Best React Usages in 2023

React has a lot of uses. Here are the best React practices you should follow in 2023 .

React is one of the most popular front-end frameworks for JavaScript. Unlike frameworks like Angular, it's very simple. So, how you write or structure your React code is up to you.

TipsMake.com will summarize for you the best React practices that you should follow to improve the results of a developing project.

Use functional components and hooks instead of classes

In React, you can use classes or functional components with hooks. However, you should use functional components and hooks more often because they provide more precise and readable code.

Consider the class component that displays data from the following NASA API:

```
class NasaData extends React.Component { ??constructor(props) { ????
super(props); ????this.state = { ??????data: [], ????}; ??} ??
componentDidMount() { ????
fetch("https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY") ??????
.then((res) => res.json()) ??????.then((json) => { ????????this.setState({
??????????data: json, ????????}); ??????}); ??} ??render() { ????
const { data } = this.state; ????if (!data.length) ??????return ( ????????
??????????
```

Fetching data.

```
{" "} ???????? ??????); ????return ( ??????> ????????
```

Fetch data using Class components

```
{" "} ????????{data.map((item) => ( ??????????{item.title} ????????))}
?????? ????); ??} }
```

You can rewrite the above code with hooks:

```
const NasaData = () => { ??const [data, setdata] = useState(null); ??
useEffect(() => { ????
```

```
fetch("https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY") ??????
.then((res) => res.json()) ??????.then((json) => { ????????setdata(json);
??????}); ??}, [data]); ??return ( ????> ??????
```

Fetch data using Class components

```
{" "} ??????{data.map((item) => ( ????????{item.title} ??????))} ???? ??
); };
```

Although the above code block is like code written in class, it is simpler, less complicated and easier to understand.

Avoid using state if possible

State in React keeps track of data that, when changed, triggers the React component to re-render. When building a React app, avoid using state as much as possible because the more state you use, the more data you have to track on the app.

One way to minimize the use of state is to declare it only when needed. For example, if you are fetching user data from an API, store the user object in state instead of fiving the individual properties.

Instead of writing:

```
const [username, setusername] = useState('') const [password, setpassword] = useS
```

Let's code like this:

```
const [user, setuser] = useState({})
```

Sort files related to the same element in a folder

When deciding on a directory structure for your React app, choose a component-centric structure. This means all related files will be stored in one directory.

For example, if making a Navbar component, create a Navbar folder containing the component files, style sheets, and other JavaScript, and the asset files used in the component.

A single directory containing all component files makes it easy to reuse, share, and debug. If you need to see how a component works, you only have to open a folder.

Avoid using index as key prop

React uses keys to uniquely identify items in an array. With the key, React can identify the item that has been changed, added, or removed from the array.

When displaying an array, you can use the index as the key.

```
const Items = () => { ??
const arr = ["item1", "item2", "item3", "item4", "item5"]; ??return ( ????
> ??????{arr.map((elem, index) => { ????????
```

1. {elem}

```
; ??????})} ???? ??); };
```

Although sometimes the above works, using the index as the key can cause errors, especially when it is necessary to change the list. Try making a list like this:

```
const arr = ["item1", "item2", "item3", "item4", "item5"];
```

Currently, the first list item 'Item1' is at index 0, but if another item has been added to the beginning of the list, index 'Item1' will change to 1. This changes the behavior of the array.

The solution here lfa uses a unique value as an index to ensure that the identity of the list item is maintained.

Choose fragment instead of div if possible

The React component needs to return code encapsulated in a tag usually or a React. You should choose to fragment where possible.

Use increased DOM size, especially on large projects because the more tags or DOM nodes you have, the more memory your web needs and the more RAM the browser 'consumes' to load the web. This leads to slow page loading, frustrating for users.

An example of removing unnecessary tags. Here, you do not use this tag when returning a single element.

```
const Button = () => { ??return Display; };
```

These are **the best ways to use React in 2023** . How do you think you should use React for programming, please share with TipsMake.com!