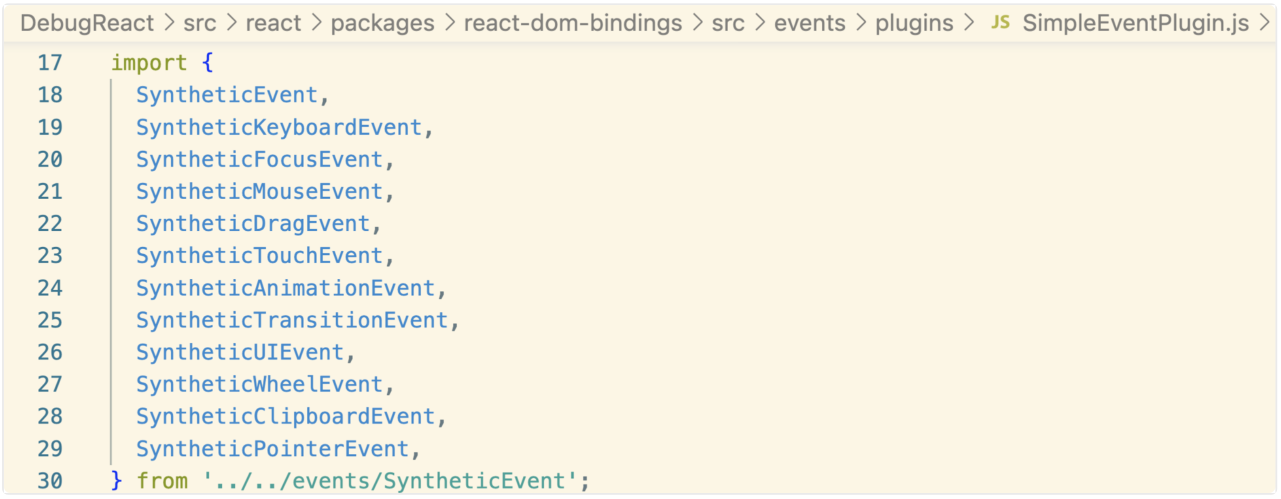
17-5 React合成事件的定义

资源

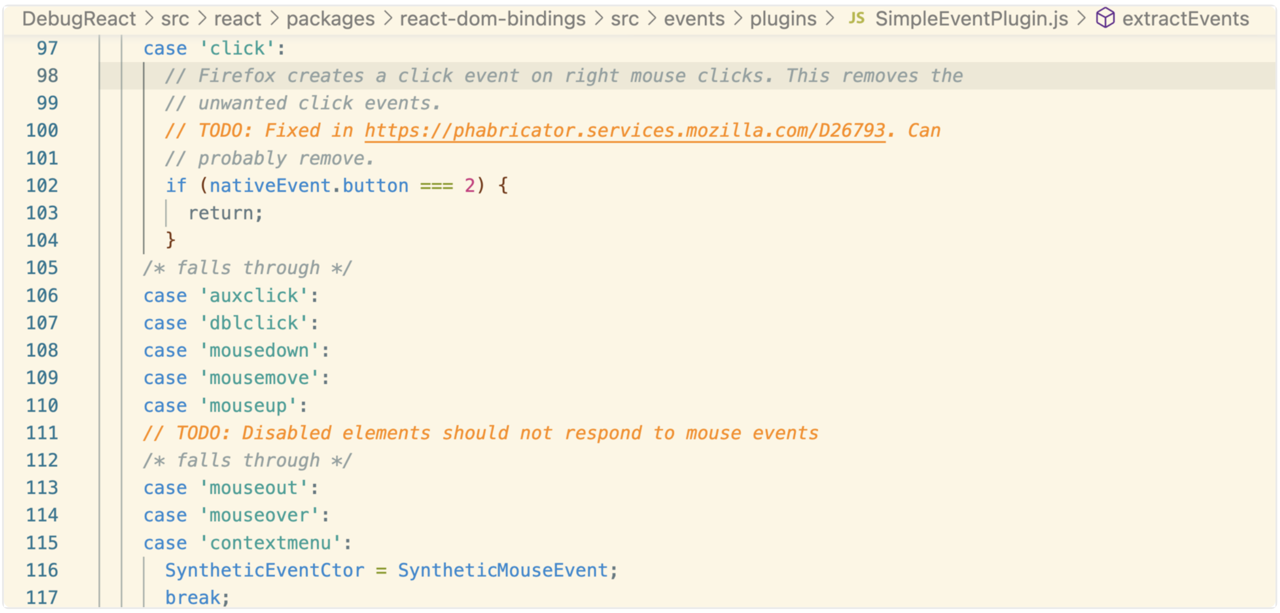
1. React16 之前的事件池
2. React17 去除事件池

这里以 click 为例：

react/packages/react-dom-bindings/src/events/plugins/SimpleEventPlugin.js


DebugReact > src > react > packages > react-dom-bindings > src > events > plugins > JS SimpleEventPlugin.js >

```js
import {
  SyntheticEvent,
  SyntheticKeyboardEvent,
  SyntheticFocusEvent,
  SyntheticMouseEvent,
  SyntheticDragEvent,
  SyntheticTouchEvent,
  SyntheticAnimationEvent,
  SyntheticTransitionEvent,
  SyntheticUIEvent,
  SyntheticWheelEvent,
  SyntheticClipboardEvent,
  SyntheticPointerEvent,
} from '../../events/SyntheticEvent';
```



DebugReact > src > react > packages > react-dom-bindings > src > events > plugins > JS SimpleEventPlugin.js > extractEvents

```
    case 'click':
      // Firefox creates a click event on right mouse clicks. This removes the
      // unwanted click events.
      // TODO: Fixed in https://phabricator.services.mozilla.com/D26793. Can
      // probably remove.
      if (nativeEvent.button === 2) {
        return;
      }
    /* falls through */
    case 'auxclick':
    case 'dblclick':
    case 'mousedown':
    case 'mousemove':
    case 'mouseup':
    // TODO: Disabled elements should not respond to mouse events
    /* falls through */
    case 'mouseout':
    case 'mouseover':
    case 'contextmenu':
      SyntheticEventCtor = SyntheticMouseEvent;
      break;
```

react/packages/react-dom-bindings/src/events/SyntheticEvent.js

JavaScript

```javascript
export const SyntheticMouseEvent: $FlowFixMe =
  createSyntheticEvent(MouseEventInterface);
```

MouseEventInterface

JavaScript

```javascript
type EventInterfaceType = {
  [propName: string]: 0 | ((event: {[propName: string]: mixed, ...}) =
};

const MouseEventInterface: EventInterfaceType = {
  ...UIEventInterface,
  screenX: 0,
  screenY: 0,
  clientX: 0,
  clientY: 0,
  pageX: 0,
  pageY: 0,
  ctrlKey: 0,
  shiftKey: 0,
  altKey: 0,
  metaKey: 0,
  getModifierState: getEventModifierState,
  button: 0,
  buttons: 0,
  relatedTarget: function (event) {
    if (event.relatedTarget === undefined)
      return event.fromElement === event.srcElement
        ? event.toElement
        : event.fromElement;

    return event.relatedTarget;
  },
  movementX: function (event) {
    if ('movementX' in event) {
      return event.movementX;
    }
    updateMouseMovementPolyfillState(event);
    return lastMovementX;
  },
  movementY: function (event) {
    if ('movementY' in event) {
      return event.movementY;
    }
    // 这里不需要调用updateMouseMovementPolyfillState()
    // 因为可以确保在复制movementX时已经运行过了。
    return lastMovementY;
  },
};
```

创建 SyntheticEvent：createSyntheticEvent

```JavaScript
function functionThatReturnsTrue() {
  return true;
}

function functionThatReturnsFalse() {
  return false;
}

// 这是一个工厂函数，故返回不同的构造函数，如果只有一个构造函数，它将是多态的，引擎
function createSyntheticEvent(Interface: EventInterfaceType) {
  function SyntheticBaseEvent(
    reactName: string | null,
    reactEventType: string,
    targetInst: Fiber | null,
    nativeEvent: {[propName: string]: mixed, ...},
    nativeEventTarget: null | EventTarget,
  ) {
    this._reactName = reactName;
    this._targetInst = targetInst;
    this.type = reactEventType;
    this.nativeEvent = nativeEvent;
    this.target = nativeEventTarget;
    this.currentTarget = null;

    for (const propName in Interface) {
      if (!Interface.hasOwnProperty(propName)) {
        continue;
      }
      const normalize = Interface[propName];
      if (normalize) {
        this[propName] = normalize(nativeEvent);
      } else {
        this[propName] = nativeEvent[propName];
      }
    }

    const defaultPrevented =
      nativeEvent.defaultPrevented != null
        ? nativeEvent.defaultPrevented
        : nativeEvent.returnValue === false;
    if (defaultPrevented) {
      this.isDefaultPrevented = functionThatReturnsTrue;
    } else {
      this.isDefaultPrevented = functionThatReturnsFalse;
    }
    this.isPropagationStopped = functionThatReturnsFalse;
    return this;
  }

  assign(SyntheticBaseEvent.prototype, {
    preventDefault: function () {
      this.defaultPrevented = true;
      const event = this.nativeEvent;
      if (!event) {
        return;
      }

      if (event.preventDefault) {
        event.preventDefault();
      } else if (typeof event.returnValue !== 'unknown') {
        event.returnValue = false;
      }
      this.isDefaultPrevented = functionThatReturnsTrue;
    },
    stopPropagation: function () {
      const event = this.nativeEvent;
      if (!event) {
        return;
      }

      if (event.stopPropagation) {
        event.stopPropagation();
      } else if (typeof event.cancelBubble !== 'unknown') {
        // ChangeEventPlugin 为 IE 注册了一个 "propertychange" 事件。
        // 该事件不支持冒泡或取消，并且任何对 cancelBubble 的引用都会抛出 "Mem
        // 对 "unknown" 的 typeof 检查可以规避这个问题（也是针对 IE 的）。
        event.cancelBubble = true;
      }

      this.isPropagationStopped = functionThatReturnsTrue;
    },

    // 在每个事件循环后，释放所有已派发的SyntheticEvent，将它们添加回对象池中。
    persist: function () {
      // React17之后，不再支持事件放入事件池。这里只是保留了个函数壳子
    },

    /**
     * 检查此事件是否应该释放回对象池。
     *  如果不应释放，则为true，否则为false。
     */
    isPersistent: functionThatReturnsTrue,
  });
  return SyntheticBaseEvent;
}
```