# let与var的区别

var没有 块级作用域 用于定义变量

会引起问题

if块级

if（条件）{

var name = ‘a’;

var func = function（）{

console.log(name);

}

}

func();//在其他地方引用func（）name可能会更改掉了。

for块级

//以前的解决方法是：闭包问题

//函数是有作用域的!!!!

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

(function (inumble) {

btn[i].addEventListener('click',function () {

i +=1;

console.log('第'+i+'个按钮被点击');

})

})

(i)

//执行函数，把i传递进上面的函数inumble= i，在上面的函数里执行，是传进去的值，不被方法外的值改变而影响

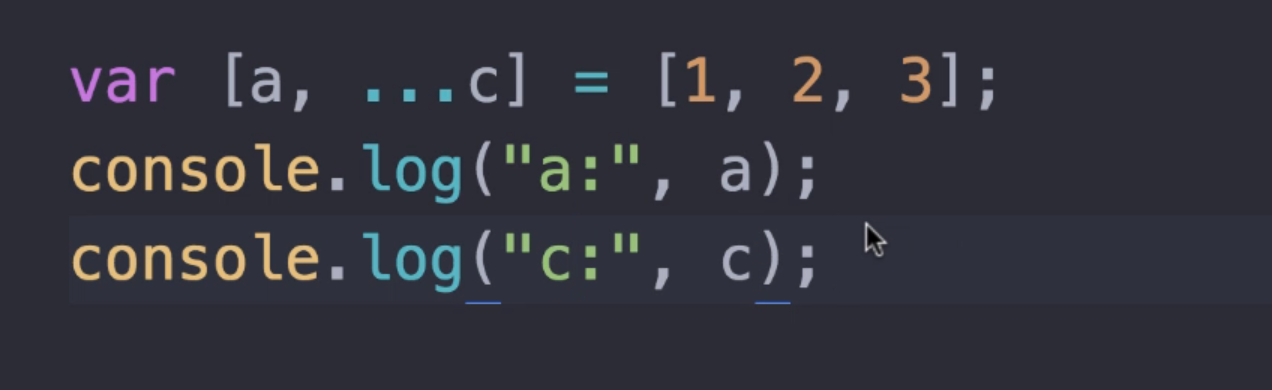
}

let 有块级作用域

# const

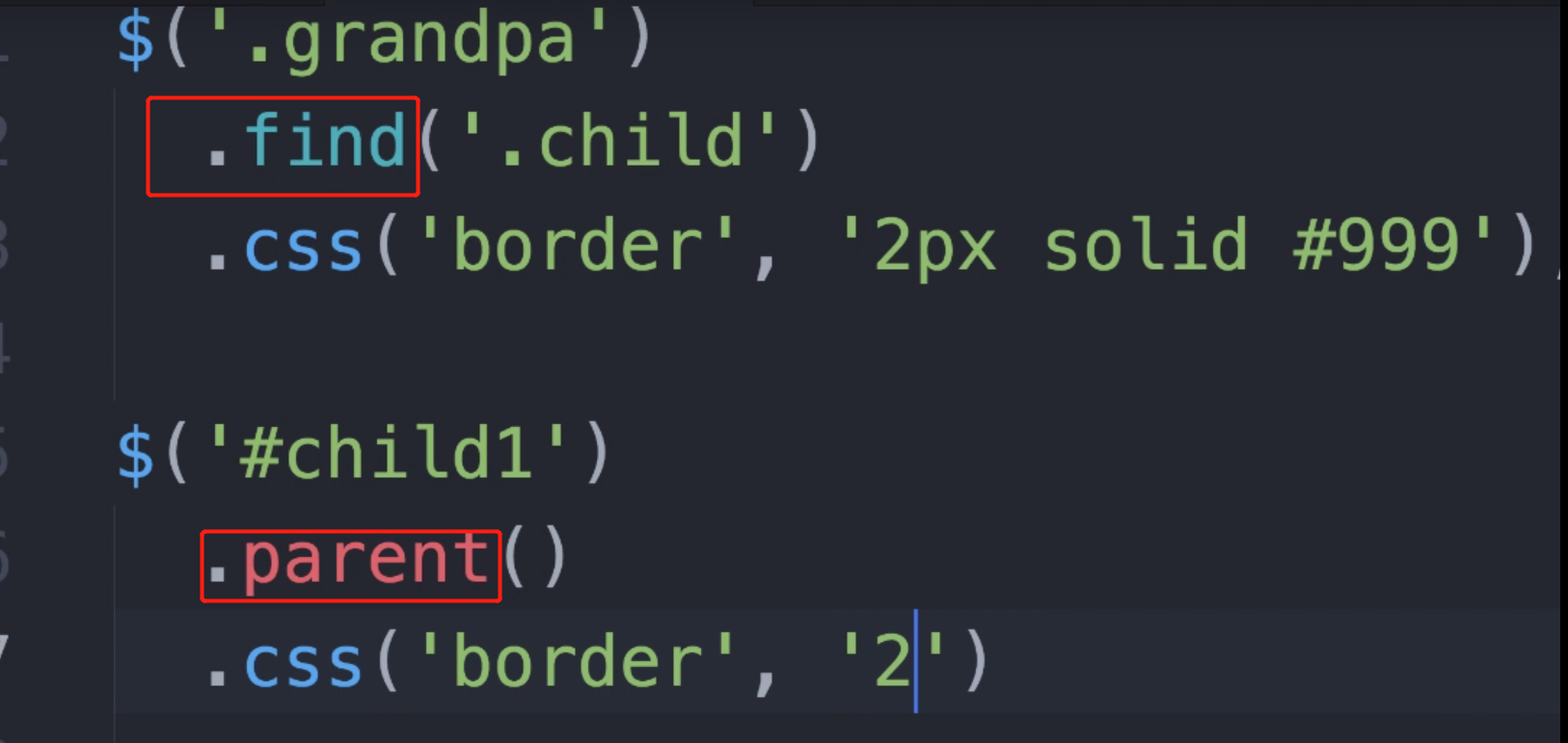
# 数组

赋值

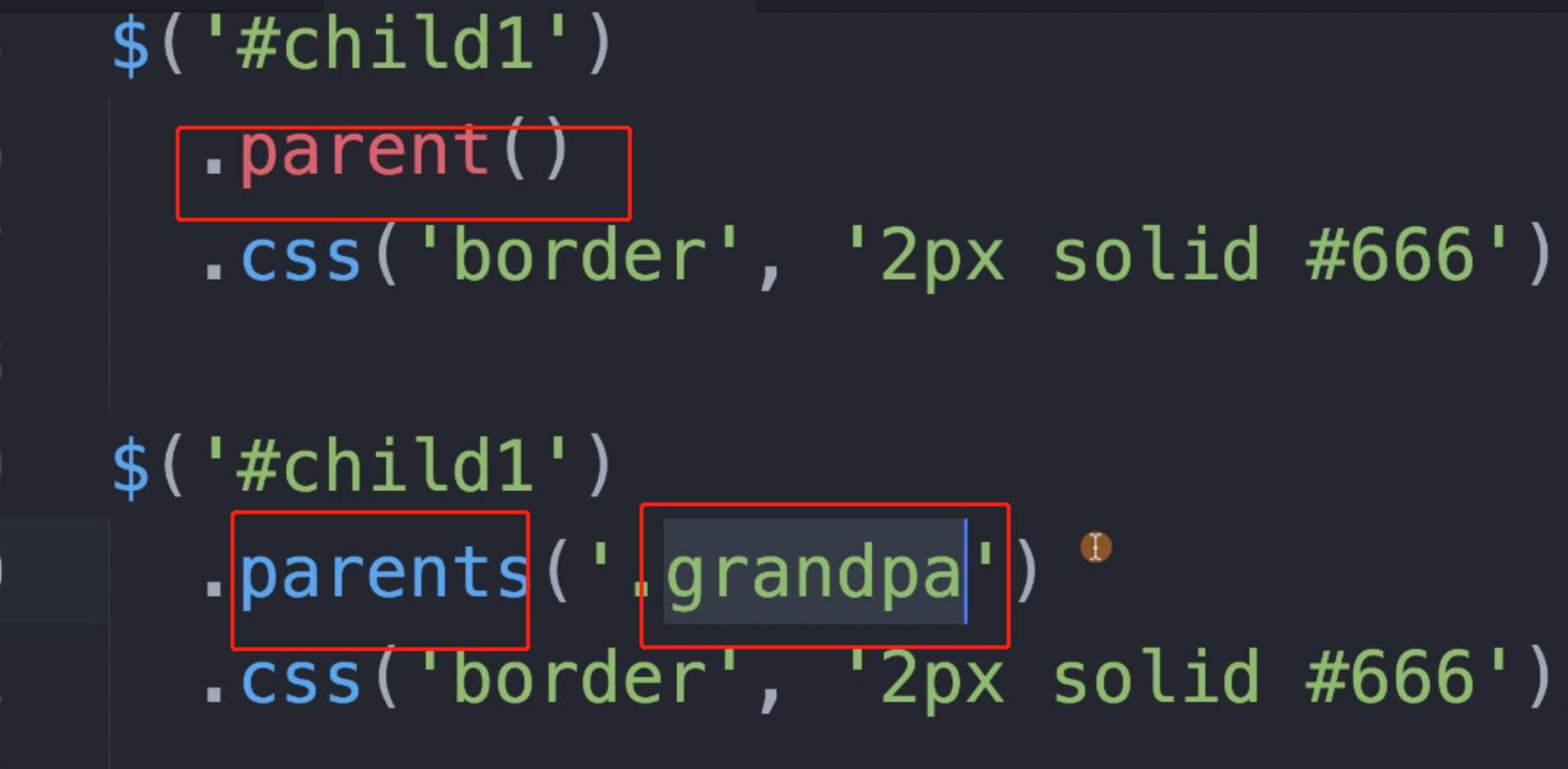


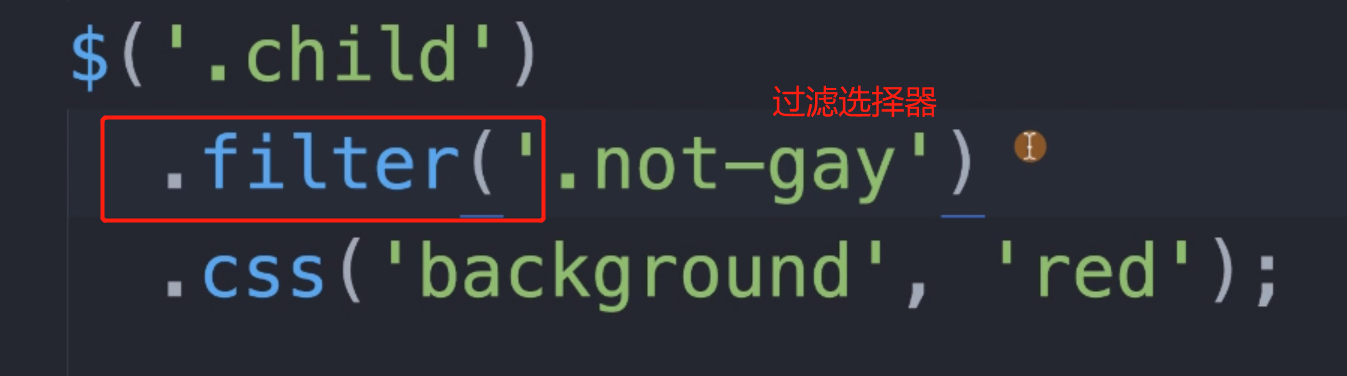
对象

# jq选择器



find是从上往下的选择，parent是从下往上一级，，parents是选中所有上一级的。。

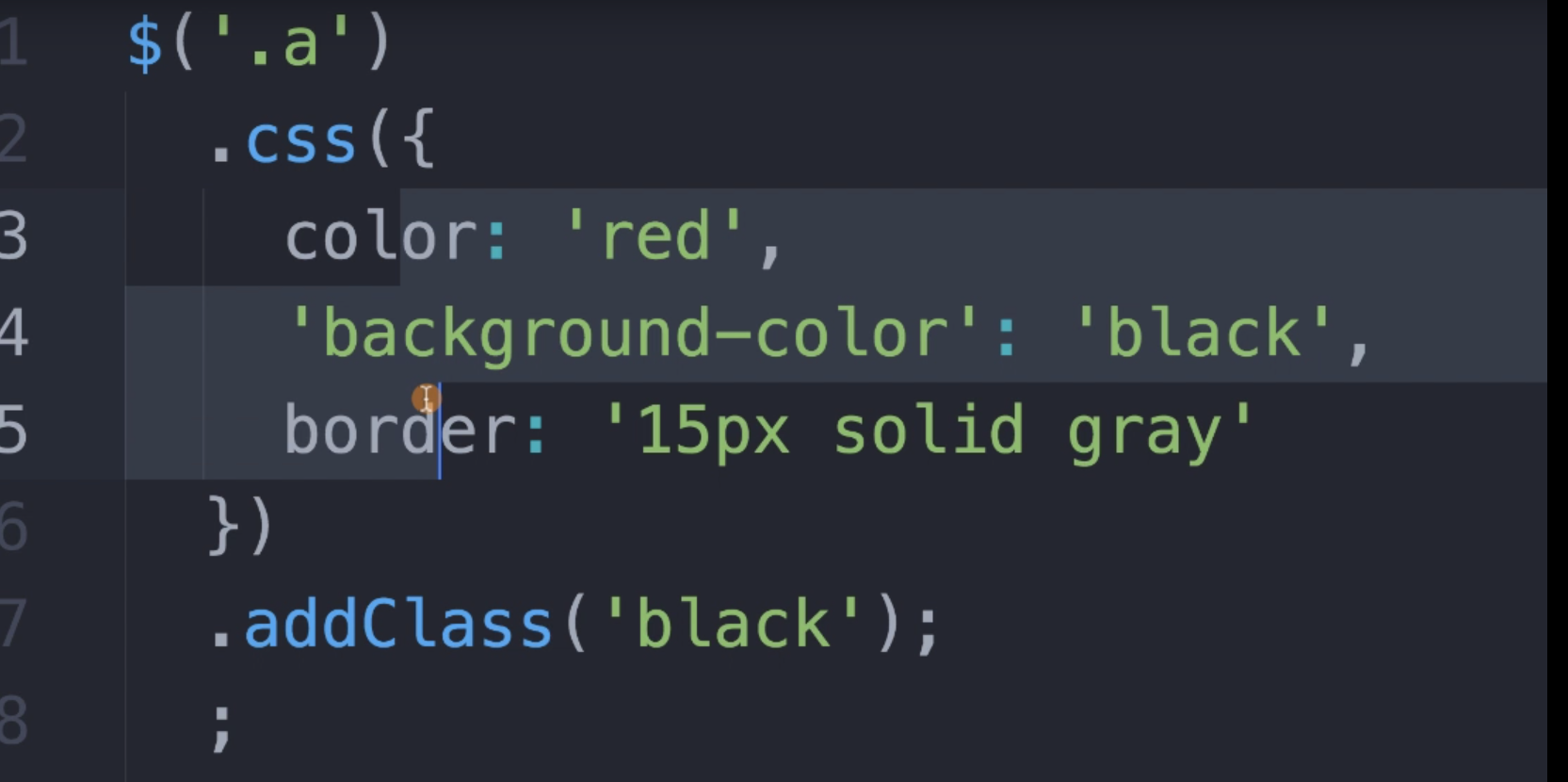




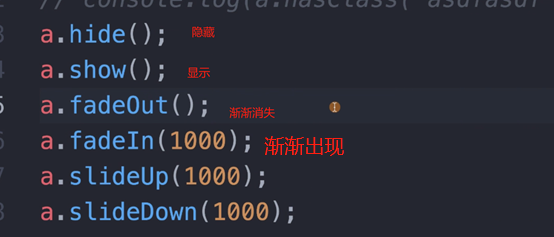
addClass 增加样式增加类名

新建一个类





removeClass去除类



# slideToggle() 方法

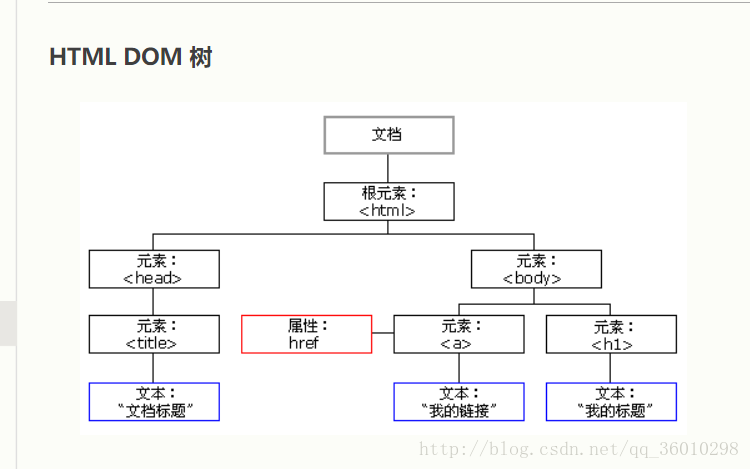
在被选元素上进行 [slideUp()](https://www.runoob.com/jquery/eff-slideup.html) 和 [slideDown()](https://www.runoob.com/jquery/eff-slidedown.html) 之间的切换

例如

$("button").click(function(){  
$("p").slideToggle();  
});点击显示/隐藏p标签

# dom操作

dom树



## 获取节点

document.getElementById(“id名”)



## 修改dom

innerHTML

// 获取<p id="p-id">...</p>

var p = document.getElementById('p-id');//获取p-id节点

// 设置文本为abc:

p.innerHTML = 'ABC'; // <p id="p-id">ABC</p>

// 设置HTML:

p.innerHTML = 'ABC <span style="color:red">RED</span> XYZ';

// <p>...</p>的内部结构已修改

## 增加dom

因为我们插入的js节点已经存在于当前的文档树，因此这个节点首先会从原先的位置删除，再插入到新的位置。

更多的时候我们会从零创建一个新的节点，然后插入到指定位置：

**var**

**list** = document.getElementById('list'),//获取节点

haskell = document.createElement('p');//创建新的节点

haskell.id = 'haskell';//p节点id

haskell.innerText = 'Haskell';//插入text文本内容

**list**.appendChild(haskell);// appendChild，把一个子节点添加到父节点的最后一个子节点

## 删除节点

找到要删除节点的父节点，，然后用removePatent()

*// 拿到待删除节点:*

**var** **self** = document.getElementById('to-be-removed');

*// 拿到父节点:*

**var** **parent** = **self**.parentElement;

*// 删除:*

**var** removed = **parent**.removeChild(**self**);

removed === **self**; *// true*

# filter

# typeof