

Web APIs 第三天

DOM- 节点操作



目录

Contents

- ◆ 节点操作
- ◆ 时间对象
- ◆ 综合案例
- ◆ 重绘和回流

学习目标

Learning Objectives

1. 掌握节点(标签)的增删改查
2. 具备编写微博发布案例的能力



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

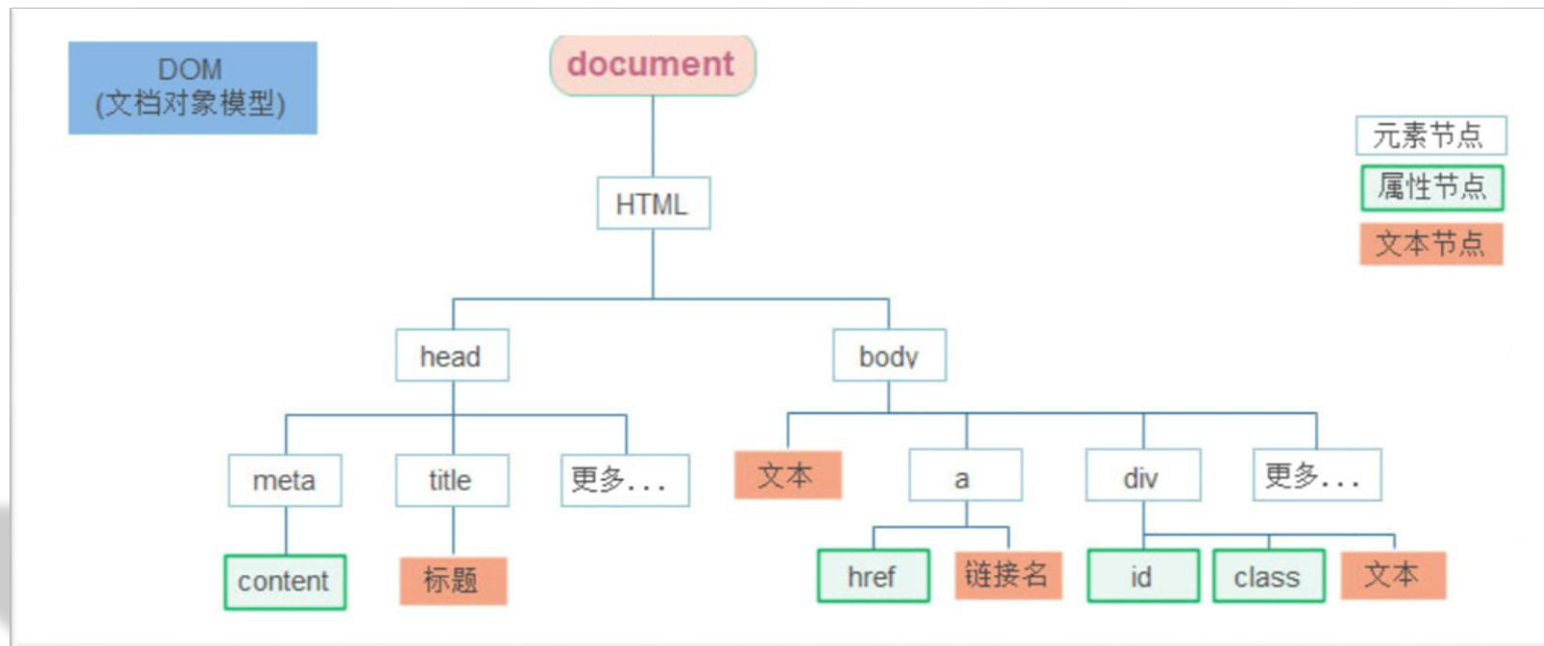
- 属性节点

- 所有的属性 比如 href

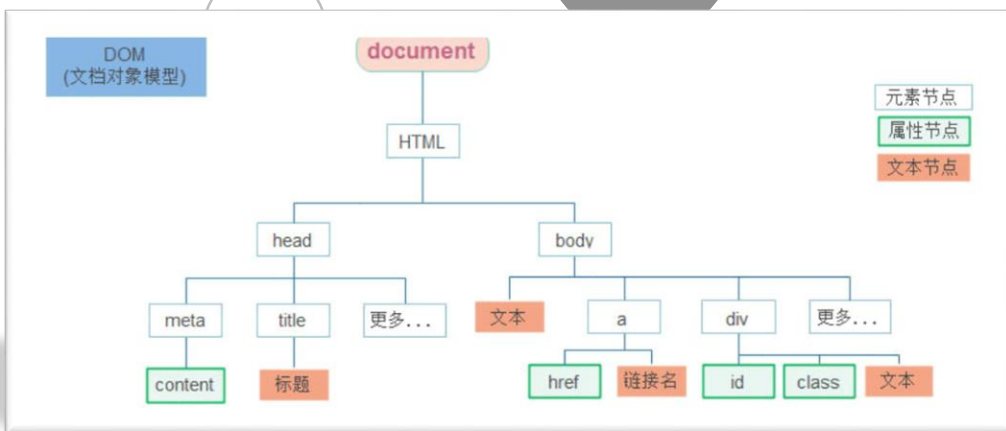
- 文本节点

- 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



节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

1.2 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

- 节点关系：

- 父节点
- 子节点
- 兄弟节点



```
<div class="erweima">  
    
  <i class="close_btn">x</i>  
</div>
```


1.2 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

- 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



案例

关闭二维码案例

需求：多个二维码，点击谁，谁关闭

分析：

- ①：需要给多个按钮绑定点击事件
- ②：关闭的是当前的父节点



1.2 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

- 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（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
```

1.2 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

- 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



总结

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



节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

1.3 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

有什么新鲜事想告诉大家？

说点什么吧...

0 / 200

发布

有什么新鲜事想告诉大家？

说点什么吧...

0 / 200

发布



英雄名称

2021/8/28 上午11:06:43

新增

1.3 增加节点

目标：能够具备根据需求新增节点的能力

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')
```

1.3 增加节点

目标：能够具备根据需求新增节点的能力

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

精品推荐


[查看全部](#)



ThinkPHP5.0博客系统实战
最牛框架，让开发变得更简单
ThinkPHP5 + Bootstrap + MySQL

Think PHP 5.0 博客系统实战项目演练


高级 • 1125人在学习



Android 网络图片加载框架详解
用轮子不够？我们拆轮子！
Glide+Fresco+UIL

Android 网络动态图片加载实战


高级 • 357人在学习



ANGULAR
Angular2 劲爆来袭
打造你的今日头条
最新框架 + 主流技术 + 项目实战
Angular 2 + Ionic + Grunt

Angular2 大前端商城实战项目演练


高级 • 22250人在学习



Android Hybrid App开发实战
H5+原生！双剑合璧打造高效高质APP！
Android × iOS

Android APP 实战项目演练


高级 • 389人在学习



UGUI
源码深度剖析

UGUI 源码深度分析案例


高级 • 124人在学习



Kami2 首页界面切换效果

Kami2 首页界面切换效果实战演练


高级 • 432人在学习



UNITY
PROVIDER
入门

UNITY 从入门到精通实战案例


高级 • 888人在学习



Cocos

我会变，你呢？

高级 • 590人在学习



Cocos

我会变，你呢？

高级 • 590人在学习

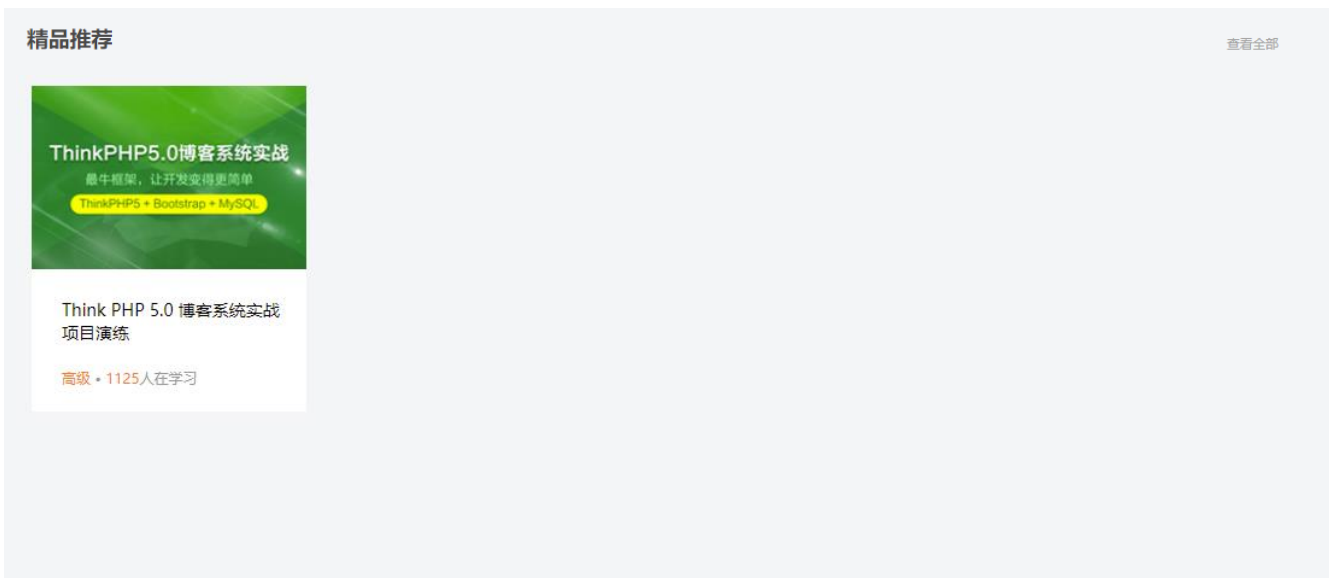
案例

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



1.3 增加节点

目标：能够具备根据需求新增节点的能力

- 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



节点操作

- DOM 节点
- 查找节点
- 增加节点
- 删除节点

1.3 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

- ◆ 节点操作
- ◆ 时间对象
- ◆ 综合案例
- ◆ 重绘和回流



时间对象

- 实例化
- 时间对象方法
- 时间戳

二. 时间对象

目标：掌握时间对象，可以让网页显示时间

- 时间对象：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. 时间对象方法
3. 时间戳



2.1 实例化

目标：能够实例化时间对象

- 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

```
let date = new Date()
```

➤ 获得指定时间

```
let date = new Date('1949-10-01')
```



时间对象

- 实例化
- 时间对象方法
- 时间戳

2.2 时间对象方法

目标：能够使用时间对象中的方法写出常见日期

- 因为时间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



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

分析：

- ①：调用时间对象方法进行转换
- ②：字符串拼接后，通过 innerText 给 标签



时间对象

- 实例化
- 时间对象方法
- 时间戳

2.3 时间戳

目标：能够获得当前时间戳

- 什么是时间戳
 - 是指1970年01月01日00时00分00秒起至现在的毫秒数，它是一种特殊的计量时间的方式
- 三种方式获取时间戳
 - 1. 使用 getTime() 方法

```
// 1. 实例化
let date = new Date()
// 2. 获取时间戳
console.log(date.getTime())
```

- 2. 简写 +new Date()

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


2.3 时间戳

目标：能够获得当前时间戳

- 什么是时间戳
 - 是指1970年01月01日00时00分00秒起至现在的毫秒数，它是一种特殊的计量时间的方式
- 三种方式获取时间：
 - 3. 使用 Date().now()

```
console.log(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

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



2. 时间对象总结

目标：掌握时间对象，可以让网页显示时间

1. 实例化时间对象

- `new Date()`

2. 时间对象方法

- 时间对象里面的方法转换实际所用

3. 时间戳

- `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

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



目录

Contents

- ◆ 节点操作
- ◆ 时间对象
- ◆ 综合案例
- ◆ 重绘和回流

案例

发布微博案例

有什么新鲜事想告诉大家？

第一条weib|

weib|

1. 微博 2. 尾部 3. 尾巴 4. 未必 5. 维保


黑马程序员
www.itheima.com

7 / 200 发布



发布微博案例

需求1

1. 注册input事件
2. 将文本的内容的长度赋值给对应的数值
3. 表单的maxlength属性可以直接限制在200个数之间

需求2

克隆预定义好的模板,将模板的hidden属性设置为false,并最终展示到页面上
判断如果内容为空,则提示不能输入为空,并且直接return
防止输入无意义空格,使用字符串.trim()去掉首尾空格,并将表单的value值设置为空字符串

需求3

获取文本域的内容,赋值给由模板克隆出来的新标签里面的content.innerText
随机获取数据数组里面的内容,替换newNode的图片和名称
利用时间对象将时间动态化 `new Date().toLocaleString()`



发布微博案例

需求4

在事件处理函数里面获取点击按钮,注册点击事件

(易错点: 必须在事件里面获取,外面获取不到)

删除对应的元素 (通过this获取对应的那条需要删除的元素)

需求5

将表单域内容重置为空

将userCount里面的内容重置为0



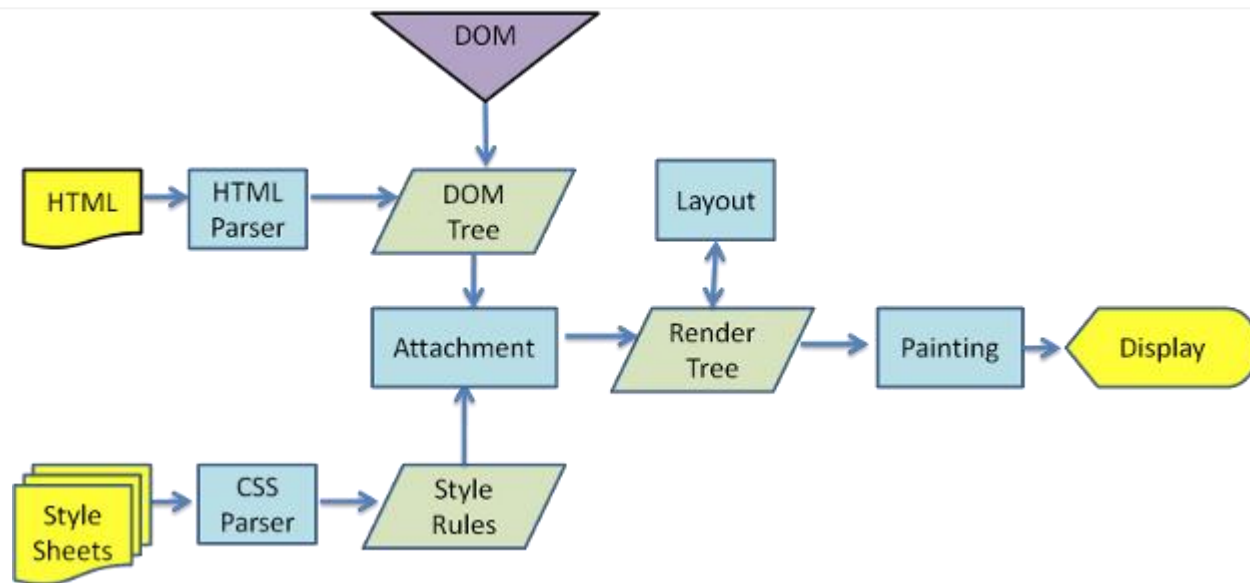
目录

Contents

- ◆ 节点操作
- ◆ 时间对象
- ◆ 综合案例
- ◆ 重绘和回流

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

2. 重绘和回流(重排)

- **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

2. 重绘和回流(重排)

- 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

2. 重绘和回流(重排)

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
```



思考

1. 整理今天笔记
2. 练习 关闭多个二维码案例
3. 练习学成在线渲染案例
4. 练习倒计时案例
5. 练习微博发布案例
6. 作业

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