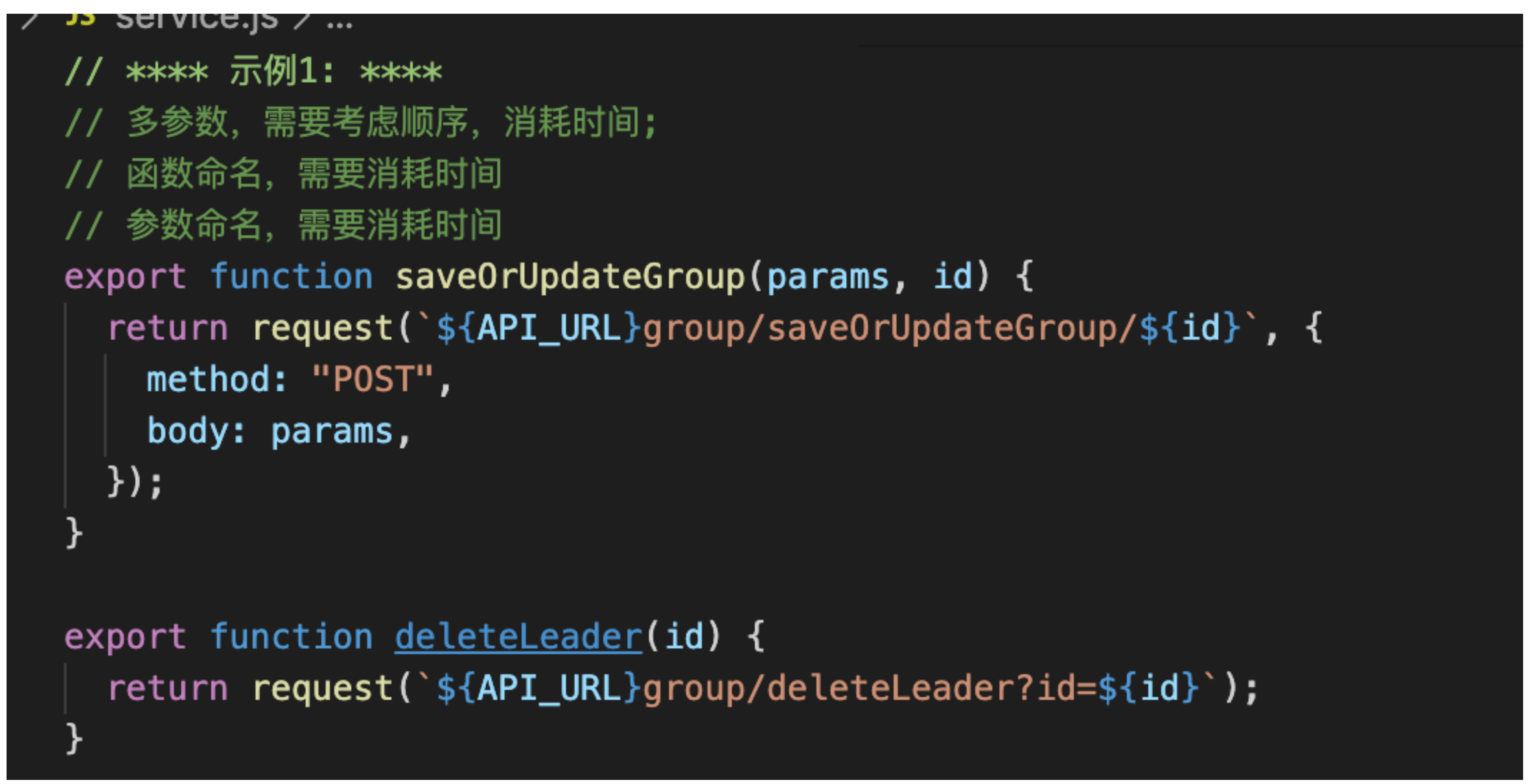
model&service优化

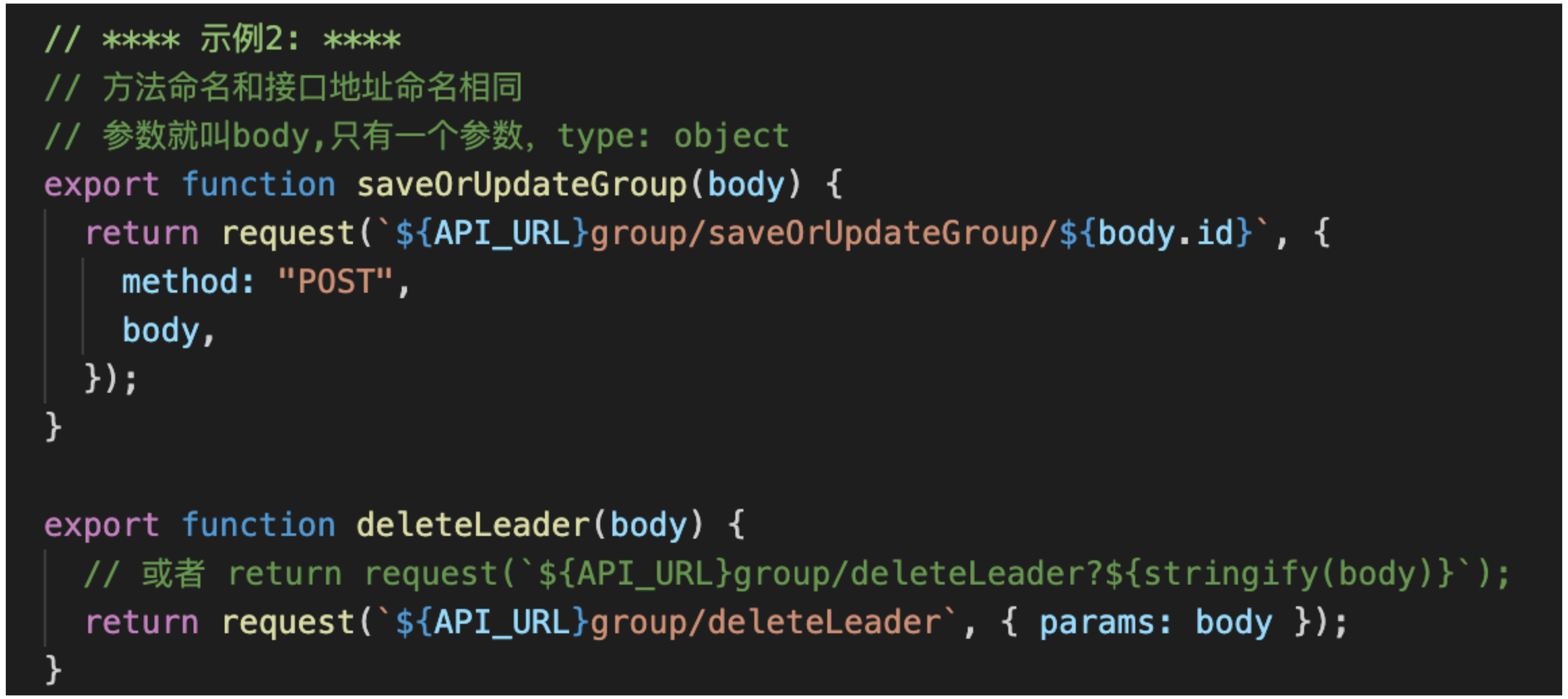
背景:

model、service经常会开发，文件会越来越大，样板代码越来越多，

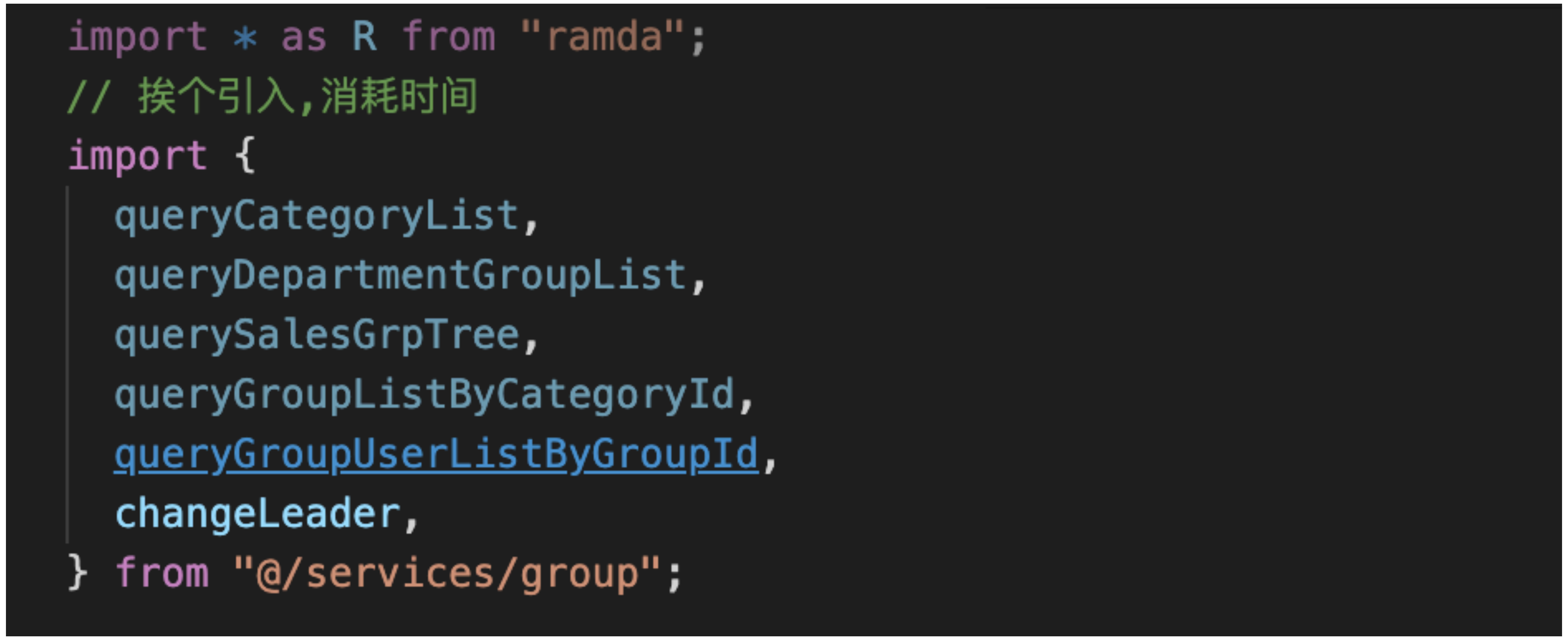
我们开发中花在这些地方的时间也比较多。



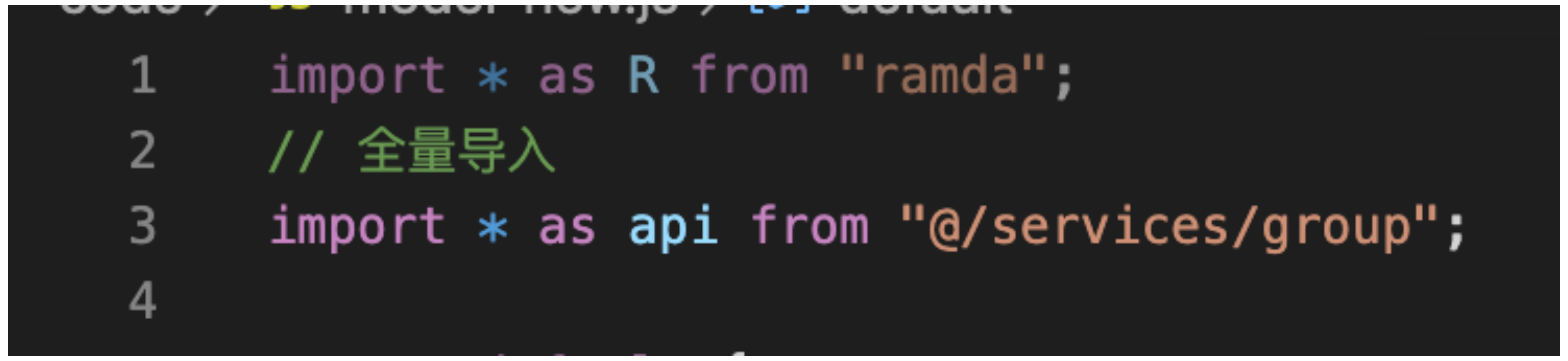
```
> JS service.js > ...
// **** 示例1: ****
// 多参数，需要考虑顺序，消耗时间;
// 函数命名，需要消耗时间
// 参数命名，需要消耗时间
export function saveOrUpdateGroup(params, id) {
  return request(`${API_URL}group/saveOrUpdateGroup/${id}`, {
    method: "POST",
    body: params,
  });
}

export function deleteLeader(id) {
  return request(`${API_URL}group/deleteLeader?id=${id}`);
}
```



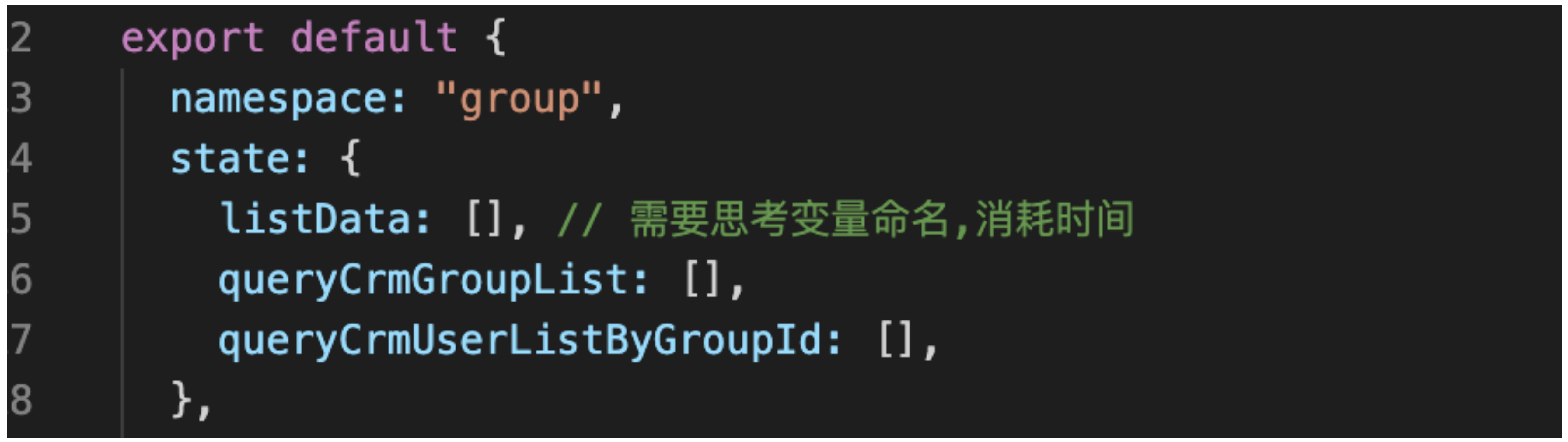
```
// **** 示例2: ****
// 方法命名和接口地址命名相同
// 参数就叫body,只有一个参数, type: object
export function saveOrUpdateGroup(body) {
  return request(`${API_URL}group/saveOrUpdateGroup/${body.id}`, {
    method: "POST",
    body,
  });
}

export function deleteLeader(body) {
  // 或者 return request(`${API_URL}group/deleteLeader?${stringify(body)}`);
  return request(`${API_URL}group/deleteLeader`, { params: body });
}
```



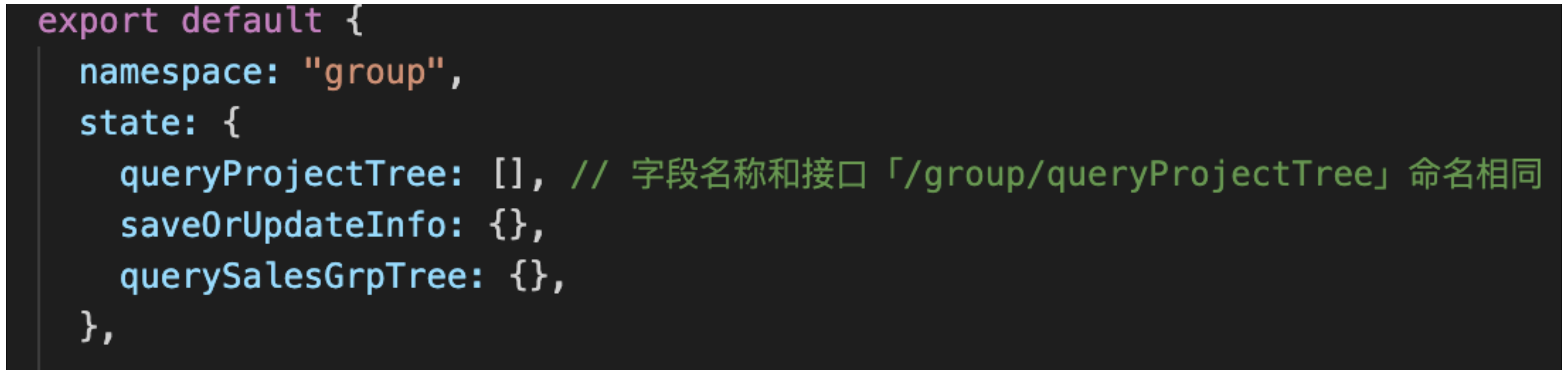
```js
import * as R from "ramda";
// 挨个引入,消耗时间
import {
  queryCategoryList,
  queryDepartmentGroupList,
  querySalesGrpTree,
  queryGroupListByCategoryId,
  queryGroupUserListByGroupId,
  changeLeader,
} from "@/services/group";
```



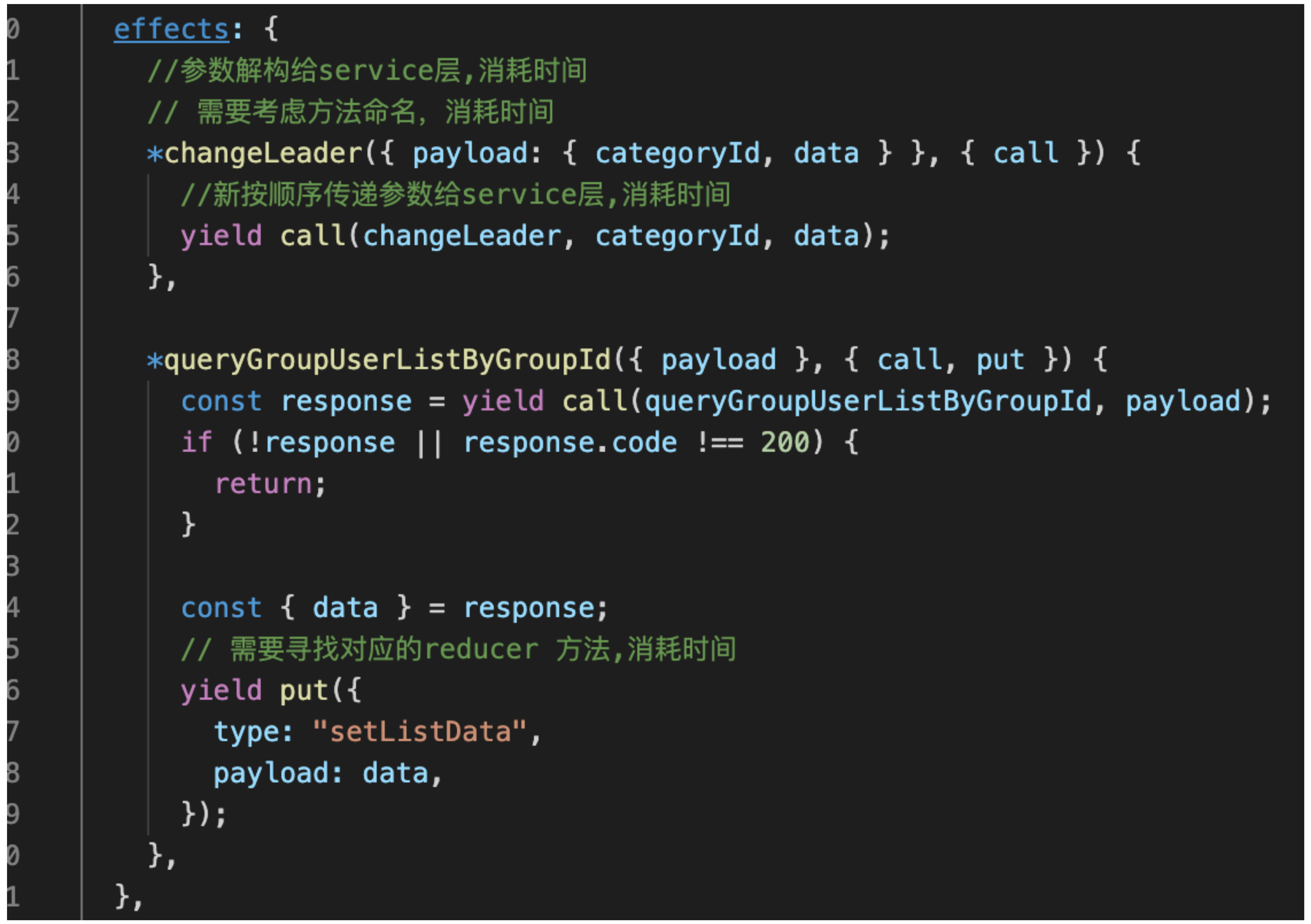
```js
import * as R from "ramda";
// 全量导入
import * as api from "@/services/group";

```



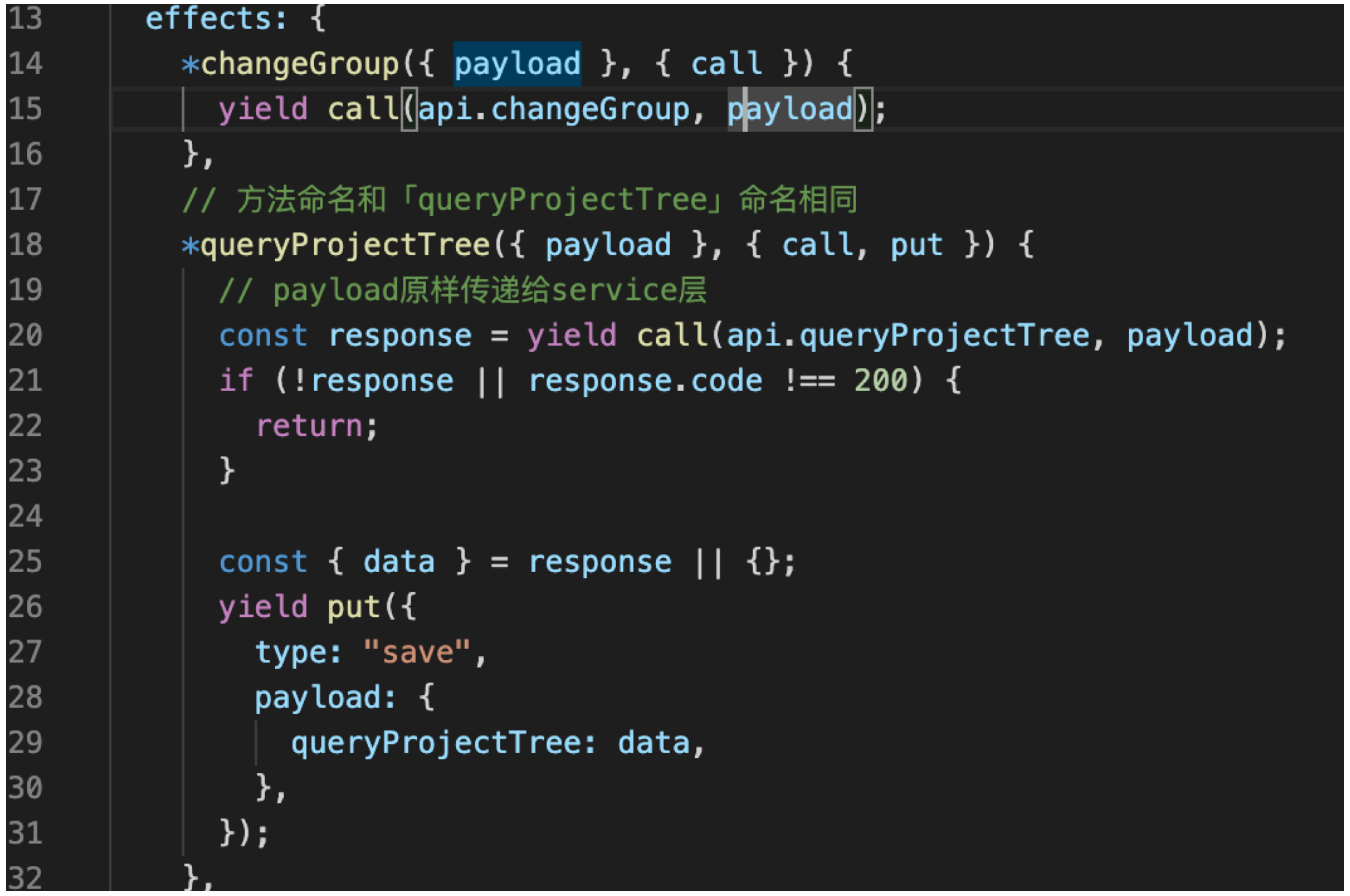
```
export default {
  namespace: "group",
  state: {
    listData: [], // 需要思考变量命名,消耗时间
    queryCrmGroupList: [],
    queryCrmUserListByGroupId: [],
  },
```



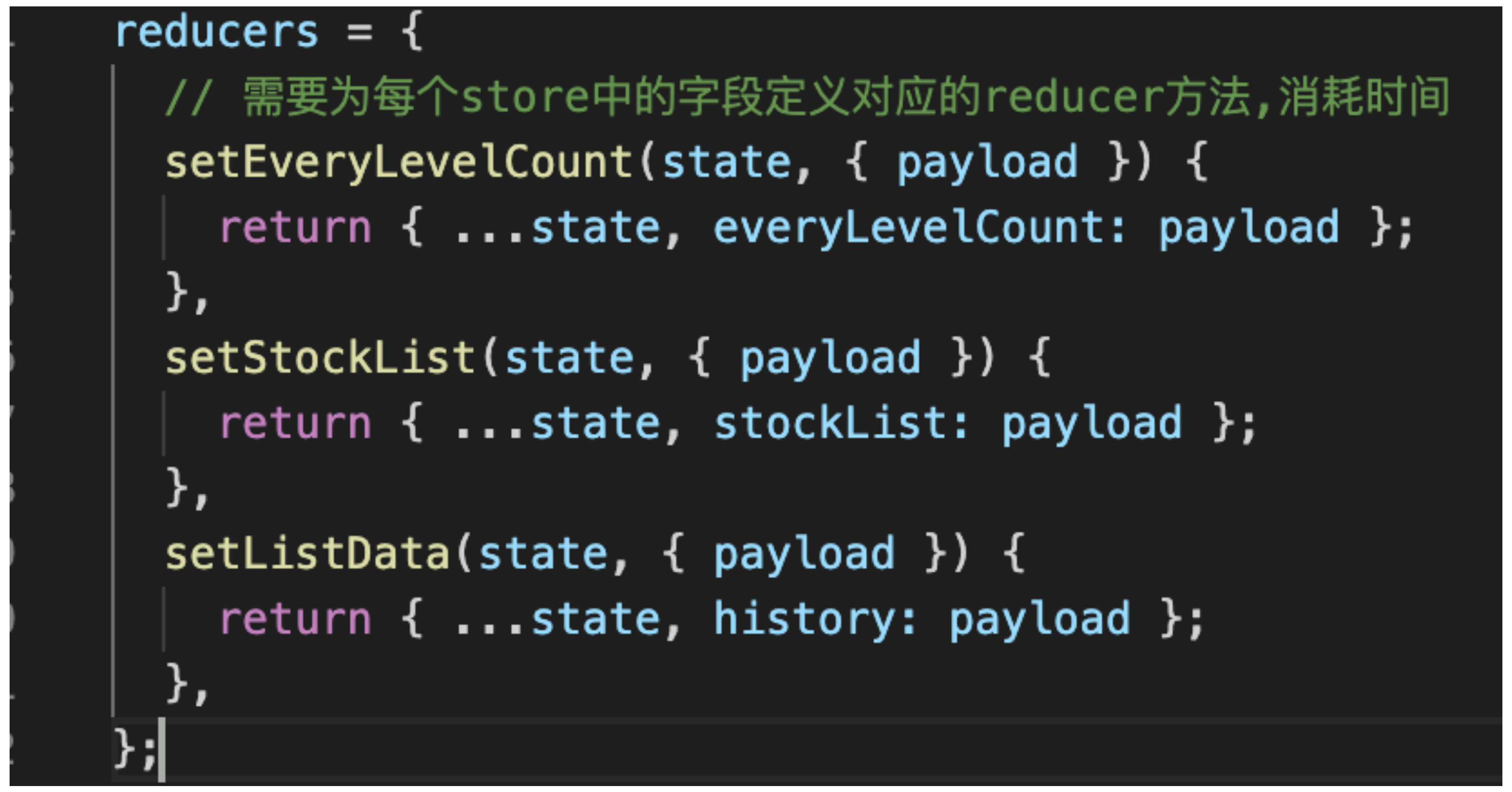
```
export default {
  namespace: "group",
  state: {
    queryProjectTree: [], // 字段名称和接口「/group/queryProjectTree」命名相同
    saveOrUpdateInfo: {},
    querySalesGrpTree: {},
  },
```



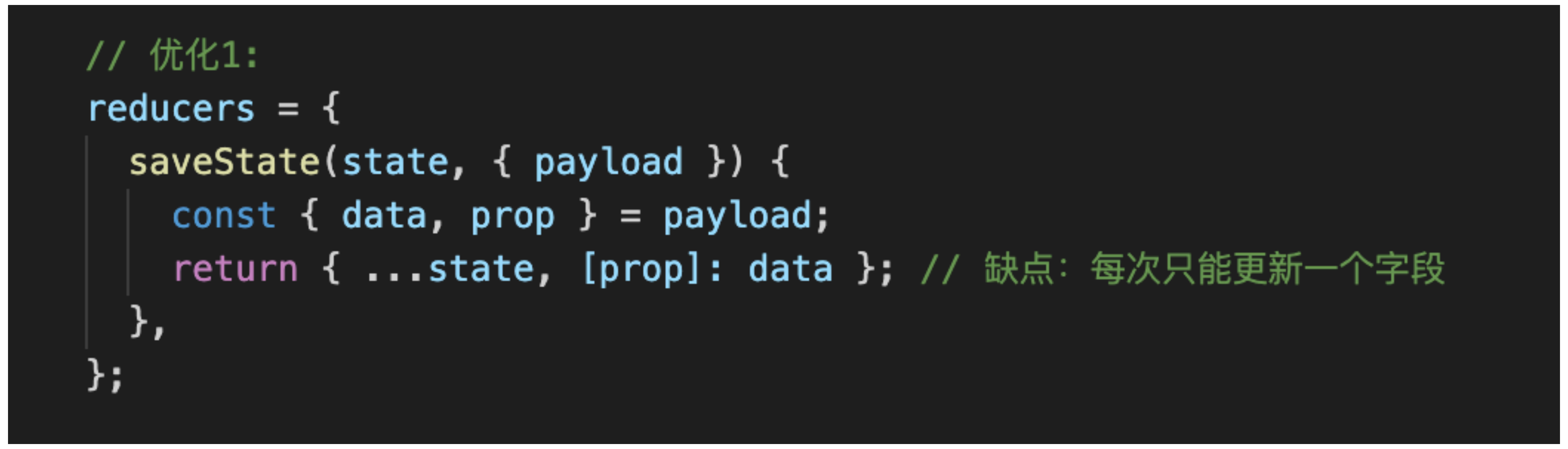
```
effects: {
  //参数解构给service层,消耗时间
  // 需要考虑方法命名, 消耗时间
  *changeLeader({ payload: { categoryId, data } }, { call }) {
    //新按顺序传递参数给service层,消耗时间
    yield call(changeLeader, categoryId, data);
  },

  *queryGroupUserListByGroupId({ payload }, { call, put }) {
    const response = yield call(queryGroupUserListByGroupId, payload);
    if (!response || response.code !== 200) {
      return;
    }

    const { data } = response;
    // 需要寻找对应的reducer 方法,消耗时间
    yield put({
      type: "setListData",
      payload: data,
    });
  },
},
```



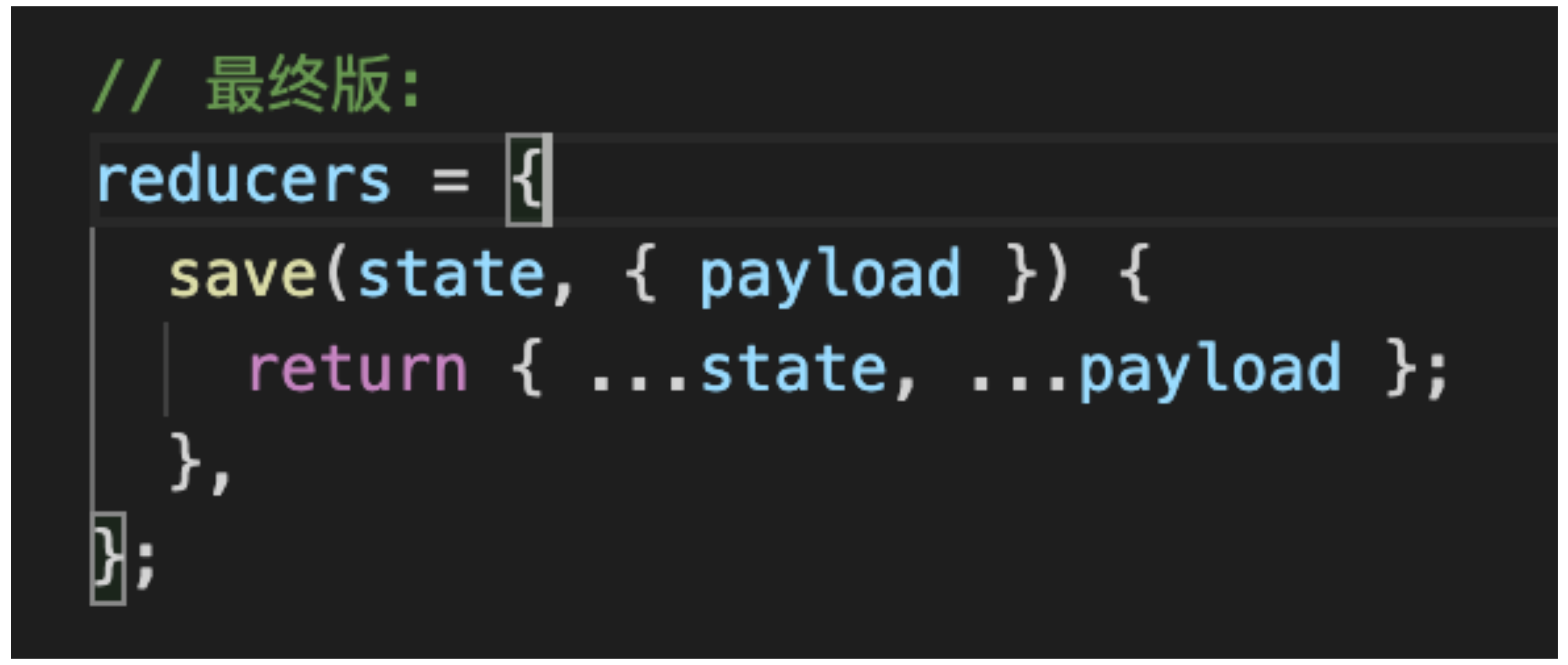
```
effects: {
  *changeGroup({ payload }, { call }) {
    yield call(api.changeGroup, payload);
  },
  // 方法命名和「queryProjectTree」命名相同
  *queryProjectTree({ payload }, { call, put }) {
    // payload原样传递给service层
    const response = yield call(api.queryProjectTree, payload);
    if (!response || response.code !== 200) {
      return;
    }

    const { data } = response || {};
    yield put({
      type: "save",
      payload: {
        queryProjectTree: data,
      },
    });
  },
```



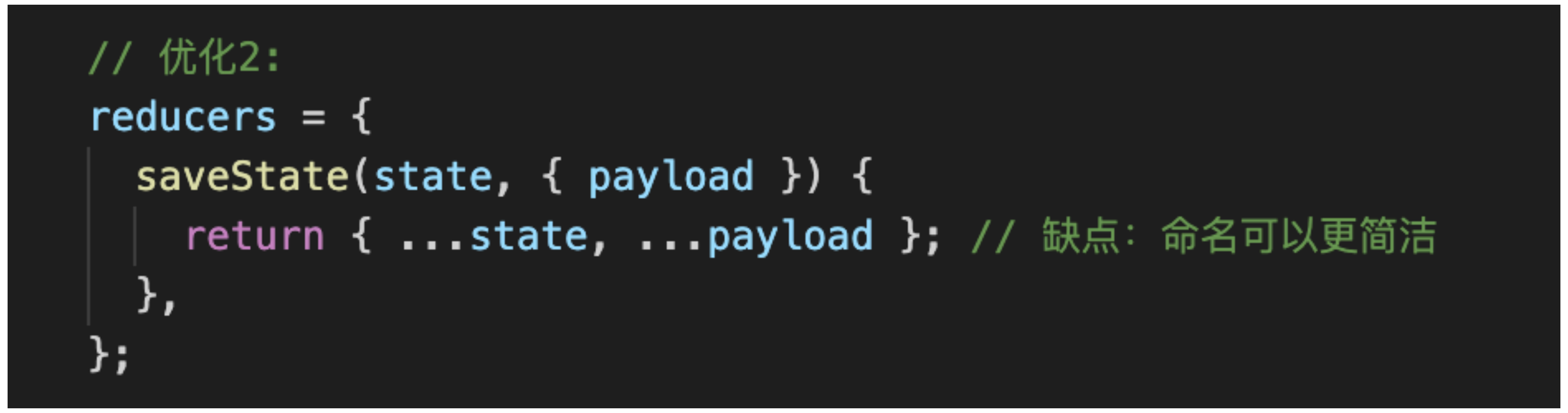
```
reducers = {
  // 需要为每个store中的字段定义对应的reducer方法,消耗时间
  setEveryLevelCount(state, { payload }) {
    return { ...state, everyLevelCount: payload };
  },
  setStockList(state, { payload }) {
    return { ...state, stockList: payload };
  },
  setListData(state, { payload }) {
    return { ...state, history: payload };
  },
};
```



```
// 优化1:
reducers = {
  saveState(state, { payload }) {
    const { data, prop } = payload;
    return { ...state, [prop]: data }; // 缺点: 每次只能更新一个字段
  },
};
```

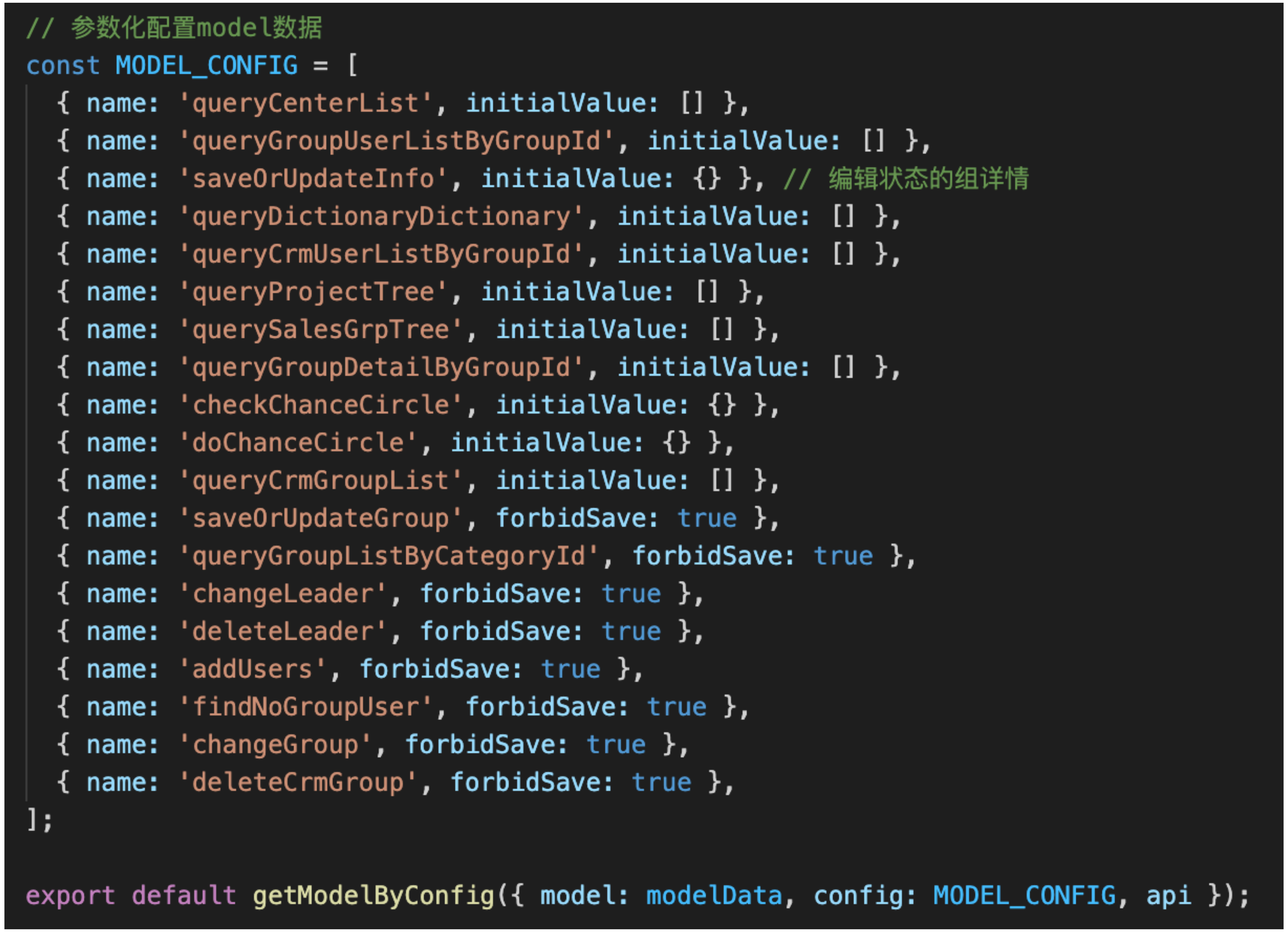


```
// 最终版:
reducers = {
  save(state, { payload }) {
    return { ...state, ...payload };
  },
};
```



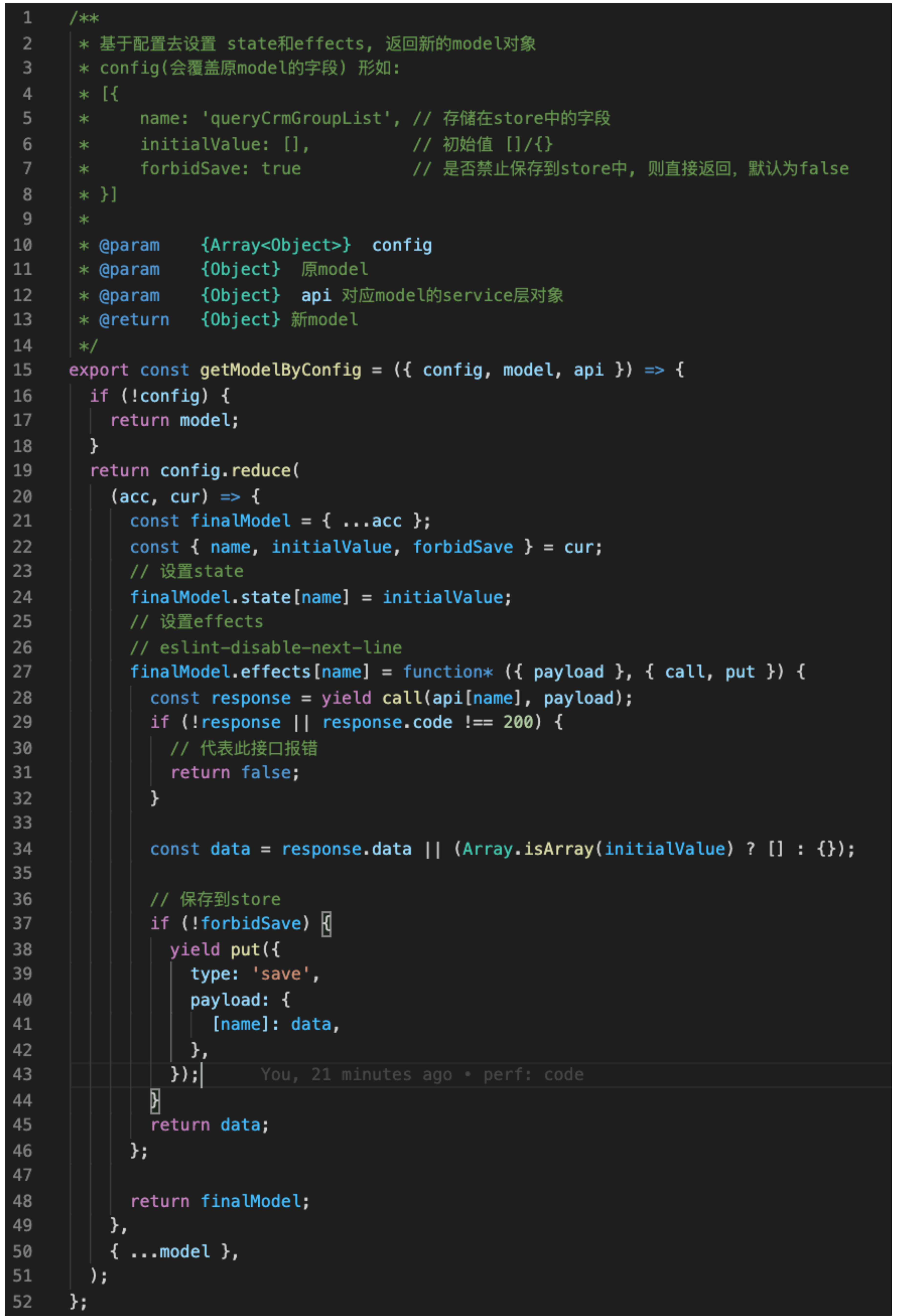
```
// 优化2:
reducers = {
  saveState(state, { payload }) {
    return { ...state, ...payload }; // 缺点: 命名可以更简洁
  },
};
```

配置式model解决样板代码问题



```javascript
// 参数化配置model数据
const MODEL_CONFIG = [
  { name: 'queryCenterList', initialValue: [] },
  { name: 'queryGroupUserListByGroupId', initialValue: [] },
  { name: 'saveOrUpdateInfo', initialValue: {} }, // 编辑状态的组详情
  { name: 'queryDictionaryDictionary', initialValue: [] },
  { name: 'queryCrmUserListByGroupId', initialValue: [] },
  { name: 'queryProjectTree', initialValue: [] },
  { name: 'querySalesGrpTree', initialValue: [] },
  { name: 'queryGroupDetailByGroupId', initialValue: [] },
  { name: 'checkChanceCircle', initialValue: {} },
  { name: 'doChanceCircle', initialValue: {} },
  { name: 'queryCrmGroupList', initialValue: [] },
  { name: 'saveOrUpdateGroup', forbidSave: true },
  { name: 'queryGroupListByCategoryId', forbidSave: true },
  { name: 'changeLeader', forbidSave: true },
  { name: 'deleteLeader', forbidSave: true },
  { name: 'addUsers', forbidSave: true },
  { name: 'findNoGroupUser', forbidSave: true },
  { name: 'changeGroup', forbidSave: true },
  { name: 'deleteCrmGroup', forbidSave: true },
];

export default getModelByConfig({ model: modelData, config: MODEL_CONFIG, api });
```

配置式model解决样板代码问题



```javascript
/**
 * 基于配置去设置 state和effects, 返回新的model对象
 * config(会覆盖原model的字段) 形如:
 * [{
 *     name: 'queryCrmGroupList', // 存储在store中的字段
 *     initialValue: [],          // 初始值 []/{}
 *     forbidSave: true           // 是否禁止保存到store中, 则直接返回, 默认为false
 * }]
 *
 * @param    {Array<Object>}  config
 * @param    {Object}  原model
 * @param    {Object}  api 对应model的service层对象
 * @return   {Object} 新model
 */
export const getModelByConfig = ({ config, model, api }) => {
  if (!config) {
    return model;
  }
  return config.reduce(
    (acc, cur) => {
      const finalModel = { ...acc };
      const { name, initialValue, forbidSave } = cur;
      // 设置state
      finalModel.state[name] = initialValue;
      // 设置effects
      // eslint-disable-next-line
      finalModel.effects[name] = function* ({ payload }, { call, put }) {
        const response = yield call(api[name], payload);
        if (!response || response.code !== 200) {
          // 代表此接口报错
          return false;
        }

        const data = response.data || (Array.isArray(initialValue) ? [] : {});

        // 保存到store
        if (!forbidSave) {
          yield put({
            type: 'save',
            payload: {
              [name]: data,
            },
          });      You, 21 minutes ago • perf: code
        }
        return data;
      };

      return finalModel;
    },
    { ...model },
  );
};
```