资料获取WX：18040505058

VUE3新特性深入解读

vue3项目的创建

安装vue-cli脚手架构建工具

vue-cli 提供一个官方命令行工具，可用于快速搭建大型单页应用。

输入命令

cnpm install -g @vue/cli

```
FF@DESKTOP-C9RNFVR MINGW64 ~/Desktop
$ cnpm install -g @vue/cli
Downloading @vue/cli to C:\Users\FF\AppData\Roaming\npm\node_modules\@vue\cli_tm
p
Copying C:\Users\FF\AppData\Roaming\npm\node_modules\@vue\cli_tmp\_@vue_cli@4.1.
1@@vue\cli to C:\Users\FF\AppData\Roaming\npm\node_modules\@vue\cli
Installing @vue/cli's dependencies to C:\Users\FF\AppData\Roaming\npm\node_modul
es\@vue\cli/node_modules
[1/36] deepmerge@^3.2.0 installed at node_modules\_deepmerge@3.3.0@deepmerge
[2/36] commander@^2.20.0 installed at node_modules\_commander@2.20.3@commander
[3/36] @vue/cli-ui-addon-widgets@^4.1.1 installed at node_modules\_@vue_cli-ui-a
ddon-widgets@4.1.1@@vue\cli-ui-addon-widgets
[4/36] @vue/cli-ui-addon-webpack@^4.1.1 installed at node_modules\_@vue_cli-ui-a
ddon-webpack@4.1.1@@vue\cli-ui-addon-webpack
```

查看版本，要求vue-cli版本在4.5以上，可以创建vue3项目

```
λ vue -V
@vue/cli 4.5.6
```

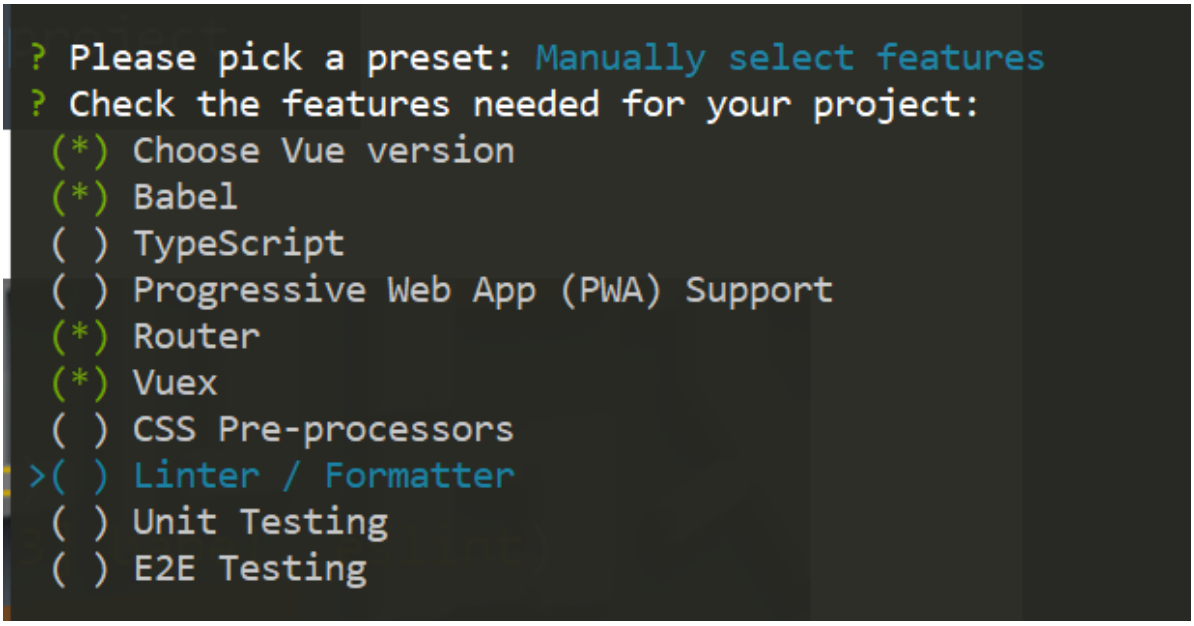
创建vue3项目

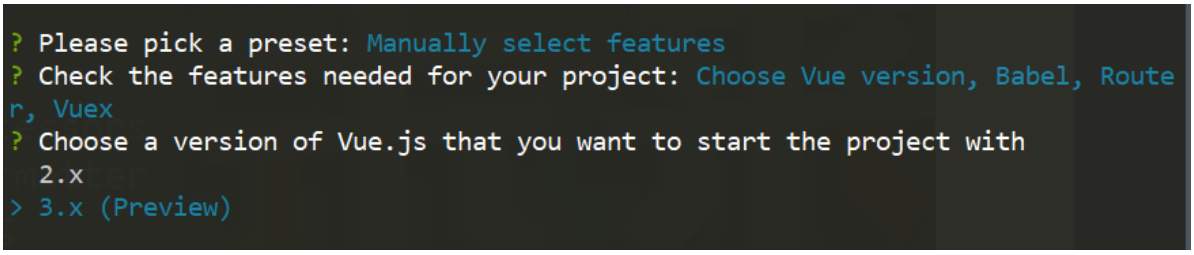
vue create 项目名称

```
λ vue create vue3-project
```

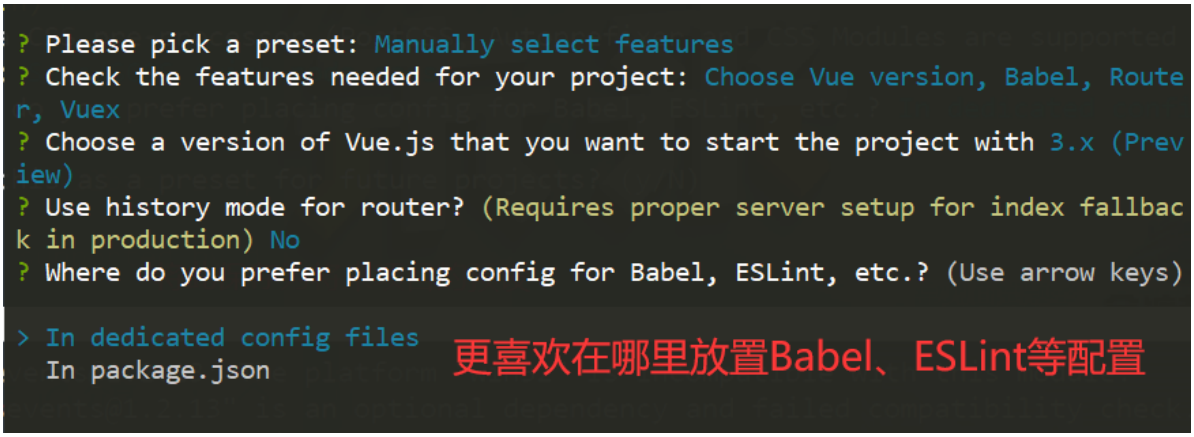
手动安装

```
Vue CLI v4.5.6
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
> Manually select features
```


? Please pick a preset: Manually select features
? Check the features needed for your project:
(*) Choose Vue version
(*) Babel
( ) TypeScript
( ) Progressive Web App (PWA) Support
(*) Router
(*) Vuex
( ) CSS Pre-processors
>( ) Linter / Formatter
( ) Unit Testing
( ) E2E Testing


? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Route
r, Vuex
? Choose a version of Vue.js that you want to start the project with
2.x
> 3.x (Preview)


? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Route
r, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x (Prev
iew)
? Use history mode for router? (Requires proper server setup for index fallbac
k in production) (Y/n)
使用路由模式


? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Route
r, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x (Prev
iew)
? Use history mode for router? (Requires proper server setup for index fallbac
k in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
In package.json
更喜欢在哪里放置Babel、ESLint等配置

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Route
r, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x (Prev
iew)
? Use history mode for router? (Requires proper server setup for index fallbac
k in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated con
fig files
? Save this as a preset for future projects? (y/N)
```

将此保存为将来项目的预设

```
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
Done in 7.66s.
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project vue3-project.
🎉  Get started with the following commands:

 $ cd vue3-project
 $ yarn serve
```

启动项目

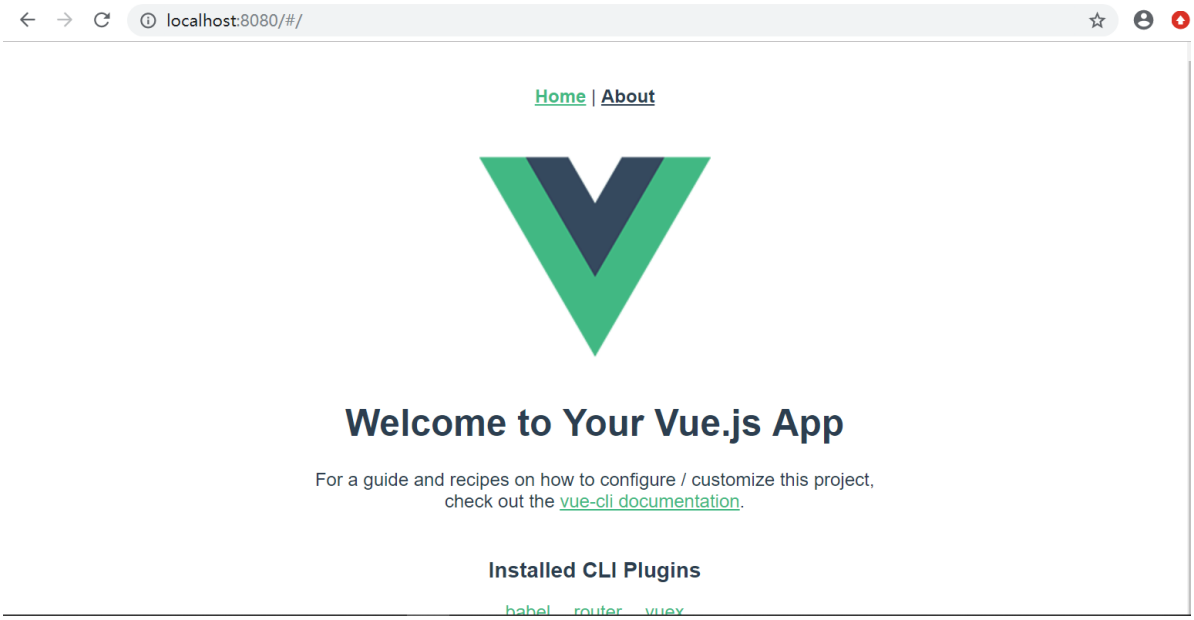
```
λ cd vue3-project
```

```
λ npm run serve
```

```
  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.124.14:8080/

  Note that the development build is not optimized.
  To create a production build, run yarn build.
```

浏览器访问



createApp

在 Vue 3 中，改变全局 Vue 行为的 API 现在被移动到了由新的 `createApp` 方法所创建的应用实例上。

vue3.0中使用createApp 来创建vue实例

```
import { createApp } from 'vue'
import App from './App.vue'
const app = createApp(App);
app.mount('#app');
```

main.js 下加载router、vuex

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

createApp(App).use(store).use(router).mount('#app')
```

setup函数

1、简介

setup函数是vue3中专门为组件提供的新属性。

2、执行时机

创建组件实例，然后初始化props，紧接着就调用setup函数，会在beforeCreate钩子之前被调用。

3、模板中使用

如果setup返回一个对象，则对象的属性将会被合并到组件模板的渲染上下文。

4、如何使用

```
<template>
  <div>
      {{name}}
  </div>
</template>

<script>
import { reactive } from "vue"
export default {
  setup(props){
    const state = reactive({
      name:'abc'
    })
    return state
  }
}
</script>
```

注意：在setup()函数中无法访问到this

reactive函数

1、简介

reactive()函数接收一个普通对象，返回一个响应式的数据对象

2、基本语法

```
按需导入reactive函数
import { reactive} from "vue"

创建响应式数据对象
const state = reactive({ id:1 })
```

3、定义响应式数据供template使用

```
1)按需导入reactive函数
import { reactive} from "vue"

2)在setup()函数中调用reactive()函数，创建响应式数据对象
setup(){
    //创建响应式数据对象
    const state = reactive({
      name:'abc'
    })
    //setup函数中将响应式数据对象return出去供template使用
    return state
}

3)在template中访问响应式数据
<template>
  <div>
```

```
      {{name}}
  </div>
</template>
```

ref的使用

1、简介

ref()函数用来根据给定的值创建一个响应式的数据对象，ref()函数调用的返回值是一个对象，这个对象上只包含一个value属性

2、基本语法

```
1)按需导入ref函数
import { ref } from "vue"

2)在setup()函数中调用ref()函数，创建响应式数据对象
setup(){
    var c = ref(10);   //初始化值为10
    return {c}
}

3)在template中访问响应式数据
<template>
  <div>
      {{c}}
  </div>
</template>
```

toRefs的使用

1、简介

toRefs()函数可以将reactive()创建出来的响应式对象，转换为普通对象，只不过这个对象上的每个属性节点，都是ref()类型的响应式数据

2、基本语法

```
1)按需导入toRefs函数
import { reactive ,toRefs } from "vue";

2)...toRefs(state)创建响应式数据对象
setup(){
    let state = reactive({id:10});
    return {
      ...toRefs(state)
    };
}

3)在template中访问响应式数据
<template>
  <div>
```

```
    <p>{{id}}</p>
  </div>
</template>
```

computed计算属性的使用

1、简介

computed()用来创建计算属性，computed()函数的返回值是一个 ref 的实例

2、基本语法

```
1)按需导入computed()
import { reactive ,toRefs ,computed} from "vue";

2)在setup()函数中调用computed()函数
setup(){
    let state = reactive({
      id:10,
      n1:computed(()=>state.id+1)  //计算属性的方式
    });
}

3)在template中访问响应式数据
<template>
  <div>
    <p>{{n1}}</p>
  </div>
</template>
```

watch的使用

1、简介

watch() 函数用来监视某些数据项的变化，从而触发某些特定的操作

2、基本语法

```
1)按需导入watch()
import { reactive ,toRefs ,watch} from "vue";

2)在setup()函数中调用watch()函数
setup() {
    let state = reactive({
      id:10,
    });
    watch(() => console.log(state.id))
  },
```

生命周期钩子函数

1、基本语法

```
1)新版的生命周期函数，可以按需导入到组件中，且只能在 setup() 函数中使用
import { onMounted, onUpdated, onUnmounted} from "vue";

2)在setup()函数中调用computed()函数
setup(){
    onMounted(() => {
      console.log('mounted!')
    })
    onUpdated(() => {
      console.log('updated!')
    })
    onUnmounted(() => {
      console.log('unmounted!')
    })
}
```

2、新旧对比

```
beforeCreate -> use setup()
created -> use setup()
beforeMount -> onBeforeMount
mounted -> onMounted
beforeUpdate -> onBeforeUpdate
updated -> onUpdated
beforeDestroy -> onBeforeUnmount
destroyed -> onUnmounted
errorCaptured -> onErrorCaptured
```

provide和inject

1、简介

provide()和 inject()可以实现嵌套组件之间的数据传递。这两个函数只能在 setup()函数中使用。父级组件中使用 provide()函数向下传递数据；子级组件中使用 inject()获取上层传递过来的数据。

2、基本语法

```
父组件：
在setup()函数中调用provide()函数
setup() {
    provide('globalColor', 'red')
  },

子组件：
在setup()函数中调用inject()函数
  setup(props){
    const state = reactive({
      color : inject("globalColor")
    })
```

```
    return state
  }
```

Vue Router 4

现在我们安装 vue-router 版本的时候，默认还是安装的 3.x 版本的，由于 vue3 的更新发生很大的变化，所以为了兼容处理，vue-router 也将发布最新版 4.x 版本了。

1、创建方式

利用createRouter 用来创建router对象

```
import { createRouter } from 'vue-router'

const router = createRouter({
  routes
})
```

2、路由模式

createWebHashHistory路由模式路径带#号

createWebHistory路由模式路径不带#号()

```
import { createRouter,createWebHashHistory } from 'vue-router'

const router = createRouter({
  history: createWebHashHistory(),
  routes
})
```

3、设置路由导航的两种方法

声明式

```
 <router-link :to="/home">
```

编程式

```
 router.push('/home')
```

声明式的常见方式

```
<router-link to="/home">home</router-link>

对象
<router-link :to="{path:'/home'}">home</router-link>
```

编程式的常见方式

```
字符串
router.push('/home')

对象
router.push({path:'/home'})
```

4、router和route的区别

router是全局路由的实例,是VueRouter的实例， 包括了路由的跳转方法，钩子函数等。

比如：$router.push

$route对象表示当前的路由信息，包含了当前 URL 解析得到的信息。包含当前的路径，参数，query对象等。

比如：$route.params

Vuex4

1、创建方式

```
创建 store 容器实例
import { createStore } from 'vuex'
const state = {
    flag:true
};
var getters = {
    getFlag(state){
        //具体的处理
    }
}
const actions={
    CHANGEFLAG({commit}){
        commit('CHANGEFLAG');
    }
};

const mutations = {
    CHANGEFLAG(state){
        //具体的处理
    }
};
const store = createStore({
    state,
    getters,
    actions,
    mutations
})
//导出store对象
export default store;
```

2、组件中的使用

```
import { useStore } from "vuex";  //导入vuex
export default {
  props:{
      name:String
  },
setup(props) {
  let store = useStore(); //定义store
  const state = reactive({});

    return {
      ...toRefs(state),
      store
    };
  },
}
```

3、vuex流程图