第十节 Rollup

1.定义

2.简单使用

1.安装

```
yarn add rollup -D
```

2.编写文件

```
// message.js
export default {
    hi: 'Hey Guys , I am mjq~'
}
// log.js
export const log = msg => {
    console.log("----------INFO--------");
    console.log(msg);
    console.log("----------INFO--------");
}
export const error = msg => {
    console.log('----------error--------');
```

```
    console.log(msg);
    console.log('----------error--------');
}
//index.js
import { log } from './logger'
import message from './message'

const msg = message.hi

log(msg)
```

3.使用命令打包
--format 打包成闭包立即执行函数的形式
--file 指定打包路径

```
yarn rollup ./src/index.js --format iife --file dist/bundle.js
```

4.打包后的效果
自动带有tree shake的效果

```
(function () {
    'use strict';

    const log = msg => {
        console.log("----------INFO--------");
        console.log(msg);
        console.log("----------INFO--------");
    };

    var message = {
        hi: 'Hey Guys , I am mjq~'
    };

    const msg = message.hi;

    log(msg);

}());
```

3.使用配置文件

1.根目录下新建rollup.config.js

```
export default {
    input: 'src/index.js', // 入口
    output: {  // 出口
        file: 'dist/bundle.js', // 编译到那个目录
        format: 'iife' // 打包模块形式
    }
}
```

2.使用命令行运行

```
yarn rollup --config
```

4.使用插件

如果项目中有其他需求,例如:

加载其他资源文件

导入commonjs模块

编译ECMAScript新特性

rollup支持使用插件的方式去扩展

插件rollup唯一的扩展途径,像webpack还有load,plugin等设置

1.安装插件

```
yarn add rollup-plugin-json -D
```

2.在rollup.config.js中使用插件

```
import json from 'rollup-plugin-json' // 导入插件

export default {
    input: 'src/index.js', // 入口
    output: {  // 出口
        file: 'dist/bundle.js', // 编译到那个目录
```

```
        format: 'iife' // 打包模块形式
    },
    plugins: [  // 插件列表
        json() // 使用插件
    ]
}
```

```
import { log } from './logger'
import message from './message'
import {name ,version } from '../package.json' // 导入json文件,提取字段

const msg = message.hi

log(msg)
log(name) // 使用字段
log(version)
```

3.打包后,可以看出自动tree shake,而且json对应的字段会被加载尽量

```
(function () {
    'use strict';

    const log = msg => {
        console.log("----------INFO--------");
        console.log(msg);
        console.log("----------INFO--------");
    };

    var message = {
        hi: 'Hey Guys , I am mjq~'
    };

    var name = "01-rollup-start";
    var version = "1.0.0";

    const msg = message.hi;

    log(msg);
```

```
    log(name);
    log(version);

}());
```

5.加载npm es模块

rollup不能自动加载第三方模块,需要使用rollup-plugin-node-resolve插件解决这个问题,使用模块名称导入模块

注意:这个插件只能解析es模块

1.安装

```
arn add rollup-plugin-node-resolve -D
```

2.rollup.config.js中使用插件

```
import json from 'rollup-plugin-json'
import resolve from 'rollup-plugin-node-resolve'

export default {
    input: 'src/index.js', // 入口
    output: {
        // 出口
        file: 'dist/bundle.js', // 编译到那个目录
        format: 'iife', // 打包模块形式
    },
    plugins: [json(), resolve()],
};
```

3.安装三方模块,使用三方插件

```
yarn add lodash-es
```

```
import { log } from './logger'
import message from './message'
```

```
import {name ,version } from '../package.json'

import _ from 'lodash-es'

const msg = message.hi

log(msg)
log(name)
log(version)
log(_.camelCase('helloWorld'))
```

6.加载commonJS模块

1.安装插件

```
yarn add rollup-plugin-commonjs -D
```

2.修改rollup.config.js文件

```
import json from 'rollup-plugin-json'
import resolve from 'rollup-plugin-node-resolve'
import commonjs from 'rollup-plugin-commonjs';
export default {
    input: 'src/index.js', // 入口
    output: {
        // 出口
        file: 'dist/bundle.js', // 编译到那个目录
        format: 'iife', // 打包模块形式
    },
    plugins: [json(), resolve(), commonjs()],
};
```

7.代码拆分

可以使用Dynamic import的方式实现

```
import('./logger').then(({log}) => {
```

```
    log('code splitting')
})
```

1.打包之后报错

```
src/index.js → dist/bundle.js...
[!] Error: UMD and IIFE output formats are not supported for code-splitting builds.
Error: UMD and IIFE output formats are not supported for code-splitting builds.
    at error (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\rollup.js:213:
30)
    at validateOptionsForMultiChunkOutput (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\roll
up\dist\shared\rollup.js:977:16)
    at Bundle.generate (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\roll
up.js:899:17)
    at async write (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\rollup.j
s:19270:31)
    at async Promise.all (index 0)
    at async build (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\bin\rollup:1466
:5)
    at async runRollup (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\bin\rollup:
1610:21)

error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

2动态导入的形式不支持IIFE的形式,
这种打包形式会把所有的代码放到一个模块中,并不像webpack一样打包有引导代码
没法办法实现代码拆分,可以使用其他的打包模块标准
3.使用命令打包其他标准,但是还是报错

```
yarn rollup --config --format amd
```

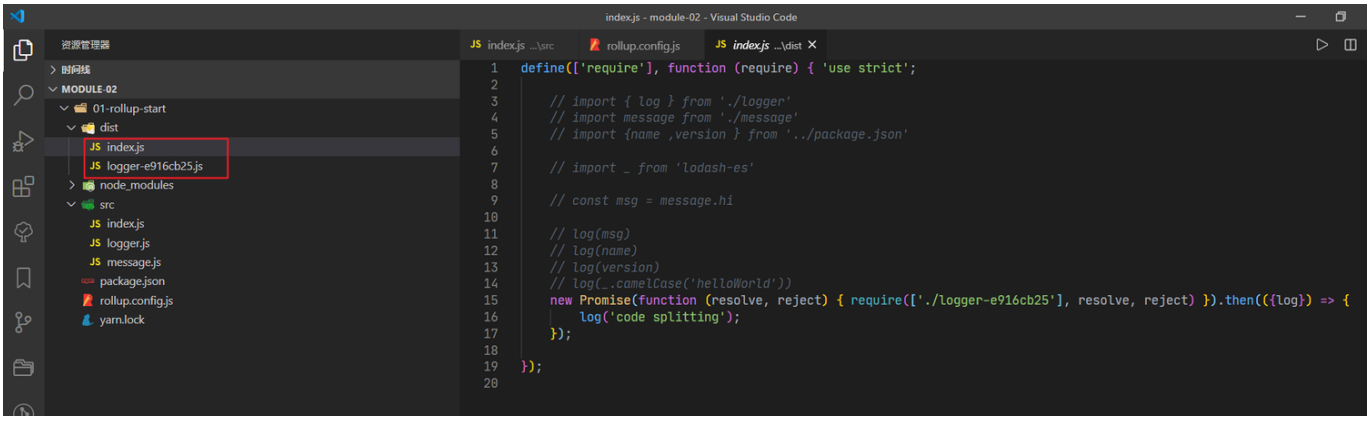
```
src/index.js → dist/bundle.js...
[!] Error: When building multiple chunks, the "output.dir" option must be used, not "output.file". To inline dynamic imports
, set the "inlineDynamicImports" option.
Error: When building multiple chunks, the "output.dir" option must be used, not "output.file". To inline dynamic imports, se
t the "inlineDynamicImports" option.
    at error (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\rollup.js:213:
30)
    at validateOptionsForMultiChunkOutput (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\roll
up\dist\shared\rollup.js:982:16)
    at Bundle.generate (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\roll
up.js:899:17)
    at async write (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\shared\rollup.j
s:19270:31)
    at async Promise.all (index 0)
    at async build (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\bin\rollup:1466
:5)
    at async runRollup (E:\workspace_learn\fed-e-code\part-02\module-02\01-rollup-start\node_modules\rollup\dist\bin\rollup:
1610:21)

error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

因为要代码拆分,打包多个文件,所以不能使用file的形式,需要使用dir

```
import json from 'rollup-plugin-json'
import resolve from 'rollup-plugin-node-resolve'
```

```
import commonjs from 'rollup-plugin-commonjs';
export default {
    input: 'src/index.js', // 入口
    // output: {
    //     // 出口
    //     file: 'dist/bundle.js', // 编译到那个目录
    //     format: 'iife', // 打包模块形式
    // },
    output: {
        dir: 'dist',
        format: 'amd'
    },
    plugins: [json(), resolve(), commonjs()],
};
```

4.打包结果拆分成2个文件

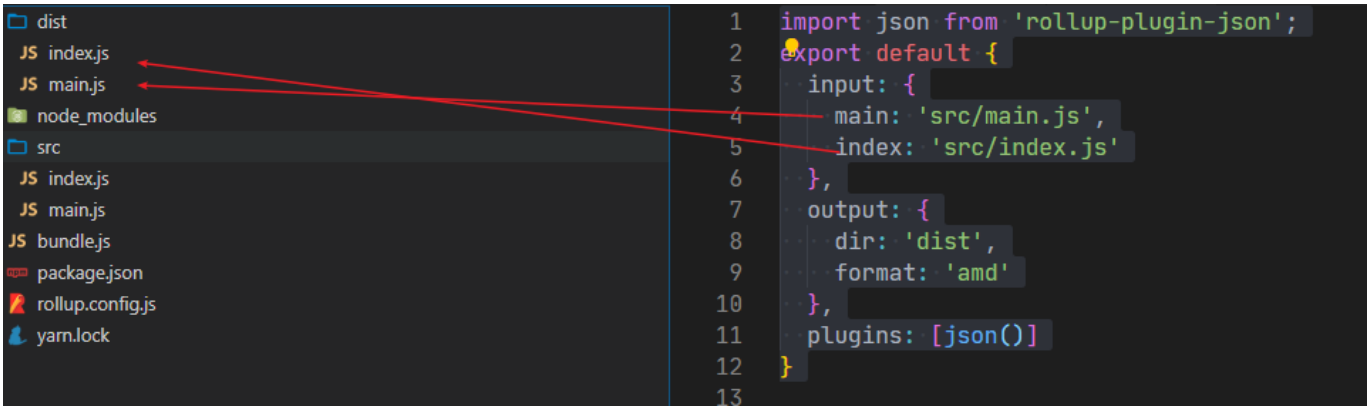


8.rollup 多入口打包

rollpu配置文件中input接收一个对象,对象的key为需要打包的一个入口

```
import json from 'rollup-plugin-json';
export default {
  input: {
    main: 'src/main.js',
    index: 'src/index.js'
  },
  output: {
    dir: 'dist',
```

```
    format: 'amd'
  },
  plugins: [json()]
}
```



可以使用require.js引入打包好的库

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <!-- 使用require.js库引用AMD标准的库 -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/require.js/2.3.6/require.js' data-main="../dist/index.js"></script>
  <script>

  </script>
</body>
</html>
```

9.rollup选用原则

优点:

- 输出结果更加扁平
- 自动移除未引用的代码,tree shake
- 打包结果依然完全可读

缺点:

- 加载非ESM的第三方模块比较复杂
- 模块最终被打包到一个函数中, 无法实现HMR(模块热替换
- 浏览器环境中,代码拆分功能依赖AMD库

如果我们正在开发应用程序(建议使用webpack)
如果我们正在开发javascript框架或者类库(vue ,react中使用rollup打包)

webpack大而全,rollup小而美