Google | طويق

React Context

أكاديمية طويق
TUWAIQ ACADEMY

وزارة الاتصالات وتقنية المعلومات
MINISTRY OF COMMUNICATIONS AND INFORMATION TECHNOLOGY

الاتحاد السعودي للأمن السيبراني والبرمجة والدرونز
SAUDI FEDERATION FOR CYBERSECURITY, PROGRAMMING & DRONES

مهارات المستقبل

stc

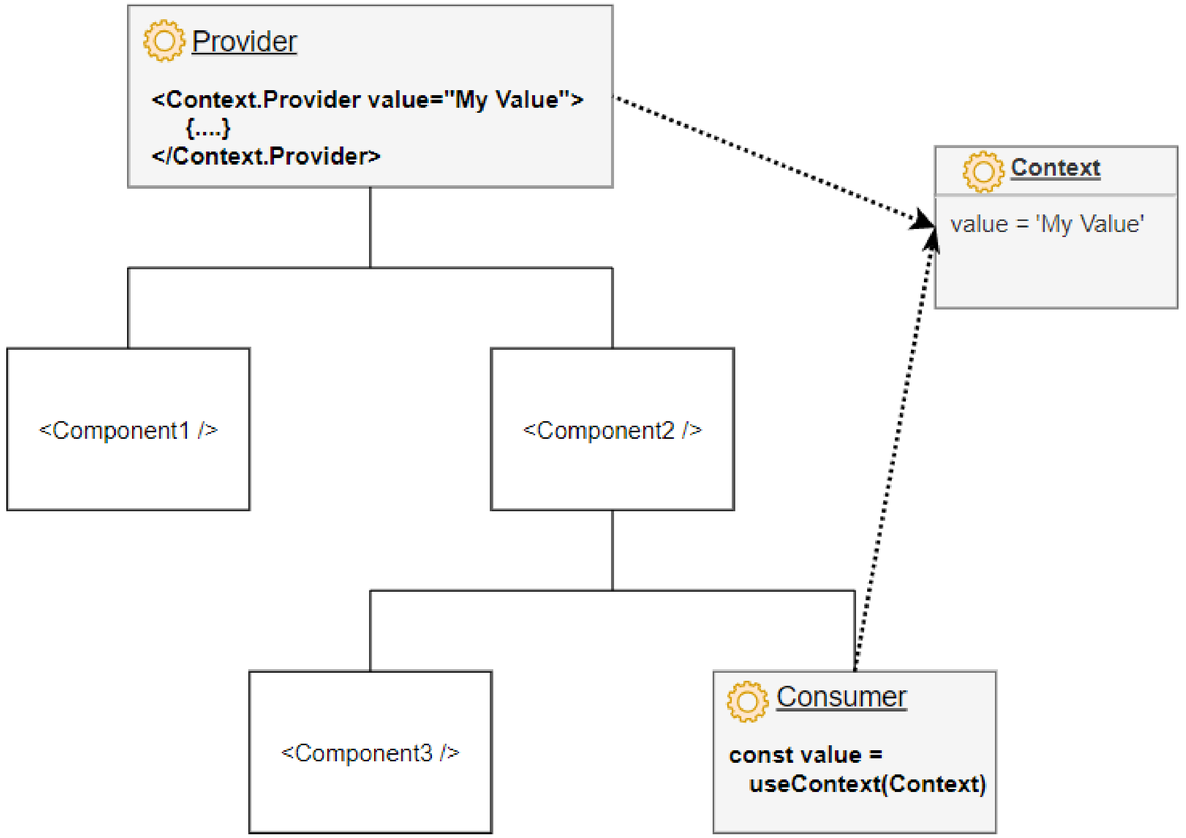
Objective

- What is the context.
- When to Use Context?
- How to use the context?
- How to name context in React?
- If-Else in JSX

What is the Context?

- Context provides a way to pass data through the component tree without having to pass props down manually at every level.

- In a typical React application, data is passed top-down (parent to child) via props, but such usage can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application. Context provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.



When to Use Context?

- Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language. For example, in the code below we manually thread through a “theme” prop in order to style the Button component:

- Using context, we can avoid passing props through intermediate elements.

You can hold inside the context:
•global state
•theme
•application configuration
•authenticated user name
•user settings
•preferred language
•a collection of services

How to use the context?

A. Creating the context

```
import { createContext } from 'react';

const Context = createContext('Default
Value');
```

How to use the context?

B. Providing the context

- **Context.Provider component available on the context instance is used to provide the context to its child components, no matter how deep they are.**
- **To set the value of context use the value prop available on the `<Context.Provider value={value} />`:function Main() {**

```
const value = 'My Context Value';

return (

<Context.Provider value={value}>

<MyComponent />

</Context.Provider>

);

}
```

How to use the context?

C. Consuming the context

- Consuming the context can be performed in 2 ways.

1. The first way, is to use the useContext(Context) React hook:

```
import { useContext } from 'react';

function MyComponent() {

const value = useContext(Context);

return <span>{value}</span>;

}
```

- The hook returns the value of the context: value = useContext(Context). The hook also makes sure to re-render the component when the context value changes.

How to use the context?

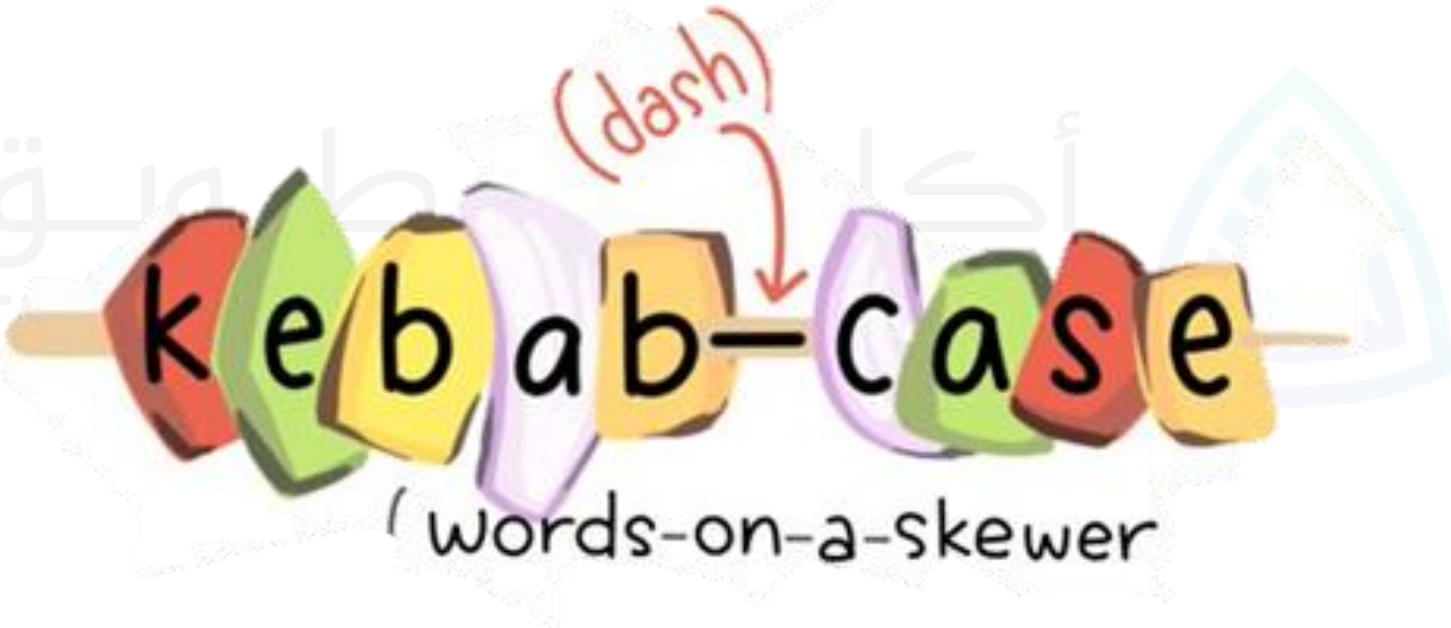
C. Consuming the context

2. The second way is by using a render function supplied as a child to Context.Consumer special component available on the context instance:

```
function MyComponent() {

return (

<Context.Consumer>

{value => <span>{value}</span>}

</Context.Consumer>

);

}
```

- Again, in case if the context value changes, <Context.Consumer> will re-render its render function.

How to name context in React?



If-Else in JSX

- if-else statements don't work inside JSX. This is because JSX is just syntactic sugar for function calls and object construction.
- define immediately-invoked function expressions inside your JSX:

```
return (
  <section>
    <h1>Color</h1>
    <h3>Name</h3>
    <p>{this.state.color || "white"}</p>
    <h3>Hex</h3>
    <p>
      {(() => {
        switch (this.state.color) {
          case "red":   return "#FF0000";
          case "green": return "#00FF00";
          case "blue":  return "#0000FF";
          default:      return "#FFFFFF";
        }
      })()}
    </p>
  </section>
);
```

Resources

- https://dmitripavlutin.com/react-context-and-usecontext/#1-how-to-use-the-context
- https://reactjs.org/docs/context.html#when-to-use-context
- https://ar.reactjs.org/docs/conditional-rendering.html
- https://react-cn.github.io/react/tips/if-else-in-JSX.html
- https://www.digitalocean.com/community/tutorials/7-ways-to-implement-conditional-rendering-in-react-applications