React Fragments – What, Why, How

#react #javascript #webdev

Tuomo Kankaanpää 25 Oct 2019 · *Updated on 10 Feb* · 4 min read

React Fragments were introduced in React 16.2.0. Even though they have been around for

a while now, many React developers I have talked to haven't started to use them yet and

the number one reason for this usually is that they have heard of them but haven't got around to learn about them.

So what are they and should you start using them? The answer to the latter is YES and the answer for the first question is what this blog post is going to tell you.

Are you new to React? Be sure to read my post [6 Things Every Beginner React Developer Should Know](#).

Problem

As the React.js docs state, a common pattern in React is for components to return multiple elements. Usually these elements are wrapped for example inside a div. In most cases the wrapper div is "irrelevant" and is only added because React components require you to return only one element. This kind of behaviour results in useless markup and sometimes even invalid HTML to be rendered, which is bad.

For example we could have a component Table that renders an HTML table and inside that table the columns are rendered with another component called Columns. It would probably look something like this.

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
```

```
    }
  }
```

This would result in an invalid HTML to be rendered because the wrapper div from Columns component is rendered inside the `<tr>`.

```
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Solution

Solution for this is, you guessed it, fragments! React fragments let you group a list of children without adding extra nodes to the DOM because fragments are not rendered to the DOM. So basically we use React.Fragment where we would normally use a wrapper div.

We can make use of fragments with `<React.Fragments>` syntax. So we could write the Columns component as follows.

```
class Columns extends React.Component {
  render() {
    return (
      <React.Fragment>
        <td>Hello</td>
        <td>World</td>
      </React.Fragment>
    );
  }
}
```

Now the Table component would render following HTML.

```
<table>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
```

```
  </table>
```

Fragments can also be declared with a short syntax which looks like an empty tag. Here is an example.

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

Typical use cases

Return multiple elements

Most common use case for React fragments is probably when you need to return multiple elements. With fragments this is easy and you don't need your typical wrapper div for the elements.

```
class Application extends React.Component {
  render() {
    return (
      <React.Fragment>
        <Header />
        <Content />
        <Footer />
      </React.Fragment>
    );
  }
}
```

Conditional rendering

React fragments can also be used when conditionally rendering elements. They make rendering groups of elements a lot easier without adding extra markup.

```
class LoginForm extends React.Component {
  render() {
```

```
    render() {
      return (
        <form>
          {this.props.isLoggedIn ? (
              <React.Fragment>
                <h3>Welcome</h3>
                <p>You are logged in!</p>
              </React.Fragment>
          ) : (
              <React.Fragment>
                <h3>Login</h3>
                <label for="username">Username</label><br/>
                <input type="text" id="username" /><br/>
                <label for="password">Password</label><br/>
                <input type="password" id="password" /><br/>
                <input type="submit" value="Login" />
              </React.Fragment>
          )}
        </form>
      );
    }
  }
```

Arrays

Fragments can also help us when rendering arrays, because fragments can have key props! Let's say you have an array of user objects and you want to render all users from the array. You need to set key prop for every user so you would need to use element like div to wrap the user info. But because fragments can have key prop, you can instead use fragments and give the key prop for them and thus you don't need to introduce any extra markup.

```
class UserList extends React.Component {
  users = [
    {
      id: 1,
      name: "Jack Bauer",
      email: "jack.bauer@ctu.gov",
      phone: "+358509283928"
    },
    {
      id: 2,
      name: "Tony Almeida",
      email: "tony.almeida@ctu.gov",
      phone: "+358508829378"
```

```
    },
    {
      id: 3,
      name: "Chloe O'brian",
      email: "chloe.obrian@ctu.gov",
      phone: "+358508899012"
    }
  ];
  render() {
    return (
      <React.Fragment>
        {this.users.map(user => (
          <React.Fragment key={user.id}>
            <h2>{user.name}</h2>
            <p>{user.email}</p>
            <p>{user.phone}</p>
          </React.Fragment>
        ))}
      </React.Fragment>
    );
  }
}
```

Should I use fragments?

So are fragments worth using instead of say a wrapper div?

Dan Abramov answered this question on StackOverflow:

- It's a tiny bit faster and has less memory usage (no need to create an extra DOM node). This only has a real benefit on very large and/or deep trees, but application performance often suffers from death by a thousand cuts. This is one cut less.
- Some CSS mechanisms like Flexbox and CSS Grid have a special parent-child relationship, and adding divs in the middle makes it hard to keep the desired layout while extracting logical components.
- The DOM inspector is less cluttered. 🙂

So the fact that fragments eliminate the wrapper div which can cause problems with invalid HTML and with styling of the components plus the fact that they are faster and the DOM is less cluttered, I'd say they are worth using.

Wrapping up

What do you think about React fragments and are you using them in your own projects? Please leave a comment below, I would love to hear from you!

Also remember to subscribe to my newsletter, to stay tuned on the latest news and posts about modern web development. I also share exclusive tips and tricks for the newsletter subscribers! You can subscribe here.

Originally published at tuomokankaanpaa.com on October 6, 2019.

Discussion (8)

Emma Goto • Oct 25 '19

I've been using fragments, but I didn't know you could pass in keys! Thanks for the tip!

Kiel • Oct 31 '19

Yes the addition of passing keys is a gamechanger... not sure if it was always there but will definitely utilize that as a standard...

i love fragments because <> and </> are quick to type, and they don't render in the dom tree so i don't have to worry about broken CSS anywhere

M. Scott Ford • Apr 14

I stumbled across this post after reading the React.Fragment documentation. I found the similarities a little bit eerie. Should the documentation give credit to this article or should this article give credit to the documentation?

Either way, I found the additional information in this article helpful. Thank you!

Tuomo Kankaanpää • May 4

I studied fragments through the official documentation + experimenting on my own, so if

those are the options I guess this article should give credit to the documentation.

Glad to hear that you liked it!

Oscar Quinteros • Oct 15 '20

on your example below how does react know to put a row instaead of a div element?

class Columns extends React.Component {
render() {
return (

Hello
World

);
}
}

Tuomo Kankaanpää • Oct 17 '20

I'm not sure if I understood correctly your question, but what I think you are looking for is that the row (tr) is inserted by the parent component and the Columns component only returns the td elements.

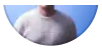
Tomer Raitz • Oct 12 '20

Great Article! Thank you

emreozgun10 • Sep 3

Great insight!

Code of Conduct • Report abuse



Tuomo Kankaanpää

Full-stack web developer programming Javascript, React and PHP. I have a YouTube channel.

LOCATION
Finland

WORK
Senior Software Developer at Jakamo

JOINED
22 Jun 2018

More from Tuomo Kankaanpää

Why I like Software Engineering 🧡

#webdev #programming #career #books

Thoughts on Github Copilot 🤖

#github #webdev #review #books

Working with React, Monorepo, Typescript and Lerna 👷

#webdev #react #typescript #monorepo