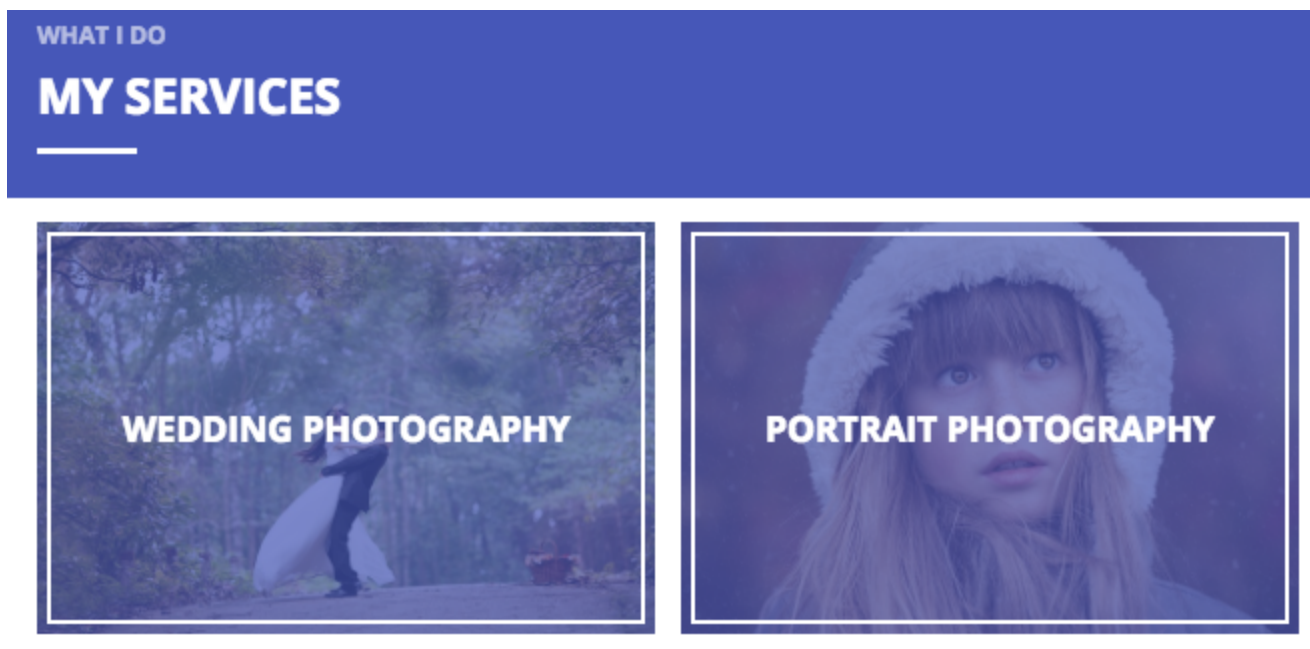
МАКЕТ ДЛЯ ВЕРТИКАЛЬНОЙ ОРИЕНТАЦИИ

Согласно макету элементы блока услуг должны располагаться друг под другом на мобильных телефонах — но только в вертикальной (*портретной*) ориентации.



МАКЕТ ДЛЯ ОСТАЛЬНЫХ СЦЕНАРИЕВ

А во всех остальных случаях, в том числе и на мобильных, блоки должны выстраиваться по два в строке:



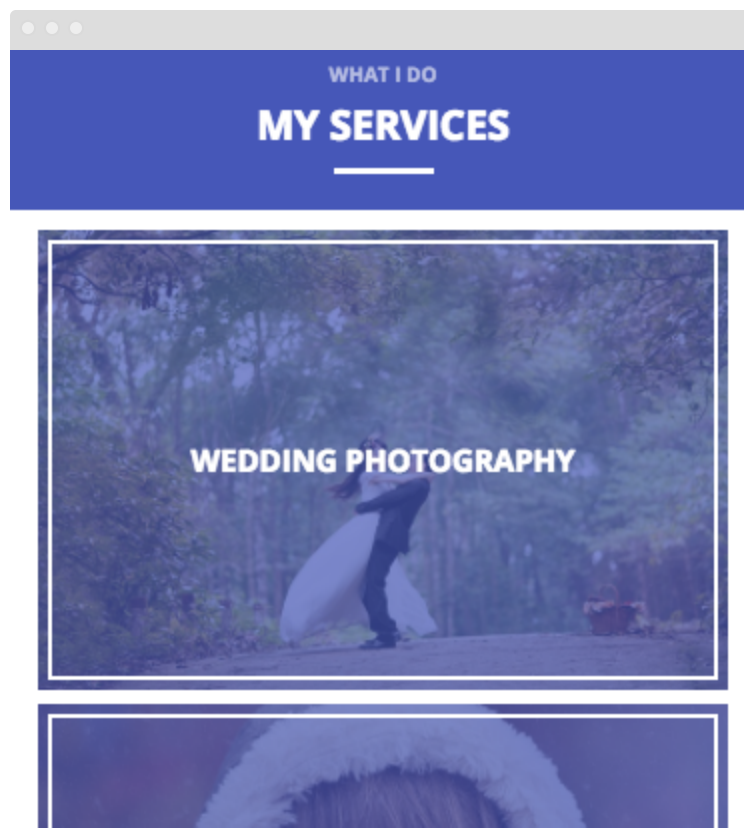
ОСНОВНЫЕ СТИЛИ

Запишем стили для отображения элементов по два в строку для всех устройств, кроме мобильных телефонов:

```
1  @media (min-width: 641px) {  
2      .section__body {  
3          display: flex;  
4          flex-wrap: wrap;  
5      }  
6  
7      .post {  
8          width: 49%;  
9      }  
10 }
```

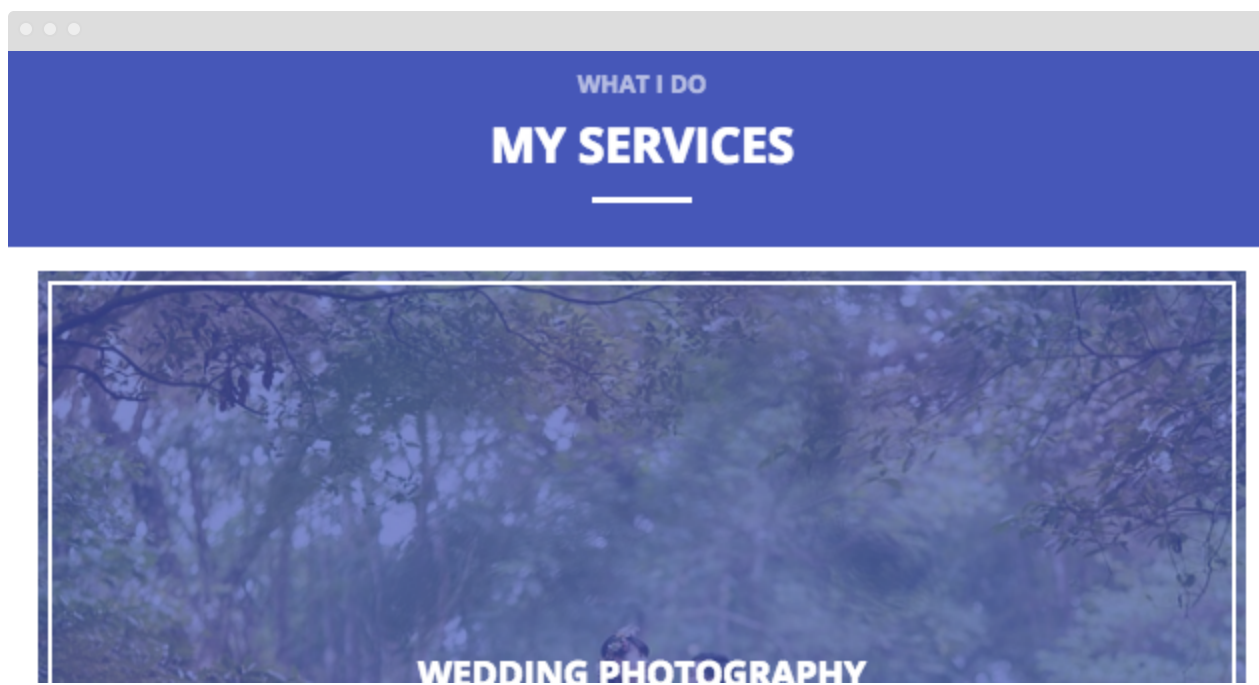

ВЕРСТКА СООТВЕТСТВУЕТ МАКЕТУ

На мобильном телефоне в вертикальной ориентации блок соответствует макету:



ГОРИЗОНТАЛЬНАЯ ОРИЕНТАЦИЯ

Но если мы повернем экран, то увидим следующее:



МЕДИАФУНКЦИИ ОПРЕДЕЛЕНИЯ ОРИЕНТАЦИИ

Нужно сделать вывод блоков по два в строку, если пользователь держит телефон горизонтально. Найти решение, подбирая ширину в медиазапросе `min-width`, нельзя, потому что информации только о ширине устройства недостаточно, чтобы понять, в какой ориентации оно находится.

Для определения ориентации устройства существуют специальные медиафункции:

- `aspect-ratio`;
- `device-aspect-ratio`;
- `orientation`.

aspect-ratio

Медиафункция `aspect-ratio` определяет соотношение ширины к высоте отображаемой области устройства.

Коэффициент записывается в виде дроби, состоящей из двух положительных целых ненулевых чисел — `width` и `height`, отображаемой области устройства разделенных знаком `/`.

ПРИМЕР РАСЧЕТА `aspect-ratio`

Например, разрешение 15" Macbook Pro – 2880*1800. Наибольший общий делитель двух этих чисел – это 360, и разделив оба числа на 360, мы получаем `aspect-ratio: 8/5`.

МИНИМУМ / МАКСИМУМ

Точно так же, как медиафункции `width` и `height`, `aspect-ratio` может использоваться с приставками `min-` или `max-`:

```
1  @media (min-aspect-ratio: 8/5) {  
2      /* css */  
3  }
```

device-aspect-ratio

Эта медиафункция определяет соотношение ширины к высоте экрана устройства и записывается как отношение значения `device-width` к `device-height`, разделенных знаком `/`.

В настоящее время является устаревшей.

orientation

Эта медиафункция определяет, в каком положении — портретном или ландшафтном — находится устройство.

Медиафункция `orientation` может принимать одно из двух возможных значений: `portrait`, когда ширина отображаемой области устройства больше или равна высоте, и `landscape` в других случаях.

РАЗНЫЙ ФОН

Используя эту медиафункцию, мы можем, например, задать разные значения свойства `background-image` блока в зависимости от ориентации устройства:

```
1  @media (orientation: landscape) {  
2    .hero {  
3      background-image: url(cat-landscape-mode.jpg);  
4    }  
5  }  
6  
7  @media (orientation: portrait) {  
8    .hero {  
9      background-image: url(cat-portrait-mode.jpg);  
10   }  
11 }
```

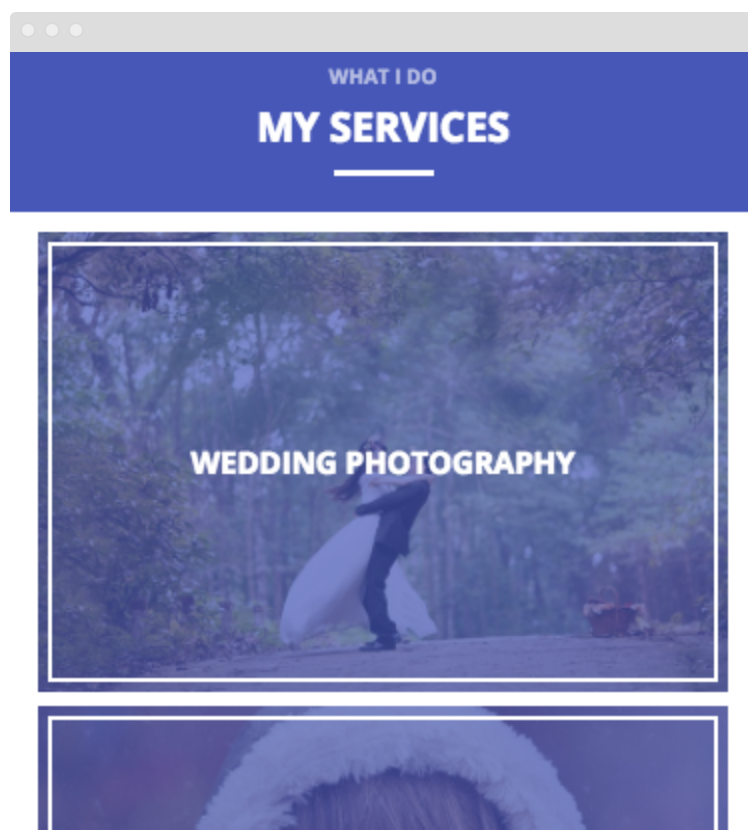
ПЕРЕПИШЕМ КОД

Возвращаясь к примеру блоков услуг, используем медиафункцию `orientation` для решения задачи отображения блоков по 2 в строку на телефонах в горизонтальной ориентации:

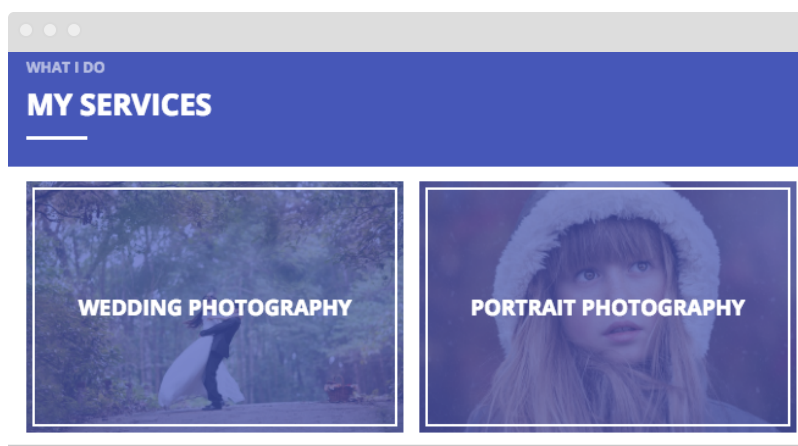
```
1  @media (min-width: 641px), (orientation: landscape)
2  and (max-width: 640px) {
3      .post {
4          width: 49%;
5      }
6  }
```

РАБОТАЕТ КАК НАДО

Блок на телефоне в вертикальной (портретной) ориентации



Блок на телефоне в горизонтальной (ландшафтной) ориентации



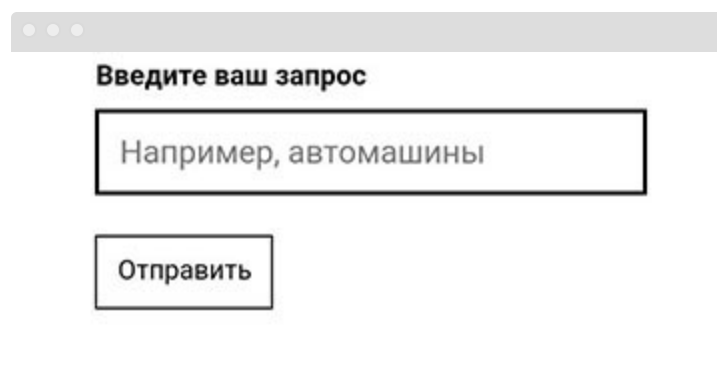
[Live Demo](#)



ОСОБЕННОСТИ HTML-ЭЛЕМЕНТОВ

ВЕРСТАЕМ ФОРМУ ПОИСКА

Сверстаем форму поиска, которой могут пользоваться со смартфона или планшета.



Введите ваш запрос

Отправить

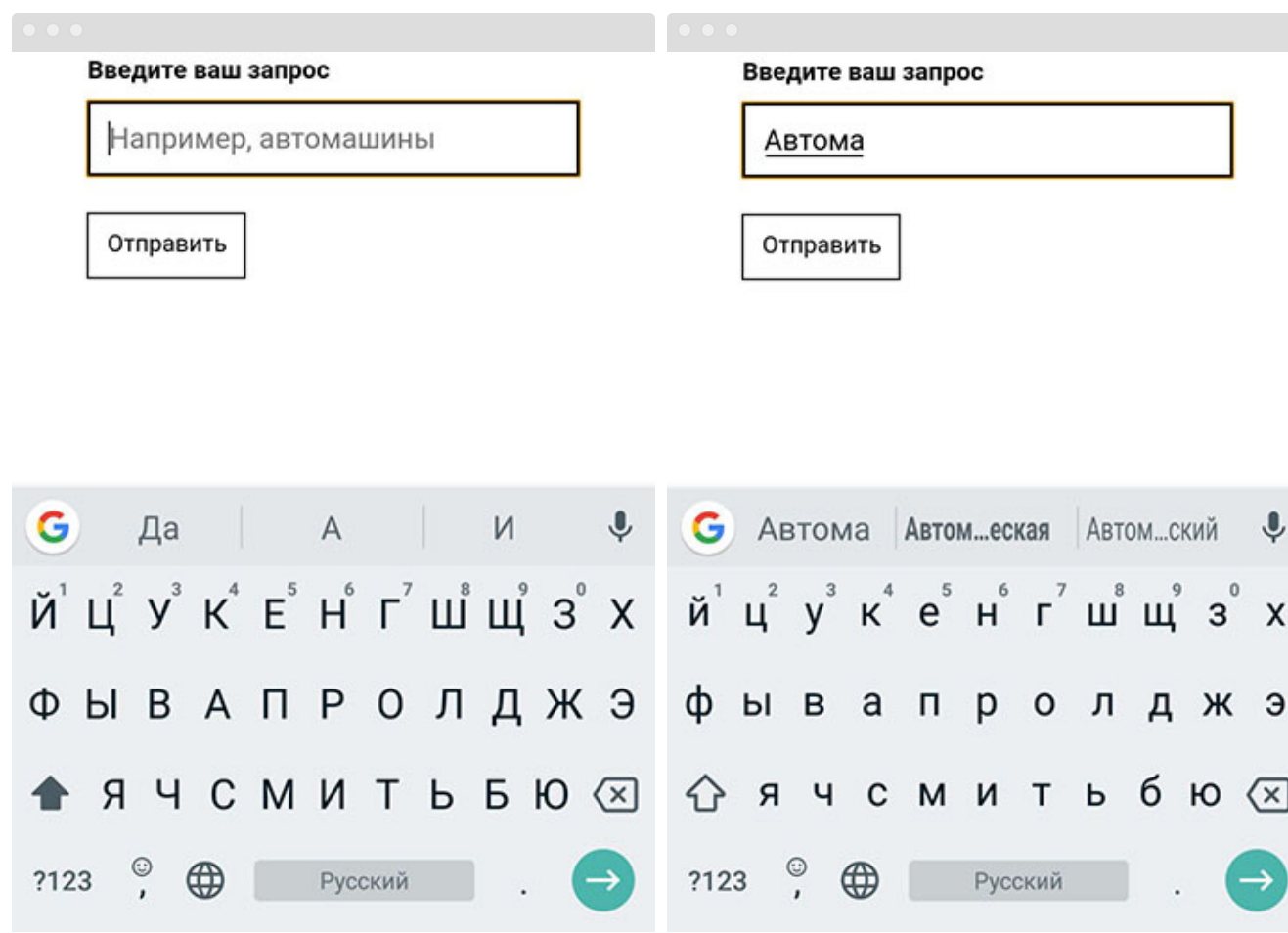
РАЗМЕТКА ФОРМЫ

Для того, чтобы сверстать поле ввода, мы использовали тег `<input>` с `type="text"`:

```
<input class="field" type="text"  
  placeholder="Например, автомашины" required>
```

КЛАВИАТУРА ПРИ ВВОДЕ

Попробуем ввести данные в поле. Когда мы устанавливаем курсор в поле и начинаем вводить запрос, появляется экранная клавиатура:



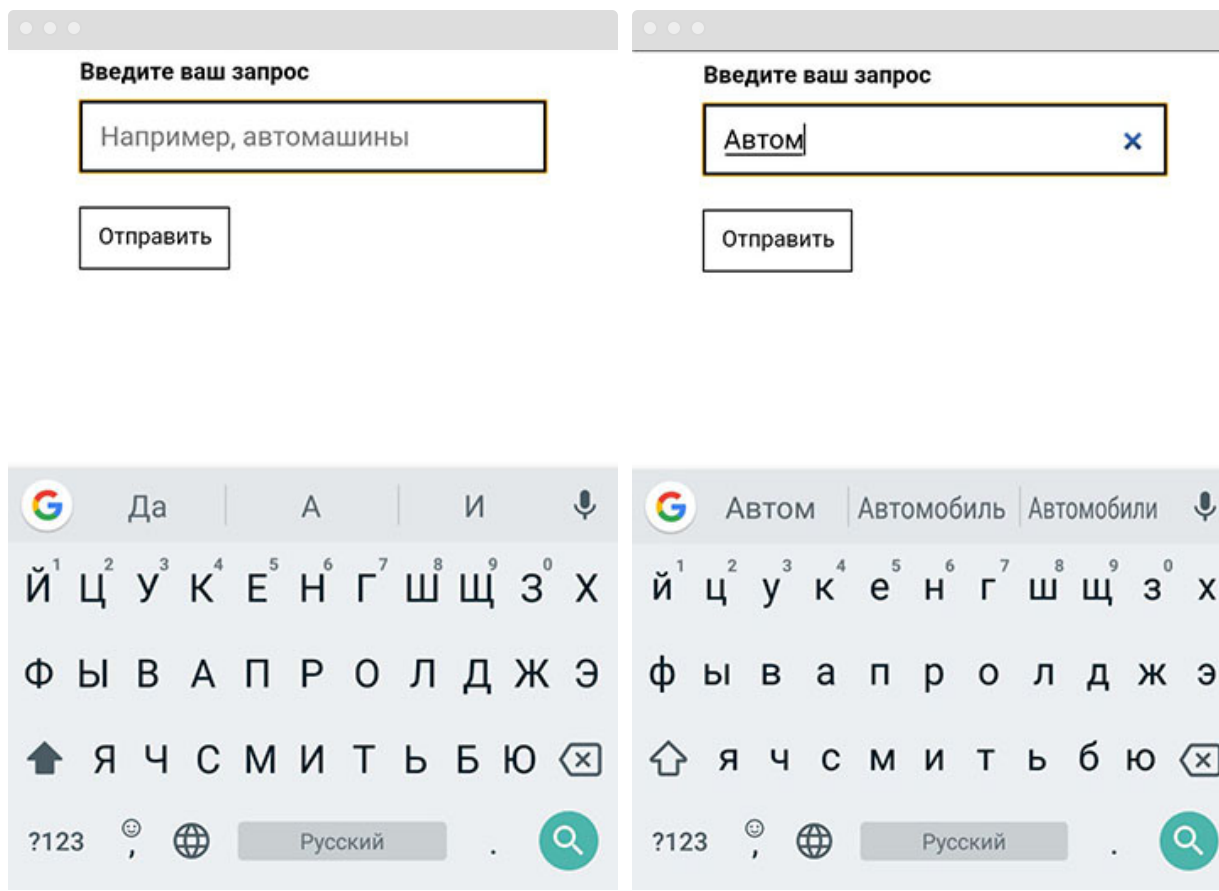
ТИПЫ В HTML5

Ранее для тега `input` мы использовали `type="text"`. Но в HTML5 появились новые типы для полей ввода. Например, укажем `type="search"` и посмотрим, что изменится:

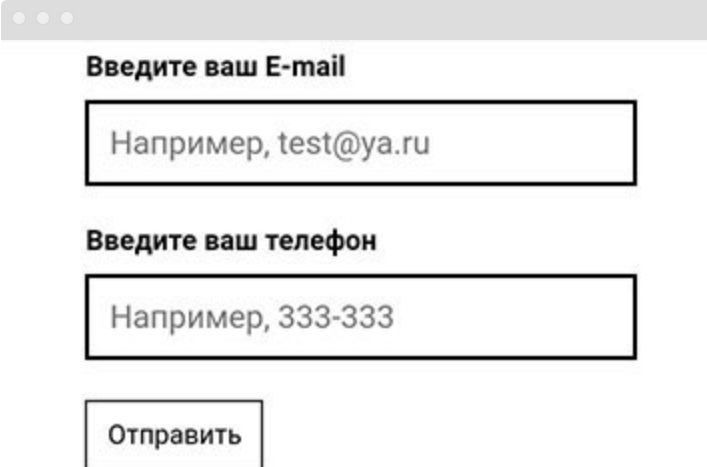
```
<input class="field" type="search"  
  placeholder="Например, автомашины" required>
```


СПЕЦИАЛЬНЫЕ СИМВОЛЫ НА КЛАВИАТУРЕ

На сенсорной клавиатуре появилась кнопка со значком лупы, которая отправляет запрос. А также в конце поля появился «крестик», нажав на который, можно быстро очистить поле.



ФОРМА ВВОДА ТЕЛЕФОНА И EMAIL



A web form with a light gray header bar containing three small white circles. Below the header, the text "Введите ваш E-mail" is followed by a text input field containing the placeholder "Например, test@ya.ru". Below this, the text "Введите ваш телефон" is followed by a text input field containing the placeholder "Например, 333-333". At the bottom is a button labeled "Отправить".

Введите ваш E-mail

Например, test@ya.ru

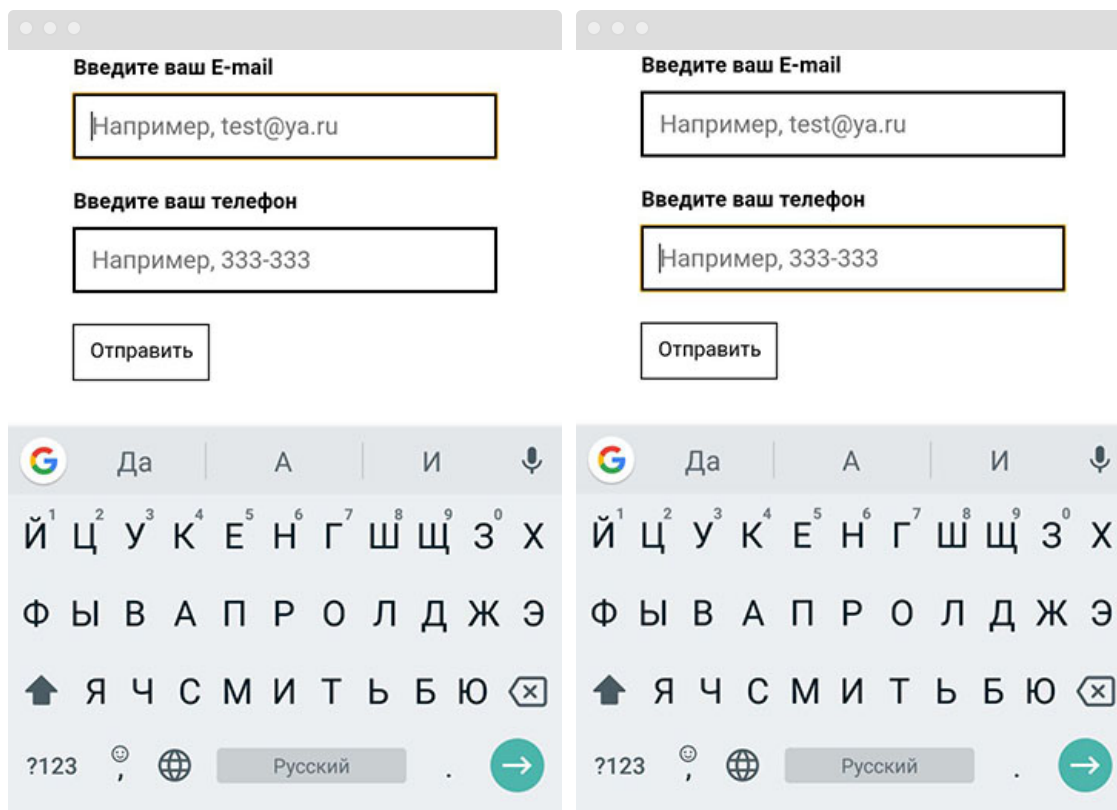
Введите ваш телефон

Например, 333-333

Отправить

НАЧНЕМ ВВОДИТЬ ДАННЫЕ

Чтобы ввести email, нужно не только поменять язык ввода на английский, но и найти значок @ — для этого переключиться на раскладку с цифрами и символами. Также придется перейти на раскладку с цифрами и для ввода телефона.



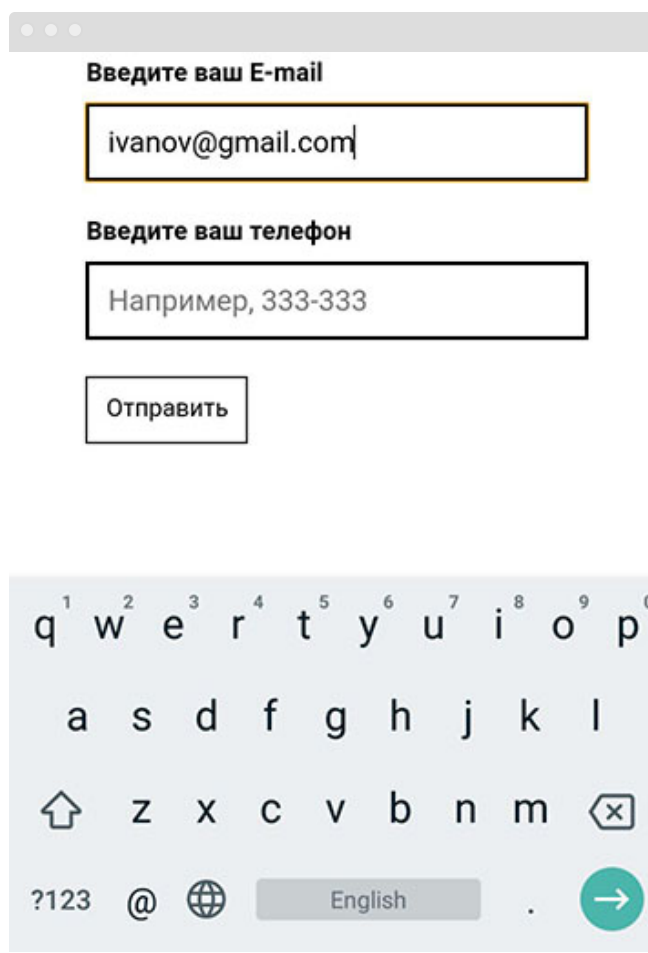
МЕНЯЕМ ПОВЕДЕНИЕ

Такое поведение возникает, потому что указан `type="text"` для тега `<input>`. Но мы можем использовать типы `email` и `tel`:

```
1 <input class="field" type="email"  
2   placeholder="Например, test@ya.ru" required>  
3 <input class="field" type="tel"  
4   placeholder="Например, 333-333" required>
```

УДОБНЫЙ ВВОД АДРЕСА ПОЧТЫ

Теперь при вводе данных в поле `email`, значок `@` удобно располагается на основной раскладке клавиатуры:



ЦИФРОВАЯ КЛАВИАТУРА

При вводе телефона у нас появляется особая — цифровая — клавиатура:

The image shows a web form with two input fields and a submit button. The first field is labeled "Введите ваш E-mail" and contains the text "ivanov@gmail.com". The second field is labeled "Введите ваш телефон" and contains the text "555-". Below the fields is a button labeled "Отправить".

Below the form is a numeric keypad with the following layout:

1	2 ABC	3 DEF	—
4 GHI	5 JKL	6 MNO	⌋
7 PRQS	8 TUV	9 WXYZ	⌫
* #	0 +	.	➡



ДОПОЛНИТЕЛЬНАЯ ПРОВЕРКА

Помимо стандартной валидации — вывода сообщения о необходимости заполнить «обязательные» поля (имеющие атрибут `required`) — для поля email добавляется автоматическая проверка на соответствие шаблону электронного адреса.

ОШИБКА В EMAIL

Если в такое поле email будет введен с ошибкой, об этом также появится сообщение при нажатии кнопки **Отправить** :

Сообщение «email должен
содержать @»

Введите ваш E-mail

test

! Please include an '@' in the email address. 'test' is missing an '@'.

Например, test@example.com

Отправить

q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ↵
?123 @ 🌐 English . ➡

Сообщение «введите часть email
после @»

Введите ваш E-mail

test@

! Please enter a part following '@'. 'test@' is incomplete.

q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ↵
?123 @ 🌐 English . ➡

ПОДДЕРЖКА В СТАРЫХ БРАУЗЕРАХ

Рассмотренные типы появились только в HTML5. Соответственно, не поддерживаются в старых браузерах.

Если не указать атрибут `type` для `<input>`, браузер считает такое поле ввода текстовым — `type="text"`. Все браузеры игнорируют неизвестные им атрибуты. Если браузер не будет «знать» указанный `type` — он просто посчитает поле текстовым.

ОТОБРАЖЕНИЕ ПОЛЕЙ В IE9

The image displays two browser window mockups side-by-side, illustrating form rendering in Internet Explorer 9. Each window has a grey title bar with three small circles on the left.

Left Window:

- Label: **Введите ваш запрос**
- Input field: A single rectangular text input box.
- Button: **Отправить** (Send)

Right Window:

- Label: **Введите ваш E-mail**
- Input field: A rectangular text input box with a vertical cursor line on the left.
- Label: **Введите ваш телефон**
- Input field: A rectangular text input box.
- Button: **Отправить** (Send)

[Live Demo](#)

ЕДИНИЦЫ ИЗМЕРЕНИЯ

rem

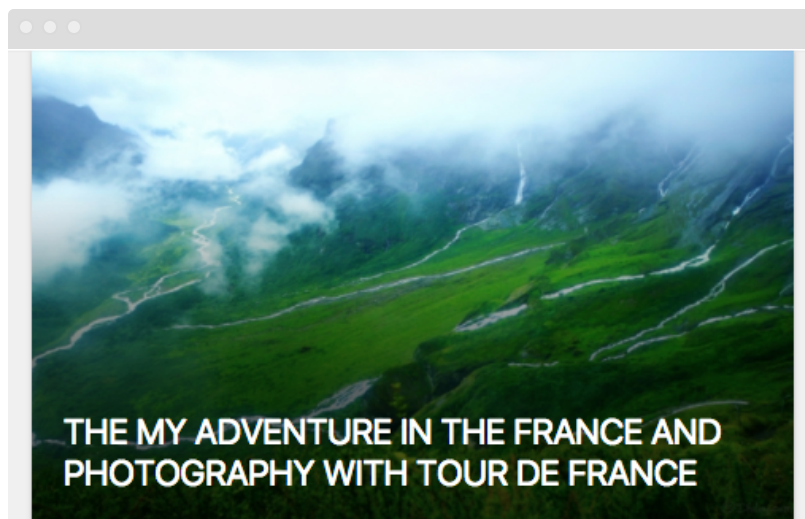


АДАПТИВНЫЙ ТЕКСТ

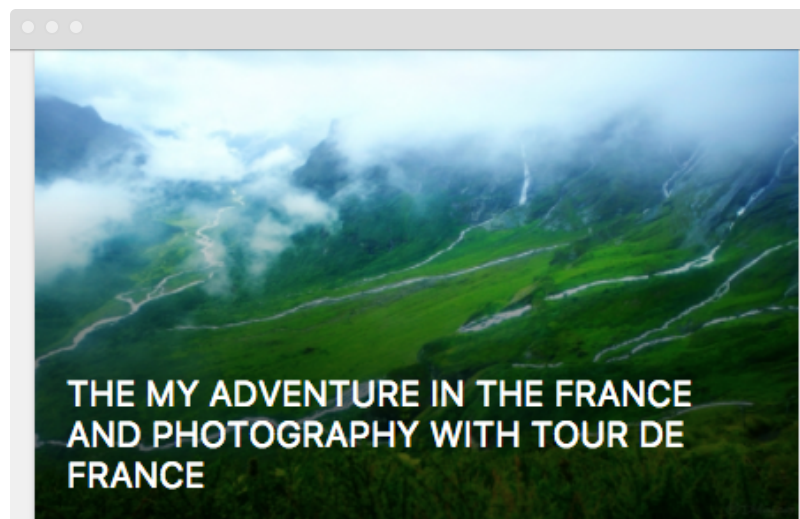
При верстке адаптивных сайтов верстальщик всегда встречается с необходимостью менять размеры шрифта для каждой платформы. Решая проблему в лоб, специалист указывает размерности текста для каждого элемента. Но можно ли сделать так, чтобы текст зависел от одного значения?

ЗАГОЛОВОК В КАРТОЧКЕ СТАТЬИ

На десктопе



На мобильном



СТИЛИ ДЛЯ ЗАГОЛОВКОВ

Для стилизации размеров заголовков мы используем `px`:

```
1  @media screen and (min-width: 768px) {  
2    .card__title {  
3      font-size: 22px;  
4    }  
5  }  
6  
7  @media screen and (max-width: 767px) {  
8    .card__title {  
9      font-size: 18px;  
10   }  
11 }
```

Было бы удобно, чтобы размер текста задавался один раз и зависел от какого-то одного значения, меняя которое, можно было бы менять и размер текста. Для такой задачи идеально подойдут единицы `rem`.



`rem` (root (корневые) `em`) — позволяют задавать размер шрифта, зависящий от свойства `font-size` корневого элемента, то есть, тега `html`.



РАЗМЕР ШРИФТА НА ДЕСКТОПЕ

Например, для устройств с шириной экрана от 768px зададим `font-size` равный `10px`:

```
1 @media screen and (min-width: 768px) {  
2     html {  
3         font-size: 10px;  
4     }  
5 }
```


ФОРМУЛА РАСЧЕТА `rem`

Значение `10px` мы взяли для более удобного расчета значений. Для того, чтобы перевести из `px` в `rem`, нужно воспользоваться следующей формулой:

$$V_{rem} = V_{px} / F_{html}$$

- `Vrem` — значение в `rem`;
- `Vpx` — значение в `px`;
- `Fhtml` — значение свойства `font-size` у тега `html`.

РАССЧИТАЕМ РАЗМЕР ШРИФТА

Теперь, используя формулу, рассчитаем размер шрифта блока

`.card__title`:

$$22\text{px} / 10\text{px} = 2.2\text{rem}$$

Запишем получившееся значение в CSS:

```
1 | .card__title {  
2 |   font-size: 2.2rem;  
3 | }
```

ФОРМУЛА РАСЧЕТА РАЗМЕРА ROOT

Чтобы сохранить изначальные значения, нужно рассчитать значение `font-size` для тега `html` для устройств, имеющих ширину экрана до 767px.

В этом случае мы будем рассчитывать `FShtml`. Исходя из следующей формулы `Vrem = Vpx / FShtml`, можно сделать вывод, что `FShtml` будет рассчитываться как

$$FShtml = Vpx / Vrem$$

РАССЧИТАЕМ ДЛЯ МОБИЛЬНЫХ

V_{px} для устройств с экраном до 767px равняется 18px, V_{rem} мы рассчитали ранее. Подставим значения:

$$FS_{html} = 18 / 2.2 = 8.18181818182$$

ПЕРЕПИСЫВАЕМ СТИЛИ

Округлим значение до целого числа и получим `8px` для свойства `font-size` у тега `html`:

```
1  .card__title {  
2    font-size: 2.2rem;  
3  }  
4  
5  @media screen and (min-width: 768px) {  
6    html {  
7      font-size: 10px;  
8    }  
9  }  
10  
11 @media screen and (max-width: 767px) {  
12   html {  
13     font-size: 8px;  
14   }  
15 }
```

ПОЛЬЗОВАТЕЛЬСКИЙ РАЗМЕР ШРИФТА

Указывать размер шрифта `html` в пикселях — не совсем оптимальное решение. Пользователь может задать удобный для него размер текста. Например, более крупный, чем стандартный. А свойство `font-size` со значением в `px` всегда будет перекрывать настройки браузера.

Более удобным вариантом будет указать свойство в процентах, тогда этот размер будет вычисляться пропорционально размеру шрифта, установленному пользователем в настройках браузера.

РАСЧЕТ РАЗМЕРА ROOT В ПРОЦЕНТАХ

Для этого надо высчитать пропорцию. По умолчанию в браузере размер текста установлен как 16px – это 100%. Для устройств с шириной экрана от 768px нужно посчитать, сколько будет 10px от 16px.

$$10 * 100 / 16 = 62.5\%$$

СЧИТАЕМ В ПРОЦЕНТАХ ДЛЯ МОБИЛЬНЫХ

Сделаем то же самое для второй группы, только вместо 10рх нам нужно использовать 8рх :

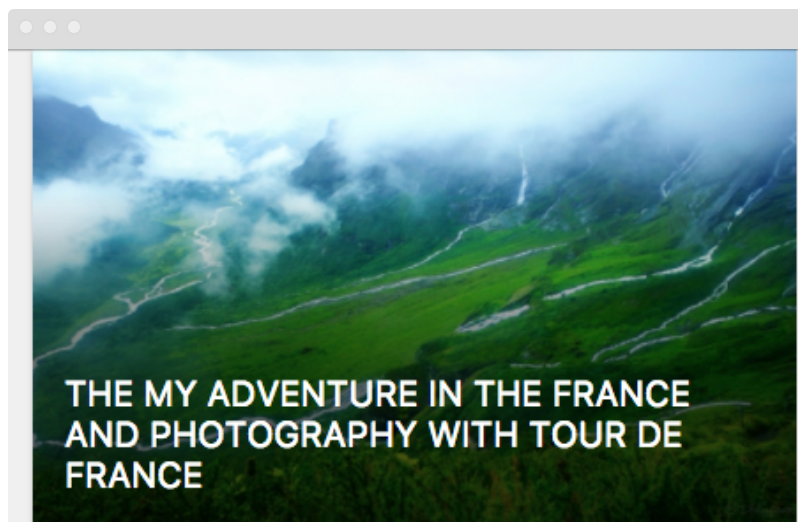
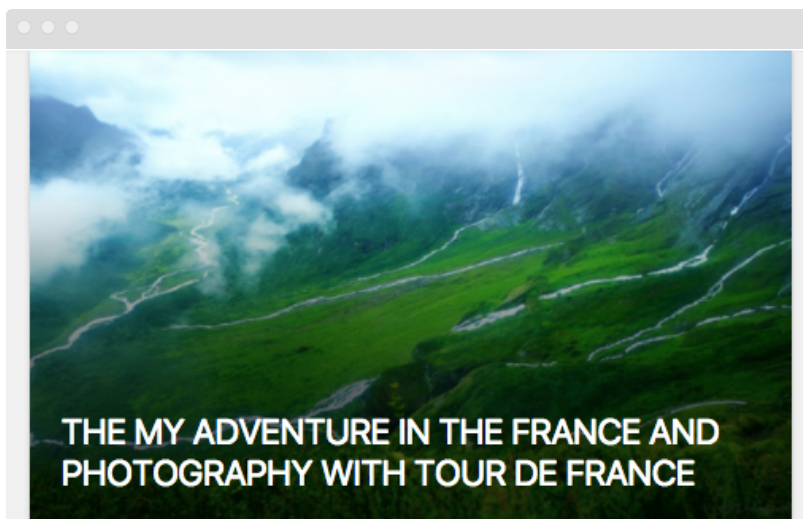
$$8.19 * 100 / 16 = 51.1875\%$$

ОКРУГЛЯЕМ И МЕНЯЕМ CSS

```
1  .card__title {  
2      font-size: 2.2rem;  
3  }  
4  
5  @media screen and (min-width: 768px) {  
6      html {  
7          font-size: 62.5%;  
8      }  
9  }  
10  
11 @media screen and (max-width: 767px) {  
12     html {  
13         font-size: 51%;  
14     }  
15 }
```

НУЖНЫЙ РЕЗУЛЬТАТ ДОСТИГНУТ

Внешний вид верстки не изменился, но код стал намного универсальнее:



ОТСТУПЫ В КАРТОЧКЕ

При верстке карточки мы использовали следующие значения для `margin` и `padding`:

```
1  @media screen and (min-width: 768px) {  
2    .card p {  
3      margin-bottom: 20px;  
4    }  
5  
6    .card__body {  
7      padding: 30px 25px 20px;  
8    }  
9  }  
10  
11 @media screen and (max-width: 767px) {  
12   .card p {  
13     margin-bottom: 16px;  
14   }  
15  
16   .card__body {  
17     padding: 24px 20px 16px;  
18   }  
19 }
```

ОТСТУПЫ В `rem`

```
1  .card p {  
2    margin-bottom: 2rem;  
3  }  
4  
5  .card__body {  
6    padding: 3rem 2.5rem 2rem;  
7  }  
8  
9  @media screen and (min-width: 768px) {  
10   html {  
11     font-size: 62.5%;  
12   }  
13 }  
14  
15 @media screen and (max-width: 767px) {  
16   html {  
17     font-size: 51%;  
18   }  
19 }
```

ПРОВЕРИМ ЗНАЧЕНИЯ

Теперь проверим, подходят ли наши коэффициенты для мобильной версии. Умножим значения в `rem` на величину шрифта для `html` мобильной версии:

– для `margin-bottom` селектора `.card p`:

$$2 * 8px = 16px$$

– для `padding-top` селектора `.card__body`:

$$3 * 8px = 24px$$

– для `padding-left` и `padding-right` селектора `.card__body`:

$$2.5 * 8px = 20px$$

[Live Demo](#)



ИТОГИ

МЕДИАФУНКЦИИ `aspect-ratio` И `orientation`

- Медиафункция `aspect-ratio` определяет соотношение ширины к высоте отображаемой области устройства.
- Коэффициент записывается в виде пропорции, состоящей из медиафункции `width` и медиафункции `height`, разделенных знаком `/`.
- `aspect-ratio` может использоваться с приставками `min-` или `max-`.
- Медиафункция `orientation` определяет, в каком режиме находится устройство – портретном или ландшафтном.
- `orientation` принимает одно из двух возможных значений: `portrait`, когда ширина отображаемой области устройства больше или равна высоте, и `landscape` в других случаях.

ОСОБЕННОСТИ HTML-ЭЛЕМЕНТОВ

- Раньше для тега `input` мы использовали `type="text"`. Сейчас, в HTML5, появились новые типы для полей ввода.
- Для создания поля поиска нужно указать `type="search"`.
- Для ввода email и телефона используют типы `email` и `tel`.
- При вводе данных с мобильных устройств в такие поля появляется специальная клавиатура — с символом `@` при вводе email и цифровая клавиатура при вводе номера телефона.
- Также кроме стандартной валидации — вывода сообщения о необходимости заполнить «обязательные» (`required`) поля — для поля `email` добавляется автоматическая проверка на соответствие шаблону электронного адреса.

ЕДИНИЦЫ ИЗМЕРЕНИЯ `rem`

- Единицы измерения `rem` позволяют задавать размер шрифта, зависящий от свойства `font-size` корневого элемента (тега `html`).
- Формула для перевода из `px` в `rem`: $V_{rem} = V_{px} / FS_{html}$.
Здесь `Vrem` – значение в `rem`, `Vpx` – значение в `px`, `FShtml` – значение свойства `font-size` у тега `html`.
- Формула расчета размера root: $FS_{html} = V_{px} / V_{rem}$.
- Размер шрифта `html` в пикселях – не совсем оптимальное решение. Более удобный вариант – указать свойство в процентах, тогда размер будет вычисляться пропорционально размеру шрифта, установленному в настройках браузера.
- По умолчанию в браузере размер текста – `16px` – это 100%. Для расчета размера root нужно высчитать пропорцию этого значения к `16px`.



НЕТОЛОГИЯ
университет интернет-профессий

Задавайте вопросы и напишите отзыв о лекции!

МИХАИЛ ЛАРЧЕНКО



larchanka@me.com



[+31621967276](tel:+31621967276)