Vue is declarative (Define a template of part)

Const app = vue.createApp {data() return}去设置Vue, app.mount(#id)

如果Vue 控制某个HTML组件， 则该组件的所有子组件都被Vue控制.

**v-bind** -> Bind/Set attribute values

**v-html** -> values is HTML element

**v-on:click** -> Button Click event trigger

**methods** ALLOW function -> pass JS function.

Method:{ functionName() { function } }

Data are stored in global.

functionName(event) : this.attribute = event.target.value -> 读取input值

preventDefault -> do not submit automatically.

Prevent modifier: Prevent browser default

Stop modifier : stopping paragation.

v-on:click -> left, right and middle click.

v-once: Only be evaluated ONCE.

Keyboard modifier (keyup/ keydown)

v-model = v-bind:value + v-on:input

(**2 way binding** , listen the event and write the event)

DOM Interaction : (templates and data binding).

**Computed Properties**: once dependence array changes. Defined amount of methods but execute differently from methods.

For performance, computed properties are better than methods for **outputting values** in most cases.

Using Method to Re-calculate events otherwise using computed properties(FOR OUTPUT).

**Watch** (option) such as methods, data, computed property(similar):

Once the data property changes, the **watch** method will re-execute again.

**Watch**:{PropertyName(**value){method}**} -> when the data meet specific condition, then, it will changed by inside method.

**Watch** Option will help to set HTTP/ setTimer Method .

Using **Watch** Option when we need to add more functionality after **the condition meet**. Using **Computed** Option to **OUTPUT values.**

**Short Summary:**

* **Method:** Data Bind(executed for every re-render), Event Bind.
  + **Use for events/data that REALLY needs to be re-evaluated all time**
* **Computed: Data Bind,** ONLY re-evaluated if one of used values change
  + **(Use for data that depends on other data)**
* **Watch:** Not directly use in template,
  + **Use for any non-data update you want to make.**
* **V-on shorthand : @; v-bind shorthand : “:”**

**Inline CSS:**

* **:style=”{attribute: }”**

**v-if (show tag after meet conditions)**

**v-else (directive after v-if) ->** Showing content conditionally

**v-show(**v-show are more like Not shown, v-else more like NOT exist)

**v-for**(using “in” as keyword) -> v-for = “I in arrayName” (Such as for loop)

* Value in {name:’val1’, age:’val2’} -> read the value of property.
* (I, index) in arrayName ->{{i}}-{{index}} : Element – element position
* V-for=”(value,key,index) in {name:’val1’,age:’val2’}”
* V-for=”num in NUMBERS” -> get #
* Using **key attribute,** :key=”i” (unique identification)

Array -> Using **push** to **add** element into this element

* Using **splice**(index, position) -> find this index and **remove** from this array

**Stop** event modifier -> This is stop processing of previous functionality

“this” property not only using in **data** but also in **methods**.

Component(Custom HTML element) : 1. **Identifier** (contain a dash( ‘-‘)

* Template

Vue create [VueAppName] ->npm run server

In vue file: <template>(Contain HTML Element), <script>(Functionality),

Component-> app.component(ComponentName)

**Props:** parent child communication (**pass** data from parent to child)

* **Props** should NOT mutated
* Once, the data passed from parent to child. It can NOT changed by parent
* Set a new value in data, and set the value to this.propName.
* Props can set type(such as string), required, default, validator(function),
* Type: String , Number Boolean Array Object Date Function Symbol
* **Using v-bind to represent the value** -> v-for = element in array; key=”array.id” ; :name=”array.name”
* **This**.$emit() -> Allows to emit custom event , listen from parent component.

Filter : let arr = [1, 2, 3, 4, 5, 6, 7, 8];

let result = arr.filter((item) => item > 5);

console.log(result); // [ 6, 7, 8 ]

,reduce let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

let result = arr.reduce(function (total, currentValue, currentIndex, arr) {

return total + currentValue;

}, 0);

console.log(result); // 55

, find let arr = [1, 2, 3, 4, 5, 6];

let result = arr.find((item) => item >= 2);

console.log(result); // 2

, map let arr = ["Tom", "Mike", "Shanguagua"];

let result = arr.map((item, index) => {

return `第${index + 1}名：${item}`;

});

console.log(result); // [ '第1名：Tom', '第2名：Mike', '第3名：Shanguagua' ]

**Provide,Inject: (work together):**

* **Only Inject what has been provide**

**Local vs Global Component:**

* Using **component** in export default, import ComponentName from location. Component:{ComponentName}
* If this **component** use in lots of other places, then We are going to use it globally(app.component() -> in main.js file)

**Styling**:

* If the style inside of vue file, then it is global styling.
* If **Style Scoped ->** Then it will only affect this file

**Slot:**

* Allow us to receive HTMl content from outside of the component.
* Using for HTML code.
* <slot name=”slotName”> to have more slots-> Sepific which slot need to be used in there. 和 <template v-slot:slotName>一起使用
* Slot 通俗的理解就是“占坑”，在组件模板中占好了位置，当使用该组件标签时候，组件标签里面的内容就会自动填坑（替换组件模板中slot位置）  
  并且可以作为承载分发内容的出口
* Slot will not change the props, component, methods….

Dynamic Component:

* <component :**is=”methodName”**></component>
  + 赋予component functionality
* <keep-alive> -> 不完全删除vue组件， 保存其状态与数据
* <teleport to=”attributeName(such as body)”>