Using the Composition and Reactivity APIs

Jim Cooper
Software Engineer

@jimthecoop | jcoop.io

Agenda

The Composition API

setup() function vs <script setup>

The Reactivity API

Ref, Reactive, and Computed

Lifecycle Hooks with the Composition API

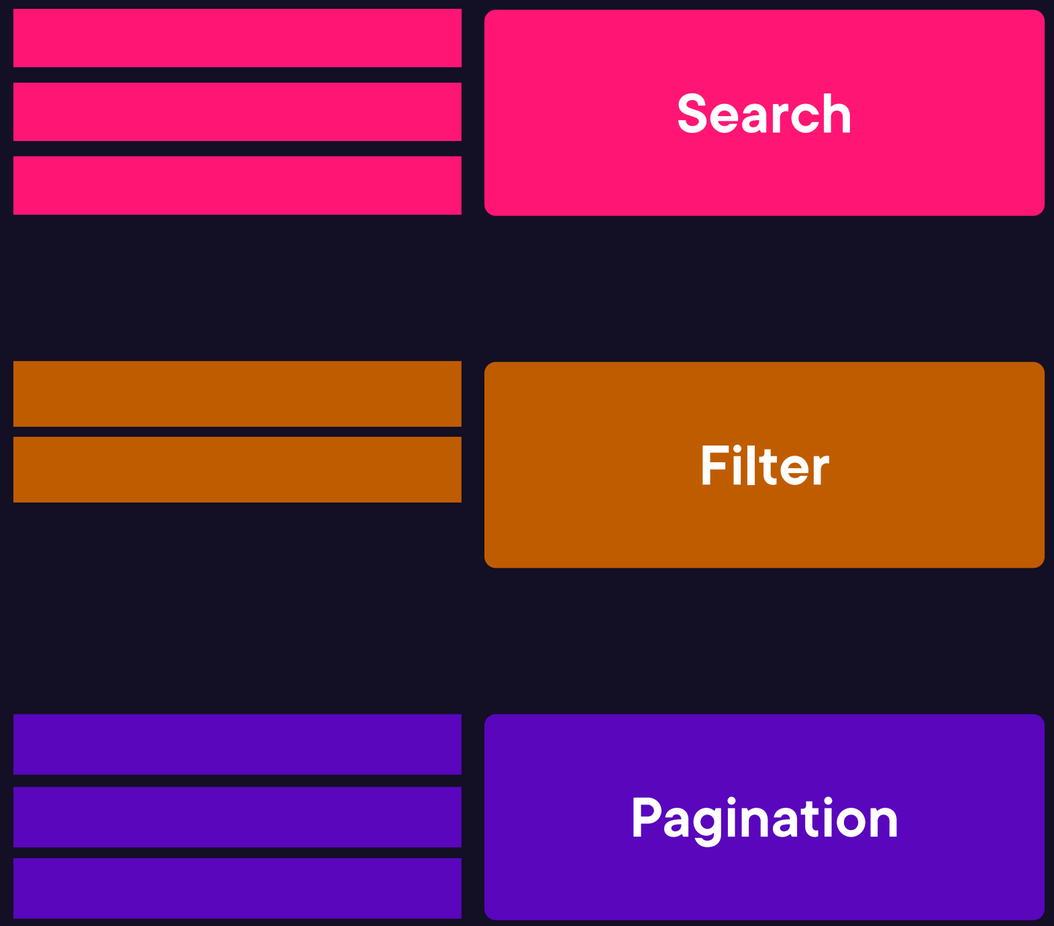
Composition API Benefits in Complex Components

Options API vs. Composition API

Options API	Composition API
Simple	Allows for abstraction / composition
Easy to see everything at a glance	Can split major concepts into separate files
Larger components become convoluted	Easier to understand in complex components

Example: Search Component

```
data() {


}
computed: {


}
methods: {


}
```



Example: Search Component

```
data() {
    
}

computed: {
    
}

methods: {
    
}
```

Search

Filter

Pagination

```
<script>
export default {



  setup() {
    let results = searchInventory();

    const search = () => { ... };

    return {
      searchResults: results,
      search,
    };

  }



}
</script>
```

```
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
  };
}
```

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 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 results = searchInventory(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,
    applyFilters,
    nextPage,
    prevPage
  };
}
```

```
export default function useSearch(searchTerm) {
  let results = searchInventory(searchTerm);

  const searchResults = computed(() => { ... });
  onMounted(() => { ... });

  return { searchResults: results };
}
```

```
export default function useFilters() {
  let filters = { };
  const applyFilters = () => { ... };

  onMounted(() => { ... });

  return { filters, applyFilters };
}
```

```
export default function usePagination() {
  let currentPage = 1;
  const nextPage = () => { ... };
  const prevPage = () => { ... };

  onMounted(() => { ... });

  return { currentPage, nextPage, prevPage };
}
```

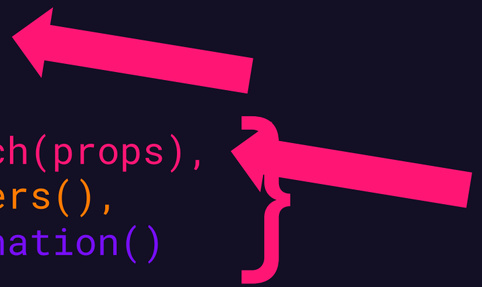
```
import useSearch from './useSearch';
import useFilters from './useFilters';
import usePagination from './usePagination';

setup(props) {
  return useSearch(searchTerm);
}
```

```
export default function useSearch(searchTerm) {
  let results = searchInventory(searchTerm);

  const searchResults = computed(() => { ... });
  onMounted(() => { ... });

  return { searchResults: results };
}
```

```
export default function useFilters() {
  let filters = { };
  const applyFilters = () => { ... };

  onMounted(() => { ... });

  return { filters, applyFilters };
}
```

```
export default function usePagination() {
  let currentPage = 1;
  const nextPage = () => { ... };
  const prevPage = () => { ... };

  onMounted(() => { ... });

  return { currentPage, nextPage, prevPage };
}
```

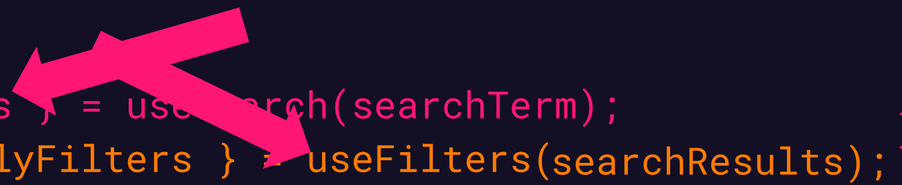
```
import useSearch from './useSearch';
import useFilters from './useFilters';
import usePagination from './usePagination';

setup(props) {
  return {
    ...useSearch(props),
    ...useFilters(),
    ...usePagination()
  }


}
```




```
export default function useSearch(searchTerm) {
  let results = searchInventory(searchTerm);

  const searchResults = computed(() => { ... });
  onMounted(() => { ... });

  return { searchResults: results };
}
```

```
export default function useFilters() {
  let filters = { };
  const applyFilters = () => { ... };

  onMounted(() => { ... });

  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) {
  const { searchResults } = useSearch(searchTerm);
  const { filters, applyFilters } = useFilters(searchResults);
  const { currentPage, nextPage, prevPage } =
    usePagination();

  return {
    searchResults,
    filters,
    applyFilters,
    currentPage,
    nextPage,
    prevPage
  }

}
```



```
export default function useSearch(searchTerm) {
  let results = searchInventory(searchTerm);

  const searchResults = computed(() => { ... });
  onMounted(() => { ... });

  return { searchResults: results };
}


export default function useFilters() {
  let filters = { };
  const applyFilters = () => { ... };

  onMounted(() => { ... });

  return { filters, applyFilters };
}


export default function usePagination() {
  let currentPage = 1;
  const nextPage = () => { ... };
  const prevPage = () => { ... };

  onMounted(() => { ... });

  return { currentPage, nextPage, prevPage };
}
```

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

```
setup(props) {
  const products = reactive(['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>
```

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

Lifecycle Hooks

Options API	Composition API
beforeCreate	*Not Needed*
created	*Not Needed*
beforeMount	onBeforeMount
mounted	onMounted
beforeUpdate	onBeforeUpdate
updated	onUpdated
beforeUnmount	onBeforeUnmount
unmounted	onUnmounted
errorCaptured	onErrorCaptured
renderTracked	onRenderTracked
renderTriggered	onRenderTriggered
activated	onActivated
deactivated	onDeactivated

Agenda

The Composition API

setup() function vs <script setup>

The Reactivity API

Ref, Reactive, and Computed

Lifecycle Hooks with the Composition API

Composition API Benefits in Complex Components

Up Next:

Styling Vue Components