18-1 实现事件注册

JavaScript
```javascript
function FunctionComponent() {
  const [count, setCount] = useReducer((x) => x + 1, 0);

  return (
    <div className="border">
      <h1>函数组件</h1>
      <button onClick={() => setCount()}>{count}</button>
    </div>
  );
}
```

type DOMEventNames

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

TypeScript
```typescript
export type DOMEventName =
  | "abort"
  | "afterblur" // Not a real event. This is used by event experiments
  // These are vendor-prefixed so you should use the exported constant
  // 'animationiteration' |
  // 'animationend |
  // 'animationstart' |
  | "beforeblur" // Not a real event. This is used by event experiment
  | "beforeinput"
  | "blur"
  | "canplay"
  | "canplaythrough"
  | "cancel"
  | "change"
  | "click"
  | "close"
  | "compositionend"
  | "compositionstart"
  | "compositionupdate"
  | "contextmenu"
  | "copy"
  | "cut"
  | "dblclick"
  | "auxclick"
  | "drag"
  | "dragend"
  | "dragenter"
  | "dragexit"
  | "dragleave"
  | "dragover"
  | "dragstart"
  | "drop"
  | "durationchange"
  | "emptied"
  | "encrypted"
  | "ended"
  | "error"
  | "focus"
  | "focusin"
  | "focusout"
  | "fullscreenchange"
  | "gotpointercapture"
  | "hashchange"
  | "input"
  | "invalid"
```

```
    | "keydown"
    | "keypress"
    | "keyup"
    | "load"
    | "loadstart"
    | "loadeddata"
    | "loadedmetadata"
    | "lostpointercapture"
    | "message"
    | "mousedown"
    | "mouseenter"
    | "mouseleave"
    | "mousemove"
    | "mouseout"
    | "mouseover"
    | "mouseup"
    | "paste"
    | "pause"
    | "play"
    | "playing"
    | "pointercancel"
    | "pointerdown"
    | "pointerenter"
    | "pointerleave"
    | "pointermove"
    | "pointerout"
    | "pointerover"
    | "pointerup"
    | "popstate"
    | "progress"
    | "ratechange"
    | "reset"
    | "resize"
    | "scroll"
    | "scrollend"
    | "seeked"
    | "seeking"
    | "select"
    | "selectstart"
    | "selectionchange"
    | "stalled"
```

```
  | "submit"
  | "suspend"
  | "textInput" // Intentionally camelCase. Non-standard.
  | "timeupdate"
  | "toggle"
  | "touchcancel"
  | "touchend"
  | "touchmove"
  | "touchstart"
  // These are vendor-prefixed so you should use the exported constant
  // 'transitionend' |
  | "volumechange"
  | "waiting"
  | "wheel";
```

不适合委托的事件

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

```typescript
// 需要分别附加到媒体元素的事件列表。
export const mediaEventTypes: Array<DOMEventName> = [
  "abort",
  "canplay",
  "canplaythrough",
  "durationchange",
  "emptied",
  "encrypted",
  "ended",
  "error",
  "loadeddata",
  "loadedmetadata",
  "loadstart",
  "pause",
  "play",
  "playing",
  "progress",
  "ratechange",
  "resize",
  "seeked",
  "seeking",
  "stalled",
  "suspend",
  "timeupdate",
  "volumechange",
  "waiting",
];

// 我们不应该将这些事件委托给容器，而是应该直接在实际的目标元素上设置它们。这主要是
export const nonDelegatedEvents: Set<DOMEventName> = new Set([
  "cancel",
  "close",
  "invalid",
  "load",
  "scroll",
  "scrollend",
  "toggle",
  // 注意："error" 事件并不是一个独占的媒体事件，也可能发生在其他元素上。我们不会
  ...mediaEventTypes,
]);
```

不同类型的事件注册

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

这里以 SimpleEvent 为例：

```JavaScript
SimpleEventPlugin.registerEvents();
// EnterLeaveEventPlugin.registerEvents();
// ChangeEventPlugin.registerEvents();
// SelectEventPlugin.registerEvents();
// BeforeInputEventPlugin.registerEvents();
```

SimpleEventPlugin

普通事件，如 click、drag、drop 等。

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

```javascript
import {
  registerSimpleEvents,
} from '../DOMEventProperties';

export {registerSimpleEvents as registerEvents};
```

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

```typescript
import type { DOMEventName } from "./DOMEventNames";

import { registerTwoPhaseEvent } from "./EventRegistry";

export const topLevelEventsToReactNames: Map<DOMEventName, string | nu
  new Map();

const simpleEventPluginEvents = [
  "abort",
  "auxClick",
  "cancel",
  "canPlay",
  "canPlayThrough",
  "click",
  "close",
  "contextMenu",
  "copy",
  "cut",
  "drag",
  "dragEnd",
  "dragEnter",
  "dragExit",
  "dragLeave",
  "dragOver",
  "dragStart",
  "drop",
  "durationChange",
  "emptied",
  "encrypted",
    "ended",
    "error",
    "gotPointerCapture",
    "input",
    "invalid",
    "keyDown",
    "keyPress",
    "keyUp",
    "load",
    "loadedData",
    "loadedMetadata",
    "loadStart",
    "lostPointerCapture",
    "mouseDown",
    "mouseMove",
    "mouseOut",
    "mouseOver",
    "mouseUp",
    "paste",
    "pause",
    "play",
    "playing",
    "pointerCancel",
    "pointerDown",
    "pointerMove",
    "pointerOut",
    "pointerOver",
    "pointerUp",
    "progress",
    "rateChange",
    "reset",
    "resize",
    "seeked",
    "seeking",
    "stalled",
    "submit",
    "suspend",
    "timeUpdate",
    "touchCancel",
    "touchEnd",
    "touchStart",
  "volumeChange",
  "scroll",
  "scrollEnd",
  "toggle",
  "touchMove",
  "waiting",
  "wheel",
];

function registerSimpleEvent(domEventName: DOMEventName, reactName: st
  topLevelEventsToReactNames.set(domEventName, reactName);
  registerTwoPhaseEvent(reactName, [domEventName]);
}

export function registerSimpleEvents() {
  for (let i = 0; i < simpleEventPluginEvents.length; i++) {
    const eventName = simpleEventPluginEvents[i];
    const domEventName = eventName.toLowerCase() as DOMEventName;
    const capitalizedEvent = eventName[0].toUpperCase() + eventName.sl
    registerSimpleEvent(domEventName, "on" + capitalizedEvent);
  }
  // Special cases where event names don't match.
  registerSimpleEvent("dblclick", "onDoubleClick");
  registerSimpleEvent("focusin", "onFocus");
  registerSimpleEvent("focusout", "onBlur");
  // 另外还有ANIMATION与TRANSITION，这里不实现了
}
```