

```

    addRequire: false, // allows
      `require()`
    experimentalModuleApi: false, //
      uses Module#_compile instead of
      `vm` (you probably don't want
      this and it is bypassed by
      default when vm.Module is
      supported)
  };

  await vm.getGlobalContext({}, options);

```

Changelog

- v6.0.0 Changes import and require to be project relative (not relative to this package on the file system).
- v5.0.0 Removes sync API, swap to async-only. Better compatibility with `--experimental-vm-modules` Node flag.
- v4.0.0 Swap to use `Module._compile` as a workaround for #2 (Node regression with experimental modules API in Node v20.10+)
- v3.0.0 ESM-only package. Node 16+

node-retrieve-globals

Execute a string of JavaScript using Node.js and return the global variable values and functions.

- Supported on Node.js 16 and newer.
- Uses `var`, `let`, `const`, `function`, `Array` and `Object` [destructuring assignment](#).
- Async-only as of v5.0.
- Can return any valid JS data type (including functions).
- Can provide an external data object as context to the local execution scope
- Transforms ESM import statements to work with current CommonJS limitations in Node's `vm`.
- Uses [Node's vm module to execute JavaScript](#)
 - ⚠ The `node:vm` module is not a security mechanism. Do not use it to run untrusted code.
 - `codeGeneration` (e.g. `eval`) is disabled by default; use `setCreateContextOptions({codeGeneration: { strings: true, wasm: true } })` to re-enable.
 - Works *with or without* `--experimental-vm-modules` flag (for `vm.Module` support). (*v5.0.0 and newer*)
 - Future-friendly feature tests for when `vm.Module` is stable and `--experimental-vm-modules` is no longer necessary. (*v5.0.0 and newer*)
- In use on:
 - [JavaScript in Eleventy Front Matter](#) (and [Demo](#))

Installation

Available on [npm](#)

npm install node-retrieve-globals

Usage

Works from Node.js with ESM and CommonJS:

```
import { RetrieveGlobals } from "node-  
retrieve-globals";  
// const { RetrieveGlobals } = await  
import("node-retrieve-globals");
```

And then:

```
let code = `var a = 1;  
const b = "hello";  
function hello() {}`;  
let vm = new RetrieveGlobals(code);  
await vm.getGlobalContext();
```

2

Returns:

```
{ a: 1, b: "hello", hello: function  
  hello() {} }
```

**Pass in your own Data and reference it in the
JavaScript code**

```
let code = `let ref = myData;`;  
let vm = new RetrieveGlobals(code);  
await vm.getGlobalContext({ myData:  
  "hello" });
```

Returns:

```
{ ref: "hello" }
```

Advanced options

```
// Defaults shown  
let options = {  
  reuseGlobal: false, // re-use  
  Node.js `global`, important if  
  you want `console.log` to log to  
  your console as expected.  
  dynamicImport: false, // allows  
  `import()`  
}
```

3