Vue Best Practices:

- Naming Conventions: (mostly from style guide documentation and simple articles)
    1. Use kebab-case for event names (things that will be emitted/listened for) (link)
        a. vue converts them into lowercase anyways
    2. Prop names should use camelCase during declaration, but kebab-case in templates (link)
        a. camelCase is standard for JavaScript (used for declarations), while kebab-case is standard for html (used in templates)
    3. Component names should always be multi-word (except root "App" components) (link)
        a. ex. UserCard instead of Card
    4. Components that are only used once should begin with the prefix "The" (link)
        a. ex. "TheFooter.vue" if footer can only be used once
    5. Child components should include parent name as prefix (link)
        a. ex. "UserCardPhoto" where "Photo" component is used in "UserCard"

- Basic Tips: (also mostly from vue style documentation)
    1. Always add :key in v-for loops (link)
    2. Never use v-if on the same element as v-for (link)
    3. For single file components, use scoped ccs style (link)

- Computed vs Methods: (link)
    - Methods: function that's a property of an object, used to group common functionality, like handling form submissions or ajax requests

```
// in component
methods: {
  reverseMessage: function () {
    return this.message.split('').reverse().join('')
  }
}
```

    - Computed: doing something new to a set of data, like filtering or combining multiple things, stored in cache

```
computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
```

    1. Don't use methods when you could use computed – computed properties are stored in cache and only re-evaluated when dependencies change, while methods will run whenever a re-render happens (link)
        a. For the above example of reversing text, computed is better because it will only run when the dependencies change (if message stays the same, it won't have to re-run)

- Watchers: (link)

    - Watchers: used to perform asynchronous operations only when necessary; good for operations that respond to changing data and that could be expensive if using computed

```
<div id="watch-example">
  <p>
    Ask a yes/no question:
    <input v-model="question">
  </p>
  <p>{{ answer }}</p>
</div>
```

    - Using {immediate: true}: triggers a watcher to be called immediately upon running (link) (example)

```
vm.$watch('a', callback, {
  immediate: true
})
// `callback` is fired immediately with current value of `a`
```

- Keep Alive: (link)
    1. Wrap component tag with keep alive to cache an instance of a component, so it only needs to be mounted once

```
<!-- Inactive components will be cached! -->
<keep-alive>
  <component v-bind:is="currentTabComponent"></component>
</keep-alive>
```