mini-vite

1. EsModule

服务器端

```JavaScript
const Koa = require('koa')
const app = new Koa()
app.use(async (ctx) => {
  const {
    request: { url, query },
  } = ctx;
  console.log("url:" + url, "query type", query.type);
  // 首页
  if (url == "/") {
    ctx.type = "text/html";
    let content = fs.readFileSync("./index.html", "utf-8");
    ctx.body = content;
  }
})
app.listen(3000, () => {
  console.log('Vite Start ....')
})
```

新建页面 index.html

```HTML
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="icon" href="/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vite App</title>
</head>
<body>
  <h1>然叔 666</h1>
```

```
  <div id="app"></div>
  <script>
  </script>
  <script type="module" src="/src/main.js"></script>
</body>
</html>
```

新建/src/main.js

```Bash
console.log('main ....')
```

添加模块解析 /index.js

/src/moduleA

```Bash
export const str = "Hello Vite";
```

/src/main.js

```Bash
import { str } from "./moduelA.js";
console.log(str);
```

```Bash
else if (url.endsWith(".js")) {
    // js文件
    const p = path.resolve(__dirname, url.slice(1));
    ctx.type = "application/javascript";
    const content = fs.readFileSync(p, "utf-8");
   ctx.body = content
  }
```

添加依赖解析

> from ('./xxx')
>
> From ('yyyy') => from ('/@modules/yyyy')

```Bash
```

```
function rewriteImport(content) {
  return content.replace(/ from ['|"]([^'"]+)['|"]/g, function (s0, s1) {
    console.log("s", s0, s1);
    // . ../ /开头的，都是相对路径
    if (s1[0] !== "." && s1[1] !== "/") {
      return `from '/@modules/${s1}'`;
    } else {
      return s0;
    }
  });
}
// 添加模块改写
 ctx.body = rewriteImport(content);
```

第三方依赖支持

/src/main.js

Bash

```
import { createApp, h } from "vue";
const App = {
  render() {
    return h("div", null, [h("div", null, String("123"))]);
  },
};
createApp(App).mount("#app");
```

Bash

```
else if (url.startsWith("/@modules/")) {
    // 这是一个node_module里的东西
    const prefix = path.resolve(
      __dirname,
      "node_modules",
      url.replace("/@modules/", "")
    );
    const module = require(prefix + "/package.json").module;
    const p = path.resolve(prefix, module);
    const ret = fs.readFileSync(p, "utf-8");
    ctx.type = "application/javascript";
    ctx.body = rewriteImport(ret);
  }
```

SFC 组件支持

Bash

```
const compilerSfc = require("@vue/compiler-sfc"); // .vue
const compilerDom = require("@vue/compiler-dom"); // 模板

  else if (url.endsWith(".css")) {
    const p = path.resolve(__dirname, url.slice(1));
    const file = fs.readFileSync(p, "utf-8");
    const content = `
    const css = "${file.replace(/\n/g, "")}"
    let link = document.createElement('style')
    link.setAttribute('type', 'text/css')
    document.head.appendChild(link)
    link.innerHTML = css
    export default css
    `;
    ctx.type = "application/javascript";
    ctx.body = content;
  } else if (url.indexOf(".vue") > -1) {
    // vue单文件组件
    const p = path.resolve(__dirname, url.split("?")[0].slice(1));
    const { descriptor } = compilerSfc.parse(fs.readFileSync(p, "utf-8"));

    if (!query.type) {
      ctx.type = "application/javascript";
      // 借用vue自导的compile框架 解析单文件组件，其实相当于vue-loader做的事情
      ctx.body = `
${rewriteImport(
  descriptor.script.content.replace("export default ", "const __script =
)}
import { render as __render } from "${url}?type=template"
__script.render = __render
export default __script
    `;
  } else if (query.type === "template") {
      // 模板内容
      const template = descriptor.template;
      // 要在server端吧compiler做了
      const render = compilerDom.compile(template.content, { mode: "module"
        .code;
      ctx.type = "application/javascript";

      ctx.body = rewriteImport(render);
```

```
    }
  }
```