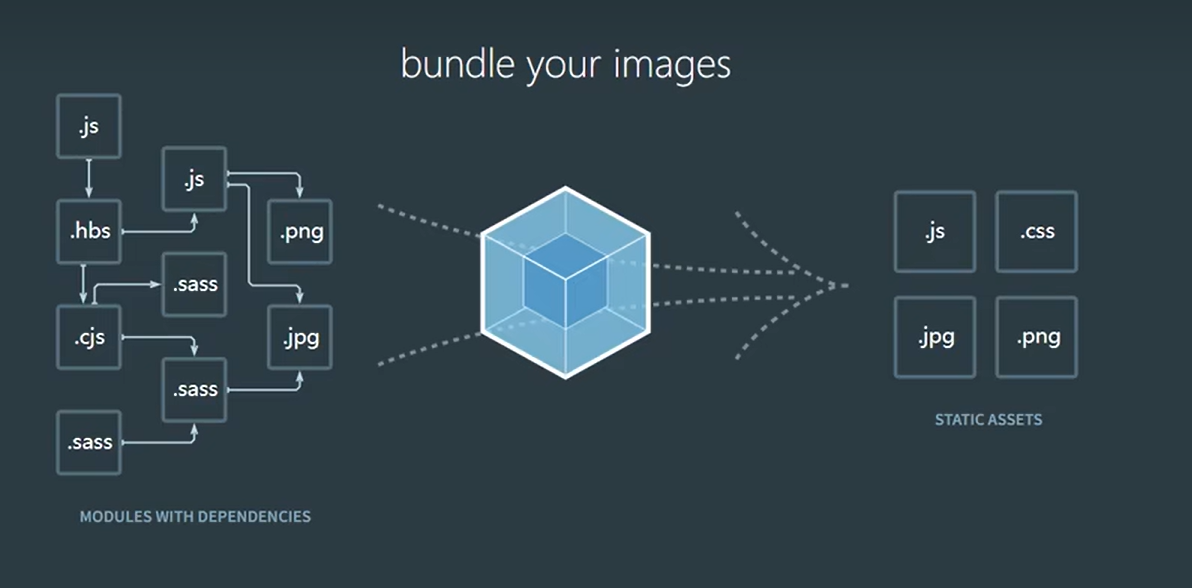
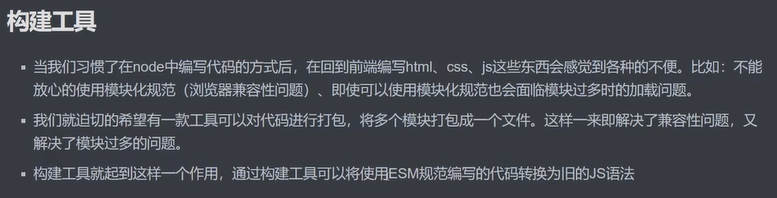
# Webpack&vite前端构建工具

## 初识构建工具





**可以理解为把一堆包合成一个包，虽然合起来的那个包大了,但请求次数少了,代码的运行速度自然也就上来了。比如我们用vue写了一个项目,但是有的浏览器（如IE）他不兼容vue的或者是ES6的语法,我们用打包工具给他打包成原始的JS文件。这样就兼容了,同时还有一个好处就是他多个模块合成了一个大的,不用一个一个去请求了。**

## 配置文件

其实我们vue构建项目自己就给你配置好了文本Pack或者是vite。实际开发根本用不到,但面试可能要用,所以这里写一下配置文件的使用。

### Entry入口设置

**Entry就是我们对哪个文件进行打包,webpack默认的src/index.js。**



我们也可以指定打包后的文件名称,下面这个打包后的文件就叫main.js。



我们可以一起打包两个文件,只要写成下面一样的数组格式就行了。



也可以像下面这样,把两个文件打包成两个文件。



### Output出口设置

**出口设置其实就是设置打包文件的名字,像下面的这样。**



**也可以设置打包文件的路径。**



我们打包名字一样他会覆盖,但如果两次打包不一样他就不会给你删打包目录,就需要有这个配置项来解决。

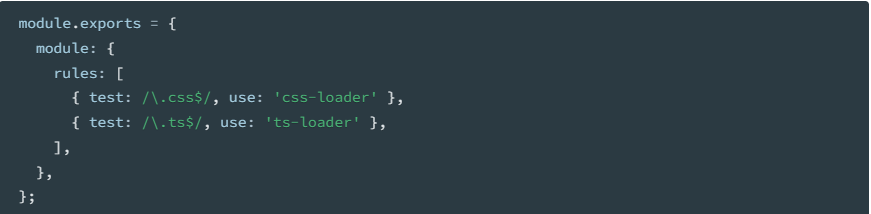


### Loader加载器

**Webpack打包时默认是只能打包JS的,如果想打包别的,像CSS,图片这种东西就要用到loader这个加载器了。**

基本的写法就是下面那种,model里面写的就是rules规则,test是正则表达式,use是使用的加载器,使用的加载器要先下载。

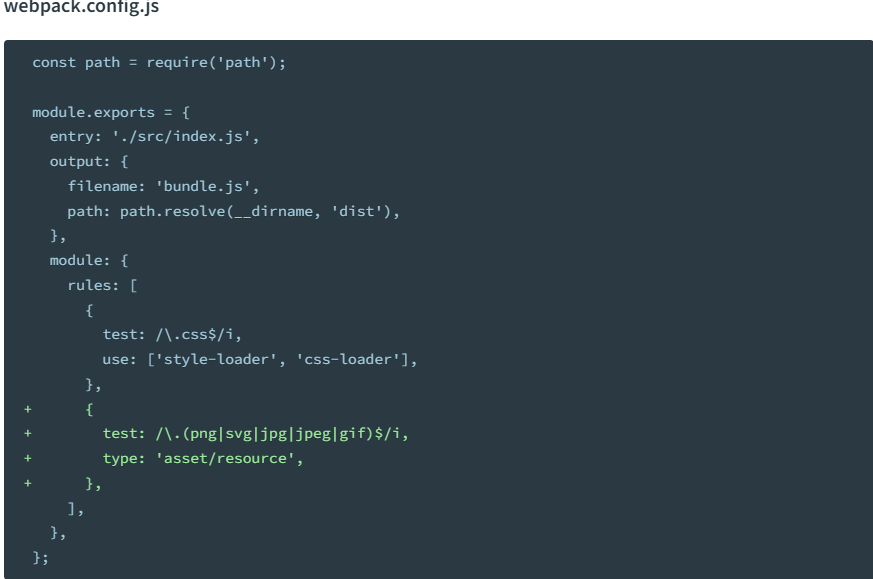




有时候需要多个加载器(比如加载CSS)需要css-laoder和style-loader这两个加载器。



加载图片用



### Babel

Babel其实也是一个loader加载器,他的作用就是把一些比较新的代码变成原始的JS。比如我们写了箭头函数()=>{},我们不用Babel的话打包后的文件里还是箭头函数。如果我们用了Bable的话打包后箭头函数就会变成function （）{}这样的普通函数,可以解决兼容性问题。

官网给出的Bable方法和之前的没多大区别。

下载



匹配使用,一条龙。



我们也可以在package.json里面配置,非常好的一集。



### Plugin(插件)

上面的那些都是打包处理代码的,这个plugin配置项,就是用这个插件解决代码打包过程的,以HtmlWebPackPlugin为例,我们的步骤为：

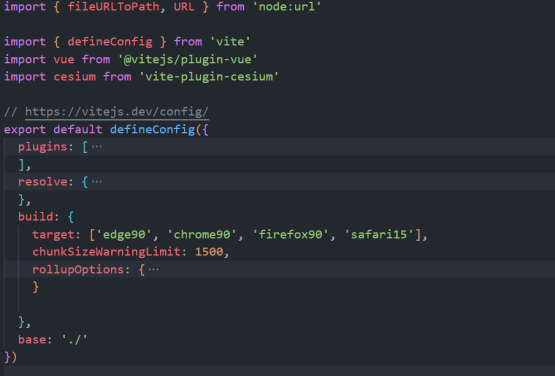
1. npm安装。
2. Require引入。
3. 在plugin选项里配置一下。
4. 他还可以在里面配置一些想要的参数**实现一个定制效果**。



**Plugin的特点就是他不处理代码。**

### Vite

Vite与webpack的区别在于他简单易用，且他编写代码的时候不打包，所以运行更快,只有最后上线的时候打包一次。配置文件也就那样

。用的时候现查就行了,毕竟这玩意是一锤子买卖，开发中只要配置一次，脚手架还会自动给你配置好。

**不过他打包有一点要注意。**

**Npm run build打包出来的HTML不能在liveserver打开,会导致路径错误,要在服务器打开,所以正确的用法是npm run build后再用npm run preview防止路径错误。**