**$event:** هرگاه برای استفاده از تابع در تگ html از یک آرگومان استفاده کنیم مانند ‌func(arg) و به event داخلی آن تگ نیاز داشته باشیم می­توانیم با $event به آن دسترسی داشته باشیم: func($event, arg)

**نکته:**

فراخوانی تابع در کد html مانند {{ func() }} باعث کاهش performanceمی­شود زیرا با تغییر هر بخش از صفحه، vue آن تابع را نیز دوباره اجرا می­کند، راه حل بهتر برای اینکار استفاده ازcomputed است.

**عبارات شرطی:**

می­توانیم از v-ifو v-else-if و v-else استفاده کنیم. همچنین می­توانیم به جای v-if از v-show استفاده کنیم اما v-show از v-else پشتیبانی نمی­کند، بهتر است برای تگ­هایی که خیلی visible و hidden می­شوند از v-show استفاده کنیم چون v-show فقط css تگ را تغییر می­دهد اما v-if کل تگ را از DOM حذف و یا به آن اضافه می­کند و ممکن است برای تگ­هایی که خیلی مخفی و آشکار می­شوند مناسب نباشد و کارایی را کاهش دهد.

دیتا را میتوان به وسیله props و $emit بین کامپوننت فرزند و والد رد و بدل کرد اما نمیتوان بین دو کامپوننت فرزند رد و بدل کرد بنابراین برای انتقال دیتا از یک کامپوننت فرزند به دیگر کامپوننت فرزند باید ابتدا دیتا را به کامپوننت والد فرستاده و از آنجا به کامپوننت فرزند دیگر منتقل کرد.

برای رد و بدل کردن دیتا بین دو کامپوننت فرزند میتوان در فایل main، یک شی دیگر از ویو ساخته و در کامپوننت های فرزند از متدهای آن شی استفاده کنیم، مانند متدهای .$emit & $on

Slot:

برای انتقال کدهای html به کامپوننت فرزند از سوی والد استفاده می­شود. برای style دادن به کدهای انتقال یافته باید style مورد نظر در کامپوننت فرزند اعمال شود، زیرا در صورتی که style کامپوننت والد در حالت scoped باشد استایلی برای کدهای در حال انتقال اعمال نمی­شود. همچنین غیر از style، کدهای دیگر ویو مانند دیتای متغیر یا v-if & v-for و ...، باید در کامپوننت والد و بین تگ­های کامپوننت فرزندی که در والد فراخوانی شده است نوشته شود.

@click.native:

هنگامی استفاده می‌شود که بخواهیم روی یک کامپوننت رخداد کلیک را تعریف کنیم، زیرا کامپوننت فراخوانی شده به Dom اضافه نشده است چون فقط کدهای html موجود در کامپوننت به Dom اضافه می‌شوند.

Lazy modifier:

هنگامی که می‌خواهیم با کلیک کردن روی رخدادی دیگر تغییرات در رخداد فعلی اعمال شود از lazy استفاده می‌کنیم. بعنوان مثال: در فرمی که از v-model استفاده کرده‌ایم با وارد کردن هر حرف مقدار ورودی تغییر می‌کند اما با تغییر آن به v-model.lazy=”” مقدار ورودی تنها زمانی اعمال می‌شود که ما روی input دیگری کلیک کنیم.

Number modifier:

هنگامی که می‌خواهیم مقدار ورودی بعنوان عدد شناخته شود از این قابلیت استفاده می‌کنیم، زیرا مقادیر ورودی string در نظر گرفته می‌شوند. مثال:‌ v-model.number=” “ یا v-model.lazy.number.

**ساختن کامپوننت داینامیک:**

برای این کار باید از تگ is=”componentsNameByVariable”></component> <component. اگر بخواهیم زمانی که کامپوننت­ها را به صورت داینامیک پیمایش می­کنیم کامپوننت قبلی از بین نرود و هنوز در صفحه بماند به این صورت عمل می­کنیم:

<keep-alive>

<component is=”componentsNameByVariable”>

</component>

</keep-alive>

:filters:{}فیلترها را می­توان بوسیله hook filters:{} ایجاد کرد، اما در مواقعی استفاده از computed property ها برای فیلتر کردن و حفظ کارایی سیستم بهتر است زیرا Vue می­­تواند تشخیص دهد که چه وقت آن property باید اجرا شود زیرا آن متدی است که شبیه یک متغیر با آن رفتار می­شود، اما در filters:{} ممکن است که کد چندین بار اجرا شود در صورتی که نیاز نیست. مانند:‌

computed:{

filteredFruits(){

return this.fruits.filter((element) => {

return element.toLowerCase().match(this.filterText)

}) } }