我叫杨村长

Github老炮：57code
https://github.com/57code

B站新星：Young村长
https://space.bilibili.com/480140591

掘金优秀作者：杨村长
https://juejin.cn/user/3251111749263 50

抖音小鲜肉：前端杨村长

WEBSITE

Development

VITE2项目工程化及原理剖析

DAY2

@杨村长

直播间定时送福利—领取彩蛋

第一天超值福利

到课就送《Vue3 & React17 进阶知识地图》
完课再送《2020web前端高频面试题全解析》

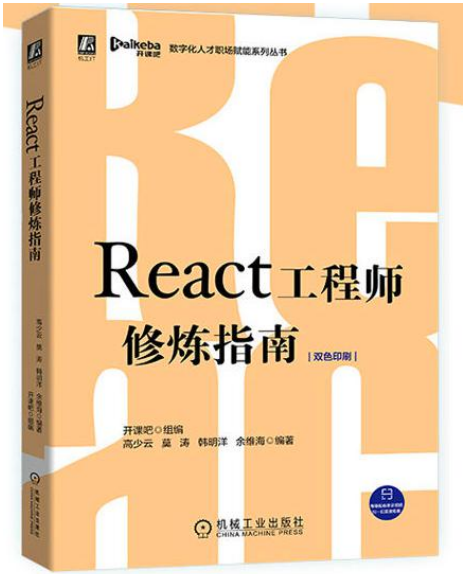
第二天超值福利(今日领取)

到课就送《Vue组件库源码解析》视频
完课再送《Vue3.0快速迁移指南》系列视频

2天全部完课的小伙伴，还有福利!

送《Vite2插件开发指南》系列视频，前端进阶必备!

每天抽取四名学员赠送
《React工程师修炼指南》



昨天内容回顾

DAY1 - 实战篇：
Vite工程化及插件编写指南

1、Vite工程化，提升工程化能力
2、后续能做的事情：继续搭建项目基础设施
3、搞清楚插件工作机制，深入理解Vite
4、会编写自己的插件，提升解决问题能力

今日学习目标

DAY2 - 原理篇：
Vite原理剖析及手写实现

1、搞清楚Vite工作原理，为什么那么快？
2、掌握node server开发，拓展思维
3、手写实现我们自己的Vite，提升内功
4、搞清楚vue3中编译器工作方式

工作原理

浏览器利用es module imports，关键变化是`index.html`中的入口文件导入方式

```
<script type="module" src="/src/main.js"></script>
```

- 第三方依赖模块预打包，并将导入地址修改为相对地址，从而统一所有请求为相对地址
- 启动一个开发服务器处理这些资源请求

```
import { createApp } from '/node_modules/.vite/vue.js?v=43d2c5b2'
import App from '/src/App.vue'

createApp(App).mount('#app')
```

vite需要根据请求资源类型做不同解析工作，比如`App.vue`，返回给用户的内容如下：

```
// 原先的script部分内容
import HelloWorld from '/src/components/HelloWorld.vue'

const __script = {
    name: 'App',
    components: {
        HelloWorld
    }
}

// 可见`template`部分转换为了一个模板请求，解析结果是一个渲染函数
import {render as __render} from "/src/App.vue?type=template"
// 将解析得到的render函数设置到组件配置对象上
__script.render = __render
__script.__hmrId = "/src/App.vue"
__script.__file = "/Users/yt/projects/vite-study/src/App.vue"
```

手写实现

第一步：创建一个node服务器，能够将`index.html`返回给浏览器。

项目根目录创建自定义服务器代码，kvite.js：

```
const Koa = require('koa')
const app = new Koa()

app.use(async (ctx) => {
  ctx.body = 'kkb vite'
})

app.listen(3000, () => {
  console.log('kvite start');
})
```

第二步：加载main.js

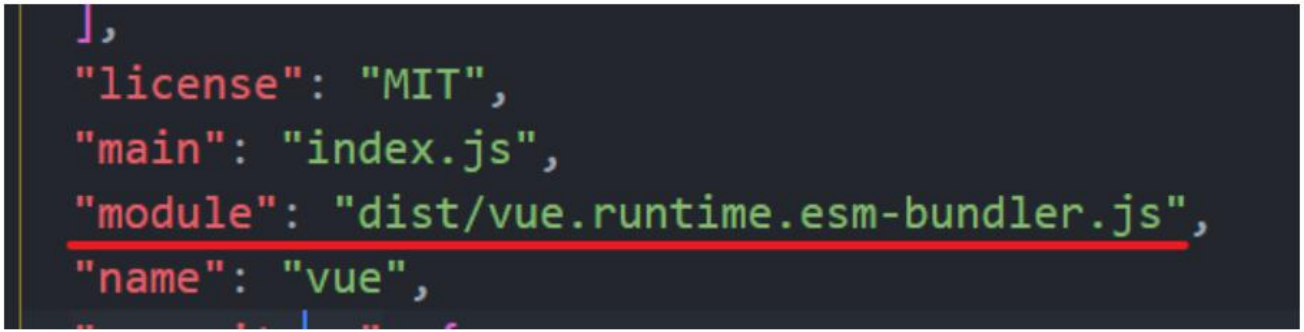
```
// 导入path
const path = require('path')

app.use(async (ctx) => {
  const url = ctx.request.url
  if (url === '/') {
    // ..
  } else if (url.endsWith('.js')) {
    // 获取js文件绝对路径，读取并返回
    const p = path.join(__dirname, url)
    ctx.type = 'text/javascript'
    ctx.body = fs.readFileSync(p, 'utf-8')
  }
})
```

第三步：裸模块路径替换，我们将`from 'vue'`替换为`from '/@modules/vue'`

```
function rewriteImport(content) {
  // 将传入内容中裸模块导入部分替换为/@modules/xx形式
  return content.replace(/ from ['|"]([^'"]+)['|"]/g, function(s0,
s1){
    // s0-待匹配字符串，s1-匹配的分组
    console.log(s0, s1);
    // 返回要替换的内容
    // 如果不是以. ../ /开头
    // 就是裸模块导入，替换这些path
    if (s1.startsWith('.') || s1.startsWith('/') ||
s1.startsWith('../')) {
      // 相对路径不做处理
      return s0
    } else {
      return ` from '/@modules/${s1}'`
    }
  })
}
```

第四步：处理依赖模块加载，例如`/@modules/vue`可以从`/node_modules/vue`中查找要加载的文件，目标文件在模块的`package.json`中有描述：



第五步：SFC请求处理，例如`App.vue`，可以使用`compiler-sfc`编译之

```
const compilerSfc = require("@vue/compiler-sfc");
```

```
else if (url.indexOf('.vue') > -1) {
  // 解析单文件组件相当于vue-loader做的事情
  // 转换script部分：将默认导出的组件对象转换为常量
  const p = path.resolve(__dirname, url.slice(1))
  const ret = compilerSfc.parse(fs.readFileSync(p, 'utf-8'))
  console.log(ret)
}
```

第六步：将得到的ast生成为JS代码

```
else if (url.indexOf('.vue') > -1) {
    // ...
    // 获取脚本部分内容
    const scriptContent = ret.descriptor.script.content
    // 转换内容中默认导出为一个常量，这样可以继续编辑它
    const script = scriptContent.replace('export default ', 'const
__script = ')
    // 返回给用户App.vue解析结果
    ctx.type = 'text/javascript'
    ctx.body = `
      // script中import仍需要重写
      ${rewriteImport(script)}
      // 注意此时并只解析了script部分，template部分转换为另一次请求单独处理
      // 为了能够和SFC请求区分，后面额外加上查询参数type=template
      import { render as __render } from '${url}?type=template'
      __script.render = __render
      export default __script
    `
}
```

第七步：处理模板请求，例如`./App.vue?type=template`，目标是编译模板为渲染函数

```
const { url, query } = ctx.request // 获取query
//...
else if (url.indexOf('.vue') > -1) {
    const p = path.resolve(__dirname, url.split("?")[0].slice(1));
    const ret = compilerSfc.parse(fs.readFileSync(p, 'utf-8'))

    if (!query.type) {}
    else if (query.type === 'template') {
      // 获取模板部分内容
      const template = ret.descriptor.template.content
      // 编译该模板为render函数
      const render = compilerDom.compile(template, { mode:
'module' }).code
      ctx.type = 'text/javascript'
      // 注意render中也有导入，需要重写import
      ctx.body = rewriteImport(render)
    }
}
```

今日总结

DAY2 - 原理篇：
Vite原理剖析及手写实现

1、搞清楚Vite工作原理，为什么那么快？
2、掌握node server开发，拓展思维
3、手写实现我们自己的Vite，提升内功
4、搞清楚vue3中编译器工作方式