CLEVERSE

REACT HOOKS 101

Aikdanai Sidhikosol
Software Engineer @Cleverse

CLEVERSE

What, why and how?

CLEVERSE

What, why and how**?**

CLEVERSE

What are Hooks?

What are Hooks?

CLEVERSE

What are Hooks?

🤔?

What are Hooks?

What are Hooks? 🎣

They are the new upcoming React feature that lets you use state and other React features (such as lifecycle) **without writing a class.**

CLEVERSE

What, **why** and how**?**

CLEVERSE

Why do we need Hooks?

Why do we use React?

😎 Reusable components

💡 Sensible data flow

👨🏻‍💻 It’s fun to code!

Problems with writing React

- 😭 **Reusing stateful logic**
- 🐳 **Huge component**
- 🔥 **Class is confusing**
  (for both humans and machines)

CLEVERSE

```
class Cat extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      id: 0,
      cat: null,
      loading: false,
      width: window.innerWidth
    };
  }

  componentDidMount() {
    this.fetch(this.state.id);
    window.addEventListener("resize", this.handleWindowResize.bind(this));
  }

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

  async fetch(id) {
    this.setState({ loading: true });
    const cat = await fetchCat(id);
    this.setState({ cat, loading: false });
  }
```

React Class Component

CLEVERSE

React Class Component

```jsx
class Cat extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      id: 0,
      cat: null,
      loading: false,
      width: window.innerWidth
    };
  }

  componentDidMount() {
    this.fetch(this.state.id);
    window.addEventListener("resize", this.handleWindowResize.bind(this));
  }

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

  async fetch(id) {
    this.setState({ loading: true });
    const cat = await fetchCat(id);
    this.setState({ cat, loading: false });
  }
```

Might have to duplicate this in some other components 🤔


```jsx
async fetch(id) {
  this.setState({ loading: true });
  const cat = await fetchCat(id);
  this.setState({ cat, loading: false });
}
```

React Class Component

```
class Cat extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      id: 0,
      cat: null,
      loading: false,
      width: window.innerWidth
    };
  }

  componentDidMount() {
    this.fetch(this.state.id);
    window.addEventListener("resize", this.handleWindowResize.bind(this));
  }

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

  async fetch(id) {
    this.setState({ loading: true });
    const cat = await fetchCat(id);
    this.setState({ cat, loading: false });
  }

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

  handleIDChange(e) {
    this.setState({
      id: e.target.value
    });
    this.fetch(e.target.value);
  }
```

Code split all over the place 😰

React Class Component

```jsx
class Cat extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      id: 0,
      cat: null,
      loading: false,
      width: window.innerWidth
    };
  }

  componentDidMount() {
    this.fetch(this.state.id);
    window.addEventListener("resize", this.handleWindowResize.bind(this));
  }

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

  async fetch(id) {
    this.setState({ loading: true });
    const cat = await fetchCat(id);
    this.setState({ cat, loading: false });
  }

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

  handleIDChange(e) {
    this.setState({
      id: e.target.value
    });
    this.fetch(e.target.value);
  }
```

```
.bind(this)
```

No.1 most forgot piece of code 😭

CLEVERSE

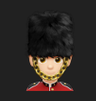
Problems with writing React

- 😭 **Reusing stateful logic**
- 🐳 **Huge component**
- 🔥 **Class is confusing**
  (for both humans and machines)

CLEVERSE

How we deal with them

Mixins (The old way)



Higher-order components

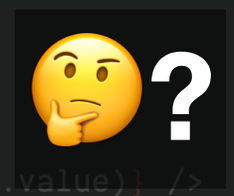
Render Props

HOCs

```
function Cat(props) {
  const { cat, loading, width, id, setID } = props;
  return (
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
      )}
    </div>
  );
}

export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

HOCs

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
```

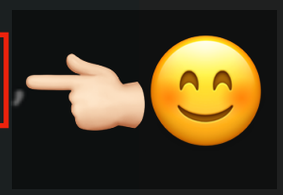


```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

CLEVERSE

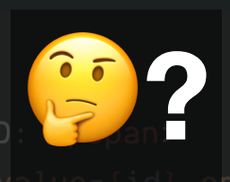
HOCs

```
function Cat(props) {
  const { cat, loading, width, id, setID } = props;

  return (
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
      )}
    </div>
  );
}

export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```



HOCs

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;
    <div>
      <section>
        <span>ID:</span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
```




```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

CLEVERSE

HOCs

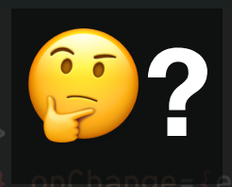
```
function Cat(props) {
  const { cat, loading, width, id, setID } = props;

  return (
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
      )}
    </div>
  );
}

export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

👈🤨 Maybe?

CLEVERSE

HOCs

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
```

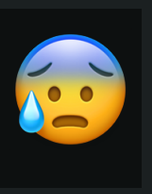


```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

CLEVERSE

HOCs

```
function Cat(props) {
  const { cat, loading, width, id, setID } = props;

  return (
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
      )}
    </div>
  );
}

export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```



HOCs

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
```

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withSubscribeCatData,
  withCatData,
  withDetectScroller,
  withAnimatedLoading,
  withAuth,
  withTracking,
  withUserAgentDetect,
  withYouForever
)(Cat);
```

🤯???

CLEVERSE

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withSubscribeCatData,
  withCatData,
  withDetectScroller,
  withAnimatedLoading,
  withAuth,
  withTracking,
  withUserAgentDetect,
  withYouForever
)(Cat);
```

HOCs

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withSubscribeCatData,
  withCatData,
  withDetectScroll
  withAnimatedLoad
  withAuth,
  withTracking,
  withUserAgentDet
  withYouForever
)(Cat);
```

HOCs

HOCs

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
```

Elements | React | Console | Sources | Network | Performance | Memory | Application | Security | Audits | AdBlock

Elements | Profiler

Search (text or /regex/)

```
▼ <Root>
  ▼ <main>
      <h1>Cat assistant (=^._.^=)∫</h1> == $r
    ▼ <withState(WindowWidth)>
      ▼ <WindowWidth id={0}>
        ▼ <SubscribeCatData id={0} width={375}>
          ▼ <CatData id={0} width={375}>
            ▼ <DetectScroller id={0} width={375} loading={false}>
              ▼ <AnimatedLoading id={0} width={375} loading={false}>
                ▼ <Auth id={0} width={375} loading={false}>
                  ▼ <Tracking id={0} width={375} loading={false}>
                    ▼ <UserAgentDetect id={0} width={375} loading={false}>
                      ▼ <YouForever id={0} width={375} loading={false}>
                        ▼ <Cat id={0} width={375} loading={false}>
                          ▼ <div>
                            ▼ <section>
                                <span>ID:</span>
                                <input value={0}></input>
                              </section>
                            ▼ <section>
                                <img src="https://ichef.bbci.co.uk/images/ic/720x405/p0517py…" alt="Mr. Bloop"></img>
                              ▼ <h2>
                                  "Mr. Bloop"
                                  ": Window size is "
                                  "375"
                                  "px!"
                                </h2>
                              </section>
                            </div>
                          </Cat>
                        </YouForever>
                      </UserAgentDetect>
                    </Tracking>
                  </Auth>
                </AnimatedLoading>
              </DetectScroller>
            </CatData>
          </SubscribeCatData>
        </WindowWidth>
      </withState(WindowWidth)>
  </main>
</Root>
```

CLEVERSE

HOCs

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
```

Elements | React | Console | Sources | Network | Performance | Memory | Application | Security | Audits | AdBlock

Elements | Profiler

Search (text or /regex/)

```
▼ <Root>
  ▼ <main>
      <h1>Cat assistant (=^._.^=)∫</h1> == $r
    ▼ <withState(WindowWidth)>
      ▼ <WindowWidth id={0}>
        ▼ <SubscribeCatData id={0} width={375}>
          ▼ <CatData id={0} width={375}>
            ▼ <DetectScroller id={0} width={375} loading={false}>
              ▼ <AnimatedLoading id={0} width={375} loading={false}>
                ▼ <Auth id={0} width={375} loading={false}>
                  ▼ <Tracking id={0} width={375} loading={false}>
                    ▼ <UserAgentDetect id={0} width={375} loading={false}>
                      ▼ <YouForever id={0} width={375} loading={false}>
                        ▼ <Cat id={0} width={375} loading={false}
                          ▼ <div>
                            ▼ <section>
                                <span>
                                  put value={0}></input>
                              </section>
                            ▼ <section>
                                <img s  ps:/  f  i.  ges  720x40  51  alt="  Bloop"></img>
                                >
                                Mr.  o
                                : W  w  i
                                75
                                !"
                              </
                              ec  n>
                            </div>
                          </Cat>
                        </YouForever
                      </User  tect>
                    ring>
                  </Auth>
                </AnimatedLoading>
              </DetectScroller>
            </CatData>
          </SubscribeCatData>
        </WindowWidth>
      </withState(WindowWidth)>
    </main>
</Root>
```

WRAPPER HELL

CLEVERSE

CLEVERSE

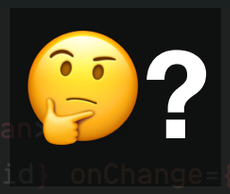
	HOCs	Render Props	Hooks 🎣
Reuse stateful logic	✅		
Explicit Props	❌	🤔?	🤔?
Composability	✅		
No “Wrapper Hell”	❌		

```jsx
function Cat(props) {
  const { cat, loading, width, id, setID } = props;
  return (
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
          <h2>
            {cat.name}: Window size is {width}px!
          </h2>
        </section>
      ) : (
        <div>No cat :(</div>
      )}
    </div>
  );
}

export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <Cat {...props} id={id} setID={setID} cat={cat} loading={loading} width={width} />
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```

Render Props

CLEVERSE

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;

    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
```



Render Props

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <Cat {...props} id={id} setID={setID} cat={cat} loading={loading} width={width} />
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```

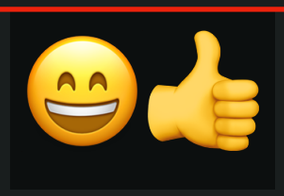
```
    )}
  </ValueHandler>
);
```

CLEVERSE

```
function Cat(props) {
const { cat, loading, width, id, setID } = props;
    <div>
      <section>
        <span>ID: </span>
        <input value={id} onChange={e => setID(e.target.value)} />
      </section>
      {loading ? (
        <section>
          <div>loading...</div>
        </section>
      ) : cat ? (
        <section>
          <img src={cat.pictureUrl} alt={cat.name} />
```

Render Props

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <Cat {...props} id={id} setID={setID} cat={cat} loading={loading} width={width} />
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```



```
    )}
  </ValueHandler>
);
```

CLEVERSE

```jsx
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <SubscribeCatData>
                {() => (
                  <DetectScroller>
                    {() => (
                      <AnimatedLoading>
                        {() => (
                          <Auth>
                            {() => (
                              <Tracking>
                                {() => (
                                  <UserAgentDetect>
                                    {() => (
                                      <Cat
                                        {...props}
                                        id={id}
                                        setID={setID}
                                        cat={cat}
                                        loading={loading}
                                        width={width}
                                      />
                                    )}
                                  </UserAgentDetect>
                                )}
                              </Tracking>
                            )}
                          </Auth>
                        )}
                      </AnimatedLoading>
                    )}
                  </DetectScroller>
                )}
              </SubscribeCatData>
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```

Render Props

CLEVERSE

Render Props

```jsx
export default props => (
  <Composer
    components={[
      <ValueHandler initialValue={0} />,
      ({ results: [valueHandler], render }) => (
        <CatData id={valueHandler.value} children={render} />
      ),
      <WindowWidth />,
      <SubscribeCatData />,
      <DetectScroller />,
      <AnimatedLoading />,
      <Auth />,
      <Tracking />,
      <UserAgentDetect />
    ]}
  >
    {([valueHandler, catData, windowWidth, ...args]) => (
      <Cat
        id={valueHandler.value}
        setID={valueHandler.setValue}
        cat={catData.cat}
        loading={catData.loading}
        width={windowWidth.width}
      />
    )}
  </Composer>
);
```

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <SubscribeCatData>
                {() => (
```

Elements | React | Console | Sources | Network | Performance | Memory | Application | Security | Audits | AdBlock

Elements | Profiler

Search (text or /regex/)

```
▼ <Root>
  ▼ <main>
      <h1>Cat assistant (=^._.^=)∫</h1> == $r
    ▼ <_default>
      ▼ <ValueHandler initialValue={0}>
        ▼ <CatData id={0}>
          ▼ <WindowWidth>
            ▼ <SubscribeCatData>
              ▼ <DetectScroller>
                ▼ <AnimatedLoading>
                  ▼ <Auth>
                    ▼ <Tracking>
                      ▼ <UserAgentDetect>
                        ▼ <Cat id={0} loading={false} width={375}>
                          ▼ <div>
                            ▼ <section>
                                <span>ID:</span>
                                <input value={0}></input>
                              </section>
                            ▼ <section>
                                <img src="https://ichef.bbci.co.uk/images/ic/720x405/p0517py…" alt="Mr. Bloop"></img>
                              ▼ <h2>
                                  "Mr. Bloop"
                                  ": Window size is "
                                  "375"
                                  "px!"
                                </h2>
                              </section>
                            </div>
                          </Cat>
                        </UserAgentDetect>
                      </Tracking>
                    </Auth>
                  </AnimatedLoading>
                </DetectScroller>
              </SubscribeCatData>
            </WindowWidth>
          </CatData>
        </ValueHandler>
      </_default>
  </main>
</Root>
```

```
);
```

Render Props

CLEVERSE

Render Props

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <SubscribeCatData>
                {() => (
```

Elements | React | Console | Sources | Network | Performance | Memory | Application | Security | Audits | AdBlock

Elements | Profiler

Search (text or /regex/)

```
▼ <Root>
  ▼ <main>
      <h1>Cat assistant (=^._.^=)∫</h1> == $r
    ▼ <_default>
      ▼ <ValueHandler initialValue={0}>
        ▼ <CatData id={0}>
          ▼ <WindowWidth>
            ▼ <SubscribeCatData>
              ▼ <DetectScroller>
                ▼ <AnimatedLoading>
                  ▼ <Auth>
                    ▼ <Tracking>
                      ▼ <UserAgentDetect>
                        ▼ <Cat id={0} loading={false} wid…
                          ▼ <div>
                            ▼ <section…
                                …ID:</span>
                                <input value={0}></i…
                              </section>
                            ▼ <sec…
                                …c="…s…che…uk…es/ic/7…5i…" alt="Mr. Bloop"></img>
                                2…
                                "…Blo…
                                …low…
                                …37…
                                …x!…
                              …</section>
                            </div>
                          </Cat>
                        </User…tect>
                      …king>
                    </Auth>
                  </AnimatedLoading>
                </DetectScroller>
              </SubscribeCatData>
            </WindowWidth>
          </CatData>
        </ValueHandler>
      </_default>
    </main>
</Root>
```

```
                );
```

WRAPPER HELL

CLEVERSE

CLEVERSE

	HOCs	Render Props	Hooks 🎣
Reuse stateful logic	✅	✅	
Explicit Props	❌	✅	🤔?
Composability	✅	❌	
No “Wrapper Hell”	❌	❌	

CLEVERSE

SPOILER ALERT

	HOCs	Render Props	Hooks 🎣
Reuse stateful logic	✅	✅	✅
Explicit Props	❌	✅	✅
Composability	✅	❌	✅
No “Wrapper Hell”	❌	❌	✅

CLEVERSE

What, why **and how?**

HOCs

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

Render Props

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <Cat {...props} id={id} setID={setID} cat={cat} loading={loading} width={width
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```

Hooks 😜

CLEVERSE

CLEVERSE

Let’s code.

HOCs

```
export default compose(
  withState("id", "setID", 0),
  withWindowWidth,
  withCatData
)(Cat);
```

Render Props

```
export default props => (
  <ValueHandler initialValue={0}>
    {({ value: id, setValue: setID }) => (
      <CatData id={id}>
        {({ cat, loading }) => (
          <WindowWidth>
            {({ width }) => (
              <Cat {...props} id={id} setID={setID} cat={cat} loading={loading} width={width
            )}
          </WindowWidth>
        )}
      </CatData>
    )}
  </ValueHandler>
);
```

Hooks

```
const { value: id, onChange: setID } = useInput(0);
const { cat, loading } = useCatData(id);
const width = useWindowWidth();
```

CLEVERSE

CLEVERSE

	HOCs	Render Props	Hooks 🎣
Reuse stateful logic	✅	✅	✅
Explicit Props	❌	✅	✅
Composability	✅	❌	✅
No “Wrapper Hell”	❌	❌	✅

CLEVERSE

But nothing is perfect.

(except you, you all are perfect in your own way)

Rules of Hooks

1️⃣ **Only call at the Top Level**

Not under any conditions, loops or nested functions.

2️⃣ **Only call from React functional components**

Not from regular JS functions, or class components.
But you can call them from custom Hooks.

CLEVERSE

About Hooks

⭕ No breaking change

✅ Class still works

🎉 But it will cover all use cases for classes!

Hooks API 🎣

useState ✅
useEffect ✅
useContext
useReducer
useCallback
useMemo
useRef
useImperativeHandle
useLayoutEffect
useDebugValue

https://reactjs.org/docs/hooks-reference.html

How to try Hooks

```
npm install react@16.8.0-alpha.1
```

```
npm install react-dom@16.8.0-alpha.1
```

Recommended

```
npm install eslint-plugin-react-hooks@next --save-dev
```

https://www.npmjs.com/package/eslint-plugin-react-hooks

My code example

https://codesandbox.io/s/03r3y8wxqn

CLEVERSE

CLEVERSE

REACT HOOKS 101

Aikdanai Sidhikosol
Software Engineer @Cleverse

CLEVERSE

REACT HOOKS 101

Aikdanai Sidhikosol
Software Engineer @Cleverse
https://steamcommunity.com/id/blutarche/

CLEVERSE IS HIRING

CLEVERSE

Full-stack Developer

CLEVERSE

REACT HOOKS 101

Aikdanai Sidhikosol
Software Engineer @Cleverse