2.10 Create Reducers For Each Action

Reducers are functions that receive state and action objects from a Redux store and return a new state to be stored back into Redux.

It's important not to directly modify the given state here. Reducers must be **pure functions** and must return a **new state**.

- Reducer functions are called from the **Container** that will be created when a user action occurs.
- When the Reducer returns a state, **Redux passes the new state** to each component, and **React renders each component** again.

2.10.1 Immutable Data Structures

- JavaScript primitive data type(number, string, boolean, undefined, and null) => **immutable**
- Object, array and function => **mutable**

Changes to the data structure are known to be buggy. Since our store consists of state objects and arrays, we need to implement **a strategy to keep the state immutable**.

There are three ways to change the state here:

ES5

```
// Example One
state.foo = '123';
// Example Two
Object.assign(state, { foo: 123 });
// Example Three
var newState = Object.assign({}, state, { foo: 123 });
```

In the example above, the first and second mutate the state object. The second example mutates because **Object.assign()** merges all its arguments into the first argument.

The third example **doesn't mutate the state**. It merges the contents of state and { foo: 123 } into a **new empty object** which is the first argument.

The spread operator introduced in **ES6** provides a simpler way to keep the state immutable.

ES6 (ES2015)

```
const newState = { ...state, foo: 123 };
```

> *For more information about the spread operator, see here.*

2.10.2 Create a Reducer for ChangeClimate

First, we will create `ChangeClimate` through **test-driven development** method.

In Part1, our app was generated through **create-react-app**, so we basically use `jest` as test runner.

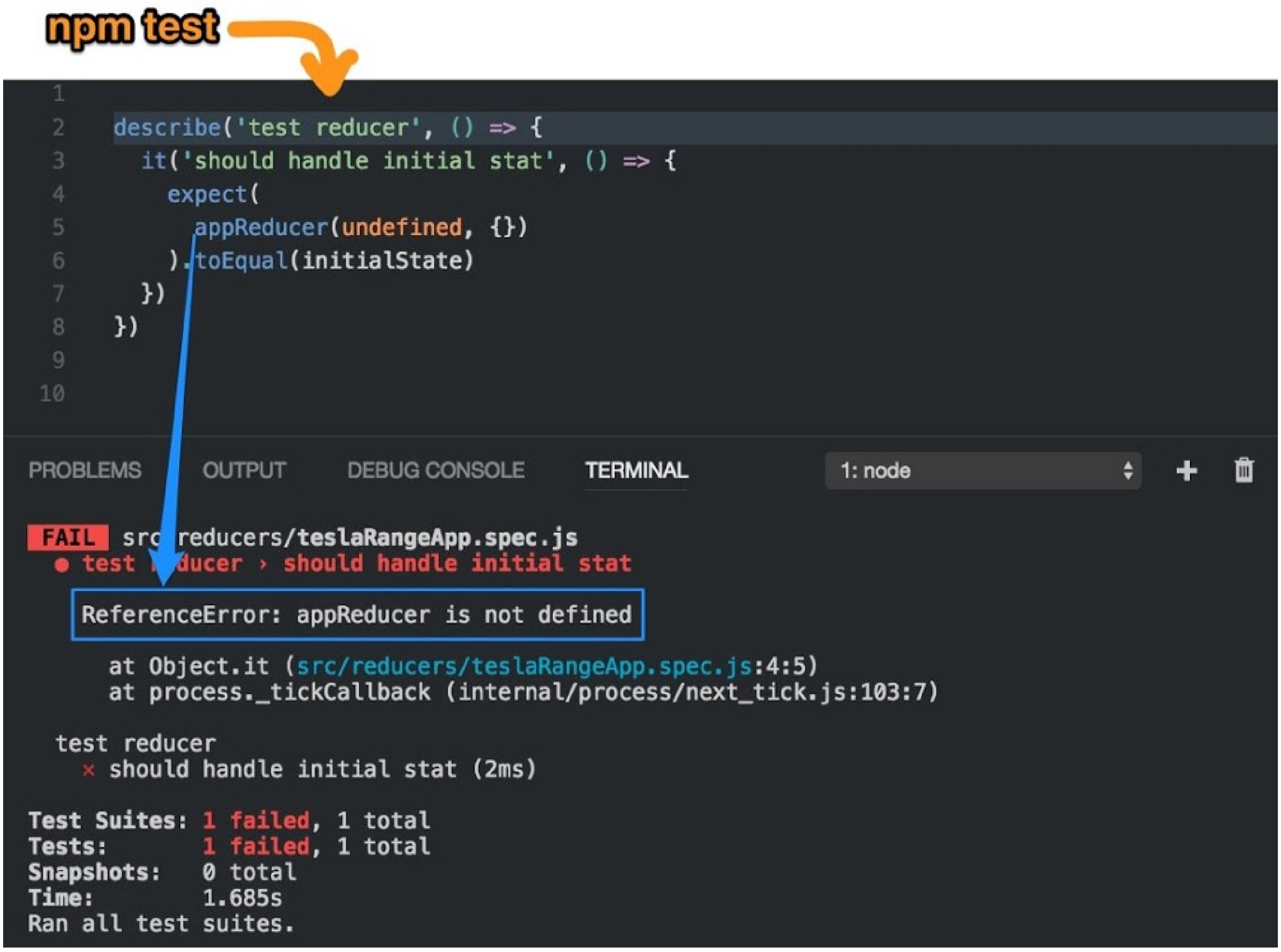
The jest looks for a test file using one of the following naming conventions:

```
Files with .js suffix in __tests__ folders
Files with .test.js suffix
Files with .spec.js suffix
```

Create **teslaRangeApp.spec.js** in src/reducers and create a test case.

```
describe('test reducer', () => {
  it('should handle initial stat', () => {
    expect(
      appReducer(undefined, {})
    ).toEqual(initialState)
  })
})
```

After create the test, run the `npm test` command. You should be able to see the following test failure message. This is because we have not written the **appReducer** yet.



```
describe('test reducer', () => {
  it('should handle initial stat', () => {
    expect(
      appReducer(undefined, {})
    ).toEqual(initialState)
  })
})
```

```
FAIL  src/reducers/teslaRangeApp.spec.js
  ● test reducer › should handle initial stat

    ReferenceError: appReducer is not defined

      at Object.it (src/reducers/teslaRangeApp.spec.js:4:5)
      at process._tickCallback (internal/process/next_tick.js:103:7)

  test reducer
    × should handle initial stat (2ms)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        1.685s
Ran all test suites.
```

To make the first test successful, we need to create **teslaRangeApp.js** in the same reducers directory and write **initial state and reducer** functions.

src/reducers/teslaRangeApp.js

```
const initialState = {
  carstats:[
    {miles:246, model:"60"},
    {miles:250, model:"60D"},
    {miles:297, model:"75"},
    {miles:306, model:"75D"},
    {miles:336, model:"90D"},
    {miles:376, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: true,
    wheels: 19
  }
}
function appReducer(state = initialState, action) {
  switch (action.type) {
    default:
      return state
  }

}
export default appReducer;
```

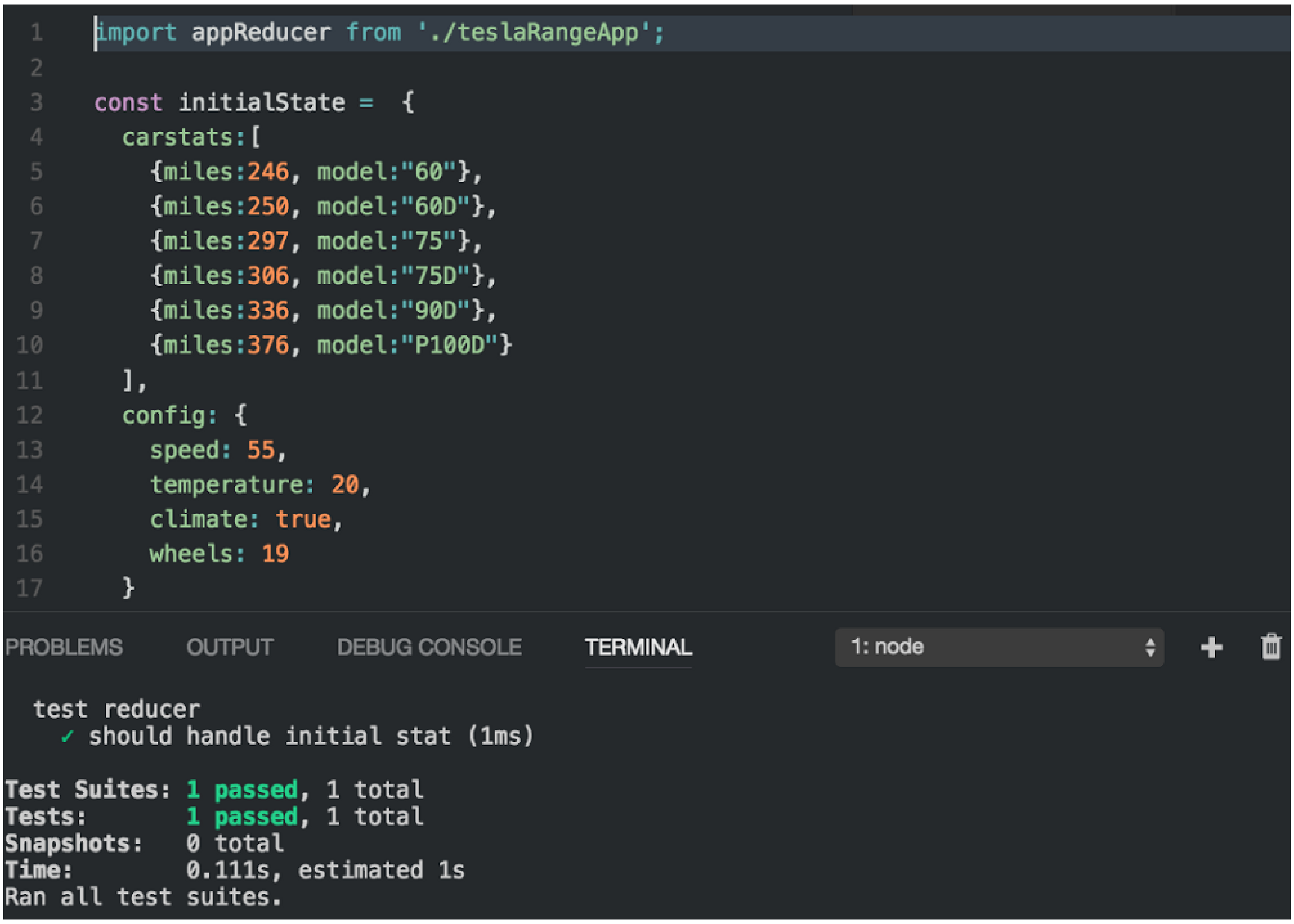
Next, import teslaRangeApp.js from teslaRangeApp.spec.js and set initialState.

src/reducers/teslaRangeApp.spec.js

```
import appReducer from './teslaRangeApp';
const initialState =  {
  carstats:[
    {miles:246, model:"60"},
    {miles:250, model:"60D"},
    {miles:297, model:"75"},
    {miles:306, model:"75D"},
    {miles:336, model:"90D"},
    {miles:376, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: true,
    wheels: 19
  }
}
describe('test reducer', () => {
  it('should handle initial stat', () => {
    expect(
      appReducer(undefined, {})
    ).toEqual(initialState)
  })
})
```

Run npm test again and the test will succeed.

In the current test case, the action type is {}, so the **initialState** is returned.



```
import appReducer from './teslaRangeApp';

const initialState =  {
  carstats:[
    {miles:246, model:"60"},
    {miles:250, model:"60D"},
    {miles:297, model:"75"},
    {miles:306, model:"75D"},
    {miles:336, model:"90D"},
    {miles:376, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: true,
    wheels: 19
  }
```

```
  test reducer
    ✓ should handle initial stat (1ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.111s, estimated 1s
Ran all test suites.
```

Now let’s test the **CHANGE_CLIMATE** action.

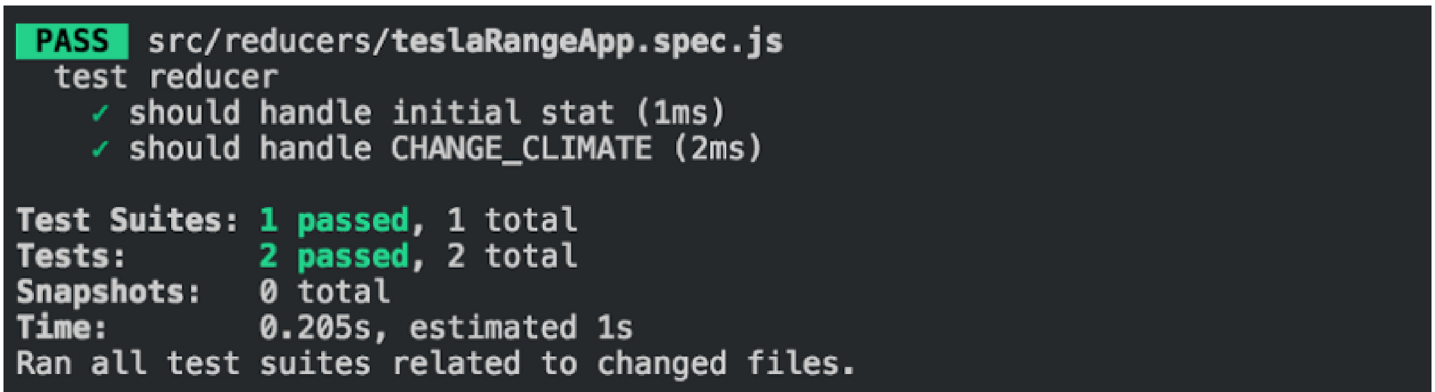
Add the following climateChangeState and CHANGE_CLIMATE test cases to teslaRangeApp.spec.js.

```
const climateChangeState = {
  carstats:[
    {miles:267, model:"60"},
    {miles:273, model:"60D"},
    {miles:323, model:"75"},
    {miles:334, model:"75D"},
    {miles:366, model:"90D"},
    {miles:409, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: false,
    wheels: 19
  }
}
it('should handle CHANGE_CLIMATE', () => {
    expect(
      appReducer(initialState,{
        type: 'CHANGE_CLIMATE'
      })
    ).toEqual(climateChangeState)
  })
```

Then add the **CHANGE_CLIMATE** case, **updateStats**, and **calculateStats**functions to teslaRangeApp.js. Then import the **BatteryService.js** that was used in part 1.

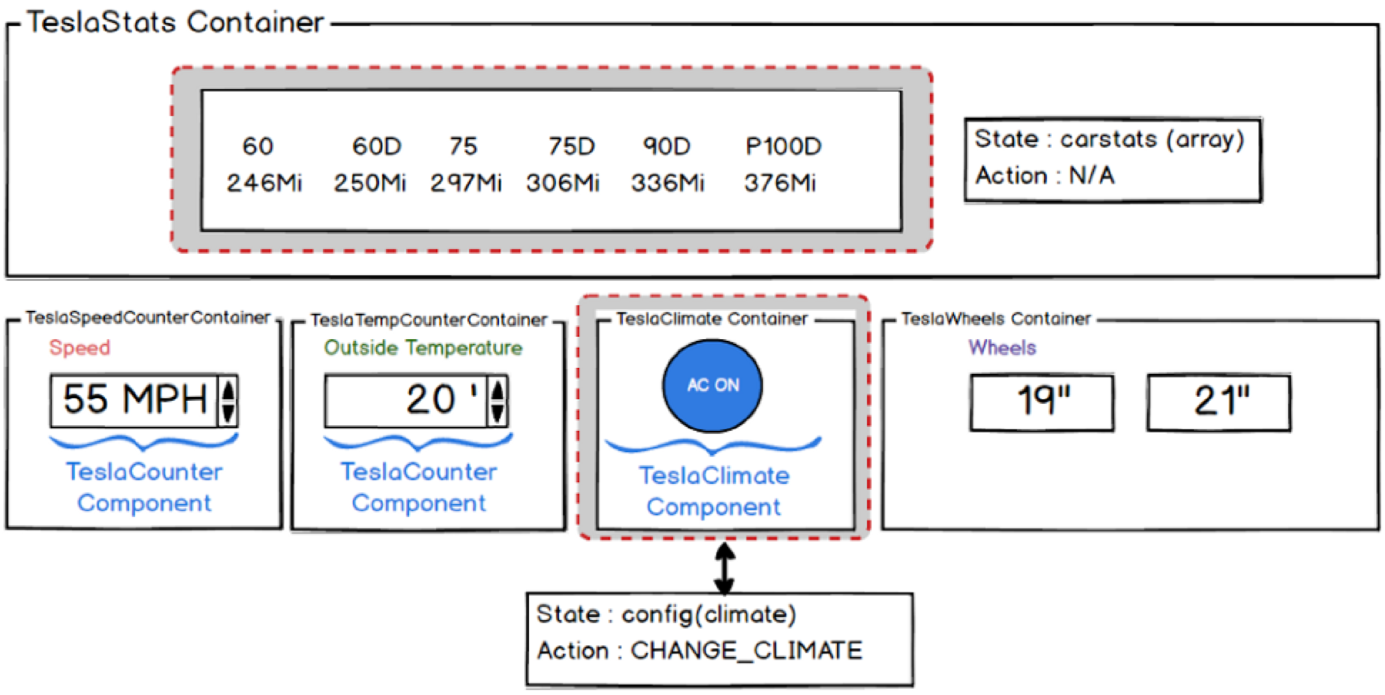
```
import { getModelData } from '../services/BatteryService';
function updateStats(state, newState) {
  return {
    ...state,
    config:newState.config,
    carstats:calculateStats(newState)
  }
}
function calculateStats(state) {
  const models = ['60', '60D', '75', '75D', '90D', 'P100D'];
  const dataModels = getModelData();
  return models.map(model => {
    const { speed, temperature, climate, wheels } = state.config;
    const miles = dataModels[model][wheels][climate ? 'on' : 'off'].speed[speed][temperature];
    return {
      model,
      miles
    };
  });
}
function appReducer(state = initialState, action) {
  switch (action.type) {
    case 'CHANGE_CLIMATE': {
      const newState = {
        ...state,
        config: {
          climate: !state.config.climate,
          speed: state.config.speed,
          temperature: state.config.temperature,
          wheels: state.config.wheels
        }
      };
      return updateStats(state, newState);
    }
    default:
      return state
  }
}
```

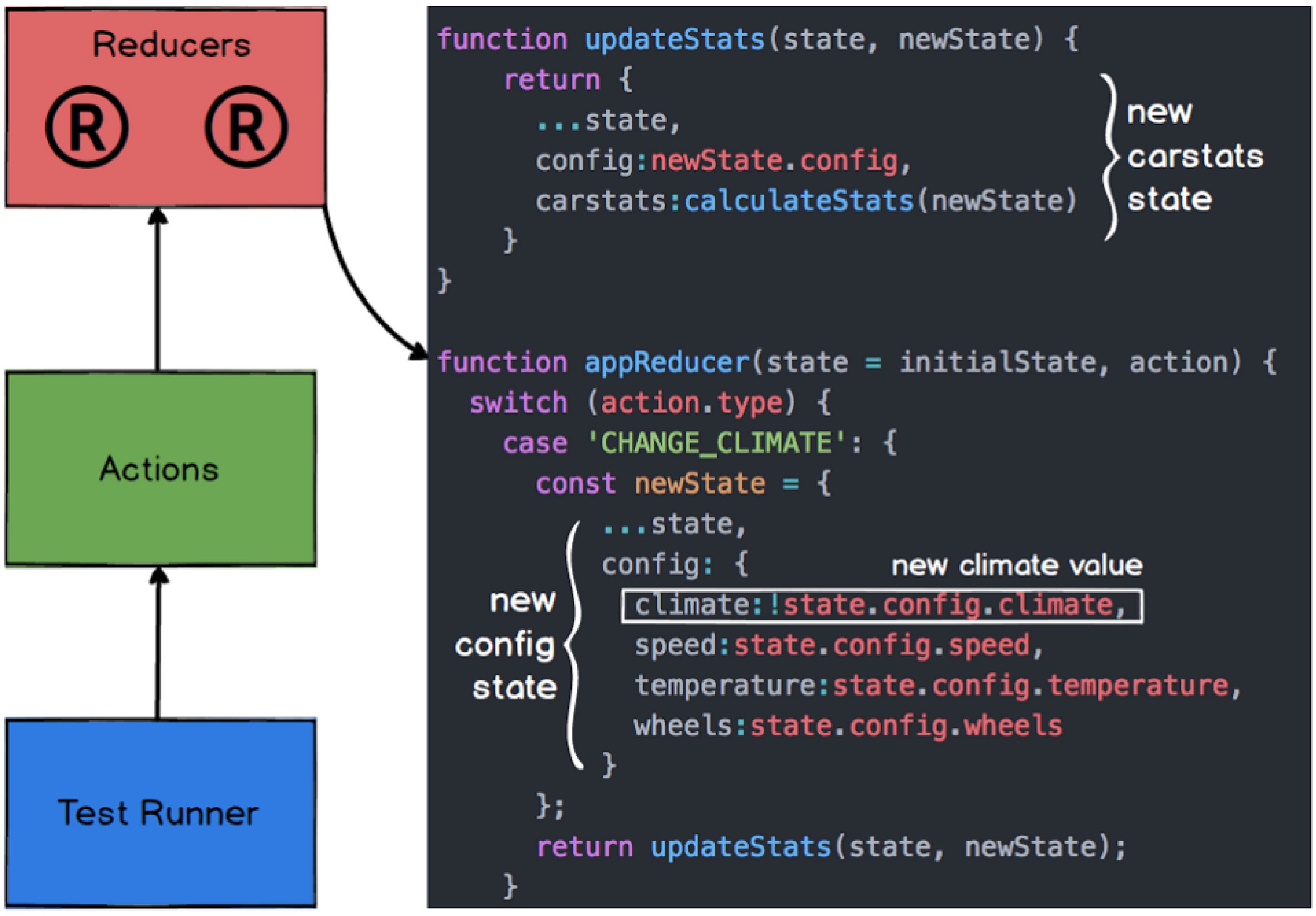
If you check the test results, you can see that the two test cases are successful.


```
 PASS  src/reducers/teslaRangeApp.spec.js
  test reducer
    ✓ should handle initial stat (1ms)
    ✓ should handle CHANGE_CLIMATE (2ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.205s, estimated 1s
Ran all test suites related to changed files.
```

What we have implemented so far is that the changes in the state that occur when the user turns the air conditioner on and off in the application through the test runner only from the **viewpoint of Action and Reducer** without Redux Store or View.





```
function updateStats(state, newState) {
    return {
      ...state,
      config:newState.config,
      carstats:calculateStats(newState)
    }
}

function appReducer(state = initialState, action) {
  switch (action.type) {
    case 'CHANGE_CLIMATE': {
      const newState = {
        ...state,
        config: {
          climate:!state.config.climate,
          speed:state.config.speed,
          temperature:state.config.temperature,
          wheels:state.config.wheels
        }
      };
      return updateStats(state, newState);
    }
```

- Check out teslaRangeApp.js as we've written it so far
- Check out teslaRangeApp.spec.js

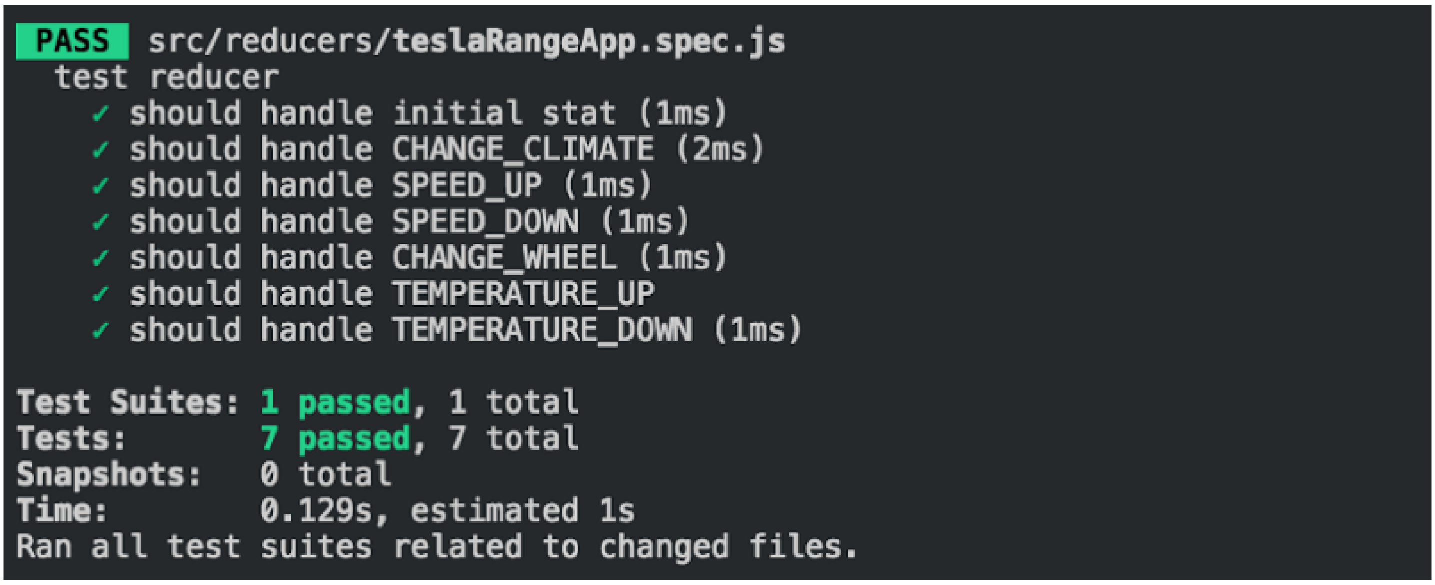
2.10.3 Create Reducer for others

If you create the rest of the test cases by referring to the above method, you finally define the **teslaRangeApp.js** file in which the reducers of all the apps are defined and the **teslaRangeApp.spec.js** to test them.

The final code can be found at:

- teslaRangeApp.js
- teslaRangeApp.spec.js

After completing the code and testing, a total of seven test cases must succeed.



```
 PASS  src/reducers/teslaRangeApp.spec.js
  test reducer
    ✓ should handle initial stat (1ms)
    ✓ should handle CHANGE_CLIMATE (2ms)
    ✓ should handle SPEED_UP (1ms)
    ✓ should handle SPEED_DOWN (1ms)
    ✓ should handle CHANGE_WHEEL (1ms)
    ✓ should handle TEMPERATURE_UP
    ✓ should handle TEMPERATURE_DOWN (1ms)

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        0.129s, estimated 1s
Ran all test suites related to changed files.
```

```
import appReducer from './teslaRangeApp';

const initialState =  {
  carstats:[
    {miles:246, model:"60"},
    {miles:250, model:"60D"},
    {miles:297, model:"75"},
    {miles:306, model:"75D"},
    {miles:336, model:"90D"},
    {miles:376, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: true,
    wheels: 19
  }
```

```
}

const climateChangeState = {

  carstats:[
    {miles:267, model:"60"},
    {miles:273, model:"60D"},
    {miles:323, model:"75"},
    {miles:334, model:"75D"},
    {miles:366, model:"90D"},
    {miles:409, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: false,
    wheels: 19
  }
}

const speedUpState = {
  carstats:[
    {miles:242, model:"60"},
    {miles:248, model:"60D"},
    {miles:292, model:"75"},
    {miles:303, model:"75D"},
    {miles:332, model:"90D"},
    {miles:371, model:"P100D"}
  ],
  config: {
    speed: 60,
    temperature: 20,
    climate: false,
    wheels: 19
  }
}

const speedDownState = {
  carstats:[
    {miles:267, model:"60"},
    {miles:273, model:"60D"},
    {miles:323, model:"75"},
    {miles:334, model:"75D"},
    {miles:366, model:"90D"},
    {miles:409, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: false,
    wheels: 19
  }
}

const wheelChangeState = {
  carstats:[
    {miles:261, model:"60"},
    {miles:268, model:"60D"},
    {miles:316, model:"75"},
    {miles:327, model:"75D"},
    {miles:359, model:"90D"},
    {miles:389, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: false,
    wheels: 21
  }
}

const temperatureUpState = {
  carstats:[
    {miles:264, model:"60"},
    {miles:272, model:"60D"},
    {miles:319, model:"75"},
    {miles:333, model:"75D"},
    {miles:367, model:"90D"},
    {miles:398, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 30,
    climate: false,
    wheels: 21
  }
}

const temperatureDownState = {
  carstats:[
    {miles:261, model:"60"},
    {miles:268, model:"60D"},
    {miles:316, model:"75"},
    {miles:327, model:"75D"},
    {miles:359, model:"90D"},
    {miles:389, model:"P100D"}
  ],
  config: {
    speed: 55,
    temperature: 20,
    climate: false,
    wheels: 21
  }
}

describe('test reducer', () => {
  it('should handle initial stat', () => {
    expect(
      appReducer(undefined, {})
    ).toEqual(initialState)
  })

  it('should handle CHANGE_CLIMATE', () => {
    expect(
      appReducer(initialState,{
        type: 'CHANGE_CLIMATE'
      })
    ).toEqual(climateChangeState)
  })

  it('should handle SPEED_UP', () => {
    expect(
      appReducer(climateChangeState,{
        type: 'SPEED_UP',
        value: 55,
        step: 5,
        maxValue: 70
      })
    ).toEqual(speedUpState)
  })
  it('should handle SPEED_DOWN', () => {
    expect(
      appReducer(speedUpState,{
        type: 'SPEED_DOWN',
        value: 60,
        step: 5,
        minValue: 45
      })
    ).toEqual(speedDownState)
  })

  it('should handle CHANGE_WHEEL', () => {
    expect(
      appReducer(speedDownState,{
        type: 'CHANGE_WHEEL',
        value: 21
      })
    ).toEqual(wheelChangeState)
  })

  it('should handle TEMPERATURE_UP', () => {
    expect(
      appReducer(wheelChangeState,{
        type: 'TEMPERATURE_UP',
        value: 20,
        step: 10,
        maxValue: 40
      })
    ).toEqual(temperatureUpState)
  })
  it('should handle TEMPERATURE_DOWN', () => {
    expect(
      appReducer(temperatureUpState,{
        type: 'TEMPERATURE_DOWN',
        value: 30,
        step: 10,
        minValue: -10
      })
    ).toEqual(temperatureDownState)
  })
})
```