20170801

作业题讲解

1、有一个div:

我想获取这一个div你有几种解决办法(不考虑兼容)：

```
document.getElementById(“div1”);
document.getElementsByTagName("div")[0];
document.getElementsByClassName(“w”)[0];
document.getElementsByName(“h”)[0];
// ->querySelector/querySelectorAll 在不需要考虑兼容的情况下（例如：移动端开发），我们经常使用这两个方法获取需要的DOM元素
document.querySelector(“#div1 / .w / div / [name='h']”);
document.querySelectorAll(“#div1 / .w / div / [name='h']”)[0];
```

2、获取当前浏览器屏幕的宽度和高度(兼容所有的浏览器)：

```
 //->获取宽(左边html为假，获取右边body的)
document.documentElement.clientWidth || document.body.clientWidth
//->获取高
document.documentElement.clientHeight || document.body.clientHeight
```

逻辑与（&&）和逻辑或（||）

1、作为条件的分隔符

```
if(1==1 && 2==2){
    //->逻辑与：左边的条件为真，右边条件也为真，整体才为真，否则就是假 =>"并且"

}

if(1==1 || 2==2){
    //->逻辑或：左右两边的条件只要有一面为真，整体就为真，反之为假 =>"或者"
}
```

2、赋值运算符

```
//->逻辑或：A||B,首先验证A的真假，如果A为真，返回的结果为A，如果A为假，返回的结果是B（不管B是真是假）
function fn(num1,num2){
//->1、参数初始化：如果参数没有传递，给参数一个默认值
    //typeof num1==="undefined"?num1=0:null;//（传统判断做法：麻烦，但是严谨）

//->2、一般我们这样处理参数初始化即可，比较的简便，但是这样不严谨：传递值了，但是传递的是false/0/空字符串，这样返回的也是右边的值
        num1 = num1 || 0;
        num2 = num2 || 0;
}
fn(10);
//----------------------
//->逻辑与：A && B，A如果为真，返回右边的B，如果A为假，返回左边的A（和逻辑或正好相反）
fn && fn();//->如果当前函数存在，则让函数执行，否则函数不执行（不严谨：如果fn是10呢？）
typeof fn==="function"?fn():null;//->和上面的意思相同，但是这种方式严谨一些
//--------------------
//->一个运算中既有 || 也有 &&，&&的优先级高于 ||
var res=0 || 2 && 3 && 0 || false;
/*
* 2&&3 ->3
* 3&&0 ->0
* 0||0 ->0
* 0||false ->false
*/
console.log(res) //=>false
```

3、获取上一个哥哥元素节点（兼容所有的浏览器）：

`previousElementSibling [不兼容]`

```
function prev(curEle){
    if("previousElementSibling" in curEle){
        return curEle.previousElementSibling;
    }
    //->不兼容的浏览器执行如下操作
    var pre = curEle.previousSibling;
    while(pre && pre.nodeType !==1){
        pre = pre.previousSibling;
    }
    return pre;
}
```

4、动态创建一个div标签，并且添加到body的最后面位置：

```
    var oDiv = document.createElement("div");
    document.body.appendChild(oDiv);
//继续，把刚才创建的克隆一份一模一样的，添加到刚才创建的div前面：最后把新创建的这个div删除；
    var cloneDiv = oDiv.cloneNode(true);//刚才创建的克隆一份一模一样的
    document.body.insertBefore(cloneDiv,oDiv);//添加到刚才创建的div前面

    document.removeChild(oDiv);//移除
```

5、实现找到第n项到第m项的内容，返回一个新的数组(原有数组不变)：

```
    ary.slice(n-1,m-1+1); => ary.slice(n-1,m)
```

作用域练习题

```
    var ary = [1, 2, 3, 4];
     function fn(ary) {
     ary[0] = 0;
     ary = [0];
     ary[0] = 100;
     return ary;
     }
     var res = fn(ary);//把全局下的ary存储的值xxxfff111当作实参传递给
     console.log(ary);//[0 2 3 4]
     console.log(res);//[100]
```

[2]

```
var foo = 1;
function bar() {
if (!foo) {//->!undefined 取反后的结果是true，条件是成立的
var foo = 10;//->不管条件是否成立，都会进行提前声明，函数也只是提前声明【把私有变量foo的结果赋值为10】
}
console.log(foo);
}
bar();
```

[3]

```
var a = 4;
function b(x, y, a) {
alert(a);// ->3
arguments[2] = 10;//函数内置的一个实参集合 = 把形参a的值也同时修改为10了
            /*
            * 在非严格模式下，arguments中的每一项值，和函数的形参存在“映射关
系”
            * ->arguments中的某一项值改变，形参也会改变
            * ->形参改变了，arguments中的这一项值也会变
            */
alert(a);// ->undefined
}
a = b(1, 2, 3);//把b执行，把函数执行的返回结果赋值给a
    =undefined
alert(a);//=>undefined
```

[3-1]

```
'use strict';//->开启js代码的严格模式：这句话需要写在当前作用域的起始位置
var a = 4;
function b(x, y, a) {
alert(a);
arguments[2] = 10;
alert(a);
}
a = b(1, 2, 3);
alert(a);
```

[4]

```
var a = 9;//0 1 0 1 2
function fn() {
    a = 0;
    return function (b) {
    return b + a++;
    }
}
var f = fn()//xxxfff000
var m = f(5);//xxxfff000(5) ->5 a=1
alert(m);//5
var n = fn()(5);//xxxfff000(5) ->5 a=1
alert(n);//5
var x = f(5);//xxxfff000(5) ->6 a=2
alert(x);//6
alert(a);//2
```

[6]

```
var num = 10;//60 65
var obj = {num: 20};
obj.fn = (function (num) {
    this.num = num * 3;//60
    num++;//21
    return function (n) {
        this.num += n;//60+5=65
        num++;22
        console.log(num);
    }
})(obj.num);
var fn = obj.fn;
fn(5);//22
obj.fn(10);//23 30
console.log(num, obj.num);//65 30
```

[7]

```
//->构造函数
function Fn() {
    var num = 100;
    this.x = 100;
    this.y = 200;
    this.getX = function () {
        console.log(this.x);
     }
 }
 Fn.prototype.getX = function () {
     console.log(this.x);
 };
 Fn.prototype.getY = function () {
     console.log(this.y);
 };
 var f1 = new Fn;//->new执行的时候，如果不需要给函数传递参数值，小括号加或者不加无所谓，都是一样的
 var f2 = new Fn;
 console.log(f1.getX === f2.getX);//false
 console.log(f1.getY === f2.getY);//true
 console.log(f1.__proto__.getY === Fn.prototype.getY);//true

//[原型查找]-----------------------------------------------------
 console.log(f1.__proto__.getX === f2.getX);//false
 console.log(f1.getX === Fn.prototype.getX);//false
 console.log(f1.constructor);//Fn
 console.log(Fn.prototype.__proto__.constructor);//Object

//[this]---------------------------------------------------------
 f1.getX();//->首先找的是私有的getX，并且getX方法中的this是f1，console.log(this.x)相当于输出f1.x =>100
 f1.__proto__.getX();//->首先找的是公有的getX，并且getX方法中的this是 f1.__proto__，console.log(this.x)相当于输出f1.__proto__.x=>undefined?
 f2.getY();//->this：f2，console.log(this.y)相当于输出f2.y =>200
 Fn.prototype.getY();//->this:Fn.prototype，console.log(this.y)相当于输出Fn.prototype.y =>undefined

 //=>总结:
 //->首先看方法执行中的this是谁（函数执行，点前面是谁，this就是谁）
 //->把js代码中的this换为刚才分析的结果
 //->按照原型链查找机制，直接判断输出的值即可

 //-----------------------------------------------------
 1、所有的函数数据类型
    ->普通函数
    ->类
    天生具备一个属性：prototype，这个属性的值是一个对象,作用：当前类的公共属性和方法都在这个对象中存储着
```

2、 prototype天生具备一个属性：constructor（构造函数），这个属性存储的值就是当前类（函数）本身

3、每一个对象数据类型
->普通对象{}
->数组和正则等
->类的实例
->函数也是对象
->prototype也是对象
也会天生自带一个属性：__proto__，这个属性存储的值是它对应类的prototype属性的值

```
//[思考题]-------------------------------------------
console.log(f1.num);//undefined
f1.num=1000;
console.log(f2.num);//undefined
Fn.prototype.num=2000;
console.log(f2.num);//2000

console.log(f1.hasOwnProperty("x"));//true
console.log(Fn.prototype.hasOwnProperty("getY"));//true
console.log(f1.hasOwnProperty("getY"));//false
console.log(f1.hasOwnProperty("hasOwnProperty"));//false
console.log(Fn.prototype.hasOwnProperty("hasOwnProperty"));//false
console.log(Object.prototype.hasOwnProperty("hasOwnProperty"));//tr
ue

//->in检测是否存在这个属性
//->hasOwnProperty检测是否为它的私有属性
Object.prototype.myHasPubProperty=function myHasPubProperty(){
    //->完成你的js
}
f1.myHasPubProperty("getY");//->true
```