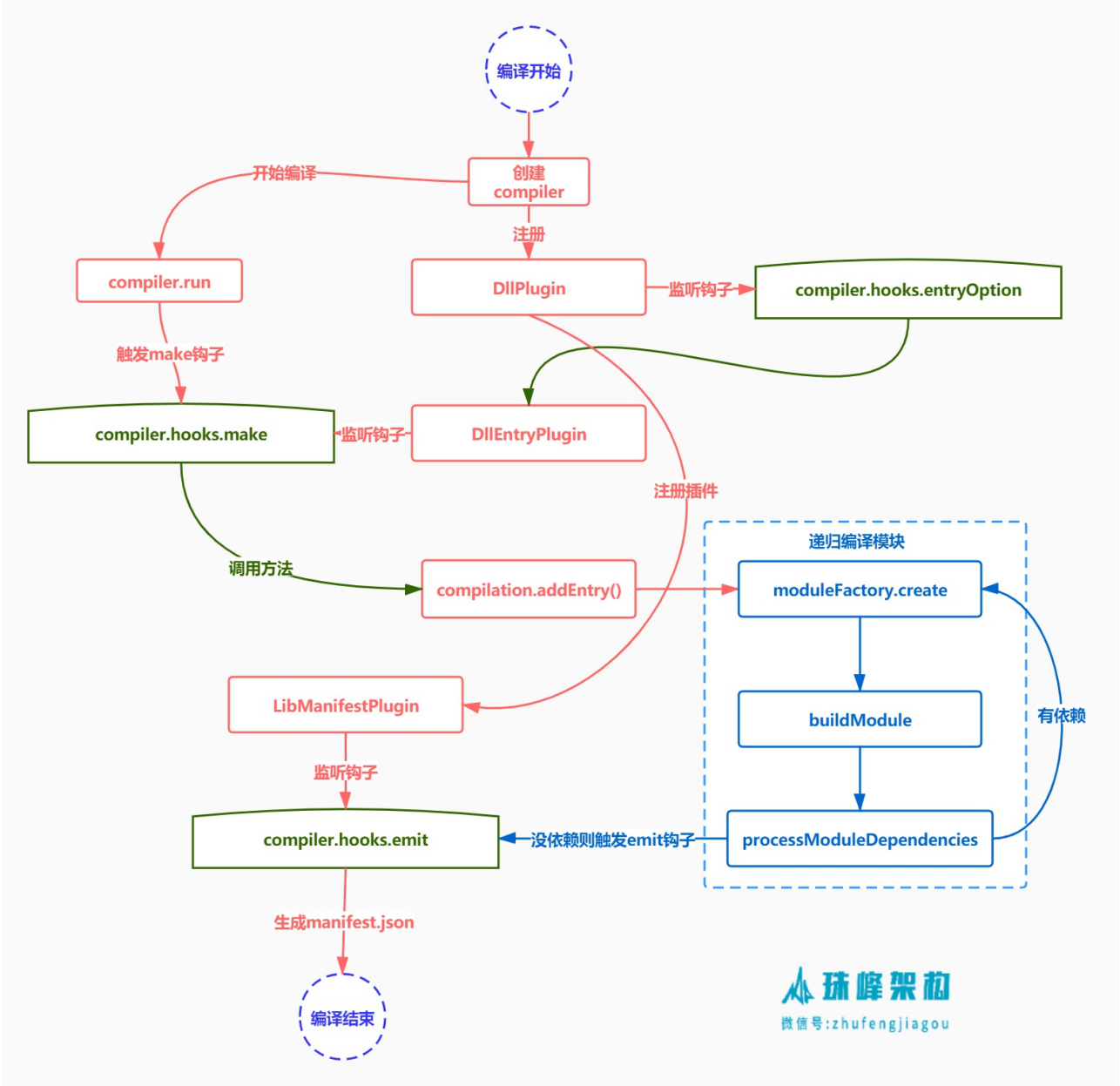
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=103 sentences=371, words=1953

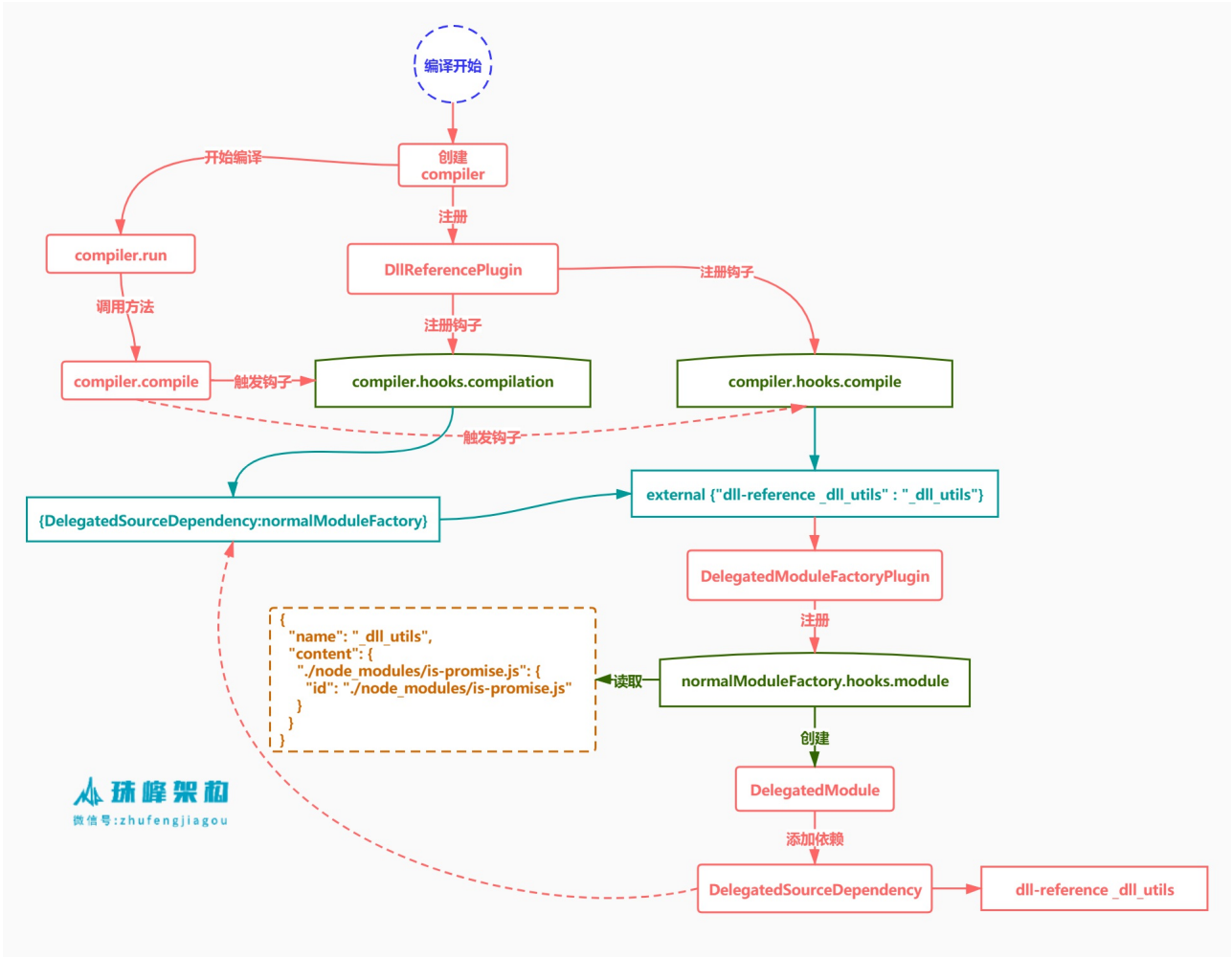
1. 什么是DLL

- `DllPlugin`和`DllReferencePlugin`提供了拆分包的方法，可以极大地提高构建时性能。术语`DLL`代表动态链接库，它最初是由Microsoft引入的。
- `.dll`为后缀的文件称为动态链接库，在一个动态链接库中可以包含给其他模块调用的函数和数据
- 把基础模块独立出来打包到单独的动态连接库里
- 当需要导入的模块在动态连接库里的时候，模块不能再次被打包，而是去动态连接库里获取

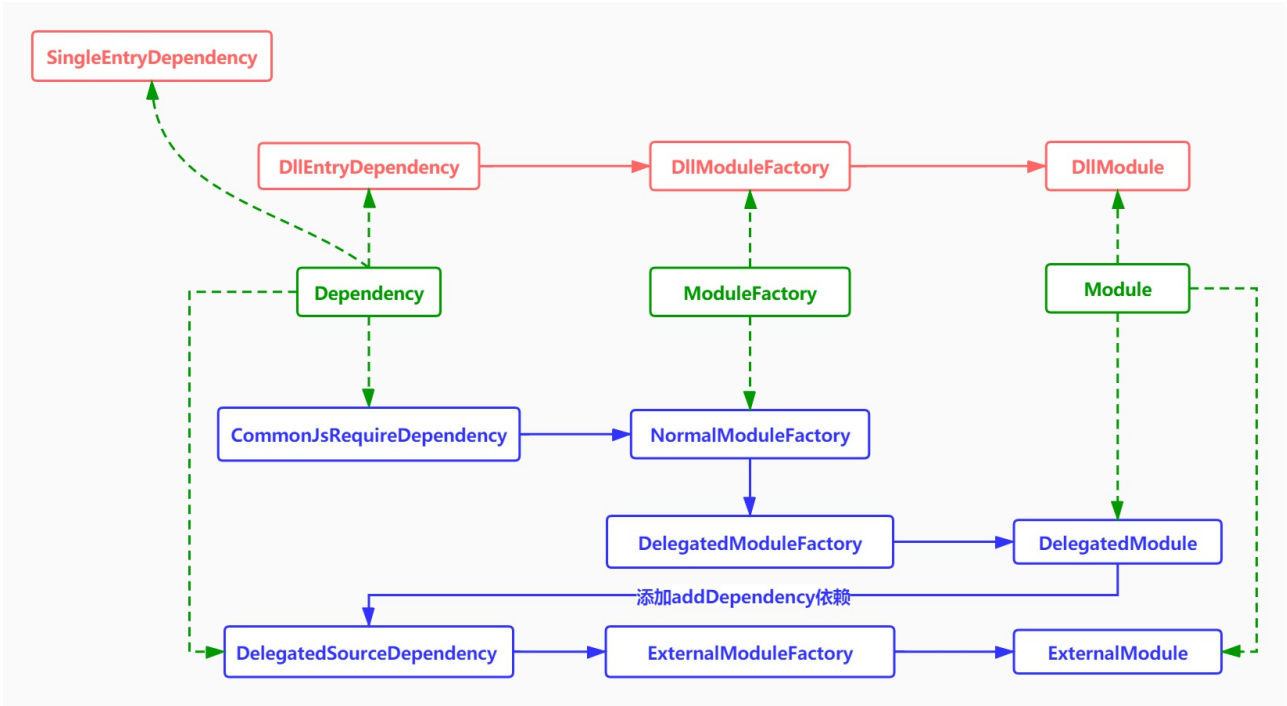
1.1 DllPlugin



1.2 DllReferencePlugin



1.3 WebpackClassDiagram #



2. 使用DLL

2.1 安装依赖

```
cnpm i webpack webpack-cli html-webpack-plugin isarray is-promise -D
```

2.2 定义Dll

- dll-plugin (https://webpack.js.org/plugins/dll-plugin/)
- DllPlugin插件： 用于打包出一个个动态连接库
- DllReferencePlugin: 在配置文件中引入DllPlugin插件打包好的动态连接库

webpack.dll.config.js

```
const path = require("path");
const DllPlugin = require("webpack/lib/DllPlugin");
const DllPlugin2 = require("./plugins/DllPlugin");
module.exports = {
  mode: "development",
  devtool: false,
  entry: {
    utils:["isarray","is-promise"]
  },
  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "utils.dll.js",
    library: "_dll_utils",
  },
  plugins: [
    new DllPlugin2({

      name: "_dll_utils",

      path: path.join(__dirname, "dist", "utils.manifest.json")
    }),
  ],
};
```

2.3 使用动态链接库文件

webpack.config.js

```
const path = require("path");
const DllReferencePlugin = require("webpack/lib/DllReferencePlugin.js");
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  mode: "development",
  devtool: false,
  entry: "./src/index.js",
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'bundle.js'
  },
  plugins: [
    new DllReferencePlugin({
      manifest: require("./dist/utils.manifest.json"),
    }),
    new HtmlWebpackPlugin({
      template: './src/index.html'
    })
  ],
};
```

2.4 html中使用

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>dlltitle>
head>
<body>
    <div id="root">div>
    <script src="utils.dll.js">script>
body>
html>
```

2.5 index.js

src\index.js

```
let isarray = require("isarray");
console.log('isarray([1, 2, 3])=',isarray([1, 2, 3]));
```

2.6 package.json

package.json

```
  "scripts": {
    "dll": "webpack --config webpack.dll.config.js",
    "build": "webpack --config webpack.config.js"
  }
```

2.7 dll.js

```
const webpack = require("webpack");
const webpackOptions = require("./webpack.dll.config");
const compiler = webpack(webpackOptions);
debugger
compiler.run((err, stats) => {
  console.log(
    stats.toJson({})
  );
});
```

2.8 build.js

```
const webpack = require("webpack");
const webpackOptions = require("./webpack.config");
const compiler = webpack(webpackOptions);
compiler.run((err, stats) => {
  console.log(
    stats.toJson({})
  );
});
```

3.打包文件分析

3.1 index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>dlltitle>
head>
<body>
    <div id="root">div>
    <script src="utils.dll.js">script>
    [illegible]
[illegible]>
```

3.2 utils.dll.js

dist\utils.dll.js

```
var _dll_utils =
  (function (modules) {
    var installedModules = {};
    function __webpack_require__(moduleId) {
      if (installedModules[moduleId]) {
        return installedModules[moduleId].exports;
      }
      var module = installedModules[moduleId] = {
        i: moduleId,
        l: false,
        exports: {}
      };
      modules[moduleId].call(module.exports, module, module.exports, __webpack_require__);
      module.l = true;
      return module.exports;
    }
    return __webpack_require__(__webpack_require__.s = 0);
  })
    ({
      "./node_modules/_is-promise@4.0.0@is-promise/index.js":
        (function (module, exports) {
          module.exports = isPromise;
          module.exports.default = isPromise;
          function isPromise(obj) {
            return !!obj && (typeof obj === 'object' || typeof obj === 'function') && typeof obj.then === 'function';
          }
        }),
      "./node_modules/_isarray@2.0.5@isarray/index.js":
        (function (module, exports) {
          var toString = {}.toString;
          module.exports = Array.isArray || function (arr) {
            return toString.call(arr) == '[object Array]';
          };
        }),
      0:
        (function (module, exports, __webpack_require__) {
          module.exports = __webpack_require__;
        })
    });
```

3.3 utils.manifest.json

dist\utils.manifest.json

```
{
  "name": "_dll_utils",
  "content": {
    "./node_modules/_is-promise@4.0.0@is-promise/index.js": {
      "id": "./node_modules/_is-promise@4.0.0@is-promise/index.js",
      "buildMeta": { "providedExports": true }
    },
    "./node_modules/_isarray@2.0.5@isarray/index.js": {
      "id": "./node_modules/_isarray@2.0.5@isarray/index.js",
      "buildMeta": {
        "providedExports": true
      }
    }
  }
}
```

3.4 bundle.js

dist\bundle.js

```
(function (modules) {
  var installedModules = {};
  function __webpack_require__(moduleId) {
    if (installedModules[moduleId]) {
      return installedModules[moduleId].exports;
    }
    var module = installedModules[moduleId] = {
      i: moduleId,
      l: false,
      exports: {}
    };
    modules[moduleId].call(module.exports, module, module.exports, __webpack_require__);
    module.l = true;
    return module.exports;
  }
  return __webpack_require__(__webpack_require__.s = "./src/index.js");
})
  ({
    "./node_modules/_isarray@2.0.5@isarray/index.js":
      (function (module, exports, __webpack_require__) {
        module.exports = (__webpack_require__("dll-reference _dll_utils"))("./node_modules/_isarray@2.0.5@isarray/index.js");
      }),
    "./src/index.js":
      (function (module, exports, __webpack_require__) {
        let isarray = __webpack_require__("./node_modules/_isarray@2.0.5@isarray/index.js");
        console.log('isarray([1, 2, 3])=', isarray([1, 2, 3]));
      }),
    "dll-reference _dll_utils":
      (function (module, exports) {
        module.exports = _dll_utils;
      })
  });
```

4.实现DllPlugin.js

4.1 DllPlugin.js

plugins\DllPlugin.js

```
const DllEntryPlugin = require("./DllEntryPlugin");
class DllPlugin {
    constructor(options) {
        this.options = options;
    }
    apply(compiler) {
        compiler.hooks.entryOption.tap("DllPlugin", (context, entry) => {
            Object.keys(entry).forEach(name => {
                new DllEntryPlugin(context, entry[name],name).apply(compiler);
            });
            return true;
        });
    }
}
module.exports = DllPlugin;
```

4.2 DllEntryPlugin.js

plugins\DllEntryPlugin.js

```
const SingleEntryDependency = require("webpack/lib/dependencies/SingleEntryDependency");
const DllEntryDependency = require("./dependencies/DllEntryDependency");
const DllModuleFactory = require("./DllModuleFactory");
class DllEntryPlugin {
    constructor(context, entries, name) {
        this.context = context;
        this.entries = entries;
        this.name = name;
    }
    apply(compiler) {
        compiler.hooks.compilation.tap(
            "DllEntryPlugin",
            (compilation, { normalModuleFactory }) => {
                const dllModuleFactory = new DllModuleFactory();
                compilation.dependencyFactories.set(
                    DllEntryDependency,
                    dllModuleFactory
                );
                compilation.dependencyFactories.set(
                    SingleEntryDependency,
                    normalModuleFactory
                );
            }
        );
        compiler.hooks.make.tapAsync("DllEntryPlugin", (compilation, callback) => {
            compilation.addEntry(
                this.context,
                new DllEntryDependency(
                    this.entries.map((entry) => new SingleEntryDependency(entry)),
                    this.name
                ),
                this.name,
                callback
            );
        });
    }
}
module.exports = DllEntryPlugin;
```

4.3 DllModuleFactory.js

plugins\DllModuleFactory.js

```
const { Tapable } = require("tapable");
const DllModule = require("./DllModule");
class DllModuleFactory extends Tapable {
    constructor() {
        super();
        this.hooks = {};
    }
    create(data, callback) {
        const dependency = data.dependencies[0];
        callback(
            null,
            new DllModule(
                data.context,
                dependency.dependencies,
                dependency.name,
                dependency.type
            )
        );
    }
}
module.exports = DllModuleFactory;
```

4.4 DllEntryDependency.js

plugins\dependencies\DllEntryDependency.js

```
const Dependency = require("webpack/lib/Dependency");
class DllEntryDependency extends Dependency {
    constructor(dependencies, name) {
        super();
        this.dependencies = dependencies;
        this.name = name;
    }
    get type() {
        return "dll entry";
    }
}
module.exports = DllEntryDependency;
```

4.5 DllModule.js

plugins\DllModule.js

```
const { RawSource } = require("webpack-sources");
const Module = require("webpack/lib/Module");
class DllModule extends Module {
    constructor(context, dependencies, name, type) {
        super("javascript/dynamic", context);
        this.dependencies = dependencies;
        this.name = name;
        this.type = type;
    }
    identifier() {
        return `dll ${this.name}`;
    }
    readableIdentifier() {
        return `dll ${this.name}`;
    }
    build(options, compilation, resolver, fs, callback) {
        this.built = true;
        this.buildMeta = {};
        this.buildInfo = {};
        return callback();
    }
    source() {
        return new RawSource("module.exports = __webpack_require__;");
    }
    size() {
        return 12;
    }
}
module.exports = DllModule;
```

5.实现LibManifestPlugin.js

5.1 LibManifestPlugin.js

plugins\LibManifestPlugin.js

```
const path = require("path");
const asyncLib = require("neo-async");
class LibManifestPlugin {
    constructor(options) {
        this.options = options;
    }
    apply(compiler) {
        compiler.hooks.emit.tapAsync(
            "LibManifestPlugin",
            (compilation, callback) => {
                asyncLib.forEach(
                    compilation.chunks,
                    (chunk, done) => {

                        const targetPath = this.options.path;
                        const name =this.options.name;
                        let content ={};
                        for(let module of chunk.modulesIterable){
                            if (module.libIdent) {
                                const ident = module.libIdent({context:compiler.options.context});
                                content[ident]= {id: module.id};
                            }
                        }
                        const manifest = {name,content};
                        compiler.outputFileSystem.mkdirp(path.dirname(targetPath), err => {
                            compiler.outputFileSystem.writeFile(
                                targetPath,
                                JSON.stringify(manifest),
                                done
                            );
                        });
                    },
                    callback
                );
            }
        );
    }
}
module.exports = LibManifestPlugin;
```

5.2 DllPlugin.js

plugins\DllPlugin.js

```
const DllEntryPlugin = require("./DllEntryPlugin");
+const LibManifestPlugin = require("./LibManifestPlugin");
class DllPlugin {
    constructor(options) {
        this.options = options;
    }
    apply(compiler) {
        compiler.hooks.entryOption.tap("DllPlugin", (context, entry) => {
            Object.keys(entry).forEach(name => {
                new DllEntryPlugin(context, entry[name],name).apply(compiler);
            });
            return true;
        });
+        new LibManifestPlugin(this.options).apply(compiler);
    }
}
module.exports = DllPlugin;
```

6. 实现DllReferencePlugin.js

6.1 DllReferencePlugin.js

plugins\DllReferencePlugin.js

```
const DelegatedSourceDependency = require("webpack/lib/dependencies/DelegatedSourceDependency");
const ExternalModuleFactoryPlugin = require("./ExternalModuleFactoryPlugin");
const DelegatedModuleFactoryPlugin = require("./DelegatedModuleFactoryPlugin");
class DllReferencePlugin {
    constructor(options) {
        this.options = options;
    }

    apply(compiler) {
        compiler.hooks.compilation.tap(
            "DllReferencePlugin",
            (compilation, { normalModuleFactory }) => {
                compilation.dependencyFactories.set(
                    DelegatedSourceDependency,
                    normalModuleFactory
                );
            }
        );

        compiler.hooks.compile.tap("DllReferencePlugin", ({normalModuleFactory}) => {
            let manifest = this.options.manifest;
            let name = manifest.name;
            let content = manifest.content;

            const externals = {};
            const source = "dll-reference " + name;
            externals[source] = name;
      new ExternalModuleFactoryPlugin("var", externals).apply(normalModuleFactory);

            new DelegatedModuleFactoryPlugin({
                source,
                context: compiler.options.context,
                content
            }).apply(normalModuleFactory);
        });
    }
}

module.exports = DllReferencePlugin;
```

6.2 ExternalModuleFactoryPlugin.js

plugins\ExternalModuleFactoryPlugin.js

```
const ExternalModule = require('webpack/lib/ExternalModule');
class ExternalModuleFactoryPlugin{
    constructor(type, externals) {
        this.type = type;
        this.externals = externals;
    }

    apply(normalModuleFactory) {

        normalModuleFactory.hooks.factory.tap(
            "ExternalModuleFactoryPlugin",

            factory => (data, callback) => {
         const dependency = data.dependencies[0];
         let request = dependency.request;
         let value = this.externals[request];
         if(value){
             callback(
                 null,
                 new ExternalModule(value, 'var', dependency.request)
             );
         }else{
             factory(data, callback);
         }
            }
        );
    }
}
module.exports = ExternalModuleFactoryPlugin;
```

6.3 DelegatedModuleFactoryPlugin.js

plugins\DelegatedModuleFactoryPlugin.js

```
const DelegatedModule = require("./DelegatedModule");
class DelegatedModuleFactoryPlugin {
    constructor(options) {
        this.options = options;
        options.type = options.type || "require";
    }
    apply(normalModuleFactory) {
        normalModuleFactory.hooks.module.tap(
            "DelegatedModuleFactoryPlugin",
            module => {
                if (module.libIdent) {
                    const request = module.libIdent(this.options);
                    if (request && request in this.options.content) {
                        const resolved = this.options.content[request];
                        return new DelegatedModule(
                            this.options.source,
                            resolved,
                            module
                        );
                    }
                }
                return module;
            }
        );
    }
}
module.exports = DelegatedModuleFactoryPlugin;
```

6.4 DelegatedModule.js

plugins\DelegatedModule.js

```
const { RawSource } = require("webpack-sources");
const DelegatedSourceDependency = require("webpack/lib/dependencies/DelegatedSourceDependency");
const Module = require("webpack/lib/Module");
class DelegatedModule extends Module {
    constructor(sourceRequest, data,originalRequest) {
        super("javascript/dynamic", null);
        this.sourceRequest = sourceRequest;
        this.request = data.id;
        this.originalRequest = originalRequest;
    }
    libIdent(options) {
        return this.originalRequest.libIdent(options);
    }
    identifier() {
        return `delegated ${this.request} from ${this.sourceRequest}`;
    }

    readableIdentifier() {
        return `delegated ${this.request} from ${this.sourceRequest}`;
    }
    size(){
        return 42;
    }
    build(options, compilation, resolver, fs, callback) {
        this.built = true;
        this.buildMeta = {};
        this.buildInfo = {};
        this.delegatedSourceDependency = new DelegatedSourceDependency(
            this.sourceRequest
        );
        this.addDependency(this.delegatedSourceDependency);
        callback();
    }
    source() {
        let str = `module.exports = (__webpack_require__("${this.sourceRequest}"))(${JSON.stringify(this.request)});`;
        return new RawSource(str);
    }
}
module.exports = DelegatedModule;
```

7. autodll-webpack-plugin

- autodll-webpack-plugin (https://www.npmjs.com/package/autodll-webpack-plugin)

7.1 webpack.config.js

```
const path = require('path');
const AutodllWebpackPlugin = require('autodll-webpack-plugin');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
    mode:'development',
    devtool:false,
    entry:'./src/index.js',
    output:{
        path:path.resolve(__dirname,'dist'),
        filename:'bundle.js',
        publicPath:''
    },
    plugins:[
        new HtmlWebpackPlugin({
            inject: true,
            template: './src/index.html',
        }),
        new AutodllWebpackPlugin({
            inject: true,
            filename: '[name].dll.js',
            entry:{
                utils:['isarray','is-promise']
            }
        })
    ]
}
```

7.2 plugin.js

node_modules_ autodll-webpack-plugin@0.4.2 (mailto:autodll-webpack-plugin@0.4.2)@autodll-webpack-plugin\lib\plugin.js

```
+const HtmlWebpackPlugin = require("html-webpack-plugin");
+        compiler.hooks.compilation.tap('AutoDllPlugin', function (compilation) {
+          if (compilation.hooks.htmlWebpackPluginBeforeHtmlGeneration) {
+            compilation.hooks.htmlWebpackPluginBeforeHtmlGeneration.tapAsync('AutoDllPlugin', doCompilation);
+          } else if (HtmlWebpackPlugin.getHooks && HtmlWebpackPlugin.getHooks(compilation)) {
+            HtmlWebpackPlugin.getHooks(compilation).beforeAssetTagGeneration.tapAsync('AutoDllPlugin', doCompilation);
+          }
+        });
```