代码分割

webpack 可将代码分割为 chunks 语块，运行时再加载

适用：抽离相同代码到一个共享块

脚本懒加载，初始下载代码更小

动态 import　import('./text.js').then(Text => {　})

①返回的是一个 Promise

②引入的是对应文件的 module.exports。

△ 在 webpack 中使用 ES-lint → webpack 在项目中集成使用 eslint 方式

①方案一：webpack 与 CI/CD 集成　在 CI/PIPELINE 中加入一个 lint pipline.

- 本地开发阶段增加 precommit 钩子

实现：安装 husky　增加 npm script　通过 lint-staged 增量检查修改的文件

②方案二：webpack 与 ESlint 集成

使用 eslint-loader. → 适合新项目

△ Webpack 实现 SSR 打包

SSR ⇒ 核心：减少请求　减少白屏时间，对 SEO 友好

代码实现思路：使用 react-dom/server 的 renderToString 方法将 React 渲染为字符串

服务端返回对应模板

webpack ssr 打包存在的问题

- 浏览器的全局变量（Node 中没有 document，window）

组件适配：将不兼容的组件根据打包环境进行适配

请求适配：将 fetch 或者 ajax 发送请求写法改成 isomorphic-fetch 或者 axios

- 样式问题：（Node.js 无法解析 css）

方案一：服务端打包通过 ignore-loader 忽略掉 css 的解析

方案二：将 style-loader 替换成 isomorphic-style-loader

△ webpack 优化打包日志　使用 friendly-errors-webpack-plugin → 优化构建提示效果

统计信息 stats：

errors-only：只发生错误时输出

minimal：只在发生错误或有新的编译时有输出

none：没有输出

normal：正常输出

verbose：全部输出

△ 构建异常与中断处理

①判断构建是否成功 ⇒ CI/CD 系统的 pipline 或发布系统需知当前构建状态

构建完后输入 echo $? 获取错误码 ⇒ 构建失败不为 0

Node.js　process.exit 规范：0 代表成功完成

②如何主动捕获并处理构建错误

webpack 在每次构建结束后会主动触发 done 这个 hook

process.exit 主动处理构建报错

可以在plugins中的事件处理函数执行

```
function() {
    this.hooks.done.tap('done', stats => {
        if (stats.compilation.error && stats.compilation.errors.length && process.argv.indexOf('--watch') == -1)
        {   console.log('build error')
            process.exit(); ——> 主动处理构建报错，添加错误码
                              └——> 在CI/CD系统中可以增加构建日志
        }
    })
}
}
```