1. nuxt2.5.13启动警告：hough the "loose" option was set to "false" in your @babel/preset-env config, it will not be used for @babel/plugin-proposal-private-property-in-object since the "loose" mode option was set to "true" for @babel/plugin-proposal-private-methods.

解决方法：

```
// nuxt.config.js
module.exports = {
    ...
    build: {
        babel: {
            plugins: [
                ["@babel/plugin-proposal-private-property-in-object", { "loose": true }]
            ]
        }
    }
}
```

不知道为何在babelrc中写不行?

2. less，less-loader使用报错：this.getOptions is not a function

```
nuxt add less less-loader@5.0

npmi less less-loader -D
```

版本不匹配造成的，将less-loader版本降低一些，稳定版本。

3. nuxt中引入svg

```
npm i nuxt-svg-sprite-loader -D

// components/SvgIcon.vue
<template>
    <svg :class="svgClass" aria-hidden="true">
        <use :xlink:href="iconName" />
      </svg>
</template>
<script>
 export default {
    name: 'SvgIcon',
    props: {
        iconClass: {
          type: String,
          required: true
        },
        className: {
          type: String,
          default: ''
        }
    },
    computed: {
        //通过iconClass 获取svg文件名
        iconName() {
          return `#icon-${this.iconClass}`
        },
        svgClass() {
          if (this.className) {
            return 'svg-icon ' + this.className
```

```
          } else {
            return 'svg-icon'
          }
        }
    }
}
</script>
<style scoped>
.svg-icon {
  width: 1em;
  height: 1em;
  vertical-align: -0.15em;
  fill: currentColor;
  overflow: hidden;
}
</style>

// plugins/svg.js
// 无需注册注册组件，components下的组件会自动注册
//预请求svg组件(通过之前的nuxt-svg-sprite-loader加载)
const req = require.context('@/assets/icons/svg', false, /\.svg$/)
const requireAll = requireContext => requireContext.keys().map(requireContext)
requireAll(req)

// nuxt.config.js
module.exports = {
    ...
    modules: {
        ['nuxt-svg-sprite-loader', {
            symbolId: 'icon-[name]'
        }]
    },
    plugins:{
        '@/plugins/svg' //注册插件文件
    }
}
```

4. 打包优化

分析包的大小，只需要配置analyze

```
// nuxt.config.js
module.exports = {
    ...,
    build: {
        // 分析
        analyze: true,
        // 分包
        optimization:{
            splitChunks: {
                // B
                minSize: 10000,
                maxSize: 250000
            }
        }
```

```
    }
}
```

4. PC H5 判断

在middleware中写入判断,然后在nuxt.config.js中router中配置中间件

> 中间件允许您定义一个自定义函数运行在一个页面或一组页面渲染之前。

中间件执行流程顺序：

1. nuxt.config.js
2. 匹配布局
3. 匹配页面

5. postcss-px-to-viewport

```
"postcss-px-to-viewport": {
  unitToConvert: "px", // 要转化的单位
  viewportWidth: 750, // UI设计稿的宽度
  unitPrecision: 6, // 转换后的精度，即小数点位数
  propList: ["*"], // 指定转换的css属性的单位，*代表全部css属性的单位都进行转换
  viewportUnit: "vw", // 指定需要转换成的视窗单位，默认vw
  fontViewportUnit: "vw", // 指定字体需要转换成的视窗单位，默认vw
  selectorBlackList: ["wrap"], // 指定不转换为视窗单位的类名，
  minPixelValue: 1, // 默认值1，小于或等于1px则不进行转换
  mediaQuery: true, // 是否在媒体查询的css代码中也进行转换，默认false
  replace: true, // 是否转换后直接更换属性值
  exclude: [/node_modules/], // 设置忽略文件，用正则做目录名匹配
  landscape: false // 是否处理横屏情况
}
```

```
postcss: {
  // 添加插件名称作为键，参数作为值
  plugins: {
    "postcss-px-to-viewport": {
      unitToConvert: "px", // 默认值`px`，需要转换的单位
      viewportWidth: , // 视窗的宽度,对应设计稿宽度
      viewportHeight: , // 视窗的高度，根据375设备的宽度来指定，一般是667，也可不配置
      unitPrecision: , // 指定`px`转换为视窗单位值的小数位数
      propList: ["*"], // 转化为vw的属性列表
      viewportUnit: "vw", // 指定需要转换成视窗单位
      fontViewportUnit: "vw", // 字体使用的视窗单位
      selectorBlaskList: [".ignore-"], // 指定不需要转换为视窗单位的类
      mediaQuery: false, // 允许在媒体查询中转换`px`
      minPixelValue: , // 小于或等于`1px`时不转换为视窗单位
      replace: true, // 是否直接更换属性值而不添加备用属性
      exclude: [], // 忽略某些文件夹下的文件或特定文件
      landscape: false, // 是否添加根据landscapeWidth生成的媒体查询条件 @media (orientation: landscape)
      landscapeUnit: "vw", // 横屏时使用的单位
      landscapeWidth:  // 横屏时使用的视窗宽度
    }
  },
  preset: {
    // 更改postcss-preset-env 设置
    autoprefixer: {}
  }
```

6. 静态资源

需要在nuxt.config.js head中配置的资源，需要放在static目录下，引入的时候，不需要写static

```
module.exports = {
    head: {
        script: [
            {src: '/js/a.js', type: 'text/javascript', charset: 'utf-8'}
        ]
    }
```

```
}
```

7. 使用IP访问

```
// package.json
{
    "config": {
        "nuxt": {
            "host": "0.0.0.0",
            "port": 3000
        }
    }
}

// or nuxt.config.js
server: {
    "host": "0.0.0.0",
    "port": 3000
}
```

8. 使用Vuex

使用模块化开发时,index.js里的state等访问时,不需要像其他模块一样使用$store.state.todos.list,直接使用$store.state.list

9.loading

页面跳转之间使用

自定义loading的调用时机,若将状态存储在Vuex中,在router.afterEach才会进入?

若想做全页面的覆盖loading,则可在<Nuxt /> 中进行v-if的判断,当非loading态时再展示

```
// store/index.js
export const state = () => ({
    loading: false
});
export const mutations = {
    changeLoading(state, payload) {
        state.loading = payload;
    }
}
// componentsloading.vue
<div v-if="$store.state.loading"></div>
// layouts/default.vue
<Nuxt v-if="!$store.state.loading" />
// plugins/loading.js
export default({app}) => {
    app.router.beforeEach((to, from, next) => {
        app.store.commit('changeLoading', true);
        next();
    });
    app.router.afterEach(() => {
        app.store.commit('changeLoading', false);
    });
}
// nuxt.config.js
module.exports = {
    loading: '@/components/Loading.vue',
    plugins: [
        '@/plugins/loading.js'
    ]
```

```
}
```

10. 服务器启动失败

getaddrinfo ENOTFOUND XXX,GetAddrInfoReqWrap.onlookup[as complete] (dns.js:57:26)

域名解析失败

需要绑定host,/etc/hosts

11. nuxt.config.js中引入assets下的文件后,打包后,文件找不到

不要在module.exports外引入文件,可以在内部引入,否则找不到文件

```
// nuxt.config.js
module.exports = {
    ...
    router: {
        extendRoutes(routes, resolve) {
            routes.splice(0);
            // 若这行代码在exports上面,则会报错:找不到文件
            const generateRoutes = require('./assets/js/routers.js').default;
            const route = generateRoutes(resolve);
             routes.push(...route);
        }
    }
}
```

12.部署上线

1. npm run build,生成.nuxt文件夹

2. 将.nuxt，nuxt.config.js，package.json，package-lock.json，static放在服务器上,执行npm install ,npm start

3. 执行npm install --production,将只安装dependencies,不安装运行时依赖,若nuxt.config中有用到运行时依赖,可能会报错,如nuxt-svg-sprite-loader

```
// package.sh
src=`pwd`
npm run build
mkdir -p upgrade
cd upgrade/
rm -rf *
cp $src/nuxt.config.js $src/package.json $src/package-lock.json .
cp -rf $src/static .
cp -rf $src/.nuxt .
 cd $src
 rm upgrade.zip
 zip -q(不显示执行过程) -r(递归执行) upgrade.zip upgrade
 // start.sh
 home="/data/nuxt"
 pm2 stop nuxt
 pm2 delete nuxt
 cd $home
 npm install
 pm2 start npm -- name nuxt -- run start
```

13. 页面渲染之前，想做一些处理，如获取权限等

两种方案：

1. nuxtServerInit：**只能在store/index.js中使用**

接收两个参数，第一个参数是Vuex上下文对象，第二个是Nuxt上下文对象

```
// store/index.js
export const actions = {
    async nuxtServerInit({commit}, {app}) {
        const res = await app.$axios.get('XXXX')
```

```
    }
};
```

2. middleware： 通过中间件，会在页面渲染之前执行

执行流程：

1)nuxt.config.js

2)匹配布局

3)匹配页面

```
// middleware/auth.js
export default async function({$axios}) {
    const res = await $axios.get('XXX');
}
// nuxt.config.js
module.exports = {
    ...
    router: {
        middleware: 'auth'
     }
};
// 也可以在某个页面使用
export default {
    ...
    middleware: 'auth'
}
```