OUR UNIT IS A FUNCTION

- Almost always, a unit test will test a single function
- We test a function *independently* of other functions
- Given we have a single function, `parse()`, that's what we're going to test
- But *where do we start*?
- Let's look at *`bargs`*' source again

BARGS' SOURCE

src/index.js

```
/**
 * Parses arguments and returns an object.
 * @param {string[]|BargsOptions} [argv] - Array of arguments to parse;
defaults to `process.argv.slice(2)`. Can also be a `BargsOptions` object.
 * @param {BargsOptions} [opts] - Options
 */
exports.parse = (argv = process.argv.slice(2), opts = {expectsValue: []})
=> {
  if (!Array.isArray(argv)) {
    opts = argv;
    argv = process.argv.slice(2);
  }
  let expectsValue = new Set(opts.expectsValue || []);
  const result = {_: []};
  let pos = 0;
  while (true) {
    let arg = argv[pos];
    if (arg === undefined) {
      return result;
    }
    if (arg.startsWith('-')) {
      if (arg === '--') {
        result._ = [...result._, ...argv.slice(++pos)];
        return result;
      }
      let [realArg, value] = arg.replace(/^-+/, '').split('=');
      if (expectsValue.has(realArg)) {
        result[realArg] = value === undefined ? argv[++pos] : value;
      } else {
        result[realArg] = value === 'false' ? false : true;
      }
    } else {
      result._ = [...result._, arg];
    }
    pos++;
  }
}

/**
 * Options for `bargs`.
 * @typedef {Object} BargsOptions
 * @property {string[]} expectsValue - Array of command-line options that
should be followed by a value
 */

 /**
  * Array of positional arguments
  * @typedef {Object} BargsArgs
  * @property {string[]} _ - Array of positional arguments
  */
```