



HTML и CSS. Адаптивная вёрстка и автоматизация

Уровень 2, с 21 ноября 2022 по 30 января 2023

Меню курса

[Главная](#) / [2. Методологии вёрстки](#) /

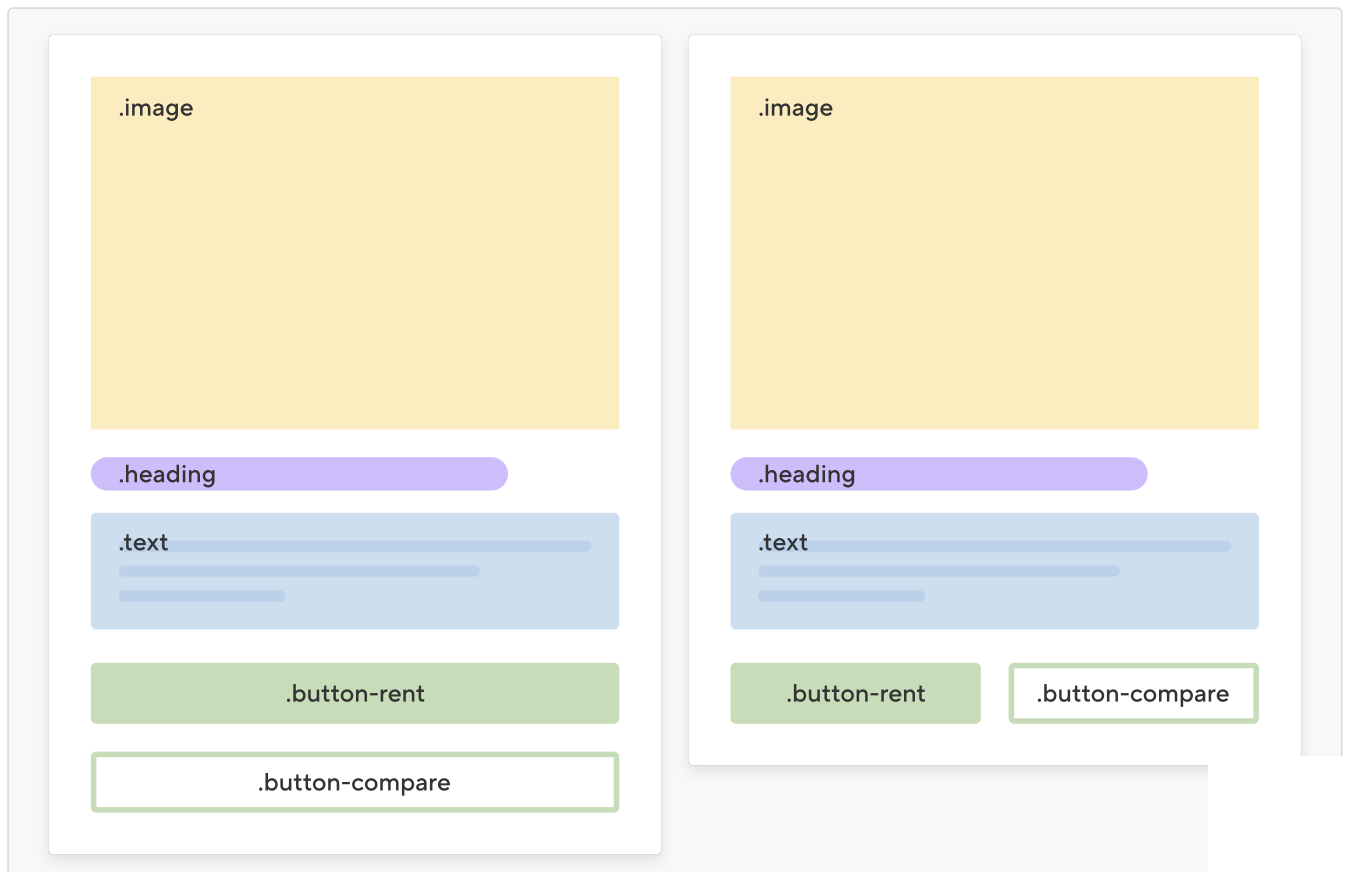
📖 2.14. Алгоритм выделения блоков

🕒 ~ 17 минут

Чтобы избежать типовых ошибок использования БЭМ, предлагаем идти по следующей схеме:

Шаг 1

По умолчанию всё — блоки, каждый фрагмент макета — это отдельный уникальный блок. Продемонстрируем на примере двух немного отличающихся карточек. В первом случае кнопки расположены друг под другом, во втором — рядом.



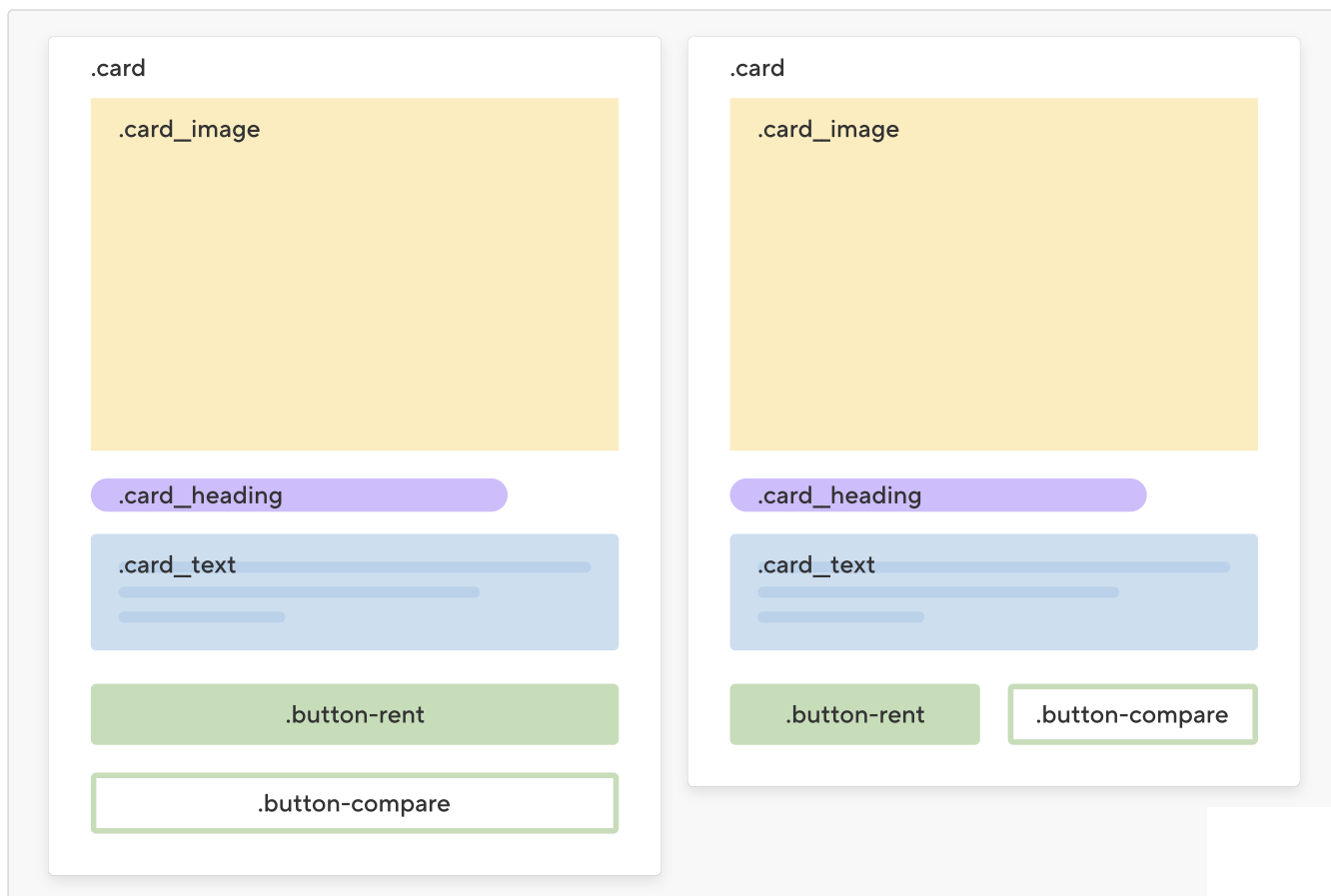
Схемы типовых карточек с заголовком, текстом, картинкой и кнопками, все они блоки

Создаём для всего отдельные блоки. Возможно, блоки будут схожи, они будут повторять друг друга целиком или частично, но мы пока что на это не обращаем внимания. Оставляем это всё на потом.

```
<section class="card">
  <h2 class="heading">Дом из мусора</h2>
  
  <p class="text">Домик «Cabana Floripa» на вершине холма острова Флорианополис
создан уругвайским художником Jaime, который использовал для его постройки
различный мусор, собранный в этом районе: старую древесину, бутылки,
керамическую плитку, осколки зеркал.</p>
  <button class="button-rent" type="button">Арендовать</button>
  <button class="button-compare" type="button">Добавить в сравнение</button>
</section>
```

Шаг 2

Начинаем уточнять, этот блок независимый или нет, используется только самостоятельно или в составе других блоков. Если объект не может существовать отдельно от родителя, то это элемент. Если внутренняя часть будет переиспользоваться, то это блок.

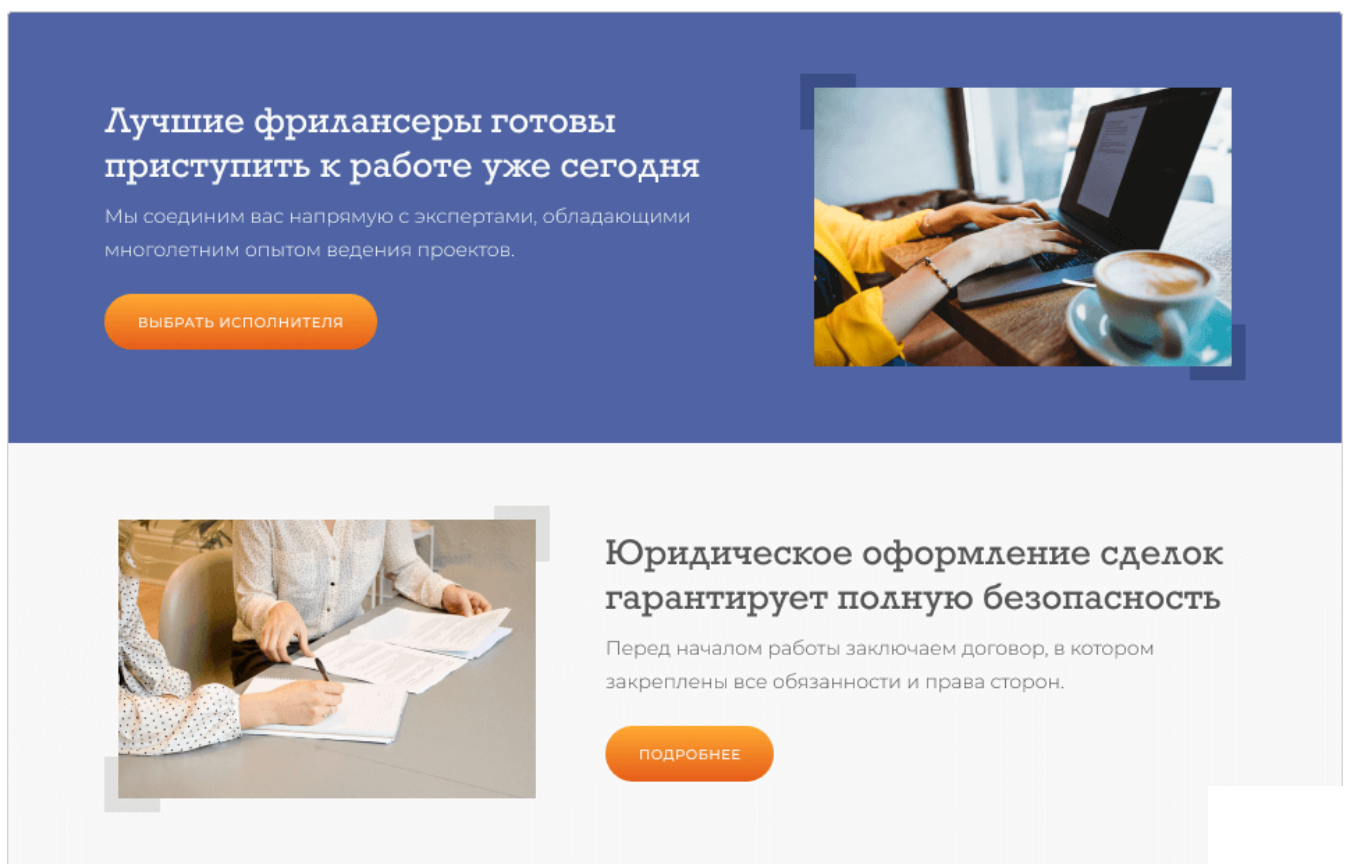


Схемы типовых карточек с заголовком, текстом, картинкой и кнопками, карточка — блок, картинка, заголовок и текст — это его элементы, а кнопки — вложенные блоки

```
<section class="card">
  <h2 class="card_heading">Дом из мусора</h2>
  
  <p class="card_text">Домик «Cabana Floripa» на вершине холма острова Флорианополис создан уругвайским художником Jaime, который использовал для его постройки различный мусор, собранный в этом районе: старую древесину, бутылки, керамическую плитку, осколки зеркал.</p>
  <button class="button-rent" type="button">Арендовать</button>
  <button class="button-compare" type="button">Добавить в сравнение</button>
</section>
```

Но всё ли так однозначно и просто в том, чтобы отделить блоки от элементов. Разберём на практике.

У нас есть макет лендинга, который состоит больше, чем из одной страницы, и, как у любого лендинга, у главной страницы есть множество разделов. Каждый из этих разделов со своим заголовком. Визуально эти заголовки практически идентичны: шрифт, размер совпадают. Заголовки различаются только цветом, светлый на контрастном фоне и тёмным, если фона нет.



Фрагмент макета лендинга с похожими разделами

Следовало ли сделать заголовок раздела отдельным независимым блоком?

Аргументы «**За**» независимый блок:

- С одной стороны, это соответствовало бы духу БЭМ: заголовки повторяются, их код идентичен.
- Если нам скажут, что заголовки нужно увеличить, было бы удобно делать это из модуля заголовка.

Аргументы «**Против**» независимого блока:

- Заголовок не может использоваться отдельно от раздела, заголовки вообще не функциональны в отрыве от того, что они озаглавливают.
- Может измениться заголовок только одного раздела, в этом случае лучше, чтобы это был элемент блока *Раздел*.
- В случае с заголовком стилей совсем мало. Это не кнопка, у которой есть состояния, обнуление свойств по умолчанию, фоны и обводки, иконки, логика и скрипты для поддержки этой логики. Это просто фрагмент текста с параметрами шрифта, у него нет никаких пользовательских действий или чего-то подобного.

Для небольшого лендинга и проекта, который не планируется развивать, можно оставить заголовки элементами, не выделять их в отдельный блок. Кода, который при этом придётся повторить в отдельных блоках, очень мало, это только параметры шрифта.

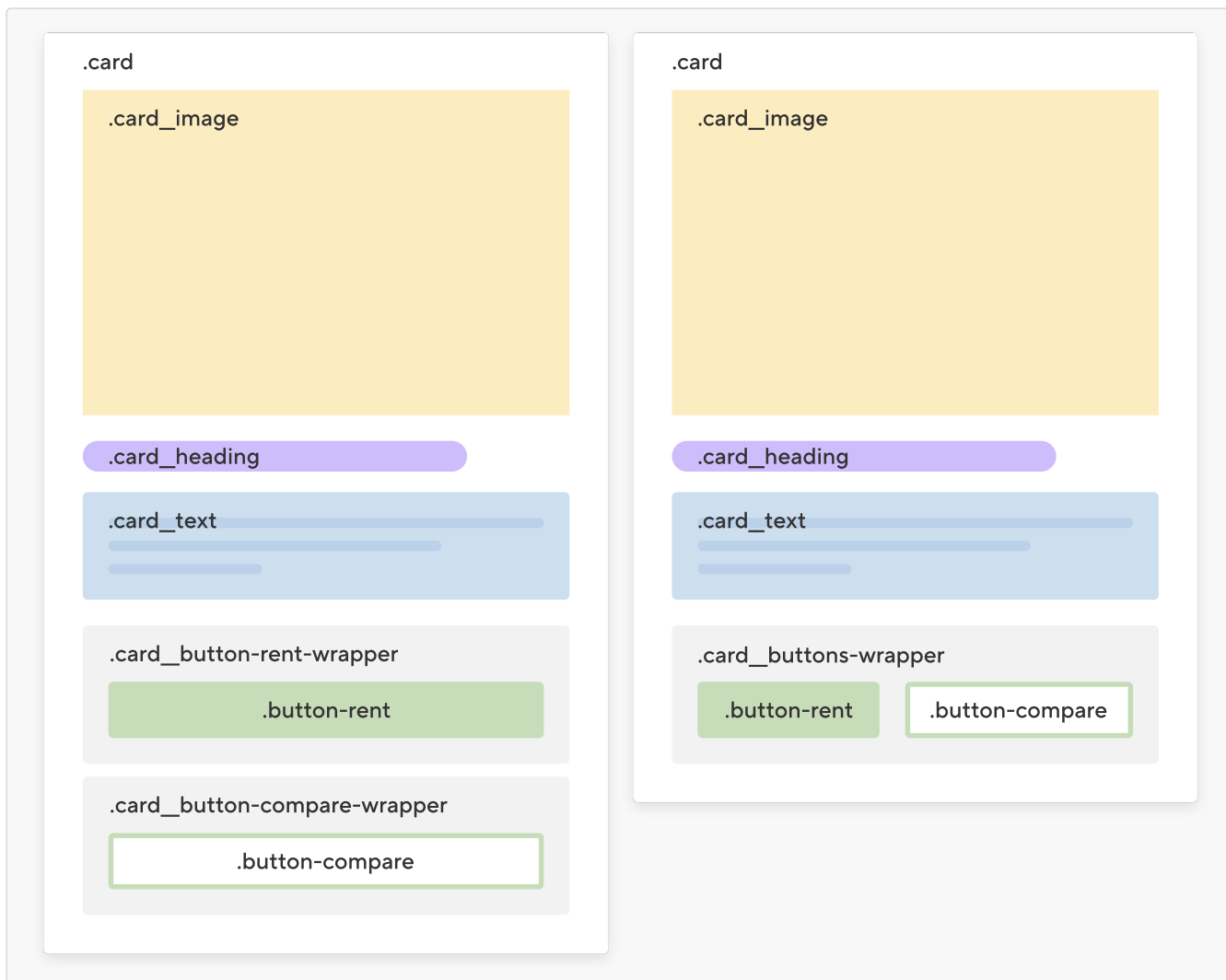
При этом нельзя утверждать, что для настоящего проекта такое решение было бы правильным.

У крупных проектов обычно бывает «дизайн-система» или *UI-kit*, в который попадают такие системные элементы, как кнопки, заголовки разных видов, карточки, списки, возможно — цитаты, другие текстовые элементы, даже виды абзацев. Если бы наш макет был для такого случая, то, вероятнее всего, заголовок был бы блоком.

Если вы хорошо разобрались, что такое блок и что такое элемент, как сочетать эти две БЭМ-сущности на одном HTML-элементе, следующие два шага можно объединить.

Шаг 3

Добавляем элементы для внешней модификации вложенных блоков. Это могут быть внешние отступы, размеры, выравнивания и другие. Сперва упростим себе задачу и элементы добавим как обёртки над вложенными блоками. Для первого вида у каждого вложенного блока с кнопкой будет своя обёртка, для второго обёртка одна, в ней нам нужно выстроить кнопки в ряд.



Управляем положением вложенного блока с кнопкой через элемент карточки

Карточка первого типа:

```
<section class="card">
  <h2 class="card__heading">Дом из мусора</h2>
  
  <p class="card__text">Домик «Cabana Floripa» на вершине холма острова
Флорианополис создан уругвайским художником Jaime, который использовал для его
постройки различный мусор, собранный в этом районе: старую древесину, бутылки,
керамическую плитку, осколки зеркал.</p>
  <div class="card_button-rent-wrapper">
    <button class="button-rent" type="button">Арендовать</button>
  </div><div class="card_button-compare-wrapper">
    <button class="button-compare" type="button">Добавить в сравнение</button>
  </div>
</section>
```

Карточка второго типа:

```

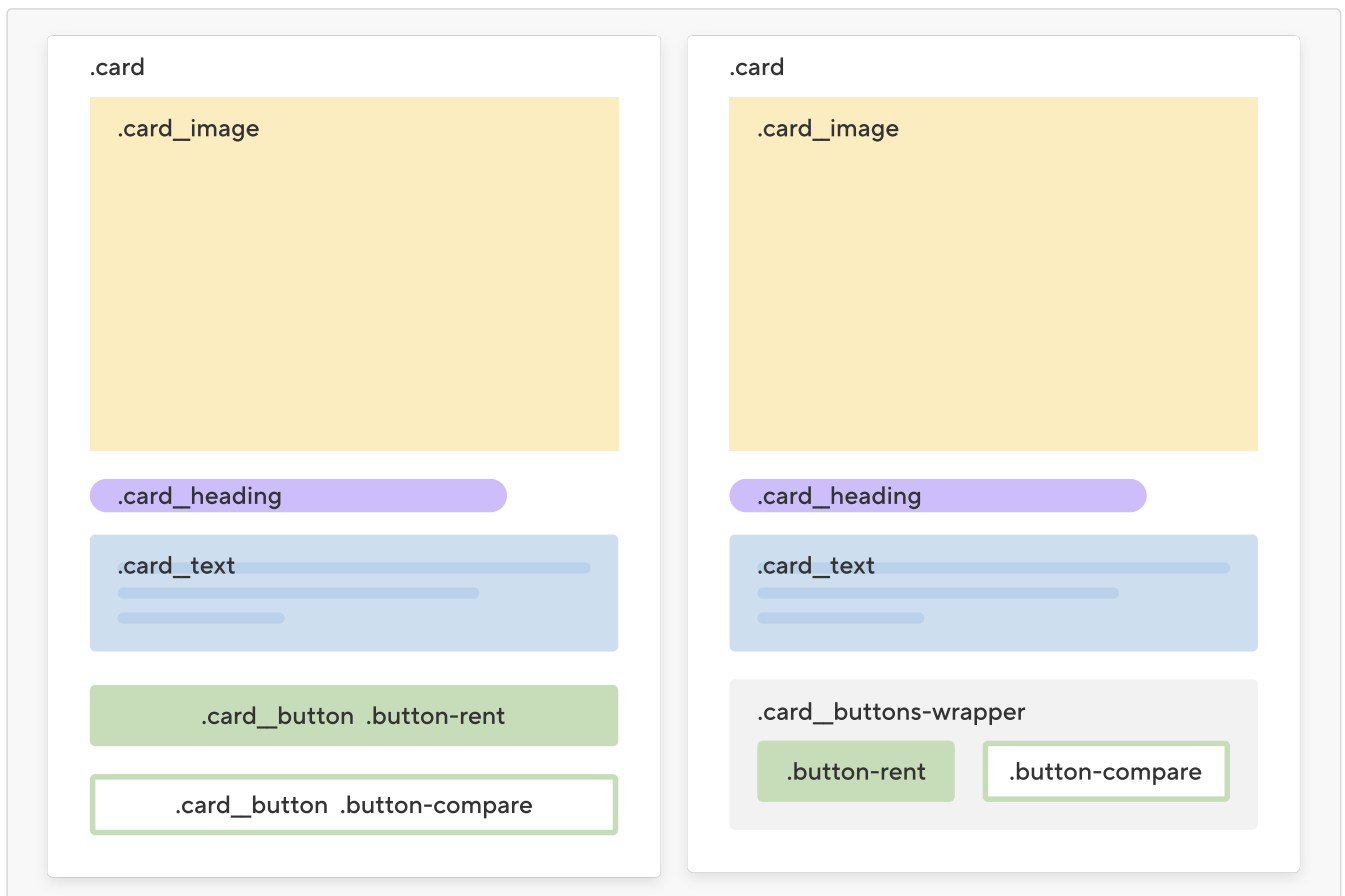
<section class="card">
  <h2 class="card__heading">Дом из мусора</h2>
  
  <p class="card__text">Домик «Cabana Floripa» на вершине холма острова
Флорианополис создан уругвайским художником Jaime, который использовал для его
постройки различный мусор, собранный в этом районе: старую древесину, бутылки,
керамическую плитку, осколки зеркал.</p>
  <div class="card__buttons-wrapper">
    <button class="button-rent" type="button">Арендовать</button>
    <button class="button-compare" type="button">Добавить в сравнение</button>
  </div>
</section>

```

Шаг 4

Оптимизируем разметку блоков, используем *миксование* и убираем лишние обёртки.

Где это возможно, один HTML-элемент будет одновременно и блоком и элементом. В нашем примере миксование можно использовать только для карточек первого типа. Для карточки второго типа всё остаётся без изменений.



Оптимизируем разметку блоков, в карточке первого типа кнопки это одновременно и элемент и блок

```

<section class="card">
  <h2 class="card__heading">Дом из мусора</h2>
  
  <p class="card__text">Домик «Cabana Floripa» на вершине холма острова
Флорианополис создан уругвайским художником Jaime, который использовал для его
постройки различный мусор, собранный в этом районе: старую древесину, бутылки,
керамическую плитку, осколки зеркал.</p>
  <button class="card__button button-rent" type="button">Арендовать</button>
  <button class="card__button button-compare" type="button">Добавить в
сравнение</button>
</section>

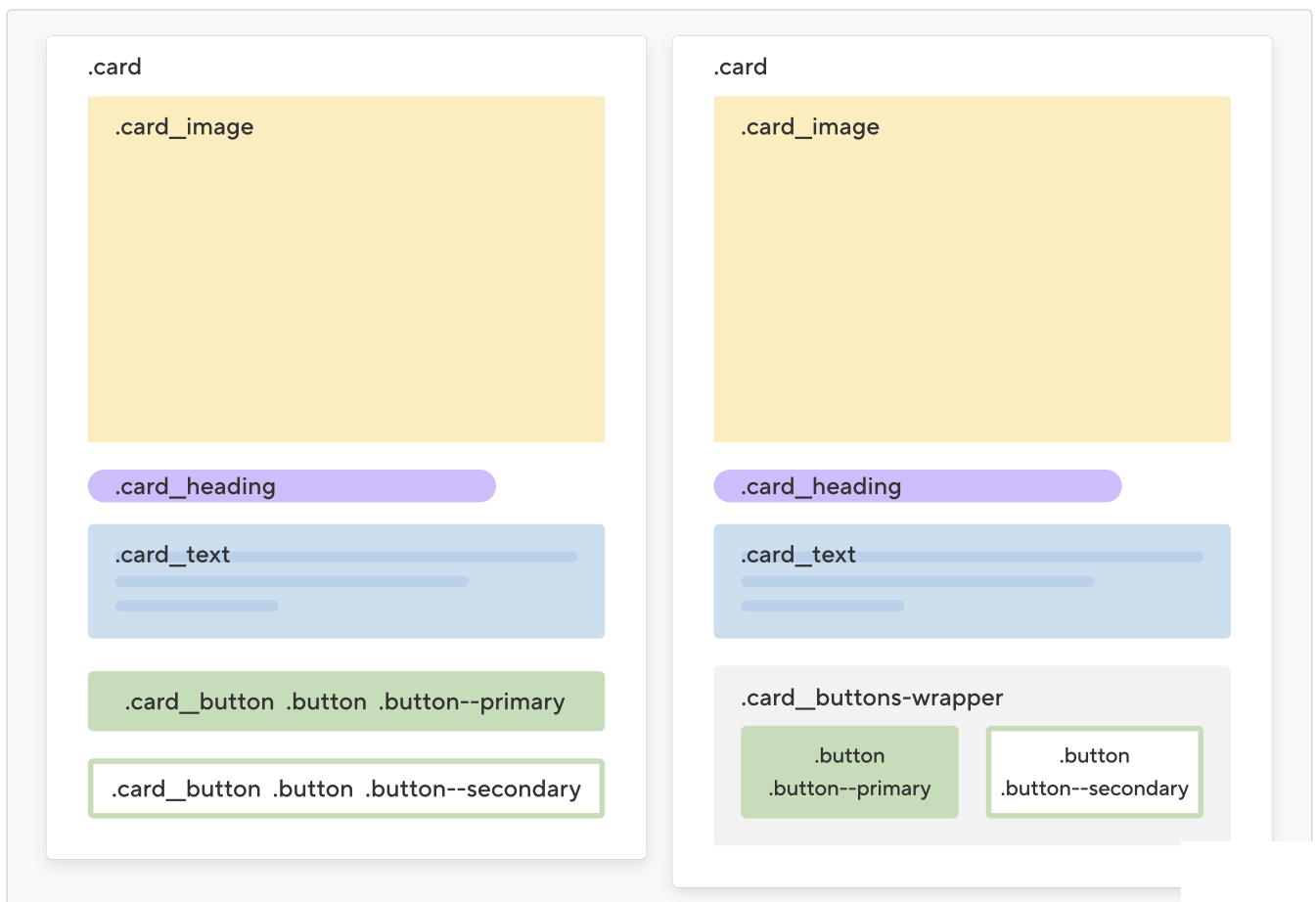
```

Шаг 5

Анализируем блоки/элементы, смотрим и выявляем одинаковые.

Это то действие, которое мы отложили на потом в *Шаге 1*.

Если мы видим, что блоки схожи, изменяются параллельно, можно слить их в один блок и переиспользовать его. Самый частый вид блока — это базовый блок, от которого будем модифицироваться. Если блок менее распространён — это модификатор блока. Так наши уникальные блоки становятся уже не уникальными и могут использоваться снова и снова.



Переиспользуемый блок кнопка и два модификатора для неё



Карточка первого типа:

```
<section class="card">
  <h2 class="card__heading">Дом из мусора</h2>
  
  <p class="card__text">Домик «Cabana Floripa» на вершине холма острова
  Флорианополис создан уругвайским художником Jaime, который использовал для его
  постройки различный мусор, собранный в этом районе: старую древесину, бутылки,
  керамическую плитку, осколки зеркал.</p>
  <button class="card__button button button--primary"
  type="button">Арендовать</button>
  <button class="card__button button button--secondary" type="button">Добавить в
  сравнение</button>
</section>
```

Карточка второго типа:

```
<section class="card">
  <h2 class="card__heading">Дом из мусора</h2>
  
  <p class="card__text">Домик «Cabana Floripa» на вершине холма острова
  Флорианополис создан уругвайским художником Jaime, который использовал для его
  постройки различный мусор, собранный в этом районе: старую древесину, бутылки,
  керамическую плитку, осколки зеркал.</p>
  <div class="card__buttons-wrapper">
    <button class="button button--primary" type="button">Арендовать</button>
    <button class="button button--secondary" type="button">Добавить в
  сравнение</button>
  </div>
</section>
```

И в конце покажем карточки того и другого типа на действующих сайтах.







Смартфон APPLE iPhone 11 Pro 512Gb, черный

✓ Есть в наличии
★★★★★
246 отзывов

Объем памяти

64Gb
128Gb
256Gb
512Gb

Цвет

Характеристики	Все характеристики
Диагональ экрана, в дюймах:	5.8
Разрешение экрана, в пикселях:	2436x1125
Встроенная память, в Гб:	512
Оперативная память, в Гб:	4
Емкость аккумулятора, в мА·ч:	3046
Количество основных камер:	три
Год выхода модели:	2019
Тип дисплея:	Super Retina XDR

Цена: 107 990 Р


Добавить в корзину

Рассрочка от 10 800 Р / месяц

Пример макета с карточкой первого типа


Новая коллекция

[Смотреть все](#)




Туфли Shein 17 500 Р

[Подробнее](#)
[В корзину](#)




Моли Grace 15 000 Р

[Подробнее](#)
[В корзину](#)



Кроссовки Brand 9 500 Р

[Подробнее](#)
[В корзину](#)



Ботинки Kelly 18 500 Р

[Подробнее](#)
[В корзину](#)

Пример макета с карточкой второго типа

Разберём ещё несколько случаев на практике.

К примеру, у нас есть два раздела *Преимущества работы с нами* и *Категории товаров*, которые очень похожи (заголовки, сетка), но есть и ряд отличий.

Преимущества работы с нами



Доставка

Доставим ваш товар
совершенно бесплатно



Гарантия

Заменим бракованный
товар на новый



Кредит

Залезть в долговую яму
стало ещё проще

Категории товаров

Бытовая техника

Холодильники
Стиральные машины
Пылесосы
Утюги и отпариватели

Электроника

Компьютеры
Электронные книги
Игровые приставки
Фото- и видеокамеры
Телефоны

Инструменты

Дрели и перфораторы
Циркулярные пилы
Электролобзики
УШМ

Макет с разделами Преимущества работы с нами и Категории товаров

Возможны три варианта реализации этого кейса:

1. Взять блок *Преимущества работы с нами*, его модифицировать, убрать специфические свойства, превратить в базовый (абстрактный) класс `section`. Для свойств, которые убрали из базового класса добавить модификатор — это будет реализация для блока *Преимущества работы с нами* `section section--profit`. Для блока *Категории товаров* использовать другую модификацию базового класса `section section--category`. Правда со временем могут возникнуть проблемы при изменении (рефакторинге), полезут баги, потребуется всё снова тестировать. Как итог, больше потраченного времени.
2. Взять блок *Преимущества работы с нами* за основу, без изменений класс `section`. А для блока *Категории товаров* сделать модификатор. Появятся дублирующие свойства, которые переопределяются для блока *Категории товаров* класс `section section--category`. Из минусов — сложнее переопределять существующие стили, появится лишний код.
3. Взять блок *Преимущества работы с нами* (`profit`), полностью его скопировать и назвать копию блоком *Категории товаров* (`category`), внести правки в стили

мы просто создаём копию блока. Этот способ самый быстрый и дешёвый. Но при таком подходе сильно раздувается кодовая база.

Замечание

Базовый класс (базовый блок) — это класс с минимальным количеством свойств. Необязательно базовый блок будет полностью самодостаточным блоком, который может быть блоком без каких-либо модификаторов.

Базовый класс для кнопки

Первичное действие

Вторичное действие

Базовый класс для кнопки содержит только сброс стилей по умолчанию и не используется сам по себе, только с модификатором

Ещё один пример, форма с настройками профиля пользователя.

Личная информация

Имя

Ваше имя

Дата рождения

Ваш день рождения

E-mail

Ваш e-mail адрес

Город

Выберите ваш город

О себе

Ваши интересы и увлечения

Отменить

Сохранить изменения

Макет формы с настройками профиля пользователя

Управляющие элементы: поля для ввода, выпадающие списки, многострочное поле для ввода имеют одинаковое оформление (шрифт, отступы, граница). Помимо стандартного набора свойств, эти элементы содержат и специфические, характерные только для этого элемента управления CSS-свойства. У многострочного поля для ввода есть CSS-свойство `resize`, которое помогает не сломать сетку (`resize: vertical;`).

У выпадающего списка `select` есть отдельные свойства для стилизации стрелки

Как лучше поступить в этом случае?

Можно каждый управляющий элемент вынести в отдельный блок и стилизовать независимо.

```
<form class="user-settings-form" action="#" method="POST">
  <label class="user-settings-form__field form-field">
    <span class="form-field__label">Имя</span>
    <input class="form-field__control input" type="text" name="add-review-name"
id="add-review-name" placeholder="Ваше имя">
  </label>
  <label class="user-settings-form__field form-field">
    <span class="form-field__label">Город</span>
    <select class="form-field__control select" name="user-settings-form-city"
id="user-settings-form-city">
      <option value="1012">Москва</option>
      <option value="1013">Санкт-Петербург</option>
      <option value="1015">Нижний Новгород</option>
    </select>
  </label>
  ...
  <label class="user-settings-form__field form-field">
    <span class="form-field__label">О себе</span>
    <textarea class="form-field__control textarea" name="user-settings-form-
about" id="user-settings-form-about" placeholder="Ваши интересы и увлечения">
  </textarea>
  </label>
  ...
  <button class="user-settings-form__button button" type="submit">Сохранить
изменения</button>
</form>
```

Стилизация каждого отдельного компонента не повлияет на другие — это плюс, но в общем стилевом файле будет дублирование кода, а это уже минус.

А можно использовать комбинацию блоков. Создать общий блок `form-control` — базовый блок для элементов управления. Он будет содержать только базовый набор CSS-правил, которые есть у всех контролов. Все тонкости стилизации отдельных элементов формы описываются в блоках соответствующих элементов управления.

```
<form class="user-settings-form" action="#" method="POST">
  <label class="user-settings-form__field form-field">
    <span class="form-field__label">Имя</span>
    <input class="form-field__control form-control input" type="text" name="add-
review-name" id="add-review-name" placeholder="Ваше имя">
  </label>
  <label class="user-settings-form__field form-field">
    <span class="form-field__label">Город</span>
    <select class="form-field__control form-control select" name="user-set
form-city" id="user-settings-form-city">
      <option value="1012">Москва</option>
      <option value="1013">Санкт-Петербург</option>
```

```

    <option value="1015">Нижний Новгород</option>
  </select>
</label>
...
<label class="user-settings-form__field form-field">
  <span class="form-field__label">0 себе</span>
  <textarea class="form-field__control form-control textarea" name="user-
settings-form-about" id="user-settings-form-about" placeholder="Ваши интересы и
увлечения"></textarea>
</label>
...
<button class="user-settings-form__button button" type="submit">Сохранить
изменения</button>
</form>

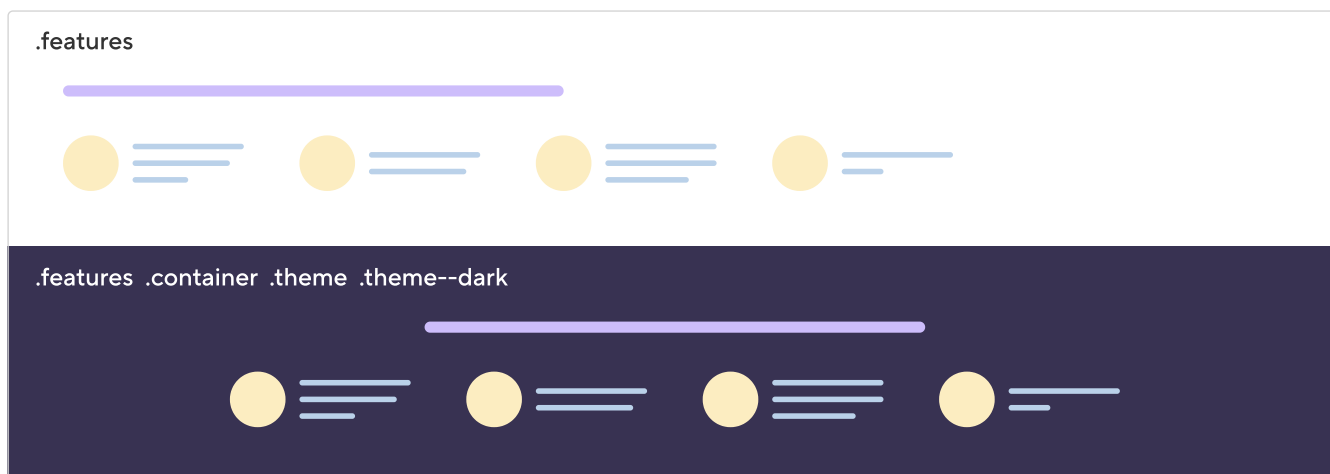
```

Шаг 6

Анализируем функциональность блока и следим за тем, чтобы он не стал «жадным».

Если класс для блока, помимо общих стилей, содержит декоративные стили, сеточные стили и другие, смотрим, можно ли поделить этот блок на несколько так, чтобы по возможности один блок отвечал за что-то одно (*один блок-одна функция*).

К примеру, создаём блок, который отвечает только за сетку; блок, который отвечает за внешний вид содержимого; блок, который отвечает за выравнивание. И всё это будут разные блоки.



Фрагмент типового макета с подписями: у одного блока несколько классов: features (базовый блок), container (центрирование контента на странице), theme и theme--dark (тёмная подложка)

```

<section class="features">
  <h2 class="features__title">Почему нас выбирают</h2>
  <ul class="features__list">
    <li class="features__item">Работаем по всей России</li>
    <li class="features__item">Строим по типовым и индивидуальным проектам

```

```
<li class="features__item">У нас работают профессиональные строители</li>
<li class="features__item">Выполняем витражное остекление</li>
</ul>
</section>

...

<section class="features container theme theme--dark">
  <h2 class="features__title">Что вы получаете, если выберете нас</h2>
  <ul class="features__list">
    <li class="features__item">Низкие цены и широкий ассортимент</li>
    <li class="features__item">100 лет – средний срок службы дома</li>
    <li class="features__item">Поддержка 24/7</li>
    <li class="features__item">Выгодные расценки и система скидок</li>
  </ul>
</section>
```

Руководствуемся правилом: «Лучше добавить лишние классы, а потом удалить, если они не понадобятся».

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

⚠ Если вы обнаружили ошибку или неработающую ссылку, выделите ее и нажмите Ctrl + Enter

Поиск по материалам

Git

[Все материалы](#)

В самом начале



- ☐ Пройдите опрос
- ☐ Укажите персональные данные
- ☐ 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](#)



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

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

Информация

Об Академии

О центре карьеры

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Услуги

Работа наставником

Для учителей

Стать автором

Остальное

Написать нам

Мероприятия

Форум

