Index.js - Git Commands Module Entry Point

This file serves as the main entry point and module exporter for a Node.js implementation of Git commands. It acts as a centralized hub that imports and exports all the individual Git command implementations.

Purpose

The `index.js` file follows the common Node.js pattern of creating a main module that aggregates and exports multiple related classes or functions. This allows users to import all Git commands from a single location.

Module Structure

Imports

```
const CatFileCommand = require('./cat-file')
const HashObjectCommand = require('./hash-object')
const LSTreeCommand = require('./ls-tree')
const WriteTreeCommand = require('./write-tree')
const CommitTreeCommand = require('./commit-tree');
const CloneCommand = require('./clone');
```

The file imports six different Git command implementations:

- **CatFileCommand**: Examines Git object contents (`git cat-file`)
- **HashObjectCommand**: Creates blob objects from files (`git hash-object`)
- **LSTreeCommand**: Lists tree object contents (`git ls-tree`)
- **WriteTreeCommand**: Creates tree objects from the working directory (`git write-tree`)
- **CommitTreeCommand**: Creates commit objects (`git commit-tree`)
- **CloneCommand**: Clones Git repositories (`git clone`)

Exports

```
module.exports = {
  CatFileCommand,
```

```
    HashObjectCommand,
    LSTreeCommand,
    WriteTreeCommand,
    CommitTreeCommand,
    CloneCommand,
}
```

Uses ES6 shorthand property syntax to export all imported classes as an object. This is equivalent to:

```
module.exports = {
    CatFileCommand: CatFileCommand,
    HashObjectCommand: HashObjectCommand,
    // ... etc
}
```

Usage Patterns

Individual Import

```
const { CatFileCommand } = require('./git-commands');
const catFile = new CatFileCommand('-p', 'abc123def456');
```

Multiple Imports

```
const { CatFileCommand, HashObjectCommand } = require('./git-commands');
```

Full Import

```
const GitCommands = require('./git-commands');
const catFile = new GitCommands.CatFileCommand('-p', 'abc123def456');
```

Project Structure Implications

This structure suggests the project is organized as:

```
project/
├── index.js              # This file - main entry point
├── cat-file.js           # CatFileCommand implementation
├── hash-object.js        # HashObjectCommand implementation
```

```
├── ls-tree.js          # LSTreeCommand implementation
├── write-tree.js       # WriteTreeCommand implementation
├── commit-tree.js      # CommitTreeCommand implementation
└── clone.js            # CloneCommand implementation
```

Git Command Coverage

The module implements core Git plumbing commands:

1. **Object Inspection**: `cat-file` - View object contents
2. **Object Creation**: `hash-object` - Create blob objects
3. **Tree Operations**: `ls-tree`, `write-tree` - List and create tree objects
4. **Commit Operations**: `commit-tree` - Create commit objects
5. **Repository Operations**: `clone` - Clone repositories

Benefits of This Pattern

- **Centralized Access**: Single import point for all commands
- **Modularity**: Each command is in its own file
- **Scalability**: Easy to add new commands by importing and exporting them
- **Clean API**: Consumers don't need to know individual file names
- **Tree Shaking**: Bundlers can eliminate unused commands