你还不会 vue 表格跨页多选?

在我们日常项目开发中,经常会有 `表格跨页多选` 的需求,接下来让我们用 `el-table` 示例一步步来实现这个需求。

动手开发

在线体验

codesandbox.io/s/priceless…

常规版本

本部分只写了一些重点代码,心急的彦祖可以直接看 `性能进阶版`

1. 首先我们需要初始化一个选中的数组 `checkedRows`

```
this.checkedRows = []
```

1. 在触发选中的时候,我们就需要把当前行数据 `push` 到 `checkedRows` ,否则就需要剔除对应行

```
<el-table ref="multipleTable" @select="handleSelectChange">
handleSelectChange (val, row) {
  const checkedIndex = this.checkedRows.findIndex(_row => _row.id === row.id)
  if (checkedIndex > -1) {
    // 选中剔除
    this.checkedRows.splice(checkedIndex, 1)
  } else {
    // 未选中压入
    this.checkedRows.push(row)
  }
}
```

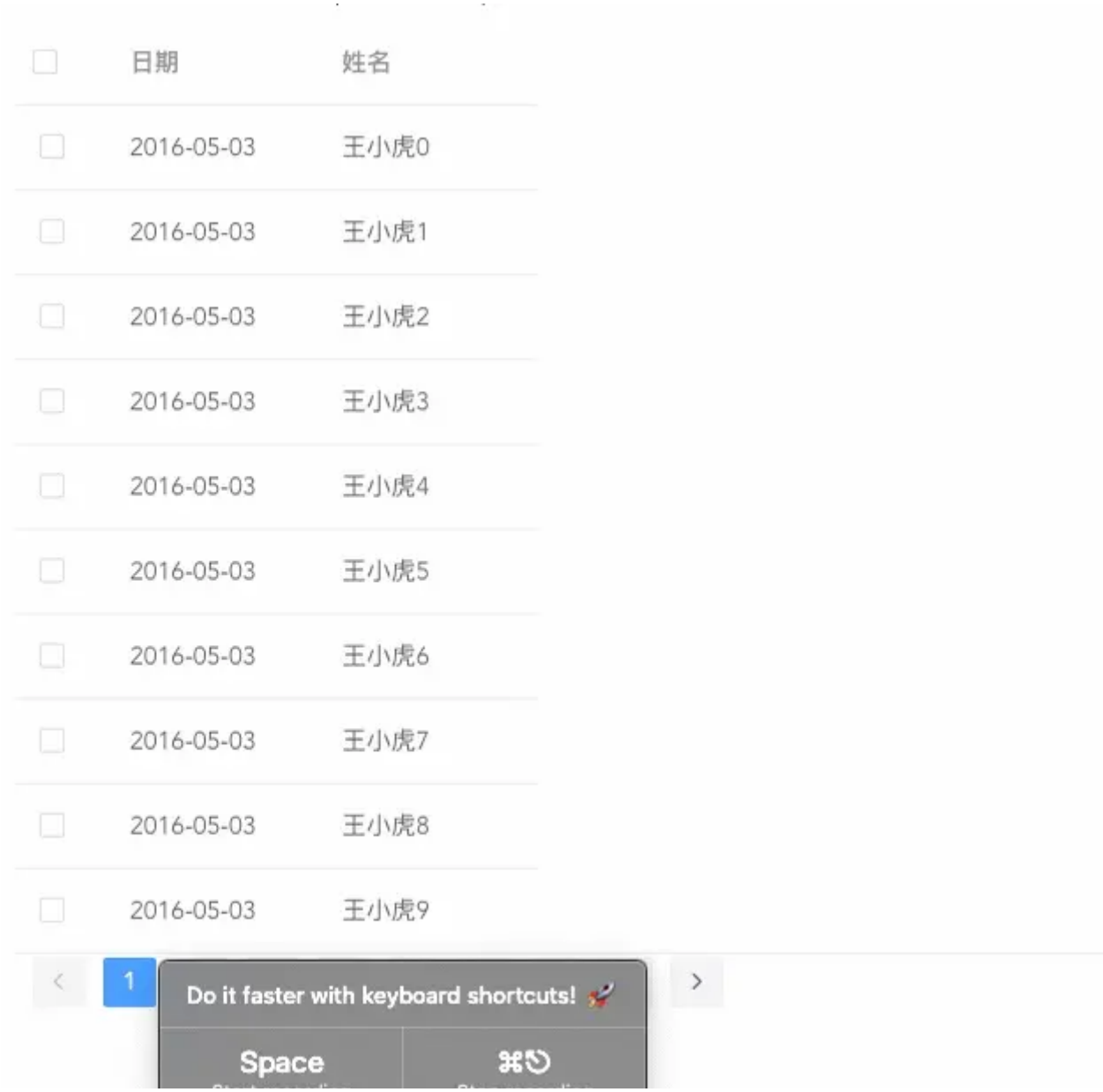
1. 实现换页的时候的回显逻辑

```
this.data.forEach(row=>{
  const checkedIndex = this.checkedRows.findIndex(_row => _row.id === row.id)
  if(checkedIndex>-1) this.$refs.multipleTable.toggleRowSelection(row,true)
```

```
4 })
```

效果预览

让我们看下此时的效果



完整代码

```
1 <template>
2   <div>
3     <el-table
4       ref="multipleTable"
5       :data="tableData"
6       tooltip-effect="dark"
7       style="width: 100%"
8       @select="handleSelectChange"
```

```
      @select-all="handleSelectAllChange"
    >
      <el-table-column
        type="selection"
        width="55"
      />
      <el-table-column
        label="日期"
        width="120"
        prop="date"
      />
      <el-table-column
        prop="name"
        label="姓名"
        width="120"
      />
    </el-table>
    <el-pagination
      background
      :current-page.sync="currentPage"
      layout="prev, pager, next"
      :total="1000"
      @current-change="currentChange"
    />
  </div>
</template>

<script>
export default {
  data () {
    return {
      currentPage: 1,
      checkedRows: [],
      pageSize: 10,
      totalData: Array.from({ length: 1000 }, (_, index) => {
        return {
          date: '2016-05-03',
          id: index,
          name: '王小虎' + index
        }
      })
    }
  },
  computed: {
    tableData () {
      const { currentPage, totalData, pageSize } = this
```

```
55        return totalData.slice((currentPage - 1) * pageSize, currentPage *
   pageSize)
56      }
57    },
58    methods: {
59      currentChange (page) {
60        this.currentPage = page
61        this.tableData.forEach(row => {
62          const checkedIndex = this.checkedRows.findIndex(_row => _row.id ===
   row.id)
63          if (checkedIndex > -1)
   this.$refs.multipleTable.toggleRowSelection(row, true)
64        })
65      },
66      handleSelectChange (val, row) {
67        const checkedIndex = this.checkedRows.findIndex(_row => _row.id ===
   row.id)
68        if (checkedIndex > -1) {
69          this.checkedRows.splice(checkedIndex, 1)
70        } else {
71          this.checkedRows.push(row)
72        }
73      },
74      handleSelectAllChange (val) {
75        this.tableData.forEach(row => {
76          this.handleSelectChange(null, row)
77        })
78      }
79    }
80  }
81  </script>
```

性能进阶版

性能缺陷分析

优秀的彦祖们,应该发现以上代码的性能缺陷了

1. `handleSelectChange` 需要执行一个 `O(n)` 复杂度的循环

2. `currentChange` 的回显逻辑内部, 有一个 `O(n^2)` 复杂度的循环

想象一下 如果场景中勾选的行数达到了 `10000` 行, 每页显示 `100` 条

那么我们每次点击换页 最坏情况就要执行 `10000 * 100` 次循环,这是件可怕的事...

重新设计数据结构

其实我们没必要把 `checkedRows` 设计成一个数组

我们可以设计成一个 `map`,这样读取值就只需要 `O(1)` 复杂度

Object 和 Map 的选择

此时应该有 彦祖会好奇,为什么要搞一个 `Map` 而不是 `Object` 呢?

其实要弄清楚这个问题,我们必须要知道他们之间的区别,网上的文章非常多,也介绍的非常详细

但有一点,是很多文章没有提及的,那就是 `Map 是有序的,Object 是无序的`

比如有个需求要获取 `第一个选中行,最后一个选中行`,那么我们利用 `Map` 实现就非常简单。

其次 我们可以用 `size` 方法轻松获取 `选中行数量`

改造代码

1.改造 `checkedRows`

```
this.crossPageMap = new Map()
```

2.修改选中逻辑(核心代码)

```
handleSelectChange (val, row) {
  // 实现了 O(n) 到 O(1) 的提升
  const checked = this.crossPageMap.has(row.id)
  if (checked) {
    this.crossPageMap.delete(row.id)
  } else {
    this.crossPageMap.set(row.id, row)
  }
}
```

3.修改换页回显逻辑

```
currentChange (page) {
  this.currentPage = page
  // 实现了 O(n^2) 到 O(n) 的提升
  this.tableData.forEach(row => {
    const checked = this.crossPageMap.has(row.id)
    if (checked) this.$refs.multipleTable.toggleRowSelection(row, true)
  })
}
```

完整代码

```vue
<template>
  <div>
    <el-table
      ref="multipleTable"
      :data="tableData"
      tooltip-effect="dark"
      style="width: 100%;height:500px"
      @select="handleSelectChange"
      @select-all="handleSelectAllChange"
    >
      <el-table-column
        type="selection"
        width="55"
      />
      <el-table-column
        label="日期"
        width="120"
        prop="date"
      />
      <el-table-column
        prop="name"
        label="姓名"
        width="120"
      />
    </el-table>
    <el-pagination
      background
      :current-page.sync="currentPage"
      layout="prev, pager, next"
      :total="1000"
      @current-change="currentChange"
    />
  </div>
</template>

<script>
export default {
  data () {
    return {
      currentPage: 1,
      crossPageMap: new Map(),
      pageSize: 10,
      totalData: Array.from({ length: 1000 }, (_, index) => {
        return {
```

```
45            date: '2016-05-03',
46            id: index,
47            name: '王小虎' + index
48          }
49        })
50      }
51    },
52    computed: {
53      tableData () {
54        const { currentPage, totalData, pageSize } = this
55        return totalData.slice((currentPage - 1) * pageSize, currentPage *
   pageSize)
56      }
57    },
58    methods: {
59      currentChange (page) {
60        this.currentPage = page
61        this.tableData.forEach(row => {
62          const checked = this.crossPageMap.has(row.id)
63          if (checked) this.$refs.multipleTable.toggleRowSelection(row, true)
64        })
65      },
66      handleSelectChange (val, row) {
67        const checked = this.crossPageMap.has(row.id)
68        if (checked) {
69          this.crossPageMap.delete(row.id)
70        } else {
71          this.crossPageMap.set(row.id, row)
72        }
73      },
74      handleSelectAllChange (val) {
75        this.tableData.forEach(row => {
76          const isChecked = this.crossPageIns.isChecked(row)
77          if (val.length === 0) {
78            // 取消全选 只有选中的需要改变状态
79            if (isChecked) this.crossPageIns.onRowSelectChange(row)
80          } else {
81            // 全选 只有未选中的才需要改变状态
82            if (!isChecked) this.crossPageIns.onRowSelectChange(row)
83          }
84        })
85      }
86    }
87  }
88  </script>
```

抽象业务逻辑

以上就是完整的业务代码部分,但是为了复用性。

我们考虑可以把其中的逻辑抽象成一个 `CrossPage` 类

设计 CrossPage 类

接收以下参数

```
`data` - 行数据
`key` - 行数据唯一值
`max` - 最大选中行数
`toggleRowSelection` - 切换行数据选中/取消选中的方法
```

提供以下方法

```
`onRowSelectChange` - 外部点行数据点击的时候调用此方法
`onDataChange` - 外部数据变化的时候调用此方法
`clear` - 清空所有选中行
`isChecked` - 判断当前行是否选中
```

构造器大致代码 如下

```
constructor (options={}) {
    this.crossPageMap = new Map()
    this.key = options.key || 'id'
    this.data = options.data || []
    this.max = options.max || Number.MAX_SAFE_INTEGER
    this.toggleRowSelection = options.toggleRowSelection
    if(typeof this.toggleRowSelection !== 'function') throw new Error('toggleRowSelection is not function')
}
```

设置私有crossPageMap

彦祖们,问题来了,我们把 `crossPageMap` 挂载到实例上,那么外部就可以直接访问修改这个变量。

这可能导致我们内部的数据逻辑错乱,所以必须禁止外部访问。

我们可以使用 修饰符来实现私有属性,具体参考

developer.mozilla.org/zh-CN/docs/...

完整代码

```
/**
 * @description 跨页选择
 * @param {Object} options
 * @param {String} options.key 行数据唯一标识
 * @param {Function} options.toggleRowSelection 设置行数据选中/取消选中的方法,必传
 */
export const CrossPage = class {
  #crossPageMap = new Map();
  constructor (options={}) {
    this.key = options.key || 'id'
    this.data = options.data || []
    this.max = options.max || Number.MAX_SAFE_INTEGER
    this.toggleRowSelection = options.toggleRowSelection
    if(typeof this.toggleRowSelection !== 'function') throw new Error('toggleRowSelection is not function')
  }
  get keys(){
    return Array.from(this.#crossPageMap.keys())
  }
  get values(){
    return Array.from(this.#crossPageMap.values())
  }
  get size(){
    return this.#crossPageMap.size
  }
  clear(){
    this.#crossPageMap.clear()
    this.updateViews()
  }
  isChecked(row){
    return this.#crossPageMap.has(row[this.key])
  }
  onRowSelectChange (row) {
    if(typeof row !== 'object') return console.error('row is not object')
    const {key,toggleRowSelection} = this
    if(this.isChecked(row)) this.#crossPageMap.delete(row[key])
    else {
      this.#crossPageMap.set(row[key],row)
      if(this.size>this.max){
        this.#crossPageMap.delete(row[key])
        toggleRowSelection(row,false)
      }
```

```
    }
  }
  onDataChange(list){
    this.data = list
    this.updateViews()
  }
  updateViews(){
    const {data,toggleRowSelection,key} = this
    data.forEach(row=>{
      toggleRowSelection(row,this.isChecked(row))
    })
  }
}

```