# k轮播图逻辑及正则表达式

## 轮播图逻辑

结构： 结构牢记（小盒子套大盒子）

样式： 外面小盒子根据图片的宽度进行设置 内部大盒子 宽度是根据图片的个数进行的设置的

行为：

1、点击按钮让图片先动起来（只点击右侧） 每次点击要准备走的距离都是-600，这个距离需要用定时器慢慢走，需要求每一步的距离 元素起始位置我们是可以获取到的， 结束位置我们也求出来；

2、 看什么时候元素停止，当刚好元素走的位置和开启求出来的结束位置一样的时候，停止定时器；

3、左边按钮和右边按钮 几乎一致， 只是点左边按钮的时候，元素要准备走的距离是600，其它的都不变，因此封装函数move(flag)

函数传参为true就是点右 false就是点左

4、无缝的操作 结构需要变化 最前面加最后一张 最后面加第一张； 在清除定时器的时候去判断是否元素走到了最后一张的位置和第一张的位置，如果走到，让元素顺便到相应的位置

5、小圆点变色 排他 关键点在求出哪一个小圆点变色，就是要拿到小圆点变色的下标，通过元素最终位置去求

6、点击小圆点，图片移动对应位置 要传递元素准备移动的最终位置 跟点击的小圆点下标相关，move里面参数要根据类型进行判断到底是点的按钮还是点的小圆点，因为传过来的参数不同

7、自动轮播 ，直接调用move传递参数为true就ok

8、自动轮播鼠标行为的同步 鼠标移入清除自动轮播 鼠标移出重启自动轮播

9、移入之后，鼠标点击按钮或者小圆点，都要去把自动轮播的下标值 进行更新， 否则没法同步；

## 正则表达式

### 1.正则的概念

是什么？

正则表达式是描述字符模式的对象。

正则表达式用于对字符串模式匹配及检索替换，是对字符串执行模式匹配的强大工具。

简单点：正则表达式是一种字符串匹配规则；正则就是让我们用来在一个字符串当中 去查找符合正则规则的字符串或者判断字符串是否符合正则规则；

为什么？

‘11111111111’

假设我想要知道一个字符串当中是否有6，该如何去做

假设我想知道字符串当中是否有数字又该如何

假设我想从字符串当中找到abcd怎么去做

假设我想知道这个电话号码是否合法

。。。。。。。。

### 2.创建方式

字面量创建

var patt=/pattern/modifiers;

构造函数创建

var patt=new RegExp(pattern,modifiers);

pattern（模式） 描述了表达式的模式

modifiers(修饰符) 用于指定全局匹配、区分大小写的匹配和多行匹配

注意：当使用构造函数创造正则对象时，需要常规的字符转义规则（在前面加反斜杠 \）。比如，以下是等价的：

var re = new RegExp("\\d+");

var re = /\d+/;

### 3.正则表达式规则写法

修饰符

修饰符用于执行区分大小写和全局匹配:

i:忽略大小写

g: 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

m:执行多行匹配

方括号

方括号用于查找某个范围内的字符：

[abc] 查找abc任意一个

[^abc] 查找不是abc的任意一个

[0-9] 查找任意一个数字 \d

[a-z] 查找任意一个小写字母

[A-Z] 查找任意一个大写字母

元字符

. 匹配任意字符不包含\n（换行和结束符）

\d 任意数字 等价于[0-9]

\D 任意非数字 等价于[^0-9]

\w 任意单词字符 数字 字母 下划线 [a-z A-Z 0-9 \_]

\W 任意非单词字符 [^a-z A-Z 0-9 \_]

\s 任意空白字符

\S 任意非空白字符

\b 单词边界 'i loveyingyou zhao li ying' /\bying/

\B 非单词边界

\n 换行符

\f 换页符

\r 回车符

\t 制表符

\v 垂直制表符

量词

+ 1个或者多个前一个字符 \d+ 代表\d有一个或者多个

\* 0个或者多个前一个字符 \d\* 代表\d有0个或者多个

？ 0个或者1个前一个字符 \d? 代表\d有0个或者1个

{n} n个前一个字符 \d{2}

{m,n} m到n个前一个字符 \d{2,4}

{m,} 至少m个前一 个字符 \d{2,}

$ 结尾 ^\d{11}$ 做严格判断

^ 开头

分组

（）分组后的反向引用

|

贪婪和非贪婪

量词后面的？代表非贪婪

### 4.字符串正则相关方法

#### 1）正则对象的方法：

Reg.test()

test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

判断字符串当中是否含有数字

判断字符串当中是否含有abcd

判断字符串是否是一个11位的数字

Reg.exec()

exec() 方法用于检索字符串中的正则表达式的匹配。

该函数返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

注意：此方法每次只会返回一个结果，如果要找到所有的，需要循环去调用必须全局匹配修饰；

#### 2）字符串方法使用正则

str.match()：在字符串中搜索符合规则的内容，搜索成功就返回内容，格式为数组，失败就返回null。如果不加g，那么返回第一次符合的结果，加g返回所有结果的数组，找一个是详细进行展示 找多个只是在一个数组展示找到的内容子串

str.search() ：在字符串搜索符合正则的内容，搜索到就返回出现的位置（从0开始，如果匹配的不只是一个字母，那只会返回第一个字母的位置）， 如果搜索失败就返回 -1 只能返回第一次；

注意：上面两个方法类似正则的方法 test和exec

Reg.exec()和str.match()

当不使用全局匹配时，两者的匹配效果都是返回第一次匹配成功的结果；

reg.test（）和str.search（） 前者返回的是true或者false 后者返回到的是匹配到的子串位置或者-1；

str.replace():查找符合正则的字符串，就替换成对应的字符串。返回替换后的内容。这个方法相当于做了两件事 先查找匹配 然后再用新的串把匹配到的串替换掉

案例:

给一个字符串，如果里面含有2个到10个连续数字就打印i love you!

判断手机号码是否合法

判断邮箱是否合法

将手机号的中间4位替换位\*\*\*\*

将身份证号的最后4位替换位\*\*\*\*

从一篇文章当中匹配所有的电话号码

作业：

完成轮播图逻辑

正则案例过一遍