Http client :- to query URLs

Important function  
  
Go's standard library includes a powerful package net/http for building HTTP clients and servers.

```

package main // Declares the main package — the entry point for the executable

import ( // Imports necessary standard packages

"fmt" // For printing output to the console

"io" // For reading from the response body

"log" // For logging errors or messages

"net/http" // For using HTTP client and request features

"time" // For setting timeouts and delays

)

func main() { // The main function where execution begins

// Create a custom HTTP client with a 10-second timeout

client := &http.Client{

Timeout: 10 \* time.Second, // Sets maximum time allowed for the request

}

// Create a new HTTP GET **request** to the specified URL

**req**, err := http.NewRequest("GET", "https://jsonplaceholder.typicode.com/posts/1", nil)

if err != nil { // If there is an error creating the request

log.Fatal(err) // Log the error and exit the program

}

// Add an HTTP header to the request to specify the desired response format

**req**.Header.Set("Accept", "application/json")

// **Send** the HTTP request **using the custom client**

**resp**, err := client.Do(**req**)

if err != nil { // If there's an error making the request (e.g., network failure)

log.Fatal(err) // Log the error and exit

}

defer resp.Body.Close() // Ensure the response body is closed after reading to free resources

// Read the response body into memory

body, err := io.ReadAll(resp.Body)

if err != nil { // If there's an error reading the body

log.Fatal(err) // Log and exit

}

// Print the HTTP status code (e.g., "200 OK")

fmt.Println("Response status:", resp.Status)

// Print the response body content as a string

fmt.Println("Response body:", string(body))

}

```