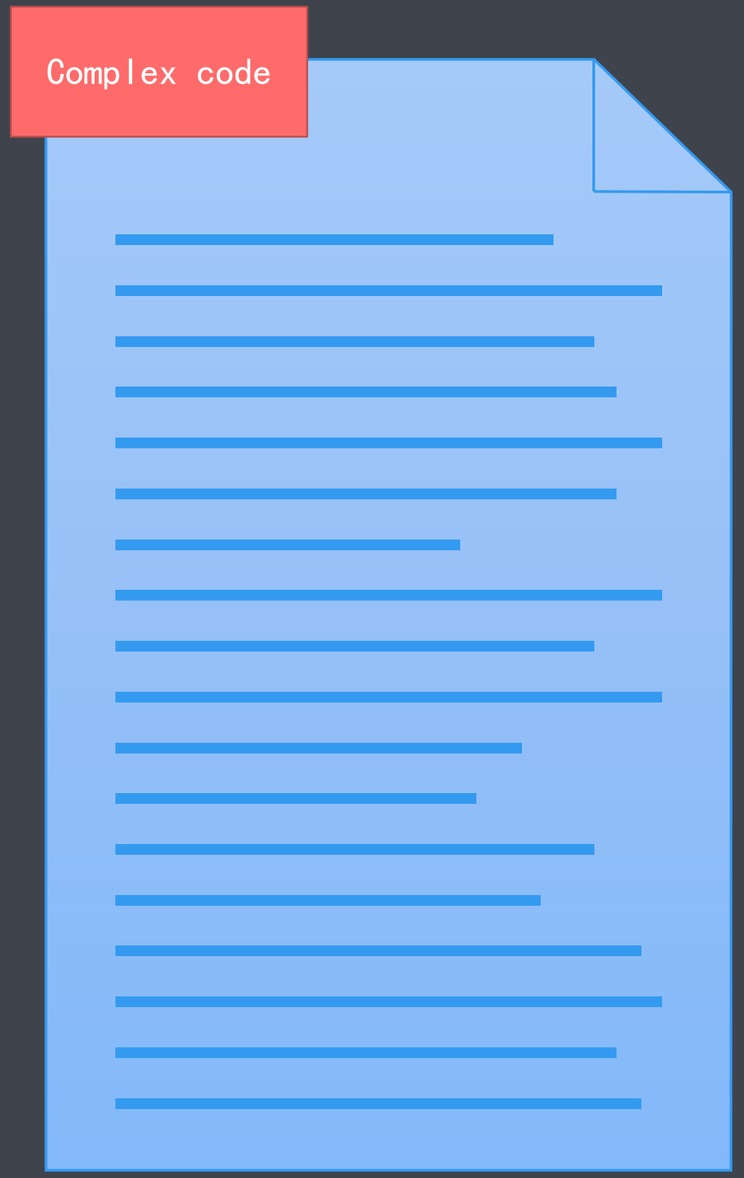
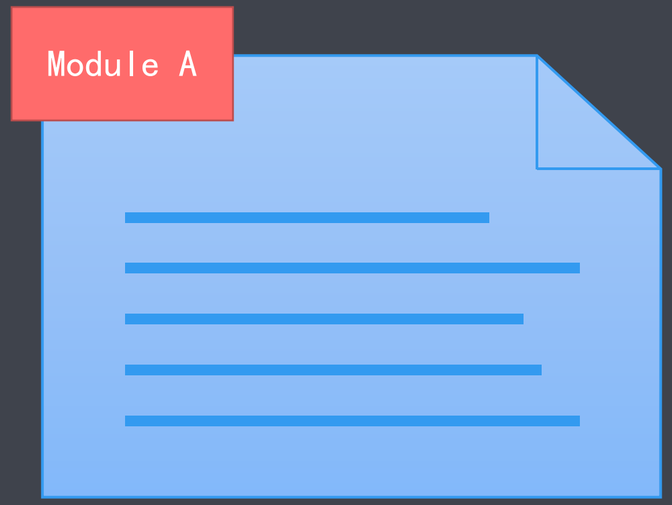
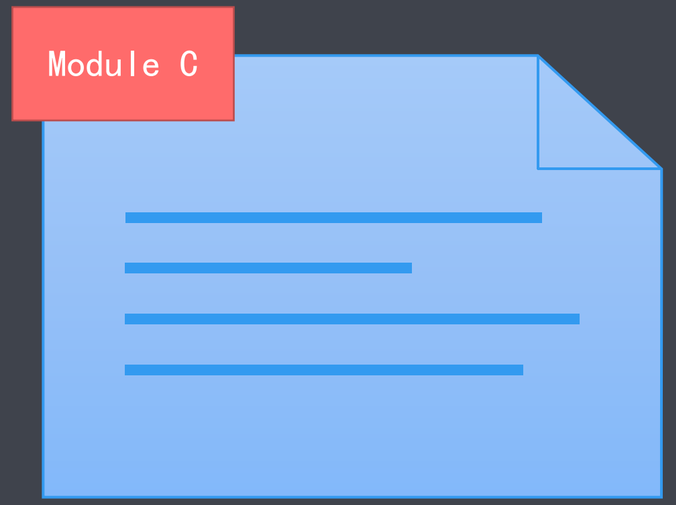
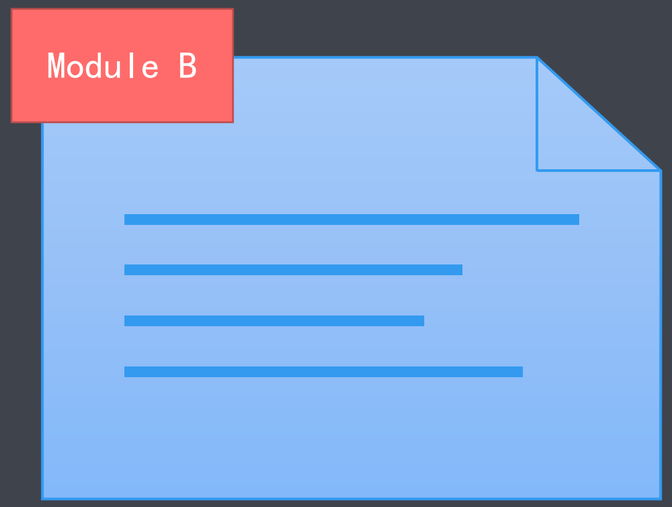
模块化开发

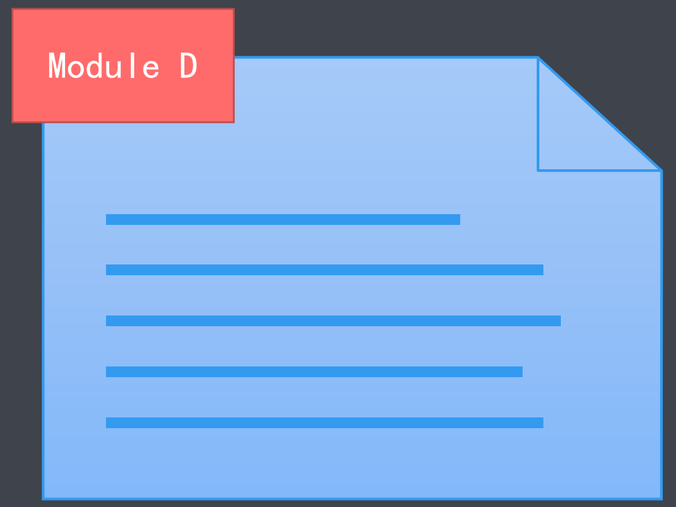
当下最重要的前端开发范式


Complex code


Module A


Module C


Module B


Module D

「模块化」只是思想

内容概要

SUMMARY

- 模块化演变过程
- 模块化规范
- 常用的模块化打包工具
- 基于模块化工具构建现代 Web 应用
- 打包工具的优化技巧

模块化演变过程

Stage 1 - 文件划分方式

- 污染全局作用域
- 命名冲突问题
- 无法管理模块依赖关系

原始方式完全依靠约定

Stage 2 - 命名空间方式

Stage 3 - IIFE

以上就是早期在没有工具和规范的情况下对模块化的落地方式

模块化规范的出现

模块化规范 + 模块加载器

CommonJS

- 一个文件就是一个模块
- 每个模块都有单独的作用域
- 通过 module.exports 导出成员
- 通过 require 函数载入模块

AMD (Asynchronous Module Definition)

Require.js

```
// 定义一个模块
define('module1', ['jquery', './module2'], function ($, module2) {
  return {
    start: function () {
      $('body').animate({ margin: '200px' })
      module2()
    }
  }
})
```

```
// 载入一个模块
require(['module1'], function (module1) {
  module1.start()
})
```

Sea.js + CMD

```
// 所有模块都通过 define 来定义
define(function (require, exports, module) {
  // 通过 require 引入依赖
  var $ = require('jquery');
  var Spinning = require('./spinning');
  // 通过 exports 对外提供接口
  exports.doSomething = ...
  // 或者通过 module.exports 提供整个接口
  module.exports = ...
});
```

这些历史对于在「和平时期」才接触前端的朋友尤为重要

模块化标准规范

最佳实践

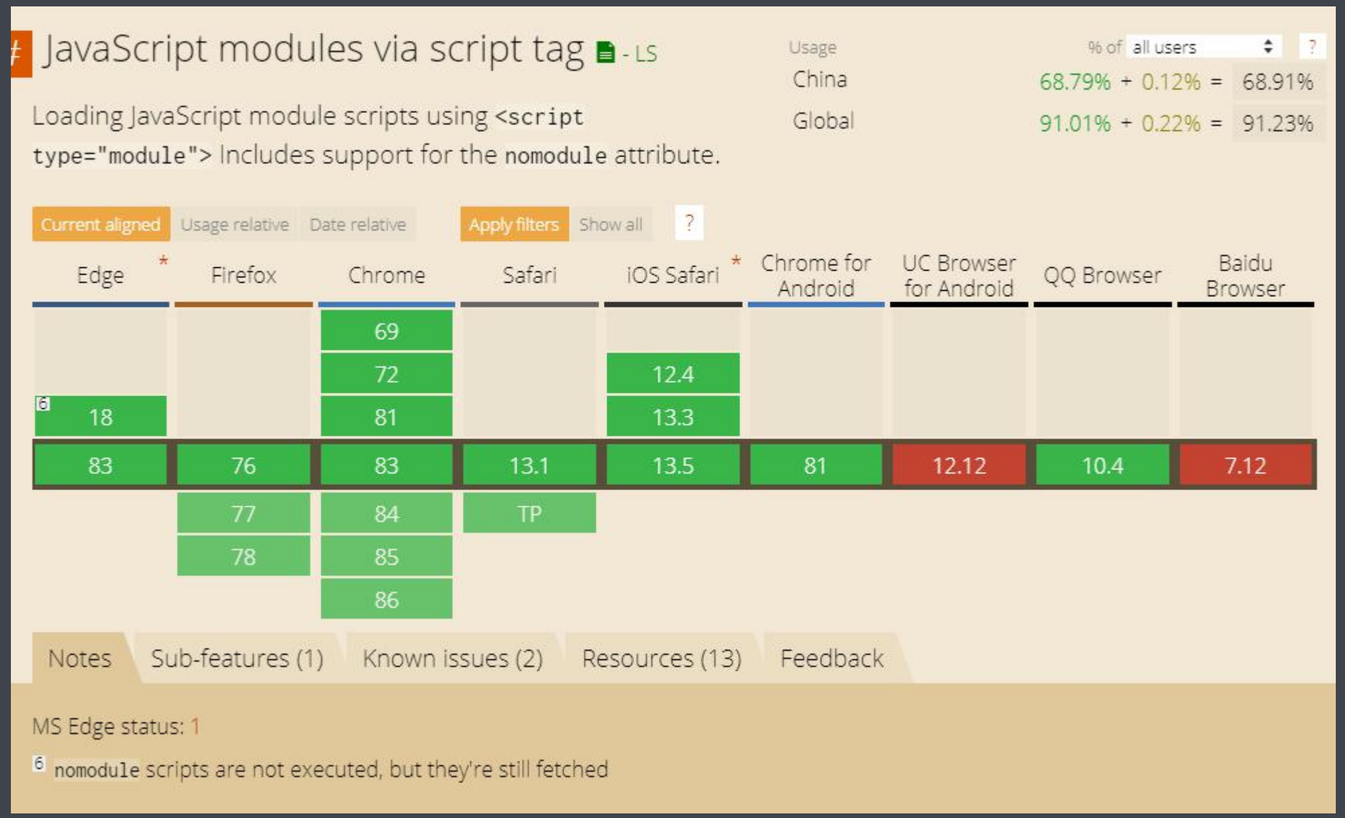
ES Modules

CommonJS

CommonJS in Node.js

ES Modules in Browers

ECMAScript 2015 (ES6)



https://caniuse.com/#feat=es6-module

ES Modules

基本特性

特性清单

- 自动采用严格模式，忽略 'use strict'
- 每个 ESM 模块都是单独的私有作用域
- ESM 是通过 CORS 去请求外部 JS 模块的
- ESM 的 script 标签会延迟执行脚本

ES Modules

导入和导出

```
// ./module.js
const foo = 'es modules'
export { foo }

// ./app.js
import { foo } from './module.js'
console.log(foo) // ⇒ es modules
```

ES Modules 导入和导出

export 用法

ES Modules 导入和导出

注意事项

ES Modules 导入和导出

import 用法

ES Modules 导入和导出

直接导出所导入的成员

ES Modules in Browser

Polyfill 兼容方案

ES Modules in Node.js

ES Modules in Node.js

与 CommonJS 模块交互

- ES Module 中可以导入 CommonJS 模块
- CommonJS 中不能导入 ES Module 模块
- CommonJS 始终只会导出一个默认成员
- 注意 import 不是解构导出对象

ES Module in Node.js

与 CommonJS 模块的差异

ES Module in Node.js

新版本进一步支持 ESM

ES Module in Node.js

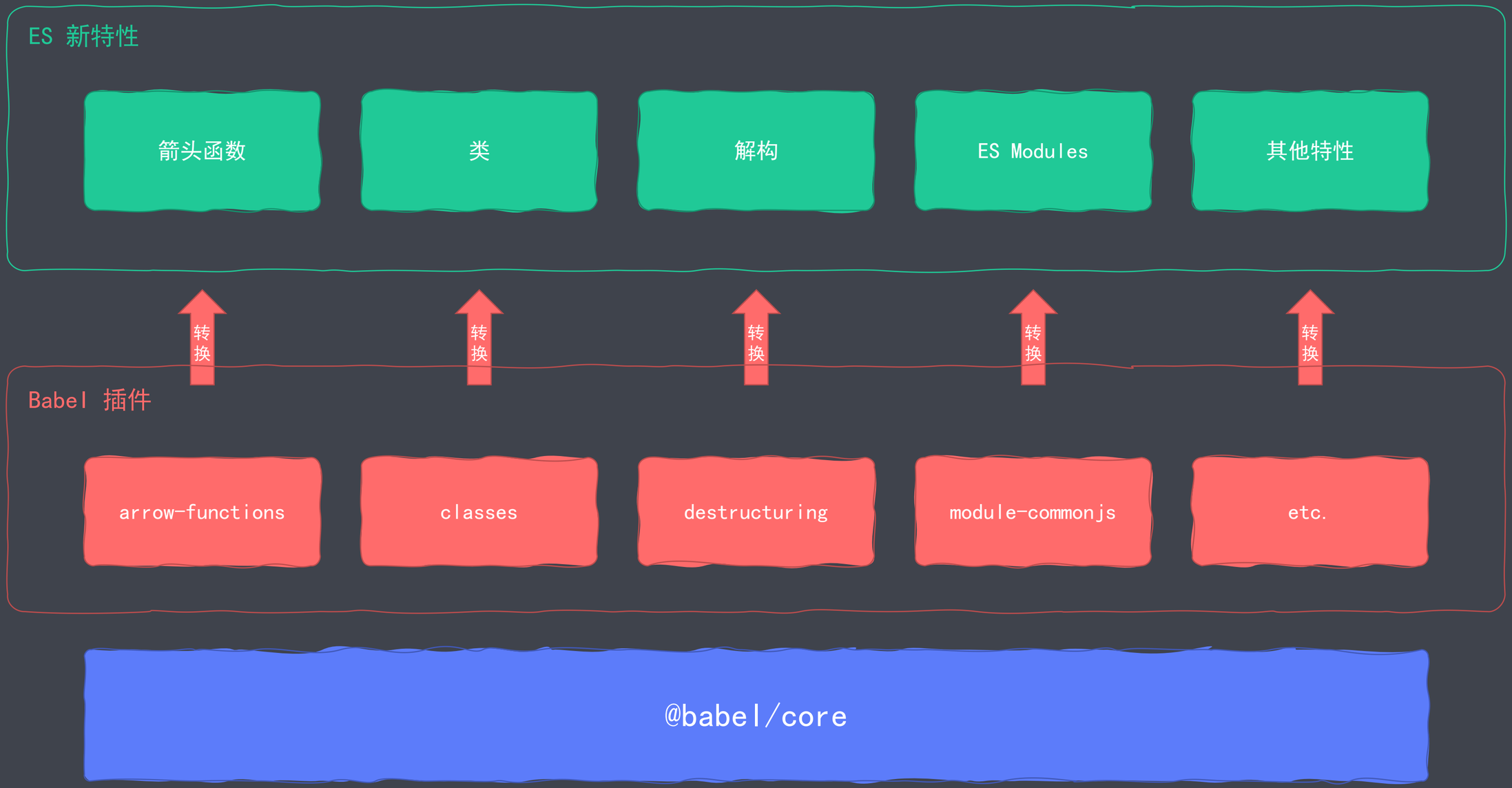
Babel 兼容方案

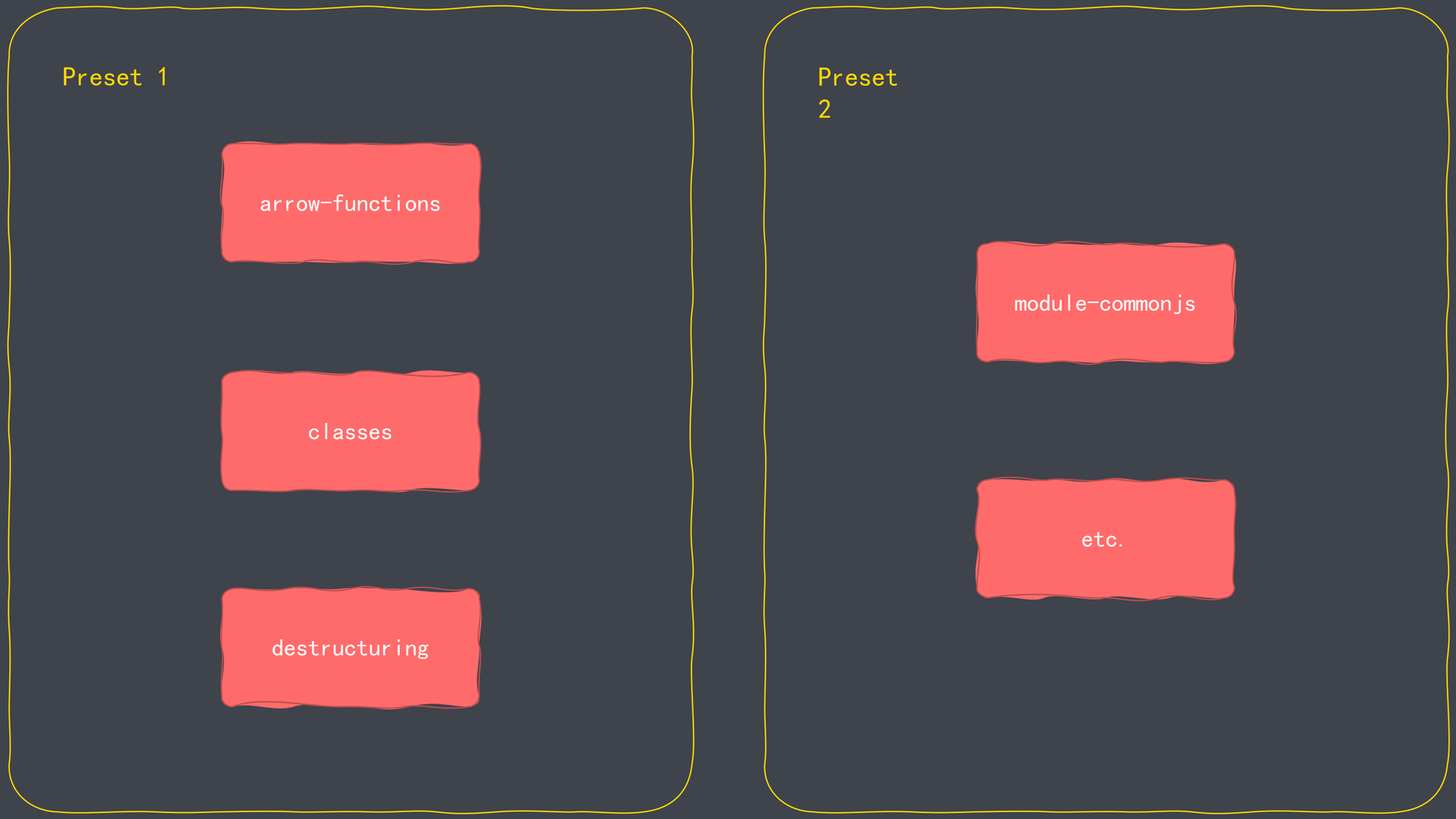
拉勾教育

—互联网人实战大学—

BABEL

L / A / G / O / U


ES 新特性
箭头函数
类
解构
ES Modules
其他特性
转换
转换
转换
转换
转换
Babel 插件
arrow-functions
classes
destructuring
module-commonjs
etc.
@babel/core

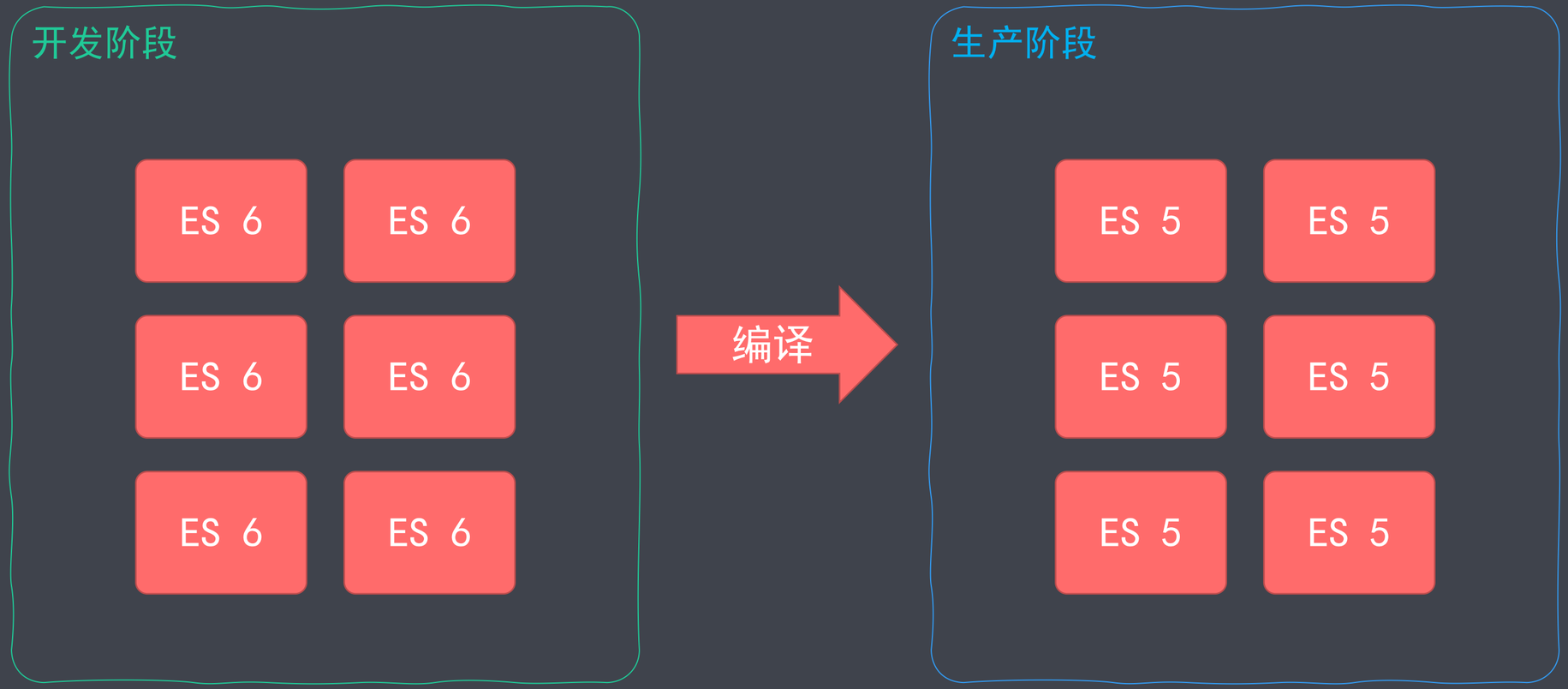

Preset 1
arrow-functions
classes
destructuring
Preset
2
module-commonjs
etc.

模块打包工具

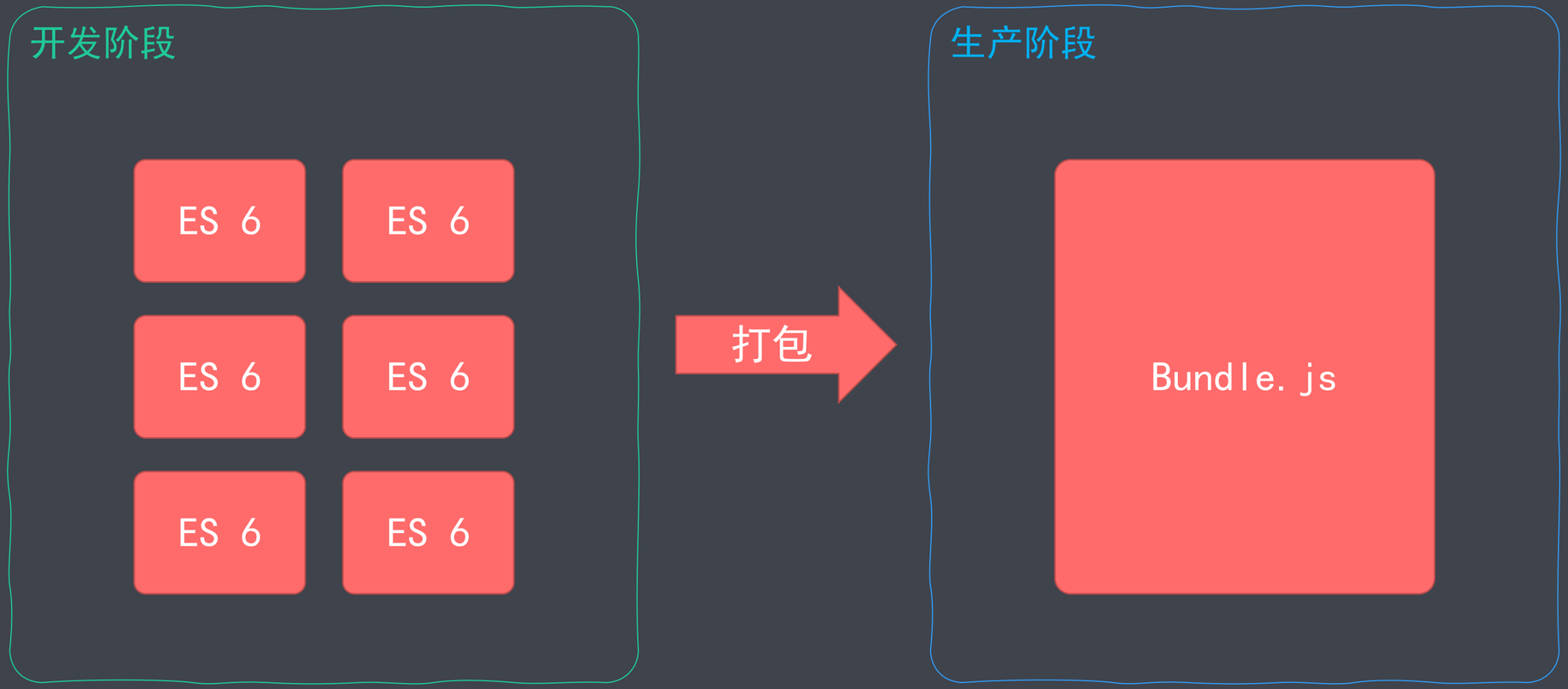
ES Modules 存在环境兼容问题

模块文件过多，网络请求频繁

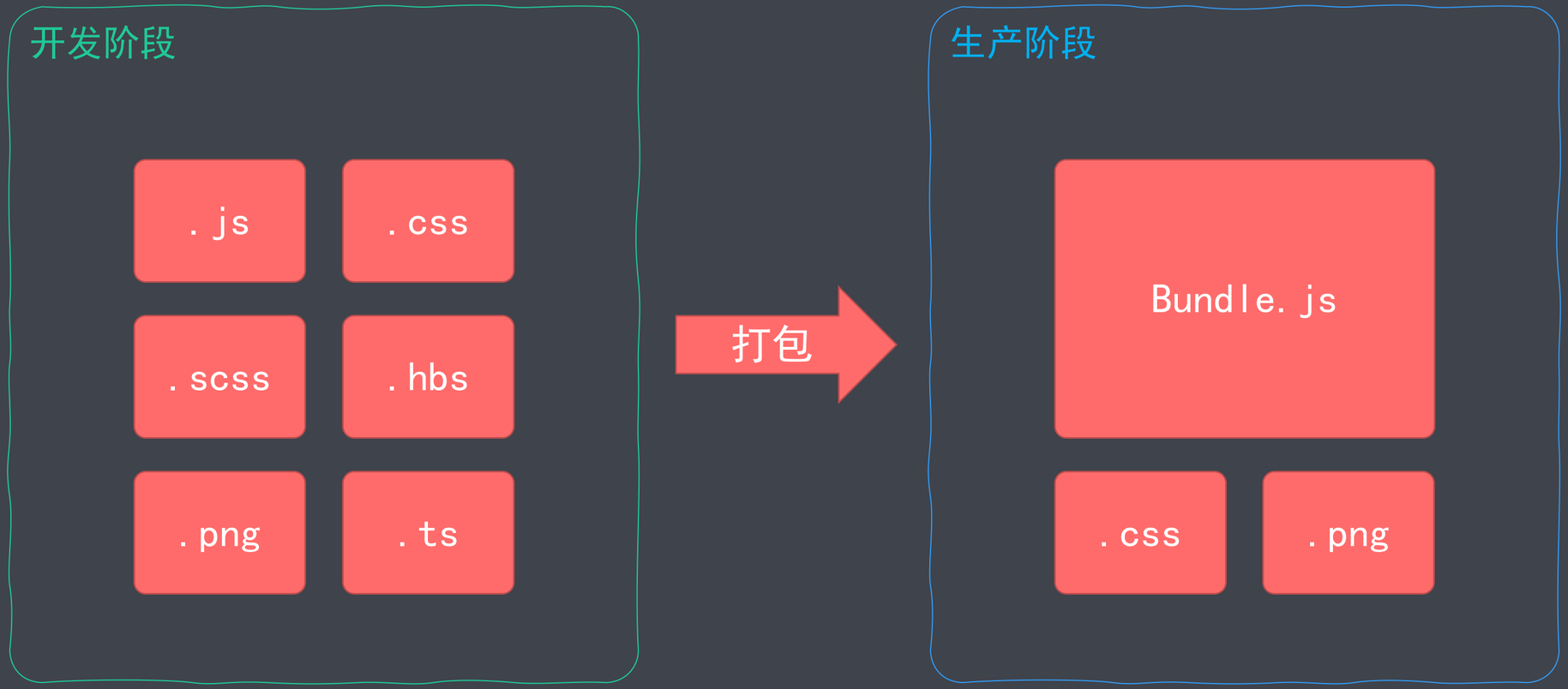
所有的前端资源都需要模块化



编译代码



模块打包



多类型模块支持

- 新特性代码编译
- 模块化 JavaScript 打包
- 支持不同类型的资源模块

模块打包工具

概要

Webpack

模块加载器（Loader）

代码拆分（Code Splitting）

资源模块（Asset Module）

Webpack 快速上手

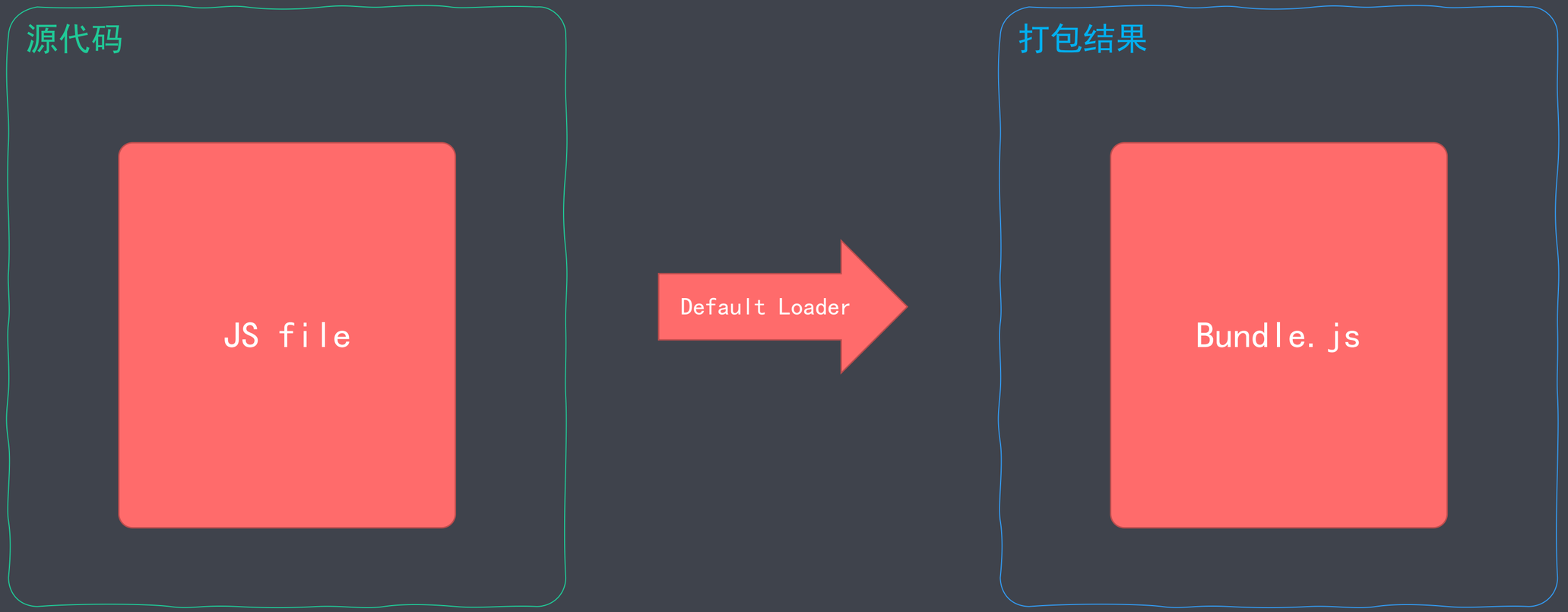
Webpack 配置文件

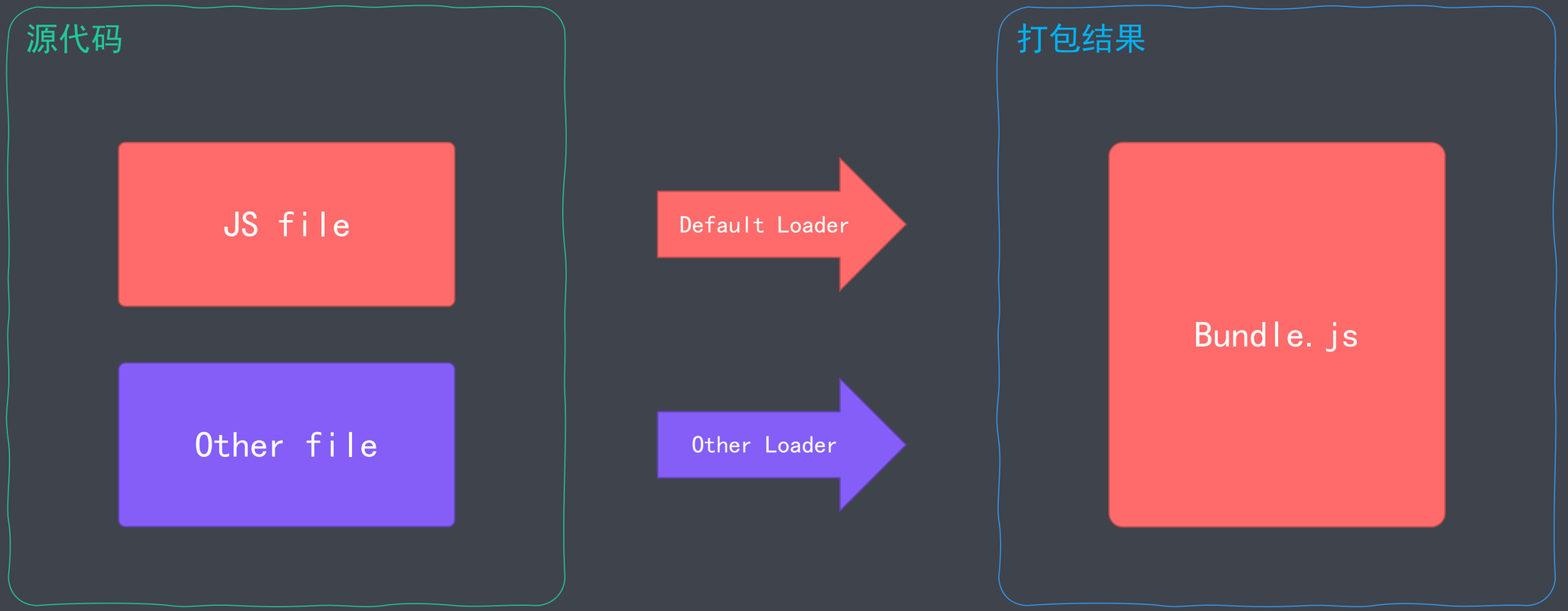
Webpack 工作模式

- 生产模式下，自动优化打包结果；
- 开发模式下，自动优化打包速度，添加一些调试过程中的辅助；
- None 模式下，运行最原始的打包，不做任何额外处理；

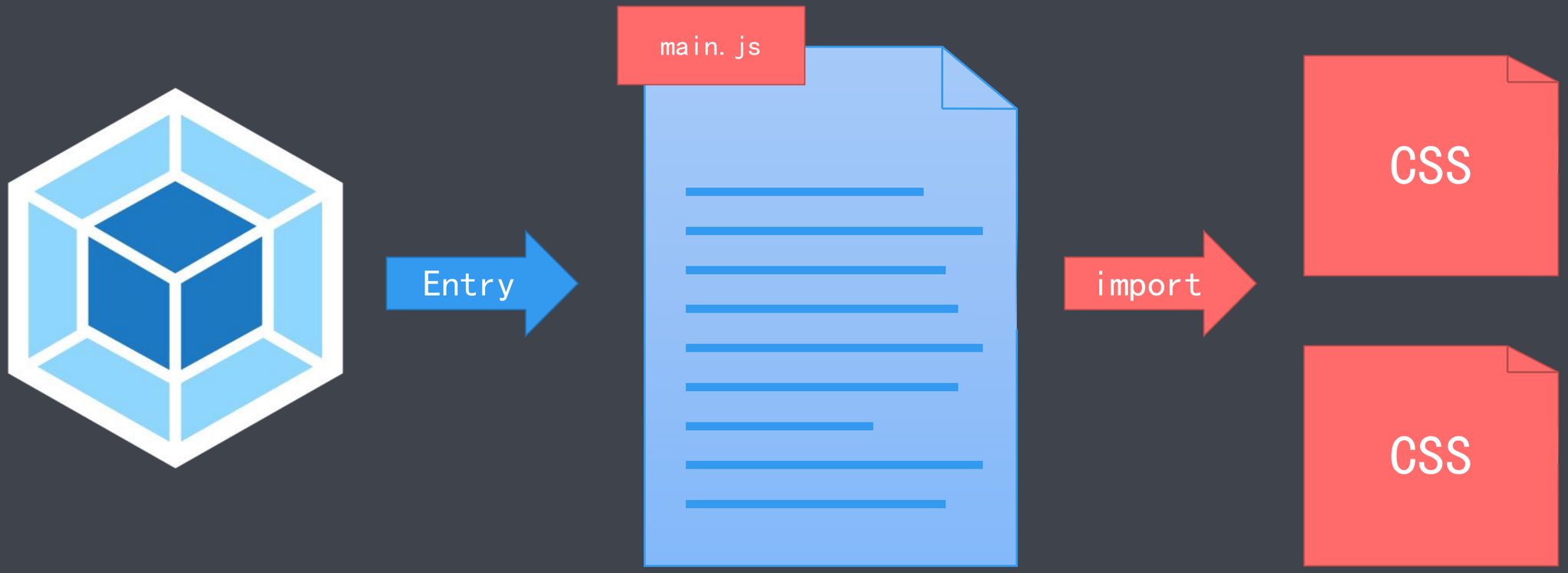
Webpack 打包结果运行原理

Webpack 资源模块加载


源代码
JS file
Default Loader
打包结果
Bundle.js

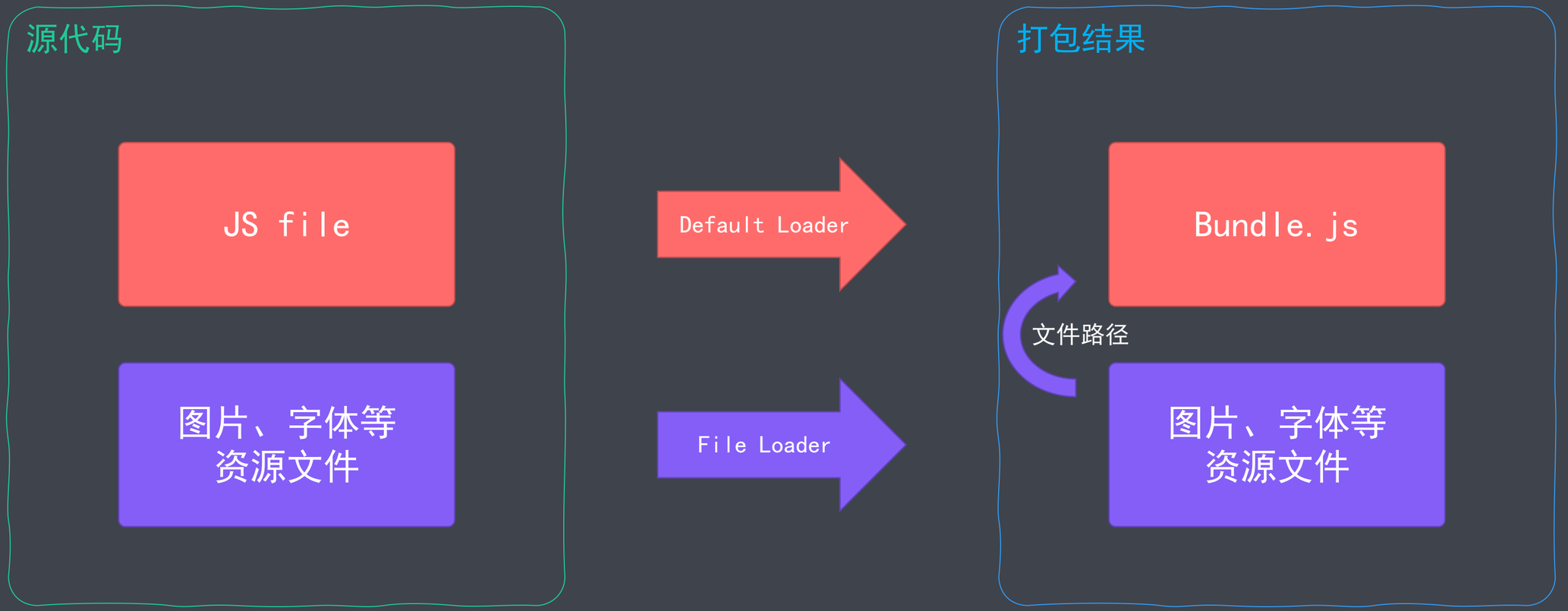

源代码
JS file
Other file
Default Loader
Other Loader
打包结果
Bundle.js

Webpack 导入资源模块


main.js
Entry
import
CSS
CSS

Webpack 文件资源加载器

大部分资源最终都


源代码
JS file
图片、字体等
资源文件
Default Loader
File Loader
打包结果
Bundle.js
文件路径
图片、字体等
资源文件

Webpack 文件资源加载器

Data URLs 与 url-loader

Data URLs

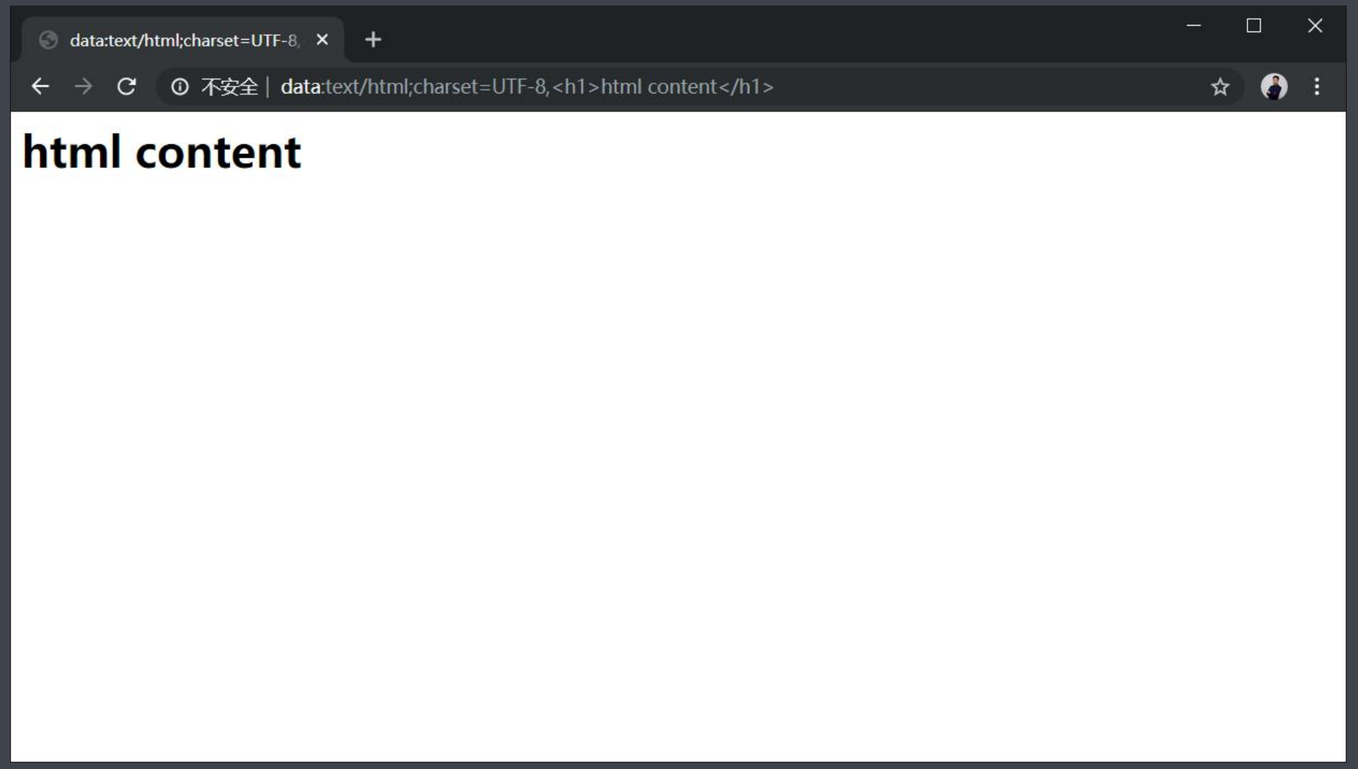
协议

媒体类型和编码

文件内容

data:[<mediatype>][;base64],<data>

data:text/html;charset=UTF-8,<h1>html content</h1>



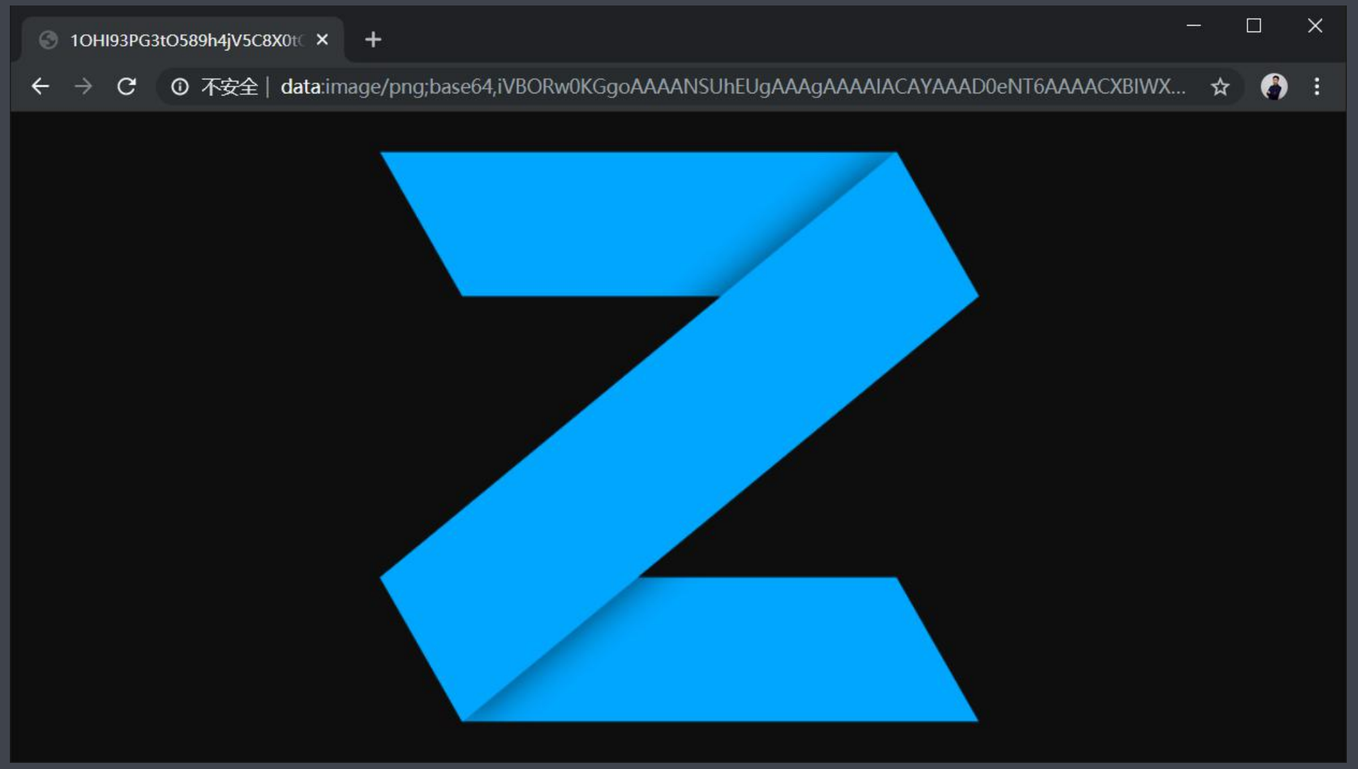

PNG


T
FONT

base64

iVBORw0KGgoAAAANSUhEUgAAAQAAAAEACAYAAABccqhmAAAgAElE
QVR4Xu2dCZhUxbXHT88+zAwgqGjXGmPieEo0KxrhE
CYgmGjdwfcaFRUFUxCVqXKNRMYrKJIHjzhpEQZYRZJFh1p693/eve6u7+
bd+249U5evv2Gm761bdarqV+ecOlUo+yuJOHqKfQhG
y6cD8WTGNbNKT1SMIoCcBsbPrdXwJgyzajZsOK3Z8/J9/kA3t/2XlZFE5ne
xR3q54mzH7qf2Oi0q2N4eNxikAxM6fp3Z4/BT/r6cZOMyWvL2TS0Ds4Pi/
3qdd/TtExb+XIHDYcJwAQNv50ekLiKiNiJqIKF/9cCBwjcBhluTtnVgCRqM+
2h7aF9oW7kHnFz9aQEgQ2GxEbgCASkBIsM7ft2/fwUcfffSv8/LyWsrLy2OF
hYWxvLw83CMvKQFHEmhvR59mV3sikWhtamqq27FjR9XWrVsr165du3r
Dhg2V6oBTyO8zglGoHXANwVFeOsvNdgHA7+OjOzo/KqHpvPPOu3T8+
PFTlbBevXpRWVlZZ5GdLKcHEkgk0s12/C5+2traqKWlhbZv3/79xx9/vGTB
ggVzFy5cuOjbb7/dRkXpqRWVlZZ5GdLKcHEkgk0s12/C5+2traqKWlhbZv3/79xx9/vGTB
ggVzFy5cuOjbb7/dRkStffr0KayoqIiVlS0xePxtpqamrauXbu2rVq1imslVk5
ED0qRu0k4BQBX89H5i1EB999u8vu+.....5CYII=

...SuQmCC


1OHI93PG3tO589h4jV5C8X0t
不安全 | ...

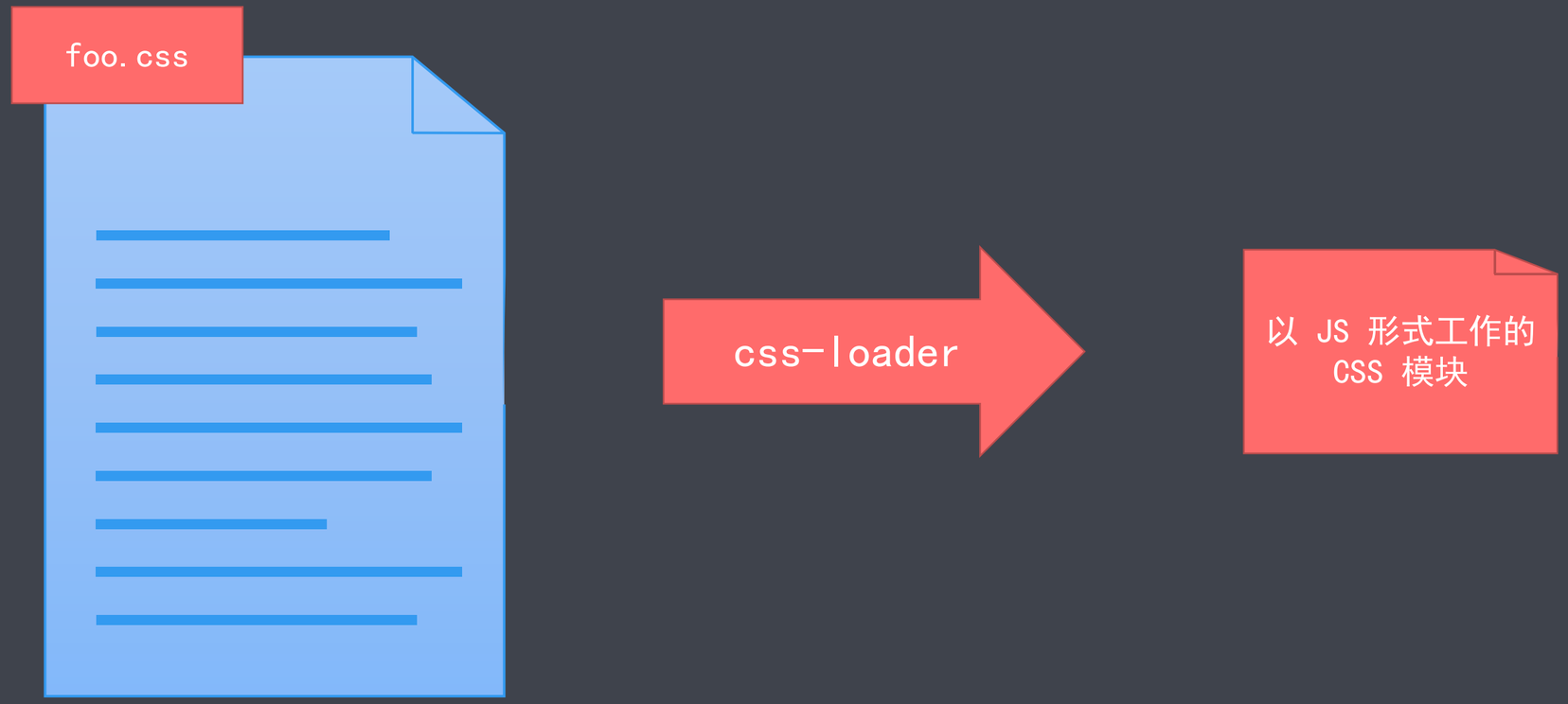
url-loader

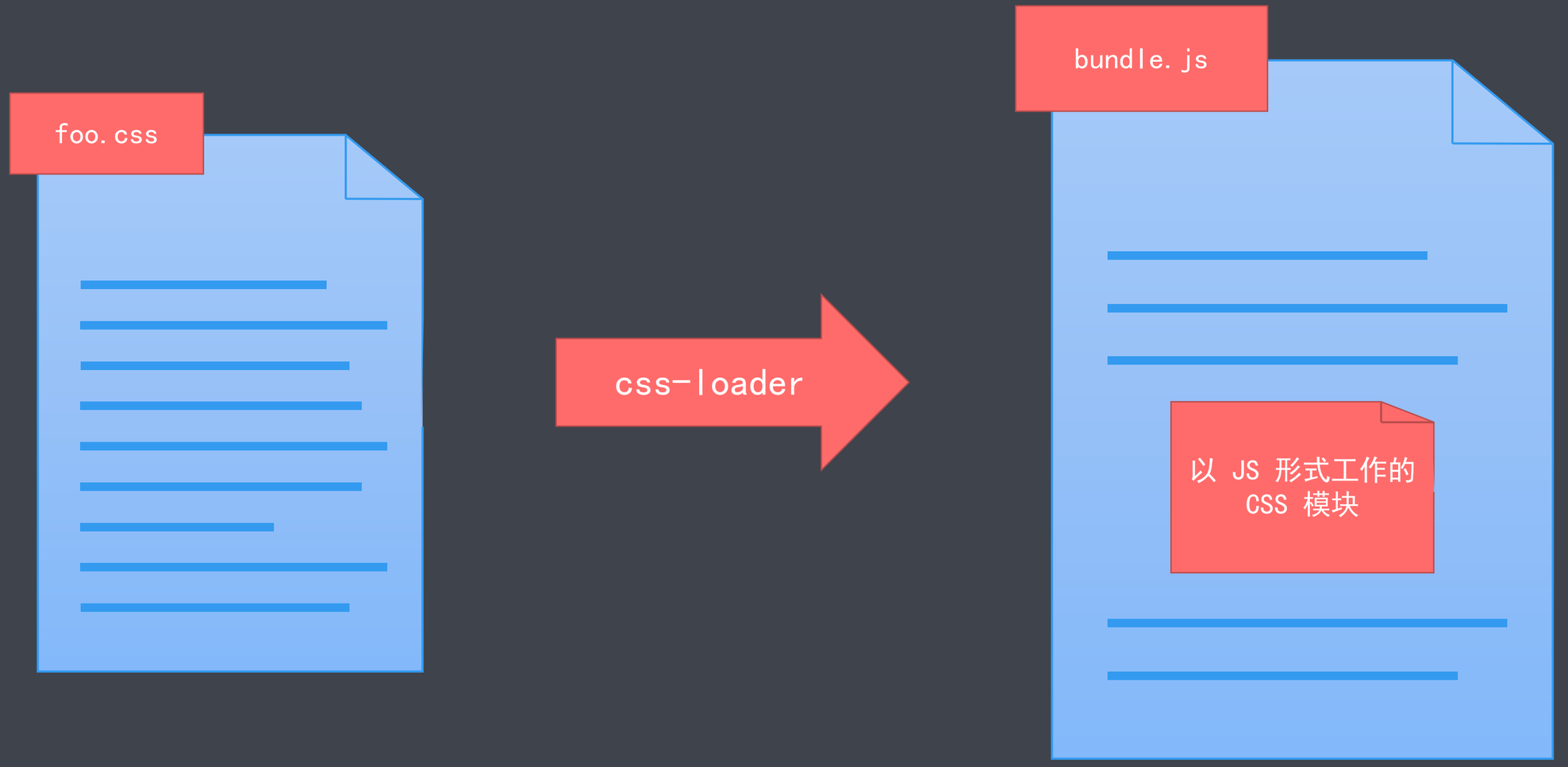
Webpack

常用加载器分类

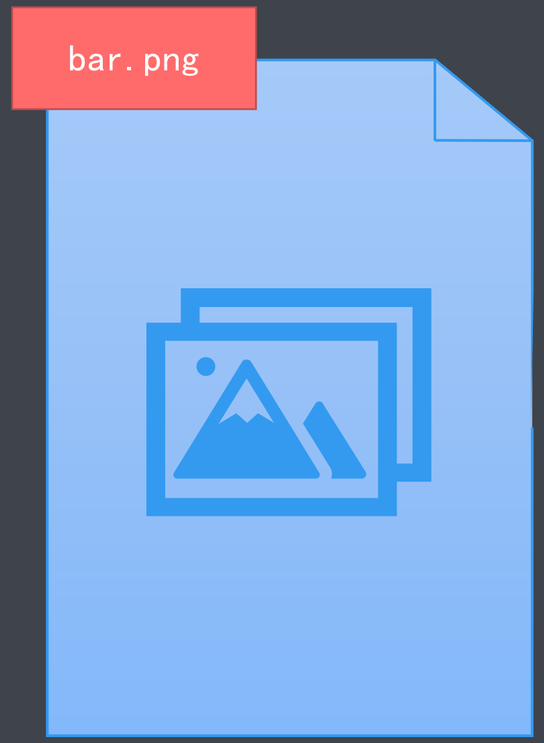
编译转换类

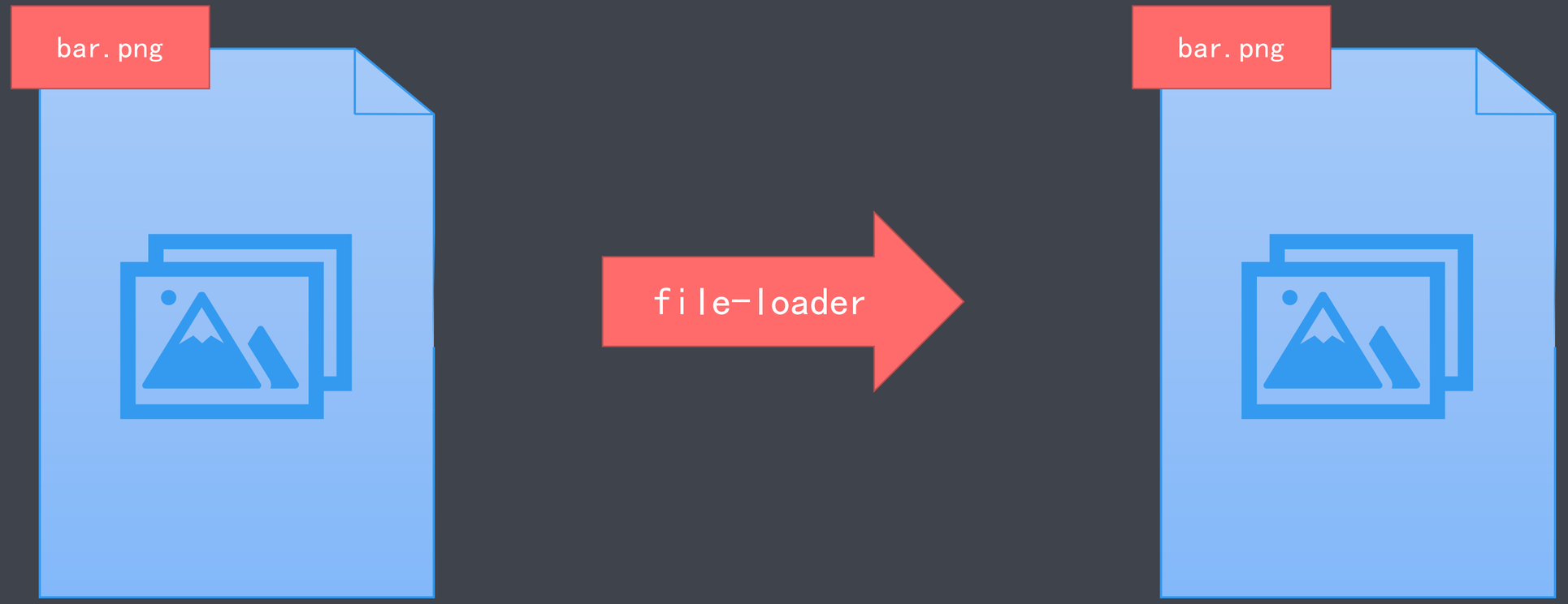

foo. css

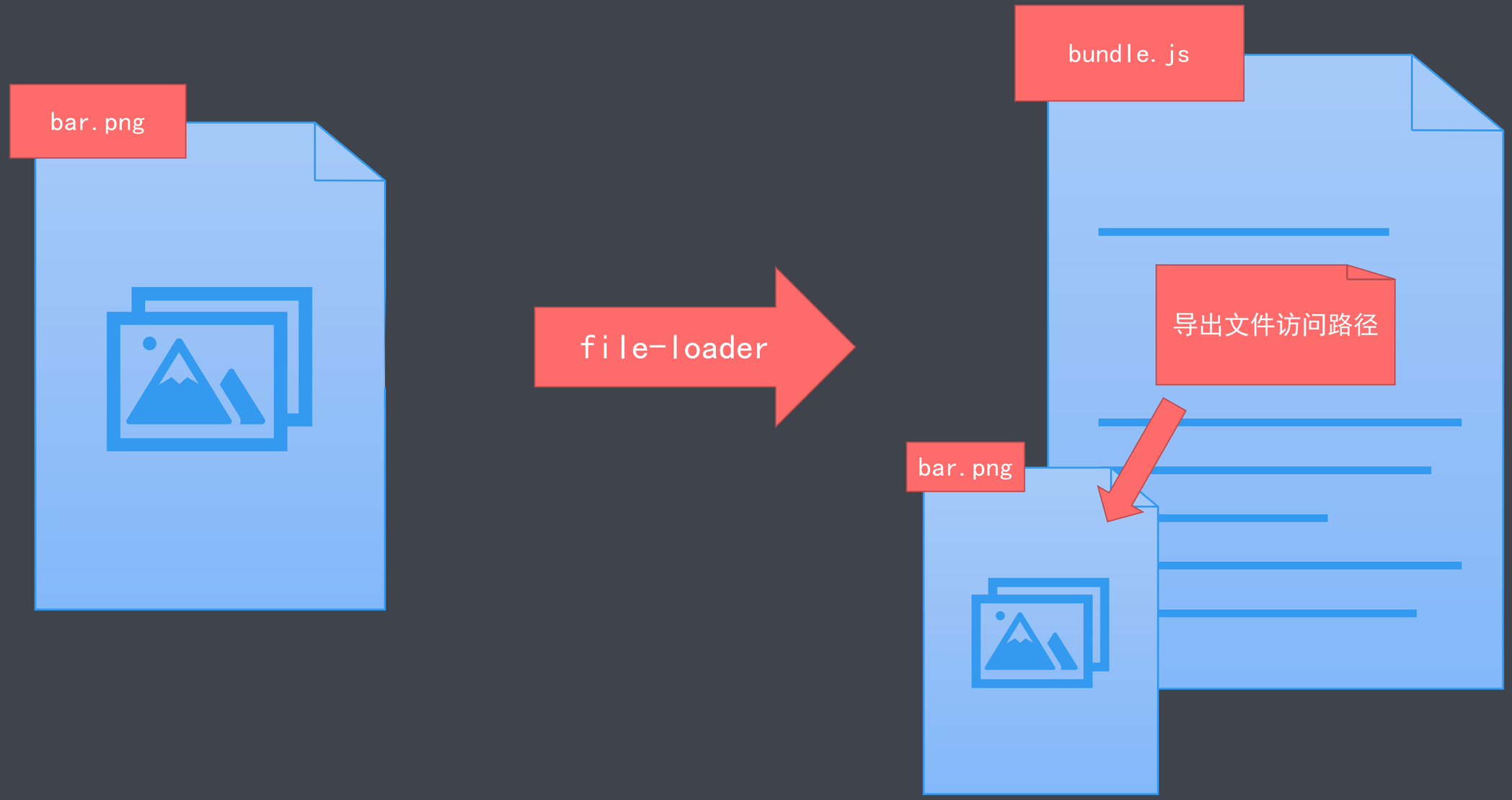

foo.css
css-loader
以 JS 形式工作的
CSS 模块


foo.css
css-loader
bundle.js
以 JS 形式工作的
CSS 模块

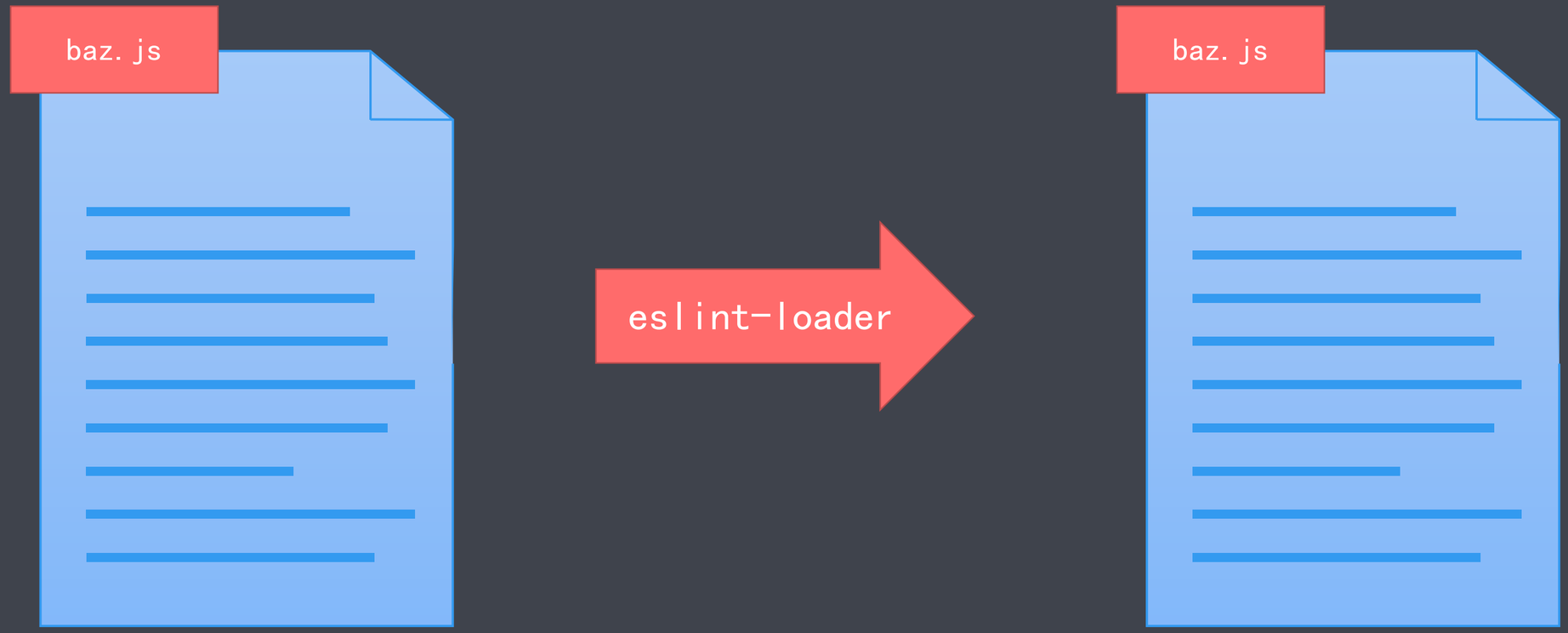
文件操作类

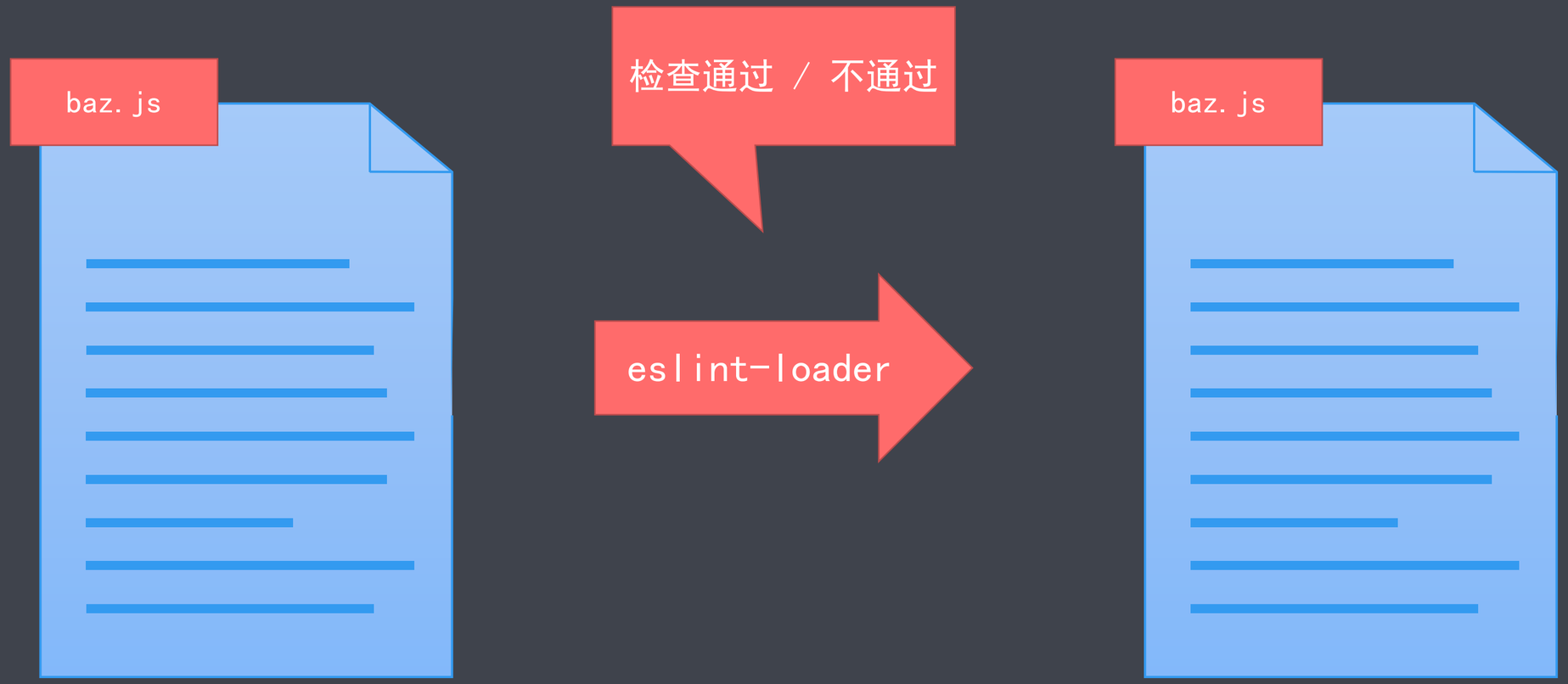

bar.png


bar.png
file-loader
bar.png


bar.png
file-loader
bundle.js
导出文件访问路径
bar.png

代码检查类


baz.js
eslint-loader
baz.js


baz.js
检查通过 / 不通过
eslint-loader
baz.js

Webpack 与 ES2015

Webpack 模块的加载方式

遵循 ES Modules 标准的 import 声明

```
import createHeading from './heading.js'
import icon from './icon.png'
import './main.css'
```

遵循 CommonJS 标准的 require 函数

```
import createHeading from './heading.js'
import icon from './icon.png'
import './main.css'

const heading = createHeading()
const img = new Image()
img.src = icon
document.body.append(heading)
document.body.append(img)
```

遵循 AMD 标准的 define 函数和 require 函数

```
define(['./heading.js', './icon.png', './main.css'], (createHeading, icon) => {
  const heading = createHeading.default()
  const img = new Image()
  img.src = icon
  document.body.append(heading)
  document.body.append(img)
})

require(['./heading.js', './icon.png', './main.css'], (createHeading, icon) =>
{ const heading = createHeading.default()
  const img = new Image()
  img.src = icon
  document.body.append(heading)
  document.body.append(img)
})
```

Webpack 模块加载方式

- 遵循 ES Modules 标准的 import 声明
- 遵循 CommonJS 标准的 require 函数
- 遵循 AMD 标准的 define 函数和 require 函数
- * 样式代码中的 @import 指令和 url 函数
- * HTML 代码中图片标签的 src 属性

Webpack

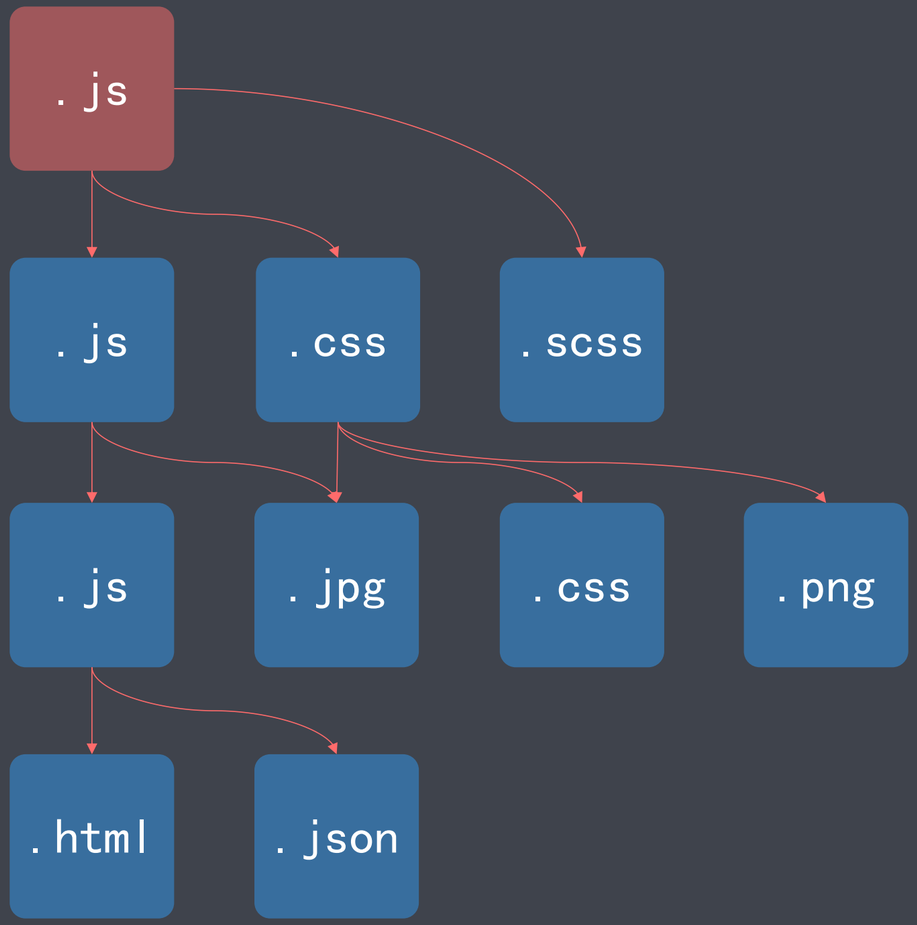
核心工作原理

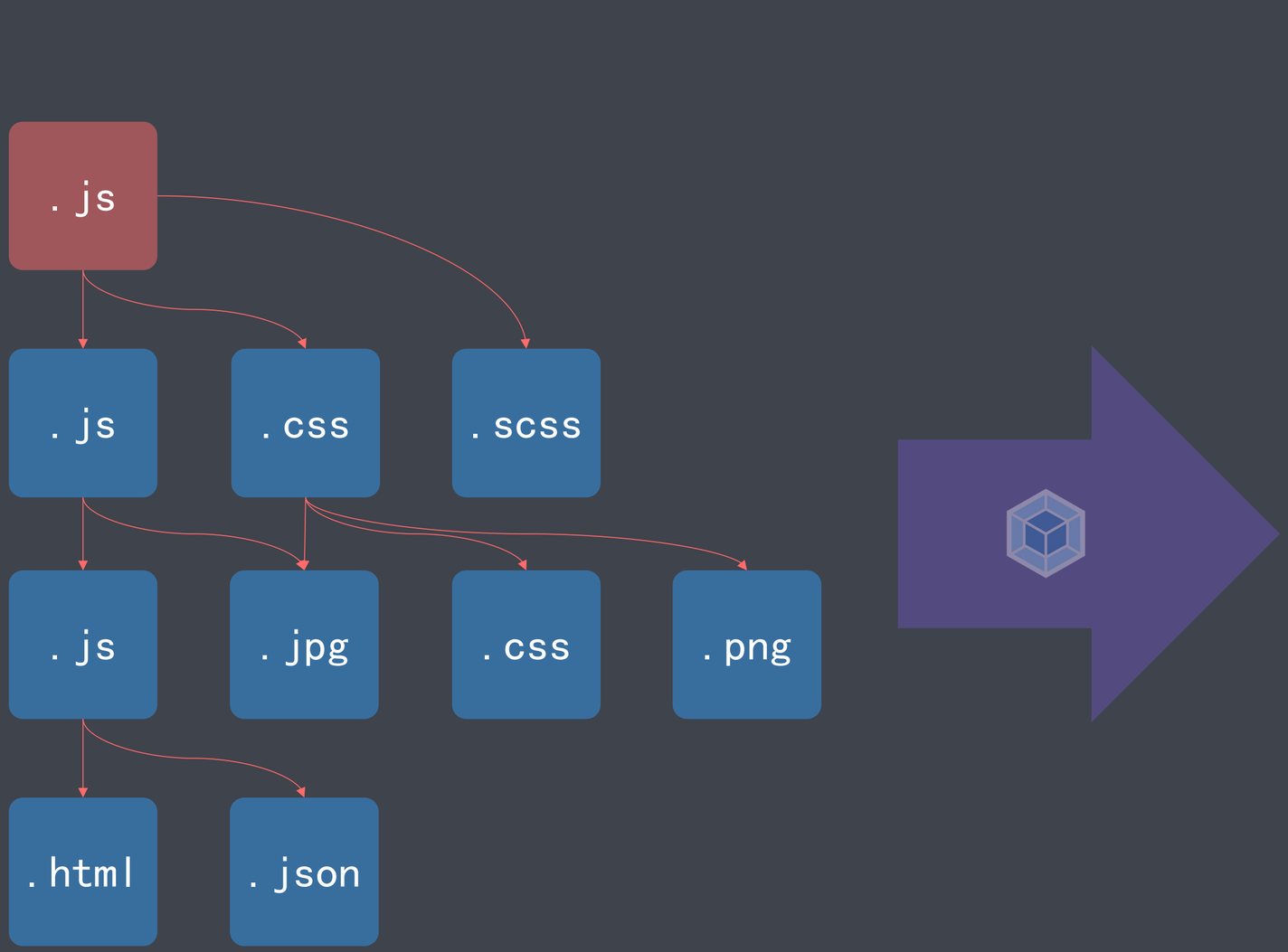

bundle your assets
.js
.hbs
.cjs
.sass
.js
.sass
.sass
.png
.jpg
.js
.css
.jpg
.png
STATIC ASSETS
MODULES WITH DEPENDENCIES

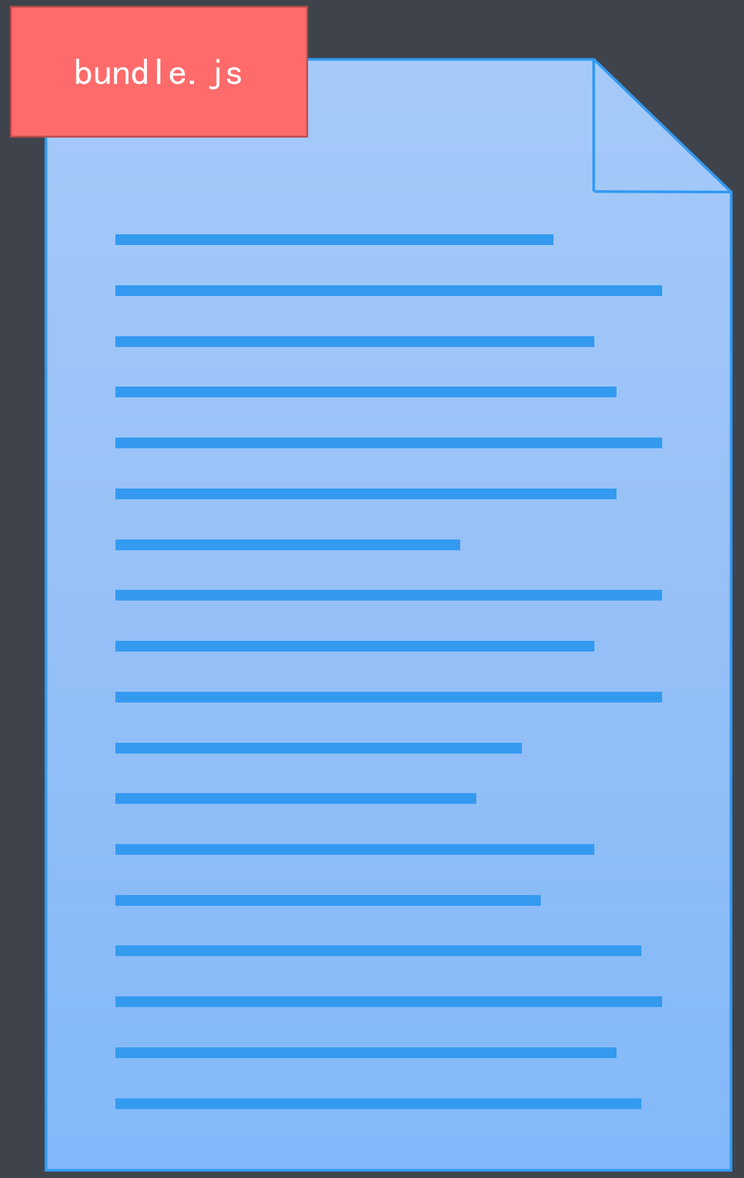
Webpack

.js	.html	.css	.png	.json
.js	.js	.jpg	.css	.scss


打包入口
.js
.html
.css
.png
.json
.js
.js
.jpg
.css
.scss


.js
.js
.css
.scss
.js
.jpg
.css
.png
.html
.json



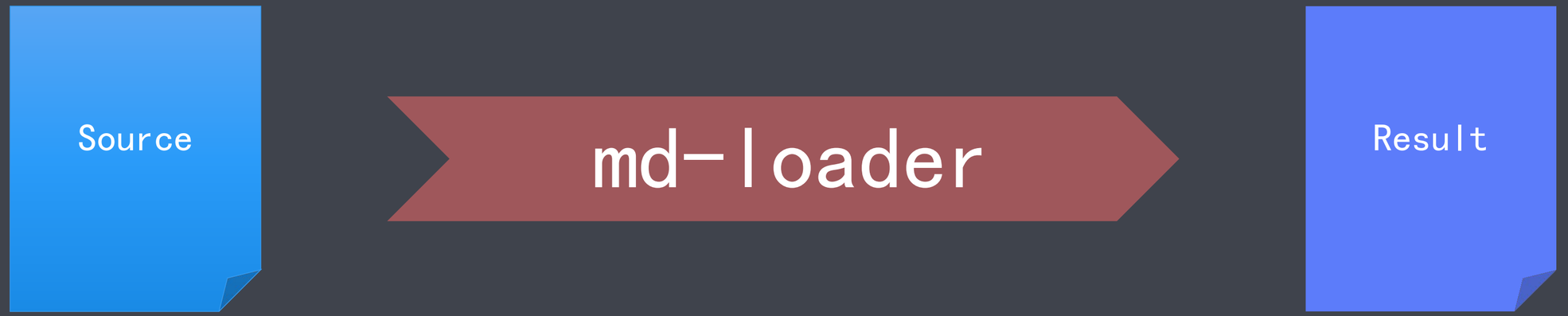

bundle.js

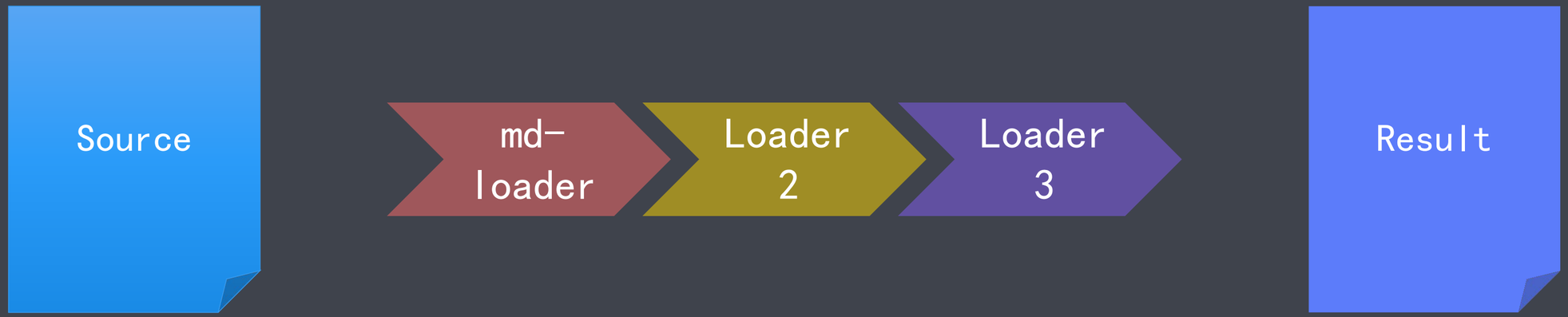
Loader 机制是 Webpack 的核心

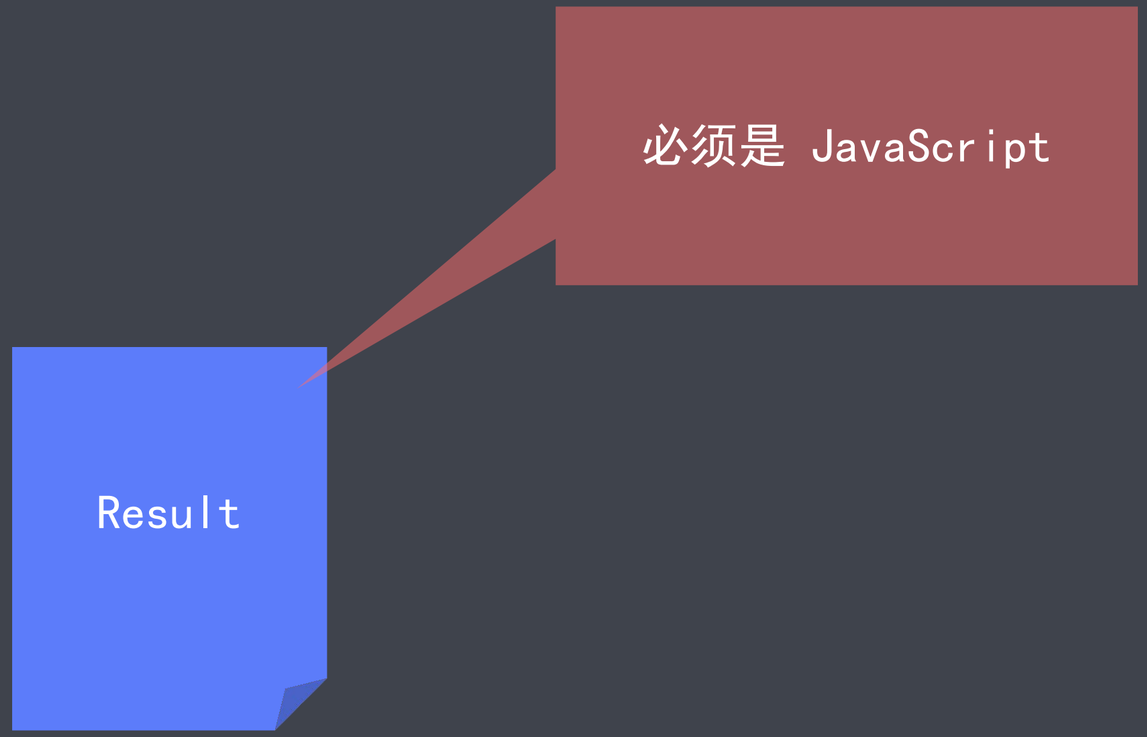
Webpack

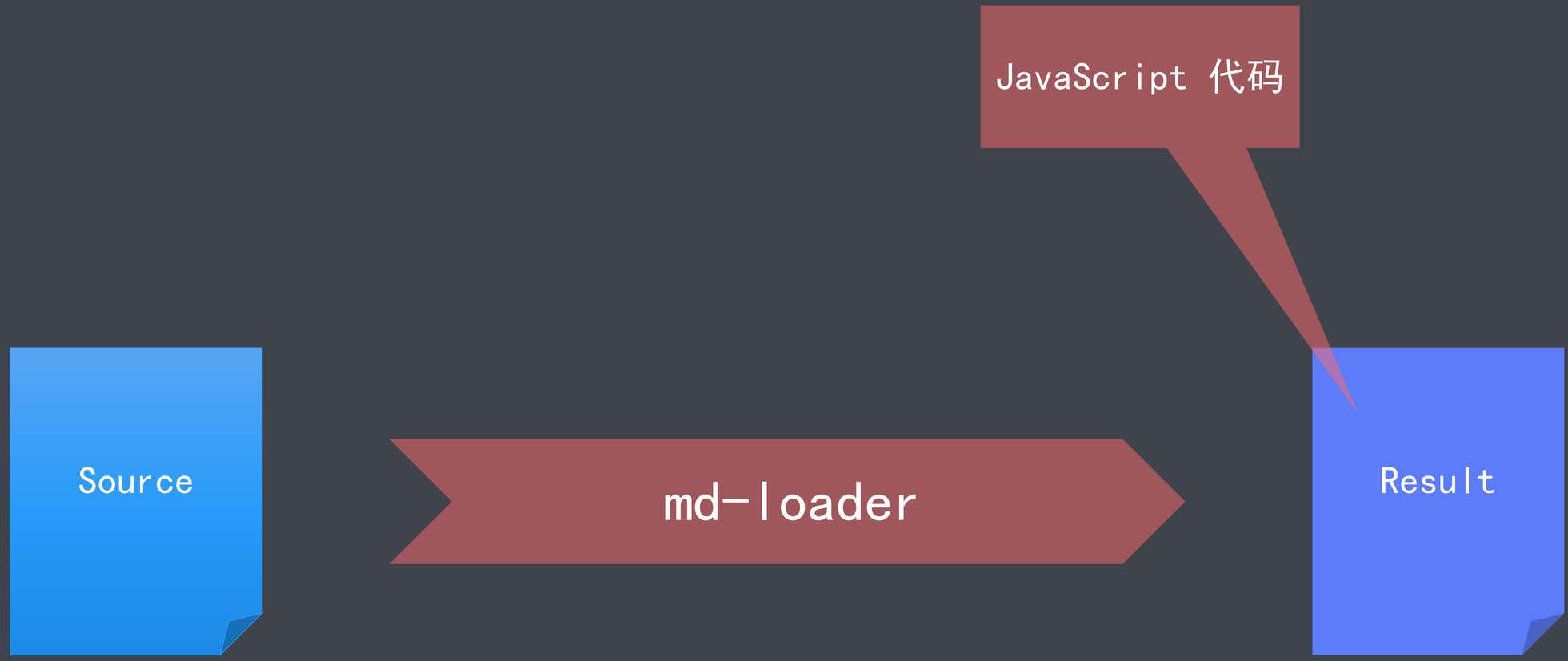
Loader 的工作原理

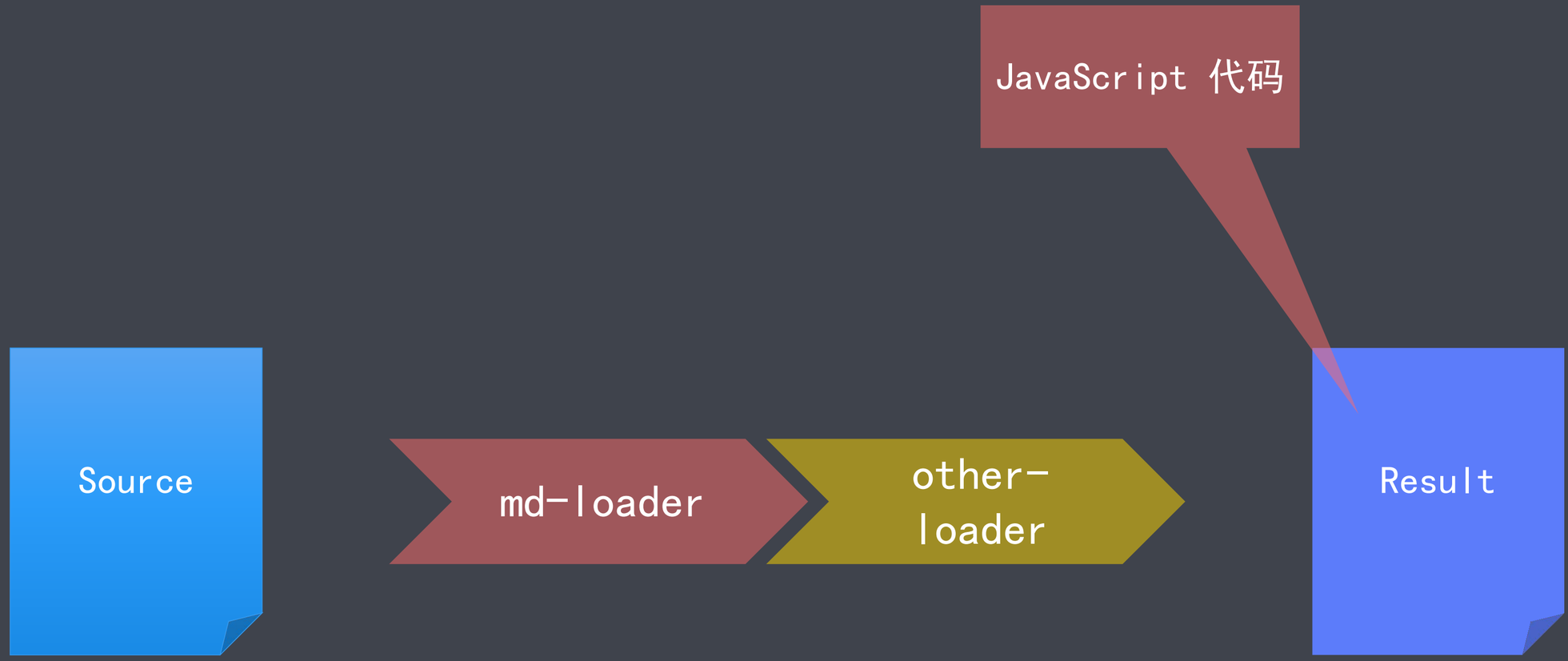
markdown-loader


Source
md-loader
Result


Source
md-loader
Loader 2
Loader 3
Result


必须是 JavaScript
Result


JavaScript 代码
Source
md-loader
Result

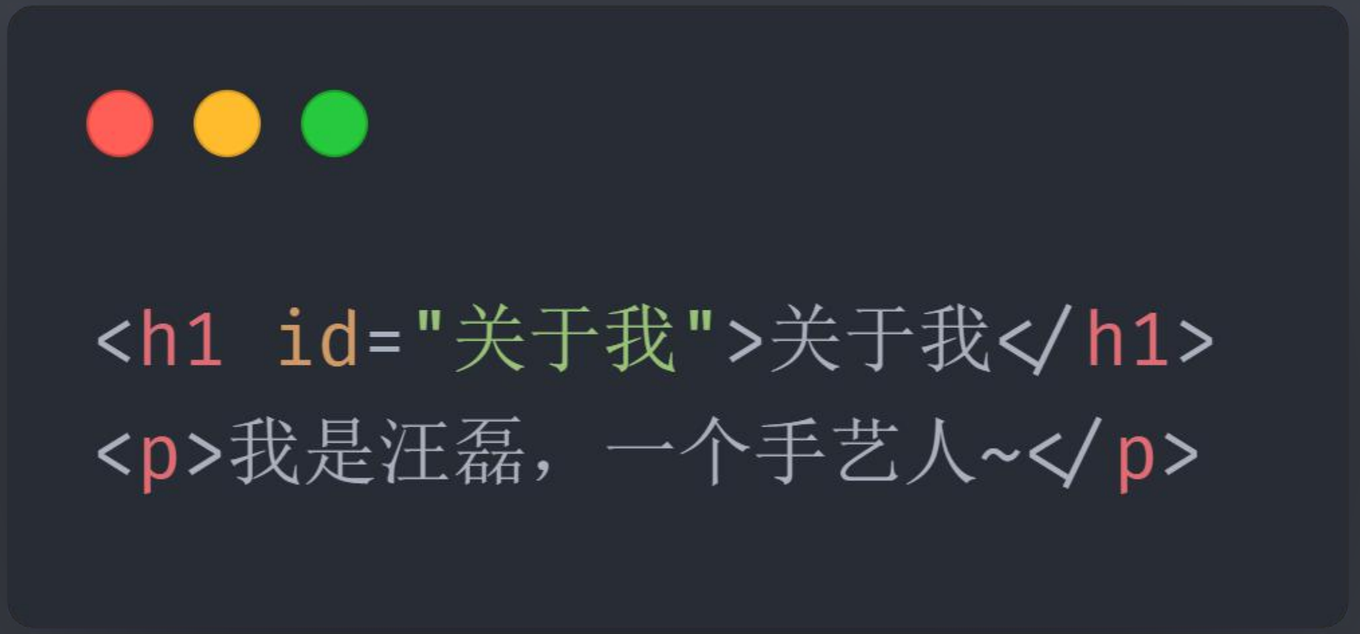

JavaScript 代码
Source
md-loader
other-
loader
Result

```
module.exports = "<h1 id=\"关于我\">关于我</h1>\n<p>我是汪磊，一个手艺人~</p>\n"
```

```
module.exports = "<h1 id="关于我">关于我</h1>
<p>我是汪磊，一个手艺人~</p>"
```

关于我

我是汪磊，一个手艺人~


<h1 id="关于我">关于我</h1>
我是汪磊，一个手艺人~

```
module.exports = "<h1 id=\"关于我\">关于我</h1>\n<p>我是汪磊，一个手艺人~</p>\n";
```

Webpack 插件机制

Webpack 常用插件

clean-webpack-plugin

Webpack 常用插件

html-webpack-plugin

Webpack 常用插件

html-webpack-plugin 选项

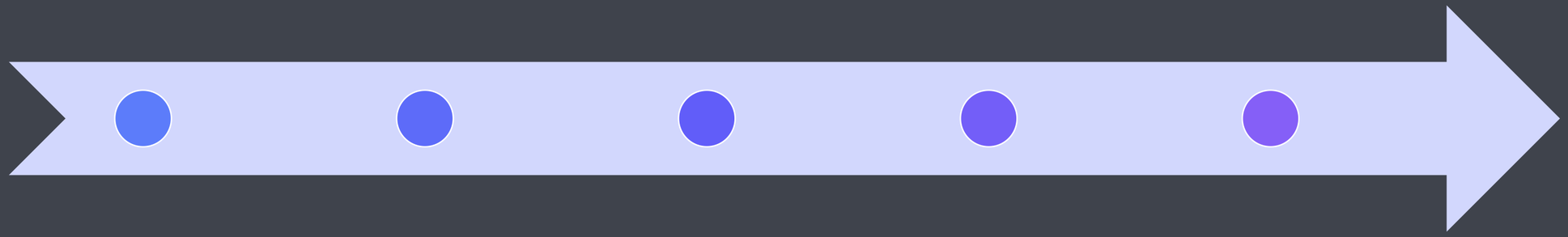
Webpack 常用插件

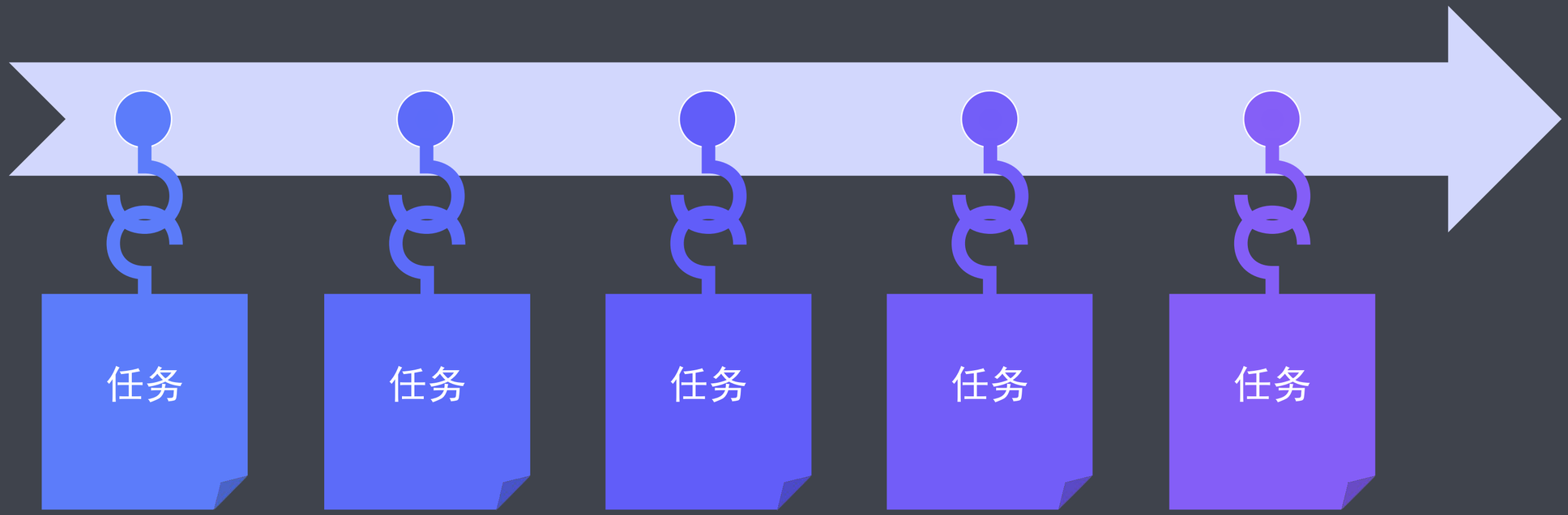
html-webpack-plugin 多实例

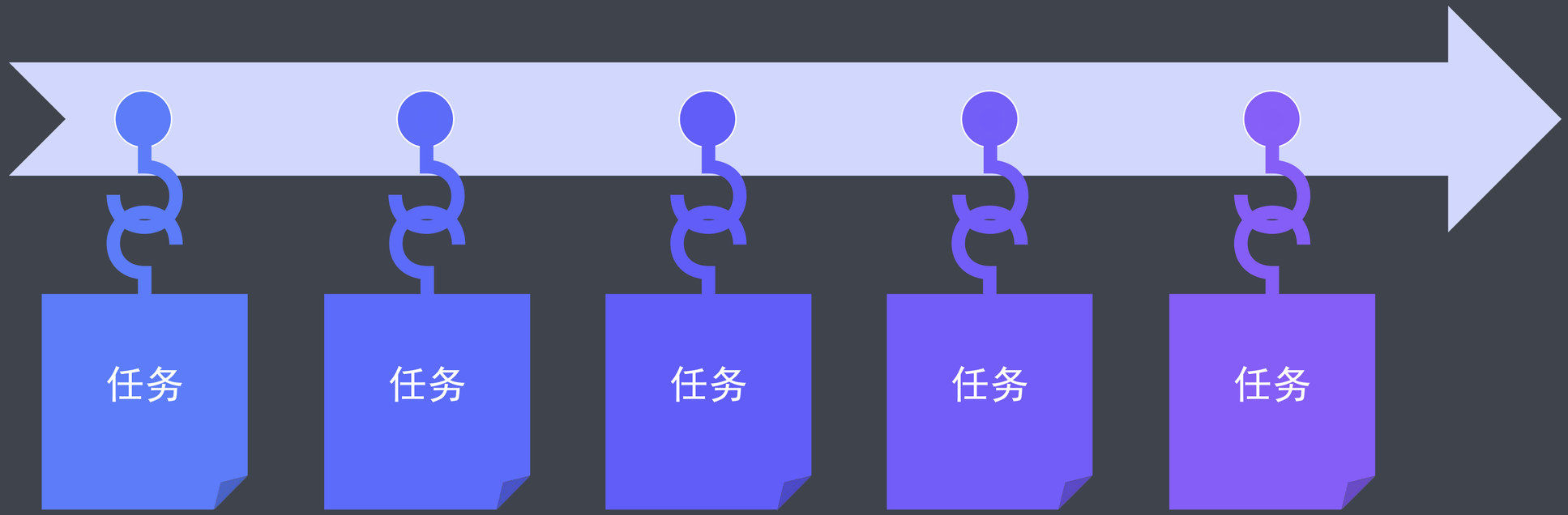
Webpack 常用插件

copy-webpack-plugin & 总结

开发一个插件




任务
任务
任务
任务
任务


任务
任务
任务
任务
任务

Webpack 开发体验

Webpack 增强开发体验

自动编译

Webpack 增强开发体验

自动刷新浏览器

Webpack Dev Server

Webpack Dev Server

静态资源访问

Webpack Dev Server

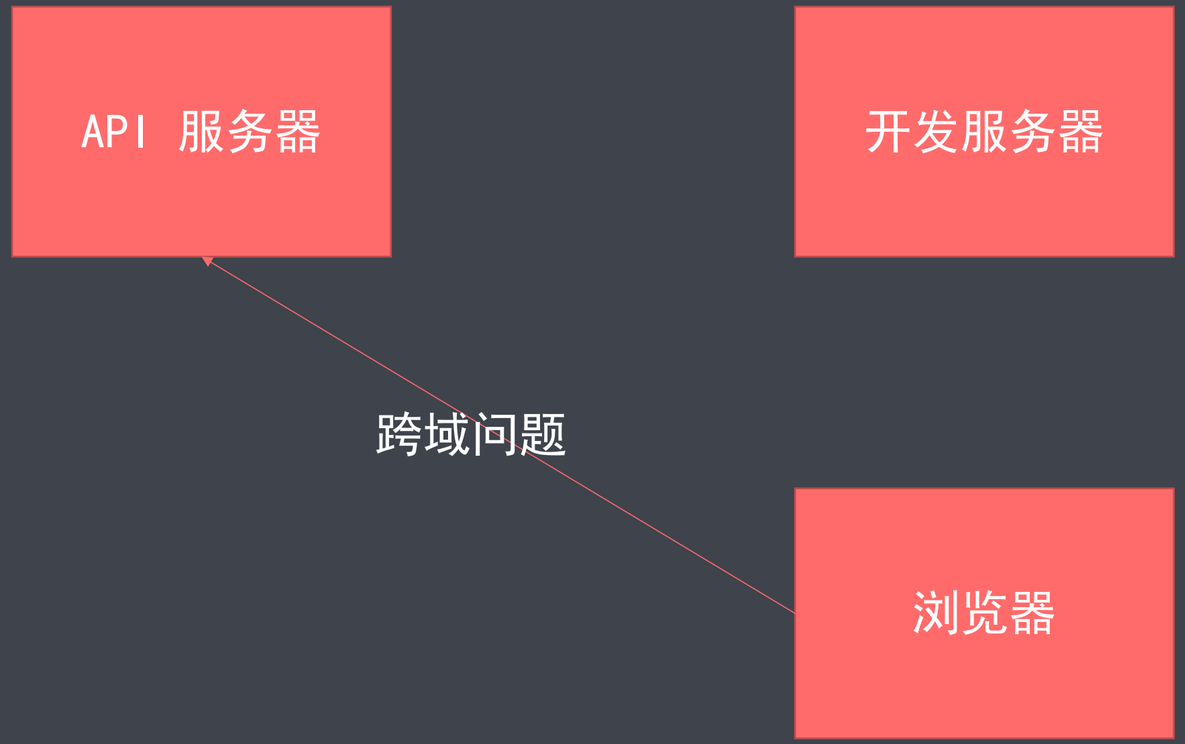
代理 API 服务

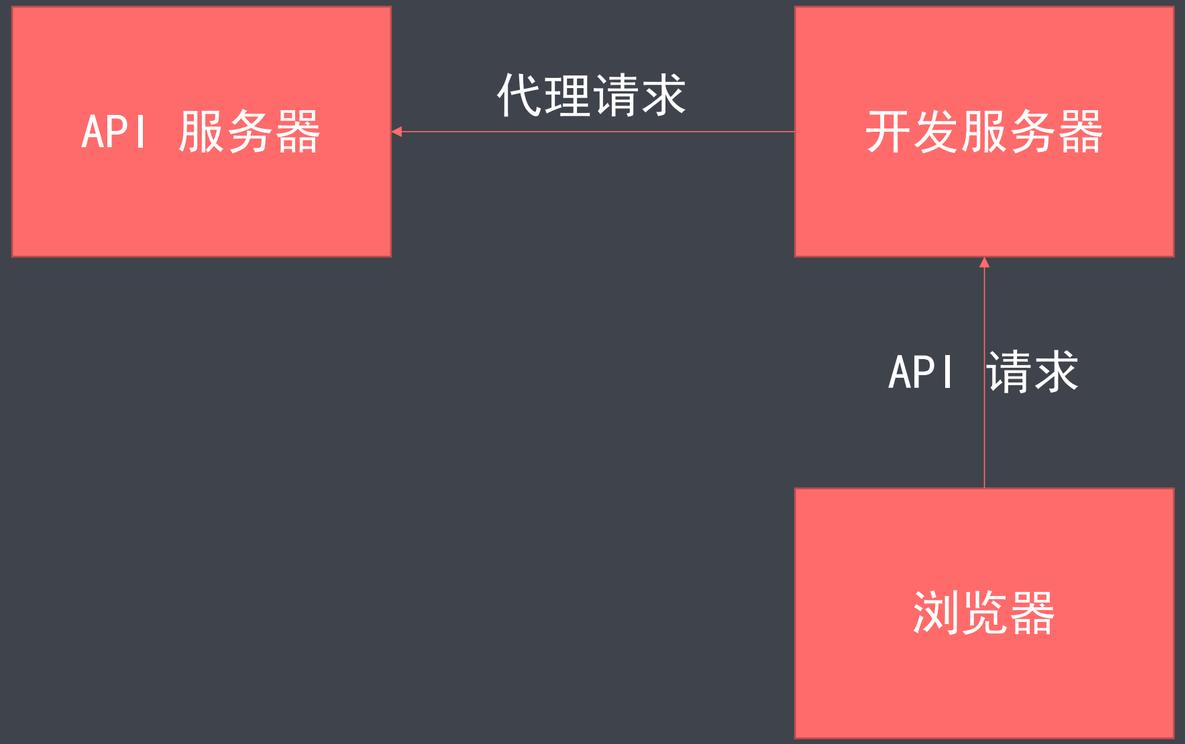
https://www.example.com/index.html

https://www.example.com/api/users

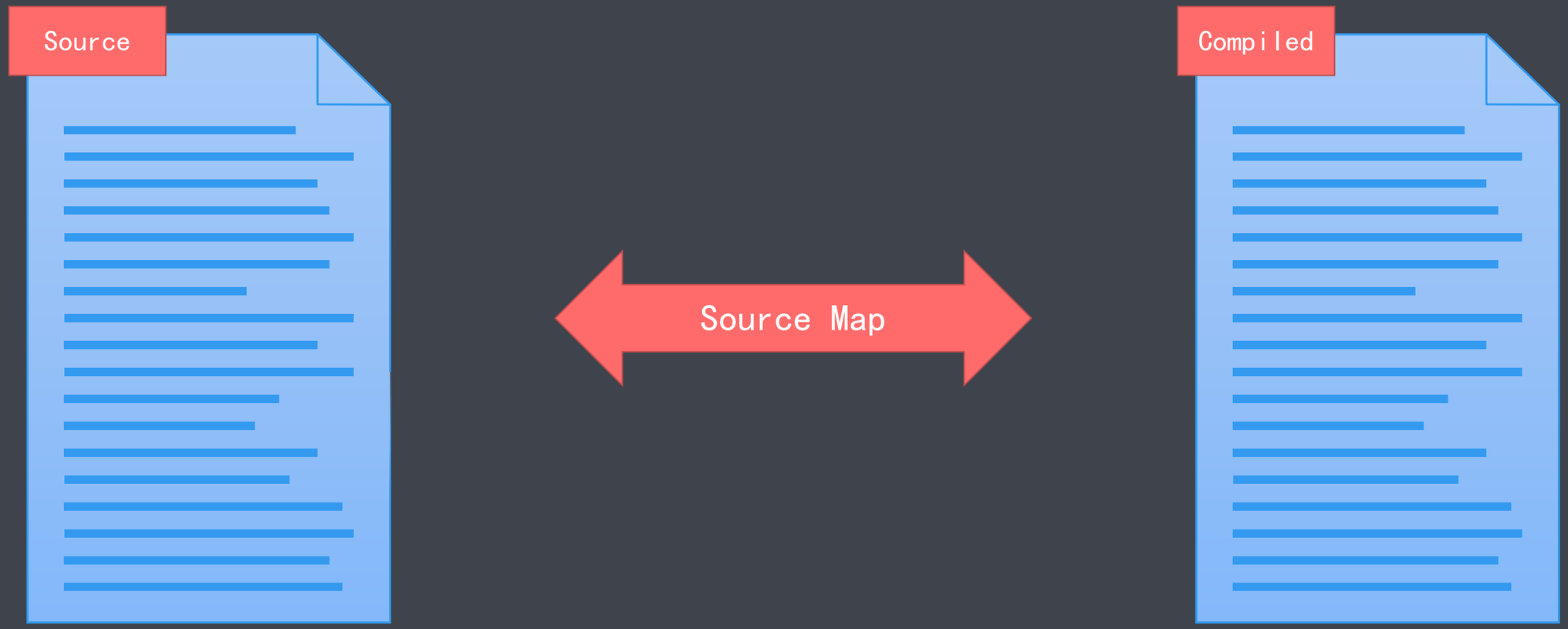
http://localhost:8080/index.html

https://www.example.com/api/users


API 服务器
开发服务器
跨域问题
浏览器


API 服务器
代理请求
开发服务器
API 请求
浏览器

Source Map


Source
Source Map
Compiled

Webpack 配置 Source Map

Webpack 配置 Source Map

eval 模式下的 Source Map

devtool	build	rebuild	production	quality
(none)	fastest	fastest	yes	bundled code
eval	fastest	fastest	no	generated code
cheap-eval-source-map	fast	faster	no	transformed code (lines only)
cheap-module-eval-source-map	slow	faster	no	original source (lines only)
eval-source-map	slowest	fast	no	original source
cheap-source-map	fast	slow	yes	transformed code (lines only)
cheap-module-source-map	slow	slower	yes	original source (lines only)
inline-cheap-source-map	fast	slow	no	transformed code (lines only)
inline-cheap-module-source-map	slow	slower	no	original source (lines only)
source-map	slowest	slowest	yes	original source
inline-source-map	slowest	slowest	no	original source
hidden-source-map	slowest	slowest	yes	original source
nosources-source-map	slowest	slowest	yes	without source content

不同 devtool 之间的差异

准备工作

不同 devtool 之间的差异

具体对比

选择合适的 Source Map

自动刷新的问题

HMR 介绍

开启 HMR

HMR 解惑

HMR API

JS 模块热替换

图片模块热替换

HMR 注意事项

生产环境优化

不同环境下的配置

不同环境的配置文件

DefinePlugin

Tree Shaking

Tree Shaking 使用

合并模块函数

Scope Hoisting

Tree Shaking & Babel

sideEffects

副作用

sideEffects 注意

Code Splitting

分包 / 代码分割

多入口打包

Multi Entry

提取公共模块

Split Chunks

动态导入

Dynamic Imports

魔法注释

Magic Comments

MiniCssExtractPlugin

提取 CSS 文件

OptimizeCssAssetsWebpackPlugin

压缩 CSS

输出文件名 Hash

substitutions