



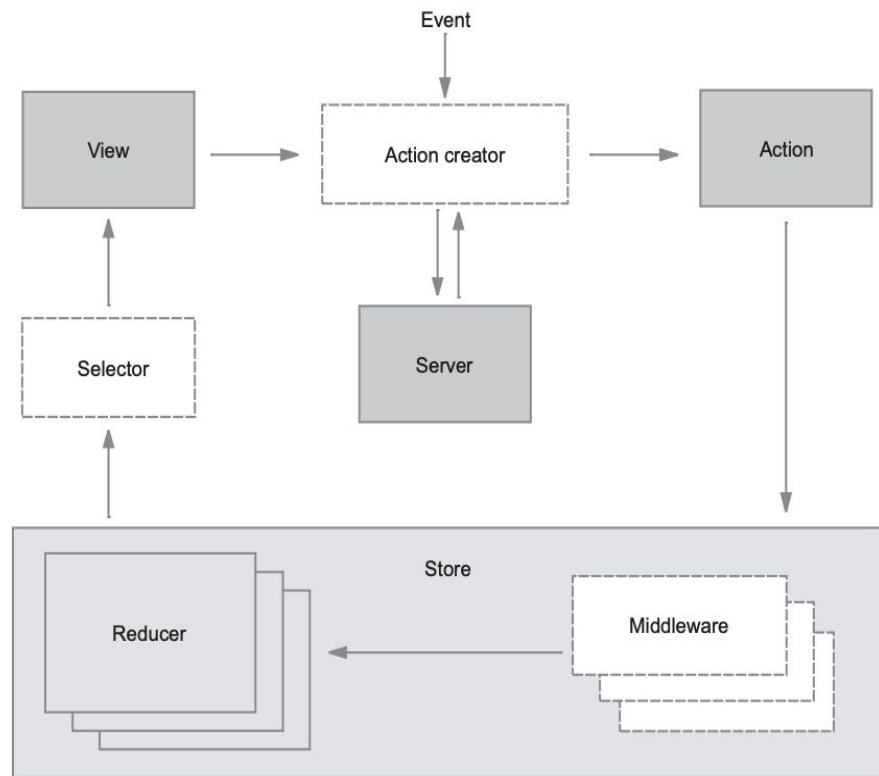
# Redux in 10 minutes #2

Yutthana

Growth Session #22 - February 15 2019

## From the previous section

- Redux is a state management library
- Basic Redux:
  - Action
  - Action Creator
  - Reducer
  - Store
  - Selector
  - View



## Goals

- Put in practice my Redux knowledge on the PTT-PM project.
- Create a simple back-end API for the auction system.
- Setup React and Redux in the PTT-PM project.

# Setup the API

Create a controller to return the auction data

```
class SampleController < ApplicationController
  before_action :skip_authorization

  def index
  end

  def auction
    @auction = Auction.find(2)
  end
end
```

```
json.auction do
  json.id @auction.id
  json.startsOn @auction.starts_on
  json.endsOn @auction.ends_on
  json.startingPrice @auction.starting_price
  json.minimumStep @auction.minimum_step
  json.quantity @auction.quantity
  json.deliveryUnit @auction.delivery_unit
  json.status @auction.status
  json.createdAt @auction.created_at
  json.updatedAt @auction.updated_at
end

json.product do
  json.id @auction.product.id
  json.sku @auction.product.sku
  json.name @auction.product.name
  json.mfi @auction.product.mfi
  json.productDensity @auction.product.product_density
  json.package @auction.product.package
  json.additionalInformation @auction.product.additional_information
  json.specificationList @auction.product.specification_list
  json.createdAt @auction.product.created_at
  json.updatedAt @auction.product.updated_at
end

json.bids @auction.bids do |bid|
  json.id bid.id
  json.price bid.price

  json.user do
    json.id bid.user.id
    json.firstName bid.user.first_name
    json.lastName bid.user.last_name
    json.email bid.user.email
    json.employeeSN bid.user.employee_sn
    json.department bid.user.department
    json.status bid.user.status
    json.role bid.user.role
    json.createdAt bid.user.created_at
    json.updatedAt bid.user.updated_at
  end
end
```

# Setup React and Redux in the PTT-PM project

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';

import App from './App';

import store from './store';

document.addEventListener('DOMContentLoaded', () => {
  ReactDOM.render(
    <Provider store={store}>
      <App/>
    </Provider>,
    document.getElementById('sample')
  );
});
```

```
import React from 'react';

import AuctionBid from './containers/AuctionBid/index.jsx';

const App = () => (
  <div className="container-fluid">
    <div className="row">
      <div className="col-12 col-lg-12 col-xl-3" />
      <div className="col-12 col-lg-8 col-xl-6">
        <AuctionBid />
      </div>
      <div className="col-12 col-lg-4 col-xl-3" />
    </div>
  </div>
);

export default App;
```

## Setup Actions



```
// Auction
export const FETCH_AUCTION_STARTED = 'FETCH_AUCTION_STARTED';
export const FETCH_AUCTION_SUCCEEDED = 'FETCH_AUCTION_SUCCEEDED';
export const FETCH_AUCTION_FAILED = 'FETCH_AUCTION_FAILED';
```

# Setup Action Creators

```
export function fetchAuctionStarted() {
  return {
    type: FETCH_AUCTION_STARTED
  };
}

export function fetchAuctionSucceeded(data) {
  return {
    type: FETCH_AUCTION_SUCCEEDED,
    payload: {
      data
    }
  };
}

export function fetchAuctionFailed(error) {
  return {
    type: FETCH_AUCTION_FAILED,
    payload: {
      error
    }
  };
}
```

```
export function fetchAuction() {
  return async function(dispatch) {
    const onSuccessed = (data) => {
      dispatch(fetchAuctionSucceeded(data));
      return data;
    };

    const onFailed = (error) => {
      dispatch(fetchAuctionFailed(error));
      return error;
    };

    try {
      dispatch(fetchAuctionStarted());

      const response = await axios.get('/sample/auction.json');
      const data = response.data;

      return onSuccessed(data);
    } catch (error) {
      return onFailed(error);
    }
  };
}
```

# Setup Reducers

```
import {
  FETCH_AUCTION_STARTED,
  FETCH_AUCTION_SUCCEEDED,
  FETCH_AUCTION_FAILED
} from '../constants';

const initialState = {
  auction: {},
  product: {},
  bids: [],
  isFetching: false,
  error: null
};
```

```
function auction(state = initialState, action) {
  switch (action.type) {
    case FETCH_AUCTION_STARTED:
      return Object.assign({}, state, {
        isFetching: true,
        error: null
      });
    case FETCH_AUCTION_SUCCEEDED:
      const { data } = action.payload;
      const { auction, product, bids } = data;

      return Object.assign({}, state, {
        auction,
        product,
        bids,
        isFetching: false
      });
    case FETCH_AUCTION_FAILED:
      const { error } = action.payload;

      return Object.assign({}, state, {
        isFetching: false,
        error
      });
    default:
      return state;
  }
}

export default auction;
```

```
import { combineReducers } from 'redux';

import auction from './auction';

const reducers = combineReducers({
  auction
});

export default reducers;
```



## Setup Stores



```
import { createStore, applyMiddleware } from 'redux';
import { composeWithDevTools } from 'redux-devtools-extension';
import thunk from 'redux-thunk';

import reducers from '../reducers';

const store = createStore(
  reducers,
  composeWithDevTools(applyMiddleware(thunk))
);

export default store;
```

## Setup Stores



```
import { createStore, applyMiddleware } from 'redux';
import { composeWithDevTools } from 'redux-devtools-extension';
import thunk from 'redux-thunk';

import reducers from '../reducers';

const store = createStore(
  reducers,
  composeWithDevTools(applyMiddleware(thunk))
);

export default store;
```

# Create container component and connect the Redux store

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import AuctionBidHeader from '../../components/AuctionBidHeader/index.jsx';
import { fetchAuction } from '../../actions';

class AuctionBid extends Component {
  componentDidMount() {
    this.props.onFetchAuction();
  }

  displayLoading() {
    return <h1>Loading ...</h1>;
  }

  displayError(error) {
    const styles = { color: 'red' };
    return <h1 style={styles}>{error.message}</h1>
  }

  render() {
    const { product, isFetching, error } = this.props;

    if (isFetching) {
      this.displayLoading();
    }

    if (error) {
      this.displayError(error);
    }

    return (
      <section className="auction-bid">
        <AuctionBidHeader title={product.name} tags={product.specificationList} />
      </section>
    );
  }
}
```

```
const mapStateToProps = (state) => {
  const { auction, product, bids, isFetching, error } = state.auction;

  return {
    auction,
    product,
    bids,
    isFetching,
    error
  };
};

const mapDispatchToProps = (dispatch) => {
  return {
    onFetchAuction() {
      dispatch(fetchAuction());
    }
  };
};

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(AuctionBid);
```


Show Example




## Conclusion

- React + Redux is fun.
- It brings your idea to implement component and business logic gradually.

- Create all components to render the whole auction page with React
- Move all logic to Redux.



Redux architecture is awesome.  
It divides the business logic into  
small unit but opinionated.



If you need to learn how one way data flow work, study the Redux architecture as it's a good starting point and it's easy to understand.



# Thanks!

## Contact Nimble

[nimblehq.co](https://nimblehq.co)

[hello@nimblehq.co](mailto:hello@nimblehq.co)

## Bangkok

399 Interchange 21 Sukhumvit Road, Unit  
#2402-03, Klong Toei, Wattana, Bangkok  
10110, Thailand

## Singapore

28C Stanley St, Singapore 068737

## Hong Kong

20th Floor, Central Tower  
28 Queen's Road, Central, Hong Kong

