函数组合

又名 【饲养函数】 -> compose 若干个纯函数、偏函数、柯里化函数组合成一个新的函数， 形成数据传递，并实现一种有序执行的效果。

举个栗子：

```
function toUpperCase(str) {
  return str.toUpperCase();
}
function exclaim(str) {
  return str + '~!'
}

console.log(toUpperCase(exclaim('you wanna something big')))
// YOU WANNA SOMETHING BIG~!
```

如此能达到目的，但是代码不优雅，不易读

所以，可以写一个组合函数，就像一个工具一样可以使用 toUpperCase 和 exclaim 的功能。

```
// 这个函数接收两个函数参数
function compose(f, g) {
  // 包裹这层函数的原因是不想让f函数执行
  return function(x) {
    return f(g(x))  // 左倾
  }
}
```

> 左倾：自右向左依次执行 高阶函数中的左倾：函数传入的另一个函数，这个函数先执行，返回给当前函数，当前函数再执行。

根据执行顺序，依次传入 exclaim 和 toUpperCase 两个功能函数。 也是左倾。

```
var f = compose(exclaim, toUpperCase)
console.log(f('now you can change me to bigcase'))
```

变量 f 接收到两个函数参数，再传入字符串执行

再举个栗子:

```
  // 再次加入若干工具函数
function split(str) {
  return str.split('');
}

function reverse(str) {
  return str.reverse();
}

function join(str) {
  return str.join('')
}
```

此时，我们就不能像刚才的组合函数一样，固定两个参数了。

tips： 利用 arguments 收集所有传入的函数参数

```
// 组合函数这样写
function compose1() {
  var args = Array.prototype.slice.call(arguments);
  var index = args.length - 1;
  return function(x) {
    // return f(g(x)); // 原来是左倾两个函数
    var res = args[index](x);//args[index](x) == g(x)
    while(index--) { // index==0 就停止了
      res = args[index](res)  // f(g)
    }
    return res;
  }
}
```

进行组合函数的使用

```
var f1 = compose1(exclaim,join,reverse,split,toUpperCase);
console.log(f1('word here')) ;// EREH DROW~!
```