什么是JavaScript？

JavaScript是一种具有面向对象、解释型的程序设计语言。它是基于对象和事件驱动且相对安全的客户端使用的脚本语言。（事件驱动即跟随当前时间点上出现的事件，调动可用资源，执行相关任务，使不断出现的问题得以解决，防止事务堆积。）

使用JavaScript的方式：

确定对象、确定属性、确定事件、用js实现功能。

JavaScript的特点：

松散性：它的变量不必具有一个明确的类型。

对象属性：JavaScript中的对象把属性名映射为任意的属性值。

继承机制：它的面向对象继承机制是基于原型的。

JavaScript的核心：

核心（ECMAScript）

文档对象模型（DOM）

浏览器对象模型（BOM）

变量：

前面输入的数据，后面要使用，需要定义变量，变量就用来存储程序当中需要用到的数据的。

特点：（1）一个变量只能存储一个数据，后来存储的数据覆盖前面存储的数据。（2）变量存储数据可以改变。

用关键字var定义变量。

数据类型：

基本数据类型：number数值型、string字符串型、boolean布尔类型、undefined未定义类型。

引用数据类型：object、function、array、null

进制：

8进制和16进制如何和十进制区分，8进制的前缀加个0,16进制的前缀加0x。

8进制：每三位二进制表示1位。

16进制：每四位二进制表示1位。

连接符号和转义字符：

+连接符：实现多个数值同时进行显示。

字符串类型和任何类型的数据相连接，最终都是字符串。

转义字符：通过反斜杠改变后边所跟字符的含义。（\n换行）

算数运算符：

+ - \* /（除） %（取余数） ++（自增运算符） --（自减）

运算符和变量或常量组合在一起形成了表达式，表达式有一个最终的结果。

2017-8-31

赋值运算符：

= += -= %= \*=

把赋值号右边的值赋给左边的变量。赋值表达式的值就是赋值号右边的值。

关系运算符：

>= < <= == === != !==

==和！=:比较数值，不比较数据类型。

===和！==:比较数值和数据类型。

逻辑运算符：

&& || ！

&&：如果左边的表达式的值为false，那么就不再计算右边表达式。

||：如果左边的表达式为true，那么就不再计算右边的表达式。

0相当于false，非0相当于true；

空字符串相当于false，非空字符串相当于true；

选择结构：

if 和switch

switch中break是停止的意思，没有break就会继续执行下面的语句，直到遇到break为止。在default下没有写break，执行完语句就会遇到右大括号也可以停止执行。如果把default放在case的前面，如果default没break，那么程序就会一直执行下去。总之程序只有遇到break或右大括号才会停止。

case 后边的值可以是常量，变量，表达式。

循环结构：

while循环解决的是具备重复件的问题。

While(){

循环体；

}

原理：先判断条件是否为真，如果为真则执行循环体一次，然后再判断条件是否为真，如果为真则再执行循环体，如此反复，直到条件为假时循环结束，循环结束后继续执行下面的语句。

do-while循环

do{

循环体；

}

while()

原理：先执行循环体一次，然后判断条件，如果条件为真则执行循环体，然后再判断条件，如此反复直到条件为假循环结束。

for循环

for(循环变量初始化语句；循环条件；改变循环变量值得语句){

循环体；

}

break:结束其所在的循环。

continue：结束的是本次循环，直接进入下一次循环。

2017-9-1

函数：

函数就是用来解决相同功能不用重复写多次的问题。

函数的参数使函数的功能更加灵活，可以让函数实现接收任意的数值。

如何自定义函数，一是否用到了不确定的数值，如果用到了就要使用参数，二再分析最终的结果是不是数值，如果是则返回。

单独使用return，它的作用就是结束函数的执行，返回到函数调用的位置。函数之间是可以相互调用的。

递归：

递归是函数自己调用自己。

在使用递归是要注意一调用次数不要太多，否则会出现内存溢出，二要有能够使调用结束下来得条件。

对象：

Js中的数据类型：

基本类型：number、string、boolean、undefined 只能存储数据

引用类型：Object是最基本的类型 特点：是数据和功能（函数）的组合。

对象：把数据和功能封装在一起， 对象.数据 对象.功能

创建对象可以使用new Object

创建对象是可以省略new。

创建对象还可以使用字面量方式

如Obj.age中的.运算符是访问对象的age。对象既可以存储数还可以包含功能。

从内存的角度来看两种类型的对比：

基本类型的变量是在栈中开辟一块内存，引用类型的变量会在栈中开辟一块内存，而对象是在堆中开辟内存，对象变量在栈中存放的是对象在堆中的内存地址。

基本类型的变量直接存储的就是数据本身。

引用类型的变量存储的是对象的内存地址。

数组：

数组属于引用类型

程序需要存储数据时要定义变量来表示，但是如果变量很多时可以用数组来解决。

创建数组的同时给数组初始化数组，数组的初始长度就是数据的个数。

数组的长度是可以改变的，数组可以存储不同类型的数据。

数组中的常用方法：

toString:把数组中的数据转成字符串并返回。

Join():使用指定的分隔符把数组中的数据连接成字符串。

Push():向数组末添加数据，并返回数组长度。

Pop():删除数组末尾的数，返回被删除的数组。

Unshift():向数组的开头添加数据，返回数组的长度。

Shift():删除数组开头的数据，返回被删除的数据。

Concat():基于当前数组创建一个新数组并返回该数组。

Slice():基于当前数组获取指定区域元素并返回一个新的数组。

Splice():删除、插入、替换。

2017-9-4

时间和日期：

Date(1000):表示从1970.1.1 08:00:00开始算多少毫秒数。

Date("6/16/2009"):内部使用了Date.parse()转换成毫秒数

Date(“May 3,2016”):

如何解决浏览器的兼容问题？

Date.UTC()方法返回日期的毫秒数。

UTC：国际协调时间（世界统一世界）。我们时间为东八区，因此要在多少点的基础上加8。

日期的方法：

通用方法：

toString():转换为字符串

toLocaleString():转换为本地格式。

Valueof():转换为毫秒数。

日期的格式化方法：

toDateString():只想得到日期的星期年月日；

toTimeString():只想得到时分秒。

toLocaleDateString()：得到本地的日期。

toLocaleTimeString()：得到本地的时分秒。

组件方法：

getTime() //获取日期的毫秒数，和valueOf()返回一致。

setTime(100) //以毫秒数设置日期，会改变整个日期

getFullYear() //获取四位年份

setFullYear(2018) //设置四位年份，返回毫秒数

getMonth()+1 //获取月份要加1

setMonth() //设置月份

getDate() //获取日期

setDate(9) //返回星期几，0表示星期日，6表示星期六

setDay(3) //设置星期几

getHours() //返回时

setHours() //设置时

getMinutes() //返回分钟

setMinutes() //设置分钟

getSeconds() //返回秒数

getMilliseconds() //返回毫秒数

setMilliseconds() //设置毫秒数

getTimezoneOffset() //返回本地时间和UTC时间相差的分钟数

字符串

Arr.sort():自动排序

字符串常用方法：

字符串中的每一个字符都有一个索引，索引从0开始。

charAt(index):根据索引得到字符串中的某一个字符

charCodeAt(index):返回索引对应的字符串的编码

String.fromCharCode():根据编码生成字符串

indexOf(str):默认返回在字符串中第一次出现的索引。

indexOf(str,position):返回从position指定的位置开始在字符串中第一次出现的索引。

lastIndexOf(str):返回在字符串中最后一次出现的索引

subStr(index1,index2)第一个参数表示截取的起始索引，第二个参数表示截取的字符的个数。

subString(index1,index2):第一个参数表示截取的起始索引，第二个参数表示截取的结束位置（不包含结束位置上的字符，到结束位置的前一位）。

b.math(“a”):以数组的形式返回，数组中是b中第一个和“a”匹配的字符串。如果没有匹配的字符串则返回null空对象。

search():字符串第一次出现的起始位置。如果没有则返回-1;

replace():默认替换第一个，原有字符串不变。

split():切割字符串存到数组里面。

toUpperCase():小写字符串变成大写字符串。

toLowerCase():大写字符串变成小写字符串。

正则表达式

isNaN():判断是否是数字，如果是则返回false。

正则表达式主要用来验证客户端的输入数据，用户填写完表单单击按钮之后，表单就会发送到服务器，在服务器端通常会用PHP，ASP，NET等服务器脚本对其进行进一步处理，因为客户端验证，可以节约大量的服务器端的系统资源，并且提供更好的用户体验。

验证方式：

test():字符串中含有和正则表达式匹配的则返回true。

excel（）:返回匹配的字符串（返回的类型是数组对象）。

new RegExp(string pattern) i:不区分大小写 g:全局匹配 m:多行匹配。如 var reg=new RegExp(“hello”,”i”);

String对象也提供了4个使用正则表达式的方法：

1. match(pattern)返回pattern中的子串或null,返回数组，数组中含有匹配的字符串。
2. search（pattern）返回字符串中pattern开始位置。
3. replace(pattern,replacement)用replacement代替pattern。
4. split(pattern)返回字符串按指定pattern拆分的数组。

2017-9-5

正则表达式规则

. :代表任意字符（换行除外）。

?:出现0次或1次、\*：出现0次或多次、+：出现1次或多次。

{m,n}:至少出现m次，最多出现n次 {m,}:至少出现m次 {m}:正好出现m次.

[]限定某一位上字符的取值范围 [^]:限定某一位上不能取的值.

\d等价于[0-9] \D等价于[^0-9] \w等价于[a-zA-Z0-9\_] \W等价于[^a-zA-Z0-9\_].

\s匹配空白字符、空格、制表符和换行符。

贪婪：+ \* ？ {m,n} {m,} {m} 最大能匹配的字符串。

惰性：+? \*? ?? {m,n}? {m,}? {m}? 最小匹配的字符串。

使用exec()方法返回匹配字符串数组。

捕获性分组。数组中第一个数据是整个正则表达式的字符串，数组中第二个数据是第一个分组中的字符串，数组中第三个数据是第二个分组中的字符串。

非捕获分组？：不想捕获。

前瞻捕获?=。

换行模式g全局匹配，m多行匹配。

Function类型

Function类型实际上是引用类型，每个函数都是function类型的对象，而且都其他引用类型一样具有属性和方法，由于函数是对象，因此函数实际上也是一个指向函数对象的引用类型变量。

函数本身就是变量，所以函数也可以作为值来使用，也就是说，不仅可以像传递参数一样把一个函数传递给另一个函数，而且可以将一个函数作为另一个函数的结果返回。

函数内部属性：

1. arguments属性：指向一个类似数组但不是数组的对象，存储的实际传递给函数的参数，而不局限于函数声明所定义的参数列表。
2. length属性：函数定义时所指定的参数个数。
3. arguments中的callee属性：它表示对函数对象本身的引用（相当于函数名）。

全局变量，任何一个函数都可以使用是公共的。

局部变量：只能被其所在的函数使用，当局部变量和全局变量同名时，全局变量无效。

this:是一个引用，指向函数被调用时的对象。

Function内部有一个prototype属性；该属性含有两个方法call(),apply();

两者的传递参数形式不一样，call(this,a,b); apply(this,[a,b])。

2017-9-6

两种数据类型的对比

JavaScript变量是松散型的（不强制类型），决定了它只是在特定时间用于保存特定值的一个名字而已，变量中存储的数据的类型也是可以改变的

作用域

作用域：在程序中定义的变量或函数能够被使用的范围。全局：在程序中的任何位置都能使用。局部：在程序的某个范围内可以使用。

全局的变量和函数，都是Window对象的属性和方法。

垃圾回收

JavaScript具有自动垃圾收集机制，它会自动管理内存分配及无用内存的回收。最常用的垃圾收集方式是标记清除。

内置对象

Glodal(全局)对象：所有在全局作用域定义的变量和函数，都是global对象的属性和方法。

2017-9-7

面向对象与原型

开发有两种模式：1.函数模式，2.面向对象

使用工厂模式创建对象：解决实例化对象产生大量重复问题。（Demo37）

构造函数创建对象：（Demo38）

工厂模式解决了重复实例化的问题，但是存在识别问题，因为根本无法清除他们到底是哪个类型，因此采用构造函数来创建特定对象。

使用构造函数的方法，和使用工厂模式的方法他们不同之处如下：

构造函数方法没有显示的创建对象(new Object());

直接将属性和方法赋值给this对象；

没有return语句

构造函数的特点

1.函数名首字母大写，为了和普通函数区分

2.使用构造函数必须使用new

继承

继承是面向对象中一个比较核心的概念，而实现继承的方式依靠原型链完成。

被继承的类型称为父类，继承的称为子类,可以把子类对象看成父类类型。

使用原型实现继承存在的问题

当原型中存在引用类型时，存在数据修改时的问题。

子类类型的对象无法给父类传参数。

原型链继承、对象冒充、组合继承、原型式继承、寄生式继承、寄生组合继承。

Math的方法

Math.min();

Math.max();

Math.ceil(num);//返回大于等于该数的最小整数

Math.floor(num);//返回小于等于该数的最大整数

Math.round(num);//四舍五入取整

Math.random();//产生随机数方法，随机数大于等于0.0小于1.0

Math.abs(num);//返回num的绝对值

Math.exp(num);//返回Math.E的num次幂

Math.log(num);//返回num的自然对数

Math.pow(num,power);//返回num的power次幂

Math.sqrt(num);//返回num的平方根

Math.acos(x);//返回x的反余弦值

Math.asin(x);//返回x的反正弦值

Math.atan(x);//返回x的反正切值

Math.atan2(y,x);//返回y/x的反正切值

Math.cos(x);//返回x的余弦值

Math.sin(x);//返回x的正弦值

Math.tan();//返回x的正切值

2017-9-8

匿名函数

匿名函数就是没有名字的函数。

把匿名函数赋值给一个变量

闭包

在一个函数内部创建另一个函数，通过另一个函数访问这个函数的局部变量。

使用闭包的优势：实现一个变量累加。

调用函数时为局部变量开辟内存，函数调用结束时，局部变量内存立刻被释放

全局变量是公共的，容易发生问题

不适用全局变量实现让age的值主键递增---闭包

闭包使用的局部变量不会立刻释放内存，会在内存驻留一段时间。

注意：尽量少用闭包，在必要时再使用闭包。

闭包中的this在运行时指向window，因为闭包并不属于这个对象的属性或方法。

BOM

浏览器对象模型：提供了独立于内容而与浏览器窗口进行交互的对象，并且每个对象都提供了很多方法与属性。

BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是window

window.closed//当窗口关闭时为真

window.defaultStatus//窗口底部状态栏显示的默认状态消息

window.document//窗口中当前显示的文档对象

window.frames//窗口中的框架对象数组

window.history//保存有窗口最近加载的url

window.length//窗口中的框架数

window.location //当前窗口的url

window.name//窗口名

window.offscreenBuffering//用于绘制新窗口内容并在完成后复制已经存在的内容，控制屏幕更新

window.opener//打开当前窗口的窗口

//Window对象的常用属性和方法

//1.系统对话框

//2.新建窗口

//3.窗口的位置和大小

//4.间隔调用和超时调用

confirm("确认") //弹出确认对话框，点击确认返回true，否则返回false。

prompt("请输入一个数据")//弹出让用户输入数据的对话框,返回用户输入的值，点击取消返回值是null,第二个参数是默认值。

print();//显示打印对话框

find();//显示查找对话框