html css javascript（js）

内容 表现 行为

什么是js

JavaScript ==> JS script(脚本语言，有逻辑，要运算)

css使用：行 内嵌 外链

js使用：同上

js运行顺序，默认是从上到下，从左到右的！(重点！)

js基本命令：

|  |  |
| --- | --- |
| alert() | 弹窗 |
| console.log() | 控制台 |
| document.write() | Dom输出 |
| document.getElementById() |  |

js数据类型：

|  |  |
| --- | --- |
| 数值型数据 | 直接输入的数字56、5.20.23.... |
| 布尔值数据 | 只有两种结果，正确或错误 true/false |
|  | undefined/null/NaN表示非数字 |

Js变量的作用，命名规则

|  |
| --- |
| js变量：记录和（在内存中）存储数据用的 |
| 格式: var 变量名=值; |
| 变量名不能数字开头，区分大小写，可以中文（最好不要用） |
| 等号在这里不是等于，它是一个赋值过程，等于判断要用”123” |
| 变量可以只声明，但不赋值，得到undefined结果 |

Js运算

|  |  |
| --- | --- |
| 比较运算 | > >= < <= == === |
| js基本运算 | + - \* / %（求余数） |
| 逻辑运算 | && || ! |
| 注意：全等于，不只是值相等，且数据类型也要一样 | |
| 三元运算：if...else...的简写 | |
| 自增自减运算：++/-- (谁在前面就先运算谁) | |
| 符号在前面，就先进行符号自增运算，再使用运算后的变量值 | |
| 变量在前面，就先使用变量的值，使用完后在进行符号自增运算 | |

If走下情形

js中 1==true 0==false null==undefined 这些都是能成立的

Js流程控制

|  |  |
| --- | --- |
| 顺序结构 | 从上到下，从左到右的顺序运行 |
| 分支结构 |  |
| 循环结构 |  |

分支结构

|  |  |
| --- | --- |
| 单路分支 | If(条件式){代码段} |
| 双路分支 | If(条件式){代码段1}else{代码段2} |
| 多路分支 | If(条件式1）{代码段2}else if(){}else{} |
| 多路分支 | Switch(条件式){case 1:代码段1 break; default:代码段n;.....} |

循环结构

|  |  |
| --- | --- |
| for循环 |  |
| while循环 |  |
| do-while循环 |  |
| continue | 停止后面代码运行，直接进入下一次循环 |
| break | 停止后面代码运行，跳出整个循环 |
| reture | 函数返回值，碰到return后，函数将会停止，里面的代码将不会继续执行下去 |

Js函数

|  |  |
| --- | --- |
| 通过function关键字去告诉浏览器，这一段代码被承包，别人不能用 | |
| 格式 | Function函数名(){代码段} |
| 使用 | 只有通过调用，才能使用函数里面的代码 |
| 注意 | 调用时一定要带括号，不带括号就表示函数本身的代码，而带上括号就是执行里面的代码 |
| 命名规则 | 函数名和变量命名规则一样（函数即变量） |

事件

|  |  |
| --- | --- |
| 事件 | 指网页交互过程中，响应用户操作而产生的一些反馈程序（js特效） |
| 由什么产生 | 按钮，表单，滚动条，图片 |
| 怎么产生的 | 点击，经过，滑动，拖拽，触屏 |
| 发生了什么 | 弹框出来，文字变色，盒子隐藏展开 |
| This关键字 | 指当前绑定这个事件的元素 |
| onfocus | 获得焦点 |
| onblur | 失去焦点 |
| window.onload | 页面加载完毕后触发 |
| window.onscroll | 窗口滚动条发生变化时触发 |
| Window.onmouseleave | 离开时触发 |
| Window.onresize | 串口尺寸发生变化是触发 |
|  |  |

实参与形参

|  |  |
| --- | --- |
| 形参 | 形式上的参数，并不参与运算，在函数运行过程中占个位置，给后面调用时的实参使用 |
| 实参 | 括号里面的参数，实际上参与运算的东西 |

作用域

|  |  |
| --- | --- |
| 作用域 | 使用范围 |
| 全局变量 | 能在所有范围里面使用的变量 |
| 局部变量 | 只能在少部分里面使用的变量，只能在当前作用域生效，即用即消 |
|  |  |
|  |  |
| 注意 | 函数运行时会先进行预读检索关键字var function  但因为只是预读，并没有真正开始运行，所以这个时候的变量我们还不知道 |

函数

|  |  |
| --- | --- |
| 自运行匿名函数 | 自动运行 |
| 函数闭包 | 让我们能在外面使用里面的局部变量， |
| 递归函数 |  |
| 注意 | 闭包函数中的局部变量将记录在内存中，销毁不掉，要少用 |

//自运行匿名行数

(function(){

console.log("1234165789")

var dfg=789;

})()

!function(){

console.log("1234165789")

var dfg=789;

}()

//console.log(d);调用不了函数里面的局部变量

//完整的闭包过程

function open(){

var m=1;

return function(){

return m=m+1;

}

}

var n=open();

//实参组

function arg(){

console.log(arguments);//输出实参数

}

arg("a",123,true);

//递归函数

function fire(val){

if(val==1){return 1}

return val\*fire(val-1)

}

fire(5);

响应式布局

window.onresize = function() {

document.documentElement.style.fontSize=document.documentElement.clientWith / 720 \* 100 + "px"

}

<!--冒泡排序-->

<script>

var arry = [12, 13, 45, 0, 21, 3, 8];

for(var i = 0; i < arry.length; i++) {

for(var j = 0; j < arry.length; j++) {

/\*从大到下排序\*/

if(arry[j] < arry[j + 1]) {

var k = arry[j];

arry[j] = arry[j + 1];

arry[j + 1] = k;

}

}

}

Sort高级用法

function sortNumber(a, b){

return a - b;//a-b 小到大 b-a大到小

}

var j=[5,2,3,7,9,1,154,454,16,3476,484,78,11.12];

//console.log(j.sort(sortNumber));

j.sort();

console.log(j.reverse())

//字符串首字母大写 hello tim my name is xxx

var str="hello tim my name is xxx";

var arr=str.split(" ");

var arry2=[];

for(var i=0;i<arr.length;i++){

arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].slice(1);

}  
对象

多个变量的集合（通过关联属性去调取）

传址

var arr1=[1,2,3];

var arr2=arr1;

arr2[2]=5;

传值

var a=6;

var b=a;

b=3;

对象单元的使用

通过关联属性去调取 对象名["属性名"] 或者 对象名.属性名

obj1.fn(666)//执行了对象里面的函数

内置对象

浏览器为我们网页已经定义好了的函数，我们只要拿过来用就行

内置函数：

浏览器为我们网页已经定义好了的函数，我们只要拿过来用就行

对象的长度无法读取

(内置对象：已经定义好了的对象，我们只要拿过来用就行)

对象单元的遍历

(通过for...in...循环结构遍历对象，不能通过对象名.属性名去读取，得用对象名["属性名"])

for(var i in obj1){

console.log(obj1[i])

}

Typeof 获取数据的类型

parseInt（“df”） 转换数据类型

Math函数

|  |  |
| --- | --- |
| Math.floor | 向下取整 |
| Math.ceil() | 向上取整 |
| Math.min |  |
| Math.max |  |
| Mth.pow(10,-43) | 宁次放 |
| Math.random |  |
| Math.round | 四舍五入 |
| Math.sqrt | 开平方根 |

数组函数

|  |  |  |
| --- | --- | --- |
| arr3.slice(3,-1) | 返回数组的一个片段 |  |
| arry.splice(random,1) | 从数组中删除元素 splice(0)全删除 |  |
| arr.pop() | 删除最后个一个元素并返回刻值 |  |
| arr.push | 向数组中添加一个新元素，返回数组的新长度 |  |
| arr.shift() | 删除数组中的第一个元素并返回 |  |
| arr.unshift(a) | 返回一个数组，在刻数组头部插入指定的元素 |  |
| arr.concat(a,b,c) | 一个由两个或多个数组合并组成的新数组 |  |
| arr.contains(a) | 数组中包含某个元素 |  |
| join("") | 通常结合 splite字符分割 数组转成字符串 |  |

字符串

|  |  |
| --- | --- |
| str.length |  |
| str.indexOf(string) | 返回括号里面参数在字符串中的索引值位置 |
| str.toUpperCase() |  |
| str.toLowerCase() |  |
| str.substring(2,2)=0 | 复制字符串片段，从哪个索引值num1开始，到哪个索引值num2结束 |
| str.replace("World","Tim") |  |
| str.substr(2,4) = 234 |  |
| str.split("o") | 用括号里面的参数来分割字符串，形成数组  复制字符串片段，从哪个索引值num1开始，到后面的几个num2字母，  num1如果为负数，则反向从1开始num2如果不设置，则默认复制到结尾 |

创建对象三种方式

|  |
| --- |
| js对象：多个变量的集合（通过关联属性去调取）  对象创建（通过键值对赋值。 键:值 ） |

var obj1={

num:1,

str:"你好",

bool:true,

length:["a","b"],

fn:function(x){

alert(x)

},

obj:{name:"张三",age:19}

}

//对象第二种创建方式

var obj2=new Object({num:1});

//对象的第三种创建方式

var obj3=new Object();

obj3.name="李四"

obj3.age=20

事件

文档或浏览器窗口中发生的一些特定交互瞬间。我们可以通过监听器（或者处理程序）预定事件，以便事件发生的时候执行相应得代码

事件流

元素产生一个事件时，此事件会在元素节点与根节点之间的路径传播，路径所经过的节点都会受到此事件，这个传播过程称成dom事件流

捕获阶段

当我们在 DOM 树的某个节点发生了一些操作（如

单击、鼠标移动上去），就会触发一个事件。这个事件从 Window

发出，不断向下传递经过子级节点直到目标节点。

在到达目标节点之前的过程，就是捕获阶段（Capture Phase）。

目标阶段

当事件不断传递，传递到了事件触发目标节点上，最终在目标节点上触发这个事件，就是目标阶段（Target Phase）。

冒泡阶段

当事件传递到达目标节点之后，将会沿着原路返回，由于这个过程类似水泡从底部冒到顶部，所以称作冒泡阶段（Bubbling Phase）。

事件委托

事件委托主要利用了事件冒泡，只指定一个事件处理程序，就可以管理一个类型的所有事件。

Event兼容性

var e=event||window.event;

键盘兼容性

key=e.keycode||e.which||e.charcode

事件监听

btn[0].addEventListener("click",function(){

console.log(2222)

},true);//冒泡

event事件对象，详细记录了事件发生时的一些相关数据（类似黑夹子）

console.log(ev.button)//鼠标按键 0表示左键

console.log(ev.clientY)//视口位置 （固定定位）

console.log(ev.offsetY)//容器位置 （绝对定位以容器为参照物）

console.log(ev.screenY)//屏幕位置 （）

console.log(ev.pageY)//页面位置 （绝对定位以页面为参照物）

阻止冒泡 阻止默认行为 阻止菜单事件 事件委托

Event.stopPropagation();

Event.preventDefault();

element.oncontextmenu=function(){

Event.preventDefault;

}

var target = e.srcElement ? e.srcElement : e.target;

最底层直接接受事件的目标dom元素

事件委托

var ul=document.getElementsByTagName("ul")

ul[0].onclick=function(event){

var e=event||window.event

var target = e.srcElement ? e.srcElement : e.target;

console.log(target)//它永远是最底层直接接受事件的目标DOM元素

}

事件

|  |  |
| --- | --- |
| onclick | 鼠标点击时触发此事件 |
| ondblclick | 鼠标双击时触发此事件 |
| onmousedown | 按下鼠标时触发此事件 |
| onmouseup | 按下鼠标后松开鼠标触发此事件 |
| onmousemove | 当鼠标移动时触发此事件 |
| onmouseover | 当鼠标移动到某对象范围的上方时触发此事件 |
| onmouseout | 当鼠标离开某对象范围时触发此事件 |
| onkeypress | 当键盘上某个键被敲下并释放时触发此事件 |
| onkeydown | 当键盘上某个键被敲下时触发此事件 |
| onkeyup | 当键盘上某个键被释放时触发此事件页面相关事件 |
| onabort | 图片在下载时被用户中断 |
| onbeforeunload | 当前页面的内容将要被改变时触发此事件 |
| onerror | 出现错误时触发此事件 |
| onload | 页面内容完成时触发此事件 |
| onresize | 当浏览器的出口大小被改变时触发此事件 |
| onscroll | 当浏览器的滚动条位置发生变化时触发此事件 |
| onstop | 当浏览器的停止按钮被按下时或这在下载的文件被中断时触发此事件 |
| onunload | 当前页面将被改动时触发此事件表单相关事件 |
|  |  |
|  |  |

onblur 当前元素失去焦点时触发此事件

onfocus 当某个元素获得焦点时触发此事件

onchange 当前元素失去焦点并且被改变时触发此事件

onreset 当表单中reset的属性被激发时触发的事件

onsubmit 当表单被提交时触发的事件滚动字幕事件

onbounce 在marquee内的内容移动至marquee显示范围之外时触发此事件

onfinish 当marquee完成需要显示的内容后触发此事件

onstart 当marquee元素开始显示内容时触发此事件编辑事件

onbeforecopy页面当前的被选择内容将要复制到浏览者系统的剪切板前触发此事件

onbeforecut当页面中的一部分或者全部的内容将被移离当前页面并移动到浏览者的系统剪切板时触发此事件

onbeforeeditfocus 当前元素将要进入编辑状态时触发此事件

onbeforepaste 内容将要从浏览者的系统剪切板粘贴到页面中时触发此事件

onbeforeupdate 当浏览者粘贴系统剪切板中的内容时通知目标对象

oncontextmenu当浏览者按下鼠标右键出现菜单时或者通过键盘的按键触发页面菜单时触发的事件

oncopy 当页面当前的被选择内容被复制时触发此事件

oncut 当页面当前的被选择内容被剪切时触发此事件

ondrag 当某个对象被拖动时触发此事件

ondragdrop 一个外部对象被鼠标拖入当前窗口时触发此事件

ondragend 当鼠标拖动结束时触发此事件

ondragenter 当对象被鼠标拖动的对象进入其容器范围内时触发此事件

ondragleave 当对象被鼠标拖动的对象离开其容器范围内时触发此事件

ondragover 当某被拖动的对象在另一对象容器范围内拖动时触发此事件

ontouch触屏事件

DOM操作：增删查改

元素节点 属性节点 文本节点

增加

createElement 创建元素节点

createTextNode 创建文本节点

1. appendChild(b) 把b添加到A里面

createAttribute 创建属性节点

setAttrbuteNode 设置属性节点

删除

a.removeChild(b) 在a里面删除b

更改

a.replaceChild(b,c) 在a里面用b替换掉c

查找

getElementById

getElementsByClassName

getElementByName

querySelector 找到第一个匹配该选择器的元素

关系查找

parentElement 父级元素

Children 子级元素

firstChild 第一个子节点（有可能是文本）

lastChild

parentNode 父级节点

childNodes 子级节点

firstElementChild 第一个标签节点

cloneNode(true)

Document.title 获得页面标题

Body.className 设置或读取class命名

Body.ID 设置或读取id命名

parentNode

parentElement

nodeName

tagName

document.documentElement html

nextSibling 同级指定节点的下一个节点

previousSibling 同级指定节点的前一个节点

clientHeight 元素的可见高度

clientWidth 元素的可见宽度

offsetHeight 元素的高度

offsetWidth 元素的宽度

scrollTop

scrollLeft 读取对象的滚动偏移量(坐标移动)

offsetTop 读取对象移动后的偏移量

offsetLeft 读取对象移动后的偏移量

box.contains(“小米”) 包含关系

clientWdith=width+padding

offsetWidth=width+padding+scrollbar(滚动条)+border

setInterval 会一直占用资源，可能会导致页面崩溃 每过一秒执行一次

setTimeout 不会消耗太多的资源 只执行一次

obj.parentElement //父级元素节点

obj.parentNode //父级节点

obj.children //所有子级元素节点（结果是数组）

obj.childNodes //所有子级节点（结果是数组）

obj.firstChild//第一个子节点

obj.firstElementChildChild//第一个元素子节点

obj.lastChild//最后一个子节点

obj.lastElementChild//最后一个元素子节点

obj.nextElementSibling //同辈的下一个元素节点

obj.nextSibling //同辈的下一个节点(文本也可能是它的下一个节点)

obj.previousElementSibling //同辈的上一个元素节点

obj.previousSibling //同辈的上一个节点(文本也可能是它的下一个节点)

//a.replaceChild(b,c) 在a里面用b替换掉c

//注意b和c一定要是节点，然后用已存在的节点b去替换c，那么已存在的b将会消失

//不想已存在的b消失，就要用克隆节点cloneNode(true) 布尔值正确为深克隆（里面内容也会一起克隆）

btn[2].onclick=function(){

var zxc=obj.nextElementSibling.cloneNode(true)

obj.parentElement.replaceChild(zxc,obj)

}

边界控制

setInterval(function() {

if(a) {

wei.style.left = (wei.offsetLeft > 0 ? wei.offsetLeft - 1 : "") + "px"

}

if(b) {

wei.style.top = (wei.offsetTop > 0 ? wei.offsetTop - 1 : "") + "px"

}

if(c) {

wei.style.left = (wei.offsetLeft < 280 ? wei.offsetLeft + 1 : "") + "px"

}

if(d) {

wei.style.top = (wei.offsetTop < 380 ? wei.offsetTop + 1 : "") + "px"

}

}, 5)

window

navigator 浏览器的信息

screen 屏幕

history

Location 地址栏

document html专用的属性和方法

prompt() 带输入功能的提示框

setInterval()循环执行

setTimeout() 延迟加载 执行一遍

Screen.availHeight　有效高度 860

screen.availWidth 有效宽度 1600

Screen.height

History.go(-1) 返回上一页

var xyx=navigator.usrAgent;

Var qwe=/Trident/i;

qwe.test(xyx)

console.log(zxc.match("Trident"))

console.log(zxc.split(" ")[3])

console.log(qwe.test(zxc))//true就是使用的ie浏览器

Location.hash 读取锚点

Location.host 读取域名

Location.href 读取整个域名网址

Location.href=”www.baidu.com” 页面跳转

Location.pathname 文件位置

Location.protovcol 协议

Location.search 地址栏参数

Location.reload() 重新加载

超链接锚点：一个页面里面的跳转

<a href=:”#a1”>k</a> <div id=”a1”></div>

<a href=”#a2”>kd</a> <a name=”a2”>ji</a>

盒子屏幕宽

box.clientWidth

创建一个标签

box.creatElement(“a”);

设置一个类属性

Box.className=”on”

三元运算符

Move(on>=2?on=2:++on)

随机位置

function test(val1, val2) {

var num = Math.floor(Math.random() \* (val2 - val1 + 1) + val1)

return num + "px"

}

子标签

box.children[0];

cookie和本地存储

//存储 cookie

cookiedocument.cookie="name="+inp[0].value+";expires="+time.toUTCString()+";path=/"

//读取cookie

var str=document.cookie;

//str="password=123456; name=阿斯顿撒"

console.log(str.split(";")[1].split("=")[1])//["password=123456"," name=阿斯顿撒"]

//删除cookie

document.cookie="name=;expires="+time.toUTCString()

本地存储

一次一条，只能字符串存储，没有时间限制

存储

localStorage.setItem("asd",["a","b"])

读取

var str=localStorage.getItem("asd")//"a,b"

删除

localStorage.removeItem("json")//"a,b"

//删除产品（存储的记录里面也要删除）

var dlt = document.querySelectorAll("li a");

for(var k = 0; k < len; k++) {

dlt[k].onclick = function() {

if(confirm("是否确定删除")) {

var li = document.getElementsByTagName("li");

for(var m = 0; m < li.length; m++) {

li[m].index = m;

}

var kai = this.parentElement

ul.removeChild(kai);

shop\_arr.splice(kai.index, 1);

var stra = JSON.stringify(shop\_arr);

localStorage.setItem("shopcar", stra);

choose()

pay()

}

}

}

//价格计算 substr（1）第一位之后的

var asdasd = document.querySelector("div span")

function pay() {

var all\_pay = 0

var check = document.querySelectorAll("li input[type='checkbox']")

for(var i = 0; i < check.length; i++) {

if(check[i].checked == true) {

var li = check[i].parentElement

var pay1 = li.children[3].innerText.substr(1)

var num1 = li.children[5].value

all\_pay += num1 \* pay1

}

}

asdasd.innerText = all\_pay

}

//全选和反选 单个

function choose() {

var check = document.querySelectorAll("li input[type='checkbox']")

var all = true

for(var i = 0; i < check.length; i++) {

if(check[i].checked == false) {

all = false

}

}

all ? qx.checked = true : qx.checked = false

}

//全选和反选

var qx = document.querySelector("div input")

var check = document.querySelectorAll("li input[type='checkbox']")

qx.onclick = function() {

if(this.checked == true) {

for(var i = 0; i < len; i++) {

check[i].checked = true

}

} else {

for(var i = 0; i < len; i++) {

check[i].checked = false

}

}

pay()

}

//数据显示

if(len == 0) {

alert("购物车为空")

} else {

for(var i = 0; i < len; i++) {

ul.innerHTML += '<li><input type="checkbox" /><img src="' + shop\_arr[i].pic + '" alt="" /><h2>' + shop\_arr[i].tit + '</h2><span>￥' + shop\_arr[i].pay + '</span><input type="button" value="-" /><input type="text" value="' + shop\_arr[i].num + '" /><input type="button" value="+" /><a href="javasrcipt:;">删除</a><hr></li>'

}

}

js中的ajax

ajax 是异步的js和xml (不要单独的文件去打开)

请求xml

btn.onclick=function(){

//第一步：打开浏览器

var xhr=new XMLHttpRequest();

//第二步：输入地址（请求方式，目标地址，是否异步处理）

xhr.open("get","xml.xml",true)

//第四步：等待反馈

xhr.onload=function(){

var dom=this.responseXML//拿到请求到的xml数据，结果是节点类型

var arr=dom.getElementsByTagName("name");

console.log(arr);

ul.innerHTML=""

for(var i=0;i<arr.length;i++){

ul.innerHTML+="<li>"+arr[i].innerHTML+"</li>"

}

}

//第三步：发送请求

xhr.send();

}

请求json

btn.onclick=function(){

var xhr=new XMLHttpRequest();

xhr.open("get","json.txt",true);

xhr.onload=function(){

var str=this.responseText;

var arr=JSON.parse(str);

console.log(arr)

ul.innerHTML=""

for(var i=num;i<num+3;i++){

ul.innerHTML+="<li>"+arr[i].name+"</li>";

}

num=num+3

}

xhr.send()

}

一键处理 xhr.onload=function(){}

分步处理

xhr.onreadystatechange=function(){

// console.log(this.status)//HTTP服务器状态码

// console.log(this.readyState)

if(this.status==404&&this.readyState==4){

console.log("路径有问题请修改")

}else if(this.status==500&&this.readyState==4){

console.log("服务器正在维修，请稍后再试")

}

}

xhr.send()

readyState状态码

0 请求未初始化

1 服务器连接创建

2 请求已接收

3 请求处理中

4 请求已完成，且响应已就绪

status状态码

200成功

404 未找到

500 服务错误

面向对象 是一种编程的思维模式，万物皆对象

面向对象的三大特 封装 继承 多态

工厂模式 利用工厂模式 修改了赵六的年龄 八戒的年龄不受影响

function test(val1,val2){

var obj=new Object()

obj.name=val1

obj.age=val2

return obj

}

工厂模式的简写 构造函数

function Test(val1,val2){

this.name=val1

this.age=val2

}

原型函数

Test.prototype.say=function(){console.log(“非常nice”) }

构造函数的使用:通过实例化 去实现

var e=new Test(“study”,22);

console.log(e.say===f.say)//不成立则资源不共用。成立则资源共用

\_\_proto\_\_构造函数的原型链 e --> Test() -->Object --> null

它表示****当我们访问一个对象的某个属性时，如果该对象自身不存在该属性， 就从它的**\_\_proto\_\_**属性上继续查找，以此类推，直到找到，若找到最后还是没有找到，则结果为undefined****

原型构造函数 构造函数里面放不共用的数据，原型里面放共用的数据

function Drag(val1){

this.name=document.getElementById(val1)

}

Drag.prototype.move=function(){

this.name.onmousedown=function(event){

var first\_x=event.offsetX

var first\_y=event.offsetY

this.onmousemove=function(event){

var move\_x=event.clientX

var move\_y=event.clientY

this.style.left=move\_x-first\_x+"px"

this.style.top=move\_y-first\_y+"px"

}

}

this.name.onmouseup=function(){

this.onmousemove=null

}

}

var box1=new Drag("box1");

box1.move()

var box2=new Drag("box2");

box2.move()

继 承 的 原 型

function Anmail(val1,val2,val3){

this.name=val1

this.color=val2

this.age=val3

}

function Cat(name,age,color){

this.style="猫"

//Anmail.call(this,name,color,age)

//Anmail.apply(this,[name,color,age])

//Anmail.apply(this,arguments)

}

Cat.prototype=new Anmail("小白",3,"白色")

Cat.prototype.constructor= Cat;

var m=new Cat()

console.log(m.name)

触屏与拖拽 触屏多用于手机端

点击事件 和 触屏有300ms延迟

var ul = document.querySelector("ul");

var firstx, firsty, lastx, lasty, ul\_left

ul.addEventListener("touchstart", function(event) {

//console.log("开始触屏")

var finger = event.targetTouches[0];

firstx = finger.clientX;

firsty = finger.clientY;

ul\_left=parseInt(ul.style.left)

}, false);

ul.addEventListener("touchmove", function(event) {

//console.log("在屏幕上移动")

var finger = event.targetTouches[0]

lastx = finger.clientX;

lasty = finger.clientY;

var num=firstx-lastx

this.style.left=-(firstx-lastx)/screen.width\*100+ul\_left+"%"

}, false);

ul.addEventListener("touchend", function(event) {

this.style.transition="1s"

if(firstx-lastx > 50) {

this.style.left=ul\_left-100+"%"

} else if(firstx-lastx < -50) {

this.style.left=ul\_left+100+"%"

}else{

this.style.left=ul\_left+"%"

}

setTimeout(function(){

ul.style.transition="0s"

},1000)

}, false);

拖拽基本不用

//拖拽开始（拖拽目标必须含有draggable="true"属性）

li.ondragstart=function(event){

console.log(123)

event.dataTransfer.setData("asd",this.innerHTML);

var img = new Image();

img.src = "../images/c1.jpg";

event.dataTransfer.setDragImage(img,40,10)

}

//拖拽目标移动

li.ondrag=function(){

console.log(456)

}

//拖拽结束

li.ondragend=function(){

console.log(789)

}

//拖拽目标进入放置的容器时触发

box.ondragenter=function(){

this.style.fontSize="50px"

}

//拖拽目标在放置的容器悬浮时触发

box.ondragover=function(event){

event.preventDefault()

this.style.background="yellow"

}

//拖拽目标离开放置的容器时触发

box.ondragleave=function(){

this.style.background="green"

}

//拖拽目标在放置的容器里面释放(需要在ondragover中组织浏览器默认行为)

box.ondrop=function(event){

//this.appendChild(li)

var str=event.dataTransfer.getData("asd");

this.innerHTML=str

关键字检索

|  |
| --- |
| 方法里面先检索是否有关键字，关键字放前面，在一行一行的运行，运行赋值时，才有值，  一行一行运行的时候如果没有赋值，那么那个检索的关键字，会是underfined。 |
| 方法里面的方法声明，也会检索到开头 |
| var a=10;  function test(){  a=100;  console.log(a);  console.log(this.a);  var a=50;  console.log(a);  a=60;  console.log(a);  } test(); |

生产100个div，每个都有标号，颜色按：红，黄，蓝，绿进行交替排序，10行10列。

|  |
| --- |
| 使用字符串拼接最后使用innerHTML增添到box中，比较省事 |
| #box{  width: 400px;  height: 400px;  margin: 0 auto;  position:relative;  display: flex;  flex-wrap: wrap;    }  #box div{  text-align: center;  line-height: 40px;  color: Aqua;  width: 40px;  height: 40px;  position: absolute;  cursor:default;    } |
| <div id="box" class="box"></div> |
| <script>  var box=document.getElementById("box");  var arr=["red","yellow","blue","green"];  var str='';  for(var i=0;i<100;i++){  str+='<div style="left:'+ i%10\*40+"px;"+'top:'+Math.floor(i/10)\*40+'px;background:'+arr[i%arr.length]+';">'+ (i+1) +'</div>';  }  box.innerHTML+=str;  </script> |