link: null
title: 珠峰架构师成长计划
description: vite.config.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=63 sentences=88, words=495

1.Vite

- Vite (法语意为 "快速的"，发音 /vit/) (https://cn.vitejs.dev/)是下一代前端开发与构建工具
- 极速的服务启动 使用原生 ESM 文件，无需打包!
- 轻量快速的热重载 无论应用程序大小如何，都始终极快的模块热重载（HMR）
- 丰富的功能 对 TypeScript、JSX、CSS 等支持开箱即用。
- 优化的构建 可选 "多页应用" 或 "库" 模式的预配置 Rollup 构建
- 通用的插件 在开发和构建之间共享 Rollup-superset 插件接口。
- 完全类型化的 API 灵活的 API 和完整 TypeS

1.1 安装依赖

```
npm install vue  --save
npm install  @vitejs/plugin-vue  vite --save-dev
```

1.2 配置文件

vite.config.js

```
import { defineConfig } from "vite";
import vue from "@vitejs/plugin-vue";

export default defineConfig({
  plugins: [vue()],
});
```

1.3 package.json

package.json

```
{
  "name": "vite2-prepare",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build"
  },
  "dependencies": {
    "vue": "^3.0.5"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^1.2.4",
    "vite": "^2.4.0"
  }
}
```

1.4 index.html

index.html

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite Apptitle>
  head>
  <body>
    <div id="app">div>
    <script type="module" src="/src/main.js">script>
  body>
html>
```

1.5 src\main.js

src\main.js

```
import { createApp } from "vue";
import App from "./App.vue";
createApp(App).mount("#app");
```

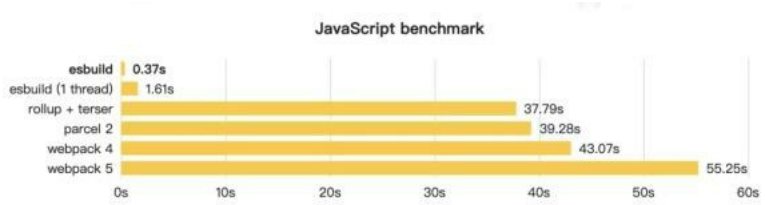
1.6 src\App.vue

src\App.vue

```
  App
```

2.esbuild 介绍

- ESbuild (https://esbuild.github.io/api/) 是一款基于 Golang 开发的一款打包工具，相比传统的打包工具，主打性能优势，在构建速度上可以快 10~100 倍



2.1 安装

```
npm install esbuild
```

2.2 使用

2.2.1 main.js

```
console.log("title");
```

2.2.2 esbuild.js

```
require("esbuild").buildSync({
  entryPoints: ["main.js"],
  outfile: "out.js",
});
```

2.3 内容类型

- content-types (https://esbuild.github.io/content-types/#javascript)
- 每个内容类型都有一个关联的 加载器，它告诉 esbuild 如何解释文件内容。默认情况下，某些文件扩展名已经为它们配置了加载器，尽管可以覆盖默认值

2.3.1 main.js

```
let title = <h1>hellohl>;
console.log("title");
```

2.3.2 esbuild.js

```
require("esbuild").buildSync({
  entryPoints: ["main.js"],
  bundle: true,
  loader: { ".js": "jsx" },
  outfile: "out.js",
});
```

2.4 plugin

- plugins (https://esbuild.github.io/plugins/#finding-plugins)
- plugins (https://github.com/esbuild/community-plugins)
- 一个 esbuild 插件是一个包含 name 和 setup 函数的对象
- 它们以数组的形式传递给构建 API 调用, setup函数在每次 BUILD API调用时都会运行一次

2.4.1 命名空间

- 每个模块都有一个关联的命名空间。默认情况下，esbuild 在 file 命名空间中运行，该命名空间对应于文件系统上的文件
- 但是 esbuild 也可以处理在文件系统上没有对应位置的"虚拟"模块
- 插件可用于创建虚拟模块。虚拟模块通常使用命名空间而不是 file 将它们与文件系统模块区分开来,通常命名空间特定于创建它们的插件

2.4.2 过滤器

- 每个回调都必须提供一个正则表达式作为过滤器。当路径与其过滤器不匹配时，esbuild 使用它来跳过调用回调
- 命名空间也可用于过滤。回调必须提供过滤正则表达式，但也可以选择提供命名空间以进一步限制匹配的路径

2.4.3 Resolve 回调

- 使用添加的回调 onResolve 将在 esbuild 构建的每个模块中的每个导入路径上运行
- 回调可以自定义 esbuild 如何进行路径解析。例如，它可以拦截导入路径并将它们重定向到其他地方。它还可以将路径标记为外部
- 回调可以返回而不提供将路径解析的责任传递给下一个回调的路径。对于给定的导入路径，onResolve 所有插件的所有回调都将按照它们注册的顺序运行，直到有人负责路径解析。如果没有回调返回路径，esbuild 将运行其默认路径解析逻辑

2.4.4 Resolve 参数

- 当 esbuild 调用由 注册的回调时 onResolve，它将为这些参数提供有关导入路径的信息：
- path 这是来自底层模块源代码的逐字未解析路径
- namespace 这是包含要解析的此导入的模块的名称空间

2.4.5 onLoad 回调

- onLoad 将为每个未标记为外部的唯一路径/命名空间对运行添加的回调
- 它的工作是返回模块的内容并告诉 esbuild 如何解释它
- 回调可以在不提供模块内容的情况下返回。在这种情况下，加载模块的责任被传递给下一个注册的回调。对于给定的模块，onLoad 所有插件的所有回调都将按照它们注册的顺序运行，直到有人负责加载模块。如果没有回调返回模块的内容，esbuild 将运行其默认的模块加载逻辑

2.4.6 onLoad 选项

- filter 每个回调都必须提供一个过滤器，它是一个正则表达式。当路径与此过滤器不匹配时，将跳过注册的回调
- namespace 这是可选的。如果提供，回调仅在提供的命名空间中的模块内的路径上运行

2.4.7 load 结果

- contents 将此设置为字符串以指定模块的内容。如果设置了此项，则不会针对此已解析路径运行更多加载回调。如果未设置，esbuild 将继续运行在当前回调之后注册的加载回调。然后，如果内容仍未设置，如果解析的路径在 file 命名空间中，esbuild 将默认从文件系统加载内容
- loader 这告诉 esbuild 如何解释内容。例如，js 加载器将内容解释为 JavaScript，css 加载器将内容解释为 CSS。js 如果未指定，则加载程序默认为。有关所有内置加载程序的完整列表，请参阅内容类型页面。

2.4.1 entry.js

```
import { Path } from "env";
console.log(`Path is ${Path}`);
```

2.4.2 env Plugin

```
let envPlugin = {
  name: "env",
  setup(build) {

    build.onResolve({ filter: /^env$/ }, (args) => ({
      path: args.path,
      namespace: "env-ns",
    }));

    build.onLoad({ filter: /.*/, namespace: "env-ns" }, () => ({
      contents: JSON.stringify(process.env),
      loader: "json",
    }));
  },
};

require("esbuild")
  .build({
    entryPoints: ["entry.js"],
    bundle: true,
    outfile: "out.js",
    plugins: [envPlugin],
  })
  .catch(() => process.exit(1));
```