JS项目

1. 跑马灯

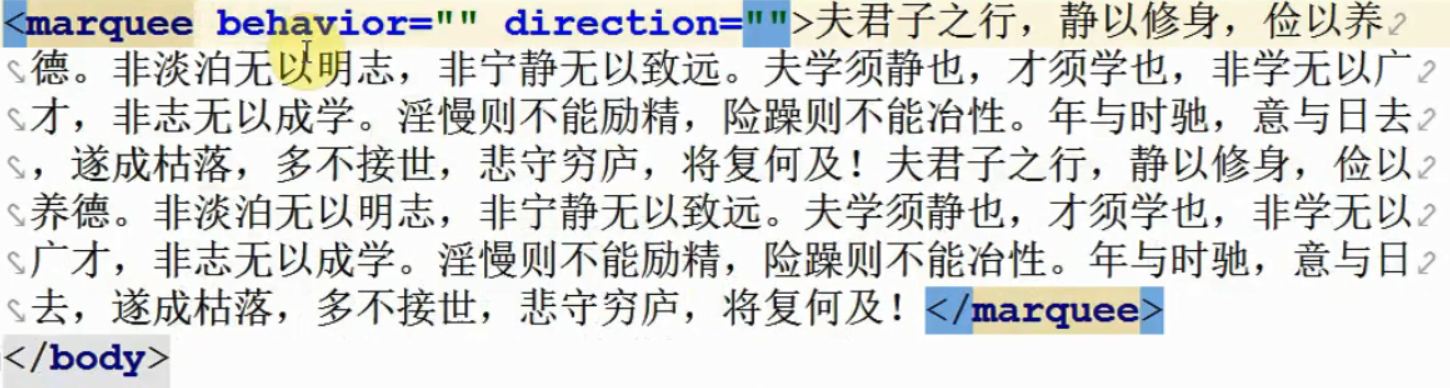
老方法: <marquee behavior=”” direction=””></marquee>

缺点: 1.不好实现无缝连接

2.开始显示内容的时候有空白

3. 性能消耗大

4.目前marquee标签基本舍弃



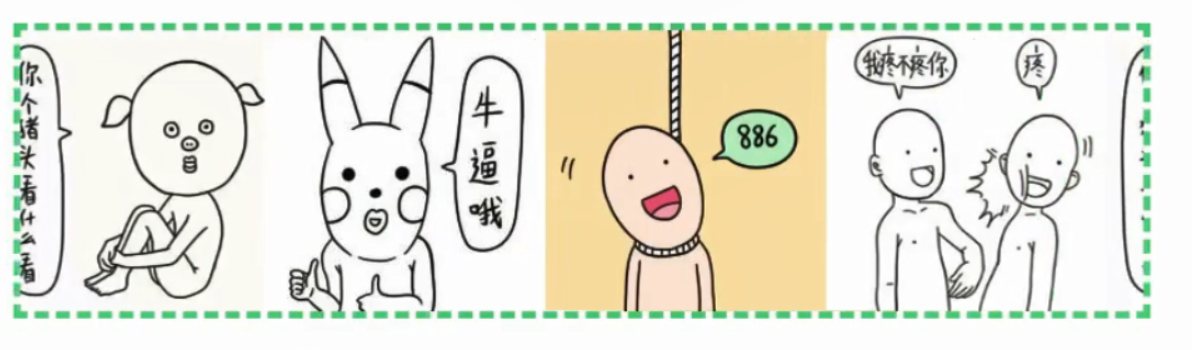
效果图:



Nowadays:

跑马灯都是基于JS动画或者CSS3动画实现的

案例展示:



原理:

让li浮动.



为实现无缝

1. 把warpper中原有的li整体克隆一份放到容器的末尾.以此来实现无缝联播.

Let wrapperList =wrapper.querySelectorAll(“li”);

Let frt=document.createDocumentFragment()



克隆/wrapper.innerHTML +=wrapper.innerHTML;

jQuery

类库.插件,UI组件,框架

1. 类库:JQ/ZEPTO … 提供一些真实项目中常用的方法,任何项目都可以把类库导入进来,调取里面的方法实现自己需要的业务逻辑
2. 插件:具备一定业务功能,例如:我们可以封装轮播图插件.选项卡插件,模态框插件等,(插件规定了当前这个功能) swiper/iscroll/jquery-drag/darepicker/ECHARTS
3. UI组件:把结构,css,js全部都封装好了.直接导入即可(偶尔需要修改)bootstrap,
4. 框架:具备一定的编程思想,要求我们按照框架的思想开发,一般框架提供了常用的类库方法,提供了强大的功能插件,有的也提供了强大的UI组件… React(React native)/Vue/Angular/Backbone/Sea.js/Require.js…

**jQuery非常优秀的js”类库”**

**>**基于原生js封装的一个类库,提供了很多方法,而且这些方法是兼容所有浏览器的

看1.11.3

dos命令一样的 cd 或者拖进来

jQuery 源码分析

(**function**(){  
 ***jQuery***=**function** ( selector, context) {  
 **return new jQuery**.**fn**.init(selector, context);  
 };  
 *//jQuery是一个类,在它的原型上提供了很多的属性和方法,供jq的实例提取出来.* jQuery.**fn**=jQuery.**prototype**={  
 **jquery**:version,  
 constructor:jQuery *//当前类的原型重定向后,自己开辟出的堆内存是没有constructor的,需要手动增加来保证自己的完整性.* };  
 *//给jq原型上提供了extend方法,同时把jq当做一个普通对象,给这个对象设置了一个私有方法.  
/\*  
\* jq是一个类(也是一个普通对象):函数的两种角色  
// \* 1.放到jq原型上的(jquery.fn/jquery.prototype这里面的jq实例调取使用的 这类方法以后用的时候直接jQuery.xxx()即可\*/* jQuery.extend = jQuery.**fn**.extend = **function**() {}  
 *//extend是把一个对象中的属性和方法扩展到指定的对象上* })()  
 *// 把下面的方法加到extend  
 // isFunction: function( obj ) {  
 // return jQuery.type(obj) === "function";  
 // },  
 //  
 // isArray: Array.isArray || function( obj ) {  
 // return jQuery.type(obj) === "array";  
 // },  
 // jQuery.fn.extend().给原型上又扩展了一个方法  
  
 // init = jQuery.fn.init = function( selector, context ) {}  
 // init.prototype = jQuery.fn;把init当做一个类,这个类的原型指向jq的原型(init的实例最后找到的就是jq这个类原型上的方法=>init的实例其实也可以理解为jq的原型*

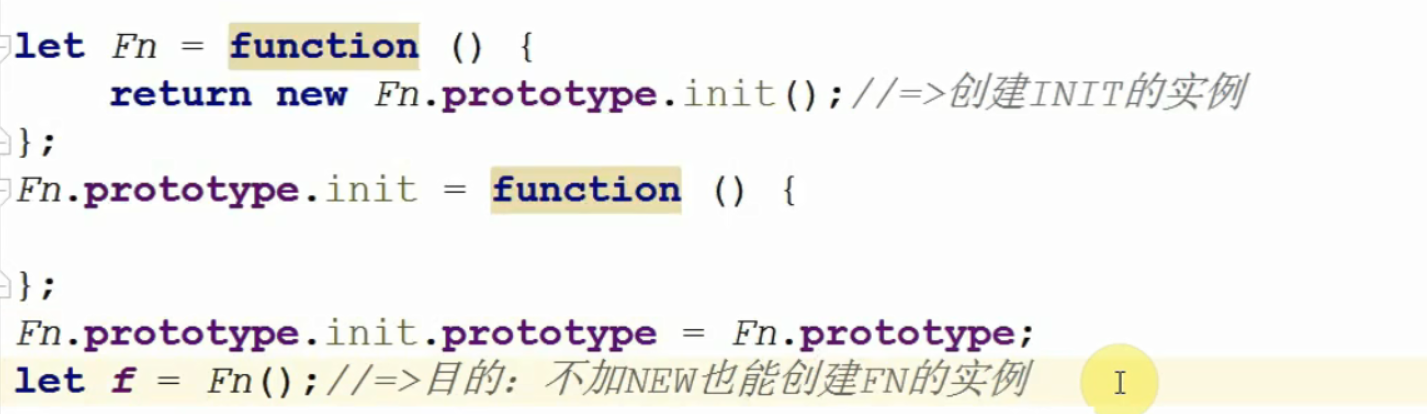
最后暴露给为window

**if** ( **typeof** noGlobal === strundefined ) {  
 window.jQuery = window.$ = jQuery;  
}

$(),和jquery( )执行都是执行jQUery=function(){}这个方法.

**$**().filter() *//创建一个jquery类的实例.可以调取jq.fn中的方法* **$**.isFunction() *// 把jq当做一个普通对象,直接的使用对象上的扩展的那些私有属性和方法(这些方法和实例没有关系)*

绕的要死



选择器的源码解析

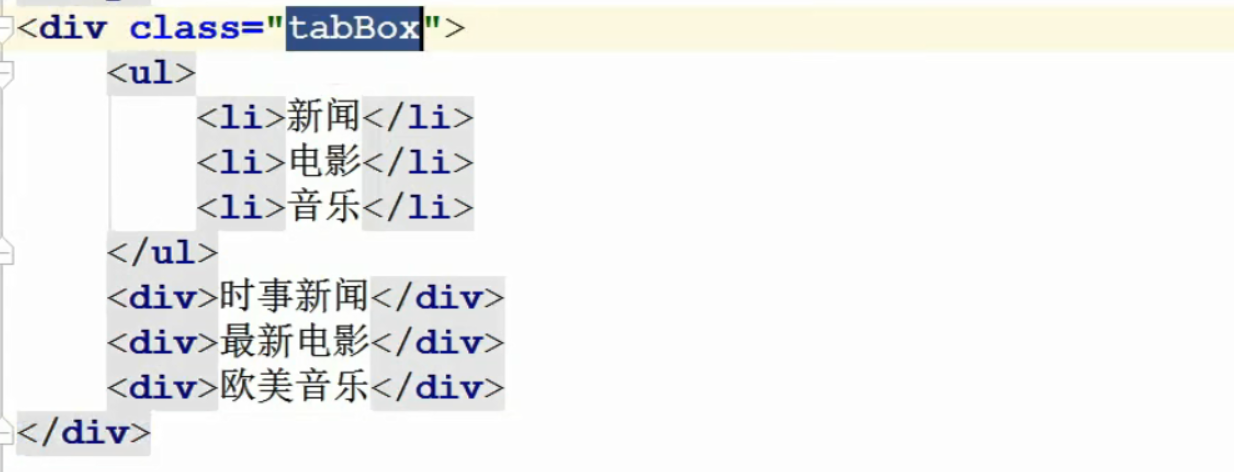
Jquery选择器:基于各种选择器创建一个JQ实例(jq对象)

1. selector 选择器的类型(一般都是字符串,但是支持函数或者元素对象)
2. context 基于选择器获取元素时候指定的上下文(默认document)

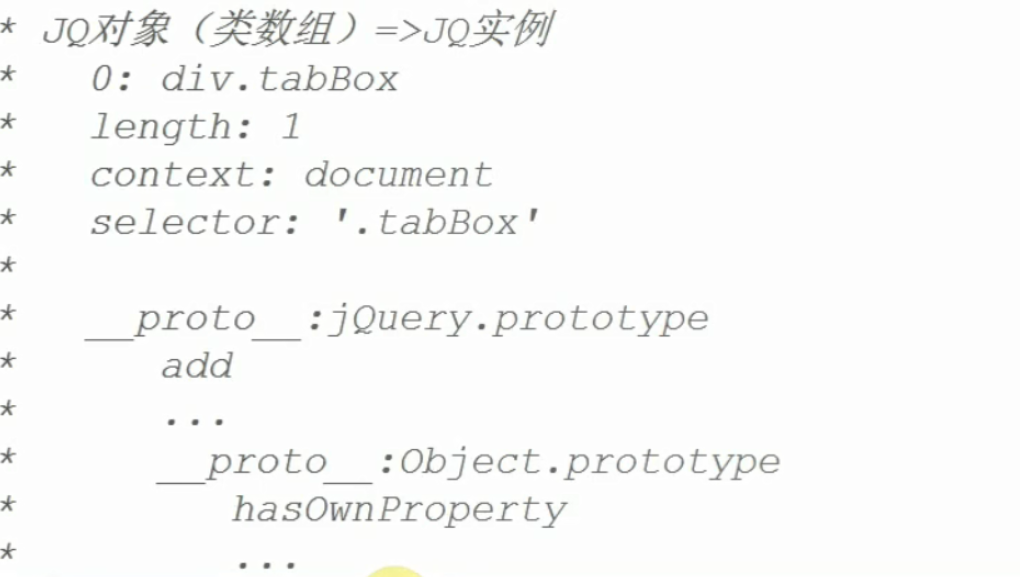
JQ对象:一个类数组结构(jq实例),这个类数组集合包含了获取到的元素

$();

tab选项卡



原型机制:



支持后代选择器 $(".taxBox li") 类似子代那种

获取页面中的元素对象

基于原生js提供的属性和方法获取 ->原生js对象

className onclick

基于jq选择器获取 => ‘jq对象’

可以调取jq原型上提供的属性和方法

add/find

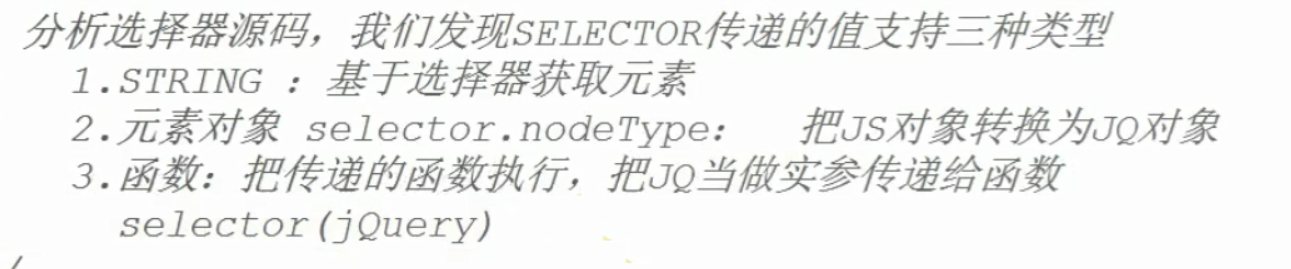
把jq对象和原生js之间相互转换

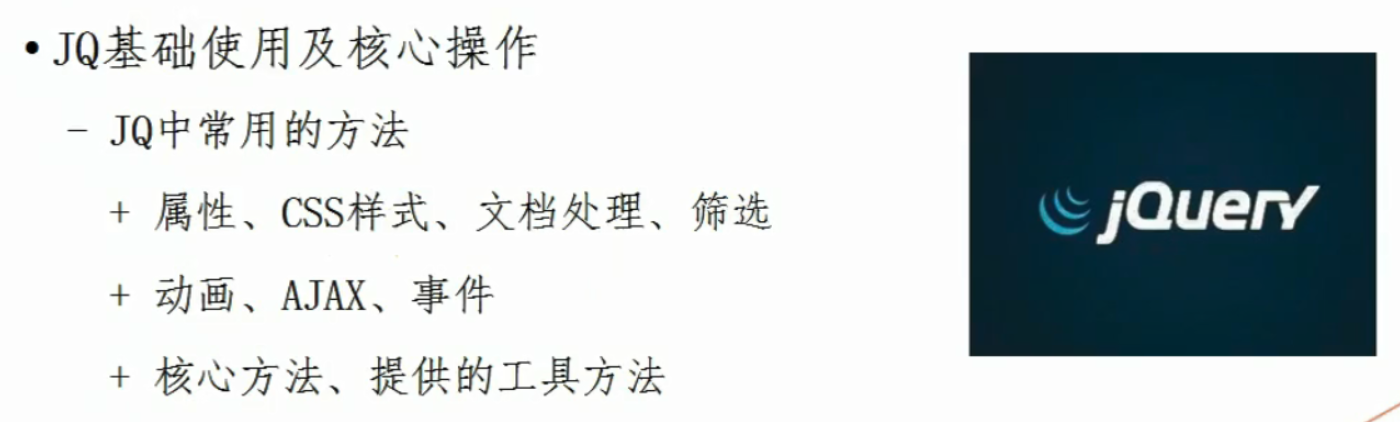
jq对象是一个类数组集合,集合中每个索引对应的都是原生js对象,我们基于索引值获取即可

**let *$tabBox***=**$**(**'.tabBox'**);*//变量名前面事宜$开始的,一般代表基于jq选择器获取的结果***let *tabBox***=***$tabBox***.get(0); *//get 是jq\原型上提供的方法,供jq实例基于索引值获取到指定的就是对象  
// $tabBox.eq(0);它也是基于索引值集合中的某一项,只不过获取的js对象,eq会把获取到的结果包裹成一个新的jq对象(jq实例返回)*



把原生js转换成jq



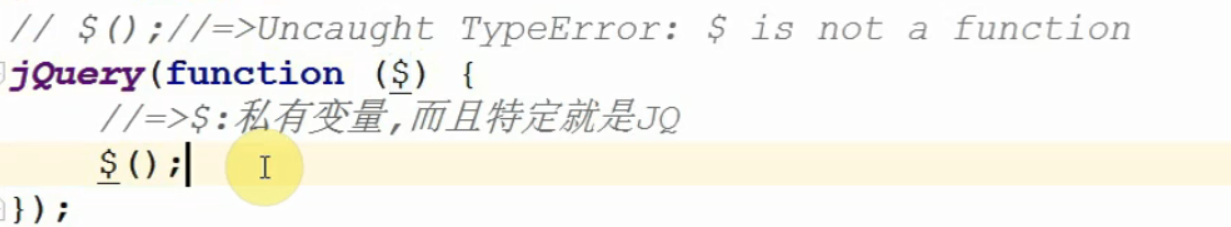


给body添加一个





JQ常用方法



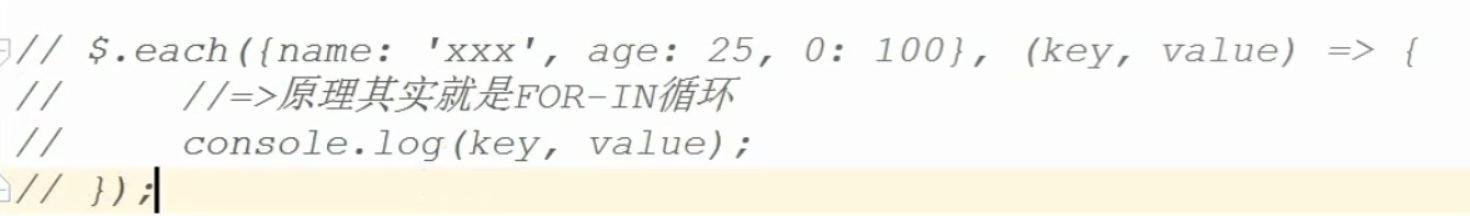
each: jq中的each方法是用来进行遍历的(类似于数组的for-each)

可遍历内容:

1. 数组

**$**.each([12,23,45],(index,item)=>{  
   
})

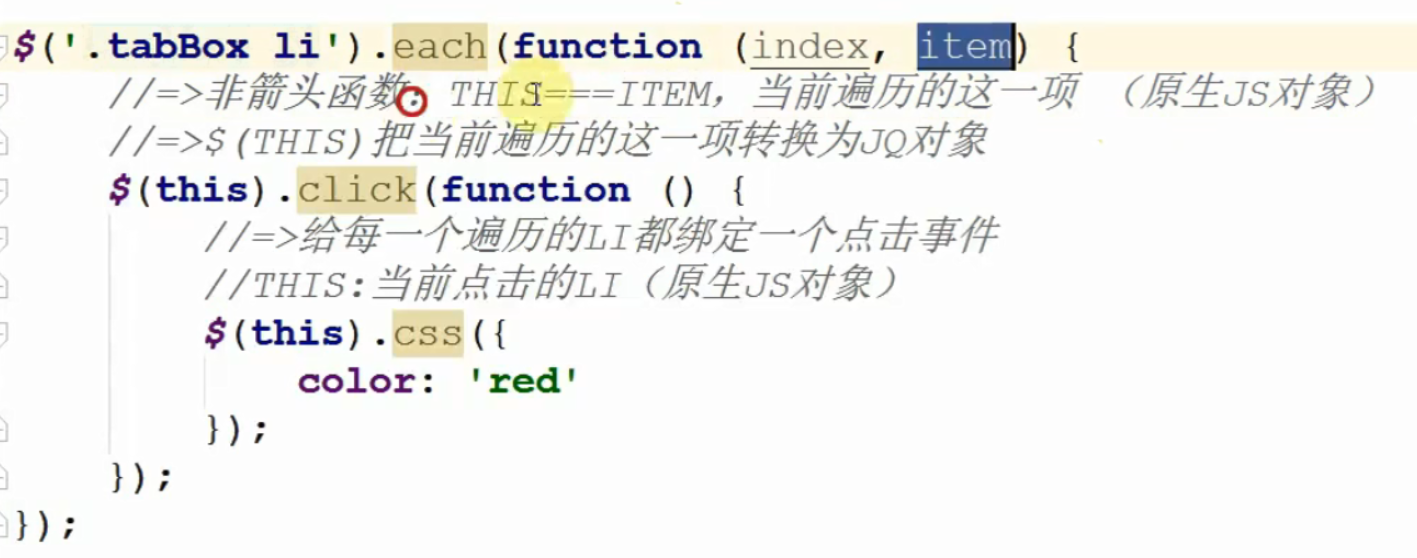
1. 对象



1. 类数组(jq对象)

三种each

1. 给jq设置私有属性 $.each
2. 给实例设置公有属性 $(selector).each
3. 内置的each



瀑布流

案例最好的是花瓣网

效果:多列的不规则排列,每一列中有很多内容,每一项内容的高度不定,最后我们按照规则排列,三列之间不能相差太多高度

实现: 首先获取需要展示的数据(假设有50条,共三列),把50条数据中的前三条以此插入到三列中(目前有的列高有的低),接下来在拿出三条数据,但是本次插入不是依次插入,而是需要先把当前三列按照高矮进行排序,

样式: 

原理: