Article Title

Carl A. Bauer*

carl.a.bauer@gmail.com

Abstract

Yet another javascript packager.

I. Introduction

II. Dependencies

There are two kinds of dependencies, *static* and *dynamic*. A static dependency is merely a technique for splitting up (and sharing) your code; a developer could easily replace static dependencies with copied and pasted code in the dependent module (resulting in a giant mess) and the program would run identically as before. The code in the dependent module need not be run to determine its static dependencies; therefore, tools like browserify can exist to bundle up static dependencies for use in the browser.

Dynamic dependencies, on the other hand, imply that the code in the dependency is required only under certain conditions. That is, the dependent module code *must run* to determine whether or not a dynamic dependency should be loaded.

A static or dynamic dependency between modules i and j (where module i depends on module j) will be specified by the symbol $d^{(s)}_{i,j}$ or $d^{(s)}_{i,j}$, respectively.

Following the CommonJS format (for static deps), a module defines static dependencies with calls to `require('module_name')` and dynamic dependencies by declaring a string literal *immediately* followed by the comment: `/*module*/`. For example:

```
1 var through = require('through');    // static
2 var react = 'react'/*module*/;       // dynamic
```

III. Bundle

A *bundle* is a set of modules defined by a single entry-point module. The set is formed by first adding the entry module to the empty set, then following the entry module's static dependencies (recursively), adding each visited module to the set. We are talking about formal sets here, so if a module is visited twice in the dependency traverse, it occurs only once in the final set.

$$b(i) = \{\, j \,|\, j \in c^{(s)}(i,j) \,\} \tag{1}$$

Every module, m, has a set of reachable dynamic dependencies defined as the set of all dynamic dependencies defined in the modules of the bundle found with

$$D^{(d)}(i) = \{\, k \,|\, k \in d^{(d)}(j),\, j \in b(i) \,\} \tag{2}$$

For example, an entry module for a bundle can access *all* dynamic dependencies defined in the bundle, whereas a static dependency of the entry module (also in the same bundle.

*

IV. Results

V. Discussion

I. Subsection One

II. Subsection Two

References

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.