

数组与其他对象的一些区别

数组api

函数(匿名函数)的一个例子

forEach分析(这里是重点!非常重要!)

小结

数组与其他对象的一些区别

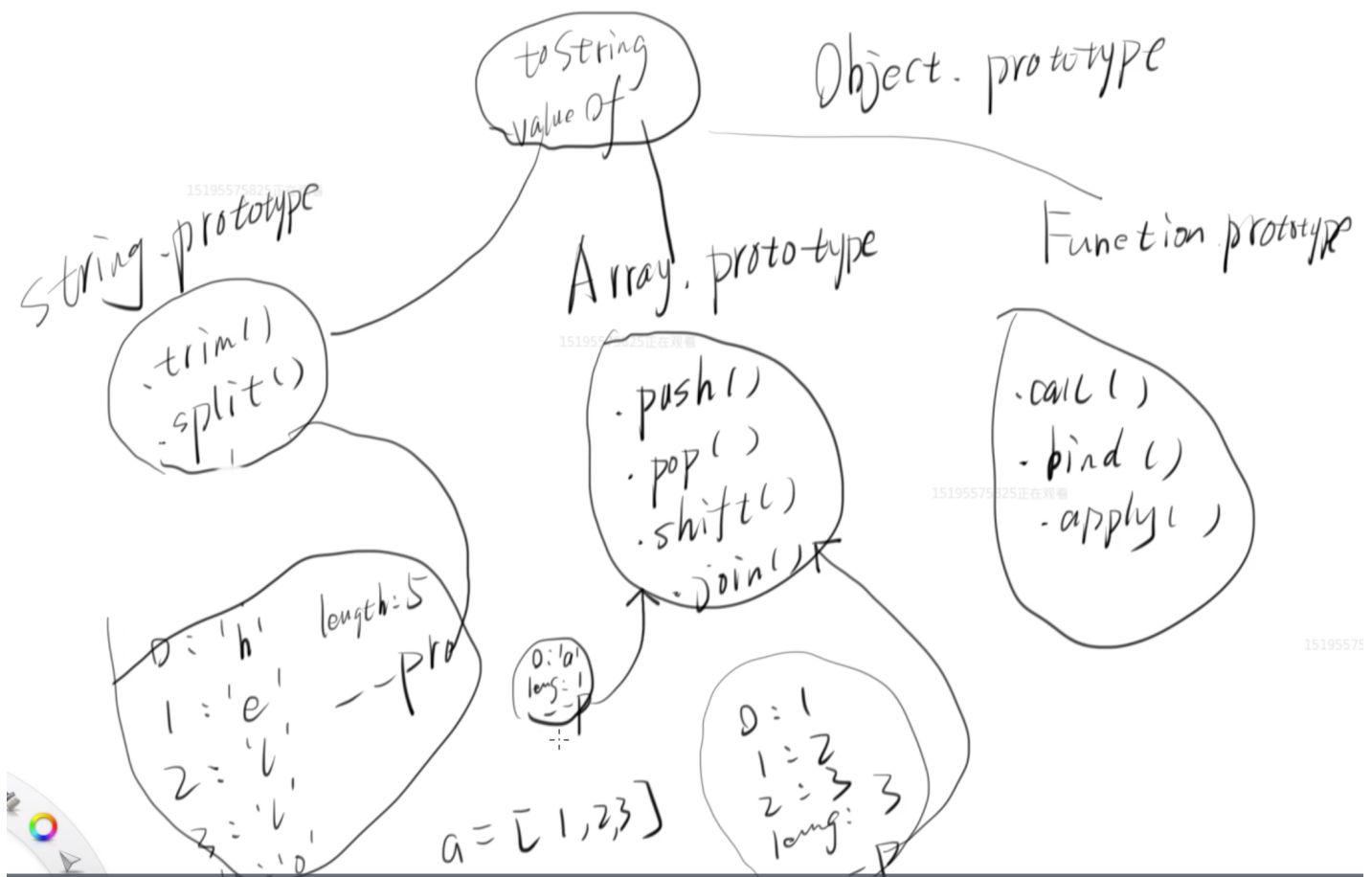
Object是由object构造出来的对象,Array是由Array构造出来的一种特殊对象!

toString,valueof是所有Object的公用属性;

所有String的公用对象里面有 `.trim()`, `.split()` (在 `String.prototype` 下面)等方法;

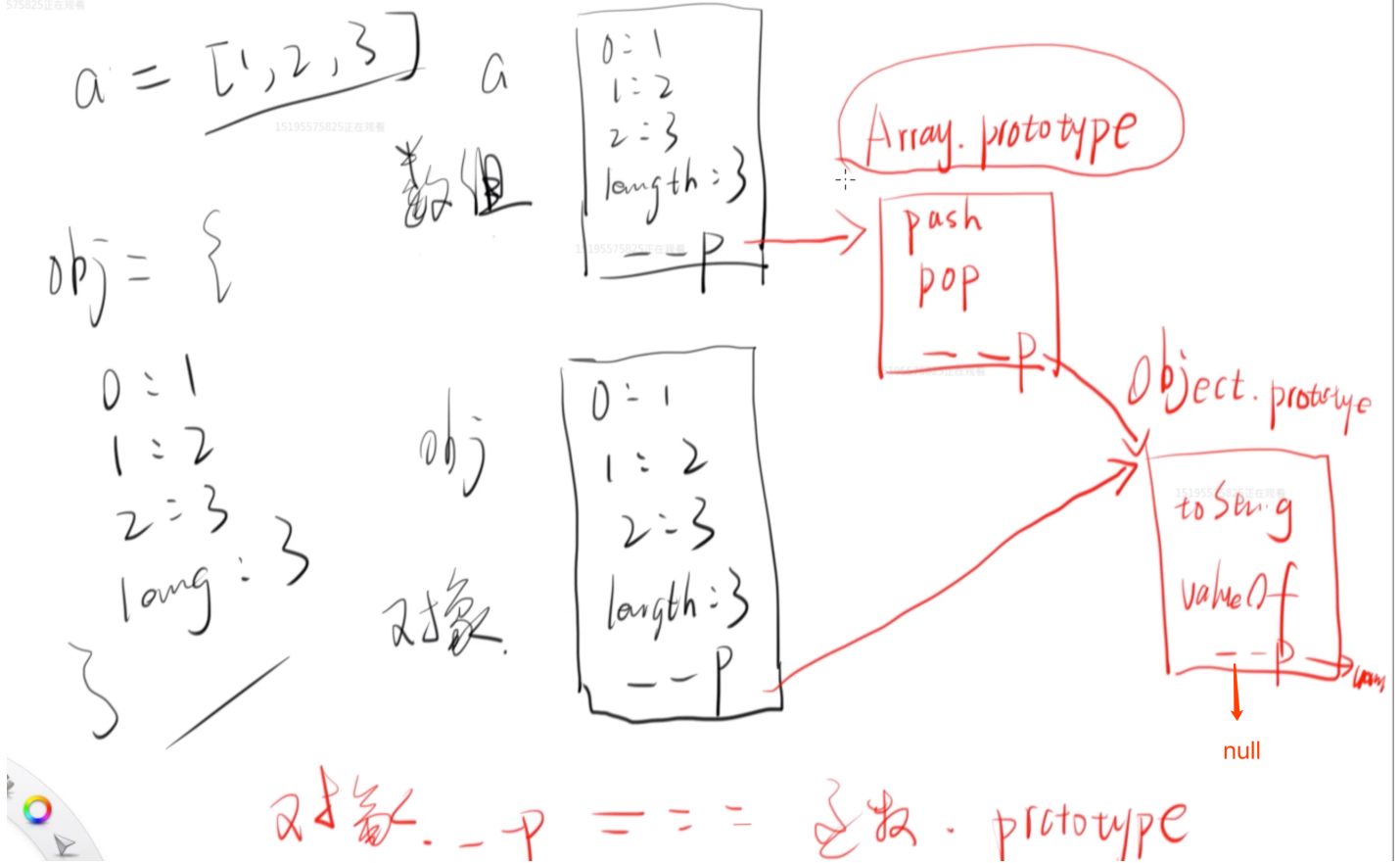
所有数组都有共有属性 `Array.prototype` :里面有 `.push()`, `.pop()` 等;

所有的函数都有共有属性 `Function.prototype` :里面有 `.call()`, `.bind()`, `.apply()` 等;



数组和对象的区别是对象没有数组的共有属性,而数组有!!

575825正在观看



伪数组:没有 `Array.prototype` 这个属性
只有Arguments是,在js里面;

数组api

forEach

```
> a = ['a', 'b', 'c', 'd']
< ▶ (4) ["a", "b", "c", "d"]
> a.forEach(function(x,y){
  console.log('value', x)
  console.log('key', y)
})
```

value a	VM956:2
key 0	VM956:3
value b	VM956:2
key 1	VM956:3
value c	VM956:2
key 2	VM956:3
value d	VM956:2
key 3	VM956:3
undefined	

每个函数都会默认返回 `return undefined` (哪怕你不写return!)

查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境

过滤输出

>> a=['a','b','c','d']
< ▶ Array(4) ["a", "b", "c", "d"]
>> a.forEach(function(x,y){
 console.log('value',x)
 console.log('key',y)
})
< undefined
value a
key 0
value b
key 1
value c
key 2
value d
key 3
>> function f2(fn){
 console.log('我接受到了一个函数')
 return undefined
}
< undefined
>> f2(function(){})
< undefined
我接受到了一个函数

每个函数都会默认return undefined
匿名函数

例子:

```
1 function x(y){ if (typeof y !=='function'){ console.log('get out!!!') return false } e  
2 undefined  
3 x(1)  
4 false  
5 get out!!! debugger eval code:4:9  
6 x( function(){} )  
7 true  
8 我接受到了一个函数  
9
```

```

← undefined
>> function x(y){
  if (typeof y !== 'function'){

    console.log('get out!!')
    return false
  }
  else{

    console.log('我接受到了一个函数')
    return true
  }
}

```

← undefined

```
>> x(1)
```

← false

get out!!

```
>> x( function(){} )
```

← true

我接受到了一个函数

```
>>
```

函数(匿名函数)的一个例子

```

>> function y_我的名字叫接受并执行同时传参给另外一个函数(y){
  y(666)
}

```

← undefined

```

>> y_我的名字叫接受并执行同时传参给另外一个函数(function(){
  console.log(arguments)//arguments就是这个函数在执行的时候接受到的参数!
})

```

← undefined

Arguments { 0: 666, ... }

```

>> y_我的名字叫接受并执行同时传参给另外一个函数(function(参数1){
  console.log(参数1)//!
})

```

← undefined

666

```

>> //y_我的名字叫接受并执行同时传参给另外一个函数:这个函数作用就是把接受的这个函数执行一下,然后传个666
//y_我的名字叫接受并执行同时传参给另外一个函数:这个函数接受了一个函数(这里为匿名函数:function(sdf){console.log(sdf)})
//这个匿名函数必须接受一个参数;这个参数(这里指的是sdf这个参数!)的值这里还不知道!必须要看另外一个函数的源代码(上面声明了!)才知道
//
y_我的名字叫接受并执行同时传参给另外一个函数(function(sdf){
  console.log(sdf)//
})

```

← undefined

666

```
>>
```

```

1 function y_我的名字叫接受并执行同时传参给另外一个函数(y){ y(666) }
2 undefined
3 y_我的名字叫接受并执行同时传参给另外一个函数(function(){ console.log(arguments)//arguments就是
4 undefined
5 Arguments { 0: 666, ... }
6 debugger eval code:2:9
7 y_我的名字叫接受并执行同时传参给另外一个函数(function(参数1){ console.log(参数1)//! })
8 undefined
9 666 debugger eval code:2:9

```

```
10 //y_我的名字叫接受并执行同时传参给另外一个函数:这个函数作用就是把接受的这个函数执行一下,然后传个666//
11 undefined
12 666
13
```

forEach分析(这里是重点!非常重要!)

a.forEach是个函数,结了两个参数,一个是数组,一个是对它的操作

```
>> function forEach(array,x){
  for (let i=0;i<array.length;i++){
    x(array[i],i)
  }
}
← undefined

>> forEach(['a','b','c'],function(value,key){
  console.log(value,key)
})
← undefined
a 0
b 1
c 2

>> //forEach所做的事:遍历['a','b','c']这个数组,对于数组中的每一项,
//都调用function(value,key){console.log(value,key)}这个匿名函数;
//调用上面这个匿名函数的时候会传2个参数;第一个参数是函数的值(value):array[i];
//第二个参数是函数的key:i,这里有三项,就传三次,每次都是不一样的值和key
forEach(['a','b','c'],function(value,key){
  console.log(value,key)
})
← undefined
a 0
b 1
c 2

>> var a=['a','b','c'];
a.forEach(function(p,q){
  console.log(p,q)
})
////这里的意思是在a上面遍历每一项,在每一项上面调用
//函数x,在调用函数x的时候传一个value(这里为参数p)和key(这里为参数q)
← undefined
a 0
b 1
c 2

>>
```

这里为forEach的源代码

```
1 function forEach(array,x){ for (let i =0;i<array.length;i++){ x(array[i],i) } }
2 undefined
3 forEach(['a','b','c'],function(value,key){ console.log(value,key) })
```

```

4 undefined
5 a 0 debugger eval code:2:9
6 b 1 debugger eval code:2:9
7 c 2 debugger eval code:2:9
8 //forEach所做的事:遍历['a','b','c']这个数组,对于数组中的每一项,//都调用function(value,key){co
9 undefined
10 a 0 debugger eval code:6:9
11 b 1 debugger eval code:6:9
12 c 2 debugger eval code:6:9
13 var a=['a','b','c']; a.forEach(function(p,q){ console.log(p,q) }) ////这里的意思是在a上
14 undefined
15 a 0 debugger eval code:3:9
16 b 1 debugger eval code:3:9
17 c 2
18

```

```

>> var obj = {0:'a',1:'b',length:2}
← undefined
>> obj.forEach=function(x){
  for (let i=0;i<this.length;i++){
    x(this[i],i)
  }
}
← function forEach()
>> obj.forEach
← function forEach()
>> obj.forEach(function(value,key){console.log(value,key)})
← undefined
a 0
b 1
>> //a.forEach接受一个函数,这个函数必须接受3(或者是2)个参数,第一个参数是a的value,
//第二个参数是a的key,第三个参数是a它自己(下面这个例子里面为d)
a=['fff','jjj','ddd'];
a.forEach(function(b,c,d){
  console.log(b,c,d)
})

```

❗ SyntaxError: illegal character [详细了解]

```

>> //a.forEach接受一个函数,这个函数必须接受3(或者是2)个参数,第一个参数是a的value,
//第二个参数是a的key,第三个参数是a它自己(下面这个例子里面为d)
a=['fff','jjj','kkk'];
a.forEach(function(b,c,d){
  console.log(b,c,d)
})

```

← undefined

```

fff 0 Array(3) [ "fff", "jjj", "kkk" ]
jjj 1 Array(3) [ "fff", "jjj", "kkk" ]
kkk 2 Array(3) [ "fff", "jjj", "kkk" ]
>>

```

a的key

a的value

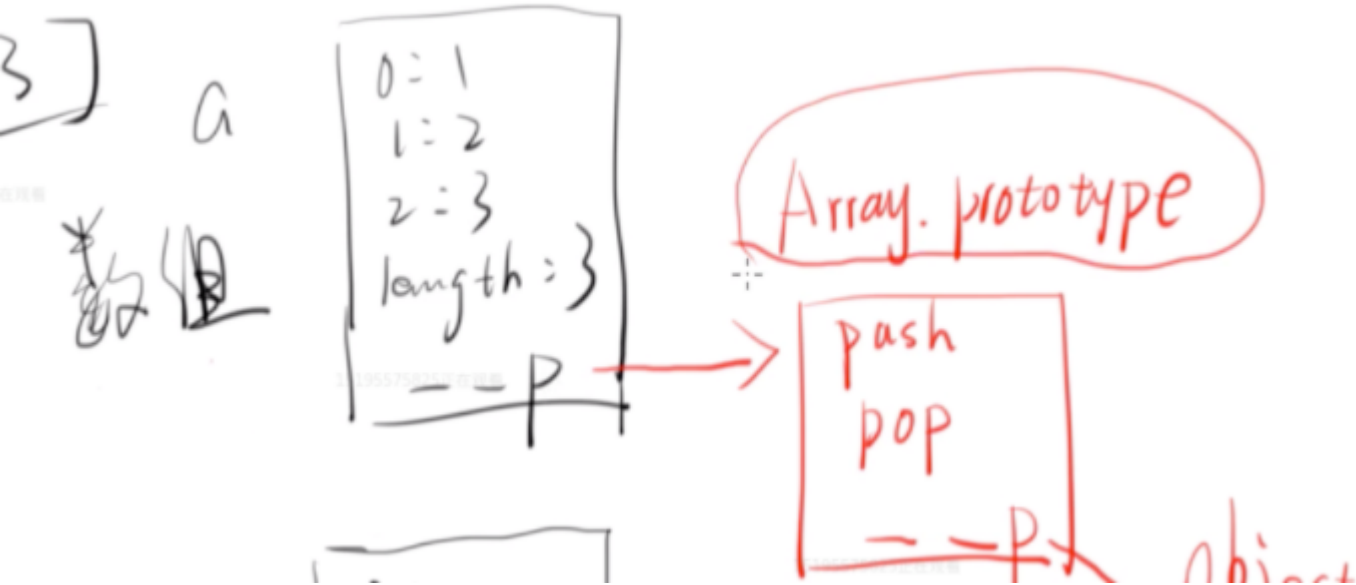
```

1 //a.forEach接受一个函数,这个函数必须接受3(或者是2)个参数,第一个参数是a的value,//第二个参数是a的ke
2 undefined
3 fff 0
4 Array(3) [ "fff", "jjj", "kkk" ]
5 debugger eval code:5:11
6 jjj 1
7 Array(3) [ "fff", "jjj", "kkk" ]
8 debugger eval code:5:11
9 kkk 2
10 Array(3) [ "fff", "jjj", "kkk" ]

```

小结

数组的本质:下划线__proto__指向Array.prototype!



forEach的理解!