### 1 webpack与grunt、gulp的不同？

grunt和gulp是基于任务和流（Task、Stream）的。类似jQuery，找到一个（或一类）文件，对其做一系列链式操作，更新流上的数据， 整条链式操作构成了一个任务，多个任务就构成了整个web的构建流程。

webpack是基于入口的。webpack会自动地递归解析入口所需要加载的所有资源文件，然后用不同的Loader来处理不同的文件，用Plugin来扩展webpack功能。

### 2 Loader和Plugin的不同

Loader直译为"加载器"。Webpack将一切文件视为模块，但是webpack原生是只能解析js文件，如果想将其他文件也打包的话，就会用到loader。 所以Loader的作用是让webpack拥有了加载和解析非JavaScript文件的能力。

Plugin直译为"插件"。Plugin可以扩展webpack的功能，让webpack具有更多的灵活性。 在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

### 3 有哪些常见的Loader？他们是解决什么问题的？

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

### 4 有哪些常见的Plugin？他们是解决什么问题的？

define-plugin：定义环境变量

commons-chunk-plugin：提取公共代码

uglifyjs-webpack-plugin：通过UglifyES压缩ES6代码

### 5 webpack的构建流程是什么

1 初始化参数：从配置文件和 Shell 语句中读取与合并参数，得出最终的参数；

2 开始编译：用上一步得到的参数初始化 Compiler 对象，加载所有配置的插件，执行对象的 run 方法开始执行编译；

3 确定入口：根据配置中的 entry 找出所有的入口文件；

4 编译模块：从入口文件出发，调用所有配置的 Loader 对模块进行翻译，再找出该模块依赖的模块，再递归本步骤直到所有入口依赖的文件都经过了本步骤的处理；

5 完成模块编译：在经过第4步使用 Loader 翻译完所有模块后，得到了每个模块被翻译后的最终内容以及它们之间的依赖关系；

6 输出资源：根据入口和模块之间的依赖关系，组装成一个个包含多个模块的 Chunk，再把每个 Chunk 转换成一个单独的文件加入到输出列表，这步是可以修改输出内容的最后机会；

7 输出完成：在确定好输出内容后，根据配置确定输出的路径和文件名，把文件内容写入到文件系统。

在以上过程中，Webpack 会在特定的时间点广播出特定的事件，插件在监听到感兴趣的事件后会执行特定的逻辑，并且插件可以调用 Webpack 提供的 API 改变 Webpack 的运行结果。

### 6 如何利用webpack来优化前端性能？

1 压缩代码。删除多余的代码、注释、简化代码的写法等等方式。可以利用webpack的UglifyJsPlugin和ParallelUglifyPlugin来压缩JS文件， 利用cssnano（css-loader?minimize）来压缩css

2 利用[CDN](https://cloud.tencent.com/product/cdn?from=10680" \t "https://www.cnblogs.com/gaoht/p/_blank)加速。在构建过程中，将引用的静态资源路径修改为CDN上对应的路径。可以利用webpack对于output参数和各loader的publicPath参数来修改资源路径

3 删除死代码（Tree Shaking）。将代码中永远不会走到的片段删除掉。可以通过在启动webpack时追加参数--optimize-minimize来实现

4 提取公共代码。

### 7 有哪些方式可以提升webpack的打包速度？

· loader：使用include和exclude指定搜索文件的范围。

· babel-loader:配置后面加上loader:'babel-loader?cacheDirectory=true'将编译过的文件缓存起来，下次只需要编译更改过的代码文件即可。

· HappyPack：使用这个插件开启loader多线程打包。

· DllPlugin：将特定的类库提前打包然后引入，减少打包类库的次数，只有当类库更新版本才进行重新打包。

· resolve.alias：配置别名，更快找到路径。

· module.noParse：确定这个文件没有其他依赖时，让webpack打包时不扫描该文件。

### 8 babel的原理

babel的转译过程也分为三个阶段：****parsing、transforming、generating****

ES6代码输入 ==》 babylon进行解析 ==》 得到AST  
==》 plugin用babel-traverse对AST树进行遍历转译 ==》 得到新的AST树  
==》 用babel-generator通过AST树生成ES5代码  
babel只是转译新标准引入的语法，比如ES6的箭头函数转译成ES5的函数；而**新标准引入的新的原生对象，部分原生对象新增的原型方法，新增的API等（如Proxy、Set等），这些babel是不会转译的。需要用户自行引入polyfill来解决**