Unit 6: Actions, Reducers

Actions

Actions:

Actions are payloads of information that send data from your application to your store. They are the *only* source of information for the store. You send them to the store using store.dispatch()

```
{
  type: ADD_TODO,
  text: 'Build my first Redux app'
}
```

Action Creators

```
function addTodo(text) {
  return {
    type: ADD_TODO,
    text
  }
}
```

Reducers

Actions describe the fact that *something happened*, but don't specify how the application's state changes in response. This is the job of reducers.

Note: This contains the most important parts when building a react-redux app

Reducers step by step

Designing the State Shape

(try to design state of a todo app before showing the next slide)

Reducers

```
{
  todos: [
    {
      text: 'Consider using Redux',
      completed: true,
    },
    {
      text: 'Keep all state in a single tree',
      completed: false
    }
  ]
}
```

Handling Actions

Now that we've decided what our state object looks like, we're ready to write a reducer for it. The reducer is a pure function that takes the previous state and an action, and returns the next state.

```
(previousState, action) => newState
```

Things you should never do inside a reducer:

Mutate its arguments;

Perform side effects like API calls and routing transitions;

Call non-pure functions, e.g. `Date.now()` or `Math.random()`.

===> Given the same arguments, it should calculate the next state and return it. No surprises. No side effects. No API calls. No mutations. Just a calculation.

Reducers

(reducers eg in the code sample)