# Creating Vue.js Components and Using Template Syntax



Jim Cooper SOFTWARE ENGINEER @jimthecoop jcoop.io

# Agenda

**Creating Components** 

Using Bindings to Display Data and Handle Events

Conditionally Displaying Elements with v-if and v-show

Repeating Elements with v-for

**Styling Components** 

**Working with Component Lifecycle Hooks** 

**Reducing Duplication with Mixins** 

**Composition and Reactivity APIs** 



# Global Components

VS

# Local Components



```
<script>
import HomePage from './components/HomePage.vue';

export default {
  name: 'App',
  components: {
    HomePage,
  },
};
</script>
```

# Local Components



```
import { createApp } from 'vue';
import App from './App.vue';

createApp(App) emonApp(#app):nount('#app');
```

Global Components



```
import { createApp } from 'vue';
import App from './App.vue';
import [HomePage] from './components/HomePage.vue';
const app = createApp(App).mount('#app');
[app.component([HomePage'][HomePage]);
```

# Global Components



</script>

# Global Components



# Global Components Drawbacks

**Global Variables** 

Increased Bundle Sizes











#### 



### Composition API

#### **Options API**

Simple

Easy to see everything at a glance

Larger components become convoluted

#### **Composition API**

Allows for abstraction / composition

Splits major concepts into separate files

Easier to understand in complex components



# Example: Search Component

```
data() {
                                                             Search
computed: {
                                                              Filter
methods: {
                                                           Pagination
```



# Example: Search Component

```
data() {
computed: {
methods: {
```

Search

**Filter** 

**Pagination** 



```
<script>
export default {
 setup() {
   let results = searchInventory();
    const search = () => { ... };
    return {
     searchResults: results,
      search,
```



```
<script>
export default {
  props: { searchTerm: { type: String } },
  components: { SearchResultsItem },
 setup(props) {
   let results = searchInventory(props.searchTerm);
    const search = () => { ... };
    return {
      searchResults: results,
      search,
</script>
```

# Lifecycle Hooks

#### **Options API**

beforeCreate created beforeMount mounted beforeUpdate updated beforeUnmount unmounted errorCaptured renderTracked renderTriggered activated deactivated

#### **Composition API**

\*Not Needed\* \*Not Needed\* onBeforeMount onMounted onBeforeUpdate onUpdated onBeforeUnmount onUnmounted onErrorCaptured onRenderTracked onRenderTriggered onActivated onDeactivated



```
import { onMounted } from 'vue';
setup(props) {
 let results = searchInventory(props.searchTerm);
  const search = () => { ... };
  const handleMounted = () => { ...};
  onMounted(handleMounted);
  return {
    searchResults: results,
    search,
```

```
setup(props) {
 let currentPage = 1;
  const handlePagingMount = () => { ... };
  const handleMount = () => {
    handleSearchMount();
    handlePagingMount();
 onMounted(handleMount);
  return {
   filters,
    currentPage,
    applyFilters,
   nextPage,
    prevPage
```

Search

**Filter** 

**Pagination** 



```
setup(props) {
  let results = searchInventory(props.searchTerm);
  let filters = { };
  let currentPage = 1;
  const search = () => { ... };
  const applyFilters = () => { ... };
  const nextPage = () => { ... };
  const prevPage = () => { ... };
  const handleSearchMount = () => { ... };
  const handleFilterMount = () => { ... };
  const handlePagingMount = () => { ... };
  const handleMount = () => {
    handleSearchMount();
    handleFilterMount();
    handlePagingMount();
  onMounted(handleMount);
  return {
    searchResults: results,
    filters,
    currentPage,
    search,
    applyFilters,
    nextPage,
    prevPage
```

```
setup(props) {
 let currentPage = 1;
  const handlePagingMount = () => { ... };
  const handleMount = () => {
    handleSearchMount();
    handlePagingMount();
 onMounted(handleMount);
  return {
   filters,
    currentPage,
    applyFilters,
   nextPage,
    prevPage
```

Search

**Filter** 

**Pagination** 



```
setup(props) {
 let filters = { };
 let currentPage = 1;
  const handlePagingMount = () => { ... };
  const handleMount = () => {
    handleSearchMount();
   handleFilterMount():
   handlePagingMount();
 onMounted(handleMount);
  return {
   searchResults: results,
   filters,
   currentPage,
   applyFilters,
   nextPage,
   prevPage
```

```
onMounted(() => { ... });
 const applyFilters = () => { ... };
 return { filters, applyFilters };
export default function usePagination() {
 let currentPage = 1;
 const nextPage = () => { ... };
 const prevPage = () => { ... };
 onMounted(() => { ... });
  return { currentPage, nextPage, prevPage };
```

```
import usePagination from './usePagination';
setup(props) {
```

```
const applyFilters = () => { ... };
  return { filters, applyFilters };
export default function usePagination() {
 let currentPage = 1;
  const nextPage = () => { ... };
  const prevPage = () => { ... };
 onMounted(() => { ... });
  return { currentPage, nextPage, prevPage };
```

```
import useSearch from './useSearch';
import useFilters from './useFilters';
import usePagination from './usePagination';
setup(props) {
  return {
    ...useSearch(props),
    ...useFilters(),
    ...usePagination()
  }
}
```

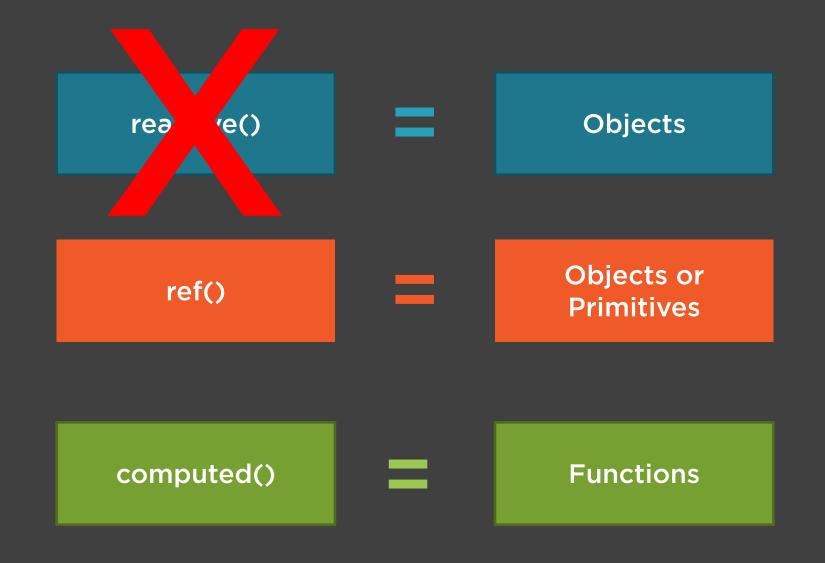
```
onMounted(() => { ... });
 const applyFilters = () => { ... };
  return { filters, applyFilters };
export default function usePagination() {
 let currentPage = 1;
  const nextPage = () => { ... };
  const prevPage = () => { ... };
 onMounted(() => { ... });
  return { currentPage, nextPage, prevPage };
```

```
import usePagination from './usePagination';
                                                                onMounted(() => { ... });
setup(props) {
 const { searchResults, search useSearch(props);
 const { filters, applyFilters } useFilters(searchResults);
 const { currentPage, nextPage, prevPage } =
   usePagination();
                                                              export default function useFilters() {
                                                                const applyFilters = () => { ... };
 return {
   searchResults,
   filters,
                                                                return { filters, applyFilters };
   currentPage,
   nextPage,
   prevPage
                                                              export default function usePagination() {
                                                                let currentPage = 1;
                                                                const nextPage = () => { ... };
                                                                const prevPage = () => { ... };
                                                                onMounted(() => { ... });
                                                                return { currentPage, nextPage, prevPage };
```

```
<template>
 Count : {{ count }}
  <button @click="increment()">Increment Count</button>
</template>
<script>
setup(props) {
 let count = 0;
  const increment = () => count = count + 1;
  return {
    count,
    increment,
</script>
```

```
<template>
  Count: {{ count }}
  <button @click="increment()">Increment Count</button>
</template>
<script>
export default {
  data() {
    return { count = 0 };
  methods: {
    increment() { this.count = this.count + 1 };
</script>
```

```
<template>
  Count: {{ count }}
  <button @click="increment()">Increment Count</button>
</template>
<script>
setup(props) {
 let count = 0;
  const increment = () => count = count + 1;
  return {
    count,
    increment,
</script>
```



```
<template>
  Count: {{ count }}
  <button @click="increment()">Increment Count</button>
</template>
<script>
export default {
  setup(props) {
   let count = 0;
    const increment = () => count = count + 1;
    return {
      count,
      increment,
</script>
```

```
<template>
  Count: {{ count }}
  <button @click="increment()">Increment Count</button>
</template>
<script>
import { ref } from 'vue';
export default {
  setup(props) {
    let count = ref(0);
    const increment = () => count.value = count.value + 1;
    return {
      count,
      increment,
</script>
```

```
<template>
 User: {{ user.firstName }} {{ user.lastName }}
  <button @click="setFirstName('Jim')">Set First Name</button>
  <button @click="setLastName('Cooper')">Set Last Name</button>
</template>
<script>
import { ref } from 'vue';
export default {
  setup(props)
   let user = ref({ firstName: '', lastName: '' });
    const setFirstName = (name) => user.value.firstName = name;
    const setLastName = (name) => user.value.lastName = name;
    return {
      user,
      setFirstName,
      setLastName
</script>
```

```
<template>
 User: {{ fullName }}
  <button @click="setFirstName('Jim')">Set First Name</button>
  <button @click="setLastName('Cooper')">Set First Name</button>
</template>
<script>
import { ref, computed } from 'vue';
export default {
  setup(props) {
    let user = ref({ firstName: '', lastName: '' });
    const setFirstName = (name) => user.value.firstName = name;
    const setLastName = (name) => user.value.lastName = name;
    return {
     liser
     fullName: computed(() => user.value.firstName + ' ' + user.value.lastName),
</script>
```

Requires Computed **Primitives** Objects .value? Method Props reactive() ref() computed()



```
setup(props)
 <u>const products = reactive(['</u>apples', 'oranges', 'grapes']);
 const filter = (text) =>
     const filteredProducts = products.filter(p => p.includes(text));
     return products.splice(0, products.length, ...filteredProducts);
  return { products, filter, }
</script>
setup(props) {
  const products = ref(['apples', 'oranges', 'grapes']);
  const filter = (text) => {
    return products.value = products.filter(p => p.includes(text));
  return { products, filter, }
</script>
```



Computed Requires **Primitives Objects** .value? Method Props ref() computed()

# Summary



**Creating Components** 

Using Bindings to Display Data and Handle Events

Conditionally Displaying Elements with v-if and v-show

Repeating Elements with v-for

**Styling Components** 

Working with Component Lifecycle Hooks

**Reducing Duplication with Mixins** 

**Composition and Reactivity APIs** 

