/\* 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.

\*\*/

function sortUserName(user) {

return user.sort((a, b) => {

const keyA = a.firstName + a.lastName + a.customerID;

const keyB = b.firstName + b.lastName + b.customerID;

return keyA.localeCompare(keyB);

});

}

/\*\*

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

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

\*\*/

function sortByType(user) {

const priority = {

systemAnalytics: 5,

engineer: 4,

productOwner: 3,

freelancer: 2,

student: 1

};

return user.sort((a, b) => priority[b.profession] - priority[a.profession]);

}

2.

/\*\* HTML

<div class="container">

<div class="header">5/8 外出確認表</div>

<div class="content">

<ol class="shop-list">

<li class="item">麵包</li>

<li class="item">短袖衣服</li>

<li class="item">飲用水</li>

<li class="item">帳篷</li>

</ol>

<ul class="shop-list">

<li class="item">暈車藥</li>

<li class="item">感冒藥</li>

<li class="item">丹木斯</li>

<li class="item">咳嗽糖漿</li>

</ul>

</div>

<div class="footer">以上僅共參考</div>

</div>

\*\*/

/\*\* CSS

.container {

font-size: 14px;

}

.container .header {

font-size: 18px;

}

.container .shop-list {

list-style: none;

margin-left: -15px;

}

.container .shop-list li.item {

color: green;

}

.container .shop-list .item {

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

/\* 因為 .shop-list li.item specificity 比 .shop-list .item 高，需用更具體選擇器或加上 !important 才能覆蓋 \*/

color: blue;

}

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

.container .shop-list li.item:nth-child(even) {

background-color: #f0f0f0;

}

\*\*/

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.

\*\*/

function getUniqueNumber (items) {

const freq = {};

for (let num of items) {

freq[num] = (freq[num] || 0) + 1;

}

for (let key in freq) {

if (freq[key] === 1) {

console.log(Number(key));

}

}

}

4.

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

/\*\*

Virtual DOM 是 React 等框架中用來模擬真實 DOM 的 JavaScript 物件。

它的主要目的是：提升效能、避免不必要的 DOM 重繪，

先更新 virtual DOM，最後一次性比對 (diff) 再應用到真實 DOM。

\*\*/

5.

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

/\*\*

never 表示永遠不會發生的值（例如拋出錯誤、無窮迴圈）

void 表示函式不回傳任何值（或 undefined）

差異在於：never 是完全沒有返回、void 是可以 undefined

\*\*/

6.

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

/\*\*

Framework base 網站通常有現成的路由、狀態管理、元件架構（如 React、Vue）；

而 non-framework 是純手寫 HTML/CSS/JS，較難維護與擴展，缺乏模組化與最佳實踐。

\*\*/

7.

/\*\*

問題原因：React 會保留 component 的狀態，即使切換了 name，<TaskCounter /> 還是用原本的狀態。

解法：給 <TaskCounter /> 加上 key={name}，強制 React 當作不同元件處理。

\*\*/

8.

/\*\*

問題整理：

- 錯誤使用 todos 解構

- setbasePoints 拼錯 → setBasePoints

- toggleTodo 應該用匿名函式

- 應使用 map 更新 todos，避免直接變動 state

\*\*/

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 [basePoints, setBasePoints] = useState(3);

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

const toggleTodo = (id) => {

setTodos(prev =>

prev.map(t =>

t.id === id ? { ...t, completed: !t.completed } : t

)

);

};

const handleStudyPointsChange = (e) => {

const value = parseInt(e.target.value);

setBasePoints(value);

setSumPoints(todos[0].studyPoint + value); // 假設只針對第 1 筆計算

};

return (

<div>

<p>課程名稱: {todos[0].text}</p>

<label>學習點數: </label>

<input

type="number"

value={todos[0].studyPoint}

onChange={handleStudyPointsChange}

/>

<p>總累積點數: {sumPoints}</p>

<button onClick={() => toggleTodo(todos[0].id)}>篩選課程</button>

</div>

);

};

9.

/\*\*

建議改善：重複傳遞 name/age 可用 Context 優化。

建立 React.createContext，使用 useContext 取代 props 傳遞。

\*\*/

const PersonContext = React.createContext();

function ParentComponent() {

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

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

return (

<PersonContext.Provider value={{ name, age }}>

<ChildComponent />

<GrandchildComponent />

</PersonContext.Provider>

);

}

function ChildComponent() {

const { name, age } = useContext(PersonContext);

return (

<div>

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

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

<GrandchildComponent />

</div>

);

}

function GrandchildComponent() {

const { name, age } = useContext(PersonContext);

return (

<div>

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

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

</div>

);

}

10.

/\*\*

達成點擊 <SearchButton /> 後 focus 到 <SearchInput />

使用 ref 傳遞給子元件 input，點擊時觸發 .focus()

\*\*/

function SearchButton({ onSearch }) {

return (

<button onClick={onSearch}> Search </button>

);

}

function SearchInput({ inputRef }) {

return (

<input ref={inputRef} />

);

}

export default function Page() {

const inputRef = useRef(null);

const handleFocus = () => {

inputRef.current?.focus();

};

return (

<>

<nav>

<SearchButton onSearch={handleFocus} />

</nav>

<SearchInput inputRef={inputRef} />

</>

);

}