## \_\_filename

**\_\_filename** 表示当前正在执行的脚本的文件名。

## \_\_dirname

**\_\_dirname** 表示当前执行脚本所在的目录。

## setTimeout(cb, ms)

**setTimeout(cb, ms)** 全局函数在指定的毫秒(ms)数后执行指定函数(cb)。

setTimeout() 只执行一次指定函数。

### **实例**

创建文件 main.js ，代码如下所示：

function printHello(){

console.log( "Hello, World!");}// 两秒后执行以上函数

setTimeout(printHello, 2000);

## setInterval(cb, ms)

**setInterval(cb, ms)** 全局函数在指定的毫秒(ms)数后执行指定函数(cb)。

返回一个代表定时器的句柄值。可以使用 **clearInterval(t)** 函数来清除定时器。

setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。

### **实例**

创建文件 main.js ，代码如下所示：

function printHello(){

console.log( "Hello, World!");}// 两秒后执行以上函数

setInterval(printHello, 2000);

## clearTimeout(t)

**clearTimeout( t )** 全局函数用于停止一个之前通过 setTimeout() 创建的定时器。 参数 **t** 是通过 setTimeout() 函数创建的定时器。

### **实例**

创建文件 main.js ，代码如下所示：

function printHello(){

console.log( "Hello, World!");}// 两秒后执行以上函数

var t = setTimeout(printHello, 2000);

// 清除定时器

clearTimeout(t);

## console

console 用于提供控制台标准输出，它是由 Internet Explorer 的 JScript 引擎提供的调试工具，后来逐渐成为浏览器的实施标准。

Node.js 沿用了这个标准，提供与习惯行为一致的 console 对象，用于向标准输出流（stdout）或标准错误流（stderr）输出字符。