javascript 复习资料5

链接：练习题 链接：javascript简易教程 链接：数组遍历 链接：正则表达式

javascript的对象

对象（object）是javascript里面很重要的一个部分。尽管在入门的时候我们不需要自己写一个对象，不过至少要知道对象的用法。

console.log()

`console.log()`是我们很常用的一个命令。这个命令里面有一个`.`号，这个符号之前的console是javascript自带的一个**对象**，而log就是这个对象的一个**方法**。`console.log()`的意思就是**调用**console这个对象自带的log方法，并且把参数写在括号里面

> 试一下在浏览器的console里面输入下面的命令`console.log(console)`，自己理解一下这个命令是什么意思，然后理解一下返回的结果

arr.length

之前在我们做数组遍历的时候写过这样的程序：

```
var arr = [1,2,3,4,5]
for(var i =0;i<arr.length;i++){
        console.log(arr[i])
}
```

上面的程序可以把数组里的每一个元素都按顺序输出。这里用到了`arr.length`。arr是我们在第一行建立的数组。而**作为**数组，arr就具有length这个属性，以及类似.forEach()这样的方法。length作为数组的一个属性描述这个数组现在有多少的元素。

> 试一下`console.log(arr)`,注意到返回结果的[1,2,3,4,5]左边还有一个黑三角形，点击那个三角形应该可以看到"length:5"这么一条，另外顺便把_proto_前面的三角形也点一下，看看里面的结果

DOM模型

下面是一段html语言

```
<ul id="ullist">
        <li>
                <a href="http://www.github.com" target="_blank">
                        text for link
                </a>
        </li>
</ul>
```

上面的html代码实际上是建立了4个对象，按照层次排下来分别是：`ul,li,a,textnode`。因为可以直接用getElementById拿到id是ullist的ul，这里以ul这个element为例：

```
var ul = document.getElementById('ullist')
console.log(ul.id);
```

document.getElementById

上面代码里面`document.getElementById()`的对象是`document`，getElementById()是document的方法。为什么getElementById前面一定要加document呢？就像log前面为什么一定要加console一个道理，getElementById是只有document这个对象才有的方法。document这个对象可以粗略的理解为整个html文档。

> 试一下console.log(document)，然后点小三角展开列表看一下document这个对象下面都有什么

getElementById这个document的方法做的事情就是把当前网页里面具有某个id的元素给找出来然后作为方法的返回值传递给`=`前面的变量，在我们的程序里就是ul这个变量

那么下面一行`console.log(ul.id)`是什么意思？

获取现在的li元素

`<ul>`这个元素现在被存在`ul`变量里面，那如果要获取里面的li元素应该怎么办呢？每一个元素都有一个children**属性**

```
var ul = document.getElementById('ullist')
console.log(ul.children)
// 输出值是[<li>...</li>]，也就是包含了ul里面所有li（现在只有一个）的类似数组的集合
```

那么现在要获取第一个（也是唯一一个）li，正确的程序就是:`ul.children[0]`。那如果要进一步获取那个元素呢？

如果`ul.children[0]`是现在的li的话，这个li的children属性就包含了a，所以正确的程序是：`ul.children[0].children[0]`

获取/设置a元素的href属性

既然ul.children[0].children[0]是现在的a元素，那这个a元素的`href`属性是什么呢？

一个数组arr的长度属性是这么写：`arr.length`，所以现在这个a元素的href属性就是`ul.children[0].children[0].href`。下面的程序就是设置a的href属性为"http://www.bing.com"

```
var ul = document.getElementById('ullist')
console.log(ul.children[0].children[0].href)
ul.children[0].children[0].href="http://www.bing.com"
```

a元素里面的文字是什么呢?

每一个元素都有一个属性叫做innerText，通过修改这个属性就可以修改元素里面的文字，当然也可以获取这个元素里面的文字。

除了innerText，每个元素还有另外一个属性叫做innerHTML。

> 自己试一下这几个命令：

```
console.log(ul.innerHTML);
console.log(ul.children[0].innerHTML);
console.log(ul.children[0].children[0].innerHTML);
```

实际上属性有两种，一种是像length和children这种只能读不能修改的，还有一种是像innerText, innerHTML和href这样子又可以读又可以写的。至于一个元素/对象有什么属性，哪些能读哪些能写就只能自己查文档去了。

如果要修改一个元素里面的文字，直接修改innerHTML这个属性就可以了。包括如果现在我想在里面link单词两边加上一个span标签，程序可以这么写：

```
ul.children[0].children[0].innerHTML = ul.children[0].children[0].innerHTML.replace("link","
<span>link</span>")
```

innerHTML这个命令在动态修改网页的时候看起来比createElement, appendChild这些命令来的方便，但是不推荐用这个来修改网页。

给上面的ul新加一个li和a元素

也就是让网页变成下面这样：

```
<ul id="ullist">
        <li>
                <a href="http://www.github.com" target="_blank">
                        text for link
                </a>
        </li>
        <li>
        <a href="http://foo.bar">
                more link
        </a></li>
</ul>
```

实现这种需求的程序之前课上已经说过了：

```
ul = document.getElementById('ullist')
var li = document.createElement('li')
var a = document.createElement('a')
var tn = document.createTextNode('more link')
ul.appendChild(li)
li.appendChild(a)
a.appendChild(tn)
```

这个程序看起来很麻烦，如果可以把createElement和appendChild一起做了就好了，比如这样`ul.createElement('li')`。但是这是不行的，因为只有document这个对象才有createElement这个方法。

所以还是一行一行的看一下上面的程序。第一行就是获取ul，2和3行分别建一个li元素和a元素，第4行建一个字符节点。后面的就是把新建的节点彼此建立一下联系。不过呢，还没有设置a的href属性和target属性。在appendChild以前应该尽快设置一下。程序修改一下成下面这样：

```
ul = document.getElementById('ullist')
```

```
var li = document.createElement('li')
var a = document.createElement('a')
a.href = "http://foo.bar"
a.target = "_blank"
var tn = document.createTextNode('more link')
ul.appendChild(li)
li.appendChild(a)
a.appendChild(tn)
```

dom的事件

先看一下之前课上讲引文处理时候用到的一个html：

```
<button type="button" name="button" onclick="txtProc()">convert</button>
```

从javascript角度看，这是一个button对象，具有type，name属性，它的innerText是"convert"，除此以外还有一个onclick的东西，这个是事件。html里面的各种元素都支持各种各样的事件，比如onclick是鼠标点击的时候会触发的事件，onmouseover是鼠标移动到这个元素上面的时候会触发的事件。看一下下面的代码：

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title></title>
        </head>
        <body>
                <div class="" onmouseover="mouseover()" onclick = "mouseclick(this)">
                        a div with events
                </div>
                <div class="" id="div2">
                        div 2
                </div>
                <script type="text/javascript">
                        function mouseover(){
                                console.log("mouse over");
                        }
                        function mouseclick(e){
                                console.log(e);
                        }
                </script>
        </body>
</html>
```

自己试一下把鼠标移动到div上面和点击div有什么效果（记得打开console）。

要给一个元素加上事件，写法就是事件名称="运行的函数(参数)"，比如上面的onmouseover="mouseover()"，onmouseover是事件名称，mouseover()是在script标签里面声明的一个函数。

再看一下 `onclick = mouseclick(this)` 这里。现在多了一个参数this。**在这里**，this代表的是发生事件的这个div（实际上this是javascript里面很容易被弄混的一个语句），所以在函数里面：

```
function mouseclick(e){
        console.log(e);
}
```

这个时候输出的就是被点击的这个div了。

语句汇总：

方法和属性

- **console.log()**: 输出括号内内容，是console对象的log方法 - **arr.length**: 数组对象的length属性，如果是读取属性：`var l = arr.length`，如果是设置属性：`a.href="#"`

DOM操作

- **var a = document.getElementById("对象id")**: 根据括号里面的id名称在html文档里查找对应的元素，并返回交给=前面
- **a.children**: a变量代表的元素下属的所有元素，是一个类似数组的集合
- **a.href = "http://foo.bar"**: 修改一个元素的属性
- **ul.children[0].innerText**: 一个元素内部的文字，可读可写
- **var li = document.createElement('li')**: 新建一个
- 元素，用li变量存储
- **li.appendChild(a)**: 把a变量代表的元素作为li变量代表元素的子代
- `<button type="button" name="button" onclick="txtProc()">convert</button>`: html语句，其中onclick="txtProc()"是给这个元素添加一个onclick事件，当按钮被点击以后执行txtProc()函数

函数

函数声明的格式：

```
function func1(arg1,arg2){
        //statements
        var val = " is "
        return arg1+val+arg2
}
```

function表示接下来声明一个函数， func1是函数名称，括号里面写的是参数的名字。这些参数（比如arg1，arg2）可以在函数里面去用。和函数里面声明的变量（比如val）不一样，参数可以在调用函数的时候去设置，但是函数里面声明的变量在调用函数的时候是碰不到的。 另外return表示返回的值。函数调用的时候是根据函数的名称来写，比如调用上面的函数

```
var r = func1("apple","red")
console.log(r);
//输出是apple is red
```

for循环

for循环的基本格式

```
for(var i=0;i<maxval;i++){
        //statement
}
```

比如遍历一个数组的全部元素

```
var a = "appletree".split("")
for(var i=0;i<a.length;i++){
        proc(a[i]);
}
function proc(e){
        console.log(e);
}
```

数组

- var a = [1,2,3]: 声明数组
- a[2]: a的第2+1个元素
- a.forEach(function(e,i,a){...}): 遍历
- var b = a.map(function(e,i,a){return ...}): 映射
- var f = a.filter(function(e,i,a){return a>1}): 筛选