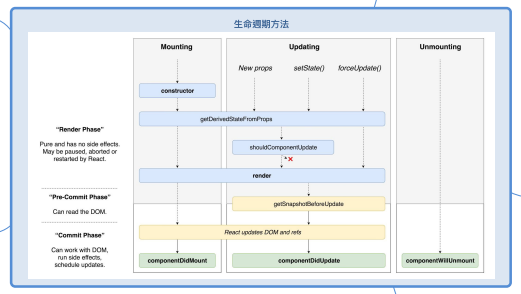
- 掛載（mounting）：初始化元件中的資料，除了render()外，在生命週期中只會被呼叫一次
  - constrsuctor：根收上層元件傳遞的props的地方，只會執行和一次
    - 可以用來作為「預先綁定」（bind the）
  - componentDidMount：在元件被建構在DOM樹中後被呼叫，只會被呼叫一次
    - 常被用來設定DOM節點的初始化
      - 可以用來新元件加上事件監聽事件
      - 可以用來設定與AJAX資料仔入
    - 可以在方法中直接呼叫setState造成顯示的更動，讓render被調用兩次
      - 可以用來在畫面DOM節點的彈跳位置錯誤
      - 可能會造成效能問題
  - getDerivedStateFromProps：會在調用 render 方法之前，在初始安裝和後續更新期間調用，在每次重渲染時都會被呼叫
    - 用於根據props的狀態來更新state
      - 會回傳一個object來更新狀態，不更新時使用null
      - ```
        state = {
          isScrollingDown: false,
          lastRow: null,
        };

        static getDerivedStateFromProps(props, state) {
          if (props.currentRow !== state.lastRow) {
            return {
            isScrollingDown: props.currentRow > state.lastRow,
            lastRow: props.currentRow,
            };
          }

          // Return null to indicate no change to state.
          return null;
        }
        ```
    - 用於加載props以外的資料
      - ```
        static getDerivedStateFromProps(props, state) {
          // Store prevId in state so we can compare when props change.
          // Clear out previously-loaded data (so we don't render stale stuff)
          if (props.id !== state.prevId) {
            return {
              externalData: null,
              prevId: props.id,
            };
          }
          // No state update necessary
          return null;
        }
        ```
    - 很罕見的，被棄用的componentWillMount只會執行一次
- 更新（updating）：元件在使用者操作過程中，會在state和props改變後被觸發操作
  - getDerivedStateFromProps()
  - shouldComponentUpdate()：讓React決定是否根據state和props的改變而更新，重新渲染元件，根據在state和props更新後，元件重新查詢新被調和
    - 如果要重新上畫新渲染，應該要使用PureComponent取代shouldComponentUpdate
      - PureComponent已經實作了shoklComponentUpdate方法
    - 如果使用方法回傳false，將render()和componentDidUpdate()不會被調用
  - render()
  - getSnapshotBeforeUpdate：在渲染完成「之前」時叫，讓元件可以在DOM被變之前擷取資訊
    - 可以用來處理增加新數以
    - ```
      getSnapshotBeforeUpdate(prevProps, prevState) {
        // Are we adding new items to the list?
        // Capture the scroll position so we can adjust scroll later.
        if (prevProps.list.length < this.props.list.length) {
        const list = this.listRef.current;
        return list.scrollHeight - list.scrollTop;
        }
        return null;
      }

      componentDidUpdate(prevProps, prevState, snapshot) {
        // If we have a snapshot value, we've just added new items.
        // Adjust scroll so these new items don't push the old ones out of view.
        // (snapshot here is the value returned from getSnapshotBeforeUpdate)
        if (snapshot !== null) {
        const list = this.listRef.current;
        list.scrollTop = list.scrollHeight - snapshot;
        }
      }
      ```
  - componentDidUpdate：在元件更新完成之後叫，在初始畫染中不會呼叫
    - 也可以在方法中直接呼叫setState，但「必須要比較props有無改變，否則將造成無窮迴圈」
    - 常被用來搭配DOM節點的操作，比較props有無改變
      - 可以用來新元件加上事件監聽事件
      - 可以用來設定與AJAX資料仔入
      - ```
        componentDidUpdate(prevProps) {
        // Typical usage (don't forget to compare props):
        if (this.props.userID !== prevProps.userID) {
          this.fetchData(this.props.userID);
        }
        }
        ```
    - componentDidUpdate(prevProps, prevState, snapshot)第後一個參數來自於getSnapshotBeforeUpdate
- 卸載（unmounting）：元件要移出真實DOM時，只有一個「componentWillUnmount」方法
  - 在元件移出DOM，調用此調用
  - 不可在其中使用setState，因為元件永遠不會被重新渲染
  - 通常用於清除事件監聽器



- 錯誤處理
  - getDerivedStateFromError(error)
  - componentDidCatch(error, info)：在下層組件發生錯誤時使用，採觸法式在commit pafse階調用，所以允許副作用
    - error代表錯誤的錯誤
    - info的物者發生錯誤的下層組件
    - 可以定義自己的元件的錯誤邊界元件
      - ```
        class ErrorBoundary extends React.Component {
        c      onstructor(p ops) {
        S      u      per(p ropS)
        t      h      is.state = {hasError: false}
        }

        C      omponentDidCatch(e rror, info) {
        t      h      is.setState({hasError: true})
        C      o      nsole.log(error, info)
        }

        r      ender() {
        i      f      (this.state.hasError) {
        r      e      t      urn <h3> Sorry, s omething went wrong.</h3>
        }

        r      e      turn t his.props.ch ildren
        }
        }
        ```
    - 解決錯誤自身的錯誤
      - React開發版本中，錯誤會冒泡至window，可以在window物件（window.onerror或window.addEventListener('error', callback)）捕獲到componentDidCatch錯誤的錯誤
      - 在React生產版本中，錯誤不會冒泡至window，這意味著任何上層組件錯誤處理程序只會在componentDidCatch()來處理錯誤
- forceUpdate：跳過組件shouldComponentUpdate方法來直接進行更新
  - 並不會像是setState方法，更改state和props後重新渲染
  - 產生的重渲染後會
  - 通過重渲染後
        ```