

1:先安装插件:

在命令行中: `cnpm i html-webpack-plugin -D`

2:在`webpack.config.js`中导入这个插件

```
webpack.config.js - webpack-study - Visual Studio Code
index.html JS main.js webpack.config.js x package.json
1  const path = require('path')
2  // 启用热更新的 第2步
3  const webpack = require('webpack')
4  // 导入在内存中生成 HTML 页面的 插件
5  // 只要是插件,都一定要 放到 plugins 节点中去
6  // 这个插件的两个作用:
7  // 1. 自动在内存中根据指定页面生成一个内存的页面
8  // 2. 自动,把打包好的 bundle.js 追加到页面中去
9  const htmlWebpackPlugin = require('html-webpack-plugin')
```

3:

```
webpack.config.js - webpack-study - Visual Studio Code
index.html JS main.js webpack.config.js x package.json
16  devServer: { // 这是配置 dev-server 命令参数的第二种形式,相对来说,这种方式麻烦一些
17    // --open --port 3000 --contentBase src --hot
18    open: true, // 自动打开浏览器
19    port: 3000, // 设置启动时候的运行端口
20    contentBase: 'src', // 指定托管的根目录
21    hot: true // 启用热更新 的 第1步
22  },
23  plugins: [ // 配置插件的节点
24    new webpack.HotModuleReplacementPlugin(), // new 一个热更新的 模块对象, 这是 启用热更新
    的第 3 步
25    new htmlWebpackPlugin({ // 创建一个 在内存中 生成 HTML 页面的插件
26      template: path.join(__dirname, './src/index.html'), // 指定 模板页面,将来会根据指定
    的页面路径,去生成内存中的 页面
27      filename: 'index.html' // 指定生成的页面的名称
28    })
29  ]
```

```
15
16  <!-- 当使用 html-webpack-plugin 之后,我们不再需要手动处理 bundle.js 的引用路径了,因为 这个
    插件,已经帮我们自动 创建了一个 合适的 script , 并且,引用了 正确的路径 -->
17  <!-- <script src="/bundle.js"></script> -->
18
```