## **1. webpack代码分割方式**

* entry配置：通过多个 entry 文件来实现
* 动态加载(按需加载)：通过主动使用import来动态加载
* 抽取公共代码：使用splitChunks配置来抽取公共代码

## **2. 基础概念**

| **概念** | **含义** |
| --- | --- |
| Entry | 入口，Webpack 执行构建的第一步将从 Entry 开始，可抽象成输入。 |
| module | 模块，在 Webpack 里一切皆模块，一个模块对应着一个文件。Webpack 会从配置的 Entry 开始递归找出所有依赖的模块。 |
| chunk | 代码块，一个 Chunk 由多个模块组合而成，用于代码合并与分割 |
| bundle | bundle就是webpack打包后的各个文件，一般和chunk是一对一的关系,bundle是由chunk 编译打包后产出的 |

## **3.项目初始化**

mkdir zhufeng\_webpack

cd zhufeng\_webpack

cnpm init -y

cnpm i webpack@next -D

## **4. webpack5初体验**

### **4.1 webpack.config.js**

**let** path = require('path');module.exports = {

mode:'development',

devtool:'none',

entry:'./src/index.js',

output:{

path:path.join(\_\_dirname,'./dist'),

filename:'main.js'

}

}

### **4.2 index.js**

index.js

**const** webpack = require('webpack');**const** webpackOptions = require('./webpack.config');**const** compiler = webpack(webpackOptions,(err,stats)=>{

**if**(err){

console.log(err);

}**else**{

console.log(stats.toJson({

assets: false,

hash: true

}));

}

});

### **4.3 hello.js**

src\hello.js

module.exports = 'hello';

### **4.4 index.js**

src\index.js

**let** hello = require('./hello');console.log(hello);

### **4.4 main.js**

(**function**(modules, runtime) {

// webpackBootstrap

"use strict";

// The module cache

**var** installedModules = {};

// The require function

**function** **\_\_webpack\_require\_\_**(moduleId) {

// Check if module is in cache

**if** (installedModules[moduleId]) {

**return** installedModules[moduleId].exports;

}

// Create a new module (and put it into the cache)

**var** module = (installedModules[moduleId] = {

i: moduleId,

l: false,

exports: {}

});

// Execute the module function

modules[moduleId].call(

module.exports,

module,

module.exports,

\_\_webpack\_require\_\_

);

// Flag the module as loaded

module.l = true;

// Return the exports of the module

**return** module.exports;

}

// the startup function

**function** **startup**() {

// Load entry module and return exports

**return** \_\_webpack\_require\_\_("./src/index.js");

}

// run startup

**return** startup();

})({

"./src/hello.js": **function**(module) {

module.exports = "hello";

},

"./src/index.js": **function**(

\_\_unusedmodule,

\_\_unusedexports,

\_\_webpack\_require\_\_

) {

**let** hello = \_\_webpack\_require\_\_("./src/hello.js");

console.log(hello);

}

});

## **5. entry分割**

### **5.1 webpack.config.js**

let path = require('path');

module.exports = {

mode:'development',

devtool:'none',+ entry:{+ main:path.join(\_\_dirname,'./src/index.js'),+ login:path.join(\_\_dirname,'./src/login.js')+ },

output:{

path:path.join(\_\_dirname,'./dist'),

filename:'[name].js'

}

}

## **6. 按需加载**

### **6.1 webpack.config.js**

**let** path = require('path');module.exports = {

mode:'development',

devtool:'none',

entry:path.join(\_\_dirname,'./src/index.js'),

output:{

path:path.join(\_\_dirname,'./dist'),

filename:'[name].js'

}

}

### **6.2 index.js**

src\index.js

**let** button = document.createElement('button');

button.innerHTML = '点我';

button.addEventListener('click',event=>{

**import**('./hello.js').then(result=>{

alert(result.default);

})

});document.body.appendChild(button);

### **6.3 hello.js**

src\hello.js

module.exports = 'hello';

### **6.4 main.js**

dist\main.js

(**function**(modules, runtime) {

**var** installedModules = {};

**var** installedChunks = {

main: 0

};

**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;

}

\_\_webpack\_require\_\_.e = **function**(chunkId) {

**return** **new** Promise((resovle, reject) => {

installedChunks[chunkId] = resovle;

**let** script = document.createElement('script');

script.src = chunkId;

document.body.appendChild(script);

}).catch(error=>{

alert('异步加载失败');

});

}

\_\_webpack\_require\_\_.t = **function**(value) {

value = \_\_webpack\_require\_\_(value);

**return** {

default:

value

};

}

window.webpackJsonp = (chunkId, moreModules) => {

**for** (moduleId **in** moreModules) {

modules[moduleId] = moreModules[moduleId];

}

installedChunks[chunkId]();

installedChunks[chunkId] = 0;

}

**function** **startup**() {

**return** \_\_webpack\_require\_\_("./src/index.js");

};

**return** startup();

})({

"./src/index.js": (**function**(module, exports, \_\_webpack\_require\_\_) {

**let** button = document.createElement('button');

button.innerHTML = '点我点我';

button.addEventListener('click', event => {

\_\_webpack\_require\_\_.e("src\_hello\_js.js").then(\_\_webpack\_require\_\_.t.bind(\_\_webpack\_require\_\_, "./src/hello.js")).then(result => {

alert(result.default);

});

});

document.body.appendChild(button);

}),

});

### **6.5 src\_hello\_js.js**

dist\src\_hello\_js.js

window.webpackJsonp("src\_hello\_js.js", {"./src/hello.js": (**function**(module, exports, \_\_webpack\_require\_\_) {

module.exports = 'hello';

})

});

## **7. splitChunks**

* webpack将会基于以下条件自动分割代码块:
  + 新的代码块被共享或者来自node\_modules文件夹
  + 新的代码块大于30kb(在min+giz之前)
  + 按需加载代码块的请求数量应该<=5
  + 页面初始化时加载代码块的请求数量应该<=3

## **7.1 webpack.config.js**

optimization:{

splitChunks: {

cacheGroups: { //设置缓存组用来抽取满足不同规则的chunk,下面以生成common为例

vendors: {

chunks: "initial",

name: 'vendors', //可以通过'name'配置项来控制切割之后代码块的命名,给多个分割之后的代码块分配相同的名称,所有的vendor 模块被放进一个共享的代码块中,不过这会导致多余的代码被下载所以并不推荐

test: /node\_modules/,//条件

priority: -10 ///优先级，一个chunk很可能满足多个缓存组，会被抽取到优先级高的缓存组中,为了能够让自定义缓存组有更高的优先级(默认0),默认缓存组的priority属性为负值.

},

commons: {

chunks: "initial",

name: 'commons',

minSize: 0,//最小提取字节数

minChunks: 1, //最少被几个chunk引用

priority: -20,

reuseExistingChunk: true// 如果该chunk中引用了已经被抽取的chunk，直接引用该chunk，不会重复打包代码

}

}

}

}

### **7.2 index.js**

**import** $ **from** 'jquery';**let** button = document.createElement('button');

button.innerHTML = '点我';

button.addEventListener('click',event=>{

**import**('./hello.js').then(result=>{

alert(result.default);

})

});document.body.appendChild(button);console.log($)

### **7.3 dist\index.html**

<!DOCTYPE html><html lang="en"><head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

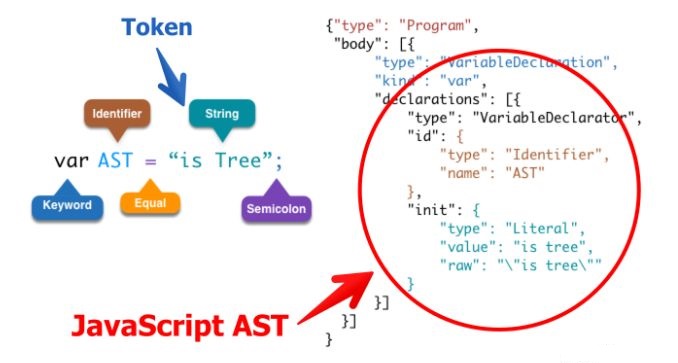
<title>Document</title></head><body><script src="src\_hello\_js.js"></script><script src="commons.js"></script><script src="vendors.js"></script><script src="main.js"></script></body></html>

* dist\commons.js
* dist\main.js
* dist\src\_hello\_js.js
* dist\vendors.js

## **8. 实现webpack**

### **8.1 AST**

* [astexplorer](https://astexplorer.net/)
* JavaScript Parser把代码转化为一颗抽象语法树（AST），这颗树定义了代码的结构，通过操纵这颗树，我们可以精准的定位到声明语句、赋值语句、运算语句等等，实现对代码的分析、优化、变更等操作



### **8.2 工具**

* [babylon](https://www.npmjs.com/package/babylon) is a JavaScript parser used in Babel.
* [babel-types](https://www.npmjs.com/package/babel-types) contains methods for building ASTs manually and for checking the types of AST nodes.
* [babel-generator](https://www.npmjs.com/package/babel-generator) Turns an AST into code.
* [babel-traverse](https://www.npmjs.com/package/babel-traverse) maintains the overall tree state, and is responsible for replacing, removing, and adding nodes.

### **8.3 index.js**

//const webpack = require('webpack');**const** webpack = require('./zfpack');**const** webpackOptions = require('./webpack.config');**const** compiler = webpack(webpackOptions,(err,stats)=>{

**if**(err){

console.log(err);

}**else**{

console.log(stats.toJson({

assets: false,

hash: true

}));

}

});

### **8.4 zfpack.js**

**const** path = require('path');**const** fs = require('fs');**const** ejs = require('ejs');**const** babylon = require('babylon');**const** t = require('babel-types');**const** generate = require('babel-generator').default;**const** traverse = require('babel-traverse').default;**let** mainTemplate = fs.readFileSync(path.join(\_\_dirname, 'main.ejs'), 'utf8');**class** **Compiler**{

**constructor**(config){

**this**.config = config;

}

run(){

**let** {entry} = **this**.config;

**this**.entry = entry;

**this**.modules = {};

**this**.buildModule(entry);

**this**.emitFiles();

}

buildModule(moduleId){

**const** originalSource = fs.readFileSync(moduleId,'utf8');

**const** ast = babylon.parse(originalSource);

**let** dependencies = [];

traverse(ast,{

CallExpression:(nodePath)=>{

**if** (nodePath.node.callee.name == 'require') {

**let** node = nodePath.node;

node.callee.name = '\_\_webpack\_require\_\_';

**let** moduleName = node.arguments[0].value;

**let** dependencyModuleId = "./"+path.posix.join(path.posix.dirname(moduleId),moduleName);

dependencies.push(dependencyModuleId);

node.arguments = [t.stringLiteral(dependencyModuleId)];

}

}

});

**let** {code} = generate(ast);

**this**.modules[moduleId] = code;

dependencies.forEach(dependencyModuleId => **this**.buildModule(dependencyModuleId));

}

emitFiles(){

**let** {output} = **this**.config;

**let** outputFile = path.posix.join(output.path,output.filename);

**let** bundle = ejs.compile(mainTemplate)({ entry:**this**.entry, modules:**this**.modules});

fs.writeFileSync(outputFile, bundle);

}

}

**function** **webpack**(config){

**let** compiler = **new** Compiler(config);

compiler.run();

}

module.exports = webpack;

### **8.5 main.ejs**

(**function**(modules, runtime) {

**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;

}

**function** **startup**() {

**return** \_\_webpack\_require\_\_("<%=entry%>");

};

**return** startup();

})({

<%**for**(moduleId **in** modules){%>

"<%-moduleId%>": (**function**(module, exports, \_\_webpack\_require\_\_) {

<%-modules[moduleId]%>

}),

<%}%>

});

### **8.6 main.js**

(**function**(modules, runtime) {

**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;

}

**function** **startup**() {

**return** \_\_webpack\_require\_\_("./src/index.js");

}

**return** startup();

})({

"./src/index.js": (**function**(module, exports, \_\_webpack\_require\_\_) {

**let** hello = \_\_webpack\_require\_\_("./src/hello.js");console.log(hello);

}),

"./src/hello.js": (**function**(module, exports, \_\_webpack\_require\_\_) {

module.exports = 'hello';

}),

});

## **9. 实现懒加载**

### **9.1 src\index.js**

**let** button = document.createElement('button');

button.innerHTML = '点我';

button.addEventListener('click',event=>{

**import**('./hello.js').then(result=>{

alert(result.default);

})

});document.body.appendChild(button);

### **9.2 zfpack.js**

const path = require('path');

const fs = require('fs');

const ejs = require('ejs');

const babylon = require('babylon');

const t = require('babel-types');

const generate = require('babel-generator').default;

const traverse = require('babel-traverse').default;+ let mainTemplate = fs.readFileSync(path.join(\_\_dirname, 'main.ejs'), 'utf8');+ let chunkTemplate = fs.readFileSync(path.join(\_\_dirname, 'chunk.ejs'), 'utf8');

class Compiler{

constructor(config){

this.config = config;

}

run(){

let {entry} = this.config;

this.entry = entry;+ this.chunks={+ main:{}+ };+ this.buildModule(entry,'main');

this.emitFiles();

}+ buildModule(moduleId,chunkId){

const originalSource = fs.readFileSync(moduleId,'utf8');

const ast = babylon.parse(originalSource,{+ plugins: ['dynamicImport']

});

let dependencies = [];

traverse(ast,{

+ CallExpression:(nodePath)=>{

if (nodePath.node.callee.name == 'require') {

let node = nodePath.node;

node.callee.name = '\_\_webpack\_require\_\_';

let moduleName = node.arguments[0].value;

let dependencyModuleId = "./"+path.posix.join(path.posix.dirname(moduleId),moduleName);

dependencies.push(dependencyModuleId);

node.arguments = [t.stringLiteral(dependencyModuleId)];

+ }else if (t.isImport(nodePath.node.callee)) {

+ let node = nodePath.node;

+ let moduleName = node.arguments[0].value;

+ let dependencyModuleId = "./"+path.posix.join(path.posix.dirname(moduleId),moduleName);+ let dependencyChunkId = dependencyModuleId.slice(2).replace(/(\/|\.)/g,'\_')+'.js';+ nodePath.replaceWithSourceString(`\_\_webpack\_require\_\_.e("${dependencyChunkId}").then(\_\_webpack\_require\_\_.t.bind(\_\_webpack\_require\_\_,"${dependencyModuleId}"))`);+ this.buildModule(dependencyModuleId,dependencyChunkId);+ }+ }+ });

let {code} = generate(ast);+ (this.chunks[chunkId]=this.chunks[chunkId]||{})[moduleId] = code;

dependencies.forEach(dependencyModuleId => this.buildModule(dependencyModuleId,chunkId));

}

emitFiles(){

let {output} = this.config;+ let chunks = Object.keys(this.chunks).forEach(chunkId=>{+ if(chunkId === 'main'){+ let outputFile = path.posix.join(output.path,output.filename);+ let mainContent = ejs.compile(mainTemplate)({ entry:this.entry, modules:this.chunks[chunkId] });+ fs.writeFileSync(outputFile, mainContent);+ }else{+ let chunkContent = ejs.compile(chunkTemplate)({chunkId,modules:this.chunks[chunkId]});+ let outputFile = path.join(output.path,chunkId);+ fs.writeFileSync(outputFile, chunkContent);+ }+ });

}

}

function webpack(config){

let compiler = new Compiler(config);

compiler.run();

}

module.exports = webpack;

### **9.3 main.ejs**

(**function**(modules, runtime) {

**var** installedModules = {};

**var** installedChunks = {

main: 0

};

**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;

}

\_\_webpack\_require\_\_.e = **function**(chunkId) {

**return** **new** Promise((resovle, reject) => {

installedChunks[chunkId] = resovle;

**let** script = document.createElement('script');

script.src = chunkId;

document.body.appendChild(script);

});

}

\_\_webpack\_require\_\_.t = **function**(value) {

value = \_\_webpack\_require\_\_(value);

**return** {

default:

value

};

}

window.webpackJsonp = (chunkId, moreModules) => {

**for** (moduleId **in** moreModules) {

modules[moduleId] = moreModules[moduleId];

}

installedChunks[chunkId]();

installedChunks[chunkId] = 0;

}

**function** **startup**() {

**return** \_\_webpack\_require\_\_("<%=entry%>");

};

**return** startup();

})({

<%**for**(moduleId **in** modules){%>

"<%-moduleId%>": (**function**(module, exports, \_\_webpack\_require\_\_) {

<%-modules[moduleId]%>

}),

<%}%>

});

### **9.4 chunk.ejs**

window.webpackJsonp(["<%-chunkId%>"], {

<%**for**(moduleId **in** modules){%>

"<%-moduleId%>": (**function**(module, exports, \_\_webpack\_require\_\_) {

<%-modules[moduleId]%>

}),

<%}%>

});