Component Communication

John Papa
DEVELOPER ADVOCATE

@john_papa   www.johnpapa.net

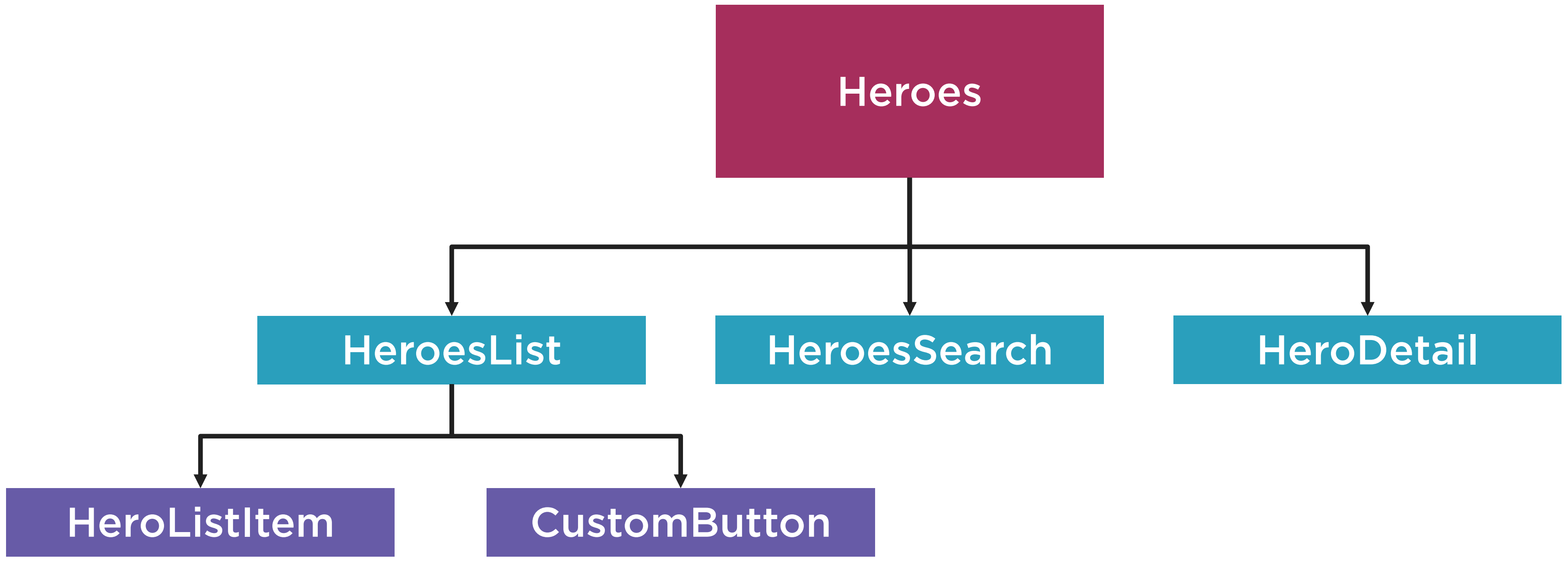
How do we …

- Create a parent and child component relationship?
- Pass values from a parent to a child component?
- Create and fire events in child components
- Listen to those events in the parent component
- Refactor one component into parent and child components
- Share logic across components

Demo

Components in Components

Component Organization



heroes.vue

```
export default {
  name: "Heroes",
  data() { },
  components: { ListHeader,  HeroList, HeroDetail }
};
```

These components are used within this parent component

Components in Components

Components may include other components in their templates

We declare them in the parent component

Demo

Passing Values to Child Components with Props

heroes.vue

We pass values from parent to a child, using bindings and props

```
<HeroDetail
  v-if="selectedHero"
  :hero="selectedHero"
/>
```

hero-detail.vue

```
export default {
  props: {
    hero: {
      type: Object,
      default: () => {},
    },
  },
}
```

Prop Tips

Casing	**camelCased prop names use kebab-cased equivalents in templates**
Types	**String, Number, Boolean, Array, Object, Function, Promise**
Dynamic vs Static	**Dynamic:** `:title="hero.name"` **Static:** `title="Mrs Awesome"`

Use v-bind (or :) with static non-strings

Validation

Any string, or undefined, null

Default value generated from a function

Simple default value

This prop is required

Child Component

```
props: {
  message: String,
  hero: {
    type: Object,
    default: () => {}
  },
  limit: {
    type: Number,
    default: 100
  },
  title: {
    type: String,
    required: true
  }
},
```

Custom Validator Functions

Define your own validation logic

Can also import shared validation functions

Child Component

```
props: {
  result: {
    type: String,
    validator: function (value) {
      return ['success', 'warning', 'danger'].indexOf(value) !== -1
    }
  }
}
```

Value must match one of these strings

Demo

Creating Parent and Child Components

Demo

Passing Objects

Demo

Communicating from Child to Parent

heroes.vue

```
<HeroDetail
  v-if="selectedHero"
  :hero="selectedHero"
  @save="saveHero"
/>
```

Method in heroes.vue which accepts the parameters

hero-detail.vue

```
methods: {
  saveHero() {
    this.$emit("save", this.theHero);
  }
}
```

Pass the hero from the child component

Fire the event

Demo

Using Mixins to Share Logic

Mixins

Distribute reusable functionality across components

Example: methods, computeds, life cycle hooks, data, watches

my-mixins.js

```
export const mymix = {
  created() {
    console.log('created lifecycle hook');
  },
  methods: {
    clear: function() {
      this.$emit('unselect');
    },
  },
};
```

heroes.vue

```
import { mymix } from './my-mixins';

export default {

  // ...

  mixins: [mymix],

  // ...

}
```

This mixin is merged into this component

When Mixins Conflict with Components

methods, components and computeds	**Merged, precedence given to the component’s method**
data	**Merged superset, precedence given to component’s data**
watch and hooks	**Both run, with mixins running before component**

What we Learned

Parent and child components

Define props

Define events

Listen to events

Share logic across components

Summary

Create small components

Communication down with props

Communicate up with events