Dll scope hoisting

DllPlugin documentation

This example demonstrates the usage of `entryOnly` option in combination with module concatenation / scope hoisting.

By default, `DllPlugin` exposes all the modules referenced in the bundle as separate entries. The manifest includes the individual modules available for use by `DllReferencePlugin`. Since all the modules are being accounted for, this prevents advanced optimizations such as tree shaking.

The `entryOnly` flag tells `DllPlugin` to only expose the modules which are configured as entry points; this affects both the manifest and the resulting bundle. Since some of the modules are no longer included in the "public contract" of the Dll, they can be optimized by merging (concatenating) multiple modules together or removing unused code. This allows taking advantage of tree shaking (scope hoisting and dead code removal) optimizations.

In this example, only `example.js` module is exposed, since it's the entry point. Modules `a.js` and `b.js` are concatenated into `example.js`. Module `cjs.js` is left as is since it's in CommonJS format.

The manifest includes `example.js` as the only exposed module and lists the exports as `["a","b","c"]` from the corresponding modules `a.js`, `b.js`, and `cjs.js`. None of the other modules are exposed.

Also, see tree shaking and scope hoisting example.

example.js

```
export { a, b } from "./a";
export { c } from "./cjs";
```

webpack.config.js

```
var path = require("path");
var webpack = require("../../");

module.exports = {
	// mode: "development" || "production",
	entry: {
		dll: ["./example"]
	},
	output: {
		path: path.join(__dirname, "dist"),
		filename: "[name].js",
		library: "[name]_[fullhash]"
	},
	optimization: {
		concatenateModules: true // this is enabled by default in production mode
	},
	plugins: [
        new webpack.DllPlugin({
            path: path.join(__dirname, "dist", "[name]-manifest.json"),
            name: "[name]_[fullhash]",
            entryOnly: true
        })
    ]
};
```

dist/dll.js

```
var dll_c76e18f6e067cdcb6208;
/******/ (() => { // webpackBootstrap
/******/ 	var __webpack_modules__ = ([
/* 0 */
/*!***************!*\
  !*** dll dll ***!
  \***************/
/*! unknown exports (runtime-defined) */
/*! runtime requirements: __webpack_require__, module */
/*! ModuleConcatenation bailout: Module Concatenation is not implemented for
DllModule */
/***/ ((module, __unused_webpack_exports, __webpack_require__) => {

module.exports = __webpack_require__;

/***/ }),
/* 1 */
/*!********************************!*\
  !*** ./example.js + 2 modules ***!
  \********************************/
/*! namespace exports */
/*! export a [provided] [no usage info] [missing usage info prevents renaming] ->
./a.js .a */
/*! export b [provided] [no usage info] [missing usage info prevents renaming] ->
./b.js .b */
/*! export c [provided] [no usage info] [missing usage info prevents renaming] ->
./cjs.js .c */
/*! other exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_require__.r, __webpack_exports__,
__webpack_require__.d, __webpack_require__, __webpack_require__.* */
/*! ModuleConcatenation bailout: Cannot concat with ./cjs.js: Module is not an
ECMAScript module */
/***/ ((__unused_webpack_module, __webpack_exports__, __webpack_require__) => {

"use strict";
// ESM COMPAT FLAG
__webpack_require__.r(__webpack_exports__);

// EXPORTS
__webpack_require__.d(__webpack_exports__, {
  "a": () => (/* reexport */ a),
  "b": () => (/* reexport */ b),
  "c": () => (/* reexport */ cjs.c)
});

;// CONCATENATED MODULE: ./b.js
// module b
function b() {
    return "b";
}

;// CONCATENATED MODULE: ./a.js
// module a
var a = "a";


// EXTERNAL MODULE: ./cjs.js
var cjs = __webpack_require__(2);
;// CONCATENATED MODULE: ./example.js




/***/ }),
/* 2 */
/*!****************!*\
  !*** ./cjs.js ***!
  \****************/
/*! default exports */
/*! export c [provided] [no usage info] [missing usage info prevents renaming] */
/*! other exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_exports__ */
/*! ModuleConcatenation bailout: Module is not an ECMAScript module */
/***/ ((__unused_webpack_module, exports) => {

// module cjs (commonjs)
exports.c = "c";


/***/ })
/******/ 	]);
```

▶ /* webpack runtime code */

```
/******/
/******/ 	// startup
/******/ 	// Load entry module and return exports
/******/ 	// This entry module doesn't tell about it's top-level declarations
so it can't be inlined
/******/ 	var __webpack_exports__ = __webpack_require__(0);
```

```
/******/ 		dll_c76e18f6e067cdcb6208 = __webpack_exports__;
/******/ 	
/******/ })()
;
```

dist/dll-manifest.json

```
{"name":"dll_c76e18f6e067cdcb6208","content":{"./example.js":{"id":1,"buildMeta":
{"exportsType":"namespace"},"exports":["a","b","c"]}}}
```

Info

Unoptimized

```
asset dll.js 4.71 KiB [emitted] (name: dll)
chunk (runtime: dll) dll.js (dll) 211 bytes (javascript) 670 bytes (runtime) [entry]
[rendered]
  > dll
  runtime modules 670 bytes 3 modules
  dependent modules 199 bytes [dependent] 2 modules
  dll dll 12 bytes [built] [code generated]
    [used exports unknown]
    dll entry
    used as library export
webpack 5.51.1 compiled successfully
```

Production mode

```
asset dll.js 694 bytes [emitted] [minimized] (name: dll)
chunk (runtime: dll) dll.js (dll) 211 bytes (javascript) 670 bytes (runtime) [entry]
[rendered]
  > dll
  runtime modules 670 bytes 3 modules
  dependent modules 199 bytes [dependent] 2 modules
  dll dll 12 bytes [built] [code generated]
    dll entry
    used as library export
webpack 5.51.1 compiled successfully
```