Flutter In Action: React : Tips & Tricks : Part 1

Posted by Khushboo Uchat Feb 9, 2021

My journey of React programming started before three years and as I was coming from hard core Angular background (**Still I am a big fan of Angular so no bad words for Angular please !!!**) , it was not a piece of Cake for me. So I thought to share some tips for newbies of React who might be facing similar challenges I faced in the beginning (and hesitating to ask silly things to senior developers). This blog might solve lots of silly questions about react syntax and ways of programming preferred in React. Before proceeding further I am assuming that you are well aware of React fundamentals.

I have divided my tips in two parts. Coding & Performance. In this part, I will share coding tips

Coding Tips:

Avoid binding event handlers manually and make habit of writing arrow functions

You must have seen lots of react examples with code like below.

Avoid

```
constructor(props) {
  // some code
  this.handleAdd = this.handleAdd.bind(this);
  this.handleSubtract = this.handleSubtract.bind(this);
}
```

Technically nothing is wrong with the above code but it becomes tedious process when you are developing large application in React and lots of event handlers are present in same component. Instead we can use arrow function syntax which will take the context automatically of parent class and saves us from this boilerplate code.

Correct Code

```
handleAdd = (event) => {
// your code
}
```

Avoid using lots of event handlers for different each input box with html forms

If you are still working with html forms in react (rather than reactive forms !!) then I believe that you must be adding event handlers for inputs to capture the values. We can have a single handler to capture the changes with unique name for each input type.

Avoid

```
<input
  type="text"
  name="number1"
  placeholder="Enter a number"
  onChange={this.onFirstInputChange}
/>
<input
  type="text"
  name="number2"
  placeholder="Enter a number"
  onChange={this.onSecondInputChange}
/>
```

Correct

<input type="text" name="number1" placeholder="Enter a number" onChange={this.onInputChange}/><input type="text" name="number2" placeholder="Enter a number" onChange={this.onInputChange}/>

```
onInputChange = (event) => {
  const name = event.target.name;
  const value = event.target.value;
  this.setState({
   [name]: value
  });
};
```

Use ES6 de structuring for better readability

Though we can have the code without es6 syntax but when your object grows then extracting the specific properties from object will be so helpful and make the code more readable.

Avoid

```
const name = event.target.name;
const value = event.target.value;
```

Correct

```
const { name, value } = event.target;
```

Use implicit return when returning objects

When returning an object from the function, we usually write like below.
Avoid

```
const getUser = () => {
return {
  name: 'XYZ',
  age: 24
}
}
```

But we can wrap it in small bracket to make it consistent with arrow function.

Correct

```
const getUser = () => ({
  name: 'XYZ',
  age: 24
})
```

The above option is used lots of times when writing code especially with Redux like action creators or mapStateToProps. It will save you lots of false typing as well as better consistency with arrow functions

Omitting prop value while passing will be boolean true by default

When we pass the property to child then we pass the value with name of the property. If we do not pass the value then the value is true by default. Below both are same.

`<Search autocomplete /> and <Search autocomplete={true} />`

React Fragment has shorthand operator

```
// Long
<React.Fragment></React.Fragment>
// short
<></>
```

If you are not using props in your constructor then you do not need super() in your constructor

Due to examples copied or code copied we all have seen super() method in constructor but not really sure why it is there. You can remove it if you are not using props inside your constructor.

Do not write inline functions in JSX if it involves business logic even if it is one line for better readability

React developers are lured to use inline function especially event handlers but if it involves some kind of operations which is logical then you can keep it out of JSX to make the code easily searchable and readable. *JSX itself is so complicated to understand for anyone who looks at it first time and finding the function inside this another headache (especaily when we are fixing the bugs created by others which is quite common).*

Avoid

```
<input onClick={event => {  // Some logic here}}/>
```

```
Correcthandler = event => {  // Some logic here}<input onClick={handler}/>
```

Keep your components small and prop names meaningful for easy debugging and understanding

When we have large components then it is quite common that we lost in the code while debugging. It can be worst if the code is not written by you as you are not able to make head or tail out of it (If you find some comments then you are the luckiest person in the world of IT in that situation). To deal with this kind of situations, we can always focus on basic norms so that later on when we look at the code it is easily understandable.

In next part, I will talk about performance tips that helps your react application to perform well
43 Views Tags: react, react.js, #technology, tips and tricks

Amreshwar Rathi
Feb 22, 2021 2:52 PM
Being started react journey in last few months , this article really helped me to take baby steps in react world . Thank you Khushboo !!

Rohit Chandiramani
Feb 10, 2021 7:24 AM
Great article! Being new to React myself, these doubts always linger. This really helped to understand the concepts in a simple and lucid way. Please keep these articles coming our way!