## Explanation of Build Files

| | UMD | CommonJS | ES Module |
|---|---|---|---|
| **Full** | vue.js | vue.common.js | vue.esm.js |
| **Runtime-only** | vue.runtime.js | vue.runtime.common.js | vue.runtime.esm.js |
| **Full (production)** | vue.min.js | | |
| **Runtime-only (production)** | vue.runtime.min.js | | |

### Terms

- **Full**: builds that contain both the compiler and the runtime.
- **Compiler**: code that is responsible for compiling template strings into JavaScript render functions.
- **Runtime**: code that is responsible for creating Vue instances, rendering and patching virtual DOM, etc. Basically everything minus the compiler.
- **UMD**: UMD builds can be used directly in the browser via a `<script>` tag. The default file from Unpkg CDN at https://unpkg.com/vue is the Runtime + Compiler UMD build ( `vue.js` ).
- **CommonJS**: CommonJS builds are intended for use with older bundlers like browserify or webpack 1. The default file for these bundlers ( `pkg.main` ) is the Runtime only CommonJS build ( `vue.runtime.common.js` ).
- **ES Module**: ES module builds are intended for use with modern bundlers like webpack 2 or rollup. The default file for these bundlers ( `pkg.module` ) is the Runtime only ES Module build ( `vue.runtime.esm.js` ).

### Runtime + Compiler vs. Runtime-only

If you need to compile templates on the fly (e.g. passing a string to the `template` option, or mounting to an element using its in-DOM HTML as the template), you will need the compiler and thus the full build.

When using `vue-loader` or `vueify` , templates inside `*.vue` files are compiled into JavaScript at build time. You don't really need the compiler in the final bundle, and can therefore, use the runtime-only build.

Since the runtime-only builds are roughly 30% lighter-weight than their full-build counterparts, you should use it whenever you can. If you wish to use the full build instead, you need to configure an alias in your bundler.

#### Webpack

```
module.exports = {
  // ...
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js' // 'vue/dist/vue.common.js' for webpack 1
    }
  }
}
```

**Rollup**

```
const alias = require('rollup-plugin-alias')

rollup({
  // ...
  plugins: [
    alias({
      'vue': 'vue/dist/vue.esm.js'
    })
  ]
})
```

**Browserify**

Add to your project's `package.json`:

```
{
  // ...
  "browser": {
    "vue": "vue/dist/vue.common.js"
  }
}
```

## Development vs. Production Mode

Development/production modes are hard-coded for the UMD builds: the un-minified files are for development, and the minified files are for production.

CommonJS and ES Module builds are intended for bundlers, therefore we don't provide minified versions for them. You will be responsible for minifying the final bundle yourself.

CommonJS and ES Module builds also preserve raw checks for `process.env.NODE_ENV` to determine the mode they should run in. You should use appropriate bundler configurations to replace these environment variables in order to control which mode Vue will run in. Replacing `process.env.NODE_ENV` with string literals also allows minifiers like UglifyJS to completely drop the development-only code blocks, reducing final file size.

**Webpack**

Use Webpack's DefinePlugin:

```
var webpack = require('webpack')

module.exports = {
  // ...
  plugins: [
    // ...
    new webpack.DefinePlugin({
      'process.env.NODE_ENV': JSON.stringify('production')
    })
```

```
  ]
}
```

**Rollup**

Use rollup-plugin-replace:

```
const replace = require('rollup-plugin-replace')

rollup({
  // ...
  plugins: [
    replace({
      'process.env.NODE_ENV': JSON.stringify('production')
    })
  ]
}).then(...)
```

**Browserify**

Apply a global envify transform to your bundle.

```
NODE_ENV=production browserify -g envify -e main.js | uglifyjs -c -m > build.js
```