DAY09 BOM 和 DOM

2023年4月29日 20:38

JavaScript的核心组成和BOM

一.JavaScript的核心组成

1.ECMAscript

ECMA：欧洲的计算机协会组织

定义了javascript的语法规范，描述了语言的基本语法和数据类型。

2.BOM：浏览器对象模型，有一套成熟的可以操作浏览器(和网页内容没有关系)的API(应用程序，通俗理解为说明文档)

3.DOM：文档对象模型，有一套成熟的可以操作页面元素的API，通过DOM可以操作页面中的元素(专门操作网页内容)

html

css

二.BOM的概述

1.BOM是浏览器对象模型，用于访问浏览器的功能，这些功能与任何网页内容无关。

2.目标：

- 获取一些浏览器的相关信息（窗口的大小）
- 操作浏览器进行页面跳转
- 获取当前浏览器地址栏的信息
- 操作浏览器的滚动条
- 浏览器的信息（浏览器的版本）
- 让浏览器出现一个弹出框（`alert` / `confirm` / `prompt`）
- ...

3.重点

BOM的核心就是 window对象

window是浏览器内置的一个对象，里面包含着操作浏览器的方法

window双重身份(两层函数)

3.1.window是ES里面的全局对象

3.2.window表示浏览器的一个窗口

获取浏览器窗口可视区的尺寸

可视区：浏览器区域可见的部分

1.这两个属性分别是用来获取浏览器窗口的宽度和高度（包含滚动条的）

window.innerHeight 和 window.innerWidth

```
// console.log(window.innerWidth, window.innerHeight);
```

2.浏览器窗口内部可视区的尺寸 - 重点

这两个方法分别是用来获取浏览器窗口的宽度和高度（**不包含滚动条的**）

document.documentElement.clientWidth
document.documentElement.clientHeight

```
// console.log(document.documentElement.clientWidth,
document.documentElement.clientHeight);
```

window: 浏览器的窗口

document: 文档区，浏览器内部显示网页内容的区域

3.下面的元素对象直接可以通过document获取。

document下面包含html,document是文档最大的对象　document -> html -> body,head

document.documentElement : html元素对象

document.body : body元素对象

document.head : head元素对象

document.title : title元素对象

浏览器卷去的尺寸

一.浏览器卷去的尺寸

document.documentElement.scrollTop : 滚动条离顶部的距离

document.documentElement.scrollLeft: 滚动条离左边的距离

```
// console.log(document.documentElement.scrollTop);

// 回到顶部
// document.onclick = function () {
//   document.documentElement.scrollTop = 0;
// }
```

二.浏览器滚动到

x 表示横向卷去的尺寸

y 表示纵向卷去的尺寸

注意: 必须传递两个数字, 一个报错，同时也可以传递对象做参数

window.scrollTo(x, y)

```
// document.onclick = function () {
//   window.scrollTo(0, 100);
// }
// document.onclick = function () {
//   window.scrollTo({
//     left: 0,//水平方向
//     top: 0, //垂直方向
//     behavior: 'smooth'  //行为，动画效果
//   });
// }
```

window常用的事件类型(可以理解成触发事件的动作)

1.onscroll：触发滚动条，执行对应的事件，属于高频事件(触发的事件次数比较大)

```
// var num = 0;
// window.onscroll = function () {
//   num++;
//   console.log(num);
// }

// 利用此事件不断获取滚动条离顶部的距离
// window.onscroll = function () {
//   console.log(document.documentElement.scrollTop);
// }
```

2.onresize:改变浏览器窗口大小(改变可视区)触发，高频事件

```
// window.onresize = function () {
//   console.log('窗口改变了');
// }
```

3.onload:页面加载完成触发事件(加载指的是页面的结构，内容，图片等)

使用此事件将js代码放置在页面的任何地方，总是要等到其他内容加载完成后触发。

```
<head>
 <style>
   * {
     padding: 0;
     margin: 0;
   }
   .box {
     height: 200px;
     width: 1000px;
     margin: 0 auto;
     background-color: #abcdef;
     font-size: 100px;
     line-height: 200px;
     text-align: center;
   }
   .box1 {
     height: 3000px;
     width: 1000px;
     margin: 0 auto;
     background-color: pink;
   }
 </style>
 <script>
  // 报错，走到这里根本不知道header是什么，header在其下面声明的。
  // console.log(header.innerHTML);
  // 这里的代码等到页面加载完成才触发
   window.onload = function () {
```

```
      console.log(header.innerHTML);//我的header里面的内容
    }
  </script>
</head>

<body>
  <div class="box">我是头部内容</div>
  <div class="box1"></div>
  <div id="header">我的header里面的内容</div>
</body>
```

window下面的方法

window.alert
window.prompt
window.confirm
window.parseInt
window.parseFloat
window.setInterval
window.clearInterval
window.setTimeout
window.clearTimeout

1.open和close

open可以打开某个页面

```
// document.onclick = function () {
//   window.open('http://www.jd.com', '_blank');
// }
// document.onclick = function () {
//   window.open('./close.html', '_blank');
// }
```

close关闭某个页面(本意是关闭open打开的页面)

```
// document.onclick = function () {
//   window.close();
// }
```

地址栏

一.地址栏的组成 - 共6部分 - 非常重要

例如:完整的url地址
http://10.31.1.10:80/about/index.html?name=zhangsan&age=18#about

1.协议(http/https):两台电脑要通信，必须使用协议，很多浏览器自动补上协议

http:超文本传输协议 默认端口：80

https:超文本传输协议(加密传输，利用SSL层) 默认端口：443

2.ip地址：ip地址是电脑的唯一标识，相当于人的身份证，因为ip地址记忆不方便，产生了域名(网址，需要注册)

3.端口：利用端口访问服务器里面具体的程序

4.路径：访问具体目录里面内容

5.携带的参数(数据)：地址栏后面通过?和&符号携带一些数据

6.哈希值：#后面的值，主要用来实现单页面跳转

二.客户端和服务器端

两台电脑在通信

浏览器端/客户端/用户端/前端：访问程序的一端

服务器端/后端：服务器就是一台昂贵的电脑，提供程序源码的电脑

端口：如果服务器端有三套程序，一个ip地址无法对于三套程序，利用端口进行划分，端口称之为访问程序的门。

三.客户端和服务器端通信的过程

查看提供的图片,提供通信的基本过程

第一步：输出域名

第二步：通过域名获取对应的ip地址 域名解析(域名和ip进行绑定)http://www.baidu.com <==> http://202.108.22.5:80

第三步：进入服务器，根据对应的端口，找到对应的程序。

第四步：通过路径访问程序的源码

第五步：服务器解析源码，返回给浏览器，浏览器解析出用户最终需要的信息。

window的子对象 - location对象 - 重点

1.理解一个完整地址栏(URL)的组成

2.href属性:获取和设置浏览器地址栏的内容

```
// console.log(location.href);//获取
// location.href = 'http://www.jd.com';//设置
// 案例:
// window.setTimeout(function () {
//    location.href = 'http://www.jd.com';//设置
// }, 3000)
```

3.search属性:获取地址栏上面的数据,包括问号(即问号后面的字符串，如果多条数据中间用 & 符号拼接)

https://suggest.taobao.com/sug?k=1&area=c2c&q=aaaa&code=utf-8&ts=1682661536027&callback=__jp2

4.hash属性:获取地址栏上面的hash值，#号后面的内容，包括#

location.hash获取地址栏

5.reload(true)方法 刷新页面,如果参数为true,通过缓存刷新。

```
// alert(1);
// location.reload();
// location.reload(true);//通过缓存刷新,第二次提升性能，页面部分内容进入缓存。
```

F5通常只是刷新本地缓存;

Ctrl+F5可以把internet临时文件夹的文件删除再重新从服务器下载，也就是彻底刷新页面了

window子对象 - history对象

浏览者通常可以使用浏览器的前进与后退按钮访问曾经浏览过的页面。

JS的history对象记录了用户曾经浏览过的页面，并可以实现浏览器前进与后退相似的导航功能。

history.go()方法-- 前进或后退指定的页面数（负数后退，正数前进）

history.length 属性-- history对象中缓存了多少个URL

history.forward();//前进一个页面

history.back();//后退一个页面

注意：在一个浏览器页面中使用，开启的新窗口或者新的页面(_blank)就不能这样操作。

本地存储 - 非常重要

1.localStorage的相关操作

1.1.概念

在HTML5中，新加入了一个localStorage对象，这个对象主要是用来作为本地存储来使用的，即能够在用户浏览器中对数据进行本地的存储。

1.2.本地存储localStorage的特点

localStorage永久存储，除非手动删除

localStorage会将5M大小的数据直接存储到本地，但只有在高版本的浏览器中才支持的。

目前所有的浏览器中都会把localStorage的值类型限定为string类型。

不同的浏览器本地存储不能相互进行访问

1.3.读(获取)写(设置)及删除操作

存

语法格式：window.localStorage.setItem(key,value) - 写入本地存储

key:自定义存储的名称 value:要存储的值

注意：key值相同会覆盖

```
// localStorage.setItem('username', 'zhangsan');
```

提前了解存储对象

JSON.stringify()：JSON(数据格式)对象下面的方法，将对象转换成字符串格式

```
// var obj = {
//    name: 'zhangsan',
//    age: 18,
//    sex: '男'
// }
// localStorage.setItem('student', JSON.stringify(obj));
// localStorage.setItem('data', [1, 2, 3, 4, 5]);
// localStorage.setItem('data', 'wangwu');
```

取

语法格式：window.localStorage.getItem(key)

如果获取的key不存在，输出null

```
// console.log(localStorage.getItem('username123'));
```

删

语法格式：window.localStorage.removeItem(key); 将localStorage中的某个键值对删除

```
// localStorage.removeItem('username');
```

window.localStorage.clear(); 将localStorage的所有内容清除
localStorage.clear();

如何查看：
进入浏览器控制面板 - application - 左侧面板 - localStorage

2.了解sessionStorage的相关操作

sessionStorage存储方式以及特点和localStorage是一样的

区别是localStorage是永久存储，sessionStorage临时存储,会话结束就消失(关闭浏览器)

```
// sessionStorage.setItem('name', 'wangwu');
// console.log(sessionStorage.getItem('name'));
// sessionStorage.removeItem('name');
```

DOM的概述

DOM: 文档(document)对象模型，有一套成熟的可以操作页面元素的API，通过DOM可以操作页面中的元素(专门操作网页内容)

其实就是操作 html 中的标签的一些能力

标签(tagName)：元素(element)，元素对象(elementObject)，节点(node)，DOM对象

DOM 的核心对象就是 docuemnt 对象(html文档内部最大的对象)

document 对象是浏览器内置的一个对象，里面存储着专门用来操作元素的各种方法

DOM元素的获取 - 查

```
<body>
  <div class="box">div1</div>
  <div id="header">div2</div>
  <div class="list">
    <ul>
      <li>11111111111111</li>
      <li>22222222222222</li>
      <li>333333333333333</li>
```

```
      <li>4444444444444444444</li>
      <li>5555555555555555555</li>
      <li>6666666666666666666</li>
    </ul>
  </div>
  <div class="footer">footer1</div>
  <div class="footer">footer2</div>
  <div class="footer">footer3</div>
</body>
```

1.querySelector()通过选择器获取元素，如果获取多个只返回第一个 - 重点

解读：参数就是添加样式的选择器(id,class,标签，后代，伪类...)

注意：一次只能获取一个元素对象。

```
// console.log(document.querySelector('box'));
// console.log(document.querySelector('#header'));
// console.log(document.querySelector('.list > ul'));
// console.log(document.querySelector('.list ul li'));//返回第一个
```

2.querySelectorAll()通过选择器获取元素，可同时获取多个元素，返回类数组 - 重点

```
// console.log(document.querySelectorAll('.list ul li'));//类数组，[li, li, li, li, li, li]
// console.log(document.querySelectorAll('.list ul li')[3]);
// var elements = document.querySelectorAll('.list ul li');
// console.log(elements.length);
// console.log(elements[0]);
```

3.getElementById()获取特定 ID 元素的节点

```
// console.log(document.getElementById('header'));
// console.log(header);
```

4.getElementsByTagName()获取相同元素的节点列表，返回类数组，使用[0]来获取

```
// console.log(document.getElementsByTagName('div'));
// console.log(document.getElementsByTagName('div')[2]);
// console.log(document.querySelectorAll('div'));
// console.log(document.querySelectorAll('div')[2]);
```

5.getElementsByClassName()获取相同类名的节点列表（IE8及以下不支持），返回类数组

```
// console.log(document.getElementsByClassName('footer'));//获取三个
```

给多个元素添加事件

```
// var list = document.querySelectorAll('.list> ul >li');
// for (var i = 0; i < list.length; i++) {
```

```
//   list[i].onclick = function () {
//     console.log('事件触发了');
//   }
// }
```

注意：forEach是数组的方法，但是元素对象也可以使用。

```
// list.forEach(function (item, index, array) {//item:元素对象
//   item.onclick = function () {
//     console.log('111111111111');
//   }
// })
```