globals

> *Global identifiers from different JavaScript environments*

It's just a JSON file, so use it in any environment.

This package is used by ESLint.

This package no longer accepts new environments. If you need it for ESLint, just create a plugin.

Install

```
$ npm install globals
```

Usage

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

console.log(globals.browser);
/*
{
    addEventListener: false,
    applicationCache: false,
    ArrayBuffer: false,
    atob: false,
    …
}
*/
```

Each global is given a value of `true` or `false`. A value of `true` indicates that the variable may be overwritten. A value of `false` indicates that the variable should be considered read-only. This information is used by static analysis tools to flag incorrect behavior. We assume all variables should be `false` unless we hear otherwise.

For Node.js this package provides two sets of globals:

- `globals.nodeBuiltin`: Globals available to all code running in Node.js. These will usually be available as properties on the `global` object and include `process`, `Buffer`, but not CommonJS arguments like `require`. See: https://nodejs.org/api/globals.html
- `globals.node`: A combination of the globals from `nodeBuiltin` plus all CommonJS arguments ("CommonJS module scope"). See: https://nodejs.org/api/modules.html#modules_the_module_scope

When analyzing code that is known to run outside of a CommonJS wrapper, for example, JavaScript modules, `nodeBuiltin` can find accidental CommonJS references.
