组件传值总结

通用方法：

- vuex
- 创建公共的bus进行传值

一、父子组件传值

在父组件中引入子组件，确立父子关系

```
// 引入子组件
import Son from "./Son.vue";
```

给子组件命名

```
components: {
    // 子组件名
    Son, // Son:Son es6语法的简写
}
```

1.1 父组件给子组件传值

将子组件展示到页面中,通过定义属性给子组件传值

```
<App :toSon='msg'></App>
```

```
data () {
    return {
        msg: 'hello App,I am your father!!'
    }
},
```

子组件通过 `props` 接收父组件传来的值，这个值相当于子组件中的 `data` 可以直接使用

```
<!-- hello App,I am your father!! -->
<h3 @click='viewData'>{{toSon}}</h3>
```

直接在页面展示

```
props:['toSon'],
methods: {
    viewData () {
        console.log(this.toSon);// hello App,I am your father!!
    }
}
```
在组件中打印

1.2 子组件给父组件传值（自定义事件）

- 通过 `$emit()` 方法将值传给父元素

```
<button @click='hello()'>给父组件传值</button>
```

```
hello:function(){
// 给父组件传递数据，flag:数据名，数据的内容{a:2,b:4}
    this.$emit("flag",{a:2,b:4});
}
```

- 父组件通过自定义事件进行接收

```
<!--@flag自定义事件（注意，这里的receiveSonData不能加括号）-->
<Son @flag="receiveSonData"></Son>
```

```
receiveSonData:function (data) {
    console.log(data);// {a:2,b:4}
}
```

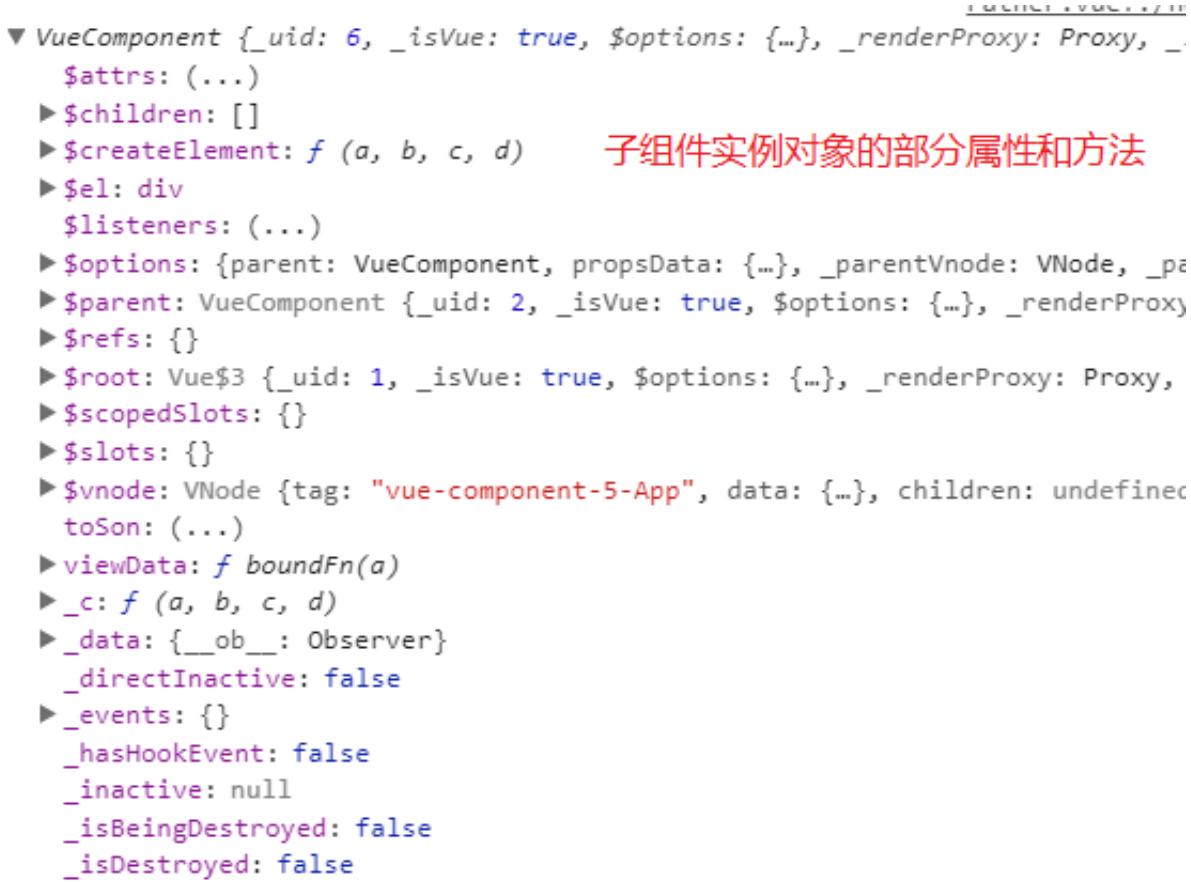
1.3 子组件给父组件传值（ `ref` ）

- 为html中展示的子组件添加ref属性

```
<App ref='app' :toSon='msg'></App>
```

```
viewAppData () {
// 在父组件中可以直接通过this.$refs.app调用子组件的方法和属性
console.log(this.$refs.app);//这里打印的是子组件的实例对象
}

```



二、通过bus来传值

2.1 bus传值的简单实现

- 新建一个bus.js文件，在需要相互传值的2个组件之中引入

```
import Vue from "vue";
var bus = new Vue();
export default bus;
```

- 通过$emit()发送数据

```
sendToBb:function(){
    // 向Bb传递数据
    bus.$emit("toBb",{name:"小明",age:"18"});
}
```

- 通过$on()来接收数据

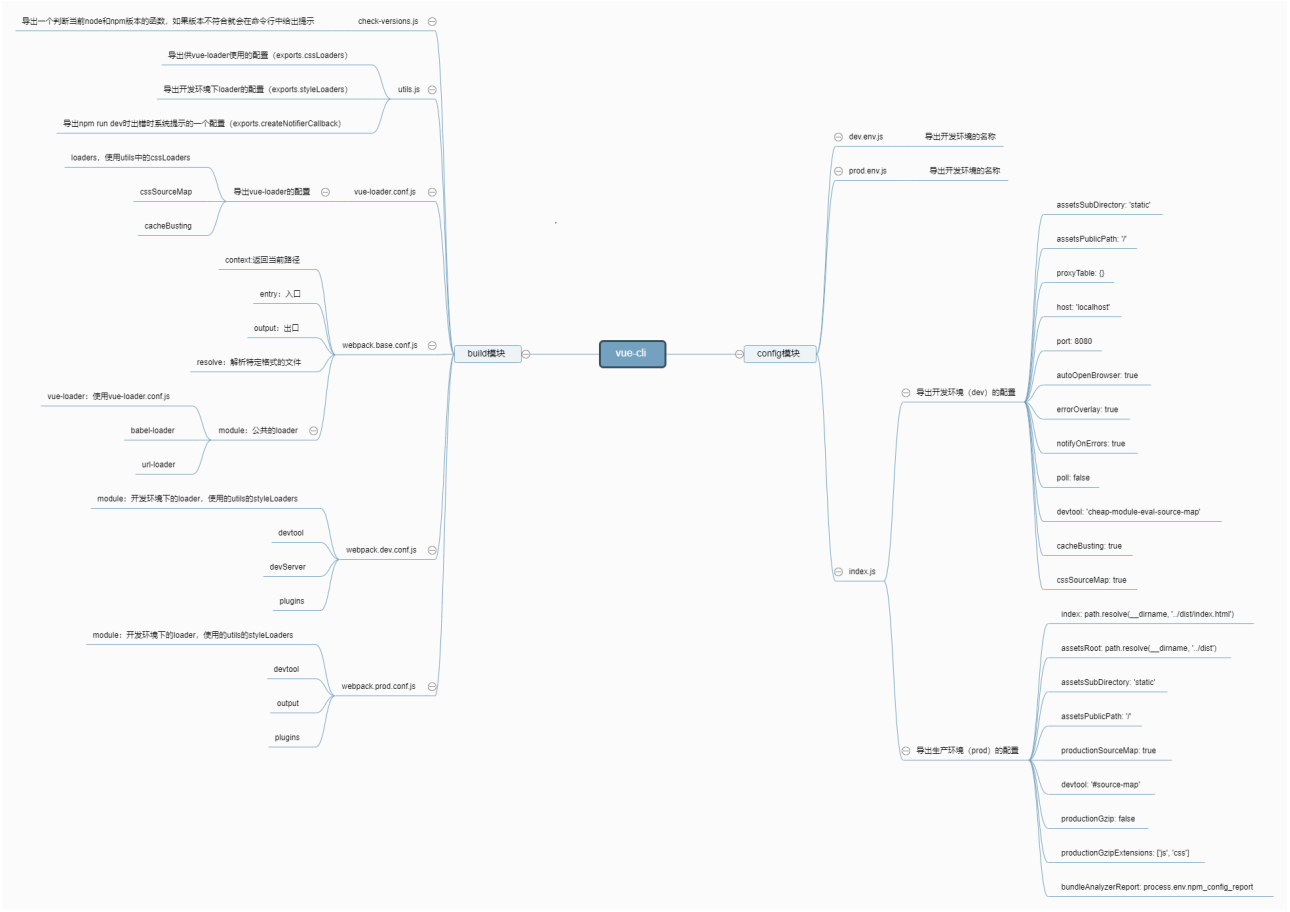
```
bus.$on("toBb",function(data) {
    console.log(data);//组件传来的数据
});
```

三、Vue-cli目录结构分析

```
|-- build                            // 项目构建(webpack)相关代码
|   |-- build.js                     // 生产环境构建代码
|   |-- check-version.js             // 检查node、npm等版本
|   |-- dev-client.js                // 热重载相关
|   |-- dev-server.js                // 构建本地服务器
|   |-- utils.js                     // 构建工具相关
|   |-- vue-loader.conf.js           // vue-loader基础配置
|   |-- webpack.base.conf.js         // webpack基础配置
|   |-- webpack.dev.conf.js          // webpack开发环境配置
|   |-- webpack.prod.conf.js         // webpack生产环境配置
|-- config                           // 项目开发环境配置
|   |-- dev.env.js                   // 开发环境变量
|   |-- index.js                     // 项目一些配置变量
|   |-- prod.env.js                  // 生产环境变量
|   |-- test.env.js                  // 测试环境变量
|-- node_modules                     // 存放 npm 下载的模块
|-- src    这是我们经常会用的一个文件夹   // 源码目录
|   |-- assets                      // 存放静态资源文件会被webpack处理解析为模块依赖
|   |-- components                    // vue公共组件
|   |-- store                         // vuex的状态管理
|   |-- router                        // 路由管理文件
|   |-- App.vue                       // 页面入口文件
|   |-- main.js                       // 程序入口文件，加载各种公共组件
|-- static                           // 静态文件，比如一些图片，json数据等
|-- test                             // 测试文件
|-- .babelrc                         // ES6语法编译配置
|-- .editorconfig                    // 定义代码格式
|-- .gitignore                       // git上传需要忽略的文件格式
|-- README.md                        // 项目说明
|-- favicon.ico
|-- index.html                       // 入口页面
|-- package.json                     // 项目基本信息

static放不会变动的文件 assets放可能会变动的文件。
```

脑图


vue-cli
build模块
check-versions.js
导出一个判断当前node和npm版本的函数，如果版本不符合就会在命令行中给出提示
utils.js
导出供vue-loader使用的配置（exports.cssLoaders）
导出开发环境下loader的配置（exports.styleLoaders）
导出npm run dev时出错时系统提示的一个配置（exports.createNotifierCallback）
vue-loader.conf.js
导出vue-loader的配置
loaders：使用utils中的cssLoaders
cssSourceMap
cacheBusting
webpack.base.conf.js
context:返回当前路径
entry：入口
output：出口
resolve：解析特定格式的文件
module：公共的loader
vue-loader：使用vue-loader.conf.js
babel-loader
url-loader
webpack.dev.conf.js
module：开发环境下的loader，使用的utils的styleLoaders
devtool
devServer
plugins
webpack.prod.conf.js
module：开发环境下的loader，使用的utils的styleLoaders
devtool
output
plugins
config模块
dev.env.js
导出开发环境的名称
prod.env.js
导出开发环境的名称
index.js
导出开发环境（dev）的配置
assetsSubDirectory: 'static'
assetsPublicPath: '/'
proxyTable: {}
host: 'localhost'
port: 8080
autoOpenBrowser: true
errorOverlay: true
notifyOnErrors: true
poll: false
devtool: 'cheap-module-eval-source-map'
cacheBusting: true
cssSourceMap: true
导出生产环境（prod）的配置
index: path.resolve(__dirname, '../dist/index.html')
assetsRoot: path.resolve(__dirname, '../dist')
assetsSubDirectory: 'static'
assetsPublicPath: '/'
productionSourceMap: true
devtool: '#source-map'
productionGzip: false
productionGzipExtensions: ['js', 'css']
bundleAnalyzerReport: process.env.npm_config_report