Tutorial 11: ReactJS (3)

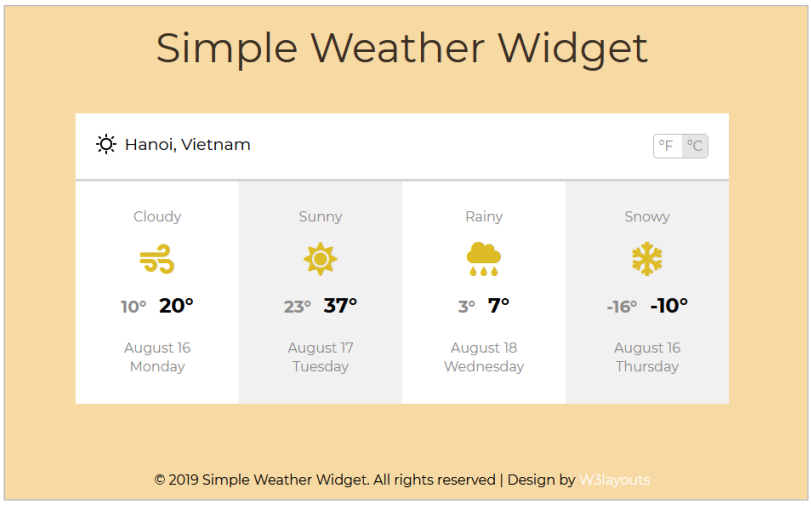
Objectives

In this tutorial, we focus on practicing:

- Create your own Class & Function components
- Handle events (basic) with use of arrow functions
- Pass data using `props`
- Init & Change `state` to re-render components
- Refactor react components by **lifting state up** into a parent component

Tutorial Exercises

In this tutorial, we aim to build a Weather App.



Download the *starter_pack* (Weather App in html version) and complete these exercises.

Exercise 1: Weather App – components (30 mins)

- Think to refactor weather app (HTML version) into react with components.
  - How many components?

    - With each of component, what are the state/ props? → decide to use Class or Function component
- Create your own components for Weather App. (bigger first)

Note: find other weather icons here
https://www.w3schools.com/icons/fontawesome5_icons_weather.asp

Exercise 2: Weather App – passing data (30 mins)

- What are the data in this app?
    - How to model these data?
- Which component to hold the data?

Note: **Lifting state up**

- *To collect data from multiple children, or*
- *To have two child components communicate with each other,*

→ *you need to declare the shared state in their parent component instead. The parent component can pass the state back down to the children by using props; this keeps the child components in sync with each other and with the parent component.*

Exercise 3: Weather App – handling events (30 mins)

- Handle user click to switch between °C or °F, the app should update all days with corresponding temperatures.

Hint: always use `setState()` to update the state. To understand more about *Why mutability is* important (read https://reactjs.org/tutorial/tutorial.html#why-immutability-is-important)

Homeworks: FlashCards

Recall: in the previous tutorial, we refactored flash-cards app into React with 03 components App, FlashCard & StatusBar.

- Continue to practice using props & state
- Can FlashCard & StatusBar are Function Components?
- Lifting state: `currentIndex` into App to update both `FlashCard` & `StatusBar` when we navigate between cards.

IFY

IFY 1: Developer Tools

The React Devtools extension for Chrome and Firefox lets you inspect a React component tree with your browser's developer tools.

```
▼<Game>
  ▼<div className="game">
    ▼<div className="game-board">
      ▼<Board>
        ▼<div>
            <div className="status">Next player: X</div>
          ▼<div className="board-row">
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            </div>
          ▼<div className="board-row">
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            </div>
          ▼<div className="board-row">
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            ▶<Square>…</Square>
            </div>
          </div>
        </Board>
      </div>
    ▼<div className="game-info">
        <div/>
        <ol/>
      </div>
    </div>
  </Game>
```

- The React DevTools let you check the props and the state of your React components.
- After installing React DevTools, you can right-click on any element on the page, click "Inspect" to open the developer tools, and the React tabs ("⚛ Components" and "⚛ Profiler") will appear as the last tabs to the right. Use "⚛ Components" to inspect the component tree.