Instructions for TVC Redux without React Counter App

1. Install redux

$ yarn add redux
$ yarn start

or ...

$ npm install redux
$ npm start

2. Make action creators:

create actions/counter.js

```
export const incrementCounter = () => {
  return { type: 'INCREMENT_COUNTER' }
}

export const decrementCounter = () => {
  return { type: 'DECREMENT_COUNTER' }
}

export const resetCounter = () => {
  return { type: 'RESET_COUNTER' }
}

export const setCounter = (num) => {
  return { type: 'SET_COUNTER', payload: num }
}
```

3. Import actions

add to index.js line 5

```
import {
  incrementCounter,
  decrementCounter,
  resetCounter,
  setCounter
} from './actions/counter'
```

4. Test actions in browser

add to index.js line 42

```
console.log(incrementCounter())
```

Test for all buttons.

5. Add reducers

create reducers/index.js

```
import { combineReducers } from 'redux'

import { counter } from './counter'

const reducers = combineReducers({
  counter
})

export default reducers
```

create reducers/counter.js

```
const INITIAL_COUNTER = { number: 0 }

export function counter(state = INITIAL_COUNTER, action) {

  console.log('state', state)
  console.log('action', action)

  return state
}
```

create store/configureStore.js

```
import { createStore } from 'redux'
import reducers from '../reducers'

export default function configureStore() {
  return createStore(
    reducers,
    window.__REDUX_DEVTOOLS_EXTENSION__ &&
    window.__REDUX_DEVTOOLS_EXTENSION__()
  )
}
```

add to index.js line 14

```
import configureStore from './store/configureStore'
const store = configureStore()
```

Test to make sure no errors.

6. Test reducer call

change index.js line 43

```
store.dispatch(incrementCounter())
```

Do rest of case statements and test.

7. Implement reducers

change reducers/counter.js line 4

```
let newState
switch (action.type) {

  case 'INCREMENT_COUNTER':
    return { number: state.number + 1}

  case 'DECREMENT_COUNTER':
    newState = Object.assign({}, state)
    if (newState.number>1)
      newState.number--
    else
      newState.number = 0
    return newState

  case 'RESET_COUNTER':
    return INITIAL_COUNTER

  case 'SET_COUNTER':
    newState = Object.assign({}, state)
    newState.number = action.payload
    newState = { ...state, number: action.payload }
    return newState

  default:
    return state
}
```

change/add index.js line 63

```
let newNum = Math.round($counterSetField.value)
if (newNum >= 0)
  store.dispatch(setCounter(newNum))
$counterSetField.value = 0
```

add index.js line 63

```
console.log('counter', store.getState().counter.number)
```

uncomment index.js line 29

```
const $counterSetField  = document.querySelector('#counter-
    set-field')
```

Test in browser.

8. Implement UI

change index.js line 19

```
const updateCounter = () => $counterDisplay.innerHTML =
    store.getState().counter.number
```

change index.js line 23

```
store.subscribe(updateCounter)
```

uncomment index.js line 28

```
const $counterBtns      = document.querySelectorAll('.btn-
    counter')
```

remove index.js line 66

```
console.log('counter', store.getState().counter.number)
```

Test fully functioning app.