

京程一灯精英班第三周笔试题 姓名：

请先盖住答案，在右侧空白处作答已加深印象。

- 1.请你说明前端开发中为何需要工程化？（5分）

- 答：

 老袁的建议是结合你自己项目的具体开发环境来详细的进行描述（下面不懂的词要谷歌）：

1.工程化为了提高效率和降低成本而生。


2.前端工程化实现的方案必须包括如下要素：开发规范（CSS Hint、Html Hint、CSS Hint），模块化开发（资源的模块化、CSS的模块化包含BEM CSS Modules、JS的模块化），组件化开发（每个包含模板(HTML)+样式(CSS)+逻辑(JS)功能完备的结构单元统前端框架/类库的思想是先组织DOM，然后把某些可复用的逻辑封装成组件来操作DOM，是DOM优先；而组件化框架/类库的思想是先来构思组件，然后用DOM这种基本单元结合相应逻辑来实现组件，是组件优先。现在可以先分文件夹再甚至用Vue等定义进一步细致的组件），组件仓库(Git、SVN着重自动化分支构建)，性能优化（自动化的压缩合并打包加戳），预编译和资源注入（包括es6/7语法转译、css预编译器处理、spirit图片生成），部署，开发流程(集成Code Review、Bug收集、敏捷开发、文件监听，动态编译)，开发文档（jsDoc），自动化测试。

3.JavaScript面向对象开发 HTML面向组件开发。

4.不把代码交给机器的程序员不是好程序员。

- 2.请绘制代码版本仓库多分支的流程图以及如何用Node实现自动化分支构建？（5分）

- 答：

 请大家找到一个仓库，务必要模拟一下整个分支开发的过程：

1.svn checkout svn地址 --username 用户名

2.svn branch 分支名（add/commit）。

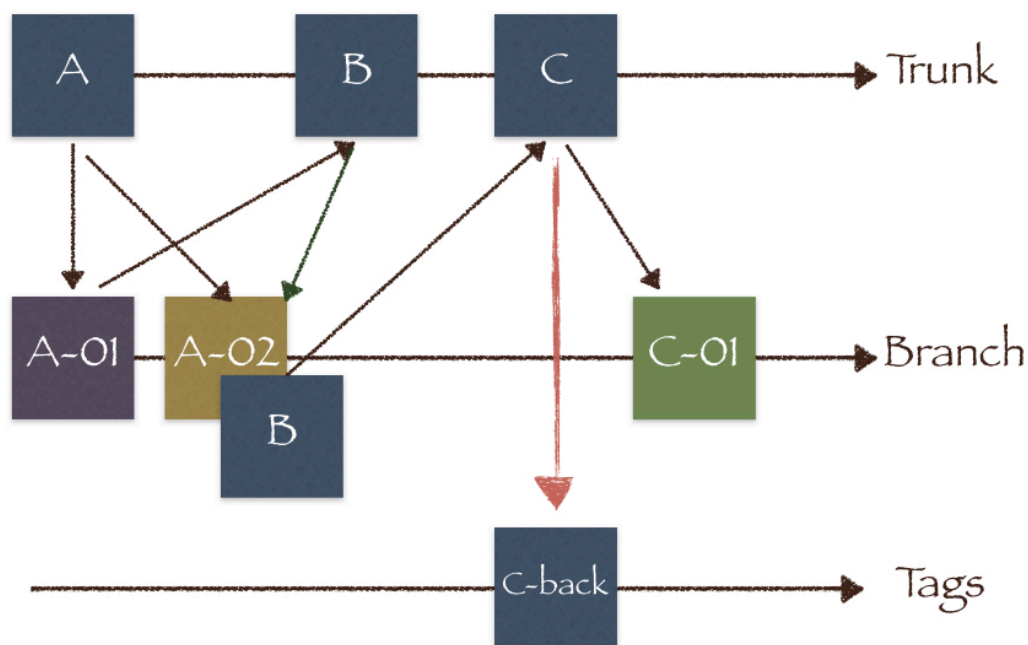
3.svn merge 主干svn地址 分支svn地址。

4.Beyond Compare -> svn resolved。

5.svn copy 主干SVN地址 /tags/2017

- 3.请说明Grunt、Gulp、Webpack、Rollup、FIS、Parcel的优缺点。（10分）

SVN开发阶段图示



- 答：

🍊 请各位务必将以上每一种构建工具均配置熟悉（尤其Yogurt）：

1.通过Gruntfile配置完成后，可以实现自动化。在执行缩小，编译，单元测试，测试等重复性任务。虽然grunt看似已垂垂老矣，但是以前写的很多项目一直用的就是grunt，温故方能知新。

2.Gulp是一个基于流的自动化构建工具。除了可以管理和执行任务，还支持监听文件、读写文件

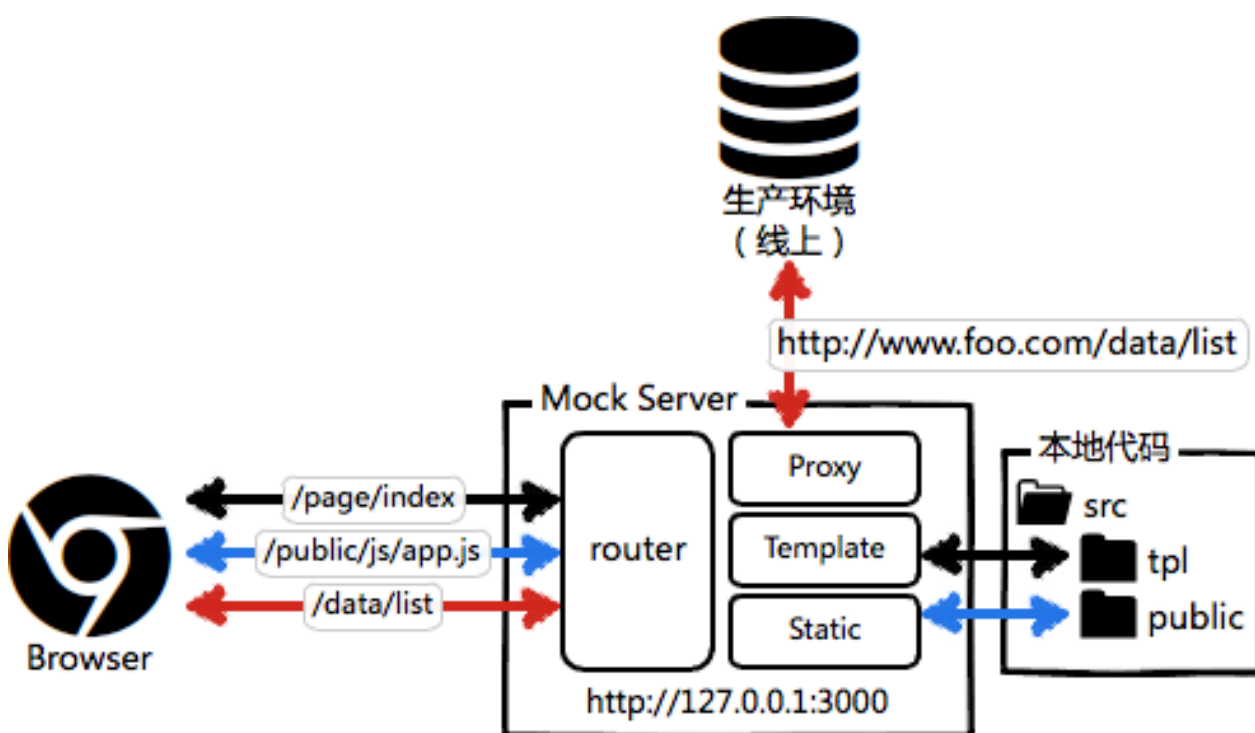
3.FIS称自己是可能是史上最强大的前端工程化方案，其实也是不无道理的。因为它不但解决前端开发中自动化工具、性能优化、模块化框架、开发规范、代码部署、开发流程等问题还有一套完整的后端解决方案。但是此项目在百度内部，且维护成本不高已看衰。不过它的经典前端工程化的思想完全能够让我们进行借鉴。

4.Rollup是一个和Webpack很类似但专注于ES6的模块打包工具。它的亮点在于，能针对ES6源码进行Tree Shaking，以去除那些已被定义但没被使用的代码并进行Scope Hoisting，以减小输出文件的大小和提升运行性能。然而Rollup的这些亮点随后就被Webpack模仿和实现。不支持Code Splitting，但打包出来的代码更小、更快。

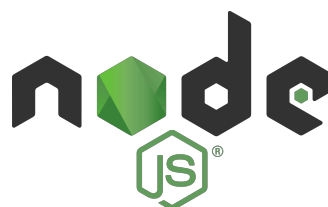
5.Parcel已快速打包、自动转换、零配置代码拆分、支持模块热替换和友好的错误记录迅速火热，但也被Webpack4抄袭未来怎么样，拭目以待。

5.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专注于构建模块化项目。

- 4.下图是一张比较粗略的工程化应用图，请你详细列举一下一个前端工程化所需要的准备步骤和所需的插件或技术。（10分）



🍊 我来说一下如果让我做我的技术栈吧，你们可以自己发挥：





PhantomCSS



- 5.请问如何书写一个Webpack的Plugin和Loader? (10分)

- 答：

🍊能够自己编写一个Loader和Plugin是面试的时候对一个高级前端的必备要求：

1.webpack的loader是一个node module 导出的一个用于转化加载的资源的功能，它的过程是从右到左，链式执行的，如果不关心上一个返回结果只关心原输入值可以用pitch。

```
module.exports = function(source) {
  this.cacheable(); //开启缓存
  //doSomeAsyncOperation(content, function(err, result) {
    //this.callback(null, result); //如果你有一些异步的操作
  //});
```

```
return source;

};
```

2.如何编写一个Plugin大家的印象应该是非常深刻的吧 😊

```
function htmlAfterWebpackPlugin(options){
    //打包的数据
    this.options = options;
}

htmlAfterWebpackPlugin.prototype.apply = function(compiler){
    compiler.plugin('compilation',function(compilation){
        compilation.plugin( '钩子函数' ,function(htmlPluginData,callback){})
    })
})
```

- 6.描述在你使用Webpack的过程中，你对它进行了哪些优化的点？（5分）

- 答：

🍊 Webpack确实很强大，但是也有很多弱点，我们一起分析下：

1.打包太慢 配置繁琐 API太多 体积巨大 难调试 日志多

2.来吧，这么几个核心的概念你要是不会我觉得你对不起我。

entry: 就是告诉webpack从哪里开始并通过依赖关系图得知要哪些文件要打包

output: 指定构建好的资源文件如何写入到指定目录，只能有一个出口 (path, publicPath, filename)

loaders: 加载器，加载器用于将资源代码转换成模块。它允许您在导入或“加载”它们时预处理文件，支持各种语言和预处理器编写模块(jsx=>js, less=>css, img=>base64等)

plugins: 插件，服务于编译期间，是一个具有apply属性的JavaScript对象，其apply属性会被webpack compiler调用，并且compiler对象可在整个编译生命周期访问

chunk：被entry所依赖的额外的代码块，同样可以包含一个或者多个文件，所有的资源都以模块的形式存在

modules: webpack中放置loaders的地方，可集中在rules数组中进行管理

resolve：设置对资源的寻址和解析规则，可通过alias和extensions来进行解析规则定制

externals：从输出的 bundle 中排除依赖，多在 library 开发或者加载中使用

manifest：文件清单,当编译器(compiler)开始执行、解析和映射应用程序时，它会保留所有模块的详细要点,使用 manifest 中的数

3.合理使用externals，公用的组件库不进行打包，采用CDN的方式。

4.利用happypack做多核编译资源

5.利用dllPlugin做资源预编译


6.神器 Tree-shaking & Scope Hoisting & code splitting

7.激活代码热更新功能(HMR)

8.利用CPU多核进行压缩

- 7.请用X-Tag 实现一个你能力范围内 Web Components。(20分)

- 答：

 很多人会跟我老袁抱怨，这么老的库学它干嘛啊。他是三大框架的祖先：


```
xtag.register('x-shouter', {
  content: '<input type="text" />',
  lifecycle:{
    created: function(){
      alert(this.firstElementChild);
      // Alerts the input specified via the 'content' property
    }
  },
  methods: {
    shout: function(message){
      setTimeout(function(){
        alert(message);
      }, this.delay);
    }
  },
  accessors: {
    delay: {
      attribute: {}
    }
  }
});
```

```

},
events: {
  tap: function(){
    this.shout(this.firstElementChild.value);
  }
}
});


```

- 8.HTML未来实现Web Components都有哪些规范，能写出几句简单的demo么。(10分)

 前端组件化其实是个持续了很久的过程了，Web Components它本身不是一个规范，他是由W3C提出的另外4个规范的合集：

- 1.Shadow Dom 管理多DOM树的层级关系，更好的合成DOM，沙箱DOM。
- 2.Custom Elements 它提供了我们自定义元素的能力。
- 3.HTML Template 它允许我们在文档中申明一段HTML，他在浏览器的解析过程中不会有任何的副作用。
- 4.HTML Imports 一个HTML文件引入另一个HTML文件。

- 9.你知道什么是Jenkins和Travis CI么，请简要概述？(10分)

 其实到了真实地企业中很少让前端工程师来配置，但是你要让人家知道原来你的公司在用（即使你公司不用）你也懂它是什么，会让面试官觉得你知道什么是持续集成：

- 1.Jenkins是一个开源软件项目，是基于Java开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。
- 2.Travis CI 是目前新兴的开源持续集成构建项目，它与jenkins，GO的很明显的特别在于采用yaml格式，简洁清新独树一帜。目前大多数的github项目都已经移入到Travis CI的构建队列中，据说Travis CI每天运行超过4000次完整构建。

- 10.请简述Webpack运行原理。(10分)

答：请参照前端架构师启蒙课 第四节《Webpack技术内幕》所讲。梳理完整Webpack运行过程已经对生成的JS如何进行模块化的组织并完成一个自己的Webpack-cli。此题非常重要!!!