Redux framework

Grafana uses Redux Toolkit to handle Redux boilerplate code.

> Some of our Reducers are used by Angular and therefore state is to be considered as mutable for those reducers.

Test functionality

reducerTester

Fluent API that simplifies the testing of reducers

Usage

```
reducerTester()
  .givenReducer(someReducer, initialState)
  .whenActionIsDispatched(someAction('reducer tests'))
  .thenStateShouldEqual({ ...initialState, data: 'reducer tests' });
```

Complex usage Sometimes you encounter a `resulting state` that contains properties that are hard to compare, such as `Dates`, but you still want to compare that other props in state are correct.

Then you can use `thenStatePredicateShouldEqual` function on `reducerTester` that will return the `resulting state` so that you can expect upon individual properties..

```
reducerTester()
  .givenReducer(someReducer, initialState)
  .whenActionIsDispatched(someAction('reducer tests'))
  .thenStatePredicateShouldEqual((resultingState) => {
    expect(resultingState.data).toEqual('reducer tests');
    return true;
  });
```

thunkTester

Fluent API that simplifies the testing of thunks.

Usage

```
const dispatchedActions = await thunkTester(initialState).givenThunk(someThunk).whenThunkIsD

expect(dispatchedActions).toEqual([someAction('reducer tests')]);
```

Typing of connected props

It is possible to infer connected props automatically from `mapStateToProps` and `mapDispatchToProps` using a helper type `ConnectedProps` from Redux. For this to work the `connect` call has to be split into two parts.

```
import { connect, ConnectedProps } from 'react-redux';

const mapStateToProps = (state: StoreState) => {
  return {
    location: state.location,
    initDone: state.panelEditor.initDone,
    uiState: state.panelEditor.ui,
  };
};

const mapDispatchToProps = {
  updateLocation,
  initPanelEditor,
  panelEditorCleanUp,
  setDiscardChanges,
  updatePanelEditorUIState,
  updateTimeZoneForSession,
};

const connector = connect(mapStateToProps, mapDispatchToProps);

type Props = OwnProps & ConnectedProps<typeof connector>;

class PanelEditorUnconnected extends PureComponent<Props> {}

export const PanelEditor = connector(PanelEditorUnconnected);
```

For more examples, refer to the Redux docs.