web APIs 第四天

正则表达式

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 能够利用正则表达式校验输入信息的合法性
2. 具备利用正则表达式验证表单的能力

目录

Contents

正则表达式

- 基本使用
- 元字符
- 替换和修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.1.1 什么是正则表达式

- 正则表达式（Regular Expression）是一种字符串匹配的模式（规则）
- **使用场景：**
  - 例如验证表单：手机号表单要求用户只能输入11位的数字 (匹配)
  - 过滤掉页面内容中的一些敏感词(替换)，或从字符串中获取我们想要的特定部分(提取)等

1.1.2 正则基本使用

- 正则基本使用分为两步：

定义规则

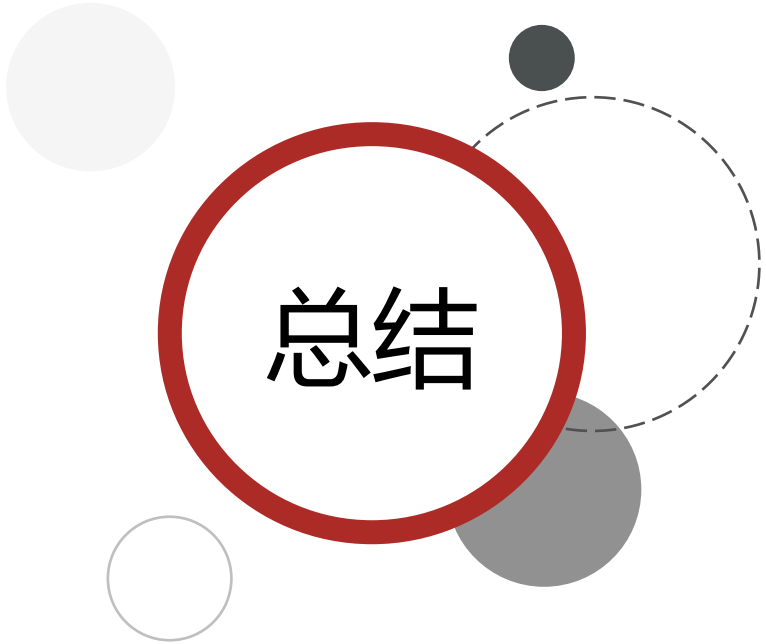
- const reg = /表达式/
- 其中 / / 是正则表达式字面量
- 正则表达式也是对象

```
const reg = /前端/
```

使用正则

- **test()** 方法 用来查看正则表达式与指定的字符串是否匹配
- 如果正则表达式与指定的字符串匹配，返回true，否则false

```
// 要检测的字符串
const str = 'IT培训,前端开发培训,IT培训课程,web前端培训,Java培训,人工智能培训'
// 1.定义正则表达式，检测规则
const reg = /前端/
// 2.检测方法
console.log(reg.test(str)) // true
```



1.正则表达式是什么?

- 是一种字符串匹配的模式（规则）

2.正则表达式有什么作用?

- 表单验证（**匹配**）
- 过滤敏感词（**替换**）
- 字符串中提取我们想要的部分（**提取**）

3.正则表达式使用分为几步?

- 定义规则
- 使用正则 test() 方法

```
// 正则表达式的基本使用
const str = 'web前端开发'
// 1. 定义规则
const reg = /web/

// 2. 使用正则 test()
console.log(reg.test(str))  // true  如果符合规则匹配上则返回true
console.log(reg.test('java开发'))  // false  如果不符合规则匹配上则返回
```

正则表达式

- 基本使用
- 元字符
- 替换和修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.2 元字符

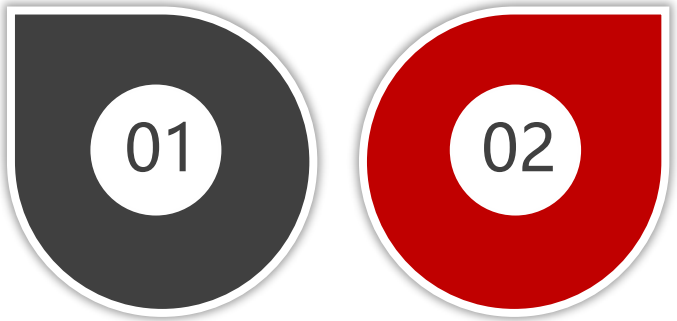
- **普通字符:**
  - 大多数的字符仅能够描述它们本身，这些字符称作普通字符，例如所有的字母和数字。
  - 普通字符只能够匹配字符串中与它们相同的字符。
  - 比如，规定用户只能输入英文26个英文字母，普通字符的话 /[abcdefghijklmnopqrstuvwxyz]/
- **元字符(特殊字符)**
  - 是一些具有特殊含义的字符，可以极大提高了灵活性和强大的匹配功能。
  - 比如，规定用户只能输入英文26个英文字母，换成元字符写法： /[a-z]/

1.2 元字符

- 为了方便记忆和学习，我们对众多的元字符进行了分类：

边界符

定义位置规则，必须用什么开头，用什么结尾

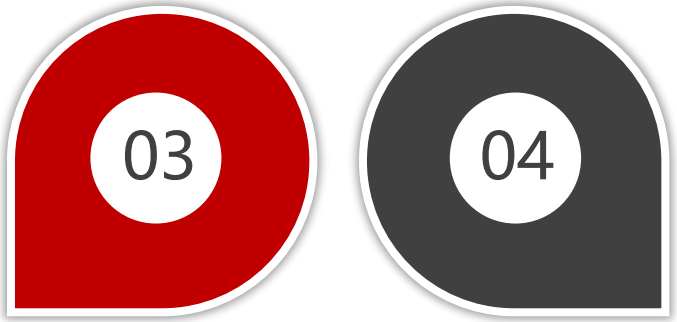


量词

定义重复次数规则

范围

表示字符的范围



字符类

区分各种字符，例如区分字母和数字

1.2 元字符

1. 边界符

- 正则表达式中的边界符（位置符）用来提示字符所处的位置，主要有两个字符

边界符	说明
^	表示匹配行首的文本(以谁开始)
$	表示匹配行尾的文本(以谁结束)

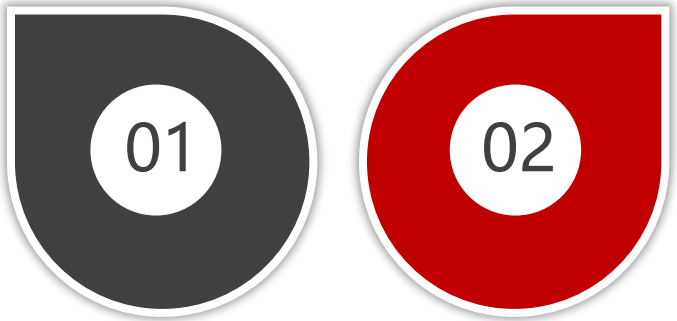
如果 ^ 和 $ 在一起，表示必须是精确匹配

1.2 元字符

- 为了方便记忆和学习，我们对众多的元字符进行了分类：

边界符

定义位置规则，必须用什么开头，用什么结尾

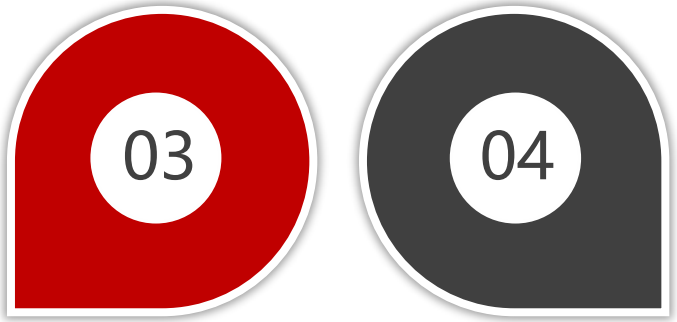


量词

定义重复次数规则

范围

表示字符的范围



字符类

区分各种字符，例如区分字母和数字

1.2 元字符

2. 量词

- 量词用来设定某个模式重复次数

量词	说明
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

注意： 逗号左右两侧千万不要出现空格

1.2 元字符

2. 量词

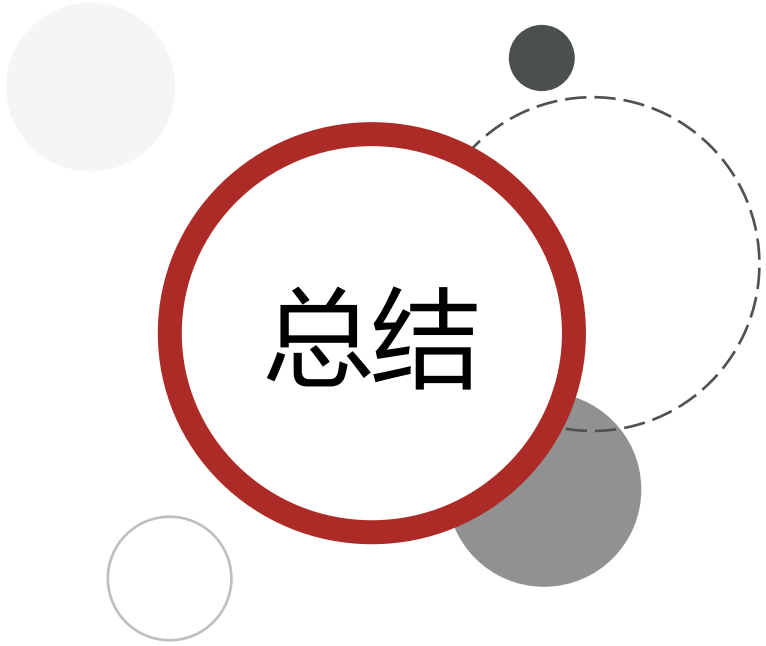
- 量词用来 设定某个模式出现的次数

```
// * 表示重复0次或者更多次
console.log(/^哈*$/.test(''))  // true
console.log(/^哈*$/.test('哈'))  // true
console.log(/^哈*$/.test('哈哈哈'))  // true
// + 表示重复1次或者更多次
console.log(/^哈+$/.test(''))  // false
console.log(/^哈+$/.test('哈'))  // true
console.log(/^哈+$/.test('哈哈哈'))  // true
// ？ 表示重复0次或者1次    0 || 1
console.log('------')
console.log(/^哈?$/.test(''))  // true
console.log(/^哈?$/.test('哈'))  // true
console.log(/^哈?$/.test('哈哈哈'))  // fasle
```

```
console.log(/^哈{2}$/.test(''))  // false
console.log(/^哈{2}$/.test('哈'))  // false
console.log(/^哈{2}$/.test('哈哈'))  // true
console.log(/^哈{2}$/.test('哈哈哈'))  // false
```

```
//  {n,}   是 >= n 的意思
console.log(/^哈{2,}$/.test(''))  // false
console.log(/^哈{2,}$/.test('哈'))  // false
console.log(/^哈{2,}$/.test('哈哈'))  // true
console.log(/^哈{2,}$/.test('哈哈哈'))  // true
```

```
//  {n,m}   是 次数 >= n  <= m 的意思 注意，逗号左右千万不要有空格
console.log(/^哈{2,4}$/.test(''))  // false
console.log(/^哈{2,4}$/.test('哈'))  // false
console.log(/^哈{2,4}$/.test('哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈哈哈'))  // false
```



1. 字符串 'p' 想要重复 3次以上怎么写?

- /^p{3,}$/

2. 字符串 'p' 想要重复6~18次之间怎么写?

- /^p{6,18}$/

3. 字符串 'p' 想要重复0次以上怎么写?

- /^p*$/ 或者 /^p{0,}$/

4. 字符串 'p' 想要至少出现一次怎么写?

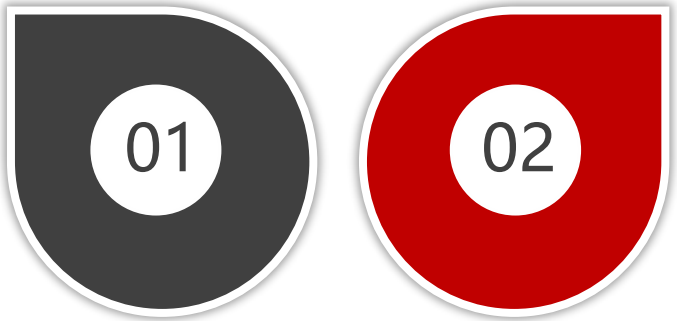
- /^p+$/或者 /^p{1,}$/

1.2 元字符

- 为了方便记忆和学习，我们对众多的元字符进行了分类：

边界符

定义位置规则，必须用什么开头，用什么结尾

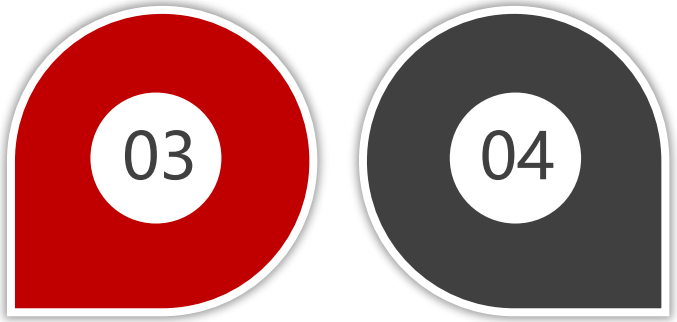


量词

定义重复次数规则

范围

表示字符的范围



字符类

区分各种字符，例如区分字母和数字

1.2 元字符

3. 范围：

表示字符的范围，定义的规则限定在某个范围，比如只能是英文字母，或者数字等等，用 [] 表示范围

[abc]	匹配包含的单个字符。也就是只有 a || b || c　这三个单字符返回true，　可以理解为多选1
[a-z]	连字符。来指定字符范围。[a-z] 表示 a 到 z 26个英文字母
[^abc]	取反符。[^a-z] 匹配除了小写字母以外的字符

1.2 元字符

3. 范围：

表示字符的分组和范围

- 使用连字符 - 表示一个范围

```
console.log(/^[a-z]$/.test('c'))  // true
```

- 比如：
  - [a-z] 表示 a 到 z 26个英文字母都可以
  - [a-zA-Z] 表示大小写都可以
  - [0-9] 表示 0~9 的数字都可以
- 认识下：

```
腾讯QQ号：  ^[1-9][0-9]{4,}$ (腾讯QQ号从10000开始)
```

1. 请说下代码代表什么意思？
   - [abc] 匹配abc其中的任何单个字符
   - [a-z] 匹配26个小写英文字母其中的任何单个字符
   - [^a-z] 匹配除了26个小写英文字母之外的其他任何单个字符

案例

用户名验证案例

需求：用户名要求用户英文字母,数字,下划线或者短横线组成，并且用户名长度为 6~16 位

分析：

①：首先准备好这种正则表达式模式 /^[a-zA-Z0-9-_]{6,16}$/

②：当表单失去焦点就开始验证.

③：如果符合正则规范, 则让后面的span标签添加 right 类

④：如果不符合正则规范, 则让后面的span标签添加 wrong 类，同时显示提示框

案例

用户名验证案例

需求：用户名要求用户英文字母,数字,下划线或者短横线组成，并且用户名长度为 6~16 位

拓展补充：

- blur 事件：当元素失去焦点时触发（不论表单元素的值是否发生改变）
- change 事件：元素失去焦点并且表单元素的值发生改变才会触发

该案例中，推荐使用 change 事件

练习

昵称案例（课堂作业）

需求：要求用户只能输入中文

分析：

①：首先准备好这种正则表达式模式　/^[\u4e00-\u9fa5]{2,8}$/

②：当表单失去焦点就开始验证.

③：如果符合正则规范, 则让后面的span标签添加 right 类

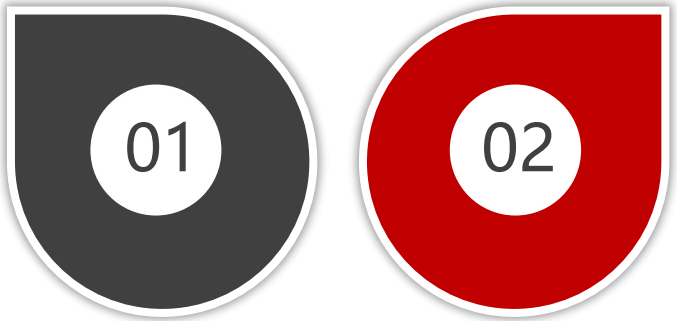
④：如果不符合正则规范, 则让后面的span标签添加 wrong 类，同时显示提示框（只能输入中文）

1.2 元字符

- 为了方便记忆和学习，我们对众多的元字符进行了分类：

边界符

定义位置规则，必须用什么开头，用什么结尾

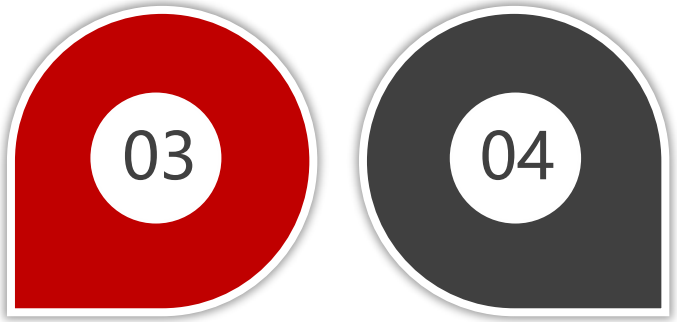


量词

定义重复次数规则

范围

表示字符的范围



字符类

区分各种字符，例如区分字母和数字

1.2 元字符

4. 字符类：

某些常见模式的简写方式，区分字母和数字

字符类	说明
\d	匹配0-9之间的任一数字，相当于[0-9]
\D	匹配所有0-9以外的字符，相当于 [^0-9]
\w	匹配任意的字母、数字和下划线，相当于[A-Za-z0-9_]
\W	除所有字母、数字和下划线以外的字符，相当于 [^A-Za-z0-9_]
\s	匹配空格（包括换行符、制表符、空格符等），相等于[\t\r\n\v\f]
\S	匹配非空格的字符，相当于 [^ \t\r\n\v\f]

```
日期格式： /^\d{4}-\d{1,2}-\d{1,2}$/
```

日期格式：2099-12-12

日期格式：2099-1-1

正则表达式

- 基本使用
- 元字符
- 替换和修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.3 替换和修饰符

- replace 替换方法，可以完成字符的替换
- **语法：**

```
字符串.replace(/正则表达式/, '替换的文本')
```

1.3 替换和修饰符

- 修饰符约束正则执行的某些细节行为，如是否区分大小写、是否支持多行匹配等
- 语法：

```
/表达式/修饰符
```

- i 是单词 ignore 的缩写，正则匹配时字母不区分大小写
- g 是单词 global 的缩写，匹配所有满足正则表达式的结果

```
console.log(/a/i.test('a'))  // true
console.log(/a/i.test('A'))  // true
```

案例

利用正则隐藏手机号中间四位

分析：

①：手机号前三位显示，中间四位隐藏为****， 后面四位显示

②：把手机号利用正则利用小括号划分为三部分

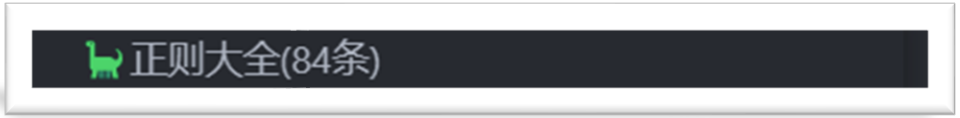
③：在replace中

$1 对应第一个小括号内容，$2 对应第2个括号内容，依次类推

④：字符串重复使用： 字符串.repeat(次数) 实现

推荐一个vscode正则插件

可以帮我们快速生成正则表达式 **any-rule**



目录

Contents

JS插件

- AlloyFinger 手势插件
- a11y-dialog 对话框插件

插件- AlloyFinger

插件: 就是别人写好的一些代码, 我们只需要复制对应的代码,就可以直接实现对应的效果

- AlloyFinger 是腾讯 AlloyTeam 团队开源的超轻量级 Web 手势插件，为元素注册各种手势事件
- **github地址**：https://github.com/AlloyTeam/AlloyFinger
- **使用步骤：**

1. 下载js库：http://alloyteam.github.io/AlloyFinger/alloy_finger.js
2. 将AlloyFinger库引入当前文件：<script src="alloy_finger.js"></script>

   或者使用在线地址：<script src="https://unpkg.com/alloyfinger@0.1.16/alloy_finger.js"></script>
3. 配置

```
new AlloyFinger(element, {  // element 是给哪个元素做滑动事件
  swipe: function (e) {
    // 滑动的时候要做的事情 e.direction 可以判断上下左右滑动 Left  Right 等
 }
})
```

拓展-M端事件

M端(移动端)有自己独特的地方。比如触屏事件 touch（也称触摸事件），Android 和 IOS 都有。

- touch 对象代表一个触摸点。触摸点可能是一根手指，也可能是一根触摸笔。触屏事件可响应用户手指（或触控笔）对屏幕或者触控板操作。
- 常见的触屏事件如下：

触屏touch事件	说明
touchstart	手指触摸到一个 DOM 元素时触发
touchmove	手指在一个 DOM 元素上滑动时触发
touchend	手指从一个 DOM 元素上移开时触发

JS插件

- AlloyFinger 手势插件
- a11y-dialog 对话框插件

插件 - a11y-dialog

- a11y-dialog 是一个轻量级、灵活的脚本，用于创建对话框窗口
- **github地址**：https://github.com/KittyGiraudel/a11y-dialog
- **使用步骤：**
  1. 下载js库： https://cdn.jsdelivr.net/npm/a11y-dialog@8/dist/a11y-dialog.min.js
  2. 将a11y-dialog库引入当前文件：<script src= “a11y-dialog.min.js"></script>
  3. 使用
     1. 准备 HTML 结构
     2. 准备 CSS 样式
     3. 准备 JS 实现对话框
     4. 具体细节内容可以查看 插件使用MD笔记

插件 – 使用经验

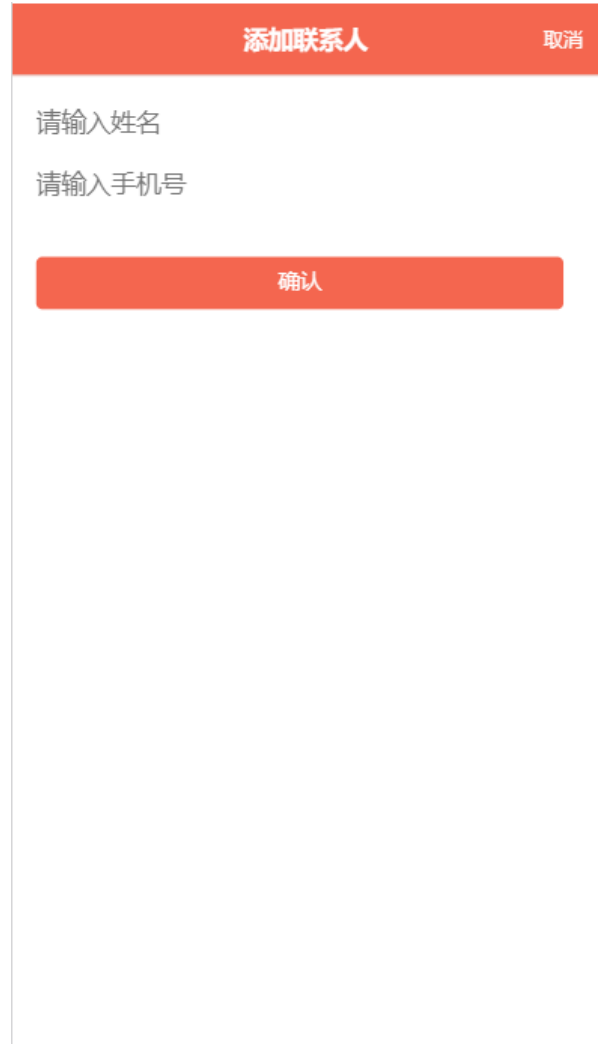
- **使用插件的思路**
  1. 明确插件的作用：通过官网了解
  2. 明确使用步骤：查看官网提供的基本使用步骤
  3. 做个Demo：按照官网的基本示例代码测试是否可以实现功能
  4. 应用到开发中
- **查找插件**
  - Github官网：https://github.com 按照关键词进行搜索
  - 按照条件标准选择插件 （功能需求、兼容性、官方文档、活跃度）
- 自主学习-其他常用插件
  - Sortable 拖拽排序插件
  - Swiper 触摸滑动插件

目录

Contents

案例 通讯录(移动端)


通讯录
周杰伦 13411112222
华 刘德华 13511112222
友 张学友 13711112222
鹏 岳云鹏 13911112222
巴 迪丽热巴 13911112222
添加联系人


添加联系人
取消
请输入姓名
请输入手机号
确认

步骤

通讯录(移动端)

- 渲染业务
- 滑动业务
- 新增业务
- 删除业务

步骤 通讯录(移动端)

渲染业务


```
// 初始化数据
const arr = [
  { name: "周杰伦", tel: "13411112222" },
  { name: "刘德华", tel: "13511112222" },
  { name: "张学友", tel: "13711112222" },
  { name: "岳云鹏", tel: "13911112222" },
]
```

核心思路：

①：使用 map方法遍历准备好的对象数组，得到HTML结构数组

②：利用 join() 方法，将HTML结构数组拼接得到字符串结构，渲染页面

- 注意封装渲染函数，后期要复用

③：字符串截取，取出姓名最后一个字作为头像

- 字符串.substring(起始索引号，[结束索引号])

```
<div class="address-book">
  <!-- 添加item 核心区域-->
  <div class="item">
    <p class="circle">华</p>
    <p class="name">刘德华</p>
    <p class="tel">13511112222</p>
    <a class="del" href="javascript:;" ">
      <i class=" iconfont icon-shanchutianchong"></i>
    </a>
  </div>
</div>
```

步骤

通讯录(移动端)

- 渲染业务
- 滑动业务
- 新增业务
- 删除业务

步骤 通讯录(移动端)

滑动业务

核心思路：

利用AlloyFinger手势插件来实现滑动效果

①：利用forEach遍历每个 item

②：遍历的同时给每个 item 添加滑动效果

- 如果是左侧滑动，则添加active类，则实现滑动效果，显示删除按钮
- 否则移除active这个类，则是隐藏删除按钮

③：需要封装一个函数 initSwipe ， 提供给渲染函数使用

步骤

通讯录(移动端)

- 渲染业务
- 滑动业务
- 新增业务
- 删除业务

步骤 通讯录(移动端)

说明：

本次案例，我们尽量减少dom操作，采取操作数据的形式

增加和删除都是针对于数组的操作，然后根据数组数据渲染页面

思路：


```
const arr = [
  { name: "周杰伦", tel: "13411112222" },
  { name: "刘德华", tel: "13511112222" },
  { name: "张学友", tel: "13711112222" },
  { name: "岳云鹏", tel: "13911112222" },
  { name: "迪丽热巴", tel: "13911112222" }
]
```

步骤 通讯录(移动端)

核心步骤：

①：点击 添加联系人按钮，先展示 对话框，点击取消会关闭对话框（插件已提供）

②：用户点击确认按钮，先进行表单校验

- 如果内容为空，则提示不能为空
- 如果姓名不符合中文，则提示 请输入2-4位的中文姓名
- 如果手机号不符合规则，则提示 请输入11位的手机号

④：如果用户输入正确，则把收集的表单数据生成对象，追加给 arr 数组

```
{ name: "xxx", tel: "13512342234" }
```

```
const arr = [
  { name: "周杰伦", tel: "13411112222" },
  { name: "刘德华", tel: "13511112222" },
  { name: "张学友", tel: "13711112222" },
  { name: "岳云鹏", tel: "13911112222" },
  { name: "迪丽热巴", tel: "13911112222" },
  { name: 'xxx', tel: '13512342234' }
]
```

⑤：重新渲染页面

⑥：当对话框关闭的时候，清空姓名和手机号内容

步骤

通讯录(移动端)

- 渲染业务
- 滑动业务
- 新增业务
- 删除业务

步骤 通讯录(移动端)

删除业务

说明：

本次案例，我们尽量减少dom操作，采取操作数据的形式

增加和删除都是针对于数组的操作，然后根据数组数据渲染页面

思路：

确定要删除的数据 → 数组删除对应数据 → 渲染页面

```
const arr = [
  { name: "周杰伦", tel: "13411112222" },
  { name: "刘德华", tel: "13511112222" },
  { name: "张学友", tel: "13711112222" },
  { name: "岳云鹏", tel: "13911112222" },
  { name: "迪丽热巴", tel: "13911112222" }
]
```

步骤 通讯录(移动端)

删除业务

核心步骤：

①： 因为新增了数据，此处采取事件委托的形式注册点击事件

- 新增元素无法直接注册事件，但是父级 book 盒子没有变化，所以给他注册点击事件

②：利用自定义属性得到当前点击元素的索引号，确定要删除第几条数据

心得总结：

事件委托的两个重要作用：

1. 减少了注册次数
2. 给新增元素注册事件

步骤 通讯录(移动端)

删除业务

核心思路：

③：删除数组中对应索引号的数据，然后重新渲染页面

- 删除方法使用 splice(起始索引号，[删除几个])

④：防止误删除，所以删除时候，可以添加confirm确认框，确认是否删除

传智教育旗下高端IT教育品牌