In Node.js, the global object provides access to various built-in variables and functions that are available throughout your application.1 It's analogous to the window object in a web browser environment.

Key Global Objects:

* console: Provides methods for logging output to the console (e.g., console.log(), console.error()).
* process: Provides information about the current Node.js process, such as its environment variables, arguments, and standard input/output streams.2
* Buffer: Represents a sequence of bytes and is used for handling binary data.3
* setTimeout(): Schedules the execution of a function after a specified delay.4
* setInterval(): Schedules the repeated execution of a function at a given interval.5
* clearInterval(): Cancels a timer set with setInterval().
* clearTimeout(): Cancels a timer set with setTimeout().
* \_\_dirname: The directory name of the current module.
* \_\_filename: The full file path of the current module.6

Example:

JavaScript

console.log(\_\_dirname); // Output: The directory where the current file is located

console.log(\_\_filename); // Output: The full path to the current file

setTimeout(() => {

console.log("This will be logged after 1 second.");

}, 1000);

Important Notes:

* While you can use global objects directly, it's generally considered good practice to avoid excessive use of global variables.7
* Global variables can lead to unintended side effects and make your code harder to maintain.8
* Consider using modules and local variables whenever possible to improve code organization and avoid potential conflicts.

By understanding and utilizing these global objects effectively, you can leverage the full power of the Node.js environment to build robust and efficient applications.