**/\* All following exams please using Javascript only 20241220 \*/ 1.**

/\*\*

There is an array, each item has such format:

{firstName: 'xxx', lastName: 'xxx', customerID: 'xxx', note: 'xxx', profession: ‘xxx’} **lastName**, **note** can be empty, **customerID** can only be a set of digital numbers. **profession** can only have ‘student’, ‘freelancer’, ‘productOwner’, ‘engineer’ or ‘systemAnalytics’.

\*\*/

/\*\*

**Q1. Please follow the principle (‘firstName’ + ‘lastName’ + ‘customerID’) to sort this array and print it out.**

1. function sortUserName(user) {

2.   user.sort((a, b) => {

3.     const keyA = (a.firstName || "") + (a.lastName || "") + a.customerID;

4.     const keyB = (b.firstName || "") + (b.lastName || "") + b.customerID;

5.     return keyA.localeCompare(keyB);

6.   });

7.   return user;

8. }

9.

10. const sortedData = sortUserName(data);

11. console.log("sortedData==>", sortedData);

12.

13. // log出來的

14. [

15.   {

16.     "firstName": "Alice",

17.     "lastName": "Smith",

18.     "customerID": "112233",

19.     "note": "",

20.     "profession": "student"

21.   },

22.   {

23.     "firstName": "Bob",

24.     "lastName": "",

25.     "customerID": "445566",

26.     "note": "",

27.     "profession": "productOwner"

28.   },

29.   {

30.     "firstName": "Charlie",

31.     "lastName": "",

32.     "customerID": "998877",

33.     "note": "",

34.     "profession": "systemAnalytics"

35.   },

36.   {

37.     "firstName": "Jane",

38.     "lastName": "",

39.     "customerID": "67890",

40.     "note": "Important client",

41.     "profession": "freelancer"

42.   },

43.   {

44.     "firstName": "John",

45.     "lastName": "Doe",

46.     "customerID": "12345",

47.     "note": "",

48.     "profession": "engineer"

49.   }

50. ]

/\*\*

Q2. Please sort by ‘profession’ to follow the principle.

**(‘systemAnalytics’ > ‘engineer’ > ‘productOwner’ > ‘freelancer’ > ‘student’’)**

\*\*/

1. // 依照題目的profession權重大小宣告

2. const profession\_order = {

3.   systemAnalytics: 1,

4.   engineer: 2,

5.   productOwner: 3,

6.   freelancer: 4,

7.   student: 5,

8. };

9.

10. const sortByType = data.sort((a, b) => {

11.   const orderA = professionOrder[a.profession] || Infinity; // 如果職業不存在，默認為最大值

12.   const orderB = professionOrder[b.profession] || Infinity;

13.   return orderA - orderB; // 若前面比後面大，則前面那個往後面擺放

14. });

15.

16. // 輸出結果

17. console.log(sortByType);

**2.**

1. <div class="container">

2. <div class="header">5/8 外出確認表</div>

3. <div class="content">

4. <ol class="shop-list">

5. <li class="item">麵包</li>

6. <li class="item">短袖衣服</li>

7. <li class="item">飲用水</li>

8. <li class="item">帳篷</li>

9. </ol>

10. <ul class="shop-list">

11. <li class="item">暈車藥</li>

12. <li class="item">感冒藥</li>

13. <li class="item">丹木斯</li>

14. <li class="item">咳嗽糖漿</li>

15. </ul>

16. </div>

17. <div class="footer">以上僅共參考</div>

18. </div>

19.

1. .container { font-size: 14px;

2. }

3. .container .header { font-size: 18px;

4. }

5. .container .shop-list { list-style: none; margin-left: -15px;

6. }

7. .container .shop-list li.item { color: green;

8. }

Q1. **/\* Explain why does this color not works, and how to fix make it work on 1st list \*/**  color: blue;

//變更前

一張含有 文字, 螢幕擷取畫面, 陳列 的圖片

AI 產生的內容可能不正確。

// 變更後

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

AI 產生的內容可能不正確。

1. 1. <ol class="shop-list first-list"> <!-- 添加 first-list -->

2. 2. <li class="item">麵包</li>

3. 3. <li class="item">短袖衣服</li>

4. 4. <li class="item">飲用水</li>

5. 5. <

6.

li class="item">帳篷</li>

6. </ol>

7.

8. .container .first-list li.item {

9. color: blue;

10. }

11.

Q2. **/\* Write styling make every other line give background color to next one \*/**

每隔一行給下一行加背景色，即給偶數行加背景色，

基於的是剛才改過文字藍色的那一區要實踐偶數行加背景色。

.container .first-list li.item:nth-child(even) { background-color: #e0f7fa; }

**3.**

/\*\* let items = [1, 1, 1, 5, 2, 3, 4, 3, 3, 3, 3, 3, 3, 7, 8, 5, 4, 9, 0, 1, 3, 2, 6, 7, 5,

4, 4, 7, 8, 8, 0, 1, 2, 3, 1];

Please write down a function to console log unique value from this array.

把陣列 arr 裡重複的元素去掉，只留下每個元素第一次出現的那個

1. function getUniqueNumber(arr) {

3.     let unique = [...new Set(arr)]; //

4.     unique.forEach(num => console.log(num));

console.log(unique);

// [ 1, 5, 2, 3, 4, 7, 8, 9, 0, 6 ]

5.   }

6.

**4.**

**/\*\*** What is **virtual DOM** and **what purpose does it aim to solve**?? **\*\*/**

A: 提升性能sss

**5.**

**/\*\*** Can you explain about the type of **never** and **what is the differ with void**? **\*\*/**

A: never

函數執行時直接拋錯誤，永遠不會返回，當然不加nerver，typescript 也會自動判斷類型，

會用在 拋出錯誤異常，或是死循環的時機點while。

1. function throwError(): never {

2.     throw new Error("发生错误！");

3.   }

B. void

void 可以理解成，不會返回值，或是不關心返回值。

2. function log(message: string): void {

3.     console.log(message);

4.   }

5.

6.

7.   button.addEventListener("click", (): void => {

8.

9.   });

10.

**6.**

**/\*\* What is difference between framework base website and normal website (none framework) \*\*/**

React / Vue / Angular 都屬於框架web，而傳統的html/js/css 原生的，則屬於非框架網站

**7.**

**/\*\*** Read the code below**,** please figure out **why after “Switch Person” button clicked,** the tasks state doesn’t update correctly**, and how to make it update as we expected \*\*/**

# /\*\* Code block start \*/

import { useState } from 'react'; export default function TaskManager() { const [isPersonAlice, setIsPersonAlice] = useState(true); return (

<div>

{isPersonAlice ? (

<TaskCounter name="Alice" />

) : (

<TaskCounter name="Bob" />

)}

<button onClick={() => { setIsPersonAlice(!isPersonAlice);

}}>

Switch Person

</button>

</div>

); } function TaskCounter({ name }) { const [tasks, setTasks] = useState(0); return (

<>

<h1>{name}'s tasks: {tasks}</h1>

<button onClick={() => setTasks(tasks + 1)}>

Complete Task

</button>

</> );

}

# /\*\* Code block end \*/

分析狀況:

組件通過樹中的位置來識別組件，而非透過props，當我們從Alice 轉換成 Bob的時候，

React會發現 TaskCounter 組件在樹中並未發生改變，只是name這個props發生了變化，  
在默認情況下，會重複使用相同組件實例。

其實加上最組件加上 key 這個props即可

1. export default function TaskManager() {

2.   const [isPersonAlice, setIsPersonAlice] = useState(true);

3.   return (

4.     <div>

5.       {isPersonAlice ? (

6.         <TaskCounter key="alice" name="Alice" />

7.       ) : (

8.         <TaskCounter key="bob" name="Bob" />

9.       )}

10.       <button

11.         onClick={() => {

12.           setIsPersonAlice(!isPersonAlice);

13.         }}

14.       >

15.         Switch Person

16.       </button>

17.     </div>

18.   );

19. }

20.

21. function TaskCounter({ name }) {

22.   const [tasks, setTasks] = useState(0);

23.   return (

24.     <>

25.       <h1>{name} 的任務數：{tasks}</h1>

26.       <button onClick={() => setTasks(tasks + 1)}>

27.         完成一個任務

28.       </button>

29.     </>

30.   );

31. }

32.

**8.**

**/\*\*** Read about the code below, **please describe the issues** and **how you will be going to improve it \*\*/**

/\*\* Code block start \*/ const TodoList = () => {

const [todos, setTodos] = useState([

{ id: 1, text: '學習 React', completed: false, studyPoint: 3 },

{ id: 2, text: '建立專案', completed: false, studyPoint: 1 }

]);

const { id, text, studyPoint } = todos;

const [basePoints, setbasePoints] = useState(3); const [sumPoints, setSumPoints] = useState(0);

const toggleTodo = (id) => {

const todo = todos.find(t => t.id === id); todo.completed = !todo.completed; setTodos(todos);

};

const handleStudyPointsChange = (e) => { basePoints(e.target.value);

setSumPoints(parseInt(value1) + parseInt(e.target.value)); };

return ( <div>

<p>課程名稱: {text}</p>

<label>學習點數: </label>

<input type="number" value={studyPoint} onChange={handleStudyPointsChange} />

<p>總累積點數: {sumPoints}</p>

<button onClick={toggleTodo(id)}>篩選課程</button>

</div>

);

}

# /\*\* Code block end \*/

1. const [todos, setTodos] = useState([

2.     { id: 1, text: '學習 React', completed: false, studyPoint: 3 },

3.     { id: 2, text: '建立專案', completed: false, studyPoint: 1 }

4.     ]);

5.

6. // todo 是一個陣列不是物件，會報錯，把這行刪掉

7. const { id, text, studyPoint } = todos;

8.

9. const [basePoints, setbasePoints] = useState(3); // 保留

10. const [sumPoints, setSumPoints] = useState(0); // 保留

11.

12. // React state 是immutable 的，原先做法會產生非預期的錯誤

13. const toggleTodo = (id) => {

14.     const todo = todos.find(t => t.id === id);

15.     todo.completed = !todo.completed;  // 因為immutable所以這邊不可以直接修改原先的 todo 這個 state

16.     setTodos(todos); // 這邊根本沒更新到，只是把原先的 todos 又丟回給 todo 這個 state，正常情況下的覆寫應該要透過 spread operator 來取舊的值，再把要更新的 key value 加進去

17.

18.     // 改成下面這個

19.     setTodos(todos.map(todo=> todo.id === id ? {...todo, completed: !todo.completed } : todo));

20.

21. };

22.

23. // 修改前

24. const handleStudyPointsChange = (e) => {

25.     basePoints(e.target.value); // basePoints 是 state 而非 修改useState的function 需要刪掉

26.     setSumPoints(parseInt(value1) + parseInt(e.target.value));

27. };

28.

29. // 修改後

30. const hadleStudyPointsChange = (id , value ) => {

31.     // 分為 id 一樣的情況  跟 id 不同

32.     const updatedTodos = todos.map((todo)=>{

33.         todo.id === id  ? {...todo, studyPoint: Number(value)} : todo ;

34.     })

35.     // 更新後的 todos

36.     setTodos(updatedTodos);

37.     // todos 更新後，也需要重新計算點數

38.

39.     const totalPoints = updatedTodos.reduce((acc, todo) => acc + todo.studyPoint, 0);

40.     setSumPoints(totalPoints);

42. }

43.

44. return (

45. <div>

46.     <p>總累積點數: {sumPoints}</p> // 加上這句，依據 setSumPoints 有計算組合

47.     {todos.map((todo)=>{

48.         <div>

49.             <p>課程名稱: {text}</p>

50.             <label>學習點數: </label>

51.             {/\* handleStudyPointsChange 用途是要修改某一筆的 todo 內容，所以需要知道該 todo 的 id 跟 要修改的內容，例如我要修改 studyPoint 的內容 \*/}

52.             <input type="number" value={todos.studyPoint} onChange={hadleStudyPointsChange(todo.id, e.target.value)}/>

53.             <p>完成狀態: {todo.completed ? '已完成' : '未完成'}</p>

54.             <button onClick={() => toggleTodo(todo.id)}>篩選課程狀態</button>

55.         </div>

56.     })}

57. </div>

58.

59. );

60.

61.

**9.**

**/\*\*** Read about the code below, **suggest how to improve the code \*\*/**

/\*\* Code block start \*/ function ParentComponent() {

const [name, setName] = useState("Naro"); const [age, setAge] = useState(12);

return (

<div>

<ChildComponent name={name} age={age} />

<GrandchildComponent name={name} age={age} />

</div>

);

}

function ChildComponent({ name, age }) { return (

<div>

<p>Name: {name}</p>

<p>Age: {age}</p>

<GrandchildComponent name={name} age={age} />

</div>

);

}

function GrandchildComponent({ name, age }) { return (

<div>

<p>Name: {name}</p>

<p>Age: {age}</p>

</div>

);

}

# /\*\* Code block end \*/

這裡運用到 createContext 的寫法，面試官也可以從我的 github side project看到有運用這個技巧在專案中。

1. import React, { useState, createContext, useContext } from "react"; // 運用 createContext 來避免 多層 Prop Drilling 的問題

2.

3. function ParentComponent() {

4.

5.     // Create context

6.     const UserContext = createContext();

7.

8.     const [name, setName] = useState("Naro");

9.     const [age, setAge] = useState(12);

10.

11. return (

12.

13.     // 改善前的寫法

14.     // {/\* ChildComponent 跟 GrandchildComponent 都傳遞了相同的 props， 在 React 中稱為 Prop Drilling，這種寫法不太靈活，所以後來出現了 context Provider的用法，

15.     // 避免 props 傳遞時，因為多個child層傳遞\*/}

16.     // <ChildComponent name={name} age={age} />

17.     // <GrandchildComponent name={name} age={age} />

18.

19.     // 優化後

20.     <UserContext.Provider value={{ name, age }}>

21.         <ChildComponent />

22.     </UserContext.Provider>

23. );

24. }

25.

26. function ChildComponent() {     return (

27.     <div>

28.         <GrandchildComponent />

29.     </div>

30. );

31. }

32. function GrandchildComponent() {

33.     const { name, age } = useContext(UserContext);

34.     return (

35.         <div>

36.             <p>Name: {name}</p>

37.             <p>Age: {age}</p>

38.         </div>

39.     );

40. }

41.

10.

**/\*\*** Read about the code below, **achieving that make input element in “SearchInput” to be focused while search button on click \*\*/**

/\*\* Code block start \*/ function SearchButton() { return (

<button> Search </button>

);

} function SearchInput() { return (

<input/>

); } export default function Page() { return (

<>

<nav>

<SearchButton />

</nav>

<SearchInput />

</>

);

}

/\*\* Code block end \*/

# **Developer Code Quality Test**

Please fork the code base below and return it as answer, we will read through the code after we got the answer. URL: [https://codesandbox.io/p/devbox/taskmanagement-3xw7ys](https://codesandbox.io/p/devbox/task-management-3xw7ys)