* 严格模式

所谓严格模式，从字面上就很好理解，即更严格的模式，在这种模式下执行，浏览器会对JS的要求更苛刻。

* + 设立"严格模式"的目的，主要有以下几个：

消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

消除代码运行的一些不安全之处，保证代码运行的安全；

提高编译器效率，增加运行速度；

为未来新版本的Javascript做好铺垫

* "严格模式"有两种调用方法，适用于不同的场合。

针对整个脚本文件 针对单个函数

* + 语法和行为改变

全局变量显式声明：严格模式下，变量都必须先用var命令声明，然后再使用。

禁止使用with语句：因为该语句无法在编译时就确定，属性到底归属哪个对象。

禁止this关键字指向全局对象

函数不能有重名的参数

禁止八进制表示法

禁止使用arguments.callee

* this的应用及指向问题

this是Javascript语言的一个关键字。它代表函数运行时，自动生成的一个内部对象，只能在 函数内部使用。随着函数使用场合的不同，this的值会发生变化。但是有一个总的原则，那就 是this指的是调用函数的那个对象。

* this的使用情况：

1、作为普通函数调用，这是函数的最通常用法，属于全局性调用，因此this就代表全局对象。

2、作为对象方法的调用，函数还可以作为某个对象的方法调用，这时this就指这个上级对象。

3、作为构造函数调用，所谓构造函数，就是通过这个函数生成一个新对象（实例）。这时，this 就指这个新对象（实例）。

* apply 、 call 、bind的介绍（函数的方法）

apply 、 call 、bind 三者都是用来改变函数的this对象的指向的；第一个参数都是this要指 向的对象，也就是想指定的上下文；从第二个参数开始，就是函数的本身的参数；

apply 、call 则是立即调用 。

bind 是返回对应函数，便于稍后调用；

我们都知道定时器的匿名函数里不能写this，它会指向window但有了bind方法以后，我们 可以随意设定this 的指向

* JSON.parse()和JSON.stringify()
* ECMAScript6简介

ECMAScript 6.0（以下简称ES6）是JavaScript语言的下一代标准，已经在2015年 6月正式发布了。它的目标，是使得JavaScript语言可以用来编写复杂的大型应用程 序，成为企业级开发语言。

* LET和CONST命令

let命令的特点：

1. ES6新增了let命令，用来声明变量。它的用法类似于var，但是所声明的变量，只在let命令所在的代码块内有效。。
2. 不存在变量提升（前置），变量一定要在声明后使用，否则报错。
3. 不允许在相同作用域内，重复声明同一个变量。报错
4. ES6允许块级作用域的任意嵌套，外层无法读取内层作用域的变量，反之可以。
5. 允许在块级作用域内声明函数

const命令的特点：

1. 声明一个只读的常量。一旦声明，常量的值就不能改变，必须立即初始化
2. 不可重复声明。
3. const的作用域与let命令相同：只在声明所在的块级作用域内有效。

* 解构赋值（本质上就是一种匹配模式）

1、数组的解构赋值

ES6允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构.

只要等号两边的模式相同，左边的变量就会被赋予对应的值

如果等号的右边不是数组，那么将会报错，同时允许使用默认值。

2、对象的解构赋值

变量必须与属性同名

对象的解构赋值的内部机制，是先找到同名属性，然后再赋给对应的变量。真正被 赋值的是后者，而不是前者。和对象的赋值顺序没有关系。

3、字符串的解构赋值。

赋值的字符串被转换成了一个类数组的对象分别赋值给对应的变量。

4、函数参数的解构赋值--应用。

* 数据结构--Set

ES6提供了数据结构set. 它类似于数组，但是成员的值都是唯一的，没有重复的值。

Set 本身是一个构造函数，用来生成 Set 数据结构，数组作为参数。

1. 创建一个Set；
2. Set类的属性--size
3. Set类的方法

Set.add(value) 添加一个数据，返回Set结构本身，允许进行链式操作。

Set.delete(value) 删除指定数据，返回一个布尔值，表示删除是否成功。

Set.has(value) 判断该值是否为Set的成员，返回一个布尔值。

Set.clear() 清除所有的数据，没有返回值。

keys()：返回键名的遍历器

values()：返回键值的遍历器

entries()：返回键值对的遍历器

forEach()：使用回调函数遍历每个成员

* 数据结构--map

JavaScript 的对象（Object），只能用字符串当作键。这给它的使用带来了很大的限制。ES6 提供了 Map 数据结构。它类似于对象，但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键。

1. 创建一个map
2. map类的属性--size
3. map类的方法

map.set(key,value) :设置键名key对应的键值value,然后返回整个map结构，如 果key已经有值，则键值会被更新，否则就生成该键，可以链式操作。

map.get(key) get方法读取key对应的键值，如果找不到key，返回undefined

map.delete(key) 删除某个键，返回true.如果删除失败，返回false.

map.has(key) 方法返回一个布尔值，表示某个键是否在当前map对象之中。

map.clear() 清除所有数据，没有返回值

map.keys() 返回键名的遍历器

map.values() 返回键名/键值的遍历器

map.entries() 返回键值对的遍历器

map.forEach(function(value,key,map本身){}) 使用回调函数遍历每个成员。

* symbol--表示独一无二的值，它是js中的第七种数据类型。

基本数据类型：null undefined number boolean string symbol

引用数据类型：object

Symbol():首字母大写。

symbol 函数前不能使用new,否则会报错，原因在于symbol是一个原始类型的 值，不是对象。

1、symbol函数接收一个字符串作为参数，表示对symbol的描述，主要是为了 在控制台显示，或者转为字符串的时候，比较容易区分。描述的内容相同仅仅表示 的是描述，不代表其他的意义。

2、symbol数据类型转换。

symbol可以利用String()、toString()转换成字符串

利用Boolean()或者！转换成布尔值(只有true值)

不能转换成数字。

不能使用任何运算符进行运算。

1. 作为对象的属性名。
2. 注意事项

不能直接使用[symbol()]作为键值，而且也不能被for...in遍历出来。但可以通过 Object.getOwnPropertySymbols(obj)方法获取一个对象所有的symbol属 性。

* 函数及=>函数

函数默认值

箭头函数有几个使用注意点。

1、函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。This不会改变了。

2、不可以当作构造函数，也就是说，不可以使用new命令，否则会抛出一个错误。

3、不可以使用arguments对象，该对象在函数体内不存在。

4、箭头函数一定是匿名函数。

* 字符串扩展方法

1. 模板字符串
2. repeat(n)方法返回一个新字符串，表示将原字符串重复n次。
3. includes()：返回布尔值，表示是否找到了参数字符串。
4. startsWith()：返回布尔值，表示参数字符串是否在源字符串的头部。
5. endsWith()：返回布尔值，表示参数字符串是否在源字符串的尾部。

* 数组的扩展方法

1. Array.from()方法用于将对象转为真正的数组
2. Array.of()方法用于将一组值，转换为数组。
3. fill()fill方法使用给定值，填充一个数组。

* 对象的扩展方法

1. 对象的简洁表示法
2. object.assign()用于对象的合并，将源对象的所有可枚举属性，复制到目标对象。

* 掌握class的写法及继承
* 综合应用

1. 了解class面相对象编程
2. 掌握ES6语法的灵活运用
3. 封装拖拽效果
4. 封装常见兼容（封装一个函数，通过传入对应的id,class,tagName能够获取对应的元素）

function getElement（）{

}

//获取id getElement(‘#box’)；//1个

//获取class getElement(‘.box’); //一组，考虑兼容

//获取tagname getElement(‘li’); //一组，如果传入非html标签，报错