Vue Developer Interview

提交方法

- 第一题请将答案写在txt文件内提交。
- 第二题只需要把相关的.js和. vue文件用ZIP□不可用RAR或其他格式）。不要把脚手架文件如node_modules和第三方插件压缩在内。
- 压缩后电邮代码至 hr@gz.questwork.com 和 don.lee@questwork.com□

电邮内必须寫下你的名字和微信号，另外再附上你的pdf簡历，否则不予处理。

第一题

阅读代码并回答以下问题：

1. button.update({ active: true, disabled: false })会输出？为什么？

```
class Button {
  constructor(opt = {}) {
    opt = opt || {}
    this.status = {
      active: typeof opt.active === 'boolean' ? opt.active : true
      disabled: typeof opt.disabled === 'boolean' ? opt.disabled : true
    }
    this.rules = opt.rules || []
  }

  static init(layout) {
    const instance = new this(layout)
    return instance.isValid ? instance : null
  }

  get isValid() {
    return !!this
  }

  getValidation(data) {
    Object.keys(this.status).forEach((k) => {
      const rule = this.rules.find((r) => r.key === k)?.value
      this.status[k] = getValidation({ rule, data }) ? data[k] :
this.status[k]
    })
    return this
  }
```

```
  update(data = {}) {
    try {
      this.getValidation({ data })
      return this
    } catch (err) {
      throw err
    }
  }

}

function getValidation({ rule, data }) {
  if (!rule) {
    return true
  }
  const { key = '', value } = rule
  const [valueAttribute] = Object.keys(value || {})
  const rowValue = data[key]

  switch (valueAttribute) {
    case '$and': {
      const arr = value['$and']
      return arr.reduce((acc, item) => (acc && getValidation({ rule: item,
data })), true)
    }
    case '$or': {
      const arr = value['$or']
      return arr.reduce((acc, item) => (acc || getValidation({ rule: item,
data })), false)
    }
    case '$empty': {
      return !rowValue === !!value['$empty']
    }
    case '$eq': {
      return rowValue === value['$eq']
    }
    case '$gt': {
      return rowValue > value['$gt']
    }
    case '$gte': {
      return rowValue >= value['$gte']
    }
    case '$in': {
      if (Array.isArray(rowValue)) {
        return !!rowValue.find((e) => (value['$in'].includes(e)))
      }
      if (typeof rowValue !== 'object') {
        return !!value['$in'].includes(rowValue)
      }
      return false
```

```
    }
    case '$lt': {
      return rowValue < value['$lt']
    }
    case '$lte': {
      return rowValue <= value['$lte']
    }
    case '$ne': {
      return rowValue !== value['$ne']
    }
    case '$notIn': {
      if (Array.isArray(rowValue)) {
        return !rowValue.find((e) => (value['$notIn'].includes(e)))
      }
      if (typeof rowValue !== 'object') {
        return !value['$notIn'].includes(rowValue)
      }
      return false
    }
    default:
      return false
  }
}


const button = Button.init({
  active: false,
  disabled: true,
  rules: [
    {
      key: 'active',
      value: {
        key: '',
        value: {
          $and: [
            {
              key: 'errors',
              value: {
                $empty: true
              }
            },
            {
              key: 'disabled',
              value: {
                $eq: false
              }
            }
          ]
        }
      }
    }
```

```
  ]
})
```

第二题

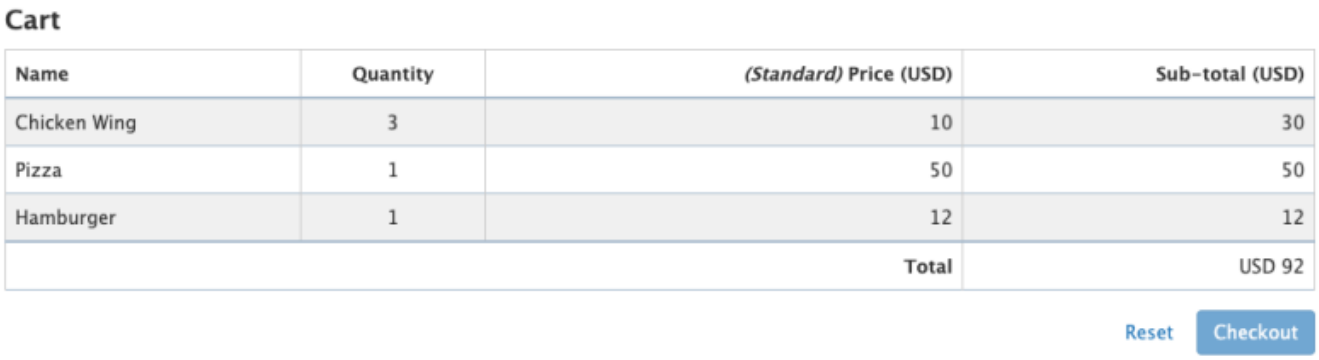
请用Vue.js 2.0或Vue.js .0以“SFC 单文件组件” 编码方式完成以下购物车题目。除了Vue.js 2.0, Vue.js 3.0, bootstrap 外，请尽量少用第三方库。不可使用Element UI, Mint UI等第三方库。

目标

- 提供前端端购物车功能

购物车外观

以下只是参考。



购物车功能

- 自动计算总金额（必须）
- 能删除已选择产品
- 能更改数量
- 能按產品名稱排序

数据结构 Data Structure

购物车数据为本地数据，不必从网上撷取。

```
[
{ id: 1, name: "Chicken Wing", category: "Food", qty: 3, price: 10 },
{ id: 2, name: "Pizza", category: "Food", qty: 1, price: 50 },
```

```
{ id: 3, name: "Hamburger", category: "Food", qty: 1, price: 12 },
{ id: 4, name: "Coca Cola", category: "Drink", qty: 2, price: 5 },
{ id: 5, name: "Orange Juice", category: "Drink", qty: 1, price: 15 },
{ id: 6, name: "Potato Chips", category: "Snack", qty: 1, price: 8 },
]
```

评分准则

- SFC组件设计
- 编码整洁
- 外观

加分项

- 组件化，模块化代码□□e.g List.vue, Button.vue)
- 使用Vue的API计算商品总价□ES6的新规范新API处理数组等