

第十五届蓝桥杯全国软件和信息技术专业人才大赛（Web 应用开发）

竞赛须知

请选手逐条仔细阅读以下须知：

1. 考试开始后，选手首先下载题目，并使用考场现场公布的解压密码解压试题。
2. 考试时间为 4 个小时，时间截止后，提交答案无效。
3. 在考试强制结束前，选手可以主动结束考试（需要身份验证），结束考试后将无法继续提交或浏览答案。
4. 选手可浏览自己已经提交的答案，被浏览的答案允许拷贝。
5. 对同一题目，选手可多次提交答案，以最后一次提交的答案为准。
6. 选手切勿在提交的代码中书写“姓名”、“考号”、“院校名”等与身份有关的信息或其它与竞赛题目无关的内容，否则成绩无效。
7. 选手必须通过浏览器方式提交自己的答案。选手在其它位置的作答或其它方式提交的答案无效。
8. 试题类型均为场景实战题。

注意事项

请选手逐条仔细阅读以下须知：

1. 基础源代码在无明确说明的情况下，请勿随意修改文件名称、文件夹名称、文件存放结构等。严格遵守题目中的注意事项。
2. 全部题目将使用前端自动化测试技术完成机器自动评分，务必严格规范根据题意操作，否则可能会影响最终机器阅卷的准确性。
3. 在评卷时使用的输入数据与试卷中给出的示例数据可能是不同的。选手的程序必须是通用的，不能只对试卷中给定的数据有效。

省赛（大学组）

试题介绍

场景实战题目均包含 2 部分内容：**题面 PDF 文档**和**基础源代码压缩包**。

1. 题面 PDF 文档可以使用本地 PDF 阅读器或者 Google Chrome 浏览器打开阅读。题面文档中会详细说明题目的背景、需求、目标。
2. 基础源代码压缩包中会给出相应题目的基础代码。一般情况下，选手需认真读题，结合需求和给出的基础源代码，通过修改、新增代码来实现题目最终目标。

部分题目可能包含前序准备步骤。例如解压缩相应的资源文件，在浏览器中预览网页效果等。大部分情况下，我们默认选手已经掌握了前端开发过程中可能涉及的基础知识和方法，不会给予单独的提示。同时，题目不会给予 IDE 开发工具的使用方法提示。

试题列表和分值

试题序号、试题名称及基础源代码文件夹名称对应如下：

- 01 智能停车系统（5 分）
- 02 布局切换（5 分）
- 03 产品 360 度展示（10 分）
- 04 多表单验证（10 分）
- 05 找回连接的奇幻之旅（15 分）
- 06 tree 命令助手（15 分）
- 07 Github 明星项目统计（20 分）
- 08 小蓝驿站（20 分）
- 09 商品浏览足迹（25 分）
- 10 NPM Download Simulator（25 分）

试题提交说明

1. 考试给出题目压缩包中包含 PDF 题面，和 10 道题目的基础代码子文件夹，和题目序号一一对应。
2. 考生需根据题意和目标，在考试给出的基础代码上新增、修改代码。请勿修改给出代码文件夹名称，层次结构。
3. 相应题目的代码完成之后，**考生需将题目对应代码文件夹压缩成 zip 或 rar 格式后上传提交，其他格式为 0 分**。例如 01 代码文件夹，应该在此文件夹基础上直接压缩后，为 01.zip 或 01.rar，然后提交压缩包到对应题目。请注意不要给压缩包添加密码。
4. 请务必严格按照规范提交题目代码，否则会造成无法被系统正确判分。

智能停车系统

介绍

蓝桥社区停车场正式对外开放，但由于停车位标记线不够完善，车主总是乱停乱放。为了使车辆有序的停放，现在给停车场绘制了醒目的停车位标记线。

本题需要在已提供的基础项目中，使用 CSS 让每辆车正确停放至停车位。

准备

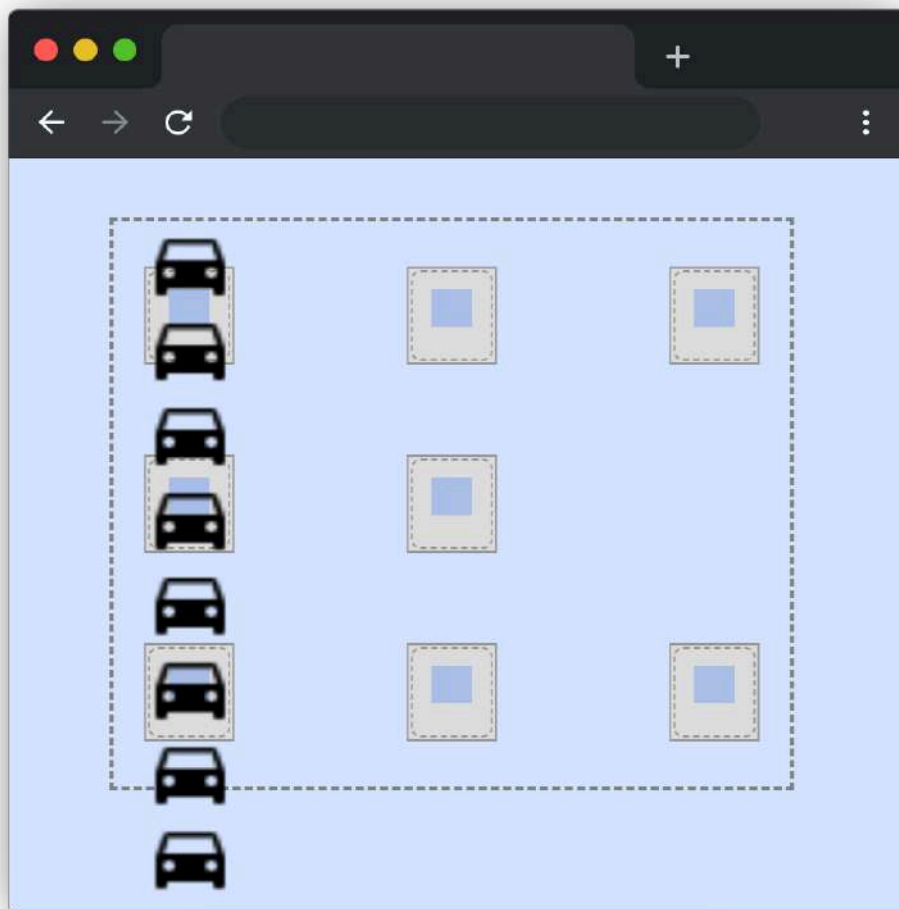
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ images
├─ css
│   └─ style.css
└─ index.html
```

其中：

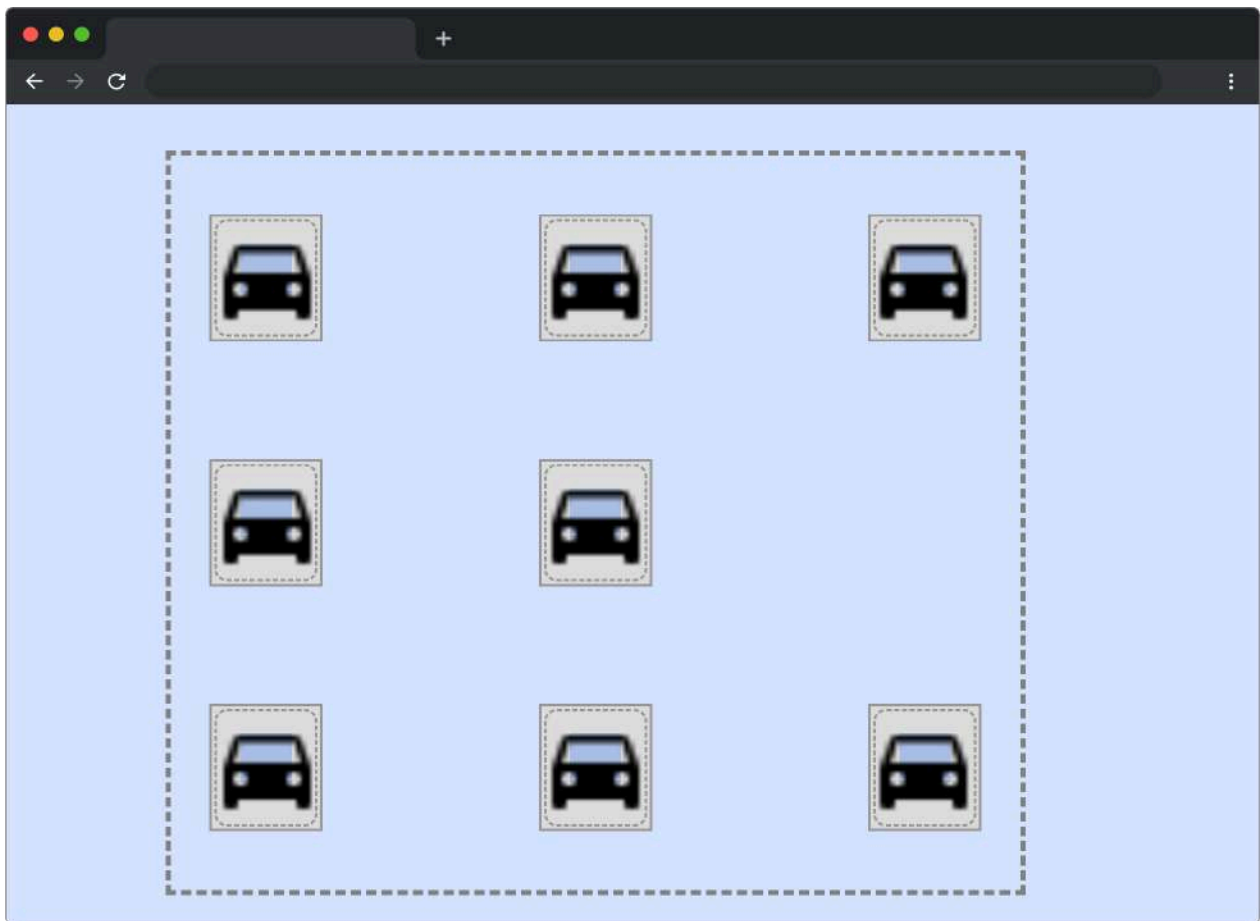
- images 是图片文件夹。
- index.html 是主页面。
- css/style.css 是待补充代码的 css 文件。

在浏览器中预览 index.html 页面效果如下：



目标

请使用 Flex 完善 `css/style.css` 中的 TODO 代码，让每辆车正确停放至停车位，最终实现下图效果。



提示：设置子元素为（`flex-wrap`）换行，在交叉轴上使用（`align-content`）两端对齐分布，在主轴上使用（`justify-content`）两端对齐分布。两端对齐分布效果即首个元素放置于起点，末尾元素放置于终点，其余元素均匀排列。

规定

- 请勿修改 `css/style.css` 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。

判分标准

- 本题完全实现题目目标得满分，否则得 0 分。

布局切换

介绍

为了提高用户体验，网站有时需要多种浏览模式。现在特邀请你为蓝桥官网设计具有经典、浏览和工具三种布局模式。使用户可以根据具体情况选择合适的模式，以便更好地浏览网页内容。

本题需要在已提供的基础项目中使用 JS 完善代码实现布局的切换。

准备

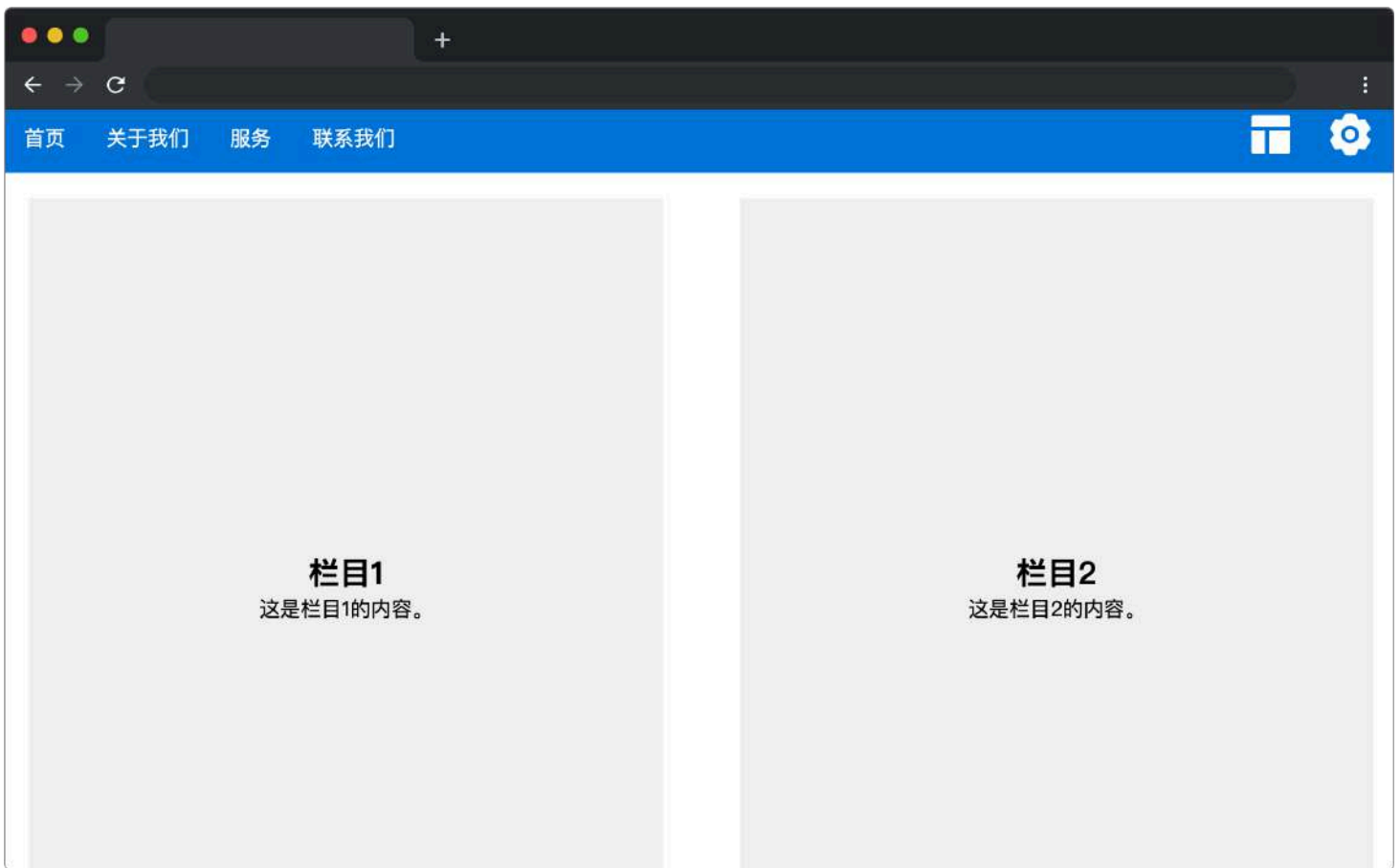
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ css
├─ images
├─ index.html
├─ effect.gif
└─ js
    └─ index.js
```

其中：

- `css` 是样式文件夹。
- `images` 是图片文件夹。
- `index.html` 是主页面。
- `effect.gif` 是最终完成效果图。
- `js/index.js` 是待补充代码的 js 文件。

在浏览器中预览 `index.html` 页面效果如下：



目标

完善 `js/index.js` 的 TODO 部分的代码，实现被点击的模式元素（`class=layout-option`）处于激活状态，即添加一个类名（`active`），其他元素（`class=layout-option`）移除激活状态，即移除类名（`active`）。

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。

判分标准

- 本题完全实现题目目标得满分，否则得 0 分。

产品 360 度展示

介绍

在电子商务网站中，用户可以通过鼠标或手势交互实现 360 度全方位查看产品，提升用户体验。现在需要你设计一个 Pipeline 管道函数，用于控制 360 度展示产品的动画序列，通过管道连接各个动画步骤，使产品以流畅的方式呈现给用户。

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：

```
├─ css
├─ effect.gif
├─ js
│   └─ index.js
│   └─ utils.js
└─ index.html
```

其中：

- `css` 是样式文件夹。
- `index.html` 是主页面。
- `js/index.js` 是动画序列代码的 js 文件。
- `js/utils.js` 是待补充代码的 js 文件。
- `effect.gif` 是页面最终的效果图。

在浏览器中预览 `index.html` 页面效果如下：



目标

请在 `js/utils.js` 文件中的 `TODO` 部分，封装一个支持异步的 `pipeline` 管道函数，能够按顺序执行一系列异步操作，每个异步操作的结果将作为下一个异步操作的输入。

函数参数说明：

- `initialValue`：管道的初始值（第一个异步步骤将以此值开始，即 `sequence` 中第一个函数的参数）。它是整个异步管道的起点。
- `sequence`：是一个由具有返回值和可以传参的函数组成的数组，函数可以是普通函数也可以是 `Promise` 函数。每个函数接收前一个步骤的输出（即该函数的参数是上一个函数的执行结果），并返回一个 `Promise`。这个数组定义了整个管道中的处理步骤和它们的顺序。

函数返回值说明：

- `pipeline` 函数返回一个 `Promise`，这个 `Promise` 最终解析为整个管道执行完成后的结果。

测试用例如下：

请注意以下用例仅供参考，实际判题时会修改测试用例，请保证代码的通用性。

- 示例 1: sequence 中都为普通函数

```
function fn1(data){
    return data + '2.开始左转'
}
function fn2(data){
    return data+'3.开始右转'
}
pipeline('1.动画开始',[fn1, fn2]).then(res => {
    console.log(res)
})
// 打印结果为: '1.动画开始2.开始左转3.开始右转'
```

- 示例 2: sequence 中都为 promise 函数

```
function fn1(data){
    return new Promise(resolve => {
        resolve(data+'->执行第一个动画')
    })
}
function fn2(data){
    return new Promise(resolve => {
        resolve(data+'->执行第二个动画')
    })
}
pipeline('动画开始',[fn1, fn2]).then(res => {
    console.log(res)
})
// 打印结果为: '动画开始->执行第一个动画->执行第二个动画'
```

完成后, 点击屏幕触发动画序列, 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/utils.js 文件外的任何内容。
- 判题时会随机提供不同参数对 pipeline 函数功能进行检测, 请保证函数的通用性, 不能仅对测试数据有效。
- 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。

判分标准

- `sequence` 参数都为普通函数且满足需求正确输出，得 5 分。
- `sequence` 参数都为 `promise` 函数且满足需求正确输出，得 5 分。

多表单验证

介绍

常见的收集信息的页面只有一个表单。但是在一些特殊情况下，一个页面可能同时存在多个表单，例如通过 `tab` 切换表单收集不同类别的信息，这时候就需要我们在表单提交时进行所有表单的校验，全部校验通过才能提交数据。

准备

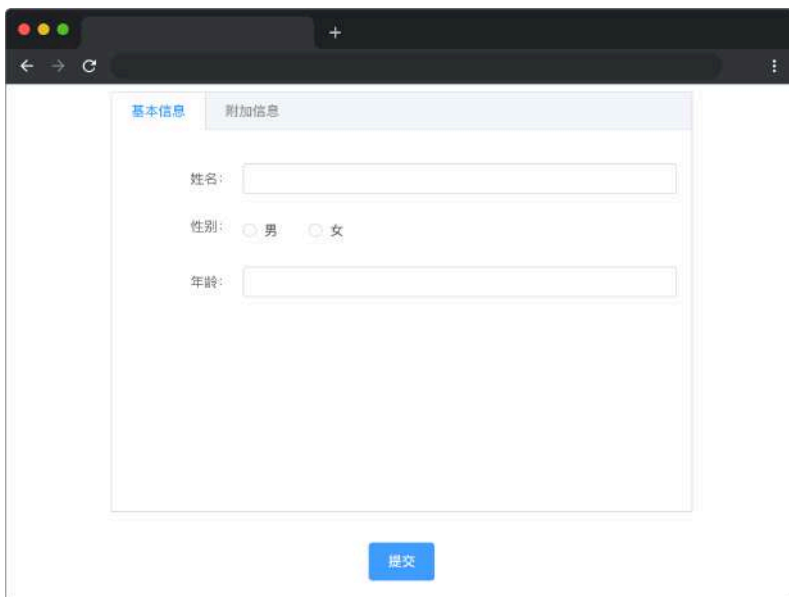
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ css
├─ index.html
└─ lib
```

其中：

- `css` 是样式文件夹。
- `lib` 是依赖文件夹。
- `index.html` 是主页面。

在浏览器中预览 `index.html` 页面效果如下：



The image shows a web browser window displaying a form. The form has two tabs at the top: '基本信息' (Basic Information) and '附加信息' (Additional Information). The '基本信息' tab is selected. Below the tabs, there are three input fields: '姓名' (Name), '性别' (Gender) with radio buttons for '男' (Male) and '女' (Female), and '年龄' (Age). At the bottom of the form, there is a blue button labeled '提交' (Submit).

目标

完善 `index.html` 中的 TODO 部分，实现多表单校验功能。

两个表单校验规则（`rules`）如下，通过 `el-form-item` 的 `prop` 进行对应：

- 姓名：必填。错误提示：请输入姓名。规则：输入的姓名只能为汉字。错误提示：只能输入汉字。
- 性别：必填。错误提示：请选择性别。
- 年龄：必填。错误提示：请输入年龄。
- 是否参加过编程比赛：必填。错误提示：请选择是否参加过编程比赛。
- 是否有过创业经历：必填。错误提示：请选择是否有过创业经历。

检验是否为汉字的正则表达式规则如下：

```
const reg = /^[^\u4e00-\u9fa5]/g;
```

表单添加好校验规则后，点击提交。效果如下：

The image displays two browser windows side-by-side, showing a web form with two tabs: '基本信息' (Basic Information) and '附加信息' (Additional Information). Both windows show validation errors for the 'Basic Information' tab.

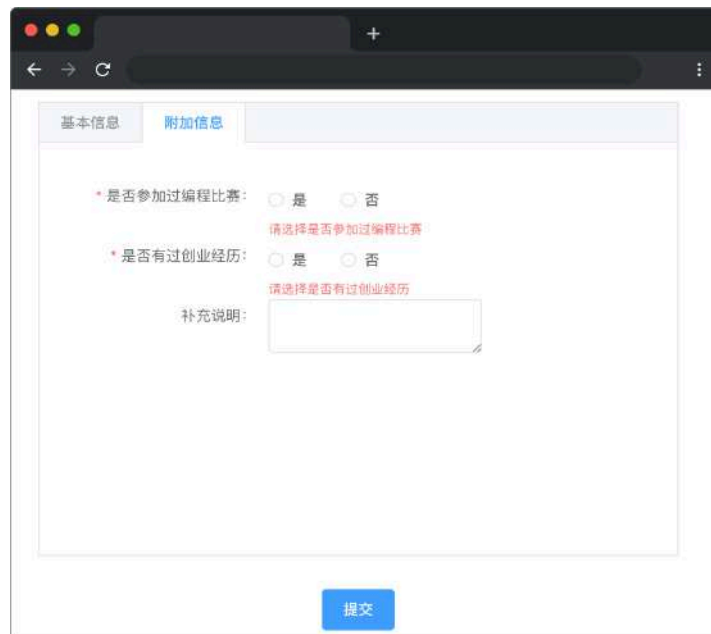
Left Window (Basic Information Tab):

- 姓名 (Name):** Input field is empty. Error message: 请输入姓名 (Please enter name).
- 性别 (Gender):** Radio buttons for '男' (Male) and '女' (Female) are present. Error message: 请选择性别 (Please select gender).
- 年龄 (Age):** Input field is empty. Error message: 请输入年龄 (Please enter age).

Right Window (Additional Information Tab):

- 姓名 (Name):** Input field contains 'lanqiao'. Error message: 只能输入汉字 (Only Chinese characters allowed).
- 性别 (Gender):** Radio buttons for '男' (Male) and '女' (Female) are present. Error message: 请选择性别 (Please select gender).
- 年龄 (Age):** Input field is empty. Error message: 请输入年龄 (Please enter age).

Both windows have a blue '提交' (Submit) button at the bottom.



本项目使用到的 `element-plus` 表单验证 API 如下：

el-form 组件参数说明

参数	说明	类型
<code>rules</code>	表单验证规则	<code>object</code>

rules 对象校验规则配置项说明

参数	说明	类型
<code>required</code>	是否必填	<code>boolean</code>
<code>message</code>	错误提示文案	<code>string</code>
<code>trigger</code>	验证逻辑的触发方式	<code>blur</code> 、 <code>change</code>
<code>validator</code>	自定义校验规则	<code>function</code>

自定义校验规则需要创建一个校验函数，参数为 `rule`、`value`、`callback`，校验通过调用 `callback` 回调即可，校验失败调用 `callback` 抛出异常信息。用法参考以下示例：

```
const validatePassword = (rule, value, callback) => {  
  if (value === '') {  
    callback(new Error("请输入密码"));  
  } else if (value !== ruleForm.pass) {  
    callback(new Error("密码输入不一致"));  
  } else {  
    callback();  
  }  
};
```

rules 配置格式示例代码：

```
{  
  x: [  
    { validator: validatePassword, trigger: 'blur' }, // 自定义验证规则  
    { required: true, message: 'xxx', trigger: 'blur' }  
  ],  
  y: [{ required: true, message: 'xxx', trigger: 'change' }], // 普通验证规则  
}
```

规定

- 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。

判分标准

- 本题完全实现题目目标得满分，否则得 0 分。

找回连接的奇幻之旅

介绍

在网络世界中，突然间失去了所有的连接。作为勇敢的冒险者，你将踏上一段惊险刺激的旅程，穿越充满谜题和挑战的网络景观，与神秘的网络幽灵对抗，解开断网之谜，找回失去的连接，带领人们重返数字世界。准备好迎接这场奇幻之旅吗？

准备

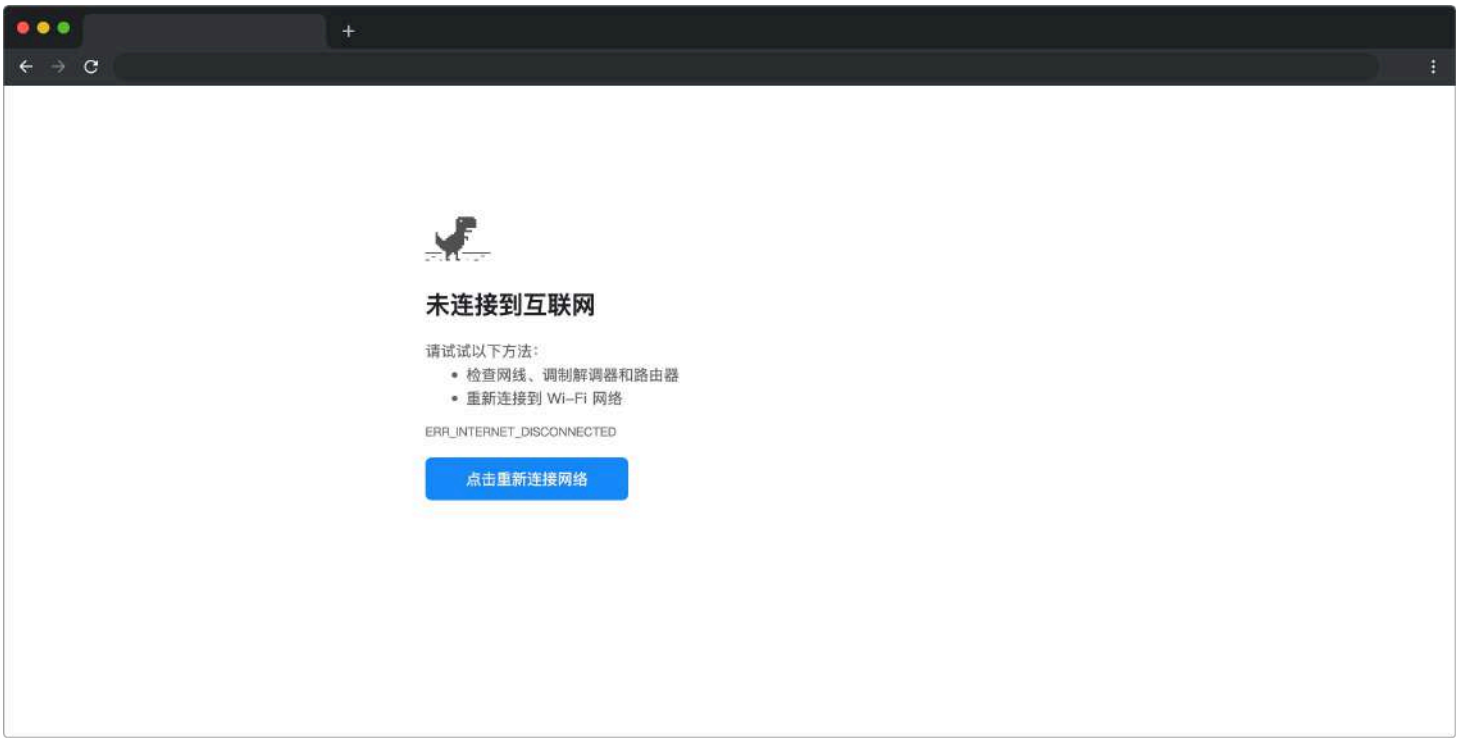
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ css
├─ images
├─ index.html
├─ effect.gif
└─ js
    └─ index.js
```

其中：

- `index.html` 是主页面。
- `css` 是样式文件夹。
- `images` 是图片文件夹。
- `effect.gif` 是项目最终完成的效果图。
- `js/index.js` 是待补充的 js 文件。

在浏览器中预览 `index.html` 页面效果如下：



目标

请在 `js/index.js` 文件中补充 `resetableOnce` 函数，实现在接收相同的函数时只执行一次。

函数返回值说明：

`resetableOnce` 函数的返回值为一个对象，格式为：`{runOnce:function, reset:function}`，对应说明如下：

- `runOnce`：一个函数，用于执行包装后的函数 `fn`。当第一次调用 `runOnce` 时，它将执行 `fn` 函数，并将结果保存，之后的每次调用将直接返回之前计算的结果。注意：**如果传入的函数（`fn`）不是同一个函数，则 `resetableOnce` 函数重新执行。**
- `reset`：一个函数，用于重置包装后的函数的状态。调用 `reset` 后可以让下一次调用 `runOnce` 时，再次执行 `fn` 函数。

函数使用示例如下：

```
// 测试用例 1: resetableOnce (fn)  fn 有参数的情况
// 定义一个加法函数
function addNumbers(a, b) {
    return a + b;
}

const test1 = resetableOnce(addNumbers);
console.log(test1.runOnce(2, 3)); // 输出: 5
console.log(test1.runOnce(4, 5)); // 第二次调用不会执行加法操作, 返回上次执行的值, 输出: 5

// 重置 once
test1.reset();
console.log(test1.runOnce(4, 5)); // 因为重置, addNumbers 再次执行, 输出: 9

// 测试用例 2: resetableOnce (fn)  fn 无参数的情况
let a = 10;
let fn = () => {
    return a = a + 1;
};
const test2 = resetableOnce(fn); // 传入了不同的函数, 之前的 once 不针对此函数生效。
console.log(test2.runOnce()); //输出 11
console.log(test2.runOnce()); // 输出 11
test2.reset();
console.log(test2.runOnce()); // 输出 12
```

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, 以免造成无法判题通过。

判分标准

- 传入同一函数满足需求能正确运行, 得 5 分。
- 本题完全实现题目目标得满分。

tree 命令助手

介绍

"tree 命令助手"是一个轻量级工具，用于生成树形结构的目录列表。它以直观的方式展示文件系统中的目录和文件层级关系，帮助用户更好地理解和浏览目录结构。通过简单的操作，您可以快速生成清晰的层级视图，从而更方便的导航和管理文件系统。

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：

```
├─ app.js
├─ css
│   └─ index.css
│   └─ lib
│       └─ a.css
│       └─ style.css
├─ index.html
└─ js
    └─ demo.js
    └─ index.js
```

其中：

- `app.js` 是待补充代码的 `js` 文件。
- 其他文件均为用来读取的普通文件（请勿新增或删除）。

目标

请在 `app.js` 文件中的 `TODO` 部分，完善 `generateTree` 函数，实现生成文件树的功能。

`generateTree` 函数接收一个文件夹路径（`dirPath`）作为参数，返回值为包含该文件夹生成的文件树对象的数组，每个对象包含一个文件或文件夹的名称，以及其对应的子文件树（如果是文件夹）。

函数返回值数据结构示例如下：

```
[
  { name: 'app.js' }, // 因为不是文件夹，没有 children
  { name: 'js', children: [{name:"demo.js"},{name:"index.js"}] } // 只有一层文件夹
  { name: 'css', children: [{name:"index.css"},{name:"lib",children:[{name:'a.css'}]}]}
  // 多层文件夹，子文件树递归显示
]
```

提示： `fs.statSync(filePath).isDirectory()` 根据给定的 `filePath` 路径返回一个布尔值，表示该路径是否为目录。 `fs.readdirSync(dirPath)` 是读取 `dirPath` 目录下的文件和子目录，并以数组的形式返回它们的名称列表。

完成后，运行 `node app.js`，终端输出的内容效果如下：

```
├─ app.js
├─ css
│   ├─ index.css
│   ├─ lib
│   │   └─ a.css
│   └─ style.css
├─ index.html
└─ js
    ├─ demo.js
    └─ index.js
```

请注意以上目录结构仅供参考，实际判题时会修改目录结构，请保证代码的通用性。

规定

- 请勿修改 `app.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。

判分标准

- 本题完全实现题目目标得满分，否则得 0 分。

Github 明星项目统计

介绍

Github 的 star 数量代表了一个项目的受欢迎程度，star 数较高的项目通常有很多值得我们学习的地方。利用图表的形式将所有项目根据其 star 数量做统计，能让人们更清晰地看出当前的各种编程语言和开源项目的流行趋势。

本题请实现一个可以对项目语言和排名进行筛选查看的 Github 明星项目统计图表。

准备

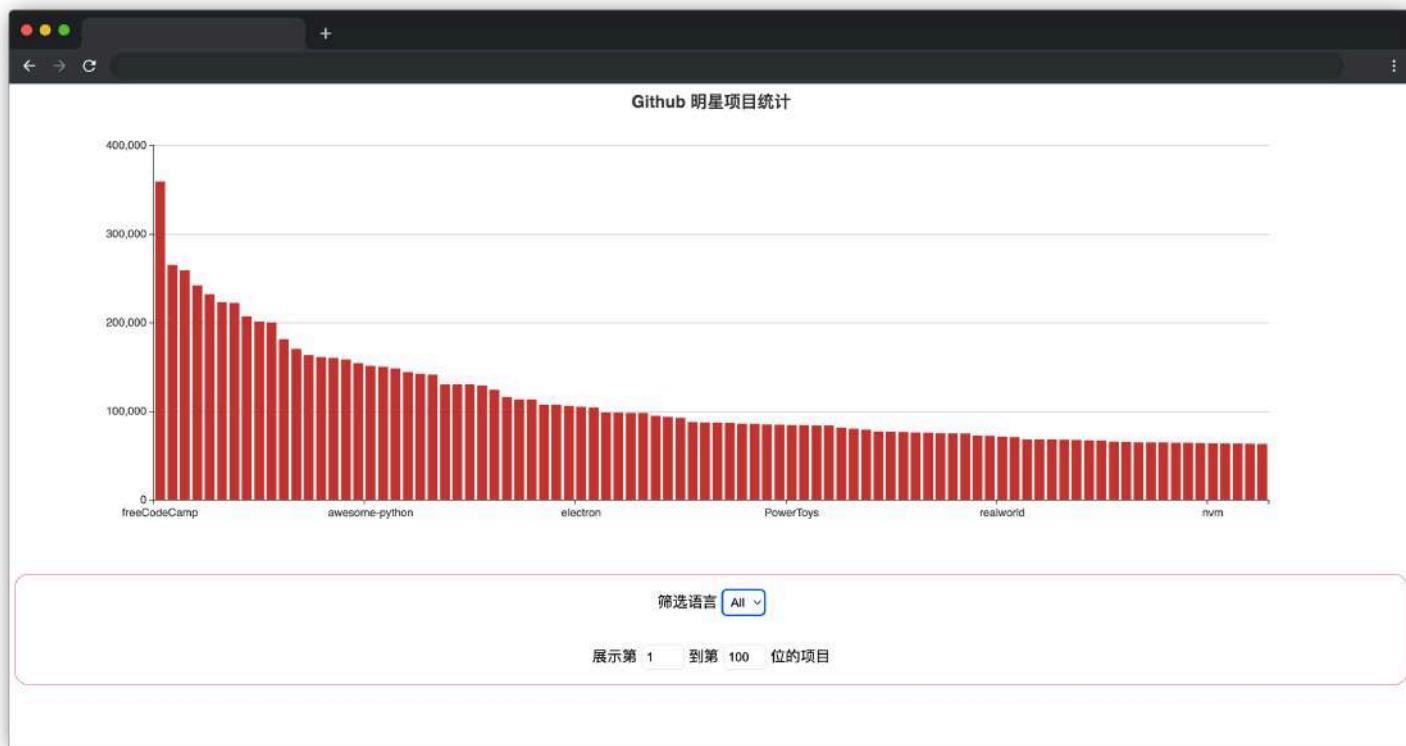
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
|— css
|— index.html
|— effect.gif
|— js
|   └─ data.json
└─ lib
```

其中：

- `css` 是样式文件夹。
- `index.html` 是主页面。
- `lib` 是依赖文件夹。
- `js/data.json` 是请求需要用到的项目数据。
- `effect.gif` 是项目最终完成的效果图。

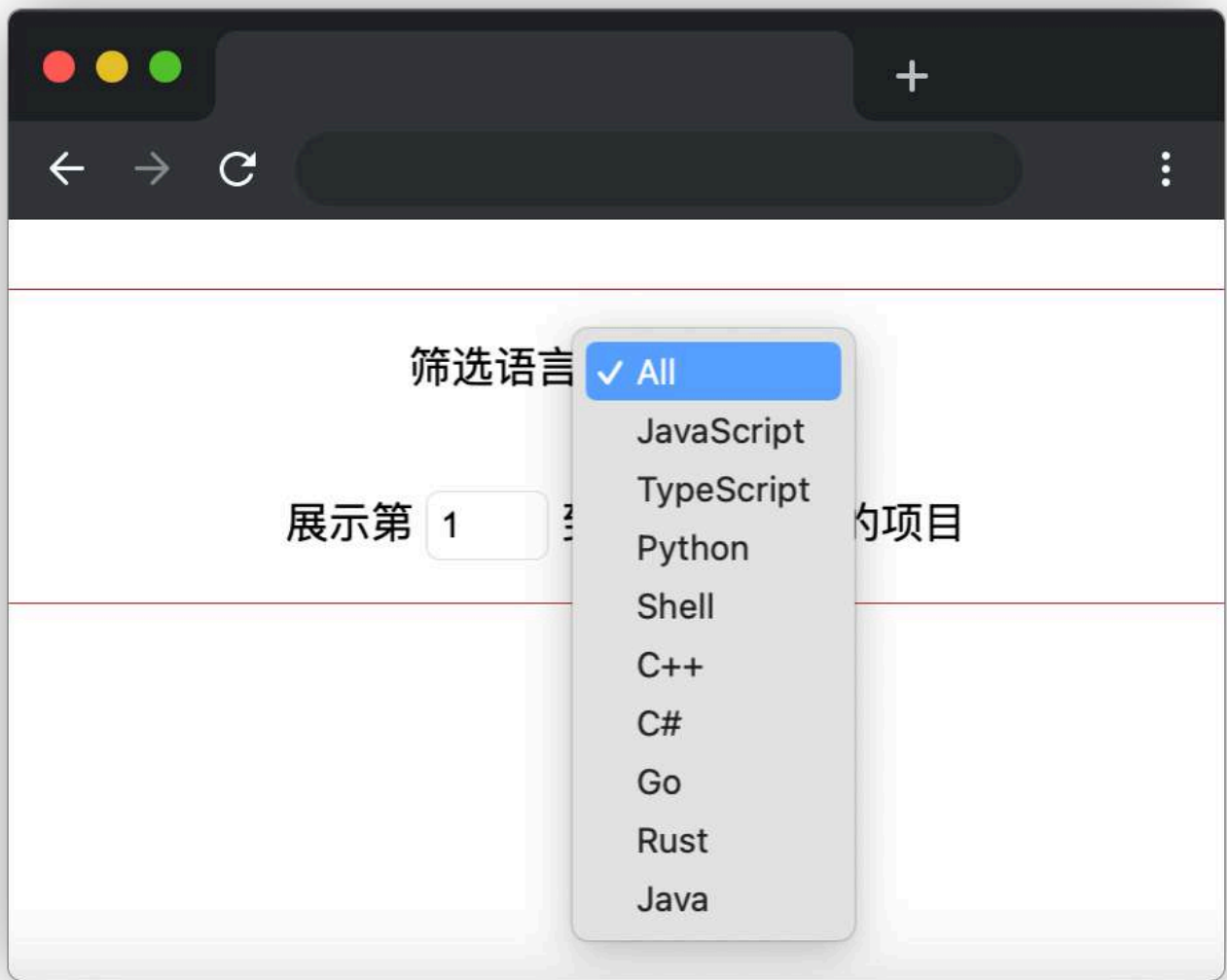
在浏览器中预览 `index.html` 页面效果如下：



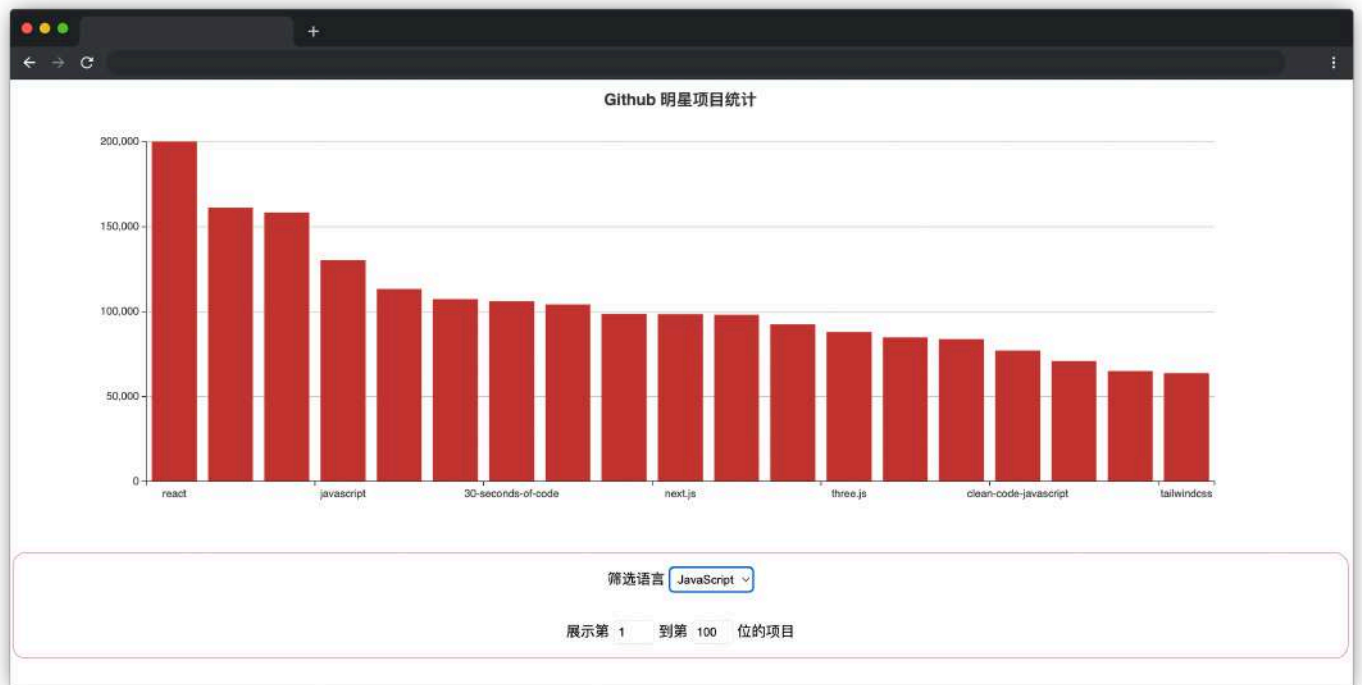
目标

请在 `index.html` 文件中补全代码，具体要求如下：

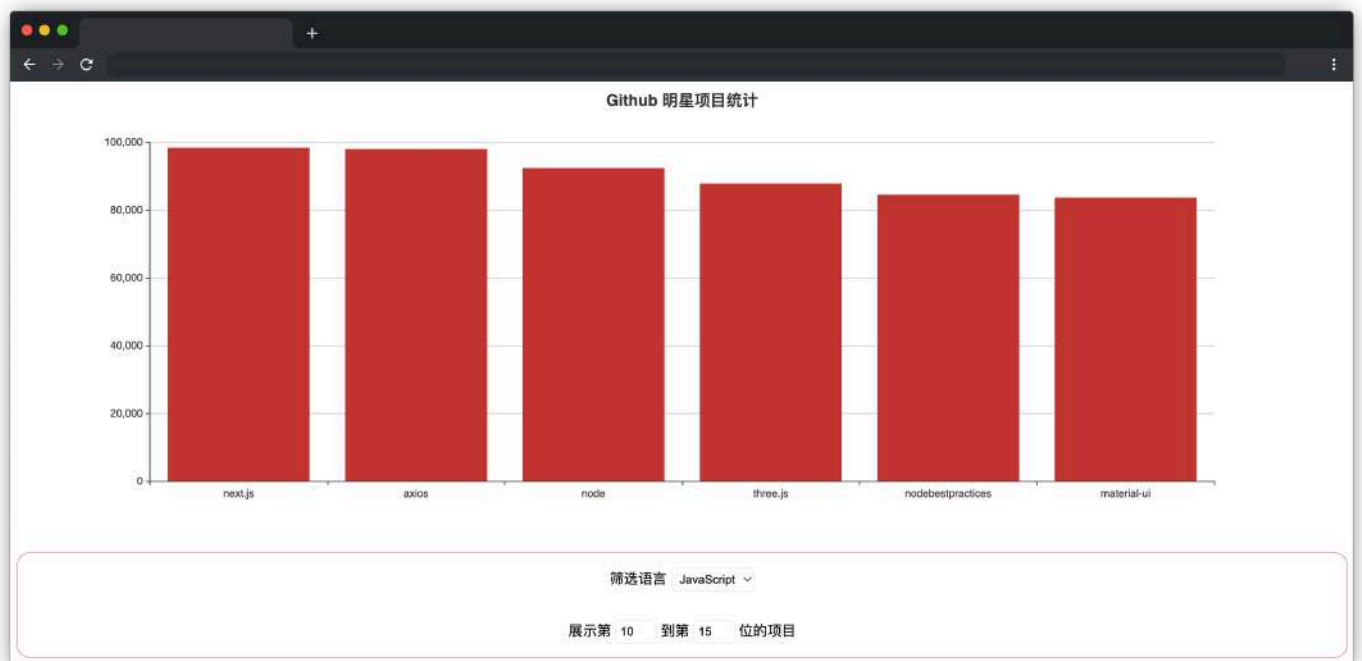
1. 将可供筛选的语言列表 `languages` 在 `select` 标签下的 `option` 元素进行渲染，`option` 的 `value` 值必须对应绑定 `languages` 数组中的值。效果如下：



2. 完善 `changeHandle` 函数，当用户选择语言和输入显示位次的时候都会调用此函数。当用户改变 `select` 筛选器的选项或改变展示项目的位次输入时，根据用户的**选择**和**输入的位次**重新渲染图表。注意：**重新渲染图表必须通过修改 `xData` 和 `yData` 的值进行，不要修改变量名称。**
- 如果用户选中的是 `All`，则展示所有的项目。
 - 如果用户选中了某个具体的语言，则将图表数据改为只显示该语言的项目。以选择 `JavaScript` 语言为例，效果如下：



- 注意：选择语言和展示位次是结合使用的，例如用户同时选择了 JavaScript 和展示 10 到 15 位的数据，则此时图表对应的数据是 使用 JavaScript 语言的项目中 star 数处于 10 到 15 位的项目（注意包含第 10 和 15 位）。效果如下：



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，以免造成无法判题通过。

判分标准

- 完成目标 1，得 7 分。
- 完成目标 2 中语言选择的切换，得 8 分
- 完成目标 2 语言切换的基础上完成显示位次的切换，得 5 分

小蓝驿站

介绍

欢迎使用小蓝驿站邮箱应用！这是一个简单易用的邮箱工具，除了提供高效的收发邮件功能，还内置了便捷的联系人管理。快来体验小蓝驿站，让联系人管理变得简单又有效。

准备

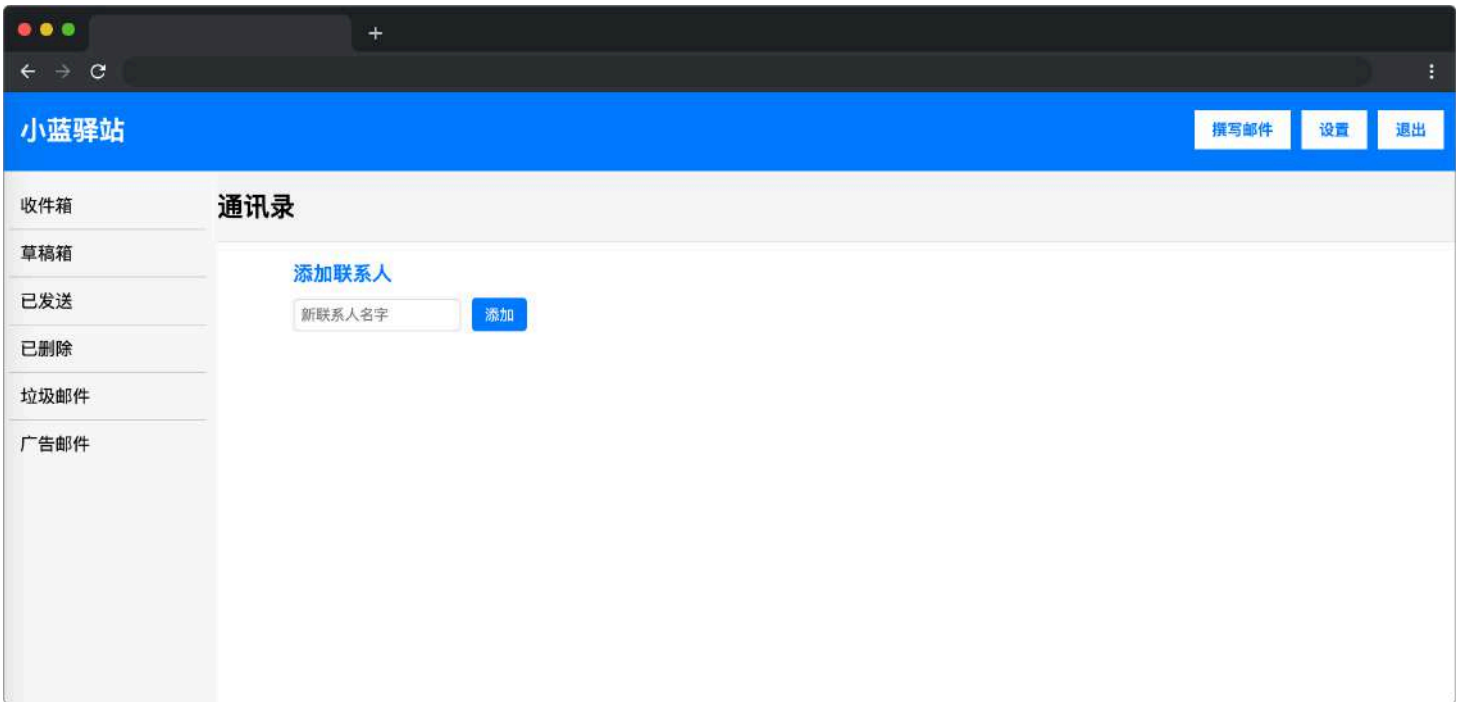
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ css
├─ data.json
├─ effect.gif
├─ index.html
└─ lib
```

其中：

- `css` 是样式文件夹。
- `data.json` 是数据文件。
- `lib` 是项目依赖文件夹。
- `index.html` 是主页面。
- `effect.gif` 是目标 2 完成后的效果图。

在浏览器中预览 `index.html` 页面效果如下：



目标

完善 `index.html` 文件中的 `TODO` 部分的代码，实现以下目标：

本题目只考察联系人的名字为英文的情况。

1. `list` 为联系人列表，`sortedContacts` 是将联系人列表 `list` 按照字母 A-Z 排序的数据，数据中的 `letter` 为**分组名称**（大写），数据中的 `contacts` 为当前分组的联系人数组，数据结构如下：

```
[
  {
    "letter": "A", // 分组名称，大写
    "contacts": [ // 当前分组联系人数组
      { "name": "Alice Adams" }, // 联系人名字，不区分大小写
      { "name": "Alex Anderson" }
    ] },
  {
    "letter": "B",
    "contacts": [...]
  }
]
```

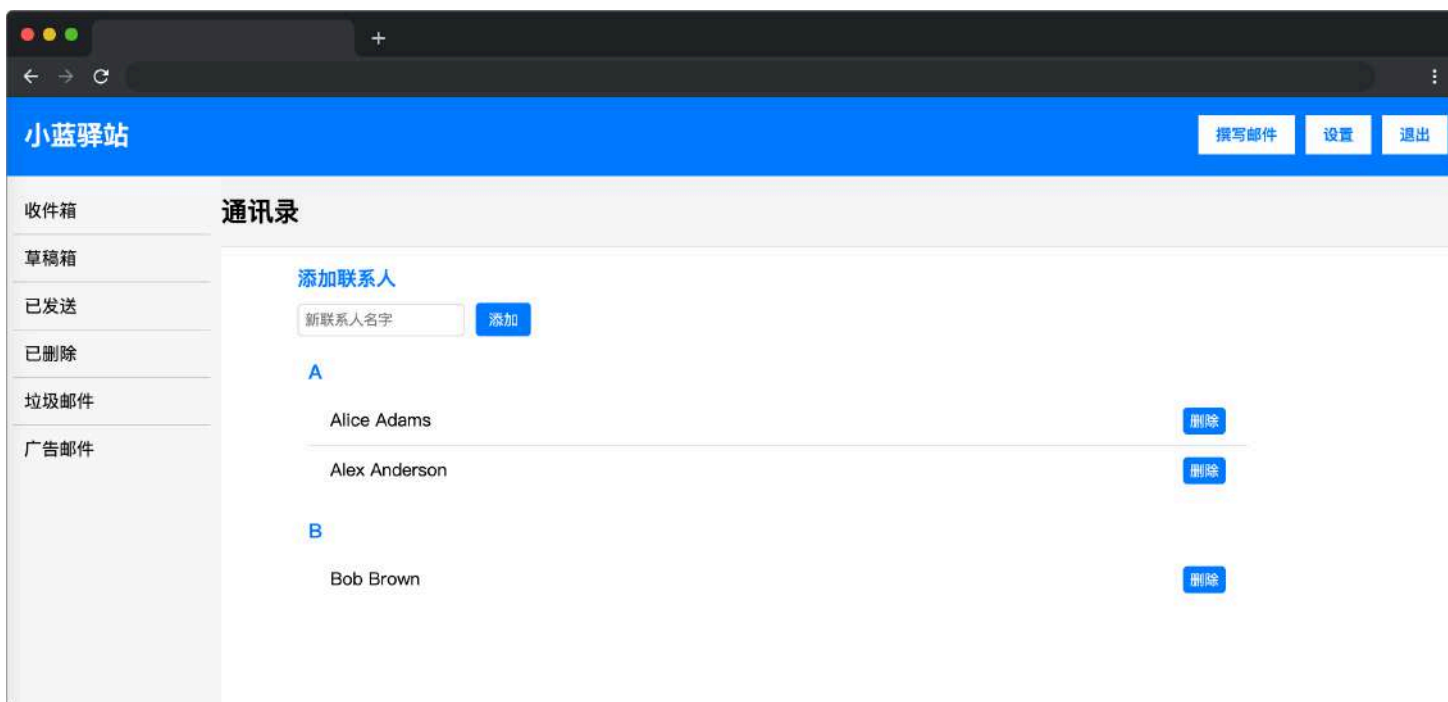
将排序后联系人数据 `sortedContacts` 按照下面示例的 DOM 结构，完成渲染。**注意：** `sortedContacts` 为动态数据，请注意代码的通用性。

```

<ul class="contacts-list">
  <!-- 以 A 为例 start -->
  <li class="contacts-group">    <!-- 字母分组渲染 DOM 结构-->
    <div class="contacts-group-title">A</div>    <!-- 分组的 字母名称 -->
    <ul>
      <li class="contact-item"> <!-- contact-item 人名渲染 dom 结构-->
        <span class="contact-name">Alice Adams</span>
        <button class="del-contact-button">删除</button>
      </li>
      <li class="contact-item"> <!-- contact-item 人名渲染 dom 结构-->
        <span class="contact-name">Alex Anderson</span>
        <button class="del-contact-button">删除</button>
      </li>
    </ul>
  </li>
  <!-- 以 A 为例 end -->
  <!-- ....省略代码 B、C、D、E ..... -->
</ul>

```

目标 1 完成后效果如下：



- 完善 `addContact` 函数，完成联系人添加功能。如果**分组名称**已存在，则直接添加联系人到该分组；如果**分组名称**不存在，则新建一个**分组名称**和联系人添加到联系人列表中。用户数输入的可能是小写字母，如 `marissa`，注意**分组名称**应该为大写 `M`，用户名字不做处理，在列表中仍然显示 `marissa`。

目标 2 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，以免造成无法判题通过。

判分标准

- 完成目标 1 得 10 分。
- 完成目标 1 的基础上，完成目标 2，得 10 分。

商品浏览足迹

介绍

商品浏览足迹的记录是一种电子商务网站或应用程序提供的功能，旨在记录用户在网站上浏览过的商品。当用户登录其帐户时，他们可以访问该页面，查看他们最近浏览的商品，以便随时可以返回对比、评估和购买。

本题会提供一组用户的商品浏览记录，需要你按照要求将其处理成所需的数据结构并返回。

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：

```
|— css
|— images
|— index.html
└─ js
    |— axios.min.js
    |— data.json
    └─ index.js
```

其中：

- `css` 是样式文件夹。
- `images` 是图片文件夹。
- `index.html` 是主页面。
- `js/axios.min.js` 是 `axios` 文件。
- `js/index.js` 是待补充的 `js` 文件。
- `js/data.json` 是页面用到的数据文件。

在浏览器中预览 `index.html` 页面效果如下：



目标

完善 `js/index.js` 文件中的 `TODO` 部分，实现以下目标：

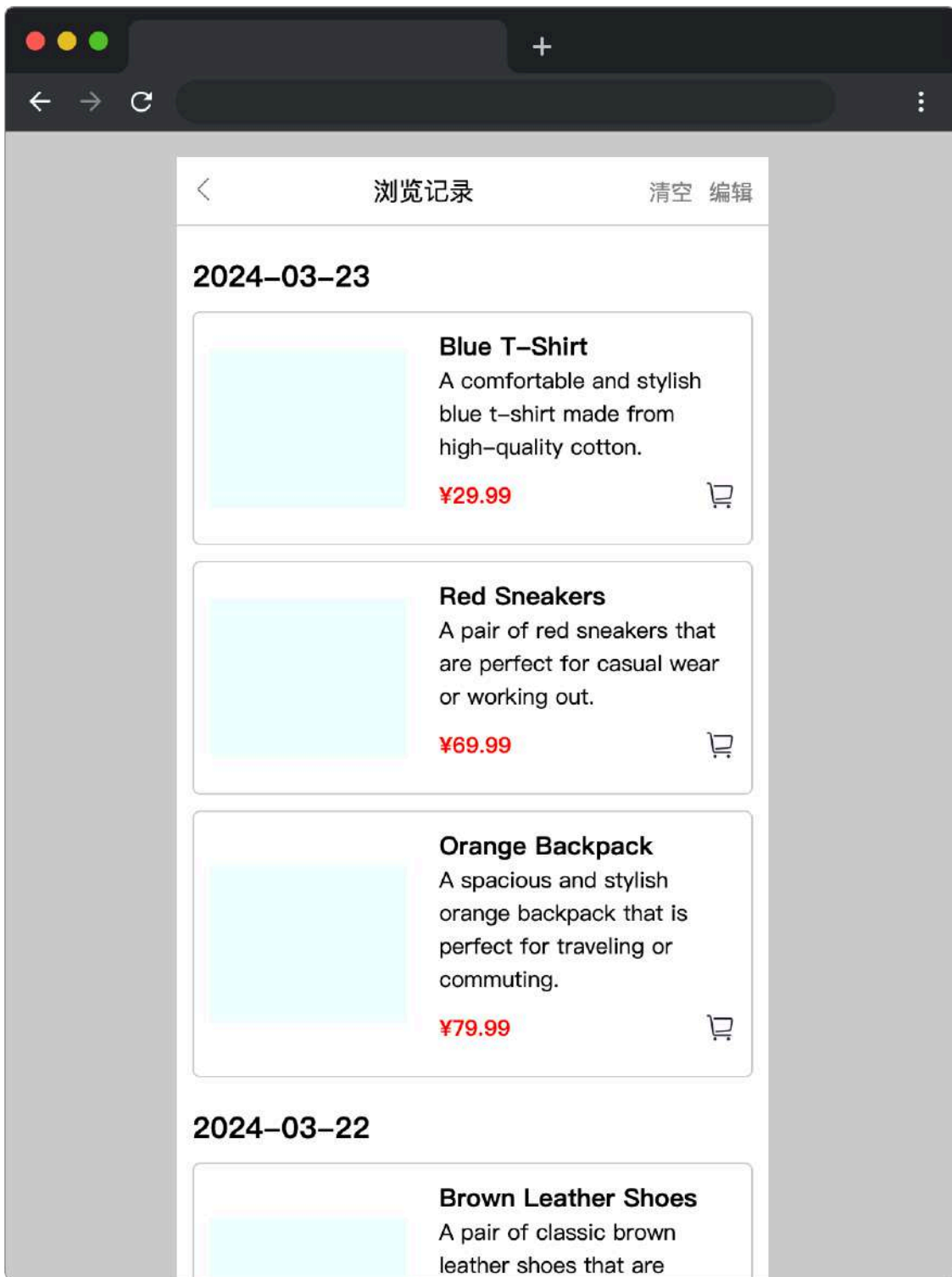
1. 在 `window.onload` 中完成数据请求（请求地址必须使用提供的常量 **MockUrl**），将请求回来的数据赋值给 `data`，并正确传递给已提供的 `getData` 函数。
2. 补充函数 `removeDuplicates`，将同一天浏览的相同商品（即字段 `id` 相同）去重并作为数组返回，数据结构与 `json` 文件中数据结构相同。

3. 补充函数 `sortByDate`，参数 `defaultData` 是函数 `removeDuplicates` 去重后的数据，根据字段 `viewed_on`（格式化为 `YYYY-MM-DD`）降序排序并作为数组返回，数据结构与 `json` 文件中数据结构相同。
4. 补充函数 `transformStructure`，将去重排序后的所有商品数据，作为一个对象存储并返回，该对象的所有 `key` 为浏览商品的当天日期（即，字段 `viewed_on` 格式化为 `YYYY-MM-DD`，月份和日期小于 10 的前面补 0），`value` 为当天浏览的所有商品列表（以数组形式表现）。

最终返回的对象数据结构如下所示：

```
{
  "2024-03-03": [
    {
      "id": "001",
      "name": "商品1",
      "description": "商品1的描述。",
      "price": 59.99,
      "viewed_on": "2024-03-22T20:02:00"
    },
    {
      "id": "002",
      "name": "商品2",
      ...
    }
  ],
  "2024-03-01": [
    {
      "id": "001",
      "name": "商品1",
      ...
    },
    {
      "id": "002",
      "name": "商品2",
      ...
    }
  ],
}
```

最终效果如下所示：



规定

- 函数 `removeDuplicates`、`sortByDate`、`transformStructure` 检测时使用的输入数据与题目中给出的示例数据可能是不同的。考生的程序必须是通用的，不能只对需求中给定的数据有效。
- 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。

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 5 分。
- 完成目标 4，得 5 分。

NPM Download Simulator

介绍

NPM Download Simulator 模拟了 npm 包下载的过程。用户可以点击 "下载 (Start Download) " 按钮开始下载，并从多个下载源中选择最快的源进行下载，模拟了实际下载的过程，包括下载进度和成功或失败的结果。

准备

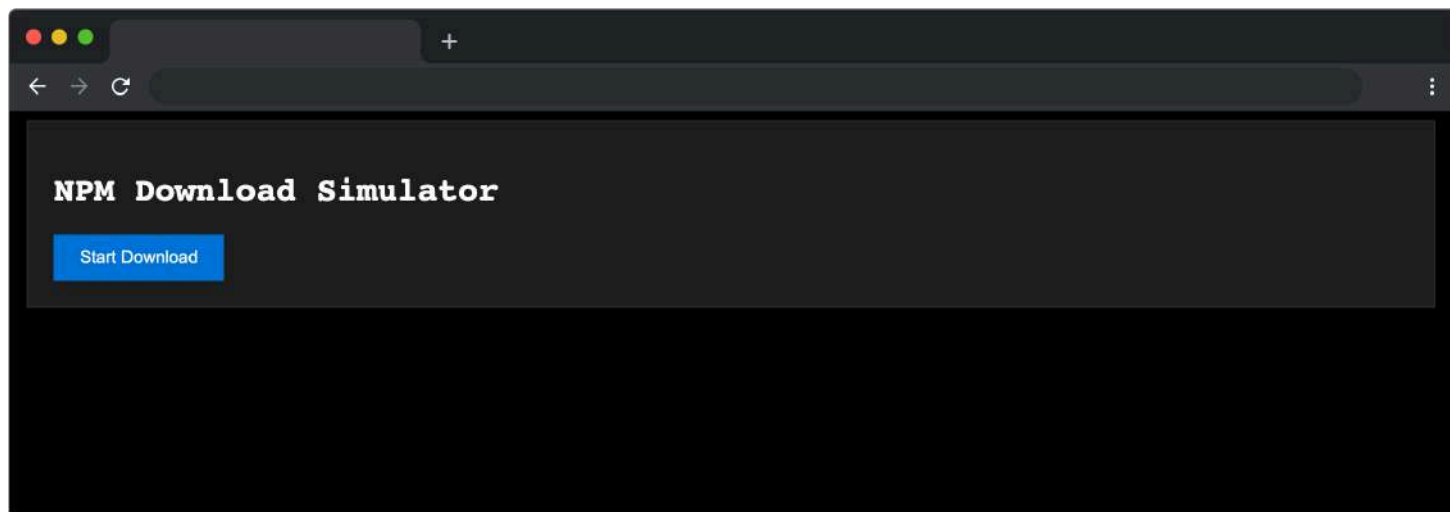
开始答题前，需要先打开本题的项目代码文件夹，目录结构如下：

```
├─ css
├─ index.html
├─ js
  └─ index.js
```

其中：

- css 是样式文件夹。
- index.html 是主页面。
- js/index.js 是待补充代码的 js 文件。

在浏览器中预览 index.html 页面效果如下：



目标

找到 `js/index.js` 文件中的 `myRace` 函数，在不使用 `Promise.race` 的情况下，完善其中的 `TODO` 部分，目标如下：

`myRace` 接收的参数应为可迭代对象，可迭代对象包含数组（`Array`）、字符串（`String`）、`Set`、`Map`。

1. 如果不是可迭代对象则函数应该根据参数类型，返回一个拒绝的 `Promise`，并抛出 `TypeError` 错误。格式为：数据类型 `is not iterable`。

示例：

```
// 1 执行代码
myRace(1)
// 控制台报错输出
Uncaught (in promise) TypeError: number is not iterable
```

```
//2 执行代码
myRace(null)
// 控制台报错输出
Uncaught (in promise) TypeError: null is not iterable
```

```
//3 执行代码
myRace(true)
// 控制台报错输出
Uncaught (in promise) TypeError: boolean is not iterable
```

2. 参数如果是可迭代对象，则 `myRace` 函数返回一个新的 `Promise` 对象，该对象代表了传入的可迭代对象中最先解决或拒绝的 `Promise` 的状态。如果可迭代对象中包含非 `Promise` 的普通值（例如字符串、数字等），那么返回的 `Promise` 会立即解决，并使用该普通值作为解决值。

示例：

```
// 测试用例 1
myRace([1,2]) //返回 Promise {<fulfilled>: 1}

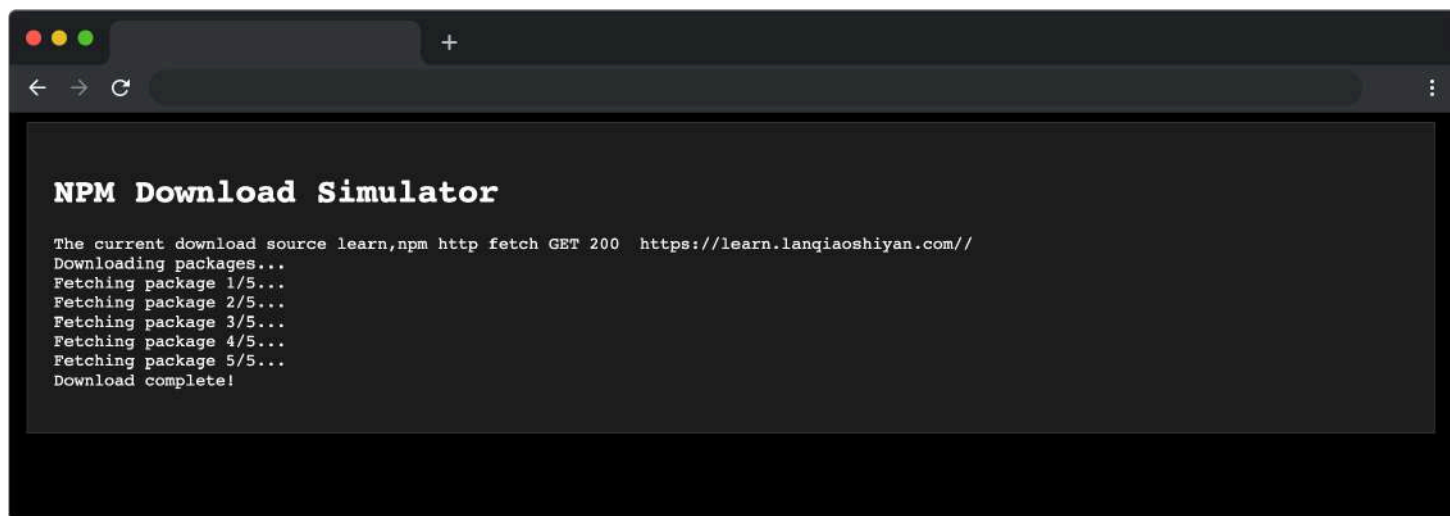
// 测试用例 2
myRace('hello') // 返回 Promise {<fulfilled>: 'h'}
```



```
// 测试用例 3
let promise1 = new Promise((resolve, reject)=> {
  reject(1)
})
let promise2 = new Promise((resolve, reject) =>{
  setTimeout(()=>{
    resolve(2)
  },1000)
})

myRace([promise1, promise2]).then(res => {
  console.log(res)
},reason=>{
  console.log(reason) // 输出 1
})
```

完成后，点击“Start Download”按钮，示例效果如下：



规定

- 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。

判分标准

- 完成目标 1，得 5 分。
- 完成目标 2，得 20 分。