JavaScript

生僻知识点

变量

- 变量提升 更多请参考深入理解JavaScript的作用域与变量提升(hoisting)
- "use strict"

•

isNAN

参考博客

```
// true
isNaN({});
                // false
isNaN(true);
                // false
isNaN(null);
isNaN(37);
                 // false
// strings
isNaN("37");
                 // false: 可以被转换成数值37
isNaN("37.37"); // false: 可以被转换成数值37.37
isNaN("37,5"); // true
isNaN('123ABC'); // true: parseInt("123ABC")的结果是 123, 但是Number("123ABC")结果是 NaN isNaN(""); // false: 空字符串被转换成0 isNaN(""); // false: 包含空格的字符串被转换成0
isNaN("");
isNaN(" ");
// dates
isNaN(new Date());
                                  // false
isNaN(new Date().toString());
                 // true: "blabla"不能转换成数值
isNaN("blabla")
                  // 转换成数值失败, 返回NaN
```

异步/同步

参考代码:

```
async function testAsync() {
  var i = 0;
  while (i<100){
    await sleep(1000)
    console.log('1 - testAsync inner ... ')
    i++
  }
  return '跑完啦。。。'
}

// 结合 'await', 实现线程休眠
function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms))
}
```

```
console.log('start ------')
testAsync().then(_re => {console.log(_re)})
console.log('end ------')
```

执行结果:

柯里化

多个连续的箭头函数与柯里化

json

json是js的严格子集;

json5是json的扩展; JSON5 仍然是 JavaScript 的严格子集,不添加任何新的数据类型,并且可以处理所有现有的 JSON 内容

尾调

ES6 的尾调用优化只在严格模式下开启,正常模式无效了解:尾递归

深拷贝

```
var deepCopy = function(src) {
   var ret = {}
   for (var k in src) {
      ret[k] = typeof src[k] ==='object' ? deepCopy(src[k]) : src[k]
   }
   return ret
}
```

踩坑

1. 可以使用 setTimeout 实现循环效果

```
function startTime(){
    var today=new Date();
    var h=today.getHours();
    var m=today.getMinutes();
    var s=today.getSeconds();// 在小于10的数字前加一个'0'
    m=checkTime(m);
    s=checkTime(s);
    document.getElementById('txt').innerHTML=h+":"+m+":"+s;
    t=setTimeout(function(){startTime()},500);
}
```

不使用setInterval的原因:

在开发环境下,很少使用间歇调用(setInterval),原因是后一个间歇调用很可能在前一个间歇调用结束前启动

2. 字符串转数字 JavaScript字符串转数字的5种方法及其陷阱