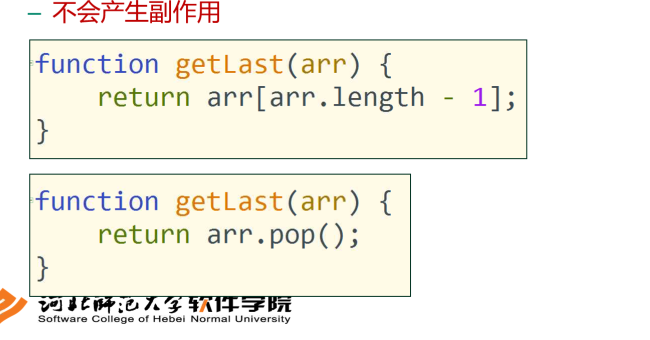


**纯函数：相同的输入得到相同的输出。不会产生副作用。**

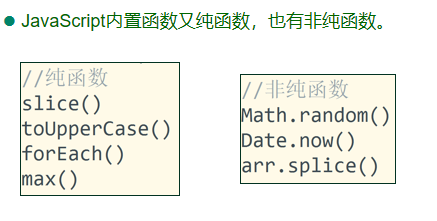
**push()和unshift()：向数组的 尾部/头部 添加若干元素，并返回 数组的 **新长度**；**

**pop()和shift()**

**从数组的 尾部/头部 删除1个元素(删且只删除1个)，并返回 **被删除的元素**；空数组是继续删除，不报错，但返回undefined；**



**Arr=[1,2,3,4,5]第一个是纯函数，多次调用都会输出5,而第二个pop二次调用就会输出不同的数。因为每次调用都会删除最后一个数。**



**Slice()：从已有的数组中返回选定的元素。**

Math.random()：

**函数返回一个浮点,  伪随机数在范围**[0，1)**，也就是说，从0（包括0）往上，但是不包括1（排除1），然后您可以缩放到所需的范围。实现将初始种子选择到随机数生成算法;它不能被用户选择或重置。**

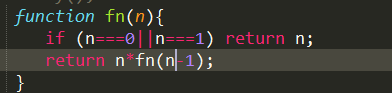
splice() **方法向/从数组中添加/删除项目，然后返回被删除的项目。**

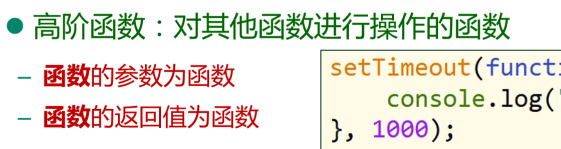
**纯函数：相同的输入总会得到相同的输出**

**纯函数的特点：可以移植，不依赖外部环境，可缓存性。**

**非纯函数的特点：依赖外部环境。**

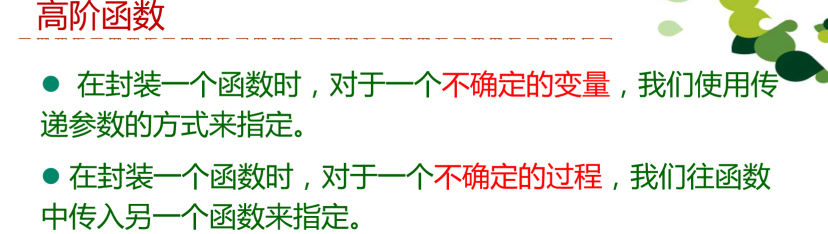
递归输出：

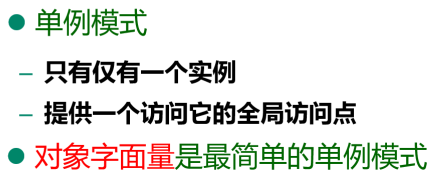




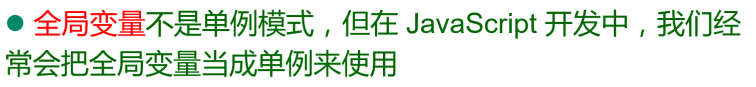
**filter()方法使用指定的函数测试所有元素，并创建一个包含所有通过测试的元素的新数组。**

**filter()基本语法： arr.filter(callback[, thisArg])**





比如 person={name:”lili”,age:20} 这就是单例模式



**不论在哪个位置都能访问到这个值**

