**Q) Disadvantage of vanilla CSS in react. CSS per Component class**

-> If we have same CSS class or Id name in multiple CSS classes, it is going to clash.

**Q) Inline CSS in React Components.**

-> <p styles= {{

font-color: red;

text-align: center

}}>Hi</p>

Here, {{}}, styles need an object, that’s why we have {} inside {}.

**Q) Dynamic and Conditional styles?**

* **making inline styles dynamic:**

<p style={{

Background-color: age>45 ? ‘red’:’green’;

}}>

HIII

</p>

* **making CSS class dynamic:**

<p className={age>45? ‘senior’:’Junior}> HII</p>

* **Making CSS class dynamic with previous class**

Let’s say we have to create a dynamic class in a such a way that, the dynamic class will be append with already existing class.

e.g. let’s say we have CSS like below

.label.invalid{}

.label.valid{}

<p className= { **` label ${ age>45? ‘invalid:’valid’ }`** }></p>

**Q) Why we use CSS Module instead of vanilla CSS?**

To remove the CSS Classes/Id’s clash, we create CSS module.

Create a CSS file with naming convention as : **Header.module.css**

Import the file as:

Import **style** from **‘./Header.module.css’**

And then use the respective classes/id like **style.paragraph**

**Q) Tailwind CSS.**

The key concept of using tailwind CSS is to remove CSS dependency. Anyone can design even if he is not aware of CSS. We don’t create explicit CSS file rather we create design inside HTML tag only.

**Q) Iterating through list in js:**

list.map((data) => ()), Here after => we should be using ( not {}

**Q) Why we use useRef().**

**a) Reading DOM input object:**

Lets say we have input box and we want to read and display the value in the web page, ideally we uses useState() hooks to perform this kind of work, This can be achieve by using useRef() hooks as it basically gets linked with input field and we don’t need to use useState(). The code is very less also no component rendering is required.

**b) Variable to map with each component rendering.**

Lets say we have variable in one component **let timer,** if we call 4 times this components , each time the timer value will get initialized and and the last value will get lost.

If we want to hold values of **timer** with respect to each component, we can create time using

useRef().

**Q) CSS design to devide page into fixed width and height:**

App.css:

    .container {

        height: 100dvh;

        width: 100wh;

        display: flex;

        }

App.js (25% and 75%):

    <div className="container">

      <SideBar style={{ flex: "0 0 25%;" }} />

      <div style={{ flex: "1" }}></div>

    </div>

**Q) Concatenating values with array:**

  const [tasks, setTasks] = useState([]);

  const taskData = useRef();

  function addTaskHandler() {

    setTasks((prev) => [...tasks, taskData.current.value]);

  }

**Q) Deleting an element from array:**

const deleteTaskHandler = (index) => {

        setTasks((prevTasks) => prevTasks.filter((\_, i) => i !== index));

  };

1. **Difference between useRefs and useState().**

Both the hooks hold object value with respect to every rendering of the components but

When the object of useRefs() gets update , It won’t re-render the components.

**\*\*) Create a object with empty array.**

export const CartContext = createContext({

items: []

});

**Q&A) Difference between map and forEach?**

If the requirement is to return any values then we use map() and if just have to iterate the list, we use forEach.

**Q&A)) Infinite Loop fix ?**