

Vue Fundamentals - 04

Component communication



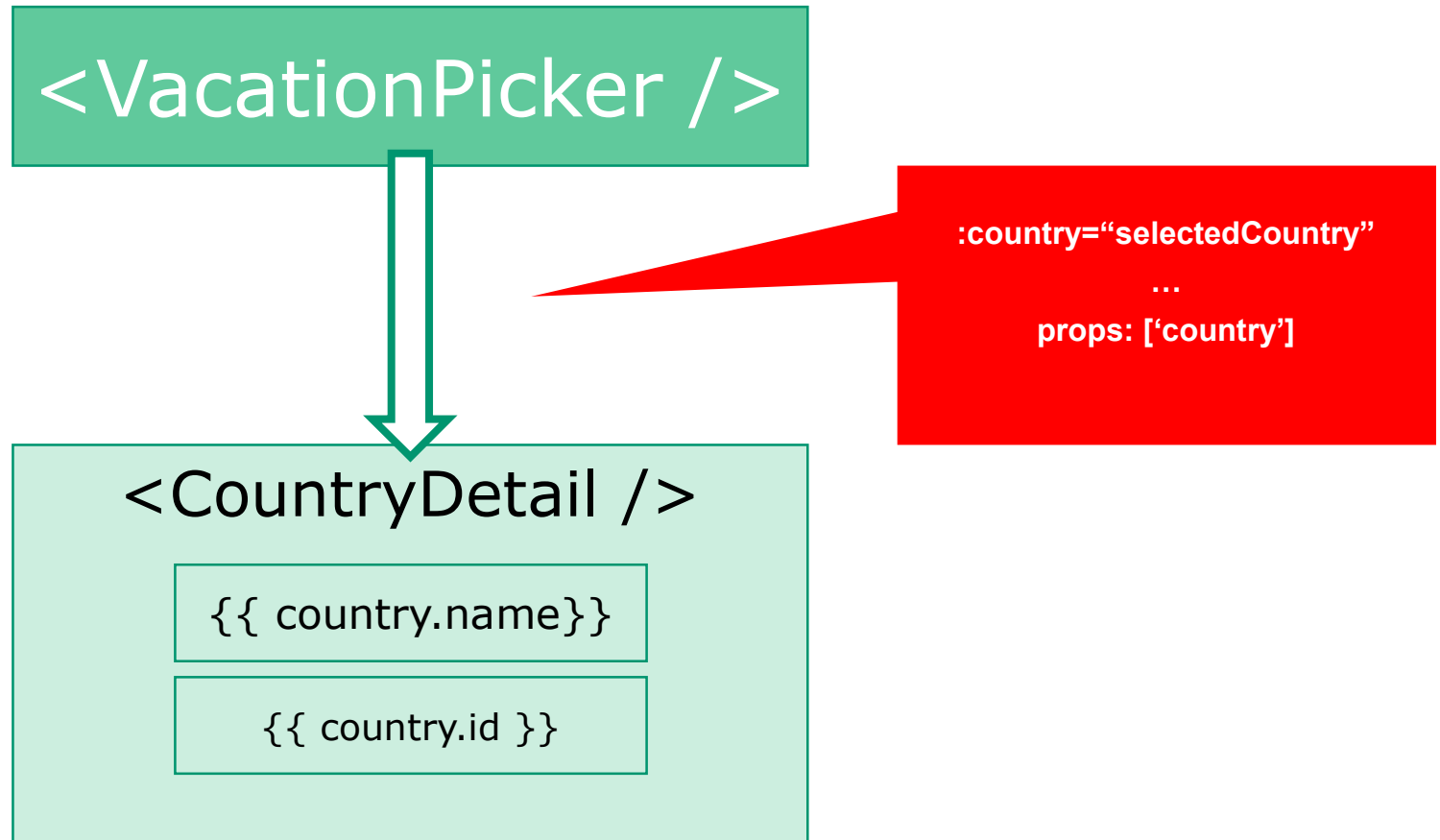
Peter Kassenaar –
info@kassenaar.com

Data flow between components

*"Data flows in to a component via
v-bind: bindings"*

*Data flows out of a component via
v-on: or @event events"*

Parent-Child flow: v-bind: or :



1. Prepare Detail component to receive data

- The data you pass to a component are called *props*.
- Props can be strings, numbers, arrays, objects and so on.
- Props is an array on the component, like:

```
export default {  
  name: "CountryDetail",  
  props: ['country'],  
}
```

We can then bind to the properties of
the passed in country with `country.id`, `country.name`, etc.

2. Update Parent component to send data down



```
<div class="col-6">
  <CountryDetail v-if="showDetails" :country="country" />
</div>
...
<script>
  // import the country data
  import data from '../data/data';
  import CountryDetail from "./CountryDetail";

  export default {
    name: 'VacationPicker',
    components: {CountryDetail},
    ...
  },
</script>
```



Move methods and computed properties

- Move or copy the necessary methods from the parent component to child component,
- In this example:
 - `getImgUrl(img)`
 - `isExpensive()`
 - `isOnsale()`

```
export default {
  name: "CountryDetail",
  props: ['country'],
  methods: {
    getImgUrl(img) {
      console.log(img);
      return require('../assets/countries/' + img);
    }
  },
  computed: {
    isExpensive() {
      return this.country.cost > 4000;
    },
    isOnSale() {
      return this.country.cost < 1000;
    }
  }
}
```

Casing of props

- HTML attributes are *case-insensitive*
- If you use camelCase on prop-names, use a hyphen in the html
 - i.e. props: ['countryDetail', 'countryName'] become
`<DetailComponent country-detail="..." country-name="..." />`
- If you are using string templates this limitation does not apply

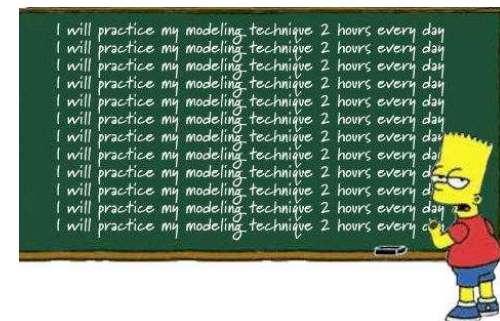
```
Vue.component('blog-post', {  
  // camelCase in JavaScript  
  props: ['postTitle'],  
  template: '<h3>{{ postTitle }}</h3>'  
})
```

```
<!-- kebab-case in HTML -->  
<blog-post post-title="hello!"></blog-post>
```

<https://vuejs.org/v2/guide/components-props.html>

Workshop

- Create a DetailComponent on your own application and pass data. OR:
 - Create an extra prop on the CountryDetailComponent and pass it.
- New: create a new component with a textbox and a button.
 - When the button is clicked, the text in the box is passed as a prop to a child component.
 - Tip: Use `v-model` on the textbox.
- Optional: implement the lifecycle hook `beforeupdate` on the child component, showing a counter that says how many times the component is updated.
- Generic Example on props: [../200-props](#)

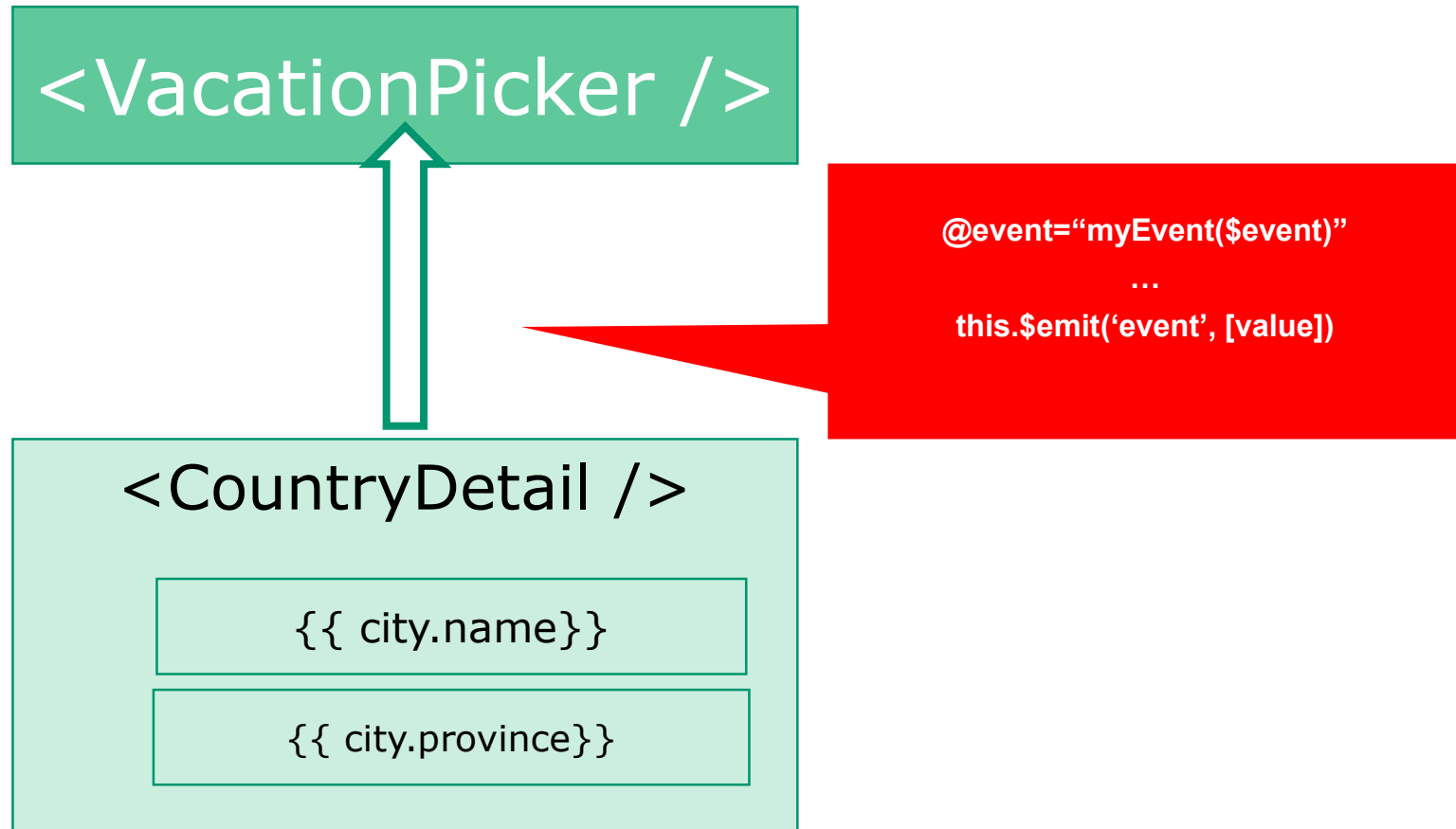




Passing data back

Communicating from child to parent component by sending events

Child-Parent flow: custom events



Binding to custom events

- Custom component can throw **custom events**, by using the `this.$emit('eventName')` method
 - It is automatically available on every component
 - You can define the name of the event yourself
 - You can pass data in the event
- In the parent component, use the well-known `@eventName="handler($event)"` notation
 - Call a local event handler to handle the event
 - `$event` is a magic variable, containing the value from the child

Vue 3 – register the event to emit

- In Vue 3 applications you *have* to tell Vue that a Child component emits an event.
- Otherwise, you'll get an error/warning in the browser console

```
export default {  
  name: "CountryDetail",  
  ...  
  emits: ['rating', 'favorite'],  
  ...  
}
```

Example custom events - Child

Prepare the child component to emit its custom event(s)

```
<span class="float-right">  
  <button @click="setRating(1)">+1</button>  
  <button @click="setRating(-1)">-1</button>  
</span>
```



```
methods: {  
  ...  
  setRating(value){  
    this.$emit('rating', value);  
  }  
}
```

Example custom events – parent

Prepare the parent component to receive custom event(s)

```
<CountryDetail v-if="showDetails"
  @rating="onRating($event)"
  :country="country" />
```

1. Catch event


```
onRating(rating){
  console.log('rating received for ' + this.country.name);
  this.data.countries[this.currentCountryIndex].rating += rating;
}
```

2. Handle event

```
<div v-if="country.rating !== 0">
  my rating:
  <span class="badge badge-secondary badge-pill">{{country.rating}}</span>
</div>
```

3. Show result in UI

Result

 Vue vacation picker

USA

Capital: Washington

my rating: 4

<< Back

Forward >>

Hide details


USA

1

+1 -1

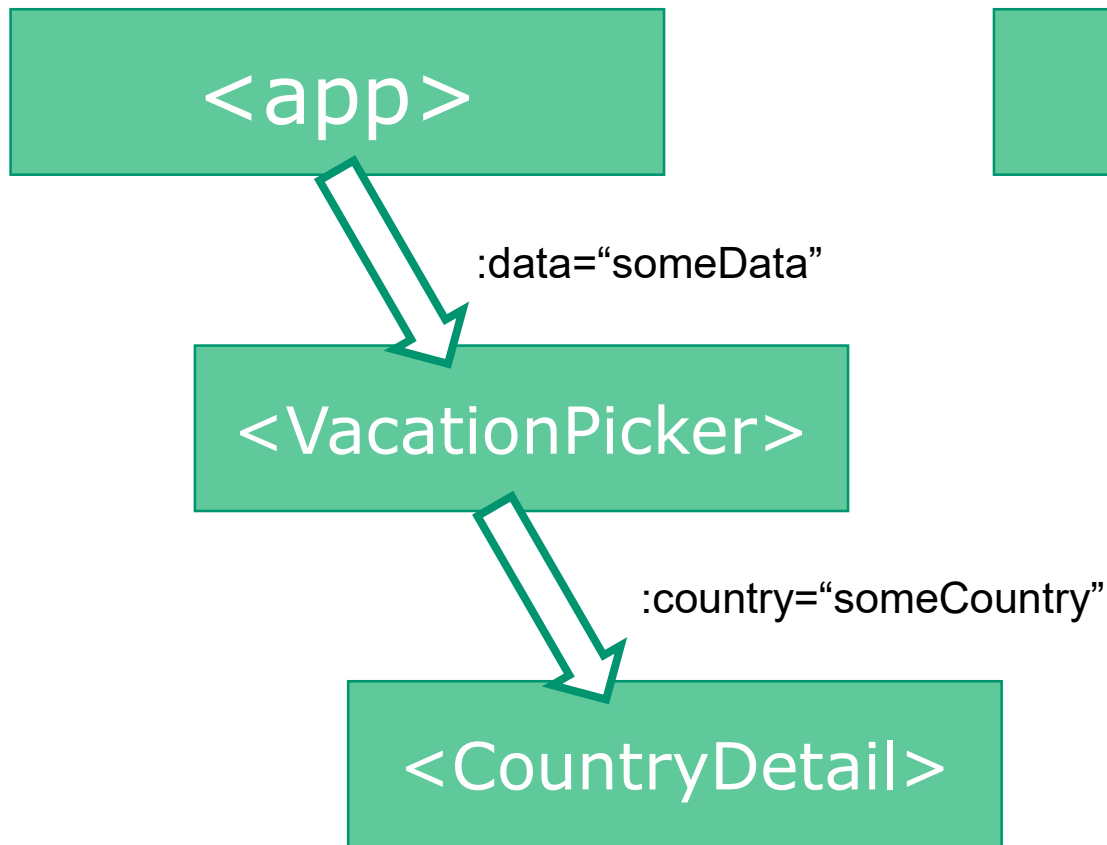
USA

Washington

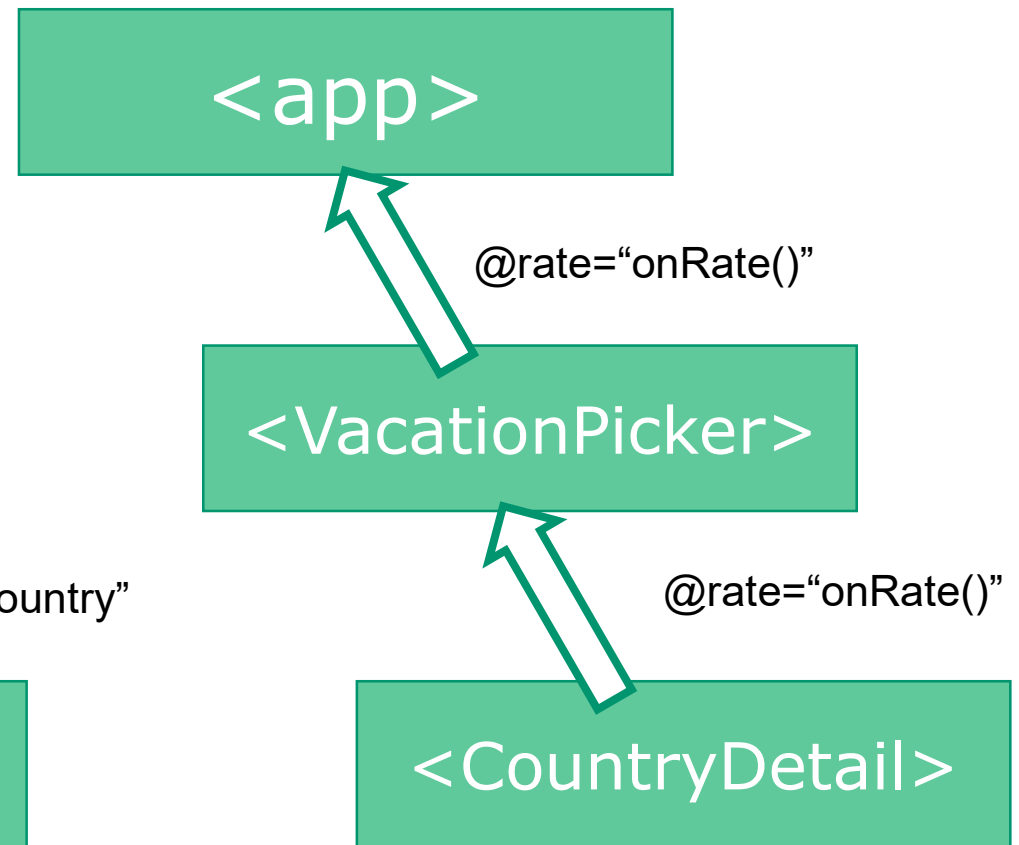


Summary

Parent → Child



Child → Parent



Workshop

- Use `../220-emit-events` as a source, or use your own project
- Add a `favorite` event to the `CountryDetail` component, so a user can mark a country as favorite.
 - Update the data model with a `favorite` property.
 - Update the child component to `$emit` the event.
 - Update the parent component to receive and handle the event
- Generic example: `../220-emit-events`

