

1. Refactor the Code (Debugging)

The following Tailwind CSS code has styling issues on a card component. Identify two problems and provide a refactored version.

```
<div class="bg-blue-500 rounded-lg text-white w-500">  
  class="text-2xl">Welcome</h2>  
<p class="mt-4 text-lg">This is a card.</p>  
</div>
```

- Problem 1: The text-2xl class on the <h2> tag is overly large for a card header
- Problem 2: w-500 class is not a valid Tailwind CSS width utility it should be in Pixels

2. Multiple Choice (Design Decision)

You're styling a button with Tailwind CSS for a mobile-first app. Which approach best ensures responsiveness?

- a) `<button class="px-4 py-2 text-sm md:text-base md:px-6 md:py-3">Click Me</button>`
- b) `<button class="p-4 text-lg">ClickMe</button>`
- c) `<button class="px-6 py-3 text-base">Click Me</button>`
- d) `<button class="p-2 md:text-sm">Click Me</button>`

Answer: a

3. True or False (Coding Standards)

- a) Tailwind CSS encourages using utility classes directly in HTML to keep styles predictable and maintainable.

Answer: True

using utility classes **directly in HTML** avoids naming collisions, and keeps styling close to the markup for better maintainability.

- b) It's a good practice to define custom Tailwind classes in a separate CSS file for reusability.

Answer: True

While Tailwind is utility-first, it **does support extracting repeated utility combinations** into reusable classes using @apply in a custom CSS file

4. Fill in the Blank (Performance)

To reduce Tailwind CSS bundle size in production, you should enable _____ in the tailwind.config.js file.

Answer: To reduce Tailwind CSS bundle size in production, you should enable purge in the tailwind.config.js file.

5. 5. Guess the Output

What will this Tailwind-styled div look like on a screen smaller than 640px?

```
<div class="bg-red-500 sm:bg-blue-500 p-4 text-white" > Hello, World!
</div>
```

- a) Red background, white text
- b) Blue background, white text
- c) No background, white text
- d) Red background, no text

Answer: a

6. Map the Items (Design Decision)

Match the Tailwind CSS utility to its purpose:

- a) flex 1 . Centers text horizontally
- b) text-center _____ 2. Enables flexbox layout
- c) space-x-4 _____ 3. Adds margin between flex items

Answers: a-2

b-1

c-3

7. Short Answer (Performance)

Your Tailwind CSS project takes 1.5 seconds to load due to a large CSS file. Suggest two strategies to optimize it without losing functionality.

- Strategy 1 : Enable PurgeCSS or Tailwind's purge option to remove unused styles.
-
- Strategy 2: Use a CDN to load Tailwind CSS and enable lazy loading for styles
-

Section 2: Vanilla JavaScript

8. Refactor the Code (Debugging)

This JavaScript code has performance and readability issues. Identify one performance issue and one readability issue, then refactor it.

```
const btn = document .getElementById( 'btn' ) ; btn. addEventListener ( click' . function ( ) {let items
document . aetElementsBvClassName ( item ) ;for (let i 0; i < items . length; i++)items C i] . style .
backgroundC010r 'blue';
```

- Performance Issue:The `getElementsByClassName('item')` is called inside the event listener on every click, leading to repeated DOM queries which can be inefficient.

-

- Readability Issue: Poor syntax, spacing, and unclear structure

- Refactored Code:

- `const btn = document.getElementById('btn');`
- `const items = document.getElementsByClassName('item');`
-
- `btn.addEventListener('click', function () {`
- `for (let i = 0; i < items.length; i++) {`
- `items[i].style.backgroundColor = 'blue';`
- `}`
- `});`

9. Multiple Choice (Output Prediction)

What does this code log to the console?

```
const arr = [1, 2, 3];
arr.forEach(function (num) {
  console.log(num * 2);
});
```

a) [2, 4, 6]

b) 2, 4, 6 (on separate lines)

d)

undefine

Answer: b

10. True or False (Coding Standards)

a) Using const for variables that won't be reassigned is a best practice in JavaScript.

Answer: True

b) Arrow functions () are always shorter and clearer than traditional function declarations.

Answer: False

11. Fill in the Blank (Performance)

To avoid memory leaks in event listeners, you should use _____ to remove them when they're no longer needed.

Answer: removeEventListener

12. Guess the Output

What's logged by this code?

```
let x = 10 ;function test ( ) {let x = 20 ;console. log (x) ;  
test ( ) ;
```

a) 10

b) 20

c) undefined

d) Error

Answer: 20

13. Map the Items (Functionality)

Match the JavaScript method to its purpose:

a) querySelector _____ 1. Executes a function after a delay

b) setTimeout _____ 2. Selects the first matching element

c) map _____ 3. Creates a new array from an array

Answers: a-2, b-1, c-3

14. Short Answer (Performance)

Your Vanilla JS app re-renders a list of 1,000 items on every button click. Suggest one technique to optimize this.

Answer: Use requestAnimationFrame to batch and optimize DOM updates for the list.

15. Multiple Choice (Debugging)

Why does this code throw an error?

```
const data undefined; console . log (data . name) ;
```

a) data is not an object

b) name is a reserved keyword

- c) data is not declared
- d) console. log is misused

Answer: A

16. Code Challenge (Coding Standards)

Write a Vanilla JS function to toggle a class active on an element with ID box .
Ensure it follows ES6+ standards and is concise.

Answer: code : `const Active= () =>
const box = document.getElementById('box');
box.classList.toggle('active');
};`

17. Refactor the Code (Debugging)

This React component causes a warning in the console. Identify the issue and refactor it.

```
function Counter ( ) {const [count, setCount]=React. useState (0) ; return (
<div>
  Count: { count } </p>
  <button onClick={ ( ) => setCount (count++) } > Increment</button>
< / div >
```

– Issue:

Refactored Code: `import { useState } from 'react';`

```
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

`export default Counter;`

18. Multiple Choice (Performance)

Which approach improves React rendering performance for a large list?

- a) Render all items with a unique key prop
- b) Use inline styles instead of Tailwind CSS
- c) Wrap the list in a div with display: none
- d) Avoid useState for list data

Answer:a

19. True or False (Coding Standards)

a) React components should always start with a capital letter to distinguish them from HTML elements.

Answer: b) It's fine to use useEffect for all side effects, even simple calculations.

Answer: a. True

20. Fill in the Blank (Design Decision)

To share state between two sibling React components, you should lift the state to their _____ component.

Answer: Parent

21. Guess the Output

What does this component render after clicking the button once?

```
function ADD ( ) {const r text, set Text] React.useState ( 'Hello' ) ; return (
<div>
< { text }</p>
< button onClick={ ( ) => set Text ( 'World' ) }>Change</button>
</div>
```

- a) Hello
- b) World
- c) Hello World
- d) Nothing

Answer:world

22. Map the Items (React Hooks)

Match the React Hook to its purpose:

- a) useState1. Handles side effects
- b) useEffect 2. Manages component state
- c) useRe f 3. Creates a mutable reference

Answers: a-2,b-1,c-3

23. Short Answer (Performance)

Your React app re-renders unnecessarily when props don't change. Name a React feature to prevent this.

Answer React.memo

24. Multiple Choice (Design Decision)

You're building a form in React with Tailwind CSS. How should you handle form state?

- a) Store all inputs in a single useState object
- b) Use a separate useState for each input
- c) Store state in Vanilla JS variables
- d) Avoid state and use onChange directly

Answer:a

25. Code Challenge (Debugging + Standards)

Write a React component that fetches a list of users from <https://jsonplaceholder.typicode.com/users> and displays their names in a Tailwind-styled list. Handle loading and errors, and ensure clean code practices.

```
import React, { useState, useEffect } from 'react';

function UserList() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchUsers = async () => {
      try {
        const response = await fetch('https://jsonplaceholder.typicode.com/users');
        if (!response.ok) throw new Error('Failed to fetch users');
        const data = await response.json();
        setUsers(data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchUsers();
  }, []);

  if (loading) return <div className="text-center text-gray-500">Loading...</div>;
  if (error) return <div className="text-center text-red-500">Error: {error}</div>;

  return (
```

```
<div className="container mx-auto p-4">

  <h2 className="text-2xl font-bold mb-4">User List</h2>

  <ul className="list-disc pl-5 space-y-2">

    {users.map((user) => (

      <li key={user.id} className="text-lg text-blue-600">

        {user.name}

      </li>

    ))}

  </ul>

</div>

);

}
```

```
export default UserList;
```