26.\_\_proto\_\_指向哪个原型就是继承哪个原型(注意是原型，直接指向函数没用)，如

Demo.prototype.case = "kiss";

function Demo() {

haode = "buhao";

}

Person.prototype.age = 19;

function Person() {

}

var person = new Person();

person.\_\_prototo\_\_ = Demo访问不了person.haode属性

而person.\_\_proto\_\_ = Demo.prototype(一般都这样写写上面的就连prototype上的属性都不能访问)能访问person.case属性

17原型链下 01：14：24

1.call和apply作用改变this指向。

call

使用方法----方法.call(里面填的东西会时方法的this全部指向这个东西，有传实参在后面传就好了);

（接用你的功能实现我的功能）如，

function Person(name, age, sex) {

this.name = name;

this.age = age;

this.sex = sex

}

function Student(name, age, sex, tel, grade) {

Person.call(this, name, age, sex);

this.tel = tel;

this.grade = grade;

}

var student = new Student('sunny', 123, 'male', 139, 2017)

apply

使用方法-------方法.apply(this指向你的东西,后面只能传数组)

2.继承的发展史

1传统模式---原型链（过多的继承了没用的属性）

2借用构造函数（注意要new出来）

3共享原型 如,

Farther.prototype.lastName = 'Deng'

function Father(){ }

function Son(){ }

Son.prototype = Father.prototype;

共享原型继承封装成方法：(注意Target和Origin是构造函数，其他的不需要直接加就好了)（Target继承Oringin的）

function inherit(Target, Origin) {

Target.protopype = Oringin.prototype

}

需要先继承后用这个方法(注意注意)（不这样的话他是改不了原型父级的原型的）

缺点：修改自己的原型会影响父级。

4.圣杯模式

function F();

F.prototype = Father.prototype;

Son.prototype = new F();

这样修改自己的原型不会影响父级的原型

封装方法

function inherit(Target, Origine) {

function O();

O.prototype = Origin.prototype; 1 注意1和2不可以写反（先new的话已经先继承了没用）

Target.prototype = new O(); 2

Target.protopype.constuctor = Target;(使原型找指向自己而不是Father.prototype)

Target.prototype.uber = Origine.prototype;(加这一步是为了想知道自己真正指向的父级是谁)

}

用的话就可以

inherit(参数就行了);

3.闭包的私有化（小媳妇）

var inherit = (function () {

var O = function () {};

return function (Target, Origin) {

O.prototype = Origin.prototype;

Target.protype = new O();

Target.prototype.constuctor = Target;

Target.prototype.uber = Origin.prototype;

}

}());

4. 命名空间

老方法 var = obj{}

新方法 利用闭包私有化。 （留一个接口）return里面的函数。如

var name = "bcd"

var init = (function () {

var name = "abc"

function callName() {

console.log(name);

}

return function () {

callName();

}

var initDeng = (function () {

var name = 123

function callName() {

console.log(name);

}

return function () {

callName();

}

下面两个人写的三个人相同的功能不影响。

name打印出bcd

init（）打印出abc

initDeng()打印出123

1.对象中的属性是function的话就是对象的方法也可以说函数。

如 var mrZhu = {

sex :male,

smoke : function() {console.log("我抽烟！！");}

}

mrZhu.smoke();打印出我抽烟了（注意调用对象属性里面的方法这样调用）

6.Math.random();随机数 返回(0, 1)的随机数

7.类数组长得像数组但又不能用数组的方法 如,

arguments是个类数组,不能用push数据进数组里面会报错.

8.成为类数组的条件

1)属性要为索引(数字)属性

2)必须要有length属性(必要条件),做好加上push属性

9.如果强行让类数组调用push方法,则会根据length属性值的位置进行扩充. 如,

var obj = {

"2" : "a",

"3" : "b",

"length" : 2,

"push" : Array.prototype.push

}

obj.push("c");

obj.push("d") 调用obj打印出obj {2 : "a", 3 : "d", length: 4}其原理是push方法封装,

Arr.prototype.push = function (target) {

obj[obj.length] = target;

obj ++;

}

1.try catch形式

try{

里面的代码发生错误，不会执行错误之后的代码且不会影响try catch之外的代码

}

catch(e（可以随便写什么。原理是系统会把错误信息封装到error对象里面把error对象传到e里面）){

他会捕捉try里面的错误代码，try没有错误时不会执行catch代码

}

2.error对象error.message和error.name

3.错误信息

4.es5.0严格模式(es3.0和es5.0冲突的部分用es5.0)注意没开启严格模式冲突部分是用3.0

启用方法 "use strict"; 在script内最上面写上字符串形式（逻辑的最顶端下面就是es5.0的模式）。

写在局部就只影响那一个写上了严格模式的函数

5.with(obj) {}改变作用域链 obj成为{}里面的代码题的最顶端的原型。es5.0不能用。

6. es2.0严格模式with，arguments.callee, func.caller不能使用，

未经声名变量不能使用（3.0是暗示全局变量），（this预编译指向window）局部this必须被赋值

(Person.call(null/undefined)赋值什么就是什么)，不能使用重复的属性和参数。

1.日期对象是系统提供好的对象。

2.查到w3c看date的属性和方法。

3.定时器(window的方法)

setInterval(function () {}, time);

每隔time时间执行一次function。 他会返回一个唯一标识可以用一个var来接受他(用来清除这个定时器clearInterval)

time只识别一次

steTimeout(function() {}, time);

隔time时间才执行这个function (只执行一次)。 clearTimeout同上

设置计时器

4.脚本化css(简介操作css)读写都是在行间。

读写css属性

dom.style.prop方法 修改行间样式(如果是css外部文件或者在head里面的都读写不到，只能读写到css的行间样式)

保留字前面加css如float ---->cssFloat

符合单词小头峰式-----> backgroundColor

写入的值必须是字符串格式----->div.style.height = "100px"

查询计算样式

window.getComputedStyle(ele,null).属性 只读 ele要获取的东西可以是div（获取的是当前元素的显示值(最终显示的不管css在哪)）

如果是相对值他会自己计算好之后返回绝对的值。

5.获取滚动条位置

window.pageXOffset/pageYOffset

document.body.scrollLeft + document.documentElement.scrollLeft 写兼容IE8一下的

6.获取视图大小

window.innerWidth/winnerHeight

ducument.documentElement.clientWidth/clientHeight 标准模式下

document.body.clientWidth/clientHeight 给怪异模式下 看html上面的！那行

document.compatMode方法可以区分怪异模式和标准模式

7.查看元素几何尺寸大小（是实时的放回不了动态的尺寸）

domEle.getBoundingClientRect();------->div.getBoundingClientRect();返回值left,right,top,bottom等

left和top代表左上角的x和y的坐标，right和bottom代表右小角的x和y的坐标

IE浏览器用这个方法不会返回width和height值 可以通过right-left计算宽，bottom-top计算高。

8.查看尺寸大小(视角上的尺寸，用的多一点)

dom.offsetWidth, dom.offsetHeifht ------>div.offsetWidth

9.查看元素位置

dom.offsetLeft/offsetTop 返回无定位父级元素，返回相对文档的坐标。对于又定位的父级元素返回相对最近有定位父级的坐标

dom.offserParent返回的是最近有定位的父级

10.让滚动条滚动

window上又三个方法

scroll(),scrollTo(); 不能累加方法一模一样

scrollBy()；能累加

括号里面填x的坐标，y的坐标

1. 实现自动阅读功能

1.事件就是实现交互体验的核心

2.绑定事件的方式

ele.onxxx = function(event) {}；

--->div.onclick = function () {} 基本都兼容

一个元素同一个事件只能绑定一次

程序this指向dom元素本身

解除事件：ele.onxxx = null/false;

obj.addEventListener("type", fn, false);

---->div.addEventListener ("click", function(){}, false ) IE9下不兼容

可以一个事件绑定多个处理函数，同一个元素执行同一个函数只能执行一次

程序this指向dom元素本身

解除事件：ele.removeListener(type, fn, false)里面的全部值要是同一个，(想要解除需要用一个函数应用，不然找不到函数)

obj.attachEvent("on+ type", fn)；

--->div.attachEvent("onclick", function(){}) IE独有的

可以一个事件绑定多个处理函数，同一个元素执行同一个函数可以执行多次

程序this指向window

ele.detachEvent("on" + type, fn)和addEventListener一样全部要是同一个

3.封装兼容性的方法要背起来addEvent(elem, type, handl);

4.事件处理模型(一个对象的一个事件类型只能出现一个模型(要么冒泡要么捕获))

事件冒泡(常规模型)：

结构上(非视觉上)嵌套关系的元素，会存在事件冒泡的功能，即同一事件上，自子元素冒泡向父元素。(自底向上)

事件捕获(不常规模型)(只有谷歌浏览器实现了)：

结构上(非视觉上)嵌套关系的元素，会存在事件捕获的功能，即同一事件上，自父元素捕获至子元素(事件源元素)。(自顶向下)

5.事件触发顺序，先捕获后冒泡。

6.事件的执行顺序是先绑定先执行(测试捕获和冒泡的先后顺序需注意绑定执行顺序)。

7.focus, blur, change, submit, reset, select等事件不冒泡

8.在每一个事件处理函数中都可以传一个形参(e)即事件对象里面有很多属性其中有一个属性方法就可以取消冒泡。

9.取消事件冒泡

w3c标准event.stopPropagation();IE9以下不支持

event.cancelBubble = true; ie独有的

10.封装冒泡函数stopBubble(event);

11.默认事件----->表单提交， a标签跳转， 右键菜单等。

12.阻止默认事件

return false；(句柄的方式阻止 只能取消句柄绑定的默认事件)

以对象属性的方式注册的事件才生效

event.preventDefault(); W3C标准，IE9一下不兼容

event.returnValue = false; 兼容IE

13.封装阻止默认事件函数cancelHandler(event);(封装不下return false)

14.事件对象

event(就是function里面的形参e)非IE存在于。 IE存在于window.event--->event || window.event

15.事件源对象(事件是在哪里执行的)

event.target 火狐只有这个

event.srcElement le只有这个

chrome浏览器两个都有

16.封装事件源对象（应用事件委托）

17.鼠标事件

click(敲击事件), mousedown(), mousemove, mouseup, contextmenu(右键出现菜单事件),

mouseover(鼠标放进去出现什么), mouseout(鼠标出来出现什么), mouseenter, mouseleave

18.区分左右键只能用moseup和mousedown事件可以区分 里面的button属性。e.button ==2 右键 e.button == 0 左键

19.键盘事件

keydown, keyup, keypress

触发顺序 keydown>keypress>keyup

keydown和keypress的区别

keydown能监听所有键盘类事件且charcode值为0(fn不能，fn键是辅助键)

keypress能监听所有字符类按键charcde为ask码(能区分字母的大小写)

20.文本类事件

input事件(改变就会触发该事件), focus(聚焦触发), blur(失去焦点触发), change(状态位发生改变才触发)(聚焦改变失去焦点才触发)

21.窗体操作类事件(window上的事件)

scroll(滚动条事件), load(效率最低,等randTree加载绘制完成之后才触发)

22.div.setCapture(); div.releaseCapture();配套使用 捕获事件 不通用只有IE有不重要

1. 事件2的作业

1.DOM只能操作html和xhtml,不能操作css(不能改变css的样式表)(可以改变行间样式，间接修改)

2.是选中html要先在body里面创建之后才可以选择。

3.document代表整个文档,在html外面。

4查

.用js选择元素括号里面填的要是字符串形式（带s都是选择出一组元素）

.getElementsById();通过ID名选择出元素 不区分id的大小写

.getElementsByTagName();通过标签名选择出元素（表单元素）

.getElementsByName();通过name选择出元素

.gtElementsByClassName();ie9一下不能用 通过class民选择出元素

下面的是静态的是副本，上面的是动态的，实时的。

.querySelector("里面css怎么选这里怎么选");

.querySelectorAll("里面css怎么选这里怎么选");选择出一组

5.生成的成组的基本上都是类数组。

6.遍历节点树（注意是节点）

parentNode 父节点（最顶端是#document）；

childNodes[第几个节点，可不写]--子节点们（所有的子节点）

firstChild-第一个子节点 lastChild-最后一个子节点

nextSibling-后一个兄弟节点 previousSibing 前一个兄弟节点

7.节点类型

元素节点 -----1 属性节点-----2 文本节点-----3 注释节点----8 document节点-------9 DocumentFragment-----11

8.基于元素点的遍历（除了chidren其他的都是IE9一下不兼容的）

parentElement（顶端是html）返回单前元素父元素节点

chidern 返回当前元素的子元素节点

firstElementChild 返回第一个元素节点

lastElementChild 返回最后一个元素节点

nextElementSibLing后一个元素节点 previousElementSibLing前一个元素节点

9.节点的四个属性

nodeName 元素的名称，以大写的形式表示，只读

nodeValue 只有文本节点和注释节点属性节点，有（text,comment）可读可写

nodeType 返回节点的类型，只可读

attributes属性节点的集合

10.hasChildNodes();方法 检测有没有子节点有turn没有false。

11.obj.prototype-->事件--->node--->Document,CharacterData,Element,Attr。

Document---->HTMLDocument,XMLDocument--->document,xmldocument

12.DOM树

getElementsById定义在Document.prototype上，Element节点上不能使用

getElementByName定义在了HTMLDocument.prototype上，不能在xml document和Element上使用

getElementByTagName定义在Document.prototype上和Element上 都能使用

HTMLDocument定义了一些常用的属性，body和head分别指其标签

document.body打印<body>标签

Document.prototype上定义了documentElement属性，指代文档的根元素，HTML中指代<html>

document.documentElement打印出<html>标签

13.增（实在js手里拿着不会显示在页面） 需要显示在页面可以document.body.appendChild("标签名")

.createElement("标签名");创建元素节点（标签）

.createTextNode("");创建节点(文本)

.creatComment("")注释节点

.createDocumentFragment()；创建文案碎片节点

14.插

appendChild(可以理解成push) 页面上已经有的剪切操作 如，

div.appendChild(span); 原来div和span是兄弟结构的变成父子结构了。（）里面不需要“”

.insertBefore(a, b); 读insert a Before b (a在b前面) 要选中他的父元素 ParentNode.insertBefore(a, b);

15删

.parent.removeChild();选中父节点删除括号里面的子标签 不需要""

child.remove(); 删掉选中的节点。

16.改

parentNode.replaceChild(new（新节点）, origin（老节点）)new代替origin

17.元素的一些属性

innerHTML = (覆盖) innerHTML += (在内容后面加东西)

innertext = 文本 (覆盖) 小心有span标签

18.元素上的方法

setAttribute("(属性名)class","on(属性值)") 如 class="on" 添加行间属性

getAttribute("标签") 取出行间属性

19.需要先定义一个标签。

总结：

1. 计算的顺序是自左向右 优先级高的先计算