MinskJS Meetup #13

Online, 19 августа 2025 в 19:00

Frontend-фреймворки: лишь перестав выбирать, мы обретаем свободу

Анна Ширяева

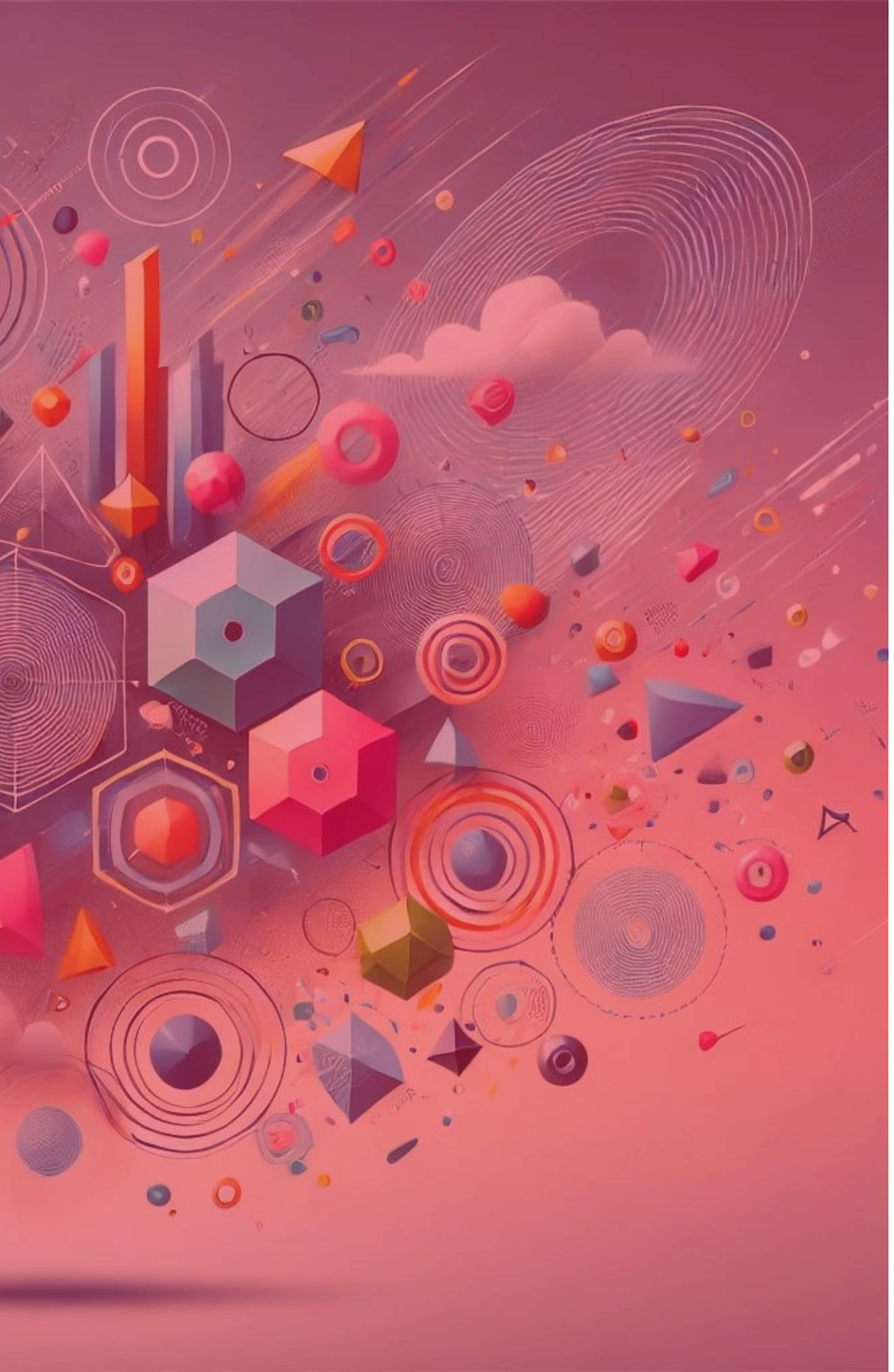
СИБУР Цифровой

О себе

- Коллекционер хобби;
- В IT с 2011;
- C# > Angular > React;
- Ведущий фронтенд-разработчик СИБУР Цифровой;
- Организатор MoscowJS ❤

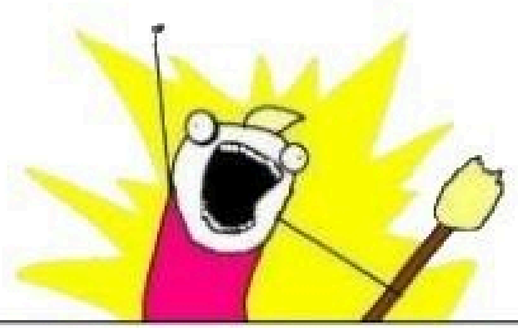
О докладе

- Чего мы хотим от фреймворков?
- Основные концепции
- Вопросы на засыпку
- Выбор
- Куда все идет?
- Итоги

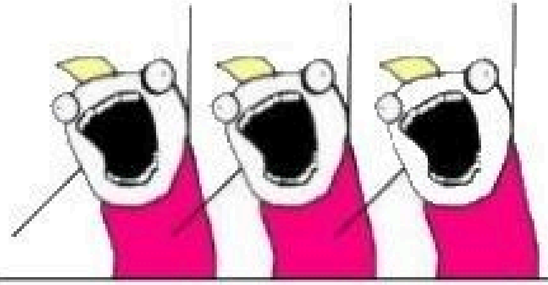


Чего мы хотим от фреймворков?

Кто мы?

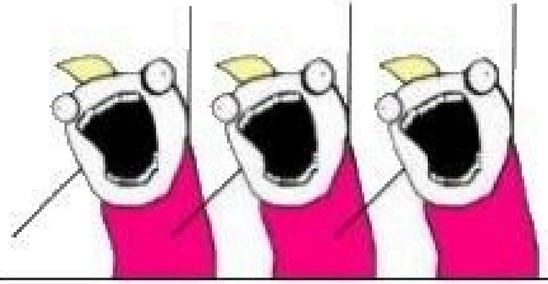


ФРОНТЕНДЕРЫ!



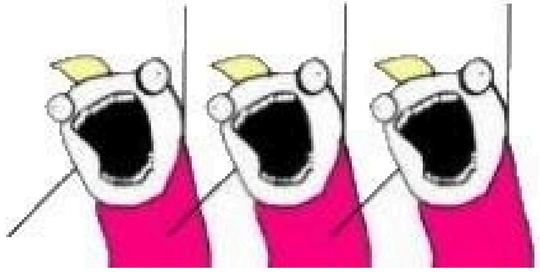
Чего мы хотим?

Чтобы все работало!



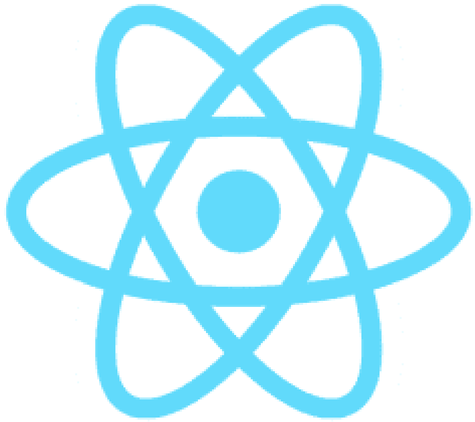
Что все?

???

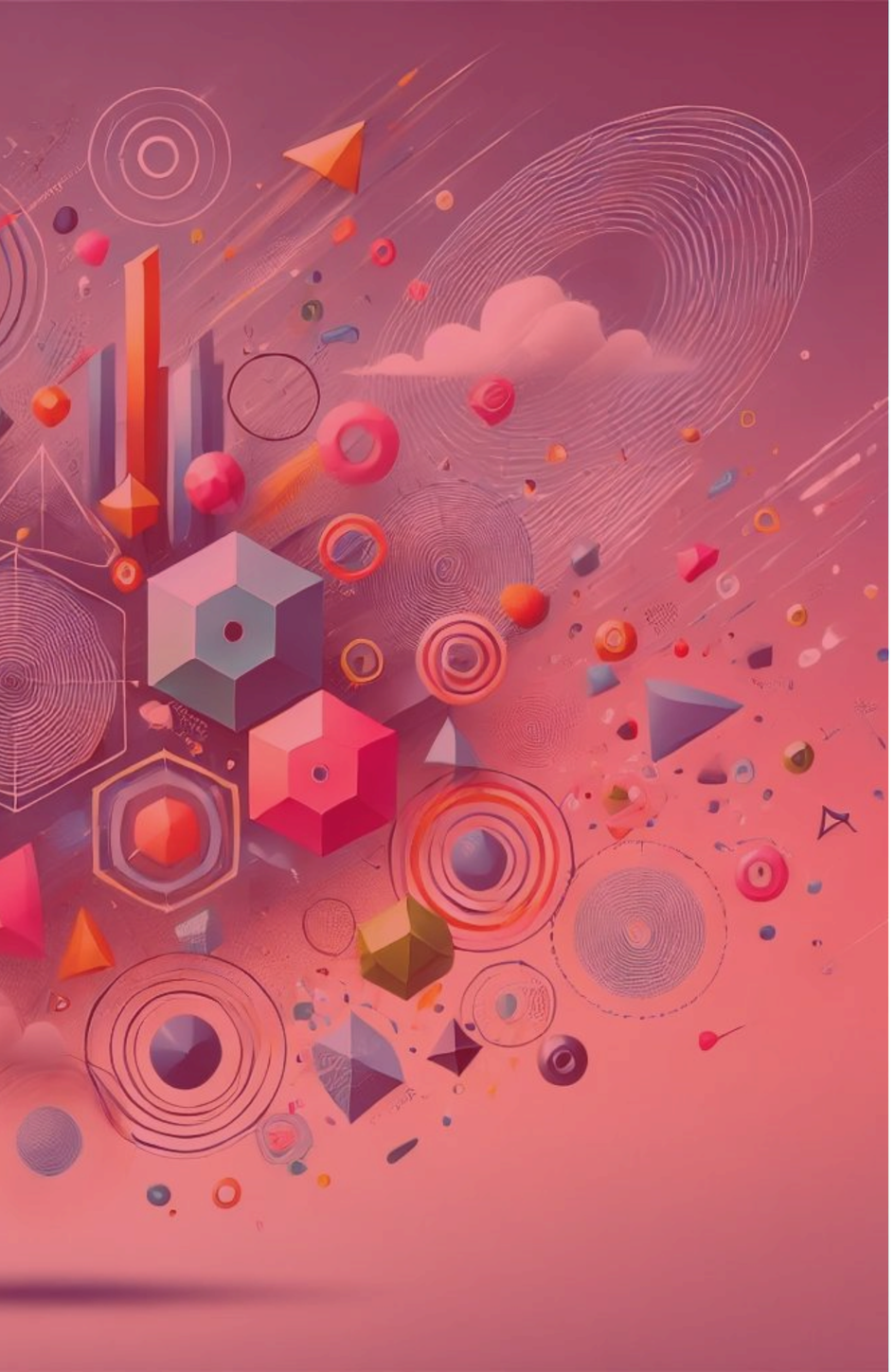


Три кита фронтенда


Knockout.

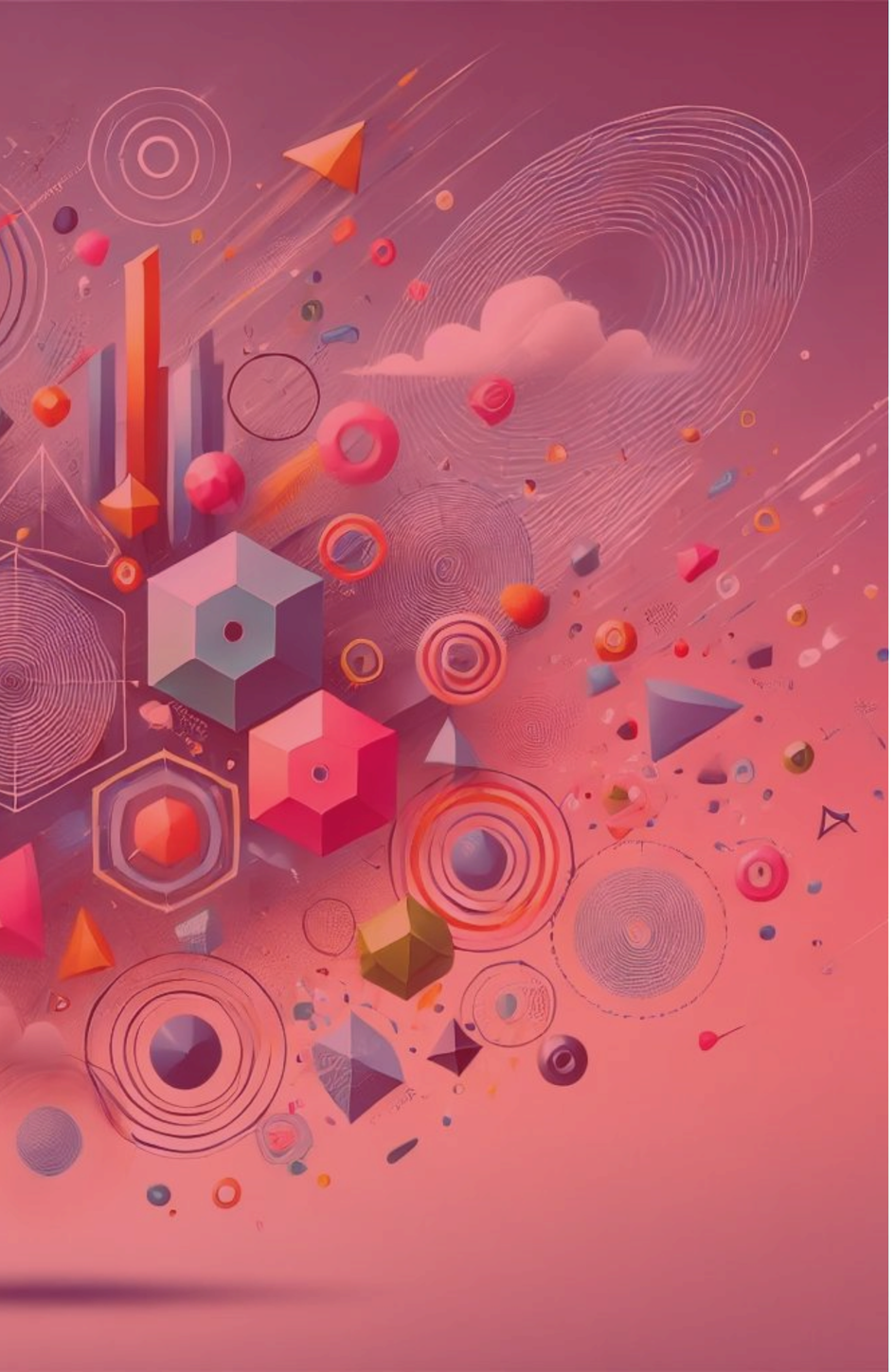


Vike



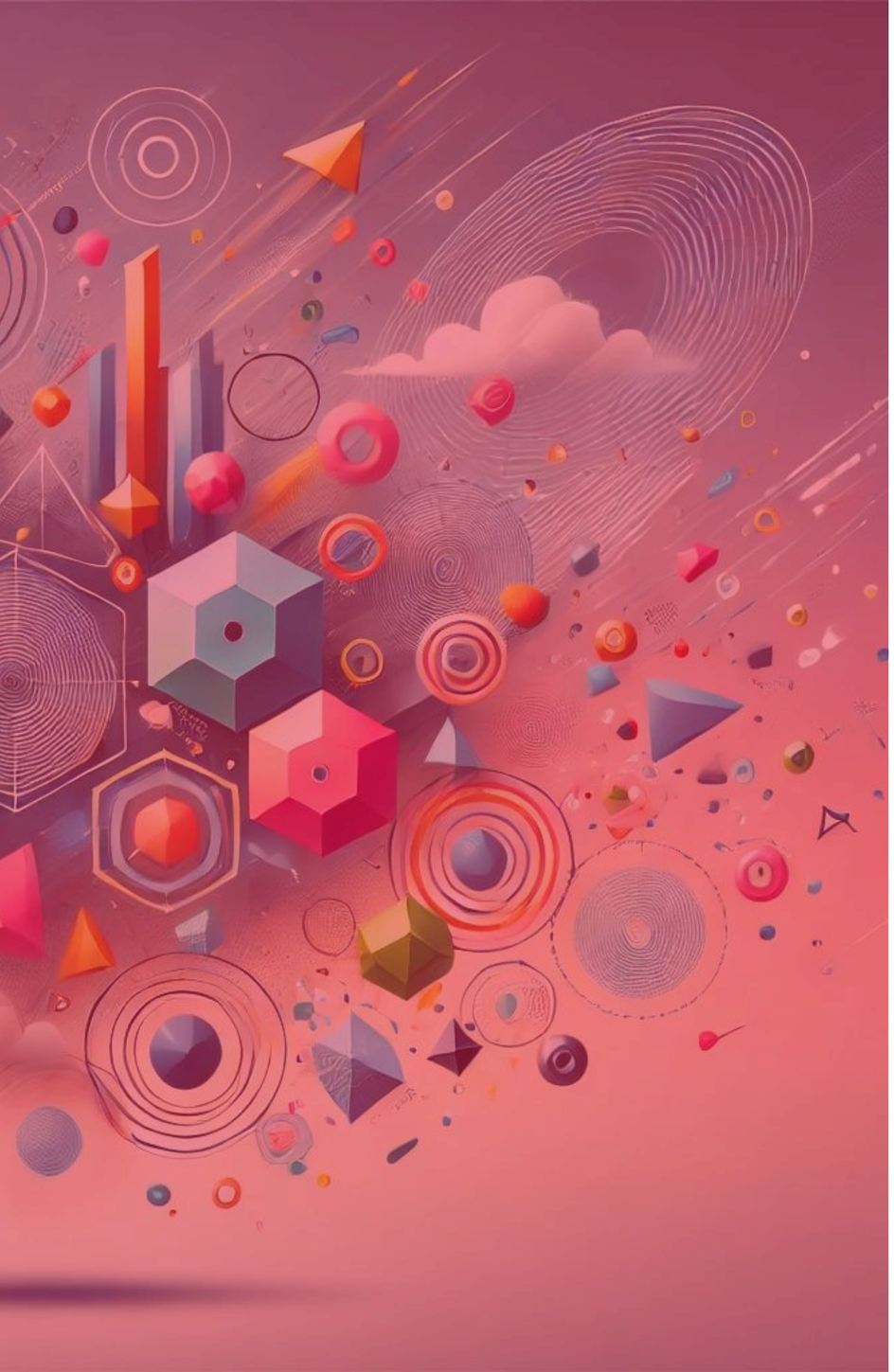
Чего мы хотим?

- Создавать страницы
- Хранить и пересчитывать данные
- Обновлять страницу
- В браузере только нужное
- Видится поисковиками
- Работает везде
- Писать легко и мало



Основные концепции

- Шаблоны
- Состояние и Реактивность
- Работа с DOM
- Компиляторы
- Стратегии рендера
- Кроссплатформенность
- DX и экосистема



Основные концепции

- **Шаблоны**
- Состояние и Реактивность
- Работа с DOM
- Компиляторы
- Стратегии рендера
- Кроссплатформенность
- DX и экосистема

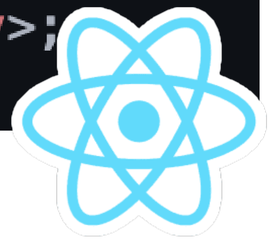
```
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```

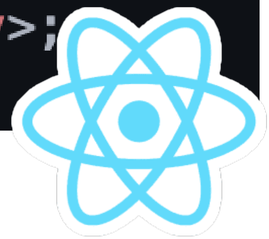
```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

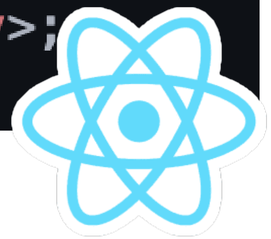
```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



```jsx
export default function DoubleCount() {
  const [count] = createSignal(10);
  const doubleCount = () => count() * 2;

  return <div>{doubleCount()}</div>;
}
```

```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

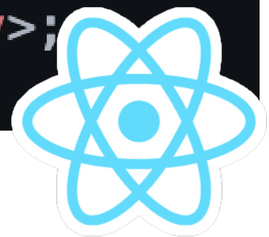
```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



```jsx
export default function DoubleCount() {
  const [count] = createSignal(10);
  const doubleCount = () => count() * 2;

  return <div>{doubleCount()}</div>;
}
```

```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```

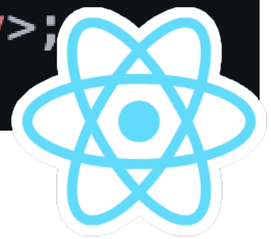


```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

```jsx
export default function DoubleCount() {
  const [count] = createSignal(10);
  const doubleCount = () => count() * 2;

  return <div>{doubleCount()}</div>;
}
```

```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```

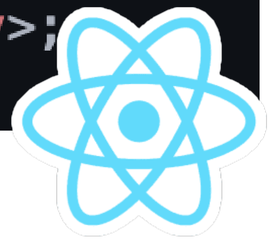


```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

```jsx
export default function DoubleCount() {
  const [count] = createSignal(10);
  const doubleCount = () => count() * 2;

  return <div>{doubleCount()}</div>;
}
```

```svelte
<script>
  let count = $state(10);
  const doubleCount = $derived(count * 2);
</script>

<div>{doubleCount}</div>
```

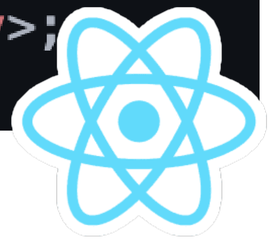
```tsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

```ts
@Component({
  selector: "app-double-count",
  template: `<div>{{ doubleCount() }}</div>`,
})
export class DoubleCountComponent {
  count = signal(10);

  doubleCount = computed(() => this.count() * 2);
}
```

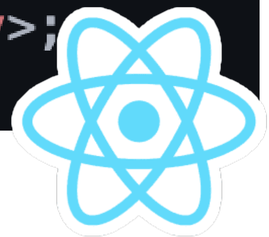
```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

```angular
@Component({
  selector: "app-double-count",
  template: `<div>{{ doubleCount() }}</div>`,
})
export class DoubleCountComponent {
  count = signal(10);

  doubleCount = computed(() => this.count() * 2);
}
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```

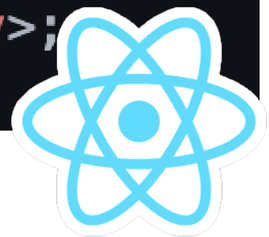


Для любителей ФП и JSX

```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

```angular
@Component({
  selector: "app-double-count",
  template: `<div>{{ doubleCount() }}</div>`,
})
export class DoubleCountComponent {
  count = signal(10);

  doubleCount = computed(() => this.count() * 2);
}
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



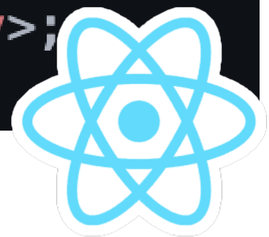
Для любителей ФП и JSX

```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

Классическая веб-страница

```angular
@Component({
  selector: "app-double-count",
  template: `<div>{{ doubleCount() }}</div>`,
})
export class DoubleCountComponent {
  count = signal(10);

  doubleCount = computed(() => this.count() * 2);
}
```

```jsx
export default function DoubleCount() {
  const [count] = useState(10);
  const doubleCount = count * 2;

  return <div>{doubleCount}</div>;
}
```



Для любителей ФП и JSX

```vue
<script setup>
import { ref, computed } from "vue";
const count = ref(10);
const doubleCount = computed(() => count.value * 2);
</script>

<template>
  <div>{{ doubleCount }}</div>
</template>
```

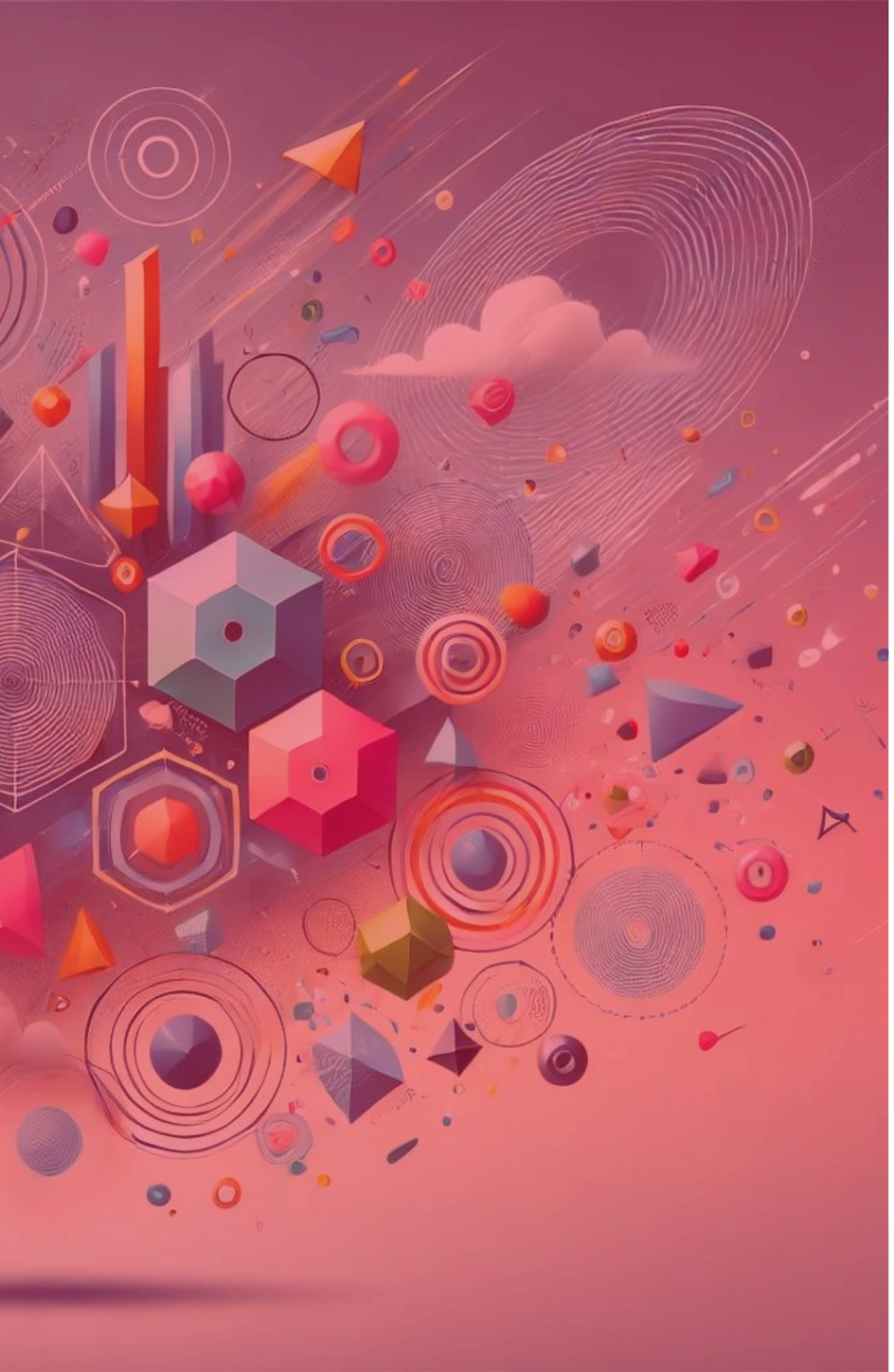
Классическая веб-страница

```angular
@Component({
  selector: "app-double-count",
  template: `<div>{{ doubleCount() }}</div>`,
})
export class DoubleCountComponent {
  count = signal(10);

  doubleCount = computed(() => this.count() * 2);
```

Классы, ООП,
Для бэкендеров

Есть

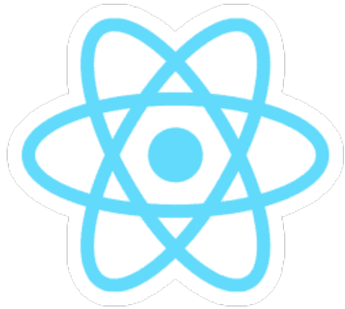
- То, что станет HTML.
- То, что станет JS.
- То, что станет CSS.
- Динамическая привязка данных.



Основные концепции

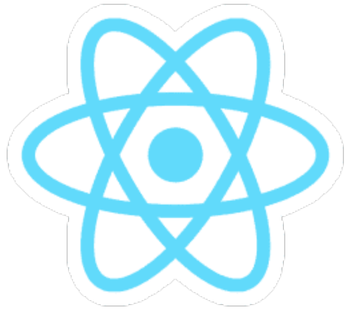
- Шаблоны
- **Состояние и Реактивность**
- Работа с DOM
- Компиляторы
- Стратегии рендера
- Кроссплатформенность
- DX и экосистема

Состояния



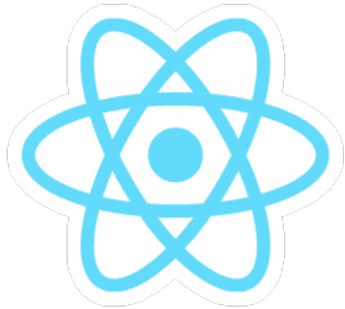
Передача между компонентами	**Input**	**Props**	**Props**

Состояния



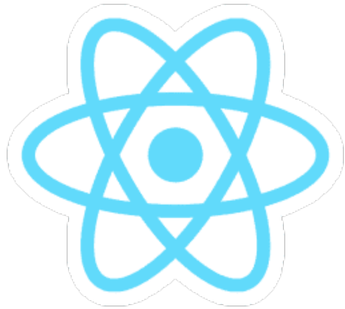
Передача между компонентами	**Input**	**Props**	**Props**
Контекст области	**Shared Service**	**Context**	**Provide / Inject**

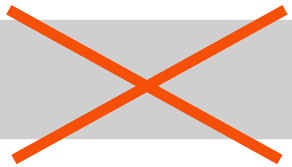
Состояния



Передача между компонентами	**Input**	**Props**	**Props**
Контекст области	**Shared Service**	**Context**	**Provide / Inject**
Глобальное хранилище	**State manager**		

Состояния

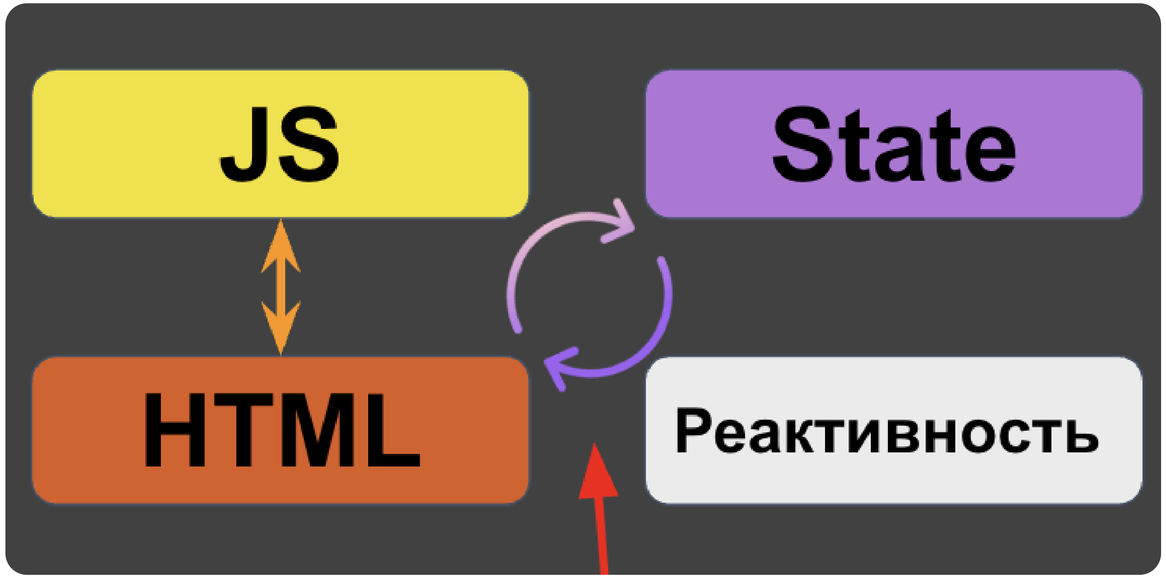


Передача между компонентами	**Input**	**Props**	**Props**
Контекст области	**Shared Service**	**Context**	**Provide / Inject**
Глобальное хранилище		**State manager**	

Реактивность

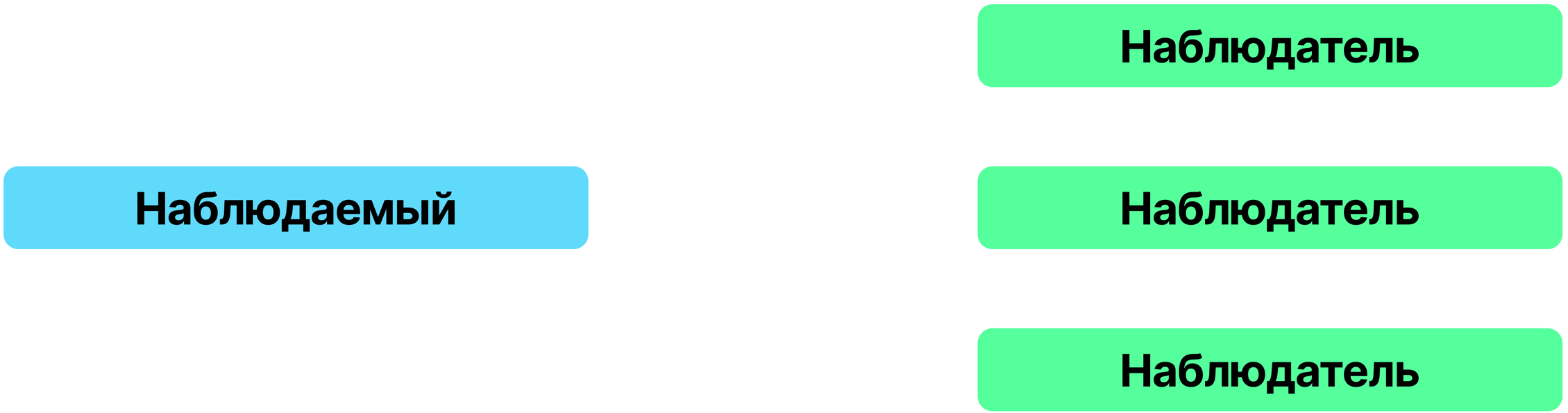
Механизм, который...

- следит за изменением стейта;
- пересчитывает связанные значения;
- обновляет UI.



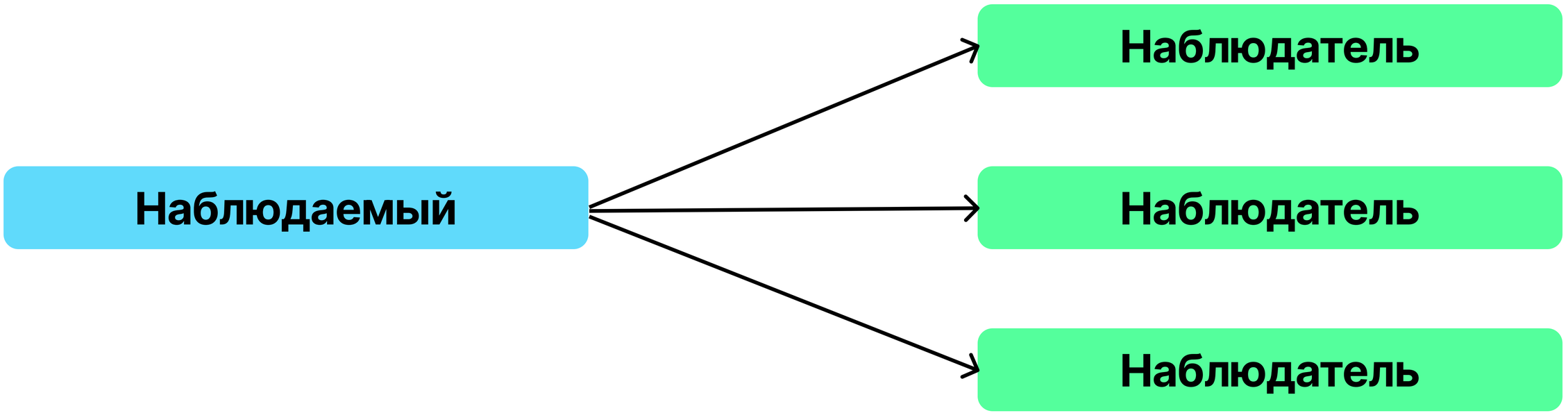
Реактивность

Основа:



Реактивность

Основа:



Реактивность

Observable – паттерн программирования

Signal – паттерн, популяризированный Solid

Proxy – нативный объект JS

Реактивность

Observable

RxJS

Signal

Proxy

Реактивность

Observable

Signal

Proxy

Реактивность

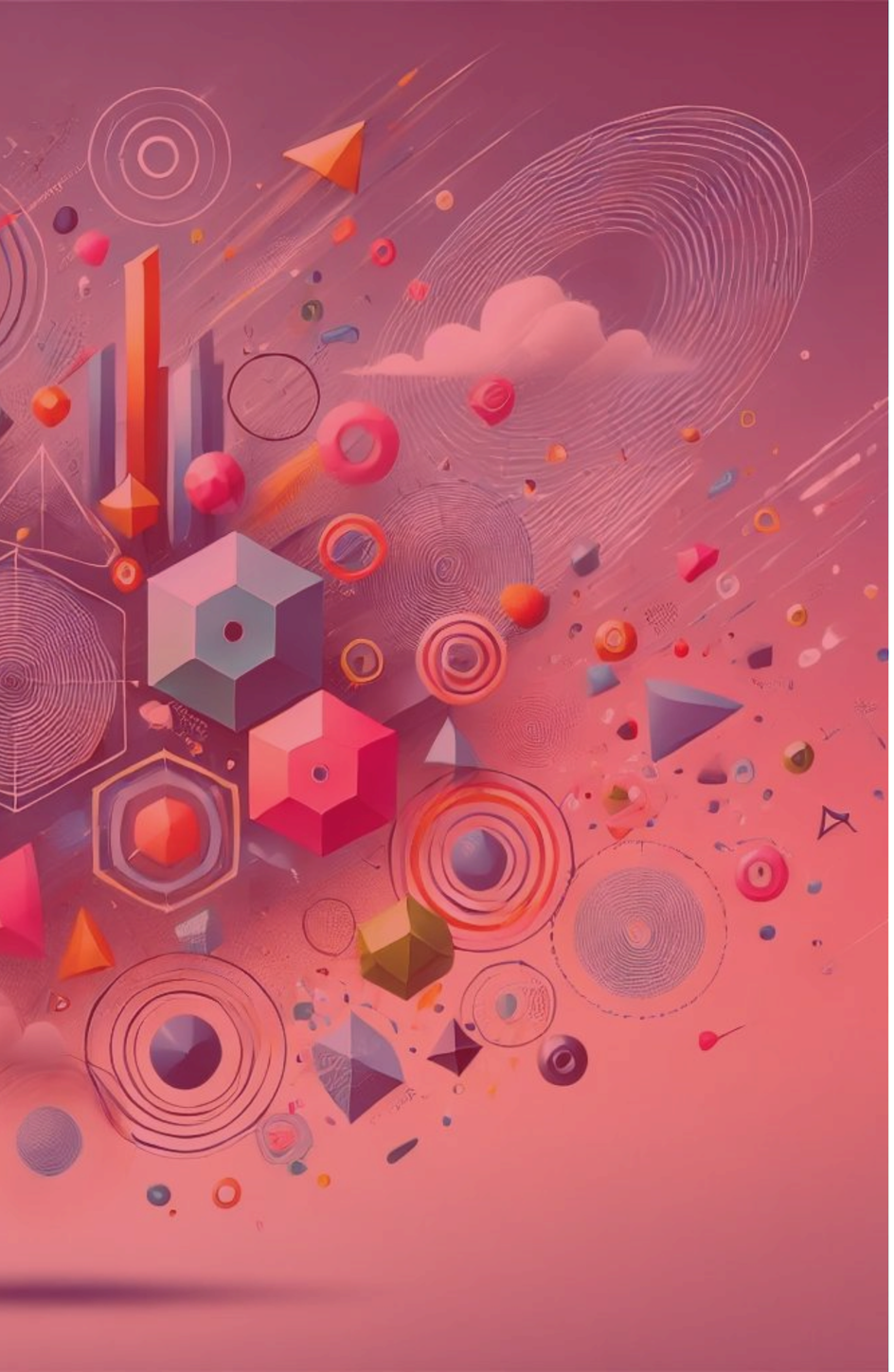
Observable

Signal

Proxy

Реактивность

- Эра сигналов!
- Fine-grained реактивность.



Основные концепции

- Шаблоны
- Состояние и Реактивность
- **Работа с DOM**
- Компиляторы
- Стратегии рендера
- Кроссплатформенность
- DX и экосистема

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- Incremental DOM (Angular)
- Virtual DOM (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- Resumable Hydration (Qwik)

Работа с DOM

- **Традиционный** (jQuery, Alpine.js)
- Incremental DOM (Angular)
- Virtual DOM (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- Resumable Hydration (Qwik)

- Прямое манипулирование DOM через API;
- Императивный подход к обновлениям.

Плюсы:

- Полный контроль;
- Нет накладных расходов.

Минусы:

- Сложность масштабирования;
- Рутинное управление состоянием.

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- **Incremental DOM** (Angular)
- Virtual DOM (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- Resumable Hydration (Qwik)

- Пошаговое построение DOM-дерева;
- Сравнение происходит на уровне инструкций;
- Меньше потребление памяти, чем VDOM.

Плюсы:

- Эффективность в больших приложениях;
- Хорошая интеграция с шаблонами.

Минусы:

- Сложнее для оптимизации;
- Менее гибкий, чем VDOM.

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- Incremental DOM (Angular)
- **Virtual DOM** (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- Resumable Hydration (Qwik)

- Виртуальное представление DOM в памяти;
- При изменениях генерируется новый VDOM;
- Алгоритм сравнения;
- Применяются только необходимые изменения к реальному DOM;

Плюсы:

- Автоматическая оптимизация групповых обновлений.

Минусы:

- Накладные расходы на создание VDOM;
- Избыточные вычисления при глубоких сравнениях.

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- Incremental DOM (Angular)
- Virtual DOM (React, Preact, Vue 2)
- **Fine-grained reactivity** (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- Resumable Hydration (Qwik)

- Система отслеживает зависимости между данными и DOM-элементами;
- При изменении данных обновляются только связанные элементы;
- Нет полного сравнения деревьев.

Плюсы:

- Минимальные обновления DOM;
- Нет накладных расходов на VDOM;
- Более предсказуемая производительность.

Минусы:

- Более сложная отладка;
- Ограниченная кроссплатформенность.

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- Incremental DOM (Angular)
- Virtual DOM (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- **Компиляция в DOM операции** (Svelte)
- Resumable Hydration (Qwik)

- Компилятор анализирует компоненты на этапе сборки;
- Генерирует оптимальный JavaScript-код для обновлений;
- В runtime нет алгоритмов сравнения.

Плюсы:

- Нулевые накладные расходы в runtime;
- Самый маленький размер бандла;
- Идеально для встраиваемых виджетов.

Минусы:

- Меньше гибкости в runtime;
- Сложнее отлаживать сгенерированный код.

Работа с DOM

- Традиционный (jQuery, Alpine.js)
- Incremental DOM (Angular)
- Virtual DOM (React, Preact, Vue 2)
- Fine-grained reactivity (Solid, Svelte, Vue 3)
- Компиляция в DOM операции (Svelte)
- **Resumable Hydration** (Qwik)

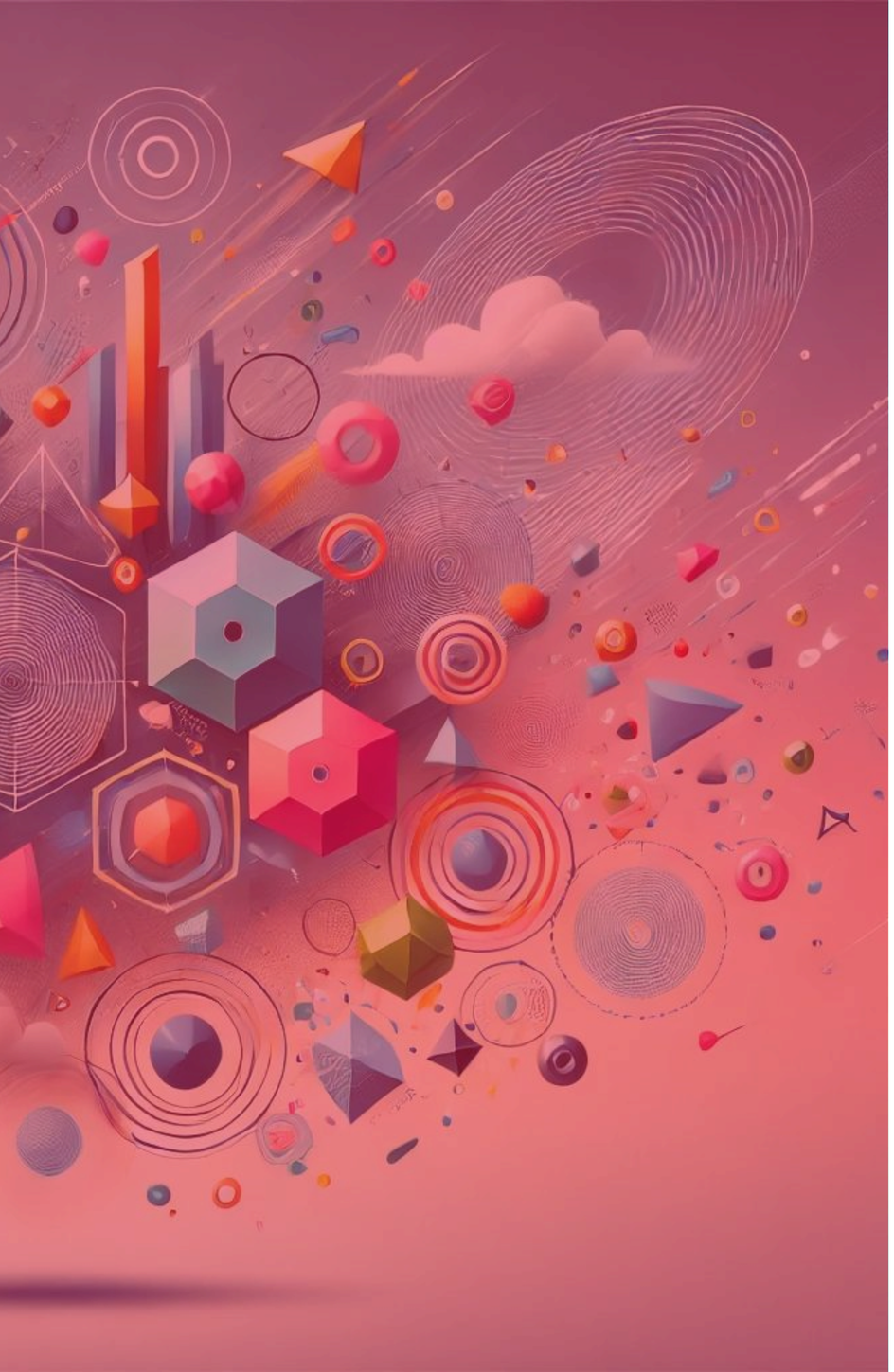
- Сервер рендерит полный HTML;
- Клиент «будит» только необходимые обработчики;
- Нет традиционной гидратации всего приложения.

Плюсы:

- Мгновенная загрузка;
- Оптимальное использование ресурсов;
- Лучший Time-to-Interactive.

Минусы:

- Новая парадигма разработки;
- Ограниченная экосистема.



Основные концепции

- Шаблоны
- Состояние и Реактивность
- Работа с DOM
- **Компиляторы**
- Стратегии рендера
- Кроссплатформенность
- DX и экосистема

Что происходит в браузере?

Интерфейс

Движок браузера

Графический движок

JavaScript движок

Что происходит в браузере?

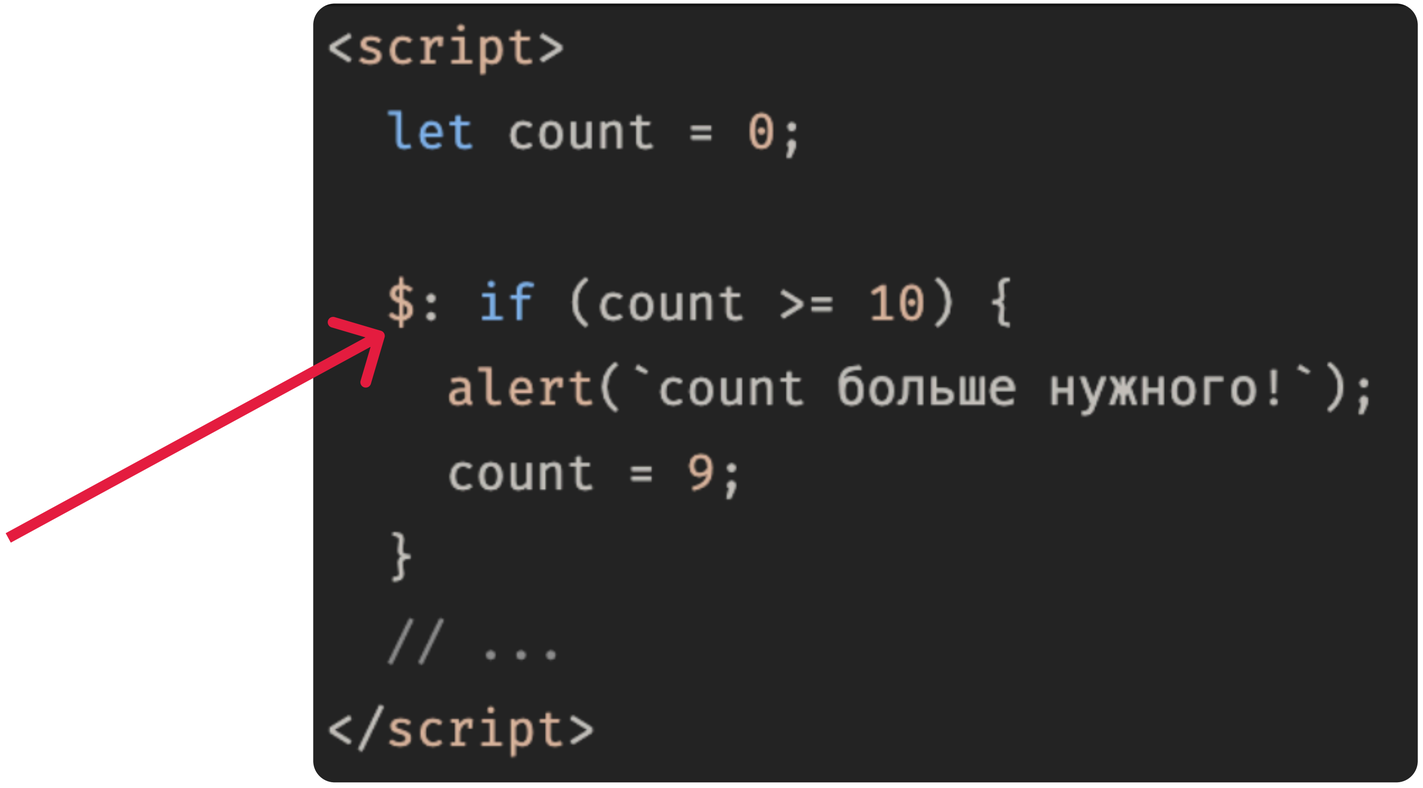
Интерфейс

Движок браузера

Графический движок

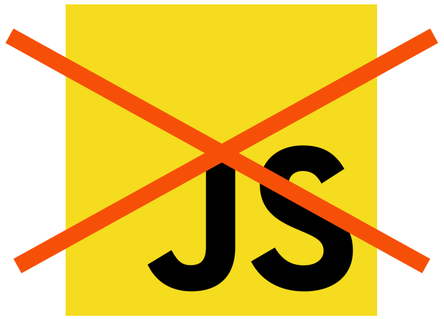
JavaScript движок

А внутри фреймворка?



```
<script>
  let count = 0;

  $: if (count >= 10) {
    alert(`count больше нужного!`);
    count = 9;
  }
  // ...
</script>
```

А внутри фреймворка?




```
<script>
  let count = 0;

  $: if (count >= 10) {
    alert(`count больше нужного!`);
    count = 9;
  }
  // ...
</script>
```

А внутри фреймворка?

- Пишем «шаблоны»
- Свои конвенции и синтаксис
- Компилятор призван это все понимать
  - поставляется как один из модулей в составе фреймворка

Задачи компиляторов:

Зависят от фреймворка.

Задачи компиляторов:

Зависят от фреймворка.

Angular

- Компилятор Ivy (с 2020)
  - AOT-компиляция (Ahead-of-Time)
  - Генерирует инструкции Incremental DOM
  - Удаляет декораторы в runtime

Задачи компиляторов:

Зависят от фреймворка.

Svelte

- Преобразует компоненты в чистый Vanilla JS
  - Удаляет runtime-зависимости
  - Генерирует точечные DOM-обновления
  - Оптимизирует анимации и переходы

Задачи компиляторов:

Зависят от фреймворка.

Solid

- Babel-плагин (необязательный)
  - Замена реактивных вызовов (createSignal) на оптимизированный код
  - Tree-shaking на уровне компонентов

Задачи компиляторов:

Зависят от фреймворка.

Qwik

- Resumable-компилятор
  - Разбивает приложение на «ленивые» куски
  - Генерирует HTML с «точками возобновления»
  - Оптимизирован для мгновенной загрузки

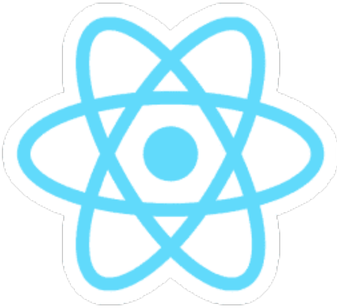
Задачи компиляторов:

Зависят от фреймворка.

Astro

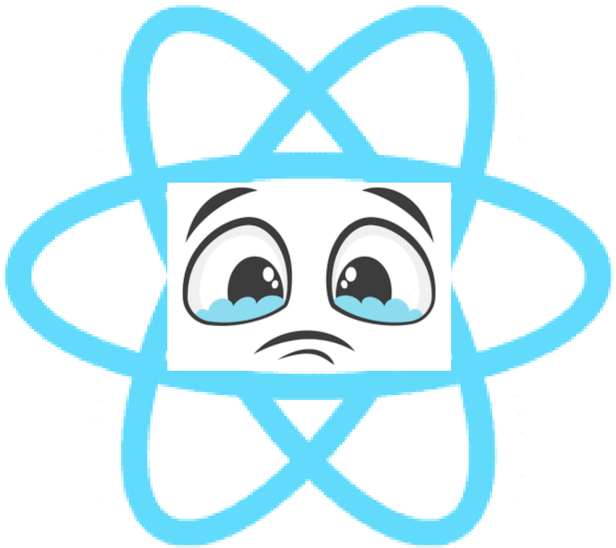
- Преобразует .astro-компоненты в статический HTML + JS
- Опциональная «островная архитектура» (Islands)
- Поддержка множества фреймворков (React, Vue внутри)

Задачи компиляторов:

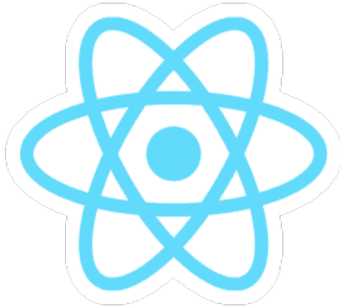


Зависят от фреймворка.

React



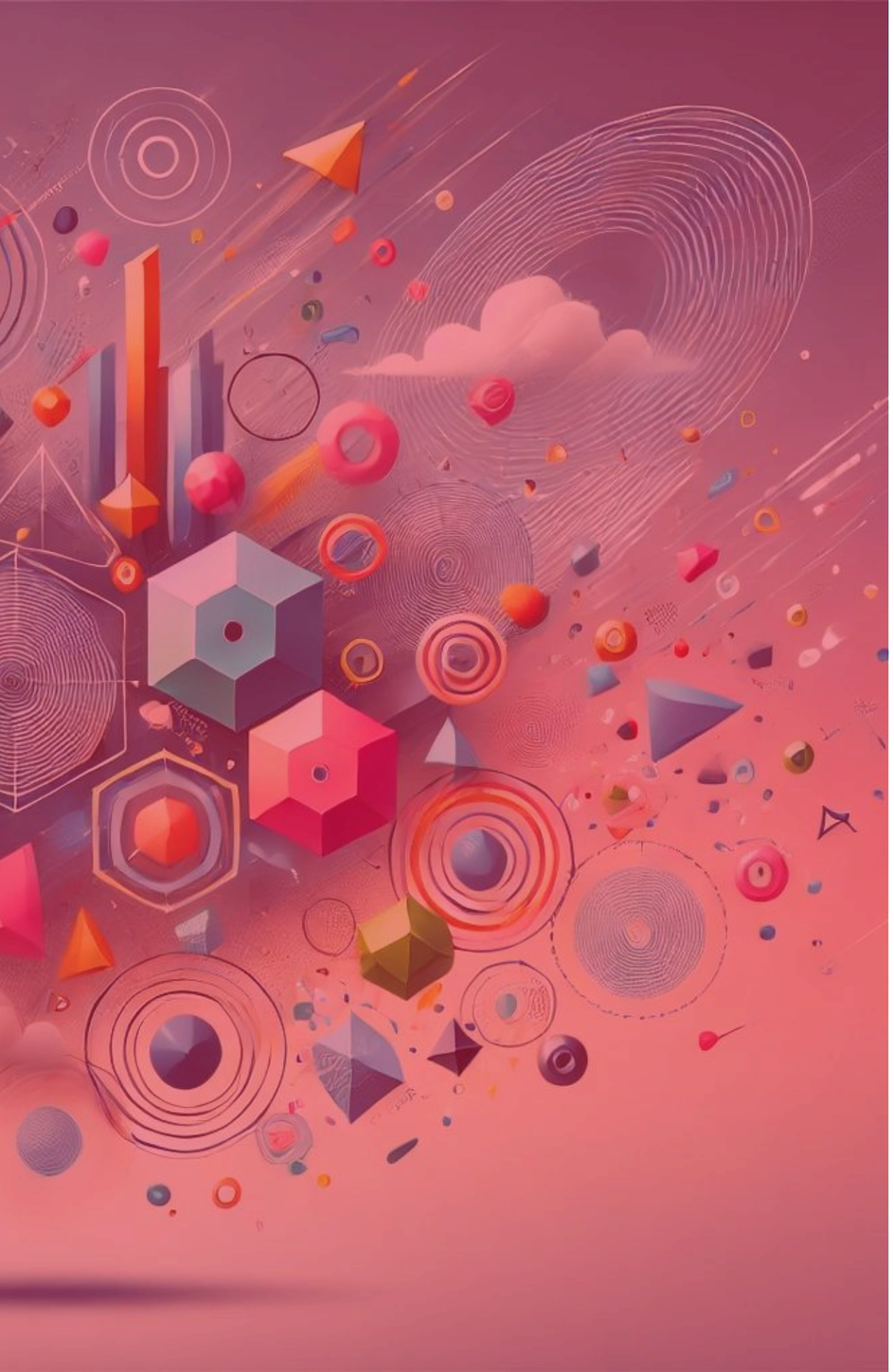
Задачи компиляторов:



Зависят от фреймворка.

React

- React Compiler в статусе Release Candidate
- Оптимизирует React код на этапе сборки
- Призван решить проблему ручной мемоизации



Основные концепции

- Шаблоны
- Состояние и Реактивность
- Работа с DOM
- Компиляторы
- **Стратегии рендера**
- Кроссплатформенность
- DX и экосистема

Стратегии рендера

- Client-Side Rendering (CSR)
  - React, Vue, Angular, …
- Server-Side Rendering (SSR)
  - Next.js (React), Nuxt (Vue), SvelteKit, …
- Static Site Generation (SSG)
  - Astro, Gatsby, Hugo, …
- Incremental Static Regeneration (ISR)
  - Next.js, Vercel
- Resumability
  - Qwik
- …

Стратегии рендера

- **Client-Side Rendering (CSR)**
  - React, Vue, Angular, ...
- Server-Side Rendering (SSR)
  - Next.js (React), Nuxt (Vue), SvelteKit, ...
- Static Site Generation (SSG)
  - Astro, Gatsby, Hugo, ...
- Incremental Static Regeneration (ISR)
  - Next.js, Vercel
- Resumability
  - Qwik
- ...

- Браузер загружает пустой HTML, затем JS-фреймворк строит интерфейс.

Плюсы:

- Быстрая навигация после первой загрузки.

Минусы:

- Плохой SEO (изначально пустой HTML);
- Долгая первоначальная загрузка.

Стратегии рендера

- Client-Side Rendering (CSR)
  - React, Vue, Angular, ...
- **Server-Side Rendering (SSR)**
  - Next.js (React), Nuxt (Vue), SvelteKit, ...
- Static Site Generation (SSG)
  - Astro, Gatsby, Hugo, ...
- Incremental Static Regeneration (ISR)
  - Next.js, Vercel
- Resumability
  - Qwik
- ...

- Сервер генерирует готовый HTML для каждой страницы и отправляет его браузеру.
- JS «оживляет» страницу уже на клиенте (гидратация).

Плюсы:

- Отличный SEO (контент в HTML).
- Быстрая первая загрузка.

Минусы:

- Серверная нагрузка.
- Задержки при гидратации.

Стратегии рендера

- Client-Side Rendering (CSR)
  - React, Vue, Angular, ...
- Server-Side Rendering (SSR)
  - Next.js (React), Nuxt (Vue), SvelteKit, ...
- **Static Site Generation (SSG)**
  - Astro, Gatsby, Hugo, ...
- Incremental Static Regeneration (ISR)
  - Next.js, Vercel
- Resumability
  - Qwik
- ...

- HTML генерируется на этапе сборки.
- Готовые страницы загружаются с CDN.

Плюсы:

- Максимальная скорость (нет запросов к серверу).
- Идеально для SEO.

Минусы:

- Не подходит для часто меняющегося контента.

Стратегии рендера

- Client-Side Rendering (CSR)
  - React, Vue, Angular, ...
- Server-Side Rendering (SSR)
  - Next.js (React), Nuxt (Vue), SvelteKit, ...
- Static Site Generation (SSG)
  - Astro, Gatsby, Hugo, ...
- **Incremental Static Regeneration (ISR)**
  - Next.js, Vercel
- Resumability
  - Qwik
- ...

- Страницы генерируются статически, но могут обновляться через заданный интервал.

Плюсы:

- Скорость SSG + актуальность контента.

Минусы:

- Сложнее настройка.

Стратегии рендера

- Client-Side Rendering (CSR)
  - React, Vue, Angular, ...
- Server-Side Rendering (SSR)
  - Next.js (React), Nuxt (Vue), SvelteKit, ...
- Static Site Generation (SSG)
  - Astro, Gatsby, Hugo, ...
- Incremental Static Regeneration (ISR)
  - Next.js, Vercel
- **Resumability**
  - Qwik
- ...

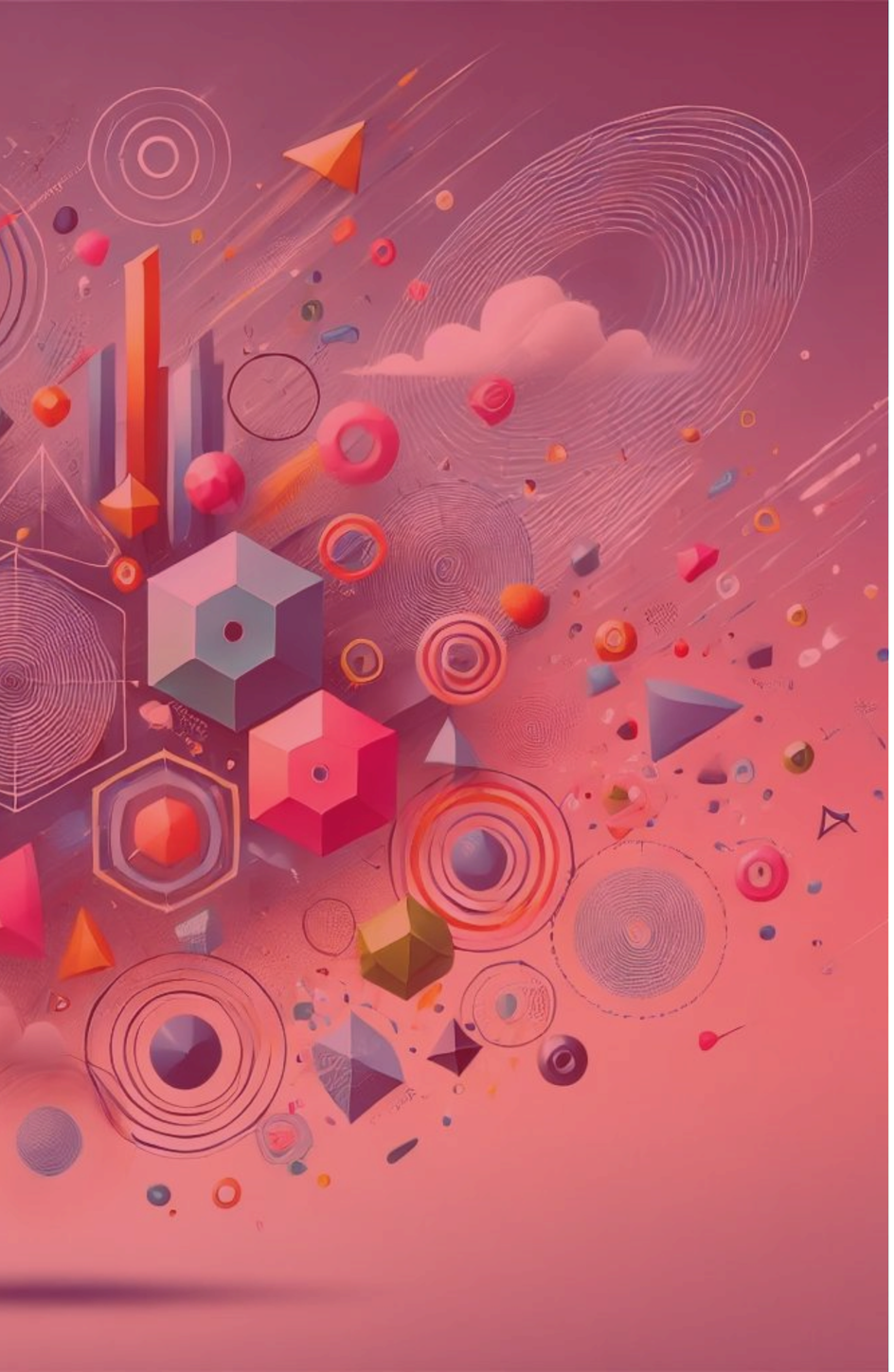
- Сервер отправляет HTML + «замороженное» состояние приложения.
- JS «просыпается» только при взаимодействии (нет гидратации).

Плюсы:

- Мгновенная загрузка.

Минусы:

- Новая концепция.



Основные концепции

- Шаблоны
- Состояние и Реактивность
- Работа с DOM
- Компиляторы
- Стратегии рендера
- **Кроссплатформенность**
- DX и экосистема

Кроссплатформенность

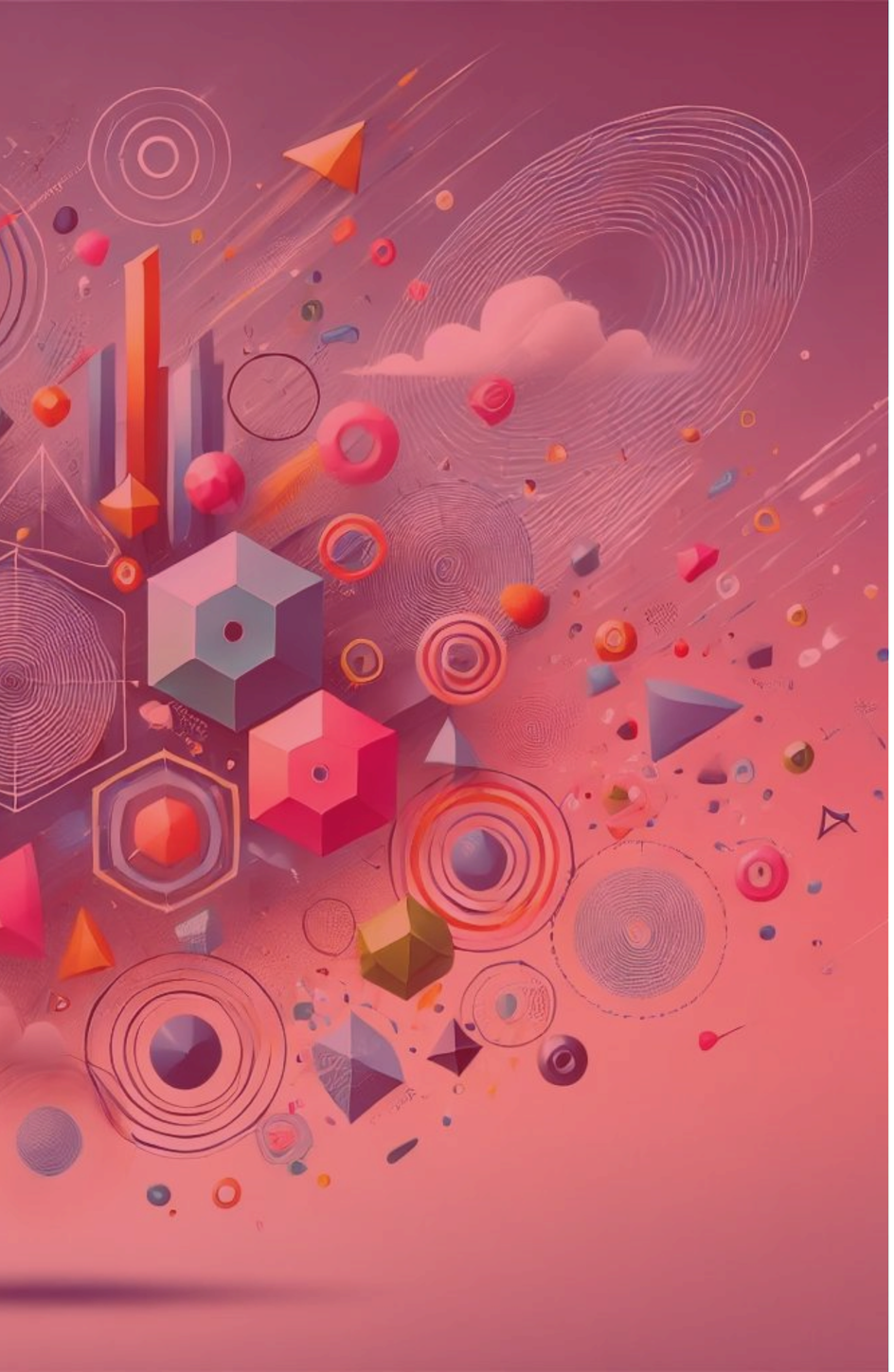
Вам какая именно нужна? :)

- Поддержка мобильных устройств?
- Компиляция в нативный код под iOS/Android/Web?
- Десктоп?
- PWA?

Кроссплатформенность

Вам какая именно нужна? :)

- Поддержка мобильных устройств
  - React Native (React)
  - Capacitor/Ionic (Angular, Vue, React)
- Компиляция в нативный код под iOS/Android/Web
  - Flutter Web
- Десктоп
  - Electron/Tauri
- PWA



Основные концепции

- Шаблоны
- Состояние и Реактивность
- Работа с DOM
- Компиляторы
- Стратегии рендера
- Кроссплатформенность
- **DX и экосистема**

DX — опыт разработчика

Вкусовщина :)

DX — опыт ~~раз~~работчика

DX — задает порог вхождения

- Удобно ли писать код;
- Как легко научиться его писать (документация);
- Сообщество.

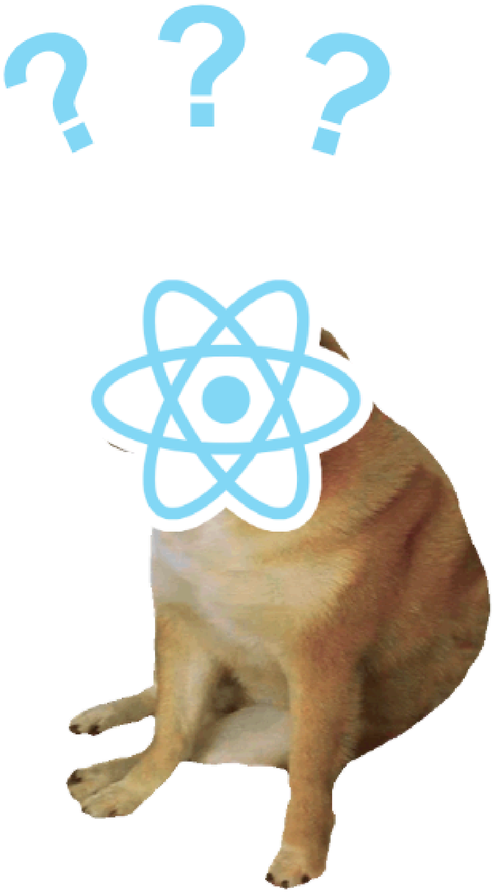
Роутинг

SSR

SM

...

Валидаторы



Надоело готовить!

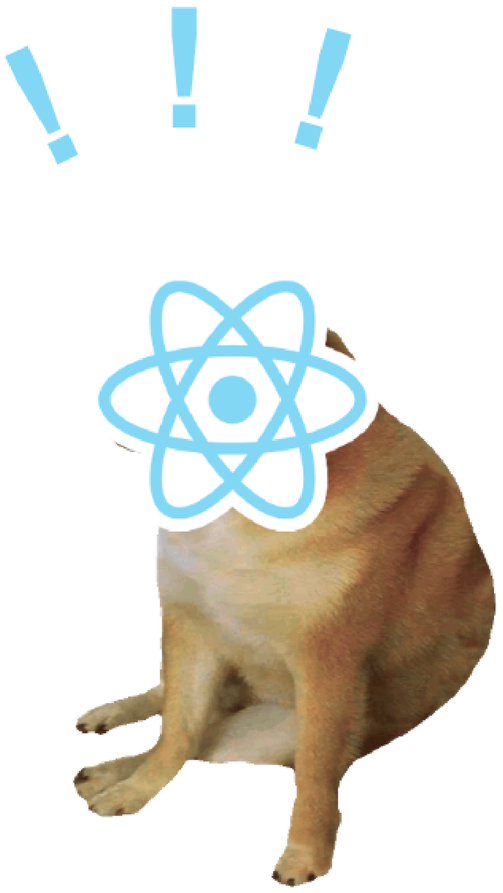
Роутинг

SSR

SSG

...

Serverless

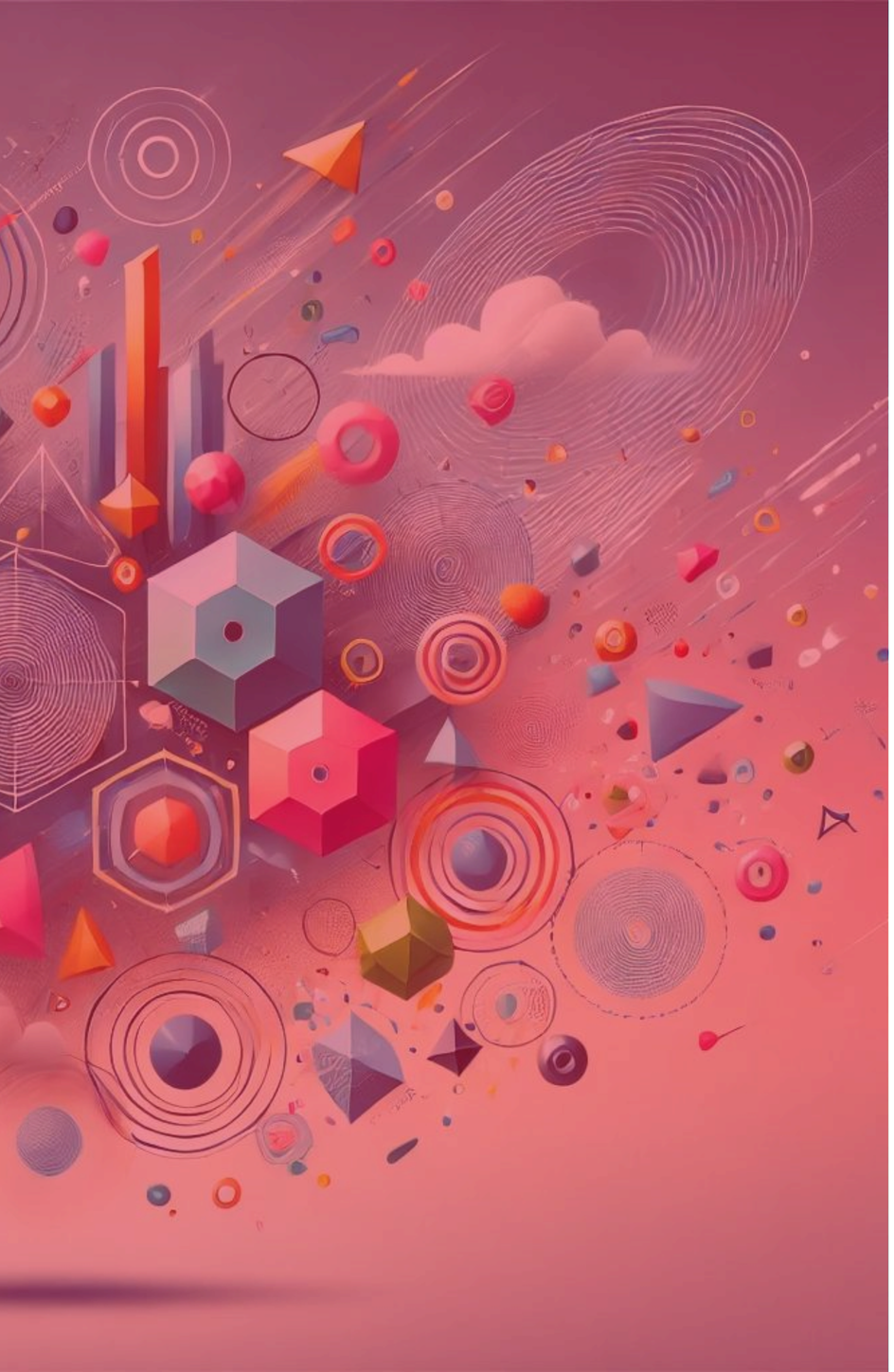


Метафреймворки!

- Надстройка на базе существующего.
- Расширяет функциональность, поставляемую из коробки.
- В частности добавляя SSR.

Метафреймворки!

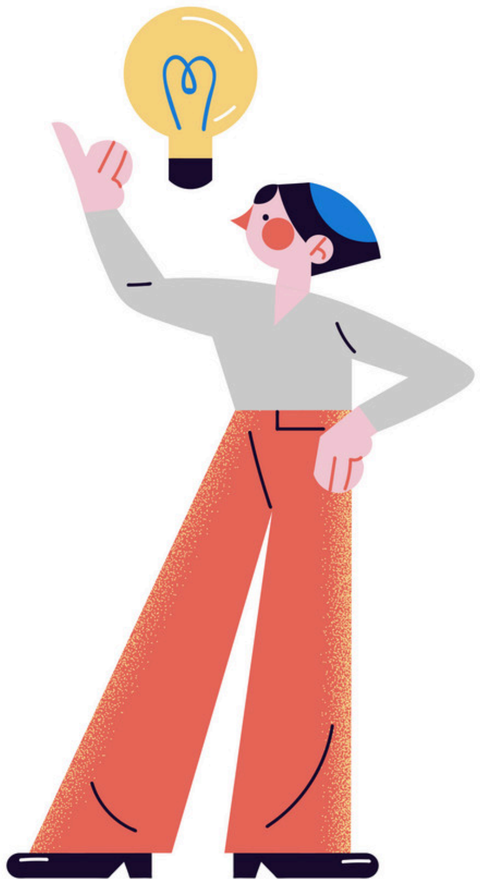
Next.js Gatsby Remix	Nuxt.js Gridsome VuePress	SvelteKit	SolidStart	Qwik City	Analog



Вопросы на засыпку

Какие фреймворки используют JSX?

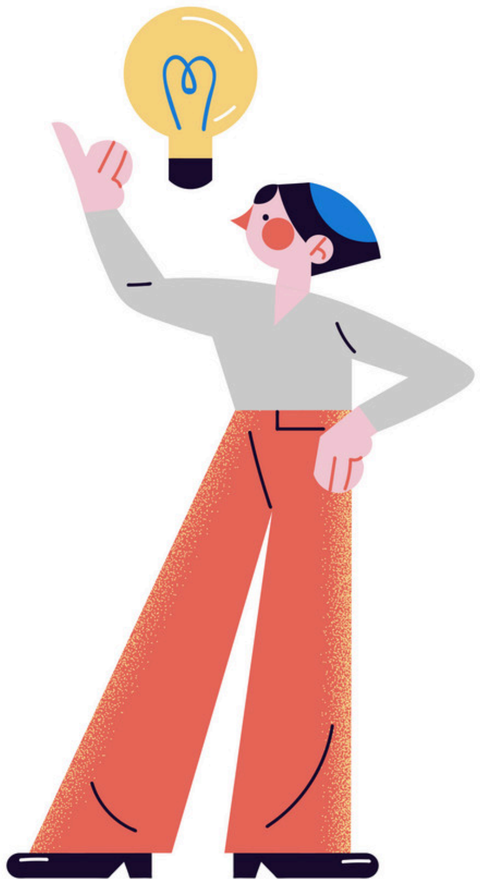
Какие фреймворки используют JSX?



- React
- Preact
- Solid

Кому не нужен сборщик?

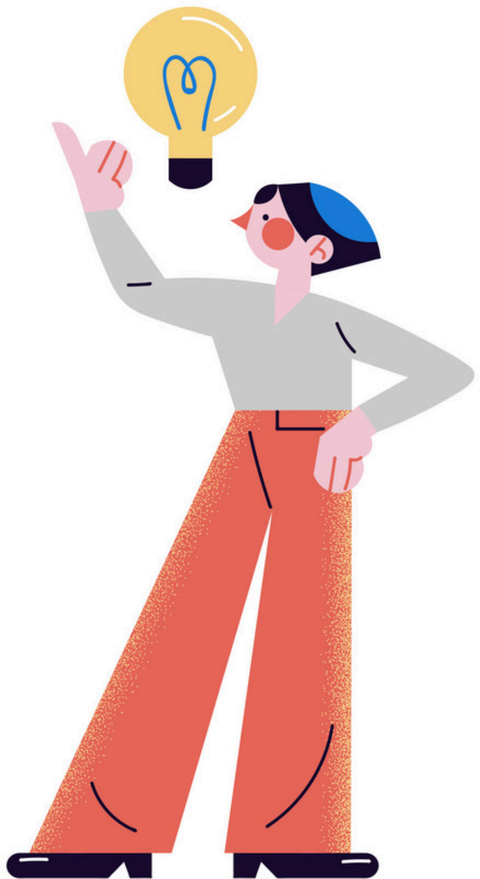
Кому не нужен сборщик?



- HTMX
- Lit
- Alpine.js
- Svelte (режим REPL)
- ...

Кто позволяет использовать все фреймворки сразу?

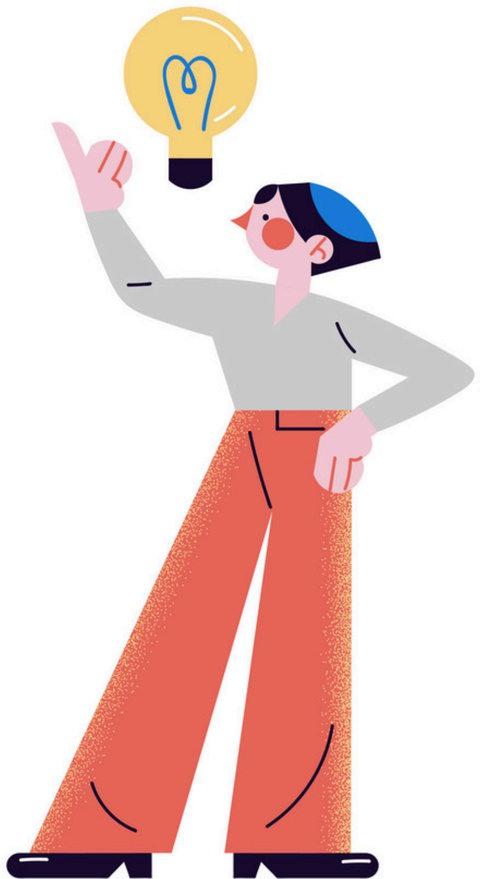
Кто позволяет использовать все фреймворки сразу?



- Astro
- Vike
- ...

Какой фреймворк позволяет запоминать действия пользователей до гидратации страницы?

Какой фреймворк позволяет запоминать действия пользователей до гидратации страницы?

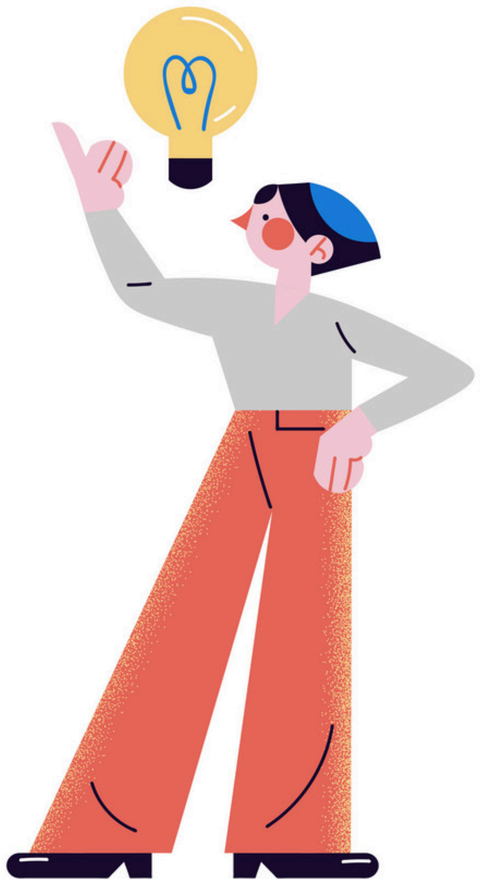


- Angular, фича **Event Replay**

```
bootstrapApplication(AppComponent, {
  providers: [provideClientHydration(withEventReplay())]
});
```

Какой фреймворк позволяет лейзилоадить аж по функциям?

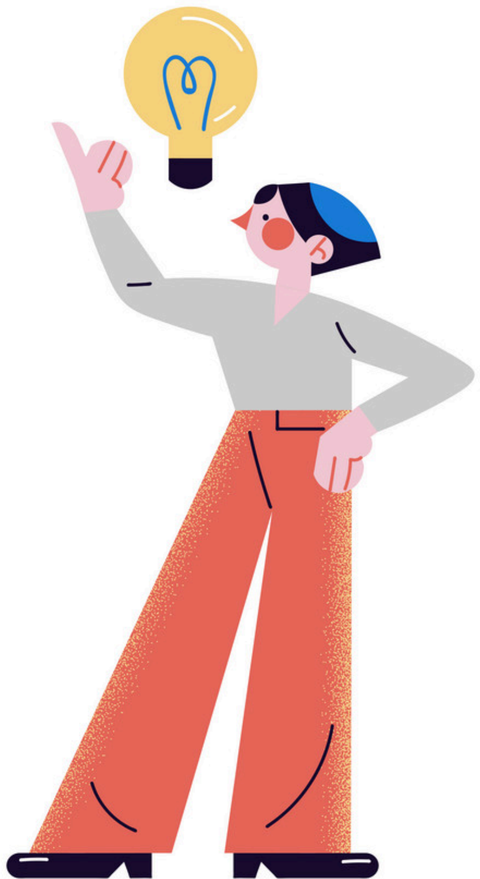
Какой фреймворк позволяет лейзилоадить аж по функциям?



- Qwik, фича **QRL**

```
export default component$(() => {
  useStyles$(AboutStyles)

  const modalVisible = useSignal(false)

  const closeModal = $(() => {
    modalVisible.value = false
  })
```

Какой фреймворк позволяет лейзилоадить аж по функциям?

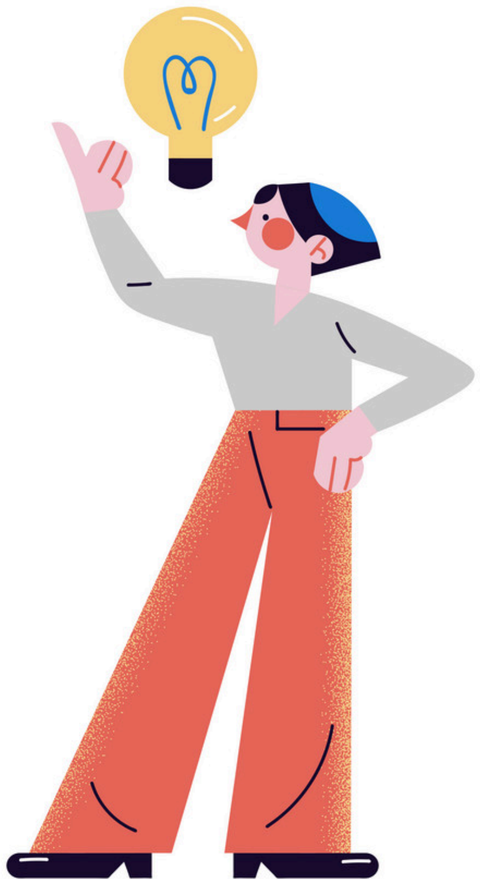


- Qwik, фича **QRL**

- URL для ленивой загрузки.
- Указывает, где в чанке находится код.
- Под капотом у оптимизатора.
- Можно использовать явно.

Как писать компоненты под любой фреймворк?

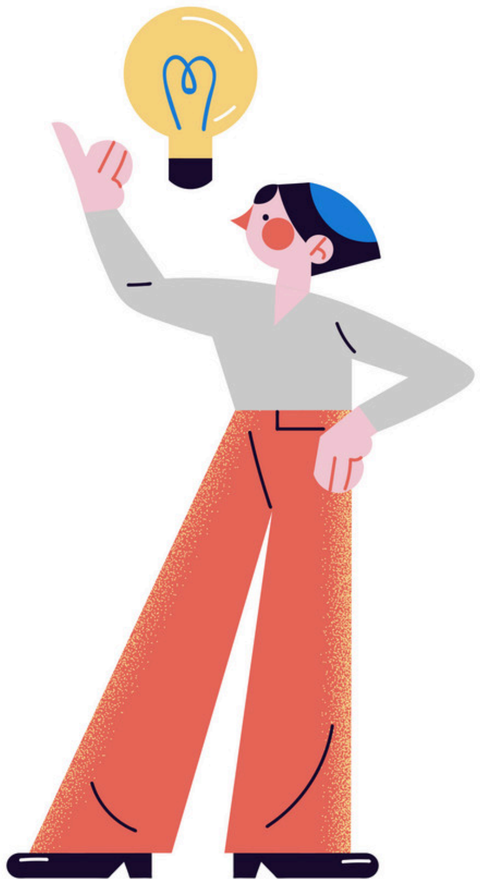
Как писать компоненты под любой фреймворк?



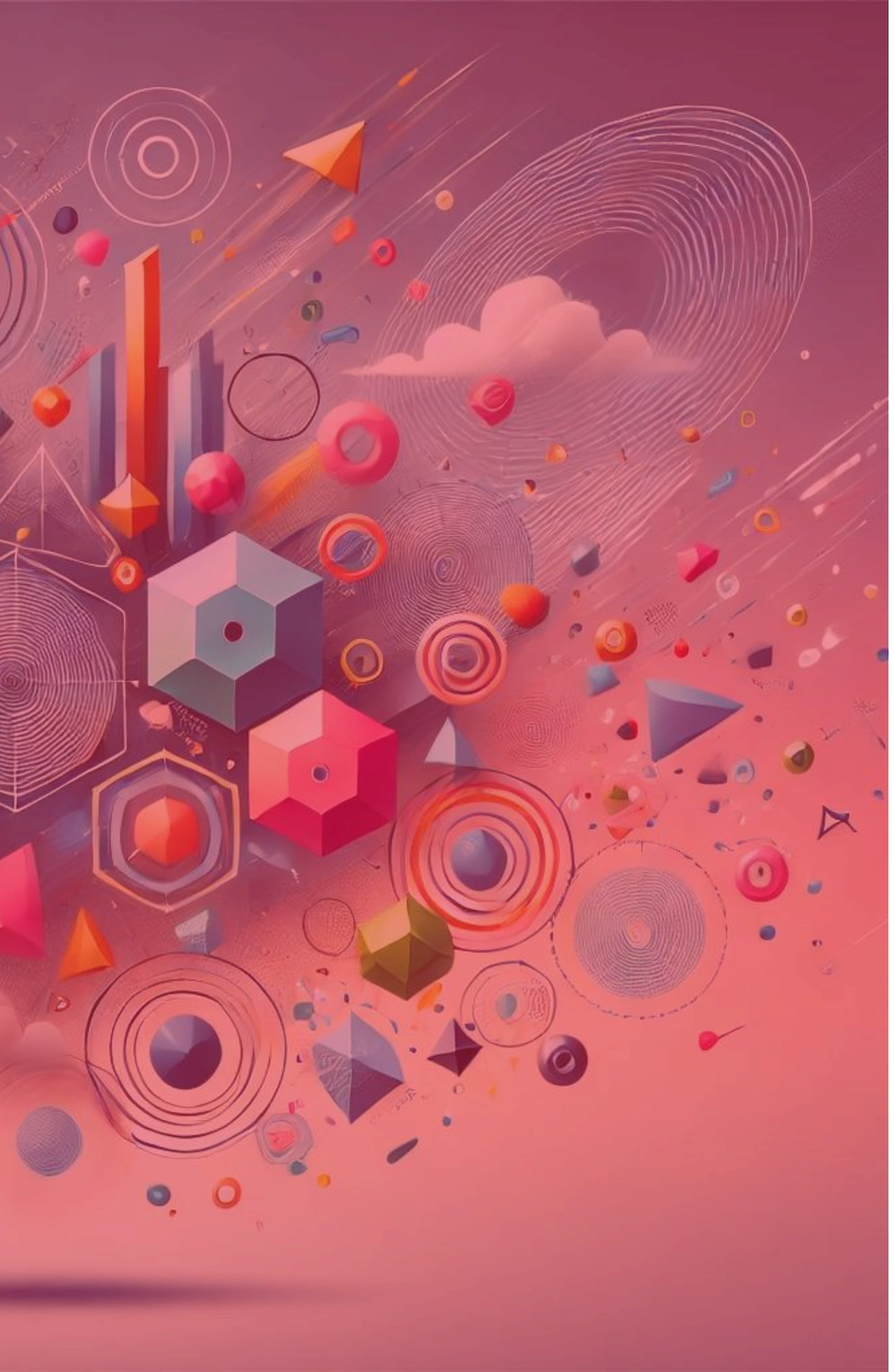
- Web-components!

Какие фреймворки работают с web-components?

Какие фреймворки работают с web-components?

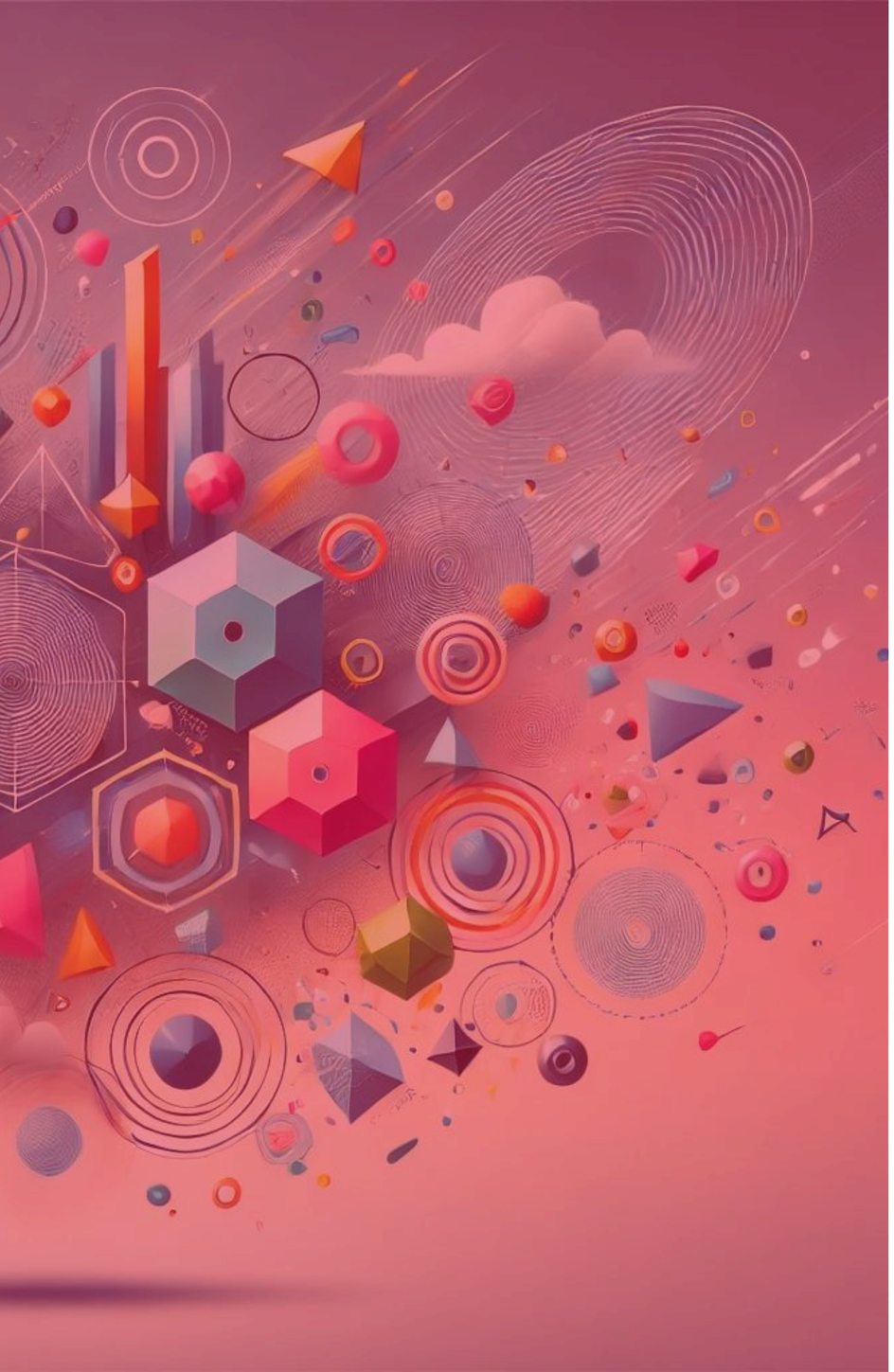


- Lit;
- Stencil;
- ...



Выбор

- Технические требования
- Команда и трудозатраты
- Корп. экосистема

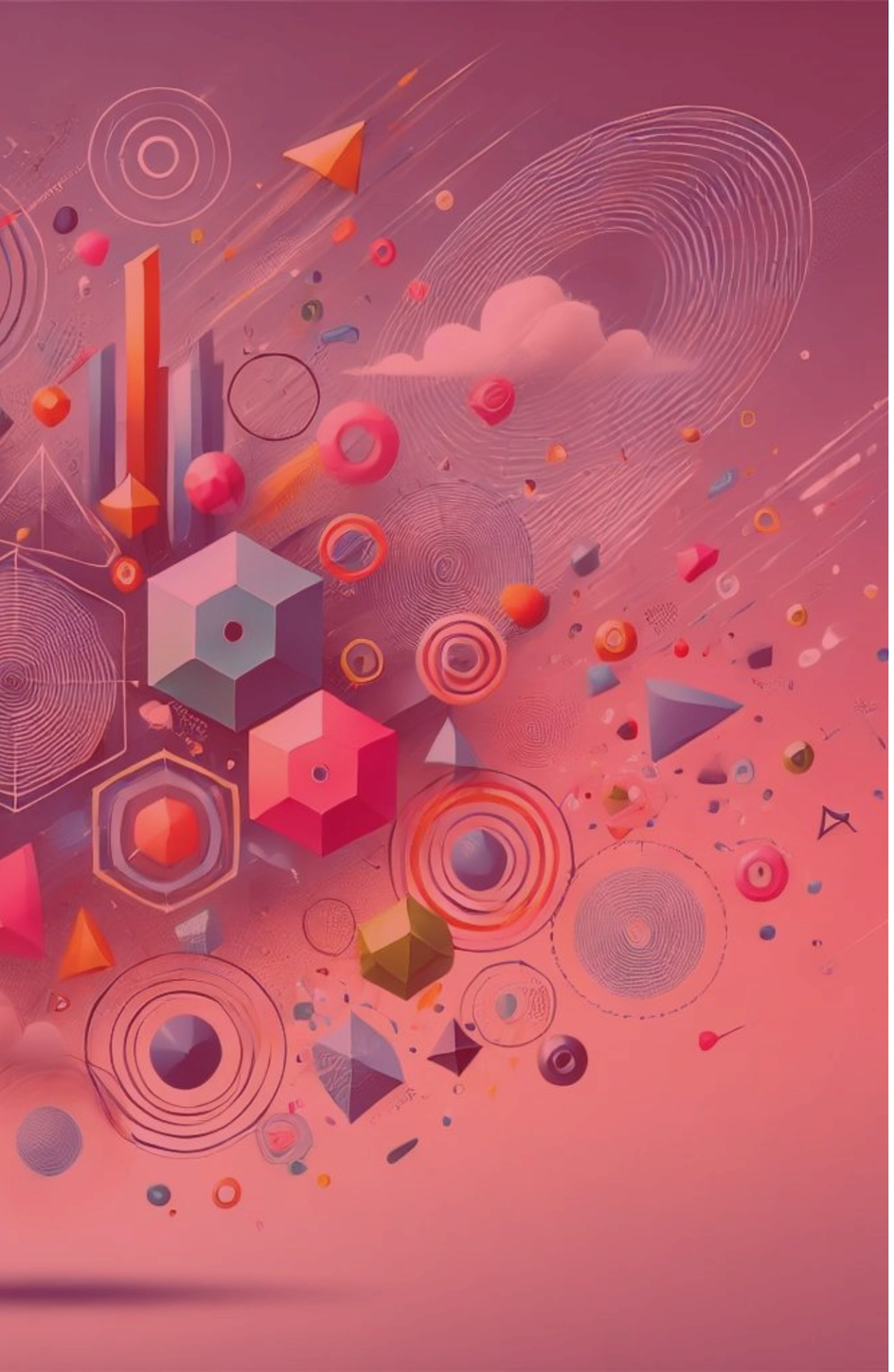


Выбор

- **Технические требования**
- Команда и трудозатраты
- Корп. экосистема

Технические требования

- Сложность
- Особые фичи
- Масштабируемость
- Интерактивность
- Производительность
- SEO, SSR
- Мобильная разработка



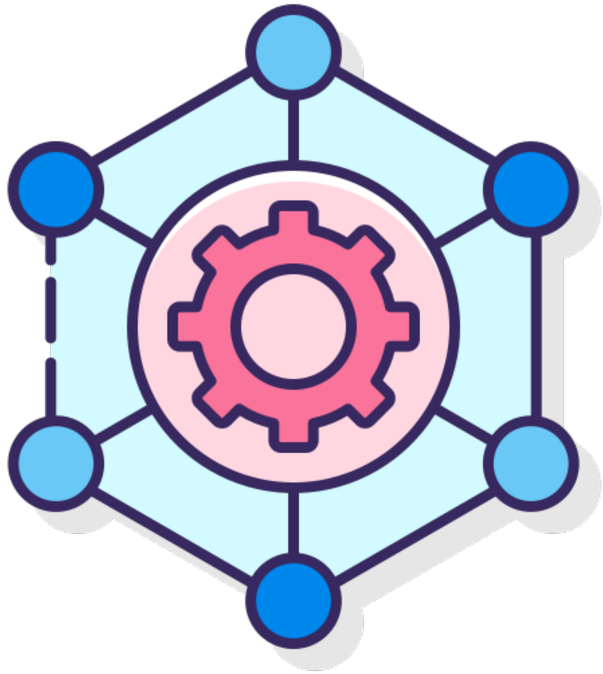
Выбор

- Технические требования
- **Команда и трудозатраты**
- Корп. экосистема

Команда и трудозатраты

Со стороны команды:

Со стороны фреймворка:



Команда и трудозатраты

Со стороны команды:

- Опыт в конкретном фреймворке
- Опыт построения фронтенд архитектуры
- Дедлайны и поддержка
- Стоимость разработки
- Стоимость поддержки

Со стороны фреймворка:

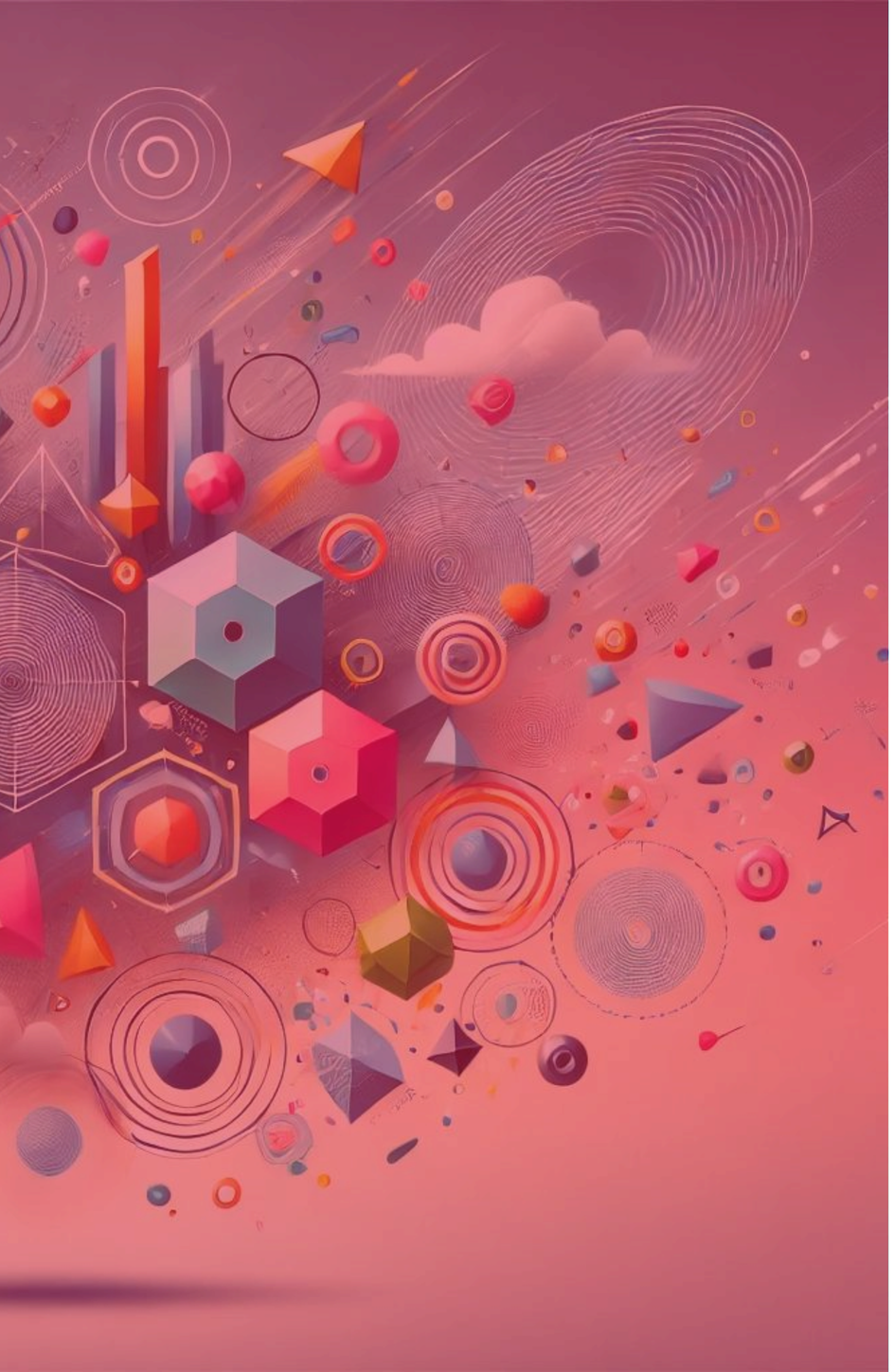
Команда и трудозатраты

Со стороны команды:

- Опыт в конкретном фреймворке
- Опыт построения фронтенд архитектуры
- Дедлайны и поддержка
- Стоимость разработки
- Стоимость поддержки

Со стороны фреймворка:

- DX
- Документация
- Сообщество
- Экосистема тулинга
- Философия



Выбор

- Технические требования
- Команда и трудозатраты
- **Корп. экосистема**

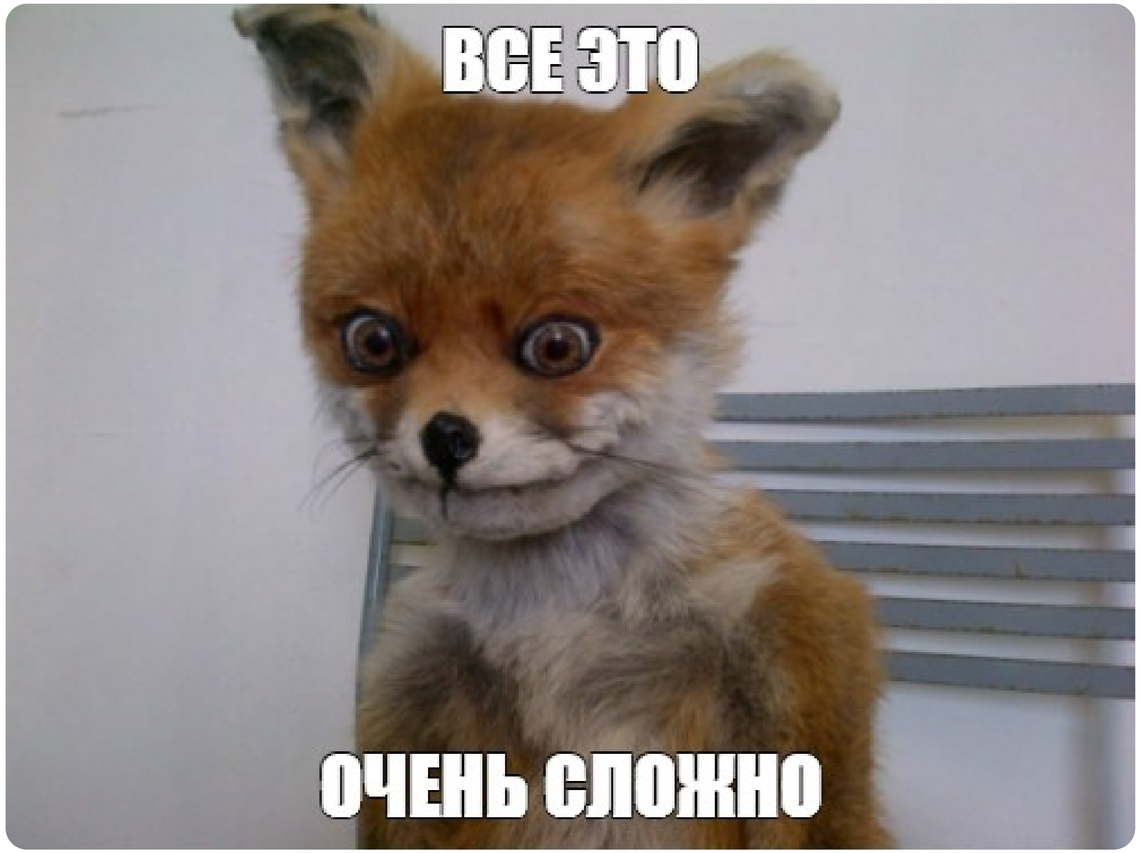
Корп. экосистема

- Есть ли корпоративные стандарты?
- Есть ли экосистема корпоративного туллинга?

Корп. экосистема

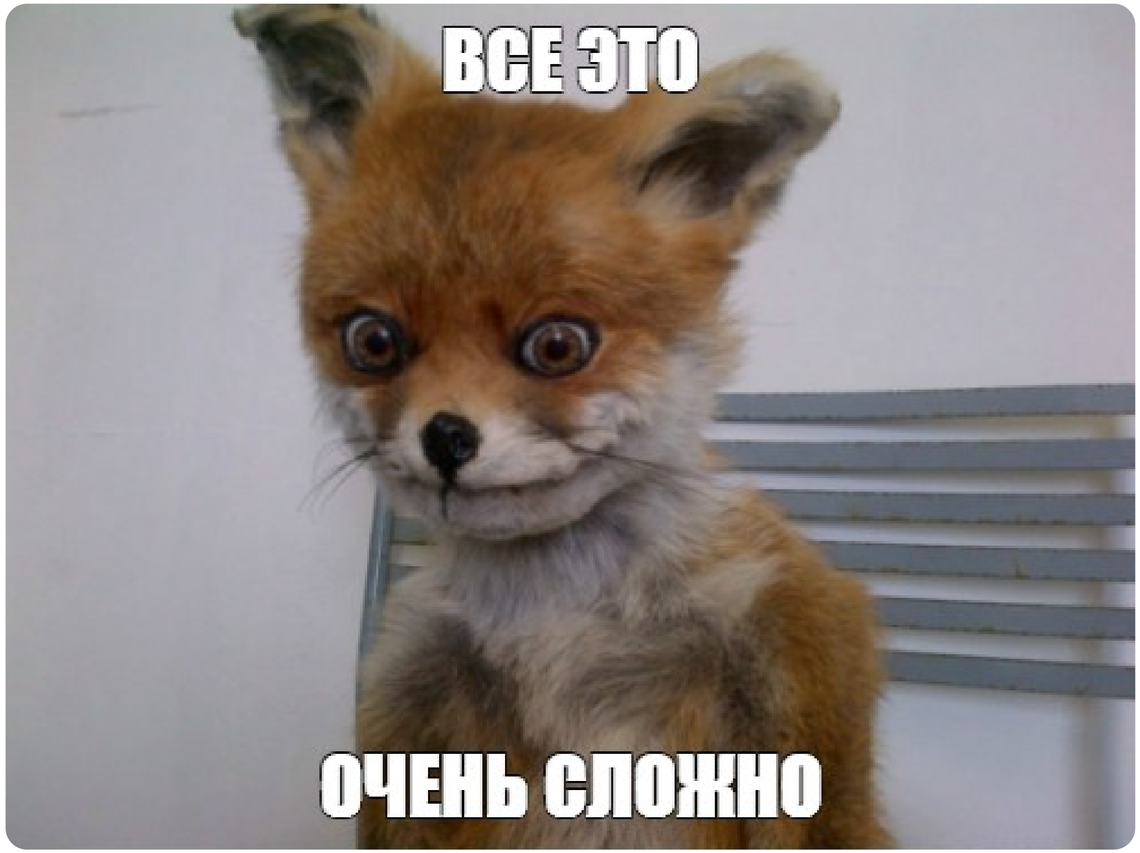
- Есть ли корпоративные стандарты?
- Есть ли экосистема корпоративного туллинга?
- Хотите ее создать?

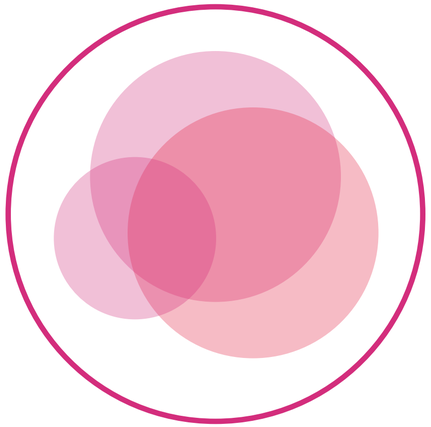
Слишком много всего!



Слишком много всего!

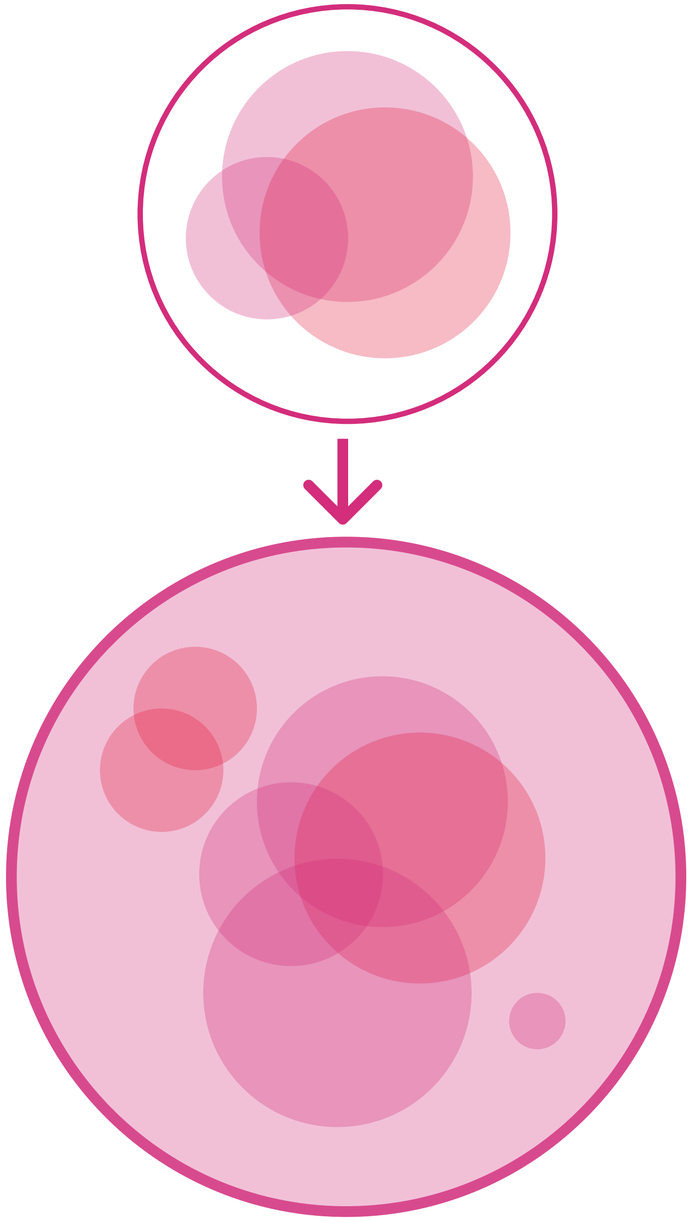
Определите
самые **критичные**
2-3 пункта





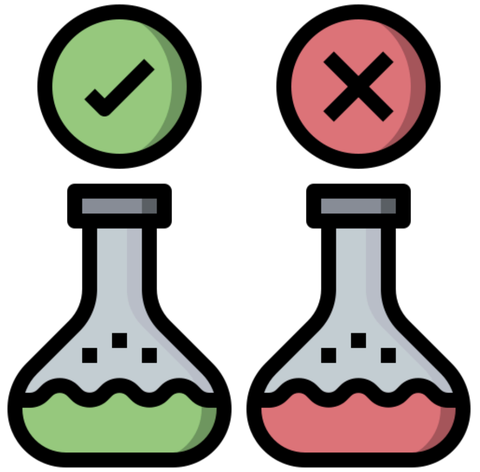
Выбор

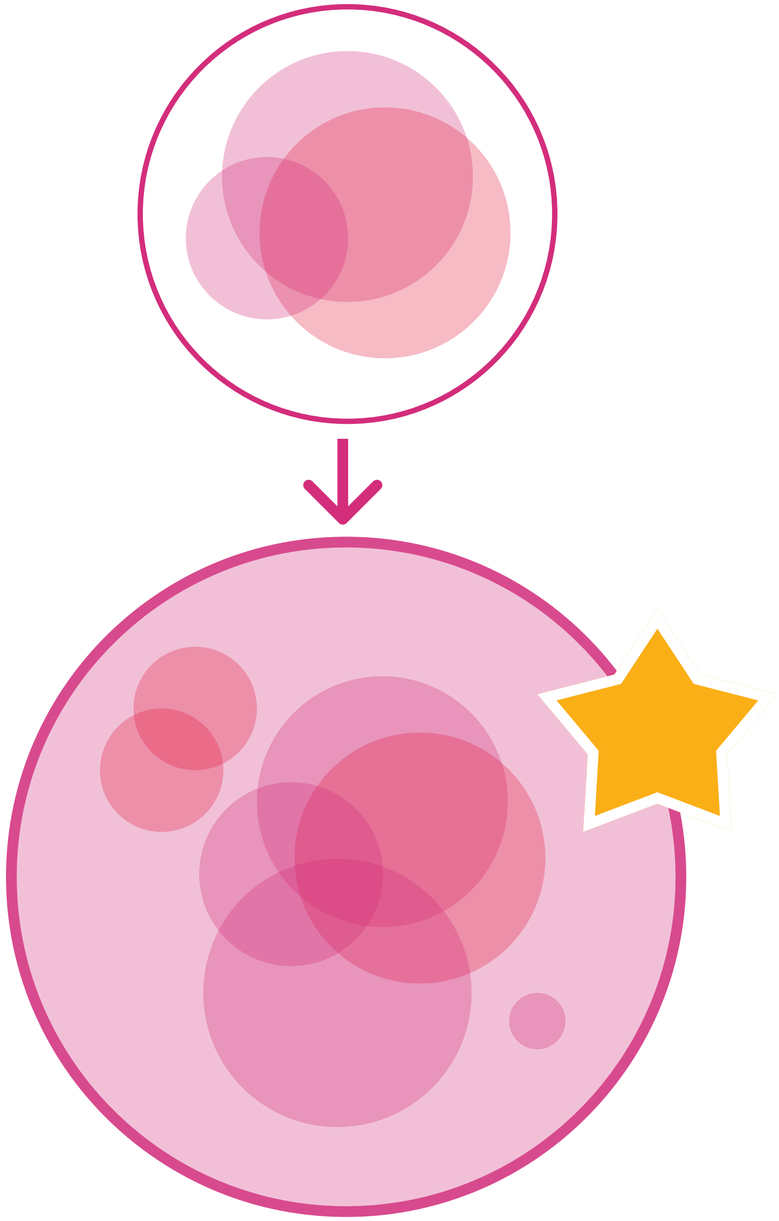
- **Технические требования**
- Команда и трудозатраты
- Корп. экосистема



Выбор

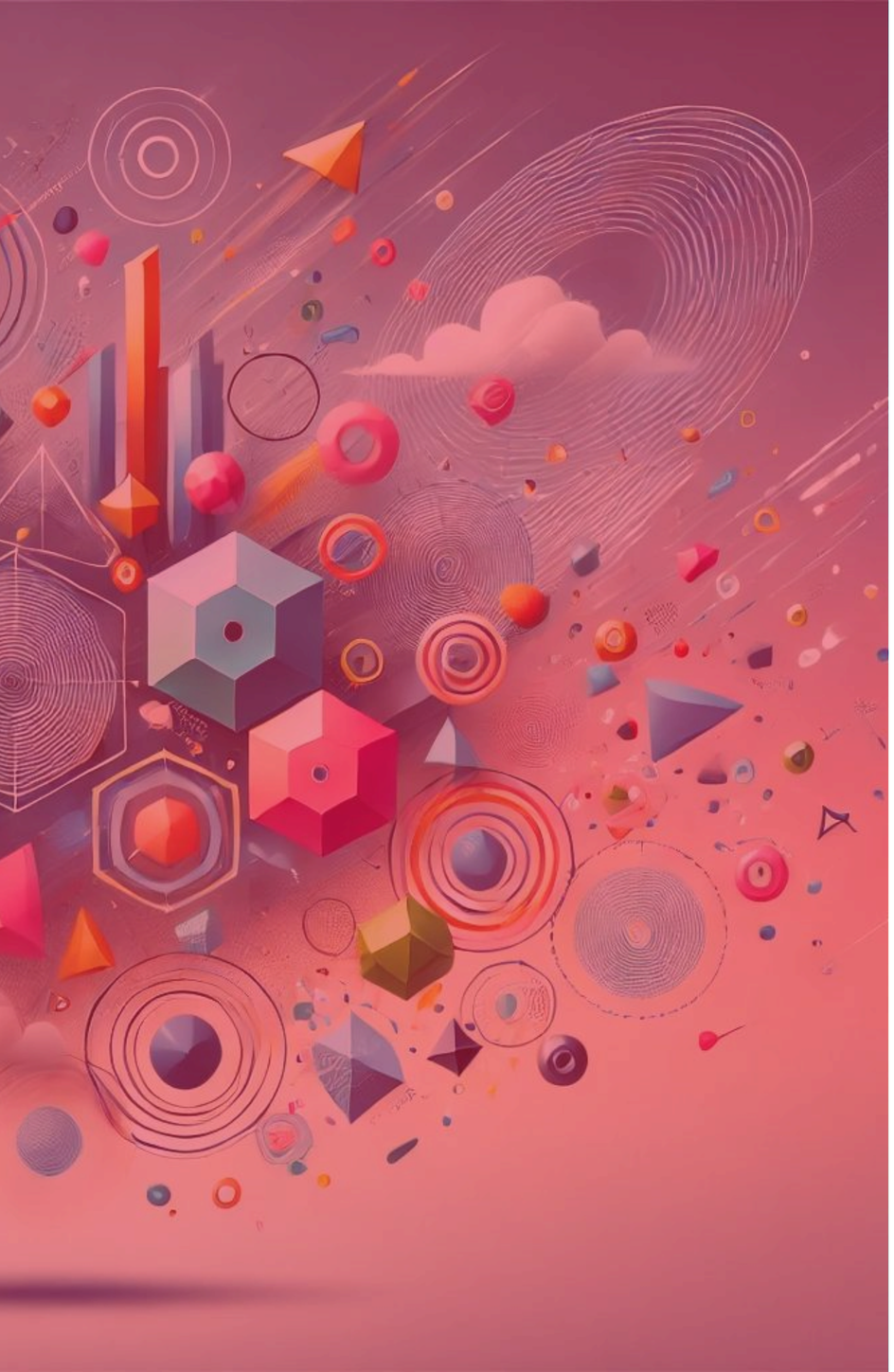
- Технические требования
- Команда и трудозатраты
- **Корп. экосистема**





Выбор

- **Технические требования**
- Команда и трудозатраты
- Корп. экосистема



Куда все идет?

Фреймворки переопыляются!

- Когда-то: классовые компоненты Angular, Vue, React.
- Сейчас: тренд на ФП.
- Vue утащил часть синтаксиса Angular.
- В react-router появилась loader функция как в Angular.
- Alpine частично использует основу реактивности Vue.
- Многие хотят компиляторы.
- ...

Идеальный фреймворк

Такого не существу...

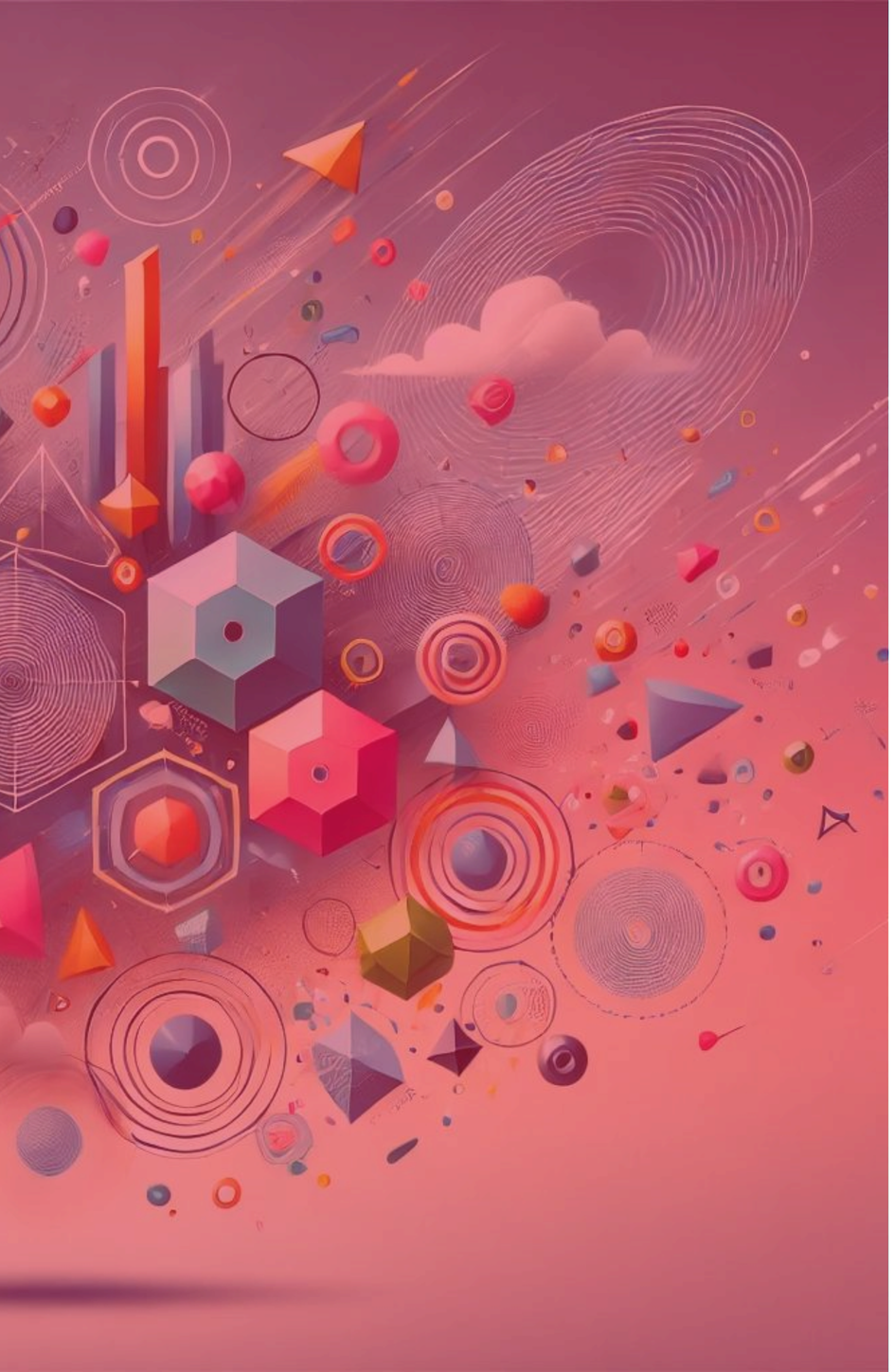
Идеальный фреймворк

- Писать максимально нативный HTML, CSS, JS.
- Компилируется и исчезает.
- Невесомый бандл.
- Fine-graned реактивность (Signals?).
- «База из коробки».
- Минимум проблем с SSR.
- Изучается за выходные.
- Нейронка хорошо пишет код на этом фреймворке.

Идеальный фреймворк

- Писать максимально нативный HTML, CSS, JS.
- Компилируется и исчезает.
- Невесомый бандл.
- Fine-graned реактивность (Signals?).
- «База из коробки».
- Минимум проблем с SSR.
- Изучается за выходные.
- Нейронка хорошо пишет код на этом фреймворке.

Потому что выполняет свои задачи и все им пользуются



Итоги

Итоги

- Не забывайте про базу (JS, HTML, CSS).
- Изучайте принципы, лежащие в основе.
- Поймите задачи, которые решают фреймворки.
- Расширяйте кругозор.
- Экспериментируйте безопасно!

Все новое — это хорошо переиспользуемое старое

Контакты

@it_wildlife