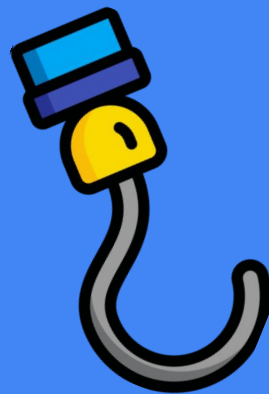


React Hooks

Por que, o que é e como usar?

Marcelo Lopes Lotufo
Freelance Software Developer

<https://marceloll.com/>



Por que? - Quais problemas pretende resolver?


O que é? - O que exatamente é um hook? Como ele funciona?

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Por que?

Por que? - Quais problemas pretende resolver?

Confusão causado por Classes e o this

 stackoverflow

React: "this" is undefined inside a component function

Asked 3 years, 8 months ago · Active 4 months ago · Viewed 71k times

ES6 React.Component doesn't auto bind methods to itself. You need to bind them yourself in constructor. Like this:

165

```
constructor (props){
  super(props);

  this.state = {
    loopActive: false,
    shuffleActive: false,
  };

  this.onToggleLoop = this.onToggleLoop.bind(this);
}
```

"However, we found that class components can encourage unintentional patterns that make these optimizations fall back to a slower path. Classes present issues for today's tools, too. For example, classes don't minify very well, and they make hot reloading flaky and unreliable. We want to present an API that makes it more likely for code to stay on the optimizable path." - [React documentation](#)

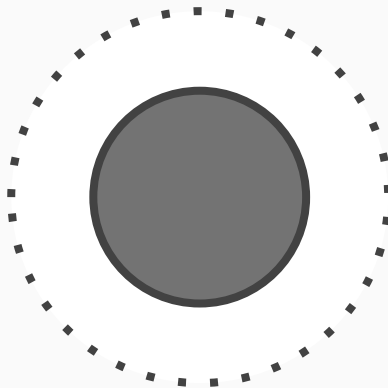
[Erro comum em relação ao uso do this e classes React](#)

Por que? - Quais problemas pretende resolver?

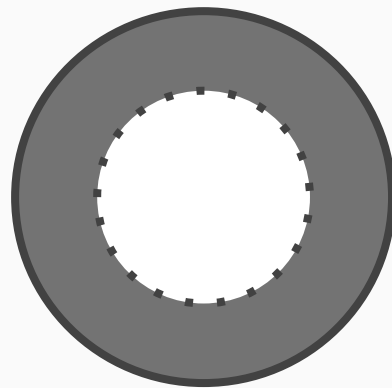
Wrapper hell e a dificuldade em dividir lógica de estado

```
<StyleWrapper theme={colors: {...}, focusColors: {...}, shadows: {...}, ...>
  <div>
    <style* { box-sizing: border-box; }/>
    <pure(StyleContext)>
      <StyleContext>
        <mapProps(mapProps(div)) theme={colors: {...}, focusColors: {...}, ...>
          <mapProps(div) theme={colors: {...}, focusColors: {...}, shadows: {...}, ...>
            <div style={color: "#d9d9d9", fontFamily: "Arial, sans-serif">
              <Overlay left=0 top=0 width=933...>
                <Broadcast channel="overlay" value={box: {...}, addLayer: fr...>
                  <div>
                    <Viewport box={left: 0, top: 0, width: 933, ...}>
                      <Broadcast channel="viewport" value={left: 0, top: 0, ...>
                        <KeyboardContext>
                          <getContext(mapProps(mapProps(div)))>
                            <mapProps(mapProps(div)) theme={colors: {...}, focu...>
                              <mapProps(div) theme={colors: {...}, focusColors: {...}, ...>
                                <div style={background: "#d9d9d9", position: "...>
                                  <getContext(mapProps(mapProps(div))) style={...>
                                    <mapProps(mapProps(mapProps(div))) style={...>
                                      <mapProps(div) style={position: "relative">
                                        <div style={position: "relative", boxSiz...>
                                          <LayoutColumn>
                                            <mapProps(mapProps(PassThrough)) theme...>
                                              <mapProps(PassThrough) theme={colors: {...}, ...>
                                                <PassThrough style={display: "flex">
                                                  <div style={display: "flex", flex...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <LayoutCell component=ShouldUpda...>
                                                    <mapProps(mapProps(PassThrough)...>
                                                      <mapProps(PassThrough) compon...>
                                                        <PassThrough component=Shoul...>
                                                          <pure(Button) keyboardFocus...>
                                                            <Button keyboardFocus=trui...>
                                                              <ButtonShell keyboardFoc...>
                                                                <mapProps(mapProps(map...>
                                                                  <mapProps(mapProps(me...>
                                                                    <mapProps(mapProps(i...>
                                                                      <mapProps(mapProps...>
                                                                        <mapProps(mapProp...>
                                                                          <mapProps(withP...>
                                                                            <withProps(wit...>
                                                                              <withHandler...>
                                                                                <div keyboa...>
                                                                                  <LayoutCel...>
                                                                                    <mapProp...>
                                                                                      </LayoutCel...>
```

Um exemplo de divisão de lógica de estado é um componente que só renderiza em certos tamanhos de tela. (i.e. MediaQuery) ou consumir um estado de um componente superior.



HOC



Render props

Wrapper hell

Por que? - Quais problemas pretende resolver?

Código espalhado pela classe com mesmo conceito

```
import * as React from "react";
import { Card, Row, Input, Text } from "../components";
import ThemeContext from "../ThemeContext";

export default class Greeting extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: "Harry",
      surname: "Potter",
      width: window.innerWidth
    };
    this.handleNameChange = this.handleNameChange.bind(this);
    this.handleSurnameChange = this.handleSurnameChange.bind(this);
    this.handleResize = this.handleResize.bind(this);
    window.addEventListener("resize", this.handleResize);
    document.title = this.state.name + ' ' + this.state.surname;
  }

  componentDidMount() {
    document.title = this.state.name + ' ' + this.state.surname;
  }

  componentWillUnmount() {
    window.removeEventListener("resize", this.handleResize);
  }

  handleNameChange(name) {
    this.setState({ name });
  }

  handleSurnameChange(surname) {
    this.setState({ surname });
  }

  handleResize() {
    this.setState({ width: window.innerWidth });
  }

  render() {
    let { name, surname, width } = this.state;
    return (
      <ThemeContext>
        {theme => (
          <Card theme={theme}>
            <Row label="Name">
              <Input value={name} onChange={this.handleNameChange} />
            </Row>
            <Row label="Surname">
              <Input value={surname} onChange={this.handleSurnameChange} />
            </Row>
            <Row label="Width">
              <Text>{width}</Text>
            </Row>
          </Card>
        )}
      </ThemeContext>
    );
  }
}
```

hooks

```
import React, { useState, useContext, useEffect } from "react";
import { Card, Row, Input, Text } from "../components";
import ThemeContext from "../ThemeContext";

export default function Greeting(props) {
  let theme = useContext(ThemeContext);

  let [name, setName] = useState("Harry");
  let [surname, setSurname] = useState("Potter");
  useEffect(() => {
    document.title = name + " " + surname;
  });

  let [width, setWidth] = useState(window.innerWidth);
  useEffect(() => {
    let handleResize = () => setWidth(window.innerWidth);
    window.addEventListener("resize", handleResize);
    return () => {
      window.removeEventListener("resize", handleResize);
    };
  });

  return (
    <Card theme={theme}>
      <Row label="Name">
        <Input value={name} onChange={setName} />
      </Row>
      <Row label="Surname">
        <Input value={surname} onChange={setSurname} />
      </Row>
      <Row label="Width">
        <Text>{width}</Text>
      </Row>
    </Card>
  );
}
```

O que é?

O que é? - O que exatamente é um hook? Como ele funciona?

São funções 'especiais' que adicionam capacidades às componentes função

```
// useState
const [state, setState] = useState(initialState);

// useEffect
useEffect(() => {
  assmbleStuff();

  return () => {
    disassembleStuff();
  }
})

// useContext
const value = useContext(CoolContext);

// Custom hook
const { t } = useTranslation();
```

```
// Exemplo de contador com hooks
export const Counter = () => {
  const [counter, setCounter] = useState(0);

  return (
    <div>
      <button onClick={() =>
        setCounter(old => old + 1)
      }> + </button>
      <div>{counter}</div>
      <button onClick={() =>
        setCounter(old => old - 1)
      }> - </button>
    </div>
  );
};
```


O que é? - O que exatamente é um hook? Como ele funciona?

As regras que devem ser seguidas para se usar um hook

Um hook deve:

- Ser usado em componentes função
- Ter um nome começando com “use” e uma letra maiúscula
- Ser chamado sempre na mesma ordem
 - Ser chamado no nível mais alto da função
 - Não estar contido em condicionais
 - Não estar contido em loops

O que é? - O que exatamente é um hook? Como ele funciona?

Como o React implementa os hooks?

```
1 // Originally written by @jamiebuilds
2
3 let hooks = null;
4
5 export function useHook() {
6   hooks.push(hookData);
7 }
8
9 function reactInternalRenderAComponentMethod(component) {
10   hooks = [];
11   component();
12   let hooksForThisComponent = hooks;
13   hooks = null;
14 }
```

“I’d say Hooks are about as much magic as calling `array.push` and `array.pop` (for which the call order matters too!)” - [Dan Abramov](#)

Recomendação: [“Deep dive: How do React hooks really work?”](#)

Como usar?

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Lista de hooks existentes na versão 16.8

- **Basic Hooks**
 - `useState`
 - `useEffect`
 - `useContext`
- **Additional Hooks**
 - `useReducer`
 - `useCallback`
 - `useMemo`
 - `useRef`
 - `useImperativeHandle`
 - `useLayoutEffect`
 - `useDebugValue`

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Como uso os seguintes hooks: `useState` - Adiciona estado local à componente função

```
// Assinatura da função useState
const [state: any, setState: function] = useState(initialState: any);
const [counter, setCounter] = useState(0);
const [expensiveCalculationValue, setExpensiveCalculation] = useState(() => allTheMaths(props));

// Passando um valor novo
setState(newState);

// Passando uma função que recebe estado antigo e retorna um novo estado
setState(oldState => {
  // Seu lindo código

  return newState;
})

// OBS: setState não faz junção de objetos de estado anteriores como o this.setState de componentes,
// você terá que fazer isso manualmente caso queira esse recurso
```

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Como uso os seguintes hooks: **useEffect** - Permite usar funções equivalentes ao ciclo de vida de um componente

```
// Função useEffect
useEffect(effect: function, dependencies: []);

// O use effect é equivalente à DidMount, DidUpdate e WillUnmount
useEffect(() => {
  // Faça aqui sua bagunça
  return () => {
    // Está na hora de limpar a bagunça
  }
},
// Mas só se as seguintes dependências permitirem
[dep1, dep2, ..., depN]
);

// OBS: O vetor de dependência pode ser:
// 'Nenhum valor' e o 'montar' efeito e 'desmontar' efeito serão executados a cada render
// [] (i.e. Vetor vazio) e o 'montar' e 'desmontar' serão executados somente uma vez
// [lista de deps] e o 'montar' e o 'demonstar' só serão executados quando alguma dependência for alterada
```

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Como uso os seguintes hooks: [useContext](#) - Permite consumir contextos de provedores acima na árvore

```
// Função useContext
const value = useContext(MyContext: ReactContext);

const theme = useContext(ThemeContext);

// Obs o useContext recebe o valor de um createContext e não o seu consumer
// Correto: useContext(MyContext)
// Incorreto: useContext(MyContext.Consumer)
// Incorreto: useContext(MyContext.Provider)

// Você ainda precisa ter um Provider na árvore acima deste componente
```

Como usar? - Quais hooks existem? Como uso? Como faço o meu?

Como criar hooks personalizados?

```
// Criando meu próprio hook
function useCustomHook () {
  const [value, setValue] = useState(window.innerWidth);

  function onResize(event) { setValue(window.innerWidth); }

  useEffect(() => {
    window.addEventListener("resize", onResize);
    return () => {
      window.removeEventListener("resize", onResize);
    }
  }, [])
  return value;
}

// Usando meu hook
const screenWidth = useCustomHook();
```


Codando...



<https://github.com/0tho/React-hooks-example>

Dúvidas?

Muito Obrigado!



Marcelo Lopes Lotufo
Freelance Software Developer

<https://marceloll.com/>