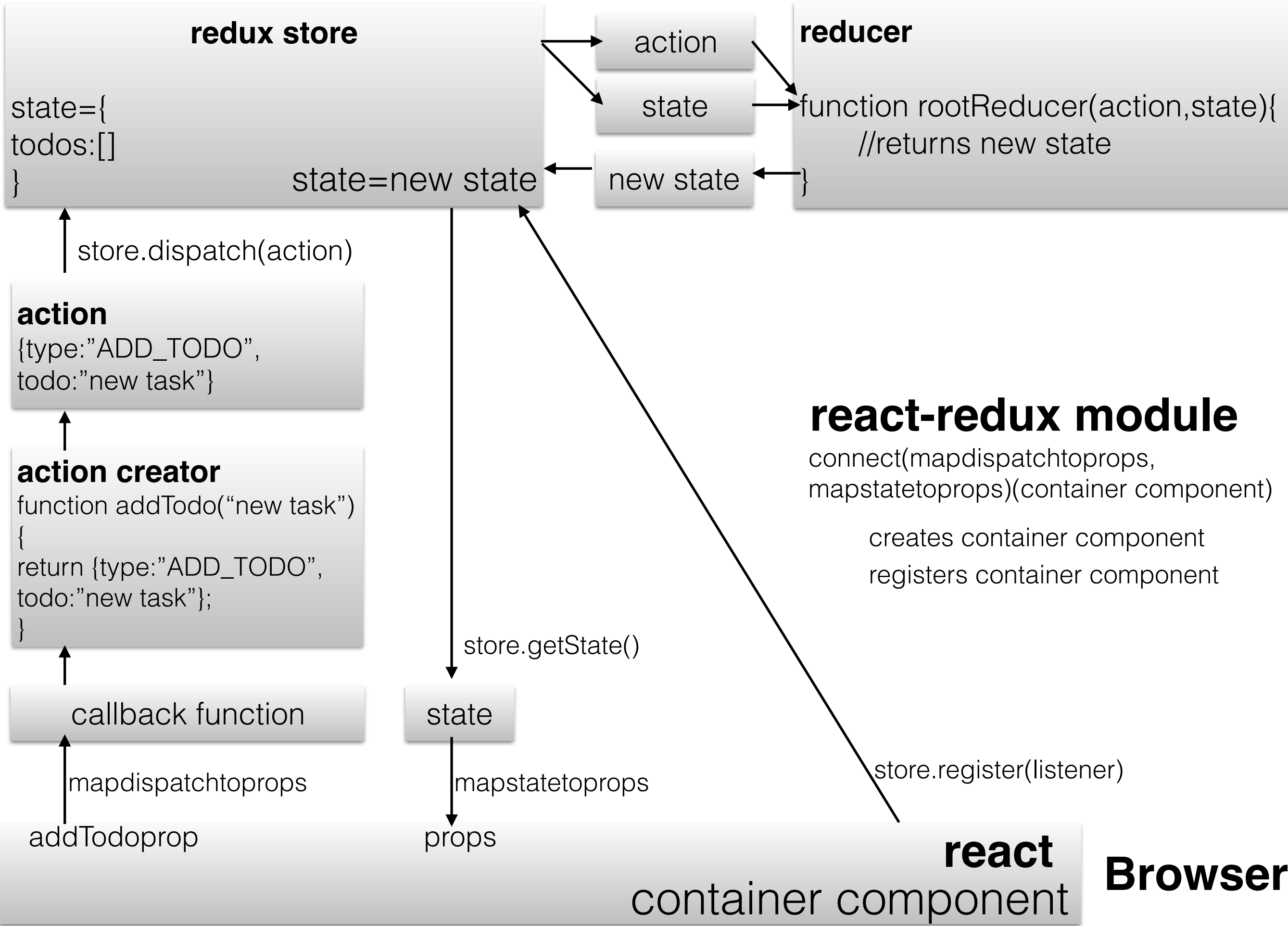


store = createStore(rootReducer, initialState={todos:[]})



react-redux connect function

```
connect(mapStateToProps,mapDispatchToProps)(containerComponent);
```

1-instantiates the container component and registers it with the redux store

2-manages the mounting and unmounting life cycle of the container component

3-maps props of the container component to the redux store state
via mapStateToProps function

4-maps props of the container component to callbacks that call store.dispatch(action)
via mapDispatchToProps

mapStateToProps

complete state mapping

```
store.state = {  
  todos:[]  
}
```

```
//returns an object that maps the state to props  
function mapStateToProps(state){  
  return {stateprop: state};  
}
```

```
//access store.state inside component  
this.props.stateprop;
```

mapStateToProps

granular state mapping: users

```
store.state = {  
  users:['areg','kobi'],  
  todos:[]  
}
```

```
//react container component  
class usersPage{..}
```

```
function mapStateToProps(state){  
  return {users:state.users};  
}
```

```
connect(mapStateToProps,mapDispatchToProps)(usersPage);
```

```
//access store.state.users  
this.props.users;
```

mapStateToProps

granular state mapping: todos

```
store.state = {  
  users:['areg','kobi'],  
  todos:[]  
}
```

```
//react container component  
class todosPage{...}
```

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

```
connect(mapStateToProps,mapDispatchToProps)(todosPage);
```

```
//access store.state.users  
this.props.todos;
```

mapDispatchToProps

Returns an object that maps callback functions to props

```
function mapDispatchToProps(dispatch){  
  return {propname: callbackFunction};  
}
```

```
function mapDispatchToProps(dispatch){  
  return {propname1: callbackFunction1,  
          propname2: callbackFunction2  
        };  
}
```

The callback function does two things:

1- calls an action creator to get an action

2- calls store.dispatch(action)

to dispatch the action returned by the action creator

mapDispatchToProps

```
function mapDispatchToProps(dispatch){
  return {
    //map the prop addTodoItemProp to the anonymous callback function
    addTodoItemprop : function(todoitem){
      //when this function is called:
      //calls action creator to get an action
      //calls store.dispatch(action) to dispatch the action to the store
    }
  };
}

//trigger the call to store.dispatch using the prop name
this.props.addTodoItemprop("do this later");
```

mapDispatchToProps

```
function addTodoItemActionCreator(todoItem){
  addTodoItemAction = {
    type:"ADD_TODO",
    todo: todoItem
  }
  return addTodoItemAction;
}

function mapDispatchToProps(dispatch){
  return {
    //map the prop addTodoItemProp to the anonymous callback function
    addTodoItemprop : function(todoitem){
      //get action from action creator
      action = addTodoItemActionCreator(todoitem);
      //call store.dispatch(action) passing the action to the redux store
      dispatch(action);
    }
  };
}

//trigger the call to store.dispatch using the prop name
this.props.addTodoItemprop("do this later");
```


mapDispatchToProps

```
function addTodoItemActionCreator(todoItem){  
  return{  
    type:"ADD_TODO",  
    todo: todoItem  
  };  
}
```

```
function mapDispatchToProps(dispatch){  
  return {  
    //map the prop addTodoItemProp to the anonymous callback function (closure)  
    addTodoItemprop: todoitem =>{  
      dispatch(addTodoItemActionCreator(todoitem));  
    }  
  };  
}
```

```
//call store.dispatch(addTodoItemActionCreator(todoItem)) using the prop name  
this.props. addTodoItemprop("do this later");
```

mapDispatchToProps

//renaming the same example

//without "ActionCreator" appended to action creator function name

//without "prop" appended to the prop name in the mapping object

//the renamed addTodoItemActionCreator action creator function name

```
function addTodoItem(todoItem){
```

```
  return {
```

```
    type:"ADD_TODO",
```

```
    todo: todoItem
```

```
  };
```

```
}
```

```
function mapDispatchToProps(dispatch){
```

```
  return {
```

```
    //the renamed prop name
```

```
    addTodoItem: todoitem=>{
```

```
      dispatch(addTodoItem(todoitem));
```

```
    }
```

```
  };
```

```
}
```

//calling the renamed prop name

```
this.props.addTodoItem("do this later");
```

mapDispatchToProps

//multiple actions with query and command mapping examples

```
function getTodosItems(){
  return {
    type:"GET_TODOS",
    todos: ["do that thing", "do another thing"]
  };
}
```

```
function addTodoItem(todoItem){
  return {
    type:"ADD_TODO",
    todo: todoItem
  };
}
```

```
function mapDispatchToProps(dispatch){
  return {
    addTodoItem:todoItem=>{
      dispatch(addTodoItem(todoItem));
    },
    getTodosItems:()=>{
      dispatch(getTodosItems());
    }
  };
}
```

```
this.props.addTodoItem("do this later");
todoItems = this.props.getTodosItems();
```

bindActionCreators helper

```
//create an actions mapping object
```

```
actions = {  
  addTodoItemprop: addTodoItemActionCreator,  
  getTodoItemsprop: getTodoItemsActionCreator  
}
```

```
//above with renamed action creators
```

```
actions = {  
  addTodoItem: addTodoItem,  
  getTodoItems: getTodoItems  
}
```

```
function mapDispatchToProps(dispatch){  
  return{  
    actions: bindActionCreators(actions,dispatch)  
  };  
}
```

```
prop.actions.addTodoItem("add task todo");  
todoitems =prop.actions.getTodoItems();
```

bindActionCreators helper

```
function mapDispatchToProps(dispatch){  
  return {  
    addTodoItem:todoitem=>{  
      dispatch(addTodoItem(todoitem));  
    },  
    getTodoItems:()=>{  
      dispatch(getTodoItems());  
    }  
  };  
}
```

```
prop.addTodoItem("add task todo");  
todoitems = prop.getTodoItems();
```

Becomes:

```
function mapDispatchToProps(dispatch){  
  return{  
    actions: bindActionCreators(actions,dispatch)  
  };  
}
```

```
prop.actions.addTodoItem("add task todo");  
todoitems =prop.actions.getTodoItems();
```

Redux-Thunk

-Redux Middleware that calls a function passing to that function the store.dispatch function and optionally the store.getState function as arguments.

-We write a function that returns the function that redux-thunk middleware calls.

This function that we write is an Action Function Creator.

-We pass the function returned by the Action Function Creator to the redux-thunk middleware by calling:

```
actionFunction = actionFunctionCreatorFunc();
```

```
//the actionFunction that we pass to redux-thunk that redux calls back  
store.dispatch(actionFunction);
```

-Just like when we write a function to return an action object and then pass that object to store.dispatch:

```
actionObject = actionObjectCreatorFunc();
```

```
store.dispatch(actionObject);
```

Redux-Thunk setup

```
import { createStore, applyMiddleware } from  
  'redux';  
import thunk from 'redux-thunk';  
import rootReducer from './reducers/index';  
  
store = createStore(  
  rootReducer,  
  applyMiddleware(thunk)  
);
```

Normal Action Dispatch (Revisited)

//ACTION **OBJECT** CREATOR

```
function addTodoItem(todoItem){  
  return {  
    type:"ADD_TODO",  
    todo: todoItem  
  };  
}
```

```
function mapDispatchToProps(dispatch){  
  return {  
    addTodoItem: todoItem=>{  
      dispatch(addTodoItem(todoItem));  
    }  
  };  
}
```

//calling the renamed prop name
this.props.addTodoItem("do this later");

Async Action Dispatch

//ACTION **FUNCTION** CREATOR

```
function addTodoItemAsync(todoItem) {  
  return function (dispatch) {  
    addTodoItemRemote(todoItem).then(  
      todoItem => dispatch(addTodoItem(todoItem)),  
      error => dispatch(displayErrorAC('Oops', todoItem, error))  
    );  
  };  
}
```

```
function mapDispatchToProps(dispatch){  
  return {  
    addTodoItem: todoItem=>{  
      dispatch(addTodoItemAsync(todoItem));  
    }  
  };  
}
```

```
//calling the renamed prop name  
this.props.addTodoItem("do this later");
```

Action Function Creator

```
//ACTION FUNCTION CREATOR
```

```
function addTodoItemAsync(todoItem) {  
  return function (dispatch) {  
    addTodoItemRemote(todoItem).then(  
      todoItem => dispatch(addTodoItem(todoItem)),  
      error => dispatch(displayError('Oops', todoItem, error))  
    );  
  };  
}
```

```
//helper function for remote API call
```

```
function addTodoItemRemote(todoItem) {  
  //Post todo item to server. Server returns added todo item.  
  return fetch('https://api.todos.com/addTodoItem',todoItem);  
}
```

```
//action creator
```

```
function addTodoItem(todoItem){  
  return {  
    type:"ADD_TODO",  
    todo: todoItem  
  };  
}
```