构建时使用Vue预渲染骨架屏

# 骨架屏介绍

* 骨架屏的作用主要是在网络请求较慢时，提供基础占位，当数据加载完成，恢复数据展示。这样给用户一种很自然的过渡，不会造成页面长时间白屏或者闪烁等情况。 常见的骨架屏实现方案有ssr服务端渲染和prerender两种解决方案！
* 使用骨架屏（Skeleton Screen）来实现瞬间加载；
* 预先定义好元素的尺寸来保证加载的稳定

# 实现思路

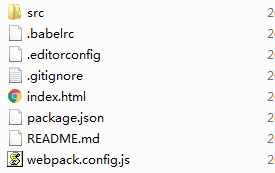
将骨架屏也看成路由组件，在构建时使用 Vue 预渲染功能，将骨架屏组件的渲染结果 HTML 片段插入 HTML 页面模版的挂载点中，将样式内联到 head 标签中。这样等前端渲染完成时，Vue 将使用客户端混合，把挂载点中的骨架屏内容替换成真正的页面内容。

# 具体实现一

## 1）使用 vue-cli 搭配的 webpack-simple项目：

vue init webpack-simple vue-skeleton

* 创建完目录结构如下



## 2）安装依赖，npm install 或者cnpm install

## 3）安装其他辅助依赖

npm install prerender-spa-plugin --save-dev --预渲染插件

npm install vue-server-renderer --save-dev --vue渲染器

npm install webpack-node-externals --save-dev -- 从node\_modules中排除所有模块

npm install html-minifier --save-dev --html压缩

## 4）修改package.json里面scripts添加

"skeleton": "webpack --config ./webpack.skeleton.conf.js && node skeleton.js"

## 5）在 /src 目录下新建一个 Skeleton.vue 文件

<template>

<div class="skeleton page">

<div class="skeleton-nav"></div>

<div class="skeleton-swiper"></div>

<ul class="skeleton-tabs">

<li v-for="i in 8" class="skeleton-tabs-item"><span></span></li>

</ul>

<div class="skeleton-banner"></div>

<div v-for="i in 6" class="skeleton-productions"></div>

</div>

</template>

<style>

.skeleton {

position: relative;

height: 100%;

overflow: hidden;

padding: 15px;

box-sizing: border-box;

background: #fff;

}

.skeleton-nav {

height: 45px;

background: #eee;

margin-bottom: 15px;

}

.skeleton-swiper {

height: 160px;

background: #eee;

margin-bottom: 15px;

}

.skeleton-tabs {

list-style: none;

padding: 0;

margin: 0 -15px;

display: flex;

flex-wrap: wrap;

}

.skeleton-tabs-item {

width: 25%;

height: 55px;

box-sizing: border-box;

text-align: center;

margin-bottom: 15px;

}

.skeleton-tabs-item span {

display: inline-block;

width: 55px;

height: 55px;

border-radius: 55px;

background: #eee;

}

.skeleton-banner {

height: 60px;

background: #eee;

margin-bottom: 15px;

}

.skeleton-productions {

height: 20px;

margin-bottom: 15px;

background: #eee;

}

</style>

## 6）接下来，再新建一个 skeleton.entry.js 入口文件

// src/entry-skeleton.js

import Vue from 'vue'

import Skeleton from './Skeleton';

// 创建一个骨架屏 Vue 实例

export default new Vue({

components: {

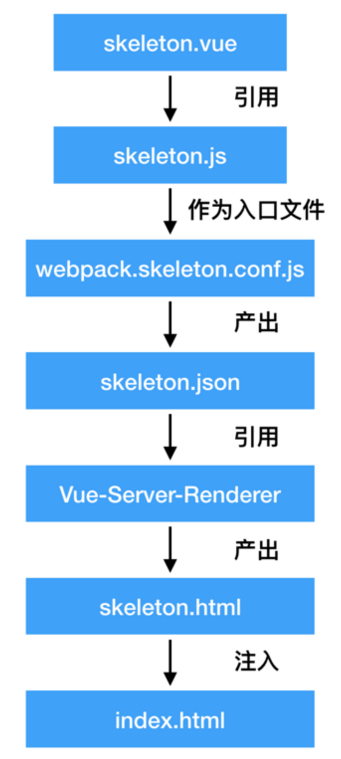
Skeleton

},

template: '<skeleton />'

});

## **7）接下来就是关键插件 vue-server-renderer，该插件本用于服务端渲染，但是在这个例子里，我们主要利用它能够把 .vue 文件处理成 html 和 css 字符串的功能，来完成骨架屏的注入，流程如下**：



## 8）此时需在根目录新建一个 webpack.skeleton.conf.js 文件，以专门用来进行骨架屏的构建。

const path = require('path')

const webpack = require('webpack')

const nodeExternals = require('webpack-node-externals')

const VueSSRServerPlugin = require('vue-server-renderer/server-plugin')

module.exports = {

target: 'node',

entry: {

skeleton: './src/skeleton.entry.js'

},

output: {

path: path.resolve(\_\_dirname, './dist'),

publicPath: '/dist/',

filename: '[name].js',

libraryTarget: 'commonjs2'

},

module: {

rules: [

{

test: /\.css$/,

use: [

'vue-style-loader',

'css-loader'

]

},

{

test: /\.vue$/,

loader: 'vue-loader'

}

]

},

externals: nodeExternals({

whitelist: /\.css$/

}),

resolve: {

alias: {

'vue$': 'vue/dist/vue.esm.js'

},

extensions: ['\*', '.js', '.vue', '.json']

},

plugins: [

new VueSSRServerPlugin({

filename: 'skeleton.json'

})

]

}

## 9）生成一个 skeleton.json 文件

* 运行npm run skeleton
* 此时会在在 /dist 目录下生成一个 skeleton.json 文件
* 这个文件在记载了骨架屏的内容和样式，会提供给 vue-server-renderer 使用

## 10）在根目录下新建一个 skeleton.js ，该文件即将被用于往 index.html 内插入骨架屏。

const fs = require('fs')

const { resolve } = require('path')

const createBundleRenderer = require('vue-server-renderer').createBundleRenderer

// 读取`skeleton.json`，以`index.html`为模板写入内容

const renderer = createBundleRenderer(resolve(\_\_dirname, './dist/skeleton.json'), {

template: fs.readFileSync(resolve(\_\_dirname, './index.html'), 'utf-8')

})

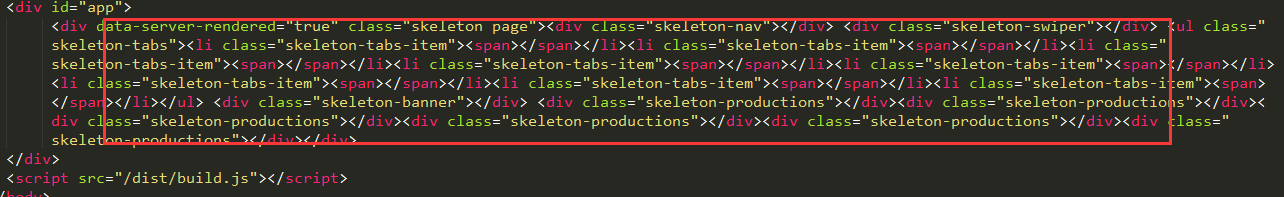
// 把上一步模板完成的内容写入（替换）`index.html`

renderer.renderToString({}, (err, html) => {

fs.writeFileSync('index.html', html, 'utf-8')

})

## 11）运行node skeleton.js ，就可以完成骨架屏的注入了。运行效果如下：



## 12）运行 npm run dev 则能看到骨架效果



# 四、具体实现二

## 1）使用 vue-cli 搭配模板例子

## 2）安装完了依赖以后， npm run dev 去运行项目

## 3）安装webpack 插件：vue-skeleton-webpack-plugin

* 使用 Vue 预渲染骨架屏
* 将骨架屏渲染结果插入 HTML 模版中
* 开发模式下插入各个骨架屏路由

安装插件npm install vue-skeleton-webpack-plugin

## 4）使用 Vue 预渲染骨架屏

使用 Vue 的预渲染功能渲染骨架屏组件，首先需要创建一个仅使用骨架屏组件的入口文件，

src/目录下建一个entry-skeleton.js

// src/entry-skeleton.js

import Vue from 'vue'

import Skeleton from './Skeleton';

// 创建一个骨架屏 Vue 实例

export default new Vue({

components: {

Skeleton

},

template: '<skeleton />'

});

## 5）创建Skeleton.vue，模板举例同上具体实现的模板

## 6）接下来创建一个用于服务端渲染的 webpack 配置对象，将刚创建的入口文件指定为 entry 依赖入口：

'use strict';

// build/webpack.skeleton.conf.js

const path = require('path')

const merge = require('webpack-merge')

const baseWebpackConfig = require('./webpack.base.conf')

const nodeExternals = require('webpack-node-externals')

const utils = require('./utils')

const config = require('../config')

const isProduction = process.env.NODE\_ENV === 'production'

const sourceMapEnabled = isProduction

? config.build.productionSourceMap

: config.dev.cssSourceMap

function resolve(dir) {

return path.join(\_\_dirname, dir)

}

let skeletonWebpackConfig = merge(baseWebpackConfig, {

target: 'node',

devtool: false,

entry: {

app: resolve('../src/entry-skeleton.js')

},

output: Object.assign({}, baseWebpackConfig.output, {

libraryTarget: 'commonjs2'

}),

externals: nodeExternals({

whitelist: /\.css$/

}),

plugins: []

})

// 启用extract-text-webpack-plugin插件

skeletonWebpackConfig.module.rules[0].options.loaders = utils.cssLoaders({

sourceMap: sourceMapEnabled,

extract: true

}),

module.exports = skeletonWebpackConfig

## 7）webpack.dev.conf.js配置

// build/webpack.dev.conf.js

* 引入已经安装的vue-skeleton-webpack-plugin插件

const SkeletonWebpackPlugin = require('vue-skeleton-webpack-plugin')

* 在plugins下添加插件

new SkeletonWebpackPlugin({ // 插件

webpackConfig: require('./webpack.skeleton.conf')

})

## 8）webpack.prod.conf.js配置

// build/webpack.prod.conf.js

* 引入已经安装的vue-skeleton-webpack-plugin插件

const SkeletonWebpackPlugin = require('vue-skeleton-webpack-plugin')

* 在plugins下添加插件

new SkeletonWebpackPlugin({ // 插件

webpackConfig: require('./webpack.skeleton.conf')

})

最后留一个思考：

一个VUE实现多页面骨架屏VUE