browserify

Browserify lets you require('modules') in the browser by bundling up all of your dependencies.

Author

Lucas Dreher

About Browserify

Topics

1. Browserify Intro
2. Install
3. Bundle Up Your First Module
4. Live Reload Development Environment
5. Browserify Transforms
6. Pros and Cons

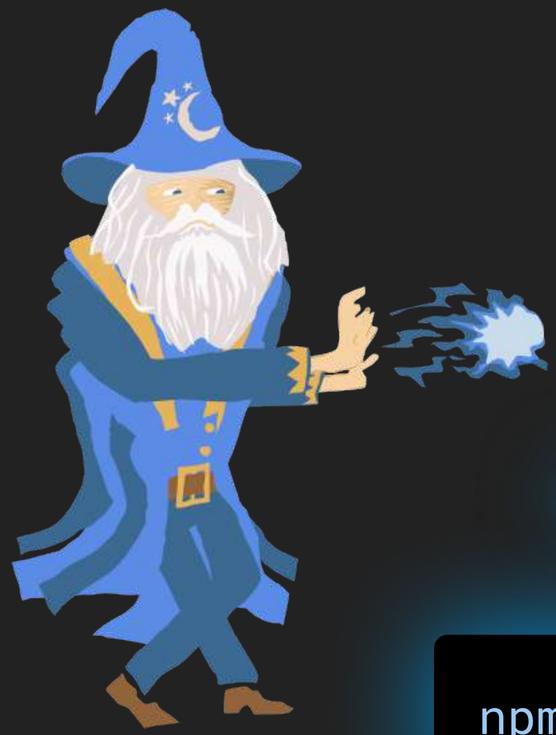
Browserify Intro

Browsers don't have the `require` method defined, but Node.js does. With Browserify you can write code that uses `require` in the same way that you would use it in Node.

With browserify, we can use the thousands of modules on `npm` in our browser-side code.

Use a node-style `require()` to organize your browser code and load modules installed by `npm`.

Browserify will recursively analyze all the `require()` calls in your app in order to build a bundle you can serve up to the browser in a single `<script>` tag.



Install

USE BROWSERIFY FROM THE COMMAND LINE

First install node, which ships with npm. Then do:

```
npm install -g browserify
```

Hello World With Browserify

Bundle Up Your First Module

Step 1

Here is a tutorial on how to use Browserify on the command line to bundle up a simple file called `main.js` along with all of its dependencies:

```
var unique = require('uniq');

var data = [1, 2, 2, 3, 4, 5, 5, 5, 6];

console.log(unique(data));
```

Step 2

Install the `uniq` module with `npm`:

```
npm install uniq
```

Hello World With Browserify

Bundle Up Your First Module

Step 3

Now recursively bundle up all the required modules starting at `main.js` into a single file called `bundle.js` with the `browserify` command:

```
browserify main.js -o bundle.js
```

Step 4

Browserify parses the AST for `require()` calls to traverse the entire dependency graph of your project.
Drop a single `<script>` tag into your html and you're done!

```
<script src="bundle.js"></script>
```

Live Reload Development Environment

If you're in the middle of writing code, you'll find running `browserify` in the terminal to regenerate bundle.js, then refreshing the browser to be time-consuming and annoying.

Enter budo!

budo is a command-line tool for automatically generating and serving your browserify bundles as you develop. It's designed to be useful for quick prototyping. Each time you save your JavaScript file budo will regenerate the bundle.js file and refresh the browser automatically.

Live Reload Development Environment

Install budo:

```
npm install -g budo
```

Run this:

```
budo index.js:bundle.js --live
```

The `--live` option enables the live reload functionality of beefy.

The `index.js:bundle.js` is telling budo to read the contents of index.js, bundle the code, and serve that bundle to the browser as bundle.js.

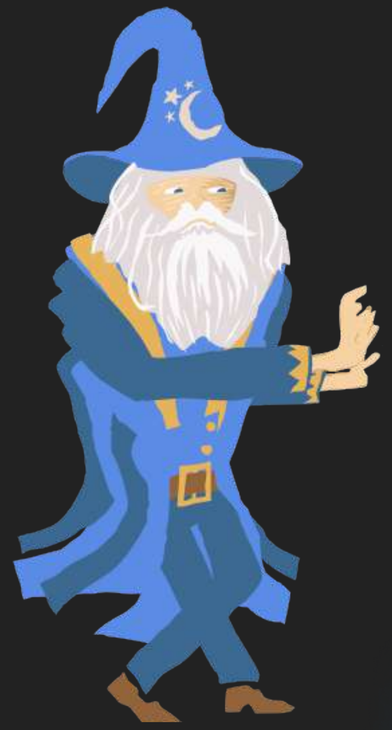
This will by default serve your index.html file at `http://localhost:9966`.

Browserify Transforms

WHEN BROWSERIFY BUNDLES OUR CODE, WE HAVE
THE OPTION TO TRANSFORM THE CODE AS WELL.

This is useful for a lot of purposes:
using `babel`, `reading files on the filesystem`, `bundling css`,
and more.

Browserify Transforms



SOME TRANSFORMATIONS

- `babelify` - turn `ES6+` code into readable vanilla `ES5` with source maps
- `cssify` - require `.css` files to add styles to the page
- `installify` - automatically installs your missing dependencies for you.

```
browserify yoursrcjs.js -o bundle.js -t babelify
```

Pros and Cons

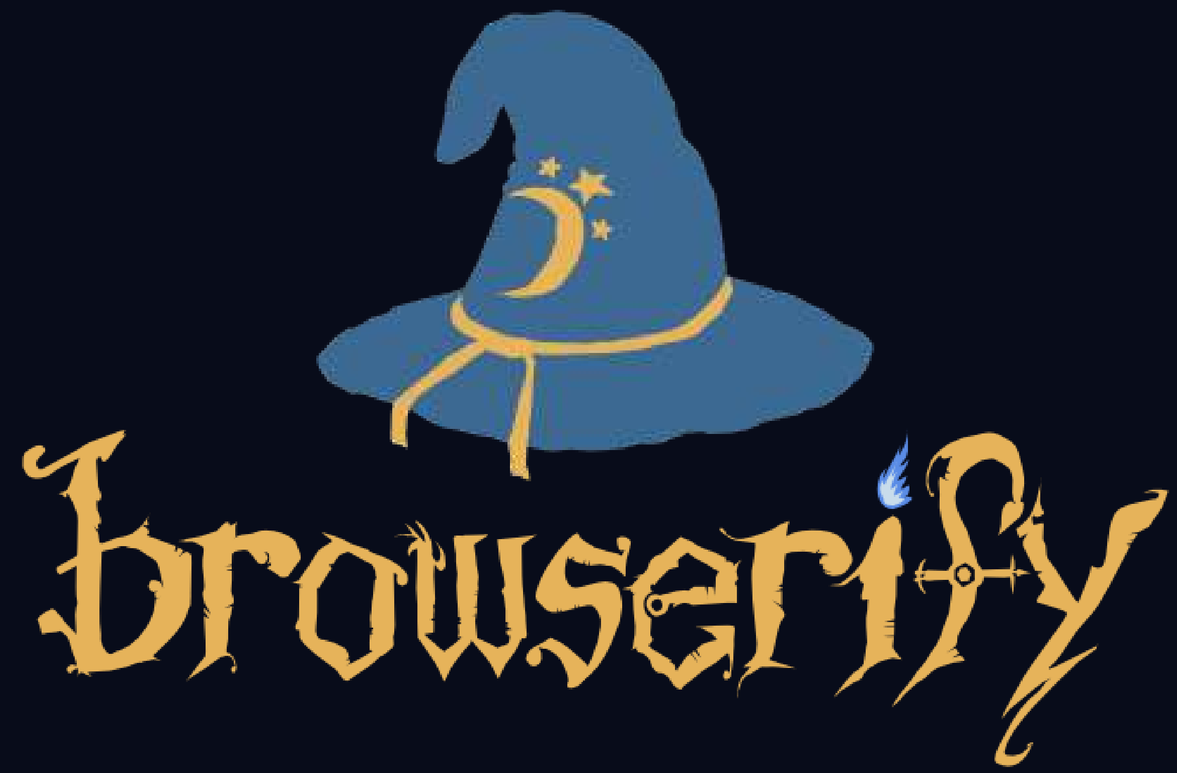
Browserify Pros and Cons

Pros

- Simple to set up and lightweight
- Use npm modules in both server and client

Cons

- No asynchronous require
- Relies on CommonJS packages
- Giant JS file / bundles with repeated code




```
browserify // thanks for watching
```