Web APIs 第四天

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

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

目录

Contents

01

日期对象

- 实例化
- 时间对象方法
- 时间戳

1. 日期对象

目标：掌握日期对象，可以让网页显示日期

- 日期对象：用来表示时间的对象
- 作用：可以得到当前系统时间

学习路径：

1. 实例化
2. 日期对象方法
3. 时间戳

1.1 实例化

目标：能够实例化日期对象

- 在代码中发现了 new 关键字时，一般将这个操作称为**实例化**
- 创建一个时间对象并获取时间
  - 获得当前时间

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

  - 获得指定时间

```
const date = new Date('2008-8-8')
console.log(date)
```

01

日期对象

- 实例化
- 日期对象方法
- 时间戳

2.2 日期对象方法

目标：能够使用日期对象中的方法写出常见日期

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

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

页面显示时间

需求：将当前时间以：YYYY-MM-DD HH:mm 形式显示在页面　　2008-08-08 08:08

分析：

①：调用日期对象方法进行转换

②：记得数字要补0

③：字符串拼接后，通过 innerText 给 标签

01

日期对象

- 实例化
- 日期对象方法
- 时间戳

2.3 时间戳

目标：能够获得当前时间戳

- **使用场景：** 如果计算倒计时效果，前面方法无法直接计算，需要借助于时间戳完成
- **什么是时间戳：**
  - 是指1970年01月01日00时00分00秒起至现在的毫秒数，它是一种特殊的计量时间的方式
- **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 剩余时间的 年月日时分秒 就是 倒计时时间
- 比如 将来时间戳 2000ms - 现在时间戳 1000ms = 1000ms
- 1000ms 转换为就是 0小时0分1秒

2.3 时间戳

三种方式获取时间戳：

1. 使用 getTime() 方法

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

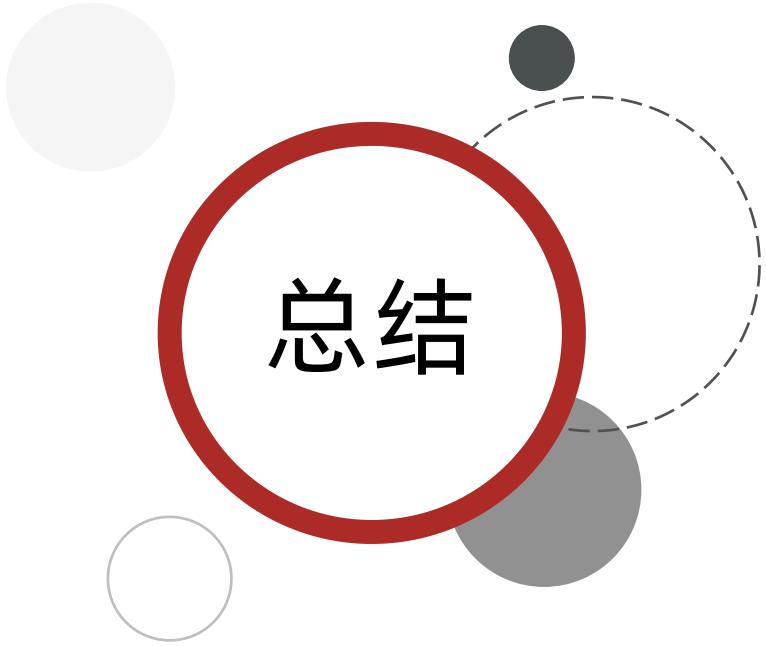
2. **简写 +new Date()**

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

3. 使用 Date.now()
   - 无需实例化
   - 但是只能得到当前的时间戳， 而前面两种可以返回指定时间的时间戳

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

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



1. 实例化日期对象
   - 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
3. 获取时间戳有哪三种方式？重点记住那个？
   - date.getTime()
   - +new Date()
   - Date.now()
   - 重点记住 +new Date() 因为可以返回当前时间戳或者指定的时间戳

案例

毕业倒计时效果

需求：计算到下课还有多少时间

分析：

①：用将来时间减去现在时间就是剩余的时间

②：核心： 使用将来的时间戳减去现在的时间戳

③：把剩余的时间转换为 天 时 分 秒

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：
   - d = parseInt(总秒数/ 60/60 /24); // 计算天数
   - 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

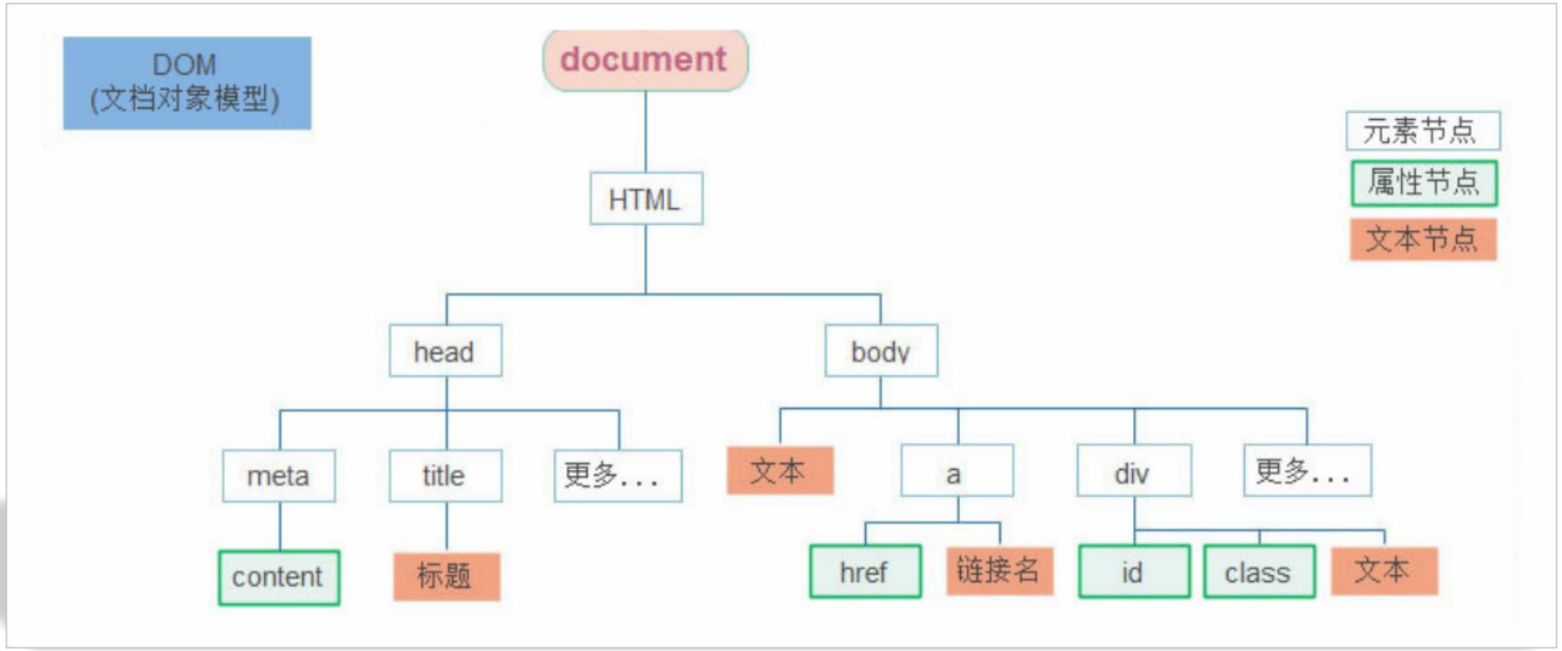
节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

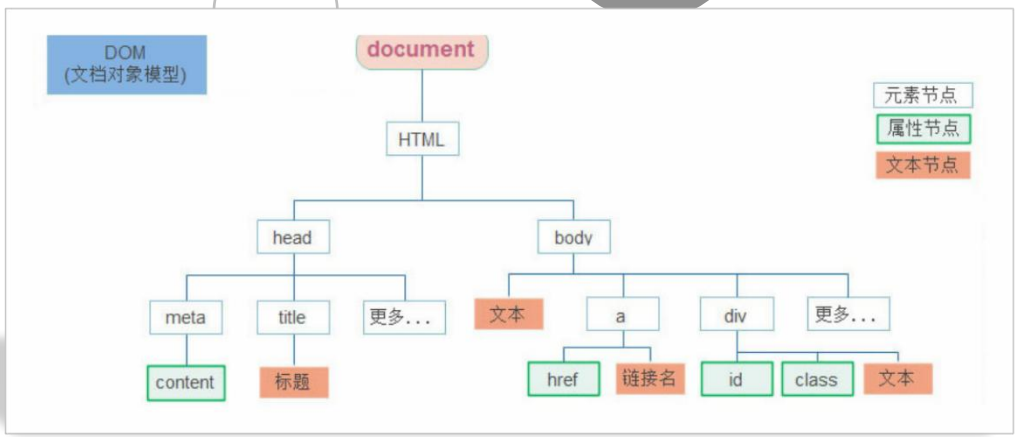
1.1 DOM节点

目标：能说出DOM节点的类型

- DOM节点
  - DOM树里每一个内容都称之为节点
- 节点类型
  - **元素节点**
    - 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
  - 其他



总结



1. 什么是DOM 节点？
   - DOM树里每一个内容都称之为节点
2. DOM节点的分类？
   - 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
3. 我们重点记住那个节点？
   - 元素节点
   - 可以更好的让我们理清标签元素之间的关系

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

目标：能够具备根据节点关系查找目标节点的能力

- 关闭二维码案例：

  点击关闭按钮， 关闭的是二维码的盒子， 还要获取erweima盒子

- 思考：
  - 关闭按钮 和 erweima 是什么关系呢？
  - 父子关系
  - 所以，我们完全可以这样做：
  - 点击关闭按钮， 直接关闭它的爸爸，就无需获取erweima元素了

```
<div class="erweima">
  <img src="./images/code.png" alt>
  <i class="close_btn">x</i>
</div>
```

- **节点关系：针对的找亲戚返回的都是对象**
  - 父节点
  - 子节点
  - 兄弟节点

2.2 查找节点

- 父节点查找：
  - parentNode 属性
  - 返回最近一级的父节点 找不到返回为null

```
子元素.parentNode
```

案例

关闭二维码案例

需求：关闭二维码案例

2.2 查找节点

- **子节点查找：**
  - childNodes
    - ✓ 获得所有子节点、包括文本节点（空格、换行）、注释节点等
  - **children 属性 （重点）**
    - ✓ 仅获得所有元素节点
    - ✓ 返回的还是一个伪数组

```
父元素.children
```

2.2 查找节点

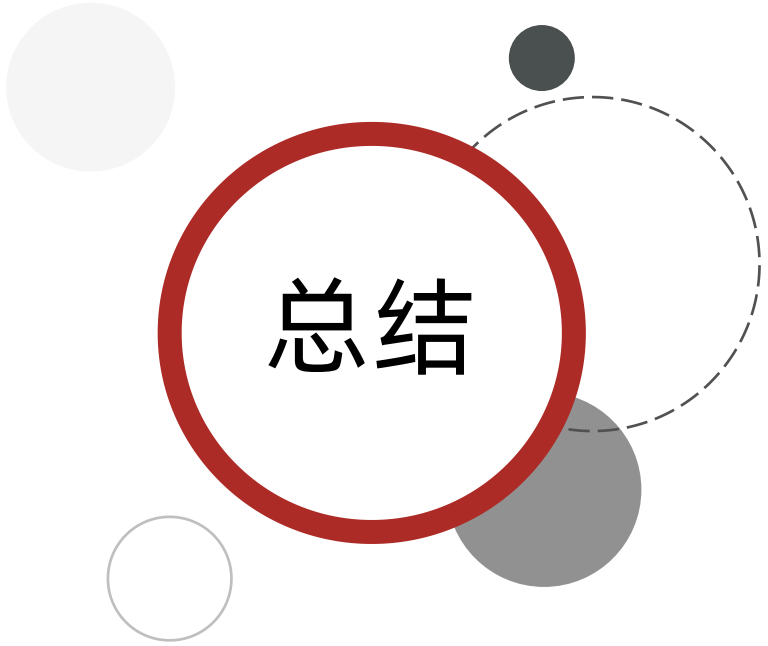
- **兄弟关系查找：**

  1. 下一个兄弟节点

  - nextElementSibling 属性

  2. 上一个兄弟节点

  - previousElementSibling 属性



1. 查找父节点用那个属性？
   - parentNode
2. 查找所有子节点用那个属性？
   - children
3. 查找兄弟节点用那个属性？
   - nextElementSibling
   - previousElementSibling

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

目标：能够具备根据需求新增节点的能力

- 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
- 学习路线：
  - 创建节点
  - 追加节点

2.3 增加节点

1.创建节点

- 即创造出一个新的网页元素，再添加到网页内，一般先创建节点，然后插入节点
- 创建元素节点方法：

```
// 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
- 插入到父元素的最后一个子元素：

```
// 插入到这个父元素的最后
父元素.appendChild(要插入的元素)
```

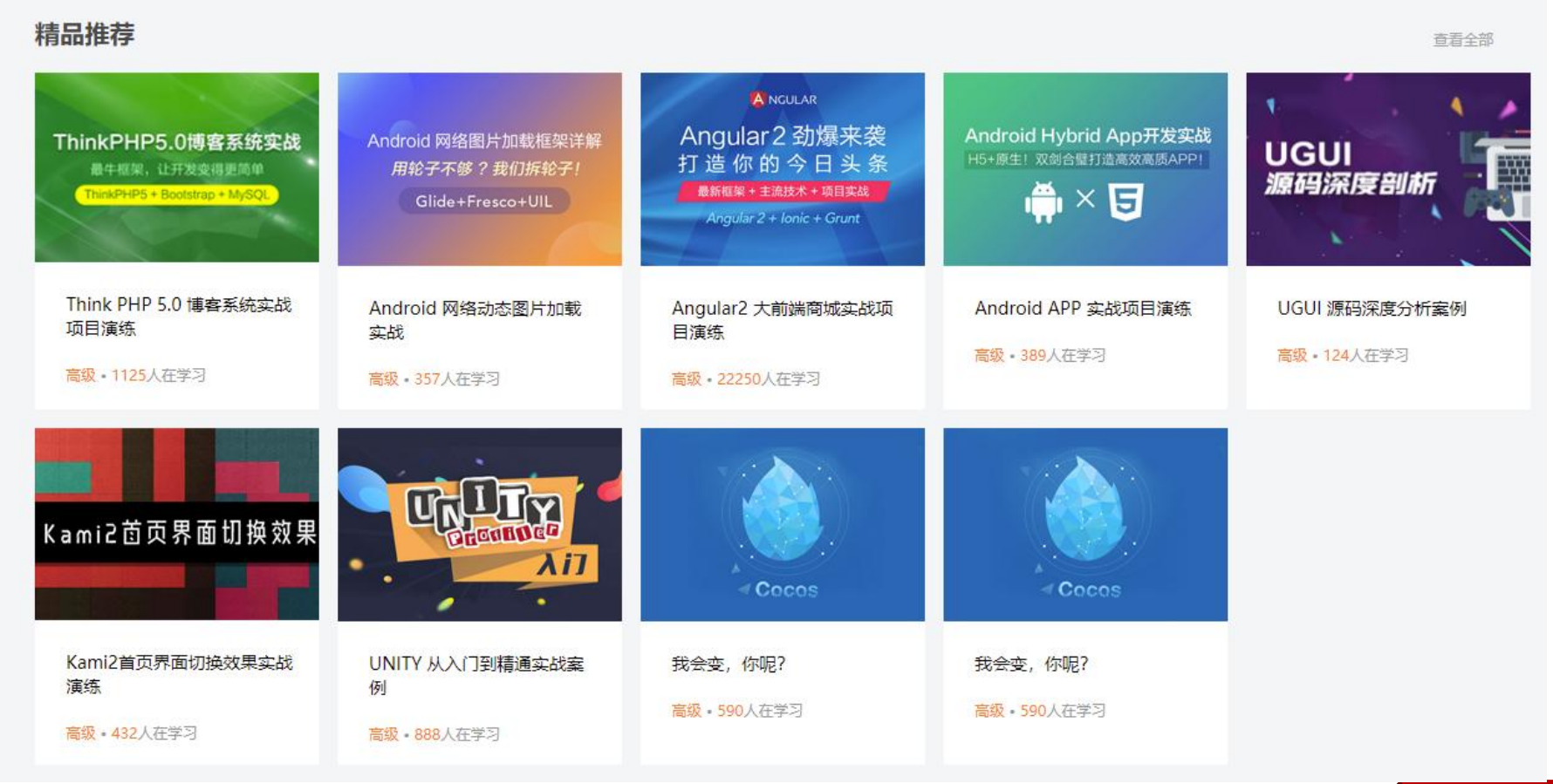
- 插入到父元素中某个子元素的前面

```
// 插入到某个子元素的前面
父元素.insertBefore(要插入的元素, 在哪个元素前面)
```

案例

学成在线案例渲染

需求：按照数据渲染页面



步骤

学成在线案例渲染

需求：按照数据渲染页面

分析：

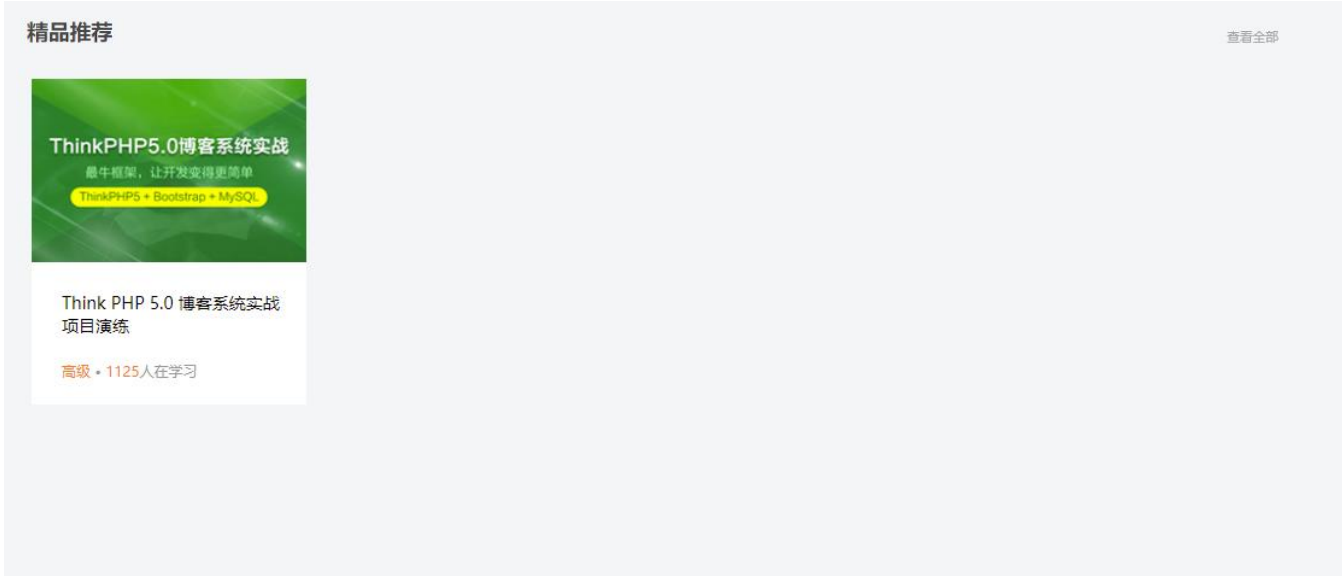
①：准备好空的ul 结构

②：根据数据的个数，创建一个新的空li

③：li里面添加内容 img 标题等

④：追加给ul

重点练习：创建节点和追加节点



2.3 增加节点

- 特殊情况下，我们新增节点，按照如下操作：
  - 复制一个原有的节点
  - 把复制的节点放入到指定的元素内部
- **克隆节点**


```
// 克隆一个已有的元素节点
元素.cloneNode(布尔值)
```

cloneNode会克隆出一个跟原标签一样的元素，括号内传入布尔值

- 若为true，则代表克隆时会包含后代节点一起克隆
- 若为false，则代表克隆时不包含后代节点
- 默认为false

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

目标：能够具备根据需求删除节点的能力

- 若一个节点在页面中已不需要时，可以删除它
- 在 JavaScript 原生DOM操作中，要删除元素必须通过父元素删除
- **语法**

```
父元素.removeChild(要删除的元素)
```

- 注：
  - 如不存在父子关系则删除不成功
  - 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html中删除节点

目录

Contents

3. M端事件

目标：了解M端常见的事件

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

- 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

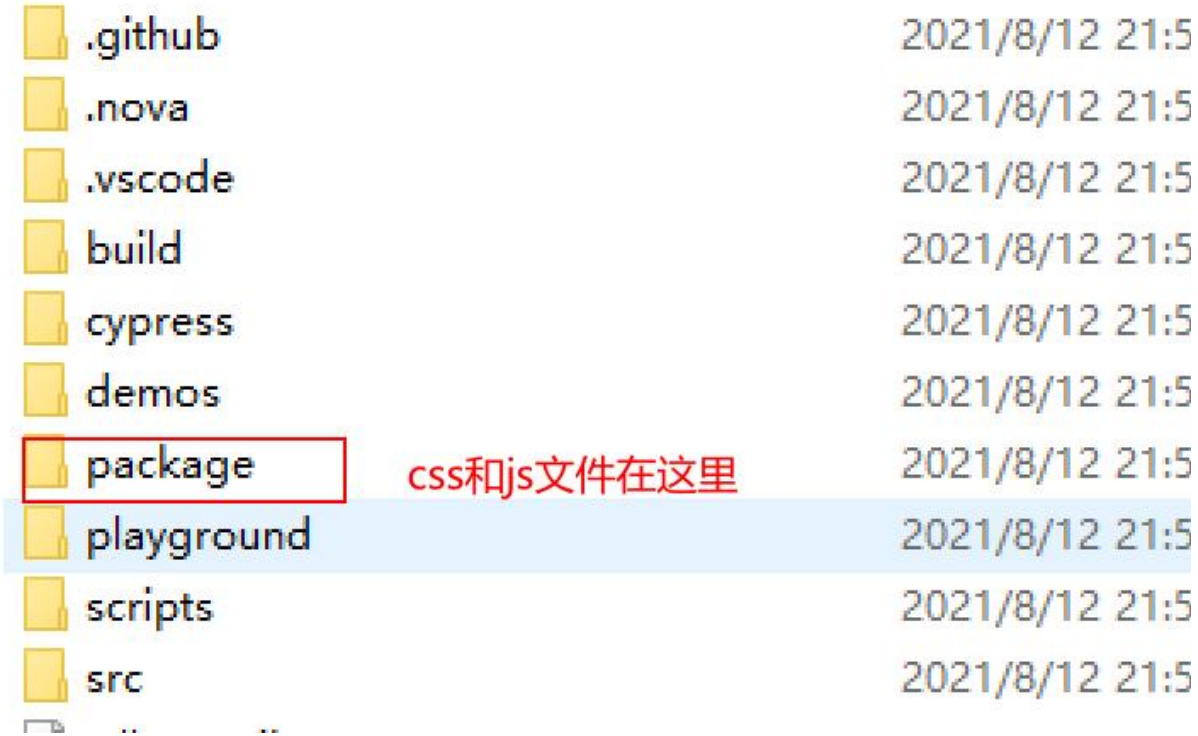
Contents

4.插件

- 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
- 学习插件的基本过程
  - 熟悉官网,了解这个插件可以完成什么需求 https://www.swiper.com.cn/
  - 看在线演示,找到符合自己需求的demo https://www.swiper.com.cn/demo/index.html
  - 查看基本使用流程 https://www.swiper.com.cn/usage/index.html
  - 查看APi文档,去配置自己的插件 https://www.swiper.com.cn/api/index.html
  - 注意：多个swiper同时使用的时候，类名需要注意区分

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

目录

Contents

案例

游乐园轮播图

案例

学生信息表案例

新增学员

姓名： 年龄： 性别: 男 薪资： 就业城市: 北京 录入

就业榜

学号	姓名	年龄	性别	薪资	就业城市	操作

步骤

学生信息表案例

业务模块：

①： 点击录入按钮可以录入数据

②： 点击删除可以删除当前的数据

步骤

学生信息表案例

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

步骤

学生信息表案例

核心思路：

①： 声明一个空的数组

②： 点击录入，根据相关数据，生成对象，追加到数组里面

③： 根据数组数据渲染页面-表格的 行

④： 点击删除按钮，删除的是对应数组里面的数据

⑤： 再次根据数组的数据，渲染页面

步骤

学生信息表案例

核心思路：

①： 声明一个空的数组

②： 点击录入模块

(1). 首先取消表单默认提交事件

(2). 创建新的对象，里面存储 表单获取过来的数据，格式如右图

(3). 追加给数组

(4). 渲染数据。 遍历数组， 动态生成tr， 里面填写对应td数据， 并追加给 tbody

(5). 重置表单

(6). 注意防止多次生成多条数据，先清空 tbody

```
const initData = [
  {
    stuId: 1001,
    uname: '欧阳霸天',
    age: 19,
    gender: '男',
    salary: '20000',
    city: '上海',
  }
]
```

步骤

学生信息表案例

核心思路：

③： 点击删除模块

(1). 采用事件委托形式，给 tbody 注册点击事件

(2). 点击链接，要删除的是对应数组里面的这个数据，而不是删除dom节点，如何找到这个数据？

(3). 前面渲染数据的时候，动态给a链接添加 自定义属性 data-id=“0”,这样点击当前对象就知道索引号了

(4). 根据索引号，利用 splice 删除这条数据

(5). 重新渲染

步骤

学生信息表案例

核心思路：

④： 点击新增需要验证表单

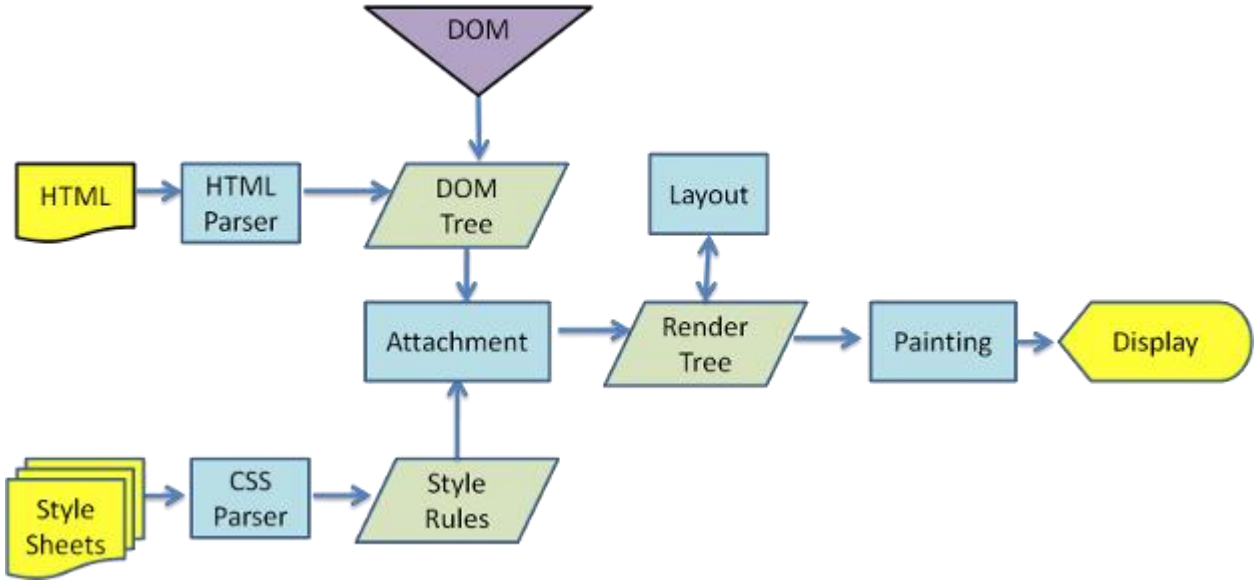
(1). 获取所有需要填写的表单， 他们共同特点都有 name属性

(2). 遍历这些表单，如果有一个值为空，则return 返回提示输入为空中断程序

(3). 注意书写的位置，应该放到新增数据的前面， 阻止默认行为的后面

4. 重绘和回流

- 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

4. 重绘和回流

- **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。**

4. 重绘和回流

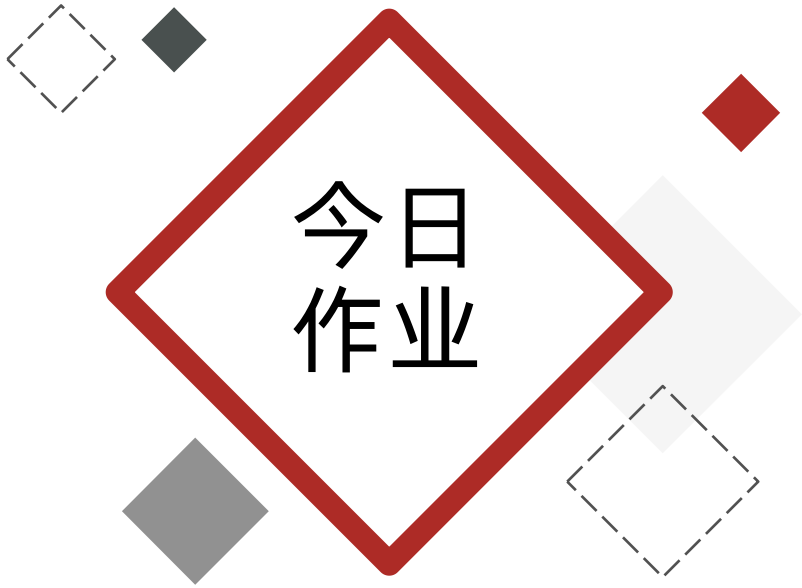
- 会导致回流（重排）的操作：
  - 页面的首次刷新
  - 浏览器的窗口大小发生改变
  - 元素的大小或位置发生改变
  - 改变字体的大小
  - 内容的变化（如：input框的输入，图片的大小）
  - 激活css伪类 （如：:hover）
  - 脚本操作DOM（添加或者删除可见的DOM元素）

简单理解影响到布局了，就会有回流

4. 重绘和回流

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
```



1. 整理今天笔记
2. 练习学成在线渲染案例、倒计时案例、微博发布案例
3. 做测试题：
4. 作业
5. 一定要预习下次课的本地存储课程

今天多一份拼搏，明日多一份欢笑

传智教育旗下高端IT教育品牌