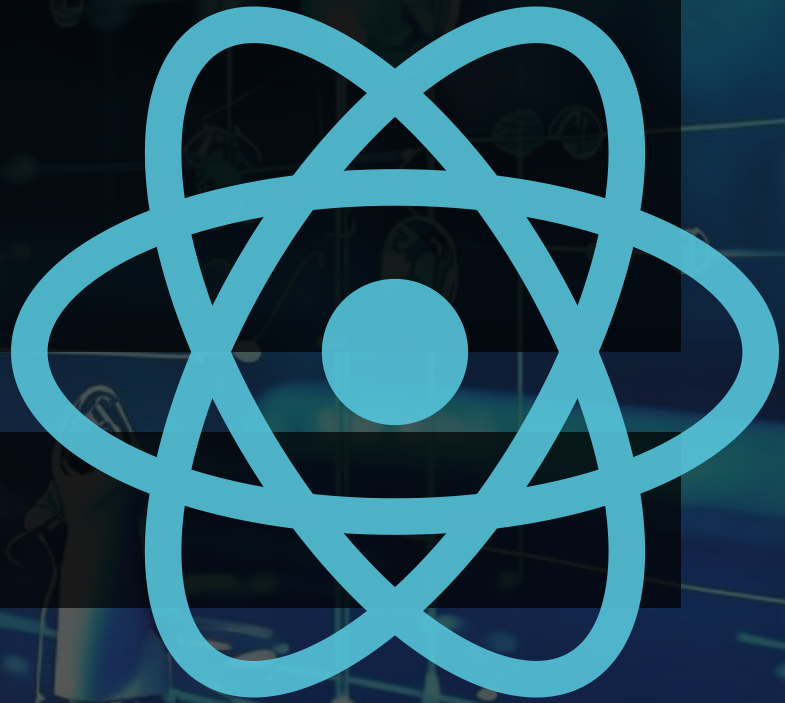
Жизнь до и после React Compiler

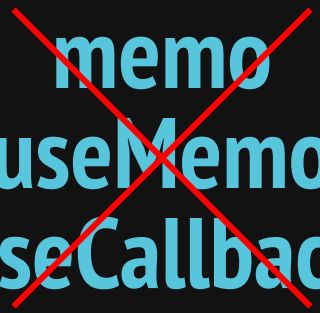


Анна Ширяева, СИБУР Цифровой

Обо мне

- Коллекционер хобби;
- В IT с 2011;
- C# > Angular > React;
- Ведущий frontend разработчик Цифрового СИБУРа;
- Организатор MocsowJS ❤️

memo
useMemo
useCallback

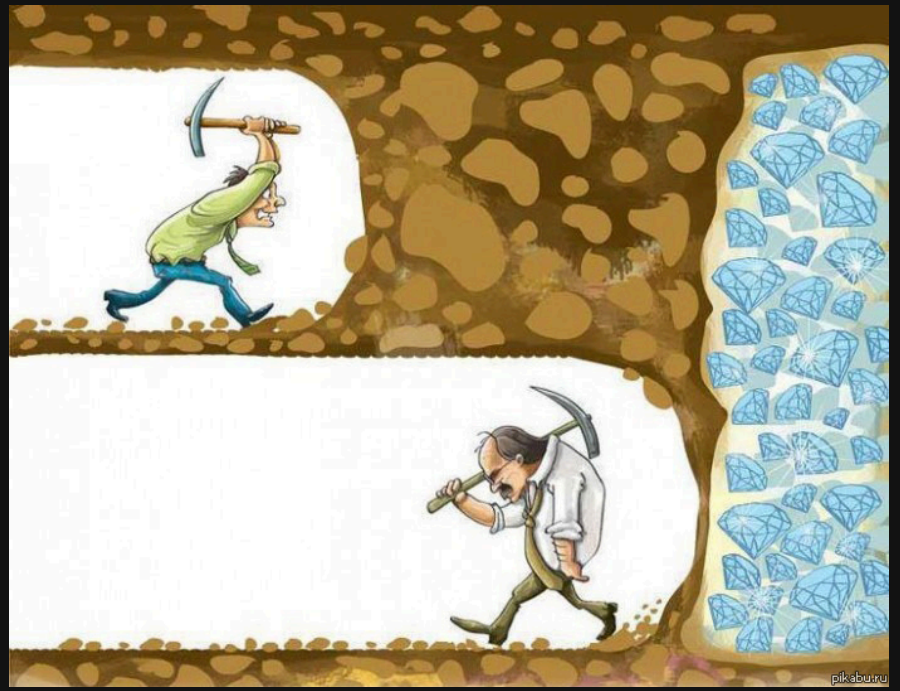

memo
useMemo
useCallback

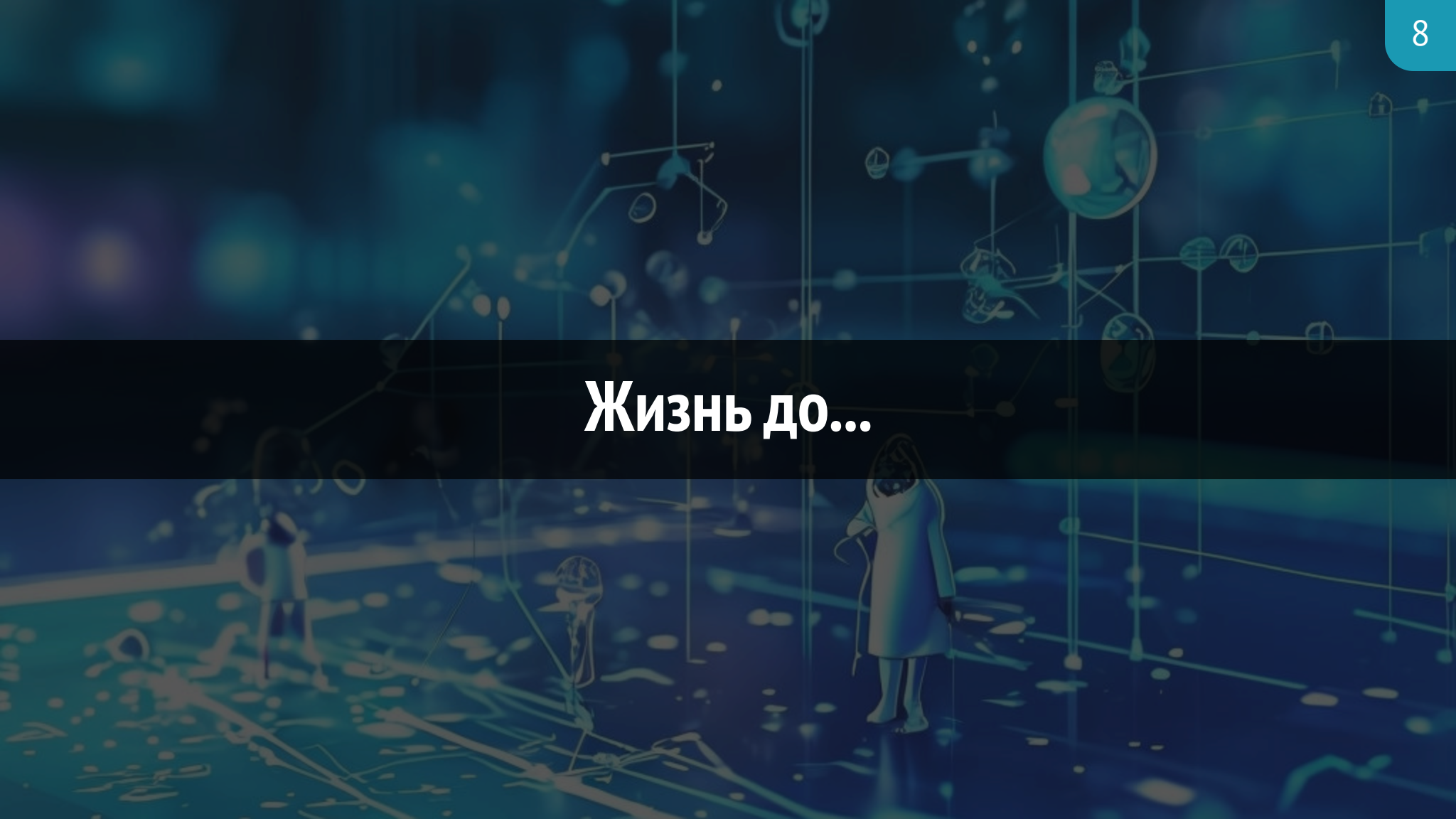

Я вам
ЗАПРЕЩАЮ
мемо

R & D

О докладе

- Жизнь до...
- Что такое React Compiler?
- Что он делает?
- Как поставить и кастомизировать?
- Почему это работает?
- На что влияет?
- Роадмап, развитие и проблемы
- Выводы





Жизнь до...

Жизнь сейчас

В начале был React...

- Не MVC framework
- Библиотека рендеринга
- Декларативный подход
- Фокус на UI компоненты, а не шаблоны
- Возможность использовать композицию компонентов

Хотим

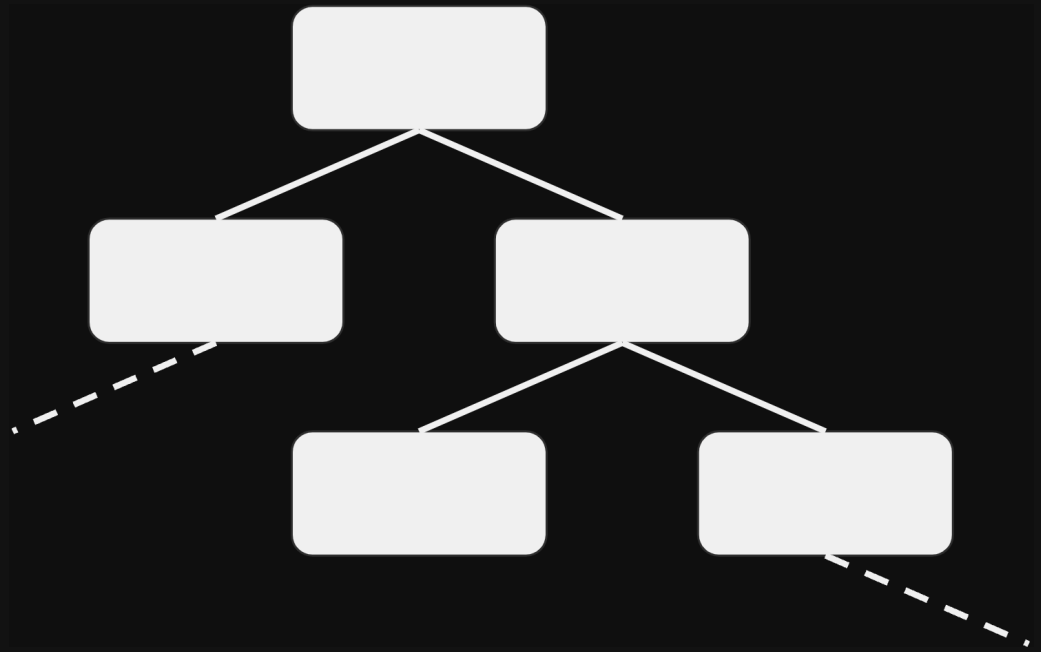
- Хранить состояние
- Отслеживать его изменение
- Выполнять ререндеры при наличии изменений
- Все было быстро
- Меньше кода

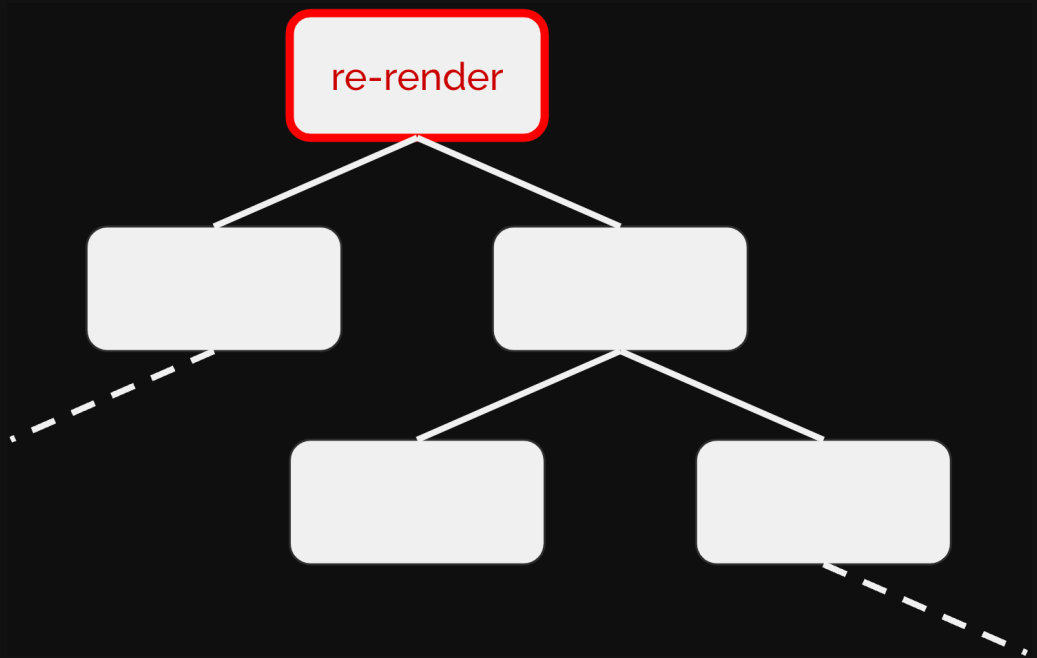
То есть...

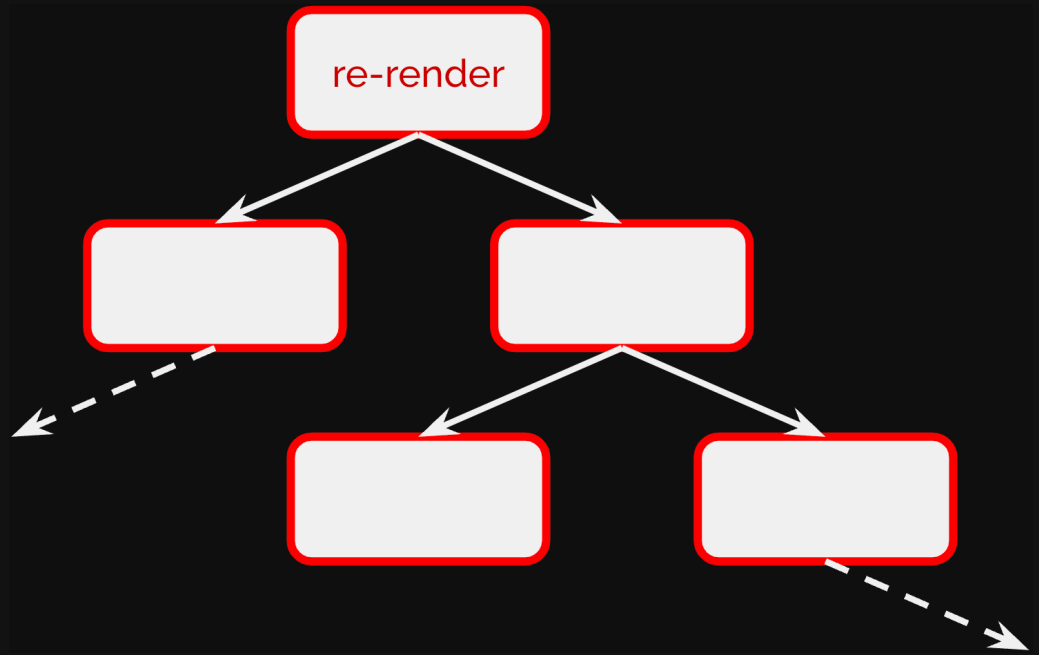
- State management
- Reactivity
- DOM updates
- Performance
- DX: Developer Experience

Что может пойти не так?

state changes = re-renders




re-render


re-render

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback

```
const GreetHolyJS = () => {
    const user = { name: 'UserName' };
    const onClick = () => {
        console.log('Hello!');
    }
    // еще логика

    return (
        <div>
            <User onClick={onClick} user={user} />
            {/* еще компоненты */}
        </div>
    )
}

const User = ({ onClick, user }) => {
    // ...
}
```

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback

```
const GreetHolyJS = () => {
    const user = useMemo(() => ({ name: 'UserName' }), []);
    const onClick = useCallback(() => {
        console.log('Hello!');
    }, []);
    // еще логика

    return (
        <div>
            <User onClick={onClick} user={user} />
            {/* еще компоненты */}
        </div>
    )
}

const User = React.memo(({ onClick, user }) {
   // ...
});
```

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback

```
const GreetHolyJS = () => {
    const user = useMemo(() => ({ name: 'UserName' }), []);
    const onClick = useCallback(() => {
        console.log('Hello!');
    }, []);
    // еще логика

    return (
        <div>
            <User onClick={onClick} user={user} />
            {/* еще компоненты */}
        </div>
    )
}

const User = React.memo(({ onClick, user }) {
   // ...
});
```

Умеем ли мы это готовить?

{ }, [ ], function ()


YEAH, I GOT THE MEMO....
makeameme.org

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback
- Composition

```
const ComplexParent = () => {
    const [count, setCount] = useState(0);

    return (
        <div>
            <button onClick={() => (setCount(prev => prev + 1))}>
                Clicked {count} times
            </button>
            <MoreStuff />
        </div>
    )
}
```

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback
- Composition

```
const ComplexParent = () => {
    const [count, setCount] = useState(0);

    return (
        <div>
            <button onClick={() => (setCount(prev => prev + 1))}>
                Clicked {count} times
            </button>
            <MoreStuff />
        </div>
    )
}
```

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback
- Composition

```
const ComplexParent = () => {
    return (
        <div>
            <ButtonWithCounter />
            <MoreStuff />
        </div>
    )
}

const ButtonWithCounter = () => {
    const [count, setCount] = useState(0);

    return (
        <button onClick={() => (setCount(prev => prev + 1))}>
            Clicked {count} times
        </button>
    )
}
```

Что там по качеству кода?

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback
- Composition
- Rules of React

```
let greeting = 'Hello';

const Welcome = () => {
    greeting = greeting + ' User!';

    return (<div>{greeting}</div>)
}
```

Например:

Сайд эффекты должны быть вне рендера

Что помогает сейчас

- Memoization:
  - memo
  - useMemo
  - useCallback
- Composition
- Rules of React

```
const Welcome = ({ greeting }) => {
    return (<div>{greeting}</div>)
}
```

Например:

Сайд эффекты должны быть вне рендера

То есть...

- State management
- Reactivity
- DOM updates
- Performance
- DX: Developer Experience

То есть...

- State management
- Reactivity
- DOM updates
- Performance + **Memo**
- DX: Developer Experience

То есть...

- State management
- Reactivity
- DOM updates
- Performance + **Memo**
- DX: Developer Experience

Универсальная таблица оценки задач
изян - 1ч
изи - 2ч
просто - 4ч
вроде просто - 6ч

То есть...

- State management
- Reactivity
- DOM updates
- Performance + **Memo**
- DX: Developer Experience

Универсальная таблица оценки задач
изян - 1ч
изи - 2ч
просто - 4ч
вроде просто - 6ч
норм - 8ч
норм так - 12ч
хз - 16ч
хз как-то - 20ч

HOLY JS/... 2024 Autumn

История реактивности

Артём Арутюнян

То есть...

- State management
- Reactivity
- DOM updates
- Performance + **Memo**
- DX: Developer Experience

Универсальная таблица оценки задач
изян - 1ч
изи - 2ч
просто - 4ч
вроде просто - 6ч
норм - 8ч
норм так - 12ч
хз - 16ч
хз как-то - 20ч
как-то сложно - 24ч
сложно - 30ч
очень сложно - 40ч

HOLY JS/... 2024 Autumn

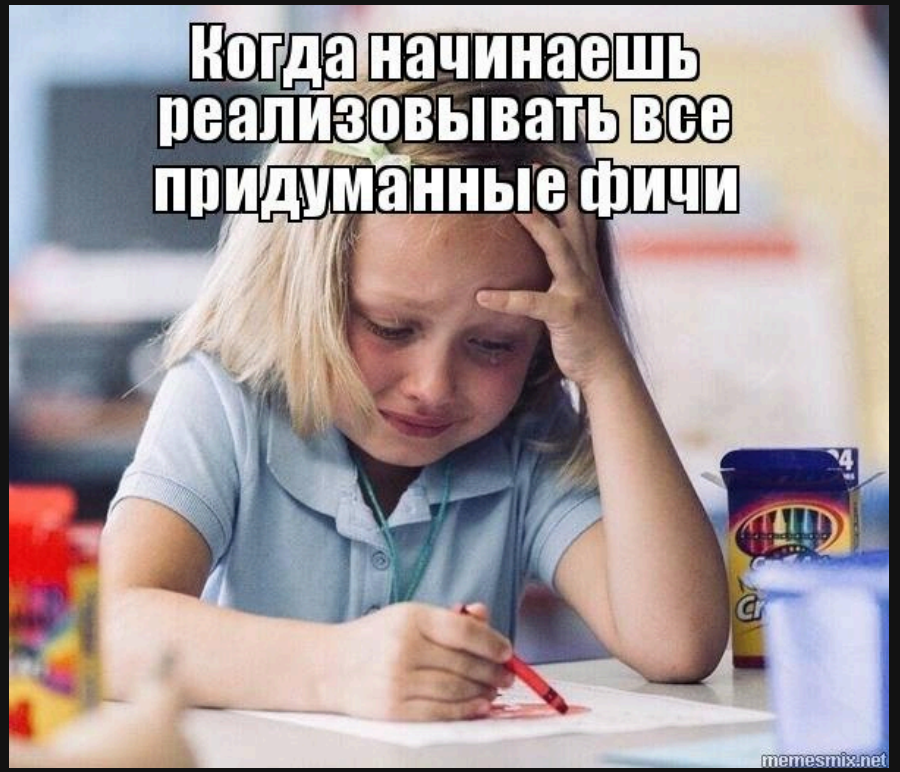
Сколько памяти ест ваша вкладка?

Антон Непша

Сбер

То есть...

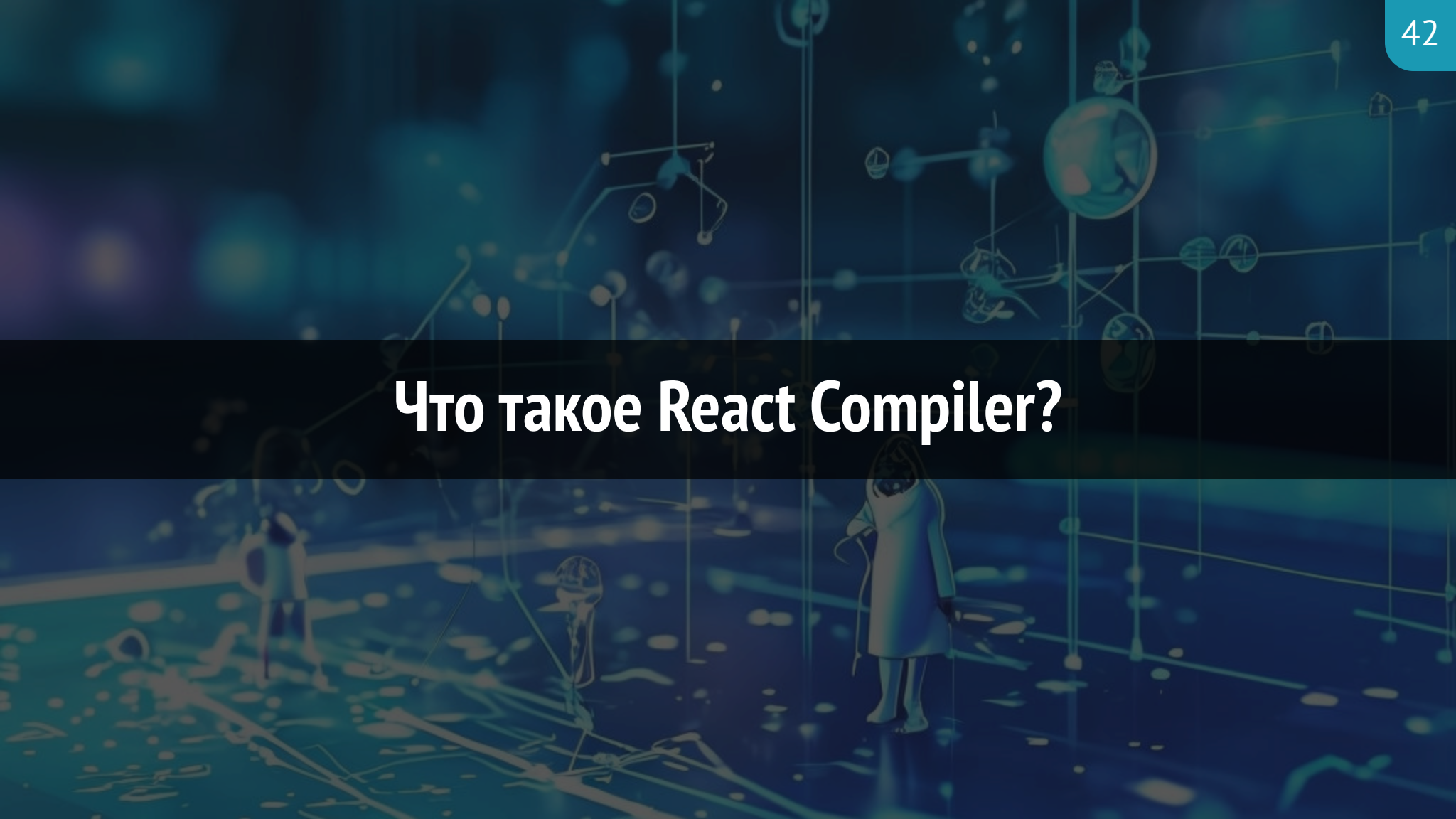
- State management
- Reactivity
- DOM updates
- Performance + **Memo**
- DX: Developer Experience



2022 React Forget

2022 React Forget → React Compiler 2024

2022 ~~React Forget~~ → React Compiler 2024



Что такое React Compiler?

React Compiler != React 19

React 19 — новая версия React
React Compiler — plugin

В каком состоянии сейчас React Compiler?

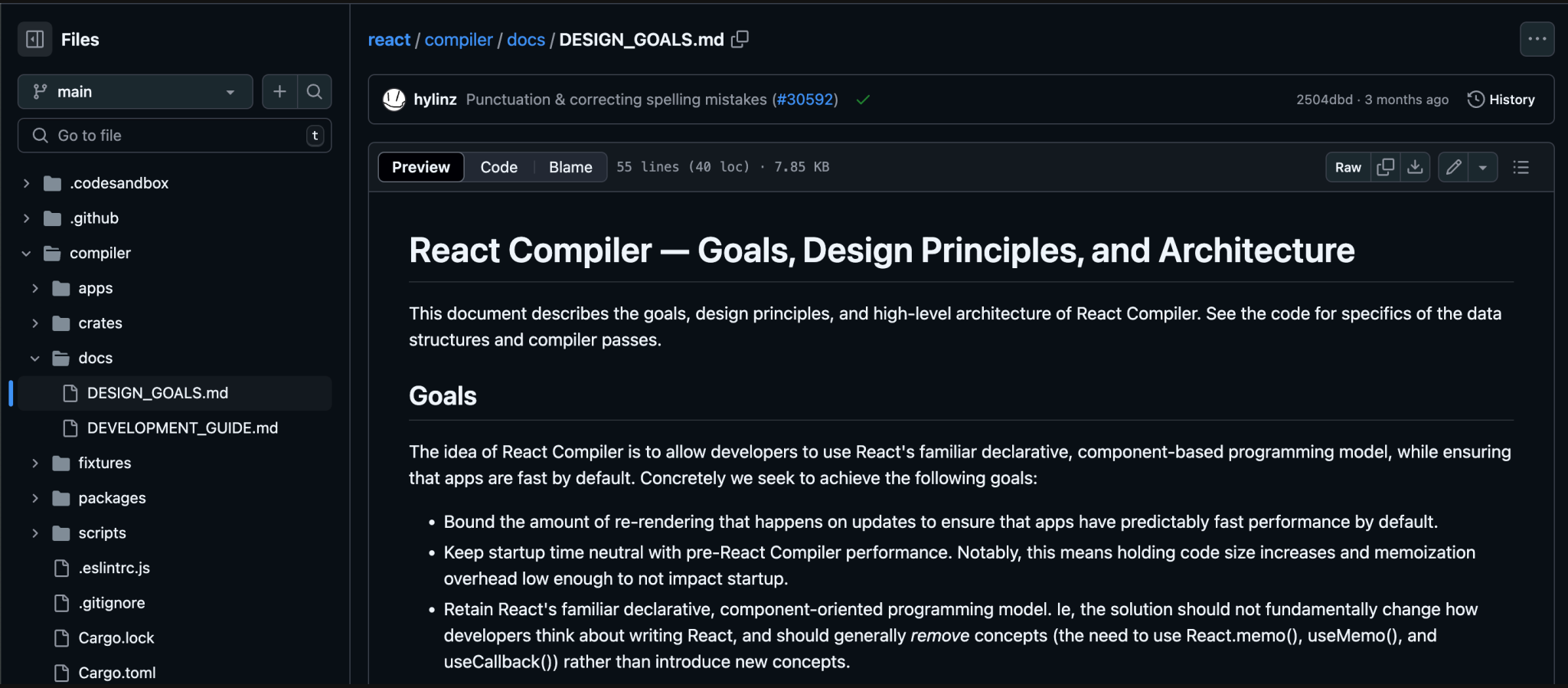
- Open-source Beta 21 октября 2024
- Поддержка React 19
- Добавлена поддержка React 17, 18

Ожидания RC от нас

- Валидный JavaScript
- Nullable/опциональные значения определены перед обращением

```
let company = null;
let person = {};
//... какие-то действия
console.log(person.name?.firstName + ' работает в ' + company);
```

- Код соответствует Rules of React


Files
main
Go to file
t
.codesandbox
.github
compiler
apps
crates
docs
DESIGN_GOALS.md
DEVELOPMENT_GUIDE.md
fixtures
packages
scripts
.eslintrc.js
.gitignore
Cargo.lock
Cargo.toml
react / compiler / docs / DESIGN_GOALS.md
hylinz Punctuation & correcting spelling mistakes (#30592)
2504dbd · 3 months ago
History
Preview
Code
Blame
55 lines (40 loc) · 7.85 KB
Raw
React Compiler — Goals, Design Principles, and Architecture
This document describes the goals, design principles, and high-level architecture of React Compiler. See the code for specifics of the data structures and compiler passes.
Goals
The idea of React Compiler is to allow developers to use React's familiar declarative, component-based programming model, while ensuring that apps are fast by default. Concretely we seek to achieve the following goals:
Bound the amount of re-rendering that happens on updates to ensure that apps have predictably fast performance by default.
Keep startup time neutral with pre-React Compiler performance. Notably, this means holding code size increases and memoization overhead low enough to not impact startup.
Retain React's familiar declarative, component-oriented programming model. Ie, the solution should not fundamentally change how developers think about writing React, and should generally remove concepts (the need to use React.memo(), useMemo(), and useCallback()) rather than introduce new concepts.

Что является его целями

- Ограничить ререндеры.
- Сдерживать увеличение бандлов и объем кэширования.
- Оставить привычную модель работы с кодом и скорее удалять лишние концепции, нежели добавлять.
- "Просто работать" на коде, соответствующему Rules of React.
- Поддерживать привычные инструменты и подходы к дебагу.
- Быть достаточно понятным и предсказуемым для разработчиков.
- Не требовать явных аннотаций или типизации кода для базовых оптимизаций.

Что НЕ является его целями?

- Идеальная оптимизация ререндеров.
- Поддерживать код, который не следует Rules of React.
- Поддерживать легаси фичи React.
- Поддерживать 100% возможностей JavaScript.

Что он делает?

Playground

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

- Внутренний кэш

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

- Внутренний кэш

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

- Внутренний кэш

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

- Внутренний кэш

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    return (
3      <div>Welcome to HolyJS!</div>
4    )
5  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = <div>Welcome to HolyJS!</div>;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  return t0;
}
```

- Внутренний кэш
- Мемоизация без useMemo

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const
- Вычисляемое преобразовывается

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const
- Вычисляемое преобразовывается

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const
- Вычисляемое преобразовывается
- ~~useMemo~~

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const
- Вычисляемое преобразовывается
- useMemo, но a * 2

React Compiler Playground

```
1  const GreetHolyJS = () => {
2    const useless = 'Hi, I am useless!';
3    const a = 1012;
4    const b = useMemo(() => (a * 2), [a]);
5    return (
6      <div>Welcome to HolyJS {b}!</div>
7    )
8  }
```

- JS

```
function anonymous_1() {
  const $ = _c(1);
  const a = 1012;
  let t0;
  t0 = 2024;
  let t1;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = <div>Welcome to HolyJS {2024}!</div>;
    $[0] = t1;
  } else {
    t1 = $[0];
  }
  return t1;
}
```

- Удаляется то, что не используется
- Статичное значение - const
- Вычисляемое преобразовывается
- useMemo, но a * 2
- Не создается лишних зависимостей

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

React Compiler Playground

```jsx
const GreetHolyJS = () => {
  const useless = 'Hi, I am useless!';
  const a = 1012;
  const b = () => (a * 2);

  useEffect(() => {
    console.log(b);
  }, [b]);

  return (
    <div>Welcome to HolyJS {b}!</div>
  )
}
```

- JS

```jsx
function anonymous_1() {
  const $ = _c(4);
  let t0;
  if ($[0] === Symbol.for("react.memo_cache_sentinel")) {
    t0 = () => 2024;
    $[0] = t0;
  } else {
    t0 = $[0];
  }
  const b = t0;
  let t1;
  let t2;
  if ($[1] === Symbol.for("react.memo_cache_sentinel")) {
    t1 = () => {
      console.log(b);
    };
    t2 = [b];
    $[1] = t1;
    $[2] = t2;
  } else {
    t1 = $[1];
    t2 = $[2];
  }
  useEffect(t1, t2);
  let t3;
  if ($[3] === Symbol.for("react.memo_cache_sentinel")) {
    t3 = <div>Welcome to HolyJS {b}!</div>;
    $[3] = t3;
  } else {
    t3 = $[3];
  }
  return t3;
}
```

~~memo~~
~~useMemo~~
~~useCallback~~

~~memo~~
~~useMemo~~
~~useCallback~~

useEffect ★

Rules of React:

- Сайд эффекты должны быть вне рендера

```
let greeting = 'Hello';

const Welcome = () => {
    greeting = greeting + ' User!';

    return (<div>{greeting}</div>)
}
```

React Compiler Playground

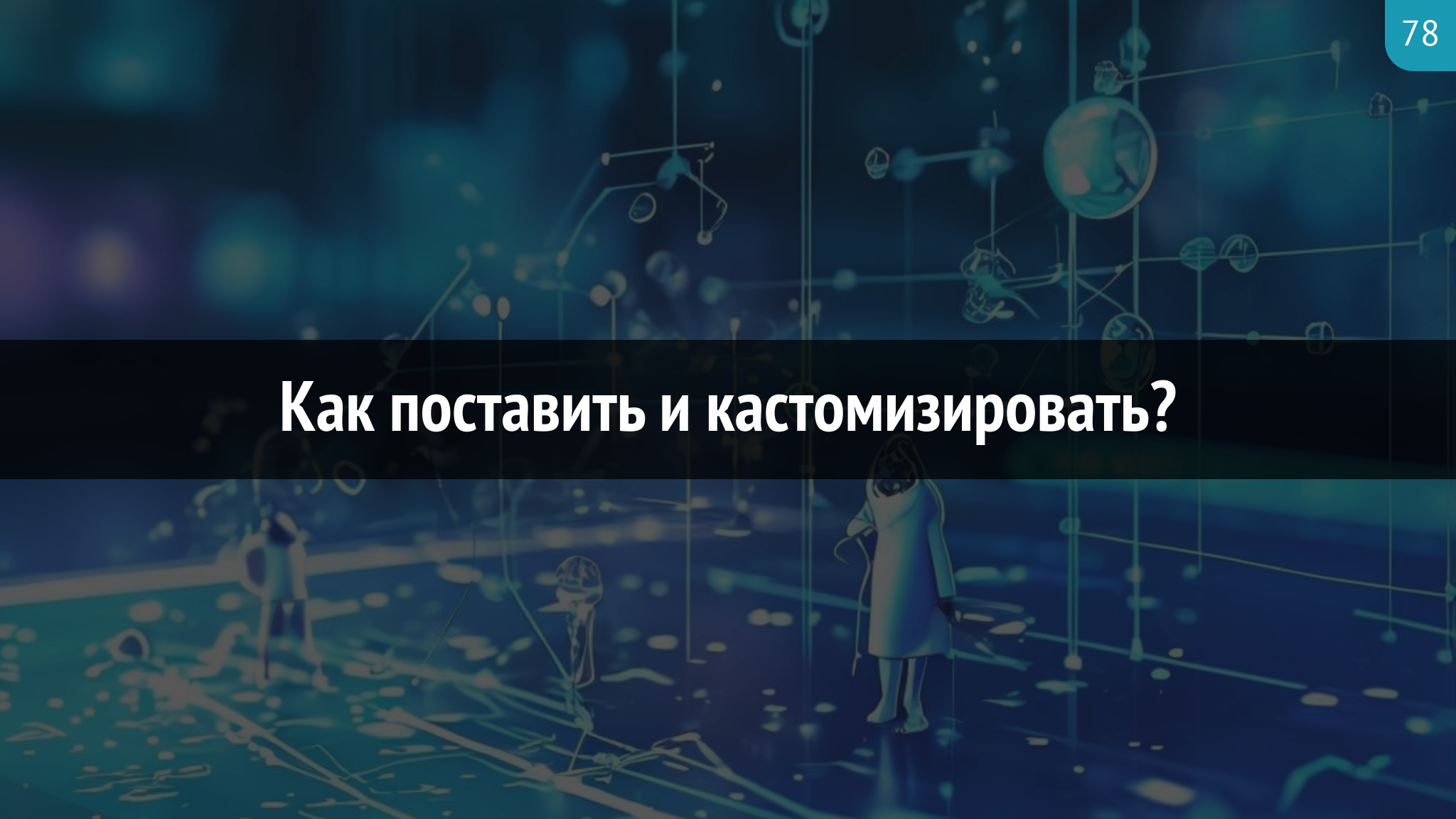
Reset Share

```
1  let greeting = 'Hello';
2
3  const Welcome = () => {
4      greeting = greeting + ' User!';
5
6      return (<div>{greeting}</div>)
7  }
```

EnvironmentConfig
HIR
PruneMaybeThrows
DropManualMemoization
InlineImmediatelyInvokedFunctionExp
MergeConsecutiveBlocks
SSA
EliminateRedundantPhi
ConstantPropagation
InferTypes
AnalyseFunctions

COMPILER ERRORS

```
InvalidReact: Unexpected reassignment of a variable which was defined outside
of the component. Components and hooks should be pure and side-effect free,
but variable reassignment is a form of side-effect. If this variable is used
in rendering, use useState instead.
(https://react.dev/reference/rules/components-and-hooks-must-be-pure#side-
effects-must-run-outside-of-render) (4:4)
```



Как поставить и кастомизировать?

Официально используется с...

- Babel
- Vite
- Webpack
- Next.js
- Remix
- Expo
- Metro (React Native)
- Rspack
- Rsbuild

Как применить?

- React 17+ и Node v16.6+
- react-compiler-healthcheck (поиск несовместимых библиотек)
- ESLint (проверка на Rules of React)
- В идеале проапгрейдиться до 19
- Установить React Compiler

Для версии 19:

```
npm install -D babel-plugin-react-compiler@beta
```

Для версии 17, 18:

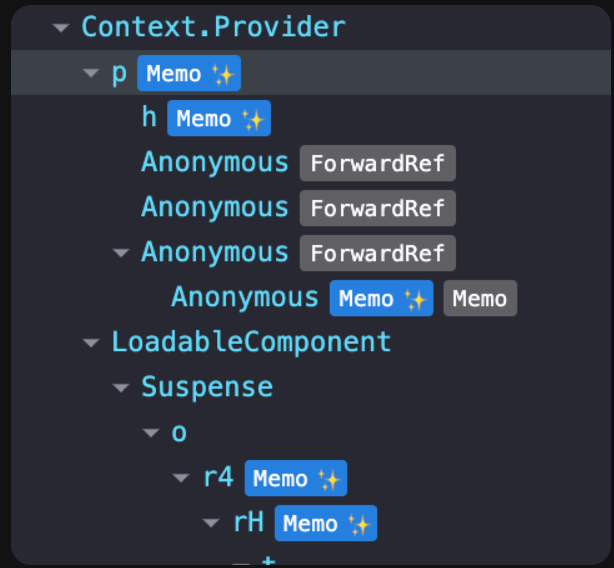
```
npm install -D react-compiler-runtime@beta
```

```
// babel.config.js
const ReactCompilerConfig = {
  target: '18' // '17' | '18' | '19'
};

module.exports = function () {
  return {
    plugins: [
      ['babel-plugin-react-compiler', ReactCompilerConfig],
    ],
  };
};
```

Как применить?

- Убедиться, что у вас React 17+ и Node v16.6+
- Поставить react-compiler-healthcheck (поиск несовместимых библиотек)
- Поставить ESLint (проверка на Rules of React)
- В идеале проапгрейдиться до 19
- Установить React Compiler
- Ставим React DevTools v5.0+
- Проверяем на наличие “Memo ✨”



Как применить?

- Убедиться, что у вас React 17+ и Node v16.6+
- Поставить react-compiler-healthcheck (поиск несовместимых библиотек)
- Поставить ESLint (проверка на Rules of React)
- В идеале проапгрейдиться до 19
- Установить React Compiler
- Ставим React DevTools v5.0+
- Проверяем на наличие “Memo ✨”
- С выключенным RC поискать лишние ререндеры
- Включить RC и проверить, починилось ли
- Если нет, а нужно, возможно вам стоит отрефакторить вручную
- Замеры делайте в production сборке

Как применить?

- Убедиться, что у вас React 17+ и Node v16.6+
- Поставить react-compiler-healthcheck (поиск несовместимых библиотек)
- Поставить ESLint (проверка на Rules of React)
- В идеале проапгрейдиться до 19
- Установить React Compiler
- Ставим React DevTools v5.0+
- Проверяем на наличие “Memo ✨”
- С выключенным RC поискать лишние ререндеры
- Включить RC и проверить, починилось ли
- Если нет, а нужно, возможно вам стоит отрефакторить вручную
- Замеры делайте в production сборке

Не хочу компилировать все?

Не хочу компилировать все?

- Ошибки ESLint
- Для дебага

Не хочу компилировать все?

- Ошибки ESLint
- Для дебага

```
function SuspiciousComponent() {
  "use no memo"; // opts out this component from being compiled by React Compiler
  // ...
}
```

Что за конфиг?

```
// babel.config.js
const ReactCompilerConfig = {
  target: '18' // '17' | '18' | '19'
};

module.exports = function () {
  return {
    plugins: [
      ['babel-plugin-react-compiler', ReactCompilerConfig],
    ],
  };
};
```

Нет доков, но есть github!

react / compiler / packages / babel-plugin-react-compiler / src / Entrypoint / **Options.ts**

Code | Blame | 271 lines (248 loc) · 8.46 KB

```
199  export const defaultOptions: PluginOptions = {
200    compilationMode: 'infer',
201    panicThreshold: 'none',
202    environment: parseEnvironmentConfig({}).unwrap(),
203    logger: null,
204    gating: null,
205    noEmit: false,
206    eslintSuppressionRules: null,
207    flowSuppressions: true,
208    ignoreUseNoForget: false,
209    sources: filename => {
210      return filename.indexOf('node_modules') === -1;
211    },
212    enableReanimatedCheck: true,
213    target: '19',
214  } as const;
215
```

Sources = компилируемые файлы!

react / compiler / packages / babel-plugin-react-compiler / src / Entrypoint / **Options.ts**

Code | Blame | 271 lines (248 loc) · 8.46 KB

```
199  export const defaultOptions: PluginOptions = {
200    compilationMode: 'infer',
201    panicThreshold: 'none',
202    environment: parseEnvironmentConfig({}).unwrap(),
203    logger: null,
204    gating: null,
205    noEmit: false,
206    eslintSuppressionRules: null,
207    flowSuppressions: true,
208    ignoreUseNoForget: false,
209    sources: filename => {
210      return filename.indexOf('node_modules') === -1;
211    },
212    enableReanimatedCheck: true,
213    target: '19',
214  } as const;
215
```

Environment = ???

react / compiler / packages / babel-plugin-react-compiler / src / Entrypoint / **Options.ts**

Code | Blame | 271 lines (248 loc) · 8.46 KB

```
199  export const defaultOptions: PluginOptions = {
200    compilationMode: 'infer',
201    panicThreshold: 'none',
202    environment: parseEnvironmentConfig({}).unwrap(),
203    logger: null,
204    gating: null,
205    noEmit: false,
206    eslintSuppressionRules: null,
207    flowSuppressions: true,
208    ignoreUseNoForget: false,
209    sources: filename => {
210      return filename.indexOf('node_modules') === -1;
211    },
212    enableReanimatedCheck: true,
213    target: '19',
214  } as const;
215
```

Нет доков, но есть github!

```
149  const EnvironmentConfigSchema = z.object({
150    customHooks: z.map(z.string(), HookSchema).default(new Map()),
151
152    /**
153     * A function that, given the name of a module, can optionally return a description
154     * of that module's type signature.
155     */
156    moduleTypeProvider: z.nullable(z.function().args(z.string())).default(null),
157
158    /**
159     * A list of functions which the application compiles as macros, where
160     * the compiler must ensure they are not compiled to rename the macro or separate the
161     * "function" from its argument.
162     *
163     * For example, Meta has some APIs such as `featureflag("name-of-feature-flag")` which
164     * are rewritten by a plugin. Assigning `featureflag` to a temporary would break the
165     * plugin since it looks specifically for the name of the function being invoked, not
166     * following aliases.
167     */
168    customMacros: z.nullable(z.array(MacroSchema)).default(null),
169
170    /**
171     * Enable a check that resets the memoization cache when the source code of the file changes.
172     * This is intended to support hot module reloading (HMR), where the same runtime component
173     * instance will be reused across different versions of the component source.
174     */
175    enableResetCacheOnSourceFileChanges: z.boolean().default(false),
176
177    /**
178     * Enable using information from existing useMemo/useCallback to understand when a value is done
179     * being mutated. With this mode enabled, Forget will still discard the actual useMemo/useCallback
180     * calls and may memoize slightly differently. However, it will assume that the values produced
181     * are not subsequently modified, guaranteeing that the value will be memoized.
```

- customHooks
- moduleTypeProvider
- customMacros
- enableResetCacheOnSourceFileChanges
- enablePreserveExistingMemoizationGuarantees
- validatePreserveExistingMemoizationGuarantees
- enablePreserveExistingManualUseMemo
- enableForest
- enableUseTypeAnnotations
- inlineJsxTransform
- validateHooksUsage
- validateRefAccessDuringRender
- validateNoSetStateInRender
- validateNoSetStateInPassiveEffects
- validateNoJSXInTryStatements
- validateMemoizedEffectDependencies
- validateNoCapitalizedCalls
- validateBlocklistedImports
- enableAssumeHooksFollowRulesOfReact
- enableTransitivelyFreezeFunctionExpressions
- enableEmitFreeze
- enableEmitHookGuards
- enableInstructionReordering
- enableFunctionOutlining
- enableJsxOutlining
- enableEmitInstrumentForget
- assertValidMutableRanges
- enableChangeVariableCodegen
- enableMemoizationComments
- throwUnknownException__testonly
- enableTreatFunctionDepsAsConditional
- disableMemoizationForDebugging
- enableChangeDetectionForDebugging
- enableCustomTypeDefinitionForReanimated
- hookPattern
- enableTreatRefLikeIdentifiersAsRefs
- lowerContextAccess

enablePreserveExistingMemoizationGuarantees

enablePreserveExistingMemoizationGuarantees

false — default

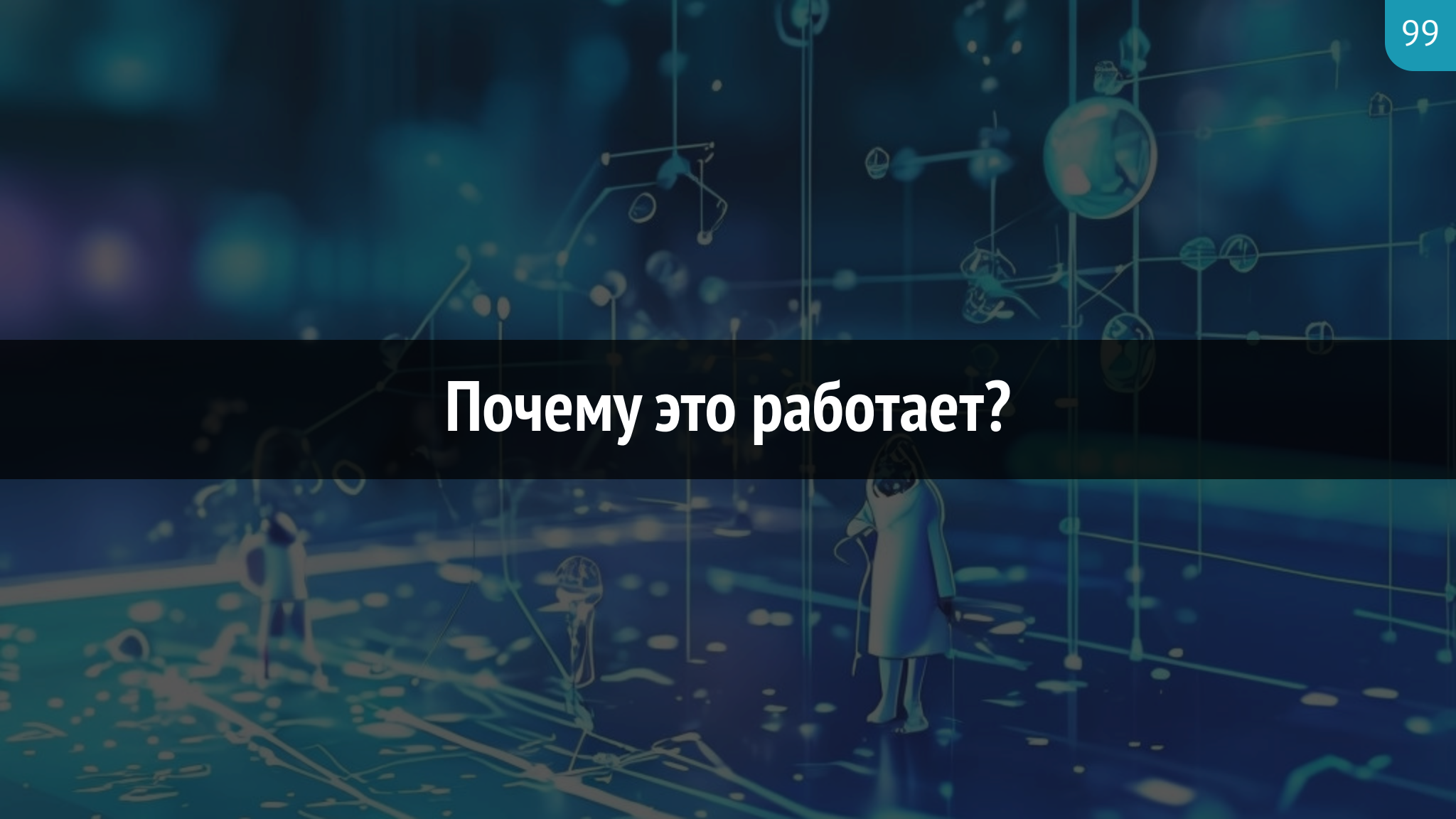
true — RC сохраняет значения из существующих useMemo/useCallback для бэкапа, но удаляет сами вызовы

enablePreserveExistingMemoizationGuarantees

false — default

true — RC сохраняет значения из существующих useMemo/useCallback для бэкапа, но удаляет сами вызовы

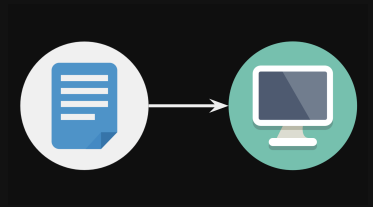
Включать/выключать для дебага



Почему это работает?

Что такое компилятор?

- Анализирует код
- Переводит код в более низкоуровневый



  - Java, C, C++, C# ...
- Способен оптимизировать и находить ошибки

COMPILER EXPLORER

Add... More Templates Share Policies 101

Go source #1

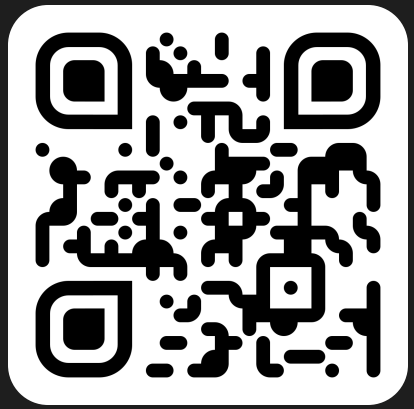
A Save/Load + Add new... Vim

```
1  // Type your code here, or load an example.
2  // Your function name should start with a capital lett
3  package main
4
5  func Square(x int) int {
6      return x * x
7  }
8
9  func main() {}
10
```

Go

Select a language...

- Android Kotlin
- Assembly
- C
- C#
- C++
- C++ (Circle)
- C++ for OpenCL



ARM gc 1.14 (Editor #1)

ARM gc 1.14 Compiler options...

A Output... Filter... Libraries Overrides + Add new... Add tool...

```
1        TEXT    "".Square(SB), LEAF|NOFRAME|ABIInternal, $-4-8
2        PCDATA  $0, $-2
3        PCDATA  $1, $-2
4        FUNCDATA        $0, gclocals·33cdeccccebe80329f1fdbee7f5874
5        FUNCDATA        $1, gclocals·33cdeccccebe80329f1fdbee7f5874
6        FUNCDATA        $2, gclocals·33cdeccccebe80329f1fdbee7f5874
7        PCDATA  $0, $0
8        PCDATA  $1, $0
9        MOVW    "".x(FP), R0
10       MUL     R0, R0, R0
11       MOVW    R0, "".~r1+4(FP)
12       JMP     (R14)
13       TEXT    "".main(SB), LEAF|NOFRAME|ABIInternal, $-4-0
14       PCDATA  $0, $-2
15       PCDATA  $1, $-2
16       FUNCDATA        $0, gclocals·33cdeccccebe80329f1fdbee7f5874
17       FUNCDATA        $1, gclocals·33cdeccccebe80329f1fdbee7f5874
18       FUNCDATA        $2, gclocals·33cdeccccebe80329f1fdbee7f5874
19       PCDATA  $0, $-1
20       PCDATA  $1, $-1
21       JMP     (R14)
```

Output (0/0) ARM gc 1.14 - cached (1303731B) ~4 lines filtered


BABEL
Babel is a JavaScript compiler.
Use next generation JavaScript, today.

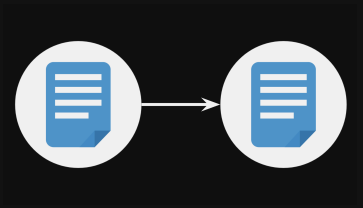
Транспилятор применяется при миграции с одной версии языка на другую, причем в обе стороны: с более старой на новую и наоборот.

> *Например, Babel —JavaScript транспилятор, позволяющий использовать новейшие функции языка на базе старых версий браузеров.*

Транспилятор поможет при переводе с одного языка программирования на другой.

Что такое транспилятор?

- Подвид / этап компиляции
- Высокоуровневый язык → высокоуровневый язык



- Кроссплатформенность

Составляющие RC

- Babel plugin — трансформирует код

```
babel-plugin-react-compiler@beta
```

- ESLint plugin — отлавливает нарушения Rules of React

```
eslint-plugin-react-compiler@beta
```

- Compiler core — основная логика анализа и оптимизации

Составляющие RC

- Babel plugin — трансформирует код

```
babel-plugin-react-compiler@beta
```

- ESLint plugin — отлавливает нарушения Rules of React

```
eslint-plugin-react-compiler@beta
```

- Compiler core — основная логика анализа и оптимизации

Open-source

Составляющие RC

- Babel plugin — трансформирует код

```
babel-plugin-react-compiler@beta
```

- ESLint plugin — отлавливает нарушения Rules of React

```
eslint-plugin-react-compiler@beta
```

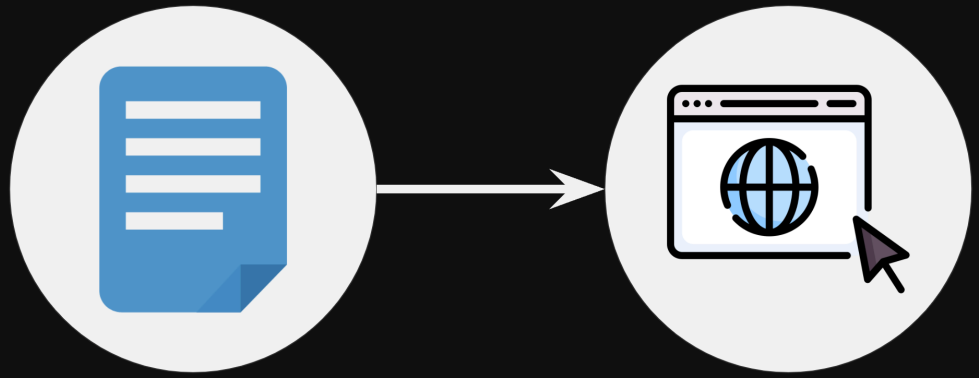
- Compiler core — основная логика анализа и оптимизации

Open-source

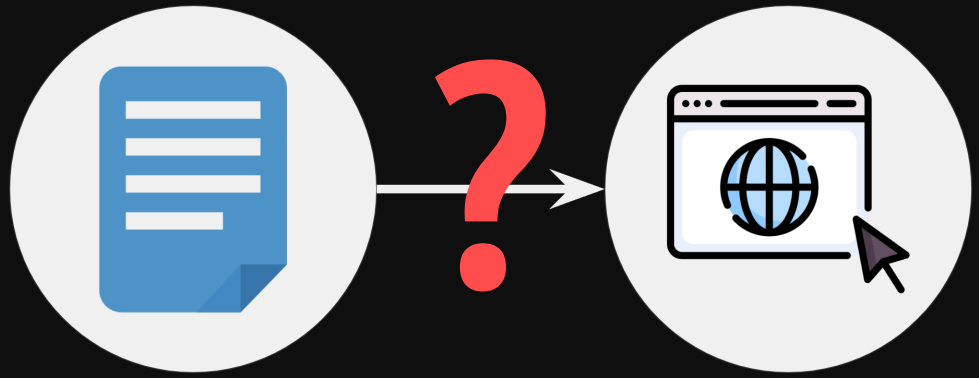
NOT open-source

RC & сборка

"Код – это просто текст, как это работает?"



"Код – это просто текст, как это работает?"



Одна из схем:

Человеко-код → Babel → Сборщик → Браузер

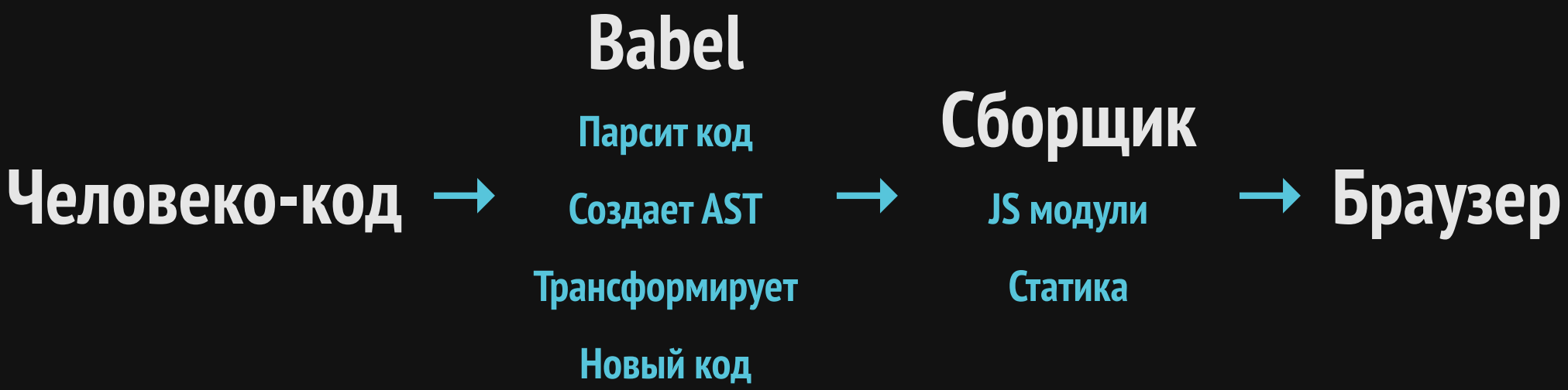
Человеко-код → Babel → Сборщик → Браузер

Парсит код

Создает AST

Трансформирует

Новый код


Человеко-код
Babel
Парсит код
Создает AST
Трансформирует
Новый код
Сборщик
JS модули
Статика
Браузер

Человеко-код → **Babel** → **Сборщик** → Браузер

Babel

Парсит код

Создает AST

+ React Compiler

(React 19)

Сборщик

JS модули

Статика

Трансформирует

На что влияет?

Человеко-код → **Babel** → **Сборщик** → **Браузер**

Babel

Парсит код

Создает AST

+ React Compiler

(React 19)

Сборщик

JS модули

Статика

Браузер

State

Реактивность

DOM

Трансформирует

React Compiler → State
Реактивность
DOM

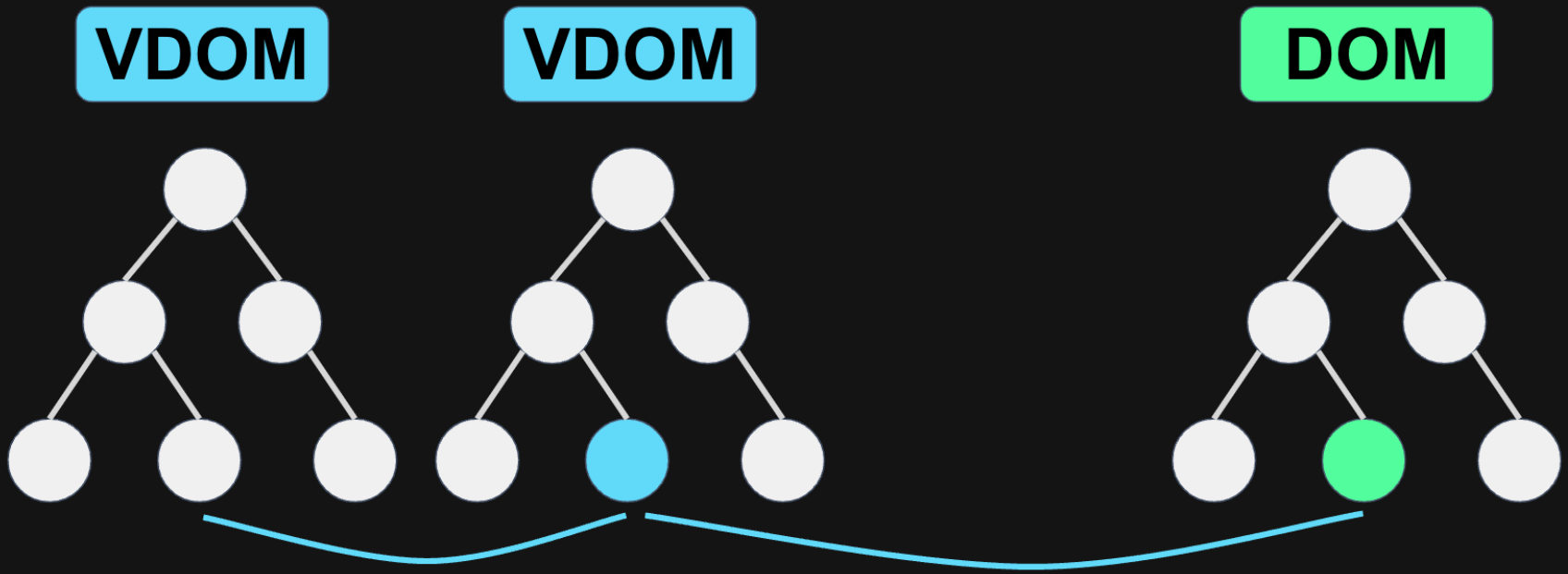
React Compiler → State / Реактивность / DOM → Performance

А что вообще во frontend-е?

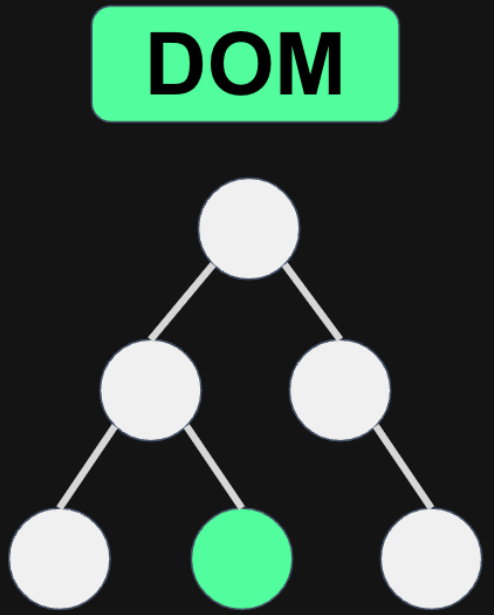
Доклад: Фантастические фреймворки и как от них не отставать

Работа с DOM

Работа с DOM



Работа с DOM




MAGIC

Компиляторы

Реактивность

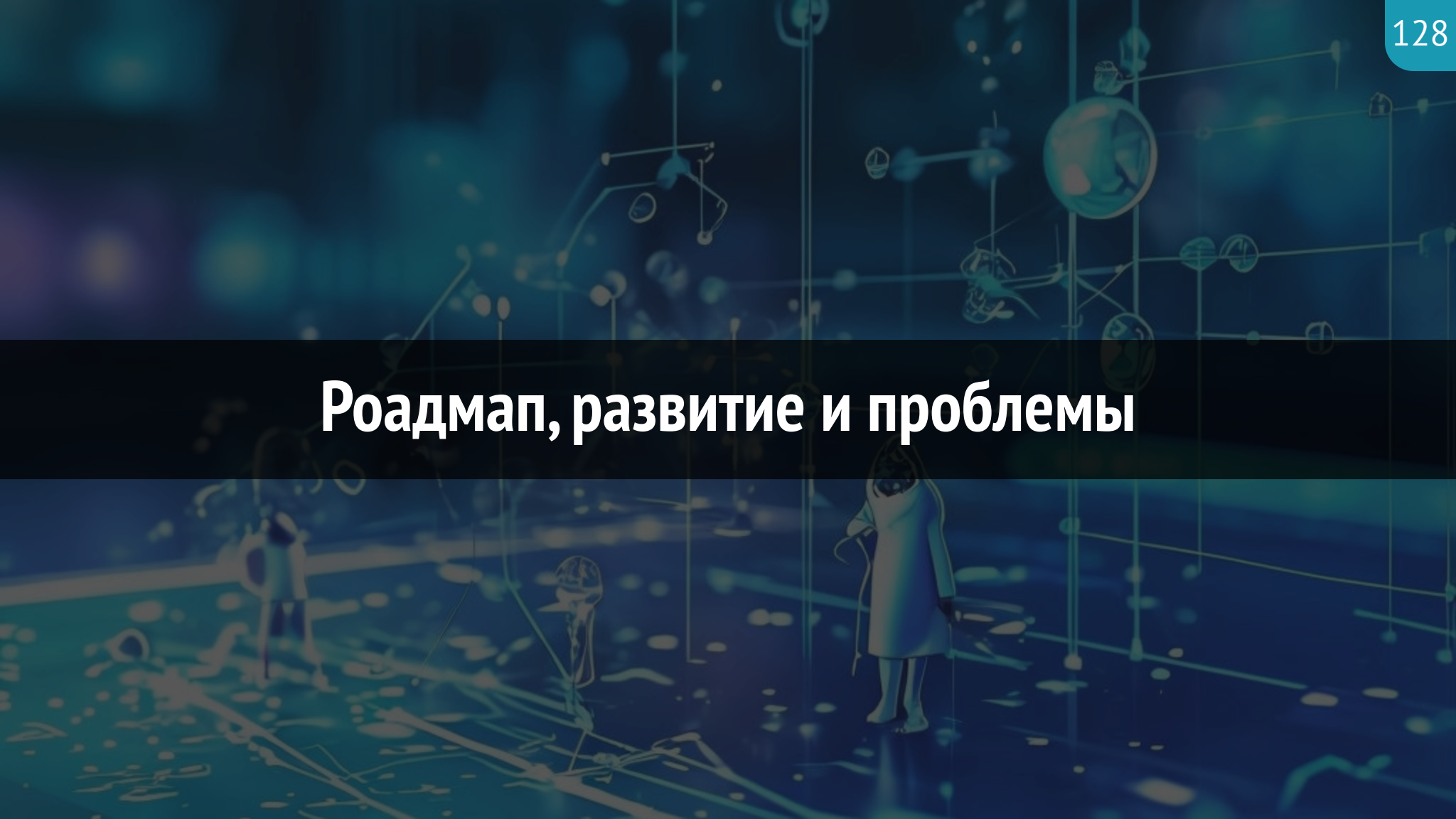
- Observable 
- Signal    

- Proxy  

Реактивность

- Observable 
- Signal    

- Proxy  



Роадмап, развитие и проблемы

Роадмап

- ✅ Experimental: Released at React Conf 2024.
- ✅ Public Beta: 21 октября 2024, собирается фидбек.
- 🚧 Release Candidate: версия, на которой должны хорошо работать большая часть приложений, следующие правилам.
- 🚧 Stable release: стабильный релиз еще после сбора фидбека.

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

```
function useVideos() {
  const data = useFragment(graphql`...`);
  const videos = [];

  if (data.kind === 'video') {
    videos.push(...data.values);
  }

  return videos;
}
```

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

```
function useVideos() {
  const data = useFragment(graphql`...`);

  return useMemo(() => {
    const videos = [];

    if (data.kind === 'video') {
      videos.push(...data.values);
    }

    return videos;
  }, [data]);
}
```

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

```
function useVideos() {
  const data = useFragment(graphql`...`);

  return useMemo(() => {
    const videos = [];

    if (data.kind === 'video') {
      videos.push(...data.values);
    }

    return videos;
  }, [data.kind, data.value]);
}
```

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

```
const edges = data.edges ?? [];
```

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

RC сам рефакторит за вас функции

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

RC сам трансформирует JSX

вместо Babel

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

Команда RC просит авторов библиотек

компилировать код

и потом публиковать в npm

Над чем ведется работа?

- Гранулярная мемоизация
- Type inference (автоназначение типов)
- Улучшение ESLint
- Function outlining
- JSX Elements inlining
- Поддержка библиотек
- Community + Working Group

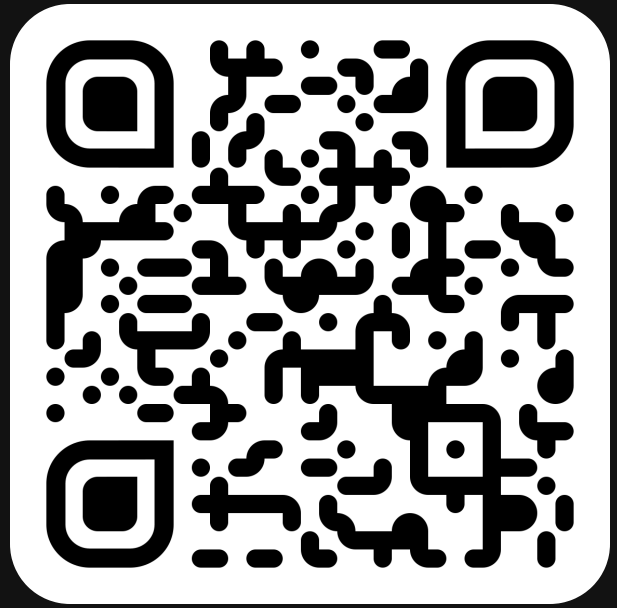
Собираются issues.

Ведутся дискуссии...

Какие проблемы?

- Proxy (MobX)
- Не оптимизирует работу некоторых библиотек
- Много issues про ref
- Еще не все кейсы оптимизируются
- ...

Nadia Makarevich: Developer Way



Выводы

Develop: совсем отказываемся от memo?

- Отказываемся там, где оптимизаций RC достаточно.
- Используем, если RC не справляется.

Develop: где может не справляться?

- Код, написанный не по Rules of React
- Плохая композиция
- Внешние библиотеки
- Proxy → MobX
- Signal?
- ...

Develop: что нужно знать, чтобы было хорошо?

- Memoization
- Composition
- Rules of React
- React Compiler
  - Применять итеративно
  - Тестировать

Research: новинка ли это?

- Концепция не новая...
- Но для React – да
- Пора забыть про React Forget

Research: какие возможности открываются?

- Не просто автомемоизация, но и рефакторинг.
- Применение сторонних инструментов оптимизации при сборке.
- Возможное изменение работы с DOM и реактивностью.
- "Когда откажутся от useState?"
- ...

Уже можно в Prod?

- У создателей уже в проде, но...
- Зависит от качества вашего кода.
- Нормально подождать стабильного релиза.
- Используя сейчас – помогаете сообществу.

@it_wildlife