CREAZIONE DI UN HTTP SERVER CON NODE.JS

1. IMPORTAZIONE DEL MODULO HTTP

UTILIZZO DEL MODULO HTTP DI NODE.JS

IL MODULO HTTP DI NODE.JS FORNISCE FUNZIONALITÀ PER CREARE SERVER HTTP E GESTIRE RICHIESTE E RISPOSTE

const http = require('http')

2. CREAZIONE DEL SERVER

UTILIZZO DEL METODO CREATESERVER()

IL METODO CREATESERVER() VIENE UTILIZZATO PER CREARE UN SERVER E RICHIEDE UNA FUNZIONE DI CALLBACK CHE VERRÀ ESEGUITA OGNI VOLTA CHE IL SERVER RICEVE UNA RICHIESTA

```
const http =
require('http');

const PORT = 3000; //
Definisci il numero di
porta desiderato

const server =
http.createServer((req,
res) => {
// Logica per gestire la
richiesta e inviare la
risposta
});

server.listen(PORT, () => {
console.log(`Server
running on port
${PORT}`);
});
```

GESTIONE DELLE RICHIESTE E DELLE RISPOSTE

ALL'INTERNO DELLA FUNZIONE DI CALLBACK, È POSSIBILE SCRIVERE LA LOGICA PER GESTIRE LE RICHIESTE E GENERARE LE RISPOSTE, UTILIZZANDO I PARAMETRI REQ E RES

INVIO DELLA RISPOSTA

IL METODO **RES.END() VIENE UTILIZZATO PER INVIARE UNA RISPOSTA AL CLIENT E CHIUDERE LA CONNESSIONE**, OPZIONALMENTE INVIANDO DATI DI RISPOSTA

CREAZIONE DI UN HTTP SERVER CON NODE.JS

3. AVVIO DEL SERVER

PARAMETRI PRINCIPALI

REQ E RES.
RAPPRESENTANO LA RICHIESTA HTTP RICEVUTA DAL CLIENT E LA RISPOSTA INVIATA DAL SERVER, MENTRE IL METODO **RES.END()** VIENE UTILIZZATO PER INVIARE UNA RISPOSTA AL CLIENT E IL METODO

```
const server = http.createServer((req, res) => {//vediamo da cosa è composta una richiesta

console.log(req);

//rispondiamo alla richiesta con una pagina html

res.setHeader('Content-type', 'text/html')

//scriviamo un titolo

res.write('<h1>Ho creato un server con Node.js</h1>');

//terminiamo il processo di risposta

res.end();

})
```

res.setHeader

2 PARAMETRI

1) NOME DELL'HEADER

Content-Type: Specifica il tipo di dati contenuti nel corpo della richiesta o della risposta. Ad esempio, "text/plain", "application/json", "image/jpeg", ecc.
Content-Length: Indica la lunghezza in byte del corpo della richiesta o della risposta.
Cache-Control: Fornisce indicazioni sulla memorizzazione nella cache dei contenuti. Può specificare direttive come "no-cache", "max-age", "public", "private", ecc.
Accept: Indica i tipi di dati che il client è disposto ad accettare dalla risposta. Ad esempio, "text/html", "application/json", "/" (qualsiasi tipo di dati).
Authorization: Contiene le credenziali di autenticazione richieste dal server per accedere alla risorsa richiesta.
User-Agent: Specifica il tipo di browser o client che ha effettuato la richiesta. Questo può essere utilizzato dal server per fornire contenuti ottimizzati per il client.
Location: Utilizzato nelle risposte di reindirizzamento per specificare l'URL a cui il client dovrebbe reindirizzarsi.
Set-Cookie: Utilizzato per impostare un cookie sulla risposta, che verrà quindi inviato dal client nelle successive richieste.

2) VALORE DELL'HEADER

STRINGHE, NUMERI, DATE, BOOLEANI, VALORI SEPARATI DA VIRGOLE, STRUTTURE DATI COMPLESSE (JSON, XML)

ALTRI METODI DI RES

res.write(chunk[, encoding][, callback])
res.end([data][, encoding][, callback])res.getheader
res.setHeader(name, value)
res.getHeader(name)
res.removeheader(name)
res.writeHead(statusCode[, statusMessage][, headers])

UTILIZZO DEL METODO LISTEN()

IL METODO LISTEN() VIENE UTILIZZATO PER FAR INIZIARE AL SERVER L'ASCOLTO DELLE RICHIESTE HTTP SU UNA PORTA SPECIFICA

METODI DI REQ

req.url: Restituisce l'URL della richiesta HTTP, inclusa la stringa di query se presente.
req.method: Restituisce il metodo HTTP della richiesta, ad esempio "GET", "POST", ecc.
req.headers: Restituisce un oggetto contenente gli header della richiesta HTTP inviata dal client.
req.params: Restituisce un oggetto contenente i parametri di percorso della richiesta. È utilizzato quando si utilizzano parametri di percorso dinamici nelle route.
req.query: Restituisce un oggetto contenente i parametri della stringa di query della richiesta HTTP. Utile quando si inviano dati tramite query string.
req.body: Restituisce i dati del corpo della richiesta. Questo metodo è disponibile solo se si utilizza un modulo di parsing del corpo della richiesta, come body-parser.
req.cookies: Restituisce un oggetto contenente i cookie inviati dal client con la richiesta.
req.get(header): Restituisce il valore dell'header specificato dalla richiesta HTTP. Il parametro header è il nome dell'header da cui si desidera ottenere il valore.
req.param(name): Restituisce il valore del parametro specificato dalla richiesta HTTP. Il parametro name è il nome del parametro da cui si desidera ottenere il valore. È una combinazione di req.body, req.query e req.params.
req.is(type): Restituisce true se il tipo di contenuto della richiesta corrisponde al tipo MIME specificato. Il parametro type è il tipo MIME da confrontare.