React

SHOW TIME

我是谁？

二哲

大学创业三两年「 Vue 」

铃盛「 React 」

今日主题

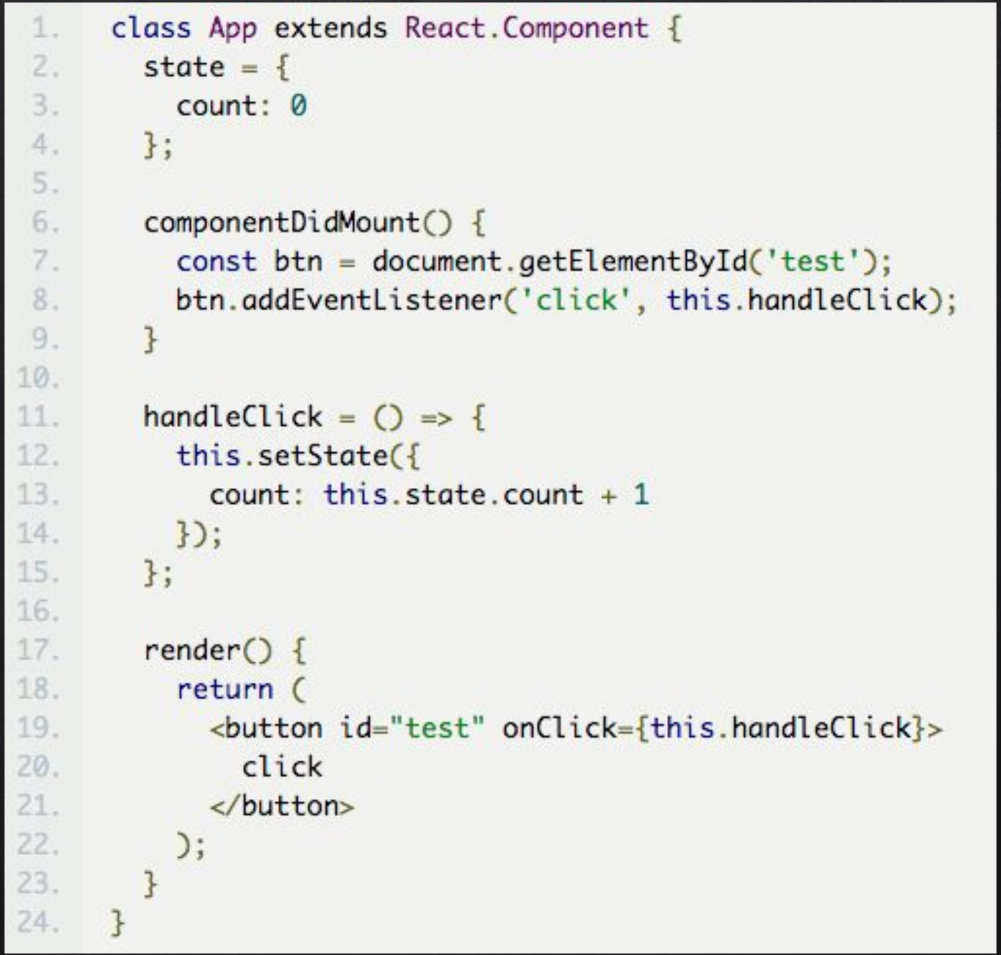
- ✘ 1. React bad parts
- ✘ 2. React hook
- ✘ 3. React state management
- ✘ 4. Q&A

“

🔨 React bad parts

1.

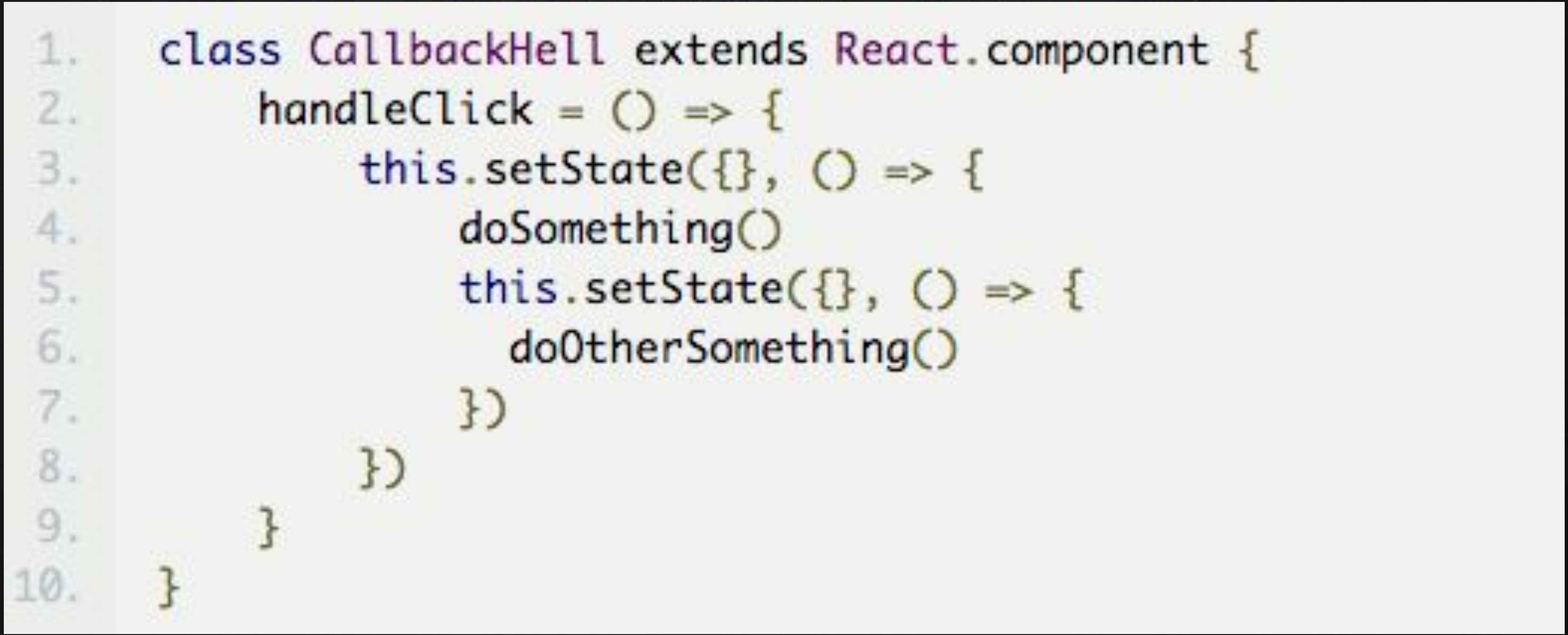
setState 同步？异步？



```
class App extends React.Component {
  state = {
    count: 0
  };

  componentDidMount() {
    const btn = document.getElementById('test');
    btn.addEventListener('click', this.handleClick);
  }

  handleClick = () => {
    this.setState({
      count: this.state.count + 1
    });
  };

  render() {
    return (
      <button id="test" onClick={this.handleClick}>
        click
      </button>
    );
  }
}
```

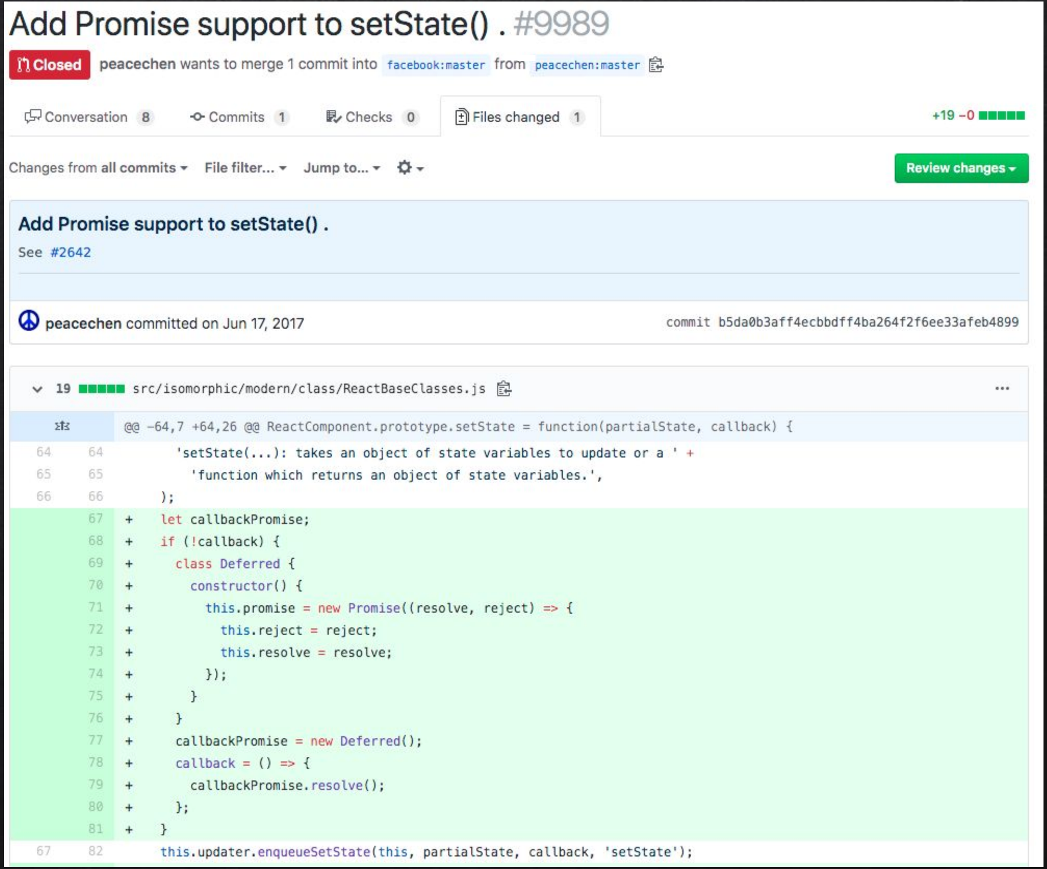
2.

setState callback hell



```
class CallbackHell extends React.component {
    handleClick = () => {
        this.setState({}, () => {
            doSomething()
            this.setState({}, () => {
              doOtherSomething()
            })
        })
    }
}
```

Add Promise support to setState()



Add Promise support to setState() . #9989

Closed peacechen wants to merge 1 commit into facebook:master from peacechen:master

Conversation 8 Commits 1 Checks 0 Files changed 1 +19 −0

Changes from all commits File filter... Jump to... Review changes

Add Promise support to setState() .

See #2642

peacechen committed on Jun 17, 2017 commit b5da0b3aff4ecbbdff4ba264f2f6ee33afeb4899

19 src/isomorphic/modern/class/ReactBaseClasses.js

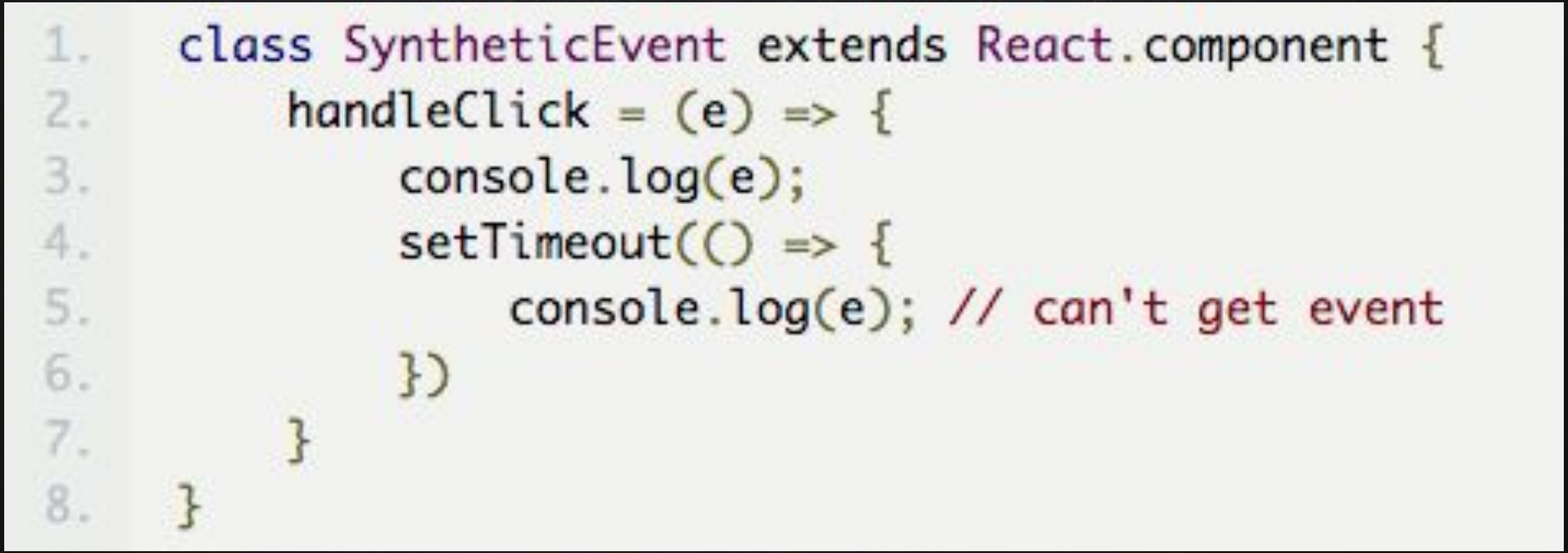
```
@@ -64,7 +64,26 @@ ReactComponent.prototype.setState = function(partialState, callback) {
       'setState(...): takes an object of state variables to update or a ' +
         'function which returns an object of state variables.',
     );
+    let callbackPromise;
+    if (!callback) {
+      class Deferred {
+        constructor() {
+          this.promise = new Promise((resolve, reject) => {
+            this.reject = reject;
+            this.resolve = resolve;
+          });
+        }
+      }
+      callbackPromise = new Deferred();
+      callback = () => {
+        callbackPromise.resolve();
+      };
+    }
     this.updater.enqueueSetState(this, partialState, callback, 'setState');
```

https://github.com/facebook/react/issues/2642

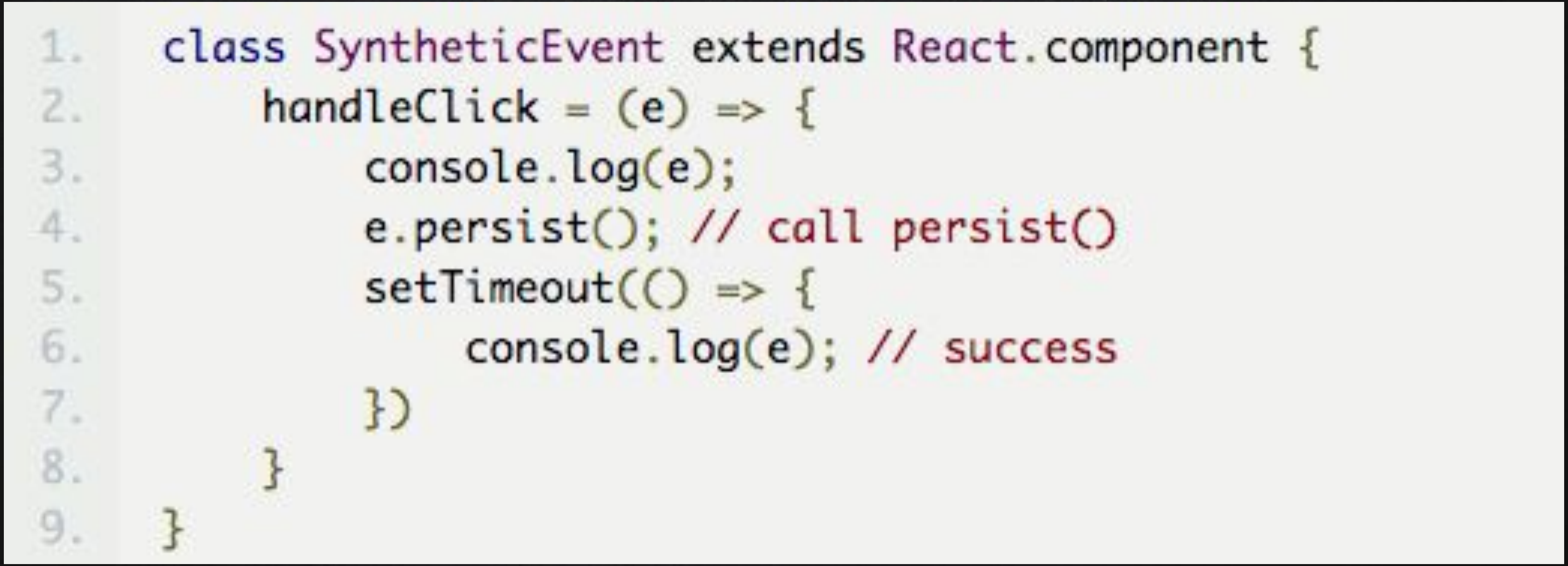
https://github.com/facebook/react/pull/9989#issuecomment-309141521

3.

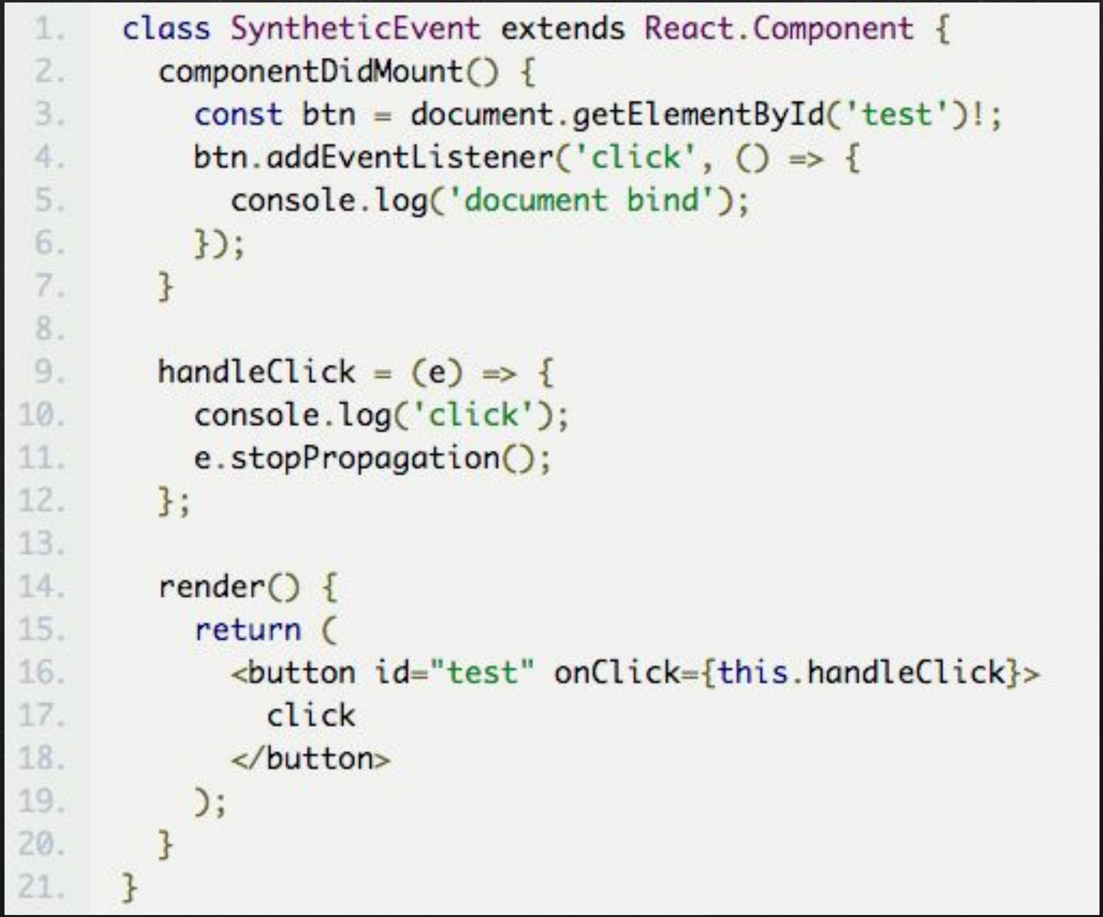
合成事件



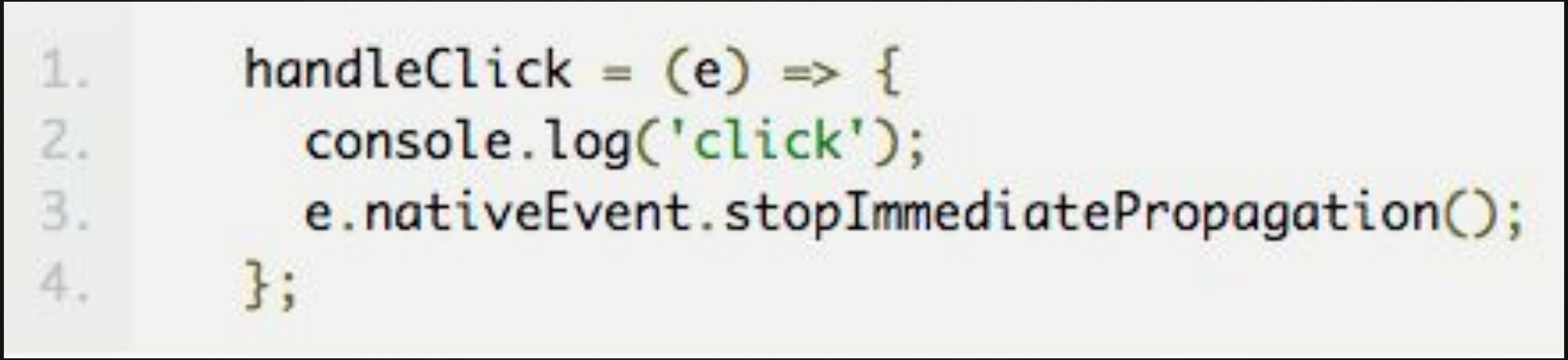
```
class SyntheticEvent extends React.component {
    handleClick = (e) => {
        console.log(e);
        setTimeout(() => {
            console.log(e); // can't get event
        })
    }
}
```



```
class SyntheticEvent extends React.component {
    handleClick = (e) => {
        console.log(e);
        e.persist(); // call persist()
        setTimeout(() => {
            console.log(e); // success
        })
    }
}
```



```
class SyntheticEvent extends React.Component {
  componentDidMount() {
    const btn = document.getElementById('test')!;
    btn.addEventListener('click', () => {
      console.log('document bind');
    });
  }

  handleClick = (e) => {
    console.log('click');
    e.stopPropagation();
  };

  render() {
    return (
      <button id="test" onClick={this.handleClick}>
        click
      </button>
    );
  }
}
```

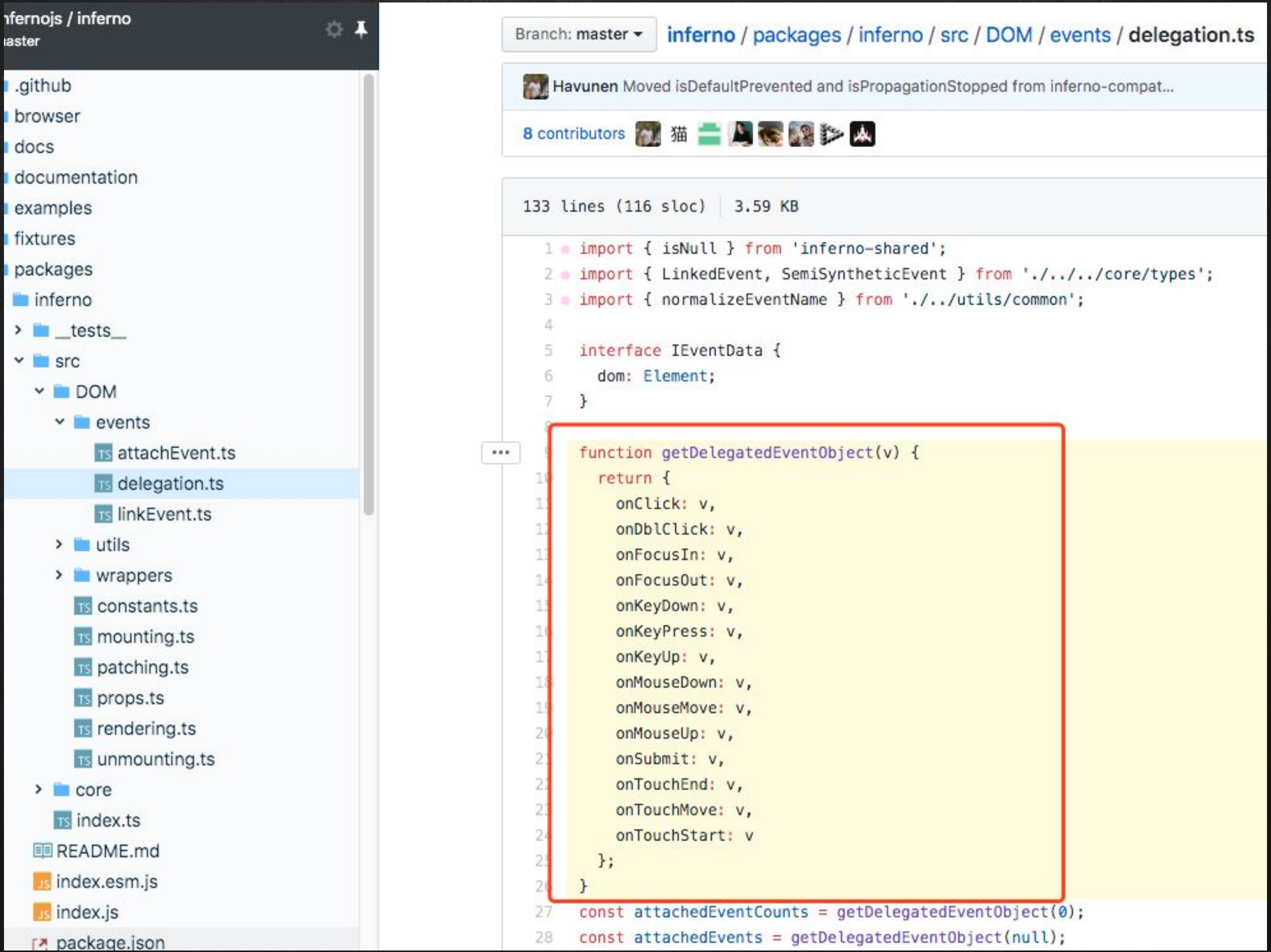


```
handleClick = (e) => {
  console.log('click');
  e.nativeEvent.stopImmediatePropagation();
};
```

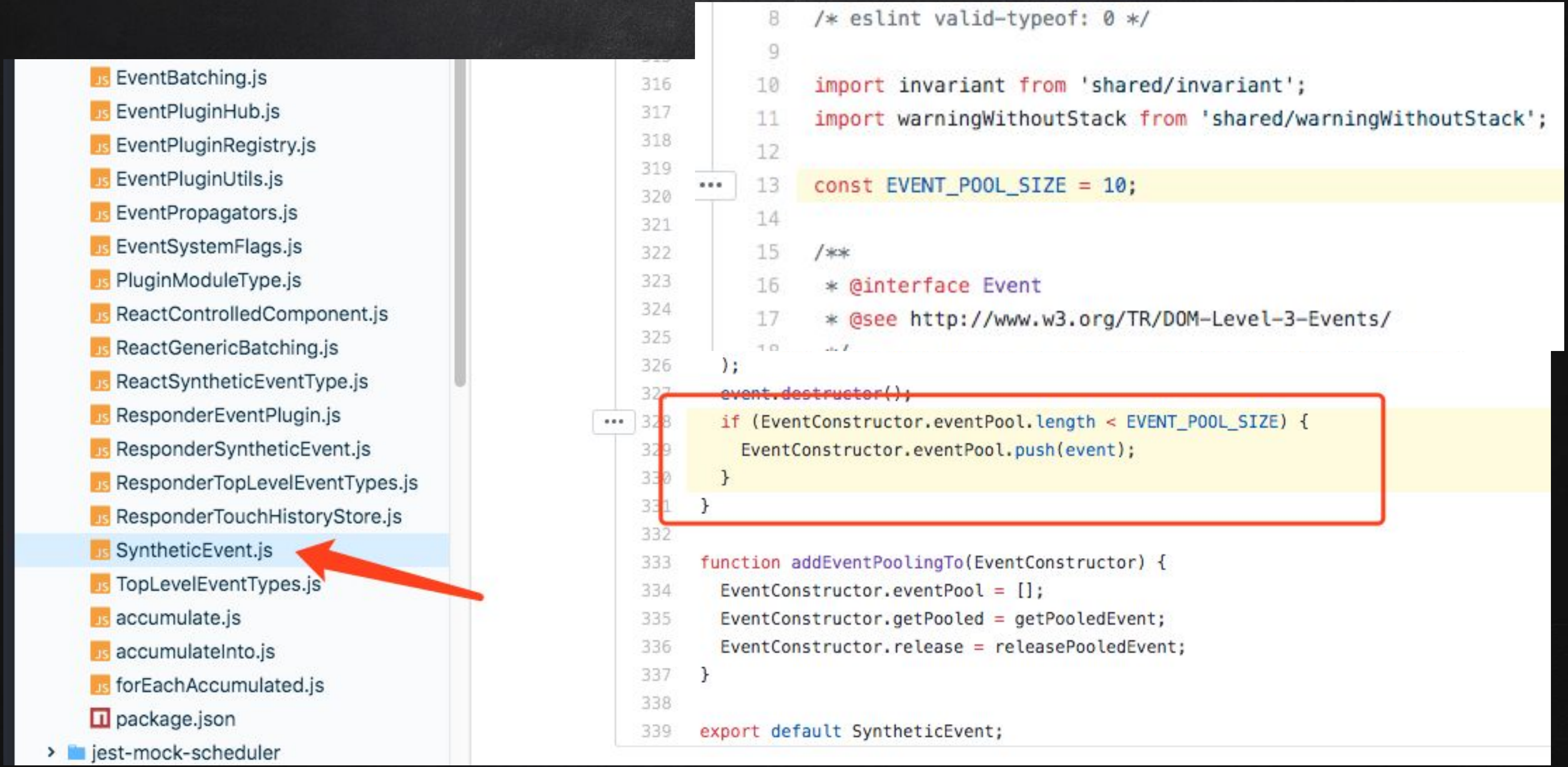
为什么要有合成事件？

目前的合成事件性能真的好吗？

有多少个合成事件？



```
import { isNull } from 'inferno-shared';
import { LinkedEvent, SemiSyntheticEvent } from './../../core/types';
import { normalizeEventName } from './../utils/common';

interface IEventData {
  dom: Element;
}

function getDelegatedEventObject(v) {
  return {
    onClick: v,
    onDblClick: v,
    onFocusIn: v,
    onFocusOut: v,
    onKeyDown: v,
    onKeyPress: v,
    onKeyUp: v,
    onMouseDown: v,
    onMouseMove: v,
    onMouseUp: v,
    onSubmit: v,
    onTouchEnd: v,
    onTouchMove: v,
    onTouchStart: v
  };
}
const attachedEventCounts = getDelegatedEventObject(0);
const attachedEvents = getDelegatedEventObject(null);
```

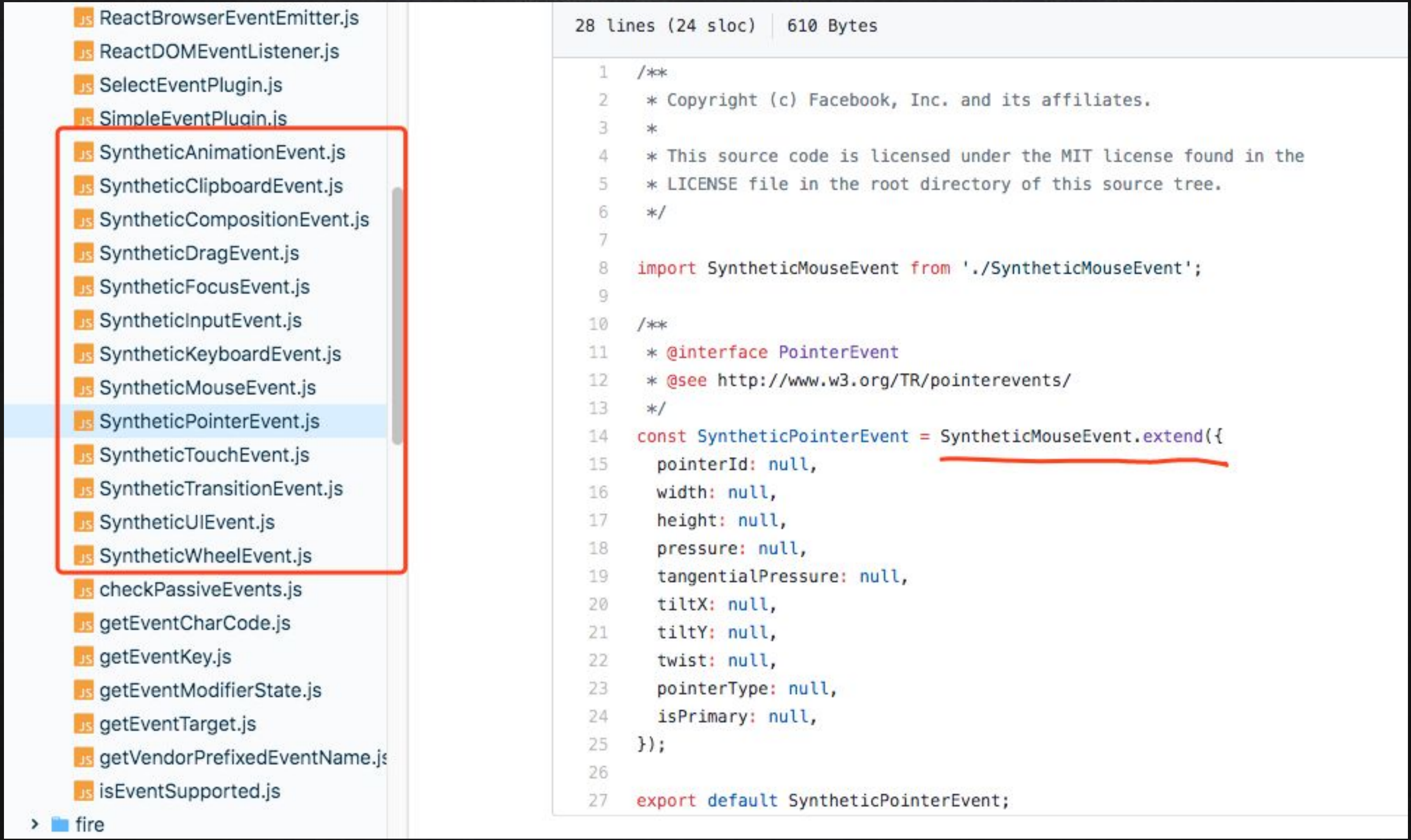


- EventBatching.js
- EventPluginHub.js
- EventPluginRegistry.js
- EventPluginUtils.js
- EventPropagators.js
- EventSystemFlags.js
- PluginModuleType.js
- ReactControlledComponent.js
- ReactGenericBatching.js
- ReactSyntheticEventType.js
- ResponderEventPlugin.js
- ResponderSyntheticEvent.js
- ResponderTopLevelEventTypes.js
- ResponderTouchHistoryStore.js
- SyntheticEvent.js
- TopLevelEventTypes.js
- accumulate.js
- accumulateInto.js
- forEachAccumulated.js
- package.json
- jest-mock-scheduler

```
/* eslint valid-typeof: 0 */

import invariant from 'shared/invariant';
import warningWithoutStack from 'shared/warningWithoutStack';

const EVENT_POOL_SIZE = 10;

/**
 * @interface Event
 * @see http://www.w3.org/TR/DOM-Level-3-Events/
```

```
  );
  event.destructor();
  if (EventConstructor.eventPool.length < EVENT_POOL_SIZE) {
    EventConstructor.eventPool.push(event);
  }
}

function addEventPoolingTo(EventConstructor) {
  EventConstructor.eventPool = [];
  EventConstructor.getPooled = getPooledEvent;
  EventConstructor.release = releasePooledEvent;
}

export default SyntheticEvent;
```

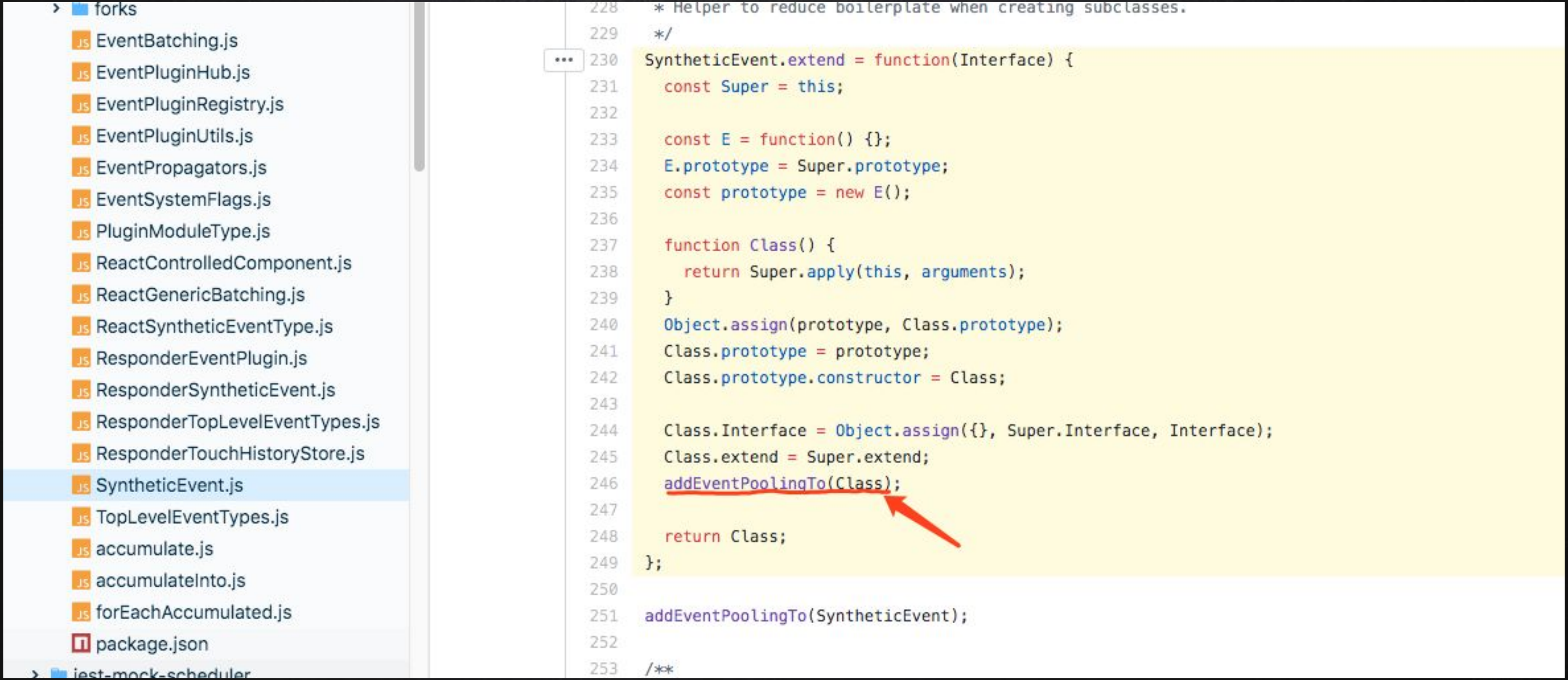
https://github.com/facebook/react/blob/67e3f3fb6e/packages/events/SyntheticEvent.js#L328



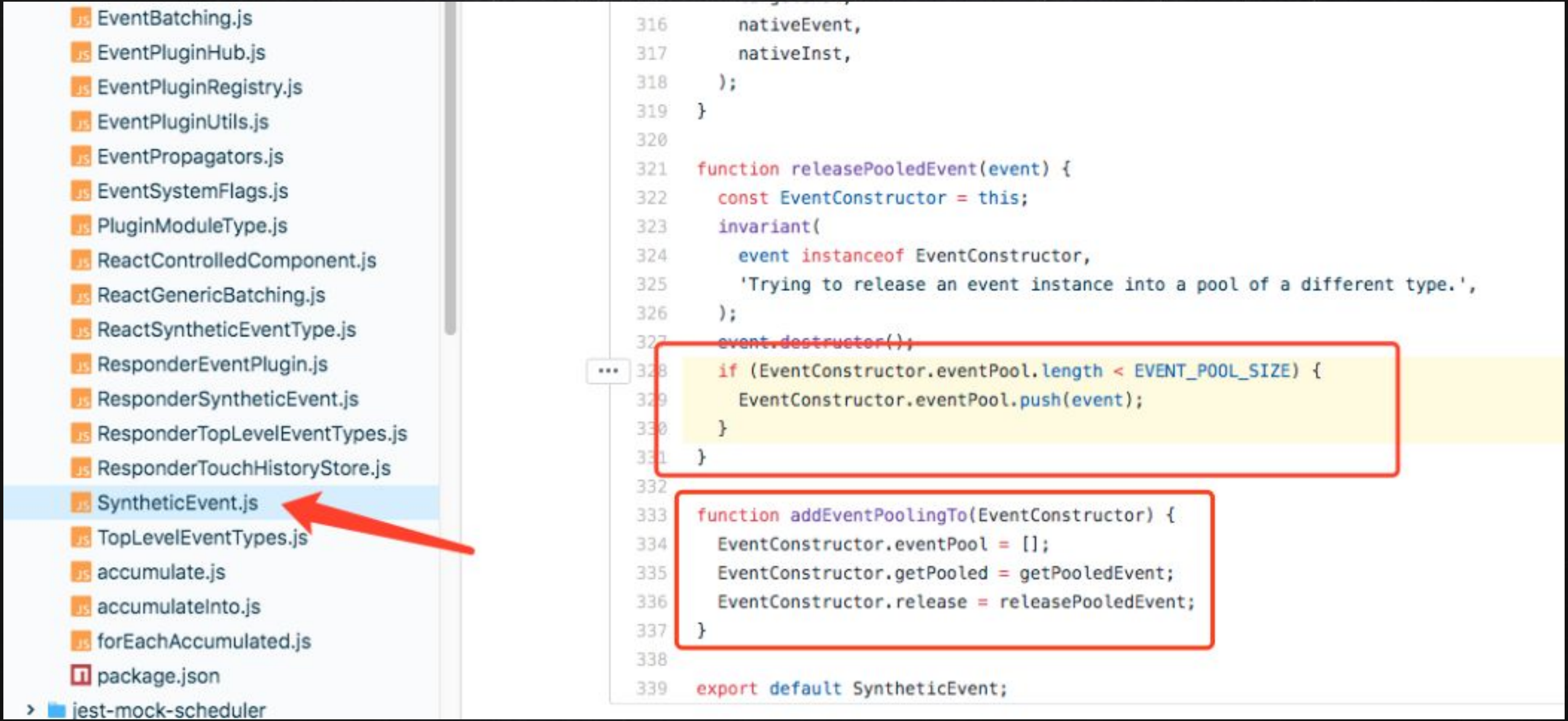
ReactBrowserEventEmitter.js
ReactDOMEventListener.js
SelectEventPlugin.js
SimpleEventPlugin.js
SyntheticAnimationEvent.js
SyntheticClipboardEvent.js
SyntheticCompositionEvent.js
SyntheticDragEvent.js
SyntheticFocusEvent.js
SyntheticInputEvent.js
SyntheticKeyboardEvent.js
SyntheticMouseEvent.js
SyntheticPointerEvent.js
SyntheticTouchEvent.js
SyntheticTransitionEvent.js
SyntheticUIEvent.js
SyntheticWheelEvent.js
checkPassiveEvents.js
getEventCharCode.js
getEventKey.js
getEventModifierState.js
getEventTarget.js
getVendorPrefixedEventName.js
isEventSupported.js
fire

28 lines (24 sloc) | 610 Bytes

```
/**
 * Copyright (c) Facebook, Inc. and its affiliates.
 *
 * This source code is licensed under the MIT license found in the
 * LICENSE file in the root directory of this source tree.
 */

import SyntheticMouseEvent from './SyntheticMouseEvent';

/**
 * @interface PointerEvent
 * @see http://www.w3.org/TR/pointerevents/
 */
const SyntheticPointerEvent = SyntheticMouseEvent.extend({
  pointerId: null,
  width: null,
  height: null,
  pressure: null,
  tangentialPressure: null,
  tiltX: null,
  tiltY: null,
  twist: null,
  pointerType: null,
  isPrimary: null,
});

export default SyntheticPointerEvent;
```

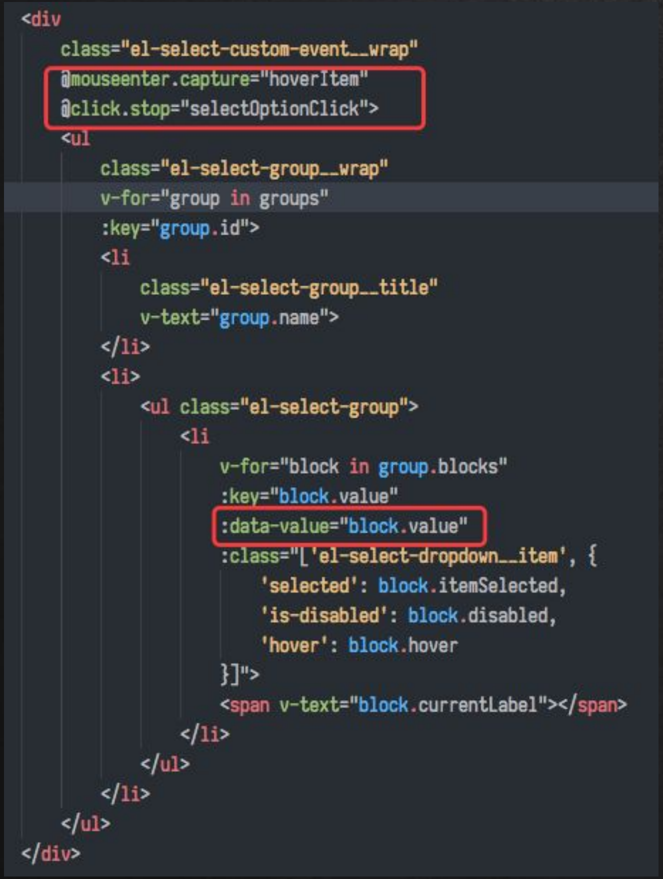


```
 * Helper to reduce boilerplate when creating subclasses.
 */
SyntheticEvent.extend = function(Interface) {
  const Super = this;

  const E = function() {};
  E.prototype = Super.prototype;
  const prototype = new E();

  function Class() {
    return Super.apply(this, arguments);
  }
  Object.assign(prototype, Class.prototype);
  Class.prototype = prototype;
  Class.prototype.constructor = Class;

  Class.Interface = Object.assign({}, Super.Interface, Interface);
  Class.extend = Super.extend;
  addEventPoolingTo(Class);

  return Class;
};

addEventPoolingTo(SyntheticEvent);

/**
```

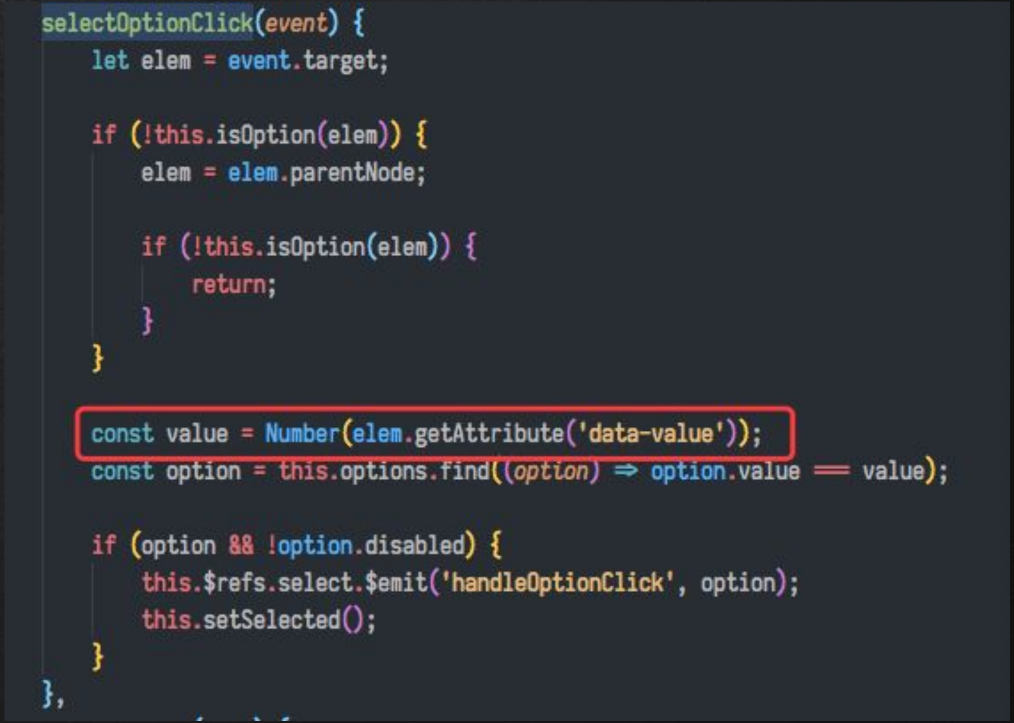


```
      nativeEvent,
      nativeInst,
    );
  }

function releasePooledEvent(event) {
  const EventConstructor = this;
  invariant(
    event instanceof EventConstructor,
    'Trying to release an event instance into a pool of a different type.',
  );
  event.destructor();
  if (EventConstructor.eventPool.length < EVENT_POOL_SIZE) {
    EventConstructor.eventPool.push(event);
  }
}

function addEventPoolingTo(EventConstructor) {
  EventConstructor.eventPool = [];
  EventConstructor.getPooled = getPooledEvent;
  EventConstructor.release = releasePooledEvent;
}

export default SyntheticEvent;
```

Vue 如何实现事件委托？



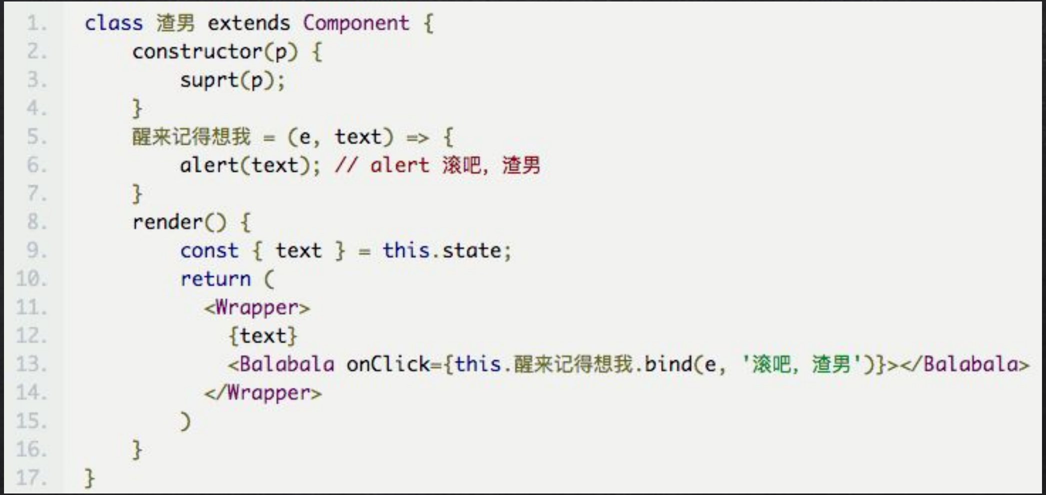
```
<div
    class="el-select-custom-event__wrap"
    @mouseenter.capture="hoverItem"
    @click.stop="selectOptionClick">
    <ul
        class="el-select-group__wrap"
        v-for="group in groups"
        :key="group.id">
        <li
            class="el-select-group__title"
            v-text="group.name">
        </li>
        <li>
            <ul class="el-select-group">
                <li
                    v-for="block in group.blocks"
                    :key="block.value"
                    :data-value="block.value"
                    :class="['el-select-dropdown__item', {
                        'selected': block.itemSelected,
                        'is-disabled': block.disabled,
                        'hover': block.hover
                    }]">
                    <span v-text="block.currentLabel"></span>
                </li>
            </ul>
        </li>
    </ul>
</div>
```



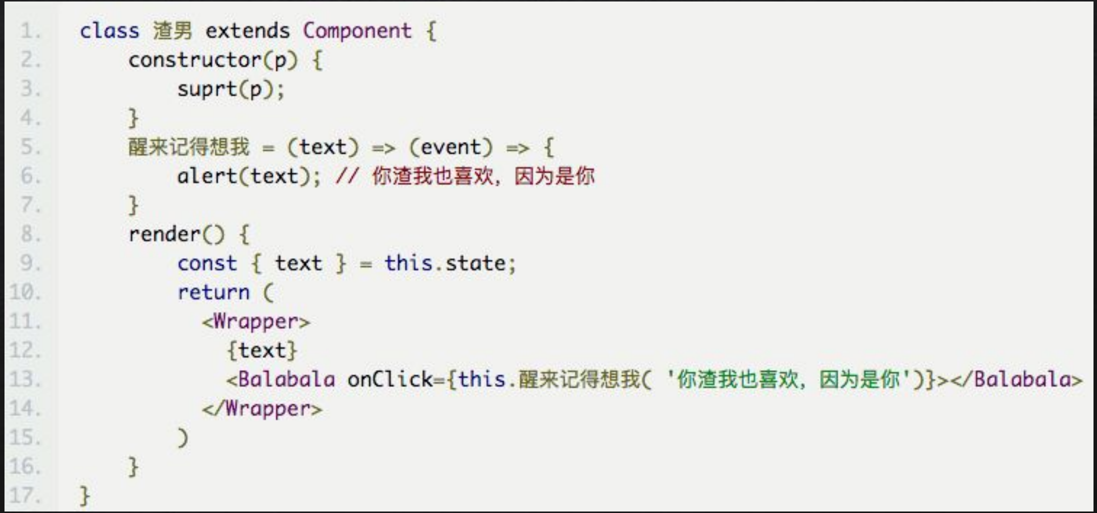
```
selectOptionClick(event) {
    let elem = event.target;

    if (!this.isOption(elem)) {
        elem = elem.parentNode;

        if (!this.isOption(elem)) {
            return;
        }
    }

    const value = Number(elem.getAttribute('data-value'));
    const option = this.options.find((option) => option.value === value);

    if (option && !option.disabled) {
        this.$refs.select.$emit('handleOptionClick', option);
        this.setSelected();
    }
},
```

4.

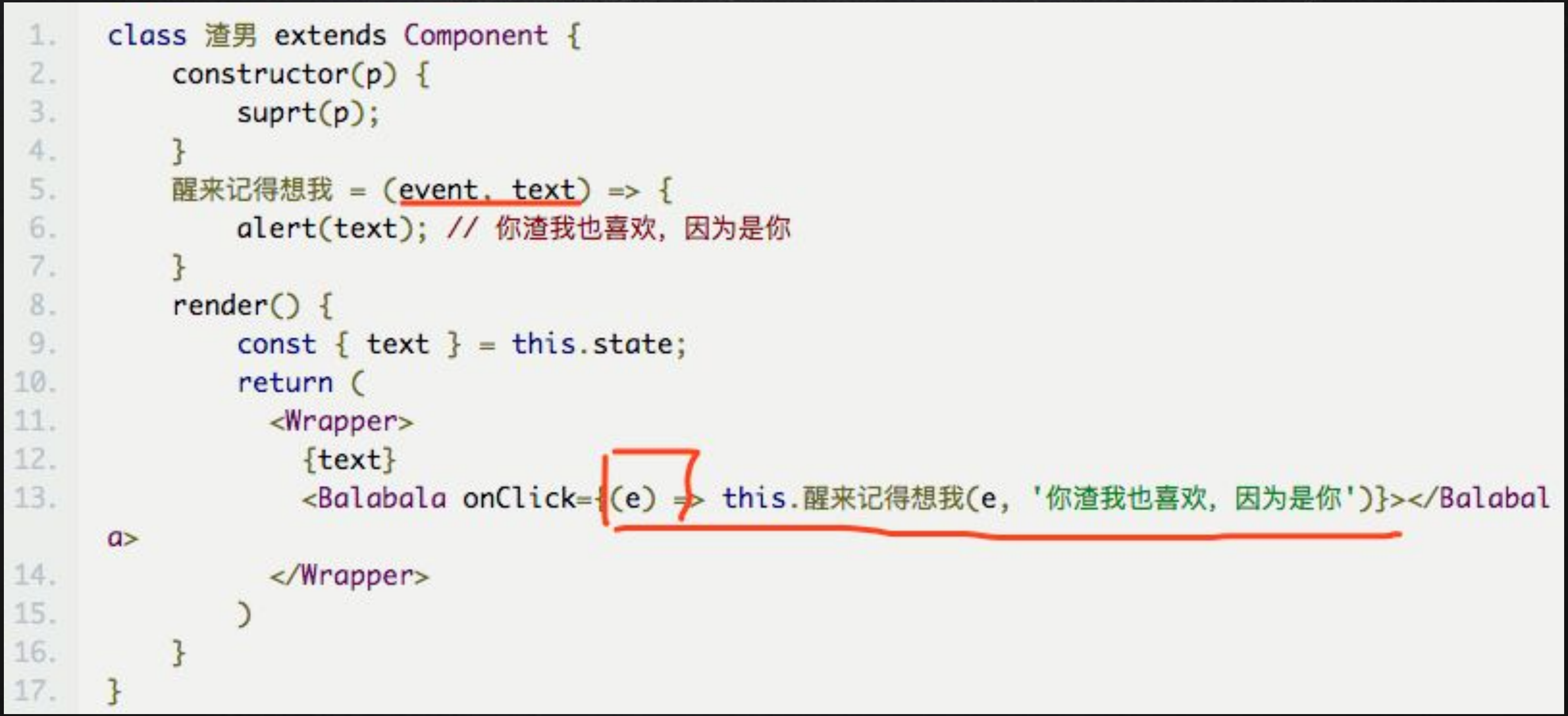
事件传参



```
class 渣男 extends Component {
    constructor(p) {
        suprt(p);
    }
    醒来记得想我 = (e, text) => {
        alert(text); // alert 滚吧，渣男
    }
    render() {
        const { text } = this.state;
        return (
          <Wrapper>
            {text}
            <Balabala onClick={this.醒来记得想我.bind(e, '滚吧，渣男')}></Balabala>
          </Wrapper>
        )
    }
}
```

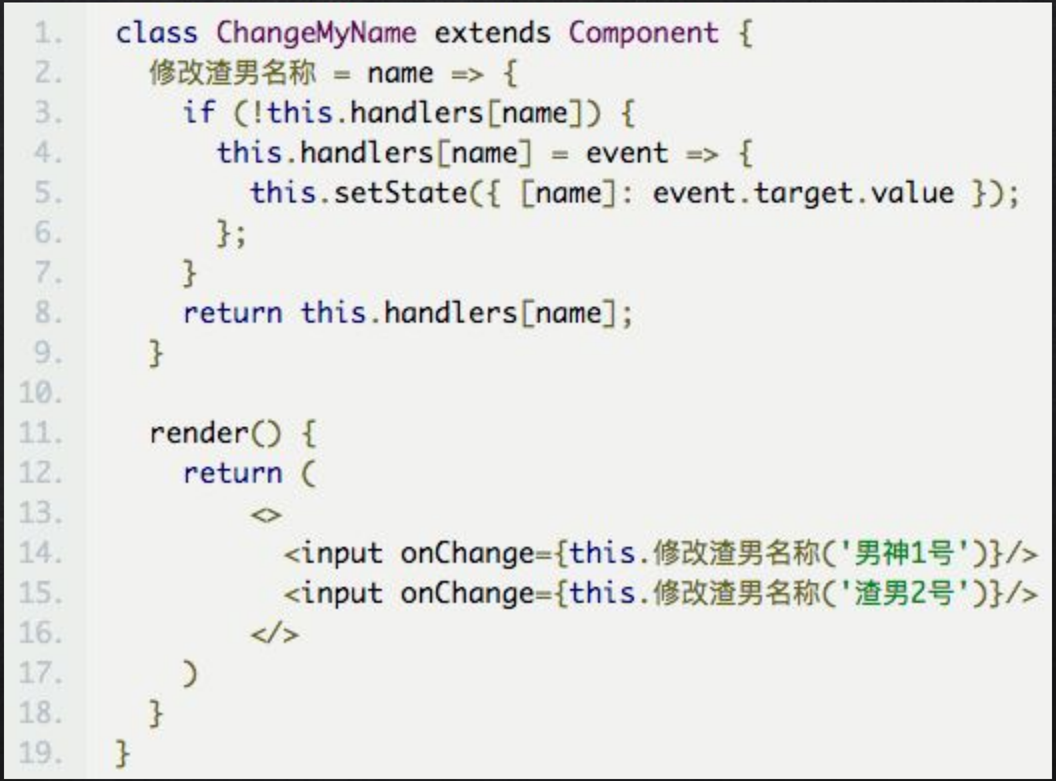


```
class 渣男 extends Component {
    constructor(p) {
        suprt(p);
    }
    醒来记得想我 = (text) => (event) => {
        alert(text); // 你渣我也喜欢，因为是你
    }
    render() {
        const { text } = this.state;
        return (
          <Wrapper>
            {text}
            <Balabala onClick={this.醒来记得想我( '你渣我也喜欢，因为是你')}></Balabala>
          </Wrapper>
        )
    }
}
```



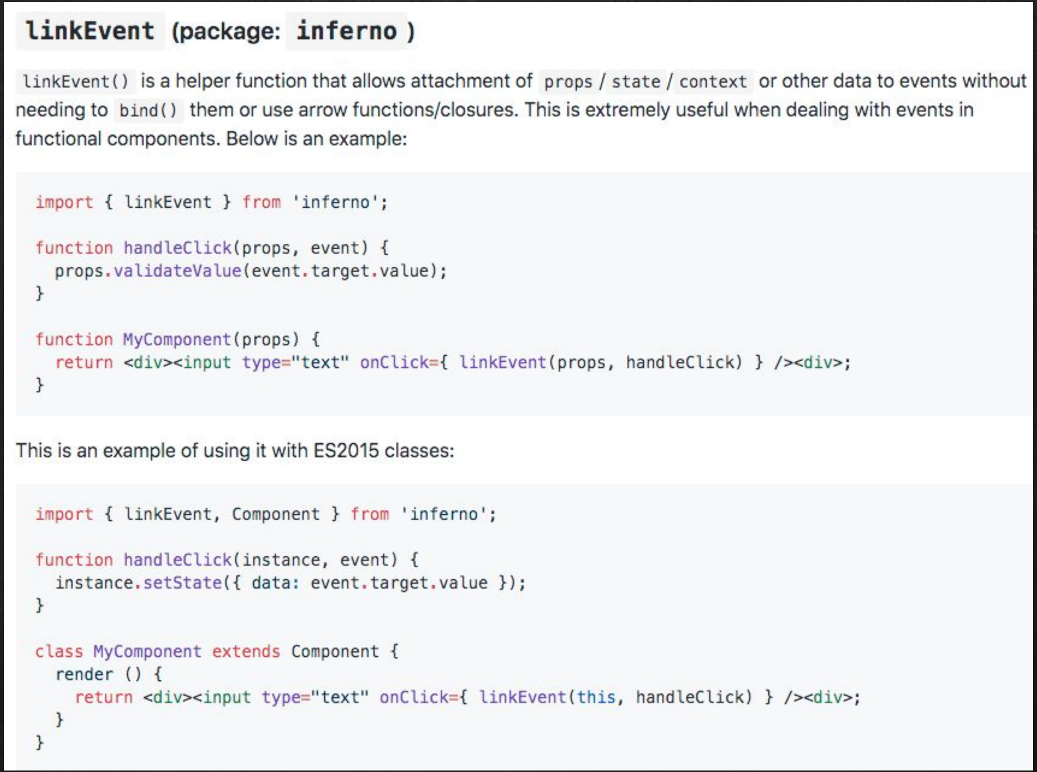
```
class 渣男 extends Component {
    constructor(p) {
        suprt(p);
    }
    醒来记得想我 = (event, text) => {
        alert(text); // 你渣我也喜欢, 因为是你
    }
    render() {
        const { text } = this.state;
        return (
          <Wrapper>
            {text}
            <Balabala onClick={(e) => this.醒来记得想我(e, '你渣我也喜欢, 因为是你')}></Balabal
a>
          </Wrapper>
        )
    }
}
```

cache 事件



```
class ChangeMyName extends Component {
  修改渣男名称 = name => {
    if (!this.handlers[name]) {
      this.handlers[name] = event => {
        this.setState({ [name]: event.target.value });
      };
    }
    return this.handlers[name];
  }

  render() {
    return (
        <>
          <input onChange={this.修改渣男名称('男神1号')}/>
          <input onChange={this.修改渣男名称('渣男2号')}/>
        </>
    )
  }
}
```

http://meckodo.com/article/how-to-bind-event-in-react



linkEvent (package: inferno)

`linkEvent()` is a helper function that allows attachment of `props` / `state` / `context` or other data to events without needing to `bind()` them or use arrow functions/closures. This is extremely useful when dealing with events in functional components. Below is an example:

```
import { linkEvent } from 'inferno';

function handleClick(props, event) {
  props.validateValue(event.target.value);
}

function MyComponent(props) {
  return <div><input type="text" onClick={ linkEvent(props, handleClick) } /><div>;
}
```

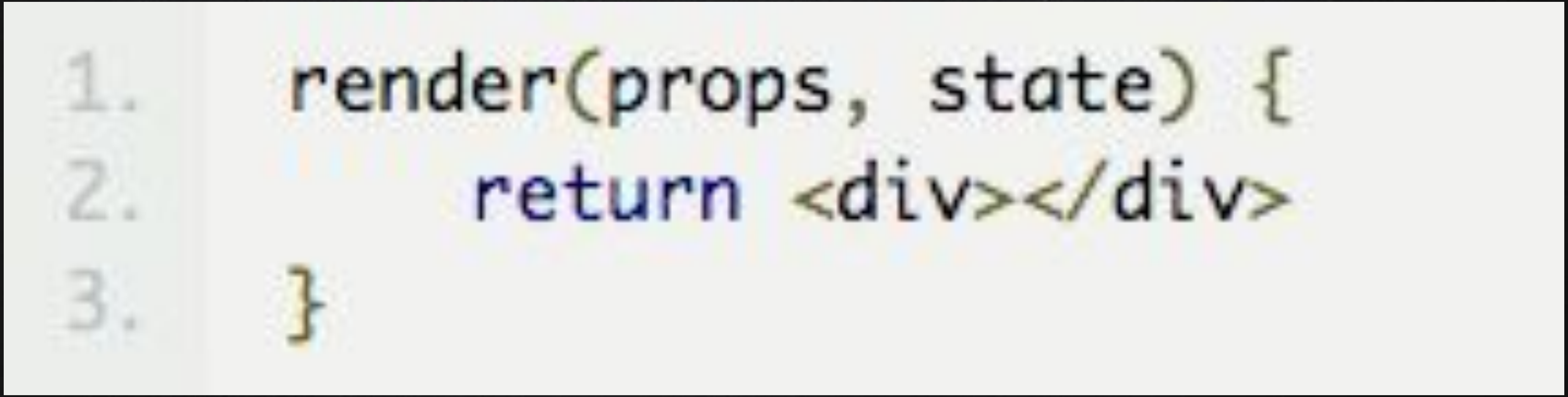
This is an example of using it with ES2015 classes:

```
import { linkEvent, Component } from 'inferno';

function handleClick(instance, event) {
  instance.setState({ data: event.target.value });
}

class MyComponent extends Component {
  render () {
    return <div><input type="text" onClick={ linkEvent(this, handleClick) } /><div>;
  }
}
```

https://github.com/infernojs/inferno/blob/master/README.md#linkevent-package-inferno

5.

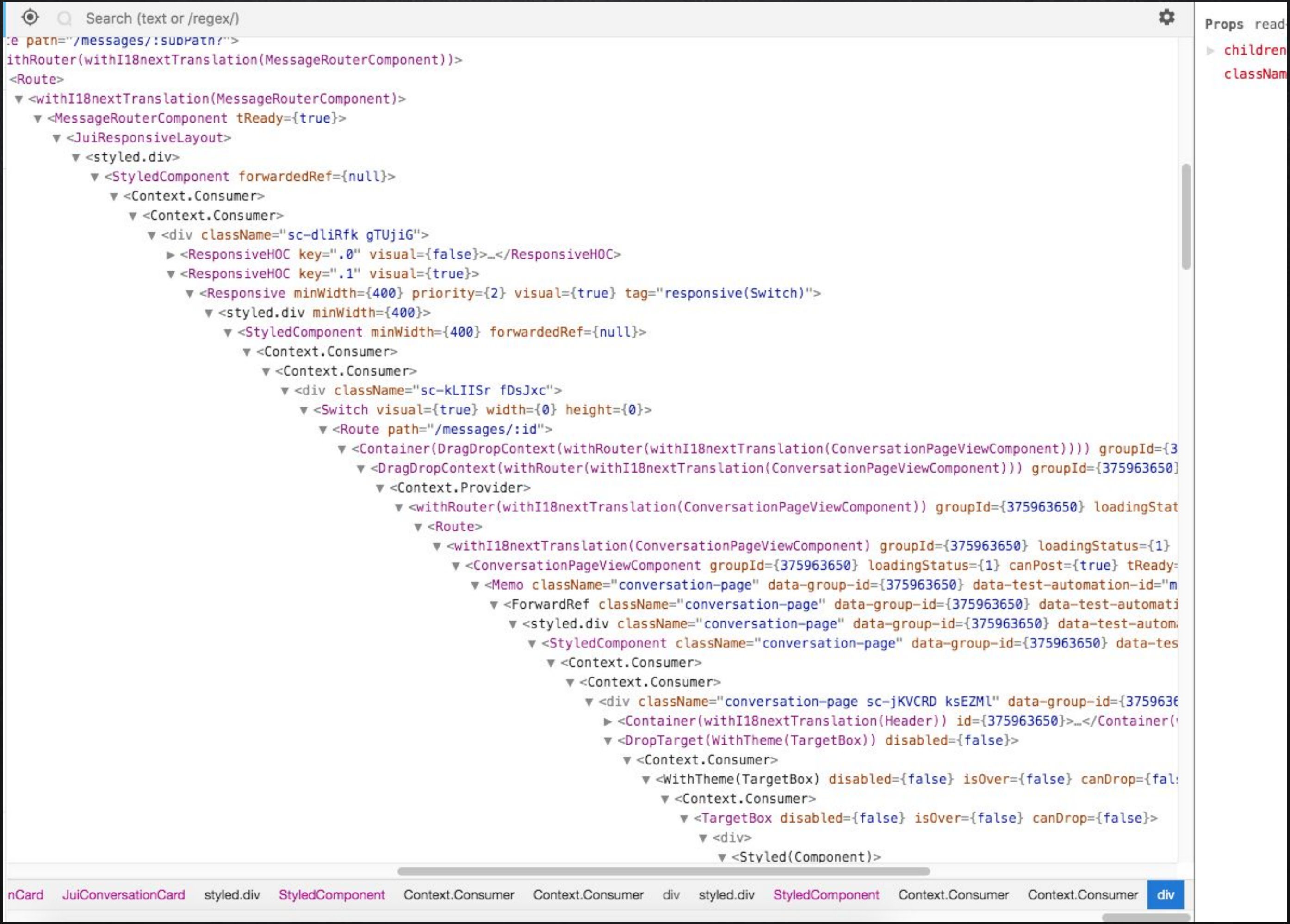
render()



```
render(props, state) {
    return <div></div>
}
```

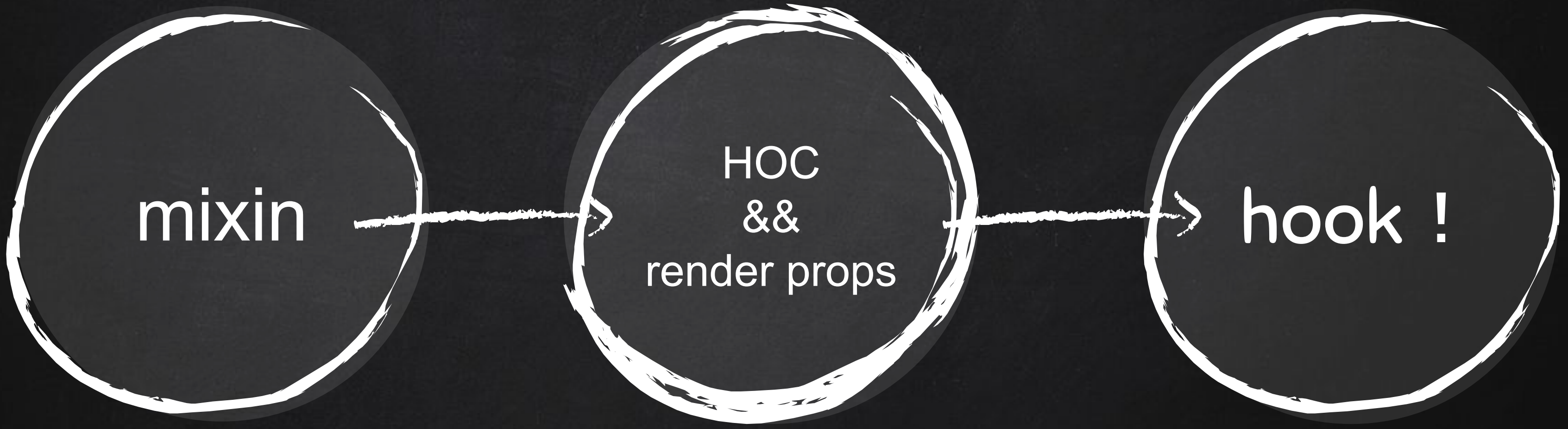
6.

逻辑复用 && 组件复用



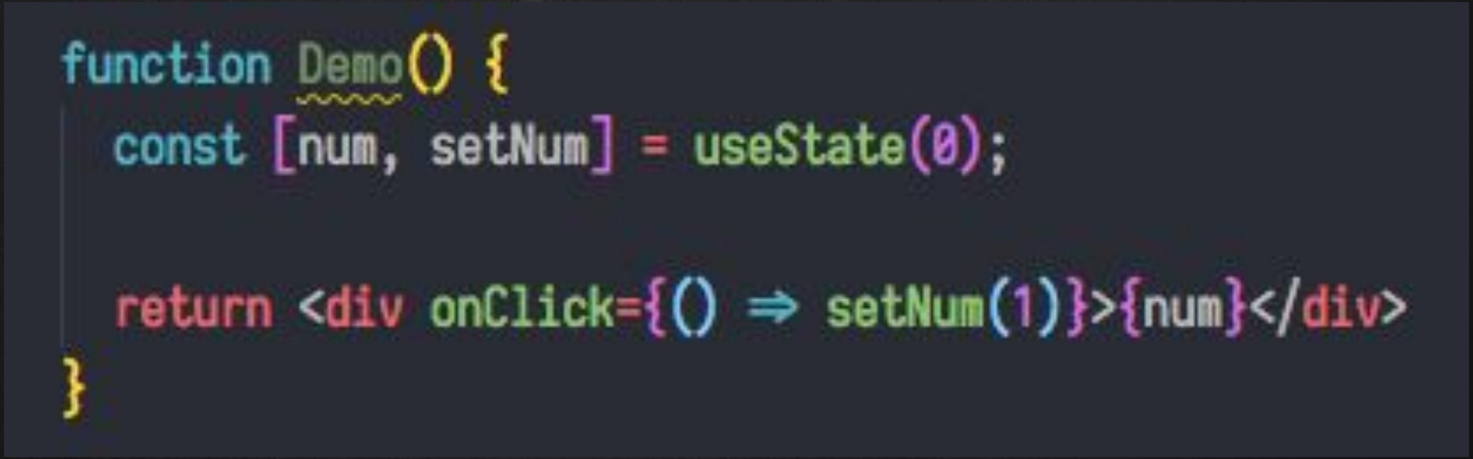
```
:e path="/messages/:subPath?">
ithRouter(withI18nextTranslation(MessageRouterComponent))>
<Route>
 ▼ <withI18nextTranslation(MessageRouterComponent)>
   ▼ <MessageRouterComponent tReady={true}>
     ▼ <JuiResponsiveLayout>
       ▼ <styled.div>
         ▼ <StyledComponent forwardedRef={null}>
           ▼ <Context.Consumer>
             ▼ <Context.Consumer>
               ▼ <div className="sc-dliRfk gTUjiG">
                 ▶ <ResponsiveHOC key=".0" visual={false}>…</ResponsiveHOC>
                 ▼ <ResponsiveHOC key=".1" visual={true}>
                   ▼ <Responsive minWidth={400} priority={2} visual={true} tag="responsive(Switch)">
                     ▼ <styled.div minWidth={400}>
                       ▼ <StyledComponent minWidth={400} forwardedRef={null}>
                         ▼ <Context.Consumer>
                           ▼ <Context.Consumer>
                             ▼ <div className="sc-kLIISr fDsJxc">
                               ▼ <Switch visual={true} width={0} height={0}>
                                 ▼ <Route path="/messages/:id">
                                   ▼ <Container(DragDropContext(withRouter(withI18nextTranslation(ConversationPageViewComponent)))) groupId={3
                                     ▼ <DragDropContext(withRouter(withI18nextTranslation(ConversationPageViewComponent))) groupId={375963650}
                                       ▼ <Context.Provider>
                                         ▼ <withRouter(withI18nextTranslation(ConversationPageViewComponent)) groupId={375963650} loadingStat
                                           ▼ <Route>
                                             ▼ <withI18nextTranslation(ConversationPageViewComponent) groupId={375963650} loadingStatus={1}
                                               ▼ <ConversationPageViewComponent groupId={375963650} loadingStatus={1} canPost={true} tReady=
                                                 ▼ <Memo className="conversation-page" data-group-id={375963650} data-test-automation-id="m
                                                   ▼ <ForwardRef className="conversation-page" data-group-id={375963650} data-test-automati
                                                     ▼ <styled.div className="conversation-page" data-group-id={375963650} data-test-autom
                                                       ▼ <StyledComponent className="conversation-page" data-group-id={375963650} data-tes
                                                         ▼ <Context.Consumer>
                                                           ▼ <Context.Consumer>
                                                             ▼ <div className="conversation-page sc-jKVCRD ksEZMl" data-group-id={375963
                                                               ▶ <Container(withI18nextTranslation(Header)) id={375963650}>…</Container(
                                                               ▼ <DropTarget(WithTheme(TargetBox)) disabled={false}>
                                                                 ▼ <Context.Consumer>
                                                                   ▼ <WithTheme(TargetBox) disabled={false} isOver={false} canDrop={fal
                                                                     ▼ <Context.Consumer>
                                                                       ▼ <TargetBox disabled={false} isOver={false} canDrop={false}>
                                                                         ▼ <div>
                                                                           ▼ <Styled(Component)>
```

nCard JuiConversationCard styled.div StyledComponent Context.Consumer Context.Consumer div styled.div StyledComponent Context.Consumer Context.Consumer div

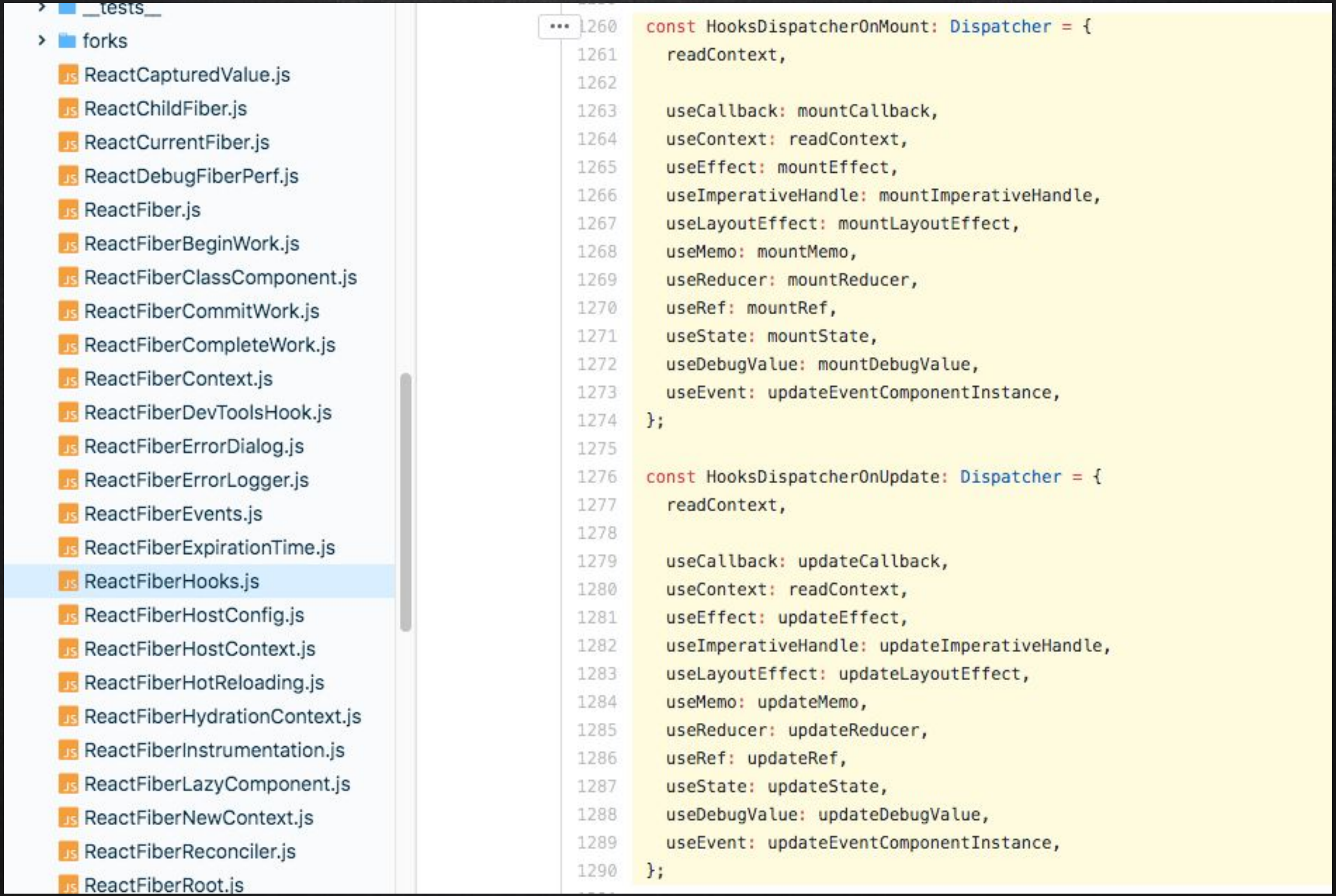

mixin
HOC
&&
render props
hook !

🎉 React hook

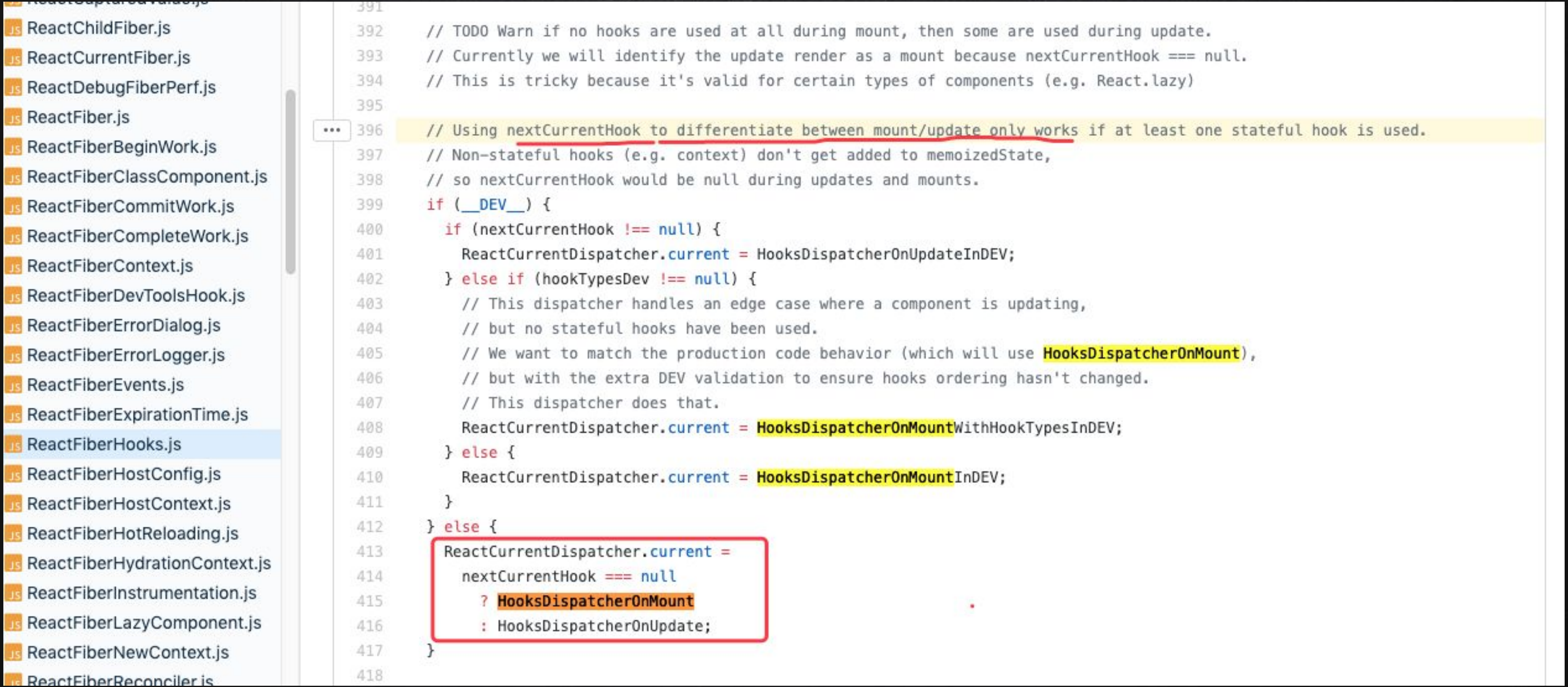
- ✘ useState
- ✘ useReducer
- ✘ useEffect
- ✘ useContext
- ✘ useRef
- ✘ ...



```jsx
function Demo() {
  const [num, setNum] = useState(0);

  return <div onClick={() => setNum(1)}>{num}</div>
}
```



```
const HooksDispatcherOnMount: Dispatcher = {
  readContext,

  useCallback: mountCallback,
  useContext: readContext,
  useEffect: mountEffect,
  useImperativeHandle: mountImperativeHandle,
  useLayoutEffect: mountLayoutEffect,
  useMemo: mountMemo,
  useReducer: mountReducer,
  useRef: mountRef,
  useState: mountState,
  useDebugValue: mountDebugValue,
  useEvent: updateEventComponentInstance,
};

const HooksDispatcherOnUpdate: Dispatcher = {
  readContext,

  useCallback: updateCallback,
  useContext: readContext,
  useEffect: updateEffect,
  useImperativeHandle: updateImperativeHandle,
  useLayoutEffect: updateLayoutEffect,
  useMemo: updateMemo,
  useReducer: updateReducer,
  useRef: updateRef,
  useState: updateState,
  useDebugValue: updateDebugValue,
  useEvent: updateEventComponentInstance,
};
```



ReactChildFiber.js
ReactCurrentFiber.js
ReactDebugFiberPerf.js
ReactFiber.js
ReactFiberBeginWork.js
ReactFiberClassComponent.js
ReactFiberCommitWork.js
ReactFiberCompleteWork.js
ReactFiberContext.js
ReactFiberDevToolsHook.js
ReactFiberErrorDialog.js
ReactFiberErrorLogger.js
ReactFiberEvents.js
ReactFiberExpirationTime.js
ReactFiberHooks.js
ReactFiberHostConfig.js
ReactFiberHostContext.js
ReactFiberHotReloading.js
ReactFiberHydrationContext.js
ReactFiberInstrumentation.js
ReactFiberLazyComponent.js
ReactFiberNewContext.js
ReactFiberReconciler.js

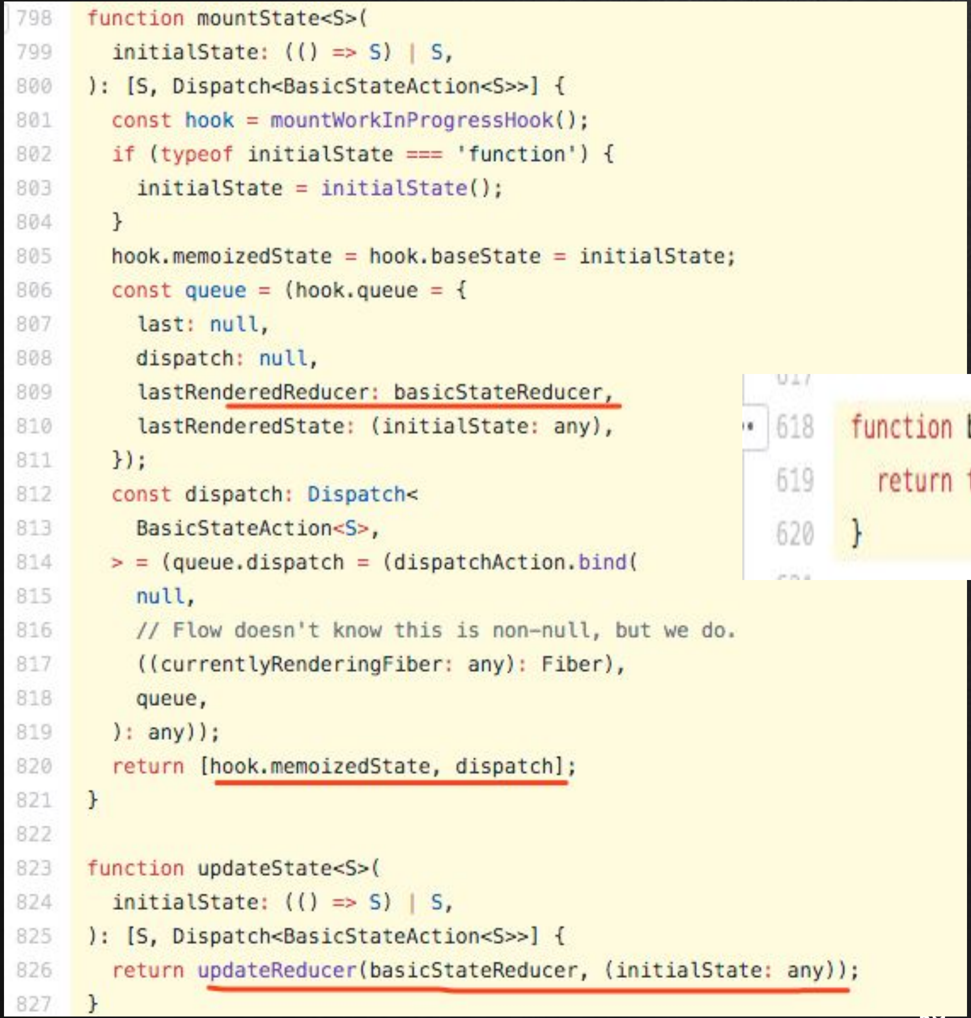
```
  // TODO Warn if no hooks are used at all during mount, then some are used during update.
  // Currently we will identify the update render as a mount because nextCurrentHook === null.
  // This is tricky because it's valid for certain types of components (e.g. React.lazy)

  // Using nextCurrentHook to differentiate between mount/update only works if at least one stateful hook is used.
  // Non-stateful hooks (e.g. context) don't get added to memoizedState,
  // so nextCurrentHook would be null during updates and mounts.
  if (__DEV__) {
    if (nextCurrentHook !== null) {
      ReactCurrentDispatcher.current = HooksDispatcherOnUpdateInDEV;
    } else if (hookTypesDev !== null) {
      // This dispatcher handles an edge case where a component is updating,
      // but no stateful hooks have been used.
      // We want to match the production code behavior (which will use HooksDispatcherOnMount),
      // but with the extra DEV validation to ensure hooks ordering hasn't changed.
      // This dispatcher does that.
      ReactCurrentDispatcher.current = HooksDispatcherOnMountWithHookTypesInDEV;
    } else {
      ReactCurrentDispatcher.current = HooksDispatcherOnMountInDEV;
    }
  } else {
    ReactCurrentDispatcher.current =
      nextCurrentHook === null
        ? HooksDispatcherOnMount
        : HooksDispatcherOnUpdate;
  }
```

https://github.com/facebook/react/blob/67e3f3fb6e/packages/react-reconciler/src/ReactFiberHooks.js#L396

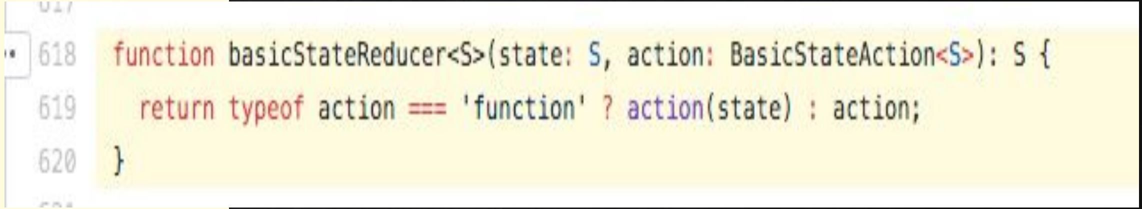
先有 useState 后有 useReducer?

Or

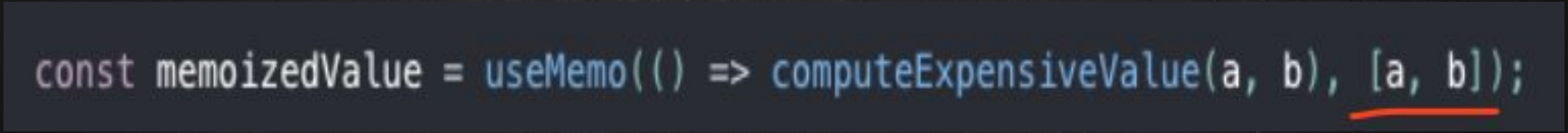
先有 useReducer 后有 useState?



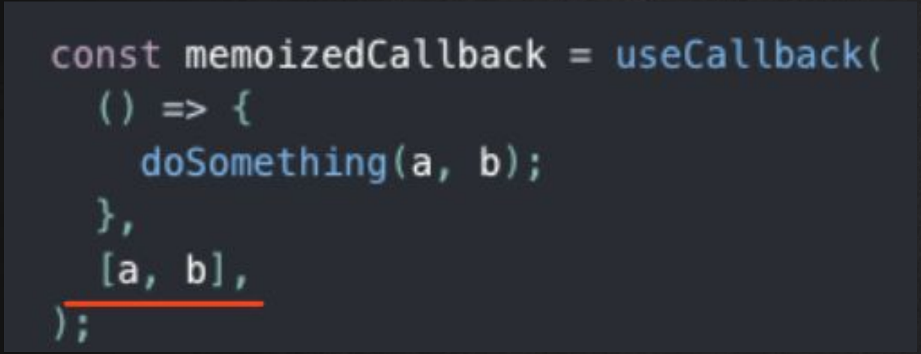
```
function mountState<S>(
  initialState: (() => S) | S,
): [S, Dispatch<BasicStateAction<S>>] {
  const hook = mountWorkInProgressHook();
  if (typeof initialState === 'function') {
    initialState = initialState();
  }
  hook.memoizedState = hook.baseState = initialState;
  const queue = (hook.queue = {
    last: null,
    dispatch: null,
    lastRenderedReducer: basicStateReducer,
    lastRenderedState: (initialState: any),
  });
  const dispatch: Dispatch<
    BasicStateAction<S>,
  > = (queue.dispatch = (dispatchAction.bind(
    null,
    // Flow doesn't know this is non-null, but we do.
    ((currentlyRenderingFiber: any): Fiber),
    queue,
  ): any));
  return [hook.memoizedState, dispatch];
}

function updateState<S>(
  initialState: (() => S) | S,
): [S, Dispatch<BasicStateAction<S>>] {
  return updateReducer(basicStateReducer, (initialState: any));
}
```



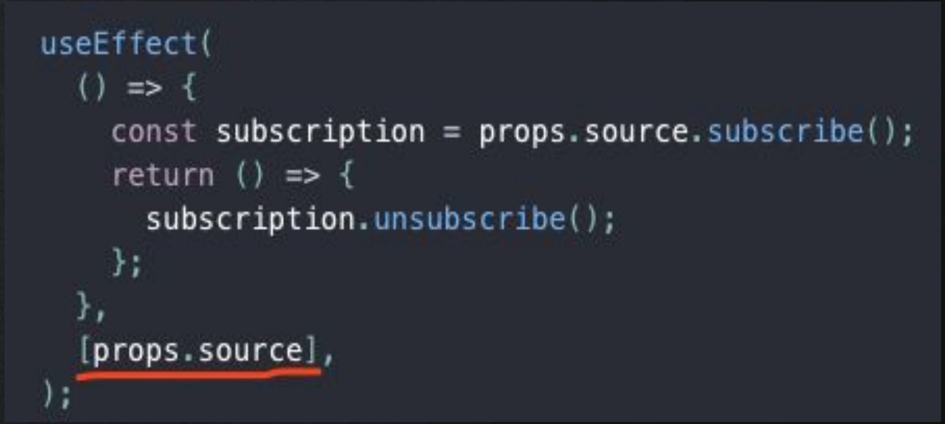
```
function basicStateReducer<S>(state: S, action: BasicStateAction<S>): S {
  return typeof action === 'function' ? action(state) : action;
}
```



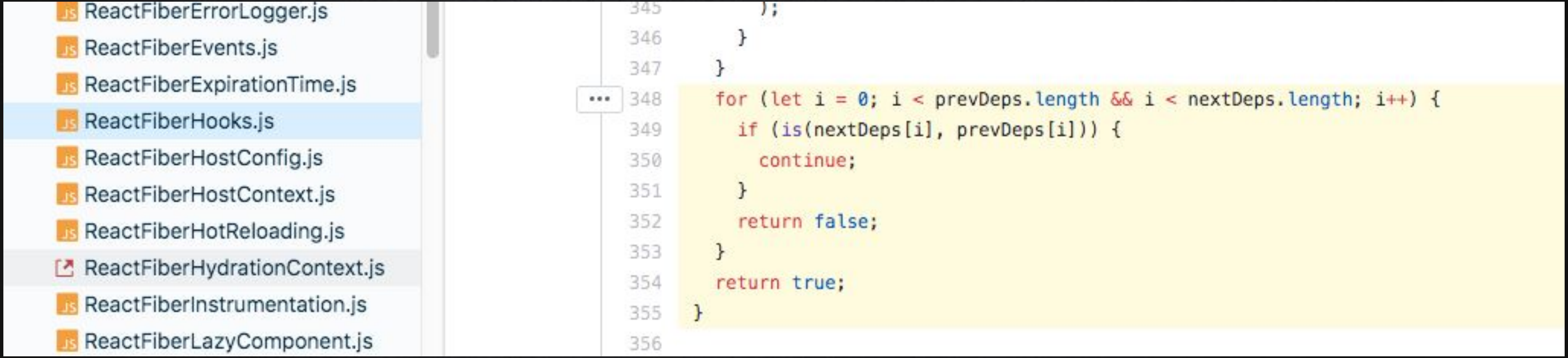
```
const memoizedValue = useMemo(() => computeExpensiveValue(a, b), [a, b]);
```



```
const memoizedCallback = useCallback(
  () => {
    doSomething(a, b);
  },
  [a, b],
);
```

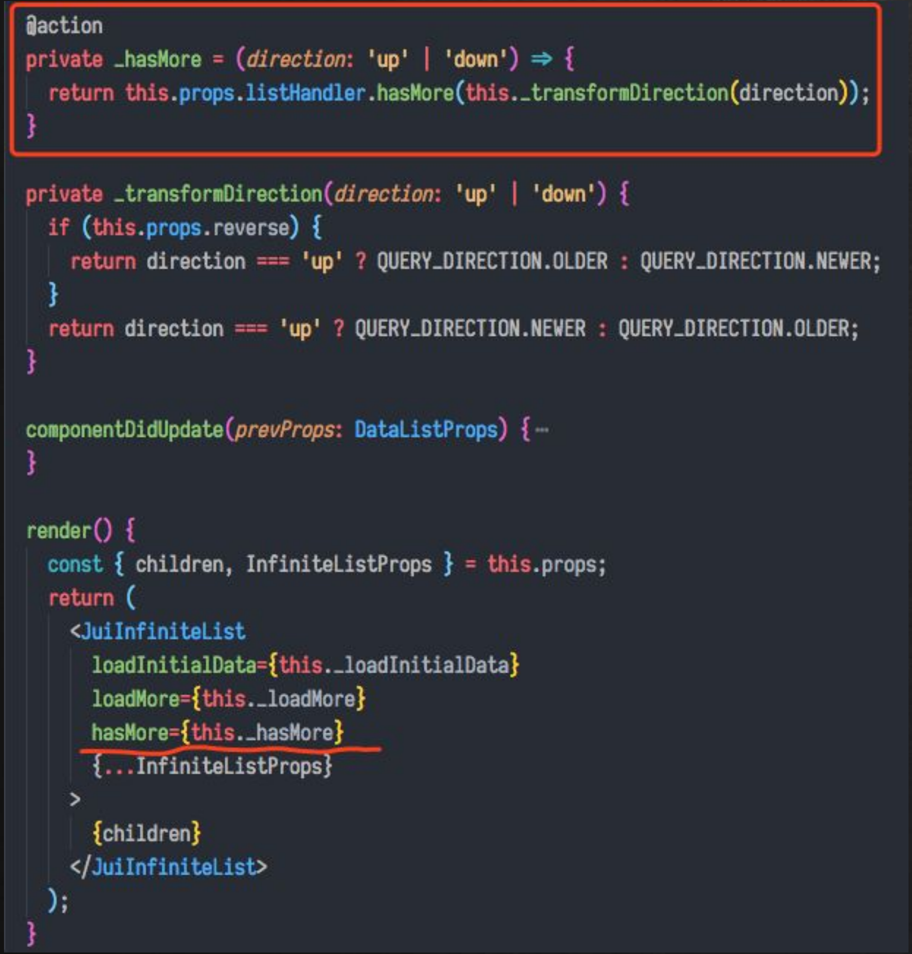


```
useEffect(
  () => {
    const subscription = props.source.subscribe();
    return () => {
      subscription.unsubscribe();
    };
  },
  [props.source],
);
```

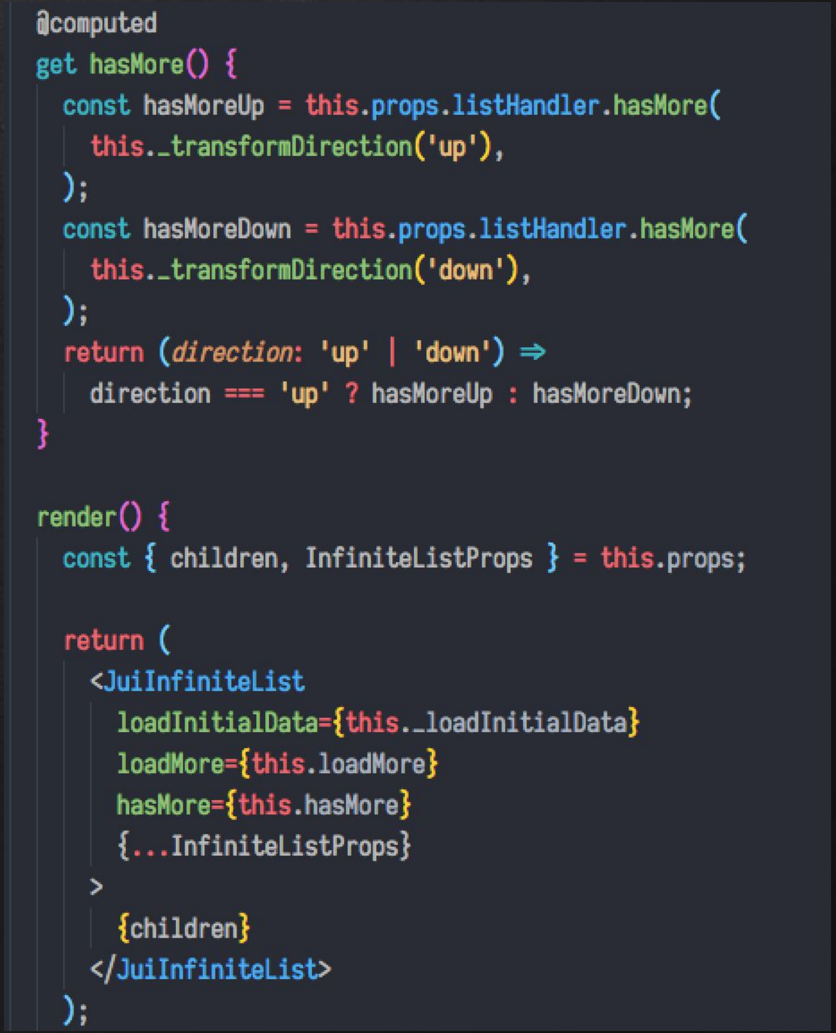


ReactFiberErrorLogger.js
ReactFiberEvents.js
ReactFiberExpirationTime.js
ReactFiberHooks.js
ReactFiberHostConfig.js
ReactFiberHostContext.js
ReactFiberHotReloading.js
ReactFiberHydrationContext.js
ReactFiberInstrumentation.js
ReactFiberLazyComponent.js

```
        );
      }
    }
    for (let i = 0; i < prevDeps.length && i < nextDeps.length; i++) {
      if (is(nextDeps[i], prevDeps[i])) {
        continue;
      }
      return false;
    }
    return true;
  }

```

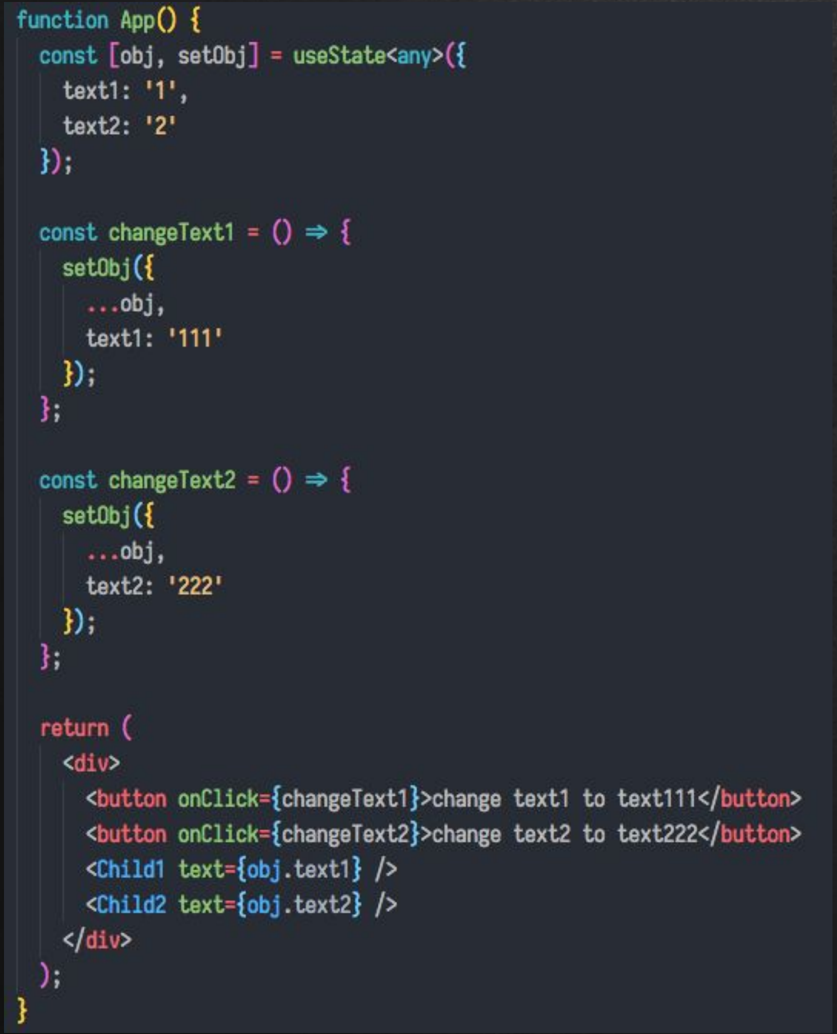


```
@action
private _hasMore = (direction: 'up' | 'down') => {
  return this.props.listHandler.hasMore(this._transformDirection(direction));
}

private _transformDirection(direction: 'up' | 'down') {
  if (this.props.reverse) {
    return direction === 'up' ? QUERY_DIRECTION.OLDER : QUERY_DIRECTION.NEWER;
  }
  return direction === 'up' ? QUERY_DIRECTION.NEWER : QUERY_DIRECTION.OLDER;
}

componentDidUpdate(prevProps: DataListProps) {…
}

render() {
  const { children, InfiniteListProps } = this.props;
  return (
    <JuiInfiniteList
      loadInitialData={this._loadInitialData}
      loadMore={this._loadMore}
      hasMore={this._hasMore}
      {...InfiniteListProps}
    >
      {children}
    </JuiInfiniteList>
  );
}
```

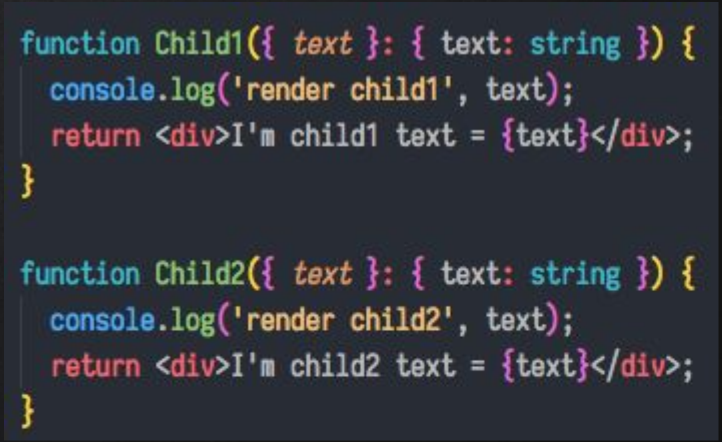


```
@computed
get hasMore() {
  const hasMoreUp = this.props.listHandler.hasMore(
    this._transformDirection('up'),
  );
  const hasMoreDown = this.props.listHandler.hasMore(
    this._transformDirection('down'),
  );
  return (direction: 'up' | 'down') =>
    direction === 'up' ? hasMoreUp : hasMoreDown;
}

render() {
  const { children, InfiniteListProps } = this.props;

  return (
    <JuiInfiniteList
      loadInitialData={this._loadInitialData}
      loadMore={this.loadMore}
      hasMore={this.hasMore}
      {...InfiniteListProps}
    >
      {children}
    </JuiInfiniteList>
  );
```

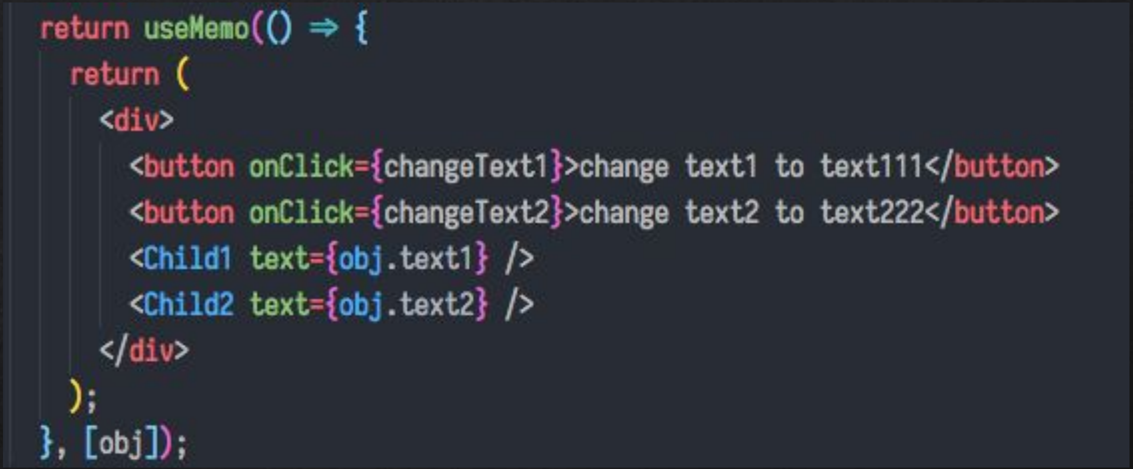
useMemo



```tsx
function App() {
  const [obj, setObj] = useState<any>({
    text1: '1',
    text2: '2'
  });

  const changeText1 = () => {
    setObj({
      ...obj,
      text1: '111'
    });
  };

  const changeText2 = () => {
    setObj({
      ...obj,
      text2: '222'
    });
  };

  return (
    <div>
      <button onClick={changeText1}>change text1 to text111</button>
      <button onClick={changeText2}>change text2 to text222</button>
      <Child1 text={obj.text1} />
      <Child2 text={obj.text2} />
    </div>
  );
}
```

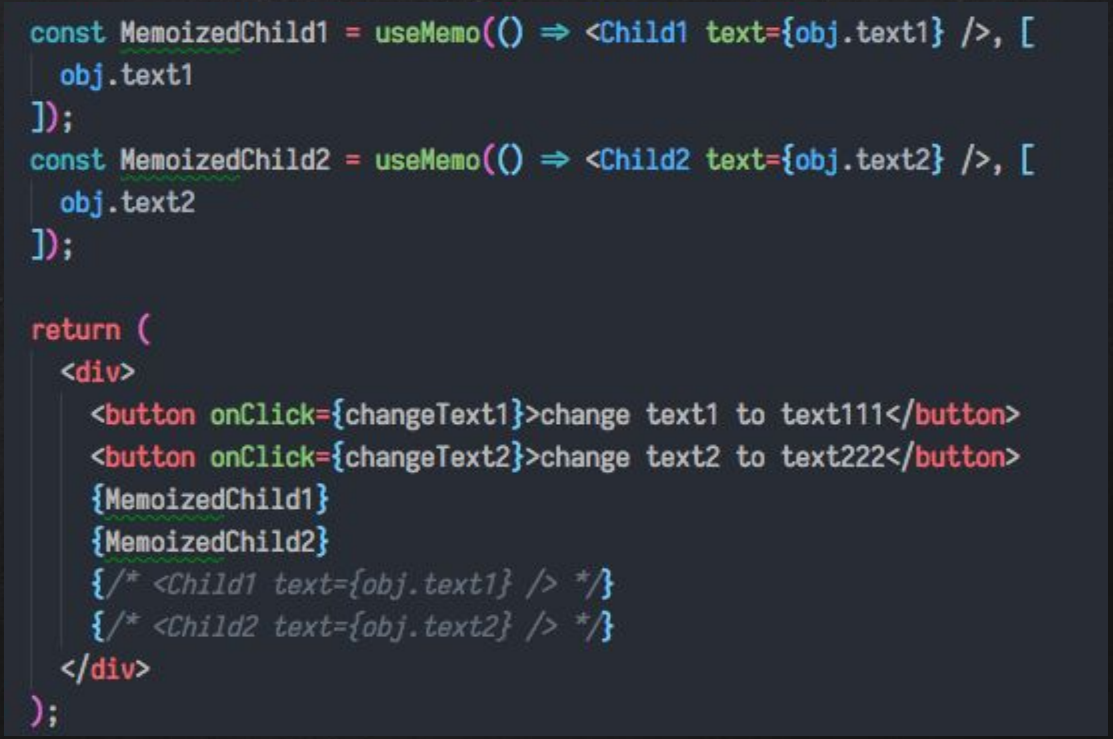


```tsx
function Child1({ text }: { text: string }) {
  console.log('render child1', text);
  return <div>I'm child1 text = {text}</div>;
}

function Child2({ text }: { text: string }) {
  console.log('render child2', text);
  return <div>I'm child2 text = {text}</div>;
}
```



```
return useMemo(() => {
  return (
    <div>
      <button onClick={changeText1}>change text1 to text111</button>
      <button onClick={changeText2}>change text2 to text222</button>
      <Child1 text={obj.text1} />
      <Child2 text={obj.text2} />
    </div>
  );
}, [obj]);
```

应该怎么写?🌰



```
const MemoizedChild1 = useMemo(() => <Child1 text={obj.text1} />, [
  obj.text1
]);
const MemoizedChild2 = useMemo(() => <Child2 text={obj.text2} />, [
  obj.text2
]);

return (
  <div>
    <button onClick={changeText1}>change text1 to text111</button>
    <button onClick={changeText2}>change text2 to text222</button>
    {MemoizedChild1}
    {MemoizedChild2}
    {/* <Child1 text={obj.text1} /> */}
    {/* <Child2 text={obj.text2} /> */}
  </div>
);
```

我们如何迁移？

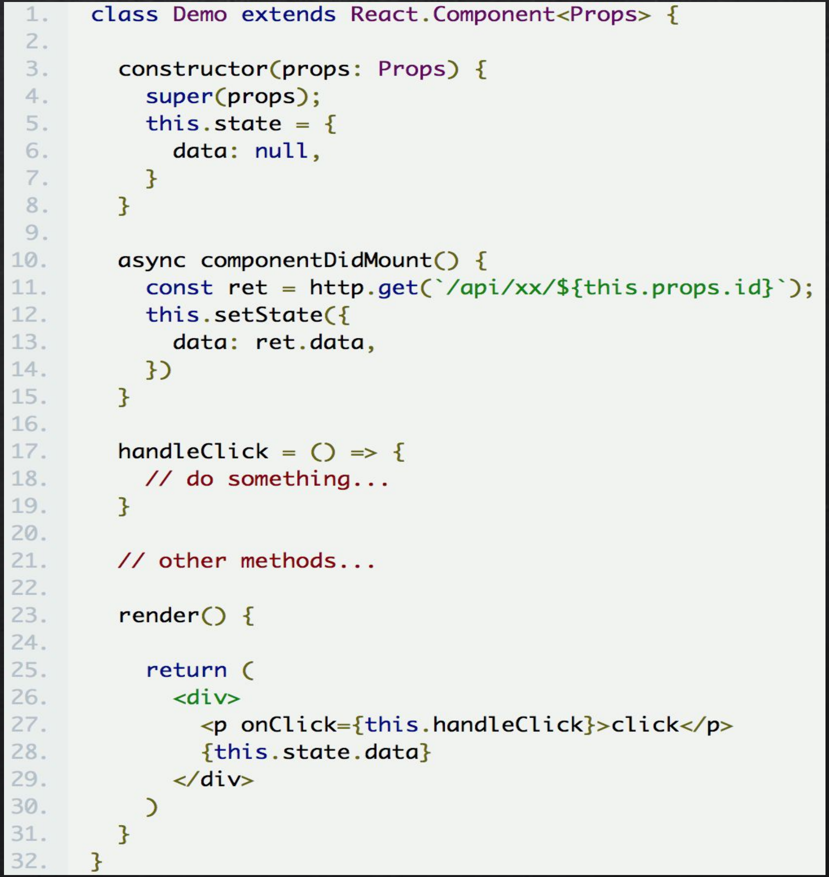
面向现代化，面向世界，面向未来

从视图和逻辑说起

```
<div>
    <input placeholder="修改名字" onChange={handleChange} />
    <p>姓名: {username}</p>
<div>
```

```
username = api.getUserName()

handleChange(e) {
    this.username = e.target.value;
}
```

Before



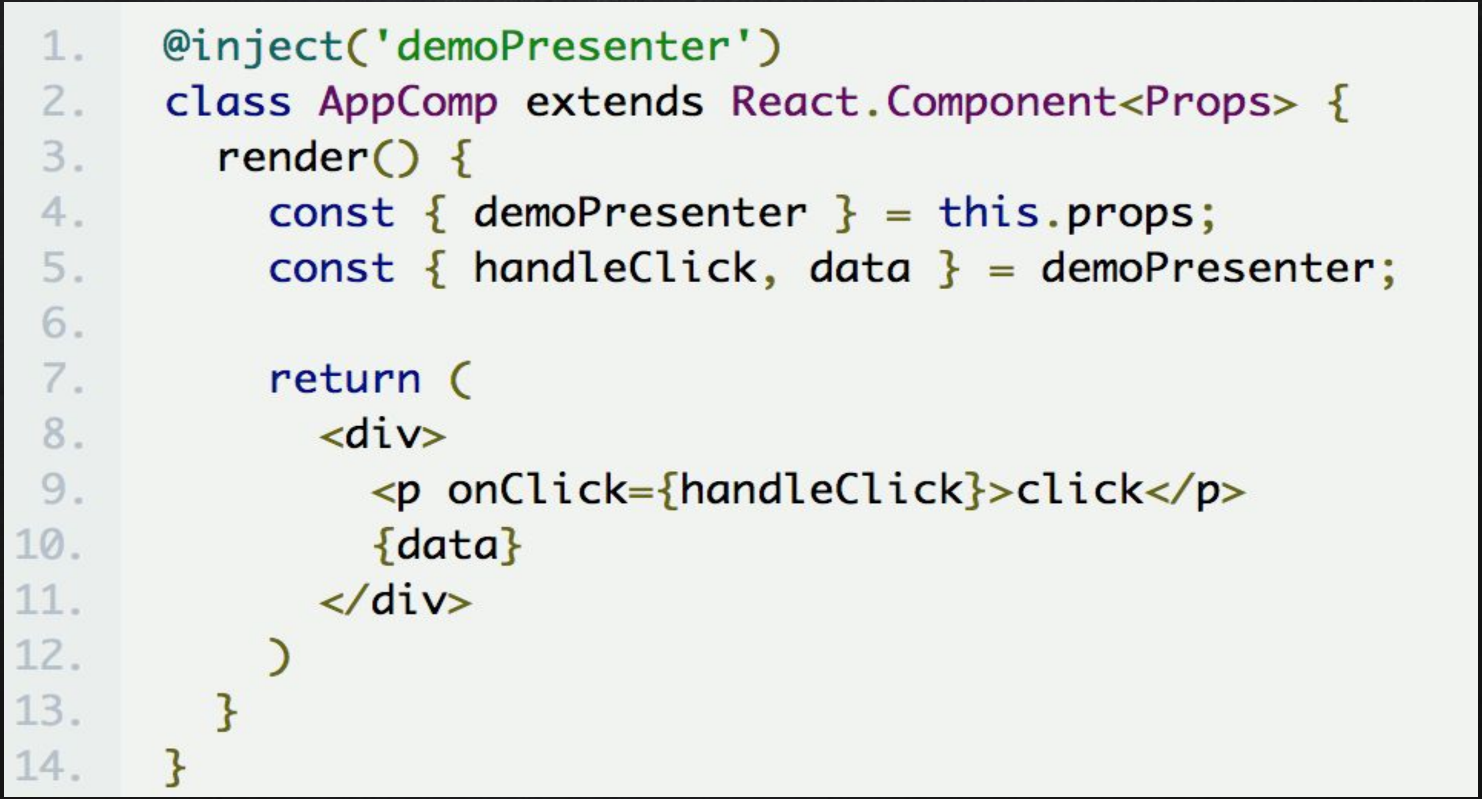
```
class Demo extends React.Component<Props> {

  constructor(props: Props) {
    super(props);
    this.state = {
      data: null,
    }
  }

  async componentDidMount() {
    const ret = http.get(`/api/xx/${this.props.id}`);
    this.setState({
      data: ret.data,
    })
  }

  handleClick = () => {
    // do something...
  }

  // other methods...

  render() {

    return (
      <div>
        <p onClick={this.handleClick}>click</p>
        {this.state.data}
      </div>
    )
  }
}
```

After

```
import { DemoPresenter } from './Demo.presenter';

class Demo extends React.Component<Props> {

  constructor(props: Props) {
    super(props);
    this.presenter = new DemoPresenter(props);
  }

  async componentDidMount() {
    const { fetchData } = this.presenter;
    await fetchData();
  }

  render() {
    const { handleClick, data } = this.presenter;
    return (
      <div>
        <p onClick={handleClick}>click</p>
        {data}
      </div>
    )
  }
}
```

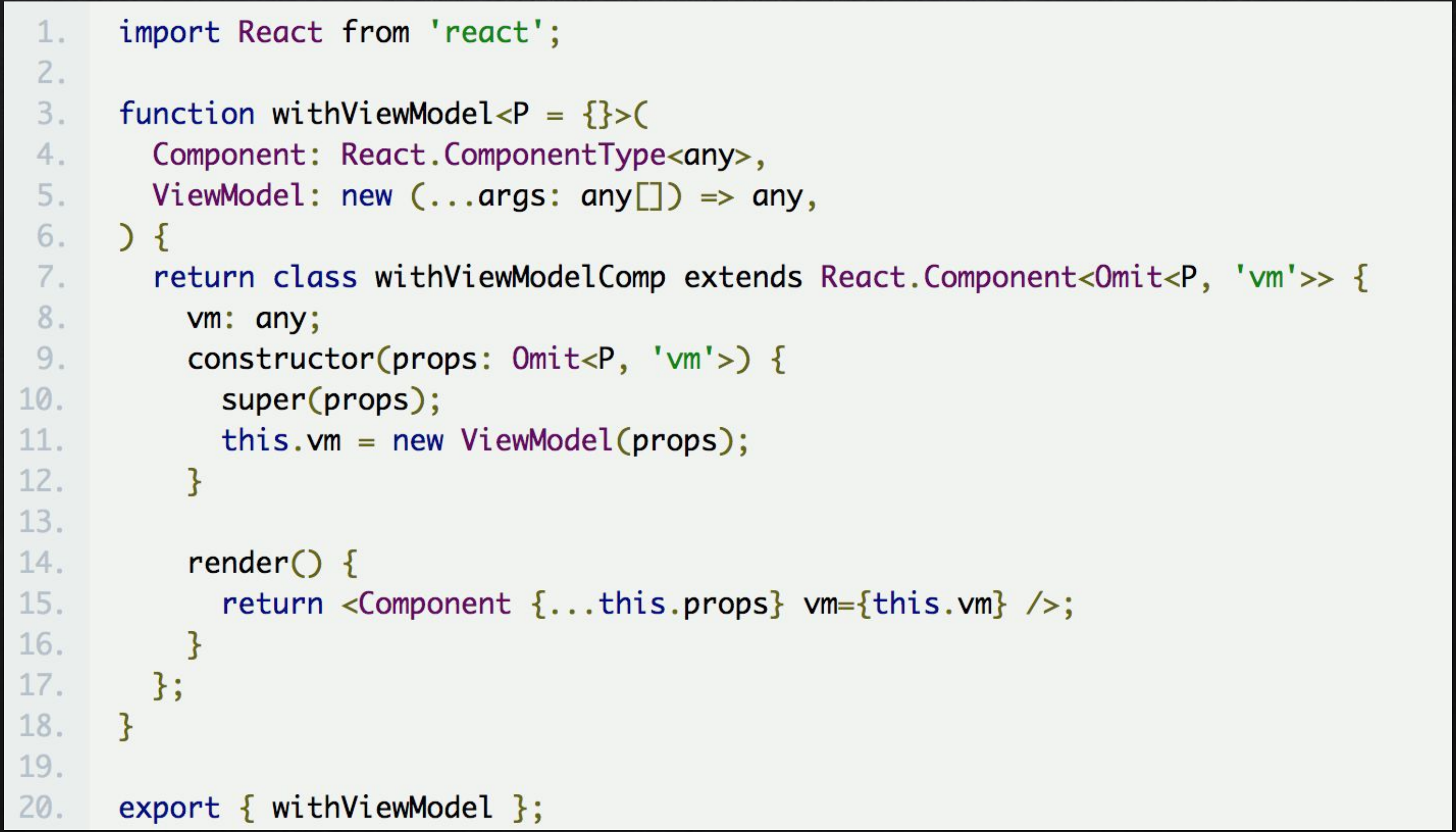
```
class DemoPresenter {

  data = {};

  constructor(props: Props) {
    this.props = props;
  }

  fetchData = async () => {
    const { id } = this.props;
    const ret = await http.get(`/api/xx/${id}`);
    this.data = ret.data;
  }

  handleClick = () => {
    // do something...
  }
}
```

从Mobx得到启发



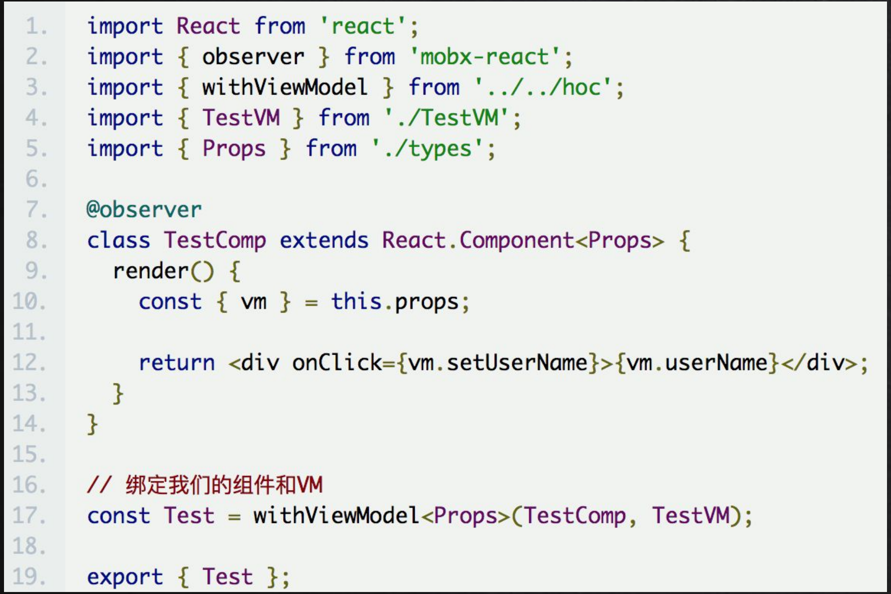
```
@inject('demoPresenter')
class AppComp extends React.Component<Props> {
  render() {
    const { demoPresenter } = this.props;
    const { handleClick, data } = demoPresenter;

    return (
      <div>
        <p onClick={handleClick}>click</p>
        {data}
      </div>
    )
  }
}
```

withViewModel

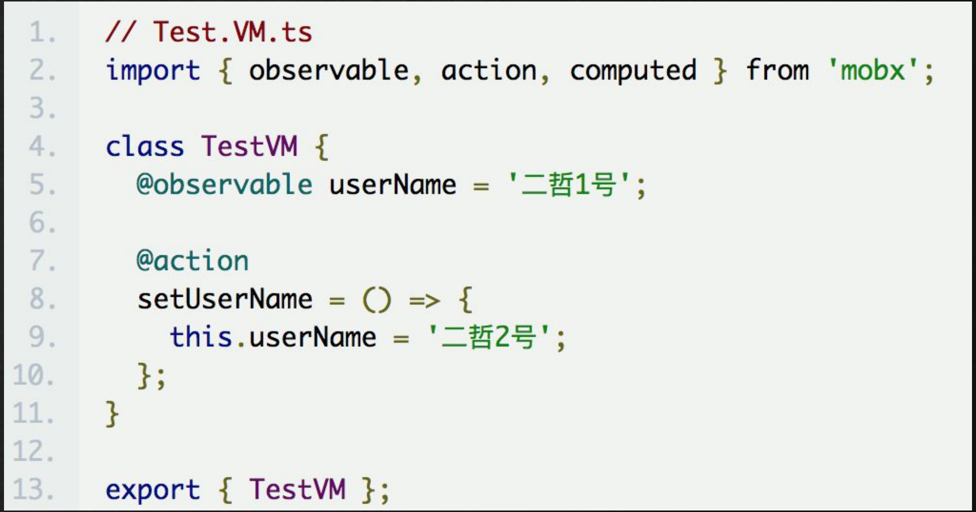


```
import React from 'react';

function withViewModel<P = {}>(
  Component: React.ComponentType<any>,
  ViewModel: new (...args: any[]) => any,
) {
  return class withViewModelComp extends React.Component<Omit<P, 'vm'>> {
    vm: any;
    constructor(props: Omit<P, 'vm'>) {
      super(props);
      this.vm = new ViewModel(props);
    }

    render() {
      return <Component {...this.props} vm={this.vm} />;
    }
  };
}

export { withViewModel };
```

How to use？

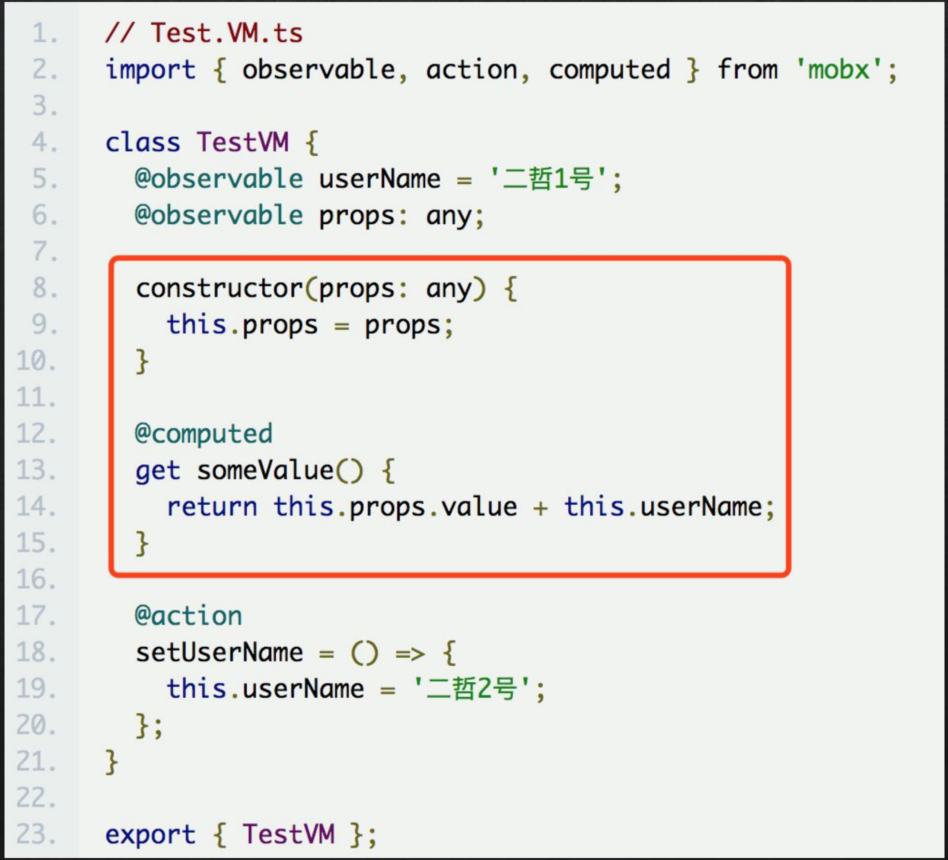


```
import React from 'react';
import { observer } from 'mobx-react';
import { withViewModel } from '../../hoc';
import { TestVM } from './TestVM';
import { Props } from './types';

@observer
class TestComp extends React.Component<Props> {
  render() {
    const { vm } = this.props;

    return <div onClick={vm.setUserName}>{vm.userName}</div>;
  }
}

// 绑定我们的组件和VM
const Test = withViewModel<Props>(TestComp, TestVM);

export { Test };
```

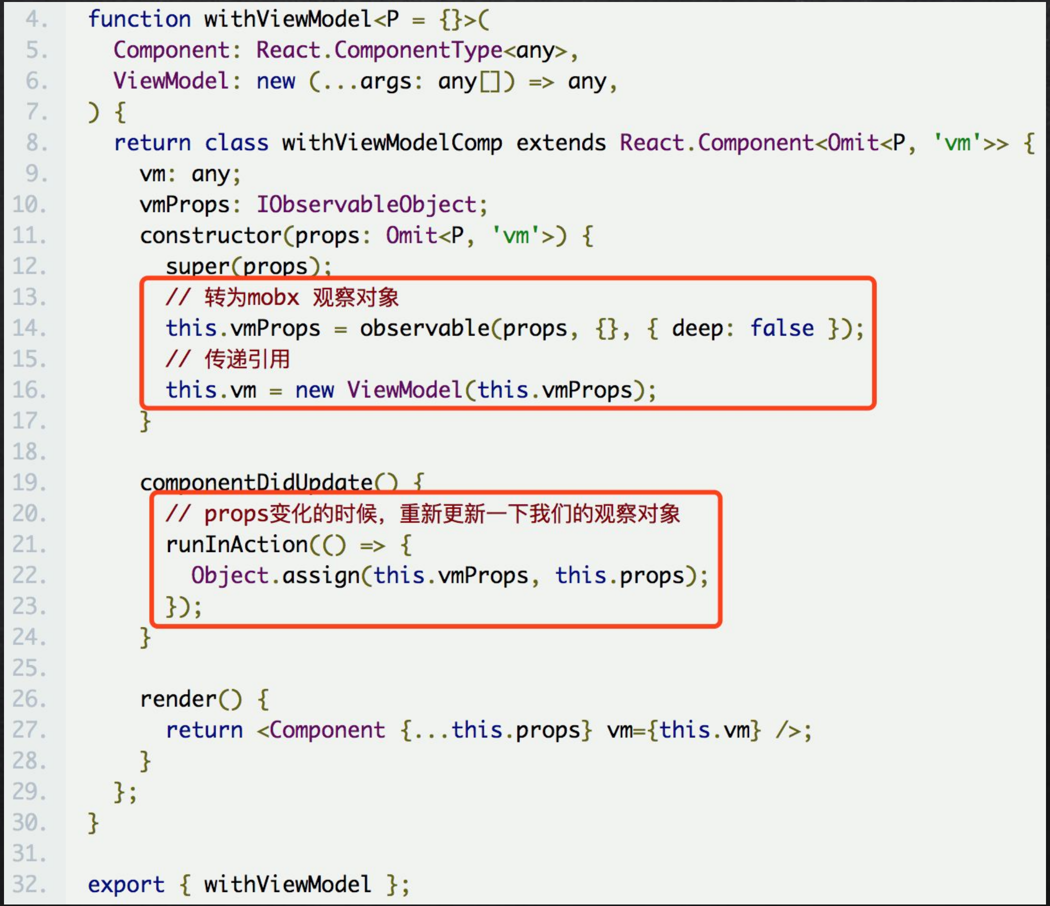


```
// Test.VM.ts
import { observable, action, computed } from 'mobx';

class TestVM {
  @observable userName = '二哲1号';

  @action
  setUserName = () => {
    this.userName = '二哲2号';
  };
}

export { TestVM };
```

Computed props 的问题



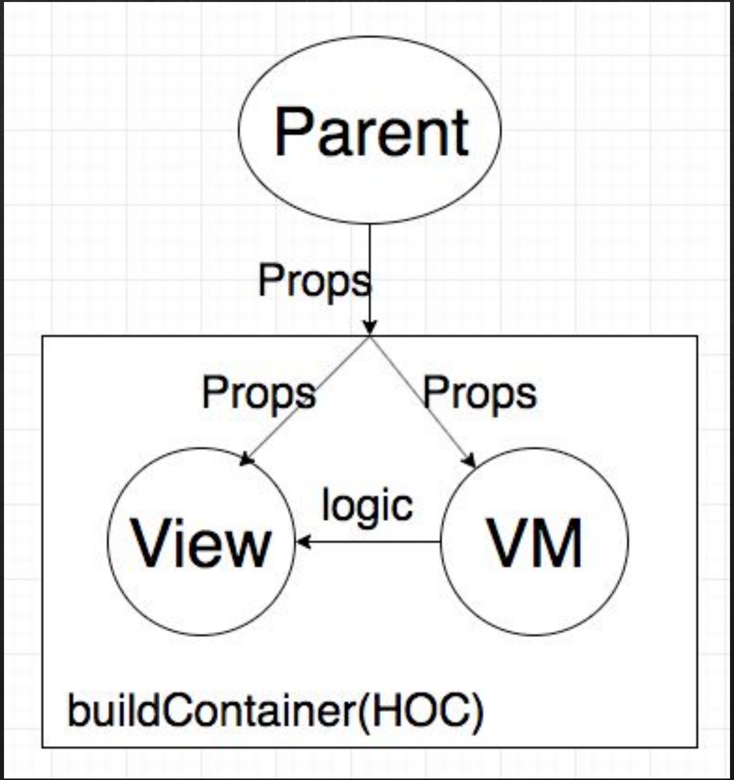
```
// Test.VM.ts
import { observable, action, computed } from 'mobx';

class TestVM {
  @observable userName = '二哲1号';
  @observable props: any;

  constructor(props: any) {
    this.props = props;
  }

  @computed
  get someValue() {
    return this.props.value + this.userName;
  }

  @action
  setUserName = () => {
    this.userName = '二哲2号';
  };
}

export { TestVM };
```

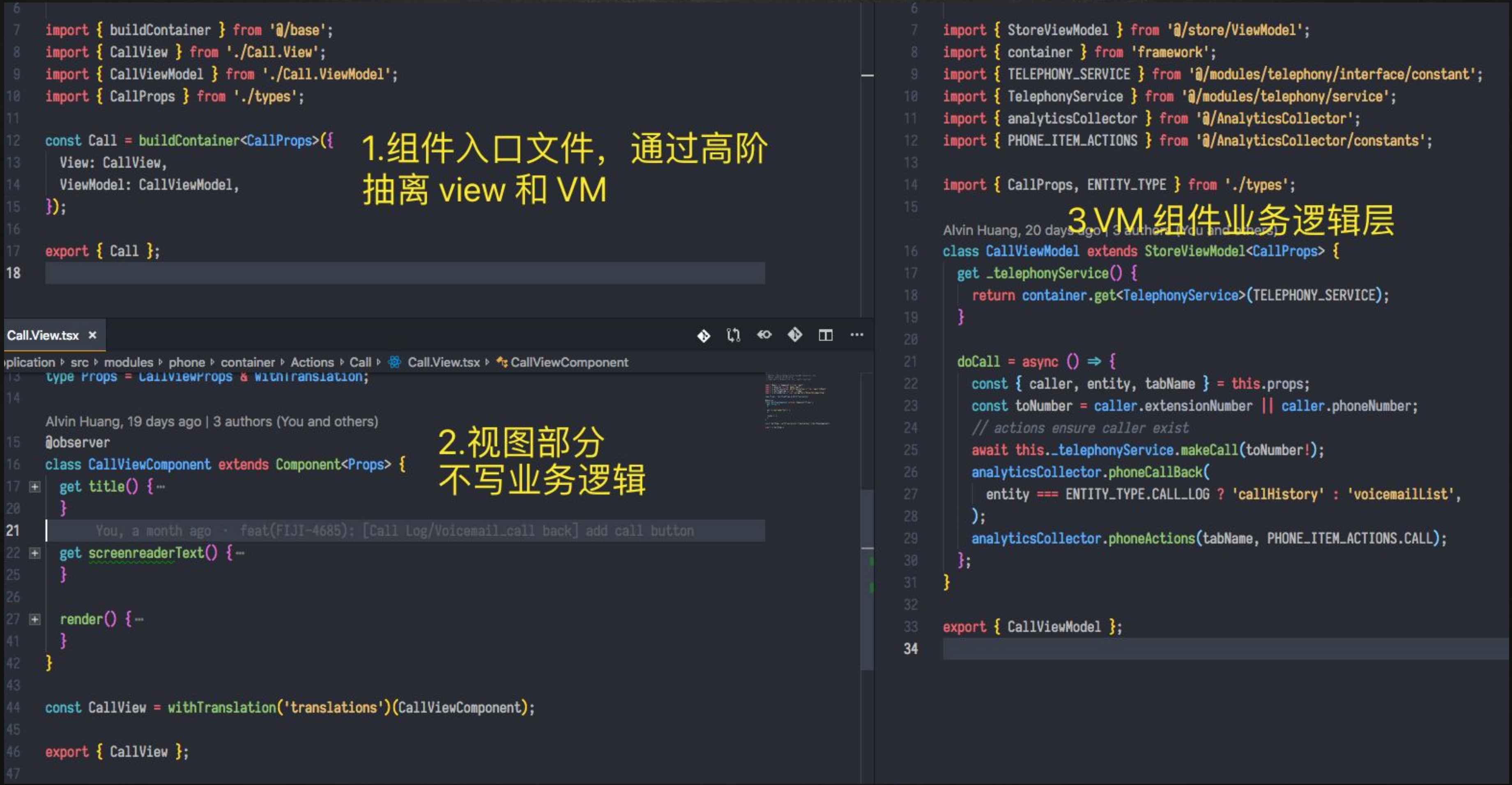


```
function withViewModel<P = {}>(
  Component: React.ComponentType<any>,
  ViewModel: new (...args: any[]) => any,
) {
  return class withViewModelComp extends React.Component<Omit<P, 'vm'>> {
    vm: any;
    vmProps: IObservableObject;
    constructor(props: Omit<P, 'vm'>) {
      super(props);
      // 转为mobx 观察对象
      this.vmProps = observable(props, {}, { deep: false });
      // 传递引用
      this.vm = new ViewModel(this.vmProps);
    }

    componentDidUpdate() {
      // props变化的时候，重新更新一下我们的观察对象
      runInAction(() => {
        Object.assign(this.vmProps, this.props);
      });
    }

    render() {
      return <Component {...this.props} vm={this.vm} />;
    }
  };
}

export { withViewModel };
```

灵感：https://github.com/mobxjs/mobx-react-lite/blob/master/src/useAsObservableSource.ts#L20-L30

Class Component





```tsx
import { buildContainer } from '@/base';
import { CallView } from './Call.View';
import { CallViewModel } from './Call.ViewModel';
import { CallProps } from './types';

const Call = buildContainer<CallProps>({
  View: CallView,
  ViewModel: CallViewModel,
});

export { Call };
```

1.组件入口文件，通过高阶抽离 view 和 VM

Call.View.tsx ×

plication › src › modules › phone › container › Actions › Call › Call.View.tsx › CallViewComponent

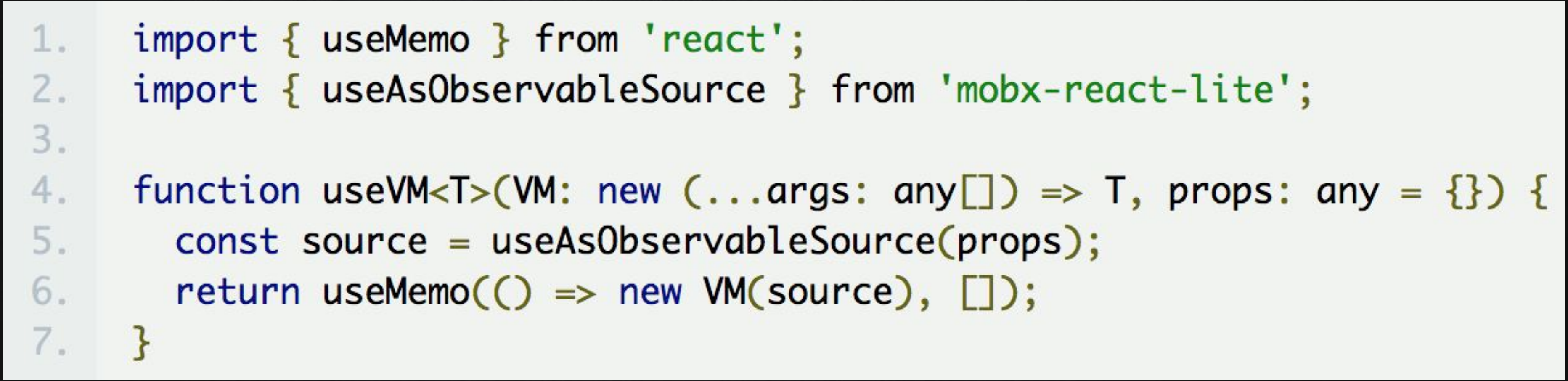
```tsx
type Props = CallViewProps & WithTranslation;

Alvin Huang, 19 days ago | 3 authors (You and others)
@observer
class CallViewComponent extends Component<Props> {
  get title() {…
  }
      You, a month ago · feat(FIJI-4685): [Call Log/Voicemail_call back] add call button
  get screenreaderText() {…
  }

  render() {…
  }
}

const CallView = withTranslation('translations')(CallViewComponent);

export { CallView };
```

2.视图部分
不写业务逻辑

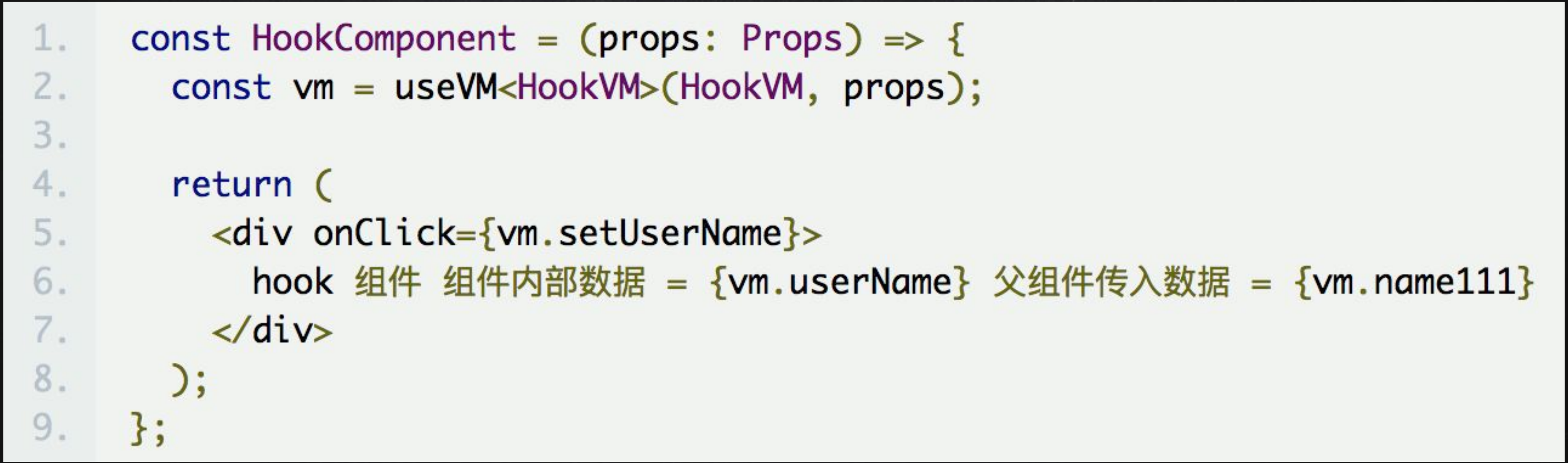
```ts
import { StoreViewModel } from '@/store/ViewModel';
import { container } from 'framework';
import { TELEPHONY_SERVICE } from '@/modules/telephony/interface/constant';
import { TelephonyService } from '@/modules/telephony/service';
import { analyticsCollector } from '@/AnalyticsCollector';
import { PHONE_ITEM_ACTIONS } from '@/AnalyticsCollector/constants';

import { CallProps, ENTITY_TYPE } from './types';

Alvin Huang, 20 days ago | 3 authors (You and others)
class CallViewModel extends StoreViewModel<CallProps> {
  get _telephonyService() {
    return container.get<TelephonyService>(TELEPHONY_SERVICE);
  }

  doCall = async () => {
    const { caller, entity, tabName } = this.props;
    const toNumber = caller.extensionNumber || caller.phoneNumber;
    // actions ensure caller exist
    await this._telephonyService.makeCall(toNumber!);
    analyticsCollector.phoneCallBack(
      entity === ENTITY_TYPE.CALL_LOG ? 'callHistory' : 'voicemailList',
    );
    analyticsCollector.phoneActions(tabName, PHONE_ITEM_ACTIONS.CALL);
  };
}

export { CallViewModel };
```

3.VM 组件业务逻辑层

Hook 版

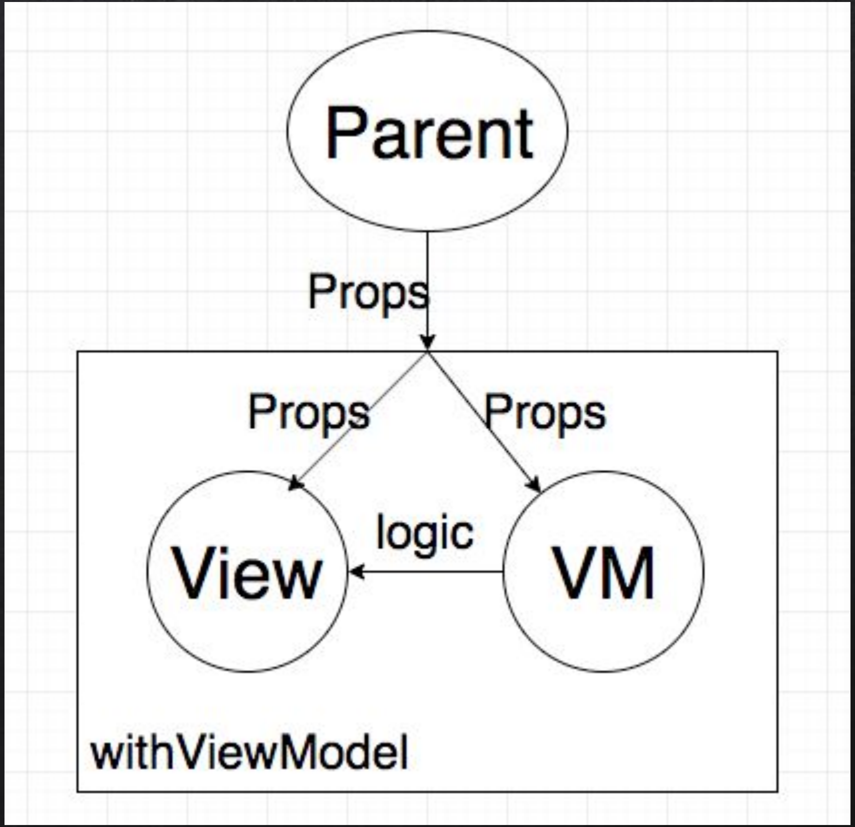


```
import { useMemo } from 'react';
import { useAsObservableSource } from 'mobx-react-lite';

function useVM<T>(VM: new (...args: any[]) => T, props: any = {}) {
  const source = useAsObservableSource(props);
  return useMemo(() => new VM(source), []);
}
```

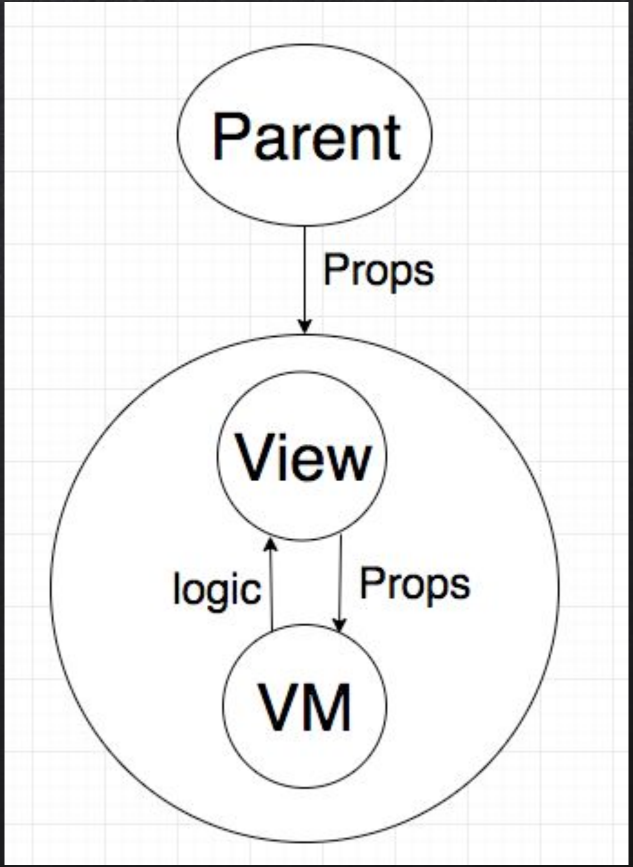


```
const HookComponent = (props: Props) => {
  const vm = useVM<HookVM>(HookVM, props);

  return (
    <div onClick={vm.setUserName}>
      hook 组件 组件内部数据 = {vm.userName} 父组件传入数据 = {vm.name111}
    </div>
  );
};
```

Class Component



Hook Component



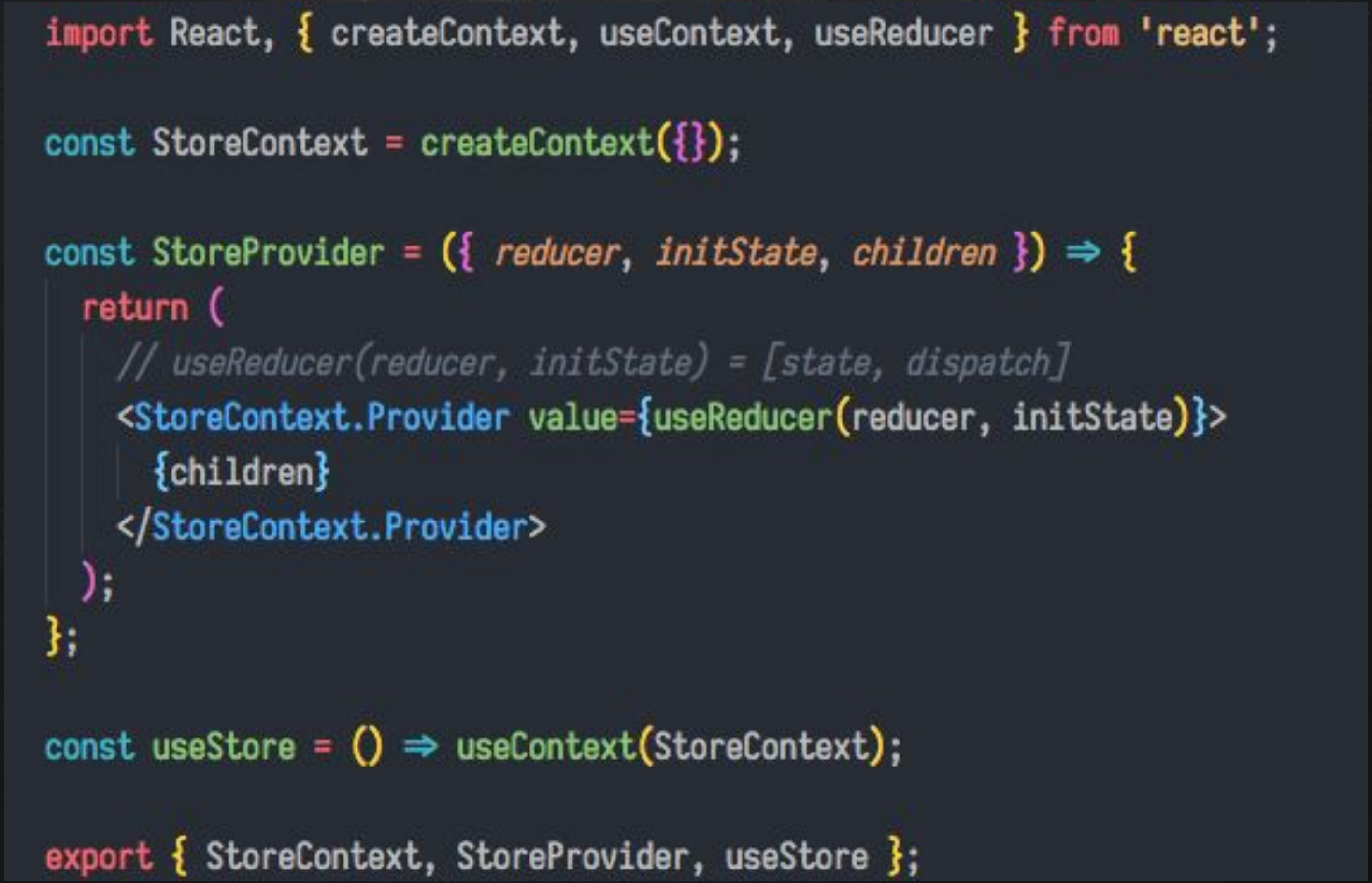
优缺点

	code	performance	test	spec	future
before	✗	✗	✓	✓	✗
after	✓	✓	✗	✗	✓

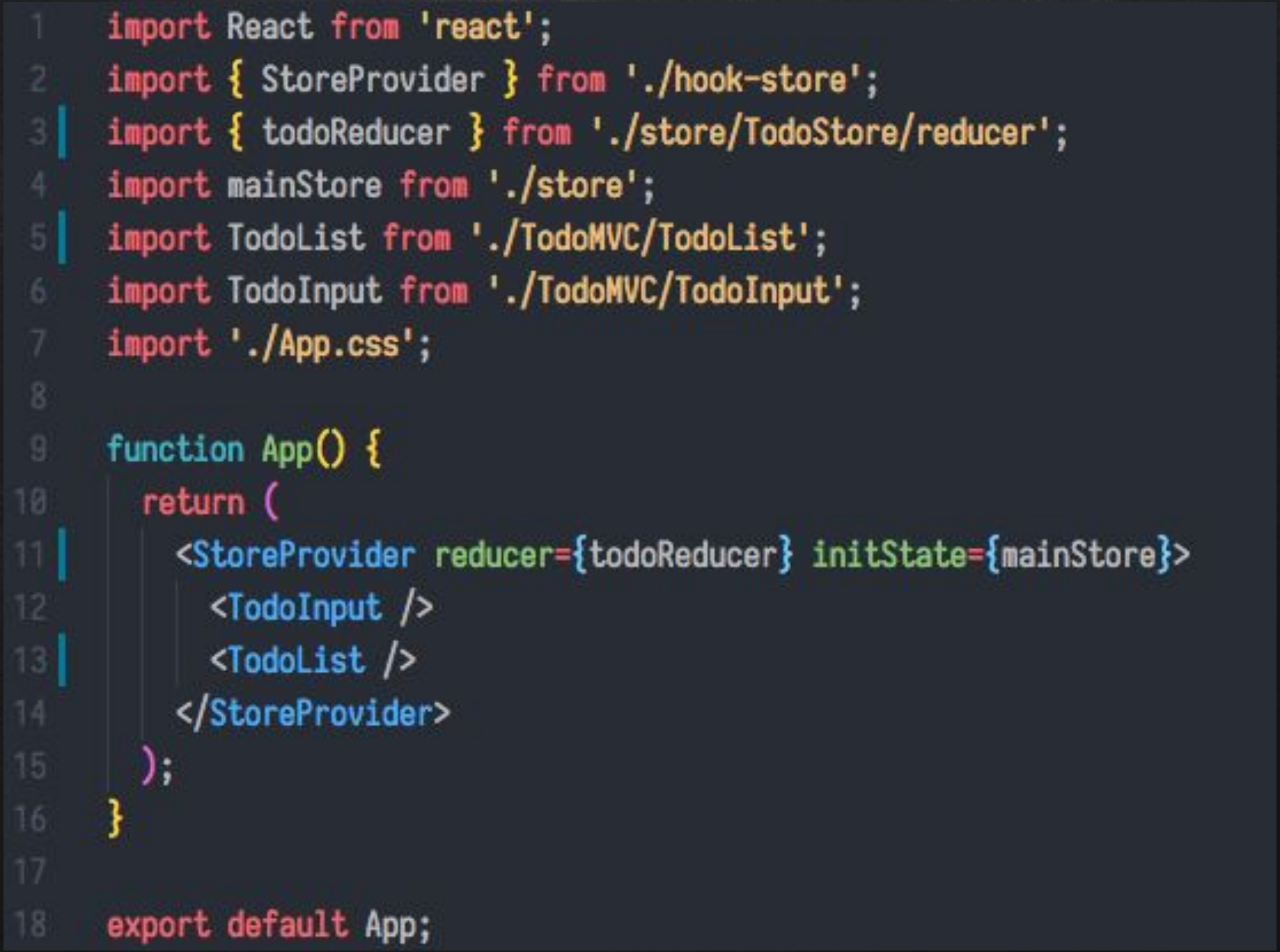
“

Hook with state management

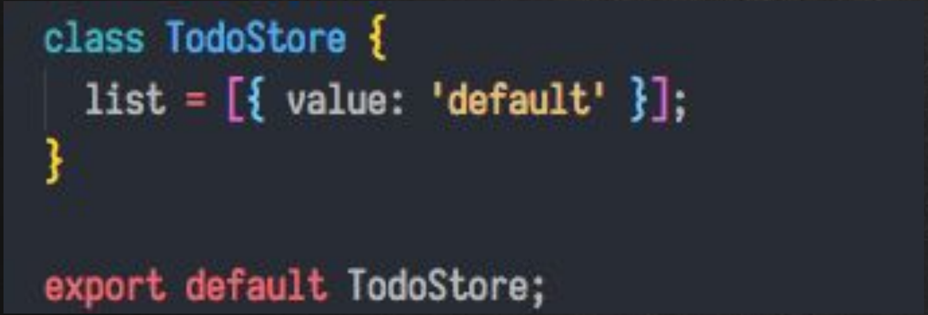
- ✘ hookStore (multi store)
- ✘ TodoStore
- ✘ TodoInput
- ✘ TodoList



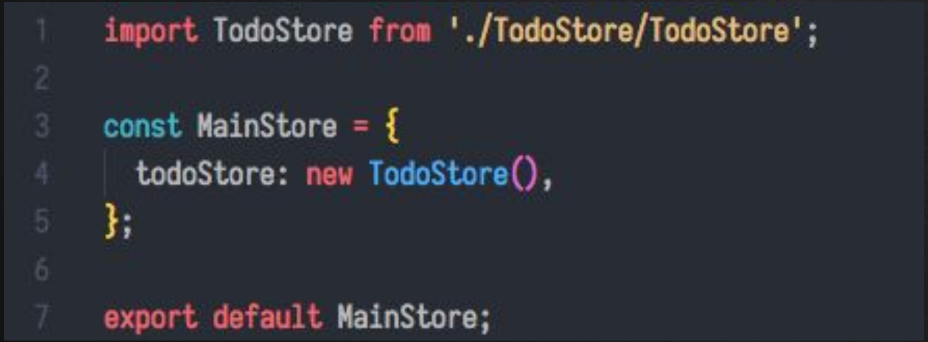
```
import React, { createContext, useContext, useReducer } from 'react';

const StoreContext = createContext({});

const StoreProvider = ({ reducer, initState, children }) => {
  return (
    // useReducer(reducer, initState) = [state, dispatch]
    <StoreContext.Provider value={useReducer(reducer, initState)}>
      {children}
    </StoreContext.Provider>
  );
};

const useStore = () => useContext(StoreContext);

export { StoreContext, StoreProvider, useStore };
```



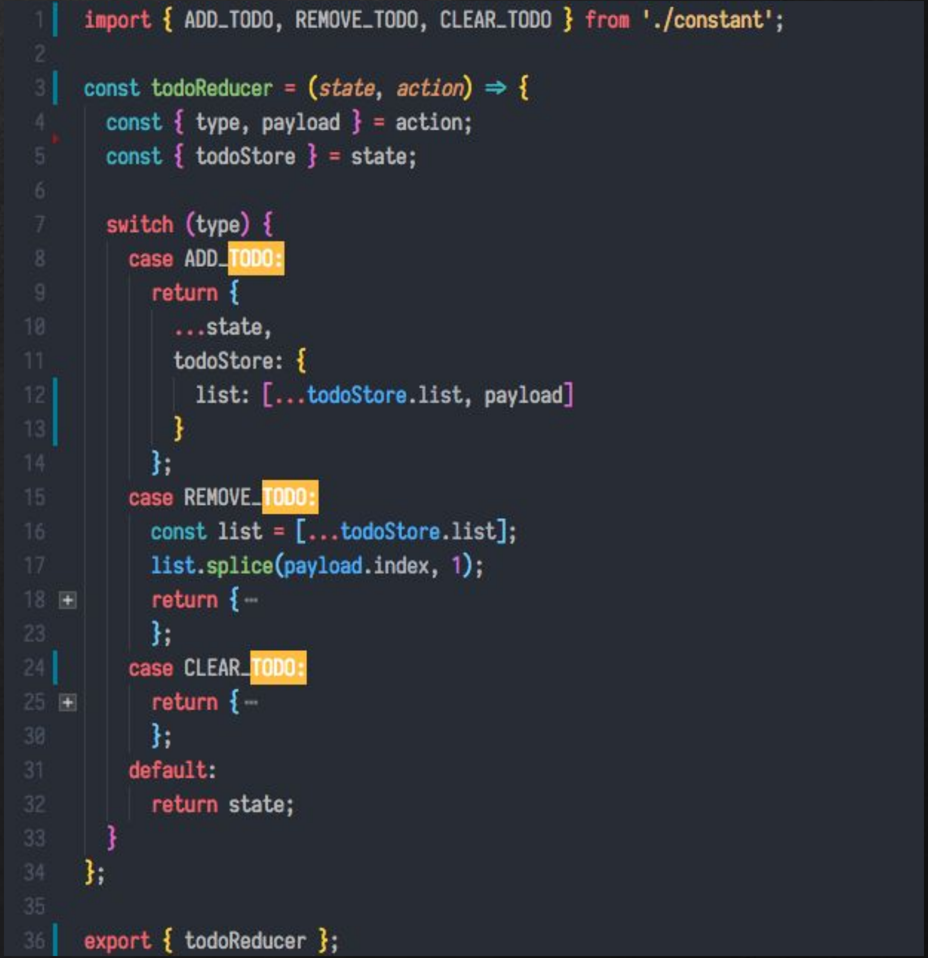
```
import React from 'react';
import { StoreProvider } from './hook-store';
import { todoReducer } from './store/TodoStore/reducer';
import mainStore from './store';
import TodoList from './TodoMVC/TodoList';
import TodoInput from './TodoMVC/TodoInput';
import './App.css';

function App() {
  return (
    <StoreProvider reducer={todoReducer} initState={mainStore}>
      <TodoInput />
      <TodoList />
    </StoreProvider>
  );
}

export default App;
```



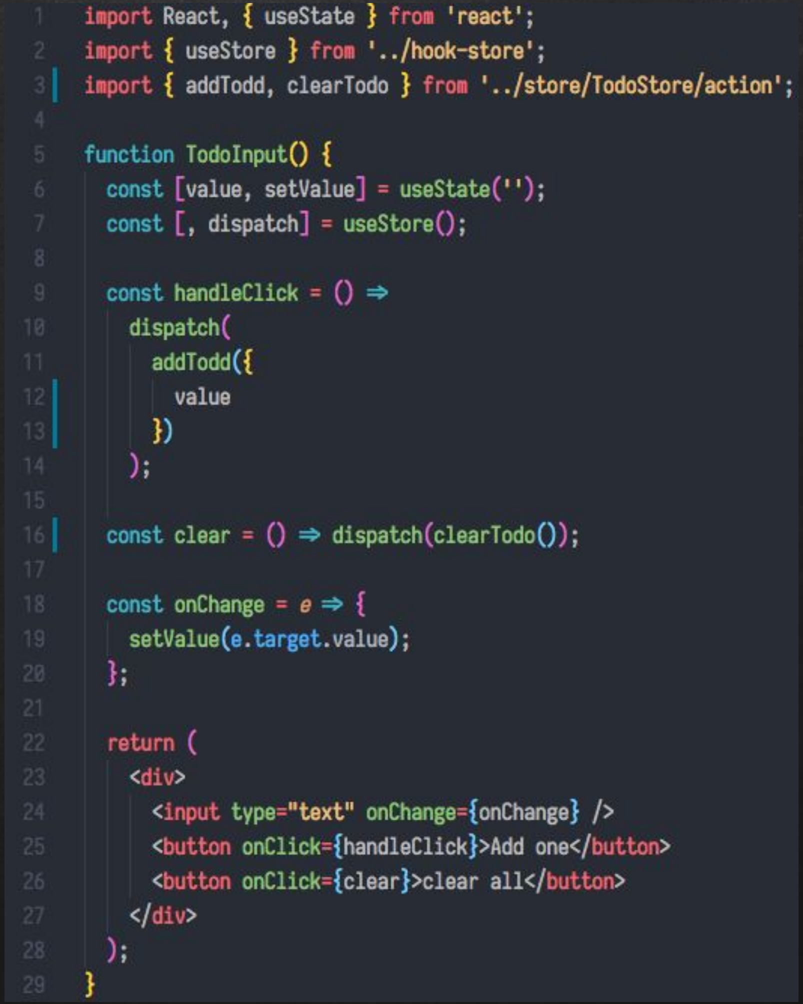
```
class TodoStore {
  list = [{ value: 'default' }];
}

export default TodoStore;
```



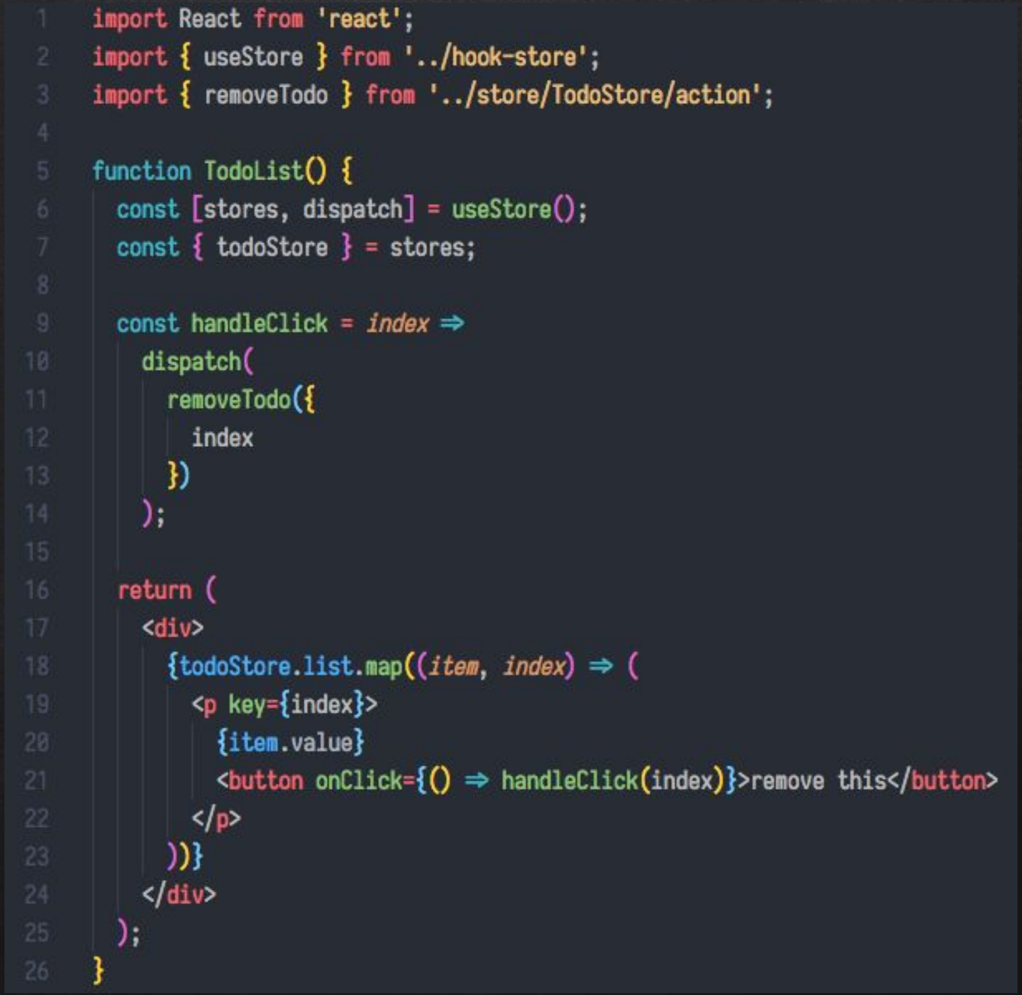
```
import TodoStore from './TodoStore/TodoStore';

const MainStore = {
  todoStore: new TodoStore(),
};

export default MainStore;
```



```
import { ADD_TODO, REMOVE_TODO, CLEAR_TODO } from './constant';

const todoReducer = (state, action) => {
  const { type, payload } = action;
  const { todoStore } = state;

  switch (type) {
    case ADD_TODO:
      return {
        ...state,
        todoStore: {
          list: [...todoStore.list, payload]
        }
      };
    case REMOVE_TODO:
      const list = [...todoStore.list];
      list.splice(payload.index, 1);
      return {⋯
      };
    case CLEAR_TODO:
      return {⋯
      };
    default:
      return state;
  }
};

export { todoReducer };
```



```
import React, { useState } from 'react';
import { useStore } from '../hook-store';
import { addTodd, clearTodo } from '../store/TodoStore/action';

function TodoInput() {
  const [value, setValue] = useState('');
  const [, dispatch] = useStore();

  const handleClick = () =>
    dispatch(
      addTodd({
        value
      })
    );

  const clear = () => dispatch(clearTodo());

  const onChange = e => {
    setValue(e.target.value);
  };

  return (
    <div>
      <input type="text" onChange={onChange} />
      <button onClick={handleClick}>Add one</button>
      <button onClick={clear}>clear all</button>
    </div>
  );
}
```



```
import React from 'react';
import { useStore } from '../hook-store';
import { removeTodo } from '../store/TodoStore/action';

function TodoList() {
  const [stores, dispatch] = useStore();
  const { todoStore } = stores;

  const handleClick = index =>
    dispatch(
      removeTodo({
        index
      })
    );

  return (
    <div>
      {todoStore.list.map((item, index) => (
        <p key={index}>
          {item.value}
          <button onClick={() => handleClick(index)}>remove this</button>
        </p>
      ))}
    </div>
  );
}
```

https://github.com/MeCKodo/React-hook-store

回顾一下

React bad parts

1. setState 异步？同步？
2. setState callback hell
3. 合成事件
4. 事件传参
5. Render
6. Mixin -> HOC & RP -> hook

React hook

1. 介绍一些基础 hook
2. Hook的第二个参数
3. useMemo
4. 我们如何迁移

State management

1. hook - store

THANKS!

Any questions?

@二哲

个人网站 :http://www.meckodo.com

Github: https://github.com/MeCKodo