Vue.js five common mistakes for beginners

Gihyo Joshua Jang - KossLab

Contents

Target Audience

- Web Developer who knows the basic concepts and syntax of Vue.js
- Frontend Developer who just started a Vue.js project
- Web Developer who is currently working on Vue.js project
- Frontend Developer who wants to know the features of Vue.js

Target Audience

- Web Developer who knows the basic concepts and syntax of Vue.js
- Frontend Developer who just started a Vue.js project
- Web Developer who is currently working on Vue.js project
- Frontend Developer who wants to know the features of Vue.js

This is not an introduction speech for Vue.js

Presentation Purpose

help you write the better and maintainable Vue.js code

All the slides and code are here below the Github link

https://github.com/joshua1988/vue-five-common-mistakes

Profile

- Web Developer, POSCO ICT
- Opensource Contributor, Kosslab
- Bootcamp Instructor in Fastcampus, 300
- Online Courses on Inflearn, 4500
- Do it! Vue.js beginning, Author, #1
- Corporate classes for Naver, ebay, SK

CAPTAIN P PANGYO

Profile

- Web Developer, POSCO ICT
- Opensource Contributor, Kosslab
- Bootcamp Instructor in Fastcampus, 300
- Online Courses on Inflearn, 4500
- Do it! Vue.js beginning, Author, #1
- Corporate classes for Naver, ebay, SK

"Fast learner who likes to share knowledge"

CAPTAIN P PANGYO

1. Reactivity

Why my page doesn't render?

What is Reactivity

What is Reactivity

- Re-render the page when data gets changed

What is Reactivity

- Re-render the page when data gets changed

```
var vm = new Vue({
  data: {
    count: 0
  }
});
vm.count += 1; // when counter changes, the page re-renders
```

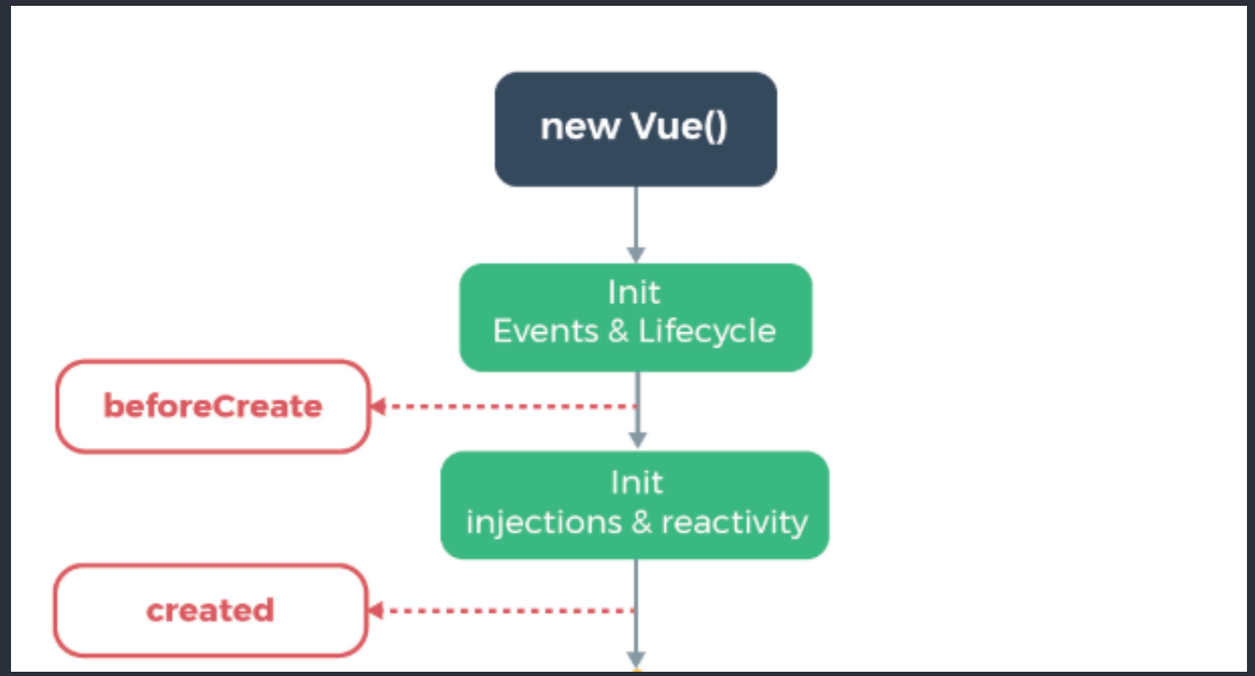
When does Reactivity set?

When does Reactivity set?

When Vue instance has been initialized

When does Reactivity set?

When Vue instance has been initialized



What to know about Reactivity

What to know about Reactivity

- The data that doesn't exist in instance initialization isn't reactive

What to know about Reactivity

- The data that doesn't exist in instance initialization isn't reactive

```
var vm = new Vue({
  data: {
    user: {
      name: 'Captain'
    }
  }
});
vm.user.age += 1; // even if age changes, it doesn't re-render
```

What to know about Reactivity

- The data that doesn't exist in instance initialization isn't reactive

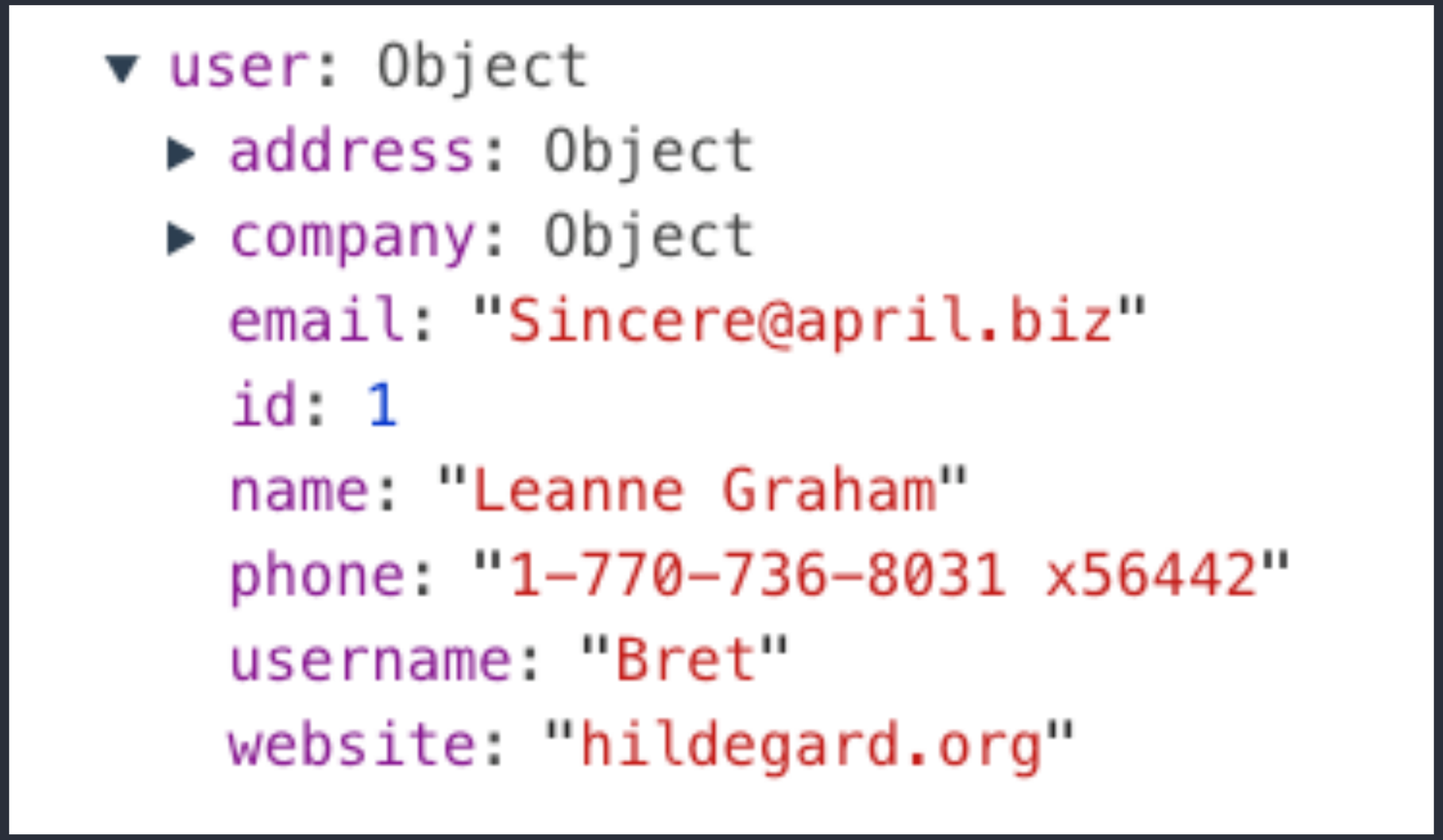
```
var vm = new Vue({
  data: {
    user: {
      name: 'Captain'
    }
  }
});
vm.user.age += 1; // even if age changes, it doesn't re-render
```

First common mistake in Reactivity

handling the states that are only needed in UI

Second common mistake in Reactivity

adding an arbitrary data to the fetched date from Backend



[Tip] treat the state the same way

```
state: {
  user: { name: 'Captain' }
},
mutations: {
  // age property wasn't initialized
  setUserAge: function(state) { state.user.age = 23; }, // X
  // dynamic property addition or deletion doesn't work
  deleteName: function(state) { delete state.user.name; } // X
}
```

Better Vue 3

Object.defineProperty() => Proxy

```
var obj = {};

// Vue 2.x
Object.defineProperty(obj, 'str', { .. });

// Vue 3.x
new Proxy(obj, { .. });
```

2. DOM Manipulation

Drop the old habit

(Old habit) DOM manipulation with Document API

(Old habit) DOM manipulation with Document API

- Accessing the certain element to manipulate

```
// Vanilla Javascript
document.querySelector('#app');

// jQuery
$('#app');
```

(Old habit) DOM manipulation with user events

- DOM manipulation based on user inputs

```
// find the button element
var btn = document.querySelector('#btn');

// find the closest certain element based on the clicked button
btn.addEventListener('click', function(event) {
  event.target.closest('.tag1').remove();
});
```

(Vue way) DOM manipulation with ref

(Vue way) DOM manipulation with ref

- reference property to get the element information

```
<!-- adding ref attribute -->
<div ref="hello">Hello Ref</div>


this.$refs.hello; // div element information
```

(Vue way) DOM manipulation with directives

- utilize the DOM information that directives provide

```
<ul>
  <li v-for="(item, index) in items">
    <span v-bind:id="index">{{ item }}</span>
  </li>
</ul>
```

Common mistake in DOM manipulation

```
<ul>
  <li @click="removeItem">
    <span>메뉴 1</span>
    <div class="child hide">메뉴 설명</div>
  </li>
  <li @click="removeItem">
    <span>메뉴 2</span>
    <div class="child hide">메뉴 설명</div>
  </li>
  ...
</ul>
```

3. Instance Lifecycle

How much do I know about the lifecycle?

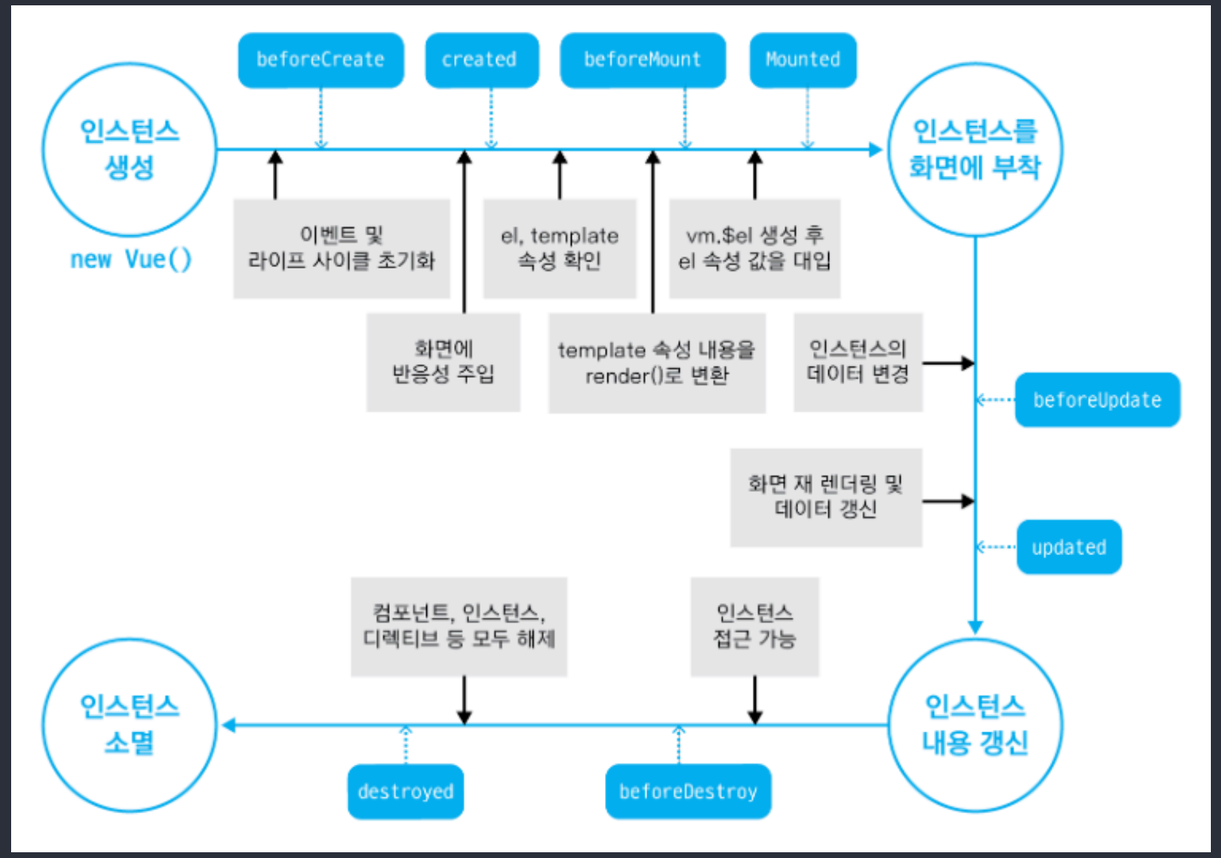
What is Instance Lifecycle?

What is Instance Lifecycle?

Steps to walk through when creating an instance

What is Instance Lifecycle?

Steps to walk through when creating an instance



35 — Image reference: Do it! Vue.js 입문(이지스퍼블리싱)

Vue.js Template Property

- A property to present what the component looks like

```
// instance options
new Vue({
  data: { str: 'Hello World' },
  template: '<div>{{ str }}</div>'
});

<!-- single file component -->
<template>
  <div>{{ str }}</div>
</template>
```

Vue.js Template Property

- A property to present what the component looks like

```
// instance options
new Vue({
  data: { str: 'Hello World' },
  template: '<div>{{ str }}</div>'
});

<!-- single file component -->
<template>
  <div>{{ str }}</div>
</template>
```

Vue.js Template Property

- A property to present what the component looks like

```
// instance options
new Vue({
  data: { str: 'Hello World' },
  template: '<div>{{ str }}</div>'
});

<!-- single file component -->
<template>
  <div>{{ str }}</div>
</template>
```

Inside the template property

Inside the template property

It is a Virtual DOM not actual DOM element.

Inside the template property

It is a Virtual DOM not actual DOM element.

```
<!-- what we type -->
<template>
  <div>{{ str }}</div>
</template>
```

Inside the template property

It is a Virtual DOM not actual DOM element.

```
<!-- what we type -->
<template>
  <div>{{ str }}</div>
</template>

// how library compiles
function render() {
  with(this) {
    return _c('div', [_v(_s(str))]);
  }
}
```

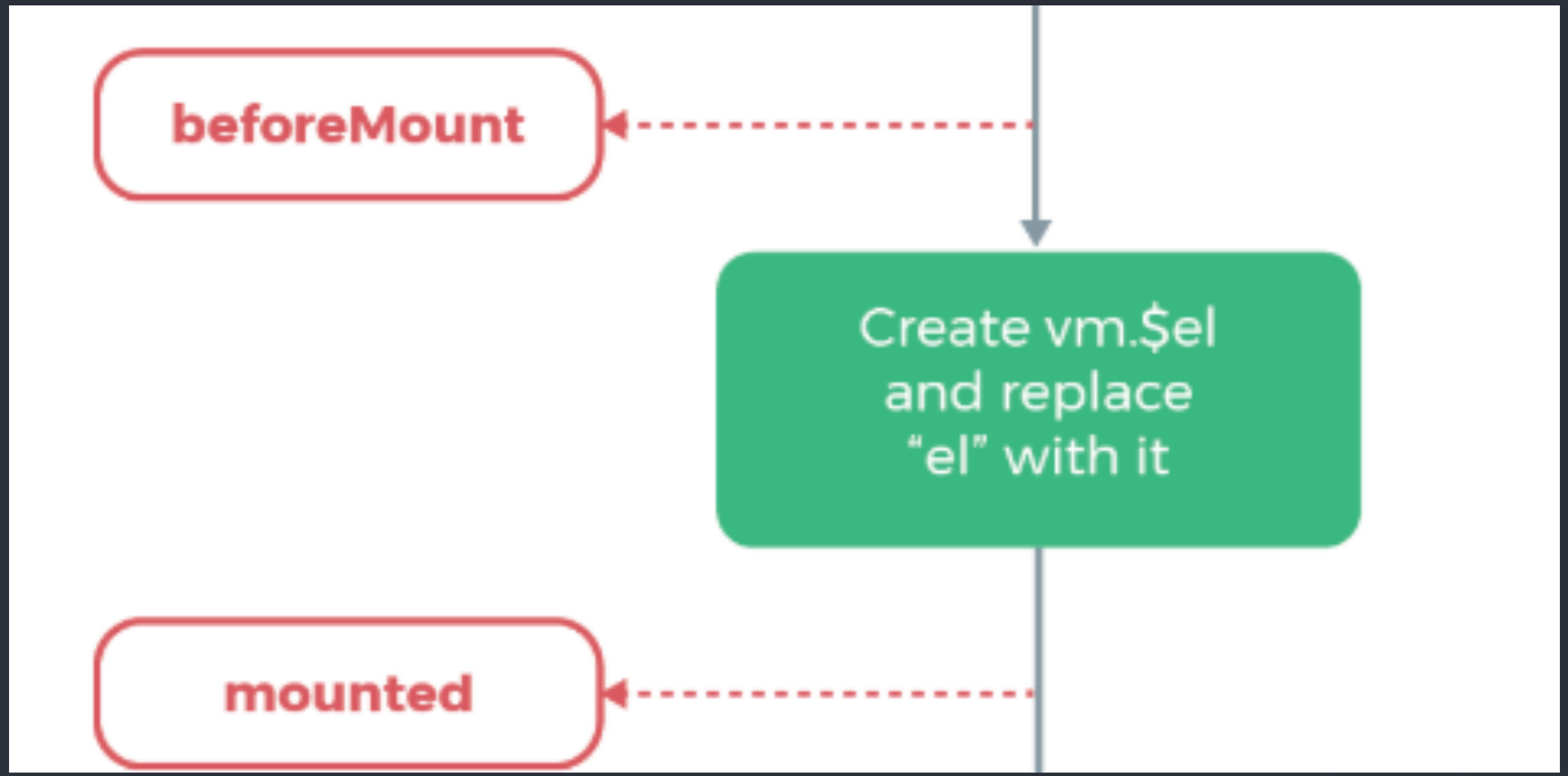
When can we access the actual DOM?

When can we access the actual DOM?

After the instance has been mounted - `mounted`

When can we access the actual DOM?

After the instance has been **mounted** - `mounted`



First common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    var ctx = document.querySelector('#myChart');
    new Chart(ctx, chartOptions);
  }
});
```

First common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    var ctx = document.querySelector('#myChart'); // null
    new Chart(ctx, chartOptions);
  }
});
```

First common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    var ctx = document.querySelector('#myChart'); // null
    new Chart(ctx, chartOptions);
  }
});
```

Second common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    this.$nextTick(function() {
      var ctx = document.querySelector('#myChart');
      new Chart(ctx, chartOptions);
    })
  }
});
```

Second common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    this.$nextTick(function() { // increase code complecity
      var ctx = document.querySelector('#myChart');
      new Chart(ctx, chartOptions);
    })
  }
});
```

Second common mistake in Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  created: function() {
    this.$nextTick(function() { // increase code complecity
      var ctx = document.querySelector('#myChart');
      new Chart(ctx, chartOptions);
    })
  }
});
```

Code that understands Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  mounted: function() {
    var ctx = document.querySelector('#myChart');
    new Chart(ctx, chartOptions);
  }
});
```

Code that understands Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  mounted: function() {
    var ctx = document.querySelector('#myChart'); // <canvas>
    new Chart(ctx, chartOptions);
  }
});
```

Code that understands Instance Lifecycle

```
<!-- template -->
<template>
  <canvas id="myChart"></canvas>
</template>

// instance options
new Vue({
  mounted: function() {
    var ctx = document.querySelector('#myChart'); // <canvas>
    new Chart(ctx, chartOptions);
  }
});
```

4. ref property

tricky ref

What is ref?

What is ref?

- access to a certain DOM element or Component

What is ref?

- access to a certain DOM element or Component
- gain the DOM information when using on DOM element

What is ref?

- access to a certain DOM element or Component
- gain the DOM information when using on DOM element
- gain the Component Instance when using on Component

What is ref?

- access to a certain DOM element or Component
- gain the DOM information when using on DOM element
- gain the Component Instance when using on Component
- provide Array type data when using with `v-for`

What is ref?

- access to a certain DOM element or Component
- gain the DOM information when using on DOM element
- gain the Component Instance when using on Component
- provide Array type data when using with `v-for`

Property to manipulate a certain UI element

First caution for ref

- Only available after template compilation

First caution for ref

- Only available after template compilation
- Can access the UI element information from `mounted`

First caution for ref

- Only available after template compilation
- Can access the UI element information from `mounted`

```
<p ref="pTag">Hello</p>

created: function() {
  this.$refs.pTag; // undefined
},
mounted: function() {
  this.$refs.pTag; // <p>Hello</p>
}
```

Second caution for ref

- Can't access the element information until it renders on the page when using with `v-if`

Second caution for ref

- Can't access the element information until it renders on the page when using with `v-if`

```
<div v-if="isUser">
  <p ref="w3c">W3C</p>
</div>

new Vue({
  data: { isUser: false },
  mounted: function() {
    this.$refs.w3c; // undefined
  },
});
```

Second caution for ref

- Can't access the element information until it renders on the page when using with `v-if`

```
<div v-if="isUser">
  <p ref="w3c">W3C</p>
</div>

new Vue({
  data: { isUser: false },
  mounted: function() {
    this.$refs.w3c; // undefined
  },
});
```

Second caution for ref

- Can't access the element information until it renders on the page when using with `v-if`

```
<div v-if="isUser">
  <p ref="w3c">W3C</p>
</div>

new Vue({
  data: { isUser: true },
  mounted: function() {
    this.$refs.w3c; // <p>W3C</p>
  },
});
```

Third caution for ref

- Can manipulate the child component but don't abuse it

```
<div id="app">
  <TodoList ref="list"></TodoList>
</div>

new Vue({
  el: '#app',
  methods: { // unnecessary access through ref to trigger the method
    fetchItems: function() { this.$refs.list.fetchTodos(); }
  }
});
```

Third caution for ref

- Can manipulate the child component but don't abuse it

```
<div id="app">
  <TodoList></TodoList>
</div>

// Use lifecycle hook instead
var TodoList = {
  methods: {
    fetchTodos: function() { .. }
  },
  created: function() { this.fetchTodos(); }
};
```

5. computed property

The last puzzle of intuitive template

What is computed?

- A property to make the template clean and simple

```
<p>{{ 'hello' + str + '!!' }}</p> <!-- template expression -->
<p>{{ greetingStr }}</p> <!-- computed -->

new Vue({
  data: { str: 'world' },
  computed: {
    greetingStr: function() {
      return 'hello' + this.str + '!!';
    }
  }
});
```

What is computed?

- A property to make the template clean and simple

```
<p>{{ 'hello' + str + '!!' }}</p> <!-- template expression -->
<p>{{ greetingStr }}</p> <!-- computed -->

new Vue({
  data: { str: 'world' },
  computed: {
    greetingStr: function() {
      return 'hello' + this.str + '!!';
    }
  }
});
```

What is computed?

- A property to make the template clean and simple

```
<p>{{ 'hello' + str + '!!' }}</p> <!-- template expression -->
<p>{{ greetingStr }}</p> <!-- computed -->

new Vue({
  data: { str: 'world' },
  computed: {
    greetingStr: function() {
      return 'hello' + this.str + '!!';
    }
  }
});
```

What is computed?

- A property to make the template clean and simple

```
<p>{{ 'hello' + str + '!!' }}</p> <!-- template expression -->
<p>{{ greetingStr }}</p> <!-- computed -->

new Vue({
  data: { str: 'world' },
  computed: {
    greetingStr: function() {
      return 'hello' + this.str + '!!';
    }
  }
})
```

What is computed?

- A property to make the template clean and simple

```
<p>{{ 'hello' + str + '!!' }}</p> <!-- template expression -->
<p>{{ greetingStr }}</p> <!-- computed -->

new Vue({
  data: { str: 'world' },
  computed: {
    greetingStr: function() {
      return 'hello' + this.str + '!!';
    }
  }
})
```

Use case #1

- Add or Delete HTML Classes

```
<li v-bind:class="{ disabled: isLastPage }"></li>
```

```
computed: {
  isLastPage: function() {
    var lastPageCondition =
      this.paginationInfo.current_page >= this.paginationInfo.last_page;
    var nothingFetched = Object.keys(this.paginationInfo).length === 0;
    return lastPageCondition || nothingFetched;
  }
}
```

Use case #1

- Add or Delete HTML Classes

```
<li v-bind:class="listItemClass"></li>

computed: {
  listItemClass: function() {
    // ..
  }
}
```

Use case #2

- Access the properties in Vuex

```
<div>
  <p>{{ this.$store.state.module1.str }}</p>
  <p>{{ module1Str }}</p>
</div>

new Vue({
  computed: {
    module1Str: function() {
      return this.$store.state.module1.str;
    }
  }
});
```

Use case #2

- Access the properties in Vuex

```
<div>
  <p>{{ this.$store.state.module1.str }}</p>
  <p>{{ module1Str }}</p>
</div>

new Vue({
  computed: {
    module1Str: function() {
      return this.$store.state.module1.str;
    }
  }
});
```

Use case #2

- Access the properties in Vuex

```
<div>
  <p>{{ this.$store.state.module1.str }}</p>
  <p>{{ module1Str }}</p>
</div>

new Vue({
  computed: {
    module1Str: function() {
      return this.$store.state.module1.str;
    }
  }
});
```

Use case #2

- Access the properties in Vuex

```
<div>
  <p>{{ this.$store.state.module1.str }}</p>
  <p>{{ module1Str }}</p>
</div>

new Vue({
  computed: {
    module1Str: function() {
      return this.$store.state.module1.str;
    }
  }
});
```

Use case #2

- Access the properties in Vuex

```
<div>
  <p>{{ this.$store.state.module1.str }}</p>
  <p>{{ module1Str }}</p>
</div>

new Vue({
  computed: {
    module1Str: function() {
      return this.$store.state.module1.str;
    }
  }
});
```

Use case #3

- Using translation library like Vue i18n

```
<p>{{ 'userPage.common.filter.input.label' }}</p>
<p>{{ inputLabel }}</p>

computed: {
  inputLabel: function() {
    return $t('userPage.common.filter.input.label');
  }
}
```

Use case #3

- Using translation library like Vue i18n

```
<p>{{ 'userPage.common.filter.input.label' }}</p>
<p>{{ inputLabel }}</p>

computed: {
  inputLabel: function() {
    return $t('userPage.common.filter.input.label');
  }
}
```

Use case #3

- Using translation library like Vue i18n

```
<p>{{ 'userPage.common.filter.input.label' }}</p>
<p>{{ inputLabel }}</p>

computed: {
  inputLabel: function() {
    return $t('userPage.common.filter.input.label');
  }
}
```

Use case #3

- Using translation library like Vue i18n

```
<p>{{ 'userPage.common.filter.input.label' }}</p>
<p>{{ inputLabel }}</p>

computed: {
  inputLabel: function() {
    return $t('userPage.common.filter.input.label');
  }
}
```

Thank You 😄

jangkeehyo@gmail.com