

REDUX PERFORMANCE

DAVID J. BRADSHAW

Principle FE Developer - ICAP Fusion Platform

- dave@bradshaw.net
- github.com/davidjbradshaw
- @davidjbrashaw

FUSION OPTIONS

dev.icapfusion.com

FUSION

Options

Activity Monitor

Agreeable Counterparties

All Trades

Historical Chart

IOI History

Interest Blotter

Interest Monitor

Legal Entity Grouping

Matching Monitor

Trade Admin

Trade Proofing

Trading Entities

Trading Mappings

EUR CCP

EUR Swaptions > Premiums > CCP (LCH) Fwd Premiums

	1y	2y	3y	4y	5y	6y	7y	8y	9y	10y	15y	20y	25y	30y
1m Opt	3	8.5	16	25	34.5	43.5	53	63.5	73.5	83.5	127	171	208	239
2m Opt	4	12	22.5	34	46	59	72	86	100	113	174	234	286	327
3m Opt	5.5	15	28	42	56.5	73.5	90	107	124	142	215	283	344	400
6m Opt	8	21	39	60	83.5	109	133	158	184	210	315	411	498	573
9m Opt	11	28.5	51	77.5	110	137	168	197	228	262	387	505	612	707
1y Opt	15	37.5	65	97	133	166	204	238	275	309	454	595	718	826
18m Opt	24	55	93.5	135	179	220	266	309	352	395	577	743	895	1037
2y Opt	33.5	73.5	122	172	222	272	323	373	422	467	677	871	1049	1207
3y Opt	52.5	110	173	236	299	363	423	483	542	601	854	1091	1308	1511
4y Opt	70.5	144	219	293	367	440	510	585	654	718	1004	1276	1524	1762
5y Opt	85.5	171	258	343	429	511	592	673	750	822	1141	1438	1716	1988
6y Opt	99.5	199	296	392	486	579	669	757	841	921	1264	1598	1900	2205
7y Opt	111.5	220	329	434	536	636	731	825	919	1008	1376	1731	2067	2400
8y Opt	122	240	358	472	582	691	795	897	994	1087	1482	1863	2234	2589
9y Opt	131.5	260	385	507	624	739	850	958	1061	1166	1586	1992	2389	2778
10y Opt	139.5	275	407	535	662	783	901	1012	1121	1234	1691	2117	2531	2951
12y Opt	152.5	300	444	586	722	855	987	1112	1236	1352	1851	2324	2789	3239
15y Opt	168	332	491	645	792	939	1080	1216	1359	1494	2066	2615	3135	3640
20y Opt	188	369	549	721	893	1064	1224	1384	1541	1691	2367	2972	3559	4126
25y Opt	205	406	604	794	980	1163	1338	1509	1681	1849	2602	3245	3877	4490
30y Opt	219	434	647	852	1055	1249	1433	1607	1784	1963	2766	3453	4120	4775

Last updated: 29/05/2019 16:30:08

Historical Chart

S 29/05/2018 29/05/2019

10D 1M 3M 6M YTD 1Y 2Y 5Y MAX

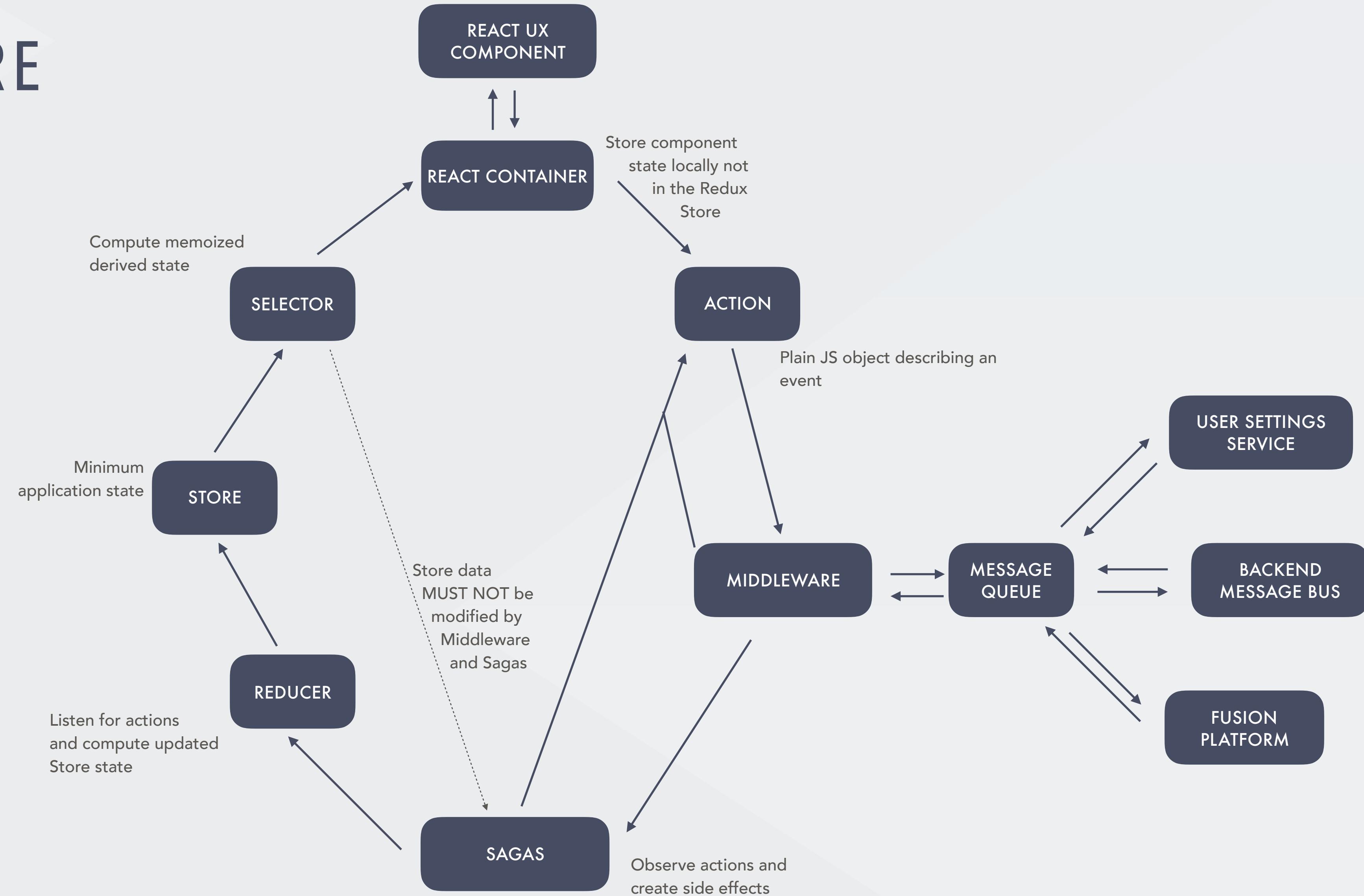
● EUR 4Y15Y (EUR CCP)

24/10/2016 04/06/2018 22/06/2018 12/07/2018 01/08/2018 21/08/2018 10/09/2018 28/09/2018 18/10/2018 07/11/2018 27/11/2018 17/12/2018 04/01/2019 24/01/2019 13/02/2019 05/03/2019 25/03/2019 12/04/2019 02/05/2019 22/05/2019

950 1,000 1,050 1,100 1,150 1,200

28/05/2019

FUSION OPTIONS ARCHITECTURE



SUBSCRIBE

```
{  
  TYPE: HEARTBEAT,  
  META: {  
    ID: 'HEARTBEAT',  
    SUBSCRIBE: {  
      TPVOL: 'LDN/PS/52/P/GLOBAL/RFQ/E/HB',  
    }  
  }  
}
```

```
{  
  type: 'mq/MSG/tpvol/LDN/PS/52/P/GLOBAL/RFQ/E/HB',  
  payload: {  
    hostName: 'ldnmtpvgy01',  
    interval: 5000,  
  },  
  meta: {  
    batch: true,  
    eventStream: false,  
  }  
}
```

dev.icapfusion.com 05:36 option2.dave.broker@icap.com

FUSION

EUR CCP EUR Cash (IRR) Fwd Premiums

	1y	2y	3y	4y	5y	6y	7y	8y	9y	10y	15y	20y	25y	30y	Other	Price	
1m Opt	5.9	1.2	3.3	4.1	1	1.9	4.6	8.8	5.5	4.1	1.1	5.4	8.6		Penguins that are 1M old	3.5	
2m Opt	8.7	5	8.5	2.7	0.8	2.7	1.6	7	1.6	7.4	7.3	1.9	0.9		Penguins that are very old by pe...	4.9	
3m Opt	2.9	7.6	6.8	7.7	7.7	1.1	3.4	0.5	4.9	3.9	9.2	9	1.4		Apples	4	
6m Opt	0.8	6.4	6.7	1.7	0.4	9.3	1.3	7.8	8.8	9.6	2.2	7.9	0.7		Oranges	8.5	
9m Opt	9	0.6	0.2	2.9	2.7	2	1.1	1.8	8	2	1.7	4.7	10				
1y Opt	3.6	6.6	2.1	5.6	6.8	0.2	4.9	7	9.4	7.9	2.1	1.2	1.3		Spread	Price 1	Price 2
18m Opt	2.1	1.4	5.3	1.8	3.4	8	3.6	7.6	1.7	3.7	7.6	0.8	2.7		1M1Y vs 1M2Y	0.9	2
2y Opt	9.7	2.4	2.9	1.4	8.5	4.8	8.4	2.6	6.7	2.9	7.7	8.7	9.3		1M1Y vs 2Y4Y	8.1	6.2
3y Opt	8.1	7	1.3	4	5	8.8	6.8	8	4.7	9.9	5	3.6	8.1				
4y Opt	7	4.4	3.8	5.2	0.5	6.3	8.8	4.6	7.4	9.2	5	3.3	6.6				
5y Opt	9.3	6.8	7	9.7	2.1	0.6	6	6.3	7	5.5	9.2	7.7	7.8				
6y Opt	9.2	7.3	3.9	8	6.4	1.2	9.7	0.8	0.5	3.6	2	7.2	5.3				
7y Opt	7	7.7	5.2	10	5.2	2.3	7.1	6.6	1.7	2.8	9.9	1	7.1				
8y Opt	3	8.4	3.2	6.5	7.8	6.3	9	4.4	5.3	2.5	1.5	4.7	1.3				
9y Opt	0.5	5.3	5.5	1.9	3.7	2.7	3.9	7.3	9.3	2.9	0.5	6.3	2.1				
10y Opt	9.6	2.9	3.2	8.4	8.9	3	6.7	4.4	3.7	0.1	9.5	6.6	5.7				
12y Opt	7.9	2	4.5	6.4	5.9	0.9	6.7	9.6	4.7	6.4	3.3	9.3	9.6				
15y Opt	1.5	7.2	3	4.4	7.7	6.8	0.3	0.5	6.3	7.8	4.4	9.6	9.9				
20y Opt	9.2	3.1	5.4	4.5	3.6	4.8	3.9	9.5	4.6	5.8	9.7	2.7	6.9				
25y Opt	9.1	8.3	4.4	4.1	4.3	5.6	7.1	7.6	8.4	7.2	2.9	1.6	6.7				
30y Opt																	

Matching Service Provided by IEnL □ Dual Int □ Sub Dual Int ■ Mkt Int ▲ Sub Mkt Int ■ Choice ▲ Sub Mkt Choice ▲ My B Int ▲ My S Int ▲ Firm's B Int ▲ Firm's S Int # Spread Int

Interest Blotter

Instruments	Fwd	Stk	PRICE	BUY Vol	SELL Vol	Status	Fin
- Outright							
EUR 1M 1Y			0.07	5.9		250	
EUR 1M 2Y			0.483	1.2	250		
EUR 3M2Y (CCP)					200		
EUR 1Y 3Y			0.957	2.1		250	
EUR 1Y6Y (CCP)					100		
EUR 3Y1Y (CCP)						200	
EUR 8Y2Y (CCP)						150	
EUR 8Y5Y (CCP)						75	

1,000 actions per second

Middlewares

Sagas

Reducers

React

ReactDOM

Browser Paint

MIDDLEWARE

To the rescue

RATE LIMITING

- Redux Throttle

```
const action = {  
  type: 'MY_ACTION',  
  meta: {  
    throttle: {  
      wait: 300,  
      leading: false  
    }  
  }  
}
```

RATE LIMITING

- Redux Throttle
- Redux Debounced

```
const action = {  
  type: 'MY_ACTION',  
  meta: {  
    debounce: {  
      time: 300,  
      leading: true,  
      trailing: false  
    }  
  }  
}
```

BATCHING

- redux-batched-actions

```
store.dispatch(  
  batchActions([  
    { type: 'ACTION1' },  
    { type: 'ACTION2' },  
    { type: 'ACTION3' },  
  ])  
)
```

BATCHING

- redux-batched-actions
- @manaflair/redux-batch

```
store.dispatch([
  { type: 'ACTION1' },
  { type: 'ACTION2' },
  { type: 'ACTION3' },
])
```

BATCHING ACTIONS

- redux-batched-actions
- @manaflair/redux-batch
- react-redux 7

```
import { batch } from "react-redux"

const { dispatch } = store

batch(() => {
  dispatch({ type: 'ACTION1' })
  dispatch({ type: 'ACTION2' })
  dispatch({ type: 'ACTION3' })
})
```



Dan Abramov
@dan_abramov



If your actions creator names start with set* and you often call multiple in a row, you might be missing the point of using Redux.

♥ 45 1:10 PM - Nov 20, 2016



BATCHING ACTIONS

- redux-batched-actions
- @manaflair/redux-batch
- react-redux 7
- redux-batched-subscribe

```
const debounceNotify = [REDACTED]
  debounce(notify => notify())

const store = createStore(
  reducer,
  [REDACTED],
  intialState,
  batchedSubscribe(debounceNotify)
)
```

REDUX-BATCHED-SUBSCRIBED

Async Action Batching

```
import raf from 'raf'
import { batchedSubscribe } from 'redux-batched-subscribe'

let notify = null
let rafId = null

const shouldBatch = action => action?.meta?.batch

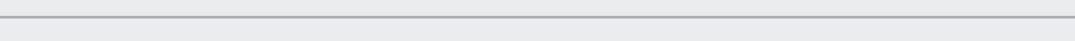
export const batchedSubscribeEnhancer = batchedSubscribe(freshNotify => (notify = freshNotify))

export const batchedSubscribeMiddleware = () => next => action => {
  const resolved = next(action)

  if (notify && rafId === null && !shouldBatch(action)) {
    notify()
  } else if (!rafId) {
    rafId = raf(() => {
      rafId = null
      notify()
    })
  }

  return resolved
}
```

ACTION PIPELINE



Middlewares

Sagas

Reducers

React

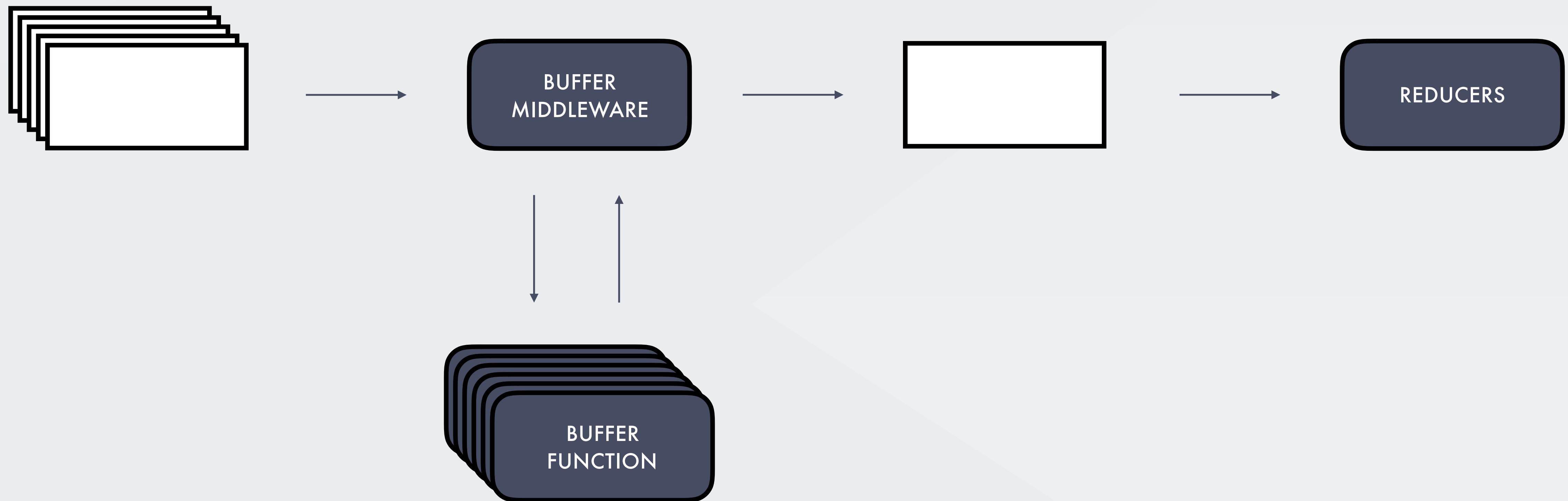
ReactDOM

Browser Paint

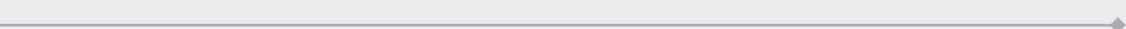
BUFFERING

*Can we turn multiple asynchronous actions
of the same type into a single action?*

BUFFERING



ACTION PIPELINE



Middlewares

Sagas

Reducers

React

ReactDOM

Browser Paint

REJECT MIDDLEWARE

```
function reject(action) {  
  console.warn('Dropped Action:', action.type)  
  return null  
}  
  
export default test => () => next => action =>  
  test(action) ? reject(action) : next(action)
```

```
rejectMiddleware(action => ignoredActionTypes.includes(action.type))
```



NOPE

A NOPE NINJA DELIVERS A WICKED DRAGON KICK



{ STOP THE ACTION OF ANOTHER PLAYER.
YOU CAN PLAY THIS AT ANY TIME. }

FIRST
EDITION

NOPE



WEBWORKERS

Not so fast

REDUCERS

Good store structure is key to performance

Reducers

Data Model

- Avoid modelling state after server API
- Avoid modelling state after what views like to consume

Reducers

Data Model

- Never hold duplicate data in the app state
- Never store derived data in the state

Reducers

Data Model

- Normalize nested objects
- Prefer Maps to Arrays

SERVER API

```
{  
  "total": 117,  
  "offset": 0,  
  "products": [  
    {  
      "id": "88cd7621-d3e1-42b7-b2b8-8ca82cdac2f0",  
      "title": "Blue Shirt",  
      "price": 9.99  
    },  
    {  
      "id": "aec17a8e-4793-4687-9be4-02a6cf305590",  
      "title": "Red Hat",  
      "price": 7.99  
    }  
  ]  
}
```

MAPPED DATA

```
{  
  "productsById": {  
    "88cd7621-d3e1-42b7-b2b8-8ca82cdac2f0": {  
      "title": "Blue Shirt",  
      "price": 9.99  
    },  
    "aec17a8e-4793-4687-9be4-02a6cf305590": {  
      "title": "Red Hat",  
      "price": 7.99  
    }  
  }  
}
```

MAPPED DATA

```
{  
  "productsById": {  
    "88cd7621-d3e1-42b7-b2b8-8ca82cdac2f0": {  
      "title": "Blue Shirt",  
      "price": 9.99  
    },  
    "aec17a8e-4793-4687-9be4-02a6cf305590": {  
      "title": "Red Hat",  
      "price": 7.99  
    }  
  },  
  "productIds": [  
    "88cd7621-d3e1-42b7-b2b8-8ca82cdac2f0",  
    "aec17a8e-4793-4687-9be4-02a6cf305590"  
  ]  
}
```

redux-ignore

Higher-order reducer to ignore redux actions

```
import { ignoreActions, filterActions } from 'redux-ignore'

ignoreActions(reducer, [ARRAY_OF_ACTIONS])
ignoreActions(reducer, (action) => !action.valid)

filterActions(reducer, [ARRAY_OF_ACTIONS])
filterActions(reducer, (action) => action.valid)
```

redux-log-slow-reducers

Logs a warning if any reducer takes over 8ms

```
import { combineReducers } from 'redux'  
import { pipe } from 'ramda'  
import logSlowReducers from 'redux-log-slow-reducers'  
  
export default (process.env.NODE_ENV === 'production'  
  ? combineReducers  
  : pipe(  
    logSlowReducers,  
    combineReducers  
  ))
```

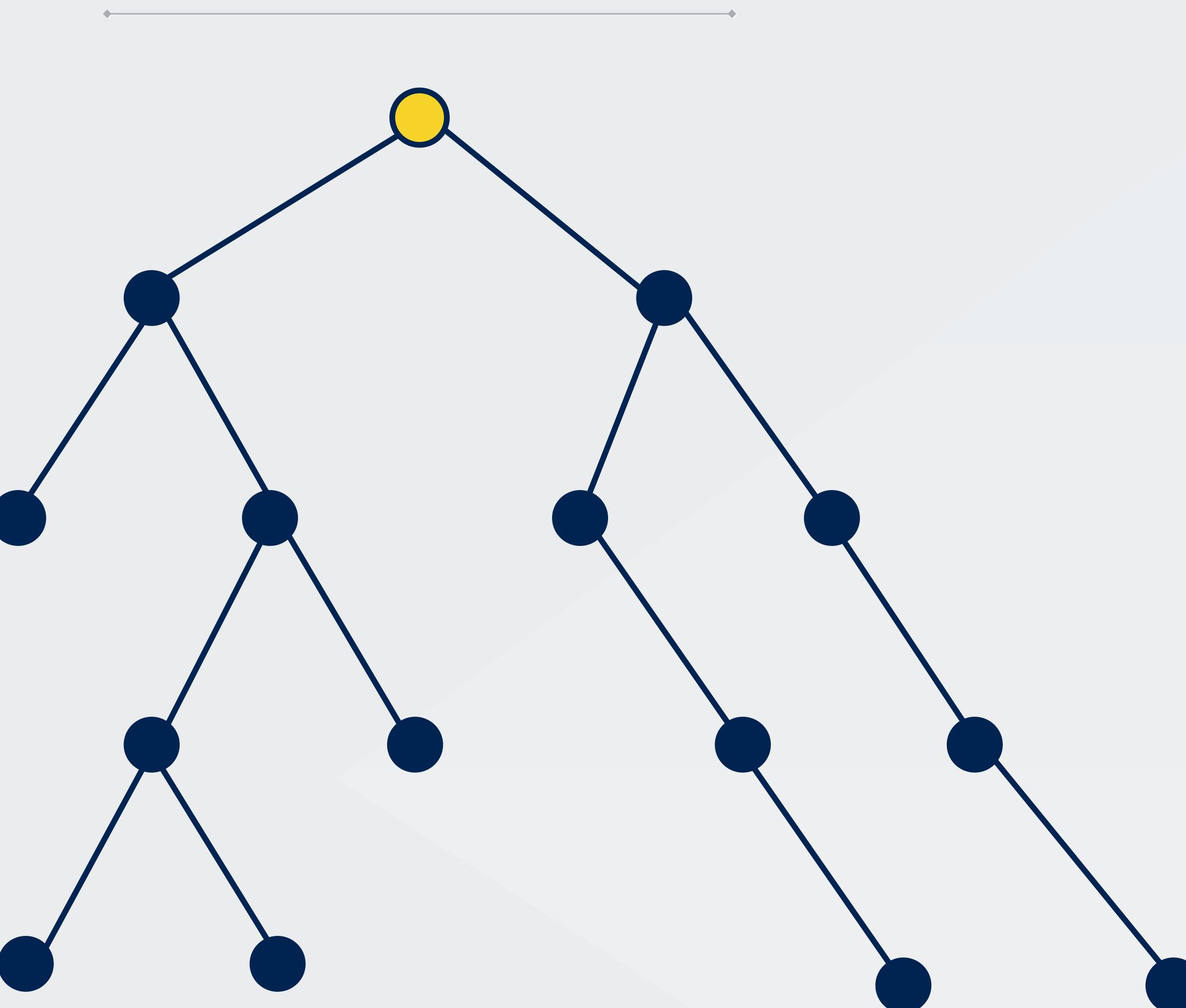
REACT-REDUX

Help it help you

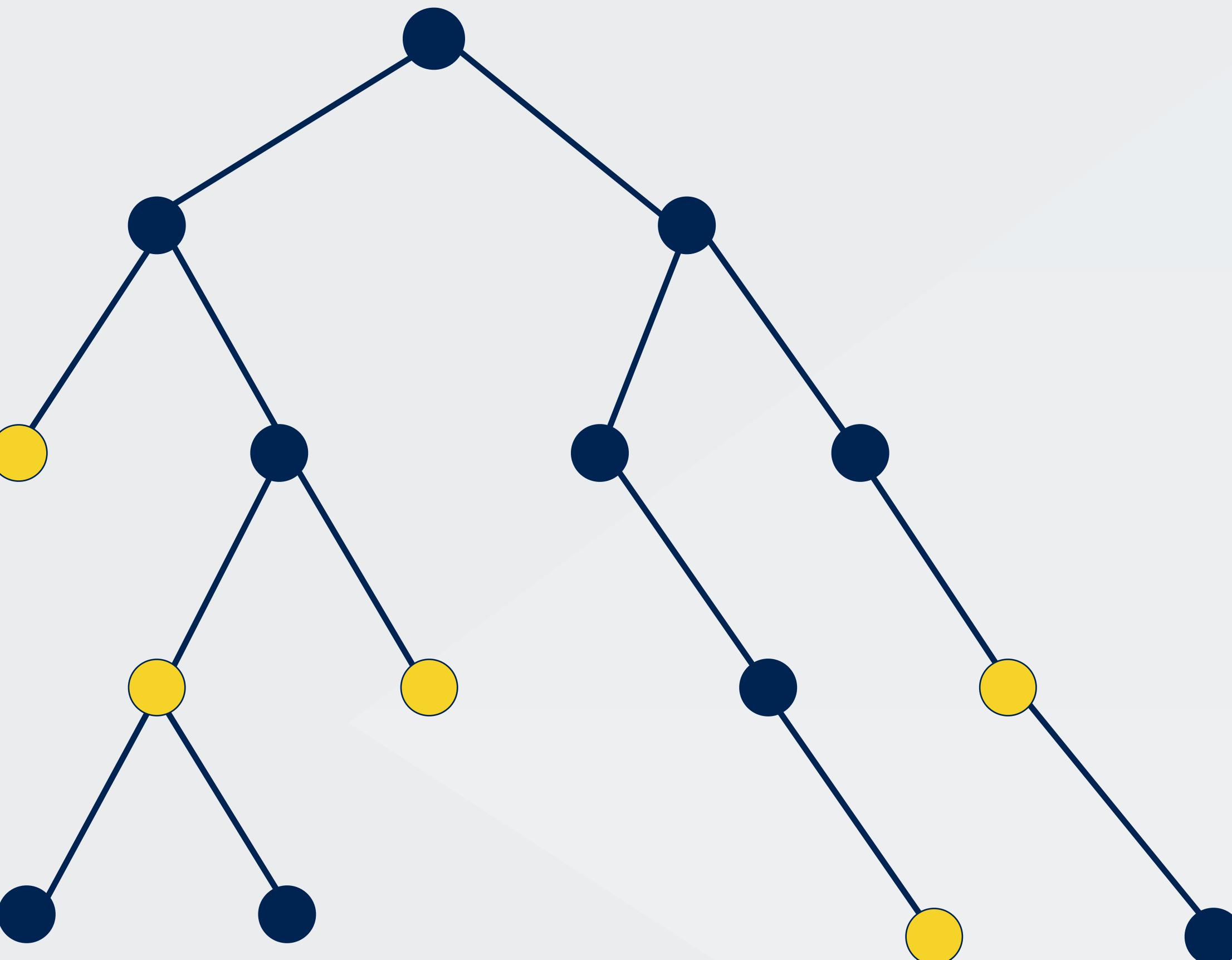
CONNECT()

Implements many performance optimizations internally, so that your own component only re-renders when it actually needs to

CONNECT()

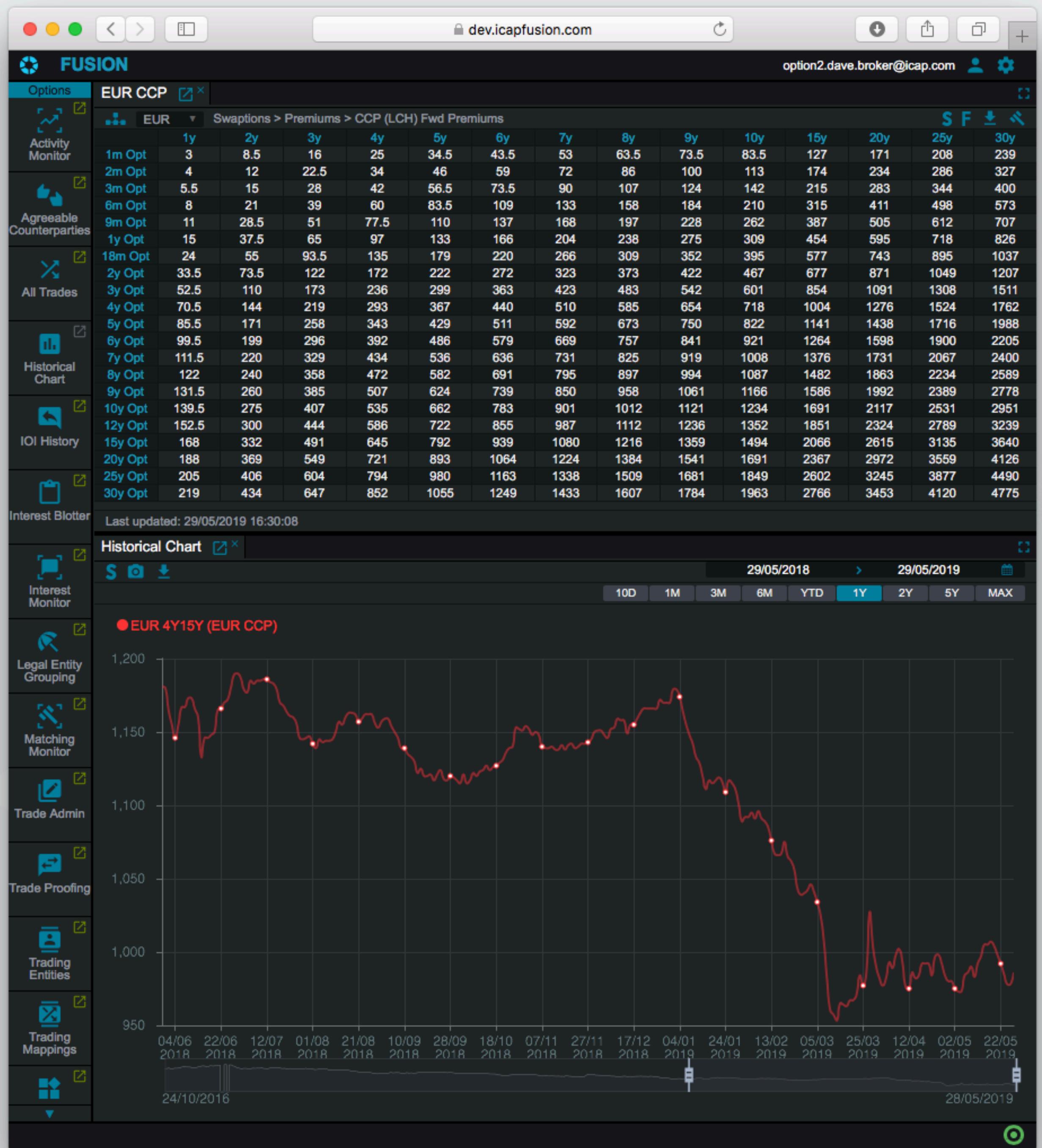


CONNECT()



FUSION OPTIONS

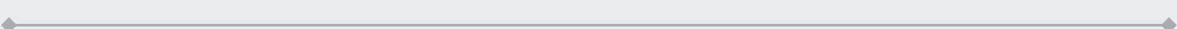
*Every cell on the grid has it's own
instance of connect*



CONNECT()

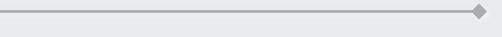
```
connect(mapStateToProps, mapDispatchToProps)(Component)
```

CONNECT()



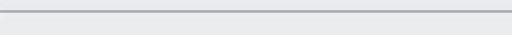
```
connect(mapStateToProps, mapDispatchToProps, mergeProps, options)(Component)
```

Options



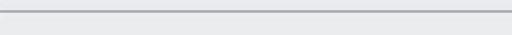
```
// spec
{
  context?: Object,
  pure?: boolean,
  areStatesEqual?: Function,
  areOwnPropsEqual?: Function,
  areStatePropsEqual?: Function,
  areMergedPropsEqual?: Function,
  forwardRef?: boolean,
}
```

Options



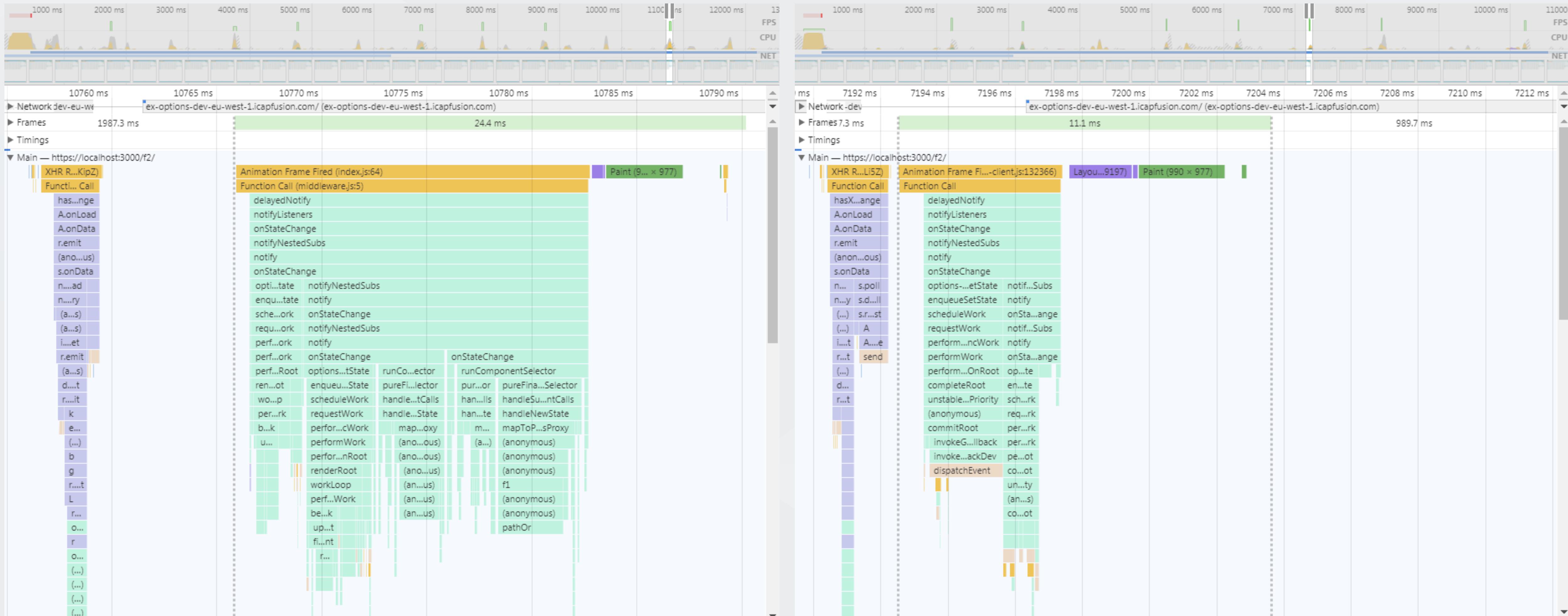
```
// Default values
{
  context: undefined,
  pure: true,
  areStatesEqual: (next, prev) => next === prev,
  areOwnPropsEqual: shallowEqual,
  areStatePropsEqual: shallowEqual,
  areMergedPropsEqual: shallowEqual,
  forwardRef: false,
}
```

Options



```
// Faster values
{
  areStatesEqual: (next, prev) => next.grid === prev.grid,
  areOwnPropsEqual: () => true,
  areMergedPropsEqual: () => true,
}
```

Options



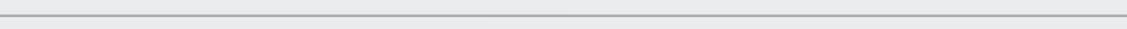
24.4 ms

11.1 ms

mergeProps

The forgotten parameter

mergeProps



```
function defaultMergeProps(stateProps, dispatchProps, ownProps) {  
  return {  
    ...ownProps,  
    ...stateProps,  
    ...dispatchProps  
  }  
}
```

mergeProps

```
import * as actionCreators from './actionCreators'

function mapStateToProps(state) {
  return { todos: state.todos }
}

function mergeProps(stateProps, dispatchProps, ownProps) {
  return Object.assign({}, ownProps, {
    todos: stateProps.todos[ownProps.userId],
    addTodo: (text) => dispatchProps.addTodo(ownProps.userId, text)
  })
}

export default connect(mapStateToProps, actionCreators, mergeProps)(TodoApp)
```

```
( ) => null !== () => null
```

mergeProps

```
import * as actionCreators from './actionCreators'

function mapStateToProps(state) {
  return { todos: state.todos }
}

function mergeProps(stateProps, dispatchProps, ownProps) {
  return Object.assign({}, ownProps, {
    todos: stateProps.todos[ownProps.userId],
    addTodo: (text) => dispatchProps.addTodo(ownProps.userId, text)
  })
}

export default connect(mapStateToProps, actionCreators, mergeProps)(TodoApp)
```

mergeProps

```
import { identity, memorizeWith } from 'ramda'
import * as actionCreators from './actionCreators'

const memorize = memorizeWith(identity)
const addTodo = memorize(
  (userId, dispatchProps) => (text) => dispatchProps.addTodo(userId, text)
)

function mapStateToProps(state) {
  return { todos: state.todos }
}

function mergeProps(stateProps, dispatchProps, ownProps) {
  return Object.assign({}, ownProps, {
    todos: stateProps.todos[ownProps.userId],
    addTodo: addTodo(ownProps.userId, dispatchProps)
  })
}

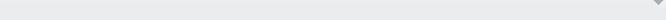
export default connect(mapStateToProps, actionCreators, mergeProps)(TodoApp)
```

mapStateToProps

```
mapStateToProps = state => ({  
  coords: {  
    x: state.map.latitude,  
    y: state.map.longitude  
  }  
})
```

{ } != { }

Reselect



mapStateToProps

- `(state) => { ... }`
- `(state, ownProps) => { ... }`
- `() => (state) => { ... }`
- `() => (state, ownProps) => { ... }`

CONCLUSION

- Normalize Data
- Memorize Selectors
- Optimize Rendering

*"There is no doubt that the grail of efficiency leads to abuse. Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: **Premature optimisation is the root of all evil.** Yet we should not pass up our opportunities in that critical 3%."*



Donald Knuth - Structured Programming with Goto statements (1974)