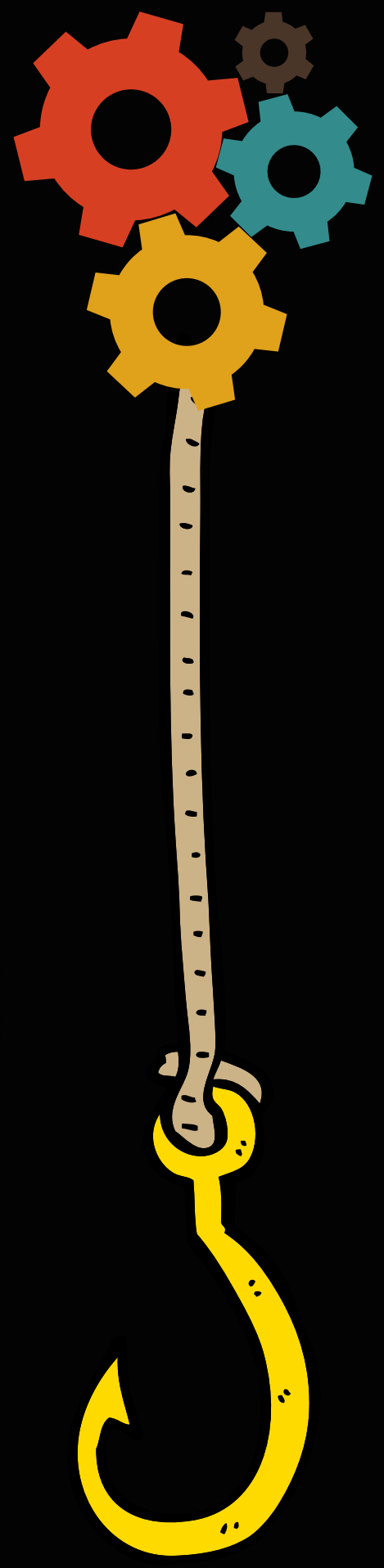
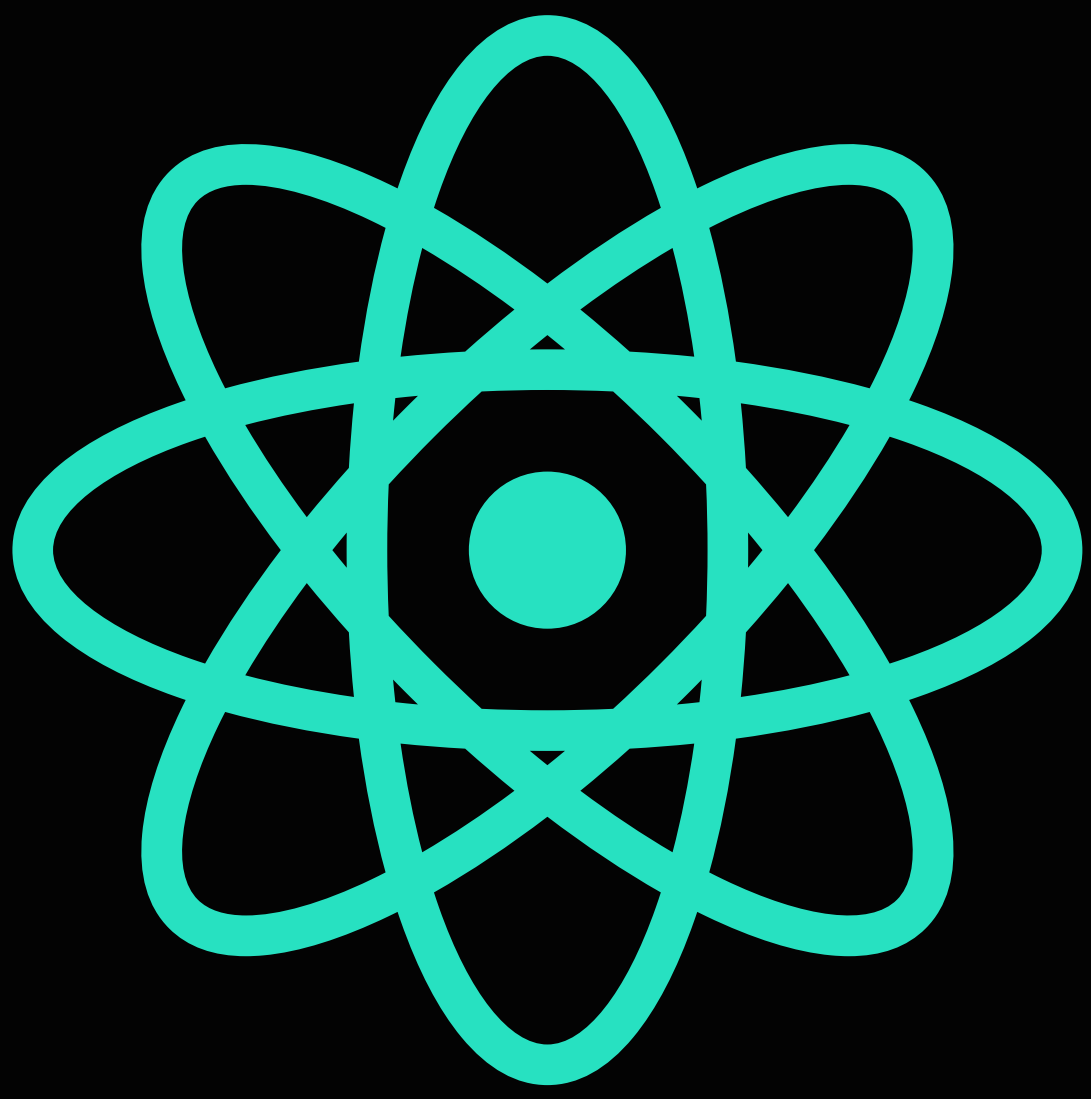
DON’T MISS IT

REACT HOOKS



React Hooks

- React Hooks are functions that enable functional components to use state, lifecycle methods, and more.
- They provide a simpler and more intuitive way to manage component logic compared to class components.



useState()

- The useState Hook allows functional components to manage state.

- It returns a state variable and a function to update that variable.

JS

```
import React, { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);

  return (
    <div>
      <h1>Counter: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
};

export default Counter;
```

Swipe

useEffect()

- The useEffect Hook enables functional components to perform side effects, such as data fetching, subscriptions, or DOM manipulation.
- It runs after every render.

JS

```
useEffect(() => {
    const interval = setInterval(() => {
      setSeconds(prevSeconds => prevSeconds + 1);
    }, 1000);

    return () => clearInterval(interval);
  }, []);
```

Swipe

useContext()

- The useContext Hook allows functional components to consume values from a Context without nesting.

- It's particularly useful for passing down global data like themes or user authentication.

JS

```
import React, { useContext } from 'react';
import ThemeContext from './ThemeContext';

const ThemedButton = () => {
  const theme = useContext(ThemeContext);

  return (
    <button style={{ background: theme.background, color: theme.foreground }}>
      Themed Button
    </button>
  );
};

export default ThemedButton;
```

Swipe

useRef()

- The useRef Hook is a powerful tool in React for accessing and interacting with DOM elements directly.

- It allows us to create a mutable reference to a DOM element and persists across renders without triggering a re-render.

JS

```
import React, { useRef, useEffect } from 'react';

const MyComponent = () => {
  const inputRef = useRef(null);

  useEffect(() => {
    inputRef.current.focus();
  }, []);

  return (
    <div>
      <h1>Focus Input Field</h1>
      <input ref={inputRef} type="text" />
    </div>
  );
};

export default MyComponent;
```

Swipe

DID YOU FIND IT HELPFUL ?

Share this with a friend who needs it!

INFINITE TECHIES