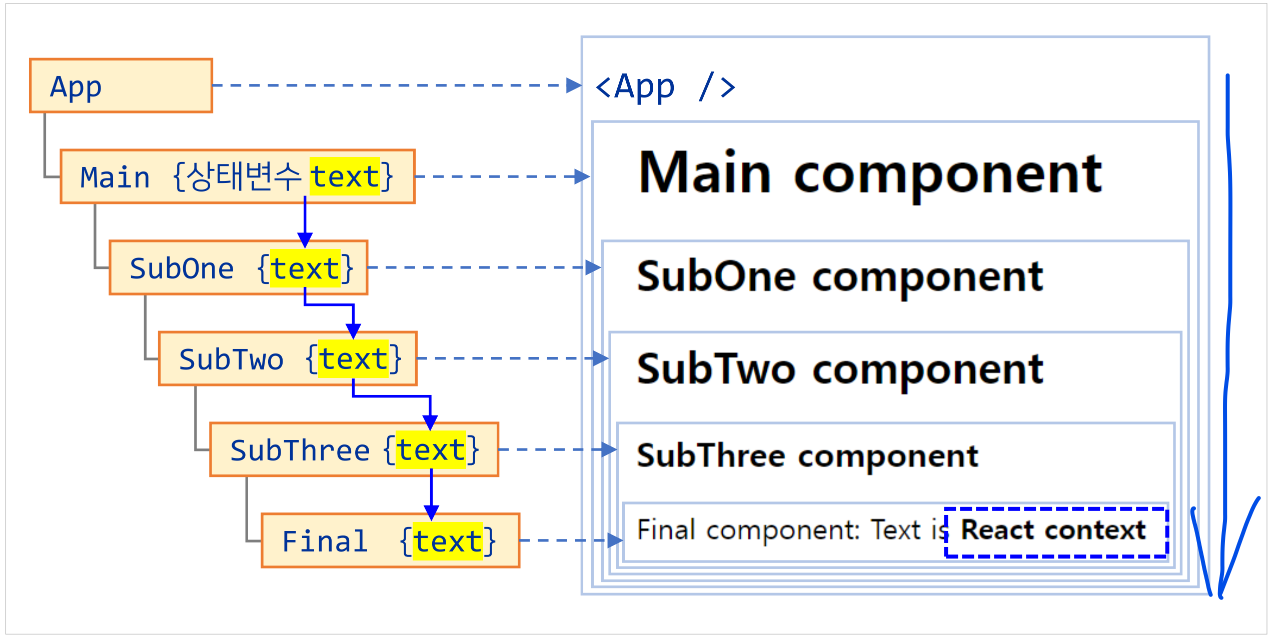
# 6. 리액트 상태관리 (3)

Prof. **Seunghyun Park** (sp@hansung.ac.kr)

Division of Computer Engineering

# 하위 컴포넌트로 데이터 전달



```
/* ch11/proj/03-1/src/Main.js */
import { useState } from "react";
import SubOne from "./SubOne";

const Main = () => {
  const [text, setText] = useState("React context");
  return (
    <>
      <h1>Main component</h1>
      <SubOne text={text} />
    </>
  )
};
export default Main;
```

하위 컴포넌트에서 데이터를 사용하기 위해
컴포넌트 계층을 따라 데이터를 전달

```
/* ch11/03-1/src/SubOne.js */
import SubTwo from "./SubTwo";

const SubOne = ({ text }) => (
  <>
    <h2>SubOne component</h2>
    <SubTwo text={text} />
  </>
)
export default SubOne;
```

```
/* ch11/03-1/src/SubTwo.js */
import SubThree from "./SubThree";

const SubTwo = ({ text }) => (
  <>
    <h2>SubTwo component</h2>
    <SubThree text={text} />
export default SubTwo;
```

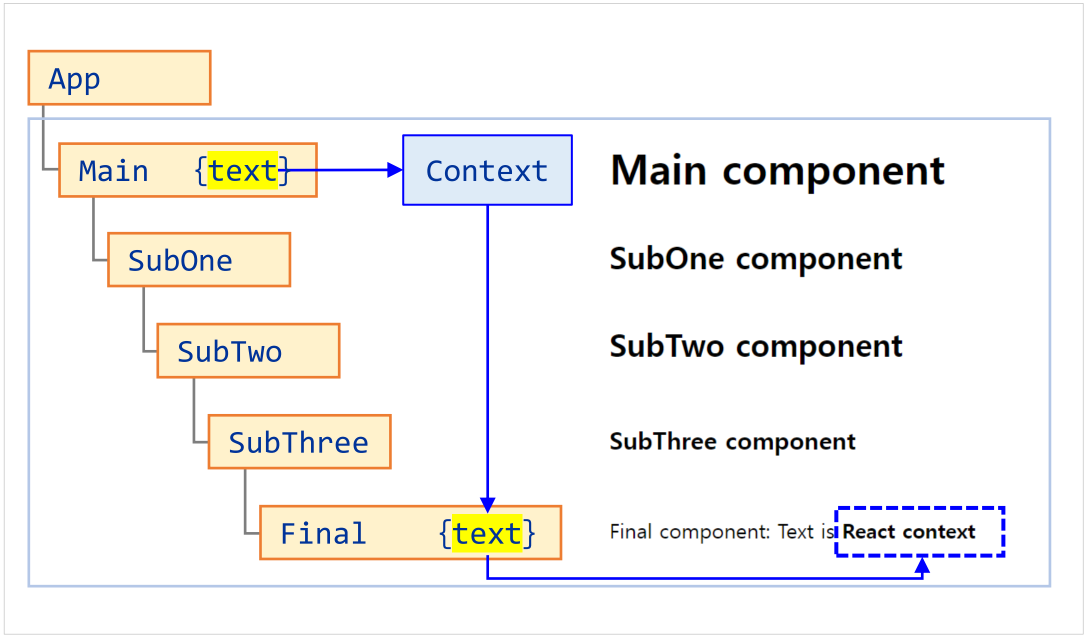
```
/* ch11/03-1/src/SubThree.js */
import Final from "./Final";

const SubThree = ({ text }) => (
  <>
    <h3>SubThree component</h3>
    <Final text={text} />
export default SubThree;
```

```
/* ch11/03-1/src/Final.js */

const Final = ({ text }) => (
  <p>Final component:
     Text is <b>{text}</b></p>
);

export default Final;
```

중간 컴포넌트에서는 실제 사용하지 않는 데이터: 중간 컴포넌트는 데이터 전달 경로로만 사용

# 리액트 Context 활용

전달 → 전달 → … 하지 않고도 가능



**Main component**

**SubOne component**

**SubTwo component**

**SubThree component**

Final component: Text is **React context**

```
/* ch11/proj/03-2/src/Main.js */
import { useState, createContext } from "react";
import SubOne from "./SubOne";

export const CreateContext = createContext()
const Main = () => {
  const [text, setText] = useState("React context");
  return (
    <CreateContext.Provider text={text}>
      <h1>Main component</h1>
      <SubOne text={text} />
    </CreateContext.Provider> );
};
export default Main;
```

Context를 생성하여
컴포넌트 트리를 포함시킴

Context 활용:
목적지 컴포넌트에서 직접 데이터 참조 가능

```
/* ch11-03-2/src/SubOne.js */
import SubTwo from "./SubTwo";

const SubOne = () => (
  <>
    <h2>SubOne component</h2>
    <SubTwo />
  </>
);
export default SubOne;
```

```
const SubTwo = () => (
  <>
    <h2>SubTwo component</h2>
    <SubThree />
  </>
);
const SubThree = () => (
  <>
    <h3>SubThree component</h3>
    <Final />
  </>
);
```

중간 컴포넌트에 데이터 전달 불필요

```
/* ch11-03-2/src/Final.js */
import { useContext } from "react";
import { CreateContext } from "./Main";

const Final = () => {
  const text = useContext(CreateContext);
  return (
    <p>Final component: Text is <b>{text}</b></p> );
};
export default Final;
```

# ContextProvider와 Custom hooks

```
/* ch11/proj/03-3/src/TextProvider.js */
import { useState, createContext, useContext } from "react";

const TextContext = createContext();
export const useText = () => useContext(TextContext);

const TextProvider = ({ children }) => {
  const [text, setText] = useState("React context");
  return (
    <TextContext.Provider value={{ text, setText }}>
      {children}
    </TextContext.Provider>
  );
};
export default TextProvider;
```

ContextProvider를 사용하는 custom hooks 정의

데이터 제공을 위한 ContextProvider를 별도로 생성하여 활용

```
/* ch11/proj/03-3/src/Main.js */
import TextProvider from "./TextProvider";
import SubOne from "./SubOne";
const Main = () => (
  <TextProvider>
    <h1>Main component</h1>
    <SubOne />
  </TextProvider>
);
export default Main;
```

```
/* ch11/proj/03-3/src/Final.js */
import { useText } from "./TextProvider";
const Final = () => {
  const { text } = useText();
  return (
    <p>Final component: Text <b>{text}</b></p>
  );
};
export default Final;
```

custom hooks을 사용하여 데이터 반환