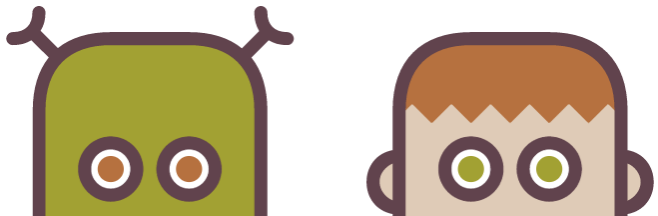
Внутреннее устройство и оптимизация бандла webpack

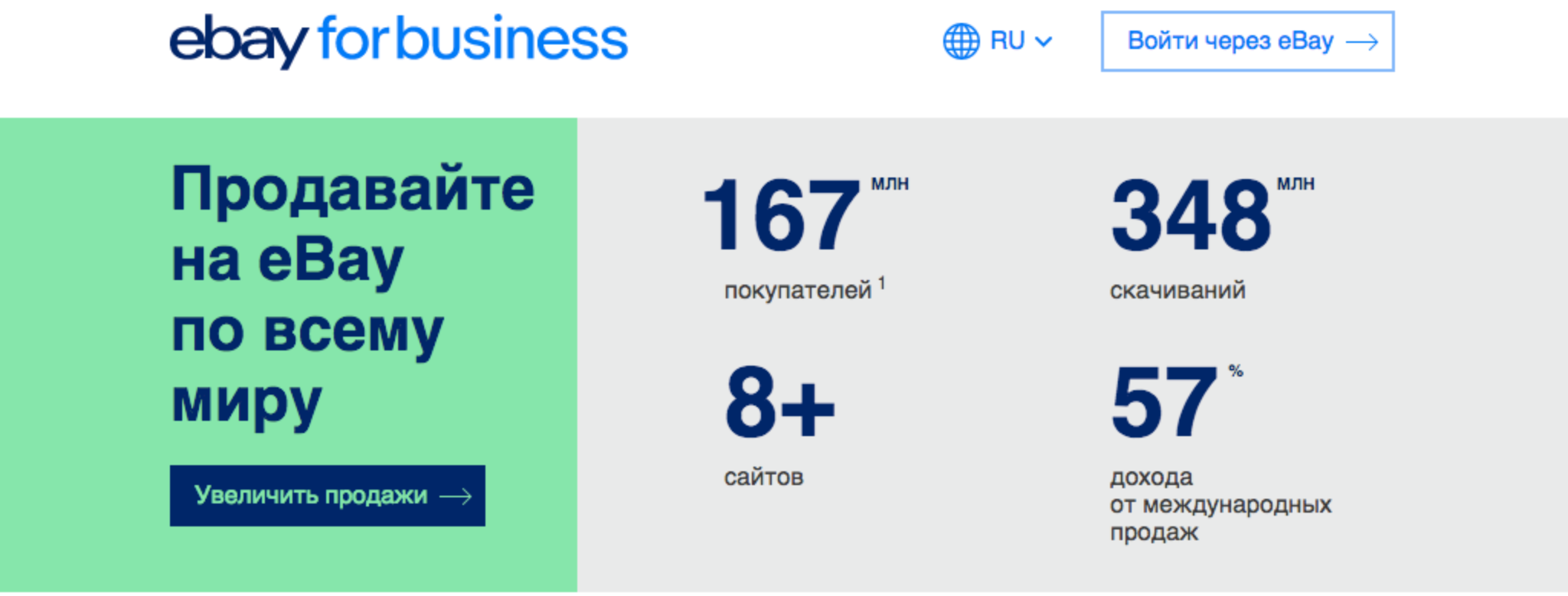
Алексей Иванов, Злые марсиане

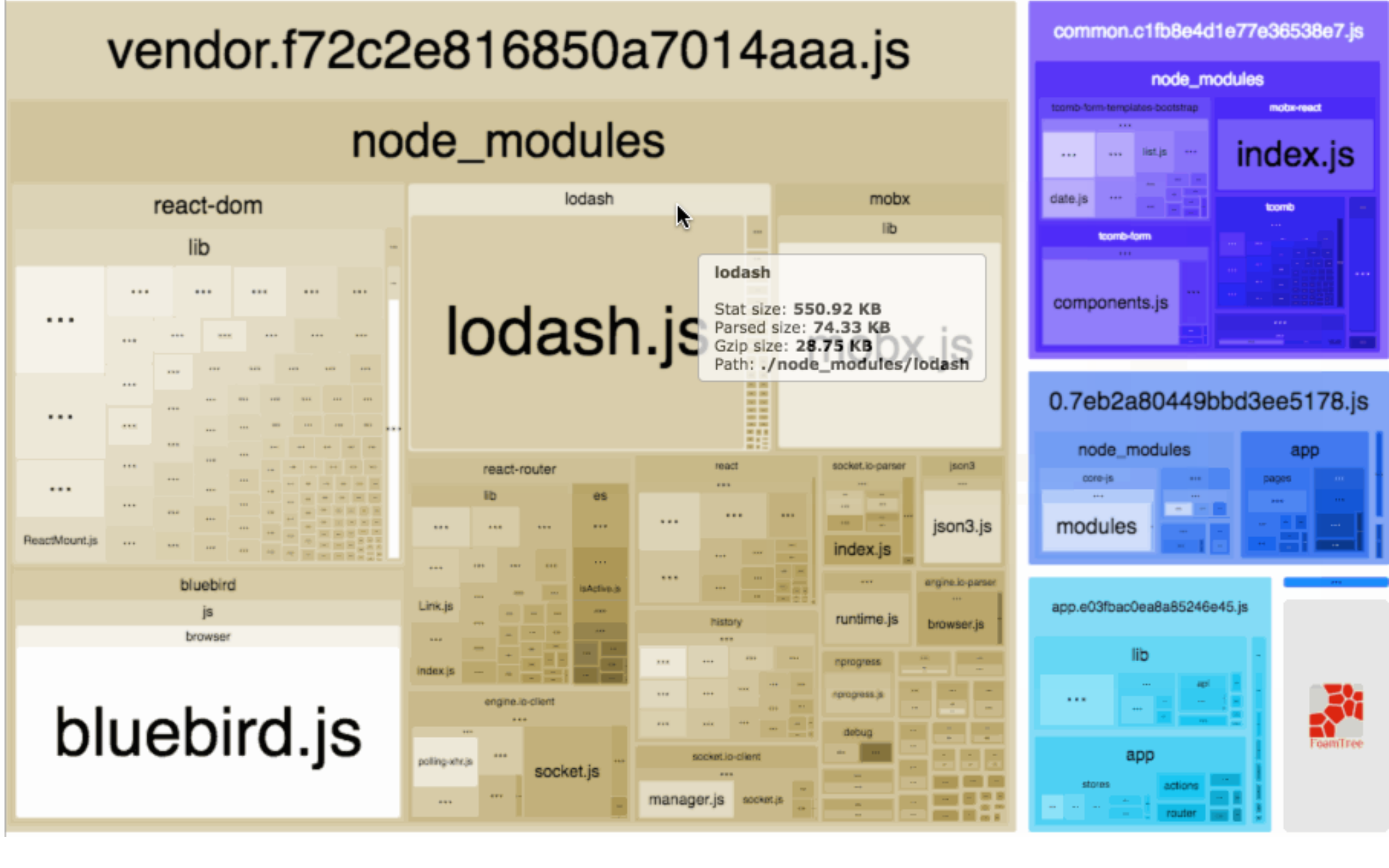


Злые марсиане

Злые марсиане

Над чем я работаю




vendor.f72c2e816850a7014aaa.js
node_modules
react-dom
lib
ReactMount.js
lodash
lodash.js
lodash
Stat size: 550.92 KB
Parsed size: 74.33 KB
Gzip size: 28.75 KB
Path: ./node_modules/lodash
mobx
lib
react-router
lib
es
Link.js
index.js
isActive.js
react
socket.io-parser
index.js
json3
json3.js
bluebird
js
browser
bluebird.js
runtime.js
engine.io-parser
browser.js
history
nprogress
nprogress.js
debug
engine.io-client
polling-xhr.js
socket.js
socket.io-client
manager.js
socket.js
common.c1fb8e4d1e77e36538e7.js
node_modules
tcomb-form-templates-bootstrap
list.js
date.js
mobx-react
index.js
tcomb
tcomb-form
components.js
0.7eb2a80449bbd3ee5178.js
node_modules
core-js
modules
app
pages
app.e03fbac0ea8a85246e45.js
lib
app
stores
actions
router
FoamTree

Проблемы

- React в бандле весит больше чем `lib/react.js`.
- Несколько версий `lodash` или `underscore`.
- `Moment.js` грузит 100+ локалей.
- Непонятные полифилы.
- Не работает tree shaking.
- Изменяется кеш для неизменившихся чанков.
- И так далее.

Для кого этот доклад

План

— CommonJS.

— Резолв путей до файлов.

— Устройство бандла изнутри.

— Глобальные константы и DefinePlugin.

— UglifyJS и dead code elimination.

— ES6 modules и tree shaking.

— Выделение чанков и асинхронная подгрузка.

— Анализ результатов сборки.

CommonJS

CommonJS

```
// module.js

const one = 1;

exports.two = 2;
module.exports = { one };
```

```
// othermodule.js

const m = require('module');

console.log(m.one);
console.log(m.two); // ошибка
```

CommonJS

- JS-файл с переменными `require`, `exports`, `module`.
- `this` равно `exports`.
- `exports` по умолчанию равно `{}`.
- Чтобы мы могли использовать npm-модули в браузере, нам нужно эмулировать в браузере это поведение.

Резолв путей

Резолв путей в `require()`

- `/`, `./`, `../` – как в файловой системе.
- Библиотеки:
  - папка с именем модуля в `node_modules`,
  - если нет в `node_modules`, ищем рекурсивно выше по дереву,
- Добавляем в конец пути `.js` или `/index.js`.

Схема резолва путей в node.js

Схема резолва путей в webpack 2

ПРОБЛЕМА 1

Несколько версий библиотек

```
./
  node_modules/
    lodash-5.0.0/
    some-lib-1.0.0/
      node_modules/
        lodash-1.0.0/
  index.js
```

Базовое устройство бандла

CommonJS модуль в браузере

```
function(module, exports, require) {

  const module = require('../path');
  // ...
  module.exports = ...;

}
```

CommonJS модуль в браузере

```
function(module, exports, __webpack_require__) {

  const module = __webpack_require__(Ø);
  // ...
  module.exports = ...;

}
```

Как выглядит простой бандл

```
(function(modules) {
  var installedModules = {};
  function __webpack_require__(moduleId) {
    /* код инициализации */
  }
  // ...
  return __webpack_require__(1); // корневой файл
})([ /* массив модулей */ ]);
```

Как выглядит простой бандл

```
(function(modules) {
  var installedModules = {};
  function __webpack_require__(moduleId) {
    /* код инициализации */
  }
  // ...
  return __webpack_require__(1); // корневой файл
})([ /* массив модулей */ ]);
```

Что делает __webpack_require__

1. Ищет инициированный модуль в кэше.
2. Создает заглушку и добавляет её в массив.

Что делает __webpack_require__

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

Что делает __webpack_require__

1. Ищет инициированный модуль в кэше.
2. Создает заглушку и добавляет её в массив.
3. Выполняет код модуля с `this` равным `module.exports`.

Что делает __webpack_require__

```
modules[moduleId].call(
   module.exports, // exports будет this для модуля
   module,
   module.exports, // по-умолчанию равно {}
   __webpack_require__
);
```

Что делает __webpack_require__

```
{
  i: moduleId,
  l: false,
  exports: // тут теперь какой-то код
}
```

Что делает __webpack_require__

1. Ищет инициированный модуль в кэше.
2. Создает заглушку и добавляет её в массив.
3. Выполняет код модуля с `this` равным `module.exports`.
4. Возвращает `module.exports`.

Как работает подключение модуля по маске

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

Require в бандле:

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

Модуль резолва пути в бандле

```
var map = {
  "./module": 1, "./module.js": 1,
  "./module2": Ø, "./module2.js": Ø,
}
function webpackContext(...) {/* код выбора модуля */};
module.exports = webpackContext;
```

Справка по Dependency management

ПРОБЛЕМА 2

moment/moment.js

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

Решение:

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

Глобальные константы и DefinePlugin

Константы в коде

```
const version = VERSION;

if (process.env.NODE_ENV !== "production") {
  const module = require('module');
  // что-то делаем
}
```

DefinePlugin

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

Преобразованный код

```
const version = "1.0.1";


if (false) {
  const module = require('module'); // не импортирован
  // что-то делаем
}
```

ПРОБЛЕМА 3

Неточная строка для замены

```
// До преобразования
const { NODE_ENV } = process.env;

// После преобразования
var _process$env = process.env,
    NODE_ENV = _process$env.NODE_ENV;
```

Ссылка на песочницу с Babel

ПРОБЛЕМА 3

Неточная строка для замены

```
var _process$env = process.env,
    NODE_ENV = "production";

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

Что будет, если не заменить process.env?

ПРОБЛЕМА 4

Полифилы node.js

`process` – стандартная переменная в node.js, webpack её заполифилит для совместимости.

Многие другие переменные из `node.js` тоже будут заменены.

ПРОБЛЕМА 5

Функции отладки в библиотеках

- React
- Redux
- ...

Сжатие кода

UglifyPlugin

— Удаляет пробелы.

— Переименовывает переменные короткими именами.

— Делает dead code elimination.

Ссылка на песочницу с UglifyJS2

СОКРАЩЕНИЕ ИМЁН И УДАЛЕНИЕ ПЕРЕМЕННЫХ

Было

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

СОКРАЩЕНИЕ ИМЁН И УДАЛЕНИЕ ПЕРЕМЕННЫХ

Стало

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

А ВОТ ТАК НЕ СРАБОТАЕТ

Было

```
var one = 1;

var two = 2;

var three = 3;


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

А ВОТ ТАК НЕ СРАБОТАЕТ

Стало

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

console.log(one+two)
```

УДАЛЕНИЕ УСЛОВИЙ

Было

```
if (true) {
  console.log(1);
}
if (false) {
  console.log(2);
}
```

УДАЛЕНИЕ УСЛОВИЙ

Стало

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

ТАК ТОЖЕ НЕ СРАБОТАЕТ

Было

```
var f = false;


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

ТАК ТОЖЕ НЕ СРАБОТАЕТ

Стало

```
var f=!1

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

ПРОБЛЕМА 6

Переменные в условиях

```
var _process$env = process.env,
    NODE_ENV = "production";

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

ES6 modules

Отличия от 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;
```

Импорты и экспорты обязательно иммутабельны.

Документация по import, export

Tree shaking

В теории webpack может точно определить, что используется у нас в приложении, и помечать их соответственно.

На практике все несколько сложнее.

TREE SHAKING

Сайд эффекты

```
export const method (() => {
   window.foo == '111';
})();
```

Пример импорта и экспорта

```
// module.js
export const one = 1;
export const two = 2;

// index.js
import { one, two } from './module';
console.log(one);
```

index.js в бандле

```
var __WEBPACK_IMPORTED_MODULE_0__module__
  = __webpack_require__(0);

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

module.js в бандле

```
const one = 1;

__webpack_exports__["a"] = one;


const two = 2;

// нет __webpack_exports__ за ненадобностью
```

ПРОБЛЕМА 7

Работает только с экспортами

```
import module3 from  './folder/module-3';
// импорт из method попадет в сборку
export const method = () => module3;
// export.default пометится как используемый
export default 1223;
```

ПРОБЛЕМА 7

Работает только с экспортами

Такой код импортирует всю библиотеку:

```
import { method } from 'lodash-es';
```

Решение:

- `import { method } from 'lodash-es/method'`
- babel-plugin-lodash

Важно!

Выключите в **Babel** обработку `modules`, чтобы он не менял их на `require` и `exports`:

```
{"presets": [
  ["latest", {
    "es2015": { "modules": false }
  }]
]}
```

Чанки

Чанки

- Синхронные и асинхронные.
- В первый файл добавляется функция `window["webpackJsonp"]`.
- В следующих файлах вызывается функция `webpackJsonp` со списком модулей и id модулей, которые надо запустить.

Чанки

```
webpackJsonp(
  [0], // id чанка
  [{ 22: function(...){...} }], // массив модулей
  [12] // id модулей для запуска
]);
```

Чанки

- Синхронные и асинхронные.
- В первый файл добавляется функция `window["webpackJsonp"]`.
- В следующих файлах вызывается функция `webpackJsonp` со списком модулей и id модулей, которые надо запустить.
- Все модули попадают в общий массив и используются оттуда.

Асинхронная подгрузка чанков

```
import('./module4').then((module4) => {
  console.log(module4);
});
```

Асинхронная подгрузка чанков

```
__webpack_require__.e(0)
  .then(__webpack_require__.bind(null, 1))
  .then((module4) => {
    console.log(module4);
  });
```

Имена файлов чанков

Создание имени чанка при наличии `[chunkhash]`:

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

CommonsChunk Plugin

Выносим общие модули в отдельный файл

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

ПРОБЛЕМА 8

Вложенные чанки

`import()`-чанки игнорируются `CommonsChunkPlugin`.

Чтобы не игнорировались, добавьте:

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

Вынос node_modules

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

ПРОБЛЕМА 9

Изменяющиеся индексы

Пример с node_modules не работает для кэша:

1. При добавлении файлов меняются индексы;
2. Код загрузки и инициализации живет в первом файле:
   — меняется стартовый индекс,
   — меняются ссылки на чанки.

ПРОБЛЕМА 9

Изменяющиеся индексы

```
(function(modules) {
  var installedModules = {};
  function __webpack_require__(moduleId) {
    /* код инициализации */
  }
  // ...
  return __webpack_require__(1); // корневой файл
})([ /* массив модулей */ ]);
```

ПРОБЛЕМА 9

Изменяющиеся индексы

Создание имени чанка при наличии `[chunkhash]`:

```
script.src = chunkId + ".” + {
  "0":"58829ØccØ88baceØ7Øb6",
  "1":"5966f9b147abØ1778e34",
}[chunkId] + ".js";
```

Фиксируем имена модулей

— `NamedModulesPlugin()` – меняет индексы на пути до файла.

— `HashedModuleIdsPlugin()` — меняет индексы на хэш содержимого.

Выносим инициализацию в отдельный файл

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

Несколько CommonsChunkPlugin

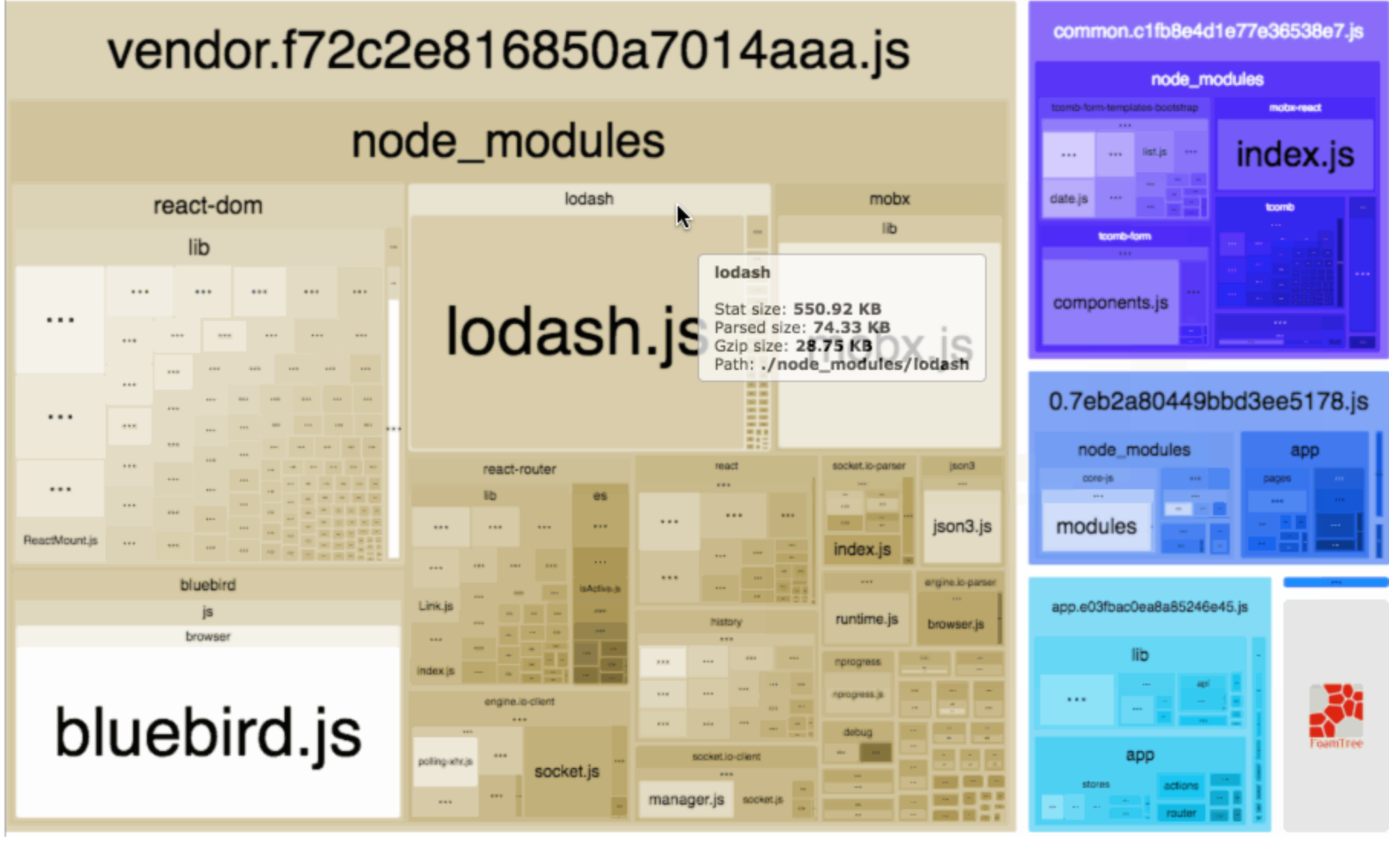
```
new webpack.optimize.CommonsChunkPlugin({
  name: 'vendor', ...
}),
new webpack.optimize.CommonsChunkPlugin({
  name: 'manifest', ...
})
```

Анализ бандла

webpack-bundle-analyzer

Строит treemap бандлов. Удобно проверять, не попали ли в бандл:

- Две версии одной библиотеки.
- Копии библиотеки в разных чанках.
- Бибиотеки, которые должны были вырезаться по условию.
- Непредвиденные зависимости у библиотек.
- Просто большие файлы.


vendor.f72c2e816850a7014aaa.js
node_modules
react-dom
lib
ReactMount.js
lodash
lodash.js
lodash
Stat size: 550.92 KB
Parsed size: 74.33 KB
Gzip size: 28.75 KB
Path: ./node_modules/lodash
mobx
lib
react-router
lib
es
Link.js
index.js
isActive.js
react
socket.io-parser
index.js
json3
json3.js
bluebird
js
browser
bluebird.js
runtime.js
engine.io-parser
browser.js
history
nprogress
nprogress.js
debug
engine.io-client
polling-xhr.js
socket.js
socket.io-client
manager.js
socket.js
common.c1fb8e4d1e77e36538e7.js
node_modules
tcomb-form-templates-bootstrap
list.js
date.js
mobx-react
index.js
tcomb
tcomb-form
components.js
0.7eb2a80449bbd3ee5178.js
node_modules
core-js
modules
app
pages
app.e03fbac0ea8a85246e45.js
lib
app
stores
actions
router
FoamTree

webpack-runtime-analyzer

Показывает отношения между файлами в графе — кто на кого ссылается, кто кого добавил в сборку. Удобно использовать, чтобы понять:

- Кто именно использует файл.
- Кто именно подключил библиотеку.

Home Errors Warnings Graph File Map Details Options

Webpack **2.2.1** / Modules: **10** / Chunks: **1**

Id	Name	Size	Occurrences	Retain	Exclusive
1	./~/css-loader!./src/main/aaa.css	185 bytes	in 2 modules	1 KB (1 module)	1 KB (1 module)
2	./~/css-loader/lib/css-base.js	1 KB	in 1 module	-	-
3	./src/main/aaa.css	905 bytes	in 3 modules	9 KB (3 modules)	7 KB (1 module)
4	./~/style-loader/addStyles.js	7 KB	in 1 module	-	-
5	../~/basisjs/src/basis.js	109 KB	in 1 module	-	-
6	./src/main/cont ^\.\/.*$	198 bytes	in 1 module	12 KB (7 modules)	3 KB (3 modules)
7	./src/main/cont/a.js	34 bytes	in 1 module	-	-
8	./src/main/cont/b.js	77 bytes	in 1 module	12 KB (5 modules)	3 KB (1 module)
9	./src/main/test.ts	3 KB	in 1 module	9 KB (4 modules)	-
	Total	**9 modules in 121 KB**			

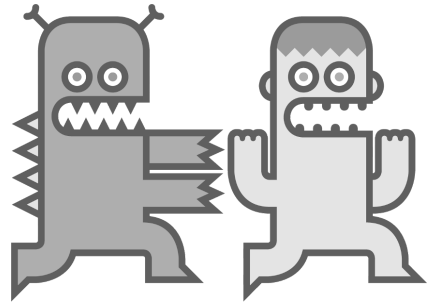
Итого

— Сделайте пустой бандл и посмотрите содержимое, там 40 строчек.

— Не бойтесь ходить в исходники и смотреть что получилось в коде.

— После добавления библиотек всегда запускайте анализатор банда и смотрите что он с собой притащил.

— После добавления чанков проверяйте их содержимое.



Внутреннее устройство и оптимизация бандла webpack

Алексей Иванов, Злые марсиане

Twitter: @iadramelk

cult of martians.com