1. 使用JS在页面(body)中输出一个内容：document.write();
2. Js基本语法
3. 注释 /\*\*/ //
4. **Js严格区分大小写**
5. Js每条语句以；结尾
6. Js会忽略多个空格和换行
7. 变量，可以保存常量，而且变量是可以改变的
8. 声明变量：用var关键字
9. 标识符：
10. 标识符可以有字母、数字、\_、$
11. 标识符不能以数字开头
12. 标识符不能是es中的关键字或保留字
13. 标识符一般采用驼峰命名法：首字母小写，每个单词的第一个字母大写
14. 数据类型（字面量的类型）：String（字符串）、Number（数值）、Boolean（布尔值）、Null（空值）、Undefined（未定义）、Object（对象）
15. 前五个都是基本数据类型，最后一个是引用数据类型
16. String字符串：引号不能嵌套
17. Number：包括整数和浮点数，NaN也是数值类型表示not a number
18. Boolean：true/false
19. Null和Undefined：

Null类型的值只有null一个，用来表示一个为空的对象；用typeof检查null时回返回object

Undefined类型的值只有undefined一个，当声明一个变量但未赋值时它就是undefined；使用typeof检查undefined时返回undefined

1. 可以使用typeof检查变量的类型 【typeof a】
2. 强制类型转换
3. 将其它数据类型转换为String：

1、调用被转换数据类型的toString()方法，该方法不会影响原变量，它会将转换的结果返回。但是，null和undefined没有toString()方法

2、调用String()函数，并将被转换数据作为参数传给函数

1. 将其它数据类型转换为Number：

1、调用Number()函数，字符串转为Number时，如果是纯数字字符串，直接转换为数字；如果字符串中有非数字的内容，则转换为NaN；如果字符串是一个空串或一个全是空格的字符串转换为0；布尔值转数字时，ture为1，false为0；Null转换为数字为0；Undefined转数字是NaN

2、parseInt()把一个字符串转换为一个整数、parseFloat()把一个字符串转换为一个浮点数，这两个函数专攻字符串转数字，将一个字符串中的**有效的**整数、小数内容取出来。如果对非String使用该俩函数，现将其转换为String然后再操作。

可以在函数中传递第二个参数，指定数字的进制

1. 将其它数据类型转换为Boolean
2. 调用Boolean()函数，数字转布尔值，除了0和NaN其余都是true；字符串转布尔值，除了空串其余都是ture；Null和Undefined转布尔为false；Object转换为布尔为ture
3. 隐式类型转换，为任意类型的值进行两次非运算
4. 运算符（操作符）：通过运算符可以对一个或多个值进行运算，并获取运算结果。typeof就是运算符返回值为字符串，用来获取一个值的类型
5. 算数运算符：

“+”：

当对非Number类型的值进行运算时，会讲这些值转换为Number然后进行运算；

在任何值和NaN做运算，结果都为NaN；

如果对两个字符串进行开发运算，会将字符串拼串并返回；

任何值和字符串做加法，都会先转换为字符串，然后和字符串做拼串并返回

1. 一元运算符：只需要一个操作数 （正负号）,对于非Number类型的值，会先转换成Number再运算，可以对其他的数据类型使用“+”将其转换为number类型
2. 自增和自减
3. 自增++：在变量自身基础上加1。对一个变量自增以后，原变量的值会立即自增1。

后++和前++，无论是哪一种都会立即使原变量的值自增1，不同的是，a++和++a的值不同，a++表达式的值等于原变量的值（自增前的值），而a的值是自增1后的值；++a表达式的值等于自增后的值 ，a的值也是自增1后的值

1. 自减--：在变量自身基础上减1。规则和自增相同
2. 逻辑运算符：！非 &&与 ||或

做“非”运算：就是取反，对于非布尔值进行运算，会先将其转换为布尔值，再进行运算；

做“与”运算：布尔值运算，第一个值是false时，不会考虑第二个值；第一个值为true时，会检查第二个值。对于非布尔值进行“与”运算，会先将其转换为布尔值，然后再运算，并返回原值（如果第一个值时true，则返回第二个值；如果第一个值为false，则直接返回第一个值）；

做“或”运算：布尔值运算，第一个值时ture时，不会考虑第二个值；第一个值为false时，会检查第二个值。对于非布尔值进行“或”运算，会先将其转换为布尔值，然后再运算，并返回原值（如果第一个值时true，则直接返回第一个值；如果第一个值为false，则返回第二个值）

1. 赋值运算符：=、+=、-=、\*=、/=、%=
2. 关系运算符：比较两个值之间的大小关系，关系成立返回true，关系不成立返回false。>、<、>=(大于或等于)、<=、==、！=

1、非数值的情况（>、<、>=、<=）：

进行比较时，会将其转换为数字再比较；任何值和NaN作任何比较都是false；

如果一侧时字符串数字，则会将其转换为数字进行比较；

如果符号两侧都是字符串型数值时，不会将其转换为数字比较，而是将其unicode编码进行比较，比较字符编码时是一位一位进行比较，如果两位一样，则比较下一位，所以可以借用它来对英文进行排序；

注意，在比较两个字符串型的数字时，可能会得到不可预期的结果，必须要转型。例如 “12312313”< +“5”

2、==相等运算符：如果值的类型不同，则会自动进行类型转换，转换为相同的类型，再作运算：

null==0；//false；undefined衍生自null，所以这两个值作相等判断时会返回true；

NaN不和任何值相等，包括它本身，因此不能判断一个值是否是NaN，需要用isNaN（）函数来判断；

3、===全等：不会做类型转换，若两个类型不同则返回false

！==不全等：不会做类型转换，若两个类型不同则返回true

1. 条件运算符（三元运算符）：条件表达式？语句1：语句2

如果条件表达式的求值结果是一个非布尔值，会将其转换为布尔值然后再运算

1. 运算符的优先级
2. JS中prompt（）弹出一个提示框，带有一个文本框，用户可以在文本框中输入一段内容，该函数需要一个字符串作为参数，该字符串将会作为提示框的提示文字；

用户输入的内容将会作为函数的返回值返回，可以定义一个变量来接收用户的输入；

prompt（）的返回值是string类型。

1. switch…case..语句，在执行时会依此将case后的表达式的值和switch后的条件进行全等比较。如果比较结果为true，则从当前case处开始执行代码，当前case后的所有代码都会执行，如果比较结果为false，则继续向下比较；

可以在case后面使用break可以确保只执行当前case语句，不执行后面的代码；

如果所有结果都为false，则执行defaul后的语句。

1. 向页面中输出：document.write();
2. break和continue

break关键字可以用来退出switch或循环语句，不能在if语句中使用break和continue；

break关键字会立即终止**离它最近**的循环语句；

可以为循环语句创建一个lable，来表示当前的循环。（eg：lable：循环语句），使用break语句时，可以在break后跟着一个lable，这样break将会结束指定的循环而不是最近的一个。

continue关键字可以用来跳过当次循环，同样continue也默认只会对离它最近的循环起作用；

也可以使用lable来跳出指定循环。

1. 测试程序的性能。

在程序执行前，开启计时器（consolt.time(“计时器名字”)）

终止计时器（console.timeEnd(“计时器名字”)）

1. 对象Object
2. 对象的基本概念及操作
3. 对象属于一种复合的数据类型，可以保存多个不同数据类型的属性
4. 对象的分类：
5. 内建对象：由ES标准中定义的对象，在任何的ES实现中都可以使用

Math、String、Number等

1. 宿主对象：由JS的运行环境提供的对象，目前来讲主要指由浏览器提供的对象

BOM、DOM

1. 自定义对象：由开发人员自己创建的对象
2. 创建对象：
3. 使用new关键字调用的函数，是构造函数constructor ，构造函数是专门用来创建对象的函数，使用typeof检查一个对象时，会返回object。

var xxx = new Object（）；

用这种方式创建对象，只能在创建完对象之后再添加属性

1. **使用对象字面量来创建一个对象**： var obj={ }；

使用对象字面量可以在创建对象时直接指定对象中的属性：var obj = {name:””,age:””};

对象字面量的属性名可以加引号也可以不加，建议不加，但如果要使用一些特殊的字符，则必须加引号

1. 使用工厂方法创建对象（用的不多）
2. 通过该方法可以大批量的创建对象
3. function 函数名（形参）{

//创建一个新的对象

var obj=new Object（）；

//向对象添加属性

obj.属性名=“属性值”（形参）；

…

//将新的对象返回

return obj；

}

1. 使用工厂方法创建的对象，使用的构造函数都是Object，所以创建的对象都是Object这个类型，就导致我们无法区分多种不同类型的对象
2. 使用构造函数来创建对象
3. 构造函数就是一个普通的函数，创建方式和普通函数没有区别。不同的是构造函数习惯上首字母大写
4. 构造函数和普通函数的区别就是调用方式的不同，普通函数就是直接调用，而构造函数需要使用new关键字来调用
5. function 函数名（形参）{

this.属性名=形参；

…

}

var 变量名 = new 函数（实参）；

1. 构造函数的执行流程：
2. 当调用构造函数时，立即创建一个新的对象
3. 将新建的对象设置为函数中的this，在构造函数中可以使用this来引用新建的对象
4. 逐行执行函数中的代码
5. 将新建的对象作为返回值返回
6. 使用同一个构造函数创建的对象，我们称为一类对象，也将一个构造函数称为一个类。我们将通过一个构造函数创建的对象称为是该类的实例
7. 使用instanceof可以检查一个对象是否是一个类的实例。

对象 instancef 构造函数

如果是则返回true，如果不是则返回false

1. 所有的对象都是Object的后代，所以任何对象和Object做instanceof检查时都会返回true
2. 向对象中添加属性：
3. 对象.属性名=属性值；
4. 对象的属性名不强求遵守标识符的规范，但使用时尽量按照标识符的规范。如果要使用特殊的属性名，不能采用.的方式操作，需要使用另一个方式：对象[“属性名”]=属性值；读取时也要采用这种方式。使用[ ]这种形式操作属性，更加灵活，在[ ]中给可以直接传递一个变量，这样变量值是多少就会读取那个变量
5. JS对象的属性值，可以是任意的数据类型。
6. 读取对象的属性：对象.属性名；如果读取对象中没有的属性，不会报错而是会返回undefined
7. 删除对象的属性：delete 对象.属性名；
8. 检查一个对象中是否含有指定的属性：in运算符： “

属性名” in 对象 如果有则返回true，没有则返回false

如果对象中没有，但是原型中有，也会返回true

1. 基本数据类型和引用数据类型
2. 基本数据类型：String、Number、Boolean、Null、Undefined

引用数据类型：Object

1. JS中的变量都存在栈内存中，
2. 基本数据类型的值直接在栈内存中存储，值与值之间是独立存在的，修改一个变量不会影响其它的变量；
3. 引用数据类型（对象）保存在对内存中，每创建一个对象，都会在堆内存中开辟一个新的空间，而变量保存的是对象的内存地址（对象的引用），如果两个变量保存的是同一个对象的引用，当通过一个变量修改属性是，另一个变量也会受到影响
4. 枚举对象中的属性
5. for（var 变量 in 对象）{ }
6. for…in语句，对象中有几个属性，循环体就会执行几次，每次执行时，会将对象中的一个属性的名字赋值给变量
7. 检查对象自身中是否含有属性：hasOwnProperty（）
8. 函数
9. 函数也是一个对象，函数中可以封装一些功能（代码），在需要时可以执行这些功能。在使用typeof检查一个函数对象时，会返回function
10. 创建函数对象：
11. var fun = new Function（）；可以将要封装的代码以字符串的形式传递给构造函数封装到函数中的代码不会立即执行，会在函数调用时执行。

调用函数的语法： fun（）；

但在开发中很少使用构造函数来创建一个函数对象。

1. **使用函数声明来创建一个函数**。 function 函数名（[形参1，形参2…形参n]）{ 语句… }
2. 使用**函数表达式来创建**一个函数。var 函数名 = function（[形参1，形参2…形参n]）{ 语句… }；
3. 函数的参数
4. 声明了形参，就相当于在函数内部声明了对应的变量，但是并不赋值。
5. 函数的实参可以是任意的数据类型。
6. 调用函数时，解析器不会检查实参的类型。所以要注意是否有可能会接收到非法的参数，如果有可能则需要对参数进行类型的检查
7. 调用函数时，解析器也不会检查实参的数量。多余的实参不会被赋值，如果实参的数量少于形参的数量，则没有对应实参的形参将是undefined
8. 函数的返回值
9. 可以使用return来设置函数的返回值
10. 返回值可以是任意的数据类型，也可以是一个对象，也可以是一个函数
11. 立即执行函数
12. 函数定义完，立即被调用，这种函数叫做立即执行函数
13. 立即执行函数值执行一次
14. （function（形参）{ 语句 …}）（实参）
15. 方法
16. 一个函数作为一个对象的属性，该函数叫做该对象的方法。
17. 对象.方法名（）
18. 作用域
19. 变量的声明提前
20. 使用var关键字声明的变量，会在所有代码执行之前被声明（但不会被赋值），但是如果声明变量时不使用var关键字，则变量不会被声明提前
21. 不使用var关键字声明的对象，在声明之前使用则会报错not defined
22. 函数的声明提前
23. 使用函数声明形式创建的函数[function 函数（）{ }]会在所有的代码执行之前就被创建，所以可以在函数声明前调用函数
24. 使用函数表达式创建的函数[var 变量 = function（）{ }]不会被声明提前，所有不能在声明之前创建
25. 全局作用域
26. 直接编写在script标签中的JS代码都在全局作用域
27. 全局作用域在页面打开时创建，在页面关闭时销毁
28. 在全局作用域中有一个全局对象window，它代表的是一个浏览器的窗口，它由浏览器创建，我们可以直接使用
29. 在全局作用域中，我们创建的变量都会作为window对象的属性保存，创建的函数都会作为window的方法保存
30. 全局作用域中的变量都是全局变量，在页面的任意部分都可以访问到
31. 函数（局部）作用域
32. 调用函数时创建函数作用域，函数执行完毕后，函数作用域销毁
33. 每调用一次函数就会创建一个新的函数作用域，他们之间时互相独立的
34. 在函数作用域中，可以访问到全局作用域的变量，在全局作用域中无法访问到函数作用域的变量
35. 当在函数作用域中操作一个变量时，它会现在自身作用域中寻找，如果有就直接使用，如果没有则向上级作用域中寻找，直到找到全局作用域，如果全局作用域中依然没有，则会报错ReferenceError
36. 在函数中要访问全局变量，可以使用window对象
37. 在函数作用域中也有声明提前的特性，使用var关键字声明的变量，会在函数中所有的代码执行之前执行，但不赋值
38. 函数声明也会在函数中所有的代码执行之前执行
39. 在函数中不使用var关键字声明的变量，都会变成全局变量
40. 定义形参就相当于在函数作用域中声明了变量
41. this
42. 解析器在调用函数时每次都会向函数内部传递一个隐含的参数，这个隐含的参数就是this，thie指向一个对象，这个对象我们成为函数执行的上下文对象，根据函数的调用方式不同，this会指向不同的对象
43. 以函数形式调用时，this永远都是window
44. 以方法形式调用时，this就是调用的方法的那个对象
45. 以构造函数形式调用时，this就是新创建的那个对象
46. 使用call（）和apply（）调用时，this就是指定的那个对象
47. 在事件的响应函数中，响应函数是给谁**绑定**的（不是触发），this就是谁
48. 原型对象prototype
49. 我们所创建的每一个函数，解析器都会向函数中添加一个属性prototype，这个属性对应着一个对象，这个对象就是原型对象
50. 如果函数作为普通函数调用prototype没有任何作用，但如果函数作为构造函数调用时，它做创建的对象中都会有一个隐含的属性只想构造函数的原型对象，可以通过\_\_proto\_\_来访问该属性
51. 原型对象就相当于一个公共的区域，所有同一个类的实例都可以访问到这个原型对象，我们可以将对象中共有的内容，统一设置到原型对象中
52. 当我们访问对象的一个属性或方法时，会首先在对象自身中寻找，如果有则直接使用，如果没有就去原型对象中寻找，如果找到就直接使用
53. 以后我们创建构造函数时，可以将这些对象共有的属性和方法，统一添加到构造函数的原型对象中，这样不用分别为每一个对象添加，也不会影响到全局作用域，就可以使每个对象都具有这些属性和方法。
54. 如果不想使用原型对象中的方法，可以直接重新给实例写一个方法
55. 原型对象也是对象，所以它也有原型，当我们使用一个对象的属性和方法时，会在自身寻找，如果有则使用，如果没有则去原型对象中寻找，如果有则使用，如果没有则去原型的原型中寻找，直到找到Object原型。Object对象的原型没有原型，如果在Object中依然没有找到，则返回undefined
56. 当我们直接在页面中打印一个对象时，实际上输出的是对象的toString（）方法的返回值，如果希望在输出对象时不输出[Object Object]，可以为对象添加一个toString（）方法
57. 垃圾回收机制（GC）
58. 程序运行中会产生垃圾，垃圾积攒过多后，会导致程序运行的速度过慢，所以我们需要一个垃圾回收的机制，来处理程序运行过程中产生的垃圾
59. 当一个对象没有任何的变量或属性对它进行引用，此时我们将永远无法操作该对象，此时这种对象就是垃圾，这种对象过多会占用大量的内存空间，导致程序变慢，必须进行清理
60. 在JS中，拥有自动的垃圾回收机制，会自动将垃圾对象从内存中销毁，我们不需要也不能进行垃圾回收的操作
61. 我们需要做的是将不再使用的对象设置为null

eg： obj=null；

1. 数组Array
2. 与普通对象使用字符串作为属性名不同，数组对象是通过数字作为索引（从0开始）操作元素
3. 数组的存储性能比普通对象好，在开发中经常使用数组来存储一些数据
4. 创建数组对象：
5. 构造函数：var arr = new Array（）；（一般不用）

使用构造函数创建数组时，可以同时添加元素，将要添加的元素作为构造函数的参数传递，元素之间用，隔开；

当只传入一个整数值时，构造函数表示创建长度为该值的数组

1. 使用字面量创建：var arr = [ ]；

使用字面量创建数组时，可以在创建时就指定数组中的元素；

当只传入一个整数值时，即创建只有一个值的数组

1. 数组中的元素可以是任意的数据类型
2. 判断一个对象是不是数组
3. 可以调用Array的isArray（）方法。Array.isArray(需要判断的对象)
4. 需要判断的对象 instanceof Array
5. 操作：
6. 向数组中添加元素：数组[索引]=值；
7. 读取数组中的元素：数组[索引]，如果读取不存在的索引，不会报错，会返回undefined
8. 获取数组的长度（元素的个数）：

对于连续的数组，可以使用 ：数组.length

对于非连续的数组，使用length会获取到最大的索引+1，尽量不要创建非连续数组

1. 修改数组的长度（元素的个数）：也可以使用 length如果修改的length大于原长度，则多处的不分会空出来；如过小于原长度，则多出的元素会被删除
2. 数组的方法
3. push（）
4. 向数组的末尾添加一个或多个元素，并返回数组新的长度
5. 将要添加的元素作为方法的参数传递
6. pop（）
7. 删除数组的最后一个元素，并返回被删除的元素
8. unshift（）
9. 向数组的开头添加一个或多个元素，并返回数组的新的长度
10. 向前面插入元素以后，其他元素的索引会一次调整
11. shift（）
12. 删除数组的第一个元素，并将被删除的元素作为返回值返回
13. slice（start，end）
14. 从某个已有的数组提取指定的元素。该方法不会改变原数组，而是将截取到的元素封装到一个新的数组中返回
15. 需要两个参数：截取开始的位置的索引和截取结束位置的索引 [开始，结束）

包括开始位置的元素，不包括结束位置的元素

1. 可以省略第二个参数，此时会截取从开始索引往后的所有元素
2. 索引可以传递一个负值，如果传递负值，则从后往前计算。-1表示倒数第一个；-2表示倒数第二个
3. splice（start，num，xx，…）
4. 删除数组中的指定元素并向数组添加新元素。该方法会影响到原数组，会将指定元素从原数组中删除，并将删除的元素作为返回值返回
5. 参数：start表示删除开始的位置的索引；num表示删除的数量；第三个及以后可以传递一些新的元素，这些元素将会自动插入到开始位置索引的前面
6. concat（其它数组/元素）
7. 连接两个或多个数组，并将新的数组返回
8. 该方法不会对原数组产生影响
9. join（）
10. 该方法可以将数组转换成一个字符串，不会影响原数组，将转换后的字符串作为结果返回
11. 在join（）中可以指定一个字符串作为参数，这个字符串将会作为数组中元素的连接符，如果不指定连接父，则默认使用“，”作为连接符
12. reverse（）
13. 该方法用来反转数组，该方法会直接修改原数组
14. sort（）
15. 该方法可以用来对数组中的元素进行排序，会影响原数组
16. 默认会按照Unicode编码进行排序
17. 即使对于纯数字的数组，使用sort（）排序时，也会按照Unicode编码来排序，所以对数字进行排序时，可能会得到错误的结果。
18. 因此我们可以自己来制定排序的规则，我们可以在sort（）添加一个回调函数来指定排序规则。

回调函数中需要定义两个形参，浏览器将会分别使用数组中的元素作为实参去调用回调函数。使用哪个元素调用不确定，但是肯定是在数组中，a一定在b前面。

浏览器会根据回调函数的返回值来决定元素的顺序，如果返回一个大于0的值，则元素会交换位置；如果返回一个小于0的值，则位置不变；如果返回一个0，则认为两个元素相等，也不交换位置。

arr.sort(function(a,b){

return a-b; //升序

return b-a;/ /降序

});

1. 数组的遍历
2. for （var i=0; i<arr.length; i++）{语句…}
3. forEach()方法： 数组.forEach(function(a,b,c){ });
4. 该方法只支持IE8以上的浏览器
5. forEach（）方法需要一个函数作为函数，像这种函数由我们创建但是不由我们调用的，我们称为回调函数
6. 数组中有几个元素，函数就会执行几次，每次执行时，浏览器会将遍历到的元素以实参的形式传递进来，我们可以来定义形参，来读取这些内容
7. 浏览器会在回调函数中传递三个参数

a参数是当前正在遍历的元素

b参数是当前正在遍历的元素的索引

c参数是正在遍历的数组

1. 函数对象（function）的方法：call（）和apply（）
2. 这两个方法都是函数对象的方法，需要通过函数对象来调用
3. 当对函数调用call（）和apply（）都会调用函数执行
4. 不用call（）和apply（）调用函数，而是直接调用函数对象时，this指向的是window
5. 在调用call（）和apply（）可以将一个对象指定为第一个参数，此时这个对象将会成为函数执行时的this。fun.call(obj); this🡪obj
6. 区别：

call（）方法可以将实参在对象之后依次传递。fun.call(obj,xx,xx);

apply（）方法需要将实参封装到一个数组中统一传递。fun.apply(obj,[xx,xx]);

1. arguments（实参）
2. 在调用函数时，浏览器每次都会传递两个隐含的参数：函数的上下文对象this和封装实参对象的arguments
3. arguments是一个类数组对象，它也可以通过索引来操作数据arguments[n]，也可以获取长度arguments.length。在调用函数时，我们所传递的实参都会在arguments中保存。
4. 我们即使不定义形参，也可以通过arguments来使用参数，只不过比较麻烦

arguments[0]表示第一个实参

1. arguments有一个callee的属性，这个属性对应一个函数对象，就是当前正在指向的函数的对象
2. Date（是个构造函数）
3. 在JS中，使用Date对象来表示一个时间
4. 创建Date对象：
5. var xx =new Date（）；如果直接使用构造函数创建一个Date对象，则会封装为当前代码执行的时间
6. 创建一个指定的时间对象，需要在函数中传递一个表示时间的字符串作为参数。

日期格式： 月份/日/年 时：分：秒

var xx = new Date(“12/03/2016 11:10:30”)

1. 属性
2. getDate（）获取当前日期对象是几日
3. getDay（）返回当前日期对象是周几。会返回0-6的值，0表示周日
4. getMonth（）获取当前日期对象的月份。会返回0-11的值，0表示1月
5. getFullYear（）获取当前日期对象的年份。
6. getTime（）获取当前日期对象的时间戳。
7. 时间戳，指的是从格林威治标准时间的1970年1月1日0时0分0秒，到当前时间所花的毫秒数。计算机底层在保存时间时使用的都是时间戳
8. Date.now（）获取当前的时间戳
9. 测试代码的执行的性能
10. 在程序执行前，开启计时器（consolt.time(“计时器名字”)）,在程序结束时，开启终止计时器（console.timeEnd(“计时器名字”)）；
11. 利用时间戳来测试：

在程序执行前，获取时间戳（var start = Date,now()），在程序结束时，再获取一个时间戳（var end = Date,now()），再用end-start得到的值则是程序的执行时间

1. Math对象
2. 和其它的对象不同，它不是一个构造函数，它属于一个工具类不用创建对象，它里面封装了数学运算相关的属性和方法。例如：Math.PI表示圆周率
3. 方法：Math.xx()
4. abs（x）用来计算一个数的绝对值
5. cell（x）可以对一个数进行向上取整，小数位只要有值就自动进1
6. floor（x）可以对一个数进行向下取整，小数部分会被舍掉
7. round（x）可以对一个数进行四舍五入取整
8. random（）可以用来生成一个0-1之间的随机数

生成一个0-x的随机数：Math.round（Math.random()\*x）;

生成一个x-y的随机数：Math.round（Math.random()\*（y-x）+x）;

1. max（a,b,c,…）可以获取多个数中的最大值

min（a,b,c,…）可以获取多个数中的最小值

1. pow（x，y）用来计算x的y次幂
2. sqrt（x）用来求一个数的平方根
3. 包装类
4. JS中提供了三个包装类，通过这三个包装类可以将基本数据类型的数据转换为对象
5. String（）可以将基本数据类型字符串转换成String对象
6. 创建一个String类型的对象

var str = new String（“xxx”）；

1. 添加属性

num.xxx = “xx”;

Number（）可以将基本数据类型数字转换成Number对象

1. 创建一个Number类型的对象

var num = new Number（x）；

Boolean（）可以将基本数据类型布尔值转换成Boolean对象

1. 创建一个Boolean类型的对象

var bool = new Boolean（true/false）；

1. 但是注意，我们在实际应用中不会使用基本数据类型的对象，如果使用基本数据类型的对象，在做一些比较时可能会带来一些不可预料的结果
2. 当我们对一些基本数据类型的值去调用属性和方法时，浏览器会**临时**使用包装类将其转换为对象，然后再调用对象的属性和方法，调用完以后，再将其转换为基本数据类型
3. String对象的方法（操作字符串），基本都不会影响到原字符串
4. length属性 获取字符串的长度
5. charAt（n）可以返回字符串中指定位置n的字符，索引从0开始
6. charCodeAt（n）可以返回字符串中指定位置n字符的Unicode编码
7. fromCharCode（）可以根据字符编码获取字符，该方法通过构造函数调用
8. concat（）可以连接两个或多个字符串，作用和“+”一样
9. indexOf（“x”，n）该方法可以检索一个字符串中是否含有指定内容，如果有则返回其第一次出现的索引，如果没有则返回-1。可以指定第二个参数指定开始查找的位置
10. lastIndexOf（）该方法的用法和indexOf（）一样，不同的是该方法从后往前找。也可以指定第二个参数，指定查找开始的位置
11. slice（）可以从字符串中截取指定的内容，不会影响原字符串，而是将截取到的内容返回。

第一个参数为开始位置的索引（包括开始位置），第二个参数为结束位置的索引（不包括结束位置），可省略第二个参数，则会截取后面的所有。也可以传递一个负数，则会从后边开始计算。

1. substring（）可以用来截取一个字符串，和slice（）类似。参数也类似，不同的是该方法不能接收负值作为参数，如果传递了一个负值，则默认为0，而且还会自动调整参数的位置，如果第二个参数小于第一个，则自动交换
2. split（“x”）可以将一个字符串拆分为一个数组。需要一个字符串作为参数，将会根据该字符串去拆分数组。如果传递一个空串作为参数，则会将每个字符都拆分为一个元素。

方法中可以传递一个正则表达式作为参数，根据正则表达式拆分字符串。这个方法即使不指定全局匹配，也会全部拆分

1. toLowerCase（）将字符串转换成大写并返回

toUpperCase（）将字符串转换为小写并返回

1. search（）可以搜索字符串中是否有指定内容，如果搜索到指定内容，则返回第一次出现的索引，如果没有则返回-1。

该方法可以接受一个正则表达式作为参数，然后会根据正则表达式去检索字符串。只会查找第一个，即使设置全局匹配也没用。

1. march（）可以根据正则表达式从一个字符串中将符合条件的内容提取出来。

默认情况下match（）只会找到第一个符合要求的内容，找到以后就停止检索，我们可以设置正则表达式为全局匹配模式，这样就会匹配到所有的内容，可以为一个正则表达式设置多个匹配模式，且顺序无所谓。

match（）会将匹配到的内容封装到一个数组中返回，即使只查询到一个结果。

1. replace（）可以将字符串中指定的内容替换为新的内容。需要两个参数：第一个为被替换的内容，可以接受一个正则表达式作为参数；第二个参数为新的内容。

默认值会替换第一个，我们可以设置正则表达式为全局匹配模式，这样就会匹配到所有的内容

1. 正则表达式
2. 用于定义一些字符串的规则，计算机可以根据正则表达式来检查一个字符串是否符合规则，或者将字符串中符合规则的内容提取出来
3. 创建正则表达式的对象
4. 构造函数创建：var reg = new RegExp（“正则表达式”，“匹配模式”）

匹配模式有两个可选值：I 忽略大小写；g 全局匹配模式

1. 使用字面量创建：var 变量 = /正则表达式/匹配模式
2. 正则表达式的方法：test（）使用这个方法可以用来检查一个字符串是否符合正则表达式的规则，如果符合则返回true，否则返回false
3. 正则表达式：
4. 单写字符，表示查找是否含有该字符
5. 使用 | 表示或者的意思；[ ]里的内容个表示或的关系

[a-z]表示任意的小写字母 [A-z]表示任意字母 [0-9]表示任意字母

1. [^ ] 表示除了
2. 量词：通过量词可以设置一个内容出现的次数，只对前边的一个内容起作用

1、{n}表示正好出现n次

2、{m，n}表示m-n次，

3、{m，}表示至少m次

4、+ 表示至少一个，相当于{1，}

5、\* 表示0个或多个，相当于{0，}

6、？ 表示0个或1个，相当于{0，1}

1. ^ 表示开头

$ 表示结尾

如果在一个正则表达式中同时使用^$则要求字符串必须完全符合正则表达式

1. . 表示任意字符

如果想检查是否有.，在正则表达式中使用\做转义字符，\.来表示. [\\表示\](file://表示/)

注意，使用构造函数时，由于它的参数是一个字符串，而\是字符串中转义字符，如果要使用\则需要使用[\\来代替](\\\\来代替)

1. \w 任意字母、数字、\_ \W 除了字母、数组、\_

\d 任意数字 \D 除了数字

\s 空格 \S 除了空格

\b 单词边界 \B除了单词边界

1. 去除字符串前后的空格. str=str.replace(/^\s\*|\s\*$/g,””);
2. DOM（宿主对象）
3. document object model 文档对象模型，JS中通过DOM来对HTML进行操作。
4. 文档的加载：浏览器在还在一个页面时，时按照自上向下的顺序加载的。读取到一行就执行一行，如果将script标签写到页面的上面，在代码执行时，页面还没有加载，DOM对象也没有加载，导致获取不到对象。

如果想把script写在页面里面，则将事件代码都写在onload函数里面。onload事件对应的响应函数会在整个页面加载完成之后才触发，window.onload=function（）{ }；这样可以确保我们的代码执行时所有的DOM对象已经加载完毕了。

1. DOM查询：
2. 获取元素节点，通过document对象调用：

如果需要读取元素节点属性，直接使用 元素.属性名 但注意class属性不能采用这种方式。读取class属性时需要使用 元素.className

1. getElementById()
2. getElementsByTagName（）可以根据标签名来获取一组元素节点对象，这个方法会给我们返回一个类数组对象，所有查询到的元素都会封装到对象中，即使查询到的元素只有一个，也会封装到数组中返回。
3. getElementsByName（）
4. 获取元素节点的子节点，通过具体的元素节点调用：
5. getElementsByTagName()方法，返回当前节点的指定标签名后代节点
6. childNodes属性，表示当前节点的所有子节点。会获取包括文本节点在内的所有节点，根据DOM标签间的空白也会当成文本节点。注意：在IE8及以下的浏览器中，不会将空白文本当成子节点。

childNodes属性，获取当前节点的所有子元素。

1. firstChild属性，表示当前节点第一个子节点（包括空白文本）
2. lastChild属性，表示当前节点最后一个子节点（包括空白文本）
3. 获取父节点和兄弟节点，通过具体节点调用：
4. parentNode属性，获取当前节点的父节点
5. previousSibling属性，获取当前节点的前一个兄弟节点（包括空白文本）

previousElementSibling属性，获取前一个兄弟元素（不包括空白文本），IE8及以下不兼容

1. nextSibling属性，获取当前节点的后一个兄弟节点
2. 其他查询方法
3. document.body保存的是body的引用
4. document.documentElementb保存的是html根标签
5. document.all/getElementByTagName（“\*”）保存的是所有元素
6. document.getElementByClassName()根据class属性值获取一组元素节点对象，但是该方法不支持IE8及以下的浏览器，虽然IE8种没有该方法，可以用下面的方法代替
7. document.querySelector()，需要一个选择器的字符串作为参数，可以根据一个CSS选择器来查询一个元素节点对象。

eg：document.querySelector(“.box1 div”)

使用该方法总会返回唯一的一个元素，如果满足条件的有多个，只会返回第一个

1. document.querySelectorAll()该方法和上一个用法类似，不同的是，它会将符合条件的元素封装到一个数组中返回，即使符合条件的元素只有一个，也返回一个数组。
2. DOM增删改
3. document.createElement：可以用来创建元素节点对象，它需要一个标签名作为参数，将会根据该标签名创建元素节点对象，并将创建好的对象作为返回值返回
4. document.createTextNode：可以用来创建文本节点对象，它需要一个文本内容作为参数，将会根据该文本内容创建文本节点对象，并将创建好的对象作为返回值返回
5. appendChild：向一个父节点中添加一个新的子节点。

用法：父节点.appendChild(子节点)

1. insertBefore：在指定的子节点前面插入新的子节点。

用法：父节点.insertBefore（新节点，旧节点）

1. replaceChild：可以使用指定的子节点替换已有的子节点。

用法：父节点.replaceChlid(新节点，旧节点)

1. removeChild：删除一个子节点。

用法：父节点.removeChlid（子节点)

**更常用的语法：**子节点.parentNode.removeChild(子节点)

1. 使用innerHtml也可以完成DOM的增删改的相关操作，一般会两种方式结合使用。
2. DOM操作CSS
3. 操作内联样式
4. 通过JS修改元素的样式：语法：元素.style.样式名=样式值
5. 注意：如果CSS的样式名中有-，这种名称在JS中是不合法的，比如background-color，需要将这种样式名改为驼峰命名法
6. 通过style属性设置的样式都是内联样式，而内联样式具有较高的优先级，所以通过JS修改的样式往往会立即显示。但是如果在样式中写了！important，则此时样式会有最高的优先级，即使荣国JS也不能覆盖该样式，此时将会导致JS修改样式失效，所以尽量不要为样式添加！important
7. 读取元素的样式：语法：元素.style.样式名
8. 通过style属性设置和读取的都是内联样式，无法读取样式表中的样式
9. 获取元素当前显示的样式
10. 语法：元素.currentStyle.样式名

用来读取当前元素正在显示的样式，如果当前元素没有设置该样式，则获取它的默认值。例如：没有给div设置width，则获取到的是auto

currentStyle仅IE浏览器支持，其他浏览器都不支持

1. 在其他浏览器中，可以使用getComputedStyle（）这个方法来获取元素的当前样式，这个方法是window的方法，可以直接使用。

需要两个参数： 第一个为要获取样式的元素

第二个可以传递一个伪元素，一般都穿null

该方法会返回一个对象，对象中封装了当前元素对应的样式。如果要获取样式要通过 对象.样式名 来获取。如果获取的样式没有设置，则会获取到真实的值，而不是默认值。例如：没有给div设置width，则获取到的是一个具体长度

但是，该方法不支持IE8及以下的浏览器

1. 通过cuttentStyle和getComputedStyle（）读取到的样式都是只读的，不能修改
2. 定义一个函数，用来获取指定元素的当前样式。

参数： obj 要获取样式的元素。 name 要获取的样式名

function getStyle（obj，name）{

if(window.getConputedStyle){

//正常浏览器的方式，具有getComputedStyle方法

return getComputedStyle（obj，null）[name];

}else{

//IE8的方式，没有getComputedStyle方法

return obj.currentStyle[name];

}

}

属性没找到返回undefined，变量没找到则会报错，为了防止报错，使用window.getComputedStyle，而不是直接使用getComputedStyle

1. 其他样式操作的属性
2. clientWidth/clientHeight这两个属性可以获取元素的可见宽度和高度，这些属性都是不带px的，可以直接进行计算。

会获取元素宽度和高度，包括内容区和内边距。

这些属性都是只读的不能修改。

1. offsetWidth/offsetHeight获取元素的整个高度和宽度，包括内容区、内边距和边框
2. offsetParent可以用来获取当前元素的定位父元素。或获取到离当前元素最近的开启了定位的祖先元素，如果所有的祖先元素都没有开启定位，则返回body
3. offsetLeft/offsetTop获取当前元素对于其定位父元素的水平/垂直偏移量

Ps：偏移量只对开启了定位的元素起作用

1. scrollHeight/scrollWidth获取元素整个滚动区域的高度和宽度

scrollLeft/scrollTop获取水平和垂直滚动条滚动的距离

当满足scrollHeight-scrollTop==clientHeight时，说明垂直滚动条到底了

当满足scrollWidth-scrollLeft==clientWidth时，说明水平滚动条到头了

Ps：onscroll事件，该事件会在滚动条滚动时触发

Ps：当整个页面滚动时，chrome认为浏览器的滚动条是body的，可以通过body.scrollTop获取，火狐等浏览器认为浏览器的滚动条是html的。则需要进行判断：

document.body.scrollTop || document.documentElement.scrollTop

document.body.scrollLeft || document.documentElement.scrollLeft

1. 事件
2. 事件，就是用户和浏览器之间的交互行为。JS与Html之间的交互是通过事件实现的。
3. 事件对象：
4. 当事件的响应函数被触发时，浏览器每次都会将一个事件对象（event）作为实参传递进响应函数，在事件对象中封装了当前事件相关的一切信息。

比如：

·鼠标的坐标(event.clientX/event.clientY）：

clientX和clientY获取鼠标在当前可见窗口的坐标。而div的偏移量是相对于整个页面的

pageX和pageY可以获取鼠标相对于当前页面的坐标。但是这两个属性在IE8中不支持，所以如果要兼容IE8则不要使用

·键盘哪个按键被按下、

·鼠标滚轮滚动的方向等。

1. IE8中，响应函数被触发时，浏览器不会传递事件对象。在IE8及以下的浏览器中，是将事件对象作为window对象的属性保存的。
2. 所以，需要判断是否支持event，解决事件对象的兼容性问题。

if（event）{event=window.enevt；}或者写成event=event || window.event；

1. event中的target属性表示的是触发事件的对象
2. 处理事件：
3. 我们可以在事件对应的属性中设置一些js代码，这样当事件被触发时，这些代码会执行。这种方法被称为结构和行为耦合，不方便维护，不推荐使用
4. 在js代码中编写代码。首先获取对象，再给对象对应事件绑定处理函数，这样当事件被触发时，其对应的函数将会被调用。这样的函数也是回调函数
5. 事件的冒泡（Bubble）：
6. 冒泡：指的就是事件的向**上**传导，当后代元素上的事件被触发时，同时祖先元素的相同事件也会被触发
7. 在开发中，大部分情况是有利的，如果不希望发生事件冒泡，可以通过事件对象来取消冒泡。 event.canBubble = true;
8. 事件的委派，将事件统一绑定到元素的共同的祖先元素，这样当后代元素上的事件触发时，会一直冒泡到祖先元素，从而通过祖先元素的响应函数来处理事件。

事件的委派是利用了冒泡，通过委派可以减少事件绑定的次数，提高程序的性能。

1. 事件的绑定
2. 对象.事件=函数

使用该形式绑定响应函数，它只能同时为一个元素的一个事件绑定一个响应函数，不能绑定多个，如果绑定了多个，则后边会覆盖掉前面的

1. 对象.addEventListener（“xxx”，function（）{ }，false）通过这个方法也可以为元素绑定响应函数
2. 参数：
3. 事件的字符串（不要on）
4. 回调函数，当事件触发时该函数会被调用
5. 是否在捕获阶段触发事件，需要一个布尔值，一般都传false
6. 使用该形式绑定响应函数，它能同时为一个元素的相同事件同时绑定多个响应函数，这样当事件被触发时，响应函数将会按照函数的绑定顺序执行
7. addEventListener()中的this，是绑定事件的对象
8. 使用该方法绑定响应函数，取消默认行为时，不能使用return false; 需要使用event.preventDefault（）取消
9. 上述方法不支持IE8及以下的浏览器，这些浏览器可以使用attachEvent(“onxxxx”,function(){ })来绑定
10. 参数：
11. 事件的字符串（要on）
12. 回调函数，当事件触发时该函数会被调用
13. 这个方法也可以同时为一个事件绑定多个处理函数，不同的是，它是后绑定先执行，执行顺序和addEventListener()相反
14. attachEvent()中的this，是window
15. 为了兼容所有浏览器，需要自己定一个函数bind，用来为指定元素绑定响应函数。
16. 需要统一两个方法的this
17. 参数：
18. obj 要绑定事件的对象
19. eventStr 事件的字符串（不要on）
20. callback 回调函数
21. function bind(obj , eventStr , callback){

if(obj.addEventListener){

//大部分浏览器兼容的方式

obj.addEventListener(eventStr , callback , false);

}else{

//IE8及以下

obj.attachEvent(“on”+eventStr , function(){

//在匿名函数中调用回调函数

callback.call(obj);

});

}

}

原因：匿名函数function（）是由浏览器调用的，而匿名函数中调用函数callback是我们自己调的，可以用call()来改变this，达到统一this的效果。

1. 事件的传播（不包括IE8及以下浏览器）
2. 三个阶段：
3. 捕获阶段：在捕获阶段时，从最外层的祖先元素向最目标元素进行事件的捕获，但是默认此事不会触发事件。
4. 目标阶段：事件捕获到目标元素，捕获结束，开始在目标元素上触发事件。
5. 冒泡阶段：事件从目标元素向他的祖先元素传递，依次触发祖先元素上的事件。
6. 如果希望在捕获阶段就触发事件，可以将addEventLinstner()中的第三个参数设置为true

一般情况下，我们不会希望在捕获阶段触发事件，所以这个参数一般都是false

1. 鼠标事件
2. 拖拽（操作）：
3. 当我们拖拽一个网页中的内容时，浏览器会默认去搜索引擎中搜索内容该，此时会导致拖拽功能的异常，这个是浏览器提供的默认行为，如果不希望发生这个行为，则可以通过return false来取消默认行为。
4. 但是，IE8不起作用。可以通过setCapture()，该方法可以捕获所有的鼠标点击事件设置到自己身上，当鼠标送开始，取消对事件的捕获通过releaseCapture().但是该方法只有IE支持，但是在火狐中调用不会报错，而如果使用chrome调用，会报错。所以需要先判断。
5. 滚轮事件
6. onmousewheel鼠标滚轮滚动的事件，会在滚轮滚动时触发，
7. 在火狐中需要使用DOMMouseScroll来绑定滚动事件，同时该事件需要通过addEventListener（）函数来绑定
8. event.wheelDelta可以获取鼠标滚轮滚动的方向，向上滚120向下滚-120。不看大小，只看正负
9. 在火狐中需要用event,detail来获取滚动的方向，向上滚-3下下滚是3。也不看大小，只看正负
10. 当滚轮滚动时，如果浏览器有滚动条，滚动条会随之滚动，这是浏览器的默认行为，如果不希望发生可以取消（return false；）
11. 在火狐中需要使用event.preventDefault（）取消。但IE8不支持，如果要兼容所有浏览器要进行判断
12. 键盘事件
13. 键盘事件一般绑定给可以获取到焦点的对象或document
14. onkeydown按键被按下。如果一直按着某个按键不松手，则事件会一直触发。当onkeydown连续触发时，第一次和第二次之间会间隔稍微长一点，其他的会非常快，是为了防止误操作的发生

onkeyup按键被松开。

1. 可以通过event.keyCode可以判断哪个按键被按下。除了keyCode，事件对象中，还提供了几个属性（altKey、ctrlKey、shiftKey），这三个属性用来判断alt、ctrl和shift是否被按下，如果被按下则返回true，否则返回false
2. 在文本框中输入内容，属于onkeydown的默认行为，如果在onkeydown中取消了默认行为（return false；），则输入的内容，不会出现在文本框中
3. 数字的keyCode是48-57。左上右下keyCode是37-40
4. BOM
5. 浏览器对象模型，BOM可以使我们通过JS来操作浏览器，在BOM中为我们提供了一组对象，用来完成对浏览器的操作
6. BOM对象：
7. Window：代表的是整个浏览器窗口，同时window也是网页中的全局对象
8. 定时器：

* setInterval（fuction，time）定时调用，可以将一个函数每隔一段时间执行一次。执行多次。该函数返回一个Number类型的数据，这个数字用来作为定时器的唯一标识。

参数：1、回调函数，该函数会每隔一段时间被调用一次

2、每次调用间隔的时间，单位是毫秒

在开启一个定时器之前需要将当前元素上的其他定时器关闭

* clearInterval（x）可以用来关闭一个定时器，方法中需要一个定时器的标识作为参数，这样将关闭标识对应的定时器

clearInterval可以接收任意参数，如果参数是一个有效的定时器的标识，则停止对应的定时器，如果参数不是一个有效的标识，则什么也不做

* setTimeout（fuction，time）延时调用，一个函数不马上执行，而是隔一段时间后再执行，且只执行一次。
* clearTimeout（）关闭一个延时调用

1. Navigator：代表的当前浏览器的信息，通过该对象可以来识别不同的浏览器
2. 由于历史原因，Navigator对象中的大部分属性已经不能帮助我们识别浏览器了，一般我们只会使用userAgent来判断浏览器的信息
3. navigator.userAgent是一个字符串，这个字符串中包含用来描述浏览器信息的内容，不同的浏览器会有不同的userAgent。如果要判断是何种浏览器，可以通过正则表达式来判断

例如 /chrome/i.test(navigator.userAgent)，判断字符串中是否有chrome字符，有则是chrome浏览器；firefox判断火狐浏览器；MSIE判断IE10及以下的IE浏览器，在IE11中已经将微软和IE相关的标识都已经去除了，所以已经基本不能通过userAgent来识别一个浏览器是否是IE了

1. 如果通过userAgent不能判断，还可以通过一些浏览器中特有的对象，来判断浏览器的信息。比如ActiveXObject，则通过“ActiveXObject” in window来判断，若返回true，则说明是IE浏览器，如果返回false则说明不是
2. Location：代表当前浏览器的地址栏信息，通过location可以获取地址栏信息，或者操作浏览器跳转页面
3. 如果直接打印location，则可以获取到地址栏的信息（当前页面的完整的路径）
4. 如果直接将location属性修改为一个完整的路径或相对的路径，则我们页面会自动跳转到该路径，并且会生成相应的历史记录
5. 方法

* assign（）用来跳转到其他页面，作用和直接修改location一样
* reload（）用来重新加载当前页面，作用和刷新按钮一样。如果在方法中传递一个true，作为参数，则会强制清空缓存刷新页面
* replace（）可以使用一个新的页面替换当前页面，调用完毕也会跳转页面。不会生成历史记录，不能使用回退按钮回退

1. History：代表浏览器的历史记录，可以通过该对象来操作浏览器的历史记录。由于隐私的原因，该对象不能获取到具体的历史记录，只能操作浏览器向前或向后翻页，而且该操作只在档次访问时有效
2. history.length属性可以获取到当次访问的连接的数量
3. 方法：

* back（）可以用来回退到上一个页面，作用和浏览器回退按钮一样；
* forward（）可以跳转下一个页面，作用和浏览器前进按钮一样；
* go（）可以用来跳转到指定的页面，它需要一个整数作为参数，1表示向前跳转一个页面，相当于forward（）；2表示向前跳转两个页面；-1:表示向后跳转一个页面‘-2表示向后跳转两个页面

1. Screen：代表用户的屏幕信息，通过该对象可以获取到用户的显示器的相关的信息

这些BOM对象在浏览器中都是作为window对象的属性保存的，可以通过window对象来使用，也可以直接使用

1. 类的操作
2. 通过style属性来修改元素的样式，每修改一个样式，浏览器就需要重新渲染一次页面，这样的执行的性能是比较差的，而且这种形式当我们要修改多个样式时，也不太方便。
3. 我们可以通过修改元素的class属性来间接的修改样式（obj.className=”xxx”），这样一来，我们只需要修改一次，即可同时修改多个样式，浏览器只需要重新渲染页面一次，性能比较好，并且这种方式，可以使行为和表现进一步分离
4. 二级菜单的实现：
5. 每一个菜单都是一个div，当div具有collapsed这个类（即class=“collapsed”）时，div就是折叠的状态，没有这个类时，就是展开的状态。collapsed中只有{height:xxpx;}
6. JSON
7. JS中的对象只有JS自己认识，其他的语言都不认识
8. JSON是一个特殊格式的字符串，这个字符串可以被任意的语言所识别，并且可以转换为任意语言中的对象，JSON在开发中主要用来数据的交互。
9. JSON（JavaScript Object Notation JS对象表示法）JSON和JS对象的格式一样，只不过JSON字符串中的属性名必须加双引号，其他的和JS语法一致。
10. JSON分类：
11. 对象{ } 例如：var obj =’{“姓名”:“xxx”，“年龄”：xx }’；
12. 数组[ ] 例如：var arr =’[1,2,3,“hello”]’；
13. JSON中允许的值：

字符串、数值、布尔值、null、对象、数组

1. JSON字符串转换为JS的对象：
2. 在JS中个，为我们提供了一个工具类，就叫JSON，这个对象帮助我们将一个JSON转换为JS对象，也可以将一个JS对象转换为JSON
3. JSON.parse（json）可以将JSON字符串转换为js对象，它需要一个JSON字符串作为参数，会将该字符串转换为JS对象并返回
4. JSON.stringify（js）可以将JS对象转换为JSON字符串，需要一个JS对象作为参数，会返回一个JSON字符串
5. JSON对象在IE7及以下的浏览器中不支持，解决方案：
6. 使用eval（）函数（不支持使用）：

eval（）这个函数可以用来执行一段字符串形式的JS代码，并将执行的结果返回。

如果使用eval（）执行的字符串中含有{ }，它会将{ }当成是代码块。

如果不希望将其当成代码块间隙，则需要在字符串前后加一个括号。

这个函数的功能很强大，但是在开发中尽量不要使用，首先它的执行性能比较差，其次它害具有安全隐患

1. 通过引入一个外部的JS文件来处理：（推荐）

<script type=”text/jacascript”rec=”json2.js”></script>