

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green color. They are positioned diagonally, with the blue one in front of the green one.

React Hooks

Prof. Guilherme Alves

O que são hooks?

Os hooks são métodos que nos permitem definir estado e realizar o controle do ciclo de vida em componentes funcionais. Foram introduzidos na versão 16.8 da biblioteca, tornando os componentes funcionais a principal recomendação da comunidade. Os hooks devem ser utilizados somente no nível superior de uma função e nunca devem ser utilizados dentro de blocos.





useState



```
1  const [state, setState] = React.useState({});  
2  
3  setState((current) => current);
```



useEffect



```
1 React.useEffect(() => {  
2     // execução de requisições assíncronas  
3     (async () => {}) ();  
4  
5     // executa a função logo antes de desmontar o componente  
6     return () => {};  
7 }, []); // quando vazio executa uma única vez e se não passar executa sempre
```

useRef



```
1  const Component = () => {
2    const refUncontrolled = React.useRef();
3
4    const handlerClick = () => {
5      const { value } = refUncontrolled.current;
6      refUncontrolled.current.focus();
7    }
8
9    return (
10     <React.Fragment>
11       <input ref={refUncontrolled} />
12       <button onClick={handlerClick}>click</button>
13     </React.Fragment>
14   );
15 };
```

useContext



```
1  const Ctx = React.createContext({});
2
3  const useCtx = () => {
4    const context = React.useContext(Ctx);
5
6    return { ...context };
7  };
8
9  const ProviderCtx = ({ children }) => {
10    const [state, setState] = React.useState({});
11
12    const change = (value) => setState({ value });
13
14    return (
15      <Ctx.Provider value={{ ...state, setState, change }}>
16        { children }
17      </Ctx.Provider>
18    );
19  };
```

useReducer

```
1  const Component = () => {
2    const [state, dispatch] = React.useReducer((state, action) => {
3      const { type, payload } = action;
4
5      switch (type) {
6        case 'increment':
7          return state + 1;
8        case 'decrement':
9          return state - 1;
10       default:
11         return state;
12     }
13   }, 0);
14
15   const decrement = () => dispatch({ type: 'decrement' });
16   const increment = () => dispatch({ type: 'increment' });
17
18   return (
19     <React.Fragment>
20       {state}
21       <button onClick={decrement}>-</button>
22       <button onClick={increment}>+</button>
23     </React.Fragment>
24   );
25 };
```