Vue.js

Filters 필터

필터

Vue.js에서는 일반 텍스트 서식을 적용할 때 사용할 수 있는 필터를 정의할 수 있습니다. 필터는 **Mustache 보간과** `v-bind` **표현식** 두 곳에서 사용할 수 있습니다. 필터는 JavaScript 표현식의 끝에 추가해야 하며 "파이프" 기호로 표시됩니다.

```HTML
<!-- Mustaches 사용시 -->
{{ message | capitalize }}

<!-- v-bind 사용시 -->
<div v-bind:id="rawId | formatId"></div>
```

> Vue 2.x 필터는 Mustache 보간과 `v-bind` 표현식 (2.1.0 이후 지원)에서 사용할 수 있습니다. 왜냐하면 필터는 주로 텍스트 변환 목적으로 설계되었기 때문입니다. 다른 디렉티브에서 보다 복잡한 데이터 변환을 하는 경우 계산된 속성을 사용해야 합니다.

https://kr.vuejs.org/v2/guide/syntax.html#필터

필터 함수는 항상 표현식의 값을 첫번째 전달 인자로 받습니다.

```JS
new Vue({
  // ...
  filters: {
    capitalize: function (value) {
      if (!value) return ''
      value = value.toString()
      return value.charAt(0).toUpperCase() + value.slice(1)
    }
  }
})
```

1번째 전달인자

필터는 체이닝 가능합니다.

```HTML
{{ message | filterA | filterB }}
```

필터는 JavaScript 함수이므로 전달인자를 사용할 수 있습니다.

```HTML
{{ message | filterA('arg1', arg2) }}
```

2번째 전달인자

3번째 전달인자

여기서, 일반 문자열 `'arg1'` 은 두번째 인자로 필터에 전달되고 `'arg2'` 는 필터의 세번째 인자로 전달됩니다.

https://kr.vuejs.org/v2/guide/syntax.html#필터

Vue.filter(id, [definition])

- 전달인자:
  - `{string} id`
  - `{Function} [definition]`
- 사용방법:

  전역 필터를 등록하거나 검색합니다.

```JS
// 등록
Vue.filter('my-filter', function (value) {
  // 처리된 값을 반환합니다
})

// getter, 필터가 등록된 경우 반환합니다
var myFilter = Vue.filter('my-filter')
```

https://kr.vuejs.org/v2/api/#Vue-filter

```
<time :datetime="new Date() | date">{{ new Date() | date('년월일') }}</time>
```

```
Vue.filter('date', function(date, format, divider){
  if ( !date instanceof Date ) {
    throw 'Date 객체를 생성해서 전달해야 합니다.'
  }
  var yyyy = date.getFullYear();
  var mm   = date.getMonth();
  var dd   = date.getDate();
  mm = mm < 10 ? '0'+mm : mm;
  dd = dd < 10 ? '0'+dd : dd;
  divider = divider || '-';
  switch(format) {
    default:
    case 'yyyy-mm-dd': return yyyy + divider + mm + divider + dd;
    case 'mm-dd-yyyy': return mm + divider + dd + divider + yyyy;
    case '년월일': return yyyy +'년 ' + mm + '월 ' + dd + '일';
    case '월일년': return mm + '월 ' + dd + '일, ' + yyyy +'년';
  }
});
```


```
<p class="currency">{{ 1283200 | currency }}</p>

<p class="currency dollar">{{ 1283200 | currency('$', 'front') }}</p>
```

```
Vue.filter('currency', function(value, sign, position) {
  var int, decimal;
  value = String(value).split('.');
  int = value[0];
  decimal = value[1];
  sign = sign || '원';
  position = position || 'back';
  int = int.split('').reverse();
  for ( var i=0, l=int.length; i<l; i+=3  ) {
    i > 0 && int.splice(i, 1, int[i]+',');
  }
  value = int.reverse().join('') + (decimal ? '.'+decimal : '');
  return position === 'back' ? value + sign : sign + value;
});
```

```
new Vue({
  el : '.demo',
  filters: {
    uppercase: function(value) {…
    },
    lowerCase: function(value) {…
    },
    capitalize: function(value) {
      return value.toString()
        .replace(/(\s|_|-)(.)/g, function($1){ return $1.toUpperCase(); })
        .replace(/(\s|_|-)/g, ' ');
    },
    titleCase: function(value) {
      return value.toString()
        .replace(/(\s|-|_)(.)/g, function($1) { return $1.toUpperCase(); })
        .replace(/(\s|-|_)/g, '')
        .replace(/^(.)/, function($1) { return $1.toUpperCase(); });
    },
    camelCase: function(value) {
      return value.toString()
        .replace(/(\s|-|_)(.)/g, function($1) { return $1.toUpperCase(); })
        .replace(/(\s|-|_)/g, '')
        .replace(/^(.)/, function($1) { return $1.toLowerCase(); });
    },
    snakeCase: function(value) {
      return value.toString().replace(/(\s|-)/g, '_').toLowerCase();
    },
    kebabCase: function(value) {
      return value.toString().replace(/(\s|_)/g, '-').toLowerCase();
    }
  }
});
```

뷰 인스턴스 객체에 국한된 형태로 필터 정의하는 방법은 filters 속성을 사용하여 내부에 정의하는 것이다.

Vue 필터(Filters) 예시

vue.filters.js — Raw

```
/*! vue.filter.js @ yamoo9.net, 2017 */
;(function(global, Vue){
  'use strict';

  Vue.filter('date', function(date, format, divider){
    if ( !date instanceof Date ) {
      throw 'Date 객체를 생성해서 전달해야 합니다.'
    }
    var yyyy = date.getFullYear();
    var mm   = date.getMonth();
    var dd   = date.getDate();
    mm = mm < 10 ? '0'+mm : mm;
    dd = dd < 10 ? '0'+dd : dd;
    divider = divider || '-';
    switch(format) {
      default:
      case 'yyyy-mm-dd': return yyyy + divider + mm + divider + dd;
      case 'mm-dd-yyyy': return mm + divider + dd + divider + yyyy;
      case '년월일': return yyyy +'년 ' + mm + '월 ' + dd + '일';
      case '월일년': return mm + '월 ' + dd + '일, ' + yyyy +'년';
    }
```


https://gist.github.com/yamoo9/6f80672681834c2b06eae2c8dc2ac292

디렉티브 Directive

Directives

Vue는 코어에 포함된 기본 디렉티브 세트(v-model과 v-show) 외에도 사용자 정의 디렉티브를 등록할 수 있습니다. Vue 2.0에서 코드 재사용 및 추상화의 기본 형식은 컴포넌트 입니다.

그러나 **일반 엘리먼트에 하위 수준의 DOM 액세스가 필요한 경우**가 있을 수 있으며 이 경우 **사용자 지정 디렉티브가 여전히 유용할 수 있습니다.**

코드 재사용/추상화 기본 형식은 컴포넌트이지만, DOM 접근이 필요한 경우 사용자 정의 디렉티브가 유용할 수 있다.

https://kr.vuejs.org/v2/guide/custom-directive.html

Directives

글로벌 Directive 정의

```
// 전역 사용자 정의 디렉티브 v-focus 등록
Vue.directive('focus', {
  // 바인딩 된 엘리먼트가 DOM에 삽입되었을 때...
  inserted: function (el) {
    // 엘리먼트에 포커스를 줍니다
    el.focus()
  }
})
```

지시어를 로컬로 등록하기 위해서 컴포넌트는 `directives` 옵션을 허용합니다.

로컬 Directive 정의

```
directives: {
  focus: {
    // 디렉티브 정의
  }
}
```

그런 다음 템플릿에서 다음과 같이 모든 요소에서 새로운 `v-focus` 속성을 사용할 수 있습니다.

HTML

```
<input v-focus>
```

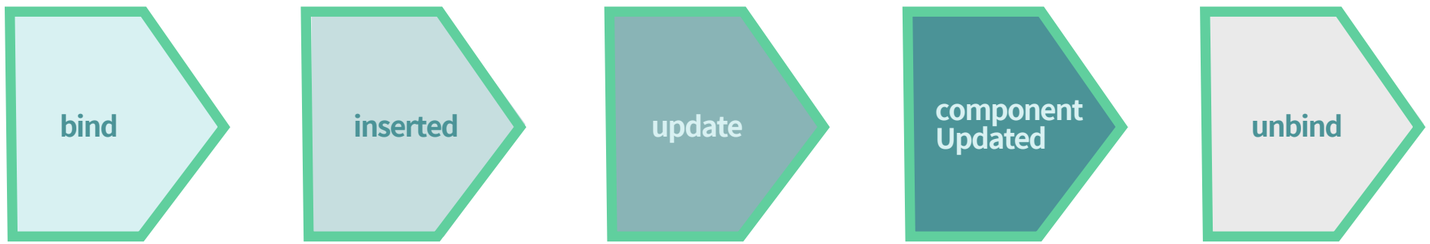
Directives

```
Vue.directive('hilighting', {
  // 바인드 처리 시, 콜백함수 실행
  bind: function(el){
    el.style.background = '#ff0';
  }
});
```

```
<p v-hilighting>
  <code>v-hilighting</code>
  사용자 정의 디렉티브가 설정된 요소는 배경색이 노란색으로 표시 처리됩니다.
</p>
```

훅 함수

디렉티브 정의 객체는 여러가지 훅 함수를 제공할 수 있습니다.(모두 선택사항입니다.)



- `bind` : 디렉티브가 처음 엘리먼트에 바인딩 될 때 한번만 호출 됩니다. 이곳에서 일회성 설정을 할 수 있습니다.
- `inserted` : 바인딩 된 엘리먼트가 부모 노드에 삽입 되었을 때 호출 됩니다. (이것은 부모 노드 존재를 보장하며 반드시 document 내에 있는 것은 아닙니다.)
- `update` : 포함하는 컴포넌트가 업데이트 된 후 호출됩니다. **그러나 자식이 업데이트 되기 전일 가능성이 있습니다** 디렉티브의 값은 변경되었거나 변경되지 않았을 수 있지만 바인딩의 현재 값과 이전 값을 비교하여 불필요한 업데이트를 건너 뛸 수 있습니다. (아래의 훅 전달인자를 참조하세요)
- `componentUpdated` : 포함하고 있는 컴포넌트와 **그 자식들** 이 업데이트 된 후에 호출됩니다.
- `unbind` : 디렉티브가 엘리먼트로부터 언바인딩된 경우에만 한번 호출됩니다.

https://kr.vuejs.org/v2/guide/custom-directive.html#훅-함수

Directives

```
Vue.directive('hook-function', {
  // 디렉티브가 요소에 첫 바인딩 시에 1회 실행
  bind: function() { console.log('bind'); },
  // 바인딩된 요소가 부모 노드에 연결되었을 때 1회 실행
  inserted: function() { console.log('inserted'); },
  // 포함하는 컴포넌트 업데이트 발생 시마다 실행
  update: function() { console.log('update'); },
  // 포함하는 컴포넌트(자식 포함) 업데이트 발생 시마다 실행
  componentUpdated: function() { console.log('componentUpdated'); },
  // 디렉티브가 요소에서 제거될 경우에 1회 실행
  unbind: function() {console.log('unbind');}
});
```

디렉티브 훅 전달인자

디렉티브 훅은 다음과 인자들을 전달합니다.

- **el**: 디렉티브가 바인딩 되는 엘리먼트 입니다. 이것은 DOM을 직접 조작하는데 사용할 수 있습니다.
- **binding**: 다음 속성을 포함하는 객체입니다.
  - **name**: `v-` 접두어가 없는 디렉티브의 이름입니다.
  - **value**: 디렉티브에 전달된 값입니다. 예를 들어 `v-my-directive="1 + 1"` 에서 값은 `2` 가 됩니다.
  - **oldValue**: 이전 값은 `update` 와 `componentUpdated` 에서만 가능합니다. 값이 변경되었는지 여부에 관계없이 사용할 수 있습니다.

> ! `el` 이외에 이 인자들을 읽기 전용으로 취급하고 절대 수정하면 안됩니다. 훅에서 정보를 공유해야 하는 경우 엘리먼트의 데이터셋을 통해 정보를 공유하는 것이 좋습니다.

https://kr.vuejs.org/v2/guide/custom-directive.html#디렉티브-훅-전달인자

Directives

```
Vue.directive('hilighting', {
  // 바인드 처리 시, 콜백함수 실행
  bind: function(el, binding){
    // 사용자가 입력한 값
    var value = binding.value || '#ff0';
    // 사용자가 입력한 값으로 설정
    el.style.background = value;
  },
  // 디렉티브 값 업데이트 시에 콜백함수 처리
  update: function(el, binding) {
    var value = binding.value;
    var old_value = binding.oldValue;
    console.log('value: %s, old_value: %s', value, old_value);
    el.style.background = value;
  }
});
```

디렉티브 훅 전달인자

디렉티브 훅은 다음과 인자들을 전달합니다.

- el: 디렉티브가 바인딩 되는 엘리먼트 입니다. 이것은 DOM을 직접 조작하는데 사용할 수 있습니다.
- binding: 다음 속성을 포함하는 객체입니다.
  - name: `v-` 접두어가 없는 디렉티브의 이름입니다.
  - value: 디렉티브에 전달된 값입니다. 예를 들어 `v-my-directive="1 + 1"` 에서 값은 `2` 가 됩니다.
  - oldValue: 이전 값은 `update` 와 `componentUpdated` 에서만 가능합니다. 값이 변경되었는지 여부에 관계없이 사용할 수 있습니다.
  - expression: 문자열로 바인딩 된 표현식 입니다. `v-my-directive="1 + 1"` 에서 표현식은 `"1 + 1"` 입니다.
  - arg: 전달인자가 있으면 디렉티브에 전달됩니다. 예를 들어 `v-my-directive: foo` 에서 arg는 `"foo"` 가 됩니다.
  - modifiers: 수정자를 포함한 오브젝트가 존재하는 경우, 예를 들어 `v-my-directive.foo.bar` 에서 수정자 객체는 `{ foo: true, bar: true }` 가 됩니다.
- vnode: Vue 컴파일러가 생성한 가상의 노드 입니다. 자세한 내용은 VNode API를 참조하십시오.
- oldVnode: 이전의 가상 노드 입니다. 오직 `update` 와 `componentUpdated` 훅에서만 사용 가능합니다.

https://kr.vuejs.org/v2/guide/custom-directive.html#디렉티브-훅-전달인자

Directives

```
<p v-hook-function  v-hilighting:background="'#433' + 'ab3'">
  <code>v-hilighting</code>
  사용자 정의 디렉티브가 설정된 요소는 배경색이 노란색으로 표시 처리됩니다.
</p>

Vue.directive('hilighting', {

  bind: function(el, binding){
    var value = binding.value || '#ff0';
    var expression = binding.expression;
    var arg = binding.arg;
    console.log('value: %s, expression: %s, arg: %s', value, expression, arg);
    el.style[arg] = value;
  },

  update: function(el, binding) {
    var arg = binding.arg;
    el.style[arg] = binding.value;
  }

});
```

```
value: #433ab3, expression: '#433' + 'ab3', arg: background
```

Directives

```
<p v-hook-function  v-hilighting.delay.animate>
  <code>v-hilighting</code>
  사용자 정의 디렉티브가 설정된 요소는 배경색이 노란색으로 표시 처리됩니다.
</p>

Vue.directive('hilighting', {

  bind: function(el, binding){
    var arg = binding.arg || 'background';
    var delay = 0;
    // 수식어 객체
    var modifiers = binding.modifiers;
    // 지연 설정이 true일 경우, 처리
    modifiers.delay && (delay = 1400);
    // 장면전환 설정이 true일 경우, 처리
    modifiers.animate && (el.style.transition = arg + ' 1s ease');
    global.setTimeout(function(){
      el.style[arg] = binding.value || '#ff0';
    }, delay);
  }

});
```

▶ *Object {delay: true, animate: true}*

사용자 정의 디렉티브

HTML

```
<div id="hook-arguments-example" v-demo:foo.a.b="message"></div>
```

훅 함수

훅 전달인자

JS

```
Vue.directive('demo', {
  bind: function (el, binding, vnode) {
    var s = JSON.stringify
    el.innerHTML =
      'name: '       + s(binding.name) + '<br>' +
      'value: '      + s(binding.value) + '<br>' +
      'expression: ' + s(binding.expression) + '<br>' +
      'argument: '   + s(binding.arg) + '<br>' +
      'modifiers: '  + s(binding.modifiers) + '<br>' +
      'vnode keys: ' + Object.keys(vnode).join(', ')
  }
})

new Vue({
  el: '#hook-arguments-example',
  data: {
    message: 'hello!'
  }
})
```

함수 약어

많은 경우에, `bind` 와 `update` 에서 같은 동작이 필요할 수 있습니다. 그러나 다른 훅은 신경 쓸 필요가 없습니다. 그 예로

```
Vue.directive('color-swatch', function (el, binding) {
  el.style.backgroundColor = binding.value
})
```

JS

bind, update 훅 함수

https://kr.vuejs.org/v2/guide/custom-directive.html#디렉티브-훅-전달인자

Directives

```
// bind, update 동시 적용이 필요한 디렉티브
Vue.directive('hilighting', function(el, binding, vnode){
    var arg = binding.arg || 'background';
    var delay = 0;
    var modifiers = binding.modifiers;
    modifiers.delay && (delay = 1400);
    modifiers.animate && (el.style.transition = arg + ' 1s ease');
    global.setTimeout(function(){
      el.style[arg] = binding.value || '#ff0';
    }, delay);
  }
);
```

객체 리터럴

디렉티브에 여러 값이 필요한 경우, JavaScript 객체 리터럴을 전달할 수도 있습니다. 디렉티브는 유효한 JavaScript 표현식을 사용할 수 있습니다.

객체 리터럴 표현식

HTML

```
<div v-demo="{ color: 'white', text: 'hello!' }"></div>
```

JS

```
Vue.directive('demo', function (el, binding) {
  console.log(binding.value.color) // => "white"
  console.log(binding.value.text)  // => "hello!"
})
```

객체 속성 접근법

https://kr.vuejs.org/v2/guide/custom-directive.html#디렉티브-훅-전달인자

Directives

```
Vue.directive('hilighting', function(el, binding, vnode){
    var arg = binding.arg || 'background';
    var delay = 0;
    var is_obj = false;
    var value = binding.value;
    var modifiers = binding.modifiers;
    modifiers.delay && (delay = 1400);
    modifiers.animate && (el.style.transition = arg + ' 1s ease');
    if ( !Array.isArray(value) && typeof value === 'object' ) {
      is_obj = true;
    }
    global.setTimeout(function(){
      if (!is_obj) {
        el.style[arg] = value || '#ff0';
      } else {
        for ( var key in value ) {
          if ( value.hasOwnProperty(key) ) {
            el.style[key] = value[key];
          }
        }
      }
    }, delay);
  }
);
```

Vue.directive(id, [definition])

- 전달인자:
  - `{string} id`
  - `{Function | Object} [definition]`
- 사용방법:

  전역 디렉티브를 등록하거나 검색합니다.

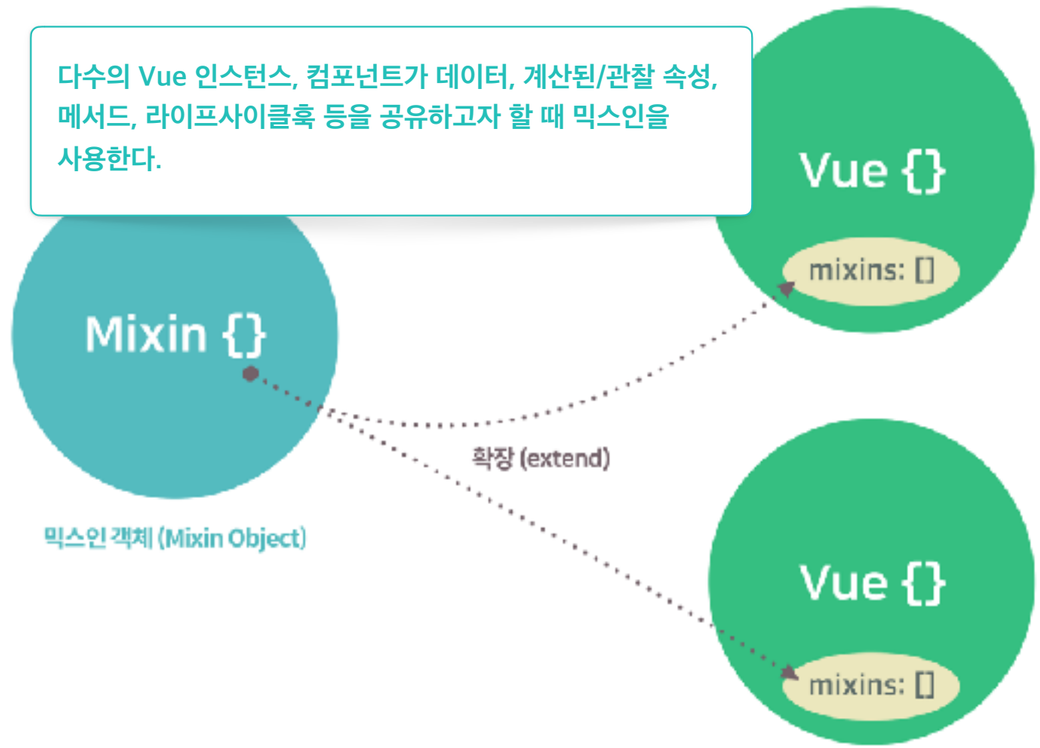
```JS
// 등록
Vue.directive('my-directive', {
  bind: function () {},
  inserted: function () {},
  update: function () {},
  componentUpdated: function () {},
  unbind: function () {}
})

// 등록 (간단한 함수 디렉티브)
Vue.directive('my-directive', function () {
  // `bind`와 `update`를 호출합니다.
})

// getter, 등록된 지시어의 경우 반환합니다.
var myDirective = Vue.directive('my-directive')
```


https://kr.vuejs.org/v2/api/#Vue-directive

믹스인 Mixins

Mixins

다수의 Vue 인스턴스, 컴포넌트가 데이터, 계산된/관찰 속성, 메서드, 라이프사이클훅 등을 공유하고자 할 때 믹스인을 사용한다.



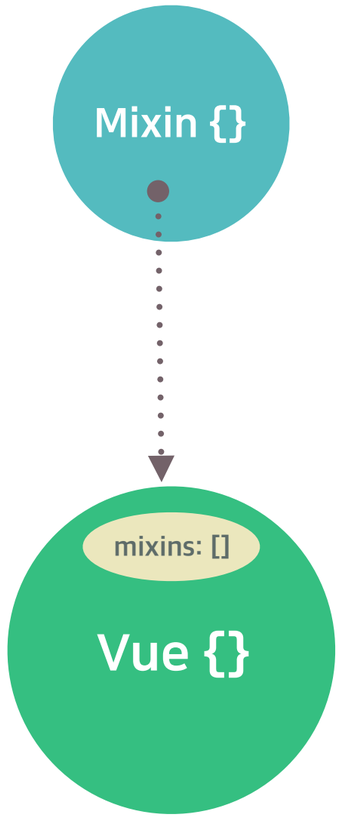
- 재사용 가능한 기능을 배포하는 유연한 방법
- Vue 컴포넌트 옵션을 모두 포함
- 컴포넌트가 믹스인을 사용할 경우, 모든 옵션이 컴포넌트에 혼합

https://kr.vuejs.org/v2/guide/mixins.html

Mixins

```
// mixin 객체 생성
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}

// mixin을 사용할 컴포넌트 정의
var Component = Vue.extend({
  mixins: [myMixin]
})

var component = new Component() // -> "hello from mixin!"
```

믹스인 객체는 일반적인 JavaScript 객체에 불과
(컴포넌트 옵션 모두 설정 가능)



Mixins

```
var mixinObj = {
  mounted: function() {
    this.hello();
  },
  methods: {
    hello: function() {
      console.log('Hello! Vue. 믹스인 호출', this.$el);
    }
  }
};

Vue.component('comp', {
  template: '<p>컴포넌트</p>',
  mixins: [mixinObj]
});

new Vue({
  el : '.demo',
  mixins: [mixinObj],
  data: {…
  }
});
```

믹스인 객체는 일반적인 JavaScript 객체에 불과
(컴포넌트 옵션 모두 설정 가능)

Mixins

- **라이프 사이클 훅은 병합 되어도 모두 실행**. (단, 믹스인 훅이 우선 실행)
- **메서드, 컴포넌트, 디렉티브는 병합 과정에서 덮어쓰기 됨**. (컴포넌트 우선)

```
var mixin = {
  created: function () {
    console.log('mixin hook called')
  }
}

new Vue({
  mixins: [mixin],
  created: function () {
    console.log('component hook called')
  }
})
```

믹스인, 컴포넌트 훅이 모두 실행

Mixins

```
var mixin = {
  methods: {
    foo: function () {
      console.log('foo')
    },
    conflicting: function () {
      console.log('from mixin')
    }
  }
}

var vm = new Vue({
  mixins: [mixin],
  methods: {
    bar: function () {
      console.log('bar')
    },
    conflicting: function () {
      console.log('from self')
    }
  }
})

vm.foo() // -> "foo"
vm.bar() // -> "bar"
vm.conflicting() // -> "from self"
```

메서드가 겹치는 경우, 컴포넌트 메서드로 덮어써진다.

Mixins

다수의 Vue 인스턴스, 컴포넌트가 데이터, 계산된/관찰 속성, 메서드, 라이프사이클훅 등을 공유하고자 할 때 믹스인을 사용한다.

```
var mixinTodoList = {
  props: {
    todoList: {
      type: Array,
      default: []
    }
  },
  data: function(){
    return {
      filter_text: ''
    }
  },
  computed: {
    filteredTodo: function() {
      var vm = this;
      return vm.todoList.filter(function(list){
        return list.match(vm.filter_text);
      });
    }
  }
};
```

상위 컴포넌트로부터
전달된 todoList 속성

사용자 입력
필터링 텍스트 데이터

계산된 속성 설정
filteredTodo

Mixins

- 전역 믹스인을 사용하여 컴포넌트를 확장할 경우, 모든 컴포넌트에 영향 (주의)
- 사용자 정의 옵션 처리 로직 주입에 활용

```
// `myOption` 사용자 정의 옵션을 위한 핸들러 주입
Vue.mixin({
  created: function () {
    var myOption = this.$options.myOption
    if (myOption) {
      console.log(myOption)
    }
  }
})

new Vue({
  myOption: 'hello!'
})
// -> "hello!"
```

created 훅은 모든 Vue 인스턴스, 컴포넌트에서 작동 (영향을 받음)

뷰 인스턴스, 컴포넌트에 사용자 정의 설정한 속성은 $options 를 통해 접근 가능

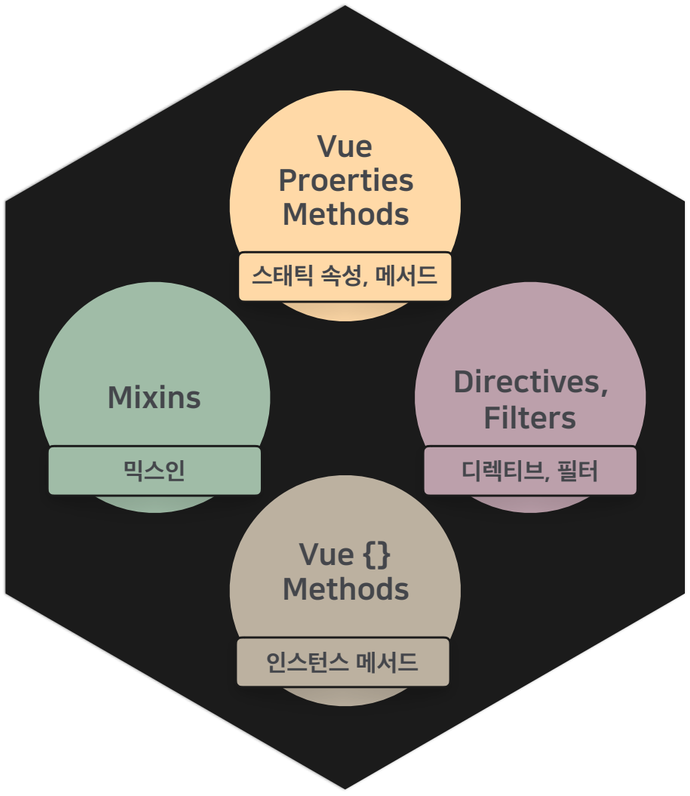
사용자 정의 옵션 설정

Mixins

- 전역 믹스인을 사용하여 컴포넌트를 확장할 경우, 모든 컴포넌트에 영향 (주의)
- 사용자 정의 옵션 처리 로직 주입에 활용

> ! 글로벌 mixin은 써드파티 컴포넌트를 포함하여 생성 된 모든 단일 Vue 인스턴스에 영향을 주기 때문에 적게 이용하고 신중하게 사용하십시오. 대부분의 경우 위 예제에서와 같이 사용자 지정 옵션 처리에만 사용해야합니다. 중복 적용을 피하기 위해 Plugins로 제공하는 것도 좋은 생각입니다.

Plugins 플러그인



플러그인 객체 (Plugin Object)

Vue.use()

뷰 생성자 (Vue Class)

https://kr.vuejs.org/v2/guide/plugins.html

Plugins

```
MyPlugin.install = function (Vue, options) {
  // 1. 전역 메소드 또는 속성 추가
  Vue.myGlobalMethod = function () {
    // 필요한 로직 ...
  }

  // 2. 전역 에셋 추가
  Vue.directive('my-directive', {
    bind (el, binding, vnode, oldVnode) {
      // 필요한 로직 ...
    }
    ...
  })

  // 3. 컴포넌트 옵션 주입
  Vue.mixin({
    created: function () {
      // 필요한 로직 ...
    }
    ...
  })

  // 4. 인스턴스 메소드 추가
  Vue.prototype.$myMethod = function (options) {
    // 필요한 로직 ...
  }
}
```

Vue 생성자

사용자 정의 옵션

플러그인 객체의 install() 메서드

전역 메서드, 속성

전역 디렉티브, 필터

전역 믹스인

인스턴스 메서드

Plugins

`Vue.use()` 글로벌 메소드를 호출하여 플러그인을 사용하십시오.

```JS
// `MyPlugin.install(Vue)` 호출
Vue.use(MyPlugin)
```

선택적으로 몇 가지 옵션을 전달할 수 있습니다.

사용자 정의 옵션

```JS
Vue.use(MyPlugin, { someOption: true })
```

`Vue.use` 는 자동으로 같은 플러그인을 두 번 이상 사용하지 못하기 때문에 같은 플러그인에서 여러 번 호출하면 플러그인이 한 번만 설치됩니다.

Plugins

```
// 플러그인 객체
var vuePlugin = {
  // 플러그인 환경설정 객체
  config: {},
  // 플러그인 설치 함수
  install: function(Vue, options){
    // 전역 디렉티브 추가
    Vue.directive('디렉티브-이름', {});
    // 전역 믹스인 추가
    // 믹스인 객체 설정
    Vue.mixin({
      mounted: function(){}
    });
    // 인스턴스 속성/메서드 추가
    Vue.prototype.$instanceProperty = '';
    Vue.prototype.$instanceMethod = function(options){};
  }
};


// 플러그인 사용 및 옵션 객체(선택사항) 설정
Vue.use(vuePlugin, {});
```

jQuery 유틸리티 메서드와 유사한 함수를 Vue 플러그인을 사용하여 유틸리티 확장

```
// 플러그인 객체
var yamoo9 = {
  install: function(Vue, options){
    // 비공개 변수,함수
    var toString = Object.prototype.toString;
    var slice    = Array.prototype.slice;
    // Vue 스태틱 메서드(Static Methods)
    Vue.type = function(o) { return toString.call(o).slice(8,-1).toLowerCase() };
    Vue.isFunction = function(o) { return this.type(o) === 'function' };
    Vue.isArray = function(o) { return this.type(o) === 'array' };
    Vue.isObject = function(o) { return this.type(o) === 'object' };
    Vue.isNodelist = function(o) {
      var type = this.type(o);
      return type === 'htmlcollection' || type === 'nodelist';
    };
    Vue.makeArray = function(o) { return slice.call(o) };
    Vue.checkTypes = function(condition, error_msg) { if (condition) { throw error_msg } };
    Vue.each = function(o, cb) {
      this.checkTypes(!o || !cb || !this.isFunction(cb), '2개의 전달인자가 필요하며, 마지막 인자는 함수여야 합니다.');
      if ( this.isArray(o) || this.isNodelist(o) ) {
          this.makeArray(o).forEach(cb);
      } else if ( this.isObject(o) ) {
        for ( var key in o ) {
          if ( o.hasOwnProperty(key) ) { cb(key, o[key], o); }
        }
      } else {
        console.info('배열, 객체 유형만 처리가 가능합니다.')
      }
    };
  }
};

// 플러그인 사용 및 옵션 객체(선택사항) 설정
Vue.use(yamoo9, {});
```

```
// 모듈 외 시스템(프론트엔드 환경)에서 플러그인 자동 설치
if ( typeof window !== 'undefined' && window.Vue ) {
  window.Vue.use(yamoo9);
}
```

모듈을 사용할 수 없는
웹 브라우저 환경에서 자동 설치

vue.js

vue.yamoo9.js

플러그인 배포

플러그인을 작성하고 커뮤니티에 배포 할 준비가되면 다음과 같이 사용자가 플러그인을 발견하는 데 도움이되는 몇 가지 일반적인 단계가 있습니다.

- 소스 코드와 배포 가능한 파일을 NPM 및 GitHub에 게시하십시오 . (코드에 맞는 라이선스를 선택했는지 확인하십시오!)
- 공식 Vue cool-stuff-discovery 저장소 (Awesome-Vue) 에 대한 요청을 엽니 다 . 많은 사람들이 플러그인을 찾기 위해 여기에옵니다.
- (선택 사항) Vue 포럼 , Vue Gitter 채널 및 해시 태그 #vuejs가 포함 된 Twitter에 게시하십시오 .

뷰 인스턴스 — 고급

Vue Instance Advanced

Vue 인스턴스 API

Options
옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- 컴포지션
- 기 타

Properties
속성

Methods
메서드

- 데이터
- 이벤트
- 라이프 사이클

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- 컴포지션
- 기 타

- data
- props
- propsData ·····> 단위 테스트 용도
- computed
- methods
- watch

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- 컴포지션
- 기 타

el

template

render ····· 문자열 템플릿 대신 자바스크립트의 완전한 프로그래밍 기능을 활용

renderError

⋮ ····· 기본 render 함수가 에러를 만나면, 대체되는 렌더 결과를 제공

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- 컴포지션
- 기 타

beforeCreate

created

beforeMount

mounted

beforeUpdate

updated

activated ·····→ keep-alive 인 컴포넌트가 활성화 될 때 호출

deactivated ·····→ keep-alive 인 컴포넌트가 비 활성화 될 때 호출

beforeDestroy

destroyed

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- **에 셋**
- 컴포지션
- 기 타

directives ·····→ 지역 디렉티브 등록 시, 사용

filters ·····→ 지역 필터 등록 시, 사용

components

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- **컴포지션**
- 기 타

parent ·····→ 상위 인스턴스 설정 (적절한 사용 요구)

mixins ·····→ 믹스인 객체 배열을 수용 (병합)

extends ·····→ 선언적으로 다른 컴포넌트 확장

provide / inject

⋮·····→ 고급 플러그인/컴포넌트 라이브러리를 위해 제공
(일반 애플리케이션 코드에서는 사용하지 않을 것을 권장)

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Options

옵션

- 데이터
- DOM
- 라이프 사이클 훅
- 에 셋
- 컴포지션
- 기 타

name ·····→ 컴포넌트 이름 등록 (컴포넌트 옵션)

delimiters ·····→ 텍스트 보간 구분 기호를 변경

functional ·····→ 단순히 가상 노드를 반환하는 render 함수로 렌더링을 훨씬 더 가볍게 만듬.

model

⋮·····→ 커스텀 컴포넌트가 v-model과 함께 사용될 때 prop와 이벤트를 커스터마이징

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Properties

속성

vm.$data

vm.$props ·····→ 컴포넌트가 전달 받은 속성을 나타내는 객체

vm.$el

vm.$options ·····→ 옵션에 사용자 정의 속성을 포함해야할 경우 유용

vm.$parent ·····→ 부모 인스턴스가 존재한다면, 부모 인스턴스 참조

vm.$root ·····→ 현재 컴포넌트 트리의 루트 Vue 인스턴스

vm.$children ·····→ 현재 인스턴스가 가지고 있는 바로 하위의 컴포넌트

vm.$slots ·····→ 각 명명된 슬롯을 프록시(우회)하는 속성

vm.$scopedSlots ·····→ 범위가 지정된 슬롯에 프로그래밍으로 액세스

vm.$refs ·····→ ref가 등록된 자식 컴포넌트를 보관하는 객체

vm.$isServer ·····→ 현재 Vue 인스턴스가 서버에서 실행중인지 여부

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Methods

메서드

데이터

이벤트

라이프 사이클

vm.$watch ·····→ 변경을 위해 Vue 인스턴스에서 표현식이나 계산된 함수를 감시

vm.$set ·····→ 전역 Vue.set의 별칭

vm.$delete ·····→ 전역 Vue.delete의 별칭

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Methods

메서드

데이터

이벤트

라이프 사이클

vm.$on ·····→ 현재 인스턴스에서 사용자 정의 이벤트 수신

vm.$once ·····→ 사용자 이벤트를 한번만 수신 (수신 후 삭제)

vm.$off ·····→ 이벤트 리스너를 제거

vm.$emit ·····→ 현재 인스턴스에서 이벤트를 트리거

https://kr.vuejs.org/v2/api/

Vue 인스턴스 API

Methods

메서드

데이터

이벤트

라이프 사이클

vm.$mount ·····→ unmounted 된 Vue인스턴스의 마운트를 수동으로 시작하는데 사용

vm.$forceUpdate ·····→ Vue 인스턴스를 강제로 다시 렌더링

vm.$nextTick ·····→ 다음 DOM 업데이트 사이클 이후 실행될 콜백을 연기

vm.$destroy

·····→ vm을 완전히 제거

https://kr.vuejs.org/v2/api/