Node.js 全局对象 Node.js 文件系统

Node.js 常用工具

util 是一个Node.js 核心模块，提供常用函数的集合，用于弥补核心JavaScript 的功能 过于精简的不足。

util.inherits

util.inherits(constructor, superConstructor)是一个实现对象间原型继承 的函数。

JavaScript 的面向对象特性是基于原型的，与常见的基于类的不同。JavaScript 没有 提供对象继承的语言级别特性，而是通过原型复制来实现的。

在这里我们只介绍util.inherits 的用法，示例如下：

```
var util = require('util');
function Base() {
    this.name = 'base';
    this.base = 1991;
    this.sayHello = function() {
    console.log('Hello ' + this.name);
    };
}
Base.prototype.showName = function() {
    console.log(this.name);
};
function Sub() {
    this.name = 'sub';
}
util.inherits(Sub, Base);
var objBase = new Base();
objBase.showName();
objBase.sayHello();
console.log(objBase);
var objSub = new Sub();
objSub.showName();
//objSub.sayHello();
console.log(objSub);
```

我们定义了一个基础对象Base 和一个继承自Base 的Sub，Base 有三个在构造函数 内定义的属性和一个原型中定义的函数，通过util.inherits 实现继承。运行结果如下：

```
base
Hello base
{ name: 'base', base: 1991, sayHello: [Function] }
sub
{ name: 'sub' }
```

注意：Sub 仅仅继承了Base 在原型中定义的函数，而构造函数内部创造的 base 属 性和 sayHello 函数都没有被 Sub 继承。同时，在原型中定义的属性不会被console.log 作 为对象的属性输出。如果我们去掉 objSub.sayHello(); 这行的注释，将会看到：

```
node.js:201
throw e; // process.nextTick error, or 'error' event on first tick
^
TypeError: Object #&lt;Sub&gt; has no method 'sayHello'
at Object.&lt;anonymous&gt; (/home/byvoid/utilinherits.js:29:8)
at Module._compile (module.js:441:26)
at Object..js (module.js:459:10)
at Module.load (module.js:348:31)
at Function._load (module.js:308:12)
at Array.0 (module.js:479:10)
at EventEmitter._tickCallback (node.js:192:40)
```

---

util.inspect

util.inspect(object,[showHidden],[depth],[colors])是一个将任意对象转换 为字符串的方法，通常用于调试和错误输出。它至少接受一个参数 object，即要转换的对象。

showHidden 是一个可选参数，如果值为 true，将会输出更多隐藏信息。

depth 表示最大递归的层数，如果对象很复杂，你可以指定层数以控制输出信息的多 少。如果不指定depth，默认会递归2层，指定为 null 表示将不限递归层数完整遍历对象。 如果color 值为 true，输出格式将会以ANSI 颜色编码，通常用于在终端显示更漂亮 的效果。

特别要指出的是，util.inspect 并不会简单地直接把对象转换为字符串，即使该对 象定义了toString 方法也不会调用。

```
var util = require('util');
function Person() {
    this.name = 'byvoid';
    this.toString = function() {
    return this.name;
    };
}
var obj = new Person();
console.log(util.inspect(obj));
console.log(util.inspect(obj, true));
```

运行结果是：

```
Person { name: 'byvoid', toString: [Function] }
Person {
  name: 'byvoid',
  toString:
```

```
    { [Function]
      [length]: 0,
      [name]: '',
      [arguments]: null,
      [caller]: null,
      [prototype]: { [constructor]: [Circular] } } }
```

util.isArray(object)

如果给定的参数 "object" 是一个数组返回true，否则返回false。

```
var util = require('util');

util.isArray([])
  // true
util.isArray(new Array)
  // true
util.isArray({})
  // false
```

util.isRegExp(object)

如果给定的参数 "object" 是一个正则表达式返回true，否则返回false。

```
var util = require('util');

util.isRegExp(/some regexp/)
  // true
util.isRegExp(new RegExp('another regexp'))
  // true
util.isRegExp({})
  // false
```

util.isDate(object)

如果给定的参数 "object" 是一个日期返回true，否则返回false。

```
var util = require('util');

util.isDate(new Date())
  // true
util.isDate(Date())
  // false (without 'new' returns a String)
util.isDate({})
  // false
```

util.isError(object)

如果给定的参数 "object" 是一个错误对象返回true，否则返回false。

```
var util = require('util');

util.isError(new Error())
  // true
util.isError(new TypeError())
  // true
util.isError({ name: 'Error', message: 'an error occurred' })
  // false
```

更多详情可以访问 http://nodejs.org/api/util.html 了解详细内容。

Node.js 全局对象 Node.js 文件系统