十五届模拟一

全部比赛地址 https://www.lanqiao.cn/contests/history/

1. 动态的 tab 栏

```
{
  /* TODO: 请在此补充代码实现tab栏动态固定 */
  top: 0;
  position: sticky;
}
```

- `position: sticky;` 是 CSS 中的一个定位属性，它可以用于创建一个在滚动过程中“粘性”定位的元素。当一个元素被设置为 `position: sticky;`时，它在滚动到特定的位置时会固定在容器中的某个位置，直到滚动到容器的底部。

2. 地球漫游

```
.earth-con {
  /* TODO:待补充代码，添加动画 */
  animation: orbit 36.5s linear infinite;
}
```

完整版

```
.earth-con {
  animation-name: orbit;
  animation-duration: 36.5s;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
}
```

3. 冰墩墩心情刻度尺

```
const range = document.getElementById("range");
const BingDunDun = document.querySelector(".BingDunDun");
BingDunDun.className = `BingDunDun not-satisfied`;

range.onchange = (e) => {
  // 进度条的值
  let value = Number(e.target.value);
  switch (value) {
    case 0:
      BingDunDun.className = `BingDunDun not-satisfied`;
      break;
    case 25:
      BingDunDun.className = `BingDunDun a-little-unsatisfied`;
      break;
    case 50:
      BingDunDun.className = `BingDunDun ordinary`;
      break;
    case 75:
      BingDunDun.className = `BingDunDun satisfied`;
      break;
    case 100:
      BingDunDun.className = `BingDunDun great`;
      break;
  }
};
```

4. 迷惑的 this

```
  handle() {
    // TODO: 待补充代码
    this.inputEl.addEventListener("input", this.handleInput.bind(this));
  },
```

5. 魔法失灵了

```
let { value } = toRefs(data);
```

- 你可以使用toRefs函数将data对象转换为响应式的Ref对象，然后通过解构赋值将value属性提取出来。

6. 燃烧你的卡路里

```
// 目标 1 <el-drawer v-model="drawer"></el-drawer>
```

```
// 定义变量
const drawer = ref(false);
// 点击切换
const submit = async () => {
  drawer.value = true;
};
// 导出
return {
  drawer,
};
```

```
// 目标 2
const sortItem = (arr, pro, compare) => {
  let i = 0;
  while (i < arr.length) {
    let tmp = arr.sort((a, z) => {
      return z[pro] - a[pro];
    })[i];
```

```
    if (tmp[pro] < compare) {
      return tmp;
    } else {
      i++;
    }
  }
};
```

7. 司龄统计

```
// 目标 1
const groupByAge = (peoples) => {
  // TODO:待补充代码，按照年龄进行分组 格式如下：
  // 输出格式示例：{
  //   1: [{ name: '杰克', age: 1 }, { name: '约翰', age: 1 }],
  //   2: [{ name: '丽莎', age: 2 }, { name: '豪尔赫', age: 2 }],
  //   5: [{ name: '艾娃', age: 5 }
  //   ...
  // }
  // 创建空对象，用于存储最终的分组结果
  const result = {};
  // 遍历每个人员，获取他们的年龄并进行分组
  for (const person of peoples) {
    const { age } = person;
    // 如果 result 对象中不存在 age 属性，则创建一个空数组并赋值给 result[age]
    if (!result[age]) {
      result[age] = [];
    }
    // 将当前的人员对象添加到 result[age] 数组中
    result[age].push(person);
  }
  // 返回按照年龄分组后的结果
  return result;
};

// 目标 2
// TODO: 设置 Echars X 轴数据和 Y 轴数据
xAxisData.value = Object.keys(groupedPeople.value);
seriesData.value = Object.values(groupedPeople.value).map(
  (item) => item.length
);
```

8. 不翼而飞的余额

```
// 目标 1
const router = createRouter({
```

```
  history: createWebHistory(),
  routes: [
    { path: "/", component: WalletPage },
    { path: "/deposit", component: DepositPage },
  ],
});
```

```
// store.js
// 存款事件
const depositMoney = (money) => {
  balance.value = Number(balance.value) + Number(money.value);
};
return {
  // ...
  depositMoney,
};

// DepositPage.js
<span id="deposit-balance">{{ store.balance }}</span>

function deposit() {
      store.depositMoney(depositAmount)
       depositAmount.value = '';
    }

return {
  // ...
  store,
  depositAmount
};
```

9. 个性化推荐

```
req.on("end", () => {
  let { interested = [] } = qs.parse(body);

  // TODO: 补充个性化页面处理代码
  interested = typeof interested === "string" ? [interested] : interested;

  let content = "";
  if (interested.length === 0) {
    content = "<div class='unselect'>你还未选择任何感兴趣的标签! </div>";
  } else {
    // console.log(1,interested)
    let tl = [];
    interested.forEach((interest) => {
      const target = data.find((item) => item.tag === interest);
      if (target) {
        tl.push(interest);
```

```
        // console.log(2,target.relevance)
        tl = tl.concat(target.relevance);
        // console.log(3,tl)
      }
    });
    let resultList = Array.from(new Set(tl));
    // console.log(4,resultList)
    resultList.forEach((tag) => {
      let t = data.find((d) => d.tag === tag);
      if (t) {
        content += `
              <div class="interest">
                <div class="tag">${t.tag}</div>
                <div>${t.content}</div>
              </div>
              `;
      }
    });

    // console.log(5,content)
  }

  const origin = fs.readFileSync(path.join(__dirname, "../customized.html"),
{
    encoding: "utf8",
  });
  content = origin.replace(/<body>.*<\/body>/, `<body>${content}</body>`);
  res.writeHead(200, { "Content-Type": "text/html" });
  res.write(content);
  res.end();
});
```

10. 贪吃蛇

```
nextStep() {
  let snakeHead; // 定义变量用于存储蛇头的位置
  let snake = this.snakeBody; // 获取蛇的身体部分
  if (this.direction === 'right') {
    // 如果蛇的移动方向是向右
    snakeHead = { ...snake[0], left: snake[0].left + this.size }; // 创建一个
新的蛇头对象，其位置在当前蛇头的右边
  } else if (this.direction === 'left') {
    // 如果蛇的移动方向是向左
    snakeHead = { ...snake[0], left: snake[0].left - this.size }; // 创建一个
新的蛇头对象，其位置在当前蛇头的左边
  } else if (this.direction === 'up') {
    // 如果蛇的移动方向是向上
    snakeHead = { ...snake[0], top: snake[0].top - this.size }; // 创建一个新
的蛇头对象，其位置在当前蛇头的上方
  } else if (this.direction === 'down') {
    // 如果蛇的移动方向是向下
    snakeHead = { ...snake[0], top: snake[0].top + this.size }; // 创建一个新
```

```
的蛇头对象，其位置在当前蛇头的下方
  }
  this.snakeBody.unshift(snakeHead); // 将新的蛇头对象添加到蛇身体的前部，使其成为
新的蛇头
  this.snakeBody.pop(); // 移除蛇身体的最后一个元素，即蛇的尾部
}
```

11. 会员卡定制

```
<span v-for="(n, $index) in otherCardMask" :key="$index">
  <transition name="slide-fade-up">
    <!-- TODO: 在下面补充代码，完成输入卡号的显示效果。
                          卡号显示输入值的前 19 位（包括空格符），效果设置为：前 5 位
（包括空格）和后 4 位显示具体输入的值，中间 10 位分两组，每组显示 5 个字符，且每组前 4 位
由 * 号代替，最后一位显示对应的输入值。例如：输入的卡号为 20 位字符串
“12345678901234567890” 显示效果为 “12345 ****0 ****5 6789”-->
    <div
      class="card-item__numberItem"
      v-if="cardNumber && $index > 4 && $index < 15 && cardNumber.length >
$index && n.trim() !== ''"
    >
      *
    </div>
    <div
      class="card-item__numberItem"
      :class="{ '-active' : n.trim() === '' }"
      :key="$index"
      v-else-if="cardNumber && cardNumber.length > $index"
    >
      {{cardNumber[$index]}}
    </div>
    <div
      class="card-item__numberItem"
      :class="{ '-active' : n.trim() === '' }"
      v-else
      :key="$index + 1"
    >
      {{n}}
    </div>
  </transition>
</span>
<!-- 目标 2 -->
<span
  class="card-item__nameItem"
  v-for="(n, $index) in cardName.replace(/\s\s+/g, ' ')"
  :key="$index + 1"
  >{{n}}</span
>
```

```
// 目标 3
// TODO
const minCardMonth = computed(() => {
  if (data.cardYear === data.minCardYear) {
    return new Date().getMonth() + 1;
  } else {
    return 1;
  }
});
// TODO
watch(
  () => {
    if (data.cardMonth < minCardMonth.value) {
      data.cardMonth = "";
    }
    return data.cardYear.value; // 返回想要监听的响应式数据，ref 定义的数据需要返回
其 value 属性
  },
  () => {
    console.log("data.cardYear 发生了变化");
  },
  {
    immediate: true,
  }
);
```

12. 表单验证器

```
//  TODO: 目标 1 当输入框的值变化时，触发 input 事件更新父组件的 v-model 值
watch(inputValue, (newValue) => {
  emit("update:modelValue", newValue);
});


//  TODO: end

// 目标 2
const is_email = (val) => {
  // TODO: 待补充代码
  return /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/.test(val);
};

// 目标 3
// TODO: 编写通用的表单验证规则 15 分
for (const field in props.rules) {
  const fieldRules = props.rules[field];
  const value = props.formData[field];

  for (const rule of fieldRules) {
    if (typeof rule === "object" && rule.validator) {
```

```
        // 处理包含 validator 函数的验证规则
        rule.validator(rule, value, (error) => {
          if (error) {
            // 如果存在错误，将错误信息添加到 errors 对象中
            errors.value[field] = error.message;
          }
        });
      } else if (rule.required && !value) {
        // 处理预定义验证规则：必填项
        errors.value[field] = rule.message;
      } else if (rule.type) {
        // 根据类型进行验证
        const validationError = validateByType(rule.type, value);
        if (!validationError) {
          if (!errors.value[field]) {
            errors.value[field] = rule.message;
          }
        }
      }
      // 验证字段长度是否在指定范围内
      if (value.length < rule.min || value.length > rule.max) {
        if (!errors.value[field]) {
          errors.value[field] = rule.message;
        }
      }
    }
  }
  // TODO: END
```

API 地址