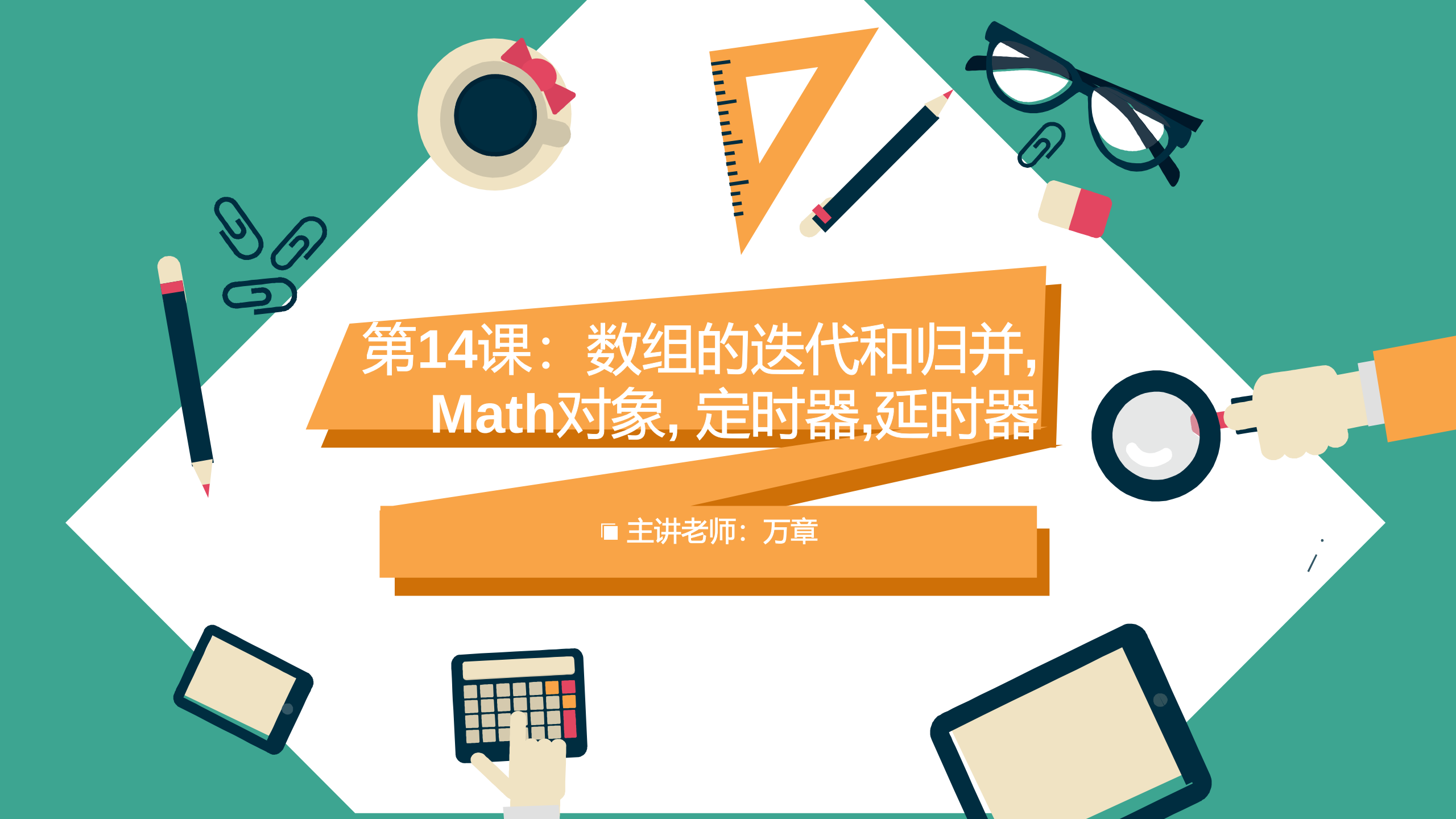
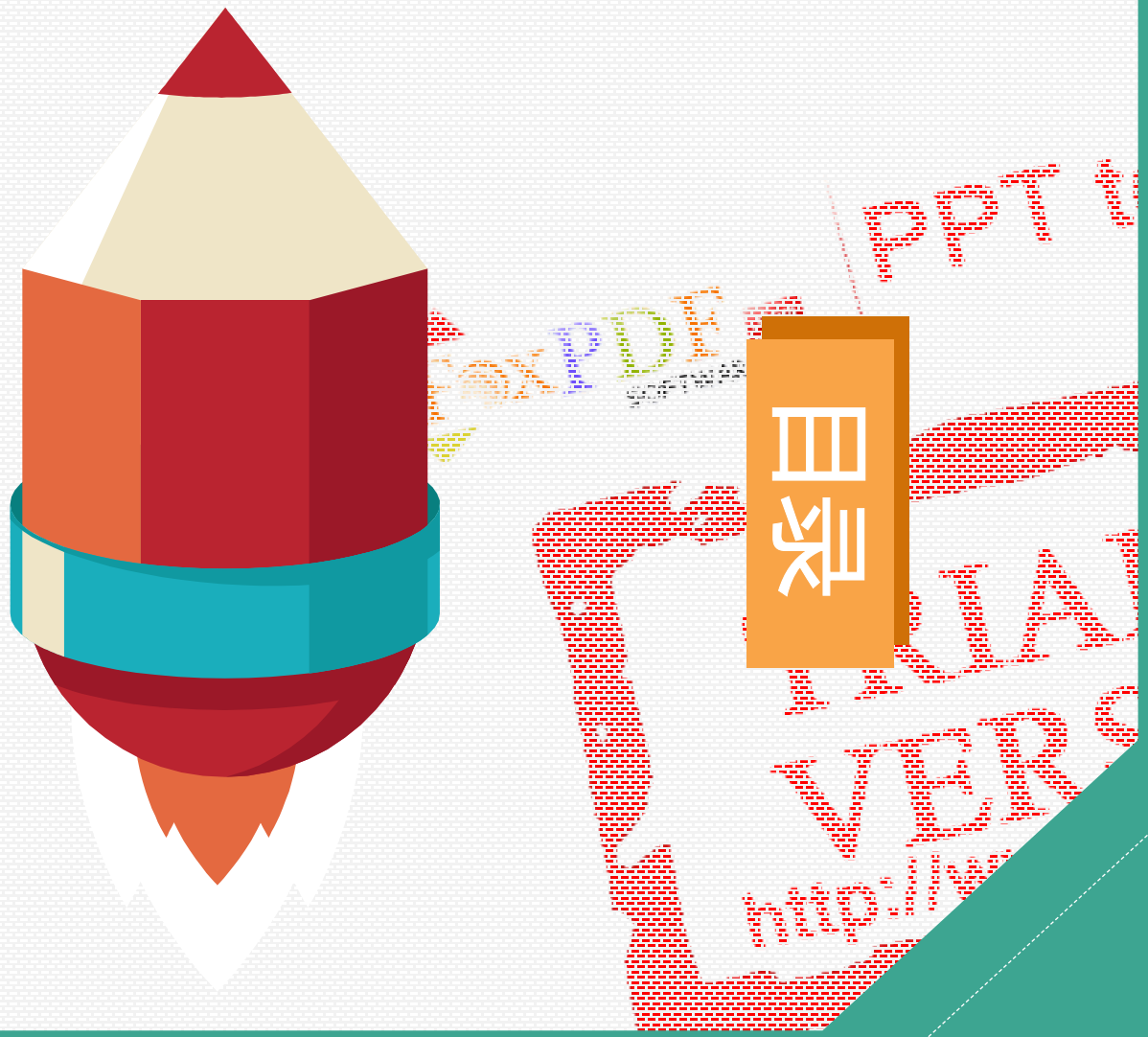


第14课：数组的迭代和归并，Math对象，定时器,延时器

主讲老师：万章



目录

- 数组的迭代和归并
- 数学对象Math
- 延时器
- 定时器

数组的迭代和归并

数组的高级方法之迭代方法

ECMAScript 5 为数组定义了5 个迭代方法。每个方法都接收两个参数：要在每一项上运行的函数和（可选的）运行该函数的作用域对象——影响this 的值。

传入这些方法中的函数会接收三个参数：数组项的值、该项在数组中的位置和数组对象本身。根据使用的方法不同，这个函数执行后的返回值可能会也可能不会影响方法的返回值。以下是这5 个迭代方法的作用。

- every()：对数组中的每一项运行给定函数，如果该函数对每一项都返回true，则返回true。
  some()：对数组中的每一项运行给定函数，如果该函数对任一项返回true，则返回true。
  filter()：对数组中的每一项运行给定函数，返回该函数会返回true 的项组成的数组。
  forEach()：对数组中的每一项运行给定函数。这个方法没有返回值。
  map()：对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组。

以上方法都不会修改数组中的包含的值。

数组的高级方法之迭代方法的every()和some()

最相似的是every()和some()，它们都用于查询数组中的项是否满足某个条件。
对every()来说，传入的函数**必须**对**每一项**都返回true，这个方法才返回true；否则，它就返回false。
对some()方法则是**只要**传入的函数对数组中的**某一项**返回true，就会返回true。

```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];

var everyResult = numbers.every(function (item, index, array) {
    return (item > 2);
});

console.log(everyResult); //false
```

every方法

确保每个数组项目都得符合return的语句条件, 参会返回true

```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];

var someResult = numbers.some(function (item, index, array) {
    return (item > 2);
});

console.log(someResult); //true
```

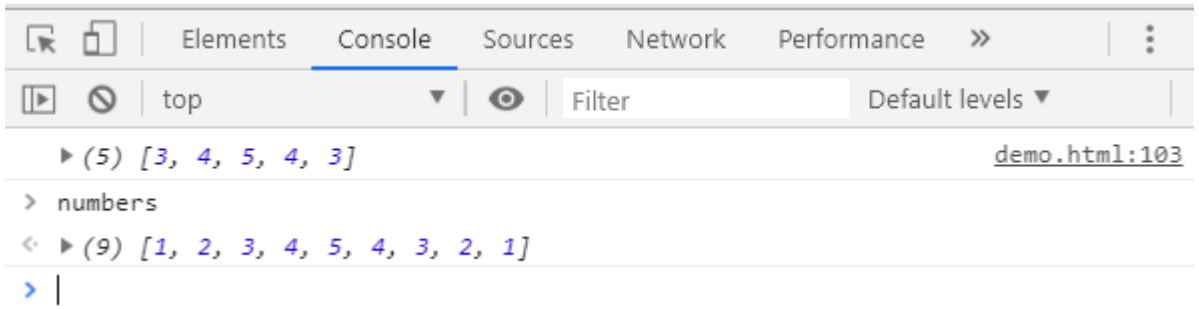
some方法

确保每个数组项目只要一个符合return的语句条件, 参会返回true

数组的高级方法之迭代方法的filter()

它利用指定的函数中return语句的条件, 把所有符合该条件的数组项目组合成一个新的数组返回
该方法不会修改原来的数组

```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];
var filterResult = numbers.filter(function (item, index, array) {
    return (item > 2);
});
console.log(filterResult); //[3,4,5,4,3]
```

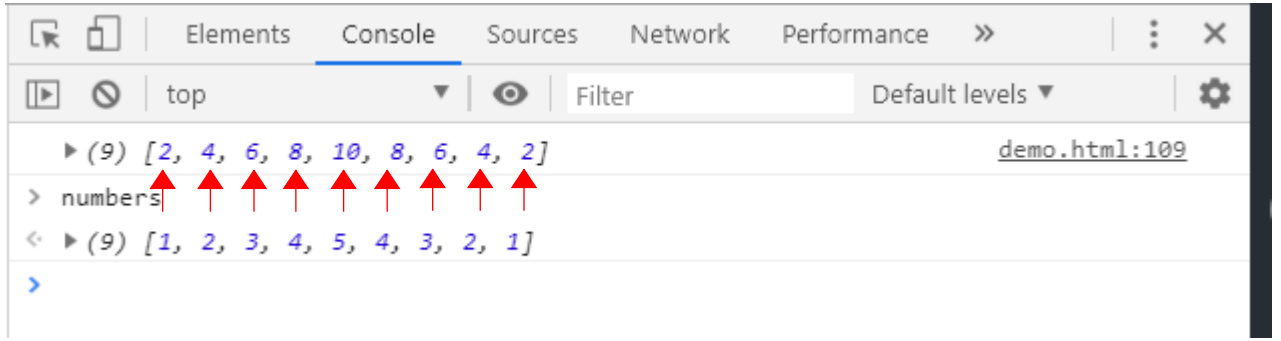


返回一个所有数值都大于2 的数组,

数组的高级方法之迭代方法的map()

map() 返回一个新数组，这个新数组的每一项都是在原始数组中的对应的数组项目传入函数经过一系列处理的结果。

```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];
var mapResult = numbers.map(function (item, index, array) {
    return item * 2;
});
console.log(mapResult); //[2,4,6,8,10,8,6,4,2]
```



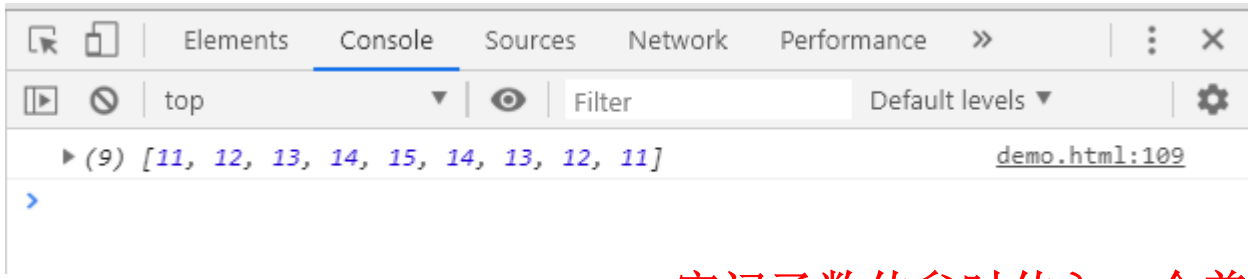
给数组中的每一项乘以2，然后返回这些乘积组成的数组

数组的高级方法之迭代方法的forEach()

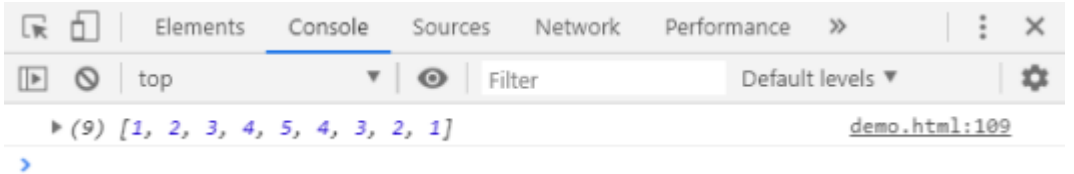
forEach()，它只是对数组中的每一项运行传入的函数。这个方法没有返回值，本质上与使用for 循环迭代数组一样。

该方法可以修改原数组

```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];
numbers.forEach(function (item, index, array) {
    array[index]+=10;
});
console.log(numbers);
```




```
var numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];
numbers.forEach(function (item, index, array) {
    item+=10;
});
console.log(numbers);
```



牢记函数传参时传入一个普通数据类型和引用类型的差别是啥

数组的高级方法之归并方法

ECMAScript 5 还新增了两个归并数组的方法：reduce()和reduceRight()。
这两个方法都会迭代数组的所有项，然后构建一个最终返回的值。其中，reduce()方法从数组的第一项开始，逐个遍历到最后。而reduceRight()则从数组的最后一项开始，向前遍历到第一项。

```
var values = [1, 2, 3, 4, 5];
var sum = values.reduce(function (prev, cur, index, array) {
    return prev + cur;
});
console.log(sum); //15
```

这两个方法都接收两个参数：一个函数和（可选的）作为归并基础的初始值。

传给reduce()和reduceRight()的函数接收4 个参数：前一个值、当前值、项的索引和数组对象。

数组的高级方法之归并方法

```
var values = [1, 2, 3, 4, 5];
```

第一次迭代

```
var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
});
```

上一次迭代的结果(1+2=3)作为下一次迭代的第一个参数

```
[1, 2, 3, 4, 5];
```

第二次迭代

```
 var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
 });
```

上一次迭代的结果(1+2+3=6)作为下一次迭代的第一个参数

```
[1, 2, 3, 4, 5];
```

第三次迭代

```
 var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
 });
```

数组的高级方法之归并方法

第四次迭代

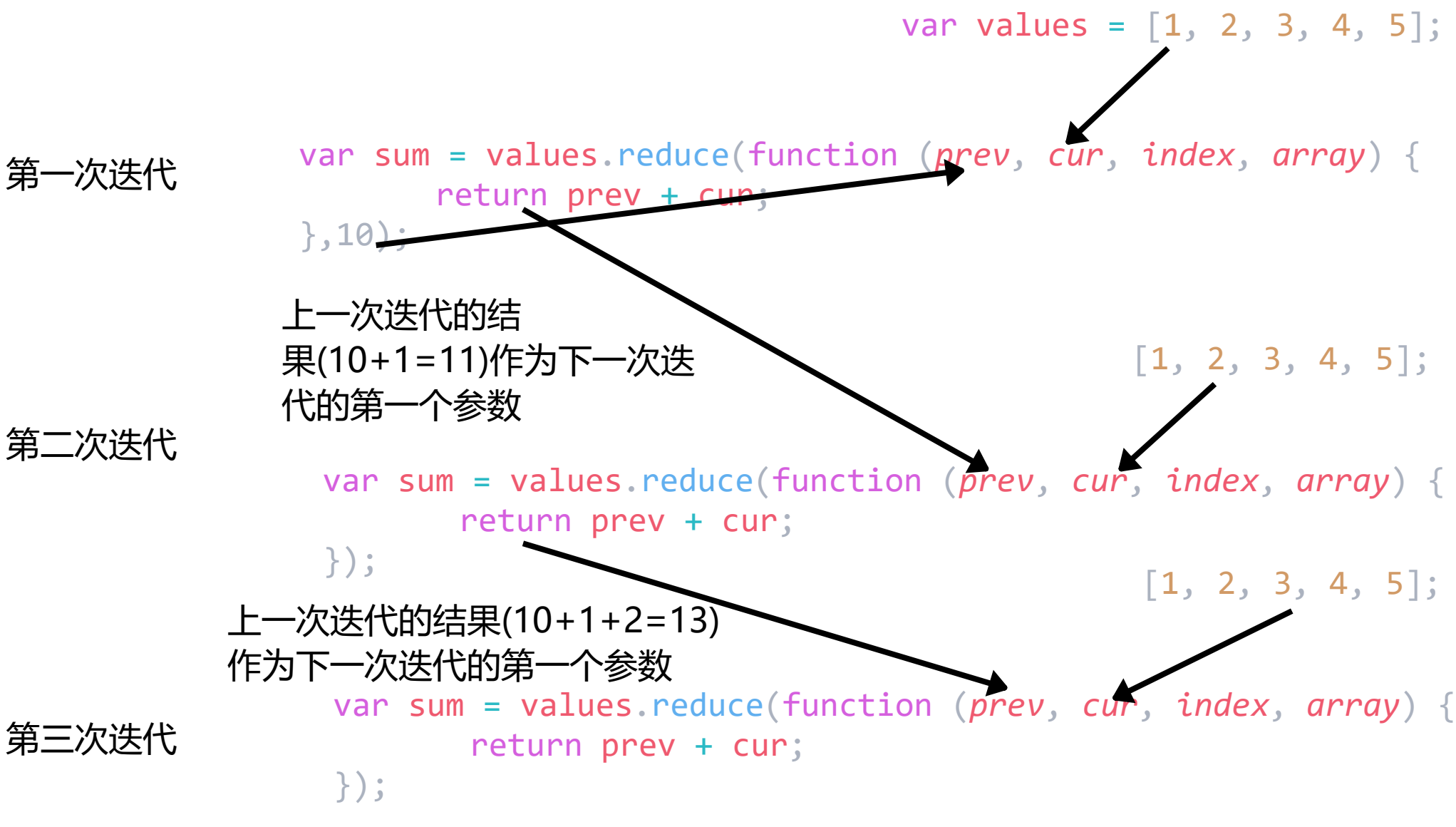
上一次迭代的结果(1+2+3+4=10)作为下一次迭代的第一个参数

```
var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
});
```

[1, 2, 3, 4, 5];

cur已经指向数组的末尾项目, 迭代停止
返回10+5=15

数组的高级方法之归并方法(有初始值版本)



数组的高级方法之归并方法(有初始值版本)

上一次迭代的结果(10+1+2+3=16)作为下一次迭代的第一个参数

第四次迭代

```
var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
});
```

[1, 2, 3, 4, 5];

上一次迭代的结果(10+1+2+3+4=20)作为下一次迭代的第一个参数

第五次迭代

```
var sum = values.reduce(function (prev, cur, index, array) {
        return prev + cur;
});
```

[1, 2, 3, 4, 5];

cur已经指向数组的末尾项目, 迭代停止

返回20+5=25

数学对象Math

数学对象Math之对象的属性(基本常量)

属　　性	说　　明
Math.E	自然对数的底数，即常量e的值
Math.LN10	10的自然对数
Math.LN2	2的自然对数
Math.LOG2E	以2为底e的对数
Math.LOG10E	以10为底e的对数
Math.PI	π的值
Math.SQRT1_2	1/2的平方根（即2的平方根的倒数）
Math.SQRT2	2的平方根

数学对象Math之方法(取最大, 最小值)

Math 对象还包含许多方法，用于辅助完成简单和复杂的数学计算。其中，min()和max()方法用于确定一组数值中的最小值和最大值。这两个方法都可以接收任意多个数值参数

```
var max = Math.max(3, 54, 32, 16);
alert(max); //54
var min = Math.min(3, 54, 32, 16);
alert(min); //3
```

数学对象Math之方法(取整)

将小数值舍入为整数的几个方法：Math.ceil()、Math.floor()和Math.round()。
这三个方法分别遵循下列舍入规则：

Math.ceil()执行向上舍入，即它总是将数值向上舍入为最接近的整数；
Math.floor()执行向下舍入，即它总是将数值向下舍入为最接近的整数；
Math.round()执行标准舍入，即它总是将数值四舍五入为最接近的整数;

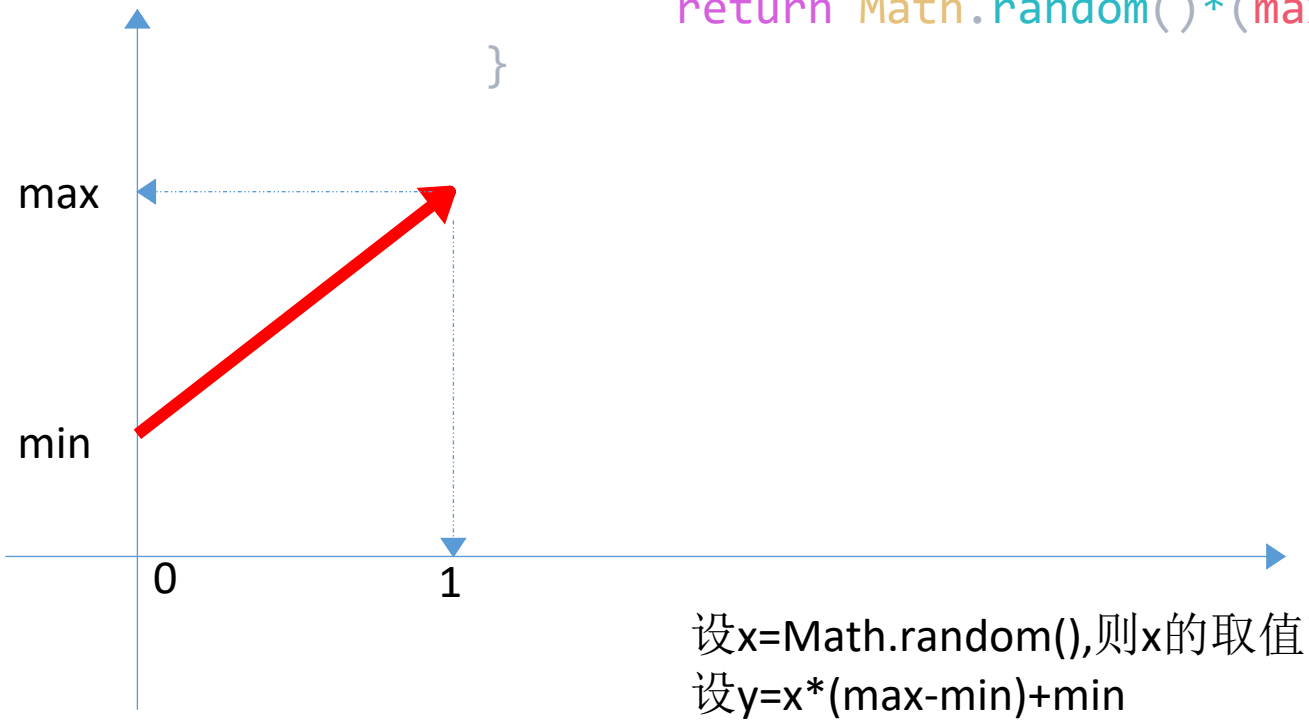
```
console.log(Math.ceil(25.9)); //26
console.log(Math.ceil(25.5)); //26
console.log(Math.ceil(25.1)); //26
console.log(Math.round(25.9)); //26
console.log(Math.round(25.5)); //26
console.log(Math.round(25.1)); //25
console.log(Math.floor(25.9)); //25
console.log(Math.floor(25.5)); //25
console.log(Math.floor(25.1)); //25
```

数学对象Math之方法(取随机数)

Math.random()方法返回大于等于0 小于1 的一个随机数

如果想要实现返回任意两个数min~max之间的随机数,我们可以改造一下

```
function random(min,max){
        return Math.random()*(max-min)+min;
}
```



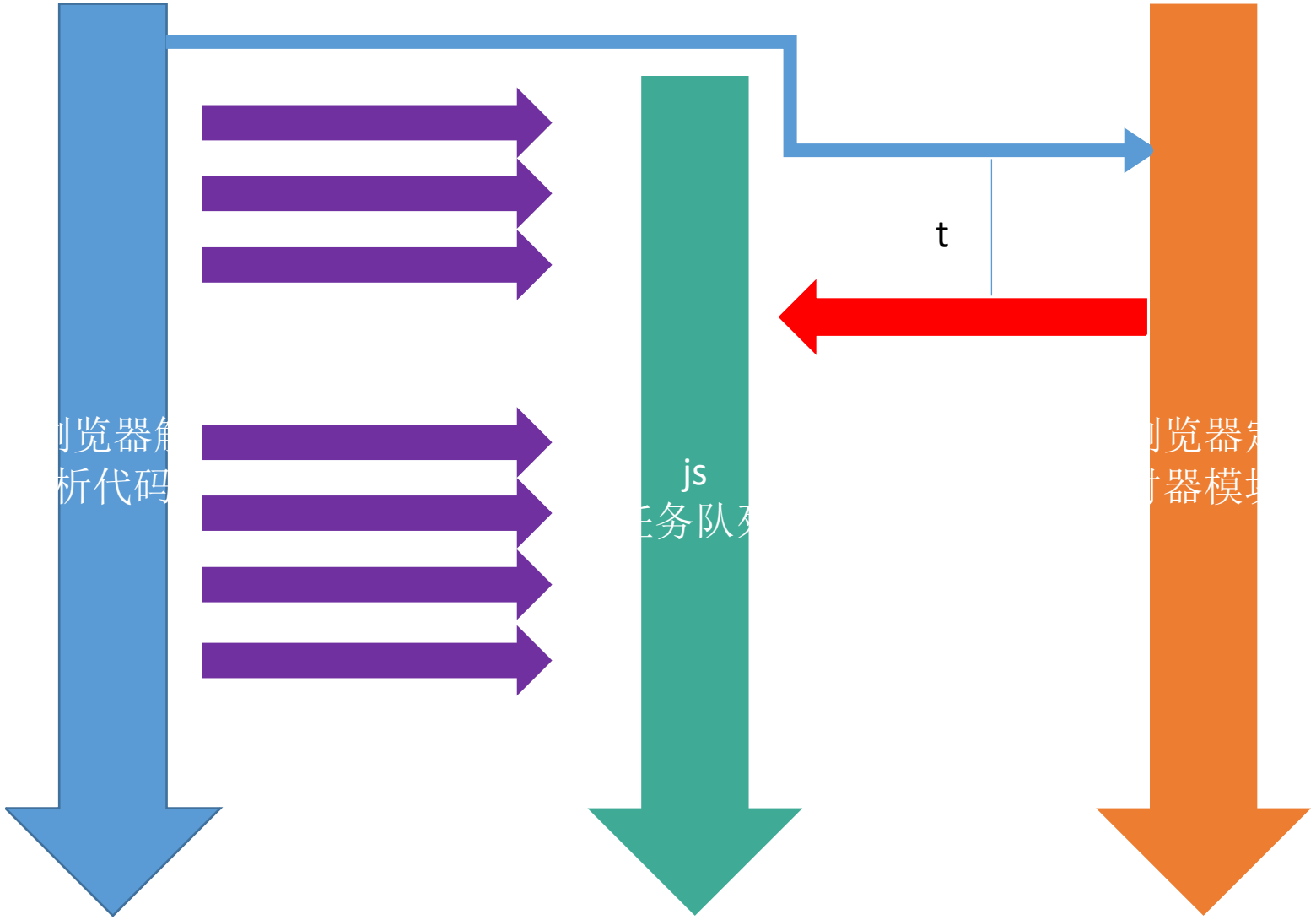
设x=Math.random(),则x的取值范围为0~1
设y=x*(max-min)+min

数学对象Math之方法(其他方法)

方　法	说　明	方　法	说　明
Math.abs(*num*)	返回*num*的绝对值	Math.asin(*x*)	返回*x*的反正弦值
Math.exp(*num*)	返回*Math.E*的*num*次幂	Math.atan(*x*)	返回*x*的反正切值
Math.log(*num*)	返回*num*的自然对数	Math.atan2(*y*,*x*)	返回*y*/*x*的反正切值
Math.pow(*num*,*power*)	返回*num*的*power*次幂	Math.cos(*x*)	返回*x*的余弦值
Math.sqrt(*num*)	返回*num*的平方根	Math.sin(*x*)	返回*x*的正弦值
Math.acos(*x*)	返回*x*的反余弦值	Math.tan(*x*)	返回*x*的正切值

延时器

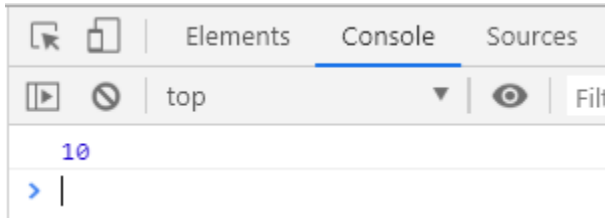
浏览器线程之延时器



setTimeout(function(){},t)

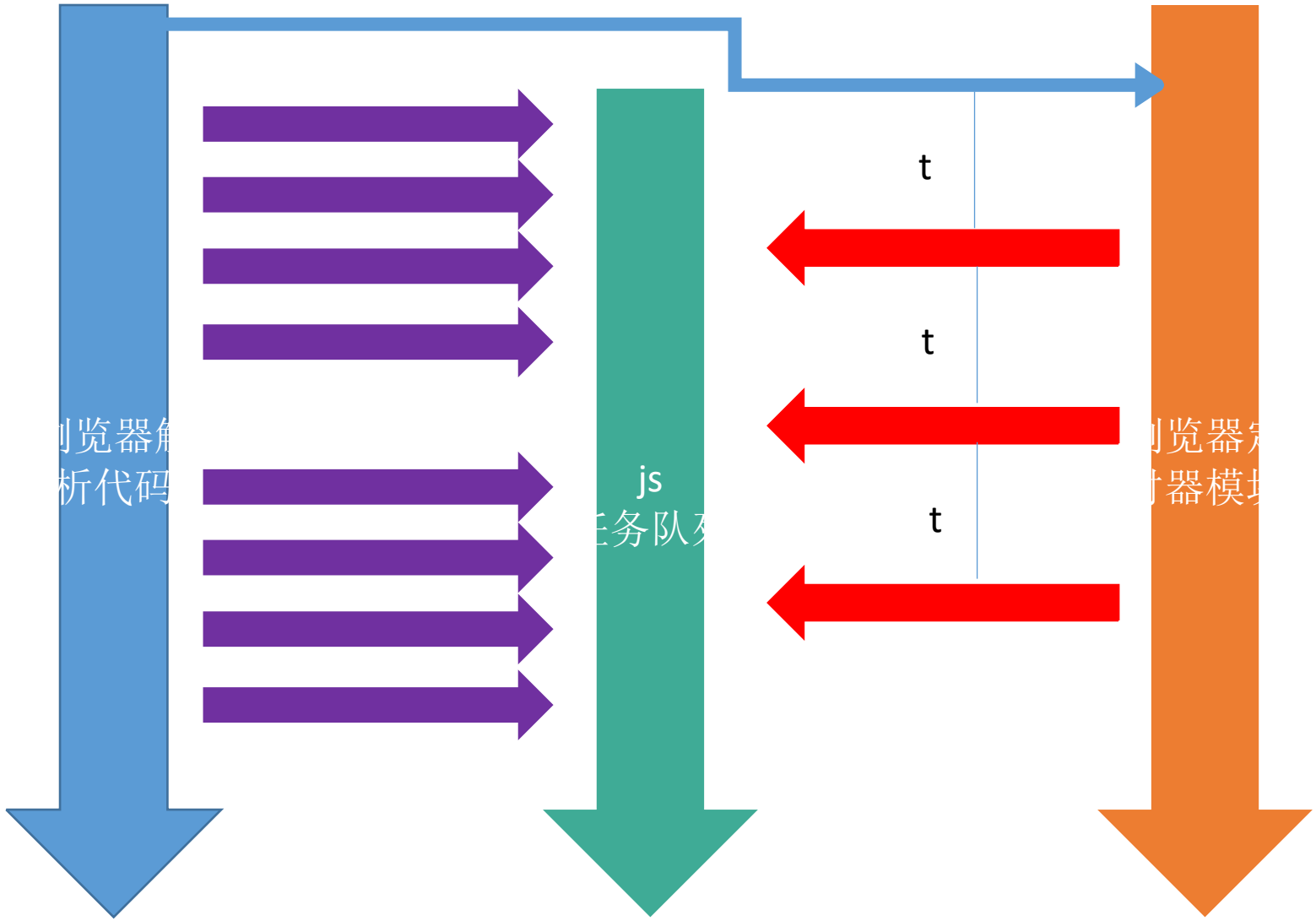
在t ms之后执行函数function(){};

```
setTimeout(function(){
    console.log(a);
},100);

let a =10;
```



定时器

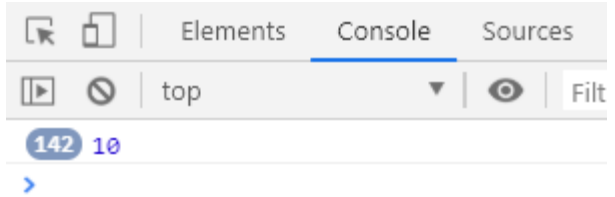
浏览器线程之定时器



setInterval(function(){},t)

每隔t ms执行一次函数function(){};

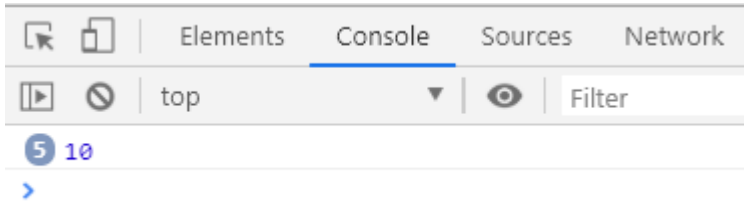
```
setInterval(function(){
    console.log(a);
},100);

let a =10;
```



永不停歇

浏览器线程之清除定时器

clearInterval(定时器名称)
删除之前设定的定时器




```
var timer=setInterval(function(){
    console.log(a);
},100);

let a =10;
setTimeout(function(){
    clearInterval(timer);
},500)
```