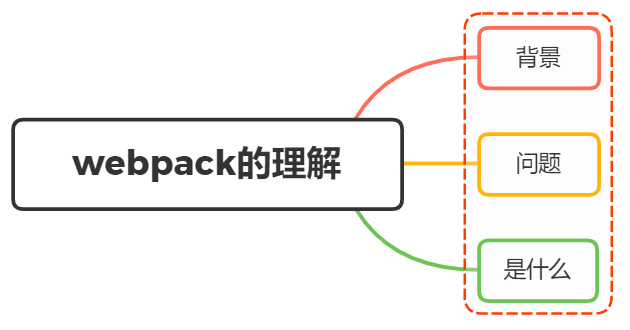
# 面试官：说说你对webpack的理解？解决了什么问题？



## 一、背景

Webpack 最初的目标是实现前端项目的模块化，旨在更高效地管理和维护项目中的每一个资源

#### 模块化

最早的时候，我们会通过文件划分的形式实现模块化，也就是将每个功能及其相关状态数据各自单独放到不同的JS 文件中

约定每个文件是一个独立的模块，然后再将这些js文件引入到页面，一个script标签对应一个模块，然后调用模块化的成员

<script src="module-a.js"></script>  
<script src="module-b.js"></script>

但这种模块弊端十分的明显，模块都是在全局中工作，大量模块成员污染了环境，模块与模块之间并没有依赖关系、维护困难、没有私有空间等问题

项目一旦变大，上述问题会尤其明显

随后，就出现了命名空间方式，规定每个模块只暴露一个全局对象，然后模块的内容都挂载到这个对象中

window.moduleA = {  
 method1: function () {  
 console.log('moduleA#method1')  
 }  
}

这种方式也并没有解决第一种方式的依赖等问题

再后来，我们使用立即执行函数为模块提供私有空间，通过参数的形式作为依赖声明，如下

// module-a.js  
(function ($) {  
 var name = 'module-a'  
  
 function method1 () {  
 console.log(name + '#method1')  
 $('body').animate({ margin: '200px' })  
 }  
  
 window.moduleA = {  
 method1: method1  
 }  
})(jQuery)

上述的方式都是早期解决模块的方式，但是仍然存在一些没有解决的问题。例如，我们是用过script标签在页面引入这些模块的，这些模块的加载并不受代码的控制，时间一久维护起来也十分的麻烦

理想的解决方式是，在页面中引入一个JS入口文件，其余用到的模块可以通过代码控制，按需加载进来

除了模块加载的问题以外，还需要规定模块化的规范，如今流行的则是CommonJS、ES Modules

## 二、问题

从后端渲染的JSP、PHP，到前端原生JavaScript，再到jQuery开发，再到目前的三大框架Vue、React、Angular

开发方式，也从javascript到后面的es5、es6、7、8、9、10，再到typescript，包括编写CSS的预处理器less、scss等

现代前端开发已经变得十分的复杂，所以我们开发过程中会遇到如下的问题：

* 需要通过模块化的方式来开发
* 使用一些高级的特性来加快我们的开发效率或者安全性，比如通过ES6+、TypeScript开发脚本逻辑，通过sass、less等方式来编写css样式代码
* 监听文件的变化来并且反映到浏览器上，提高开发的效率
* JavaScript 代码需要模块化，HTML 和 CSS 这些资源文件也会面临需要被模块化的问题
* 开发完成后我们还需要将代码进行压缩、合并以及其他相关的优化

而webpack恰巧可以解决以上问题

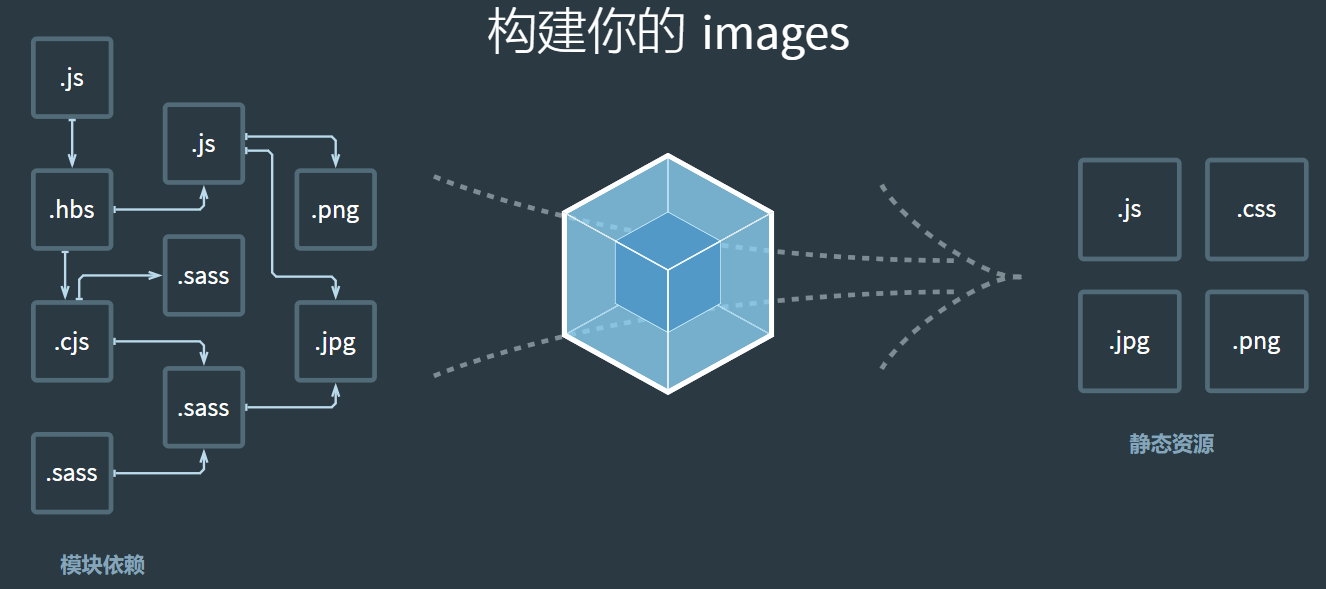
## 三、是什么

webpack 是一个用于现代JavaScript应用程序的静态模块打包工具

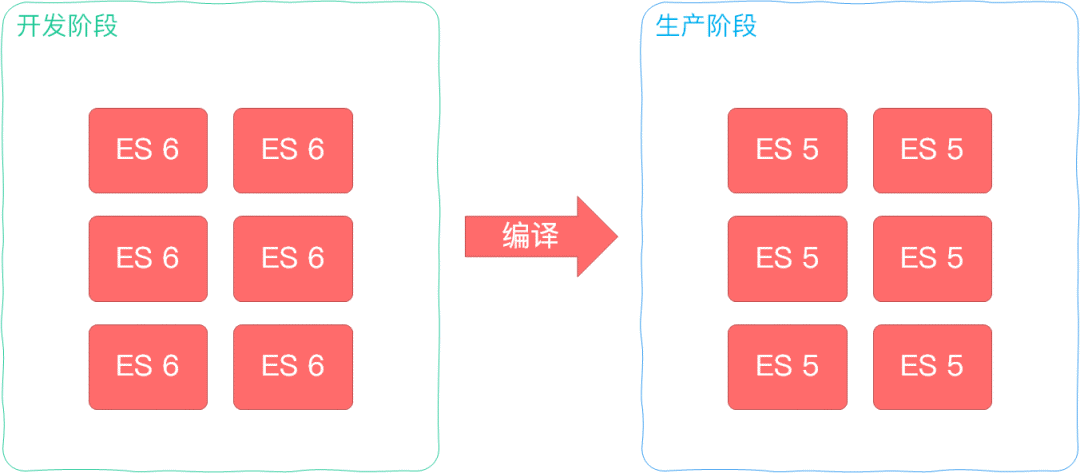
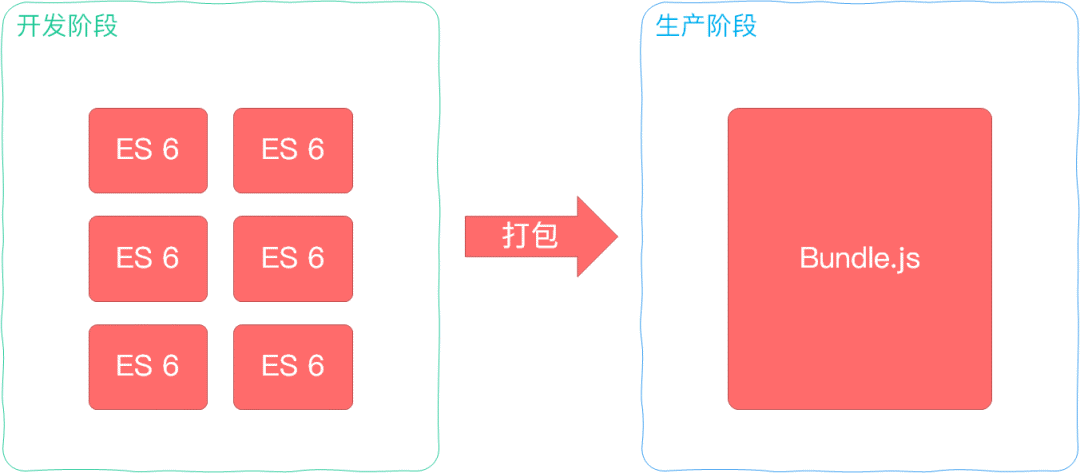
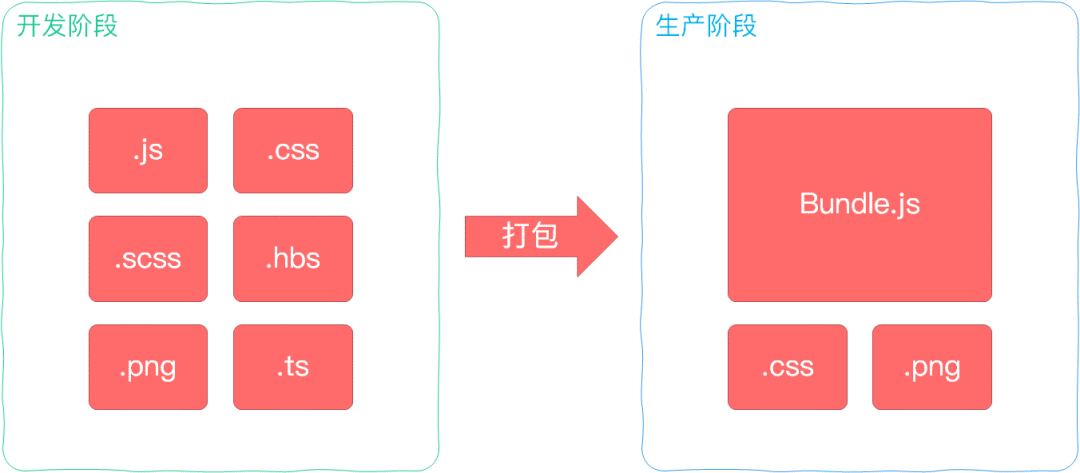
* 静态模块

这里的静态模块指的是开发阶段，可以被 webpack 直接引用的资源（可以直接被获取打包进bundle.js的资源）

当 webpack处理应用程序时，它会在内部构建一个依赖图，此依赖图对应映射到项目所需的每个模块（不再局限js文件），并生成一个或多个 bundle



#### webpack的能力：

**编译代码能力**，提高效率，解决浏览器兼容问题  **模块整合能力**，提高性能，可维护性，解决浏览器频繁请求文件的问题  **万物皆可模块能力**，项目维护性增强，支持不同种类的前端模块类型，统一的模块化方案，所有资源文件的加载都可以通过代码控制 

## 参考文献

* https://webpack.docschina.org/concepts/
* https://zhuanlan.zhihu.com/p/267875652