---

link: null
title: 珠峰架构师成长计划
description: index.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=102 sentences=342, words=1959

---

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) {

  "use strict";

  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/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

7.1 webpack.config.js

```
 optimization:{
    splitChunks: {
     cacheGroups: {
          vendors: {
            chunks: "initial",
            name: 'vendors',
            test: /node_modules/,
            priority: -10
          },
           commons: {
            chunks: "initial",
            name: 'commons',
            minSize: 0,
            minChunks: 1,
            priority: -20,
            reuseExistingChunk: true
          }
     }
   }
 }
```

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

```
<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>Documenttitle>
head>
<body>
<script src="src_hello_js.js">script>
<script src="commons.js">script>
<script src="vendors.js">
```

- 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('./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__("");
  };
  return startup();
})({
  for(moduleId in modules){%>
          "": (function(module, exports, __webpack_require__) {

           }),

});
```

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__("");
  };
  return startup();
})({
  for(moduleId in modules){%>
          "": (function(module, exports, __webpack_require__) {

           }),

});
```

9.4 chunk.ejs

```
window.webpackJsonp([""], {
    for(moduleId in modules){%>
            "": (function(module, exports, __webpack_require__) {

             }),

});
```