# 生命周期

|  |
| --- |
| https://img2018.cnblogs.com/blog/1158910/201908/1158910-20190818204215037-2038459873.png  **红框内的是Nuxt的生命周期(运行在服务端)，**  **黄框内同时运行在服务端&&客户端上，**  **绿框内则运行在客户端**  红框、黄框内的周期都不存在Window对象  export default {   middleware(*context*){  console.log(this) // undefined       console.log(window) // 服务端报错    },    validate({*params*,*query*}){      console.log(this) // undefault      console.log(window) // 服务端报错      return true;    },    asyncData() {  console.log(this) // undefined      console.log(window) // 服务端报错    },    fetch() {  console.log(this) // 执向组件      console.log(window) // 服务端报错    },    created () {  console.log(this) // 执向组件      console.log(window) // undefined    },    mounted () {  console.log(this) // 执向组件      console.log(window) // Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, frames: Window, …}    }  } |

## nuxtServerInit

|  |
| --- |
| 适用场景是对 store 操作  store/index.js  export *const* actions = {    nuxtServerInit(*store*, *context*) {      // 通常初始化数据到store当中      console.log('store：',*store*)      console.log('context：',*context*)    }  } |

## middleware中间件

|  |
| --- |
| 中间件执行流程顺序：nuxt.config.js->匹配布局->匹配页面 |

### nuxt.config.js配置

|  |
| --- |
| nuxt.config.js  router:{    middleware:'auth'  },  middleware/auth.js  export default (*context*)*=>*{    // context 服务端上下文    // 全局守卫业务    console.log('middleware：', *context*)  }  或  export default ({*store*,*route*,*redirect*,*params*,*query*,*req*,*res*})*=>*{    // context 服务端上下文    // 全局守卫业务    console.log('middleware：', *store*)  } |

### layouts布局页面配置

|  |
| --- |
| middleware/auth.js  export default (*context*)*=>*{    console.log('布局页面层级中间件：', *context*)  }  =======================================  layouts/default.vue  <script>  export default {    // middleware:'auth', 布局页面层级中间件定义  // 或    middleware(*context*) {      console.log('布局页面层级中间件：', context)    }  }  </script> |

### 普通页面配置

|  |
| --- |
| Pages/index.vue  在当前页面实例化之前执行  // SSR  // middleware:'auth'  // 或  middleware(*context*){  console.log('middleware pages：', context)  }, |

## validate

|  |
| --- |
| 参数校验，校验失败，则自动跳转到错误页面  Pages/index.vue    //参数的有效性    validate({*params*,*query*}) {      //校验业务      console.log('validate')      return true;    }, |

## asyncData

|  |
| --- |
| data(){      return {        a:1      }    },  //读数据，返回给组件    asyncData(*context*){      //异步业务逻辑，读取服务端数据      console.log('asyncData')      return {        b:2      }    }, |

## fetch

|  |
| --- |
| //读数据 , vuex    fetch({*store*}){      //异步业务逻辑，读取服务端数据提交给vuex      console.log('fetch')    }, |

## beforeCreate/ created

|  |
| --- |
| //SSR && CSR 有机会拿到window,拿不到时undefined    beforeCreate(){      console.log('beforeCreate')    },    created(){      console.log('created',this) // this指向当前组件    }, |

## beforeMount/ mounted

|  |
| --- |
| //CSR   window this指向组件    beforeMount(){},    mounted(){      console.log('mounted')    }, |

## beforeUpdate/ updated

|  |
| --- |
| beforeUpdate(){},    updated(){      console.log('updated')    }, |

## beforeDestroy/ destroyed

|  |
| --- |
| beforeDestroy(){},    destroyed(){      console.log('destroyed')    }, |

## activated/ deactivated

|  |
| --- |
| // 服务端渲染 激活、失活 不存在    activated(){},    deactivated(){}, |

# 路由

|  |
| --- |
|  |

# 路由守卫

## 前置

### 全局前置守卫

|  |
| --- |
| 第一种：    router:{  middleware:'auth',  }  Middleware/auth.js  export default ({*store*,*route*,*redirect*,*params*,*query*,*req*,*res*})*=>*{    // context 服务端上下文    // 全局守卫前置业务    // store 状态树信息    // route 一条目标路由信息    // redirect 强制跳转    // params,query 校验参数合理性    console.log('middleware nuxt.config outside 全局守卫前置业务')    redirect('/login')  }  =========================================  第二种:      middleware({*store*,*route*,*redirect*,*params*,*query*}){      //context 服务端上下文      // 全局守卫前置业务      // store 状态树信息      // route 一条目标路由信息      // redirect 强制跳转      // params,query 校验参数合理性      console.log('middleware layouts 全局守卫前置业务')      redirect('/reg')    }, |

### 组件独享前置守卫

|  |
| --- |
| pages/index.vue    middleware({*store*,*route*,*redirect*,*params*,*query*}){      //context 服务端上下文      // 组件独享前置守卫      // store 状态树信息      // route 一条目标路由信息      // redirect 强制跳转      // params,query 校验参数合理性      console.log('middleware pages 组件独享前置守卫')      redirect('/login')    } |

### 插件全局前置守卫

|  |
| --- |
| nuxt.config.js   plugins: [      '~/plugins/router',    ],  plugins/ router.js  export default ({*app*,*redirect*,*params*,*query*,*store*}) *=>* {    // app == vue实例    // redirect 跳转函数 √  *app*.router.beforeEach((*to*,*from*,*next*) *=>* {      // 全局前置的守卫， 插件      // next() next(true) next(false)      // next('/login') 不能跳转      // console.log('插件配置，全局前置',to)      //  next()     if(*to*.name=='login' || *to*.name=='reg'){        next()      }else {        redirect({name:'login'})      }    })  *app*.router.afterEach((*to*,*from*)*=>*{      console.log('插件全局后置守卫')    })  } |

## 后置

### 组件独享后置守卫

|  |
| --- |
| pages/index.vue  beforeRouteLeave(*to*,*from*,*next*){  *let* bl=window.confirm('是否要离开');     next(bl)  } |

### 插件全局后置守卫

|  |
| --- |
| nuxt.config.js   plugins: [      '~/plugins/router',    ],  plugins/ router.js  export default ({*app*,*redirect*,*params*,*query*,*store*}) *=>* {  *app*.router.afterEach((*to*,*from*)*=>*{      console.log('插件全局后置守卫')    })  } |

# 参数检验-错误路由定制

|  |
| --- |
| <nuxt-link to="/goods/1?a=1&b=2">商品01</nuxt-link>  <nuxt-link :to="{name:'goods-id',params:{id:2},query:{a:11,b:22}}">商品02</nuxt-link>  <nuxt-link :to="{name:'goods-id',params:{id:3},query:{a:111,b:222}}">商品03</nuxt-link>    name:'goods-detail',    validate({*params*,*query*}){      return typeof params.id === 'number'    },  layouts/error.vue  <template>    <div>      <h1 v-if="error.statusCode">{{error.message}}</h1>      <h1 v-else>应用发送异常</h1>      <button @click="$router.replace('/index')">跳转到首页</button>    </div>  </template>  <script>  export default {    //接受错误信息   error: {statusCode,message}    props:['error']  }  </script> |

# 统一动效和独享动效

|  |
| --- |
| asset/css/ transition.css  /\*  路由统一动效 \*/  /\* 动画形式 \*/  .page-enter-active, .page-leave-active{  *transition*:  opacity .5s;  }  /\* 入 退 \*/  .page-enter, .page-leave-active{  *opacity*:  0;  }  /\* 单路由动画 ->组件内部 transition: '类名' \*/  .test-enter-active, .test-leave-active{  *transition*:  .5s ease all;  }   .test-enter, .test-leave-active{  *margin-left*: -1000px;  }  nuxt.config.js  /\*   \*\* 全局样式   \*/  css: [    'assets/css/transition.css'  ],  ============================================================  独享动效  export default {    name:'goods-detail',    transition:'test' //'动画名' 可以使用全局样式test 或 当前组件样式test  }  </script>  <style scoped>    .test-enter-active, .test-leave-active{  *transition*:  .5s ease all;    }     .test-enter, .test-leave-active{  *margin-left*: -1000px;    }  </style> |

# 数据交互

|  |
| --- |
| 安装：@nuxtjs/axios、@nuxtjs/proxy  nuxtjs自带的模块都要在modules配置  next.config.js  modules: [     '@nuxtjs/axios'  ],  ------------------  static\data\list.json  {  *"title"*:"nuxt数据"  }  ------------------  pages\index.vue  async asyncData({*$axios*}){  *let* res = await $axios({url:'/data/list.json'})     console.log('读取到的静态资源',res.data)     return {        title:res.data.title,     }  },  async fetch({*$axios*}){  *let* res = await $axios({url:'/data/list.json'})    console.log('res',res.data.title)  }, |

# 跨域

|  |
| --- |
| nuxt.config.js  axios:{    proxy:true,    // 开启axios跨域    prefix:'/api', // baseUrl  },  proxy: {    '/api/':{      target:'http://localhost:3001',//代理转发的地址      changeOrigin:true,      pathRewrite:{        '^/api': '' // 把api开头替换为空      }    }  },  pages\index.vue  async asyncData({*$axios*}){    //读取跨域数据  *let* res2 = await $axios({url:'/api/goods/home',params:{\_limit:1}})    console.log('读取到的跨域资源',res2.data)    return {      data:res2.data.data    }  }, |

# 拦截器配置和token携带

|  |
| --- |
| nuxt.config.js  plugins: [     {       src:'~/plugins/axios',       ssr:true//服务端     }  ],  plugins\axios.js  export default *function*({*$axios*,*redirect*,*route*,*store*}) {    //基本配置  *$axios*.defaults.timeout=10000;    //请求拦截  *$axios*.onRequest(*config=>*{      console.log('请求拦截')  *config*.headers.token='eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFsZXgiLCJfaWQiOiI1ZThhMGQ2MzczNDg2MDIzYTRmZDY4ZGYiLCJpYXQiOjE1ODY2MDIwODcsImV4cCI6MTU4NjY4ODQ4N30.bgzJALn6DiKMQuhohPjjml3cgjZbtXiU2Kfg9t1lcGo';      return *config*;    })    //响应拦截  *$axios*.onResponse(*res=>*{      // console.log('响应拦截')      if(*res*.data.err === 2 && *route*.fullPath !== '/login'){        redirect('/login?path='+*route*.fullPath)      }      return *res*    })    //错误处理  *$axios*.onError(*error=>*{      //处理      return *error*;    })  } |

# loading页配置与定制

|  |
| --- |
| nuxt.config.js  /\*    \*\* 定义系统默认loading效果，或者指定一loading组件    \*/    // loading: { color: '#399', height:'3px' },    loading: '~/components/loading.vue',  components\loading.vue  <template>    <div v-if="loading" class="spinner"></div>  </template>  <script>  export default {    data:()*=>*({      loading:false    }),    methods:{      // start finish是nuxt约定好的      start(){        this.loading = true;      },      finish(){        this.loading = false;      },    }  }  </script>  <style scoped>  .spinner {  *width*: 60px;  *height*: 60px;  *background-color*: #399;    *position*: fixed;  *left*:50%;*top*:50%;  *margin-left*:-30px;  *margin-top*:-30px;  *-webkit-animation*: rotateplane 1.2s infinite ease-in-out;  *animation*: rotateplane 1.2s infinite ease-in-out;  }    @-webkit-keyframes *rotateplane* {    0% { *-webkit-transform*: perspective(120px) }    50% { *-webkit-transform*: perspective(120px) rotateY(180deg) }    100% { *-webkit-transform*: perspective(120px) rotateY(180deg)  rotateX(180deg) }  }    @keyframes *rotateplane* {    0% {  *transform*: perspective(120px) rotateX(0deg) rotateY(0deg);  *-webkit-transform*: perspective(120px) rotateX(0deg) rotateY(0deg)    } 50% {  *transform*: perspective(120px) rotateX(-180.1deg) rotateY(0deg);  *-webkit-transform*: perspective(120px) rotateX(-180.1deg) rotateY(0deg)    } 100% {  *transform*: perspective(120px) rotateX(-180deg) rotateY(-179.9deg);  *-webkit-transform*: perspective(120px) rotateX(-180deg) rotateY(-179.9deg);    }  }  </style> |

# vuex定义和使用

## 模块方式

|  |
| --- |
| `store` 目录下的每个 `.js` 文件会被转换成为状态树（当然，`index` 是根模块）  store\index.js |

## Classic方式

|  |
| --- |
| Classic(不建议使用)：`store/index.js`返回创建Vuex.Store实例的方法 |

# 状态管理

|  |
| --- |
|  |

# 状态持久化和token校验

|  |
| --- |
| 安装: cookie-universal-nuxt  思想：登录时，同步vuex && cookie, 强制刷新后,nuxtServerInit钩子，取出cookies，同步vuex，  axios拦截器读取vuex  nuxt.config.js  modules: [    'cookie-universal-nuxt'  ],  pages\login.vue      login(){        this.$axios({          url:'/api/login',          method:'post',          data:{            username:this.username,            password:this.password          }        }).then(  *res=>*{            if(res.data.err==0){              //同步vuex && cookie              this.$cookies.set('user',res.data)              this.$store.commit('user/M\_UPDATE\_USER',res.data)              if(!this.$route.query.path || /login|reg/.test(this.$route.query.path)){                this.$router.replace('/user')              }else{                this.$router.replace(this.$route.query.path)              }            }else{              this.message=res.data.msg            }            }        )      }  store\index.js  //actions  export *const* actions = {    nuxtServerInit(*store*, {app:{*$cookies*}}) {      //初始化token东西到store当中  *let* user = $cookies.get('user') ? $cookies.get('user') : {err:2,msg:'未登录',token:''};      store.commit('user/M\_UPDATE\_USER',user)    }  }  plugins\axios.js  //响应拦截    $axios.onResponse(*res=>*{      // console.log('响应拦截')        if(res.data.err === 2 && route.fullPath !== '/login'){        redirect('/login?path='+route.fullPath)      }      return res    }) |

# UI库服务端渲染 路由观测

|  |
| --- |
| nuxt.config.js  plugins: [     {       src:"~/plugins/element-ui",       ssr:true, //不支持ssr的插件只会在客户端运行不要给true       // mode:'server'//client  // v2.4+     },  ],    /\*    \*\* 全局样式    \*/    css: [      'element-ui/lib/theme-chalk/index.css',    ],    /\*    \*\* Build configuration    \*/    build: {      transpile:[/^element-ui/], 打包为element-ui开头      /\*      \*\* You can extend webpack config here      \*/      extend (*config*, *ctx*) {      }    }  plugins\element-ui.js  import Vue from 'vue'  //整体引入全局使用  import ElementUI from 'element-ui'  Vue.use(ElementUI);  // 按需引入全局使用  import {Button} from 'element-ui'  Vue.use(Button) |

# 全局定义方法、过滤器、指令、样式

|  |
| --- |
| nuxt.config.js  plugins: [    '~/plugins/mixins'  ],  plugins\mixins.js  //全局方法  import Vue from 'vue';  *let* show = () *=>* console.log('全局方法')  *Vue*.prototype.$show=show;//服务端钩子内部不可以使用，this不会执行vue实例  //全局过滤器  import \* as filters from '../assets/script/filters';  *Object*.keys(filters).forEach(*key=>*Vue.filter(key,filters[key]));  //全局指令  import direc1 from '../assets/script/directives/direc1'  import direc2 from '../assets/script/directives/direc2'  Vue.directive('direc1',direc1)  Vue.directive('direc2',direc2)  //全局组件  import UcButton from '../components/global/uc-button';  Vue.component('uc-button',UcButton)  //混入methods  Vue.mixin({    methods:{      $seo(*title*,*content*,*payload* = []){        return {          title,          meta:[{            hid:'description',            name:'keywords',            content          }].concat(payload)        }      }    }  })  --------------------------------------------  assets\script\filters.js  export *function* fillzero(*n*){    return n < 10 ? '0' + n : '' + n;  }  export *const* date = *time* *=>* {  *let* d = new *Date*();    d.setTime(time);  *let* year = d.getFullYear();  *let* month = d.getMonth()+1;  *let* date = d.getDate();  *let* hour = d.getHours();  *let* min = d.getMinutes();  *let* sec = d.getSeconds();    return `${year}年${fillzero(month)}月${fillzero(date)}日 ${fillzero(hour)}:${fillzero(min)}:${fillzero(sec)}`  }  --------------------------------------------  assets\script\directives\direc1.js  *function* direc1(*el*,*binding*,*vnode*){    console.log('全局指令1',el,binding,vnode)  }  export default {    bind(*el*,*binding*,*vnode*){      direc1(el,binding,vnode)    }  }  --------------------------------------------  assets\script\directives\direc2.js  *function* direc2(*el*,*binding*,*vnode*){    console.log('全局指令2')  }  export default {    inserted(*el*,*binding*,*vnode*){      direc2(el,binding,vnode)    },    componentUpdated(*el*,*binding*,*vnode*){      direc2(el,binding,vnode)    }  }  --------------------------------------------  components\global\uc-button\index.vue  <template>    <button>全局按钮</button>  </template>  <script>  export default {    name:'uc-button'  }  </script>  --------------------------------------------  使用  pages\goods\comment.vue    <div v-direc1="'red'">direc1</div>    <div v-direc2>{{title}}</div>  <uc-button/>    <div>发布时间:{{data.time | date}}</div>     <div>发布时间:{{4 | fillzero}}</div>    mounted(){      this.$show()    }    head(){      return this.$seo(this.data.title,this.data.des,[{}])    } |

# meta信息注入

|  |
| --- |
| nuxt.config.js  head: {      title: process.env.npm\_package\_name || '',      meta: [        { charset: 'utf-8' },        { name: 'viewport', content: 'width=device-width, initial-scale=1' },        { hid: 'description', name: 'description', content: process.env.npm\_package\_description || '' }      ],      link: [        { rel: 'icon', type: 'image/x-icon', href: '/favicon.ico' }      ]    },  ------------------------------------------    //head选项，页面meta配置    head(){      return {        meta:[          {name:'keywords',content:this.collectionName}        ]      }    },    head(){      return this.$seo(this.data.title,this.data.des,[{}])    } |

# Sass、scss在服务端使用， 全局主题使用

|  |
| --- |
| npm i node-sass sass-loader –save  安装后直接使用  <style lang="scss" scoped>  $bg: #399;  .box{  *background*: $bg  }  .box2{  *background*: $theme-bg  }  ----------------------  npm i @nuxtjs/style-resources --save  assets\scss\global.scss  $theme-bg:#393;  nuxt.config.js  modules: [    '@nuxtjs/style-resources'  ],  styleResources:{    scss:[      './assets/scss/global.scss'    ]  },  .box2{  *background*: $theme-bg  } |

# 定义入口html模板

|  |
| --- |
| app.html  <!DOCTYPE html>  <html {{HTML\_ATTRS}}>  <head {{HEAD\_ATTRS}}>    {{HEAD}}    <!-- 自定义的内容 -->    <!-- <script src="外部资源"></script> -->  </head>  <body {{BODY\_ATTRS}}>    {{APP}}  </body>  </html> |

# 资源指向与引入

|  |
| --- |
| static目录下，打包时，不会优化，直接搬到打包目录下  <h4>内部资源指向</h4>  <!-- 相对路径找到一些需要压缩的资源 assets -->  <img src="../assets/img/btns.png" alt="">  <img src="~assets/img/btns.png" alt="">  <!-- 绝对路径找到无需压缩的资源static -->  <img src="/img/bg.jpg" alt="">  .bgimg{  *height*: 50px;  *background*: url('../assets/img/takeSbmComment.png') no-repeat  *background*: url('~assets/img/takeSbmComment.png') no-repeat  }  ----------------------------------  <h4>外部资源</h4>  nuxt.config.js  script:[    {src:'https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js'}  ],  link:[    {rel:'stylesheet',href:'https://fonts.googleapis.com/css?family=Roboto'}  ]  app.html  <script src="外部资源"></script>  pages\user.vue  head:{    script:[      {src:'https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js'}    ]  }, |

# typeScript

|  |
| --- |
| types\vue.d.ts  //重写vue类型接口  import Vue from 'vue';  import {NuxtAxiosInstance} from '@nuxtjs/axios'  import {NuxtCookies} from 'cookie-universal-nuxt'  declare *module* 'vue/types/vue' {  *interface* Vue {      $axios: NuxtAxiosInstance;      $seo:Function;      detail:{title:*string*,des:*string*};      $show:()*=>void*;      collectionName:*string*;      $cookies: NuxtCookies;      username:*string*;      password:*string*;    }  } |

# vuex加入装饰器

|  |
| --- |
| *npm install vuex-class –S*  import { State, Getter, Action, Mutation } from "vuex-class";  export default *class* Index extends *Vue* {    @State    bNav: *boolean* | *undefined*; //装饰一个实例属性bNav引用到state.bNav    @State(*state* *=>* state.home.data)    home?: *object*[];    @State("user")    data!: *object*; //外部state.user 做 组件内的data使用    @Getter    getNav!: *string*; //抓取getters的key，作为组件实例属性使用    //...mapActions('user',['A\_UPDATE\_USER']),    @Action("user/A\_UPDATE\_USER")    A\_UPDATE\_USER!: (*payload*: *object*) *=>* *void*;    //...mapMutations('user',['M\_UPDATE\_USER']),    @Mutation("user/M\_UPDATE\_USER")    M\_UPDATE\_USER!: (*payload*: *object*) *=>* *void*;    getStore() {      //编程式访问vuex      //发出actions请求给user模块      // this.$store.dispatch('user/A\_UPDATE\_USER',{err:0,msg:'登录成功',token:'假token',data:{title:"user模块的actions提交过来的数据"}})      // this.A\_UPDATE\_USER({err:0,msg:'登录成功',token:'假token',data:{title:"123"}})      //发出mutations请求给user模块      // this.$store.commit('user/M\_UPDATE\_USER',{err:0,msg:'登录成功',token:'假token',data:{title:"组件携带过去的数据"}})      this.M\_UPDATE\_USER({err:0,msg:'登录成功',token:'假token',data:{title:"456"}})    }    //计算属性定义  *get* xx(): *string* {      return this.bNav ? "真" : "假";    }  } |

# 资源配置

|  |
| --- |
|  |

# 部署

|  |
| --- |
|  |

# Nuxt for ts

|  |
| --- |
|  |

# nuxt.config.js配置

## mode

|  |
| --- |
| 设置渲染模式   * 'spa': 没有服务器端渲染（只有客户端路由导航等） * 'universal': 默认，同构应用程序（服务器端呈现+客户端路由导航等） |