

# Vue3 API

蓝桥杯Web组 省赛备赛

# 浏览器API

- CSS选择器
- 文档结构与遍历
- 属性
- 元素内容
- 操作节点
- 操作样式

# Vue3

- 模板语法
- 响应式API
- 生命周期API
- 组件API

# Vue3学习最佳资源

官方文档: <https://vuejs.org>

# 模板语法

Vue 使用基于 HTML 的模板语法，允许您声明性地将渲染的 DOM 绑定到底层组件实例的数据。所有 Vue 模板都是语法有效的 HTML，可以被符合规范的浏览器和 HTML 解析器解析。

## 1. 数据绑定

```
<template>
  <span>Message: {{ msg.toUpperCase() }}</span>
</template>
```

## 2. 属性绑定

```
<script setup>
const dynamicId = ref('foo')
</script>

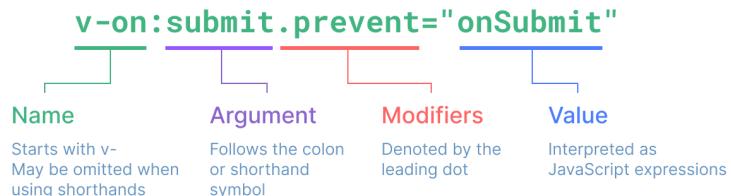
<template>
  <span :id="dynamicId">Message: {{ msg }}</span>
</template>
```

### 3. 多属性绑定

```
<script setup>
const items = {
  foo: 'bar',
  baz: 42
}
</script>

<template>
  <div v-bind="items"></div>
</template>
```

### 4. 完整的指令语法



详见：<https://vuejs.org/guide/essentials/template-syntax.html>

## 5. 条件渲染 (操作DOM)

```
<template>
  <div v-if="awesome">Vue is awesome!</div>
  <div v-else>Oh no 😢 </div>
</template>
```

## 6. 条件渲染 (控制 display )

```
<template>
  <div v-show="awesome">Vue is awesome!</div>
</template>
```

详见: <https://vuejs.org/guide/essentials/conditional.html>

## 7. 列表渲染

```
<template>
  <ul>
    <li v-for="item in items" :key="item.id" :text="item.text">
      {{ item.text }}
    </li>
  </ul>
</template>
```

详见: <https://vuejs.org/guide/essentials/list.html>

## 8. 事件绑定

```
<template>
  <button @click="count++>Add 1</button>
  <p>Count is: {{ count }}</p>

  <!-- using $event special variable -->
  <button @click="warn('Form cannot be submitted yet.', $event)"> Submit </button>

  <!-- using inline arrow function -->
  <button @click="(event) => warn('Form cannot be submitted yet.', event)"> Submit </button>
</template>

<script setup>
  function warn(message, event) {
    // now we have access to the native event
    if (event) {
      event.preventDefault()
    }
    alert(message)
  }
</script>
```

详见：<https://vuejs.org/guide/essentials/event-handling.html>

## 9. class 的绑定

```
<template>
  <div :class="{ active: isActive }"></div>
</template>

<template>
  <div :class="[isActive ? activeClass : '', errorClass]"></div>
</template>
```

## 10. style 的绑定

```
<script>
const activeColor = ref('red')
const fontSize = ref(30)
</script>

<template>
  <div :style="{ color: activeColor, fontSize: fontSize + 'px' }"></div> ←注意单位，和浏览器JS操作样式类似→
</template>
```

## 11. 双向绑定

```
<p>Message is: {{ message }}</p>
<input v-model="message" placeholder="edit me" />
```

# 响应式API

💡 不要忘记导入Vue的API！

## 1. `setup()` 和 `<script setup>` 语法

```
<script>
import { ref } from 'vue'

export default {
  // `setup` is a special hook dedicated for the Composition API.
  setup() {
    const count = ref(0)

    // expose the ref to the template
    return {
      count
    }
  }
}
</script>
```

使用SFC（单文件组件）时，可以使用 `<script setup>` 语法糖，但是因为我们的蓝桥杯不使用构建工具，所以应该用不上了！

## 2. ref() 和 reactive()

```
<script>
import { ref } from 'vue'

export default {
  setup() {
    const count = ref(0)
    function increment() {
      // 在<script>里需要用.value访问，在模板里不需要
      count.value++
    }
    return {
      count,
      increment
    }
  }
}
</script>
```

ref 可以持有任何值类型，包括对象、数组或 JavaScript 内置数据结构如 Map。ref 自带深度响应性，可以观测嵌套的对象。

reactive 本身就是一个响应式对象，对于初学者来说由于其有使用限制，所有情景下都使用 ref 即可。

详见：<https://vuejs.org/guide/essentials/reactivity-fundamentals.html#deep-reactivity>

### 3. 计算属性

```
1 <script setup>
2 import { reactive, computed } from 'vue'
3
4 const author = reactive({
5   name: 'John Doe',
6   books: [
7     'Vue 2 - Advanced Guide',
8     'Vue 3 - Basic Guide',
9     'Vue 4 - The Mystery'
10    ]
11 })
12
13 // a computed ref
14 const publishedBooksMessage = computed(() => {
15   return author.books.length > 0 ? 'Yes' : 'No'
16 })
17 </script>
18
19 <template>
20   <p>Has published books:</p>
21   <span>{{ publishedBooksMessage }}</span>
22 </template>
```

计算属性自动跟踪其响应式依赖， 默认只读。详见<https://vuejs.org/guide/essentials/computed.html>。

## 4. 倾听器

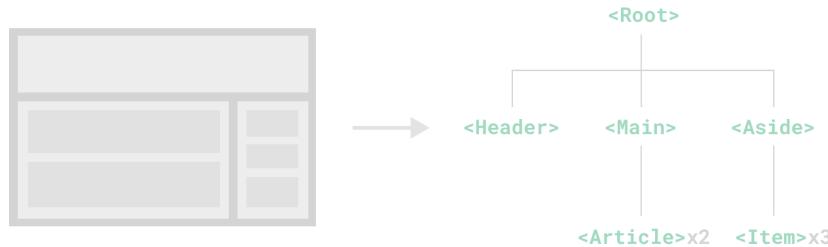
计算属性允许我们声明式地计算派生值。然而，在某些情况下，我们需要在状态变化时执行“副作用”，例如修改DOM，或根据异步操作的结果更改另一部分状态。详见：<https://vuejs.org/guide/essentials/watchers.html>。

```
<script setup>
import { ref, watch } from 'vue'

const question = ref('')
const answer = ref('Questions usually contain a question mark.')
const loading = ref(false)

// watch works directly on a ref
watch(question, async (newQuestion, oldQuestion) => {
  if (newQuestion.includes('?')) {
    loading.value = true
    answer.value = 'Thinking ...'
    try {
      const res = await fetch('https://yesno.wtf/api')
      answer.value = (await res.json()).answer
    } catch (error) {
      answer.value = 'Error! Could not reach the API. ' + error
    } finally {
      loading.value = false
    }
  }
})
```

# 组件API



## 1. 属性传递

```
<script>
export default {
  props: {
    title: String, likes: Number
  }, // 对象语法，值为构造函数
  setup(props) { // setup() receives props as the first argument.
    console.log(props.foo)
  }
}
</script>
```

详见：<https://vuejs.org/guide/components/props.html>

## 2. 事件传递

```
!--> MyComponent -->
<button @click="$emit('someEvent', 1, 2, 3)">Click Me</button> !-- 传递参数 -->
```

在 `setup` 函数中声明事件：

```
export default {
  emits: ['inFocus', 'submit'],
  setup(props, { emit }) {
    emit('submit')
  }
}
```

详见： <https://vuejs.org/guide/components/events.html>

# 生命周期API

例如，可以使用 `onMounted` 钩子在组件完成初始渲染并创建 DOM 节点后运行代码：

```
<script>
export default {
  setup() {
    onMounted(async () => {
      console.log(`the component is now mounted.`)
      await fetchData()
    })
  }
}
</script>
```

详见：<https://vuejs.org/guide/built-ins/lifecycle.html>

# 这里略去不讲，但你应该了解的话题

1. 可写的计算属性: <https://vuejs.org/guide/essentials/computed.html#writable-computed>
2. 表单输入的值绑定: <https://vuejs.org/guide/essentials/forms.html#value-bindings>
3. 侦听器的其它用法：深度侦听、立即侦听、watchEffect：  
<https://vuejs.org/guide/essentials/watchers.html#watchers>
4. 模板引用: <https://vuejs.org/guide/essentials/template-refs.html>
5. 定义 model : <https://vuejs.org/guide/components/v-model.html>
6. 定义 slots : <https://vuejs.org/guide/components/slots.html>
7. 依赖注入 : <https://vuejs.org/guide/components/provide-inject.html>
8. 逻辑复用 : <https://vuejs.org/guide/reusability/composables.html>