Understanding the Node.js Wrapper Function and Special Variables

In Node.js, every JavaScript file is wrapped inside a special function before it is executed. This allows each file to have its own private scope, preventing variables from leaking into the global scope.

The Wrapper Function

Node.js wraps your code like this internally:

```
(function(exports, require, module, __filename, __dirname) {
  // Your entire file code lives here
});
```

This is known as the **Module Wrapper Function**. Because of this, your Node.js file gets access to the following special variables:

Special Variables in Node.js

`__filename`

- Returns the absolute path of the current file.

```
console.log(__filename);
// Example: /Users/haris/project/app.js
```

`__dirname`

- Returns the absolute path of the directory that contains the current file.

```
console.log(__dirname);
// Example: /Users/haris/project
```

`exports` and `module.exports`

- Used to export variables or functions from a module.

`require`

- Used to import modules.

`module`

- Represents the current module and contains metadata about it.

Why This Matters

This wrapper allows each file to be treated as a separate module. It ensures:

- Code isolation (no global scope pollution)
- Access to helpful variables like `__dirname` and `__filename`
- A consistent module system using CommonJS

Example

```
// test.js
console.log('Filename:', __filename);
console.log('Directory:', __dirname);
```

When you run this with `node test.js`, it will print the full path of the file and its directory.

Understanding the wrapper function and special variables is key to mastering how Node.js modules work internally.