

第三届 FEDAY(前端开发日)

UCloud

100offer

upyun

Broadview

图灵教育

GenePlan

StuQ

CSDN
不止于代码

W3C

掘金

前端早读课

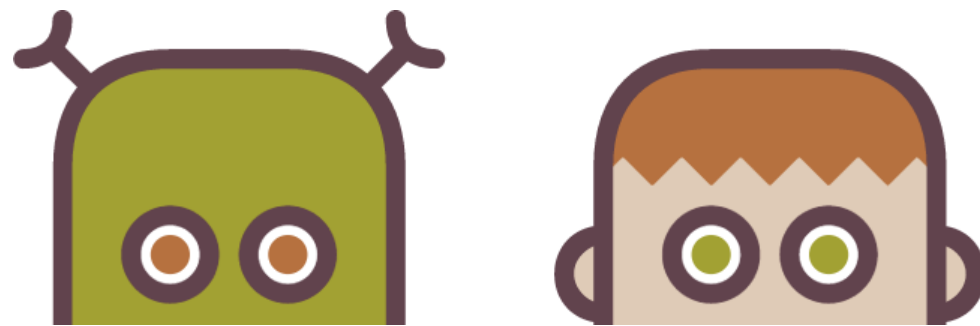
中生代技术

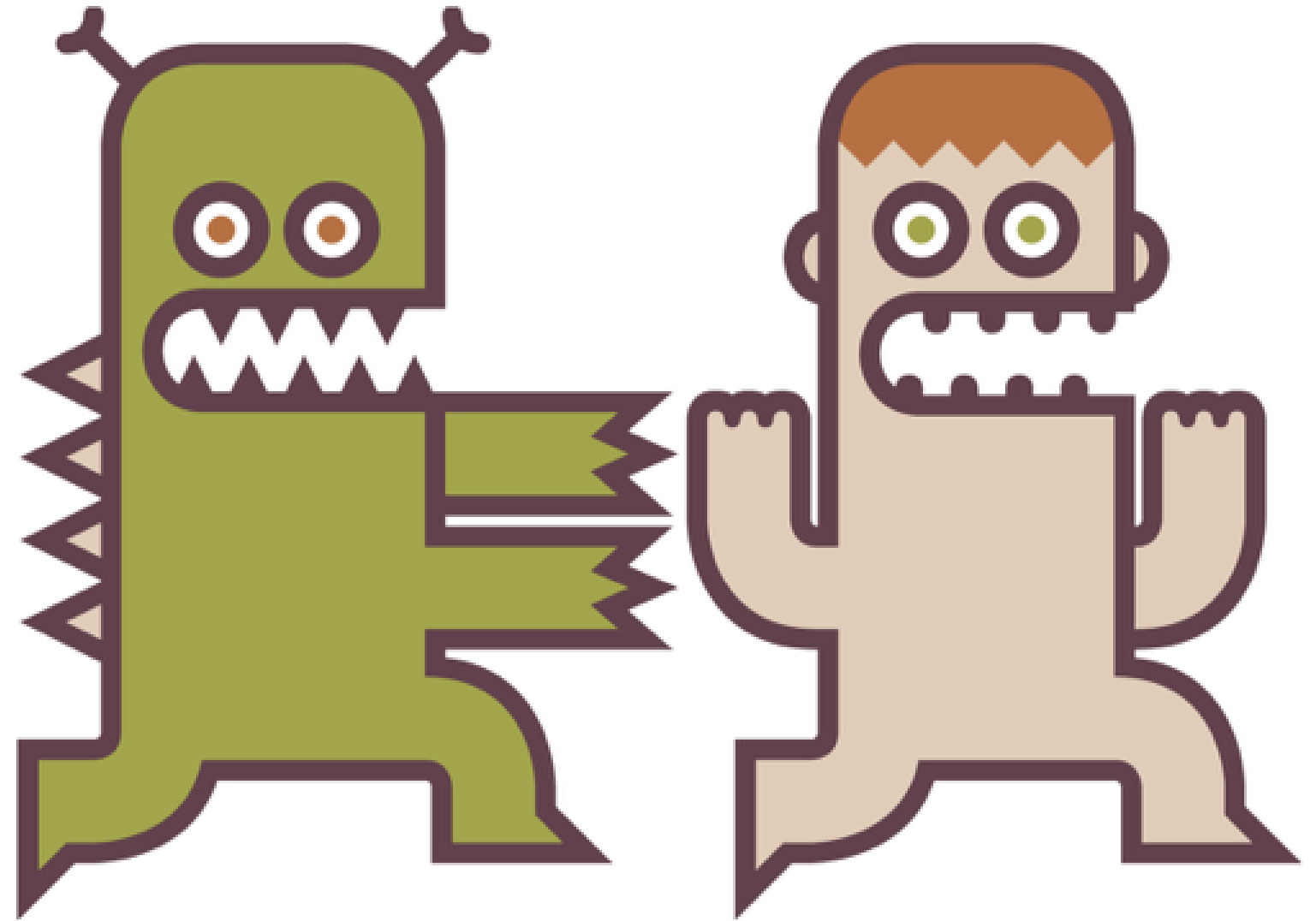
segmentfault

码云

webpack bundle inner structure and optimization

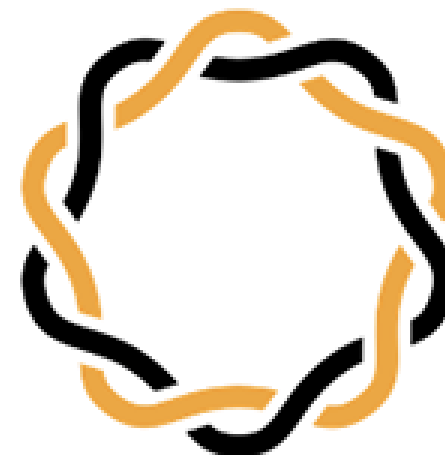
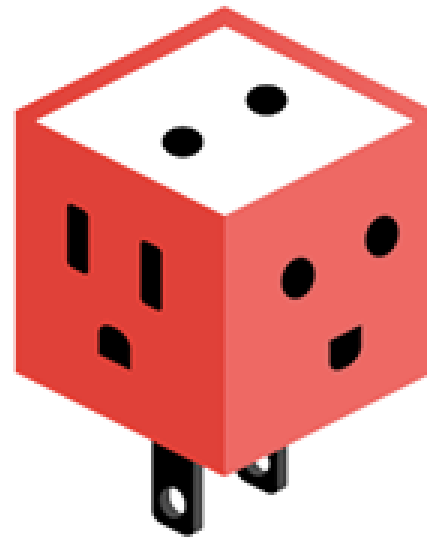
Alexey Ivanov, Evil Martians





EVIL MARTIANS

Evil Martians



Evil Martians



What I'm working on

ebay for business



Войти через eBay →

Продавайте
на eBay
по всему
миру

Увеличить продажи →

167^{млн}
покупателей¹

8+
сайтов

348^{млн}
скачиваний

57%
дохода
от международных
продаж

Treemap sizes:

Stat

Parsed

Gzipped

.f72c2e816850a7014aaa.js

node_modules

bluebird.js

mobx.js

lodash.js

common.c1fb8e4d1e77e36538e7.js

node_modules

index.js

0.7eb2a80449bbd3ee5178.js

node_modules

app

modules

app.e03fbac0ea8a85246e45.js

app

stores

router

lib



Problems

- Multiple versions of `lodash` or `underscore` .

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.
- Strange polyfills.

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.
- Strange polyfills.
- React's size in bundle is larger than in `lib/react.js`.

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.
- Strange polyfills.
- React's size in bundle is larger than in `lib/react.js`.
- Tree shaking doesn't work.

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.
- Strange polyfills.
- React's size in bundle is larger than in `lib/react.js`.
- Tree shaking doesn't work.
- Content of ALL chunks is changing on every file save, so cache doesn't work.

Problems

- Multiple versions of `lodash` or `underscore`.
- `Moment.js` loading 100+ locales.
- Strange polyfills.
- React's size in bundle is larger than in `lib/react.js`.
- Tree shaking doesn't work.
- Content of ALL chunks is changing on every file save, so cache doesn't work.
- etc.

**You bundle
probably has
them too**

Plan

— CommonJS.

Plan

- CommonJS.
- Bundle structure.

Plan

- CommonJS.
- Bundle structure.
- String replacement with DefinePlugin.

Plan

- CommonJS.
- Bundle structure.
- String replacement with DefinePlugin.
- UglifyJS and dead code elimination.

Plan

- CommonJS.
- Bundle structure.
- String replacement with DefinePlugin.
- UglifyJS and dead code elimination.
- Creating chunks and loading them via async.

Plan

- CommonJS.
- Bundle structure.
- String replacement with DefinePlugin.
- UglifyJS and dead code elimination.
- Creating chunks and loading them via async.
- ES6 modules, tree shaking and other ways of bundle optimization.

CommonJS

COMMONJS

Things to emulate

```
// module.js
```

```
// this == exports == {}
```

```
exports.a = 1;
```

```
// {a:1}
```

```
this.b = 2;
```

```
// {a:1, b:2}
```

```
module.exports = {c:3};
```

```
// {c:3}
```

```
// othermodule.js
```

```
const m = require('module');
```

```
console.log(m.a); // error
```

```
console.log(m.b); // error
```

```
console.log(m.c); // 3
```

**How can we
emulate it?**

CommonJS module in a browser

```
function(module, exports, require) {  
  
    const module = require('../path');  
  
    // ...  
  
    module.exports = ...;  
  
}
```

CommonJS module in a browser

```
function(module, exports, __webpack_require__) {  
  
    const module = __webpack_require__(0);  
  
    // ...  
  
    module.exports = ...;  
  
}
```

How does a simple bundle look like

```
(function(modules) {  
    var installedModules = {};  
  
    function __webpack_require__(moduleId) {  
  
        /* init code */  
  
    }  
  
    // ...  
  
    return __webpack_require__(1); // root file  
})([ /* modules array */ ]);
```

How does a simple bundle look like

```
(function(modules) {  
  var installedModules = {};  
  
  function __webpack_require__(moduleId) {  
  
    /* init code */  
  
  }  
  
  // ...  
  
  return __webpack_require__(1); // root file  
  
})([ /* modules array */ ]);
```

What does `__webpack_require__` do

1. Checks if the module was already initialized via `installedModules`.

WHAT DOES `__WEBPACK_REQUIRE__` DO

Why do we need `installedModules`

```
// counter.js
```

```
let c = 0;
```

```
module.exports = {  
  increment: () => c++,  
  decrement: () => c--,  
  getCount = () => c  
};
```

```
// file1.js
```

```
require('counter').increment();
```

```
// file2.js
```

```
const c = require('counter');  
console.log(c.getCount()); // 1
```

What does `__webpack_require__` do

1. Checks if the module was already initialized via `installedModules`.
2. Creates a placeholder and places it inside the `installedModules`.

What does `__webpack_require__` do

```
{  
  i: moduleId,  
  l: false,  
  exports: {}  
}
```


What does `__webpack_require__` do

1. Checks if the module was already initialized via `installedModules`.
2. Creates a placeholder and places it inside the `installedModules`.
3. Calls the module function via `this = module.exports`.

What does `__webpack_require__` do

```
modules[moduleId].call(  
  module.exports, // exports will be equal to 'this'  
  module,  
  module.exports, // equals {} by default  
  __webpack_require__  
);
```

What does `__webpack_require__` do

```
{  
  i: moduleId,  
  l: false,  
  exports: // now we have some variables here  
}
```

What does `__webpack_require__` do

1. Checks if the module was already initialized via `installedModules`.
2. Creates a placeholder and places it inside the `installedModules`.
3. Calls the module function via `this = module.exports`.
4. Returns `module.exports`.

Bundle example one more time

```
(function(modules) {  
  var installedModules = {};  
  
  function __webpack_require__(moduleId) {  
  
    /* init code */  
  
  }  
  
  // ...  
  
  return __webpack_require__(1); // root file  
  
})([ /* modules array */ ]);
```

**What can go
wrong?**

PROBLEM 1

Library paths in `require()`

We still need to emulate Node.js and NPM:

- Look for a folder with `module_name` inside `node_modules`.
- If not found, try to recursively find at parent folders.

PROBLEM 1

Multiple versions of one library

./

node_modules/

lodash-4.17.0/

some-lib-1.0.0/

node_modules/

lodash-3.0.0/

index.js

PROBLEM 1

Multiple versions of one library

```
(function(modules) {  
    // ...  
})(  
    /* lodash-4.17.0 modules */,  
    /* lodash-3.0.0 modules */,  
    /* rest of the modules */  
]);
```

PROBLEM 1

Solution

```
module.exports = {  
  resolve: {  
    alias: {  
      lodash: path.join(process.cwd(), 'node_modules/lodash'),  
    }  
  }  
};
```

**Require with
expression**

How require with expression works

```
const p = '4';  
const m = require("./module" + p);
```

Only works if we can make a RegExp from the string part of the path.

Generated context module

```
var map = {  
  " ./module ": 1, " ./module.js ": 1,  
  " ./module2 ": 0, " ./module2.js ": 0,  
}  
  
function webpackContext(filename) {  
  /* code to select module */  
};  
  
module.exports = webpackContext;
```

How require with expression works

```
const p = '4';
```

```
const m = require("./module" + p);
```

Requiring by expression in bundle:

```
const m = __webpack_require__(3)("./module" + p);
```

Bundle with all dependencies

```
(function(modules) {  
    // ...  
})(  
    /* all modules with matched filenames */,  
    /* name resolving module */,  
);
```

PROBLEM 2

moment/moment.js

```
require('./locale/' + name); // 118 locales
```

Solution:

```
new webpack.ContextReplacementPlugin(  
  /moment[\/\\]locale$/,  
  /en|cn/  
)
```


DefinePlugin

Variables in code

```
const version = VERSION;
```

```
if (process.env.NODE_ENV !== "production") {  
  const module = require("module");  
  // do something  
}
```

DefinePlugin config

```
new webpack.DefinePlugin({  
  "process.env.NODE_ENV": JSON.stringify("production"),  
  VERSION: JSON.stringify("1.0.1"),  
});
```

DefinePlugin config

```
new webpack.DefinePlugin({  
  "process.env.NODE_ENV": '"production"',  
  VERSION: '"1.0.1"',  
});
```

Transformed code

```
const version = "1.0.1";
```

```
if (false) {
```

```
    const module = require("module"); // not imported
```

```
    // do something
```

```
}
```

PROBLEM 3

Not using exact variable names

// Before replacing

```
const NODE_ENV = process.env.NODE_ENV;  
if (NODE_ENV === "production") { require("module"); }
```

// After replacing

```
const NODE_ENV = "production";  
if (NODE_ENV !== "production") { __webpack_require__(0); }
```

**What will happen if
process.env was
not replaced?**

PROBLEM 4

node.js polyfills

`process` is a standard variable in `node.js`; webpack will polyfill it for compatibility.

Many other `node.js` variables will be polyfilled too.

Libraries that use `process.env.NODE_ENV`:

- React,
- Redux,
- etc.

Dead code elimination with UglifyJS

SHORTENING VARIABLE NAMES AND REMOVING UNUSED ONES

Before

```
function text() {  
    var one = 1;  
    var two = 2;  
    var three = 3;  
    return one + two;  
}
```

SHORTENING VARIABLE NAMES AND REMOVING UNUSED ONES

After

```
function text(){var t=1,e=2  
return t+e}
```

WILL NOT WORK

Before

```
var one = 1;
```

```
var two = 2;
```

```
var three = 3;
```

```
console.log(one + two);
```

WILL NOT WORK

After

```
var one=1,two=2,three=3
```

```
console.log(one+two)
```

REMOVE CONDITION OPERATORS

Before

```
if (true) {  
    console.log(1);  
}
```

```
if (false) {  
    console.log(2);  
}
```

REMOVE CONDITION OPERATORS

After

```
console.log(1);
```

ALSO WILL NOT WORK

Before

```
var f = false;
```

```
if (f) {  
    console.log(1);  
}
```


ALSO WILL NOT WORK

After

```
var f=!1
```

```
f&&console.log(1)
```

PROBLEM 5

Variables in conditional operators

```
const NODE_ENV = "production";  
if (NODE_ENV !== "production") {  
  __webpack_require__(0);  
}
```

Chunks

Synchronous chunks

Just two or more JavaScript files in the html:

```
<head>
```

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

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

```
</head>
```

Updated bundle code in main.js

```
(function(modules) {  
    window["webpackJsonp"] = function() { /* ... */ }  
    var installedModules = {};  
    function __webpack_require__(moduleId) { /* init code */ }  
    return __webpack_require__(1); // root file  
})([ /* modules array */ ]);
```

chunk.js code

```
webpackJsonp(  
  [0], // Mark chunks as loaded  
  [{ 22: function(...){...} }], // Load modules to array  
  [12] // Run modules after loading  
]);
```

chunk.js code

```
webpackJsonp(  
  [0], // Mark chunks as loaded  
  [{ 22: function(...){...} }], // Load modules to array  
  [12] // Run modules after loading  
]);
```

chunk.js code

```
webpackJsonp(  
  [0], // Mark chunks as loaded  
  [{ 22: function(...){...} }], // Load modules to array  
  [12] // Run modules after loading  
]);
```


chunk.js code

```
webpackJsonp(  
  [0], // Mark chunks as loaded  
  [{ 22: function(...){...} }], // Load modules to array  
  [12] // Run modules after loading  
]);
```

Asynchronous chunks

```
import('./module').then(module) => {  
  console.log(module);  
});
```

Async chunk loading

```
__webpack_require__.(0)
  .then(__webpack_require__.bind(null, 1))
  .then((module) => {
    console.log(module);
  });
```

Inside webpack_require.e()

Putting together chunk name with [chunkhash] :

```
__webpack_require__.e(chunkId) {  
  script.src = chunkId + "." + {  
    "0": "588290cc688bace070b6",  
    "1": "5966f9b147ab01778e34",  
  }[chunkId] + ".js";  
  // ...  
}
```

CommonsChunk Plugin

Moving common modules to a separate file

```
new webpack.optimize.CommonsChunkPlugin({  
  name: 'common',  
  filename: '[name].[chunkhash].js',  
  minChunks: 2  
})
```

PROBLEM 6

Nested chunks

`import()` -chunks will be ignored by `CommonsChunkPlugin`.

To include them:

```
new webpack.optimize.CommonsChunkPlugin({  
    . . . ,  
    children: true  
})
```

Moving node_modules content to a separate file

```
new webpack.optimize.CommonsChunkPlugin({  
  name: "vendor",  
  minChunks: function (module) {  
    return module.context &&  
      module.context.indexOf("node_modules") !== -1;  
  }  
})
```


PROBLEM 7

Changing indexes

node_modules example will not work for cache:

1. When you add new files, indexes will change;
2. Initialization code lives in the first file:
 - start file index can change,
 - links to chunk files can change.

PROBLEM 7

Changing indexes

```
(function(modules) {  
    var installedModules = {};  
  
    function __webpack_require__(moduleId) {  
  
        /* init code */  
  
    }  
  
    // ...  
  
    return __webpack_require__(1); // root module  
  
})([ /* modules array */ ]);
```

PROBLEM 7

Changing indexes

Putting together chunk name with `[chunkhash]` :

```
script.src = chunkId + "." + {  
  "0": "588290cc688bace070b6",  
  "1": "5966f9b147ab01778e34",  
}[chunkId] + ".js";
```

Fix module names

- `NamedModulesPlugin()` — replaces indexes with file names.
- `HashedModuleIdsPlugin()` — replaces indexes with content hashes.

Move initialization code to a separate file

```
new webpack.optimize.CommonsChunkPlugin({  
  name: "manifest",  
  minChunks: Infinity  
}),
```

Multiple CommonsChunkPlugin

```
new webpack.optimize.CommonsChunkPlugin({  
  name: 'vendor', ...  
}),  
  
new webpack.optimize.CommonsChunkPlugin({  
  name: 'manifest', ...  
})
```

ES6 modules

Differences from CommonJS

// CommonJS

```
const m = require('./m');
```

```
exports.one = 1;
```

```
module.exports = 2;
```

// ES Modules

```
import m from './m';
```

```
export const one = 1;
```

```
export default 2;
```

Import and exports are always immutable.

Why do we need it?

Bundle size with different options

```
import { chunk } from 'lodash-es';  
console.log(chunk([1,2,3,4], 2));
```

Tree Shaking	Exports Mangling	Scope Hoisting	Pure Module	Minimized	Gzipped
no	no	no	no	223.271	40.568
yes	yes	no	no	139.317	33.595
yes	yes	yes	no	90.235	31.023
yes	yes	no	yes	4.164	1.479
yes	yes	yes	yes	3.137	1.474

<https://twitter.com/jdalton/status/893109185264500737>

Tree shaking

TREE SHAKING

Import and export example

```
// module.js
```

```
export const one = 1;
```

```
export const two = 2;
```

```
// index.js
```

```
import { one, two } from './module';
```

```
console.log(one);
```

TREE SHAKING

Side effects

```
export const method = (() => {  
  window.foo == foo == 'foo';  
  
  return foo;  
})();
```

Import and export example

```
// module.js
```

```
export const one = 1;
```

```
export const two = 2;
```

```
// index.js
```

```
import { one, two } from './module';
```

```
console.log(one);
```

index.js in bundle

```
var __WEBPACK_IMPORTED_MODULE_0__module__  
    = __webpack_require__(0);
```

```
console.log(__WEBPACK_IMPORTED_MODULE_0__module__["a"]);
```

module.js in bundle

```
const one = 1;
```

```
__webpack_exports__["a"] = one;
```

```
const two = 2;
```

```
// no __webpack_exports__ because it will not be imported
```


PROBLEM 8

According to the spec, all children must be loaded and evaluated

```
import module3 from './folder/module-3';  
// import from 'method' will be added to bundle  
export const method = () => module3;  
// export.default will be marked as used  
export default 1223;
```

Bundle size with different options

```
import { chunk } from 'lodash-es';  
console.log(chunk([1,2,3,4], 2));
```

Tree Shaking	Exports Mangling	Scope Hoisting	Pure Module	Minimized	Gzipped
no	no	no	no	223.271	40.568
yes	yes	no	no	139.317	33.595
yes	yes	yes	no	90.235	31.023
yes	yes	no	yes	4.164	1.479
yes	yes	yes	yes	3.137	1.474

<https://twitter.com/jdalton/status/893109185264500737>

Scope hoisting

SCOPE HOISTING

ModuleConcatenationPlugin

```
// a.js
```

```
export const a = 123;
```

```
// b.js
```

```
import { a } from 'a.js';
```

BEFORE

ModuleConcatenationPlugin

```
[(function(/* ... */) {  
    const a = 123;  
    __webpack_exports__["a"] = a;  
}),  
(function(/* ... */) {  
    var __module__ = __webpack_require__(1);  
    console.log(__module__["a"]);  
})]
```

AFTER

ModuleConcatenationPlugin

```
[(function(/* ... */) {  
    const a = 123;  
    console.log(a);  
})]
```

Bundle size with different options

```
import { chunk } from 'lodash-es';  
console.log(chunk([1,2,3,4], 2));
```

Tree Shaking	Exports Mangling	Scope Hoisting	Pure Module	Minimized	Gzipped
no	no	no	no	223.271	40.568
yes	yes	no	no	139.317	33.595
yes	yes	yes	no	90.235	31.023
yes	yes	no	yes	4.164	1.479
yes	yes	yes	yes	3.137	1.474

<https://twitter.com/jdalton/status/893109185264500737>

Pure module

Pure module

Experimental feature. Not in master brunch.

```
// package.json  
  
{  
  "pure-module": true  
}
```

Pure module

```
// big-module/index.js
```

```
export { a } from './a';
```

```
export { b } from './b';
```

```
export { c } from './c';
```

```
// file.js
```

```
import { a } from 'big-module';
```

```
console.log(a);
```

Without pure module

```
(function(modules) {  
    // ...  
})([  
    /* pure-module/index.js */,  
    /* pure-module/a.js */,  
    /* pure-module/b.js */,  
    /* pure-module/c.js */,  
]);
```

With pure module

```
(function(modules) {  
    // ...  
})(  
    /* pure-module/a.js */,  
];
```

Bundle size with different options

```
import { chunk } from 'lodash-es';  
console.log(chunk([1,2,3,4], 2));
```

Tree Shaking	Exports Mangling	Scope Hoisting	Pure Module	Minimized	Gzipped
no	no	no	no	223.271	40.568
yes	yes	no	no	139.317	33.595
yes	yes	yes	no	90.235	31.023
yes	yes	no	yes	4.164	1.479
yes	yes	yes	yes	3.137	1.474

<https://twitter.com/jdalton/status/893109185264500737>

Bundle analysis

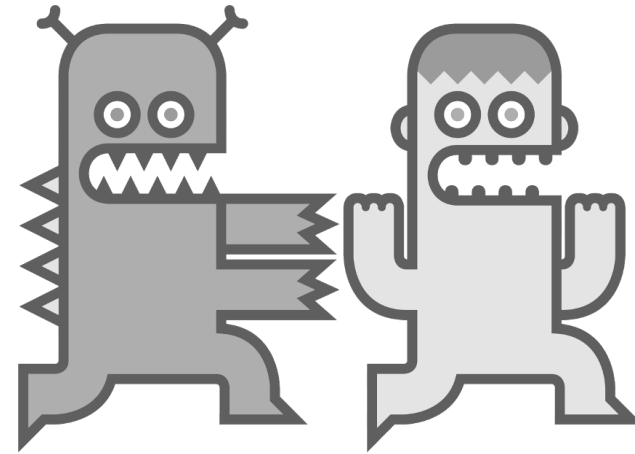
Bundle analysis

[webpack-bundle-analyzer](#) — useful for finding two versions of the library, copies of the library in different chunks, libraries that should be removed by `if` condition, unexpected dependencies, large files.

[webpack-runtime-analyzer](#) — useful for finding which file imported a specific file and why a specific library was imported.

Summary

- Create an empty bundle file and study its content — it's only 40 lines.
- Don't be afraid to look inside bundle and look at the resulted code.
- After you add a new library, run bundle analyzer and look at what was added with it.
- After you create a chunk, run bundle analyzer and look at its content.



webpack bundle inner structure and optimization

Alexey Ivanov, Evil Martians

[@iadramelk](#) [evl.ms](#) [@evilmartians](#)