

Vladislav Makeev

#### useState

```
const [ cookies, setCookies ] = useState(0)
```

```
const [ cookies, setCookies ] = useState(0)
function handleClick() {
    setCookies(prev => prev + 1)
}
```

```
const [ cookies, setCookies ] = useState(0)
function handleClick() {
    setCookies(cookies + 1)
}
```

```
const [ cookies, setCookies ] = useState(0)
function handleClick() {
    setCookies(cookies + 1)
function handleTripleClick() {
    handleClick()
    handleClick()
    handleClick()
```

```
const [ cookies, setCookies ] = useState(0)
function handleClick() {
    setCookies(prev => prev + 1)
function handleTripleClick() {
    handleClick()
    handleClick()
    handleClick()
```

## useEffect

```
useEffect(() => {
    let newCookies = Number(localStorage.getItem('cookieClickInReact.Cookies'))
    if (newCookies) {
        setCookies(newCookies)
    }
    }, []
)
```

```
useEffect(() => {
    let newCookies = Number(localStorage.getItem('cookieClickInReact.Cookies'))
    if (newCookies) {
        setCookies(newCookies)
  }, []
useEffect(() => {
        localStorage.setItem('cookieClickInReact.Cookies', cookies)
    }, [cookies]
```

# useRef

#### const ref = useRef(0)

```
const ref = useRef(0)
useEffect(() => {
    ref.current++;
})
```

```
const ref = useRef(0)
useEffect(() => {
    ref.current++;
useEffect(() => {
    ref.current++;
}, [])
```

```
const ref = useRef(0)
useEffect(() => {
    ref.current++;
useEffect(() => {
    ref.current++;
}, [])
useEffect(() => {
    ref.current++;
   [cookies])
```

```
const ref = useRef(0)
useEffect(() => {
    ref.current++;
                  // 5
useEffect(() => {
    ref.current++;
                  // 1
}, [])
useEffect(() => {
    ref.current++;
                    //3
   [cookies])
```

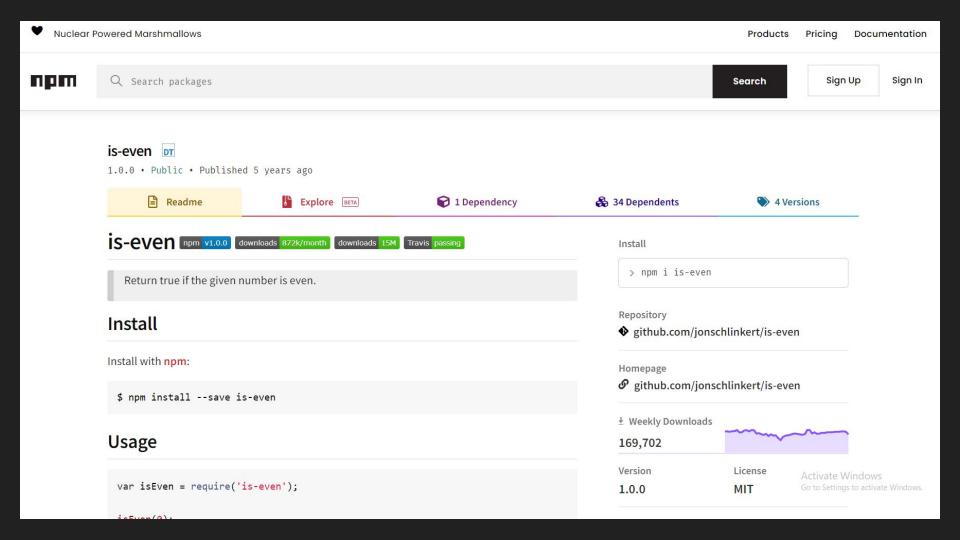
```
const ref = useRef(0)
useEffect(() => {
    ref.current = cookies
}, [cookies])
return (<div>{cookies}, {ref.current}</div>)
```

```
const curRef = useRef(0)
useEffect(() => {
    curRef.current.focus()
}, [cookies])
return (
    <div>Cookies: <input
     ref={curRef} type="text" value={cookies}/>
    </div>
```

#### useMemo

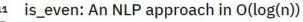
```
const [ cookies, setCookies ] = useState(0)
const [ playerId, setPlayerId ] =
useState(123_456_789)
const playerIdIsEven = isEven(playerId)
return (
    <div>
        <h1>PlayerId: {playerIdIsEven}</h1>
        <div>Cookies: {cookies}</div>
    </div>
```

```
private bool IsEven(int number){
   if (number == 1) return false;
   else if (number == 2) return true;
   else if (number == 3) return false:
   else if (number == 4) return true;
   else if (number == 5) return false;
   else if (number == 6) return true;
   else if (number == 7) return false;
   else if (number == 8) return true;
   else if (number == 9) return false;
   else if (number == 10) return true;
   else if (number == 11) return false;
   else if (number == 12) return true;
   else if (number == 13) return false;
   else if (number == 14) return true;
   else if (number == 15) return false;
   else if (number == 16) return true;
   else if (number == 17) return false;
   else if (number == 18) return true;
   else if (number == 19) return false;
   else if (number == 20) return true;
   else if (number == 21) return false;
   else if (number == 22) return true:
```











Sometimes numbers can be scary. Numbers written out as friendly English text are easier on the eyes, so here's an is\_even which works with English numbers and a helper function which gets them into the right format. Runs in O(log(n)), since we only look at each digit once or twice.

```
from math import log, floor
ones = ['zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', '
teens = [*ones, 'ten', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fift
         'sixteen', 'seventeen', 'eighteen', 'nineteen']
tens = ['oops', 'oof ouch owie', 'twenty', 'thirty', 'forty', 'fifty', '
exponents = ['thousand', 'million', 'billion', 'trillion', 'quadrillion'
             'sextillion', 'septillion', 'octillion', 'nonillion', 'deci
def to_english(n):
    result = ''
    while n \ge 1000:
        l = floor(log(n) / log(1000))
        r = floor(n / 1000 ** 1)
        n = n % 1000 ** 1
        exponent = exponents[1 - 1]
        result += f'{to_english(r)}-{exponent} '
```

```
const [ cookies, setCookies ] = useState(0)
const [ playerId, setPlayerId ] = useState(123_456_789)
const playerIdIsEven = isEven(playerId)
return (
    <div>
        <h1>PlayerId: {playerIdIsEven}</h1>
        <div>Cookies: {cookies}</div>
    </div>
```

```
const [ cookies, setCookies ] = useState(0)
const [ playerId, setPlayerId ] = useState(123_456_789)
const playerIdIsEven = useMemo(
    () => isEven(playerId), [playerId]
return (
    <div>
        <h1>PlayerId: {playerIdIsEven}</h1>
        <div>Cookies: {cookies}</div>
    </div>
```

## useCallback

```
const [ cookies, setCookies ] = useState(0)
const [ playerId, setPlayerId ] = useState(123_456_789)
const playerIdIsEven = () => isEven(cookies)
return (
   <div>
      <PlayerComponent isEven={playerIdIsEven}>
      </PlayerComponent>
      <div>Cookies: {cookies}</div>
   </div>
```

```
const [ cookies, setCookies ] = useState(0)
const [ playerId, setPlayerId ] = useState(123_456_789)
const playerIdIsEven = useCallback(
    () => isEven(cookies), [cookies]
return (
   <div>
      <PlayerComponent isEven={playerIdIsEven}>
      </PlayerComponent>
      <div>Cookies: {cookies}</div>
   </div>
```

## useId

```
const id = useId()
return (
    <div id={id}></div>
```

### useContext

```
const [ cookies, setCookies ] = useState(0)
const CookieContext = createContext(cookies)
return (
    <CookieContext.Provider value={cookies}>
        <CookieComponent></CookieComponent>
    </CookieContext.Provider>
```

const cookies = useContext(CookiesContext)

<u>Context Custom Hook</u>

const cookies = useContext(CookiesContext)

<u>Context Custom Hook</u>