18-2 实现事件绑定与事件委托

在 React 初始化渲染的时候，会调用函数 `listenToAllSupportedEvents` 来绑定事件。

packages/react-dom/src/client/ReactDOMRoot.ts

```TypeScript
export function createRoot(container: Container): RootType {
  const root: FiberRoot = createFiberRoot(container);
  listenToAllSupportedEvents(container);
  return new ReactDOMRoot(root);
}
```

listenToAllSupportedEvents

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

```JavaScript
const listeningMarker = "_reactListening" + Math.random().toString(36)

export function listenToAllSupportedEvents(rootContainerElement: Event
  if (!(rootContainerElement as any)[listeningMarker]) {
    // sy 防止重复绑定
    (rootContainerElement as any)[listeningMarker] = true;
    allNativeEvents.forEach((domEventName) => {
      // 单独处理selectionchange事件，因为它不会冒泡，需要在文档上处理。
      if (domEventName !== "selectionchange") {
        if (!nonDelegatedEvents.has(domEventName)) {
          // ! 这些事件都是委托在rootContainerElement上的
          // nonDelegatedEvents中都是不需要委托的事件，如cancel、close、inv
          listenToNativeEvent(domEventName, false, rootContainerElemen
        }
        listenToNativeEvent(domEventName, true, rootContainerElement);
      }
    });
  }
}
```

listenToNativeEvent

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

```JavaScript
export function listenToNativeEvent(
  domEventName: DOMEventName,
  isCapturePhaseListener: boolean,
  target: EventTarget
): void {
  let eventSystemFlags = 0;
  if (isCapturePhaseListener) {
    eventSystemFlags |= IS_CAPTURE_PHASE;
  }
  addTrappedEventListener(
    target,
    domEventName,
    eventSystemFlags,
    isCapturePhaseListener
  );
}
```

addTrappedEventListener

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

```JavaScript
function addTrappedEventListener(
  targetContainer: EventTarget,
  domEventName: DOMEventName,
  eventSystemFlags: EventSystemFlags,
  isCapturePhaseListener: boolean
) {
  // 获取对应事件，事件定义在ReactDOMEventListener.js中
  // 如DiscreteEventPriority对应dispatchDiscreteEvent，ContinuousEventPr
  let listener = createEventListenerWrapperWithPriority(
    targetContainer,
    domEventName,
    eventSystemFlags
  );

  if (isCapturePhaseListener) {
    // ! 捕获阶段
    addEventCaptureListener(targetContainer, domEventName, listener);
  } else {
    addEventBubbleListener(targetContainer, domEventName, listener);
  }
}
```

createEventListenerWrapperWithPriority

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

```JavaScript
export function createEventListenerWrapperWithPriority(
  targetContainer: EventTarget,
  domEventName: DOMEventName,
  eventSystemFlags: EventSystemFlags,
): Function {
  // 根据事件名称，获取优先级。比如click、input、drop等对应DiscreteEventPrior
  // message也许处于Scheduler中，根据getCurrentSchedulerPriorityLevel()获
  const eventPriority = getEventPriority(domEventName);
  let listenerWrapper;
  switch (eventPriority) {
    case DiscreteEventPriority:
      listenerWrapper = dispatchDiscreteEvent;
      break;
    case ContinuousEventPriority:
      listenerWrapper = dispatchContinuousEvent;
      break;
    case DefaultEventPriority:
    default:
      listenerWrapper = dispatchEvent;
      break;
  }
  return listenerWrapper.bind(
    null,
    domEventName,
    eventSystemFlags,
    targetContainer,
  );
}
```

getEventPriority

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

```TypeScript
export function getEventPriority(domEventName: DOMEventName): EventPri
  switch (domEventName) {
    // Used by SimpleEventPlugin:
    case "cancel":
    case "click":
    case "close":
    case "contextmenu":
    case "copy":
    case "cut":
    case "auxclick":
    case "dblclick":
    case "dragend":
    case "dragstart":
    case "drop":
    case "focusin":
    case "focusout":
    case "input":
    case "invalid":
```

```
    case "keydown":
    case "keypress":
    case "keyup":
    case "mousedown":
    case "mouseup":
    case "paste":
    case "pause":
    case "play":
    case "pointercancel":
    case "pointerdown":
    case "pointerup":
    case "ratechange":
    case "reset":
    case "resize":
    case "seeked":
    case "submit":
    case "touchcancel":
    case "touchend":
    case "touchstart":
    case "volumechange":
    // Used by polyfills: (fall through)
    case "change":
    case "selectionchange":
    case "textInput":
    case "compositionstart":
    case "compositionend":
    case "compositionupdate":
    // Only enableCreateEventHandleAPI: (fall through)
    case "beforeblur":
    case "afterblur":
    // Not used by React but could be by user code: (fall through)
    case "beforeinput":
    case "blur":
    case "fullscreenchange":
    case "focus":
    case "hashchange":
    case "popstate":
    case "select":
    case "selectstart":
      return DiscreteEventPriority;
    case "drag":
```

```
    case "dragenter":
    case "dragexit":
    case "dragleave":
    case "dragover":
    case "mousemove":
    case "mouseout":
    case "mouseover":
    case "pointermove":
    case "pointerout":
    case "pointerover":
    case "scroll":
    case "toggle":
    case "touchmove":
    case "wheel":
    // Not used by React but could be by user code: (fall through)
    case "mouseenter":
    case "mouseleave":
    case "pointerenter":
    case "pointerleave":
      return ContinuousEventPriority;
    case "message": {
      // 我们可能在调度器回调中。
      // 最终，这种机制将被替换为检查本机调度器上的当前优先级。
      const schedulerPriority = Scheduler.getCurrentPriorityLevel();
      switch (schedulerPriority) {
        case ImmediatePriority:
          return DiscreteEventPriority;
        case UserBlockingPriority:
          return ContinuousEventPriority;
        case NormalPriority:
        case LowPriority:
          return DefaultEventPriority;
        case IdlePriority:
          return IdleEventPriority;
        default:
          return DefaultEventPriority;
      }
    }
    default:
      return DefaultEventPriority;
```

```
  }
}
```

捕获阶段

addEventCaptureListener

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

```JavaScript
export function addEventCaptureListener(
  target: EventTarget,
  eventType: string,
  listener: Function,
): Function {
  target.addEventListener(eventType, listener, true);
  return listener;
}
```

冒泡阶段

addEventBubbleListener

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

```JavaScript
export function addEventBubbleListener(
  target: EventTarget,
  eventType: string,
  listener: Function,
): Function {
  target.addEventListener(eventType, listener, false);
  return listener;
}
```