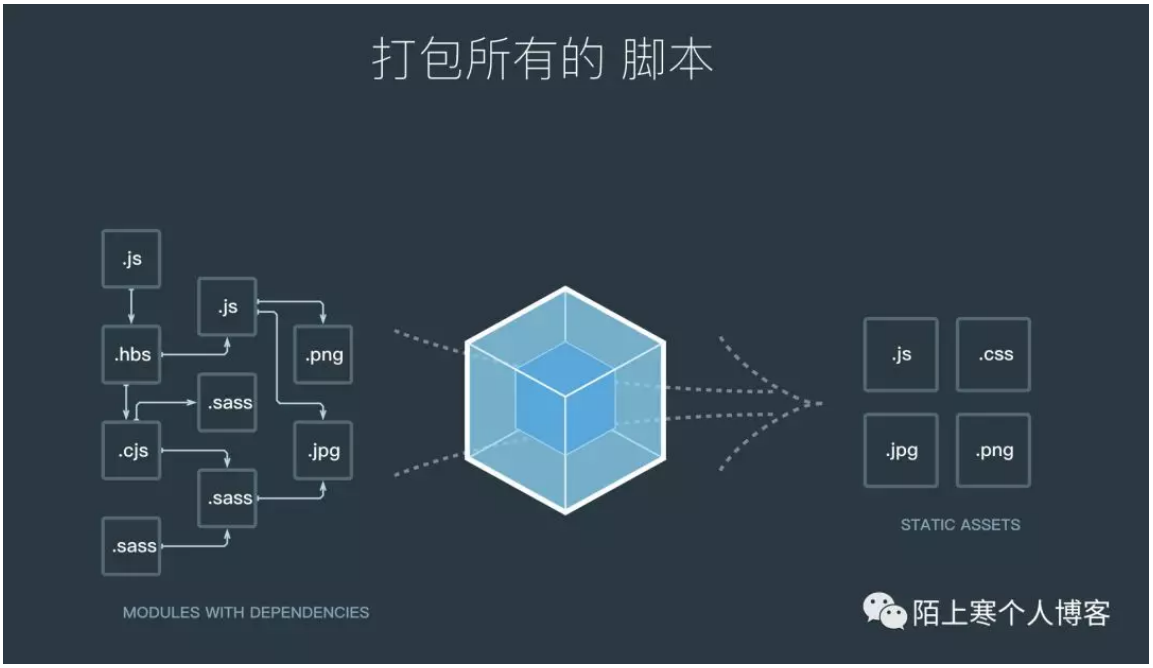
什么是webpack

webpack是一个打包模块化javascript的工具，在webpack里一切文件皆模块，通过loader转换文件，通过plugin注入钩子，最后输出由多个模块组合成的文件，webpack专注构建模块化项目。

WebPack可以看做是模块打包机：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

官网的图片形象的展示了webpack的定义



几个常见的loader

- 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

几个常见的plugin

- define-plugin：定义环境变量
- terser-webpack-plugin：通过TerserPlugin压缩ES6代码

- html-webpack-plugin 为html文件中引入的外部资源，可以生成创建html入口文件
- mini-css-extract-plugin：分离css文件
- clean-webpack-plugin：删除打包文件
- happypack：实现多线程加速编译

webpack与grunt、gulp的不同?

Webpack与Gulp、Grunt没有什么可比性，它可以看作模块打包机，通过分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其转换和打包为合适的格式供浏览器使用。**Gulp/Grunt是一种能够优化前端的开发流程的工具，而WebPack是一种模块化的解决方案，不过Webpack的优点使得Webpack在很多场景下可以替代Gulp/Grunt类的工具。**

他们的工作方式也有较大区别：

Grunt和Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，工具之后可以自动替你完成这些任务。

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个（或多个）浏览器可识别的JavaScript文件。

三者都是前端构建工具，grunt和gulp在早期比较流行，现在webpack相对来说比较主流，不过一些轻量化的任务还是会用gulp来处理，比如单独打包CSS文件等。

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。

所以总结一下：

从构建思路来说

gulp和grunt需要开发者将整个前端构建过程拆分成多个Task，并合理控制所有Task的调用关系

webpack需要开发者找到入口，并需要清楚对于不同的资源应该使用什么Loader做何种解析和加工

对于知识背景来说

gulp更像后端开发者的思路，需要对于整个流程了如指掌 webpack更倾向于前端开发者的思路

webpack有哪些优点

- 专注于处理模块化的项目，能做到开箱即用，一步到位
- 可通过plugin扩展，完整好用又不失灵活

- 使用场景不局限于web开发
- 社区庞大活跃，经常引入紧跟时代发展的新特性，能为大多数场景找到已有的开源扩展
- 良好的开发体验

webpack的缺点

webpack的缺点是只能用于采用模块化开发的项目

分别介绍bundle，chunk，module是什么

bundle：是由webpack打包出来的文件，

chunk：代码块，一个chunk由多个模块组合而成，用于代码的合并和分割。

module：是开发中的单个模块，在webpack的世界，一切皆模块，一个模块对应一个文件，webpack会从配置的entry中递归开始找出所有依赖的模块。

分别介绍什么是loader?什么是plugin?

loader：模块转换器，用于将模块的原内容按照需要转成你想要的内容

plugin：在webpack构建流程中的特定时机注入扩展逻辑，来改变构建结果，是用来自定义webpack打包过程的方式，一个插件是含有apply方法的一个对象，通过这个方法可以参与到整个webpack打包的各个流程(生命周期)。

什么 是模块热更新?

模块热更新是webpack的一个功能，他可以使得代码修改过后不用刷新浏览器就可以更新，是高级版的自动刷新浏览器。

devServer中通过hot属性可以控制模块的热替换

1，通过配置文件

```
const webpack = require('webpack');
const path = require('path');
let env = process.env.NODE_ENV == "development" ? "development" : "production";
const config = {
    mode: env,
  devServer: {
          hot:true
  }
}
    plugins: [
          new webpack.HotModuleReplacementPlugin(),  //热加载插件
    ],
module.exports = config;
```

2，通过命令行

```
    "scripts": {
```

```
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "NODE_ENV=development webpack-dev-server --config webpack.develop.config.js --hot",
  },
```

什么是Tree-shaking

Tree-shaking可以用来剔除javascript中不用的死代码，它依赖静态的es6模块化语法，例如通过哦import 和export 导入导出，Tree-shaking最先在rollup中出现，webpack在2.0中将其引入，css中使用Tree-shaking需要引入Purify-CSS

通过webpack处理长缓存

浏览器在用户访问页面的时候，为了加快加载速度，会对用户访问的静态资源进行存储，但是每一次代码升级或是更新，都需要浏览器去下载新的代码，最方便和简单的更新方式就是引入新的文件名称。在webpack中可以在output纵输出的文件指定chunkhash,并且分离经常更新的代码和框架代码。通过NameModulesPlugin或是HashedModuleIdsPlugin使再次打包文件名不变。

如何提高webpack的构建速度

1. 通过externals配置来提取常用库
2. 利用DllPlugin和DllReferencePlugin预编译资源模块 通过DllPlugin来对那些我们引用但是绝对不会修改的npm包来进行预编译，再通过DllReferencePlugin将预编译的模块加载进来。
3. 使用Happypack 实现多线程加速编译

要注意的第一点是，它对file-loader和url-loader支持不好，所以这两个loader就不需要换成happypack了，其他loader可以类似地换一下

1. 使用Tree-shaking和Scope Hoisting来剔除多余代码
2. 使用fast-sass-loader代替sass-loader
3. babel-loader开启缓存

babel-loader在执行的时候，可能会产生一些运行期间重复的公共文件，造成代码体积大冗余，同时也会减慢编译效率
可以加上cacheDirectory参数或使用 transform-runtime 插件试试

```
// webpack.config.js
use: [{
    loader: 'babel-loader',
    options: {
        cacheDirectory: true
}]
// .babelrc
```

```
{
        "presets":  [
                "env",
                "react"
        ],
        "plugins":  ["transform-runtime"]
}
```

1. 不需要打包编译的插件库换成全局"script"标签引入的方式

比如jQuery插件，react, react-dom等，代码量是很多的，打包起来可能会很耗时
可以直接用标签引入，然后在webpack配置里使用 expose-loader 或 externals 或 ProvidePlugin 提供给模块内部使用相应的变量

```
// @1
use:  [{
        loader:  'expose-loader',
        options:  '$'
        },  {
        loader:  'expose-loader',
        options:  'jQuery'
        }]
// @2
externals:  {
          jquery:  'jQuery'
        },
// @3
new  webpack.ProvidePlugin({
      $:  'jquery',
     jQuery:  'jquery',
     'window.jQuery':  'jquery'
  }),
```

1. 优化构建时的搜索路径

在webpack打包时，会有各种各样的路径要去查询搜索，我们可以加上一些配置，让它搜索地更快
比如说，方便改成绝对路径的模块路径就改一下，以纯模块名来引入的可以加上一些目录路径
还可以善于用下resolve alias别名 这个字段来配置
还有exclude等的配置，避免多余查找的文件，比如使用babel别忘了剔除不需要遍历的