Another way to make the properties reactive is to use “*reactive()*” function.

The first difference from developer perspective is that using “*ref()*” you need to declare each single property

const dice = ref(0);  
const rolls = ref([]);

and when you send the properties to the template you need to list them individually:

return { dice, rolls, total, roll, restart };

With “*reactive()*” function you need to collect all properties in one object:

const game = reactive(  
 {  
 dice: 0,  
 rolls: []  
 }  
)

With this syntax you can pass to the template just the object (game object) instead of all properties (dice, rolls…):

return {  
 game, //data  
 roll, restart //functions  
};

Another difference is in the properties usage. You can avoid to use “*.value*” but you need to use it as object attribute “*game.dice*”. For example:

function restart() {  
 game.dice=0  
 game.rolls = [];  
}

An additional thing: if you want to avoid to use “*game.dice*” and “*game.rolls*” and you want use just “*dice*” and “*rolls*”, you could use the object destructuring when you send properties to template (in the return clausole of setup method).

The template

Thanks to the usage of destructuring object, the template is still the same of “*ref()*” approach.

<template>  
 <h1>Number is: {{ dice }}</h1>  
 <div>Number of rolls: {{ rolls.length }}</div>  
 <div>Total: {{ total }}</div>  
 <button [@click](http://twitter.com/click)="roll()">Let's roll the dice</button>  
 <button [@click](http://twitter.com/click)="restart()">Restart</button>  
 <ul>  
 <li v-for="(t, index) in rolls" :key="index">  
 {{ t }}  
 </li>  
 </ul>  
</template>

Walking through the code

<script>  
// ##001 : import from vue3:  
// - reactive for making properties reactive;  
// - computed for computed function like total  
// - toRefs for destructuring object for template  
import { reactive, computed, toRefs } from "vue";  
export default {  
 name: 'RollTheDiceReactive',  
 // ##002 : implement setup function  
 setup() {  
 // ##003 : declare and initialize reactive object  
 const game = reactive(  
 {  
 dice: 0,  
 rolls: [],  
 // ##004 : we include also computed properties in game object  
 total: computed(  
 () => {  
 let temptotal = 0;  
 for (let i=0 ; i< game.rolls.length; i++) {  
 temptotal = temptotal + game.rolls[i]  
 }  
 return temptotal;  
 }  
 )  
 }  
 )  
 // ##005: implement roll function (inside setup() )  
 function roll() {  
 game.dice = Math.floor(Math.random() \* Math.floor(5)) + 1;  
 game.rolls.unshift(game.dice);  
 }// ##06: implement restart function (inside setup() )  
 function restart() {  
 game.dice=0  
 game.rolls = [];  
 }// ##007: expose to the template all stuff (object, functions etc)  
 return {  
 ...toRefs(game), //data  
 roll, restart //functions  
 };  
 }  
}  
</script>