

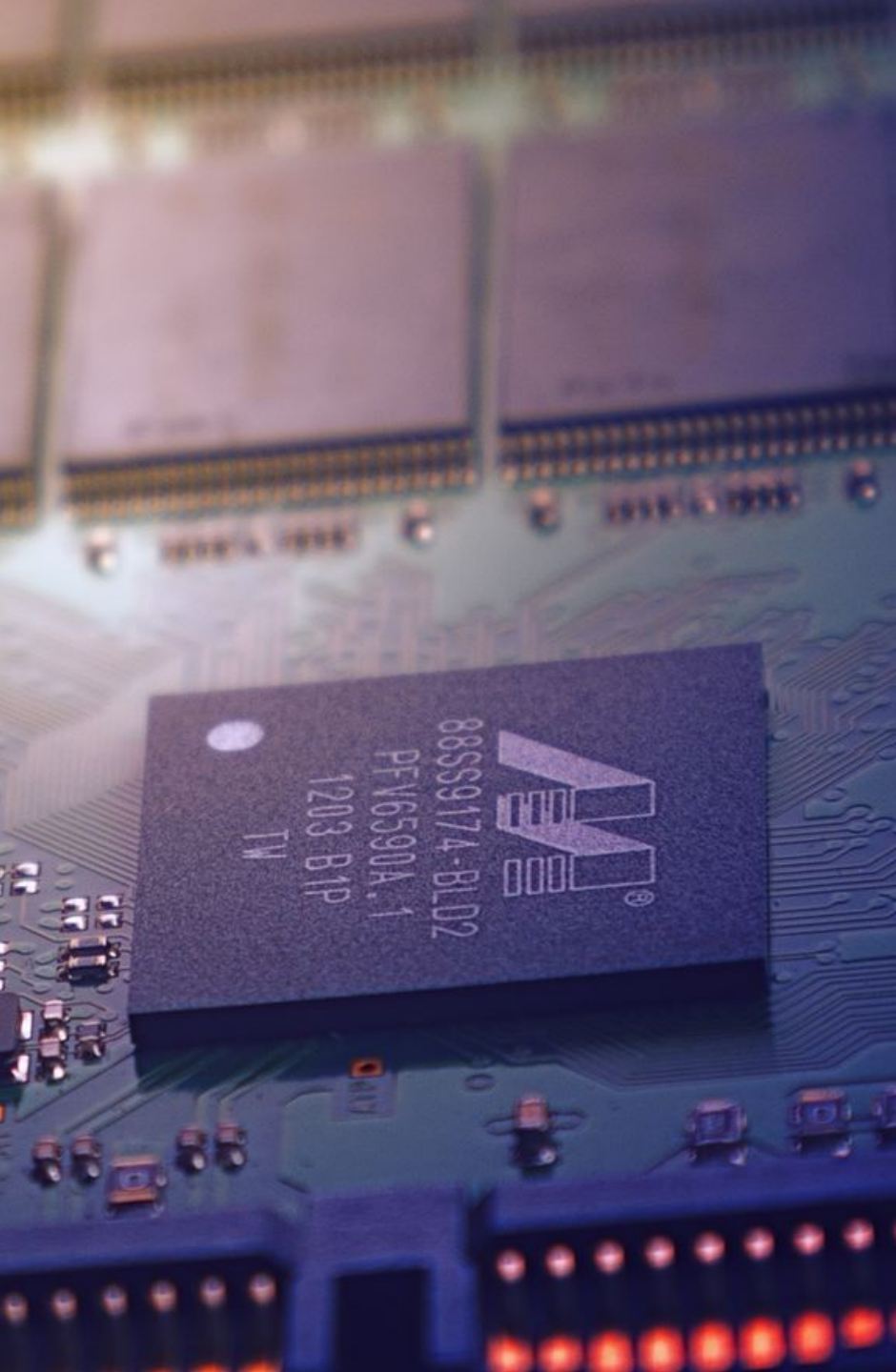
Разработка сложных компонентов

—
деливерим быстро, поддерживаем легко

We are Lamoda



- Ведущий fashion e-commerce России
- 10 млн. пользователей
- 100 внутренних систем
- Наш фронтенд — это онлайн-магазин

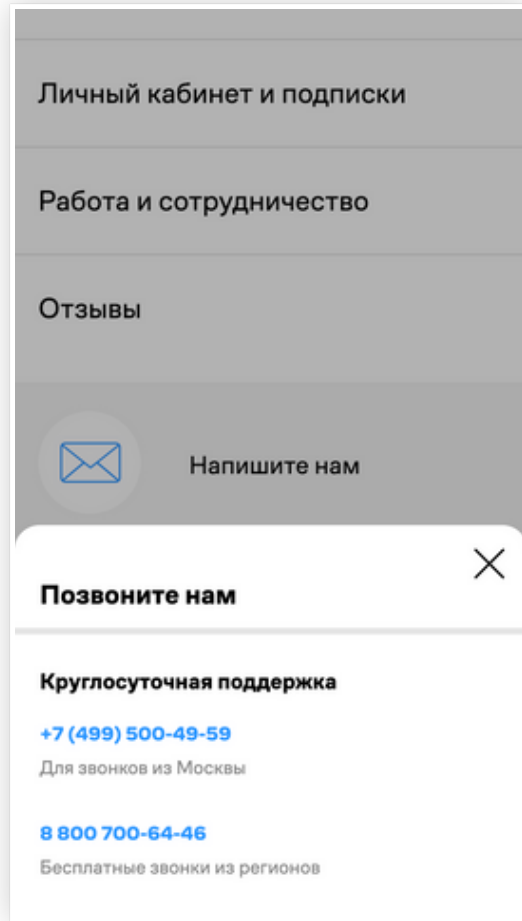


Фронтенд в Lamoda

- desktop site / m.-site / WebView / баннеры / лендинги
- 12 фронтенд-разработчиков
- Разрабатываем UI, используя компоненты
- Среди челленджей — сложные компоненты

Какой компонент
считать **сложным**?

Какой компонент считать сложным?



- Обычное модальное окно
- Разработка заняла 3 месяца
- 3 урока (3 принципа)

Навуевертил! или Disclaimer



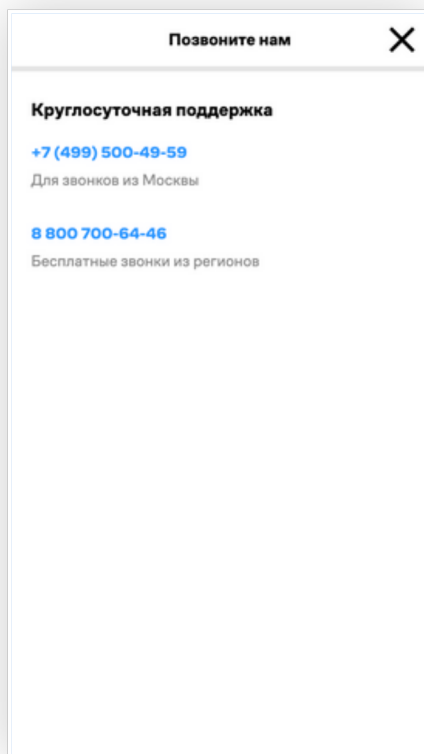
- Примеры кода на Vue.js
- Принципы универсальны

Принцип #1

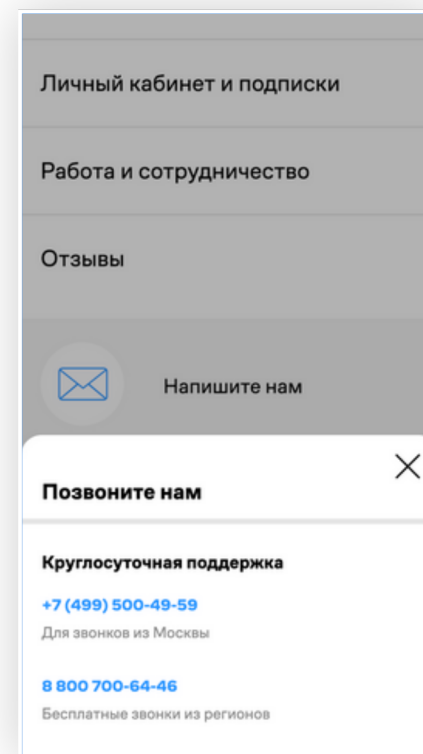
Компонент — это песочница

Принцип #1: Компонент — это песочница

Задача: А/В тест моб. модала



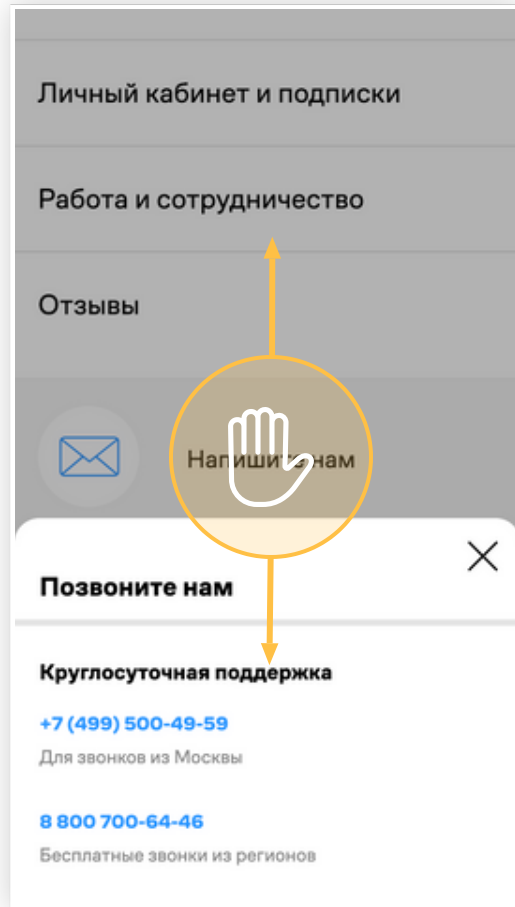
A (old)



B (new)

Принцип #1: Компонент — это песочница

Как будем делать свайп?



- Готовых компонентов для Vue нет
- Hammer.js не умеет следить за пальцем
- Делать с нуля не хочется

Выход: Swiper.js

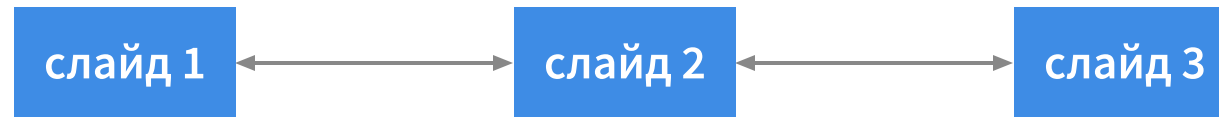


 23,365 stars 8,617 forks

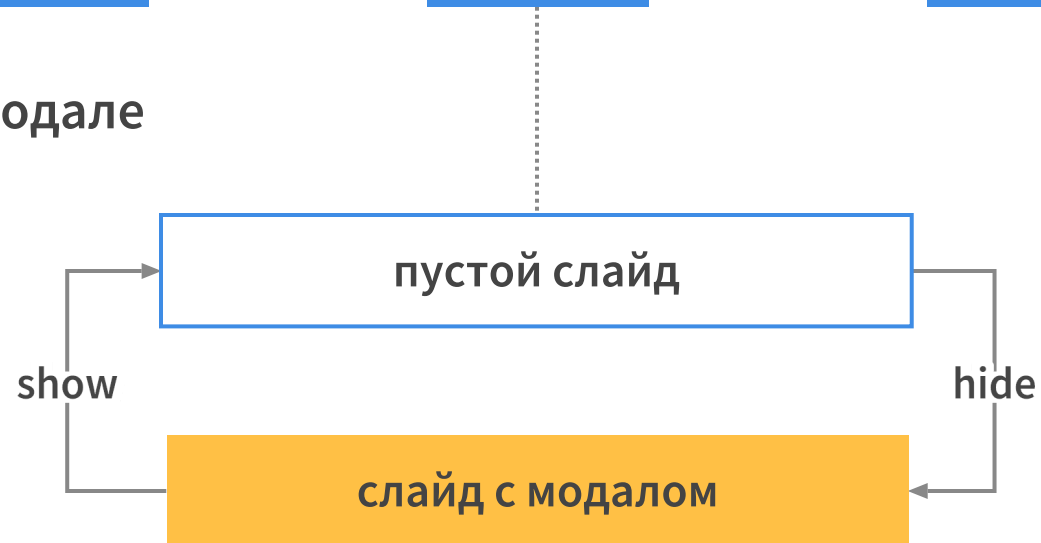
- Популярный touch-слайдер
- Уже используется для галерей
- `followFinger: true`

Выход: Swiper.js

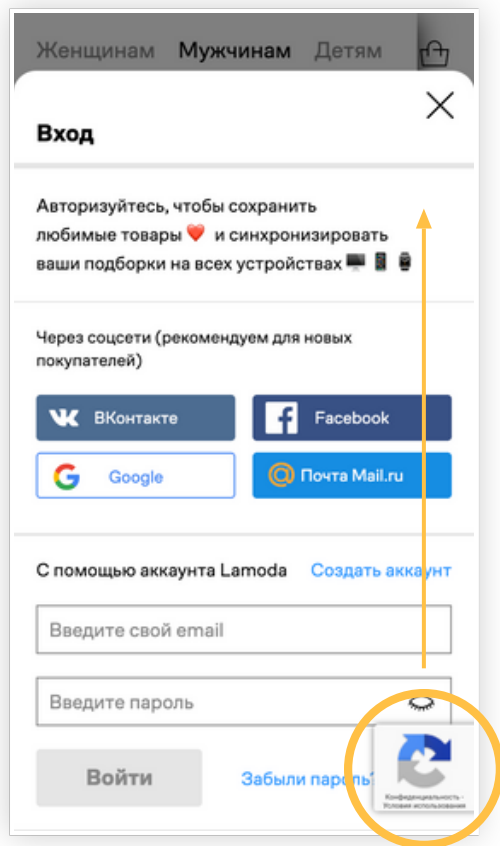
- Обычное применение (галерея)



- Применение в модале



Работает, но есть проблемы



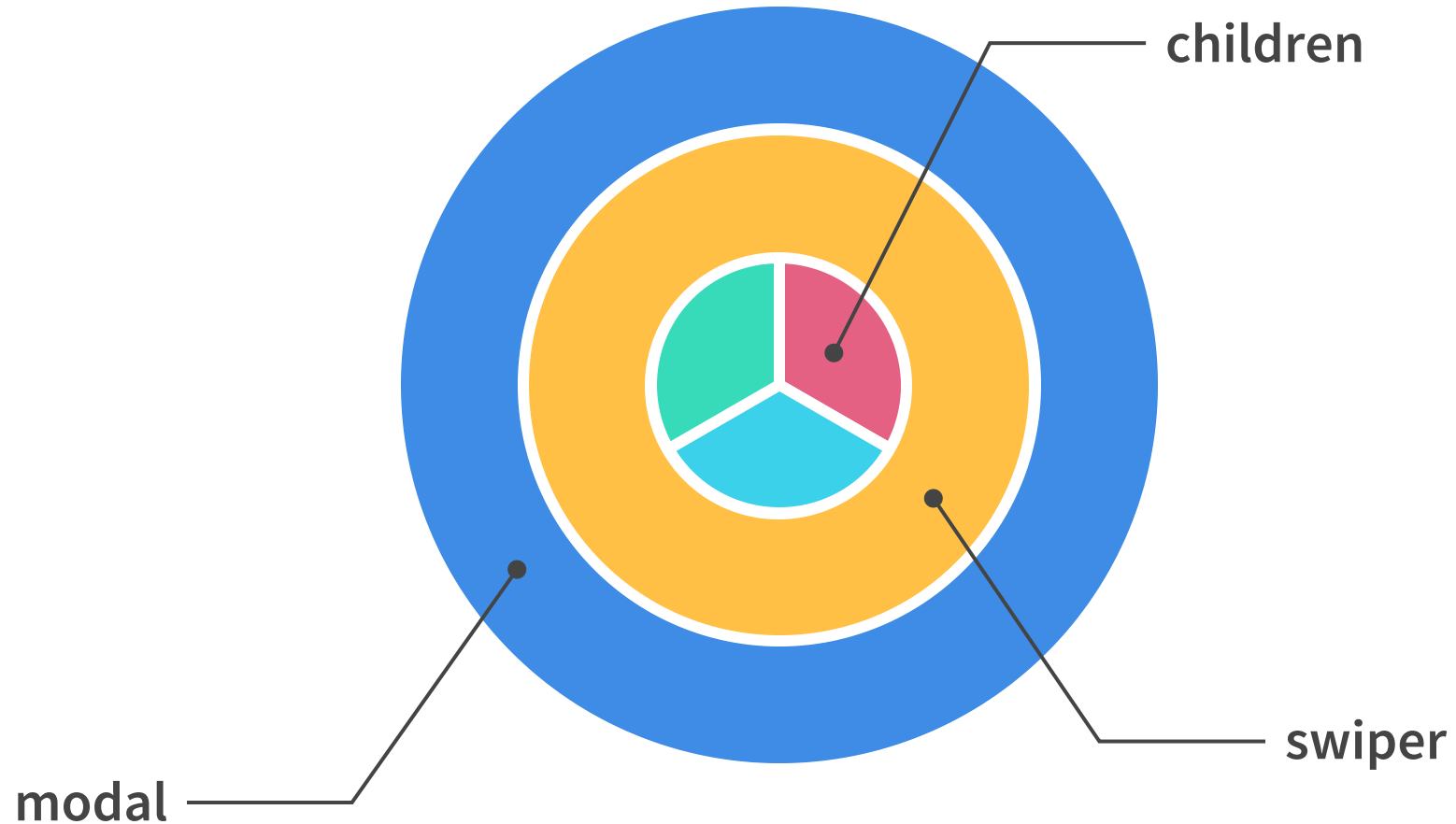
- reCAPTCHA использует position: fixed
- Swiper использует transform на слайдах
- Swiper теперь — containing block для капчи
- Результат: капча может “улететь”

Работает, но есть проблемы



- Галерея тоже использует Swiper
- Внутри модала свайп выключен
- Результат: свайп в галерее не работает

Swiper — небезопасная среда для children



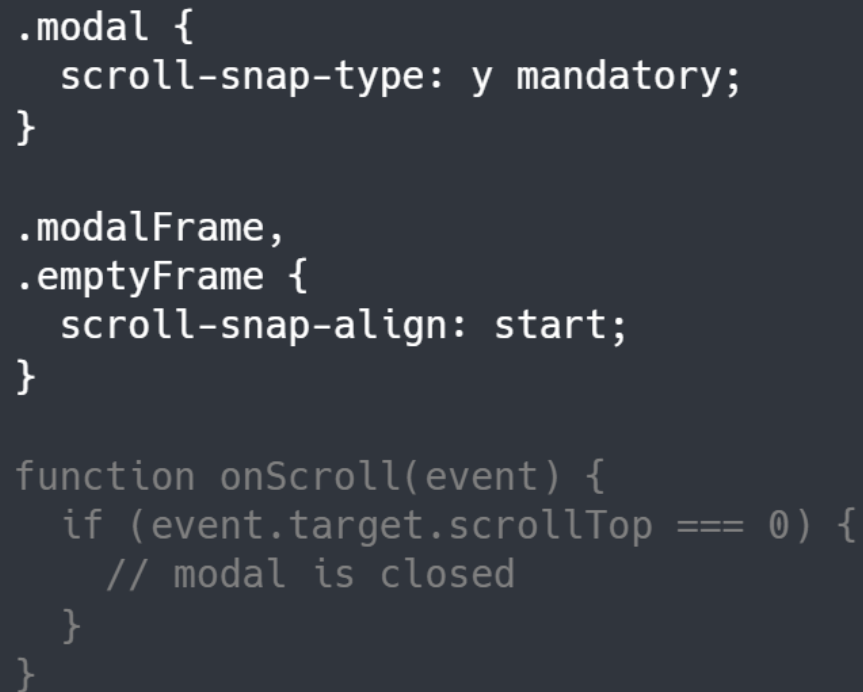
Альтернатива: CSS Scroll Snap



iOS Safari *	Chrome for Android	UC Browser for Android	Samsung Internet
12.4			10.1
13.3	80	12.12	11.1
13.4			

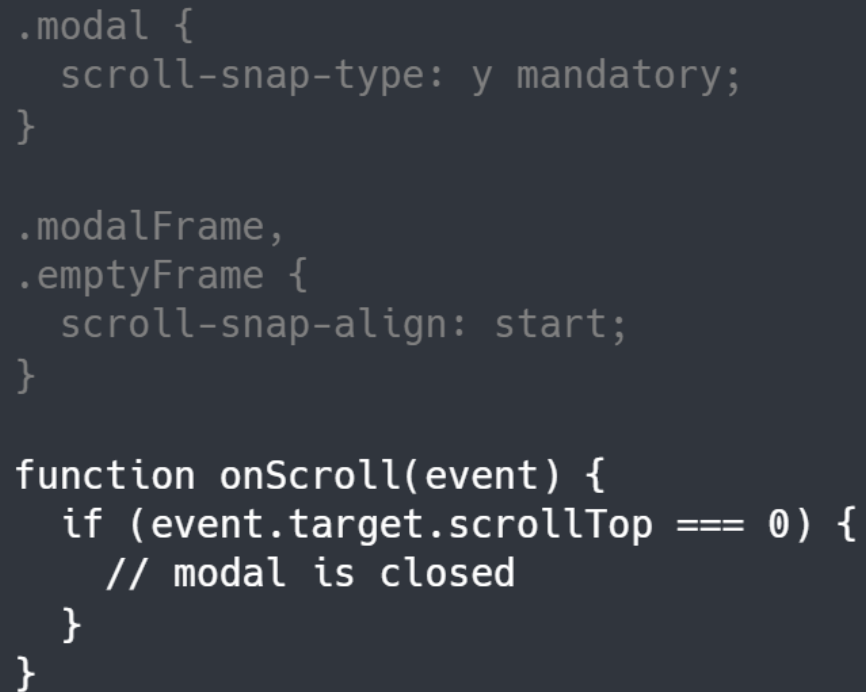
- Целиком нативное решение
- Поддерживается практически везде
- В нашем проекте — на стадии тестирования
- Ждем свободы от сайд-эффектов

Альтернатива: CSS Scroll Snap



```
.modal {  
  scroll-snap-type: y mandatory;  
}  
  
.modalFrame,  
.emptyFrame {  
  scroll-snap-align: start;  
}  
  
function onScroll(event) {  
  if (event.target.scrollTop === 0) {  
    // modal is closed  
  }  
}
```

Альтернатива: CSS Scroll Snap



```
.modal {  
  scroll-snap-type: y mandatory;  
}  
  
.modalFrame,  
.emptyFrame {  
  scroll-snap-align: start;  
}  
  
function onScroll(event) {  
  if (event.target.scrollTop === 0) {  
    // modal is closed  
  }  
}
```

Компонент — это песочница

- Не должно быть сайд-эффектов для children
- Промежуточный слой между компонентом и children — зло

Компонент — это песочница

- **Плюсы**

- Компонент проще переиспользовать
- Надежный UI kit

- **Минусы**

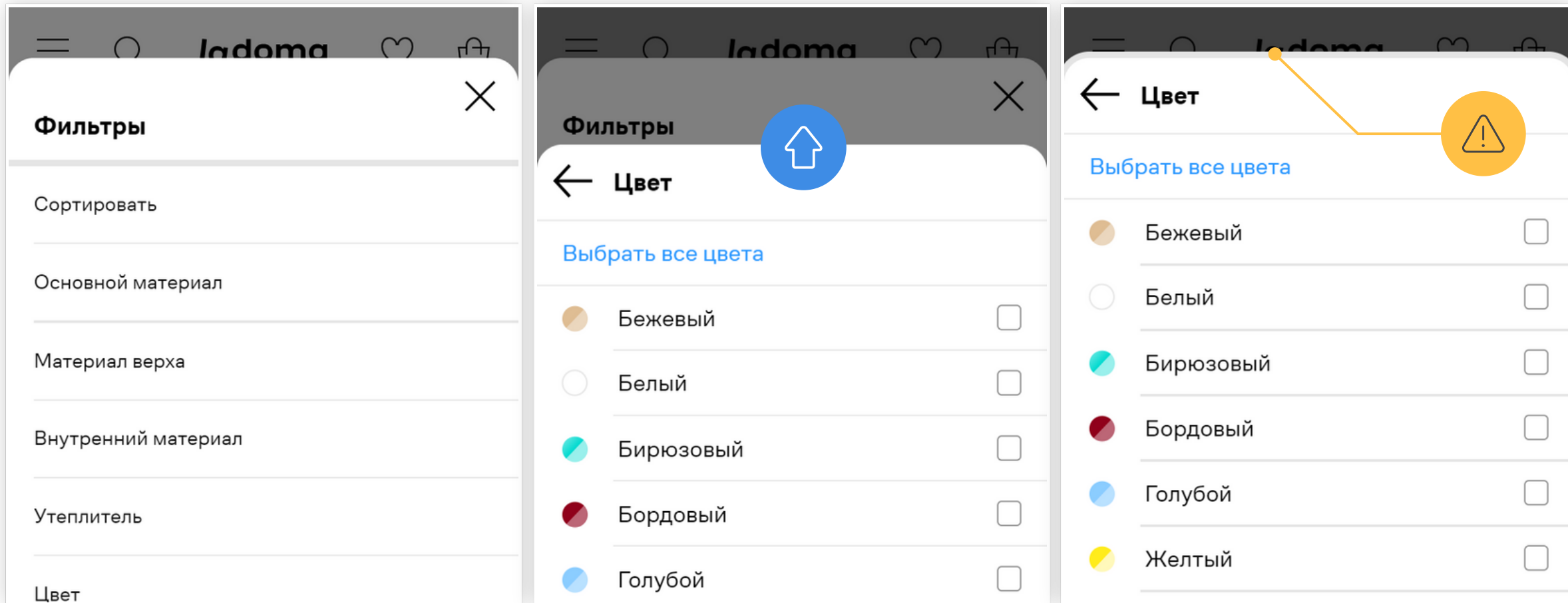
- “Чистое” решение — не всегда на поверхности
- Могут пострадать функции самого компонента

Принцип #2

Fancy-фича может подождать

Принцип #2: Fancy-фича может подождать

Фича: индикатор перекрытия модалов



Модалу стало сложнее

- Нужно следить за порядком модалов
- Нужно узнавать высоту контента
- Нужно создавать стор для данных
- Нужно вычислять видимость

...А местами вообще не работает



- Индикатор сделан как ::before
- overflow: hidden обрезает углы фото
- Заодно обрезается и индикатор
- Делать надежно — значит усложнить верстку

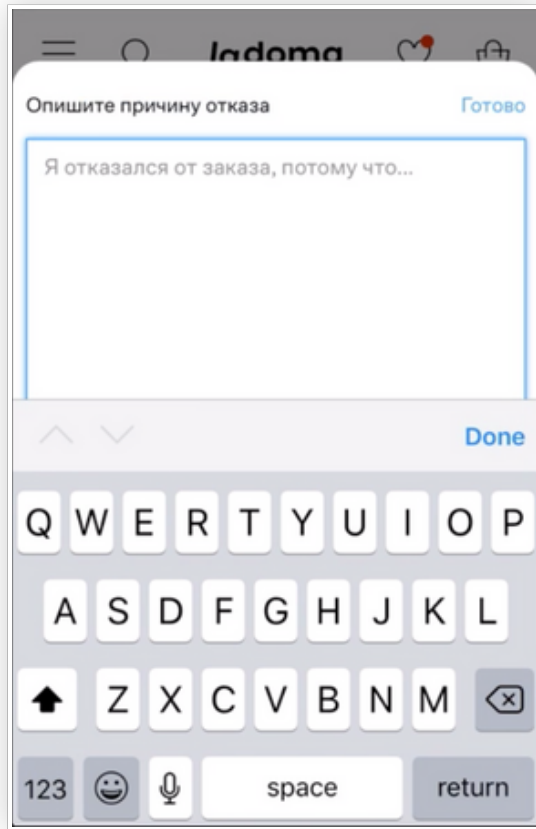


Принцип #2: Fancy-фича может подождать

Релиз отложен... а стоило ли?

- Индикатор — полезный, но некритичный UI-элемент
- Челленджи в плане реализации

Урок был усвоен



- Фича: закрывать модал по кнопке done
- Как отследить нажатие?
- На Android кнопки вообще нет
- В итоге добавили кнопку в UI

Ғансу-фича может подождать

- Вовремя обнаруживаем ғансу-фичу в компоненте
- Ищем компромисс в реализации

Fancy-фича может подождать

- **Плюсы**

- быстрее релиз компонента
- проще поддержка

- **Минусы**

- недовольный дизайнер

Принцип #3

Не объединяй и властвуй



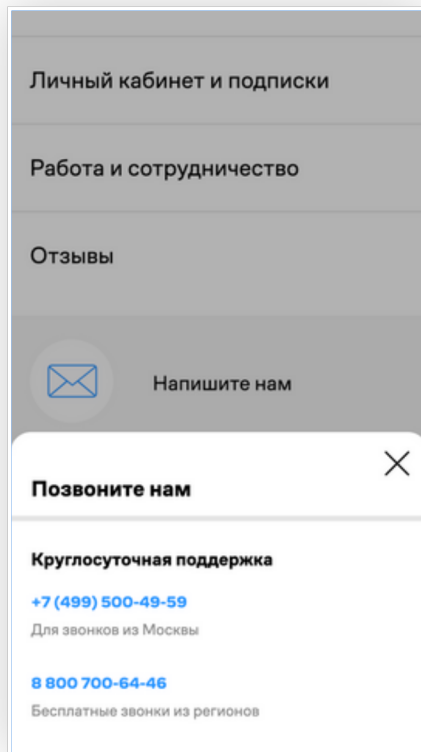
Принцип #3: Не объединяй и властвуй

“Модалка может все!”

- Старый + новый дизайн (А/В)
- Новые крутые фишки
- Возможность кастомизации
- Лишние ответственности
- 15 props / 5 slots / 700 lines

Принцип #3: Не объединяй и властвуй

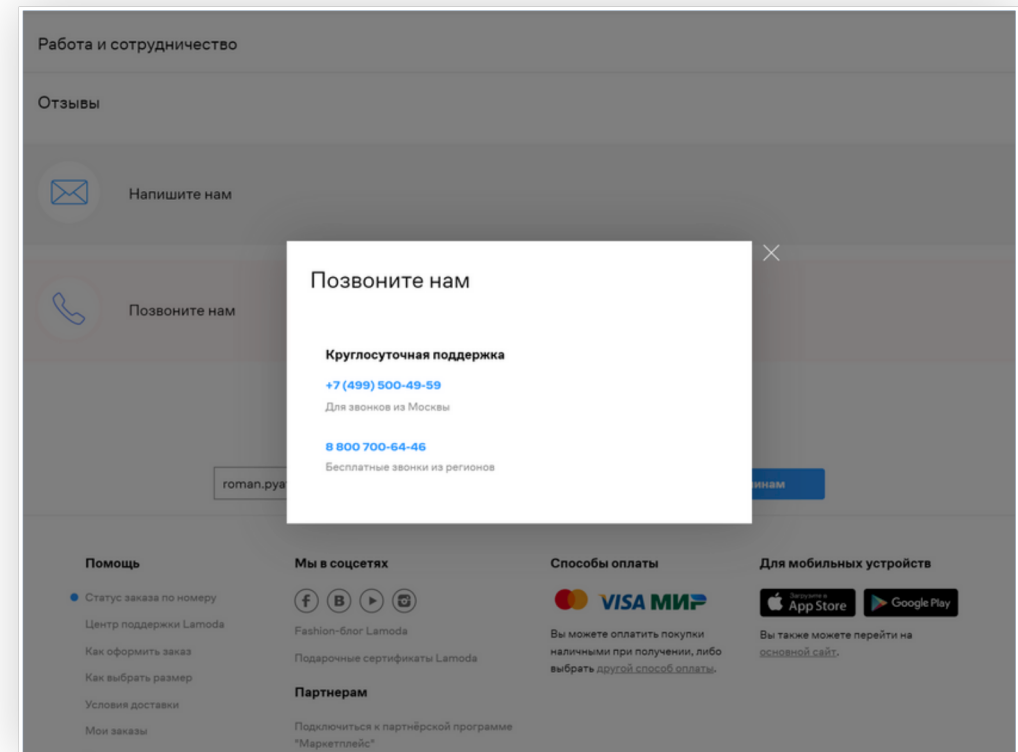
...И может стать еще больше



mobile (m-modal)



responsive (x-modal)

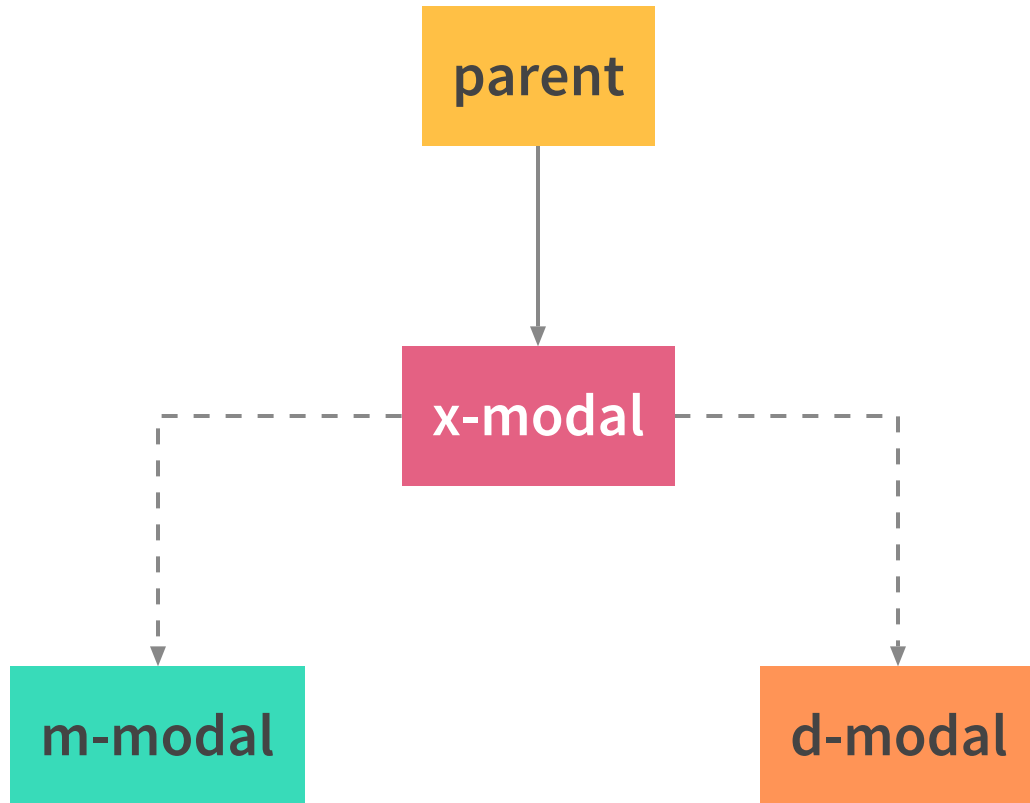


desktop (d-modal)

Перспектива: компонент-монстр

- Еще + 50% к объему кода
- Обмажемся if'ами
- Будем перебивать мобильные стили
- Конфликт между командами `mobile vs. desktop`

А что если не объединять?



- Оставить компоненты жить отдельно
- Перейти на общий API
- Переключаться на нужную реализацию автоматически

А что если не объединять?

```
import MModal from 'components/m-modal/m-modal.vue';
import DModal from 'components/d-modal/d-modal.vue';
import deviceProps from 'utils/device-props';

const XModal = {
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      deviceProps.screenSize === 'phone' ? MModal : DModal,
      context.data,
      context.children,
    );
  },
};
```

А что если не объединять?

```
import MModal from 'components/m-modal/m-modal.vue';
import DModal from 'components/d-modal/d-modal.vue';
import deviceProps from 'utils/device-props';

const XModal = {
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      deviceProps.screenSize === 'phone' ? MModal : DModal,
      context.data,
      context.children,
    );
  },
};
```

А что если не объединять?

```
import MModal from 'components/m-modal/m-modal.vue';
import DModal from 'components/d-modal/d-modal.vue';
import deviceProps from 'utils/device-props';

const XModal = {
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      deviceProps.screenSize === 'phone' ? MModal : DModal,
      context.data,
      context.children,
    );
  },
};
```

А что если не объединять?

```
import MModal from 'components/m-modal/m-modal.vue';
import DModal from 'components/d-modal/d-modal.vue';
import deviceProps from 'utils/device-props';

const XModal = {
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      deviceProps.screenSize === 'phone' ? MModal : DModal,
      context.data,
      context.children,
    );
  },
};
```


Тот случай, когда копипаста не напрягает



Адаптивный UI

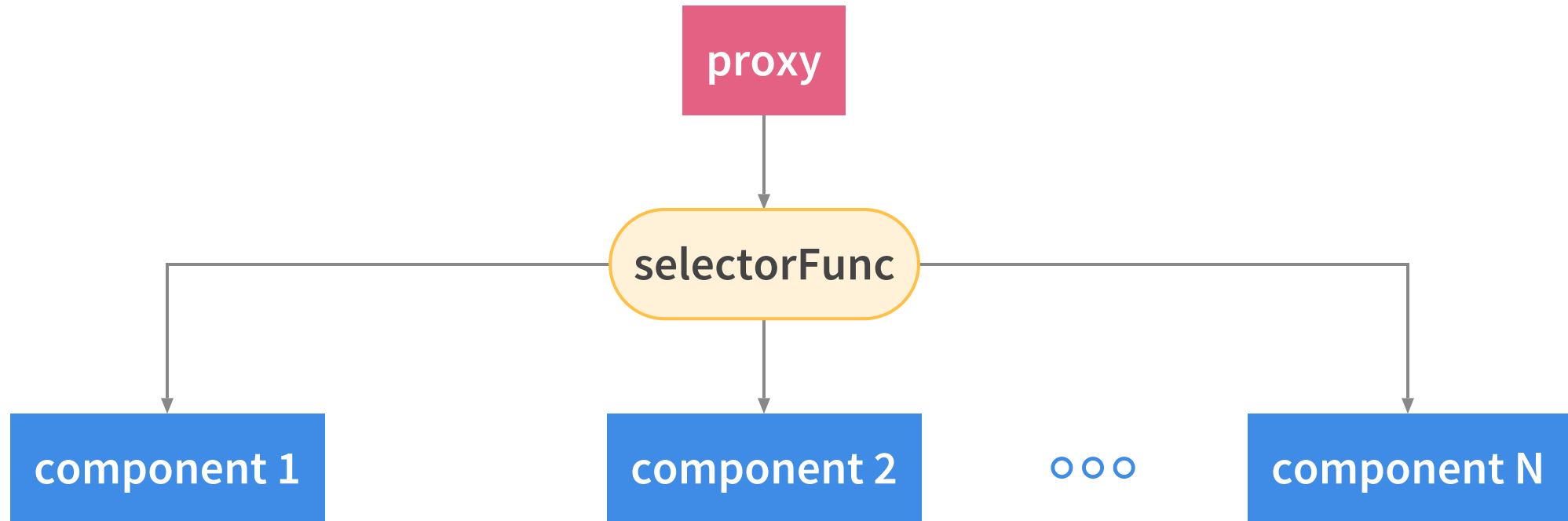


A/B тестирование




Несколько платформ
(окружений)

Общее решение: проху-компонент



Паттерн проектирования Proху (Заместитель)

Общее решение: проху-компонент



```
const createProxyComponent = selectorFunc => ({
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      selectorFunc(),
      context.data,
      context.children,
    );
  },
});
```

Принцип #3: Не объединяй и властвуй

Общее решение: проху-компонент



```
const createProxyComponent = selectorFunc => ({
  functional: true,
  render(createElement, context) {
    return context.parent.$createElement(
      selectorFunc(),
      context.data,
      context.children,
    );
  },
});
```

Не объединяй и властвуй

- Разные реализации компонента живут отдельно
- Паттерн Proxu помогает склеить части

Не объединяй и властвуй

- **Плюсы**

- Не нужно путаться в условиях
- Проще работать независимо

- **Минусы**

- Копипаста
- Проблемы с тулзами

Выводы

Выводы

- **Сложным может оказаться любой компонент**
- **Три полезных принципа**
 - Компонент — это песочница
 - Fancy-фича может подождать
 - Не объединяй и властвуй
- **Быстрее ТТМ и легче поддерживать код**



Роман Пятаков

Frontend Tech Lead / Lamoda



romankipper

Спасибо за внимание! :)