web APIs 第四天

日期对象、DOM节点、移动端滑动

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

学习目标

Learning Objectives

1. 理解节点(标签)的增删改查
2. 具备编写增加学生信息表案例的能力

01

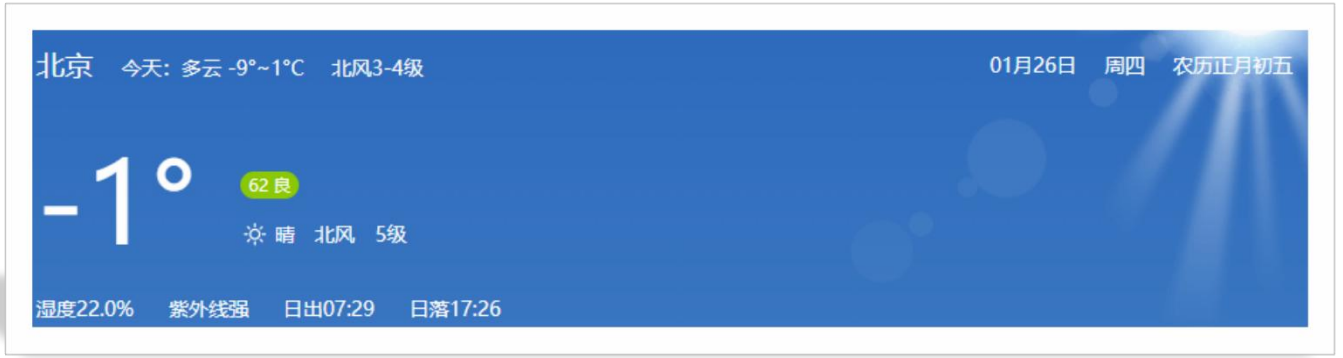
日期对象

- 实例化
- 时间对象方法
- 时间戳

1. 日期对象 Date

- 日期对象：用来表示日期和时间的对象
- 作用：可以得到当前系统日期和时间

Date是JavaScript内置对象



1.1 实例化

- 日期对象使用必须先实例化：创建一个日期对象并获取时间
- 在代码中发现了 new 关键字时，一般将这个操作称为**实例化**
  - 获得当前日期

```
const date = new Date()
```

  - 获得指定日期

```
const date = new Date('2099-9-9')
```

2.2 日期对象方法 - 格式化日期对象

使用场景：因为日期对象返回的数据我们不能直接使用，所以需要转换为实际开发中常用的格式（格式化日期对象）

方法	作用	说明
getFullYear()	获得年份	获取四位年份
getMonth()	获得月份	取值为 0 ~ 11
getDate()	获取月份中的每一天	不同月份取值也不相同
getDay()	获取星期	取值为 0 ~ 6
getHours()	获取小时	取值为 0 ~ 23
getMinutes()	获取分钟	取值为 0 ~ 59
getSeconds()	获取秒	取值为 0 ~ 59

总结

1. 实例化日期对象怎么写？
   - new Date()
2. 日期对象方法里面月份和星期有什么注意的？
   - 月份是0~11, 星期是 0~6

方法	作用	说明
getFullYear()	获得年份	获取四位年份
getMonth()	获得月份	取值为 0 ~ 11
getDate()	获取月份中的每一天	不同月份取值也不相同
getDay()	获取星期	取值为 0 ~ 6
getHours()	获取小时	取值为 0 ~ 23
getMinutes()	获取分钟	取值为 0 ~ 59
getSeconds()	获取秒	取值为 0 ~ 59

案例

页面显示自动变化的日期和时间案例

需求：将当前日期以：2088年8月8号 08:08:56 显示到页面中

分析：

①：封装一个函数 getDateTime，显示 2088年8月8号 08:08:56 格式

②：记得时间的数字要补0 成为 08:08:56 格式

③：字符串拼接后，通过 innerText 给 div标签显示到页面中，并添加定时器效果

2.2 日期对象方法

格式化日期对象另外一种方法

方法	作用	说明
toLocaleString()	返回该日期对象的字符串（包含日期和时间）	2099/9/20 18:30:43
toLocaleDateString()	返回日期对象日期部分的字符串	2099/9/20
toLocaleTimeString()	返回日期对象时间部分的字符串	18:30:43

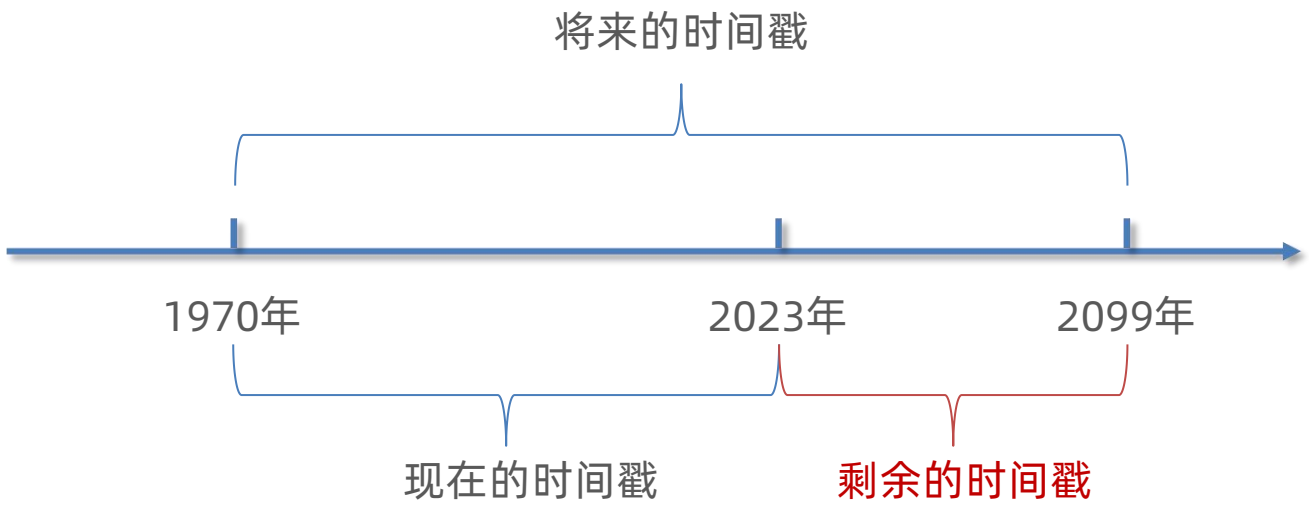
01

日期对象

- 实例化
- 日期对象方法
- 时间戳

2.3 时间戳

- **什么是时间戳：**
  - 是指1970年01月01日00时00分00秒起至现在的总毫秒数(数字型)，它是一种特殊的计量时间的方式
- **使用场景：** 计算倒计时效果，需要借助于时间戳完成
- **算法：**
- 将来的时间戳 - 现在的时间戳 = 剩余时间毫秒数
- 剩余时间毫秒数转换为年月日时分秒 就是倒计时时间



2.3 时间戳

三种方式获取时间戳：

getTime()

需要实例化

```
const date = new Date()
console.log(date.getTime())
```

+new Date()

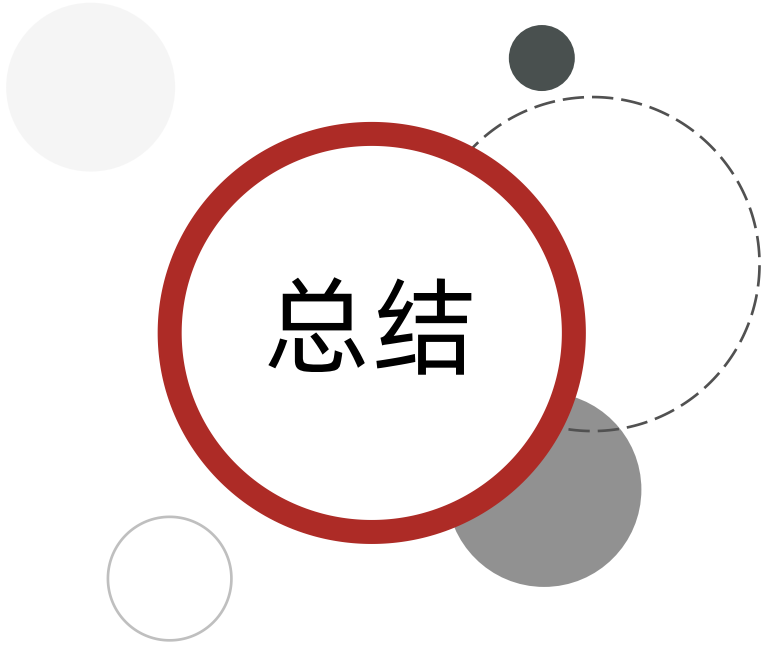
本质转换为数字

```
console.log(+new Date())
```

Date.now()

无需实例化
但是只能得到当前的时间戳

```
console.log(Date.now())
```



1. 为什么需要时间戳？
   - 计算倒计时效果，需要借助于时间戳完成
2. 时间戳是什么？
   - 是计量时间的方式，是指1970年01月01日00时00分00秒起至现在的总毫秒数
3. 获取时间戳有哪三种方式？重点记住那个？
   - date.getTime()
   - +new Date()　　使用较为简单
   - Date.now()

案例

倒计时效果

需求：计算到下课还有多少时间（18:30）

分析：

①：核心： 使用将来的时间戳减去现在的时间戳，封装函数 getTimer

②：把剩余的时间戳转换为 时、分、秒

③：把计算时分秒写到对应 span 盒子里面

④： 添加定时器效果自动变化时间

注意：

1. 通过时间戳得到是毫秒，需要转换为秒在计算
2. 转换公式：
   - h = parseInt(总秒数/ 60/60 %24) // 计算小时
   - m = parseInt(总秒数 /60 %60) // 计算分数
   - s = parseInt(总秒数%60) // 计算当前秒数

目录

Contents

02

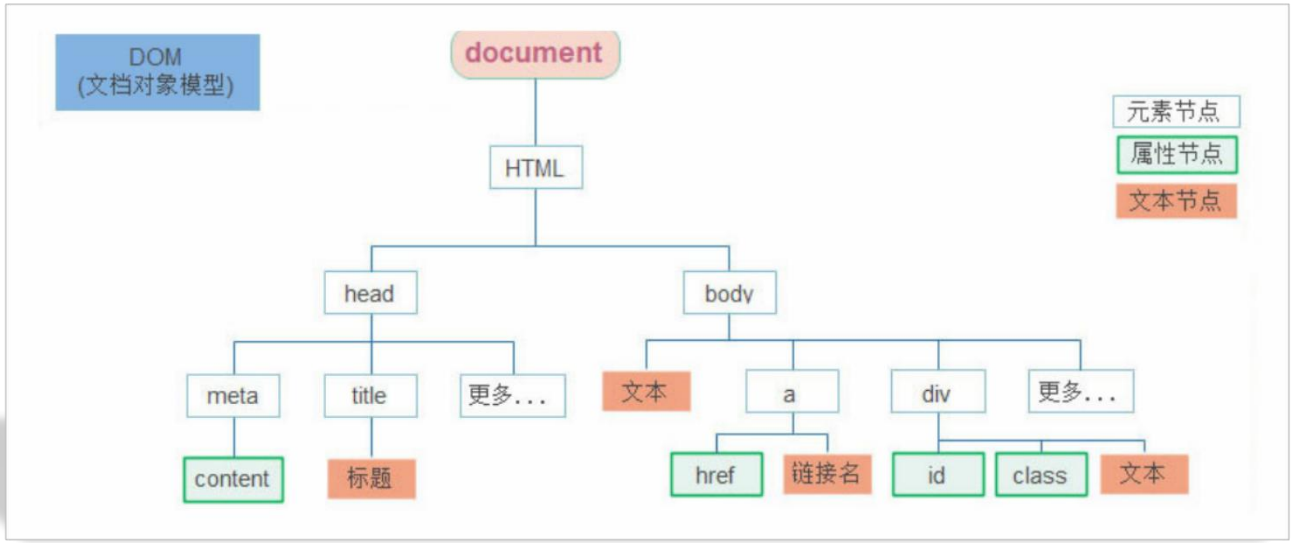
节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

2.1 DOM节点

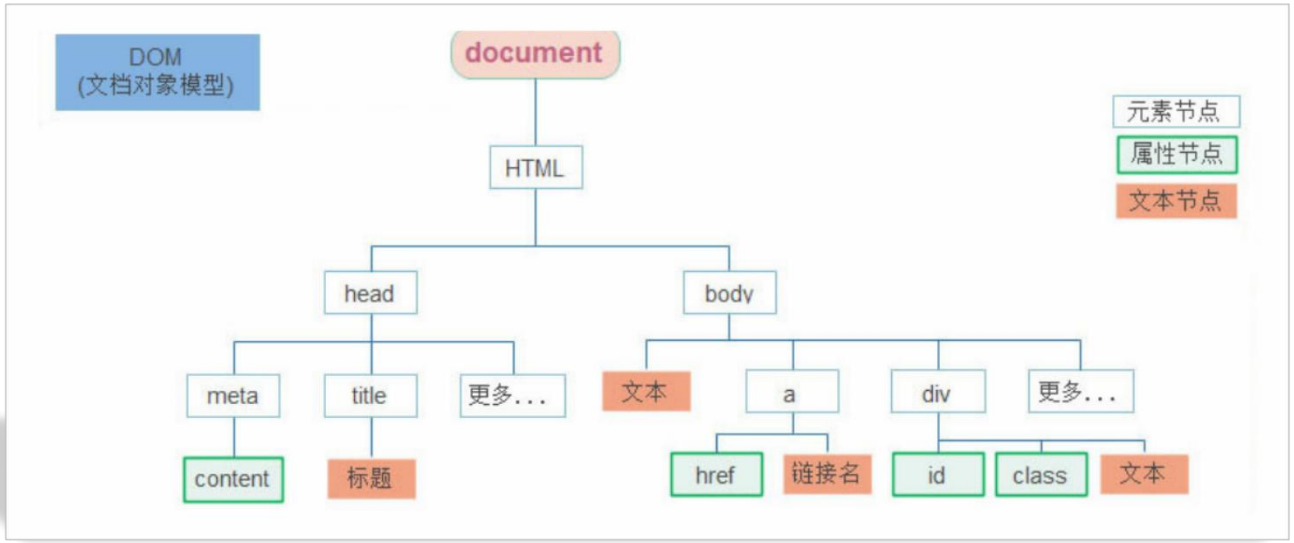
- **DOM树：** DOM 将 HTML文档以树状结构直观的表现出来，我们称之为 DOM 树 或者 节点树
- 节点（Node）是DOM树(节点树)中的单个点。包括文档本身、元素、文本以及注释都属于是节点。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>标题</title>
</head>
<body>
  文本
  <a href="">链接名</a>
  <div id="" class="">文本</div>
</body>
</html>
```

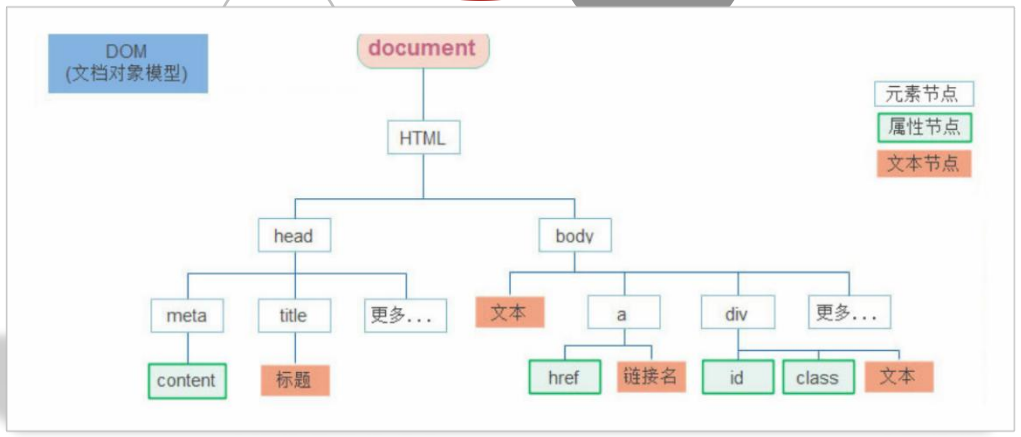


2.1 DOM节点

- **DOM树：** DOM 将 HTML文档以树状结构直观的表现出来，我们称之为 DOM 树 或者 节点树
- 节点（Node）是DOM树(节点树)中的单个点。包括文档本身、元素、文本以及注释都属于是节点。
  - **元素节点（重点）**
    - 所有的标签 比如 body、 div
    - html 是根节点
  - 属性节点
    - 所有的属性 比如 href
  - 文本节点
    - 所有的文本
- 学习节点有什么好处？
  - 利用节点关系可以更好的操作元素（比如查询更方便）



总结



1. 什么是DOM树？
   - DOM 将 HTML文档以树状结构直观的表现出来
2. 什么是DOM 节点？
   - DOM树里每一个点都称之为节点
3. DOM节点的分类？我们重点记住哪个？
   - 元素节点 比如 div标签
   - 属性节点 比如 class属性
   - 文本节点 比如标签里面的文字

02

节点操作

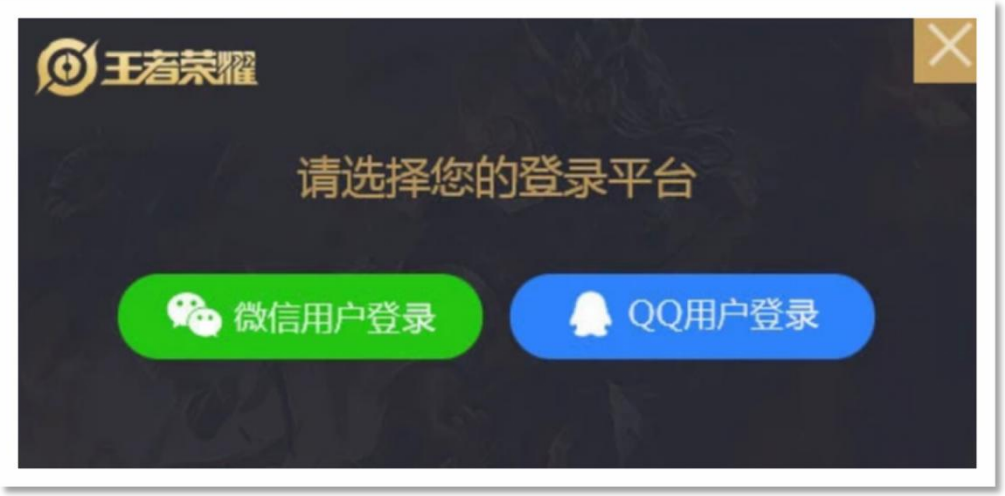
- DOM 节点
- 查找节点
- 增加节点
- 删除节点

2.2 查找节点

查找节点：利用节点关系查找节点，返回的都是对象

- 父节点
- 子节点
- 兄弟节点

- 有了查找节点可以使我们选择元素更加方便



2.2 查找节点

- 父节点查找：

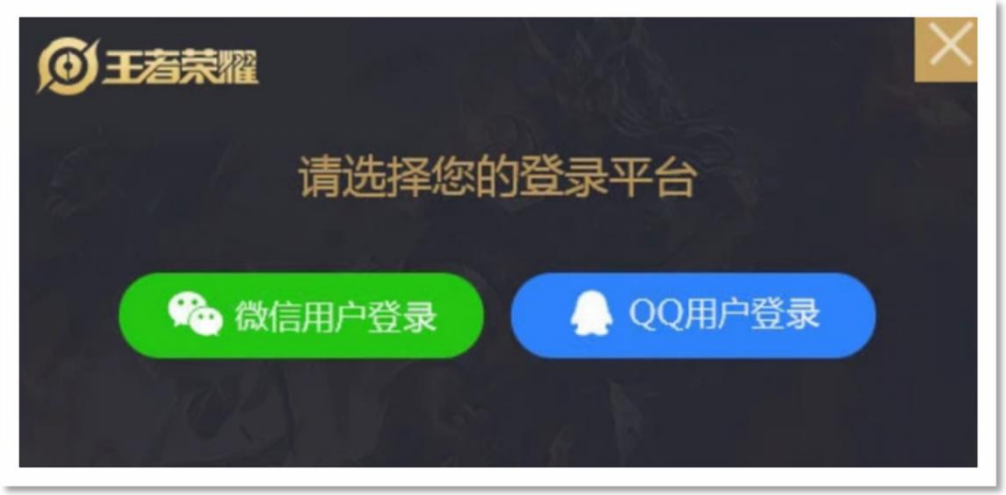
parentNode	返回最近一级的父节点对象， 找不到返回为 null

```
子元素.parentNode
```

案例

关闭王者荣耀案例

需求：点击关闭按钮，关闭弹窗



分析：

①：点击关闭按钮，关闭父元素

2.2 查找节点

- **子节点查找：**

```
<ul>
    <li>我是第1个孩子</li>
    <li>我是第2个孩子</li>
    <li>我是第3个孩子</li>
    <li>我是第4个孩子</li>
</ul>
```

children（重点）

- 获得所有子元素节点，返回的是一个伪数组
- 节点对象.children

2.2 查找节点

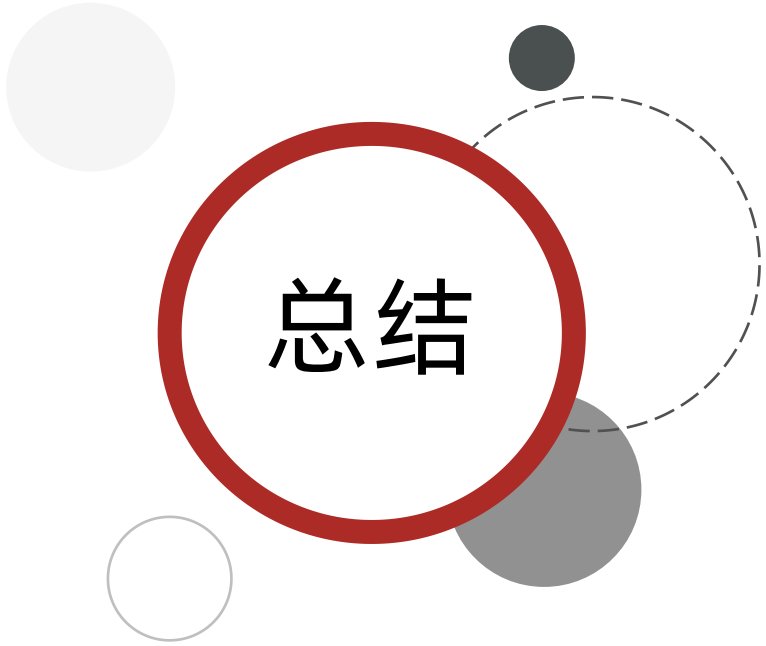
- 兄弟关系查找：

下一个兄弟

- 节点对象.nextElementSibling
- 该属性获取节点的下一个兄弟节点

上一个兄弟

- 节点对象.previousElementSibling
- 该属性获取节点的上一个兄弟节点



1. 查找所有子元素节点用那个属性？返回值是什么？
   - 节点对象.children
   - 返回的是伪数组
2. 查找兄弟节点有哪2个属性？
   - nextElementSibling 下一个兄弟
   - previousElementSibling 上一个兄弟

02

节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

2.3 增加节点

- 很多情况下，我们需要在页面中增加元素
  - 比如，点击发布按钮，可以新增一条信息
- 一般情况下，我们新增节点，按照如下操作：
  - 创建一个新的节点
  - 把创建的新的节点放入到指定的元素内部

2.3 增加节点

1.创建节点

- 即创造出一个新的网页元素
- 创建元素节点语法：

```
// 创造一个新的元素节点
document.createElement('标签名')
```

2.3 增加节点

2.追加节点

- 要想在界面看到，还得插入到某个父元素中
- 父元素最后一个子节点之后，插入节点元素

```
element.append()
```

- 父元素第一个子元素的之前，插入节点元素

```
element.prepend()
```

02

节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

2.4 删除节点

- 若一个节点在页面中已不需要时，可以删除它
- 语法

```
element.remove()
```

  - 把对象从它所属的 DOM 树中删除
  - 删除节点和隐藏节点（display:none） 有区别的： 隐藏节点还是存在的，但是删除，则从DOM树中删除

目录

Contents

3. 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

目录

Contents

4.插件-swiper

- 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
- 学习插件的思路：

1. **看官网。**了解这个插件可以完成什么需求　　https://www.swiper.com.cn/
2. **查看基本使用流程 。**　　https://www.swiper.com.cn/usage/index.html
3. **写个小demo。**看在线演示,找到符合自己需求的demo　　https://www.swiper.com.cn/demo/index.html
4. **应用的开发中。**

4.插件-swiper

```
<link rel="stylesheet" href="./css/swiper.min.css">
<script src="./js/swiper.min.js"></script>
```

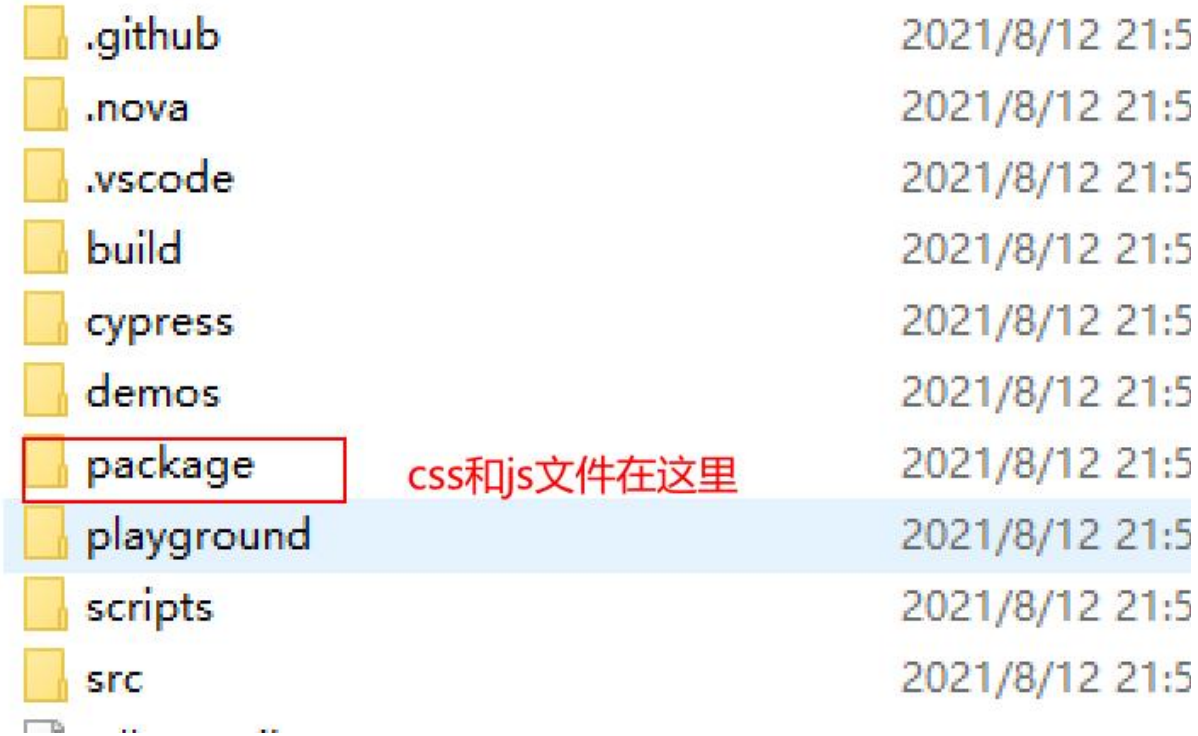
- 小demo的实现步骤

1. 下载相关js文件和css文件引入到html页面中
2. 找到自己喜欢的轮播图 demo，新窗口中打开，然后复制相关css、html、js代码
3. 填充自己的图片

```
<div class="swiper-wrapper">
  <!-- 每个小图片模块 -->
  <div class="swiper-slide">
    <img src="./images/slider01.jpg" alt="">
  </div>
  <!-- 每个小图片模块 -->
  <div class="swiper-slide">
    <img src="./images/slider02.jpg" alt="">
  </div>
```

4.插件

- 1. 本地文件


.github 2021/8/12 21:5
.nova 2021/8/12 21:5
.vscode 2021/8/12 21:5
build 2021/8/12 21:5
cypress 2021/8/12 21:5
demos 2021/8/12 21:5
package 2021/8/12 21:5
css和js文件在这里
playground 2021/8/12 21:5
scripts 2021/8/12 21:5
src 2021/8/12 21:5

插件- AlloyFinger

- 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
 }
})
```

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
请输入姓名
请输入手机号
添加联系人

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：遍历准备好的对象数组

②：遍历数组，我们使用新方法 forEach()

```
arr.forEach(function (element, index) { /* … */ })
```

- element 是数组元素
- index 是数组元素的索引号

③：利用字符串拼接，渲染页面

- 注意封装渲染函数，后期要用

```
<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>
```

④：里面需要用到字符串截取，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])

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)

新增业务

说明：

本次案例，我们尽量减少dom操作，采取操作数据的形式，为了后期Vue做铺垫

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（数据驱动视图）

思路：

新增数据 → 追加给数组 → 利用数组渲染页面

```
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)

新增业务

核心步骤：

①： 用户点击按钮，先判断姓名和电话是否为空，如果为空则提示不能为空，终止执行

②： 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' }
]
```

③： 重新渲染页面，然后清空姓名和手机号表单

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，为了后期Vue做铺垫

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（数据驱动视图）

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：

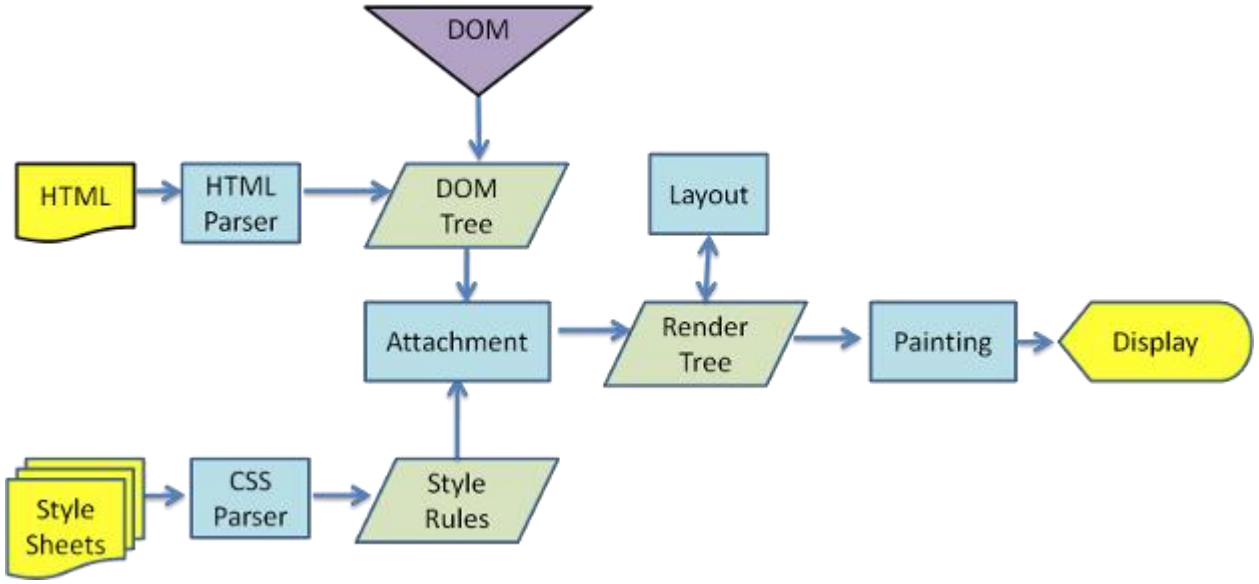
③：删除数组中对应索引号的数据，然后重新渲染页面

- 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

拓展-重绘和回流（导师讲解）

1. 浏览器是如何进行界面渲染的



- 解析（Parser）HTML，生成DOM树(DOM Tree)
- 同时解析（Parser） CSS，生成样式规则 (Style Rules)
- 根据DOM树和样式规则，生成渲染树(Render Tree)
- 进行布局 Layout(回流/重排):根据生成的渲染树，得到节点的几何信息（位置，大小）
- 进行绘制 Painting(重绘)：根据计算和获取的信息进行整个页面的绘制
- Display：展示在页面上

拓展-重绘和回流（导师讲解）

- **回流(重排)**

当 Render Tree 中部分或者全部元素的尺寸、结构、布局等发生改变时，浏览器就会重新渲染部分或全部文档的过程称为 回流。

- **重绘**

由于节点(元素)的样式的改变并不影响它在文档流中的位置和文档布局时(比如：color、background-color、outline等)，称为重绘。

- **重绘不一定引起回流，而回流一定会引起重绘。**

拓展-重绘和回流（导师讲解）

- 会导致回流（重排）的操作：
  - 页面的首次刷新
  - 浏览器的窗口大小发生改变
  - 元素的大小或位置发生改变
  - 改变字体的大小
  - 内容的变化（如：input框的输入，图片的大小）
  - 激活css伪类 （如：:hover）
  - 脚本操作DOM（添加或者删除可见的DOM元素）

简单理解影响到布局了，就会有回流

拓展-重绘和回流（导师讲解）

思考下述代码的重绘重排过程!

```
let s = document.body.style
s.padding = "2px" // 重排 + 重绘
s.border = "1px solid red" // 再一次 重排 + 重绘
s.color = "blue" // 再一次重绘
s.backgroundColor = "#ccc" // 再一次 重绘
s.fontSize = "14px" // 再一次 重排 + 重绘
```

传智教育旗下高端IT教育品牌