18-4 实现合成事件

这里以 click 为例：

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

TypeScript

```
function extractEvents(
  dispatchQueue: DispatchQueue,
  domEventName: DOMEventName,
  targetInst: null | Fiber,
  nativeEvent: AnyNativeEvent,
  nativeEventTarget: null | EventTarget,
  eventSystemFlags: EventSystemFlags,
  targetContainer: EventTarget
): void {
  // click->onClick
  const reactName = topLevelEventsToReactNames.get(domEventName);
  if (reactName === undefined) {
    return;
  }

  let SyntheticEventCtor = SyntheticEvent;
  switch (domEventName) {
    case "click":
      // Firefox creates a click event on right mouse clicks. This rem
```

```
      // unwanted click events.
      // TODO: Fixed in https://phabricator.services.mozilla.com/D2679
      // 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;
  }

  const inCapturePhase = (eventSystemFlags & IS_CAPTURE_PHASE) !== 0;
  // 如果是 scroll 事件，或者是 scrollend 事件，那么只会在冒泡阶段触发
  const accumulateTargetOnly =
    !inCapturePhase &&
    (domEventName === "scroll" || domEventName === "scrollend");

  const listeners = accumulateSinglePhaseListeners(
    targetInst,
    reactName,
    nativeEvent.type,
    inCapturePhase,
    accumulateTargetOnly,
    nativeEvent
  );

  if (listeners.length > 0) {
    const event: ReactSyntheticEvent = new SyntheticEventCtor(
      reactName,
      domEventName,
      null,
```

```
      nativeEvent,
      nativeEventTarget
    );

    dispatchQueue.push({ event, listeners });
  }
}
```

processDispatchQueueItemsInOrder

packages/react-dom-bindings/src/events/ReactDOMEventListener.ts

```TypeScript
function processDispatchQueueItemsInOrder(
  event: ReactSyntheticEvent,
  dispatchListeners: Array<DispatchListener>,
  inCapturePhase: boolean
): void {
  let previousInstance;
  if (inCapturePhase) {
    for (let i = dispatchListeners.length - 1; i >= 0; i--) {
      const { instance, currentTarget, listener } = dispatchListeners[
      if (instance !== previousInstance && event.isPropagationStopped(
        return;
      }
      executeDispatch(event, listener, currentTarget);
      previousInstance = instance;
    }
  } else {
    for (let i = 0; i < dispatchListeners.length; i++) {
      const { instance, currentTarget, listener } = dispatchListeners[
      if (instance !== previousInstance && event.isPropagationStopped(
        return;
      }
      executeDispatch(event, listener, currentTarget);
      previousInstance = instance;
    }
```

```
  }
}
```

合成事件类型定义

packages/react-dom-bindings/src/events/ReactSyntheticEventType.ts

```TypeScript
import type { Fiber } from "react-reconciler/src/ReactInternalTypes";

type BaseSyntheticEvent = {
  isPersistent: () => boolean;
  isPropagationStopped: () => boolean;
  _targetInst: Fiber;
  nativeEvent: Event;
  target?: any;
  relatedTarget?: any;
  type: string;
  currentTarget: null | EventTarget;
};

export type KnownReactSyntheticEvent = BaseSyntheticEvent & {
  _reactName: string;
};
export type UnknownReactSyntheticEvent = BaseSyntheticEvent & {
  _reactName: null;
};

export type ReactSyntheticEvent =
  | KnownReactSyntheticEvent
  | UnknownReactSyntheticEvent;
```

合成事件定义

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

```JavaScript
```

```
import { Fiber } from "react-reconciler/src/ReactInternalTypes";

type EventInterfaceType = {
  [propName: string]: 0 | ((event: { [propName: string]: any }) => any
};

function functionThatReturnsTrue() {
  return true;
}

function functionThatReturnsFalse() {
  return false;
}

function createSyntheticEvent(Interface: EventInterfaceType) {
  function SyntheticBaseEvent(
    reactName: string | null,
    reactEventType: string,
    targetInst: Fiber | null,
    nativeEvent: { [propName: string]: any },
    nativeEventTarget: null | EventTarget
  ): void {
    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;
  }

  // $FlowFixMe[prop-missing] found when upgrading Flow
  Object.assign(SyntheticBaseEvent.prototype, {
    // $FlowFixMe[missing-this-annot]
    preventDefault: function () {
      this.defaultPrevented = true;
      const event = this.nativeEvent;
      if (!event) {
        return;
      }

      if (event.preventDefault) {
        event.preventDefault();
        // $FlowFixMe[illegal-typeof] - flow is not aware of `unknown`
      } else if (typeof event.returnValue !== "unknown") {
        event.returnValue = false;
      }
      this.isDefaultPrevented = functionThatReturnsTrue;
    },

    // $FlowFixMe[missing-this-annot]
    stopPropagation: function () {
      const event = this.nativeEvent;
      if (!event) {
        return;
      }

      if (event.stopPropagation) {
        event.stopPropagation();
```

```
        // $FlowFixMe[illegal-typeof] - flow is not aware of `unknown`
      } else if (typeof event.cancelBubble !== "unknown") {
        // The ChangeEventPlugin registers a "propertychange" event fo
        // IE. This event does not support bubbling or cancelling, and
        // any references to cancelBubble throw "Member not found".  A
        // typeof check of "unknown" circumvents this issue (and is al
        // IE specific).
        event.cancelBubble = true;
      }

      this.isPropagationStopped = functionThatReturnsTrue;
    },
  });
  return SyntheticBaseEvent;
}

const modifierKeyToProp = {
  Alt: "altKey",
  Control: "ctrlKey",
  Meta: "metaKey",
  Shift: "shiftKey",
};

function modifierStateGetter(keyArg) {
  const syntheticEvent = this;
  const nativeEvent = syntheticEvent.nativeEvent;
  if (nativeEvent.getModifierState) {
    return nativeEvent.getModifierState(keyArg);
  }
  const keyProp = modifierKeyToProp[keyArg];
  return keyProp ? !!nativeEvent[keyProp] : false;
}

function getEventModifierState(nativeEvent: { [propName: string]: any
  return modifierStateGetter;
}

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

```
const EventInterface = {
  eventPhase: 0,
  bubbles: 0,
  cancelable: 0,
  timeStamp: function (event: { [propName: string]: any }) {
    return event.timeStamp || Date.now();
  },
  defaultPrevented: 0,
  isTrusted: 0,
};
export const SyntheticEvent = createSyntheticEvent(
  EventInterface as EventInterfaceType
);

const UIEventInterface = {
  ...EventInterface,
  view: 0,
  detail: 0,
};

let lastMovementX;
let lastMovementY;
let lastMouseEvent;

function updateMouseMovementPolyfillState(event: { [propName: string]:
  if (event !== lastMouseEvent) {
    if (lastMouseEvent && event.type === "mousemove") {
      // $FlowFixMe[unsafe-arithmetic] assuming this is a number
      lastMovementX = event.screenX - lastMouseEvent.screenX;
      // $FlowFixMe[unsafe-arithmetic] assuming this is a number
      lastMovementY = event.screenY - lastMouseEvent.screenY;
    } else {
      lastMovementX = 0;
      lastMovementY = 0;
    }
    lastMouseEvent = event;
  }
}

/**
 * @interface MouseEvent
```

```
 * @see http://www.w3.org/TR/DOM-Level-3-Events/
 */
const MouseEventInterface = {
  ...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;
    }
    // Don't need to call updateMouseMovementPolyfillState() here
    // because it's guaranteed to have already run when movementX
    // was copied.
    return lastMovementY;
  },
```

```
};

export const SyntheticMouseEvent = createSyntheticEvent(
  MouseEventInterface as EventInterfaceType
);
```

代码 diff

Changes 5 | History

5 changed files

- packages/react-dom-bindi.../DOMPluginEventSystem.ts
- packages/react-dom-bindings/s.../SimpleEventPlugin.ts
- packages/react-dom-bindi.../ReactDOMEventListener.ts
- packages/react-dom-bin.../ReactSyntheticEventType.ts
- packages/react-dom-bindings/src/e.../SyntheticEvent.ts

packages/react-dom-bindings/src/events/DOMPluginEventSystem.ts

```diff
@@ -11,6 +11,7 @@ import {
 import { Fiber } from "react-reconciler/src/ReactInternalTypes";
 import { HostComponent } from "react-reconciler/src/ReactWorkTags";
 import getListener from "./getListener";
+import { ReactSyntheticEvent } from "./ReactSyntheticEventType";

 export type AnyNativeEvent = Event | KeyboardEvent | MouseEvent | TouchEvent;

@@ -21,7 +22,7 @@ export type DispatchListener = {
 };

 type DispatchEntry = {
-  event: AnyNativeEvent; // todo 实现合成事件
+  event: ReactSyntheticEvent;
   listeners: Array<DispatchListener>;
 };
```

Changes 5 | History

5 changed files

- packages/react-dom-bindi.../DOMPluginEventSystem.ts
- packages/react-dom-bindings/s.../SimpleEventPlugin.ts
- packages/react-dom-bindi.../ReactDOMEventListener.ts
- packages/react-dom-bin.../ReactSyntheticEventType.ts
- packages/react-dom-bindings/src/e.../SyntheticEvent.ts

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

```diff
+ let SyntheticEventCtor = SyntheticEvent;
+ switch (domEventName) {
+   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;
+ }
+
  const inCapturePhase = (eventSystemFlags & IS_CAPTURE_PHASE) !== 0;
  // 如果是 scroll 事件，或者是 scrollend 事件，那么只会在冒泡阶段触发
  const accumulateTargetOnly =
@@ -43,7 +70,15 @@ function extractEvents(
  );

  if (listeners.length > 0) {
-   dispatchQueue.push({ event: nativeEvent, listeners });
+   const event: ReactSyntheticEvent = new SyntheticEventCtor(
+     reactName,
+     domEventName,
+     null,
+     nativeEvent,
+     nativeEventTarget
+   );
+
+   dispatchQueue.push({ event, listeners });
```

Summary (required)

Description

Commit to **lesson18-4**

Changes 5 | History

5 changed files

- packages/react-dom-bindi.../DOMPluginEventSystem.ts
- packages/react-dom-bindings/s.../SimpleEventPlugin.ts
- packages/react-dom-bindi.../ReactDOMEventListener.ts
- packages/react-dom-bin.../ReactSyntheticEventType.ts
- packages/react-dom-bindings/src/e.../SyntheticEvent.ts

Summary (required)

Description

Commit to **lesson18-4**

packages/react-dom-bindings/src/events/ReactDOMEventListener.ts

```diff
export function createEventListenerWrapperWithPriority(
  targetContainer: EventTarget,
@@ -128,26 +129,34 @@ export function processDispatchQueue(
}

function processDispatchQueueItemsInOrder(
-  event: Event,
+  event: ReactSyntheticEvent,
  dispatchListeners: Array<DispatchListener>,
  inCapturePhase: boolean
): void {
+  let previousInstance;
  if (inCapturePhase) {
-    // 捕获阶段，从上往下执行
    for (let i = dispatchListeners.length - 1; i >= 0; i--) {
      const { instance, currentTarget, listener } = dispatchListeners[i];
+      if (instance !== previousInstance && event.isPropagationStopped()) {
+        return;
+      }
      executeDispatch(event, listener, currentTarget);
+      previousInstance = instance;
    }
  } else {
    for (let i = 0; i < dispatchListeners.length; i++) {
      const { instance, currentTarget, listener } = dispatchListeners[i];
+      if (instance !== previousInstance && event.isPropagationStopped()) {
+        return;
+      }
      executeDispatch(event, listener, currentTarget);
+      previousInstance = instance;
    }
  }
}

function executeDispatch(
-  event: Event,
+  event: ReactSyntheticEvent,
  listener: Function,
  currentTarget: EventTarget
): void {
```