模块化 & 构建工具面试题集锦

# 第一章 模块化章节

## 1.说说你对AMD和Commonjs的理解

1) CommonJS是服务器端模块的规范，Node.js采用了这个规范。CommonJS规范加载模块是同步的，也就是说，只有加载完成，才能执行后面的操作。AMD规范则是非同步加载模块，允许指定回调函数

2) AMD推荐的风格通过返回一个对象(return 模块对象)做为模块对象，CommonJS的风格通过对module.exports或exports的属性赋值来达到暴露模块对象的目的

## 2. Commonjs和ES6模块化的区别

1) CommonJS是服务器端模块的规范，Node.js采用了这个规范。但目前也可用于浏览器端，需要使用Browserify进行提前编译打包.

2) ES6模块化规范专门针对于浏览器端，但目前浏览器支持不是很好(只有Chrome浏览器支持), 也需要使用Browserify进行打包

3) 暴露的方式和暴露的本质：

|  |
| --- |
| 1、commonjs暴露的方式   - module.exports = value;   - exports.xxx = value;   - 暴露的本质是exports对象 2、ES6中暴露的方式   - export xxx （常规暴露，暴露的本质是对象，接收的时候只能以对象的解构赋值的方式来接收值）   - export default （默认暴露，暴露任意数据类型，暴露什么数据类型，接收什么数据类型） |

## 谈谈你对js模块化的理解

1. 当工业级的项目开发的足够大的时候，如果将所有的js代码定义在一个js文件的话，使得复杂度提升，后期维护难度加大。
2. 如果将一个大的js文件根据一定的规范拆分成几个小的文件的话将会便于管理，可以提高复用性。
3. 模块化在项目中十分的重要，一个复杂的项目肯定有很多相似的功能模块，如果每次都需要重新编写模块肯定既费时又耗力。但是引用别人编写模块的前提是要有统一的“打开姿势”，如果每个人有各自的写法，那么肯定会乱套，所有会引出模块化规范的使用
4. 常用的JavaScript模块化规范有四种： Commonjs， AMD(require.js), CMD(sea.js), ES6模块化

## 模块化好文地址：

* 1. <http://www.jb51.net/article/107245.htm>

# 第二章：项目构建章节

## loader和plugin区别

loader用于加载某些资源文件，因为webpack本身只能打包CommonJS规范的js文件，对于其他资源，例如css，图片等，是没有办法加载的，这就需要对应的loader将资源转换 plugin用于扩展webpack的功能，直接作用于webpack，loader只专注于转换文件，而plugin不仅局限于资源加载

Loader只能处理单一文件的输入输出，而Plugin则可以对整个打包过程获得更多的灵活性，譬如 ExtractTextPlugin，它可以将所有文件中的css剥离到一个独立的文件中，这样样式就不会随着组件加载而加载了。

## 2.什么是chunk



Webpack提供一个功能可以拆分模块，每一个模块称为chunk，这个功能叫做Code Splitting。你可以在你的代码库中定义分割点，调用require.ensure，实现按需加载

## 3.如何开发一个loader，原理是啥

A loader is a node module exporting a function.

缓存： Webpack Loader 同样可以利用缓存来提高效率，并且只需在一个可缓存的 Loader 上加一句 this.cacheable() 异步：在一个异步的模块中，回传时需要调用 Loader API 提供的回调方法 this.async()

## 4.打包原理

webpack打包，最基本的实现方式，是将所有的模块代码放到一个数组里，通过数组ID来引用不同的模块

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*/ ([

/\* 0 \*/

/\*\*\*/ function(module, exports, \_\_webpack\_require\_\_) {

\_\_webpack\_require\_\_(1);

\_\_webpack\_require\_\_(2);

console.log('Hello, world!');

/\*\*\*/ },

/\* 1 \*/

/\*\*\*/ function(module, exports) {

var a = 'a.js';

console.log("I'm a.js");

/\*\*\*/ },

/\* 2 \*/

/\*\*\*/ function(module, exports) {

var b = 'b.js';

console.log("I'm b.js");

/\*\*\*/ }

/\*\*\*\*\*\*/ ]);

可以发现入口entry.js的代码是放在数组索引0的位置，其它a.js和b.js的代码分别放在了数组索引1和2的位置，而webpack引用的时候，主要通过\_\_webpack\_require\_\_的方法引用不同索引的模块。

## 5.webpack和gulp的区别

webpack是一种模块化打包工具，主要用于模块化方案，预编译模块的方案；gulp是工具链、构建工具，可以配合各种插件做js压缩，css压缩，less编译 替代手工实现自动化工作。

Grunt/Gulp更多的是一种工作流；提供集成所有服务的一站式平台； gulp可以用来优化前端工作流程。

## 6.如何写一个plugin

Compiler在开始打包时就进行实例化，实例对象里面装着与打包相关的环境和参数，包括options、plugins和loaders等。

compilation对象，它继承于compiler，所以能拿到一切compiler的内容。Compilation表示有关模块资源，已编译资源，Compilation在每次文件变化重新打包时都进行一次实例化

apply方法：当安装这个插件的时候，这个apply方法就会被webpack compiler调用。

function HelloWorldPlugin(options) {

// Setup the plugin instance with options...

}

HelloWorldPlugin.prototype.apply = function(compiler) {

compiler.plugin('done', function() {

console.log('Hello World!');

});

};

module.exports = HelloWorldPlugin;

## 7.webpack打包后文件体积过大怎么办？

很多方法：异步加载模块（代码分割）；提取第三方库（使用cdn或者vender）；代码压缩；去除不必要的插件；去除devtool选项，dllplugin等等

## 8.请说出三种减低页面加载时间的方法

1、压缩css、js文件

2、合并js、css文件，减少http请求

3、外部js、css文件放在最底下

4、减少dom操作，尽可能用变量替代不必要的dom操作