redux-actions

redux-actions використовується для скорочення коду при описі action та reducer

<https://redux-actions.js.org/introduction/tutorial>

Встановлення

npm install --save redux-actions

Створення action

[Methods](/api/createaction#methods)

* [createAction](/api/createaction#createaction)
  + [createAction(type)](/api/createaction#createactiontype)
  + [createAction(type, payloadCreator)](/api/createaction#createactiontype-payloadcreator)
  + [createAction(type, payloadCreator, metaCreator)](/api/createaction#createactiontype-payloadcreator-metacreator)
* [createActions](/api/createaction#createactions)
  + [createActions(actionMap)](/api/createaction#createactionsactionmap)
  + [createActions(actionMap, ...identityActions)](/api/createaction#createactionsactionmap-identityactions)

Приклад [createAction(type)](file:///D:\api\createaction#createactiontype). Створюємо два action для збільшення/зменшення на вказану кількість

|  |  |  |
| --- | --- | --- |
| без redux-actions | з redux-actions | Загальна форма  [createAction(type)](file:///D:\api\createaction#createactiontype) |
| Створення actions, action-creators | | | |
| --------------- Створюємо типи actions --------------------  const ActionTypes={  INCREMENT : ’ INCREMENT’,  DECREMENT : 'DECREMENT'  }  --------------- Описуємо action-creators ------------------  const increment = (amount)=>{  return {  type: ActionTypes. INCREMENT,  payload: amount,  }  }  const decrement = (amount)=>{  return {  type: ActionTypes. DECREMENT,  payload: amount  }  } | import { createAction } from ‘redux-actions’  --- описуємо одразу action-creators ---  const increment = createAction('INCREMENT');  const decrement = createAction('DECREMENT'); | --- описуємо одразу action-creators ---  const action\_creator = **createAction**( назва action ); |
| Приклади створення actions | | |
| store.dispatch( **inctrement(5)** )  store.dispatch( **decrement(5)** ) | store.dispatch( **inctrement(5)** )  store.dispatch( **decrement(5)** )  Результатом виклику **inctrement(5)** буде об’єкт  {  type: 'INCREMENT',  payload: 5  } | store.dispatch( action\_creator ( параметр ) )  Параметр одразу передається у payload, тобто параметр і є значенням поля payload |
|  | Створення одразу декількох action |  |
|  | import { createActions } from 'redux-actions';  const { increment , decrement } = **createActions**({  INCREMENT:undefined,  DECREMENT:undefined }) | const { actionCreator1 , actionCreator2, … } = **createActions**({  'назва1':undefined,  'назва2':undefined, …}) |

Приклад [createAction(type, payloadCreator)](file:///D:\api\createaction#createactiontype-payloadcreator). Створюємо два action для збільшення/зменшення на вказану кількість (додаємо як поле у об’єкт payload). Вказуємо функцію, яка створює payload.

|  |  |  |
| --- | --- | --- |
| без redux-actions | з redux-actions | Загальна форма  [createAction(type, payloadCreator)](file:///D:\api\createaction#createactiontype-payloadcreator) |
| Створення actions, action-creators | | | |
| --------------- Створюємо типи actions --------------------  const ActionTypes={  INCREMENT : ’ INCREMENT’,  DECREMENT : 'DECREMENT'  }  --------------- Описуємо action-creators ------------------  const increment = (amount)=>{  return {  type: ActionTypes. DECREMENT,  payload: {  amount : amount  }  }  }  const decrement = (amount)=>{  return {  type: ActionTypes. INCREMENT,  payload: {  amount : -amount  }  }  } | import { createAction, handleAction } from ‘redux-action’  --- описуємо одразу action-creators ---  const increment = createAction(  'INCREMENT',  (amount) => ({amount : amount })  );  const increment = createAction(  'DECREMENT',  (amount) => ({amount : - amount })  ); | const action\_creator = **createAction**(  назва action ,  функція, що створює об’єкт payload  ); |
| Приклади створення actions | | |
| store.dispatch( **inctrement(5)** )  store.dispatch( **decrement(5)** ) | store.dispatch( **inctrement(5)** )  store.dispatch( **decrement(5)** )  Результатом виклику **inctrement(5)** буде об’єкт  {  type: 'INCREMENT',  payload: { amount :5}  } | store.dispatch( action\_creator ( параметр ) )  Параметр передається у функцію, яка створює об’єкт payload |
|  | Створення одразу декількох action |  |
|  | import { createActions } from 'redux-actions';  const { increment , decrement } = **createActions**(  {  'INCREMENT' : (amount)=>({amount:amount}) ,  'DECREMENT' : (amount)=>({amount: - amount})  }) | const { actionCreator1 , actionCreator2, … } = **createActions**({  'назва1': функція1, що створює payload ,  'назва2': функція2, що створює payload,  …  }) |

===============================================================================================================================================

Опис редюсерів

[Methods](/api/handleaction#methods)

* [handleAction](/api/handleaction#handleaction)
  + [handleAction(type, reducer, defaultState)](/api/handleaction#handleactiontype-reducer-defaultstate)
  + [handleAction(type, reducerMap, defaultState)](/api/handleaction#handleactiontype-reducermap-defaultstate)
* [handleActions](/api/handleaction#handleactions)
  + [handleActions(reducerMap, defaultState)](/api/handleaction#handleactionsreducermap-defaultstate)

Приклад.

|  |  |  |
| --- | --- | --- |
| без redux-actions | з redux-actions | Загальна форма  [handleAction(type, reducer, defaultState)](file:///D:\api\handleaction#handleactiontype-reducer-defaultstate) |
| Створення reducer | | | |
| const reducer=(state=defaultState,action) =>  {  switch (action.type){  case ActionTypes.INCREMENT:  return {  . . . state,  counter:state.counter + 1  }  . . . . . . .  }  } | import { handleAction } from 'redux-actions';  ------- описаний раніше action-creator -----  const **increment** = createAction('INCREMENT');  ------- відповідний reducer ----------  const reducerIncrement = handleAction(  **increment**,  (state, action) => ({  ...state,  counter: state.counter + 1  }),  defaultState  ); | редюсер = handleAction(  type,  reducer,  defaultState,  ) |
| Створення декількох reducer | | |
|  |  |  |
|  | ------------- Створені раніше actions -------------  const { increment , decrement } = **createActions**(  {  INCREMENT:undefined,  DECREMENT:undefined  }  )  ------------- reducers -----------------------  const reducer = handleActions(  {  [increment]: state => ({ ...state,  counter: state.counter + 1  }),  [decrement]: state => ({ ...state,  counter: state.counter - 1  })  },  defaultState  ); | [handleActions(reducerMap, defaultState)](file:///D:\api\handleaction#handleactionsreducermap-defaultstate)  const reducer = handleActions(  {  action1: (state, action) => (new state),  action2: (state, action) => (new state),  . . . . . . . .  },  defaultState  ); |

Написання спільного reducer для декількох action

### combineActions(...types)

This allows you to reduce multiple distinct actions with the same reducer.

-------------- Опис action creators ----------------

const { increment, decrement } = **createActions**({

INCREMENT: amount => ({ amount }),

DECREMENT: amount => ({ amount: -amount }),

})

-------------- Опис reducer ------------------------

increment(5)

{

type: …,

payload: { amount:5 }

}

|  |  |
| --- | --- |
|  | const reducer = handleActions(  {  [**combineActions (increment, decrement)**]: (state,{ payload: { amount } }) => {    return {  ...state,  counter: state.counter + amount  };  }  },  { counter: 10 }  ); |
|  | const reducer = handleActions(  {  [**increment**]: (state,{ payload: { amount } }) => {  return {  ...state,  counter: state.counter + amount  };  },  [**decrement**]: (state,{ payload: { amount } }) => {  return {  ...state,  counter: state.counter + amount  };  }  },  { counter: 10 }  ); |

===========================================================================================================================

Приклад. З просторами імен

|  |  |
| --- | --- |
|  |  |
|  | **EXAMPLE**  const actionCreators = createActions({  APP: {  COUNTER: {  INCREMENT: [amount => ({ amount }), amount => ({ key: 'value', amount })],  DECREMENT: amount => ({ amount: -amount }),  SET: undefined // given undefined, the identity function will be used  },  NOTIFY: [  (username, message) => ({ message: `${username}: ${message}` }),  (username, message) => ({ username, message })  ]  }  });  expect(actionCreators.app.counter.increment(1)).to.deep.equal({  type: 'APP/COUNTER/INCREMENT',  payload: { amount: 1 },  meta: { key: 'value', amount: 1 }  });  expect(actionCreators.app.counter.decrement(1)).to.deep.equal({  type: 'APP/COUNTER/DECREMENT',  payload: { amount: -1 }  });  expect(actionCreators.app.counter.set(100)).to.deep.equal({  type: 'APP/COUNTER/SET',  payload: 100  });  expect(  actionCreators.app.notify('yangmillstheory', 'Hello World')  ).to.deep.equal({  type: 'APP/NOTIFY',  payload: { message: 'yangmillstheory: Hello World' },  meta: { username: 'yangmillstheory', message: 'Hello World' }  });  When using this form, you can pass an object with key namespace as the last positional argument (the default is /).  **EXAMPLE**  createActions({ ... }, 'INCREMENT', { namespace: '--' })  **createActions(actionMap, ...identityActions)**  identityActions is an optional list of positional string arguments that are action type strings; these action types will use the identity payload creator.  const { actionOne, actionTwo, actionThree } = createActions(  {  // function form; payload creator defined inline  ACTION\_ONE: (key, value) => ({ [key]: value }),  // array form  ACTION\_TWO: [  first => [first], // payload  (first, second) => ({ second }) // meta  ]  // trailing action type string form; payload creator is the identity  },  'ACTION\_THREE'  );  expect(actionOne('key', 1)).to.deep.equal({  type: 'ACTION\_ONE',  payload: { key: 1 }  });  expect(actionTwo('first', 'second')).to.deep.equal({  type: 'ACTION\_TWO',  payload: ['first'],  meta: { second: 'second' }  });  expect(actionThree(3)).to.deep.equal({  type: 'ACTION\_THREE',  payload: 3  }); |
| REDUCER | handleAction(  'APP/COUNTER/INCREMENT',  (state, action) => ({  counter: state.counter + action.payload.amount  }),  defaultState  ); |