Web APIs 第一天

DOM-获取DOM元素、修改属性

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. 能获取DOM元素并修改元素属性
2. 具备利用定时器间歇函数制作焦点图切换的能力

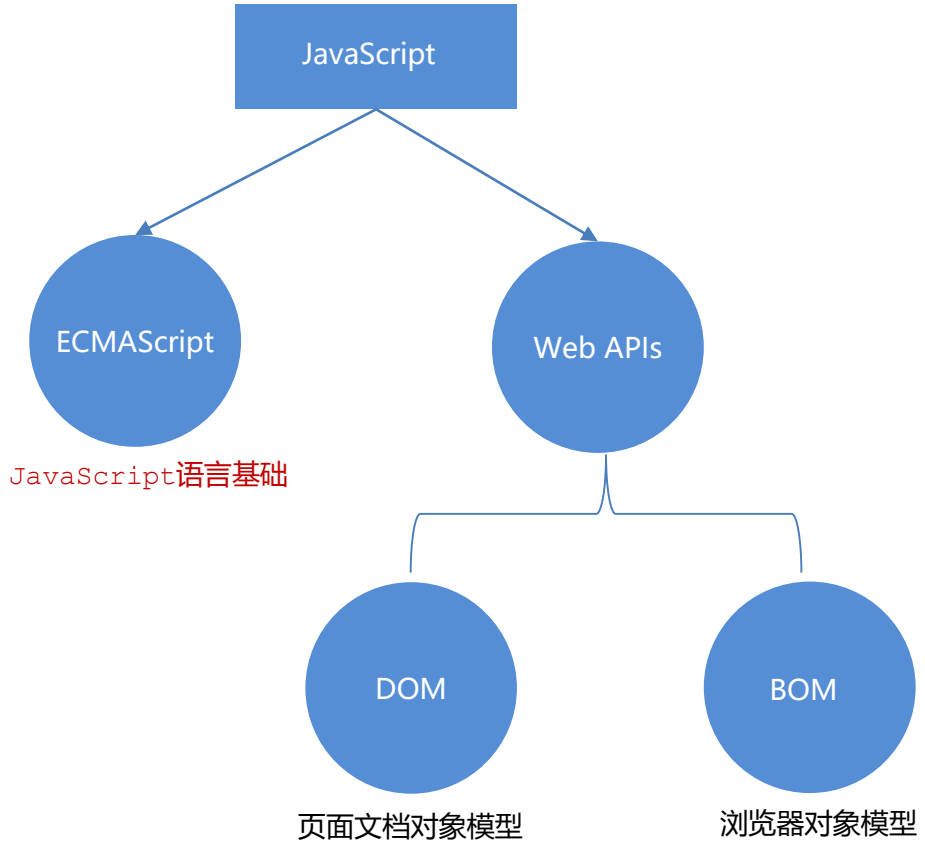
Web API 基本认知

- 作用和分类
- 什么是DOM
- DOM树
- DOM对象

一、Web API 基本认知

1. 作用和分类

- 作用: 就是使用 JS 去操作 html 和浏览器
- 分类：DOM (文档对象模型)、BOM（浏览器对象模型）

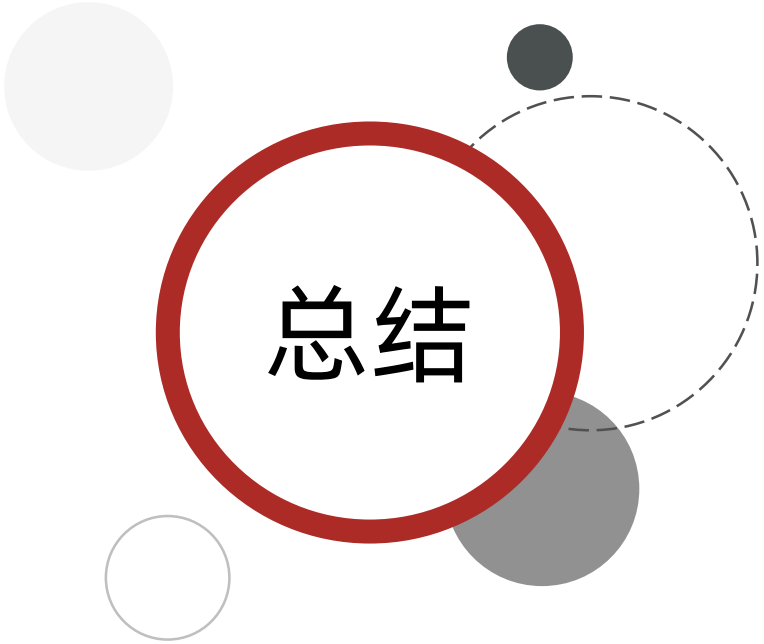


Web API 基本认知

- 作用和分类
- 什么是DOM
- DOM树
- DOM对象

2. 什么是DOM

- DOM（Document Object Model——文档对象模型）是用来呈现以及与任意 HTML 或 XML文档交互的API
- 白话文：DOM是浏览器提供的一套专门用来 **操作网页内容** 的功能
- DOM作用
  - 开发网页内容特效和实现用户交互



1. Web API阶段我们学习那两部分？
   - DOM
   - BOM
2. DOM 是什么？有什么作用？
   - DOM 是文档对象模型
   - 操作网页内容，可以开发网页内容特效和实现用户交互

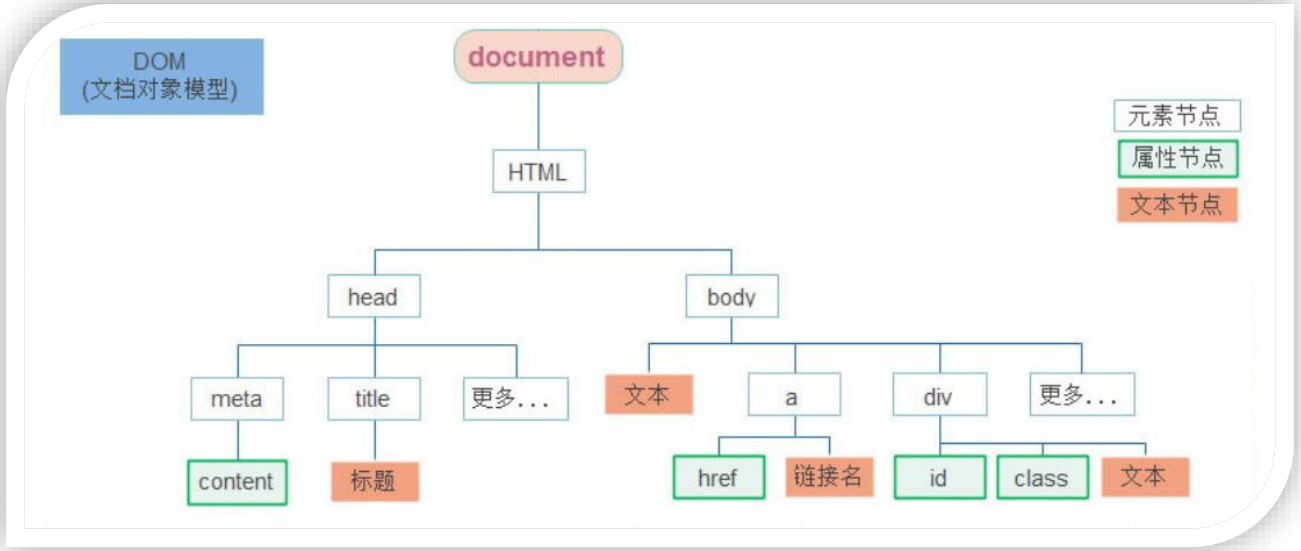
Web API 基本认知

- 作用和分类
- 什么是DOM
- DOM树
- DOM对象

3. DOM树

- DOM树是什么
  - 将 HTML 文档以树状结构直观的表现出来，我们称之为文档树或 DOM 树
  - 描述网页内容关系的名词
  - 作用：文档树直观的体现了标签与标签之间的关系

```html
<!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>
```

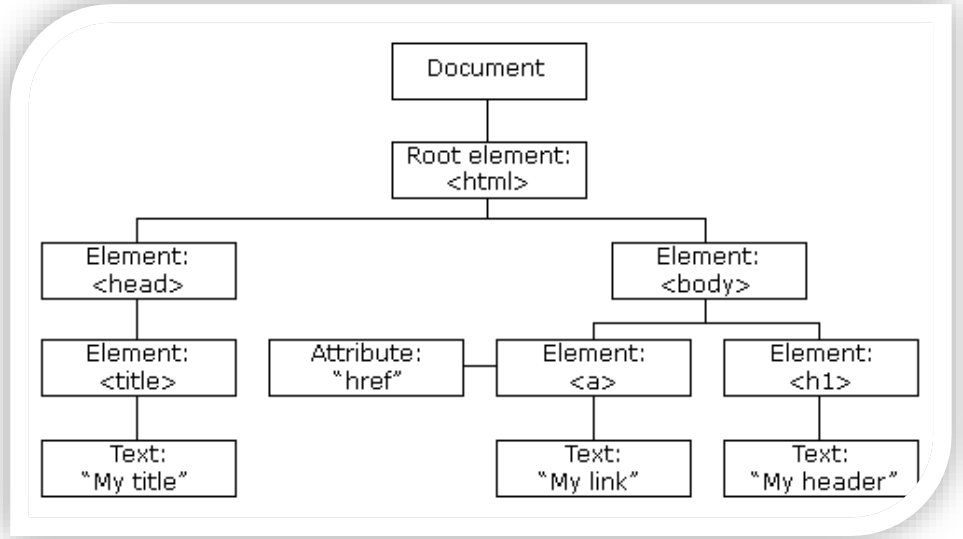


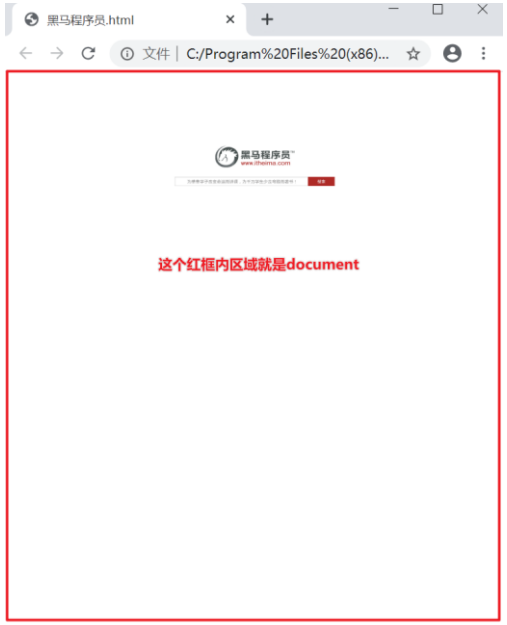
Web API 基本认知

- 作用和分类
- 什么是DOM
- DOM树
- DOM对象

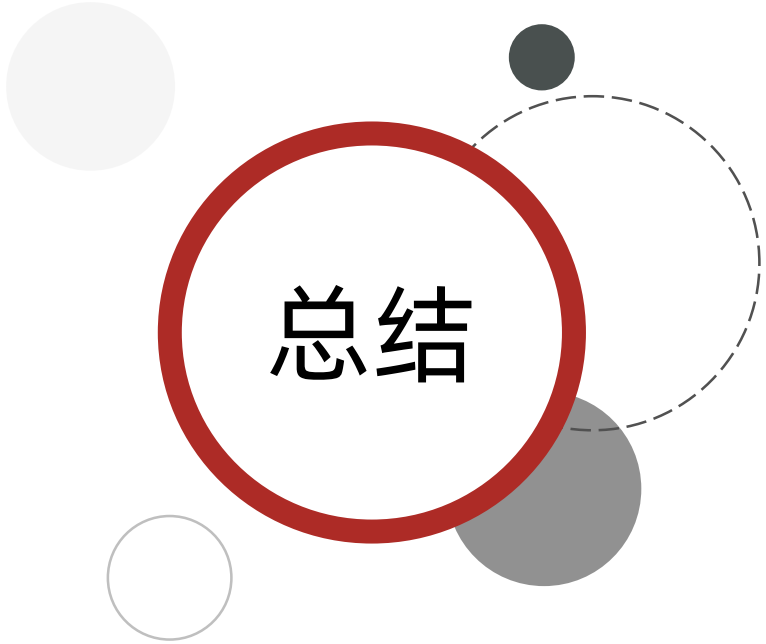
4. DOM对象（重要）

- DOM对象：浏览器根据html标签生成的 JS对象
  - 所有的标签属性都可以在这个对象上面找到
  - 修改这个对象的属性会自动映射到标签身上
- DOM的核心思想
  - 把网页内容当做对象来处理
- document 对象
  - 是 DOM 里提供的一个**对象**
  - 所以它提供的属性和方法都是**用来访问和操作网页内容的**
    - ✓ 例：document.write()
  - 网页所有内容都在document里面





小结



1. DOM 树是什么?
   - 将 HTML 文档以树状结构直观的表现出来，我们称之为文档树或 DOM 树
   - 作用：文档树直观的体现了标签与标签之间的关系
2. DOM对象怎么创建的?
   - 浏览器根据html标签生成的 JS对象（DOM对象）
   - DOM的核心就是把内容当对象来处理
3. document 是什么?
   - 是 DOM 里提供的一个**对象**
   - 网页所有内容都在document里面

目录

Contents

获取DOM元素

- 根据CSS选择器来获取DOM元素（重点）
- 其他获取DOM元素方法（了解）

二、获取DOM对象

目标：能查找/获取DOM对象

提问：我们想要操作某个标签首先做什么？

- 肯定首先选中这个标签，跟 CSS选择器类似，选中标签才能操作

- **查找元素DOM元素就是选择页面中标签元素**

学习路径：

1. 根据CSS选择器来获取DOM元素（重点）
2. 其他获取DOM元素方法（了解）

二、获取DOM对象

1. 根据CSS选择器来获取DOM元素（重点）

1.1 选择匹配的第一个元素

语法：

```
document.querySelector('css选择器')
```

参数：

包含一个或多个有效的CSS选择器 **字符串**

返回值：

CSS选择器匹配的第一个元素,一个 HTMLElement对象。

如果没有匹配到，则返回null。

多参看文档：https://developer.mozilla.org/zh-CN/docs/Web/API/Document/querySelector

二、获取DOM对象

1. 根据CSS选择器来获取DOM元素（重点）

1.2 选择匹配的多个元素

语法：

```
document.querySelectorAll('css选择器')
```

参数：

包含一个或多个有效的CSS选择器 **字符串**

返回值：

CSS选择器匹配的NodeList 对象集合

例如：

```
document.querySelectorAll('ul li')
```

1. 获取一个DOM元素我们使用谁？
   - querySelector()
2. 获取多个DOM元素我们使用谁？
   - querySelectorAll()
3. querySelector() 方法能直接操作修改吗？
   - 可以
4. querySelectorAll() 方法能直接修改吗？ 如果不能可以怎么做到修改？
   - 不可以， 只能通过遍历的方式一次给里面的元素做修改

二、获取DOM对象

1. 根据CSS选择器来获取DOM元素（重点）

```
document.querySelectorAll('css选择器')
```

得到的是一个**伪数组**：

- 有长度有索引号的数组
- 但是没有 pop() push() 等数组方法

想要得到里面的每一个对象，则需要遍历（for）的方式获得。

注意事项

哪怕只有一个元素，通过querySelectAll() 获取过来的也是一个**伪数组**，里面只有一个元素而已

练习

请控制台依次输出 3个 li 的 DOM对象

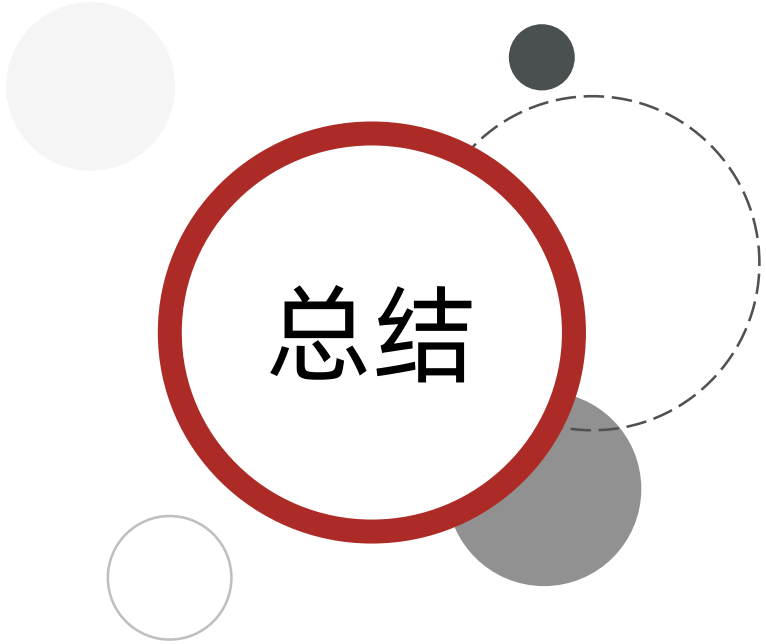
```
<ul class="nav">
    <li>我的首页</li>
    <li>产品介绍</li>
    <li>联系方式</li>
</ul>
```

二、获取DOM对象

2. 其他获取DOM元素方法（了解）

```
// 根据id获取一个元素
document.getElementById('nav')
// 根据 标签获取一类元素   获取页面 所有div
document.getElementsByTagName('div')
// 根据 类名获取元素   获取页面 所有类名为 w的
document.getElementsByClassName('w')
```

小结



1. 获取页面中的标签我们最终常用那两种方式？
   - querySelectorAll()
   - querySelector()
2. 他们两者的区别是什么？
   - querySelector() 只能选择一个元素， 可以直接操作
   - querySelectorAll() 可以选择多个元素，得到的是伪数组，需要遍历得到每一个元素
3. 他们两者小括号里面的参数有神马注意事项？
   - 里面写css选择器
   - **必须是字符串，也就是必须加引号**

目录

Contents

设置/修改DOM元素内容

- document.write() 方法
- 对象.innerText 属性
- 对象.innerHTML 属性

三、设置/修改DOM元素内容

目标：能够修改元素的文本更换内容

DOM对象都是根据标签生成的,所以操作标签,本质上就是操作DOM对象。

就是操作对象使用的点语法。

如果想要修改标签元素的里面的**内容**，则可以使用如下几种方式：

学习路径：

1. document.write() 方法
2. 对象.innerText 属性
3. 对象.innerHTML 属性

三、设置/修改DOM元素内容

目标：能够修改元素的文本更换内容

1. document.write()

- 只能将文本内容追加到 </body> 前面的位置
- 文本中包含的标签会被解析

举例说明

```
// 永远都只是追加操作，且只能位置 </body> 前
document.write('Hello World!');
document.write('<h3>你好，世界！ </h3>')
```

三、设置/修改DOM元素内容

目标：能够修改元素的文本更换内容

2. 元素innerText 属性

- 将文本内容添加/更新到任意标签位置
- 文本中包含的标签不会被解析

举例说明

```
// innerText 将文本内容添加 / 更新到任意标签位置
let info = document.getElementById('info')
// intro.innerText = '嗨~ 我叫李雷！'
info.innerText = '<h4>嗨~ 我叫李雷！</h4>'
```

三、设置/修改DOM元素内容

目标：能够修改元素的文本更换内容

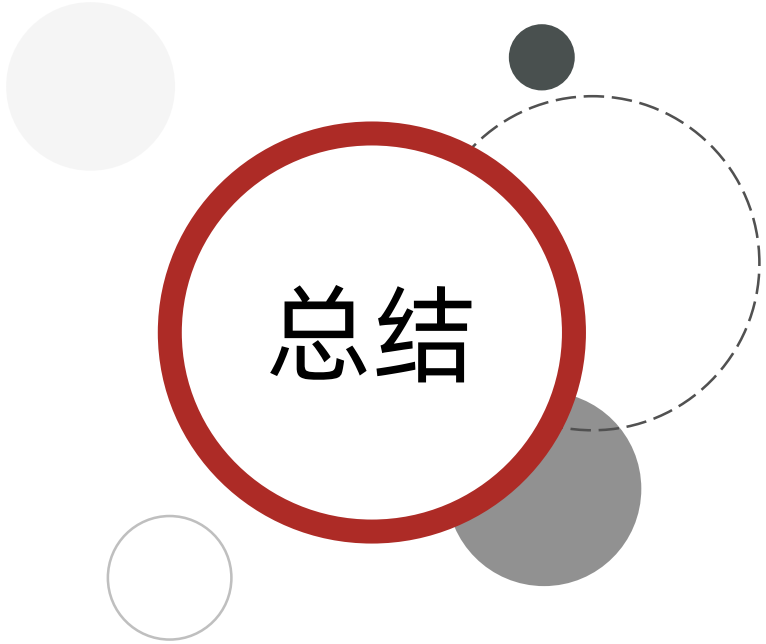
3. 元素.innerHTML 属性

- 将文本内容添加/更新到任意标签位置
- 文本中包含的标签会被解析

举例说明

```
// 2. innerHTML属性
box.innerHTML = '<h3>前端程序员<br>的头发都很多</h3>'
```

小结



1. 设置/修改DOM元素内容有哪3钟方式？
   - document.write() 方法
   - 元素.innerText 属性
   - 元素.innerHTML 属性
2. 三者的区别是什么？
   - document.write() 方法　只能追加到body中
   - 元素.innerText 属性　只识别内容，不能解析标签
   - 元素.innerHTML 属性　　能够解析标签
   - 如果还在纠结到底用谁，你可以选择innerHTML

三、设置/修改DOM元素内容

案例

随机抽取的名字显示到指定的标签内部

需求：将名字放入span 盒子内部

分析：

①：获取span 元素

②：得到随机的名字

③：通过innerText 或者 innerHTML 讲名字写入元素内部

目录

Contents

设置/修改DOM元素属性

- 设置/修改元素**常用**属性
- 设置/修改元素**样式**属性
- 设置/修改 **表单元素** 属性

四、设置/修改DOM元素属性

1. 设置/修改元素常用属性

- 还可以通过 JS 设置/修改标签元素属性，比如通过 src更换 图片
- 最常见的属性比如： href、title、src 等
- **语法：**

```
对象.属性 = 值
```

举例说明

```
// 1. 获取元素
let pic = document.querySelector('img')
// 2. 操作元素
pic.src = './images/b02.jpg'
pic.title = '这是一个图片'
```

四、设置/修改DOM元素属性

案例

页面刷新，图片随机更换

需求：当我们刷新页面，页面中的图片随机显示不同的图片

案例

页面刷新，图片随机更换

需求：当我们刷新页面，页面中的图片随机显示不同的图片

分析：

①：随机显示，则需要用到随机函数

②：更换图片需要用到图片的 src 属性，进行修改

③：核心思路：

1. 获取图片元素
2. 随机得到图片序号
3. 图片.src = 图片随机路径

设置/修改DOM元素属性

- 设置/修改元素**常用**属性
- 设置/修改元素**样式**属性
- 设置/修改 **表单元素** 属性

四、设置/修改DOM元素属性

2. 设置/修改元素**样式**属性

还可以通过 JS 设置/修改标签元素的样式属性。

- 比如通过 轮播图小圆点自动更换颜色样式
- 点击按钮可以滚动图片，这是移动的图片的位置 left 等等

学习路径：

1. 通过 style 属性操作CSS
2. 操作类名(className) 操作CSS
3. 通过 classList 操作类控制CSS

四、设置/修改DOM元素属性

2. 设置/修改元素**样式**属性

1.通过 style 属性操作CSS

语法：

```
对象.style.样式属性 = 值
```

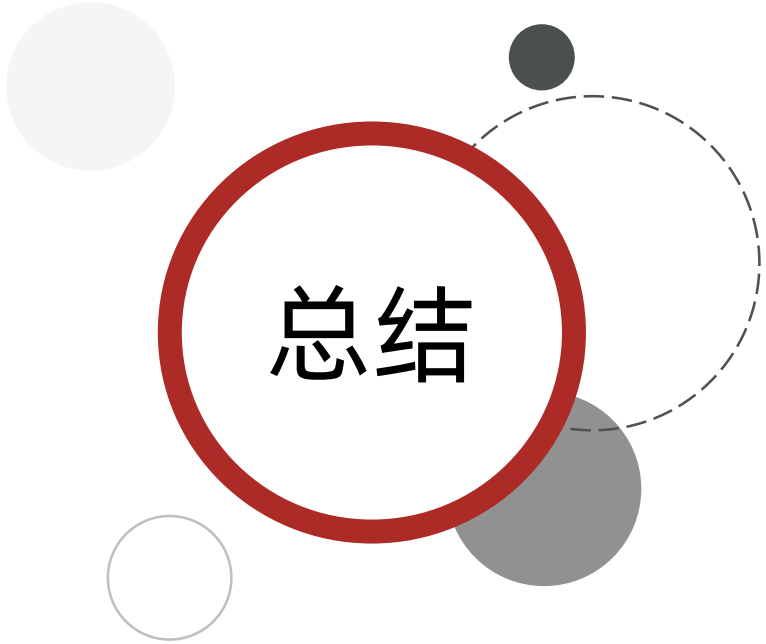
举例说明

```
let box = document.querySelector('.box')
// 2. 修改背景颜色
box.style.backgroundColor = 'red'
box.style.width = '300px'
box.style.marginTop = '50px'
```

注意：

1. 修改样式通过style属性引出
2. 如果属性有-连接符，需要转换为小驼峰命名法
3. 赋值的时候，需要的时候不要忘记加css单位

小结



1. 设置/修改元素样式属性通过_____style_____属性引出来？
2. 如果需要修改一个div盒子的样式，比如 padding-left, 如何写？
   - element.style.paddingLeft = '300px'
   - 小驼峰命名法
3. 因为我们是样式属性，一定别忘记，大部分数字后面都需要加单位

```
let box = document.querySelector('.box')
// 2. 修改背景颜色
box.style.backgroundColor = 'red'
box.style.width = '300px'
box.style.marginTop = '50px'
```

案例

页面刷新，页面随机更换背景图片

需求：当我们刷新页面，页面中的背景图片随机显示不同的图片

分析：

①： 随机函数

②： css页面背景图片 background-image

③： 标签选择body， 因为body是唯一的标签，可以直接写 document.body.style

四、设置/修改DOM元素属性

2. 设置/修改元素**样式**属性

2. 操作类名(className) 操作CSS

如果修改的样式比较多，直接通过style属性修改比较繁琐，我们可以通过借助于css类名的形式。

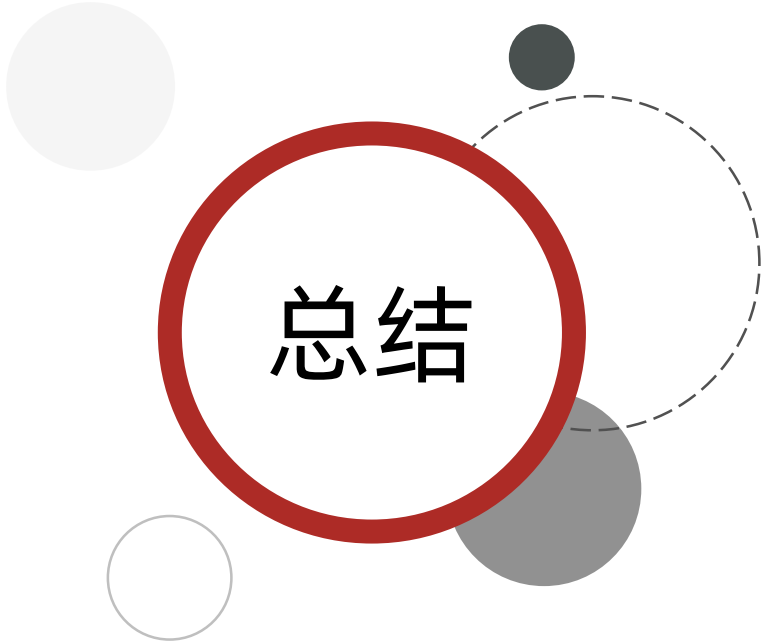
语法：

```
// active 是一个css类名
元素.className = 'active'
```

注意：

1. 由于class是关键字, 所以使用className去代替
2. className是使用新值换旧值, 如果需要添加一个类,需要保留之前的类名

小结



1. 使用 className 有什么好处?
   - 可以同时修改多个样式
2. 使用 className 有什么注意事项?
   - 直接使用 className 赋值会覆盖以前的类名

四、设置/修改DOM元素属性

2. 设置/修改元素**样式**属性

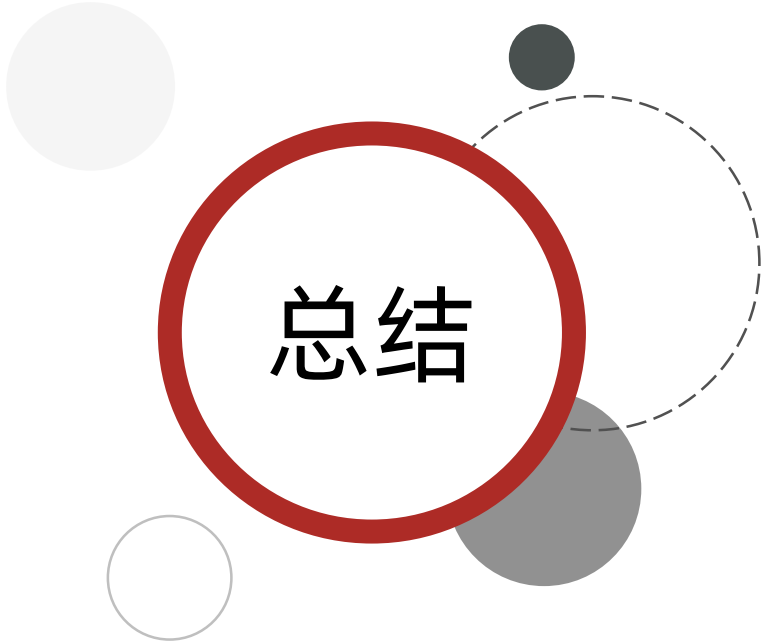
3. 通过 classList 操作类控制CSS

为了解决className 容易覆盖以前的类名，我们可以通过classList方式追加和删除类名

语法：

```
// 追加一个类
元素.classList.add('类名')
// 删除一个类
元素.classList.remove('类名')
// 切换一个类
元素.classList.toggle('类名')
```

小结



1. 使用 className 和classList的区别？
   - 修改大量样式的更方便
   - 修改不多样式的时候方便
   - classList 是追加和删除不影响以前类名

设置/修改DOM元素属性

- 设置/修改元素**常用**属性
- 设置/修改元素**样式**属性
- 设置/修改 **表单元素** 属性

四、设置/修改DOM元素属性

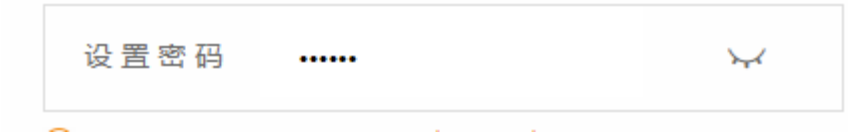
3. 设置/修改 表单元素 属性

表单很多情况，也需要修改属性，比如点击眼睛，可以看到密码，本质是把表单类型转换为文本框

正常的有属性有取值的 跟其他的标签属性没有任何区别

- 获取: DOM对象.属性名
- 设置: DOM对象.属性名 = 新值

```
表单.value = '用户名'
表单.type = 'password'
```



全选	商品	商家	价格
	小米手机	小米	¥1999
	小米净水器	小米	¥4999
	小米电视	小米	¥5999

四、设置/修改DOM元素属性

3. 设置/修改 表单元素 属性

表单属性中添加就有效果,移除就没有效果,一律使用布尔值表示 如果为true 代表添加了该属性 如果是false 代表移除了该属性

比如： disabled、checked、selected

全选	商品	商家	价格
☐	小米手机	小米	￥1999
☐	小米净水器	小米	￥4999
☐	小米电视	小米	￥5999

目录

Contents

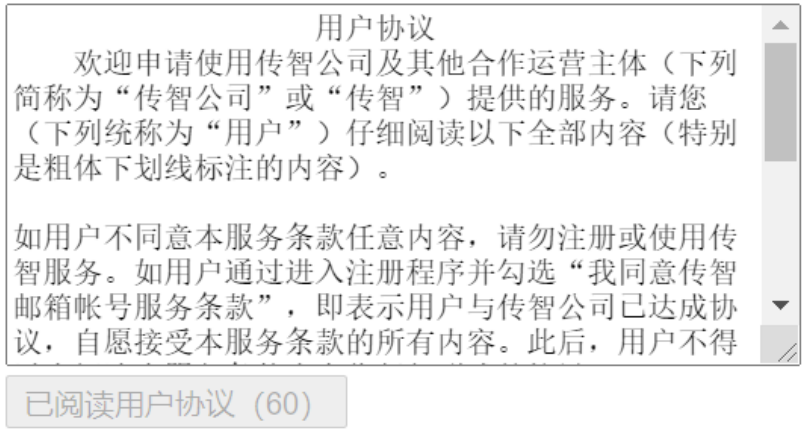
定时器-间歇函数

- 定时器函数介绍
- 定时器函数基本使用

五、定时器-间歇函数

目标：能够说出定时器函数在开发中的使用场景

- 网页中经常会需要一种功能：每隔一段时间需要自动执行一段代码，不需要我们手动去触发
- 例如：网页中的倒计时
- 要实现这种需求，需要定时器函数
- 定时器函数有两种，今天我先讲间歇函数



定时器-间歇函数

- 定时器函数介绍
- 定时器函数基本使用

五、定时器-间歇函数

目标：能够使用定时器函数重复执行代码

定时器函数可以开启和关闭定时器

1. 开启定时器

```
setInterval(函数, 间隔时间)
```

- 作用：每隔一段时间调用这个函数
- 间隔时间单位是毫秒

举例说明

```
function repeat() {
    console.log('前端程序员，就是头发多咋滴~~')
}
// 每隔一秒调用repeat函数
setInterval(repeat, 1000)
```

注意：

1. 函数名字**不需要加括号**
2. **定时器返回的是一个id数字**

五、定时器-间歇函数

目标：能够使用定时器函数重复执行代码

定时器函数可以开启和关闭定时器

2. 关闭定时器

```
let 变量名 = setInterval(函数, 间隔时间)
clearInterval(变量名)
```

注意：

1. 函数名字**不需要加括号**
2. **定时器返回的是一个id数字**

一般不会刚创建就停止，而是满足一定条件再停止

定时器函数小结

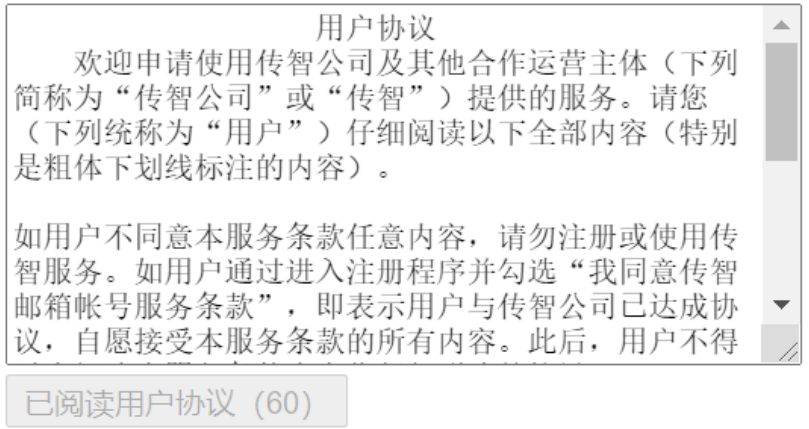
1. 定时器函数有什么作用？
   - 可以根据时间自动重复执行某些代码
2. 定时器函数如何开启？
   - setInterval(函数名, 时间)
3. 定时器函数如何关闭？

```
let 变量名 = setInterval(函数, 间隔时间)
clearInterval(变量名)
```

案例

倒计时效果

需求：按钮60秒之后才可以使用



分析：

①：开始先把按钮禁用（disabled 属性）

②：一定要获取元素

③：函数内处理逻辑

秒数开始减减

按钮里面的文字跟着一起变化

如果秒数等于0 停止定时器 里面文字变为 同意 最后 按钮可以点击

目录

Contents

案例

网页轮播图效果

需求：每隔一秒钟切换一个图片

分析：

①：获取元素（图片和文字）

②：设置定时器函数

设置一个变量++

更改图片张数

更改文字信息

③：处理图片自动复原从头播放

如果图片播放到最后一张就是第9张

则把变量重置为0

注意逻辑代码写到图片和文字变化的前面

今日复习路线

1. 整理今天笔记
2. 练习 随机变化背景图片案例
3. 练习用户倒计时协议案例
4. 重点练习自动播放轮播图案例
5. 最后作业

你生活的起点并不是那么重要，重要的是最后你能到达哪里

传智教育旗下高端IT教育品牌