

El módulo predefinido `path` es muy utilizado y se encuentra muy relacionado con los nombres de entradas y las rutas de un sistema de ficheros. Por esta razón, vamos a dedicarle su propia lección.

La lección comienza presentando el concepto de ruta. A continuación, mostramos cómo consultar el separador utilizado en el sistema operativo en el que se está ejecutando `node`. Después, se presenta varias funciones con las que trabajar con distintos aspectos de una ruta dada.

Al finalizar la lección, el estudiante sabrá:

- Cómo consultar el separador de rutas.
- Cómo comprobar si una ruta es absoluta o relativa.
- Cómo obtener el nombre de una entrada a partir de su ruta.
- Cómo obtener la extensión de un archivo.
- Cómo obtener la ruta al directorio padre de una ruta.
- Cómo crear una ruta a partir de sus partes.
- Cómo normalizar una ruta.
- Cómo resolver una ruta.

Introducción

Recordemos que una *ruta* (*path*) indica la ubicación exacta de una entrada del sistema de ficheros como, por ejemplo, la de un archivo o un directorio. Indica los directorios por los que hay que pasar hasta llegar a la entrada en cuestión, así como el nombre de la entrada. Cada una de las partes se separa de la anterior por un *separador* (*separator*), un carácter especial que depende del sistema operativo. En **Windows**, se usa el carácter `\`; y en **Linux**, `/`.

He aquí la misma ruta, pero en sistemas operativos distintos:

```
/mi/directorio/hacia/el/archivo.txt  
\\mi\directorio\hacia\el\archivo.txt
```

Como **Node** es multiplataforma, debe de lidiar con el separador. Buscando así hacer los paquetes independientes de la plataforma. Una tarea un poco desagradable. Pero para ayudar a hacer los paquetes más robustos, se recomienda utilizar el módulo `path`.

Veamos los miembros más utilizados de este módulo.

Separador

Tal como acabamos de ver, el separador es un carácter especial que delimita los elementos de una ruta. Para conocer cuál se usa en el sistema operativo en el que se está ejecutando `node`, podemos usar la propiedad `path.sep`. Una cadena de texto que contiene el carácter separador: `\` en **Windows** y `/` en sistemas **POSIX**.

Ruta absoluta

Recordemos que las rutas pueden ser relativas o absolutas. Una *ruta absoluta* (*absolute path*) es aquella que indica la localización completa de la entrada del sistema de ficheros. Comenzando en la raíz. Ejemplo: `/home/me/.bashrc`.

En cambio, una *ruta relativa* (*relative path*) es aquella que se indica a partir del directorio actual o del padre. Como por ejemplo: `me`, `../me` o `./me`.

Para saber si una ruta dada es absoluta, se puede utilizar la función `isAbsolute()`:

```
function isAbsolute(path) : boolean
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string	Ruta a analizar.

Ejemplos:

```
> path.isAbsolute("/esto/es/un/ejemplo")
true
> path.isAbsolute("esto/es/un/ejemplo")
false
> path.isAbsolute("./esto/es/un/ejemplo")
false
> path.isAbsolute("../esto/es/un/ejemplo")
false
>
```

Nombre de la entrada

Toda entrada del sistema de archivos tiene una ruta que indica donde se encuentra y un nombre único dentro de su directorio padre. Para conocer el nombre de la entrada referenciada por una ruta, se puede utilizar la función `basename()` del módulo `path`:

```
function basename(path) : string
function basename(path, ext) : string
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string	Ruta a analizar.
<code>ext</code>	string	Extensión a suprimir.

La función no comprueba si la ruta existe. Simplemente trabaja sobre ella y extrae su nombre. Cuando se pasa el parámetro `ext`, si la entrada finaliza en la extensión indicada, el nombre devuelto no la contendrá; en otro caso, sí lo hará.

Veamos la función en acción:

```
> path.basename("/esto/es/un/ejemplo.txt")
'ejemplo.txt'
> path.basename("/esto/es/un/ejemplo.txt", ".txt")
'ejemplo'
>
```

Extensión de archivo

La **extensión** (*extension*) es el sufijo que se añade a los archivos, que ayuda a identificar el tipo de su contenido. Por ejemplo: `.txt`, `.js`, `.jsx`, `.pdf`, `.doc`, `.odt`, etc. Si necesitamos conocer la extensión de un nombre de archivo, podemos utilizar la función `extname()`:

```
function extname(path) : string
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string	Ruta o nombre de archivo a analizar.

Veamos unos ejemplos ilustrativos:

```
> path.extname("archivo.txt")
'.txt'
> path.extname("/mi/archivo.txt")
'.txt'
>
```

Directorio padre

También es posible obtener la ruta del directorio padre de una ruta. Para ello, se usa la función `dirname()`:

```
function dirname(path) : string
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string	Ruta a analizar.

Ejemplo ilustrativo:

```
> path.dirname("/esto/es/un/ejemplo.txt")
'/esto/es/un'
>
```

Creación de ruta

En muchas ocasiones, se desea crear una ruta a partir de sus elementos. Para facilitar la obtención de la ruta final, se puede utilizar la función `join()` del módulo `path`:

```
function join(...path) : string
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string[]	Las partes que forman la ruta.

Esta función se utiliza muchísimo. No la olvide.

Sirva lo siguiente como ejemplo:

```
> path.join("esto", "es", "un", "ejemplo")
'esto/es/un/ejemplo'
> path.join("/esto", "es/un", "ejemplo")
'/esto/es/un/ejemplo'
>
```

Observe que la ruta final consiste en la concatenación de las partes indicadas mediante el separador.

Normalización de ruta

La **normalización** (*normalization*) es la operación mediante la cual se obtiene una ruta que, por una parte, resuelve los especificadores de directorios `.` (directorio actual) y `..` (directorio padre) y, por otra parte, reemplaza la duplicidad del separador.

No hay nada como unos ejemplos ilustrativos:

```
> path.normalize("/esto/es/un/buen/./ejemplo")
'/esto/es/un/buen/ejemplo'
> path.normalize("/esto/es/un/buen/../ejemplo")
'/esto/es/un/ejemplo'
> path.normalize("/esto/es/un/buen//ejemplo")
'/esto/es/un/buen/ejemplo'
>
```

Resolución de ruta

La función `resolve()` se utiliza para obtener una ruta absoluta a partir de sus partes. Cuando los elementos resultan en una ruta relativa, la función devuelve una absoluta respecto al directorio actual. Su signatura es como sigue:

```
function resolve(...path) : string
```

Parámetro	Tipo de datos	Descripción
<code>path</code>	string[]	Las partes que forman la ruta.

El comportamiento de esta función se ve mejor mediante unos ejemplos ilustrativos:

```
> process.cwd()
'/home/me'
> path.resolve("esto", "es", "un/buen", "..", "ejemplo")
'/home/me/esto/es/un/buen/ejemplo'
> path.resolve("/esto", "es", "un/buen", "..", "ejemplo")
'/esto/es/un/ejemplo'
>
```

Observe que la función realiza un `join()` y después un `normalize()`. Añadiendo como sufijo la ruta del directorio actual si la ruta resultante es relativa.