Day27笔记

|  |
| --- |
| JavaScript事件与继承  一、JavaScript事件  JavaScript事件是浏览器和用户之间互动的核心机制。事件由三部分组成：事件源、事件类型、事件处理程序，这三者共同构成了事件的完整过程。  1.事件的三要素  事件源：触发事件的HTML元素，如按钮或文本框。  事件类型：事件的类型，如`click`（点击）、`mouseover`（鼠标悬停）等。  事件处理程序：响应事件的JavaScript函数。  2.事件流  事件捕获：事件从最不具体的元素向最具体的元素传播。类似将点击事件从`document`传递到目标元素。  事件冒泡：事件从最具体的元素向外层元素传播，直到`document`为止。它是事件的默认传播方式。  3.事件绑定示例  <button id="btn">点我啊</button>  <script>  var btn = document.getElementById("btn");  btn.onclick = function() {  alert("按钮被点击了！");  }  </script>  这段代码展示了如何为一个按钮绑定点击事件。当按钮被点击时，会弹出一个提示框。绑定事件的方式非常直观，直接通过`onclick`属性将函数赋值给事件处理程序。  4.事件冒泡的应用  在复杂的网页中，利用事件冒泡机制，可以在父元素上统一管理子元素的事件处理，而不需要为每个子元素单独添加事件。这不仅减少了代码的冗余，还提高了性能。  二、JavaScript继承  JavaScript中的继承是面向对象编程中的一个重要概念。主要通过\*\*原型链\*\*实现。  1.原型链  概念：每个构造函数都有一个原型对象，这个原型对象又指向另一个构造函数的原型对象，形成了一条链。实例对象可以通过这条链继承属性和方法。  示例：  function Animal() {  this.type = "animal";  }  Animal.prototype.getType = function() {  console.log(this.type);  }  function Dog() {  this.name = "dog";  }  Dog.prototype = new Animal();  var myDog = new Dog();  myDog.getType(); // 输出 "animal"  `Dog`继承了`Animal`的属性和方法。当调用`myDog.getType()`时，会输出`Animal`中的`type`属性。  2.组合继承  概念：结合原型链和构造函数两者的优点，使用原型链继承方法，使用构造函数继承属性。  function Animal(name) {  this.name = name;  }  Animal.prototype.sayName = function() {  console.log(this.name);  }  function Dog(name, age) {  Animal.call(this, name);  this.age = age;  }  Dog.prototype = new Animal();  Dog.prototype.sayAge = function() {  console.log(this.age);  }  var myDog = new Dog("Buddy", 5);  myDog.sayName(); // 输出 "Buddy"  myDog.sayAge(); // 输出 "5"  `Dog`不仅继承了`Animal`的属性`name`，还添加了自己的属性`age`和方法`sayAge`。这种继承方式能够最大化代码复用，并确保每个实例都拥有独立的属性。  3.继承的注意事项  当使用原型链继承时，需要注意所有实例共享同一个原型对象。这意味着一个实例的改动会影响其他实例。因此，在构造函数中定义实例属性，而在原型中定义共享方法是更合理的选择。  日期2024.8.21 |