一、各个正则表达式的理解

(\b\w+\b)(?:\s\1)+：\b 的意思是在单词的边界处断定位置

\w 的意思是匹配所有的单词字母

\b 也是在单词的边界处断定位置

\s 的意思是匹配所有的空格元素或者转行

\1 的意思是匹配和前面的单词完全一样的单词

\gi 的意思是不区分字母的大小写

^[1][3,4,5,7,8][0-9]{9}$：1 的意思是匹配首字母是 1

[3,4,5,7,8]的意思在第二个字母的位置匹配的是[3,4,5,7,8]列表中的数组

[0,9]{9}的意思是将匹配九次，每一次的匹配条件是该字符是否是[0,9]中的数字

^\w+ ((-\w+)|(\.\w+))*\ :

匹配开头以 至少有一次以上的 A-Za-z0-9 任何字符

（包括下划线），后面可以有中划线，或者可以有点号，或者是 A-Za-z0-9 任何字符，

以上条件至少有一次的匹配。

@[A-Za-z0-9]+

接着下一个字符是@，@后边是这些字符，至少又一次；

((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$/

后面这些可以有.号，-号，字符，至少有一次，结尾是 A-Za-z0-9，至少有一次。

二、对继承不同方式的理解：

原型链继承

```
//原型链
function Country1() {
    this.name = "国家";
}
function DevelopingCountry1() {

}
DevelopingCountry1.prototype=new Country1();
DevelopingCountry1.prototype.sayHi=function () {
    console.log("Hi,i am a developing country.")
}


function PoorCountry1 (){

}
PoorCountry1.prototype=new Country1();
PoorCountry1.prototype.saySad= function () {
    console.log("I am a sad poor country.")
}
```

让新实例的原原型等于父类的实例

构造函数继承:

```
//构造函数
function Country2() {
    this.name = "国家";
}
function DevelopingCountry2() {
    Country2.call(this);
}
DevelopingCountry2.prototype.sayHi=function sayHi() {
    console.log("Hi,i am a developing country.");
}
function PoorCountry2() {
    Country2.call(this);

}
PoorCountry2.prototype.saySad= function () {
    console.log("I am a sad poor country.")
}
function DevelopedCountry2() {
    Country2.call(this);
}
DevelopedCountry2.prototype.sayHappy=function () {
    console.log("I am a Happy developed country.")
}
```

用 call()将父类的构造函数引入了子类函数，只继承了父类构造函数的属性，没有继承父类原型的属性。

Object.create():

```
//Object.creat
function Country() {
    this.name = "国家";
}
var DevelopingCountry=Object.create(Country);
DevelopingCountry.sayHi=function () {
    console.log("Hi,i am a developing country.");
}



var PoorCountry=Object.create(Country,);
PoorCountry.saySad=function () {
    console.log("I am a sad poor country.")
}
var DevelopedCountry=Object.create(Country);
DevelopedCountry.sayHappy=function () {
    console.log("I am a Happy developed country.")
}
```

Object.create()方法创建一个新对象，使用现有对象来提供新创建的对象的__proto__

三 Map、Set、Array 之间的区别和使用。

Array： 数组对象，是使用单独的变量名来存储一系列的值。

Set： ES6 提供了新的数据结构。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。

Map： ES6 提供了新的数据结构。它类似于对象，也是键值对的集合，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。Map 结构提供了“值—值”的对应。