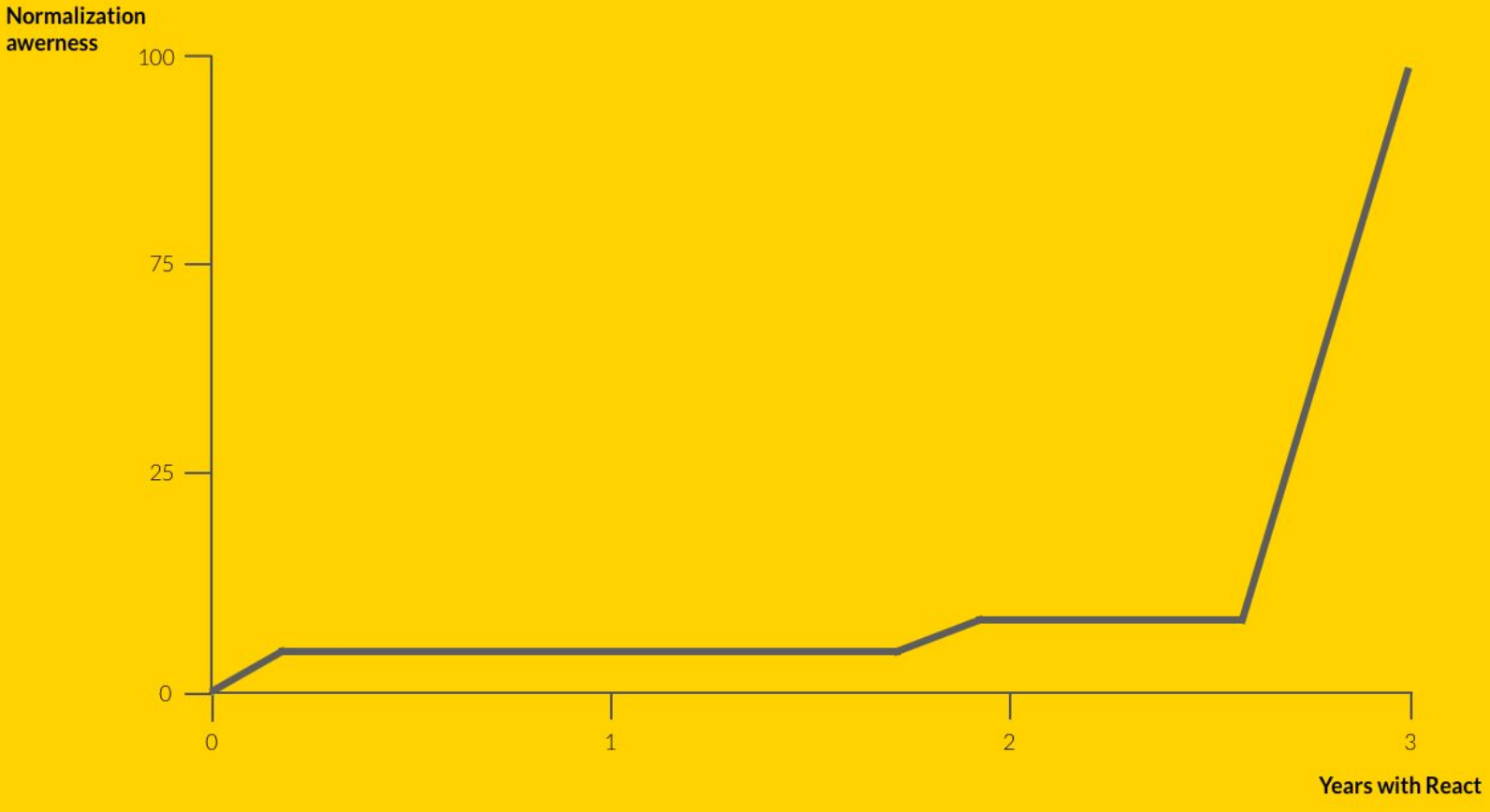
Your state is not so consistent and predictable after all?
Normalize it!

Michał Wołczecki-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)

Techopedia.com

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

Some fairly common product data

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

Some fairly common product data

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

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

Some fairly common product data

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

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

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

```
const products = (state = [], action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return [
        ...state,
        {
          ...action.product,
          variants: action.product.variants.map(
            variant => variant.attributes.map(
              attribute => attribute.name.toUpperCase()
            )
          )
        }
      ]
    }
  }
}
```

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

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

Three types of data (entities):

- Products
- Variants
- Attributes

All of them could be moved to separate lookup tables

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

```
const entities = {
  products: {
    byIds: {
      product1: {
        id: 'product1',
        name: 'Product One',
        price: 999,
        variants: ['variant11', 'variant12']
      }
    }
  },
  variants: {
    byIds: {
      variant11: {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: ['attrColorRed']
      },
      variant12: {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: ['attrColorRed']
      }
    }
  },
  attributes: {
    byIds: {
      attrColorRed: {
        id: 'attrColorRed',
        type: 'color',
        name: 'red',
        value: '#FF0000'
      }
    }
  }
}
```

```
const entities = {
  products: {
    byIds: {
      product1: {
        id: 'product1',
        name: 'Product One',
        price: 999,
        variants: ['variant11', 'variant12']
      }
    }
  },
  variants: {
    byIds: {
      variant11: {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: ['attrColorRed']
      },
      variant12: {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: ['attrColorRed']
      }
    }
  },
  attributes: {
    byIds: {
      attrColorRed: {
        id: 'attrColorRed',
        type: 'color',
        name: 'red',
        value: '#FF0000'
      }
    }
  }
}
```

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!

```
// entities.reducer.js
import { combineReducers } from 'redux'

import products from 'products.reducer.js'
import variants from 'variants.reducer.js'
import arguments from 'arguments.reducer.js'

const entities = combineReducers(
  {
    products,
    variants,
    arguments
  }
)
```

Entry point for our normalization

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

```
// products.reducer.js
import { combineReducers } from 'redux'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return {
        ...state,
        [action.product.id]: {
          ...action.product,
          variants: [action.product.variants.map(variant => variant.id)]
        }
      }
    }
    default:
      return state
  }
}

const products = combineReducers({
  byIds,
})

export default products
```

Products reducer

```
// variants.reducer.js
import { combineReducers } from 'redux'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return {
        ...state,
        ...action.product.variants.reduce((acc, variant) => {
          acc[variant.id] = {
            ...variant,
            attributes: variant.attributes.map(attribute => attribute.id)
          }

          return acc
        }, {})
      }
    }
    default:
      return state
  }
}

const variants = combineReducers({
  byIds,
})

export default variants
```

Variants reducer

```
// attributes.reducer.js
import { combineReducers } from 'redux'
import { flatMap } from 'lodash'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      const allAttributes = flatMap(action.product.variants, variant => variant.attributes)

      return {
        ...state,
        ...allAttributes.reduce((acc, attribute) => {
            acc[attribute.id] = {
              ...attribute,
              name: attribute.name.toUpperCase()
            }

            return acc
        }, {})
      }
    }
    default:
      return state
  }
}

const attributes = combineReducers({
  byIds,
})

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

```
const data = [
  {
    id: 'product1',
    name: 'Product One',
    price: 999,
    variants: [
      {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      },
      {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [
          {
            id: 'attrColorRed',
            type: 'color',
            name: 'red',
            value: '#FF0000'
          }
        ]
      }
    ]
  }
]
```

```
const data = {
  entities: {
    attributes: {
      attrColorRed: {
        id: 'attrColorRed',
        type: 'color',
        name: 'red',
        value: '#FF0000'
      }
    },
    variants: {
      variant11: {
        id: 'variant11',
        img: 'http://test.com/img11.jpg',
        attributes: [ 'attrColorRed' ]
      },
      variant12: {
        id: 'variant12',
        img: 'http://test.com/img12.jpg',
        attributes: [ 'attrColorRed' ]
      }
    },
    products: {
      product1: {
        id: 'product1',
        name: 'Product One',
        price: 999,
        variants: [ 'variant11', 'variant12' ]
      }
    }
  },
  result: [ 'product1' ]
}
```

```
// products.reducer.js
import { combineReducers } from 'redux'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return {
        ...state,
        ...action.entities.products
      }
    }
    default:
      return state
  }
}

const products = combineReducers({
  byIds,
})

export default products
```

Products reducer

```
// variants.reducer.js
import { combineReducers } from 'redux'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return {
        ...state,
        ...action.entities.variants
      }
    }
    default:
      return state
  }
}

const variants = combineReducers({
  byIds,
})

export default variants
```

Variants reducer

```
// attributes.reducer.js
import { combineReducers } from 'redux'
import { flatMap } from 'lodash'

const byIds = (state = {}, action) => {
  switch (action.type) {
    case 'ADD_PRODUCT': {
      return {
        ...state,
        ...action.entities.attributes
      }
    }
    default:
      return state
  }
}

const attributes = combineReducers({
  byIds,
})

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