

通用方法文档

产品版本：V1.0

文档版本：202401

北京有生博大软件股份有限公司

目录

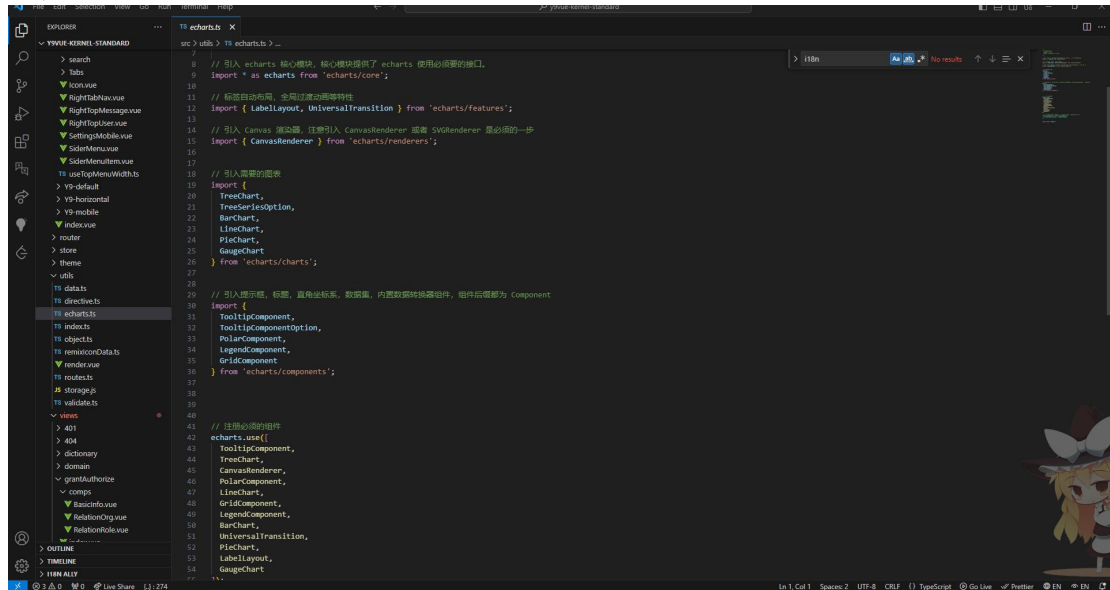
1. echart 图	3
2. 方法封装	4
3. 缓存配置	6
4. 自定义 v-指令	7
5. 路由数据的处理	8

在项目文件中，你可以看到很多地方都是用了 `utils` 目录下的工具文件。
`src/utils` 目录下是封装的一些方法和一些数据整合。

1. echarts 图

文件位置：`src/utils/echarts.ts`

如果项目中需要使用到 echarts 图，未防止在每个文件中重复引入，遂在 `src/utils` 目录下创建了 `echarts.ts` 文件，用于引入 echarts 所需的图表、渲染器等，也有便于统一管理。



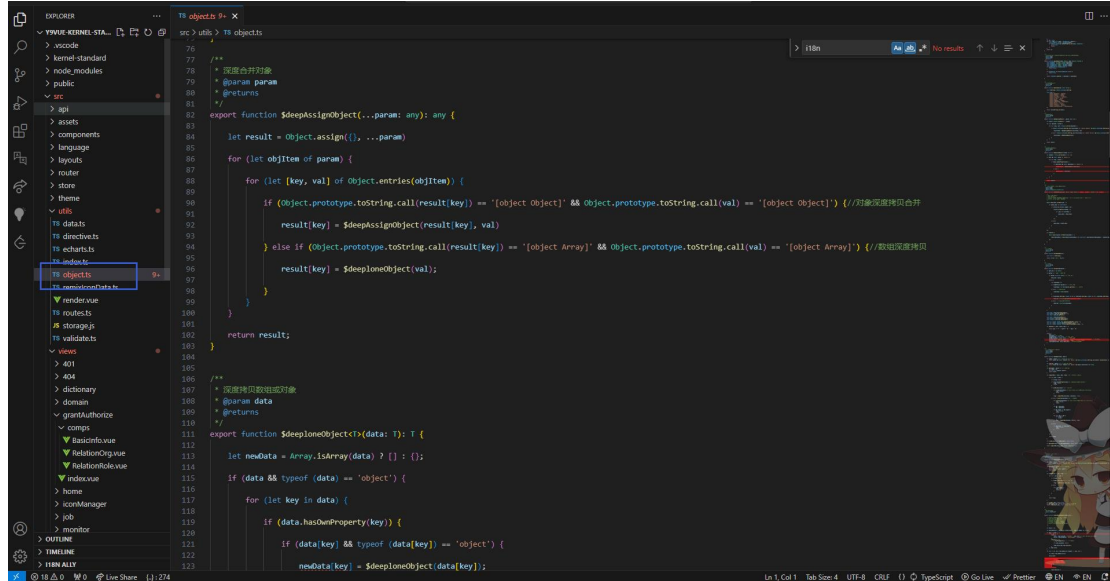
```
1 // 引入 echarts 核心模块，核心模块提供了 echarts 使用必须的接口。
2 import * as echarts from 'echarts/core';
3
4 // 按需引入图表，全局注册功能等特性
5 // 新增自动布局，全屏过渡动画等特性
6 import { LabelLayout, UniversalTransition } from 'echarts/features';
7
8 // 引入 Canvas 渲染器，注册引入 CanvasRenderer 或者 SVGRenderer 是必须的一步
9 import { CanvasRenderer } from 'echarts/renderers';
10
11 // 引入需要的图表
12 import {
13   TreeChart,
14   TreemapOption,
15   BarChart,
16   LineChart,
17   PieChart,
18   GaugeChart,
19 } from 'echarts/charts';
20
21 // 引入轴、标题、坐标、数据、内部数据转换器组件，组件后缀都为 Component
22 import {
23   TooltipComponent,
24   TooltipComponentOption,
25   PolarComponent,
26   LegendComponent,
27   GridComponent,
28 } from 'echarts/components';
29
30 // 注册必须的组件
31 echarts.use([
32   TooltipComponent,
33   TreeChart,
34   CanvasRenderer,
35   PolarComponent,
36   LineChart,
37   GridComponent,
38   LegendComponent,
39   UniversalTransition,
40   PieChart,
41   LabelLayout,
42   GaugeChart
43 ]);
```

2. 方法封装

2.1. 对象、数组

文件位置: **src/utlis/object.ts**

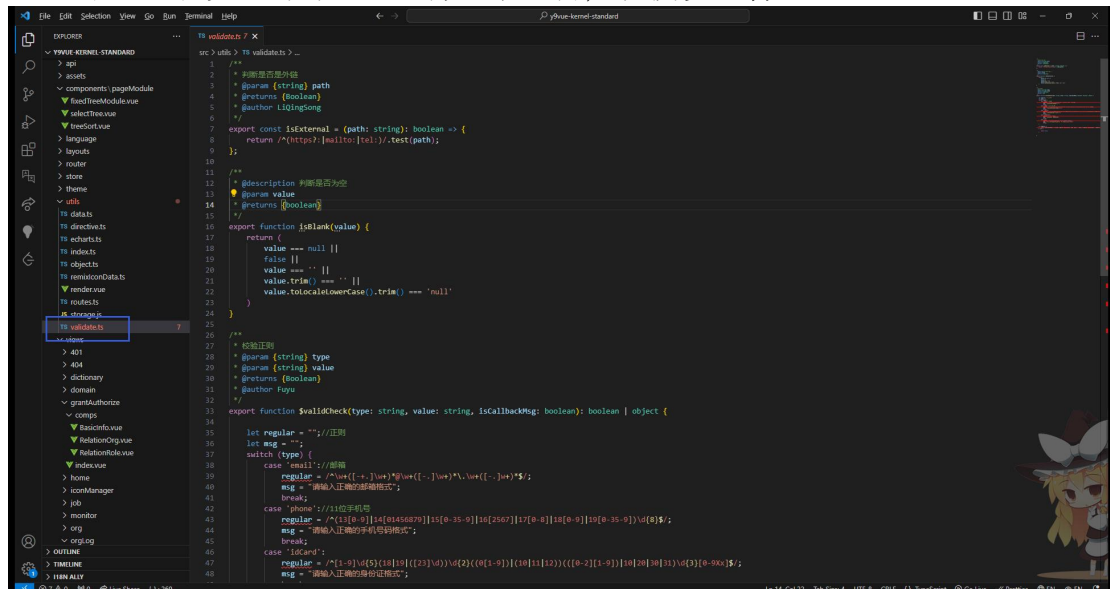
文件内有很多封装的一些方法, 比如比较两个对象是否相等、深度合并对象、深度拷贝数组或对象等。



2.2. 检验正则

文件位置: **src/utlis/validate.ts**

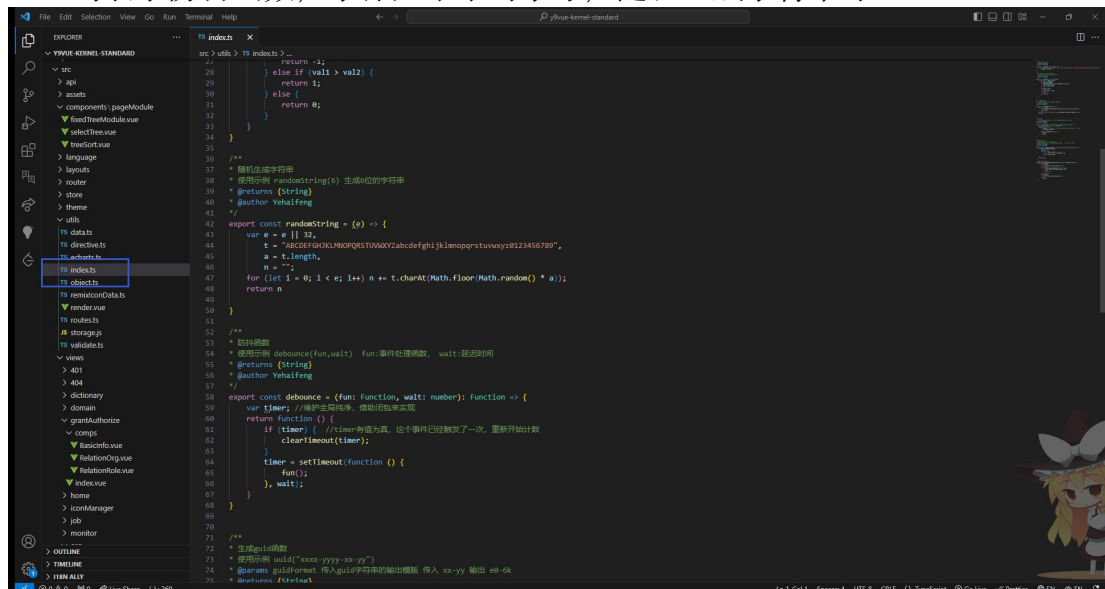
对全文中涉及到的正则进行一个整合, 以便统一管理。



2.3. 字体大小设置、防抖等

文件位置: **src/utils/index.ts**

封装了防抖函数, 字体大中小的字号, 随机生成字符串等。



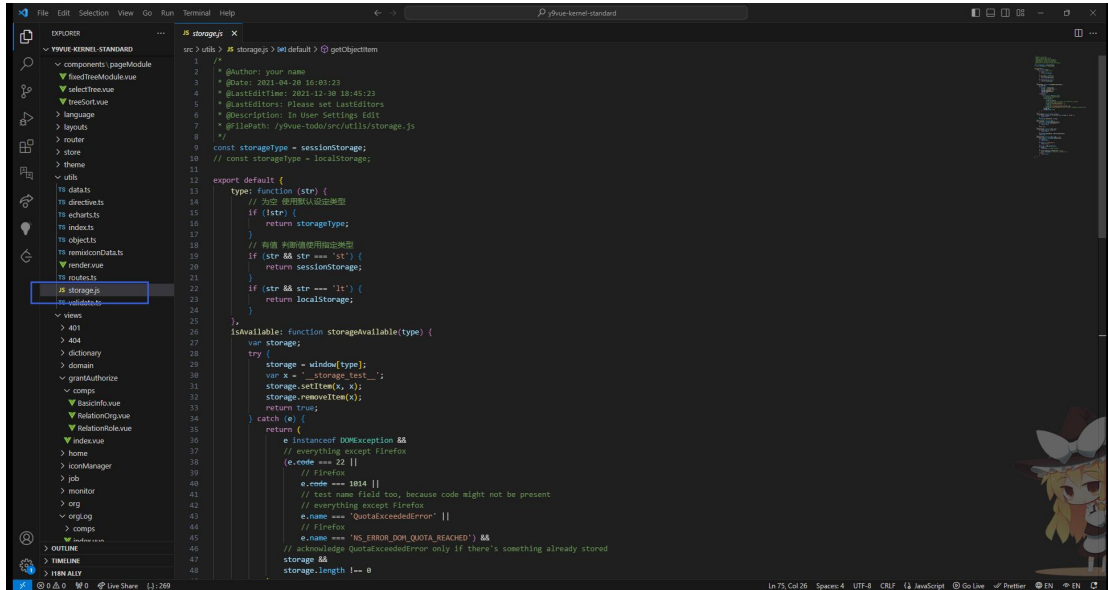
```
24 src > utils > ts index.ts
25
26     return -1;
27   } else if (val1 > val2) {
28     return 1;
29   } else {
30     return 0;
31   }
32 }
33
34 /**
35  * 随机生成字符串
36  * 使用示例 randomString() 生成6位的字符串
37  * @param {String}
38  * @author Vohalifeng
39 */
40 export const randomString = (p) => {
41   var e = [] | 32,
42       t = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",
43       n = t.length,
44       m = "";
45   for (let i = 0; i < e; i++) m += t.charAt(Math.floor(Math.random() * n));
46   return m
47 }
48
49 /**
50  * 防抖函数
51  * 使用示例 debounce(func,wait) func:事件处理函数, wait:延迟时间
52  * @param {String}
53  * @author Vohalifeng
54 */
55 export const debounce = (fun: Function, wait: number): Function => {
56   var timer; //防止全局污染, 借助闭包来实现
57   return function () {
58     if (timer) { //timer不为null, 这个事件已经被触发了一次, 重新开始计数
59       clearTimeout(timer);
60     }
61     timer = setTimeout(function () {
62       fun();
63     }, wait);
64   };
65 }
66
67 /**
68  * 生成fontSize函数
69  * 使用示例 size("xxxx-yyyy-zz-zz")
70  * @param {String} 传入fontSize字符串的输出模版 传入 xx-yy 输出 ee-ok
71  * @author: ZChalmei
72 */
```

3. 缓存配置

文件位置: **src/utils/storage.js**

本项目默认采用 `sessionStorage` 的存储方式, 如果更改, 可以直接在 `storage` 文件中将 `storageType` 改为 `localStorage`, 这样项目中有引入该文件进行的存储操作都会变成 `storageType` 设置的类型进行数据存储。

如果想单个更改, 可以单独使用 `sessionStorage` 或 `localStorage` 去进行操作。

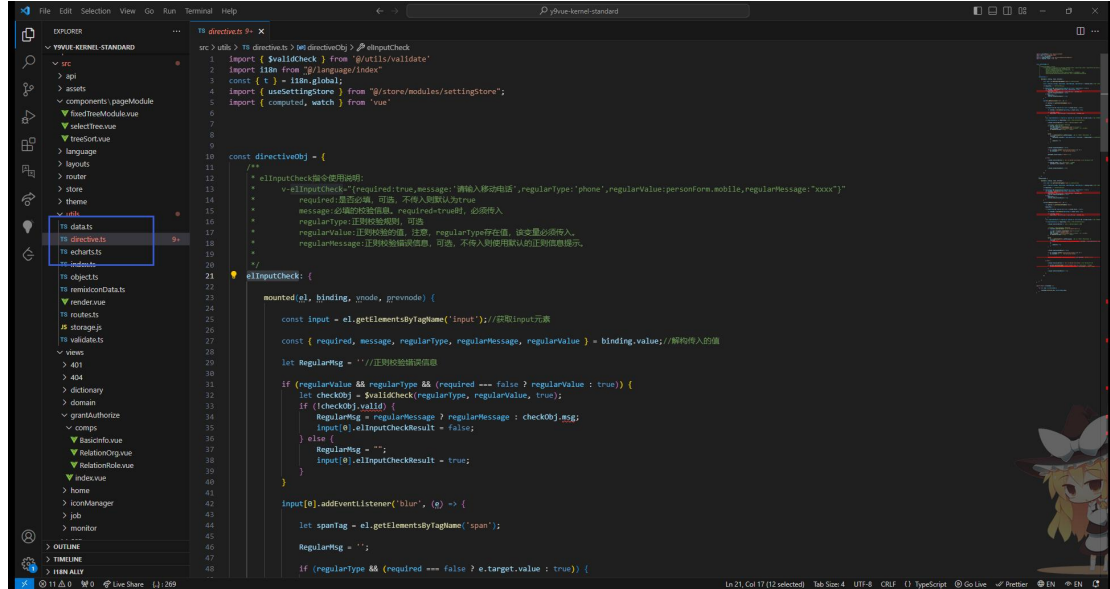


```
1  *
2  * @Author: your name
3  * @Date: 2021-04-20 16:03:23
4  * @LastEditTime: 2021-12-30 18:46:19
5  * @Description: please set lastEditors
6  * @Description: In User Settings Edit
7  * @FilePath: /yvue-todo/src/utils/storage.js
8  */
9  const storageType = sessionStorage;
10 // const storageType = localStorage;
11
12 export default {
13   type: function (str) {
14     // 空字符串返回指定类型
15     if (!str) {
16       return storageType;
17     }
18     // 有值 判断值使用指定类型
19     if (str && str === 'st') {
20       return sessionStorage;
21     }
22     if (str && str === 'lt') {
23       return localStorage;
24     }
25   },
26   isAvailable: function (storageAvailable(type) {
27     var storage;
28     try {
29       storage = window[type];
30       var x = '- storage test -';
31       storage.setItem(x, x);
32       storage.removeItem(x);
33       return true;
34     } catch (e) {
35       return (
36         e instanceof DOMException &&
37         // everything except Firefox
38         (e.code === 22 ||
39           // Firefox
40           e.code === 1014 ||
41           // test name field too, because code might not be present
42           // everything except Firefox
43           e.name === 'QuotaExceededError' ||
44           // Firefox
45           e.name === 'NS_ERROR_DOM_QUOTA_REACHED') &&
46         // acknowledge QuotaExceededError only if there's something already stored
47         storage &&
48         storage.length !== 0
49       );
50     }
51   });
52 }
```

4. 自定义 v-指令

文件位置: **src/Utils/directive.ts**

自定义表单验证, 与表单错误处理的 v-指令。后续需要添加新的指令可以直接添加在对象内。

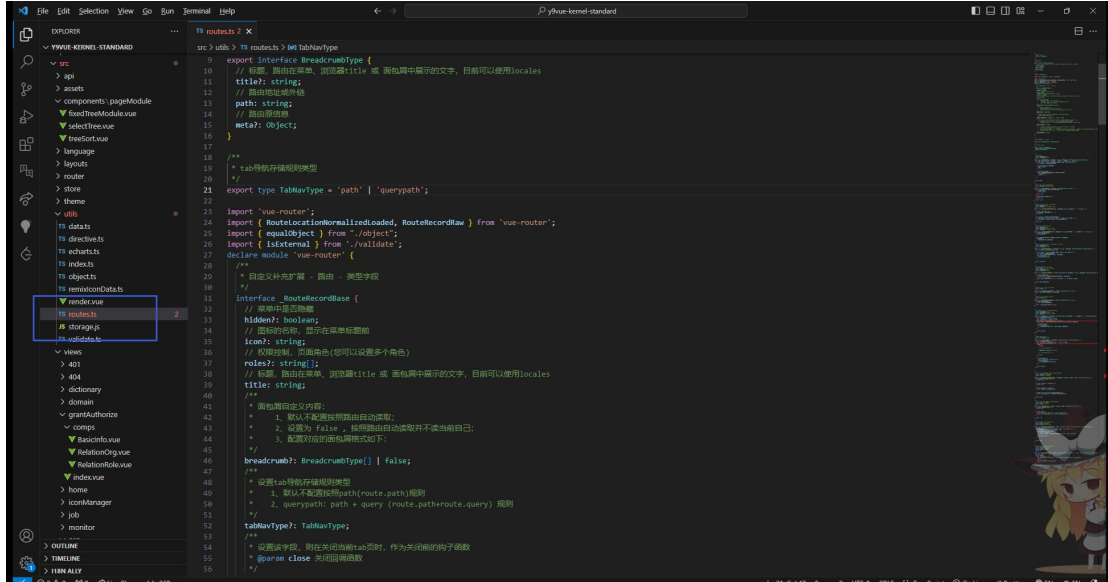


```
src > utils > ts directive.ts @ directive.ts | elInputCheck
1 import { $validCheck } from '@utils/validate'
2 import { $t } from '@utils/language/index'
3 const { t } = $t.global
4 import { useSettingStore } from '@store/modules/settingStore';
5 import { computed, watch } from 'vue'
6
7
8
9
10 const directiveObj = {
11
12   /**
13    * elInputCheck指令使用说明:
14    * v-elInputCheck="{required:true,message:'请输入手机号码',regularType:'phone',regularValue:personform_mobile,regularMessage:'xxxx'}"
15    * required:是否必填。 必填, 不得为空或false
16    * message:必填校验提示信息, required=true时, 必须传入
17    * regularType:正则校验规则, 可选
18    * regularValue:正则校验的值, 注意, regularType存在值, 该值是必须传入
19    * regularMessage:正则校验失败消息, 可选, 不得为空或undefined正则校验提示。
20    */
21 }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

5. 路由数据的处理

文件位置: **src/utils/routes.ts**

文件中包含面包屑类型、路由类型字段, 以及一些路由相关的逻辑处理等。



```
ts routes.ts X
src > utils > ts routes.ts > ts TabNavType
9 export interface BreadcrumbType {
10 // 标题: 路由名称, 浏览器title 页 面包屑中显示的文字, 目前可以使用locales
11 title?: string;
12 // 路由地址外链
13 path: string;
14 // 路由类型
15 meta?: Object;
16 }
17
18 /**
19 * tab导航存储视图的类型
20 */
21 export type TabNavType = 'path' | 'querypath';
22
23 import 'vue-router';
24 import { RouteLocationNormalizedLoaded, RouteRecordRaw } from 'vue-router';
25 import { equalObject } from './object';
26 import { isInternal } from './validate';
27 declare module 'vue-router' {
28 /**
29 * 自定义补充扩展 - 路由 - 类型字段
30 */
31 interface RouteRecordBase {
32 // 路由是否隐藏
33 hidden?: boolean;
34 // 路由的名称, 显示在菜单标题前
35 icon?: string;
36 // 权限控制, 页面角色(你可以设置多个角色)
37 roles?: string[];
38 /**
39 * 面包屑自定义内容:
40 * 1. 默认不配置按照路由自动获取;
41 * 2. 设置为 false, 按照路由自动获取并不受当前自己;
42 * 3. 配置对应的面包屑格式如下:
43 */
44 breadcrumb?: BreadcrumbType | false;
45 /**
46 * 设置tab存储视图的类型
47 * 1. 默认不配置按照path(route.path)规则
48 * 2. querypath: path + query (route.path+route.query) 规则
49 */
50 TabNavType?: TabNavType;
51 /**
52 * 设置关闭字段, 则在关闭当前tab页时, 作为关闭的钩子函数
53 * @param close 关闭回调函数
54 */
55 }
```