## **1.初始化项目**

$ mkdir zhufengskeleton

$ cd zhufengskeleton

$ npm init -y

$ cnpm i css-tree @babel/core @babel/preset-env @babel/preset-react babel-loader cross-env fs-extra html-webpack-plugin webpack webpack-cli webpack-dev-server -D

$ cnpm i react react-dom -S

$ cnpm i puppeteer -D

## **2.React项目构建**

### **2.1 webpack.config.js**

**const** HtmlWebpackPlugin = require('html-webpack-plugin');**const** {resolve} = require('path');module.exports = {

mode:'development',

devtool:false,

entry: "./src/index.js",

output: {

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

filename: "main.js"

},

module: {

rules: [{

test: /\.js$/,

use: [

{

loader:'babel-loader',

options:{

presets:["@babel/preset-env","@babel/preset-react"]

}

}

],

exclude: /node\_modules/

}]

},

devServer: {

contentBase: resolve(\_\_dirname,'dist')

},

plugins: [

**new** HtmlWebpackPlugin({

template: './src/index.html'

})

]

}

### **2.2 src\index.js**

src\index.js

**import** React **from** 'react';**import** ReactDOM **from** 'react-dom';

ReactDOM.render((

<div>

<img src="http://img.zhufengpeixun.cn/zhufengjg.jpg" width="100%"></img>

<button>点我点我</button>

</div>

),document.getElementById('root'));

### **2.3 src\index.html**

src\index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>skeleton</title>

</head>

<body>

<div id="root"><!-- shell --></div>

</body>

</html>

## **3. 创建插件**

### **3.1 skeleton\index.js**

skeleton\index.js

**const** SkeletonPlugin = require('./SkeletonPlugin')module.exports = {

SkeletonPlugin

}

### **3.2 SkeletonPlugin.js**

skeleton\SkeletonPlugin.js

**const** PLUGIN\_NAME = 'SkeletonPlugin';**const** defaultOptions = {

}**class** **SkeletonPlugin** {

**constructor**(options){

**this**.options = {...defaultOptions,...options};

}

apply(compiler) {

compiler.hooks.done.tap(PLUGIN\_NAME, **async** () => {

console.log(PLUGIN\_NAME,'done');

})

}

}module.exports = SkeletonPlugin;

### **3.3 webpack.config.js**

const HtmlWebpackPlugin = require('html-webpack-plugin');

const {resolve} = require('path');+const {SkeletonPlugin} = require('./skeleton');

module.exports = {

mode:'development',

devtool:false,

entry: "./src/index.js",

output: {

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

filename: "main.js"

},

module: {

rules: [{

test: /\.js$/,

use: [

{

loader:'babel-loader',

options:{

presets:["@babel/preset-env","@babel/preset-react"]

}

}

],

exclude: /node\_modules/

}]

},

devServer: {

contentBase: resolve(\_\_dirname,'dist')

},

plugins: [

new HtmlWebpackPlugin({

template: './src/index.html'

}),+ new SkeletonPlugin({+ + })

]

}

## **4. 启动服务**

### **4.1 webpack.config.js**

webpack.config.js

const HtmlWebpackPlugin = require('html-webpack-plugin');

const {resolve} = require('path');

const {SkeletonPlugin} = require('./skeleton');

module.exports = {

mode:'development',

devtool:false,

entry: "./src/index.js",

output: {

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

filename: "main.js"

},

module: {

rules: [{

test: /\.js$/,

use: [

{

loader:'babel-loader',

options:{

presets:["@babel/preset-env","@babel/preset-react"]

}

}

],

exclude: /node\_modules/

}]

},

devServer: {

contentBase: resolve(\_\_dirname,'dist')

},

plugins: [

new HtmlWebpackPlugin({

template: './src/index.html'

}),

new SkeletonPlugin({+ staticDir: resolve(\_\_dirname,'dist'),+ port:8000,+ origin:'http://localhost:8000'

})

]

}

### **4.2 SkeletonPlugin.js**

skeleton\SkeletonPlugin.js

const PLUGIN\_NAME = 'SkeletonPlugin';+const Server = require('./Server');

const defaultOptions = {

}

class SkeletonPlugin {

constructor(options){

this.options = {...defaultOptions,...options};

}

apply(compiler) {

compiler.hooks.done.tap(PLUGIN\_NAME, async () => {+ await this.startServer(); + await this.server.close();

})

}+ async startServer(){+ this.server = new Server(this.options);+ await this.server.listen();+ }

}

module.exports = SkeletonPlugin;

### **4.3 Server.js**

skeleton\Server.js

**const** http = require('http')**const** express = require('express');**class** **Server** {

**constructor**(options) {

**this**.options = options;

}

listen() {

**const** app = **this**.app = express();

app.use('/',express.static(**this**.options.staticDir));

**this**.listenServer = http.createServer(app);

**return** **new** Promise( (resolve) =>{

**this**.listenServer.listen(**this**.options.port, () => {

console.log(`server listen at port: ${**this**.options.origin}`);

resolve();

})

});

}

**async** close() {

**return** **new** Promise( (resolve) =>{

**this**.listenServer.close(() => {

console.log('server closed!');

resolve();

})

});

}

}module.exports = Server;

## **5. 启动puppeteer**

### **5.1 webpack.config.js**

const HtmlWebpackPlugin = require('html-webpack-plugin');

const {resolve} = require('path');

const {SkeletonPlugin} = require('./skeleton');

module.exports = {

mode:'development',

devtool:false,

entry: "./src/index.js",

output: {

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

filename: "main.js"

},

module: {

rules: [{

test: /\.js$/,

use: [

{

loader:'babel-loader',

options:{

presets:["@babel/preset-env","@babel/preset-react"]

}

}

],

exclude: /node\_modules/

}]

},

devServer: {

contentBase: resolve(\_\_dirname,'dist')

},

plugins: [

new HtmlWebpackPlugin({

template: './src/index.html'

}),

new SkeletonPlugin({

staticDir: resolve(\_\_dirname,'dist'),

port:8000,

origin:'http://localhost:8000',+ device: 'iPhone 6'

})

]

}

### **5.2 SkeletonPlugin.js**

skeleton\SkeletonPlugin.js

const PLUGIN\_NAME = 'SkeletonPlugin';

const Server = require('./Server');+const Skeleton = require('./Skeleton');

const defaultOptions = {

}

class SkeletonPlugin {

constructor(options){

this.options = {...defaultOptions,...options};

}

apply(compiler) {

compiler.hooks.done.tap(PLUGIN\_NAME, async () => {

await this.startServer();+ this.skeleton= new Skeleton(this.options);+ await this.skeleton.initialize();+ const skeletonHtml = await this.skeleton.genHtml(this.options.origin);+ console.log('skeletonHtml',skeletonHtml);+ await this.skeleton.destroy();

await this.server.close();

})

}

async startServer(){

this.server = new Server(this.options);

await this.server.listen();

}

}

module.exports = SkeletonPlugin;

### **5.3 Skeleton.js**

skeleton\Skeleton.js

**let** puppeteer = require('puppeteer');**class** **Skeleton** {

**constructor**(options = {}) {

**this**.options = options

}

**async** initialize() {

**this**.browser = **await** puppeteer.launch({ headless: false });

}

**async** newPage() {

**const** { device } = **this**.options;

**const** page = **await** **this**.browser.newPage();

**await** page.emulate(puppeteer.devices[device]);

**return** page;

}

**async** genHtml(url) {

**const** page = **await** **this**.newPage()

**const** response = **await** page.goto(url, { waitUntil: 'networkidle2' });

**if** (response && !response.ok()) {

**throw** **new** Error(`${response.status} on ${url}`)

}

**return** 'html';

}

**async** destroy() {

**if** (**this**.browser) {

**await** **this**.browser.close();

**this**.browser = null

}

}

}module.exports = Skeleton;

## **6. 截取骨架内容**

### **6.1 SkeletonPlugin.js**

skeleton\SkeletonPlugin.js

const PLUGIN\_NAME = 'SkeletonPlugin';

const Server = require('./Server');

const Skeleton = require('./Skeleton');+const {resolve} = require('path');+const {readFileSync,writeFileSync} = require('fs');

const defaultOptions = {

}

class SkeletonPlugin {

constructor(options){

this.options = {...defaultOptions,...options};

}

apply(compiler) {

compiler.hooks.done.tap(PLUGIN\_NAME, async () => {

await this.startServer();

this.skeleton= new Skeleton(this.options);

await this.skeleton.initialize();

const skeletonHtml = await this.skeleton.genHtml(this.options.origin);+ const originPath = resolve(this.options.staticDir,'index.html');+ const orgiginHtml = await readFileSync(originPath, 'utf8');+ const finalHtml = orgiginHtml.replace('<!-- shell -->',skeletonHtml);+ await writeFileSync(originPath,finalHtml,'utf8');+ await this.skeleton.destroy();+ await this.server.close();+ process.exit(0);

})

}

async startServer(){

this.server = new Server(this.options);

await this.server.listen();

}

}

module.exports = SkeletonPlugin;

### **6.2 Skeleton.js**

skeleton\Skeleton.js

let puppeteer = require('puppeteer');+let {readFileSync} = require('fs');+let {resolve} = require('path');+let {sleep} = require('./utils');

class Skeleton {

constructor(options = {}) {

this.options = options

}

async initialize() {

this.browser = await puppeteer.launch({ headless: false });

}

async newPage() {

const { device } = this.options;

const page = await this.browser.newPage();

await page.emulate(puppeteer.devices[device]);

return page;

}+ async makeSkeleton(page) {+ const { defer = 5000 } = this.options;+ const scriptContent = await readFileSync(resolve(\_\_dirname, 'skeletonScript.js'), 'utf8');+ await page.addScriptTag({ content: scriptContent })+ await sleep(defer);+ await page.evaluate((options) => {+ Skeleton.genSkeleton(options);+ }, this.options)+ }

async genHtml(url) {

const page = await this.newPage()

const response = await page.goto(url, { waitUntil: 'networkidle2' });

if (response && !response.ok()) {

throw new Error(`${response.status} on ${url}`)

}+ await this.makeSkeleton(page);+ const { styles, html } = await page.evaluate(() => Skeleton.getHtmlAndStyle());+ let result = `+ <style>${styles.join('\n')}</style>+ ${html}+ `;+ return Promise.resolve(result);

}

async destroy() {

if (this.browser) {

await this.browser.close()

this.browser = null

}

}

}

module.exports = Skeleton;

### **6.3 skeletonScript.js**

skeleton\skeletonScript.js

window.Skeleton = (**function** () {

**const** $$ = document.querySelectorAll.bind(document);

**const** REMOVE\_TAGS = ['title', 'meta', 'style','script'];

**function** **genSkeleton**(options = {}) {

}

**function** **getHtmlAndStyle**() {

**const** styles = Array.from($$('style')).map(style => style.innerHTML || style.innerText);

Array.from($$(REMOVE\_TAGS.join(','))).forEach(ele => ele.parentNode.removeChild(ele));

**const** html = document.body.innerHTML;

**return** { html, styles };

}

**return** {genSkeleton,getHtmlAndStyle};

}());

### **6.4 utils.js**

skeleton\utils.js

**function** **sleep**(duration) {

**return** **new** Promise((resolve) => {

setTimeout(resolve, duration)

})

}module.exports = {

sleep

}

## **7. 元素转换**

### **7.1 webpack.config.js**

webpack.config.js

const HtmlWebpackPlugin = require('html-webpack-plugin');

const {resolve} = require('path');

const {SkeletonPlugin} = require('./skeleton');

module.exports = {

mode:'development',

devtool:false,

entry: "./src/index.js",

output: {

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

filename: "main.js"

},

module: {

rules: [{

test: /\.js$/,

use: [

{

loader:'babel-loader',

options:{

presets:["@babel/preset-env","@babel/preset-react"]

}

}

],

exclude: /node\_modules/

}]

},

devServer: {

contentBase: resolve(\_\_dirname,'dist')

},

plugins: [

new HtmlWebpackPlugin({

template: './src/index.html'

}),

new SkeletonPlugin({

staticDir: resolve(\_\_dirname,'dist'),

port:8000,

origin:'http://localhost:8000',

device: 'iPhone 6',+ image: {+ color: '#EFEFEF',+ },+ button: {+ color: '#EFEFEF',+ }

})

]

}

### **7.2 skeletonScript.js**

skeleton\skeletonScript.js

window.Skeleton = (function () {+ const SMALLEST\_BASE64 = 'data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA7';+ const CLASS\_NAME\_PREFEX = 'sk-';

const $$ = document.querySelectorAll.bind(document);

const REMOVE\_TAGS = ['title', 'meta', 'style', 'script'];+ const styleCache = new Map();+ const setAttributes = (ele, attrs) => {+ Object.keys(attrs).forEach(k => ele.setAttribute(k, attrs[k]));+ };+ const addStyle = (selector, rule) => {+ if (!styleCache.has(selector)) {+ styleCache.set(selector, rule)+ }+ }+ function imgHandler(ele, options={}) {+ const {width, height} = ele.getBoundingClientRect();+ const attrs = {+ width,+ height,+ src: SMALLEST\_BASE64+ };+ setAttributes(ele, attrs);+ const className = CLASS\_NAME\_PREFEX + 'image';+ const rule = `{ background: ${options.color} !important;}`;+ addStyle(`.${className}`, rule);+ ele.classList.add(className)+ }+ function buttonHandler(ele,options={}) {+ const classname = CLASS\_NAME\_PREFEX + 'button'+ const rule = `{+ color: ${options.color} !important;+ background: ${options.color} !important;+ border: none !important;+ box-shadow: none !important;+ }`+ addStyle(`.${classname}`, rule)+ ele.classList.add(classname)+ }

function genSkeleton(options = {}) {+ const rootElement = document.documentElement;+ ;(function traverse(options) {+ let { button, image } = options;+ const buttons = [];+ const imgs = [];+ ;(function preTraverse(ele) {+ if (ele.children && ele.children.length > 0) {+ Array.from(ele.children).forEach(child => preTraverse(child))+ }+ if (ele.tagName === 'BUTTON') {+ return buttons.push(ele);+ }+ if (ele.tagName === 'IMG') {+ return imgs.push(ele)+ }+ })(rootElement);+ buttons.forEach(e => buttonHandler(e, button))+ imgs.forEach(e => imgHandler(e, image));+ })(options);+ let rules = ''+ for (const [selector, rule] of styleCache) {+ rules += `${selector} ${rule}\n`;+ }+ const styleEle = document.createElement('style')+ styleEle.innerHTML = rules;+ document.head.appendChild(styleEle)

}

function getHtmlAndStyle() {

const styles = Array.from($$('style')).map(style => style.innerHTML || style.innerText);

Array.from($$(REMOVE\_TAGS.join(','))).forEach(ele => ele.parentNode.removeChild(ele));

const html = document.body.innerHTML;

return { html, styles };

}

return { genSkeleton, getHtmlAndStyle };

}());

## **8. cssTree**

* [astexplorer](https://astexplorer.net/)



### **8.1 cssTree.js**

**const** fs= require('fs')**const** path= require('path')**const** csstree = require('css-tree');**let** createCode = **async** **function** (scssFilePath) {

**let** cssString = fs.readFileSync(scssFilePath,'utf8')

**let** ast = csstree.parse(cssString);

csstree.walk(ast, **function** (node) {

**if** (node.type == 'Dimension' && node.unit =='px') {

node.value = node.value/75;

node.unit ='rem';

}

});

**let** output = csstree.generate(ast);

fs.writeFile(path.join(\_\_dirname,'output.css'), output, **function** () {

console.log('最终代码写入到output.css')

})

}**let** scssFilePath= path.join(\_\_dirname,'input.css');

createCode(scssFilePath);

### **8.2 input.css**

.avatar{

width: 750px;

}

### **8.3 output.css**

.avatar{width:10rem}

### **8.4 ast.json**

{

"type": "StyleSheet",

"loc": null,

"children": [

{

"type": "Rule",

"prelude": {

"type": "SelectorList",

"children": [

{

"type": "Selector",

"children": [

{

"type": "ClassSelector",

"name": "avatar"

}

]

}

]

},

"block": {

"type": "Block",

"children": [

{

"type": "Declaration",

"property": "width",

"value": {

"type": "Value",

"loc": null,

"children": [

{

"type": "Dimension",

"value": "750",

"unit": "px"

}

]

}

}

]

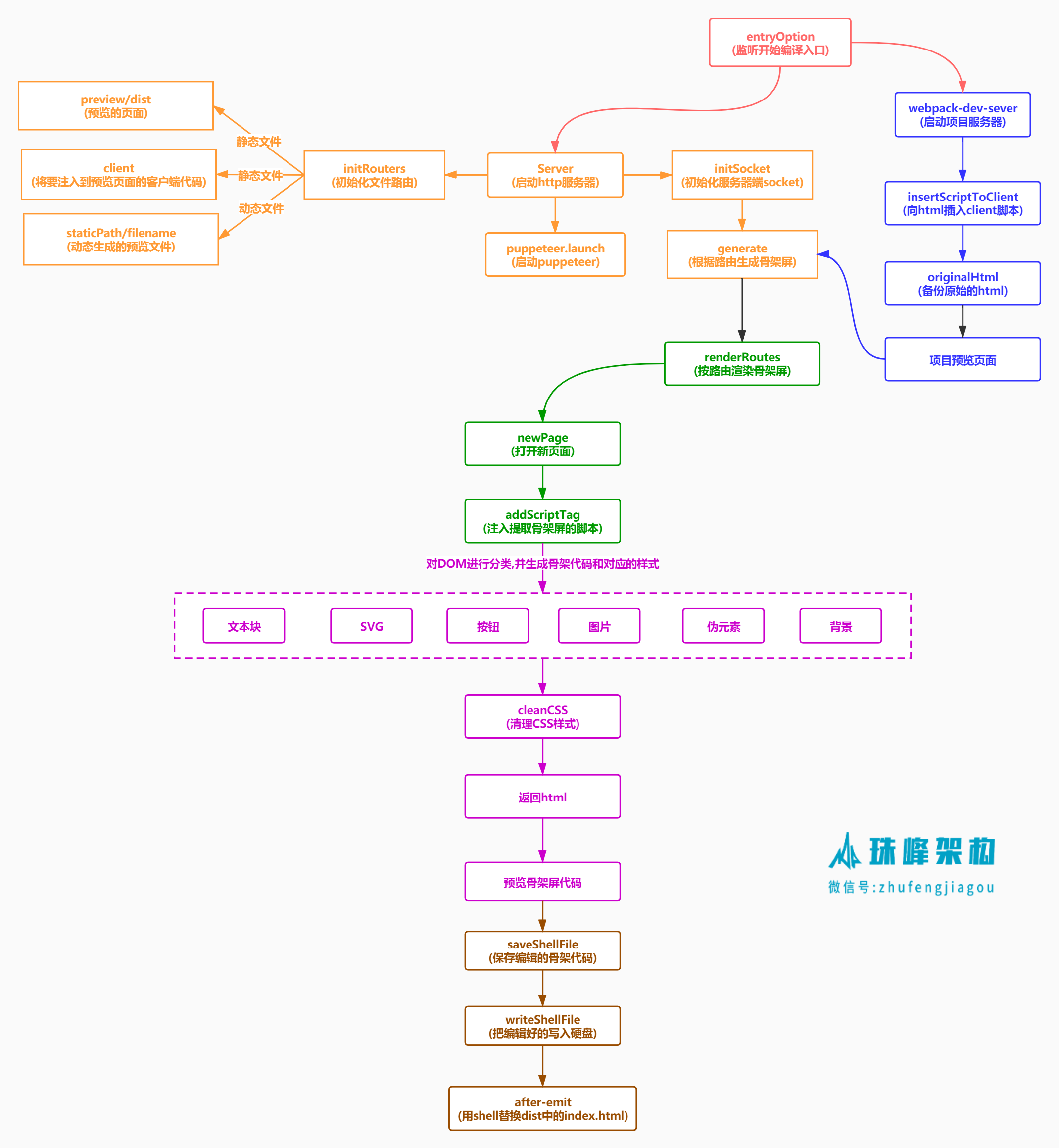
}

}

]

}

## **9. 参考**



Powered by [idoc](https://github.com/jaywcjlove/idoc" \t "https://zhufeng-document.vercel.app/html/_blank). Depe