

Your state is not so consistent and predictable after all?
Normalize it!



Michał Wołczycki-Klim

Javascript developer who likes styling 🐢 🐢 🐢 Working around web apps for 5 years and counting. My first website was made for club party that I was organising.

[Twitter](#)

[Github](#)

[LinkedIn](#)

What's going on with state normalization?

Normalization
awerness

100

75

25

0

0

1

2

3

Years with React



What it actually means to normalize state shape?

Normalization is the process of reorganizing data in a database so that it meets two basic requirements: (1) There is no redundancy of data (all data is stored in only one place), and (2) data dependencies are logical (all related data items are stored together)

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ]  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32  }  
33 ]
```

Some fairly common
product data

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ]  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32  }  
33 ]
```

Some fairly common product data

- Data structure is complicated, hard to read and reason about


```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ]  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32   }  
33 ]
```

Some fairly common product data

- Data structure is complicated, hard to read and reason about
- Some data is repeated

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ]  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32  }  
33 ]
```

Some fairly common product data

- Data structure is complicated, hard to read and reason about
- Some data is repeated
- Corresponding reducer logic could be complicated

○○○

```
1 const products = (state = [], action) => {
2   switch (action.type) {
3     case 'ADD_PRODUCT': {
4       return [
5         ...state,
6         {
7           ...action.product,
8           variants: action.product.variants.map(
9             variant => variant.attributes.map(
10              attribute => attribute.name.toUpperCase()
11            )
12          )
13        }
14      ]
15    }
16  }
17 }
```

Our products reducer
has to keep logic that
only concerns
attributes elements

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ]  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32  }  
33 ]
```

Three types of data (entities):

- Products
- Variants
- Attributes

All of them could be moved to separate lookup tables

○○○

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ],  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32  }  
33 ]
```



○○○

```
1 const entities = {  
2   products: {  
3     byIds: {  
4       product1: {  
5         id: 'product1',  
6         name: 'Product One',  
7         price: 999,  
8         variants: ['variant11', 'variant12']  
9       }  
10    },  
11  },  
12  variants: {  
13    byIds: {  
14      variant11: {  
15        id: 'variant11',  
16        img: 'http://test.com/img11.jpg',  
17        attributes: ['attrColorRed']  
18      },  
19      variant12: {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: ['attrColorRed']  
23      }  
24    },  
25  },  
26  attributes: {  
27    byIds: {  
28      attrColorRed: {  
29        id: 'attrColorRed',  
30        type: 'color',  
31        name: 'red',  
32        value: '#FF0000'  
33      }  
34    }  
35  }  
36 }
```

```
1 const entities = {
2   products: {
3     byIds: {
4       product1: {
5         id: 'product1',
6         name: 'Product One',
7         price: 999,
8         variants: ['variant11', 'variant12']
9       }
10    },
11  },
12  variants: {
13    byIds: {
14      variant11: {
15        id: 'variant11',
16        img: 'http://test.com/img11.jpg',
17        attributes: ['attrColorRed']
18      },
19      variant12: {
20        id: 'variant12',
21        img: 'http://test.com/img12.jpg',
22        attributes: ['attrColorRed']
23      }
24    },
25  },
26  attributes: {
27    byIds: {
28      attrColorRed: {
29        id: 'attrColorRed',
30        type: 'color',
31        name: 'red',
32        value: '#FF0000'
33      }
34    }
35  }
36 }
```

Benefits

- Much clearer structure. We can clearly see what data is in each entity and what entities are there
- We can probably simplify reducer logic since we no longer have to deal with nested data
- No repetition of data. If we want to update our red color attribute we will have to make only in one place
- Having item id we can obtain its data without digging through nested object
- Performance gain! Since we decoupled entities we should see less components rerenders when updating specific data
- More flexibility!



```
1 // entities.reducer.js
2 import { combineReducers } from 'redux'
3
4 import products from 'products.reducer.js'
5 import variants from 'variants.reducer.js'
6 import arguments from 'arguments.reducer.js'
7
8 const entities = combineReducers(
9   {
10     products,
11     variants,
12     arguments
13   }
14 )
```

Entry point for our normalization

- Use combineReducers function to split your reducers logic
- Store all your entities in one reducer

```
1 // products.reducer.js
2 import { combineReducers } from 'redux'
3
4 const byIds = (state = {}, action) => {
5   switch (action.type) {
6     case 'ADD_PRODUCT': {
7       return {
8         ...state,
9         [action.product.id]: {
10           ...action.product,
11           variants: [action.product.variants.map(variant => variant.id)]
12         }
13       }
14     }
15     default:
16       return state
17   }
18 }
19
20 const products = combineReducers({
21   byIds,
22 })
23
24 export default products
```

Products reducer


```
1 // variants.reducer.js
2 import { combineReducers } from 'redux'
3
4 const byIds = (state = {}, action) => {
5   switch (action.type) {
6     case 'ADD_PRODUCT': {
7       return {
8         ...state,
9         ...action.product.variants.reduce((acc, variant) => {
10           acc[variant.id] = {
11             ...variant,
12             attributes: variant.attributes.map(attribute => attribute.id)
13           }
14         }, {}),
15       }
16     }
17   }
18   default:
19     return state
20 }
21
22 const variants = combineReducers({
23   byIds,
24 })
25
26 export default variants
```

Variants reducer

```
1 // attributes.reducer.js
2 import { combineReducers } from 'redux'
3 import { flatMap } from 'lodash'
4
5 const byIds = (state = {}, action) => {
6   switch (action.type) {
7     case 'ADD_PRODUCT': {
8       const allAttributes = flatMap(action.product.variants, variant => variant.attributes)
9
10      return {
11        ...state,
12        ...allAttributes.reduce((acc, attribute) => {
13          acc[attribute.id] = {
14            ...attribute,
15            name: attribute.name.toUpperCase()
16          }
17
18          return acc
19        }, {})
20      }
21    }
22    default:
23      return state
24  }
25 }
26
27 const attributes = combineReducers({
28   byIds,
29 })
30
31 export default attributes
```

Attributes reducer



```
1 import { normalize, schema } from 'normalizr'
2 import data from 'data.js'
3
4 const attribute = new schema.Entity('attributes')
5 const variant = new schema.Entity('variants', {
6   attributes: [attribute]
7 })
8 const product = new schema.Entity('products', {
9   variants: [variant]
10 })
11
12 const mySchema = new schema.Array(product)
13
14 const normalizedData = normalize(data, mySchema);
```

Normalizr library can
do a lot of heavy
lifting for us

○○○

```
1 const data = [  
2   {  
3     id: 'product1',  
4     name: 'Product One',  
5     price: 999,  
6     variants: [  
7       {  
8         id: 'variant11',  
9         img: 'http://test.com/img11.jpg',  
10        attributes: [  
11          {  
12            id: 'attrColorRed',  
13            type: 'color',  
14            name: 'red',  
15            value: '#FF0000'  
16          }  
17        ],  
18      },  
19      {  
20        id: 'variant12',  
21        img: 'http://test.com/img12.jpg',  
22        attributes: [  
23          {  
24            id: 'attrColorRed',  
25            type: 'color',  
26            name: 'red',  
27            value: '#FF0000'  
28          }  
29        ]  
30      }  
31    ]  
32   }  
33 ]
```



○○○

```
1 const data = {  
2   entities: {  
3     attributes: {  
4       attrColorRed: {  
5         id: 'attrColorRed',  
6         type: 'color',  
7         name: 'red',  
8         value: '#FF0000'  
9       }  
10    },  
11    variants: {  
12      variant11: {  
13        id: 'variant11',  
14        img: 'http://test.com/img11.jpg',  
15        attributes: [ 'attrColorRed' ]  
16      },  
17      variant12: {  
18        id: 'variant12',  
19        img: 'http://test.com/img12.jpg',  
20        attributes: [ 'attrColorRed' ]  
21      }  
22    },  
23    products: {  
24      product1: {  
25        id: 'product1',  
26        name: 'Product One',  
27        price: 999,  
28        variants: [ 'variant11', 'variant12' ]  
29      }  
30    }  
31  },  
32  result: [ 'product1' ]  
33 }
```

○○○

```
1 // products.reducer.js
2 import { combineReducers } from 'redux'
3
4 const byIds = (state = {}, action) => {
5   switch (action.type) {
6     case 'ADD_PRODUCT': {
7       return {
8         ...state,
9         ...action.entities.products
10       }
11     }
12     default:
13       return state
14   }
15 }
16
17 const products = combineReducers({
18   byIds,
19 })
20
21 export default products
```

Products reducer

○○○

```
1 // variants.reducer.js
2 import { combineReducers } from 'redux'
3
4 const byIds = (state = {}, action) => {
5   switch (action.type) {
6     case 'ADD_PRODUCT': {
7       return {
8         ...state,
9         ...action.entities.variants
10       }
11     }
12     default:
13       return state
14   }
15 }
16
17 const variants = combineReducers({
18   byIds,
19 })
20
21 export default variants
```

Variants reducer

○○○

```
1 // attributes.reducer.js
2 import { combineReducers } from 'redux'
3 import { flatMap } from 'lodash'
4
5 const byIds = (state = {}, action) => {
6   switch (action.type) {
7     case 'ADD_PRODUCT': {
8       return {
9         ...state,
10        ...action.entities.attributes
11      }
12    }
13    default:
14      return state
15  }
16 }
17
18 const attributes = combineReducers({
19   byIds,
20 })
21
22 export default attributes
```

Attributes reducer

Normalize it!

Useful links:

- <https://redux.js.org/recipes/structuringreducers/normalizingstateshape>
- <https://egghead.io/lessons/javascript-redux-normalizing-the-state-shape>
- <https://www.quora.com/What-is-a-lookup-table>
- <https://hackernoon.com/redux-patterns-rethinking-byid-and-byhash-structures-854e8a0fa32d>
- <https://www.techopedia.com/definition/1221/normalization>