# Vue组件

软通大学
ISOFTSTONE UNIVERSITY

# 本节目标

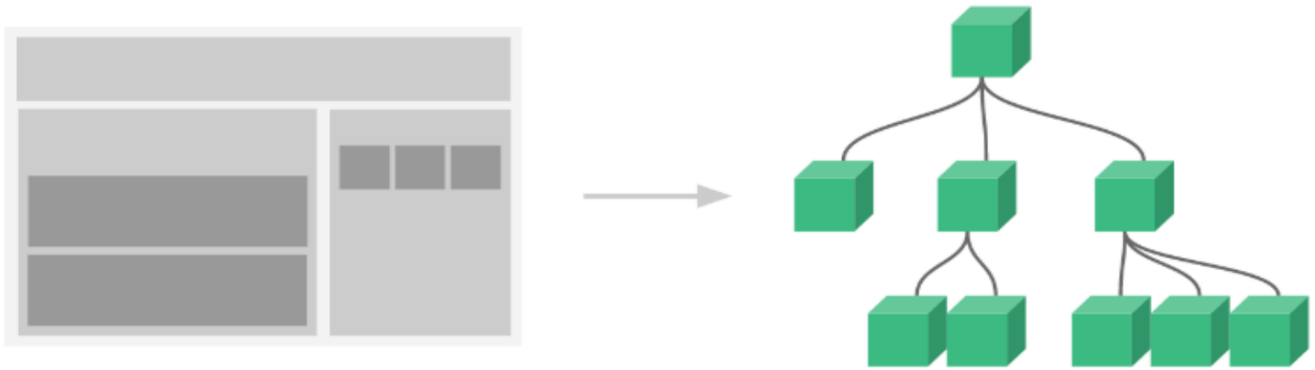
- 培养vue组件思维
- vue组件基本使用
- vue组件切换
- vue组件传值
- 熟悉插槽

# 目录 CONTENTS

# 01 Vue组件基础

# Vue组件基础

组件（Component）是 Vue.js 最强大的功能之一。
组件可以扩展 HTML 元素，封装可重用的代码。
组件系统让我们可以用独立可复用的小组件来构建大型应用，几乎任意类型的应用的界面都可以抽象为一个组件树



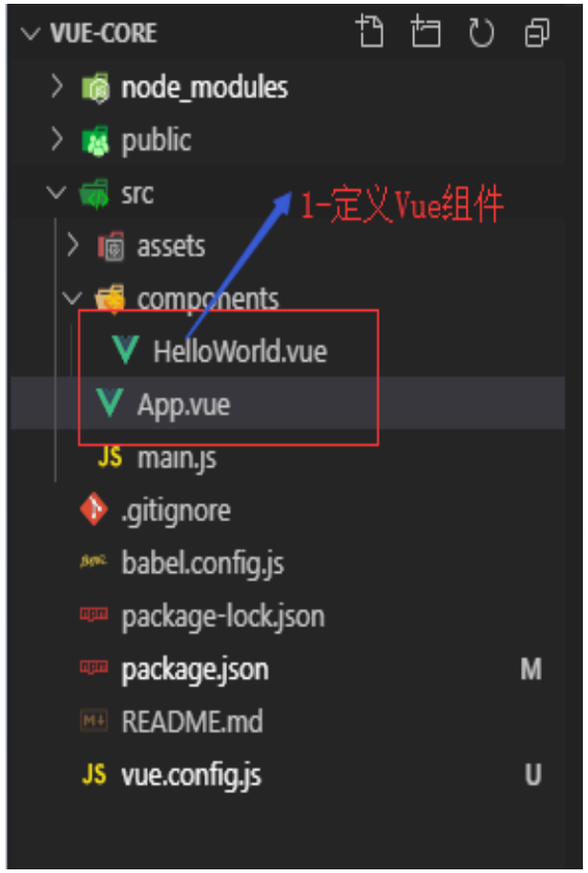
**使用组件的基本方式是:**

1. **创建组件**
2. **使用组件的<scripts>内import导入组件，并注册组件**
3. **使用组件名作为标签元素引用组件**

# Vue组件基础

**默认VueCli3.x创建的vue项目就是组件化的单页面应用.**

新建一个Vue-Core项目，修改基础项目配置，执行npm run dev...分析项目运行流程




一个vue文件就是一个vue组件:vscode添加vueHelper插件快速创建vue

```vue
App.vue
src > App.vue > {}"App.vue" > template > div#app > img
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <HelloWorld></HelloWorld>
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'app',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

3-使用组件

导入组件

2-注册helloWorld
App.vue ：父组件
HelloWorld: 子组件

组件须知:

1:组件里的template部分只能包含一个根元素

2:组件可以复用，组件里还能再包含组件，只要导入组件并注册组件，就可以使用组件

3:A组件包含B组件，A是父组件，B是子组件。A组件又被包含在Vue实例中，Vue实例相当于是A的父组件



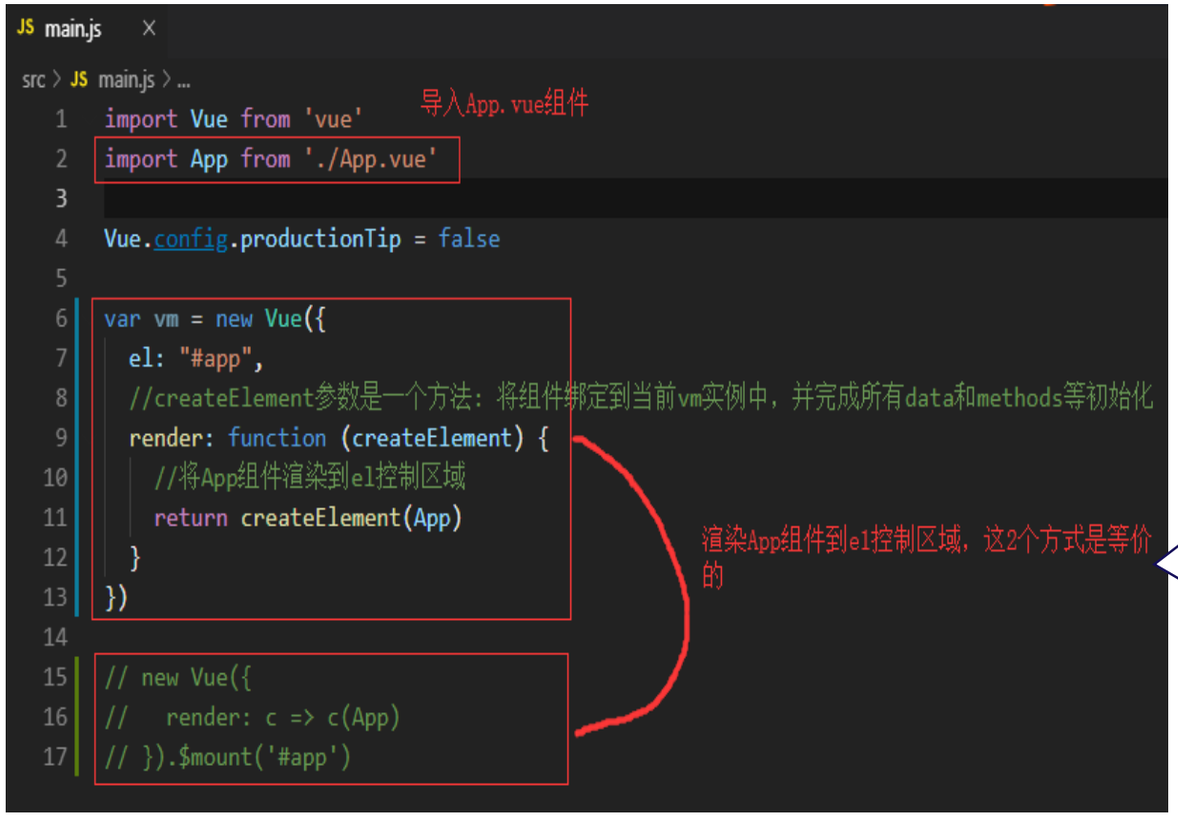


软通大学 ISOFTSTONE UNIVERSITY

# Vue组件基础

**Render方法渲染组件：**

入口main.js中，默认使用render方法将App.vue组件渲染到模板html界面中




**render方法渲染组件，实际上是 = 组件注册+引用组件**，但通常render方法只是在入口组件使用，而且它是属于vue实例的方法，如果是组件中使用组件，是没有render方法的，因为入口main.js会自动会插入到index.html模板文件中，所以render方法内部实际做的动作实际上就是注册组件，并在模板文件的app区域引用了它

# 02 组件中的data和methods

# 组件中的data和methods

组件中的data属性必须是一个函数，因为这样可以确保组件中data绑定的数据只属于组件内部
组件中的methods属性定义和Vm实例methods属性定义一样

```
<script>
export default {
  data(){
    return {
      count:0
    }
  },
  //组件中的methos定义
  methods:{
     reset(){
        this.count = 0
     }
  }
}
...
```

**案例**: 定义Counter计算器组件，在App.vue中重复使用该组件，验证data属性的函数写法结论?

# 03 动态组件

# 动态组件

所谓动态组件就是同一位置的多个组件切换显示:

```
<template>
 <div>
    <button @click="comName='Login'">登录</button>
    <button @click="comName='Register'">注册</button>
    <!-- component组件绑定is属性：指定加载的组件名称 -->
    <component v-bind:is="comName"></component>
 </div>
</template>
```

*: 点击登录或者注册的时候，改变data绑定的comName属性值，就可以实现切换组件

```
<script>
//导入组件
import Login from "./components/Login"
import Register from "./components/Register"

export default {
  data(){
    return {
      comName:"Login"
    }
  },
  components:{
    Login,
    Register
  }
}
```

# 动态组件切换动画

```
<template>
  <div>
    <button @click="comName='Login'">登录</button>
    <button @click="comName='Register'">注册</button>
    <!-- 为组件切换设置动画 -->
    <!-- mode="out-in": 设置过渡模式，out-in:表示入场动画完全退出之后，离场动画再开始进入-->
    <transition mode="out-in">
      <component v-bind:is="comName"></component>
    </transition>
  </div>
</template>
```

(transition包裹动画元素)

```
<style>
      .v-enter { opacity: 0; transform:
        translateX(150px);}
      .v-enter-to { opacity: 1; transform:
        translateX(0px);}
      .v-enter-active { transition: all 0.8s ease; }
      .v-leave {opacity: 1; transform: translateX(0px);}
      .v-leave-to {opacity: 0; transform:
        translateX(150px);}
      .v-leave-active {transition: all 0.8s ease;}
</style>
```

(6个css)

*: v-move和v-leave-active绝对定位解决组件切换的错位问题

```
.v-move {transition: all 0.8s ease;}
.v-leave-active { position: absolute;}
```

# 04 组件传值

# 组件传值

如上述的例子中，Toggle中有Login和Register, Toggle是父组件，Login和Register是子组件，App中有HelloWorld，App是父组件，HelloWorld是子组件。父组件与子组件需要数据交互，也就是存在数据传递的情况

- 父组件向子组件传值：Toggle中传递msg到Login和Register
  父组件通过v-bind:属性名='属性值'的方式发送数据到子组件, 子组件通过props属性，接受来自父组件的数据

- 子组件向父组件传值
  vue中，子组件向父组件传值是通过事件绑定实现的，并不能够向父给子传递那样，通过v-bind和props来完成，父组件在引用子组件处，通过v-on:事件名称="xx"绑定一个方法，比如: v-on:func="m"，2. 子组件内，当需要与父组件通信时，通过触发某一事件的函数，比如v-on:clic，在这个函数处理内调用: this.$emit('func','参数') ==>触发父组件的func事件绑定的方法，也就是m。
  this.$emit('func','xx'): 这里的func就是父组件v-on绑定的事件名称
  this.$emit('func','xx')：这里的xx就是子组件要向父组件传递的数据，作为事件方法的参数传递，父组件中的m方法定义参数就可以接收数据了

# 父组件向子组件传值

```
<template>
 <div>
  <button @click="comName='Login'">登录</button>
  <button @click="comName='Register'">注册</button>
  <transition mode="out-in">
    <!-- 父组件通过v-bind:属性名='属性值'的方式发送数据到子组件 -->
    <component v-bind:is="comName"
    v-bind:msg="toSonMsg"></component>
  </transition>
 </div>
</template>
```

(父组件)

```
<template>
 <div>
   <h3>这是登录组件: {{msg}}:{{title}}</h3>
 </div>
</template>

<script>
export default {
  //通过prop属性，接受来自父组件的数据
  data(){
   return {
     title:"Login"
   }
  },
  props:["msg"]
}
<script>
```

(子组件)

# 子组件向父组件传值

```
<div>
  <button @click="comName='Login'">登录</button>
  <button @click="comName='Register'">注册</button>
  <transition mode="out-in">
   <!-- 父组件通过v-on：事件名称='方法名'的方式发送事件到子组件 -->
   <component v-bind:is="comName“
v-bind:msg="toSonMsg"
v-on:func="fromJson"></component>
  </transition>
  <p>From子组件的消息:{{formJsonData}}</p>
 </div>
```

1-父组件v-on发送事件到子组件

```
methods:{
  //事件方法：由子组件触发
  fromJson(data){
   //data是来自子组件触发方法传递过来的数据参数
   this.formJsonData = data;
  }
 }
```

3-父组件执行方法，接收子组件数据

```
<div>
   <h3>这是登录组件: {{msg}}:{{title}}</h3>
   <button @click="send">Say a Word to Father</button>
 </div>

 ...
data(){
   return {
    title:"Login",
    sonmsg:{name:'我是Login',say:"孝敬给您100W"}
   }
  },
  methods:{
   send(){
    <!—子组件触发父组件方法fromJson-->
    this.$emit("func",this.sonmsg)
   }
  }
```

2-子组件触发父组件方法，并传递参数

# 05 插槽

# 插槽基本用法和插槽的作用

```
<template>
  <div>
    <!-- 插槽的基本使用 -->
    <Hello>你好</Hello>
  </div>
</template>
```

*:默认组件使用:组件标签元素包含的内容是没有任何作用的

```
<template>
  <div>
    <!-- slot插槽:分发引用组件时的元素内容 -->
    <h3>Hello Vue: <slot></slot></h3>
  </div>
</template>
```


(项目运行后)

插槽的作用:<slot>标签的位置被引用组件的标签元素内包含的内容替换了





软通大学 ISOFTSTONE UNIVERSITY

# 具名插槽

所谓具名插槽，就是给插槽定义个名字，让插槽根据名字去替换内容

```
<div>
    <!-- 具名插槽:在内容上绑定slot="名字" -->
    <Hello>
        <div slot="girl">
            受欢迎的女人:漂亮、美丽、购物、逛街
        </div>
        <div slot="boy">
            受欢迎的男人:有钱、很有钱、非常有钱
        </div>
        <!-- 默认插槽 -->
        <div>
            这里是屌丝聚集的地方
        </div>
    </Hello>
</div>
```

```
<template>
  <div>
    <!-- slot插槽:分发引用组件时的元素内容 -->
    <h3>男人和女人</h3>
    <!-- 替换name是gril的组件元素内容 -->
    <slot name="girl"></slot>
    <div style="height:1px;background-color:red;"></div>
    <!-- 替换name是boy的组件元素内容 -->
    <slot name="boy"></slot>
    <div style="height:1px;background-color:red;"></div>
    <!-- 替换没有name的组件元素内容 -->
    <slot></slot>
  </div>
</template>
```

*: 插槽就是将组件元素内容替换到组件内,具名插槽就是对应名字替换

# 作用域插槽

作用域插槽其实际意义就相当于组件传值功能，使用作用域插槽，可以将组件元素和组件模板两者之间互相传递数据。

- 组件元素--->组件模板:
  组件元素标签v-bind:属性绑定数据到slot插槽所在的组件模板，组件模板内通过props获取数据

- 组件模板--->组件元素
  组件模板<slot>标签v-bind:绑定数据，组件元素内Slot替换标签元素定义slot-scope="a"接收数据,a是任意的，a只是一个接收数据的变量名。
  特别需要注意的是：
  slot-scope="a" 表示接收来自slot绑定的数据，a只是一个接收数据的变量名，
  类似于a = slot绑定的数据，而组件模板内sloat绑定的数据格式 :cs="item"，
  其实质是: {cs:{id:1,name:'李　白'}}，cs也只是一个变量名。所以，如果要在组件元素内显示item对象的内容应该是这么写: `<div slot-scope="a">{{a.cs.id}}---{{a.cs.name}}</div>`

# 作用域插槽

```
<div>
  <!-- 1:组件元素绑定数据到组件模板内 -->
  <Hello2 :cikeList="cikes">
    <!-- 定义slot-scope="cikes"接收来自slot标签绑定的数据 -->
    <!-- a表示接收slot :cs的值，名字可以任意 -->
    <!-- 注意: Hello.vue中 slot: cs="item" -->
    <!-- a接收的数据，它的数据格式是key:value形式的对象:
{cs:{id:1,name:"李白"}}，而不是直接item对象-->
    <!-- <div slot-scope="a">{{a}}</div> -->
    <div slot-scope="a">{{a.cs.id}}---{{a.cs.name}}</div>
  </Hello2>
 </div>
```

1-组件元素绑定数据到组件模板内

4-组件元素接收组件模板绑定的数据

2-组件模板内接收组件元素绑定的数据

```
<template>
 <div>
  <ul>
   <!--3:slot通过v-bind绑定了item数据发送到了组件
元素上，组件元素通过slot-scope接收 -->
    <li v-for="item in cikeList" :key="item.id">
      3- 组件模板内绑定数据到组件元素去
      <slot :cs="item"></slot>
    </li>
  </ul>
 </div>
</template>
<script>
export default {
 //2:-接收来自组件元素绑定的数据
 props: ["cikeList"]
};
</script>
```

# 本节总结

- 组件的创建、导入与引用
- 组件的data必须是一个函数
- 组件切换以及组件切换时的动画
- 组件传值
- 插槽的用法

# Vue组件综合案例

评论列表

软通大学
ISOFTSTONE UNIVERSITY

# 本节目标

- 掌握Vue组件的基本用法
- 掌握Vue组件传值
- 熟悉插槽的使用

# 01 组件传值实现评论列表

# 组件传值实现评论列表

评论人：

评论内容：

发表评论

评论人：张三丰 我的武功天下第一

评论人：小小 我是一致小小鸟

安装bootstrap作为前端界面效果展示

# 插槽方式实现评论列表

技术分析:

**CommentBox.vue子组件点击发表评论**

评论数据信息保存到本地localStorage中，并通过this.$emit(“func”) 通知父组件，加载localStorage的评论数据

**App.vue入口组件展示评论：**

1：在生命周期方法created钩子函数内加载localStorage的评论数据

2：v-on:func绑定事件给子组件

3：methods属性中定义事件方法，处理子组件向父组件传递数据时的回调通知

# 02 插槽方式实现评论列表

# 插槽方式实现评论列表

技术分析:

**App.vue入口组件展示评论:**

1：生命周期created钩子函数内加载localstorage数据展示评论

2：在引用CommentBox评论子组件时，通过v-bind:list 绑定评论数据到CommentBox子组件内部

3：作用域插槽slot-scope=“a”，接收CommentBox的属性绑定传递给组件元素的item数据

**CommentBox.vue评论组件:**

1:export default{ ..props:[“list”] } 定义props属性接收来自组件元素传递的数据

2:点击发表评论时候，直接给list重新赋值最新的数据即可。

3: li元素显示的内容通过slot插槽分发，并通过:cs=“item“ 属性绑定的形式传递给组件元素

```
<div>
  <!-- slot插槽 -->
  <CommentBox :list="list">
    <div slot-scope="a">
      <span class="badge">评论人：  {{ a.cs.user }}</span> {{ a.cs.content }}
    </div>
  </CommentBox>
</div>
```

(组件元素)

```
...
 <ul class="list-group">
     <li class="list-group-item" v-for="item in
list" :key="item.id">
        <!-- 显示li元素的内容分发给组件元素 -->
        <slot :cs="item"></slot>
     </li>
   </ul>
```

(组件模板)

# 03 refs标记属性

# 插槽方式实现评论列表

ref标记属性是vue提供给开发者快速获取组件实例，也能快速获取dom元素(获取到的是一个原生js对象)的技术

```
<template>
  <div>
    <h3 id="myh3" ref="myh3">哈哈哈， 今天天气太好了！！！ </h3>
    <hr />
    <RefsDemo ref="demo"></RefsDemo>
  </div>
</template>
```

- **获取dom**

  **this.$refs.myh3.innerText**

- **获取组件**

  a) 获取组件:调用组件methods属性中的方法: **this.$refs.demo.show();**

  b) 获取组件:获取组件data属性中的数据： **this.$refs.demo.msg**

# 本节总结

- Vue组件传值应用
- 插槽的应用
- refs标记属性

# 本节练习

- 课堂案例: 分别用组件传值方式和插槽方式实现评论列表?


软通大学
ISOFTSTONE UNIVERSITY