React Hooks

Marie Hooper | Feb 6, 2019

Hooks are an upcoming feature that let you make stateful logic reusable in a cleaner way.

Why hooks?

- Reusing stateful logic between components can get complicated
  - Mixins (ew)
  - Higher order components
  - Render props
- It can be difficult to decouple logic from implementation details
- Create more functional components than class-based components

Duplicate state logic

```
1  import React from 'react'
2  import getTopPopSongs from '../api/getTopPopSongs'
3  import SearchResults from './SongSearchResults'
4
5  export default class TopPopSongs extends React.Component {
6    state = {
7      value: '',
8      searchResults: {
9        items: [],
10       totalItems: 0
11     }
12   }
13
14   setValue = event => {
15     this.setState({
16       value: event.currentTarget.value,
17       searchResults: getTopPopSongs(event.currentTarget.value)
18     })
19   }
20
21   componentDidMount() {
22     this.setState(prevState => ({
23       searchResults: getTopPopSongs(prevState.value)
24     }))
25   }
26
27   render() {
28     const { searchResults, value } = this.state
29     return (
30       <div className="content">
31         <h1>Top Pop Songs</h1>
32         <input
33           className="shadow-input"
34           type="text"
35           placeholder="Search an artist or song name"
36           value={value}
37           onChange={this.setValue}
38         />
39         <br />
40         <SearchResults searchResults={searchResults} />
41       </div>
42     )
43   }
44 }
45
```

```
1  import React from "react";
2  import getTopPublicUniversities from "../api/getTopPublicUniversities";
3  import SearchResults from "../components/UniversitySearchResults";
4
5  export default class TopPublicUniversities extends React.Component {
6    state = {
7      value: "",
8      searchResults: {
9        items: [],
10       totalItems: 0
11     }
12   };
13
14   setValue = event => {
15     this.setState({
16       value: event.currentTarget.value,
17       searchResults: getTopPublicUniversities(event.currentTarget.value)
18     });
19   };
20
21   componentDidMount() {
22     this.setState(prevState => ({
23       searchResults: getTopPublicUniversities(prevState.value)
24     }));
25   }
26
27   render() {
28     const { searchResults, value } = this.state;
29     return (
30       <div className="content">
31         <h1>Top Public Universities</h1>
32         <input
33           type="text"
34           placeholder="Search a school name or location"
35           value={value}
36           onChange={this.setValue}
37         />
38         <br />
39         <SearchResults searchResults={searchResults} />
40       </div>
41     );
42   }
43 }
44
```

Duplicate state logic everywhere we want to use search

```
state = {
  value: "",
  searchResults: []
};

setValue = event => {
  this.setState({
    value: event.currentTarget.value,
    searchResults: getTopPublicUniversities(event.currentTarget.value)
  });
};

componentDidMount() {
  this.setState(prevState => ({
    searchResults: getTopPublicUniversities(prevState.value)
  }));
}
```

Higher order component

Allows us to create a function that encapsulates the search functionality and inject it into a given component.

Downsides:

- Hard to determine which props are coming from where and to avoid name conflicts
- Creates an additional component in your DOM tree
- Harder to test than standalone hooks

```
1  import React from "react";
2
3  export default function withSearch(WrappedComponent, search) {
4    return class WithSearch extends React.Component {
5      state = {
6        value: "",
7        searchResults: []
8      };
9
10     setValue = event => {
11       this.setState({
12         value: event.currentTarget.value,
13         searchResults: search(event.currentTarget.value)
14       });
15     };
16
17     componentDidMount() {
18       this.setState(prevState => ({
19         searchResults: search(prevState.value)
20       }));
21     }
22
23     render() {
24       return (
25         <WrappedComponent
26           {...this.props}
27           value={this.state.value}
28           setValue={this.setValue}
29           searchResults={this.state.searchResults}
30         />
31       );
32     }
33   };
34 }
```

Higher order component ‘withSearch’ in use

```
1  import React from "react";
2  import getTopPopSongs from "../api/getTopPopSongs";
3  import withSearch from "./withSearch";
4  import SearchResults from "../components/SongSearchResults";
5
6  function TopPopSongs({ value, setValue, searchResults }) {
7    return (
8      <div className="content">
9        <h1>Top Pop Songs</h1>
10       <input
11         className="shadow-input"
12         type="text"
13         placeholder="Search an artist or song name"
14         value={value}
15         onChange={setValue}
16       />
17       <br />
18       <SearchResults searchResults={searchResults} />
19     </div>
20   );
21 }
22
23 export default withSearch(TopPopSongs, getTopPopSongs);
```

component

search fn

Render Props

A component with a render prop takes a function that returns a React element and calls it instead of implementing its own render logic

Downsides:

- Although this is a more clear abstraction than HOC, it's still an abstraction.
- You are creating a new anonymous function which may be a performance concern eventually in a large app
- Triangle of Doom

```
1  import React from "react";
2
3  export default class Search extends React.Component {
4    state = {
5      value: "",
6      searchResults: {
7        items: [],
8        totalItems: 0
9      }
10   };
11
12   setValue = event => {
13     this.setState({
14       value: event.currentTarget.value,
15       searchResults: this.props.search(event.currentTarget.value)
16     });
17   };
18
19   componentDidMount() {
20     this.setState(prevState => ({
21       searchResults: this.props.search(prevState.value)
22     }));
23   }
24
25   render() {
26     return this.props.children({
27       value: this.state.value,
28       setValue: this.setValue,
29       searchResults: this.state.searchResults
30     });
31   }
32 }
33
```

Render prop ‘Search’ in use

```
1  import React from "react";
2  import getTopPopSongs from "../api/getTopPopSongs";
3  import Search from "./Search";
4  import SearchResults from "../components/SongSearchResults";
5
6  export default function TopPopSongs() {
7    return (
8      <Search search={getTopPopSongs}>
9        {({ value, setValue, searchResults }) => (
10         <div className="content">
11           <h1>Top Pop Songs</h1>
12           <input
13             className="shadow-input"
14             type="text"
15             placeholder="Search an artist or song name"
16             value={value}
17             onChange={setValue}
18           />
19           <br />
20           <SearchResults searchResults={searchResults} />
21         </div>
22       )}
23     </Search>
24   );
25 }
```

Hooks

- Allow us to write a whole app using just functional components instead of class-based components every time we need state
- Allow us to group related logic in effects rather than having to group and sometimes repeat functionality in lifecyle methods
- Custom hooks are just functions, so they are easy to test, reuse and understand

```
1  import React, { useEffect, useState } from "react";
2  import getTopPopSongs from "../api/getTopPopSongs";
3  import SearchResults from "../components/SongSearchResults";
4
5  export default function TopPopSongs() {
6    const [value, setValue] = useState("");
7    const [searchResults, setSearchResults] = useState({
8      items: [],
9      totalItems: 0
10   });
11
12   useEffect(
13     () => {
14       const results = getTopPopSongs(value);
15       setSearchResults(results);
16     },
17     [value]
18   );
19
20   return (
21     <div className="content">
22       <h1>Top Pop Songs</h1>
23       <input
24         className="shadow-input"
25         type="text"
26         placeholder="Search an artist or song name"
27         value={value}
28         onChange={event => {
29           setValue(event.currentTarget.value);
30         }}
31       />
32       <SearchResults searchResults={searchResults} />
33     </div>
34   );
35 }
36
```

Rules of Hooks

- Only call hooks at the top level (not inside a loop, function, or conditional statement)
- Only call hooks from React functions (not from regular JS functions)

Types of React Hooks

- State hooks (useState)
- Effect hooks (useEffect)
- Custom hooks (useSomeCoolThing)
- Other hooks (useContext, useRef, useReducer, etc.)

Class component versus function component with Hooks

```python
import React from 'react'
import getTopPopSongs from '../api/getTopPopSongs'
import SearchResults from './SongSearchResults'

export default class TopPopSongs extends React.Component {
  state = {
    value: '',
    searchResults: {
      items: [],
      totalItems: 0
    }
  }

  setValue = event => {
    this.setState({
      value: event.currentTarget.value,
      searchResults: getTopPopSongs(event.currentTarget.value)
    })
  }

  componentDidMount() {
    this.setState(prevState => ({
      searchResults: getTopPopSongs(prevState.value)
    }))
  }

  render() {
    const { searchResults, value } = this.state
    return (
      <div className="content">
        <h1>Top Pop Songs</h1>
        <input
          className="shadow-input"
          type="text"
          placeholder="Search an artist or song name"
          value={value}
          onChange={this.setValue}
        />
        <br />
        <SearchResults searchResults={searchResults} />
      </div>
    )
  }
}
```

```python
import React, { useEffect, useState } from "react";
import getTopPopSongs from "../api/getTopPopSongs";
import SearchResults from "../components/SongSearchResults";

export default function TopPopSongs() {
  const [value, setValue] = useState("");
  const [searchResults, setSearchResults] = useState({
    items: [],
    totalItems: 0
  });

  useEffect(
    () => {
      const results = getTopPopSongs(value);
      setSearchResults(results);
    },
    [value]
  );

  return (
    <div className="content">
      <h1>Top Pop Songs</h1>
      <input
        className="shadow-input"
        type="text"
        placeholder="Search an artist or song name"
        value={value}
        onChange={event => {
          setValue(event.currentTarget.value);
        }}
      />
      <SearchResults searchResults={searchResults} />
    </div>
  );
}
```

Custom Hooks

Allows us to combine hooks for reuse throughout different components.

Benefits:

- Decouple stateful logic from implementation details
- Reuse stateful logic across components

```
1 import { useEffect, useState } from "react";
2
3 export default function useSearch(search) {
4   const [value, setValue] = useState("");
5   const [searchResults, setSearchResults] = useState({
6     items: [],
7     totalItems: 0
8   });
9
10  useEffect(
11    () => {
12      const results = search(value);
13      setSearchResults(results);
14    },
15    [value]
16  );
17
18  return [value, setValue, searchResults];
19 }
20
```

Custom Hook useSearch() in use

```
1 import React from "react";
2 import getTopPopSongs from "../api/getTopPopSongs";
3 import useSearch from "./useSearch";
4 import SearchResults from "../components/SongSearchResults";
5
6 export default function TopPopSongs() {
7   const [value, setValue, searchResults] = useSearch(getTopPopSongs);
8
9   return (
10    <div className="content">
11      <h1>Top Pop Songs</h1>
12      <input
13        className="shadow-input"
14        type="text"
15        placeholder="Search an artist or song name"
16        value={value}
17        onChange={event => {
18          setValue(event.currentTarget.value);
19        }}
20      />
21      <br />
22      <SearchResults searchResults={searchResults} />
23    </div>
24  );
25 }
26
```

Resources 📚

- React docs on hooks are really good as a starting point: https://reactjs.org/docs/hooks-intro.html
- Dan Abramov does a really good job explaining hooks: https://medium.com/@dan_abramov/making-sense-of-react-hooks-fdbde8803889
- CSS Tricks has a good article too: https://css-tricks.com/intro-to-react-hooks/
- Some hook recipes to explore use cases for hooks: https://usehooks.com/
- Look, they're coming to Vue too!: https://css-tricks.com/what-hooks-mean-for-vue/
- For a deeper dive on Higher Order Components, Tyler McGinnis crushes it here: https://tylermcginnis.com/react-higher-order-components/
- Mixins being publicly shamed in React: https://reactjs.org/blog/2016/07/13/mixins-considered-harmful.html

One last example...

```
1  import { useState } from "react";
2
3  export default function useForm(initialValues) {
4    const [values, setValues] = useState(initialValues);
5
6    function mergeValue(key, value) {
7      setValues(prevValues => ({
8        ...prevValues,
9        [key]: value,
10     }));
11   }
12
13   function changeValue(event) {
14     const { checked, id, name, type, value } = event.currentTarget;
15     if (type === "checkbox") {
16       mergeValue(id, checked);
17     } else if (type === "radio") {
18       mergeValue(name, id);
19     } else {
20       mergeValue(id, value);
21     }
22   }
23
24   return [values, changeValue];
25 }
```

```
1  function SomeForm() {
2    const [values, setValue] = useForm({
3      name: "",
4    });
5
6    return (
7      <Label htmlFor="name">
8        Name
9        <Input id="name" value={values.name} onChange={setValue} />
10     </Label>
11   );
12 }
```

Hook for form data management! ✧