08.箭头扩展和尾调用

学习要点：

1.箭头扩展

2.尾调用优化

本节课我们来开始学习箭头一些别的用法，以及尾调用优化的方法。

一. 箭头扩展

1. 箭头也支持一些内置函数的使用，比如 sort()排序；

```
let arr = [3, 1, 2].sort((a, b) => a - b);
console.log(arr);

//翻译后的代码为：
let arr = [3, 1, 2].sort(function (a, b) {
    return a - b;
});
```

2. 箭头函数不支持 arguments 绑定，请直接使用...other 模式(rest 运算符)；

```
//下面这种写法不支持
let fn = (x, y) => {
    return arguments[0] + arguments[1]
};

//不确定参数，使用...
let fn = (...other) => {
    return other[0] + other[1]
};
console.log(fn(10, 20));
```

3. 箭头函数和普通函数一样，都可以被 typeof 和 instanceof；

```
console.log(typeof fn);
console.log(fn instanceof Function);
```

二. 尾调用优化

1. 什么是尾调用？即在一个函数的最后可执行的一步调用了其它函数；

```
function go(x) {
    return x + 20;
}

let fn = function (x) {
    return go(x);
```

```
};
console.log(fn(10));
```

2. 那？什么又是尾调用优化？为何要优化？因为：每次尾调用都会创建栈帧；
3. 如果尾调次数过多，而内存中的调用栈越来越大，可能就会出现程序问题；
4. 尤其是在递归函数的问题上，尾调用优化适合在这种场景中使用；
5. 首先要说明，尾调用优化必须是在 ES6 的严格模式下，'use strict';
6. 严格模式，可以设置为全局作用域，也可以在函数体内有效；
7. 严格模式对变量、对象和函数做了一些代码规范等等，具体规范可以搜索；
8. 而对于尾调用，必须严格按照三个规则，才能执行严格模式下的优化，如下：

(1) .尾调用必须 return 返回；　　// go(x); 错误
(2) .尾调用 return 返回不得含其它操作 // return go(x) + 1; 错误
(3) .尾调用 return 返回的不是函数，而是函数赋值的变量，不在尾部；

```
//let result = go(x);
//return result;
```

```
'use strict';

function fn(x) {
    if (x <= 1) {
        return 1;
    }
    return fn(x - 1);
}

console.log(fn(10));
```