Markdown 文档解析

介绍

Markdown 因为其简洁的语法大受欢迎，已经成为大家写博客或文档时必备的技能点，众多博客平台都提倡用户使用 Markdown 语法进行文章书写，然后再发布后，实时的将其转化为常规的 HTML 页面渲染。

本题需要在已提供的基础项目中，使用 Nodejs 实现简易的 Markdown 文档解析器。

准备

开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├── docs.md
├── images
│   └── md.jpg
├── index.html
└── js
    ├── index.js
    └── parse.js
```

其中：

- `index.html` 是主页面。
- `images` 是图片文件夹。
- `docs.md` 是需要解析的 Markdown 文件。
- `js/index.js` 是提供的工具脚本，用于快速验证代码结果。
- `js/parse.js` 是需要补充的脚本文件。

目标

在 `js/parse.js` 中实现几种特定的 Markdown 语法解析，目前初始文件中已实现标题解析（即从 `#` 前缀转换为 `<hn>` 标签），请你继续完善该文件 TODO 部分，完成剩余语法解析操作，具体需求如下：

1. 对分隔符进行解析，Markdown 中使用 `---` （三条及以上的短横线） 作为分隔符，将其解析成为 `<hr>` 标签：

```
<!-- Markdown -->
----

<!-- 对应 HTML  -->
<hr />
```

2. 对引用区块进行解析，Markdown 中使用 `>` 作为前缀，将其解析成为 `<blockquote>` 标签：

```
<!-- Markdown 单个引用区块 -->
> 引用区块1
<!-- Markdown 引用区块 -->
> 多级引用区块2
> 多级引用区块2

<!-- 对应 HTML  -->
<blockquote>
  <p>引用区块1</p>
</blockquote>

<blockquote>
  <p>多级引用区块2</p>
  <p>多级引用区块2</p>
</blockquote>
```

3. 对无序列表进行解析，Markdown 中使用 `*` 或者 `-` 作为前缀，将其解析成为 `<ul>` 标签：

```
<!-- Markdown  -->
* 无序列表
* 无序列表
* 无序列表
或者：
- 无序列表
- 无序列表
- 无序列表

<!-- 对应 HTML  -->
<ul>
  <li>无序列表</li>
  <li>无序列表</li>
  <li>无序列表</li>
</ul>
```

4. 对图片进行解析，Markdown 中使用 `![alt](link)` 表示，将其解析成为 `<img>` 标签：

```
<!-- Markdown  -->
![图片](./images/md.jpg)

<!-- 对应 HTML  -->
<img src="./images/md.jpg" alt="图片" />
```

5. 对文字效果进行解析，比如粗体效果，和行内代码块，将其分别解析成 `<b>` 和 `code` 标签：

```
<!-- Markdown  -->
这是**粗体**的效果文字，这是内嵌的`代码行`

<!-- 对应 HTML  -->
这是<b>粗体</b>的效果文字，这是内嵌的<code>代码行</code>
```

在验证代码效果时，你可以在终端运行：

```
node ./js/index.js
```

程序会将解析的结果输出到 `index.html` 文件中，然后通过浏览器查看输出的 `index.html` 是否符合解析要求（注意：程序不会实时的将结果更新到 `index.html` 文件中，在你的代码变更后，请重新执行上述命令）。

在题目所提供的数据的情况下，完成后的效果如下：

规定

- 请勿修改 `js/parse.js` 文件外的任何内容。
- 请严格按照考试步骤操作，切勿修改考试默认提供项目中的文件名称、文件夹路径、class 名、id 名、图片名等，以免造成判题无法通过。
- 完成答题后，考生需将题目对应代码文件夹压缩成 zip 格式后上传提交，其他压缩包格式为 0 分。

判分标准

- 完成目标 1，得 5 分。
- 完成目标 2，得 5 分。
- 完成目标 3，得 10 分。
- 完成目标 4 与 目标 5，得 5 分。

组课神器

介绍

在很多教育网站的平台上，课程的章节目录会使用树型组件呈现，为了方便调整菜单，前端工程师会为其赋予拖拽功能。本题需要在已提供的基础项目中，完成可拖拽树型组件的功能。

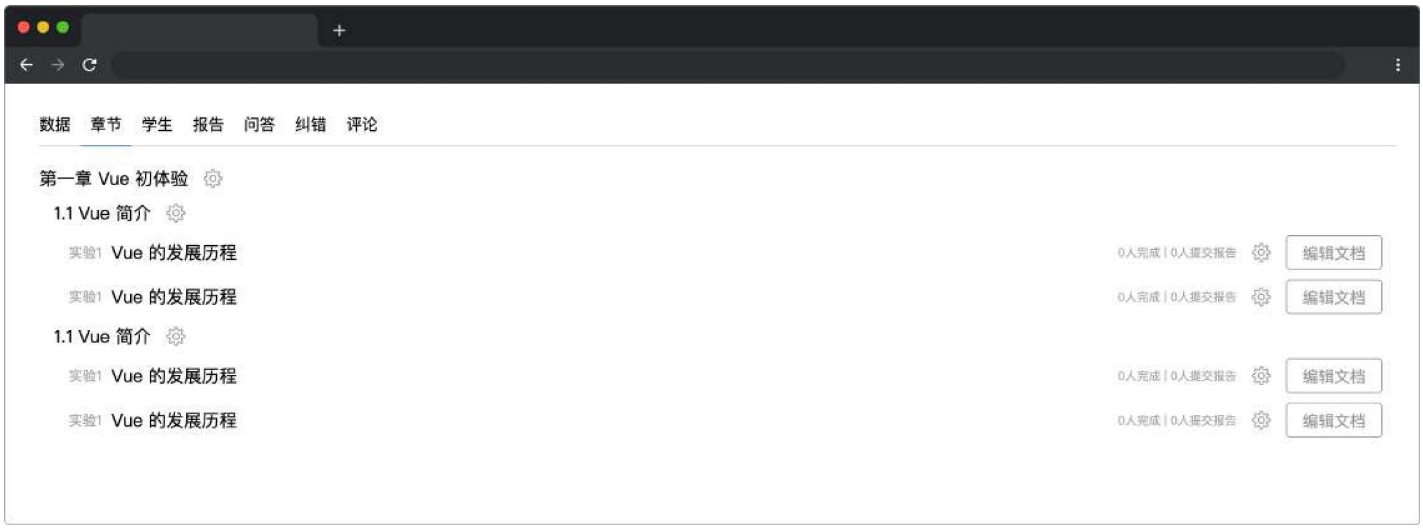
准备

```
├── effect.gif
├── css
│   └── style.css
├── index.html
├── images
└── js
    ├── data.json
    ├── axios.min.js
    └── index.js
```

其中：

- `index.html` 是主页面。
- `images` 是图片文件夹。
- `js/index.js` 是需要补充代码的 `js` 文件。
- `js/data.json` 是存放数据的 json 文件。
- `js/axios.min.js` 是 axios 文件。
- `css/style.css` 是样式文件。
- `effect.gif` 是完成的效果图。

在浏览器中预览 `index.html` 页面，显示如下所示：



目标

请在 `js/index.js` 文件中补全代码。具体需求如下：

1. 补全 `js/index.js` 文件中 `ajax` 函数，功能为根据请求方式 `method` 不同，拿到树型组件的数据并返回。具体如下：

- 当 `method === "get"` 时，判断 `localStorage` 中是否存在 `key` 为 `data` 的数据，若存在，则从 `localStorage` 中直接获取后处理为 json 格式并返回；若不存在则从 `./js/data.json` （必须使用该路径请求，否则可能会请求不到数据）中使用 ajax 获取并返回。
- 当 `method === "post"` 时，将通过参数 `data` 传递过来的数据转化为 json 格式的字符串，并保存到 `localStorage` 中， `key` 命名为 `data` 。

最终返回的数据格式如下：

```
[
    {
        "id": 1001,
        "label": "第一章 Vue 初体验",
        "children": [ ... ]
    },
    {
      "id": 1006,
      "label": "第二章 Vue 核心概念",
      "children": [
          {
              "id": 1007,
              "label": "2.1 概念理解",
              "children": [
                  {
                      "id": 1008,
                      "label": "聊一聊虚拟 DOM",
                      "tag":"文档 1"
                  },
                  ...
              ]
          },
          {
              "id": 1012,
              "label": "2.2 Vue 基础入门",
              "children": [
                  {
                      "id": 1013,
                      "label": "Vue 的基本语法",
                      "tag":"实验 6"
                  },
                  ...
              ]
          }
      ]
  }
]
```

2. 补全 `js/index.js` 文件中的 `treeMenusRender` 函数，使用所传参数 `data` 生成指定 DOM 结构的模板字符串（完整的模板字符串的 `HTML` 样例结构可以在 `index.html` 中查看），并在包含 `.tree-node` 的元素节点上加上指定属性如下：

属性名	属性值	描述
`data-grade`	`${grade}`	表示菜单的层级，整数，由 `treeMenusRender` 函数的 `grade` 参数值计算获得，章节是 1，小节是 2，实验文档是 3。
`data-index`	`${id}`	表示菜单的唯一 id，使用每层菜单数据的 `id` 字段值。

3. 补全 `js/index.js` 文件中的 `treeDataRefresh` 函数，功能为：根据参数列表 `{ dragGrade, dragElementId }, { dropGrade, dropElementId }` 重新生成拖拽后的树型组件数据 `treeData` （`treeData` 为全局变量，直接访问并根据参数处理后重新赋值即可）。

方便规则描述，现将 `data.json` 中的数据扁平化处理，得到的数据顺序如下：

```
[
  { grade: "1", label: "第一章 Vue 初体验", id: "1001" },
  { grade: "2", label: "1.1 Vue 简介", id: "1002" },
  { grade: "3", label: "Vue 的发展历程", id: "1003" },
  { grade: "3", label: "Vue 特点", id: "1004" },
  { grade: "3", label: "一分钟上手 Vue", id: "1005" },
  { grade: "1", label: "第二章 Vue 核心概念", id: "1006" },
  { grade: "2", label: "2.1 概念理解", id: "1007" },
  { grade: "3", label: "聊一聊虚拟 DOM", id: "1008" },
  { grade: "3", label: "感受一下虚拟 DOM", id: "1009" },
  { grade: "3", label: "聊一聊 MVVM 设计模式", id: "1010" },
  { grade: "3", label: "Vue 中的 MVVM 设计模式", id: "1011" }, // 即将被拖拽的元素节点
  { grade: "2", label: "2.2 Vue 基础入门", id: "1012" },
  { grade: "3", label: "Vue 的基本语法", id: "1013" },
  { grade: "3", label: "第一步，创建 Vue 应用实例", id: "1014" },
];
```

拖拽前后的规则说明如下：

- 情况一：若拖拽的节点和放置的节点为同级，即 `treeDataRefresh` 函数参数列表中 `dragGrade == dropGrade` ，则将 `id == dragElementId` （例如： `1011` ）的节点移动到 `id==dropElementId` （例如： `1008` ）的节点后，作为其后第一个邻近的兄弟节点。最终生成的数据顺序如下：

```
[
  { grade: "1", label: "第一章 Vue 初体验", id: "1001" },
  { grade: "2", label: "1.1 Vue 简介", id: "1002" },
  { grade: "3", label: "Vue 的发展历程", id: "1003" },
  { grade: "3", label: "Vue 特点", id: "1004" },
  { grade: "3", label: "一分钟上手 Vue", id: "1005" },
  { grade: "1", label: "第二章 Vue 核心概念", id: "1006" },
  { grade: "2", label: "2.1 概念理解", id: "1007" },
  { grade: "3", label: "聊一聊虚拟 DOM", id: "1008" },
  // 在目标元素节点下方插入
  { grade: "3", label: "Vue 中的 MVVM 设计模式", id: "1011" },
  { grade: "3", label: "感受一下虚拟 DOM", id: "1009" },
  { grade: "3", label: "聊一聊 MVVM 设计模式", id: "1010" },
  // 移除被拖拽的元素节点
  { grade: "2", label: "2.2 Vue 基础入门", id: "1012" },
  { grade: "3", label: "Vue 的基本语法", id: "1013" },
  { grade: "3", label: "第一步，创建 Vue 应用实例", id: "1014" },
];
```

- 情况二：若拖拽的节点和放置的节点为上下级，即 `treeDataRefresh` 函数参数列表中 `dragGrade - dropGrade == 1` ，则将 `id == dragElementId` （例如： `1011` ）的节点移动到 `id==dropElementId` （例如： `1002` ）的节点下，并作为其第一个子节点。最终生成的数据顺序如下：

```
[
  { grade: "1", label: "第一章 Vue 初体验", id: "1001" },
  { grade: "2", label: "1.1 Vue 简介", id: "1002" },
  // 在目标元素节点下方插入
  { grade: "3", label: "Vue 中的 MVVM 设计模式", id: "1011" },
  { grade: "3", label: "Vue 的发展历程", id: "1003" },
  { grade: "3", label: "Vue 特点", id: "1004" },
  { grade: "3", label: "一分钟上手 Vue", id: "1005" },
  { grade: "1", label: "第二章 Vue 核心概念", id: "1006" },
  { grade: "2", label: "2.1 概念理解", id: "1007" },
  { grade: "3", label: "聊一聊虚拟 DOM", id: "1008" },
  { grade: "3", label: "感受一下虚拟 DOM", id: "1009" },
  { grade: "3", label: "聊一聊 MVVM 设计模式", id: "1010" },
  // 移除被拖拽的元素节点
  { grade: "2", label: "2.2 Vue 基础入门", id: "1012" },
  { grade: "3", label: "Vue 的基本语法", id: "1013" },
  { grade: "3", label: "第一步，创建 Vue 应用实例", id: "1014" },
];
```

最终效果可参考文件夹下面的 gif 图，图片名称为 `effect.gif` （提示：可以通过 VS Code 或者浏览器预览 gif 图片）。

规定

- 请勿修改 `js/index.js` 文件外的任何内容。
- 请严格按照考试步骤操作，切勿修改考试默认提供项目中的文件名称、文件夹路径、class 名、id 名、图片名等，以免造成判题无法通过。
- 完成答题后，考生需将题目对应代码文件夹压缩成 zip 格式后上传提交，其他压缩包格式为 0 分。

判分标准

- 完成目标 1，得 5 分。
- 完成目标 2，得 10 分。
- 完成目标 3，得 10 分。