

# Vue Events

# Vue Events

- **Props** are used to pass data from a parent component to a child
- **Events** are used to pass data from a child component to its parent
- Events have a name and an optional payload
- Vue component instance methods
  - `$emit` publishes events
  - `$on` subscribes to events
- `v-on` (`@`) template directive also subscribes to an event

```
<div @event-name="expression">
```
- If *expression* is a method name, event arguments are passed to it
- Vue Devtools provides logging and filtering of all Vue events



# Emitting Events

- Vue events emitted from a component can only be subscribed to from the same component instance in a parent component
- Not like DOM events that have a capture and bubbling phase
- For example, a child component named `child` can emit an event named "foo" on a button click

```
<button @click="$emit('foo', value1, value2)">  
  Alpha  
</button>
```

in template of  
`child` component

# Subscribing to Events ...

- A parent component can subscribe to this event on an instance of `Child`

```
<Child @foo="parentMethod" />
```

in template of parent component

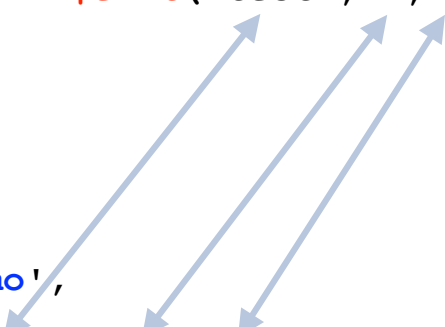
- The event payload values *value1* and *value2* are passed as separate arguments to `parentMethod`



# ... Subscribing to Events

- A component can subscribe to events it emits

```
<template>
  <div>
    <button @click="$emit('test', 1, 2)">
      Trigger
    </button>
  </div>
</template>
<script>
export default {
  name: 'EventDemo',
  mounted() {
    this.$on('test', (n1, n2) => {
      console.log(n1, n2);
    });
  }
};
</script>
```



but it seems easier  
to just use a method

```
<template>
  <div>
    <button @click="test(1, 2)">
      Trigger
    </button>
  </div>
</template>
<script>
export default {
  name: 'EventDemo',
  methods: {
    test(n1, n1) {
      console.log(n1, n2);
    }
  }
};
</script>
```

# Event Bus

- Pattern for sharing data between components that do not have a direct parent-child relationship
- Alternative to using a state management library like Vuex that some developers prefer



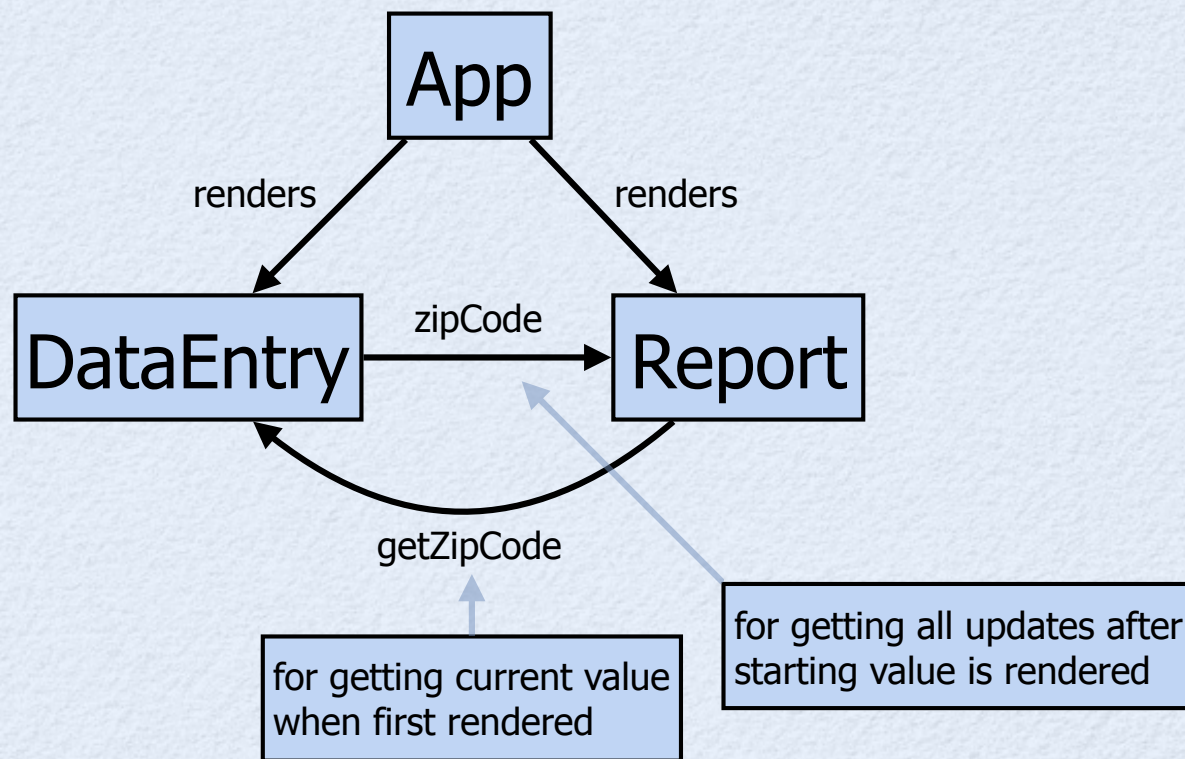
# Event Bus Example

- All **Vue** objects support the methods **\$emit**, **\$on**, **\$once**, and **\$off**
- **\$once** can be used in place of **\$on** to only listen to next occurrence of a given event
- Create a **Vue** object and export it so all components that wish to publish or subscribe to events can do so using this object

```
import Vue from 'vue';           event.js
export const EventBus = new Vue();
```

# Event Bus - App.vue

- This component renders the **DataEntry** and **Report** components
- Provides a button to toggle whether **Report** component is rendered





# Event Bus - App.vue

```
<template>
  <div>
    <DataEntry />
    <Report v-if="showReport" />
    <button @click="showReport = !showReport">
      Toggle Report
    </button>
  </div>
</template>

<script>
  import DataEntry from './components/DataEntry';
  import Report from './components/Report';
  export default {
    name: 'App',
    components: {DataEntry, Report},
    data() {
      return {showReport: true};
    }
  };
</script>
```

could use `v-show` instead of `v-if`,  
but we want to exercise  
the `mounted` method in `Report`

# Event Bus - DataEntry.vue

- This component renders a number `<input>` for entering a zip code
- Emits `zipCode` event with value every time it changes
- Subscribes to `getZipCode` events and emits current value when one is received



# Event Bus - DataEntry.vue

```
<template>
  <div>
    <label for="zipCode">Zip</label>
    <input type="number" v-model="zipCode" />
  </div>
</template>

<script>
  import {EventBus} from '../event';
  export default {
    name: 'DataEntry',
    data() {
      return {zipCode: 0};
    },
    mounted() {
      // When value of zipCode is requested, emit it.
      EventBus.$on('getZipCode',
        () => EventBus.$emit('zipCode', this.zipCode));
    },
    watch: {
      // When value of zipCode changes, emit it.
      zipCode() {
        EventBus.$emit('zipCode', this.zipCode);
      }
    }
  };
</script>
```

# Event Bus - Report.vue

- This component displays current zip code
- **mounted** method
  - called every time an instance is rendered
  - subscribes to `zipCode` events to get value updates
  - emits a `getZipCode` event to get current value
- **destroyed** method
  - called when no longer rendered
  - unsubscribes from `zipCode` events



# Event Bus - Report.vue

```
<template>
  <div>The zip code is now {{zipCode}}.</div>
</template>

<script>
  import {EventBus} from '../event';
  export default {
    name: 'Report',
    data() {
      return {
        listener: null,
        zipCode: 0
      };
    },
    mounted() {
      // Subscribe to "zipCode" event.
      this.listener = zipCode => (this.zipCode = zipCode);
      EventBus.$on('zipCode', this.listener);

      // Request current value.
      EventBus.$emit('getZipCode');
    },
    destroyed() {
      // Unsubscribe this listener from "zipCode" event.
      EventBus.$off('zipCode', this.listener);
    }
  };
</script>
```

# Vuex

- A more robust approach to state management that is more complex and more capable than Event Bus approach
- Covered later