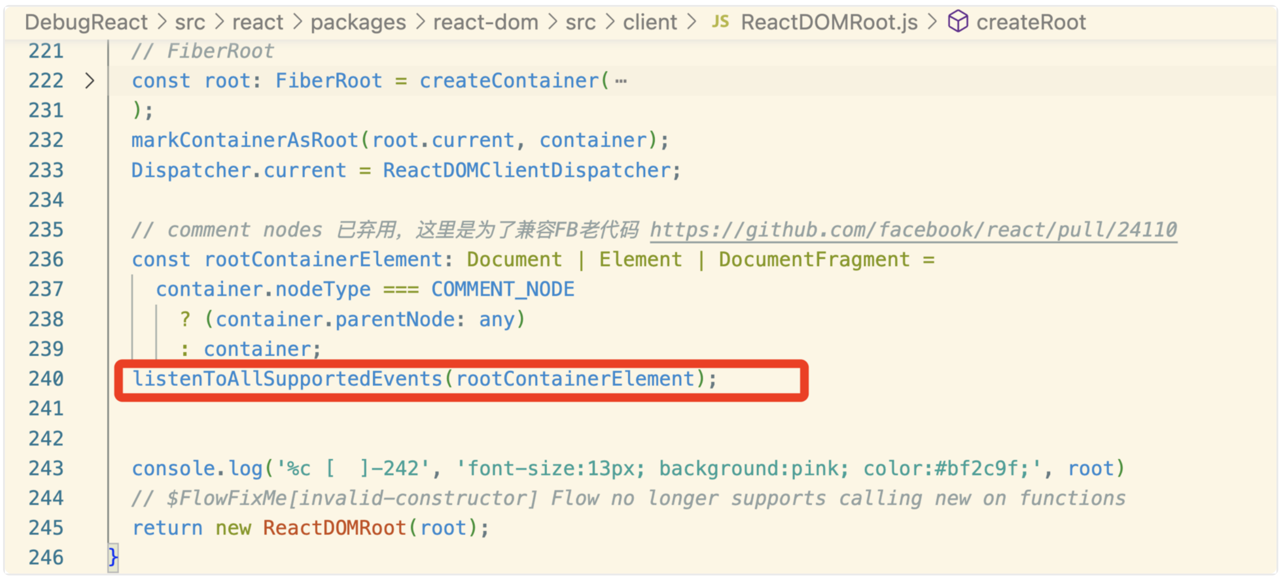
17-3 React 中的事件绑定与事件委托

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



DebugReact > src > react > packages > react-dom > src > client > JS ReactDOMRoot.js > createRoot

```
  // FiberRoot
  const root: FiberRoot = createContainer(⋯
  );
  markContainerAsRoot(root.current, container);
  Dispatcher.current = ReactDOMClientDispatcher;

  // comment nodes 已弃用，这里是为了兼容FB老代码 https://github.com/facebook/react/pull/24110
  const rootContainerElement: Document | Element | DocumentFragment =
    container.nodeType === COMMENT_NODE
      ? (container.parentNode: any)
      : container;
  listenToAllSupportedEvents(rootContainerElement);


  console.log('%c [  ]-242', 'font-size:13px; background:pink; color:#bf2c9f;', root)
  // $FlowFixMe[invalid-constructor] Flow no longer supports calling new on functions
  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: any)[listeningMarker]) {
    // sy 防止重复绑定
    (rootContainerElement: 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);
      }
    });

    // 单独处理selectionchange事件
    const ownerDocument =
      (rootContainerElement: any).nodeType === DOCUMENT_NODE
        ? rootContainerElement
        : (rootContainerElement: any).ownerDocument;
    if (ownerDocument !== null) {
      // selectionchange事件也需要去重，但它附加在document上。
      if (!(ownerDocument: any)[listeningMarker]) {
        (ownerDocument: any)[listeningMarker] = true;
        listenToNativeEvent('selectionchange', false, ownerDocument);
      }
    }
  }
}
```

listenToNativeEvent

react/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,
  isDeferredListenerForLegacyFBSupport?: 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,
  );

  let isPassiveListener: void | boolean = undefined;
  if (passiveBrowserEventsSupported) {
    // sy
    // 浏览器引入了一种干预措施，使这些事件在document上默认为passive状态。
    // React不再将它们绑定到document上，但是现在改变这一点将会撤销之前的性能优势
    // 因此，我们现在在根节点上手动模拟现有的行为。
    // https://github.com/facebook/react/issues/19651
    if (
      domEventName === 'touchstart' ||
      domEventName === 'touchmove' ||
      domEventName === 'wheel'
    ) {
      isPassiveListener = true;
    }
  }

  // React17之后，事件委托在targetContainer，但是兼容之前的版本委托在document
  targetContainer =
    enableLegacyFBSupport && isDeferredListenerForLegacyFBSupport
      ? (targetContainer: any).ownerDocument
      : targetContainer;

  let unsubscribeListener;
  // 当启用legacyFBSupport时，是为了当我们想要向container添加一次性事件监听器时
  // 这应该只与enableLegacyFBSupport一起使用，因为需要与内部FB www事件工具提供
  // 这通过在调用后立即移除事件监听器来实现。我们也可以尝试在addEventListener上使
  // 一些浏览器今天不支持这一点，考虑到这是为了支持传统代码模式，它们可能需要支持这
  if (enableLegacyFBSupport && isDeferredListenerForLegacyFBSupport) {
    const originalListener = listener;
    listener = function (...p) {
      removeEventListener(
        targetContainer,
        domEventName,
        unsubscribeListener,
        isCapturePhaseListener,
      );
      return originalListener.apply(this, p);
    };
  }

  if (isCapturePhaseListener) {
    // ! 捕获阶段
    // sy
    if (isPassiveListener !== undefined) {
      // touchstart、touchmove、wheel
      unsubscribeListener = addEventCaptureListenerWithPassiveFlag(
        targetContainer,
        domEventName,
        listener,
        isPassiveListener,
      );
    } else {
      // sy
      // click、contextmenu、drag、drop、input、mousedown、change等事件
      unsubscribeListener = addEventCaptureListener(
        targetContainer,
        domEventName,
        listener,
      );
    }
  } else {
    if (isPassiveListener !== undefined) {
      // touchstart、touchmove、wheel
      unsubscribeListener = addEventBubbleListenerWithPassiveFlag(
        targetContainer,
        domEventName,
        listener,
        isPassiveListener,
      );
    } else {
      // click、contextmenu、drag、drop、input、mousedown、change等事件
      // sy
      unsubscribeListener = addEventBubbleListener(
        targetContainer,
        domEventName,
        listener,
      );
    }
  }
}
```

createEventListenerWrapperWithPriority

react/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,
  );
}
```

捕获阶段

支持 passive

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

JavaScript

```javascript
export function addEventCaptureListenerWithPassiveFlag(
  target: EventTarget,
  eventType: string,
  listener: Function,
  passive: boolean,
): Function {
  target.addEventListener(eventType, listener, {
    capture: true,
    passive,
  });
  return listener;
}
```

addEventCaptureListener

react/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

支持 passive

```javascript
```

```
export function addEventBubbleListenerWithPassiveFlag(
  target: EventTarget,
  eventType: string,
  listener: Function,
  passive: boolean,
): Function {
  target.addEventListener(eventType, listener, {
    passive,
  });
  return listener;
}
```

addEventBubbleListener

JavaScript

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