

17 | 项目打包与优化：前端必备的Webpack打包配置详解

2023-05-31 Barry 来自北京

《Python实战·从0到1搭建直播视频平台》



你好，我是 Barry。

前端实战篇即将步入尾声。之前的课程中，整体的前端项目开发已经告一段落，在本地即可访问项目的完整功能，但是，如果想让更多的人能在浏览器直接访问我们的平台，又该如何实现呢？

shickey.com转载分享

为了让更多的用户访问视频平台界面。这节课我们就一起来学习一下，如何将我们的前端代码打包上线。

shickey.com转载分享

想要高效地打包代码，我们需要用到模块管理工具 Webpack。这款工具你应该不陌生，因为我们在最初构建项目的时候就用过它。

只有全面了解 Webpack，你才能在后续项目部署的时候轻车熟路，快速实现项目管理和项目打包。而且，Webpack 也是在技术面试中也是个高频考点，难度系数也比较高。不过不要有

心理负担，耐心学完这节课，你就能找到掌握 Webpack 的诀窍了。

Webpack 初识

Webpack 是一种模块打包工具，它可以将多个 JavaScript 模块打包成一个或多个 JavaScript 文件，从而减少网页加载时的体积。Webpack 主要用于构建大型的、复杂的应用程序，比如游戏、后端服务等。

Webpack 的工作原理是将源代码分割成一个个模块，然后使用 loader 将这些模块打包成一个或多个 JavaScript 文件。每个模块都可以看作一个独立的文件，它们之间通过依赖关系相互联系。

在 Webpack 当中，常用的模块打包器有 Babel、Parcel、Compression 等。除去模块打包，它们还可以对代码进行压缩、优化等处理。

另外，Webpack 还支持多种插件，这些插件可以扩展其功能，比如支持多种 loaders、支持代码分割、支持自动化测试等。

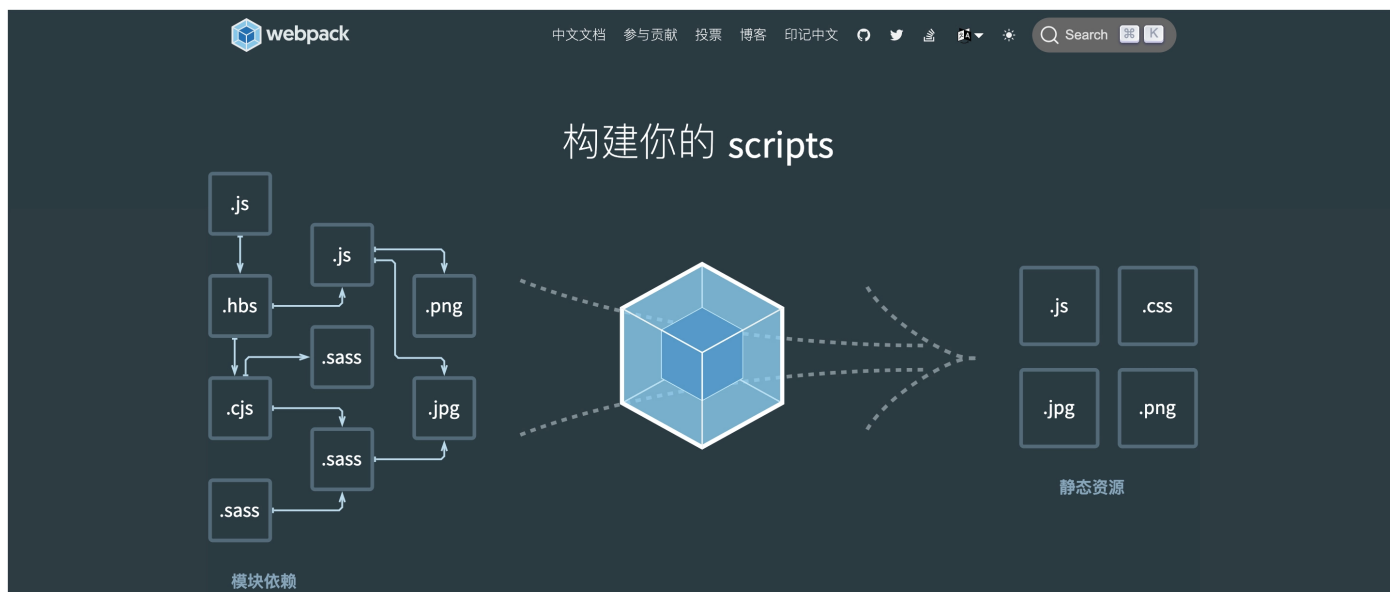
分析了这么多，不难看出 Webpack 是一种非常强大的模块打包工具，它可以帮助开发者快速构建高质量的网页应用程序。

Webpack 详解

了解了 Webpack 的作用之后呢，我们还需要深入了解 Webpack 的核心内容，为后续在项目中应用它打好基础。当然如果你想更全面地了解这个工具，也可以在课后直接访问

🔗 [Webpack 的中文官网](#)。

shikey.com转载分享



首先我们需要详细了解 Webpack 的**核心概念**。这些概念对应的官网链接我也为你做了整理，方便你课后拓展学习。

🔗 入口

🔗 输出

🔗 loader

🔗 插件

🔗 模式

🔗 浏览器兼容性

🔗 环境

shikey.com转载分享


接下来我们就依次来学习前面列出的每一种属性，了解其核心作用和使用技巧。

入口 (entry) shikey.com转载分享

Webpack 的入口起点是指 Webpack 在开始打包之前所加载的第一个模块或文件。在 Webpack 中，入口起点通常是 entry (或 module) 属性。Webpack 会从入口起点开始打包，直到遇到第一个不允许打包的模块或文件为止。

入口起点的设置非常重要，因为它会影响 Webpack 打包的策略。通常情况下，Webpack 会使用入口起点来决定哪些模块需要打包、哪些模块应该被忽略。如果入口起点设置得不合理，可能会导致打包结果不符合预期，甚至导致项目无法正常运行。

在项目当中我们默认的入口是 main.js，具体的配置路径在项目的 build 文件夹下面的 webpack.base.conf.js 文件内。你可以结合后面的具体代码案例来加深理解。

 复制代码

```
1 module.exports = {  
2   context: path.resolve(__dirname, '../'),  
3   entry: {  
4     app: './src/main.js'  
5   }  
6   //其中还有其他的一些应用  
7 }
```

因为我们本身使用的是 Vue-cli 脚手架，所以这些代码不需要你单独再写一遍，搭建过程就自动配置好了。不过你一定要对它的实现原理非常熟悉，这样需要根据项目需求调整 Webpack 配置的时候，你才能够快速上手。

了解了入口配置之后，我们紧接着来看一下输出的配置，这是我们最终生成打包文件的核心配置。

输出 (output)

Webpack 的输出 (output) 是指打包后生成的可执行文件或资源文件的位置和文件名等相关信息。


在 Webpack 的配置当中，我们可以通过 output 属性来指定输出位置，配置打包后的文件输出路径、文件名和文件描述等，output 可以接受一个对象，这个对象包含后面这些属性。

filename: 指定打包后生成的文件名，可以包含占位符，如[name].[hash].js，其中[name]表示模块名称，[hash]表示文件哈希值。

path: 指定打包后生成的文件存放路径，一般设置为项目的静态资源目录。

`publicPath`: 指定打包后的文件生成的路径，可以避免在构建时对资源文件进行绝对路径解析，例如项目中图片、样式等静态资源文件的访问路径。

我也把在项目中具体的配置写在了后面，供你参考。

 复制代码

```
1 module.exports = {
2   context: path.resolve(__dirname, '../'),
3   entry: {
4     app: './src/main.js'
5   },
6   output: {
7     path: config.build.assetsRoot,
8     filename: '[name].js',
9     publicPath: process.env.NODE_ENV === 'production' ?
10       config.build.assetsPublicPath :
11       config.dev.assetsPublicPath
12   }
13 }
```

和入口部分一样，`output` 部分也是在项目构建的时候创建好的，通常来说也不需要我们做更改，但是你要明白它的作用。

loader

接下来要学习的是处理模块的插件 `loader`。`Webpack loader` 是一个用来处理模块的插件，`loader` 的主要作用是对模块进行转换和处理，它可以将各种类型的文件转换成 `Webpack` 可以处理的模块。

具体来说，它可以接受一个输入文件，然后对其进行解析、压缩、混淆、编译等操作，最终输出一个可以供 `Webpack` 打包的模块。常见的 `loader` 包括 `babel-loader`、`css-loader`、`file-loader`、`url-loader` 等，你可以参考后面这张表格来了解它们的用途。

名称	用途
babel-loader	可以将 ES6 + 语法转换为 ES5 语法
css-loader	可以将 CSS 文件转换为 JS 模块
file-loader	可以将文件复制到输出目录中
url-loader	可以压缩和提取公共的图像资源等



通过 loader 的支持，Webpack 能够更好地处理和转换不同的模块类型，从而提升开发效率和构建质量。

只看原理有点抽象，我们通过一个简单的案例来直观感受一下，在项目中我们该如何使用 loader。

复制代码

```
1 module: {
2   rules: [{
3     test: /\.vue$/,
4     loader: 'vue-loader',
5     options: vueLoaderConfig
6   }, {
7     test: /\.less$/,
8     loader: "style-loader!css-loader!less-loader",
9   }, {
10    test: /\.css$/,
11    loader: 'babel-loader',
12    include: [resolve('src'), resolve('test'), resolve('node_modules/webpack-de
13  ], {
14    test: /\.(png|jpe?g|gif|svg)(\?.*)?$/,
15    loader: 'url-loader',
16    options: {
17      limit: 10000,
18      name: utils.assetsPath('img/[name].[hash:7].[ext]')
```

```
19     }
20   }, {
21     test: /\. (mp4|webm|ogg|mp3|wav|flac|aac) (\?.*)?$/,
22     loader: 'url-loader',
23     options: {
24       limit: 10000,
25       name: utils.assetsPath('media/[name].[hash:7].[ext]')
26     }
27   }, {
28     test: /\. (woff2?|eot|ttf|otf) (\?.*)?$/,
29     loader: 'url-loader',
30     options: {
31       limit: 10000,
32       name: utils.assetsPath('fonts/[name].[hash:7].[ext]')
33     }
34   }]
35 }
```

通过这个案例，loader 的用法就一目了然了。可以看到，loader 的配置文件也是放在 webpack.base.conf.js 文件里的。

对于.vue、.less、.js 文件，我们使用 babel-loader 转换。

.vue 文件需要使用 vue-loader 处理。

.less 文件需要使用 style-loader、css-loader 处理。

.css 文件需要使用 css-loader 处理。

通过配置 loader，我们可以轻松地支持各种类型的模块，提升开发效率。


插件 (plugin)

如果我们想要扩展 Webpack 功能的模块，想要修改 Webpack 的打包过程和处理逻辑，那么我们该如何实现呢？这时候就需要用到 Webpack 插件了。

Webpack 中的插件是一种用于扩展 Webpack 功能的模块，它可以用来执行各种任务，例如打包优化、代码分割、资源管理等。在 Webpack 的配置中，插件可以通过 plugins 属性来指定。

Webpack 提供了许多内置的插件，例如 `html-webpack-plugin`、`CopyWebpackPlugin` 等。同时，Webpack 还支持自定义插件，开发者可以编写自己的插件来实现特定的功能。

接下来，对于如何使用 `html-webpack-plugin`，我们还是结合案例来体会。

 复制代码

```
1 plugins: [  
2   new webpack.DefinePlugin({  
3     'process.env': require('../config/dev.env')  
4   }),  
5   new HtmlWebpackPlugin({  
6     filename: 'index.html',  
7     template: 'index.html',  
8     inject: true  
9   }),  
10  new CopyWebpackPlugin([  
11    {  
12      from: path.resolve(__dirname, '../static'),  
13      to: config.dev.assetsSubDirectory,  
14      ignore: ['.*']  
15    }  
16  ])  
17 ]
```

这段代码配置了 `HtmlWebpackPlugin` 插件，目的是生成一个名为 `"index.html"` 的 HTML 文件，并使用名为 `"index.html"` 的文件作为模板，将 Webpack 打包生成的脚本和样式标签直接注入到生成的 HTML 文件中。

具体的参数用途我也给你列在了下面，你大致了解就可以。

shickey.com转载分享

参数名	用途
filename: 'index.html'	指定生成的 HTML 文件的名称。在这个例子中，它将是名为 "index.html" 的文件。
template: 'index.html'	指定使用哪个模板来生成 HTML 文件。在这个例子中，它将使用名为 "index.html" 的文件作为模板。
inject: true	指示 HtmlWebpackPlugin 将 Webpack 打包生成的脚本和样式标签直接注入到生成的 HTML 文件中。



模式 (mode)

最后，我们一起来了解一下 Webpack 是如何适应不同的构造环境的。也许你会有疑问，为什么要使用不同的构造环境呢？


这是因为不同的构造环境会提供不同的编译选项和优化策略，这样可以满足更多开发者的需求。

Webpack 中的 mode 是一个配置选项，主要用于指定 Webpack 的工作模式。它有三种取值：none、development 和 production。三种模式具体的应用场景是后面这样。

模式	描述
none	Webpack 不会对代码做任何优化，直接将模块打包成一个数组，调用模块时直接调用数组中的序号。这个模式主要用于测试和调试，不需要对代码进行优化。
development	Webpack 对代码做了一些基本的优化，例如减小代码体积、删除注释、压缩字符串等。在开发模式下，开发者可以使用一些工具来查看代码，例如 sourcemap 等。
production	Webpack 对代码进行了更多的优化，例如 Tree Shaking、Code Splitting 等。在生产模式下，代码会更加紧凑，加载更快，这样可以提高应用程序的性能。



在 Webpack 的配置中，我们可以根据需求去选择不同模式去构建应用程序，具体可以通过设置 mode 属性来实现。你可以参考后面这个例子。

 复制代码

```
1 module.exports = {  
2   mode: 'development'  
3 };
```

到这里，我们就理清了 Webpack 里最核心的部分。

这里我想带你系统梳理一下，Webpack 在 Vue 项目里各阶段发挥了哪些作用。看完以后，你就会明白，为什么我们在项目的脚手架构建环节就已经设置好了对应的 Webpack 配置项，仍然需要专门学习了解这些配置项的作用和用法。

Webpack 在 Vue 项目的构建和打包阶段发挥着不可或缺的作用。

在开发阶段，Webpack 可以通过各种插件和工具提供开发环境，可以让你能够更加高效地开发和调试项目。同时，在生产环境中，Webpack 还可以帮助开发人员将 Vue 项目中的代码和资源打包成一个或多个静态文件，从而在浏览器中加载和运行。

通过 Webpack 的代码分割和优化等功能，可以提高 Vue 项目的性能和加载速度，同时也可以更好地管理和组织 Vue 项目中的代码和依赖项。

项目实践

shikey.com 转载分享

下面我们进入项目实践环节，演练一下如何使用 Webpack 实现项目打包上线，这里我们需要重点关注 Webpack 打包的命令操作。

shikey.com 转载分享

首先，你要把你本地的项目服务停掉。然后在项目根目录下，运行 “npm run build” 命令。

 复制代码

```
1 npm run build
```

这时候就能生成生产环境下的打包文件。这个命令会编译、压缩 Vue 项目，并将所有的代码和资源打包成一个或多个静态文件，用于在生产环境中运行。后面的截图展示的就是对应的执行过程。

Version: webpack 3.12.0
Time: 31029ms

Asset	Size	Chunks	Chunk Names
static/fonts/element-icons.535877f.woff	28.2 kB	[emitted]	
static/fonts/element-icons.732389d.ttf	56 kB	[emitted]	
static/img/header_day.fac7459.png	395 kB	[emitted] [big]	
static/img/online8.56c476f.png	1.2 MB	[emitted] [big]	
static/img/kkb2.cdbef86.png	76.8 kB	[emitted]	
static/media/1.3ae4202.mp4	1.5 MB	[emitted] [big]	
static/img/kkb1.d30af68.png	449 kB	[emitted] [big]	
static/img/ad.61902c0.jpg	342 kB	[emitted] [big]	
static/js/0.87eca9fd446952e3f74e.js	329 kB	0 [emitted]	vendor--async
static/js/1.753d5d00d9efd86e70fe.js	5.49 kB	1 [emitted]	
static/js/2.80fd7e1733e78d48ea01.js	5.96 kB	2 [emitted]	
static/js/3.053a903064d815d6b67d.js	850 kB	3 [emitted] [big]	
static/js/4.d368eff97be21a4f07f.js	27.7 kB	4 [emitted]	
static/js/5.82ec82363a6555c36398.js	9.85 kB	5 [emitted]	
static/js/6.0f9a6fbb128cae9169e0.js	5 kB	6 [emitted]	
static/js/7.d0a709b5a64bf09278a4.js	6.23 kB	7 [emitted]	
static/js/8.acd0783473bcd4381478.js	12.1 kB	8 [emitted]	
static/js/9.c9c31bc4ade0e1159838.js	10.8 kB	9 [emitted]	
static/js/10.9a9c3459cb4b564da1dc.js	8.33 kB	10 [emitted]	
static/js/11.0396ccb178ad04e2b193.js	4.13 kB	11 [emitted]	
static/js/12.4aa2b9930d311b348eb2.js	1.1 kB	12 [emitted]	
static/js/13.e53a3ca81860035a052.js	2.41 kB	13 [emitted]	
static/js/14.b5ba4b24ee02f1a1168.js	3.59 kB	14 [emitted]	
static/js/15.cf2ada32e512d8102537.js	3.15 kB	15 [emitted]	
static/js/16.1f1448f9f74d577c65ed.js	3.33 kB	16 [emitted]	
static/js/17.42837cfac8cf37f7f6cc.js	4.54 kB	17 [emitted]	
static/js/vendor.3674068ce19a6f0ffdd13.js	1.01 MB	18 [emitted] [big]	vendor
static/js/app.2eddb59b5fbdcad798a60.js	18.3 kB	19 [emitted]	app
static/js/manifest.7edd67e3da4fe32b886c.js	1.9 kB	20 [emitted]	manifest
static/css/app.6602aa98405097a257a7fc9db0b25c7b.css	350 kB	19 [emitted] [big]	app
static/css/app.6602aa98405097a257a7fc9db0b25c7b.css.map	570 kB	[emitted]	
static/js/0.87eca9fd446952e3f74e.js.map	566 kB	0 [emitted]	
static/js/1.753d5d00d9efd86e70fe.js.map	22.3 kB	1 [emitted]	vendor--async
static/js/2.80fd7e1733e78d48ea01.js.map	30.9 kB	2 [emitted]	



如果你看到这张图，就代表打包成功了。这时候，你会在你的项目目录下看到一个 dist 访问文件，其中包含了 Vue 项目的所有静态资源，如 HTML、CSS、JS、图片等。进行到这里，我们就成功完成了项目打包。

```
Build complete.  
  
Tip: built files are meant to be served over an HTTP server.  
Opening index.html over file:// won't work.  
  
New major version of npm available! 6.4.1 -> 9.6.7  
Change log: https://github.com/npm/cli/releases/tag/v9.6.7  
Run npm install -g npm to update!
```

shikey.com转载分享



接下来，我们需要将生成的打包文件 dist 上传到 Web 服务器，这里你可以使用 FTP 等工具。我推荐你使用 **FileZilla** 来完成这一步，它支持直接的拖拽式操作，对初学者比较友好。详细的下载说明，你可以直接参考 [FileZilla 中文官网](#)。

与此同时，我们还要在 Web 服务器上配置 Nginx 或 Apache 等服务器，这样才能将打包文件提供给用户访问。你可以根据自己的需求，配置 Web 服务器的虚拟主机、域名解析、SSL

证书等，这个是非常灵活的，你自己选择就可以。

好，项目的打包上线工作大功告成，我们的视频平台可以支持更多的用户访问和使用，恭喜你坚持到这里。

总结

今天的学习告一段落，我们一起来总结回顾一下本节课的内容。

善于使用 Webpack 能让 Vue 项目的开发更加便捷、高效。作为模块管理工具，它不但可以通过整合各种资源（如 CSS、图片等），减少网页加载时的体积，还能将分散的代码打包成一个或多个模块从而减少网页加载时间。

Webpack 的学习重点就是了解每一个配置项的作用，这样才能逐步强化我们的 Webpack 应用能力，更好地管理项目，也可应对一些项目优化问题。

在项目打包和配置发布上线环节，通过打包命令生成 dist 文件之后，你可以尝试放到自己的服务器上部署访问一下，看看效果。这个过程你一定要在课后实践一下，相信在完成之后你会收获满满。

到这里，我们完成了前端开发的全部内容。从下节课开始，我们的后端开发之旅即将开启，实现前后端联调的目标就在眼前，相信你已经迫不及待了，让我们继续前进。

思考题

shikey.com转载分享

在 Webpack 中，loader 中 test 选项用于匹配需要被处理的文件。你知道它是通过什么样的方式进行文件类型匹配的吗？

shikey.com转载分享

欢迎你在留言区和我交流互动，如果这节课对你有帮助，也推荐你分享给身边的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (2)



曾泽浩

2023-06-15 来自广东

老师可以加餐讲一下怎么部署到服务器吗？

作者回复：可以的，推进我们正常课程，之后我会给安排一节如何实现服务器的部署。



peter

2023-05-31 来自北京

请教老师几个问题：

Q1：VSCode中编辑vue文件，而vue文件是node解析的，这个关系应该是在VSCode中指定的吧。请问，是在VSCode的什么地方指定的？

Q2：webpack能打包Java后端吗？

Q3：output的path和public有什么区别啊？

Q4：不允许打包的标志是什么？

文中提到“Webpack 会从入口起点开始打包，直到遇到第一个不允许打包的模块或文件为止”，怎么判断不运行打包？

Q5：loader部分，vue会被babel-loader和vue-loader处理。对于一个vue文件，是被两个loader都处理吗？还是选一个处理？

Q6：打包后vue文件还存在吗？

我的理解：vue文件在开发的时候存在，打包后并不存在vue文件；vue会被转换为js文件。

作者回复：1、node.js是什么？你还记得吗？Node.js是一个基于Chrome V8引擎的JavaScript运行环境，所以不是vscode。

2、Webpack是一个用于构建前端应用程序的打包工具，所以不能打包后端

3、path：指定输出文件的路径。public：指定公共路径，用于在HTML文件中引用资源文件。

4、它是一种情况，例如配置文件中的 entry 选项指定了多个入口文件。配置文件中的 entry 选项指定的入口文件不存在。配置文件中的 output 选项指定的输出文件路径不存在。配置文件中的 output 选项指定的文件名与输入文件名不一致。这几种情况都不允许继续打包。

5、对于一个 Vue 文件，它会被 vue-loader 处理，不需要进行选择。

6、你的理解是正确的，Webpack在打包过程中会处理vue文件，并将它们转换成JavaScript代码和相关的HTML、CSS、JS和图片文件等。

