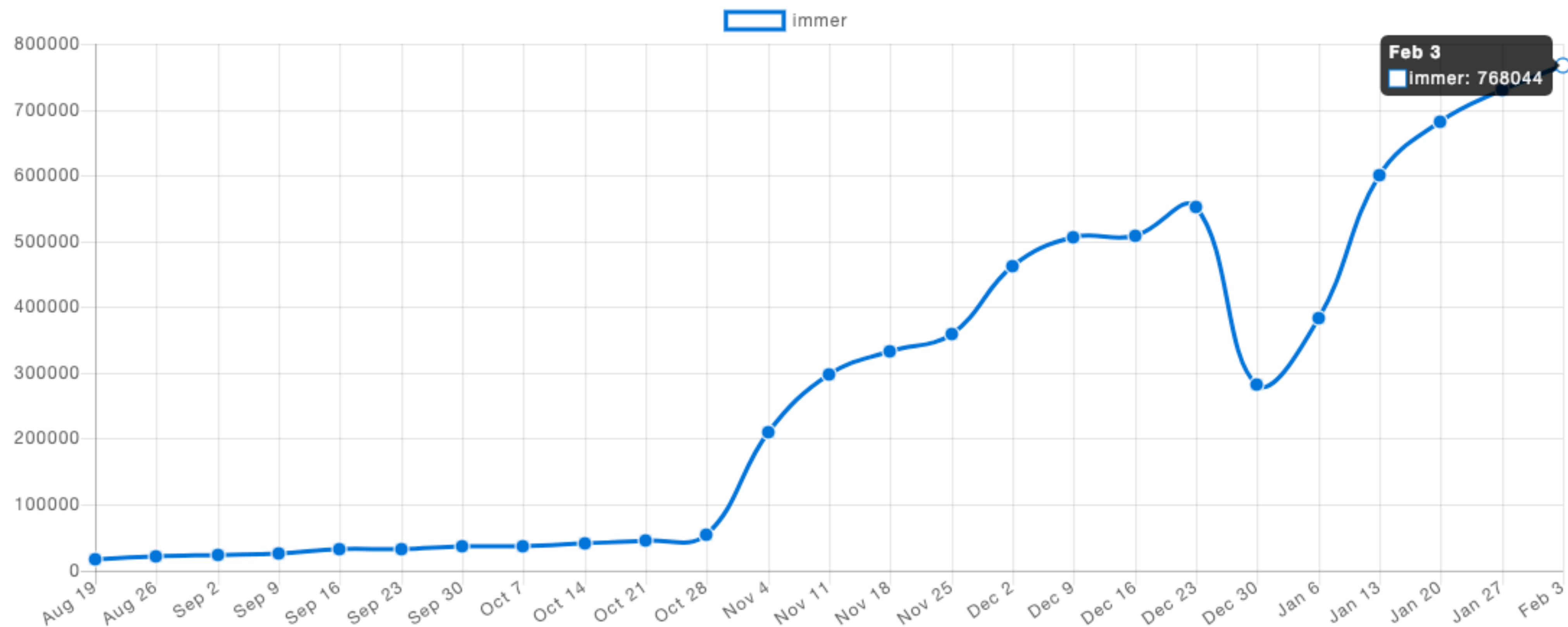


리액트 개발자라면 꼭! 사용해볼 가치가 있는 불변성 관리 도구

Immer.js



불변성을 지키는건

... (spread 연산자) 사용하면 쉬움



```
const object = {  
  a: 1,  
  b: 2,  
  c: 3  
};  
  
const nextObject = {  
  ...object,  
  d: 4  
};  
  
console.log(object !== nextObject); // true
```

배열이면 배열의 내장함수로
대부분의 경우 충분



```
const items = [  
  {  
    id: 1,  
    text: 'spread 연산자로 새 객체 생성',  
    checked: false  
  },  
  {  
    id: 2,  
    text: '배열 내장함수로 새 배열 생성',  
    checked: false  
  },  
  {  
    id: 3,  
    text: 'immer 를 사용해서 편하게!',  
    checked: false  
  }  
];
```



```
const nextItems = items.concat({  
  id: 4,  
  text: '새 데이터 추가!',  
  checked: false  
});
```



```
const nextItems = items.map(item => {  
  if (item.id === 1) {  
    return {  
      ...item,  
      checked: !item.checked  
    };  
  }  
  return item;  
});
```




```
const nextItems = items.filter(item => item.id !== 3);
```

이런건 쉬운데,
가끔씩 구조가 복잡해질때가 있다!

객체 안의 객체



```
const object = {  
  remains: '이건 그대로 두고',  
  updateMe: {  
    value: '이것만 업데이트 해봅시다',  
    anotherValue: '그런데 이건 그대로 유지시켜주세요',  
    inside: {  
      number: 1, // 이것도 바꿔봅시다!  
      anotherNumber: 2,  
    }  
  }  
}
```



```
const nextObject = {  
  ...object,  
  updateMe: {  
    ...object.updateMe,  
    value: '업데이트 됨!',  
    inside: {  
      ...object.updateMe.inside,  
      number: object.updateMe.inside.number + 1,  
    }  
  }  
}
```



```
const nextObject = {  
  ...object,  
  updateMe: {  
    ...object.updateMe,  
    value: '업데이트 됨!',  
    inside: {  
      ...object.updateMe.inside,  
      number: object.updateMe.inside + 1,  
    }  
  }  
}
```

매우 헛갈림!!!
(슬라이드 준비하면서 실수)

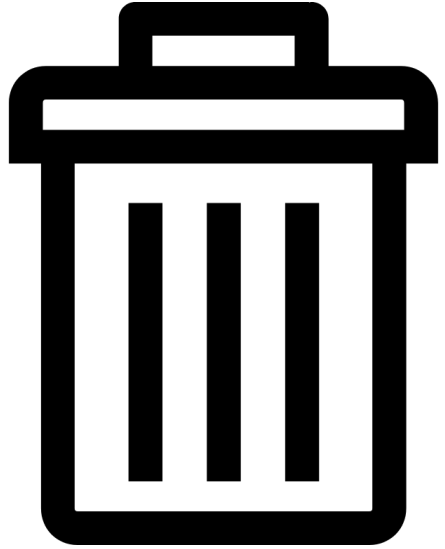
객체 안에 객체 안에 배열?



```
const object = {  
  a: {  
    b: {  
      items: [  
        {  
          id: 1,  
          text: 'hello' // -> 'bye'  
        },  
        {  
          id: 2,  
          text: 'world'  
        }  
      ]  
    },  
    c: 1,  
    d: 2  
  },  
  e: 3  
};
```




```
const nextObject = {  
  ...object,  
  a: {  
    ...object.a,  
    b: {  
      ...object.a.b,  
      items: object.a.b.items.map(item =>  
        item.id === 1 ? { ...item, text: 'bye' } : item  
      )  
    }  
  }  
};
```

구조가 복잡해지면,
가독성이 솔직히... 

가능하면 상태의 객체 구조가
복잡해지지 않도록
개발하는것이 좋아요

그러나 어쩔 수 없는 상황에..!


도와주세요 immer맨!



```
import produce from "immer"

const baseState = [
  {
    todo: "Learn typescript",
    done: true
  },
  {
    todo: "Try immer",
    done: false
  }
]

const nextState = produce(baseState, draftState => {
  draftState.push({todo: "Tweet about it"})
  draftState[1].done = true
})
```



```
const nextObject = {
  ...object,
  updateMe: {
    ...object.updateMe,
    value: '업데이트 됨!',
    inside: {
      ...object.updateMe.inside,
      number: object.updateMe.inside.number + 1,
    }
  }
}
```



```
const nextObject = produce(object, draft => {
  draft.updateMe.value = '업데이트 됨!';
  draft.updateMe.inside.number += 1;
});
```

```
const nextObject = {
  ..object,
  a: {
    ...object.a,
    b: {
      ...object.a.b,
      items: object.a.b.items.map(item =>
        item.id === 1 ? { ..item, text: 'bye' } : item
      )
    }
  }
};
```

```
const nextObject = produce(object, draft => {
  const item = draft.a.b.items.find(id => id === 1);
  item.text = 'bye';
});
```


CodeSandbox 에서 실습하기

<https://bit.ly/immer-tutorial>

첫번째 실습: <https://bit.ly/2RP9rkB>
두번째 실습: <https://bit.ly/2GISNIr>