Day28笔记

|  |
| --- |
| BOM  1. 什么是BOM  BOM（浏览器对象模型）是用于控制浏览器窗口及其各项功能的JavaScript API集合。它包含了一系列对象，能够访问和操作浏览器的功能，如窗口的大小、位置、导航历史、用户屏幕信息以及网络请求等。  alert("Hello, World!"); // 弹出提示框  var width = window.innerWidth; // 获取窗口的内部宽度  var location = window.location.href; // 获取当前页面的URL  BOM的核心是`window`对象，它在浏览器中既是全局对象，又是浏览器窗口的接口。通过BOM，我们可以实现许多与浏览器交互的操作，如弹出窗口、页面重定向、获取屏幕分辨率等。BOM虽然功能强大，但由于其各浏览器实现存在差异性，使用时需注意兼容性问题。  2. 常见的BOM对象与方法  2.1 window 对象  `window`对象是BOM的核心，表示浏览器窗口或框架。它提供了浏览器窗口相关的属性和方法。  console.log(window.name); // 获取窗口的名称  window.open('https://www.example.com', '\_blank'); // 在新窗口打开链接  通过`window`对象，我们可以获取和操作浏览器窗口的属性，如名称、大小、位置等。`window`对象还与ECMAScript中的全局对象紧密关联，因此全局变量和函数也是`window`对象的属性和方法。  2.2 navigator 对象  `navigator`对象包含了浏览器的信息，如应用程序的名称、版本号、用户代理等。  console.log(navigator.userAgent); // 输出浏览器的用户代理字符串  `navigator`对象让我们能够检测浏览器的信息，对于实现不同浏览器的兼容性处理非常有用。特别是在处理用户代理（user agent）时，可以根据不同浏览器的特性加载不同的脚本或样式。  2.3 location 对象  `location`对象提供了当前文档的位置信息（URL），并允许重定向页面。  console.log(location.href); // 输出当前页面的URL  location.href = "https://www.newpage.com"; // 重定向到新的URL  `location`对象是处理页面导航的核心对象，通过它可以轻松获取当前URL信息并执行页面跳转操作。它在实现动态导航和单页应用程序（SPA）的路由管理时尤为重要。  2.4 screen 对象  `screen`对象包含有关用户屏幕的信息，如分辨率、颜色深度等。  console.log(screen.width); // 获取屏幕的宽度  console.log(screen.height); // 获取屏幕的高度  `screen`对象主要用于处理与用户设备屏幕相关的操作，例如根据屏幕分辨率调整页面布局。虽然它的使用场景较为有限，但在响应式设计中非常实用。  2.5 history 对象  `history`对象用于操作浏览器的会话历史，可以实现页面的前进、后退功能。  history.back(); // 返回前一个页面  history.forward(); // 前进到下一个页面  `history`对象的前进、后退功能使得我们能够更好地控制用户的浏览体验，特别是在单页应用中，通过`pushState`和`replaceState`等方法还能实现浏览器地址栏的动态更新而不刷新页面。  3. 兼容性与BOM的注意事项  虽然BOM提供了强大的功能，但由于各浏览器厂商在实现上存在差异，部分属性和方法在不同浏览器中的支持情况并不完全一致。  在使用BOM时，应始终考虑到兼容性问题。可以通过检测特性是否存在来确保代码的健壮性，例如使用`if`语句检测`window.screen`对象是否存在。了解并应对不同浏览器之间的差异，对于编写跨平台兼容性好的代码至关重要。  通过这次学习，我对BOM的功能和应用有了更深入的理解。BOM在JavaScript编程中提供了非常重要的功能，使得我们能够更好地与浏览器交互。在实际开发中，合理使用BOM能够极大提升用户体验，但也需要注意浏览器兼容性问题，并在必要时进行相应的处理。  日期2024.8.22 |

Day29笔记

|  |
| --- |
| AJAX学习笔记  1. 什么是AJAX？  AJAX（Asynchronous JavaScript And XML）是一种用于创建交互式网页应用的开发技术，旨在通过异步数据传输（HTTP请求）来增强网页与服务器之间的数据交换。其特点是能够在不重新加载整个页面的情况下，对页面的部分内容进行更新。  AJAX技术是现代Web开发中的核心部分。它的异步特性使得用户能够在不感知页面刷新的情况下进行操作，提高了用户体验。尽管名称中包含XML，但实际应用中JSON更为普遍，因为它更轻量且易于解析。  2. HTTP请求与响应  HTTP（超文本传输协议）是Web浏览器与服务器之间通信的基础。AJAX通过JavaScript对HTTP的请求和响应进行操作，从而实现数据的动态加载。  3. XMLHttpRequest对象  `XMLHttpRequest`对象是实现AJAX功能的关键，它用于在后台与服务器交换数据。常用的方法包括：  `open(method, url, async)`：设置请求类型（如GET或POST）、目标URL以及是否异步。  `setRequestHeader(header, value)`：设置HTTP请求头。  `send(data)`：发送请求数据。  var request = new XMLHttpRequest();  request.open('GET', 'https://api.example.com/data', true);  request.send();  request.onreadystatechange = function() {  if (request.readyState === 4 && request.status === 200) {  console.log(request.responseText);  }  };  `XMLHttpRequest`提供了一个简洁的接口用于与服务器进行通信，尤其是在处理异步请求时，它避免了页面的阻塞。然而，随着Fetch API的引入，`XMLHttpRequest`的使用逐渐减少，但它仍然是理解AJAX的基础。  4. AJAX的封装  为了简化AJAX的使用，通常会对其进行封装。  var myAjax = {  get: function(url, params, success, error) {  var httpRequest = new XMLHttpRequest();  httpRequest.open('GET', url + "?" + new URLSearchParams(params).toString(), true);  httpRequest.send();  httpRequest.onreadystatechange = function() {  if (httpRequest.readyState === 4 && httpRequest.status === 200) {  success(JSON.parse(httpRequest.responseText));  } else if (httpRequest.readyState === 4) {  error(httpRequest.responseText);  }  };  }  };  封装AJAX有助于提高代码的可复用性和可维护性。通过封装，我们可以统一处理请求的细节，减少冗余代码，同时也可以更容易地应对跨浏览器的兼容性问题。  5. 实战操作：GET与POST请求  GET请求  var request = new XMLHttpRequest();  request.open('GET', 'https://api.example.com/data?param=value', true);  request.send();  POST请求  var request = new XMLHttpRequest();  request.open('POST', 'https://api.example.com/data', true);  request.setRequestHeader('Content-type', 'application/json;charset=UTF-8');  request.send(JSON.stringify({key: 'value'}));  GET请求适用于获取数据且不会改变服务器状态的操作，POST则更适合提交数据。理解两者的区别并合理选择，可以提高请求的效率与安全性。  总结  AJAX通过异步数据传输大大提升了Web应用的交互性，理解并熟练掌握其基本原理和常用操作是前端开发的重要技能。通过对`XMLHttpRequest`的深入理解和灵活应用，可以实现更加高效和友好的用户体验。  日期2024.8.23 |