学习记录

1. 学习Chrome浏览器的快捷按键

   开发者工具：option＋command＋i

   javascript控制台：option＋command＋j

   或者按option＋command＋c也可以打开

   command + d 收藏至书签

2. 学习Vuex文档 使用计算值属性时的mapState辅助函数

```js
// 在单独构建的版本中辅助函数为 Vuex.mapState
import { mapState } from 'vuex'

export default {
  // ...
  computed: mapState({
    // 箭头函数可使代码更简练
    count: state => state.count,

    // 传字符串参数 'count' 等同于 `state => state.count`
    countAlias: 'count',

    // 为了能够使用 `this` 获取局部状态，必须使用常规函数
    countPlusLocalState (state) {
      return state.count + this.localCount
    }
  })
}
```

当映射的计算属性的名称与 state 的子节点名称相同时，我们也可以给 `mapState` 传一个字符串数组。

```js
computed: mapState([
  // 映射 this.count 为 store.state.count
  'count'
])
```

当有局部计算值属性时，我们需要添加对象展开运算符 ...

对象展开运算符

`mapState` 函数返回的是一个对象。我们如何将它与局部计算属性混合使用呢？通常，我们需要使用一个工具函数将多个对象合并为一个，以使我们可以将最终对象传给 `computed` 属性。但是自从有了对象展开运算符（现处于 ECMASCript 提案 stage-4 阶段），我们可以极大地简化写法：

```js
computed: {
  localComputed () { /* ... */ },
  // 使用对象展开运算符将此对象混入到外部对象中
  ...mapState({
    // ...
  })
}
```

3. 学习ES6箭头函数

由于大括号被解释为代码块，所以如果箭头函数直接返回一个对象，必须在对象外面加上括号，否则会报错。

```
// 报错
let getTempItem = id => { id: id, name: "Temp" };

// 不报错
let getTempItem = id => ({ id: id, name: "Temp" });
```

下面是 rest 参数与箭头函数结合的例子。

```
const numbers = (...nums) => nums;

numbers(1, 2, 3, 4, 5)
// [1,2,3,4,5]

const headAndTail = (head, ...tail) => [head, tail];

headAndTail(1, 2, 3, 4, 5)
// [1,[2,3,4,5]]
```

使用注意点

箭头函数有几个使用注意点。

（1）函数体内的`this`对象，就是定义时所在的对象，而不是使用时所在的对象。

（2）不可以当作构造函数，也就是说，不可以使用`new`命令，否则会抛出一个错误。

（3）不可以使用`arguments`对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替。

（4）不可以使用`yield`命令，因此箭头函数不能用作 Generator 函数。

上面四点中，第一点尤其值得注意。`this`对象的指向是可变的，但是在箭头函数中，它是固定的。

4. generator函数

函数使用function* 来定义 同时可以有多个返回值使用关键字yield 也可以使用return　function()并没有调用该方法 只是声明了一个generator对象 需要手动next()调用　并返回 {value: , result: }结构　result为boolean类型 当为true时调用结束　　如果yield 在return 之后调用 无效

```
"use strict";
function* fibonacci() {
    yield 1;
    yield 2;
}

var it = fibonacci();
console.log(it);          // "Generator {  }"
console.log(it.next());   // 1
console.log(it.next());   // 2
console.log(it.next()); //undefined
```

已解决

1. 学习商城源码，更改搜索历史书序问题。


```
import { localCache as LCC } from './index';

/**
 * [刷新本地搜索历史存储]
 * @param {string}
 */
export function refreshLocalHistory(val) {
  if (val.trim().length > 0) {
    let array = LCC.getLocalLimited('search-history');
    if (!array) {
      array = [];
    }
    let valIndex = array.findIndex((value) => value == val);
    if (valIndex > -1) {
      array.splice(valIndex, 1);
    }
    // array.unshift(val);
    array.push(val);
    // 本次存储的数量限制在15个，思考这里做15的处理和在 module 处理有何异同
    LCC.setLocalLimited('search-history', {
      data: array.slice(0, 15)
    });
  }
}
```

2. 箭头函数两个箭头是什么意思？ a(param1)(param2)... a(偏函数)

```
getters: {
  // ...
  getTodoById: (state) => (id) => {
    return state.todos.find(todo => todo.id === id)
  }
}
```

```
store.getters.getTodoById(2) // -> { id: 2, text: '...', done: false }
```

嵌套箭头函数 官方案例

back to top

edit

嵌套的箭头函数

箭头函数内部，还可以再使用箭头函数。下面是一个 ES5 语法的多重嵌套函数。

```
function insert(value) {
  return {into: function (array) {
    return {after: function (afterValue) {
      array.splice(array.indexOf(afterValue) + 1, 0, value);
      return array;
    }};
  }};
}

insert(2).into([1, 3]).after(1); //[1, 2, 3]
```

上面这个函数，可以使用箭头函数改写。

```
let insert = (value) => ({into: (array) => ({after: (afterValue) => {
  array.splice(array.indexOf(afterValue) + 1, 0, value);
  return array;
}})});

insert(2).into([1, 3]).after(1); //[1, 2, 3]
```

未解决

1.