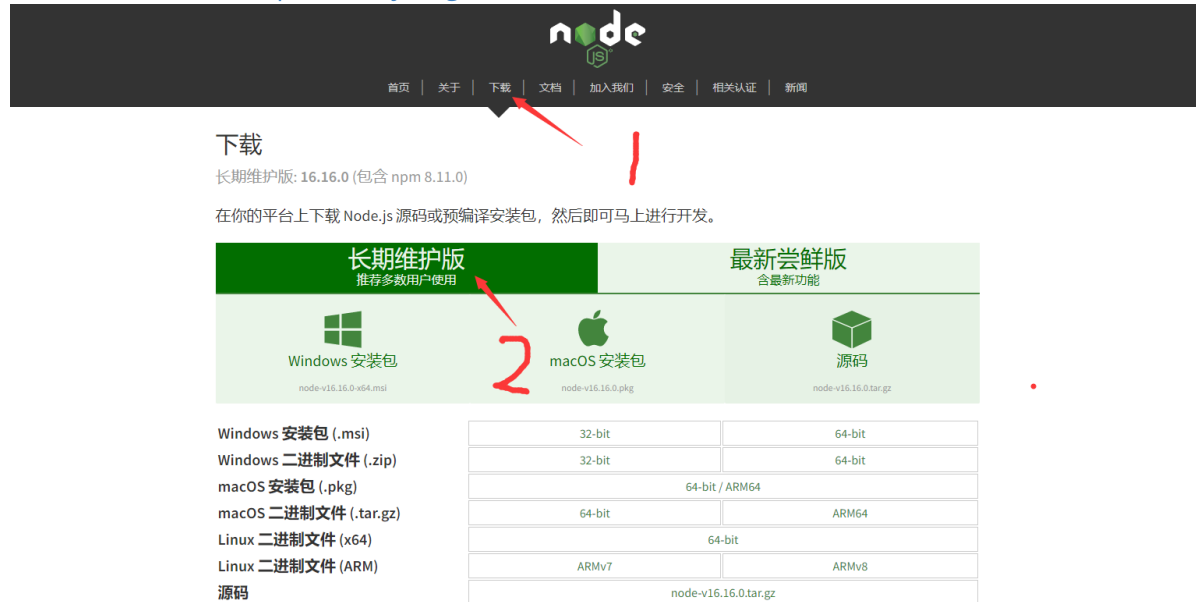


node.js基础使用

Node.js 是一个 JavaScript 运行环境。它让 JavaScript 可以开发后端程序，node不是一门语言,而是一个构建在 Chrome(谷歌浏览器) JavaScript V8 解析引擎之上的一个 JavaScript 运行时（环境）。

一.安装node

node官网下载地址: <https://nodejs.org/zh-cn/download/>



The screenshot shows the Node.js download page. A red arrow points to the '下载' (Download) link in the top navigation bar. Another red arrow points to the '长期维护版' (Long-term support) button, which is highlighted with a red '2'. Below this, there are links for Windows, macOS, and Linux installers, as well as source code. A table lists various system architectures supported by the source code.

下载

长期维护版: 16.16.0 (包含 npm 8.11.0)

在你的平台下载 Node.js 源码或预编译安装包，然后即可马上进行开发。

长期维护版
推荐多数用户使用

最新尝鲜版
含最新功能

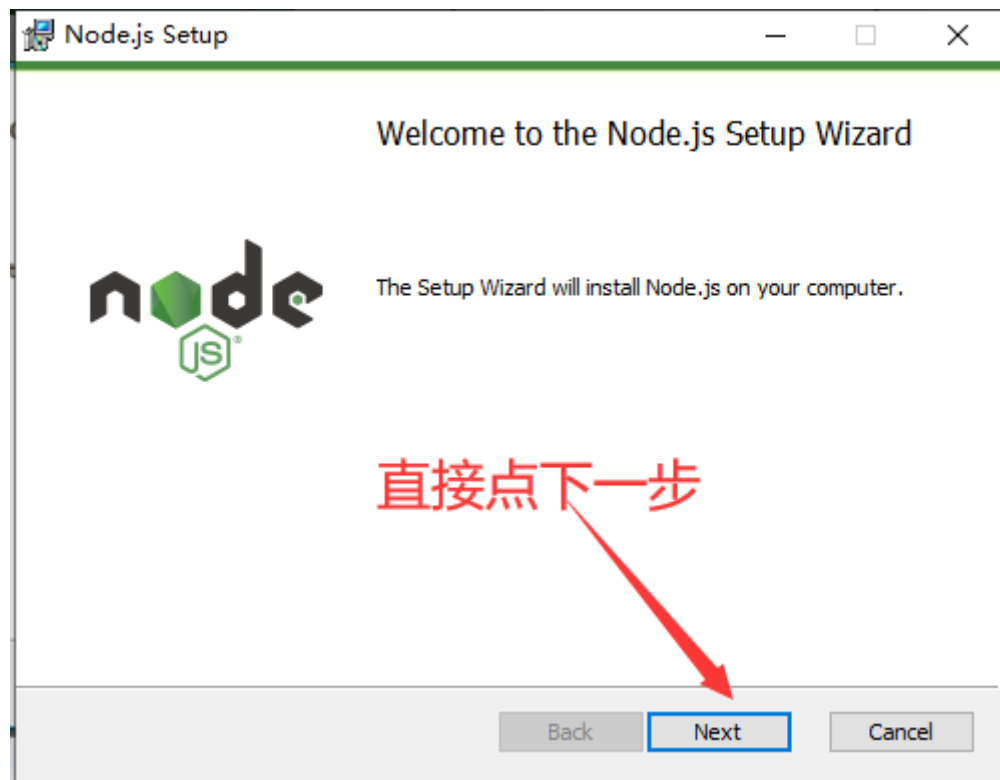
Windows 安装包
node-v16.16.0-x64.msi

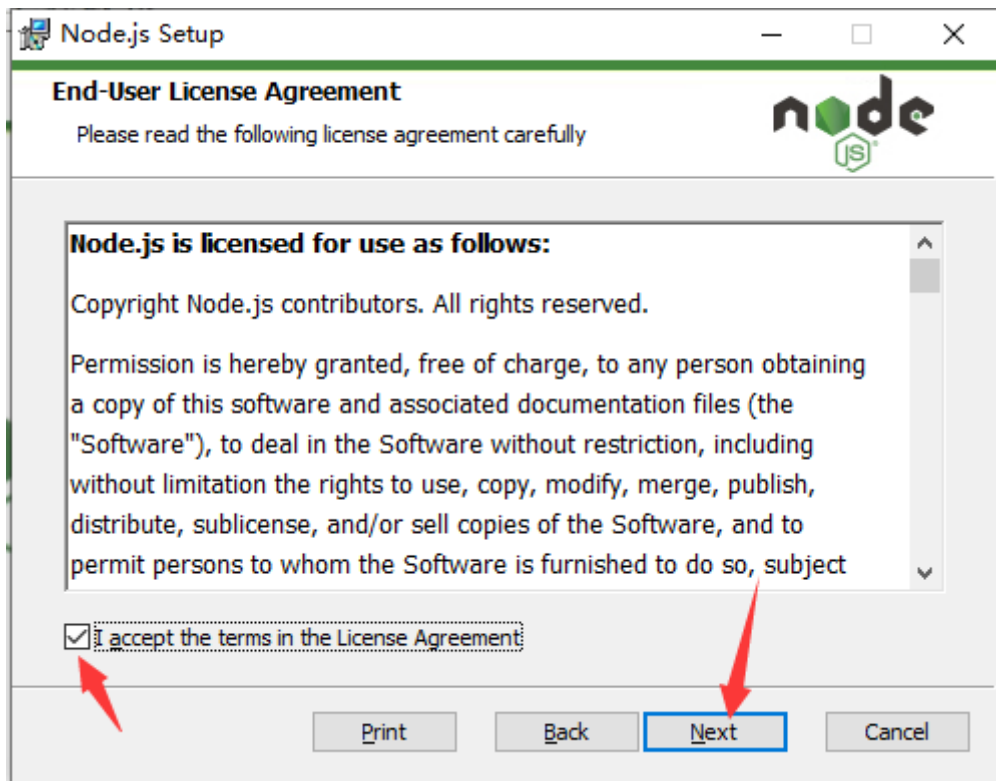
macOS 安装包
node-v16.16.0.pkg

源码
node-v16.16.0.tar.gz

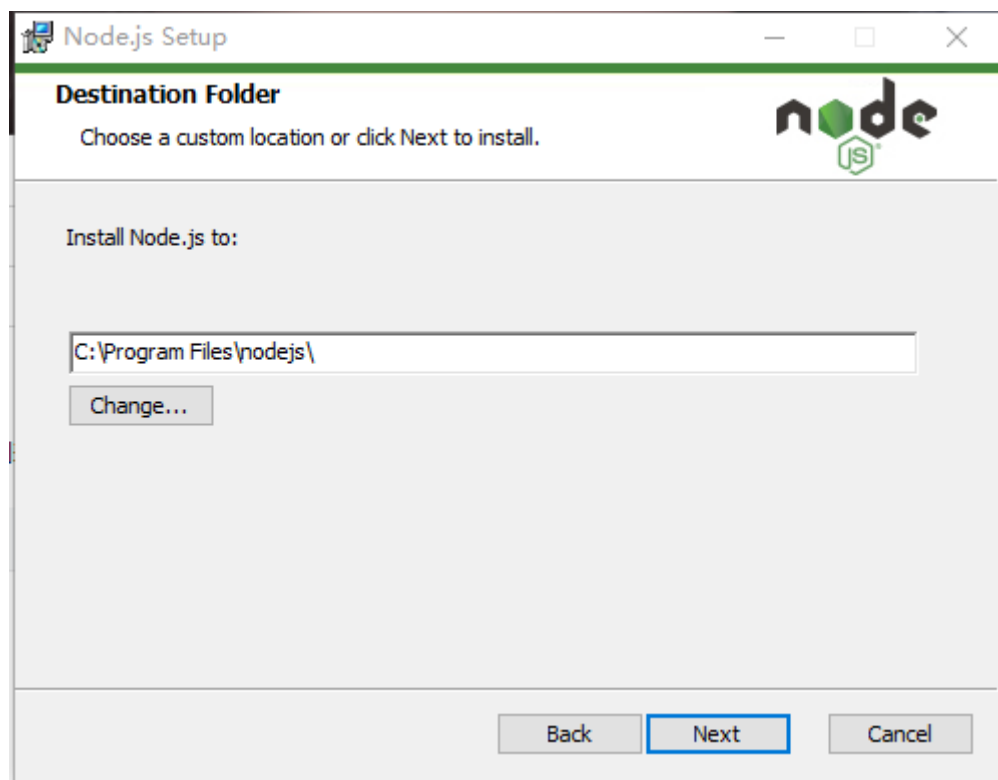
Windows 安装包 (.msi)
Windows 二进制文件 (.zip)
macOS 安装包 (.pkg)
macOS 二进制文件 (.tar.gz)
Linux 二进制文件 (x64)
Linux 二进制文件 (ARM)
源码

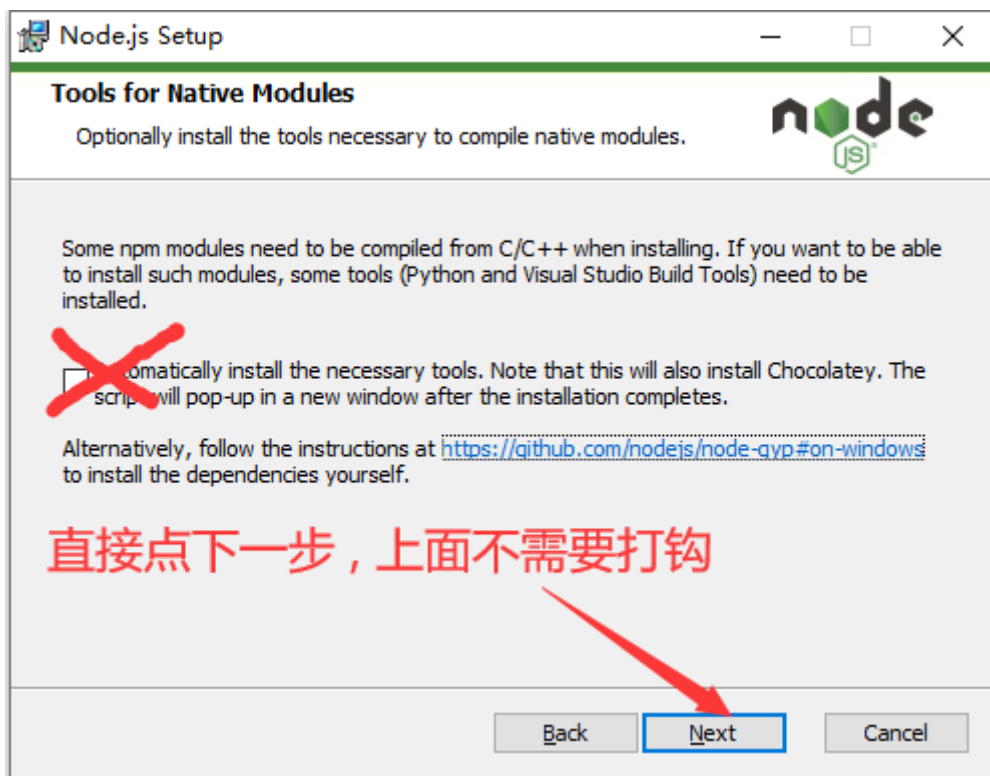
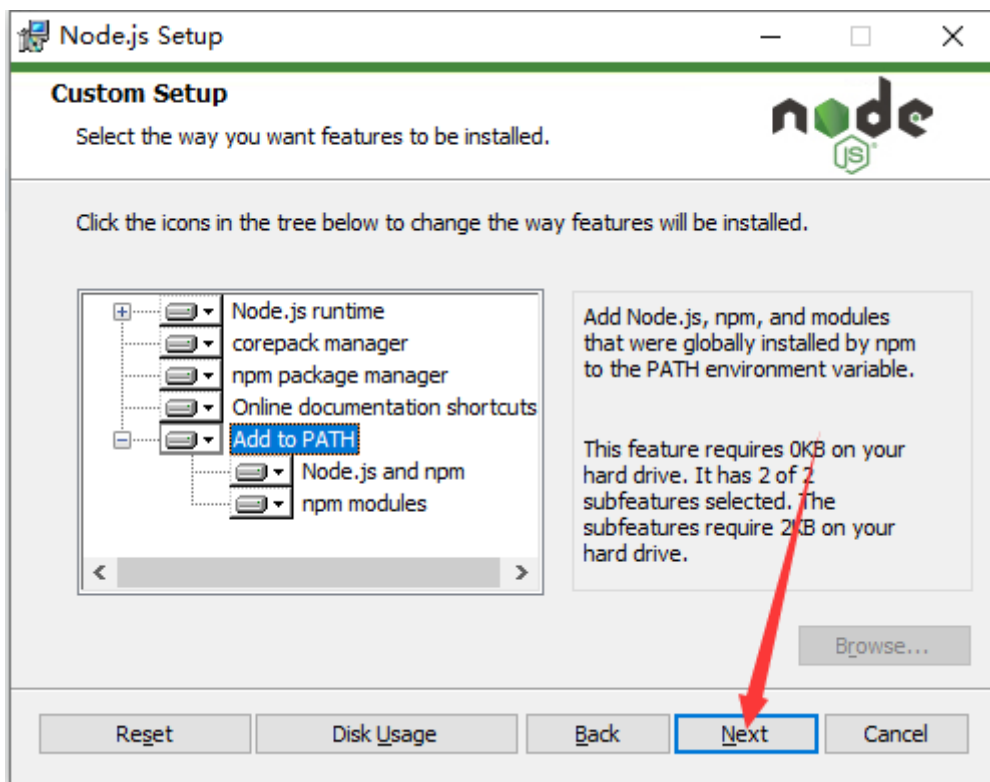
32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	
ARMv7	ARMv8
node-v16.16.0.tar.gz	

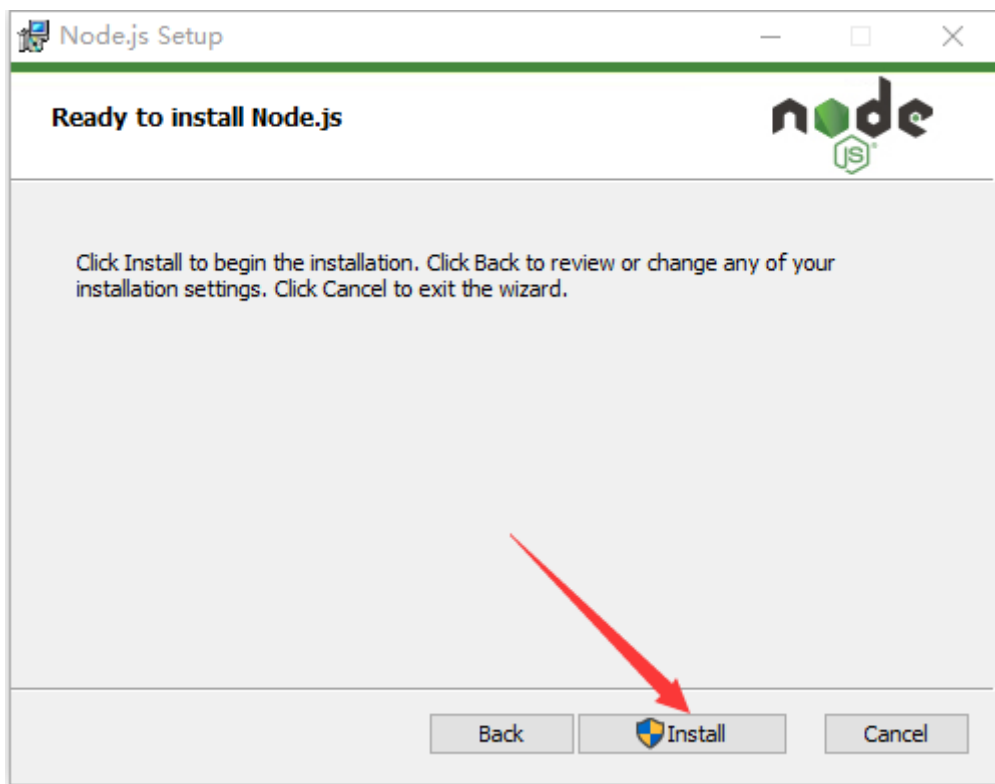




安装路径默认c盘 不要修改







注意点: 电脑上有 360 电脑管家 鲁大师等某些壁纸软件, 最好是卸载掉 不要去用 因为这些会阻止 node的运行

node基础命令

1. 打开cmd命令, 运行 `node -v` 测试是否安装完成;
2. 使用 `node 文件名` 运行一个js文件;
3. 直接 `npm init` 初始化一个node项目 之后回车就行 (快捷方式直接输入: `npm init -yes`)
4. 完成后就可以看到 文件夹里面多了 `package.json` 这个文件是记录node里面的包

二.模块化

每一个js文件都是一个模块 node应用也是由模块组成

1.自定义模块

需求: 1.js 引入 2.js的数据

1.引入require方法

```
// 在1.js中 引入2.js文件
require('./2.js')
// 返回 当前目录的绝对路径
console.log(__dirname);
// 返回当前文件的绝对路径
console.log(__filename);

// 但是如果你想再1.js中去使用name变量 那就不行
// console.log(name);

let retext=require('./2.js');
// 直接调用 2.js里面的name属性
console.log(retext.name);
```

2.输出module.exports对象

```
//2.js里的代码
let name='帅哥';
// console.log(name);

// 被引入的模块数据
module.exports={ // 相当于 我把name这个变量 暴露出去 给引入这个文件的人使用
  name
}

//直接打印module.exports
// console.log(module.exports); //拿到空对象 {}
```

3.解构赋值的写法

```
let name='小恐龙';
let obj={
  name:'血小板',
  age:18
}
module.exports={
  name,
  obj
}
```

```
let {name,obj}=require('./2.js')
console.log(name,obj);
```

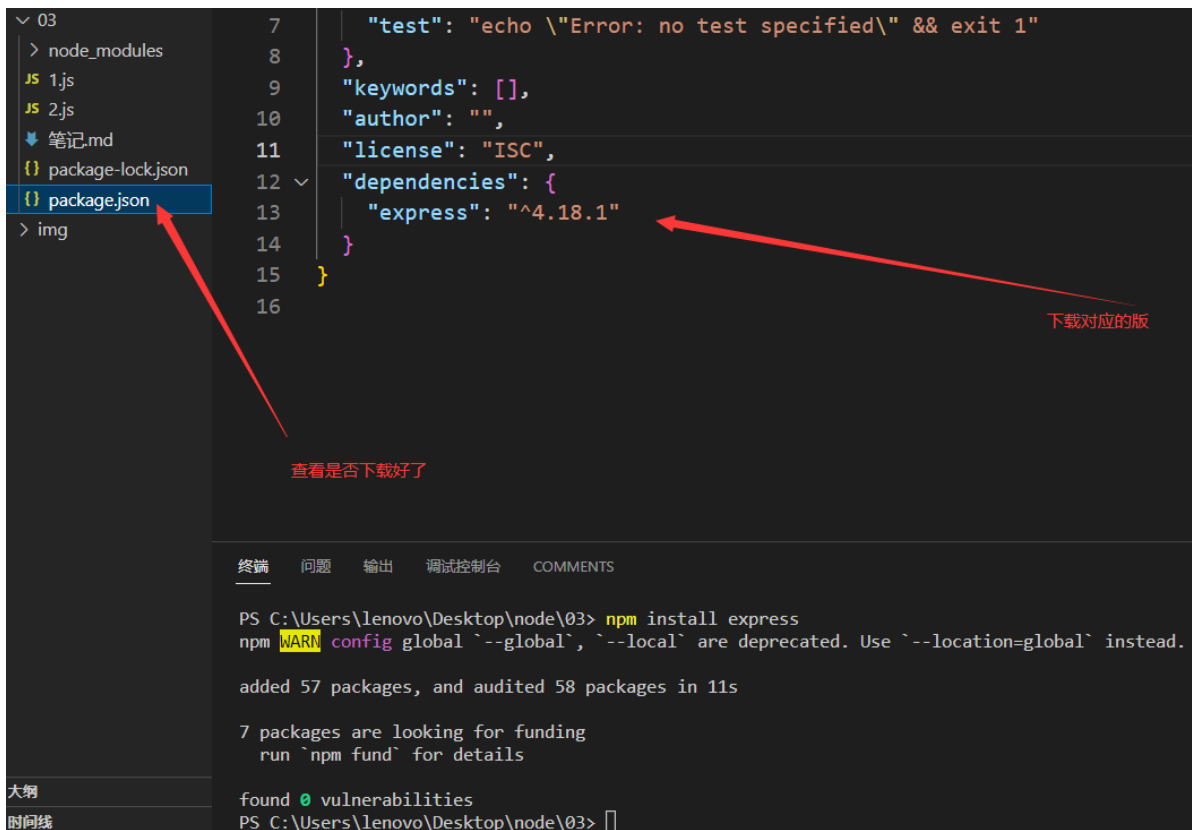
4.总结require

- require 函数 每个模块都有
- 引入时要注意文件夹命名规范 不要出现 空格 . 中文等字符 最好见名知意
- 参数 传你被引用的模块的路径(不能省略 ./)
- require('./2.js') 可以省略写法 require('./2') 这其实是防错机制 但是 路径具体到文件名 最好以免出现问题

2.引入第三方模块

1. 初始化一下项目 `npm init -yes`
2. 下载一个第三方的包 比如说 `npm install express`

```
PS C:\Users\lenovo\Desktop\node\03> npm install express
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
[#####.....] - reify:express: sill audit bulk request {
```



vue的介绍

vue官网: <https://cn.vuejs.org/guide/introduction.html>

vue是一套构建用户界面的渐进式框架，Vue 只关注视图层，采用自底向上增量开发的设计。Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件

使用列子:

```
<!DOCTYPE html>
<html lang="zh-cn">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>vue基本运用</title>
  <!-- 第一步 导入vue.js文件 -->
  <script src="https://cdn.bootcss.com/vue/2.2.2/vue.min.js"></script>
</head>

<body>
  <!-- 第三步 给网页内容 插值 -->
  <div id="app">{{ message }}</div>

  <script>
    //第二步创建 vue实例化对象
    let app = new Vue({
      el: '#app',
      data: {
        message: '帅气的封夕你好!'
      }
    })
  </script>
```

```
</body>
```

```
</html>
```

4.vue常用指令

v-html指令(相当于原生js中的innerHTML)

```
<div class="app">
  {{message}}
  <!-- v-html 绑定值 -->
  <p v-html="age"></p>

  <!--绑定一个标签对象 是可以被解析出来-->
  <div v-html='text'></div>

</div>
</div>

<script>
  // 创建实例
  let app = new Vue({
    el: '.app',
    data(){
      return{
        message: '帅气封夕,在线解答',
        age: '男',
        text: '<span>新加入的span标签</span>'
      }
    }
  })
  app.mount('.app')
</script>
```

v-text指令 (相当于原生js中的innerText)

```
<div class="app">
  <!-- v-text 绑定文本 -->
  <p v-text="message"></p>

  <!--绑定一个标签对象 直接被当前文本打印-->
  <p v-text="text"></p>
</div>
<script>
  // 创建实例
  let app = new Vue({
    el: '.app',
    data(){
      return{
        message: '帅气封夕,在线解答',
        age: '男',
        text: '<span>新加入的span标签</span>'
      }
    }
  });
</script>
```

v-model 双向绑定

```
<div id="app">
  <!-- v-model 双向绑定 -->
  <input type="text" v-model="username" placeholder="输入..." />
  <p>{{username}}</p>
</div>

<script>
  let app = new Vue({
    el: '#app',
    data() {
      return {
        message: '我是封夕',
        username: ''
      }
    }
  });
</script>
```

v-on 事件绑定

```
<div id="app">
  <!-- v-on 事件绑定 -->
  <button v-on:click="fn">点击事件</button>

  <!-- v-on简写 -->
  <button @click="fn">我是简写方式</button>

  <!-- 双击 -->
  <button @dblclick="fn1">双击</button>
</div>

<script>
  // 创建实例
  let app = new Vue({
    el: '#app',
    data() {
      return {
        fn: function() {
          alert('封夕老师yyds')
        },
        fn1: function() {
          console.log('封夕老师真帅');
        }
      }
    }
  })
</script>
```

v-for 循环

语法: v-for="(item,index) in array"

item: 数组中的每一项

index : 索引值

如果只需要第一个参数item ,index可以不写,括号可以省略

```
<div id="app">
  <!-- v-for 循环 -->
  <ul>
    <li v-for="(item,index) in array">{{item}} -- {{index}}</li>
  </ul>
</div>
<script>
  // 创建实例
  let app = new Vue({
    el: '#app',
    data(){
      return{
        //定义个数组
        array: ['啤酒', '可乐', '瓜子', '白菜']
      }
    }
  })
</script>

<!--
  遍历对象语法: v-for="(value,key,index) in obj"
  value: 属性值
  key: 属性
  index : 索引值
-->
<div id="app">
  <!-- v-for 循环 -->
  <ul>
    <li v-for="(item) in obj">{{item}}</li>
  </ul>
</div>
<script>
  // 创建实例
  let app = new Vue({
    el: '#app',
    data(){
      return{
        //定义个对象
        obj:{
          name: '封夕',
          age: 24,
          like: '打游戏'
        }
      }
    }
  })
</script>
```

v-show 显示和隐藏

```
<div id="app">
```

```

<!-- v-show 显示和隐藏 获取的是一个表达式的值 值为真的图片显示 假那就隐藏 -->
<button @click="taggle">{{btext}}</button>
<br>


<script>
  // 创建实例
  let app = new Vue({
    el: '#app',
    data(){
      return{
        btext: '隐藏',
        show: true, //初始值默认让图片显示
        taggle: ()=>{
          // console.log(this.show);
          // 取反操作 比如原来是true 点击之后取反变false
          this.show = !this.show;

          //三目判断一下 然后文字是隐藏 那就改为显示 否则的就 隐藏
          this.btext = this.btext == '隐藏' ? '显示' : '隐藏'
        }
      }
    }
  })
</script>
</div>

```

v-if 显示与隐藏 和v-show对比的区别 就是是否删除dom节点 默认值为false

```

<div id="app">
  <!-- v-if 判断表达式 条件成立那就存在dom树节点中,条件不成立 那就会在dom树中移除 -->
  <h2 v-if="age>=20">
    封夕哥太帅了
  </h2>

  <script>
    let app = new Vue({
      el: '#app',
      data(){
        return{
          age: 28
        }
      }
    });
  </script>
</div>

```

