React 購物車實作筆記

基本概念

1. 購物車資料結構

```
const initialProducts = [
  {
    id: 0, // 唯一識別碼
    name: '小熊餅乾', // 商品名稱
    count: 1, // 商品數量
  },
  {
    id: 1,
    name: '巧克力豆餅乾',
    count: 5,
  },
  {
    id: 2,
    name: '小老板海苔',
    count: 2,
  },
]
```

資料結構說明：

- `id`: 用於唯一識別每個商品，在 React 中作為 key 使用
- `name`: 商品名稱，用於顯示
- `count`: 商品數量，用於計算和顯示

2. 狀態管理

```
const [products, setProducts] = useState(initialProducts)
```

狀態管理說明：

- `products`: 當前購物車中的商品列表
- `setProducts`: 更新商品列表的函數
- `useState`: React Hook，用於管理組件狀態

核心功能實作

1. 增加商品數量

```
const onIncrease = (productId) => {
  // 1. 使用 map 遍歷所有商品
  const nextProducts = products.map((v) => {
    // 2. 找到要修改的商品
    if (v.id === productId) {
      // 3. 創建新物件，數量加1
      return { ...v, count: v.count + 1 }
    } else {
      // 4. 其他商品保持不變
      return v
    }
  })
  // 5. 更新狀態
  setProducts(nextProducts)
}
```

使用範例：

```
// 點擊按鈕時增加商品數量
<button onClick={() => onIncrease(product.id)}>+</button>
```

2. 減少商品數量

```
const onDecrease = (productId) => {
  const nextProducts = products.map((v) => {
    if (v.id === productId) {
      return { ...v, count: v.count - 1 }
    } else {
      return v
    }
  })
  setProducts(nextProducts)
}
```

使用範例：

```
// 點擊按鈕時減少商品數量
<button
  onClick={() => {
    if (product.count <= 1) {
      if (confirm('你確定要刪除此商品?')) {
        onRemove(product.id)
      }
    } else {
      onDecrease(product.id)
```

```
    }
  }}
>
  -
</button>
```

3. 移除商品

```
const onRemove = (productId) => {
  // 使用 filter 過濾掉要刪除的商品
  const nextProducts = products.filter((v) => v.id !== productId)
  setProducts(nextProducts)
}
```

使用範例：

```
// 點擊刪除按鈕時移除商品
<button
  onClick={() => {
    if (confirm('你確定要刪除此商品?')) {
      onRemove(product.id)
    }
  }}
>
  刪除
</button>
```

4. 進階功能：數量為0時自動刪除

```
function handleDecreaseClick(productId) {
  // 1. 先減少所有商品的數量
  let nextProducts = products.map((product) => {
    if (product.id === productId) {
      return {
        ...product,
        count: product.count - 1,
      }
    } else {
      return product
    }
  })

  // 2. 過濾出數量大於0的商品
  nextProducts = nextProducts.filter((p) => p.count > 0)

  // 3. 檢查是否需要刪除商品
```

```
  if (nextProducts.length < products.length) {
    // 如果有商品被刪除，顯示確認對話框
    if (confirm('你確定要刪除此商品?')) {
      setProducts(nextProducts)
    }
  } else {
    // 如果只是減少數量，直接更新
    setProducts(nextProducts)
  }
}
```

最佳實踐

1. 狀態更新模式

```
// 正確：使用不可變更新
const nextProducts = products.map((item) =>
  item.id === id ? { ...item, count: item.count + 1 } : item
)

// 錯誤：直接修改狀態
products[0].count += 1 // 不要這樣做！
```

2. 使用者體驗優化

```
// 1. 加入載入狀態
const [isLoading, setIsLoading] = useState(false)

// 2. 加入錯誤處理
try {
  setProducts(nextProducts)
} catch (error) {
  console.error('更新失敗:', error)
  alert('更新失敗，請重試')
}

// 3. 加入動畫效果
;<div
  className="cart-item"
  style={{
    transition: 'all 0.3s ease',
  }}
>
  {/* 商品內容 */}
</div>
```

3. 程式碼組織

```
// 1. 將相關功能分組
const cartOperations = {
  increase: (id) => {
    /* ... */
  },
  decrease: (id) => {
    /* ... */
  },
  remove: (id) => {
    /* ... */
  },
}

// 2. 使用自定義 Hook
function useCart() {
  const [products, setProducts] = useState([])

  const addToCart = (product) => {
    /* ... */
  }
  const removeFromCart = (id) => {
    /* ... */
  }

  return { products, addToCart, removeFromCart }
}
```

擴展功能

1. 加入價格功能

```
const initialProducts = [
  {
    id: 0,
    name: '小熊餅乾',
    count: 1,
    price: 100,
    subtotal: 100, // 小計 = 數量 * 單價
  },
]

// 更新小計
const updateSubtotal = (products) => {
  return products.map((product) => ({
    ...product,
    subtotal: product.count * product.price,
```

```
  }))
}
```

2. 計算總金額

```
// 計算購物車總金額
const calculateTotal = (products) => {
  return products.reduce(
    (sum, product) => sum + product.count * product.price,
    0
  )
}

// 顯示總金額
;<div className="total">總金額: NT${calculateTotal(products)}</div>
```

3. 本地儲存

```
// 儲存到 localStorage
useEffect(() => {
  localStorage.setItem('cart', JSON.stringify(products))
}, [products])

// 從 localStorage 讀取
useEffect(() => {
  const savedCart = localStorage.getItem('cart')
  if (savedCart) {
    setProducts(JSON.parse(savedCart))
  }
}, [])
```

4. 加入購物車動畫

```
// CSS
.cart-item {
  transition: all 0.3s ease;
}

.cart-item-enter {
  opacity: 0;
  transform: translateX(-100%);
}

.cart-item-enter-active {
  opacity: 1;
```

```
  transform: translateX(0);
}

// 使用
<div className="cart-item">
  {/* 商品內容 */}
</div>
```

效能優化

1. 使用 useMemo

```
const total = useMemo(
  () =>
    products.reduce((sum, product) => sum + product.count * product.price, 0),
  [products]
)
```

2. 使用 useCallback

```
const handleIncrease = useCallback(
  (id) => {
    setProducts(
      products.map((product) =>
        product.id === id ? { ...product, count: product.count + 1 } : product
      )
    )
  },
  [products]
)
```

3. 虛擬列表

```
import { FixedSizeList } from 'react-window'

const CartList = () => (
  <FixedSizeList
    height={400}
    width={300}
    itemCount={products.length}
    itemSize={50}
  >
    {({ index, style }) => <div style={style}>{/* 渲染商品 */}</div>}
  </FixedSizeList>
)
```