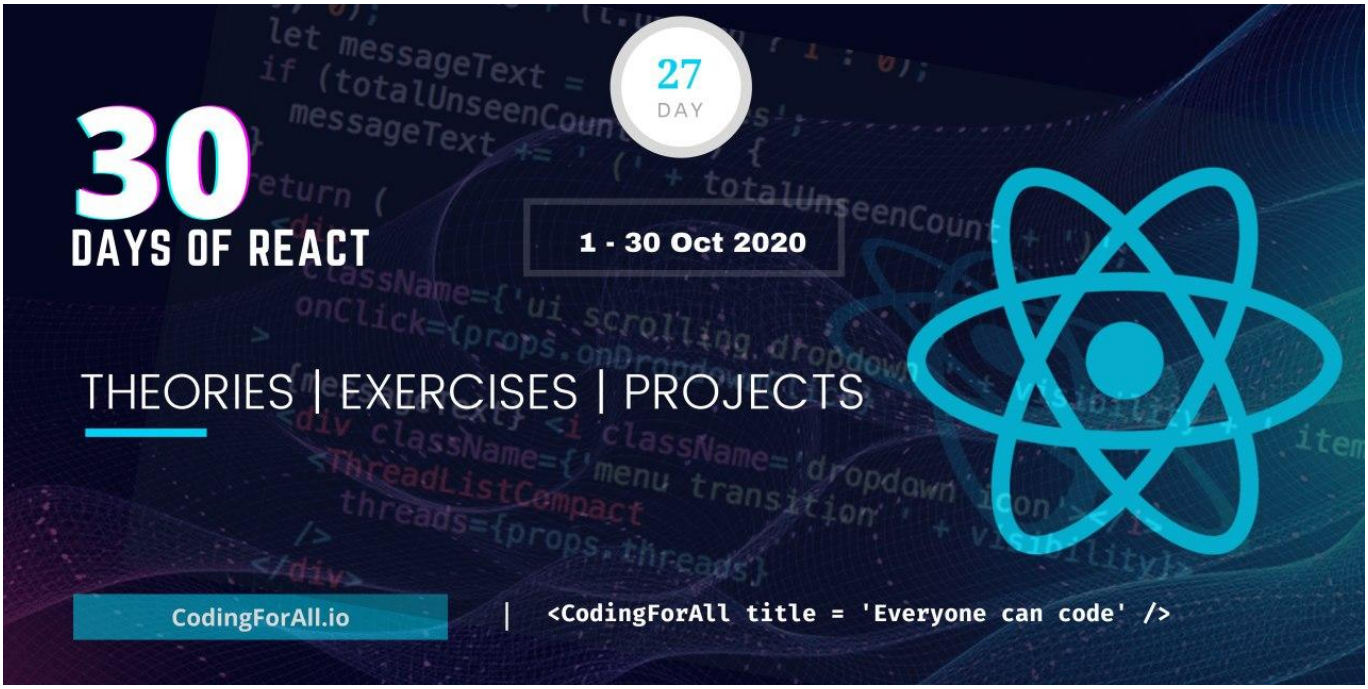
30 Days Of React: useRef

Author: Asabeneh Yetayeh

October, 2020

<< Day 25 | Day 27>>



useRef

In this challenge we have covered, how to handle uncontrolled input data. In this section, we will use the useRef hooks to get input data or to access any DOM element in your React application.

The useRef returns a mutable ref object whose .current property is initialized to the passed argument (initialValue). The returned object will persist for the full lifetime of the component.

In the following example, we see how to get data from input and access elements from the DOM tree using useRef hook.

Getting Data from input

Let's get data from uncontrolled input element.

```
import React, { useRef } from 'react'
import ReactDOM from 'react-dom'

const App = (props) => {
  const ref = useRef(null)
```

```
  const onClick = () => {
    let value = ref.current.value
    alert(value)
  }
  return (
    <div className='App'>
      <h1>How to use data from uncontrolled input using useRef</h1>
      <input type='text' ref={ref} />
      <br />
      <button onClick={onClick}>Get Input Data</button>
    </div>
  )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```

Focus

Using the useRef we can trigger the focus event on input.

```
import React, { useRef } from 'react'
import ReactDOM from 'react-dom'

const App = (props) => {
  const ref = useRef(null)
  const onClick = () => {
    ref.current.focus()
  }
  return (
    <div className='App'>
      <h1>How to focus on input element useRef</h1>
      <input type='text' ref={ref} />
      <br />
      <button onClick={onClick}>Click to Focus on input</button>
    </div>
  )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```

Getting Content from DOM tree

Don't touch the DOM when you develop a React application because React has its own way to manipulate the DOM using the virtual DOM. In case, we get interested to get some content from the DOM tree we can use the useRef hook. See the example:

```
import React, { useRef } from 'react'
import ReactDOM from 'react-dom'

const App = (props) => {
  const ref = useRef(null)
  const onClick = () => {
    let content = ref.current.textContent
    alert(content)
    console.log(content)
  }
  return (
    <div className='App'>
      <h1 ref={ref}>How to getting content from the DOM tree</h1>
      <button onClick={onClick}>Getting Content</button>
    </div>
  )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement)
```

Accessing and Styling a DOM element

We can access and style an element from the DOM tree. See the example below:

```
import React, { useRef } from 'react'
import ReactDOM from 'react-dom'

const App = (props) => {
  const ref = useRef(null)
  const onClick = () => {
    ref.current.style.backgroundColor = '#61dbfb'
    ref.current.style.padding = '50px'
    ref.current.style.textAlign = 'center'
  }
  return (
    <div className='App'>
      <h1 ref={ref}>How to style HTML from the DOM tree using useRef</h1>
      <button onClick={onClick}>Style it</button>
    </div>
  )
}

const rootElement = document.getElementById('root')
ReactDOM.render(<App />, rootElement
```

Exercises

1. Develop the following application. The application generates 27 hexadecimal colors by default. If the generate button get clicked it will generate another new 27 hexadecimal colors.

🎉 CONGRATULATIONS ! 🎉

<< Day 25 | Day 27>>