

Как поставить разработку фронтенда на поток



АНТОН ЖУКОВ

https://t.me/mister_cheater

Дисклеймер

В ходе своего рассказа, я буду предлагать очень радикальные вещи.

Смысл моих предложений - решить проблемы бизнеса. Задетые чувства фронтендеров будут, но тут они не очень важны.

Как устроена галера: есть пулл проектов и пулл разработчиков. Разработчиков нужно кидать туда сюда, оптимизируя издержки, пока согласуются бюджеты проектов.

Структура повествования

1. **Коммерческая разработка и интересы бизнеса**
2. Как достигается надежность
3. Выбор фронтенд фреймворка
4. Архитектура фронтенда

Коммерческая разработка и интересы бизнеса

Соотношение
цена/качество,
трудозатраты/качество
должно быть приемлемым

Исправленные баги не
должны возвращаться

Нужна предсказуемость
издержек

Рефакторинг и новые
фичи не должны ломать
старый функционал

Баг в продакшене может принести
финансовый ущерб клиенту и
испортить нашу репутацию

Стандартизация и
унификация всего

Сопровождение и поддержка
должны быть простыми и
требующими минимум
ресурсов

Репутация – очень большая
ценность. Наши продукты не
должны расстраивать
клиентов

Минимизация роли конкретного
человека в процессе разработки
(управление рисками)

Структура повествования

1. Коммерческая разработка и интересы бизнеса
2. **Как достигается надежность**
3. Выбор фронтенд фреймворка
4. Архитектура фронтенда

Как достигается надежность?



Как достигается надежность?

<TechLeader>

Я лично гарантирую всё

Как достигается надежность?

<TechLeader>

Лично гарантирует всё,
**внедрил инструменты
статического анализа**

Static analysis

TypeScript дает статические проверки типов, позволяет выявлять ошибки в compile-time, а не в runtime

Prettier обеспечивает единый Code Style на уровне команды

ESLint дает статические проверки и рекомендации по Code Style

Зачем нужен Prettier



<TechLeader проекта>

Молю 🙏 давай отступы делать между объектами с стор

```
export default {
  setCustomerAddressesLoading() {},

  setCustomerAddressesItems() {},

  markCustomerAddressForDeletion() {},

  undoCustomerAddressDeletion() {},

  deleteCustomerAddress() {},

  setSelectedCustomerAddress() {},
};
```

Постараюсь добавить правило в линтер



<Разработчик>

Пофиксил !88 (0703252f)



<TechLeader проекта>

И тут отформатируй, пожалуйста

Suggested change

Apply suggestion

```
46 - disabled() { return this.state === CHECKOUT_STEP_STATE.DISABLED; },
47 - opened() { return this.state === CHECKOUT_STEP_STATE.OPENED; },
48 - filled() { return this.state === CHECKOUT_STEP_STATE.FILLED; },
46 + disabled() {
47 +   return this.state === CHECKOUT_STEP_STATE.DISABLED;
48 + },
50 + opened() {
51 +   return this.state === CHECKOUT_STEP_STATE.OPENED;
52 + },
53 + filled() {
54 +   return this.state === CHECKOUT_STEP_STATE.FILLED;
55 + },
56 +
```

changed this line in [version 7 of the diff](#) 1 month ago



<Разработчик>

Отформатировал

Зачем нужен ESLint

- Найти существующие ошибки в коде;
- Избежать глупых ошибок;
- Избежать бесконечные циклы в условиях цикла for;
- Убедится, что все методы getter возвращают что-то;
- Не разрешить выражения console.log (и аналогичные);
- Проверить наличие дубликатов cases в switch;
- Устранить недоступный код;

```
$ ./node_modules/.bin/eslint "**/*.js,jsx"
```

```
/Users/chamilton/Development/Node/sandbox/src/index.js
```

```
1:1 error    Unexpected var, use let or const instead      no-var
2:1 warning  Unexpected console statement                  no-console
4:9 error    'message' is never reassigned. Use 'const' instead  prefer-const
```

```
* 3 problems (2 errors, 1 warning)
```

```
2 errors and 0 warnings potentially fixable with the `--fix` option.
```

Зачем нужен TypeScript

billing	Billing ▾ {
lastname	string nullable: true
firstname	string nullable: true
telephone	string nullable: true
email	string nullable: true
legal_type	integer
	Тип плательщика:
	<ul style="list-style-type: none"> 1 - Юридическое лицо 2 - Физическое лицо 3 - ИП
	Enum:
	<ul style="list-style-type: none"> > Array [3]
has_proxy	boolean nullable: false
	If has proxy
inn	string nullable: true

Swagger

```
import { z } from "zod";

const Billing = z.object({
  lastname: z.string().nullable(),
  firstname: z.string().nullable(),
  telephone: z.string().nullable(),
  email: z.string().nullable(),
  legal_type: z.enum([1, 2, 3]),
  has_proxy: z.boolean(),
  inn: z.string().nullable(),
  /* ... */
});
```

Код на TypeScript

Зачем нужен TypeScript

mode text/javascript

CodeMirrorEditor.vue:87

- 6 ▶ Uncaught TypeError: Cannot read properties of undefined (reading 'map') [codemirror.js:2412](#)
- at prepareMeasureForLine ([codemirror.js:2412:17](#))
 - at coordsCharInner ([codemirror.js:2758:27](#))
 - at coordsChar ([codemirror.js:2726:19](#))
 - at posFromMouse ([codemirror.js:2977:18](#))
 - at CodeMirror.onMouseDown ([codemirror.js:7353:15](#))
 - at HTMLDivElement.<anonymous> ([codemirror.js:3962:22](#))
- 4 ▶ Uncaught TypeError: Cannot read properties of undefined (reading 'map') [codemirror.js:2412](#)
- at prepareMeasureForLine ([codemirror.js:2412:17](#))
 - at cursorCoords ([codemirror.js:2662:47](#))
 - at drawSelectionCursor ([codemirror.js:3178:15](#))
 - at prepareSelection ([codemirror.js:3168:9](#))
 - at TextareaInput.prepareSelection ([codemirror.js:9457:18](#))
 - at endOperation_R2 ([codemirror.js:3884:46](#))
 - at endOperations ([codemirror.js:3842:9](#))
 - at [codemirror.js:3829:7](#)
 - at finishOperation ([codemirror.js:2058:7](#))
 - at endOperation ([codemirror.js:3826:15](#))

>

Как достигается надежность?

<TechLeader>

Проводит Code review.

Лично гарантирует
многое, но уже **внедрил
инструменты**
статического анализа,
**мониторинга ошибок в
рантайме и мониторинг
производительности**

Monitoring and Observability

Frontend Monitoring позволяет поймать ошибки, которые не заметили на этапе написания кода. Инструменты (Sentry, LogRocket, DataDog)

Необходимо ловить ошибки, как на уровне компонентов, так и на уровне менеджера состояния

Performance Monitoring позволяет обнаружить регрессии в производительности при разработке. Инструменты (Lighthouse CI, Perfume.js)

Как достигается надежность?

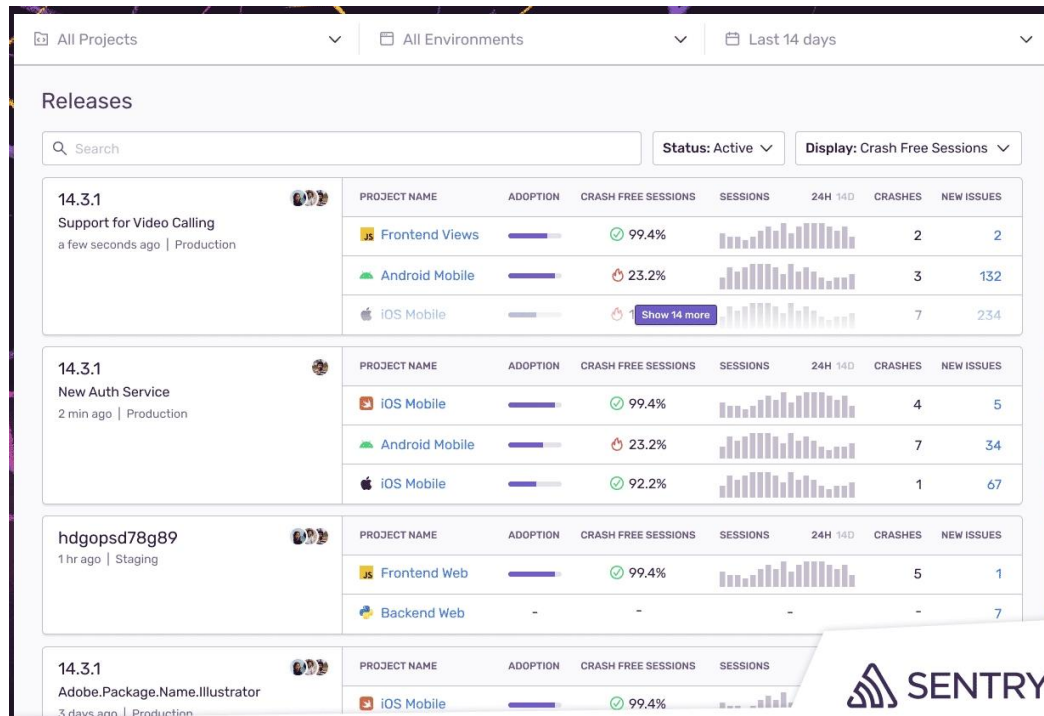
Frontend Monitoring позволяет поймать ошибки, которые не заметили на этапе написания кода. Инструменты (Sentry, LogRocket, DataDog)

Необходимо ловить ошибки, как на уровне компонентов, так и на уровне менеджера

```

<App >
  <Header />
  <LeftSideBar />
  <ErrorBoundary>
    <Main />
  </ErrorBoundary>
</App>

```



Как достигается надежность?

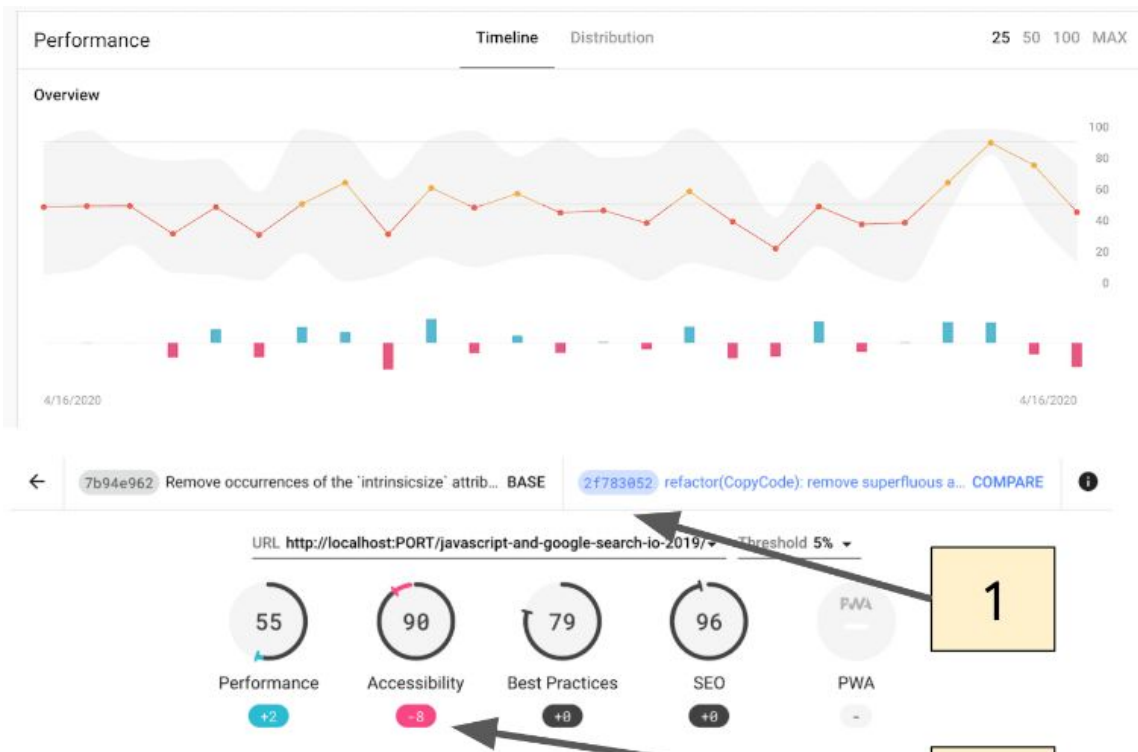
Performance Monitoring
позволяет обнаружить
регрессии в
производительности при
разработке. Инструменты
(Lighthouse CI, Perfume.js)

```
✓ .lighthouseci/ directory writable
✓ Configuration file found
✓ Chrome installation found
△ GitHub token not set
Healthcheck passed!
```

```
Automatically determined .dist as 'staticDistDir'.
Set it explicitly in lighthouseci.json if incorrect.
```

```
Started a web server on port 64444...
Running Lighthouse 5 time(s) on http://localhost:64444/index.html
Run #1...done.
Run #2...done.
Run #3...done.
Run #4...done.
Run #5...done.
Done running Lighthouse!
```

```
Uploading median LHR of http://localhost:64444/index.html...success!
```



Как достигается надежность?

<TechLeader>

Проводит Code review.

Опирается на
инструменты

статического анализа,
мониторинга ошибок в
рантайме и мониторинг
производительности.

**Проверяет, что баги
фиксируются с тестами**

Automated tests

Unit-tests проверяют, что конкретные кусочки
кода корректно работают в конкретных
сценариях

- В процессе разработки в сложных компонентах можно использовать для того, чтобы не ошибиться
- **Для устранения регрессий после фикса багов.** Если пофиксить баг и написать тест, то он уже просто так никогда не вернется


```
describe('method "setSelectedShippingMethod" should work correctly', () :void => {  
  test('when methods = [delivery,pickup], selectedShippingMethod = delivery', () :void => {...});  
  
  test('when methods = [delivery], selectedShippingMethod = pickup', () :void => {  
    const state : {shippingMethods: [{type: string, title: string}...] = {  
      shippingMethods: [  
        {  
          type: 'delivery',  
          title: 'Доставка',  
        },  
      ],  
    };  
  
    mutations.setSelectedShippingMethod(state, selectedShippingMethod: 'pickup');  
  
    expect(state.selectedShippingMethod).toEqual( expected: 'delivery');  
  });  
});
```

Как достигается надежность?

<TechLeader>

Проводит Code review.

Опирается на автоматизированные инструменты: E2E-тесты, статический анализ, мониторинг ошибок в рантайме, мониторинг производительности. Проверяет, что баги фиксируются с тестами

Automated tests

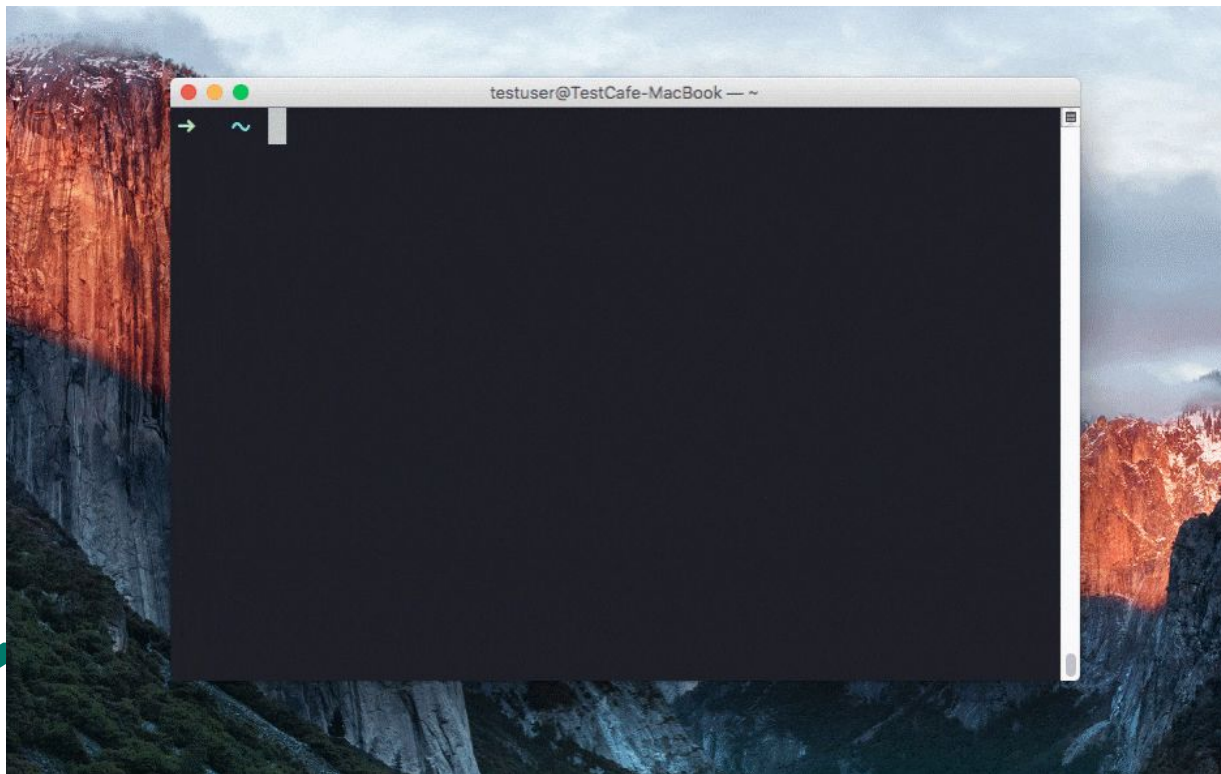
E2E-tests проверяют, что приложение в целом работает. И корректно отрабатывают какие-то пользовательские сценарии.

Пример: пользователь как ИП зашел на чекаут, ввёл адрес, заполнил личные данные, дал согласие на их обработку, выбрал метод оплаты и завершил заказ

Как достигается надежность?

Automated tests

E2E-tests проверяют, что приложение в целом работает. И корректно обрабатывают какие-то пользовательские сценарии.



Как достигается надежность?

Manual tests

<TechLeader>

Проводит Code review.




Опирается на автоматизированные инструменты: E2E-тесты, статический анализ, мониторинг ошибок в рантайме, мониторинг производительности.

Проверяет, что баги фиксируются с тестами

<QA>

Обеспечивает финальный контроль качества

Как достигается надежность?

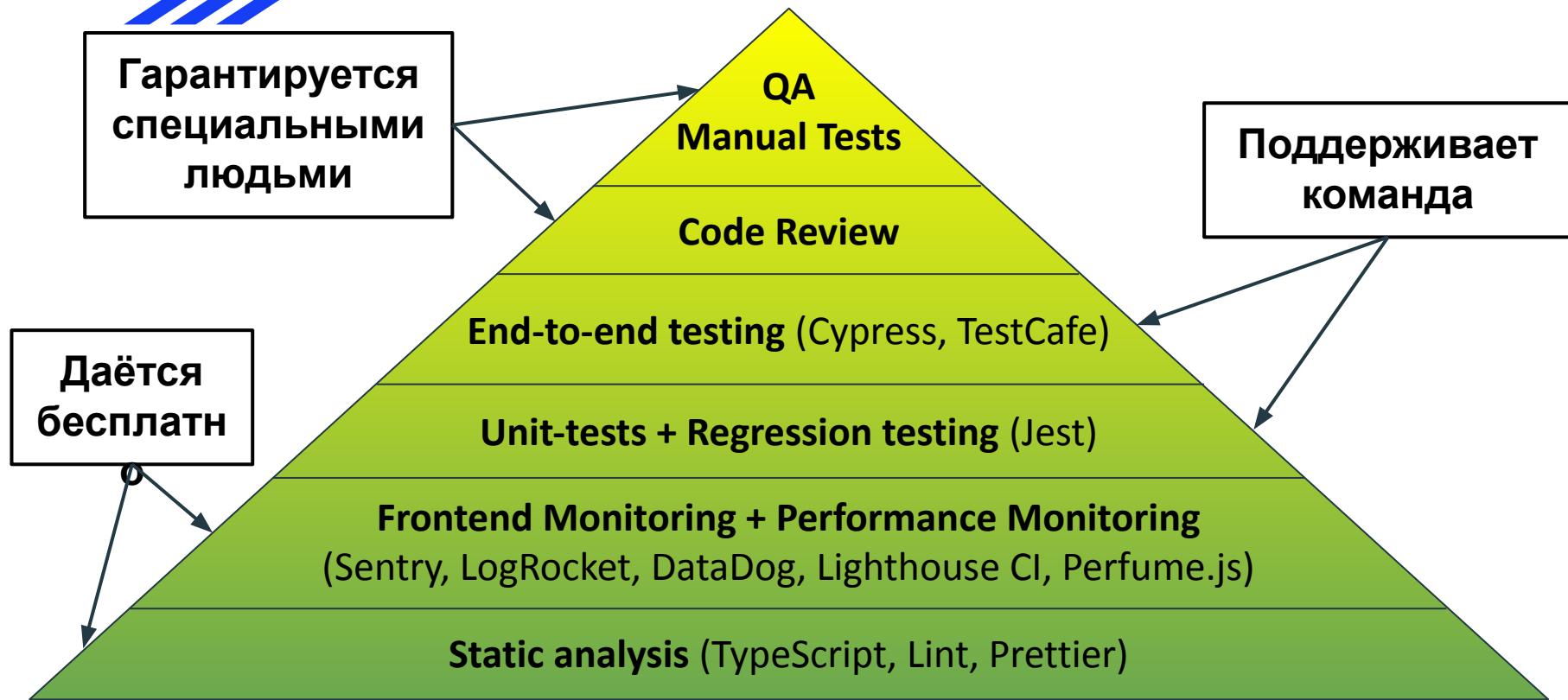
3 checks passed		
✓	 Static Tests (ESLint, Prettier)	Details
✓	 Unit Tests (Jest)	Details
✓	 E2E Tests (TestCafe)	Details

VS

<TechLeader>

Я лично гарантирую всё

Как достигается надежность?



Как достигается надежность?

Чтобы двигаться дальше в сторону повышения надежности и минимизации издержек, необходимо для наших целевых фреймворков создать эталонные шаблоны, где всё это преднастроено, использовать эти шаблоны для новых проектов.

Так же в шаблон должен входить Storybook, менеджер состояния, настройка стилей, шрифтов, иконок, картинок и т.д.



End-to-end testing (Cypress, TestCafe)

Unit-tests + Regression testing (Jest)

Frontend Monitoring + Performance Monitoring
(Sentry, LogRocket, DataDog, Lighthouse CI, Perfume.js)

Static analysis (TypeScript, Lint, Prettier)

Как достигается надежность?

Чтобы двигаться еще дальше в сторону повышения надежности и минимизации издержек, необходимо вынести переиспользуемую логику в независимые NPM-пакеты.

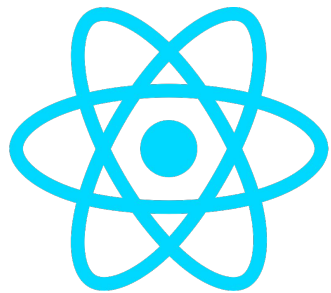
Что может входить в такие пакеты:

- Работа Magento через GraphQL; (и прочими фреймворками)
- Специфичные для русского ECommerce вещи типа валидации ИНН, БИК, Р/С, email, телефона;
- Работа с адресами;
- Работу с картой (Yandex Map, OpenStreetMap);
- Работа с Dadata;
- Общие вещи типа механизма валидации;
- Набор анимаций;
- Renderless-компоненты не привязанные к дизайну – модалки, инпуты, радиогруппы, чекбокс группы, табы, степ-бай-степ компоненты.

Структура повествования

1. Коммерческая разработка и интересы бизнеса
2. Как достигается надежность
3. **Выбор фронтенд фреймворка**
4. Архитектура фронтенда

Выбор фронтенд фреймворка



React



Vue.js

NEXT



Vue 2 заканчивает свой жизненный цикл

How long will Vue 2 be supported?

Vue 2.7 is the current, and final minor release of Vue 2.x. Vue 2.7 receives 18 months of LTS (long-term support) starting from its release date on July 1st, 2022. During this period, Vue 2 will receive necessary bug and security fixes, but will no longer receive new features.

Vue 2 will reach End of Life (EOL) on December 31st, 2023. After that date, Vue 2 will continue to be available in all existing distribution channels (CDNs and package managers), but will no longer receive updates, including security and browser compatibility fixes.

Сторонние разработчики отказываются от поддержки Vue 2

Vue-yandex-maps

v1.x-beta Руководство Примеры API Я.Карт Язык GitHub

Создание карты

Маркер с кастомным балуном

Map

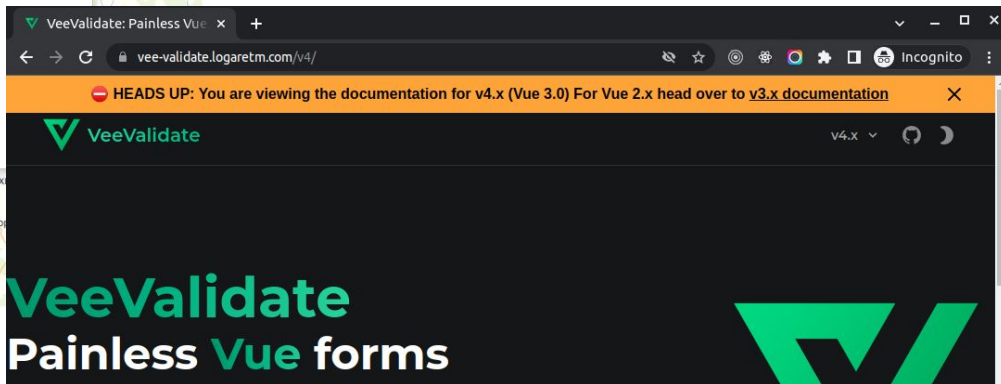
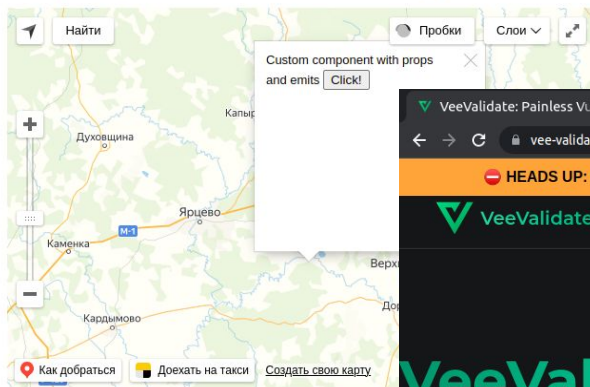
CustomBalloon

Перемещение маркера по клику

Использование кластерера

Использование коллекций

Маркер с кастомным балуном



Что команда разработки Vue думает про TypeScript

A type system like TypeScript can detect many common errors via static analysis at build time. This reduces the chance of runtime errors in production, and also allows us to more confidently refactor code in large-scale applications. TypeScript also improves developer ergonomics via type-based auto-completion in IDEs.

Vue is written in TypeScript itself and provides first-class TypeScript support. All official Vue packages come with bundled type declarations that should work out-of-the-box.

Почему стоит отказаться от Mixins

Mixins ломают вывод типов в TypeScript. По своей природе Mixins основаны на неявных мутациях, и типизации не поддаются. Есть альтернативные решения основанные на композиции, где с типизацией всё ок.

⚠ No Longer Recommended

In Vue 2, mixins were the primary mechanism for creating reusable chunks of component logic. While mixins continue to be supported in Vue 3, [Composition API](#) is now the preferred approach for code reuse between components.

Users coming from Vue 2 may be familiar with the [mixins](#) option, which also allows us to extract component logic into reusable units. There are three primary drawbacks to mixins:

1. **Unclear source of properties:** when using many mixins, it becomes unclear which instance property is injected by which mixin, making it difficult to trace the implementation and understand the component's behavior. This is also why we recommend using the refs + destructure pattern for composables: it makes the property source clear in consuming components.
2. **Namespace collisions:** multiple mixins from different authors can potentially register the same property keys, causing namespace collisions. With composables, you can rename the destructured variables if there are conflicting keys from different composables.
3. **Implicit cross-mixin communication:** multiple mixins that need to interact with one another have to rely on shared property keys, making them implicitly coupled. With composables, values returned from one composable can be passed into another as arguments, just like normal functions.

For the above reasons, we no longer recommend using mixins in Vue 3. The feature is kept only for migration and familiarity reasons.

Команда разработки Vue выбрала движение в сторону функциональности и композиции

Why Composition API?

Better Logic Reuse

The primary advantage of Composition API is that it enables clean, efficient logic reuse in the form of **Composable functions**. It solves **all the drawbacks of mixins**, the primary logic reuse mechanism for Options API.

Composition API's logic reuse capability has given rise to impressive community projects such as **VueUse**, an ever-growing collection of composable utilities. It also serves as a clean mechanism for easily integrating stateful third-party services or libraries into Vue's reactivity system, for example **immutable data**, **state machines**, and **RxJS**.

More Flexible Code Organization

Many users love that we write organized code by default with Options API: everything has its place based on the option it falls under. However, Options API poses serious limitations when a single component's logic grows beyond a certain complexity threshold. This limitation is particularly prominent in components that need to deal with multiple **logical concerns**, which we have witnessed first hand in many production Vue 2 apps.

Better Type Inference

In recent years, more and more frontend developers are adopting **TypeScript** as it helps us write more robust code, make changes with more confidence, and provides a great development experience with IDE support. However, the Options API, originally conceived in 2013, was designed without type inference in mind. We had to implement some **absurdly complex type gymnastics** to make type inference work with the Options API. Even with all this effort, type inference for Options API can still break down for mixins and dependency injection.

This had led many developers who wanted to use Vue with TS to lean towards Class API powered by `vue-class-component`. However, a class-based API heavily relies on ES decorators, a language feature that was only a stage 2 proposal when Vue 3 was being developed in 2019. We felt it was too risky to base an official API on an unstable proposal. Since then, the decorators proposal has gone through yet another complete overhaul, and finally reached stage 3 in 2022. In addition, class-based API suffers from logic reuse and organization limitations similar to Options API.

In comparison, Composition API utilizes mostly plain variables and functions, which are naturally type friendly. Code written in Composition API can enjoy full type inference with little need for manual type hints. Most of the time, Composition API code will look largely identical in TypeScript and plain JavaScript. This also makes it possible for plain JavaScript users to benefit from partial type inference.

Smaller Production Bundle and Less Overhead

Code written in Composition API and `<script setup>` is also more efficient and minification-friendly than Options API equivalent. This is because the template in a `<script setup>` component is compiled as a function inlined in the same scope of the `<script setup>` code. Unlike property access from `this`, the compiled template code can directly access variables declared inside `<script setup>`, without an instance proxy in between. This also leads to better minification because all the variable names can be safely shortened.

```

1 <script>
2 export default {
3   props: ['color'],
4   emits: ['update:name'],
5   data() {
6     return {
7       name: 'John Doe',
8       age: 30,
9       users: ['Jane', 'Mark', 'Bob'],
10    }
11  },
12  computed: {
13    details() {
14      return `${this.name} is ${this.age} years old`;
15    },
16    userList() {
17      return this.users.join(', ');
18    }
19  },
20  methods: {
21    updateName(newName) {
22      this.name = newName;
23      this.$emit('update:name', newName);
24    },
25    updateAge(newAge) {
26      this.age = newAge;
27    },
28    addUser(user) {
29      this.users.push(user);
30    },
31    removeUser(username) {
32      this.users = this.users.filter(user => user !== username);
33    }
34  },
35  watch: {
36    name(newName, oldName) {
37      // Do something when name changes
38    }
39  },
40  mounted() {
41    // Do something when component is mounted
42  },
43  updated() {
44    // Do something when component is updated
45  },
46 }
47 </script>

```

Options API

<https://www.webmound.com>

```

1 <script setup>
2 import { computed, onMounted, onUpdated, ref, watch } from 'vue';
3
4 const emits = defineEmits(['update:name']);
5
6 // Personal details section
7 const name = ref('John Doe');
8 const age = ref(30);
9 const details = computed(() => `${name.value} is ${age.value} years old`);
10 const updateName = (newName) => {
11   name.value = newName;
12   emits('update:name', newName);
13 }
14 const updateAge = (newAge) => {
15   age.value = newAge;
16 }
17 watch(name, (newName, oldName) => {
18   // Do something when name changes
19 });
20
21 // Users list section
22 const users = ref(['Jane', 'Mark', 'Bob']);
23 const userList = computed(() => users.value.join(', '));
24 const addUser = (user) => {
25   users.value.push(user);
26 }
27 const removeUser = (username) => {
28   users.value = users.value.filter(user => user !== username);
29 }
30
31 // Lifecycle hooks section
32 onMounted(() => {
33   // Do something when component is mounted
34 });
35
36 onUpdated(() => {
37   // Do something when component is updated
38 });
39 </script>

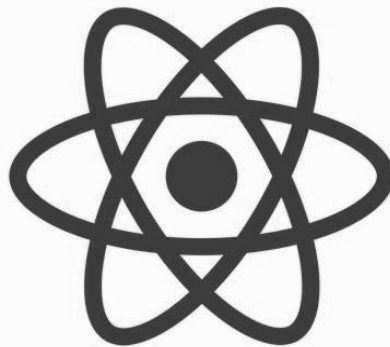
```

Composition API

Команда разработки React сделала тот же
выбор еще в 2018-2019 году

1. **useState**
2. **useReducer**
3. **useEffect**
4. **useRef**
5. **useLayoutEffect**
6. **useContext**
7. **useImperativeHandle**
8. **useMemo**
9. **useCallback**

ALL REACT HOOKS



Структура повествования

1. Коммерческая разработка и интересы бизнеса
2. Как достигается надежность
3. Выбор фронтенд фреймворка
4. Архитектура фронтенда

Коммерческая разработка и интересы бизнеса

Соотношение
цена/качество,
трудозатраты/качество
должно быть приемлемым

Исправленные баги не
должны возвращаться

Нужна предсказуемость
издержек

Рефакторинг и новые
фичи не должны ломать
старый функционал

Баг в продакшене может принести
финансовый ущерб клиенту и
испортить нашу репутацию

Стандартизация и
унификация всего

Сопровождение и поддержка
должны быть простыми и
требующими минимум
ресурсов

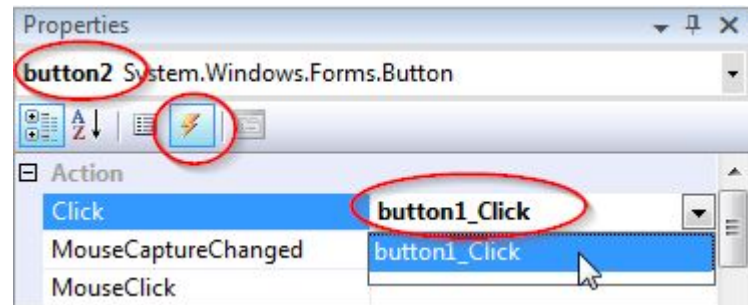
Репутация – очень большая
ценность. Наши продукты не
должны расстраивать
клиентов

Минимизация роли конкретного
человека в процессе разработки
(управление рисками)

Без менеджера состояния

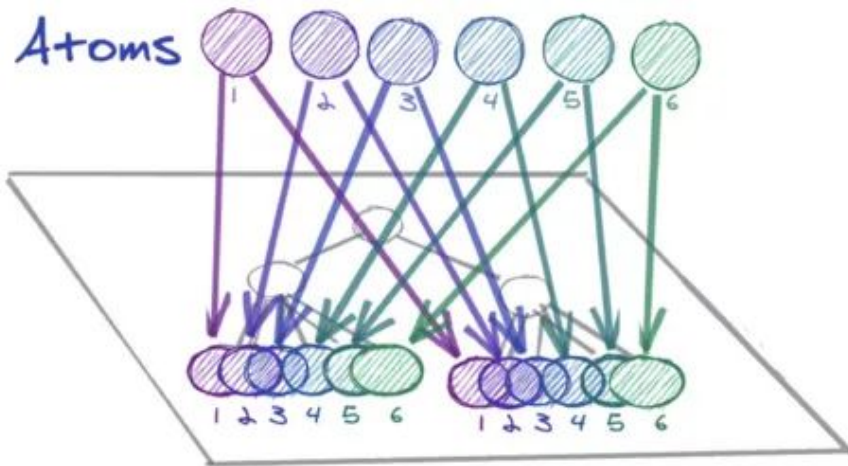
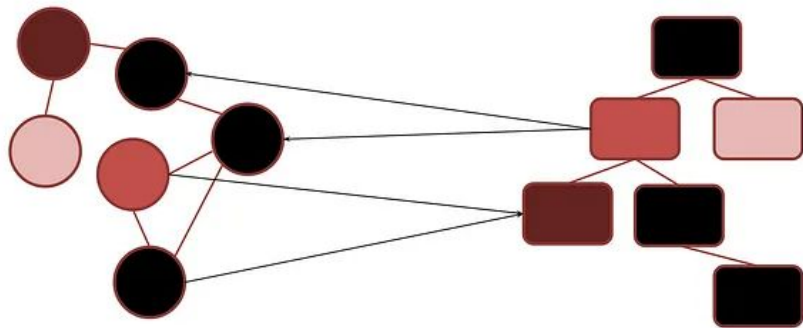
```

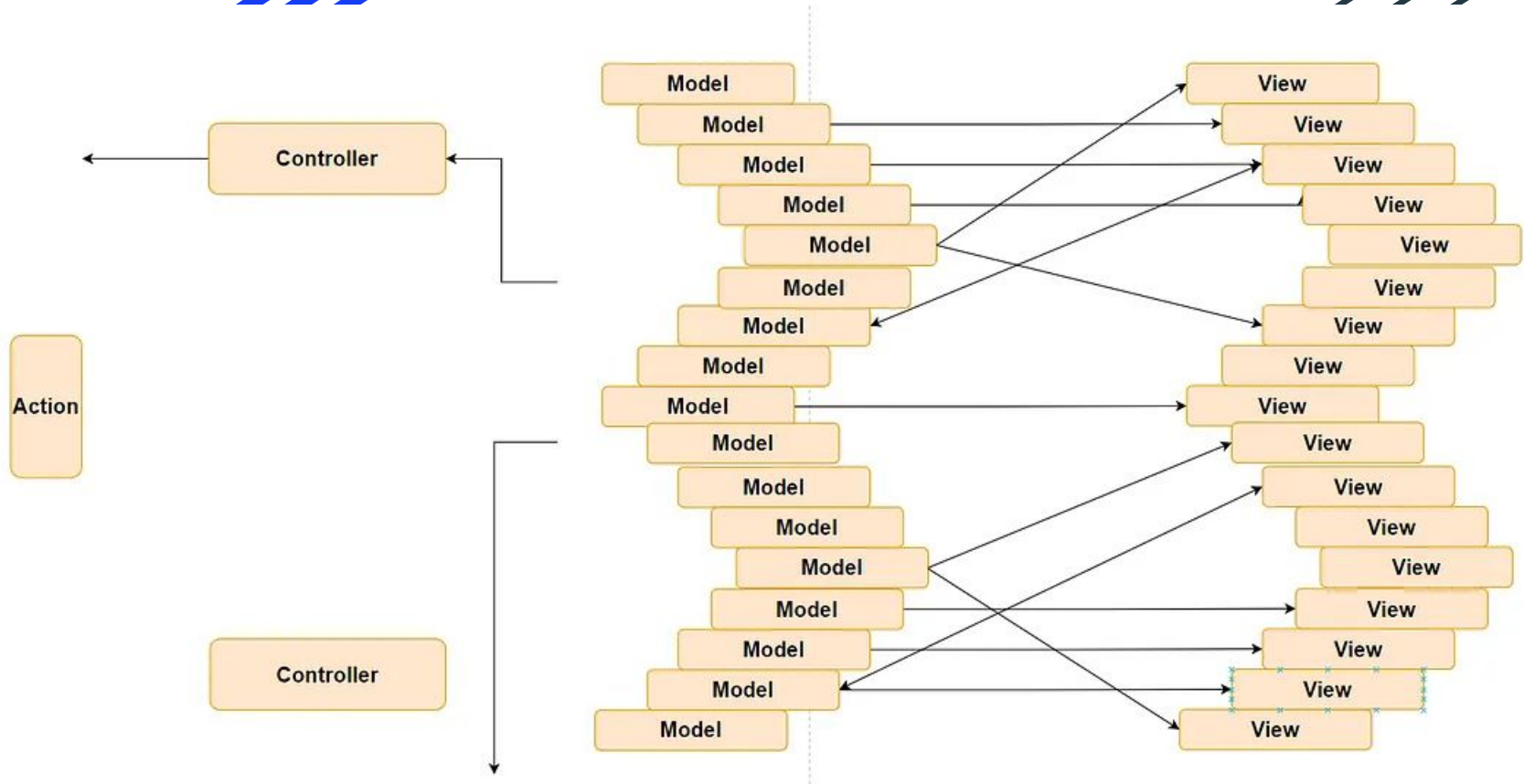
Form1.cs [Design] =
btnClickMe_Click
private void btnClickMe_Click(object sender, EventArgs e)
{
    lblButtonClick.Text = "This text is set on Form1
  
```



Model

Components





Пользовательский интерфейс является отражением состояния приложения

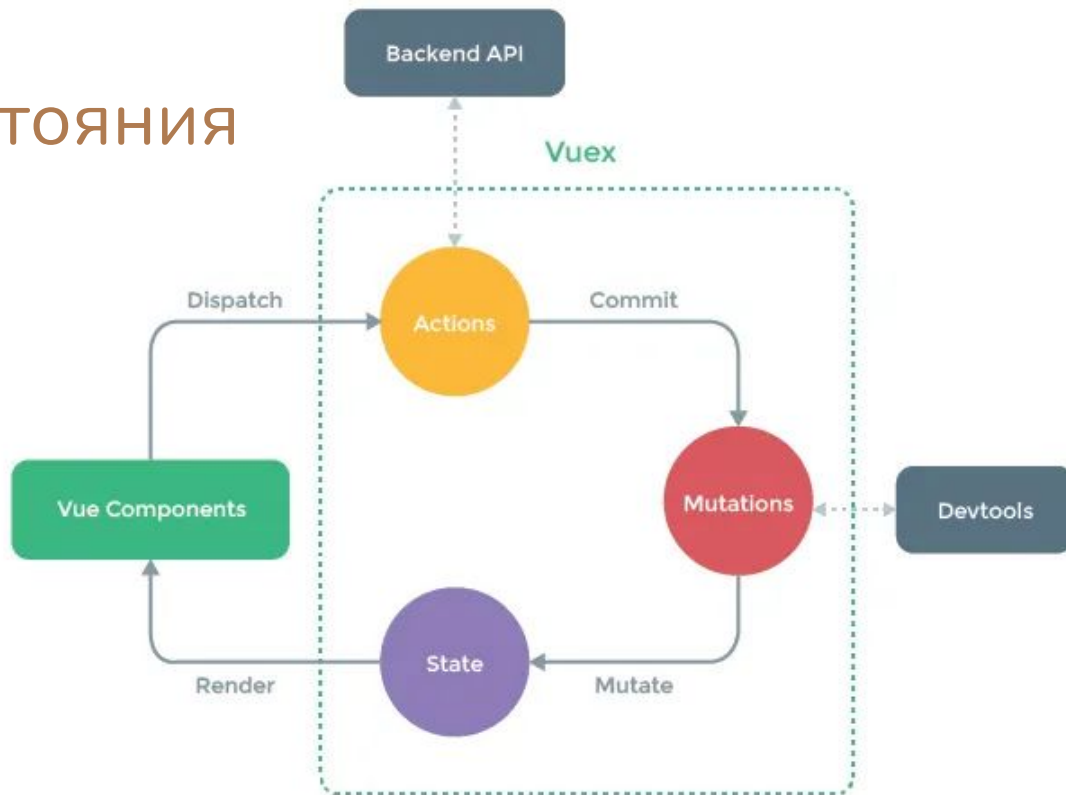
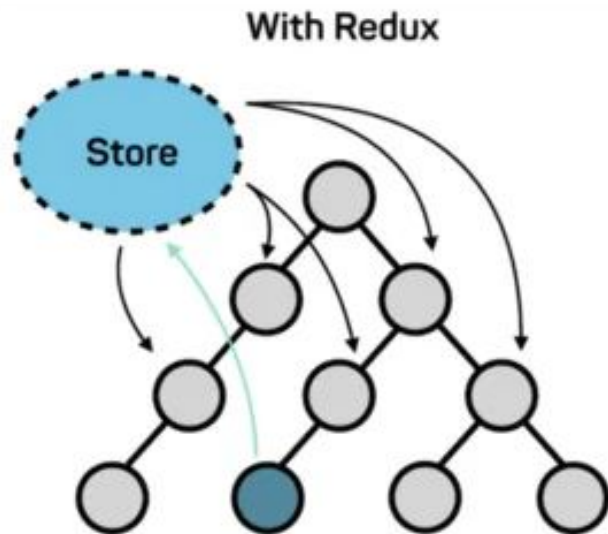
$$\text{UI} = f(\text{state})$$

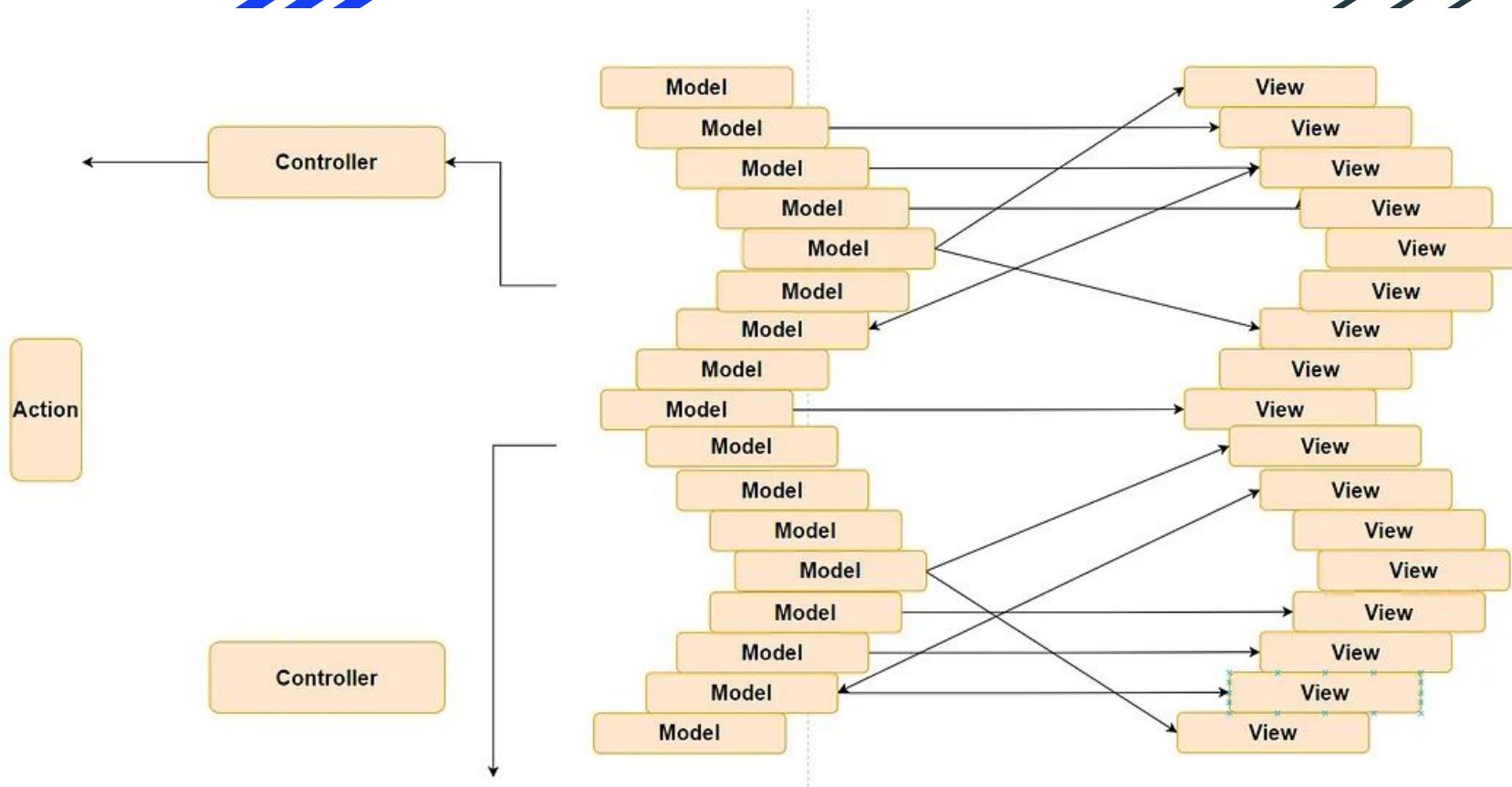
The layout
on the screen

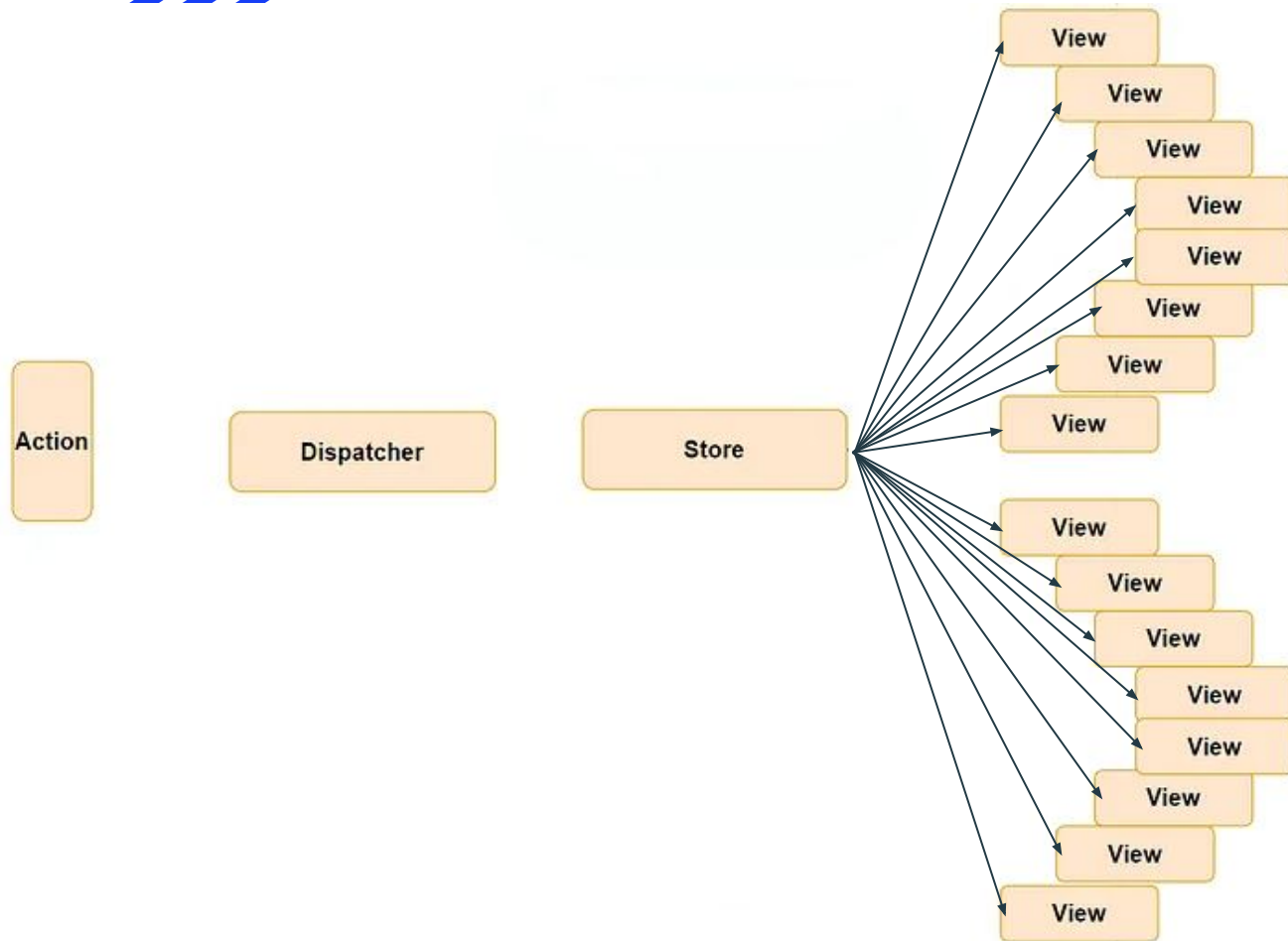
Your
build
methods

The application state

С менеджером состояния







The screenshot shows a web browser at `localhost:3000` displaying a ReactJS Boilerplate application. The page has a header with the text "ReactJS Minimal" and a "Home" button. Below the header, there is a section titled "Start your next re" with a play button icon and a description "A minimal React-Redux bo".

Overlaid on the right side of the browser is the Redux DevTools interface. The top bar shows "Inspector" and "Home Page - ReactJS Boilerplate". The left pane shows a list of actions, including "@@INIT", "boilerplate/Home/CHANGE_USER...", and "boilerplate/App/LOAD_REPOS...". The right pane shows the state tree, which includes a "global" object with "loading" and "currentUser" properties, and a "userData" object with a "repositories" array.

Below the "Start your next re" section, there is a "Try me!" heading and a text input field labeled "Show Github repositories by @js1599". Below the input field is a list of repository links:

- [Debugger-Learning-Material](#)
- [NBA-Swipe-React](#)
- [NBA-Swipe](#)
- [TheRebaseTest](#)
- [nakleiderer/boardtownuas-website](#)

Controlled

Controlled by local state



```
<label>
  Name:
  <input
    type="text"
    value={this.state.value}
    onChange={this.handleChange}
  />
</label>
```

Uncontrolled

Native HTML input



```
<label>
  Name:
  <input type="text" />
</label>
```

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение
Авдиева, 1

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение

Выберите вариант из списка

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение
ул Авдиева, д|

ул Авдиева, д 1

ул Авдиева, д 2

ул Авдиева, д 4

ул Авдиева, д 5

Address

Validation

Select

Error Label

Address (Uncontrolled)

Validation

Button

Select

Modal

Error Label

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение
Авдиева, 1

Мои адреса



Адрес №8



Ростовская обл, г. Таганрог, Авдиева, 2



Адрес №9

Ростовская обл, г. Таганрог, ул Авдиева,
д 9999

Адрес №10

Ростовская обл, г. Таганрог, ул
Галицкого, д 59а

ПРИВЕЗТИ СЮДА

Stepper

Address (Uncontrolled)

Validation

Button

Select

Modal

Error Label

Оформление заказа

Корзина

27 423 Р | 1 товар



ИЗМЕНИТЬ

Способ получения

 ТАГАНРОГ

ИЗМЕНИТЬ

ДОСТАВКА

САМОВЫВОЗ

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение
ул Авдеева, д 2

Оформление заказа

Корзина

27 423 ₽ | 1 товар



ИЗМЕНИТЬ

Способ получения

Доставка

Таганрог, Авдеева, 2.
Продавец: ТехноНиколь

2 265 ₽ | 23 Мая



ИЗМЕНИТЬ

Получатель

Индивидуальный предприниматель

Жуков Антон
+7 (920) 759-53-88



ИЗМЕНИТЬ

Оплата

Безналичная оплата

Чек будет отправлен по почте



ИЗМЕНИТЬ

Доступен новый тип оплаты 

ЗАВЕРШИТЬ ОФОРМЛЕНИЕ ЗАКАЗА

Stepper

Address (Uncontrolled)

Validation

Button

Select

Modal

Error Label

Оформление заказа

Корзина

27 423 ₽ | 1 товар

Способ получения

📍 ТАГАНРОГ

ДОСТАВКА

САМОВЫВОЗ

Адрес

ВЫБРАТЬ ИЗ М

Улица, дом, строение
ул Авдиева, д 2

В данный момент доставка по этому адресу невозможна

Оформление заказа

Корзина

27 423 ₽ | 1 товар



ИЗМЕНИТЬ

Способ получения

Доставка

Таганрог, Авдиева, 2.
Продавец: ТехноНиколь

2 265 ₽ | 23 Мая



ИЗМЕНИТЬ

Пол

Индиви

Жуков

+7 (920



ИЗМЕНИТЬ

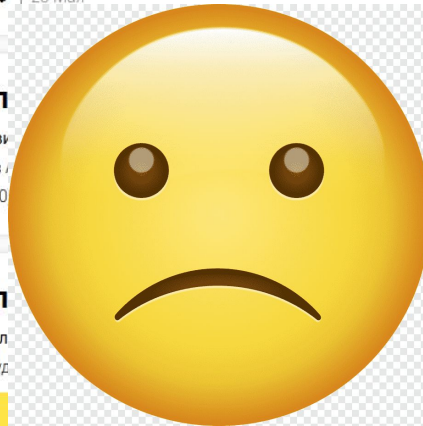
Опл

Безнал

Чек буд



ИЗМЕНИТЬ



ЗАВЕРШИТЬ ОФОРМЛЕНИЕ ЗАКАЗА

С менеджером состояния и Flux

Оформление заказа

Корзина

27 423 ₽ | 1 товар



Способ доставки



ТАГАНРОГ

ДОСТАВКА

САМОВЫВОЗ

Адрес

ВЫБРАТЬ ИЗ МОИХ АДРЕСОВ

Улица, дом, строение
ул Авдиева, д 2

В данный момент доставка по этому адресу невозможна

```
state.checkout.step = 2 // shipping
state.shipping.address.error = "В данный момент
доставка по этому адресу невозможна"
```

Onion-архитектура



onion architecture

Q Все

Картинки

Видео

Книги

Покупки



uncle bob clean code



clean



node js

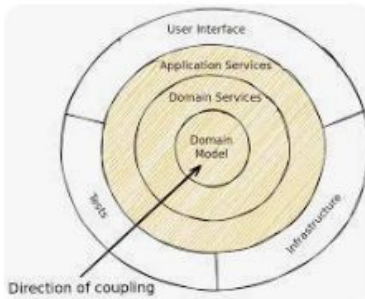


hexagonal



CodeGuru

Onion Architecture: Definition,...



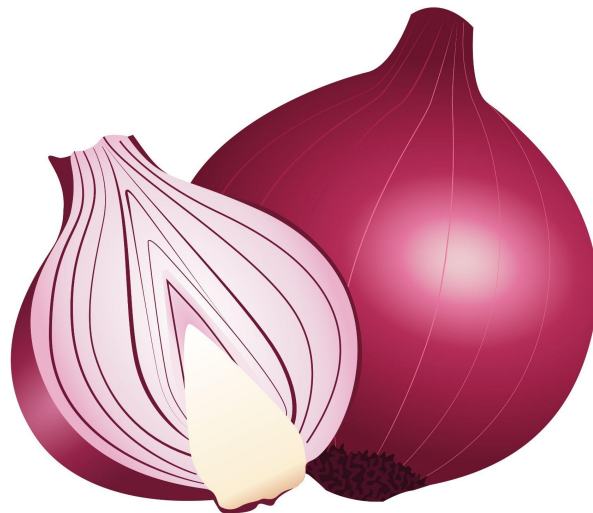
DEV Community

Onion Architecture 🍷 - DEV Co...

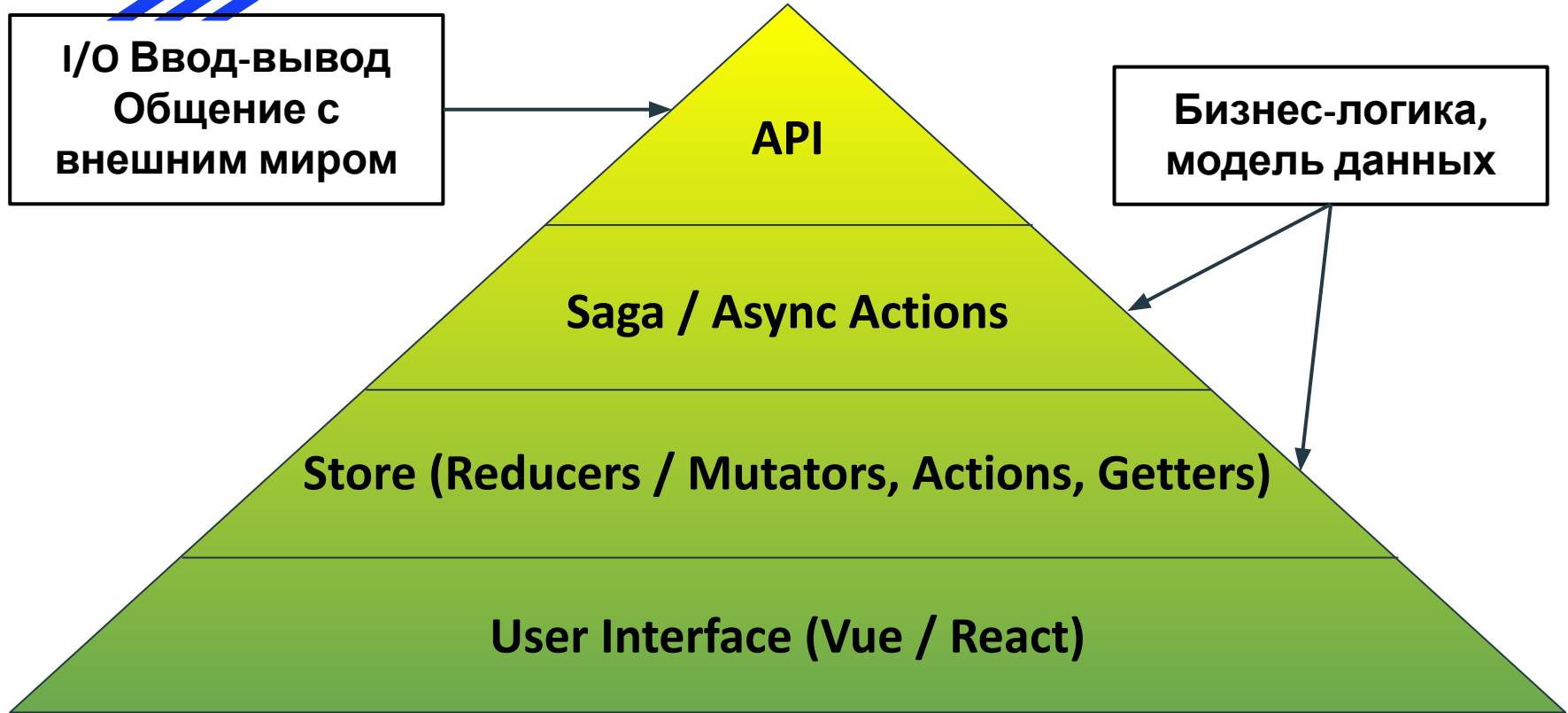


@hgraca

Onion Architecture



Архитектура фронтенда



API, I/O, TypeScript

billing	Billing ▾ {
lastname	string nullable: true
firstname	string nullable: true
telephone	string nullable: true
email	string nullable: true
legal_type	integer
	Тип плательщика:
	<ul style="list-style-type: none"> 1 - Юридическое лицо 2 - Физическое лицо 3 - ИП
	Enum:
	<ul style="list-style-type: none"> > Array [3]
has_proxy	boolean nullable: false
	If has proxy
inn	string nullable: true

Swagger

```
import { z } from "zod";

const Billing = z.object({
  lastname: z.string().nullable(),
  firstname: z.string().nullable(),
  telephone: z.string().nullable(),
  email: z.string().nullable(),
  legal_type: z.enum([1, 2, 3]),
  has_proxy: z.boolean(),
  inn: z.string().nullable(),
  /* ... */
});
```

Код на TypeScript

Redux-Saga

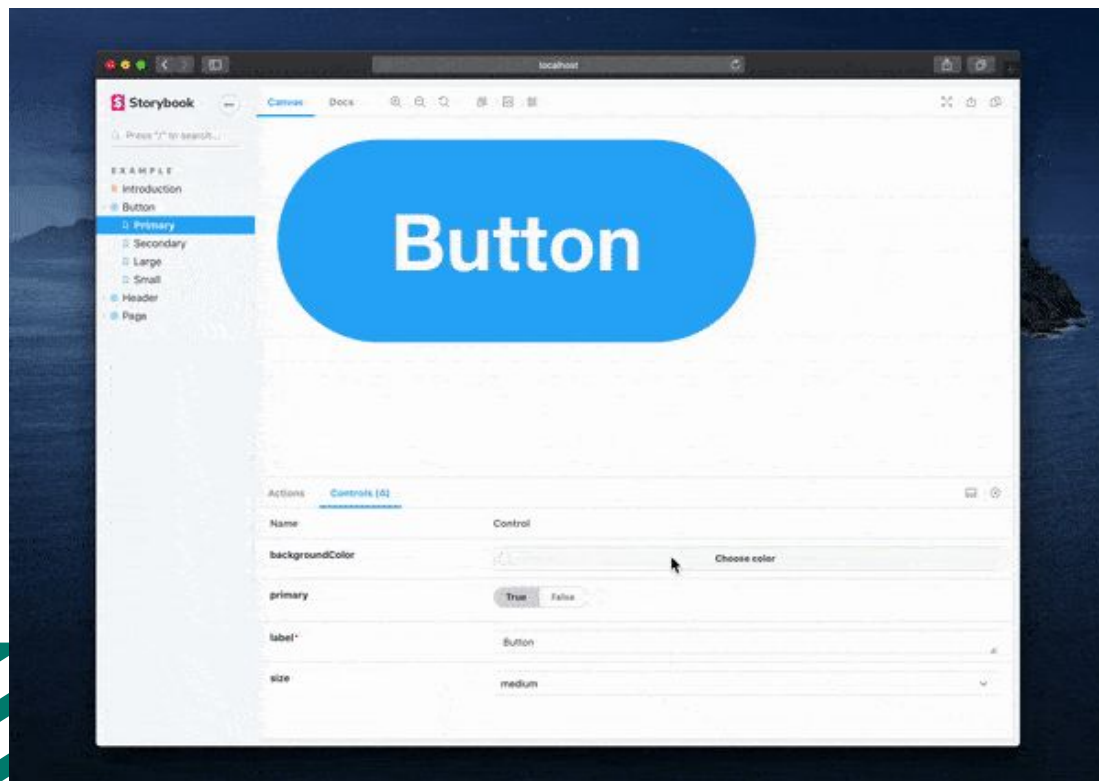
```
import {put, all, takeLatest, call} from 'redux-saga/effects';
import {getMovies} from '../api/movies';
import {saveMovies} from '../actions/MoviesActions';
import {GET_MOVIES_SAGA} from '../actions/ActionTypes';

function* searchMovies(action) {
  const {name} = action.payload;
  let movies = yield call(getMovies, name);
  yield put(saveMovies(movies));
}

const sagas = [takeLatest(GET_MOVIES_SAGA, searchMovies)];

export default function* sagasRoot() {
  yield all([...sagas]);
}
```

UI, StoryBook



Коммерческая разработка и интересы бизнеса

Соотношение
цена/качество,
трудозатраты/качество
должно быть приемлемым

Исправленные баги не
должны возвращаться

Нужна предсказуемость
издержек

Рефакторинг и новые
фичи не должны ломать
старый функционал

Баг в продакшене может принести
финансовый ущерб клиенту и
испортить нашу репутацию

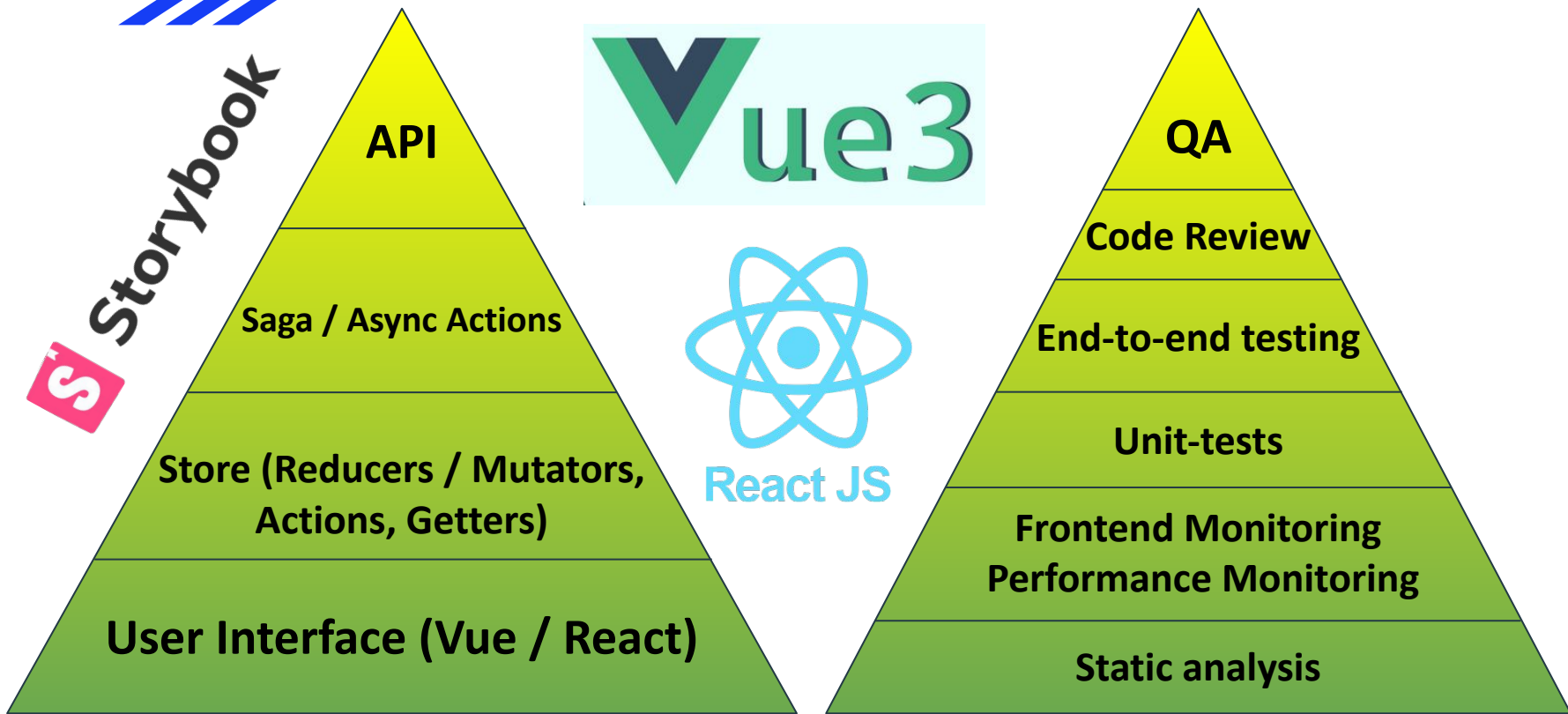
Стандартизация и
унификация всего

Сопровождение и поддержка
должны быть простыми и
требующими минимум
ресурсов

Репутация – очень большая
ценность. Наши продукты не
должны расстраивать
клиентов

Минимизация роли конкретного
человека в процессе разработки
(управление рисками)

Как поставить разработку фронтенда на поток



Спасибо за внимание!
Вопросы?



Антон Жуков

https://t.me/mister_cheater