Node.js: come gestire gli argomenti della linea di comando

In questo articolo vedremo come creare una semplice applicazione da riga di comando in Node.js.

Installiamo il modulo yargs per il parsing e la gestione avanzata degli argomenti da riga di comando.

```
npm install yargs --save
```

La nostra applicazione genererà una password casuale di lunghezza `length` specificata dall'utente tramite il comando `create`.

```
node app.js create --length=12
```

Per ottenere questo risultato, creiamo prima una funzione di utility che, data una lunghezza definita da un numero intero, genera la password casuale corrispondente.

```
'use strict';

const generate = passwordLength => {
    const chars =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ@#!$%&?*%()[]{}<>|'.split('');
    let output = '';

    for(let i = 0; i < passwordLength; i++) {
        let char = chars[Math.floor(Math.random() *
(chars.length - 1))];
```

```
        output += char;
    }

    return output;
};

module.exports = generate;
```

A questo punto dobbiamo definire il comando `create` con l'opzione obbligatoria `length` che deve essere un numero intero.

```
'use strict';

const yargs = require('yargs');
const createPassword = require('./lib/generate');

yargs.command({
    command: 'create',
    describe: 'Generate a password',
    builder: {
        length: {
            describe: 'Password length',
            demandOption: true,
            type: 'number'
        }
    },
    handler(argv) {
        console.log(createPassword(argv.length));
    }
});

yargs.parse();
```

Il metodo `command()` definisce un comando. Al suo interno l'oggetto `builder` definisce le opzioni del comando. Nel nostro caso l'opzione `length` è obbligatoria (`demandOption`) e deve essere un numero (`type`).

Il metodo `handler()` definisce un callback per gestire il comando. Il suo unico argomento, `argv`, è un oggetto letterale avente come proprietà tutte le opzioni definite in precedenza i cui valori saranno quelli inseriti dall'utente.