Mixins

Basics

Mixins are a flexible way to distribute reusable functionalities for Vue components. A mixin object can contain any component options. When a component uses a mixin, all options in the mixin will be “mixed” into the component’s own options.

Example:

```js
// define a mixin object
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}

// define a component that uses this mixin
var Component = Vue.extend({
  mixins: [myMixin]
})

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

Option Merging

When a mixin and the component itself contain overlapping options, they will be “merged” using appropriate strategies. For example, hook functions with the same name are merged into an array so that all of them will be called. In addition, mixin hooks will be called **before** the component’s own hooks:

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

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

```
// => "mixin hook called"
// => "component hook called"
```

Options that expect object values, for example `methods`, `components` and `directives`, will be merged into the same object. The component's options will take priority when there are conflicting keys in these objects:

```
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"
```

JS

Note that the same merge strategies are used in `Vue.extend()`.

Global Mixin

You can also apply a mixin globally. Use with caution! Once you apply a mixin globally, it will affect **every** Vue instance created afterwards. When used properly, this can be used to inject processing logic for custom options:

```
// inject a handler for `myOption` custom option
Vue.mixin({
  created: function () {
    var myOption = this.$options.myOption
    if (myOption) {
      console.log(myOption)
    }
  }
})

new Vue({
```

JS

```
  myOption: 'hello!'
})
// => "hello!"
```

! Use global mixins sparsely and carefully, because it affects every single Vue instance created, including third party components. In most cases, you should only use it for custom option handling like demonstrated in the example above. It’s also a good idea to ship them as **Plugins** to avoid duplicate application.