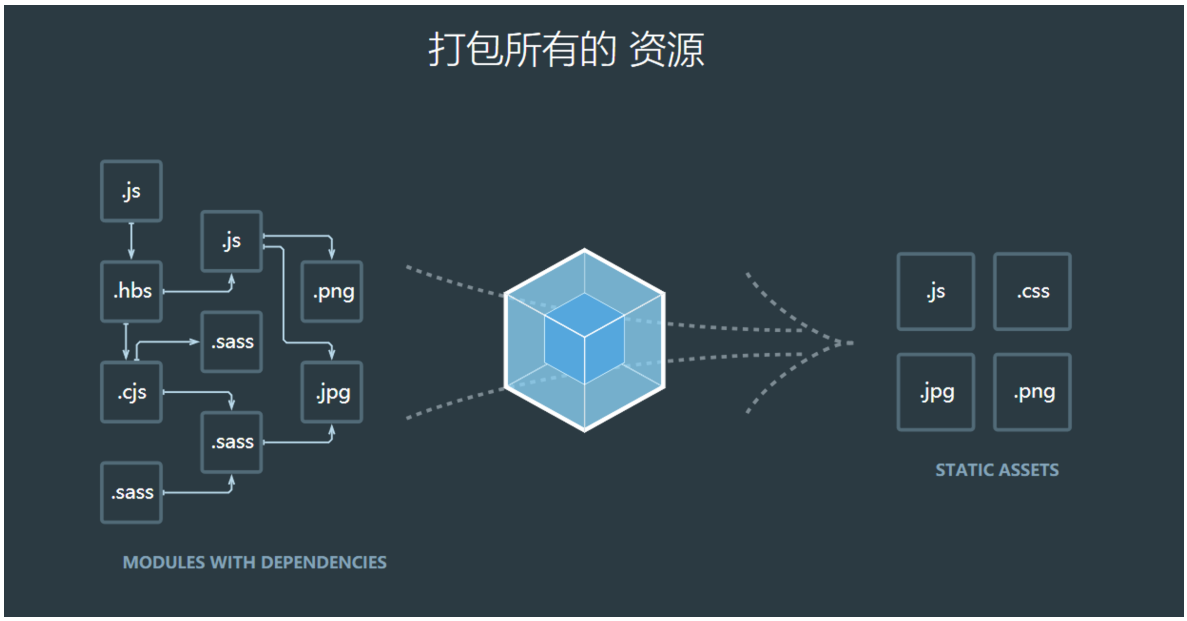
webpack

webpack 是一个项目构建工具，也称之为“模块打包器”。



介绍

webapck 的功能：

1. 压缩代码
2. 合并代码
3. 代码转换（当有一些代码浏览器不能识别的时候，例如 JS 的最新版本代码，就可以通过 webpack 转换成浏览器能识别的代码）

核心

1. entry：入口；
2. output：出口（输出）；
3. loaders：模块转换器；
4. plugins：扩展插件；

一、项目初始化

- **1. 生成 `package.json` 文件**

```
npm init -y
```

2. 下载 webpack

```
npm i webpack webpack-cli -D
```

二、文件打包

1. 创建文件

在项目根目录创建一个 `src/index.js` 文件，在该文件随意写点代码，例如：

```
console.log('hello');
console.log('world');
```

2. 执行打包命令

```
npx webpack --mode=development
```

`mode` 用于指定打包模式，development 表示“开发模式”，production 表示“生产模式”。

以上打包操作，采用的都是 webpack 的默认配置，例如：默认入口文件为： `src/index.js` ，默认出口文件为： `dist/main.js` 。

三、自定义配置

所有的自定义配置都在配置文件： `webpack.config.js` 中进行设置，如果没有该文件，则需要在项目根目录手动创建。

该配置文件的基本格式如下：

```
module.exports = {
    // 项目入口文件的配置
    entry: {},
     // 项目出口文件的配置
    output: {},
    // 项目构建环境：production（生产环境）、development（开发环境）、none
    mode: 'development',
    // loaders 的配置
    module: {},
```

```
    // 插件的配置
    plugins: [],
    // 开发服务配置
    devServer: {}
}
```

四、常用 plugins

所有 plugins 的使用，都需要先下载对应的依赖包，然后再在配置文件中进行配置。

1. html-webpack-plugin

- 下载：

```
npm i html-webpack-plugin -D
```

- 配置

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

plugins: [
    new HtmlWebpackPlugin({
        template: './src/index.html'
    })
]
```

- 执行打包命令 `npx webpack` 。

2. clean-webpack-plugin

自动删除打包目录下的多余的内容。

- 下载

```
npm i clean-webpack-plugin -D
```

- 配置

```
const { CleanWebpackPlugin } = require('clean-webpack-plugin');

plugins: [
    new CleanWebpackPlugin()
],
```

五、常用 loaders

所有的 loaders 在使用前，也需要先下载，然后在配置文件的 `module` 属性中进行配置。

注：loaders 不需要手动引入，直接使用即可。

1. 打包 css

1. 新建一个 `src/css/index.css` 文件，随意设置一些 css 代码，例如：

```
body {
    background-color: black;
}
```

2. 在入口 `index.js` 文件中引入 css：

```
import './css/index.css';
```

3. 下载 loader

```
npm i style-loader css-loader -D
```

4. 配置

```
rules: [
    {
        test: /\.css$/,    // 文件名匹配
        exclude: /node_modules/,    // 不需要解析的文件名匹配
        use: ['style-loader', 'css-loader']    // 使用的 loader
    }
]
```

5. 执行打包命令

2. 打包 less

1. 新建一个 `src/css/index.less` 文件，随意设置一些 less 代码，例如：

```
body {
    h1 {
        color: white;
    }
}
```

2. 引入

```
import './css/index.less';
```

3. 下载 loader

```
npm i less less-loader -D
```

4. 配置

```
 rules: [
     {
         test: /\.less$/,
         exclude: /node_modules/,
         use: ['style-loader', 'css-loader', 'less-loader']
     }
 ]
```

5. 执行打包命令

3. 打包 css 中的图片

1. 在 `src/index.css` 中引入背景图片：

```
body {
    background-color: black;
    background-image: url('../images/pic.JPG');
}
```

默认情况下，webpack 不能对该图片样式进行打包处理。

2. 下载 loader

```
npm i url-loader file-loader -D
```

3. 配置

```
rules: [
    {
        test: /\.(png|jpg|jpeg|gif|webp|svg)$/i,
        exclude: /node_modules/,
        use: [{
            loader: 'url-loader',
            options: {
                limit: 1024 * 8, // 8kb 以下的图片采用 base64 的编码方式处理图片
                outputPath: 'images'   // 设置图片打包后的路径
            }
        }]
    }
]
```

4. 执行打包命令

4. 打包 html 中的图片

1. 在 `src/index.html` 中加入以下代码：

```
<img src="./images/pic.png"/>
```

2. 下载 loader

```
npm i html-loader -D
```

六、入口和出口

1. 单出入口

```
const path = require('path');
module.exports = {
    // 项目入口文件的配置
    entry: {
        entry1: './src/index.js',
    },
    // 项目出口文件的配置
    output: {
        path: path.resolve(__dirname, 'dist'),
        filename: 'bundle.js'    // bundle.js 为出口文件名称，可自定义
    }
}
```

- **2. 多出入口**

```
const path = require('path');
module.exports = {
    // 项目入口文件的配置
    entry: {
        entry1: './src/index.js',
        entry2: './src/app.js'
    },
    // 项目出口文件的配置
    output: {
        path: path.resolve(__dirname, 'dist'),
        filename: '[name].js'
    },
}
```