

คพ494

หัวข้อพิเศษทางวิทยาการคอมพิวเตอร์ 1

VUE.JS – BINDING



Standard
Library

COURSE OUTLINE

- ▶ Introduction
 - instance, syntax, setup
- ▶ **Interact with DOM**
 - **binding, event, styling**
- ▶ v-if, v-for
- ▶ Module
- ▶ Component
- ▶ FORM
- ▶ Animations
- ▶ Http
- ▶ Routing
- ▶ Vue Native

ConfigGlobal

silent

optionMergeStrategies

devtools

errorHandler

ignoredElements

keyCodes

performance2.2

productionTip2.2

APIGlobal

Vue.extend

Vue.nextTick

Vue.set

Vue.delete

Vue.directive

Vue.filter

Vue.component

Vue.use

Vue.mixin

Vue.compile

Vue.version

Lifecycle HooksOptions

beforeCreate

created

beforeMount

mounted

beforeUpdate

updated

activated

deactivated

beforeDestroy

destroyed

DataOptions

data

props

propsData

computed

methods

watch

DOMOptions

el

template

render

renderError2.2

AssetsOptions

directives

filters

components

CompositionOptions

parent

mixins

extends

provide / inject2.2

MiscOptions

name

delimiters

functional

model2.2

DataMethods

vm.\$watch

vm.\$set

vm.\$delete

EventsMethods

vm.\$on

vm.\$once

vm.\$off

vm.\$emit

LifecycleMethods

vm.\$mount

vm.\$forceUpdate

vm.\$nextTick

vm.\$destroy

PropertiesInstance

vm.\$data

vm.\$props2.2

vm.\$el

vm.\$options

vm.\$parent

vm.\$root

vm.\$children

vm.\$slots

vm.\$scopedSlots2.1

vm.\$refs

vm.\$isServer

Directiveshtml

v-text

v-html

v-show

v-if

v-else

v-else-if2.1

v-for

v-on

v-bind

v-model

v-pre

v-cloak

v-once

DirectivesHooks

bind

inserted

update

componentUpdated

unbind

Attributeshtml

key

ref

slot

ComponentsBuilt-In

component

transition

transition-group

keep-alive

slot

v-modelModifiers

v-model.lazy

v-model.number

v-model.trim

EventModifiers

v-on:click.native

v-on:click.stop

v-on:click.prevent

v-on:click.passive2.3

v-on:click.capture

v-on:click.self

v-on:click.once2.1.4

v-bindModifiers

v-bind.prop

v-bind.camel2.1

v-bind.sync2.3

MouseModifiers

v-on:click.left2.2

v-on:click.right2.2

v-on:click.middle2.2

KeyboardModifiers

v-on:keyup.ctrl2.1

v-on:keyup.alt2.1

v-on:keyup.shift2.1

v-on:keyup.meta2.1

v-on:keyup.enter

v-on:keyup.tab

v-on:keyup.delete

v-on:keyup.esc

v-on:keyup.up

v-on:keyup.down

v-on:keyup.left

v-on:keyup.right

<https://vuejs-tips.github.io/cheatsheet/>

CREATE VUE

▶ Vue Object/ Instance

```
var vm = new Vue({  
  // options  
})
```

▶ Data Example

```
newTodoText: '',  
visitCount: 0,  
hideCompletedTodos: false,  
todos: [],  
error: null
```

▶ Vue Data Object

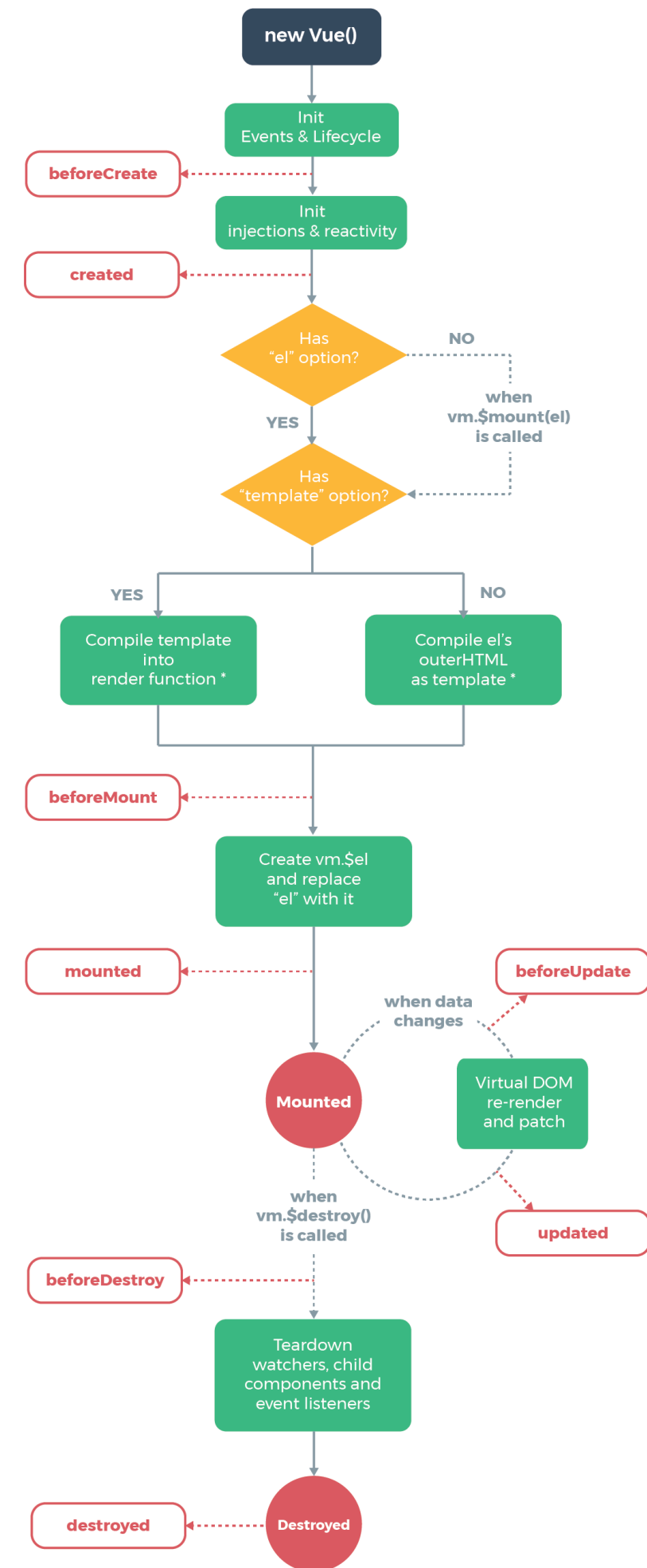
```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!'  
  }  
})
```

CREATE VUE

Vue Object/ Instance Example

```
// Our data object  
var data = { a: 1 }
```

```
// The object is added to a Vue instance  
var vm = new Vue({  
  data: data  
})
```



OUTPUT

- **Text** → using the “Mustache” syntax (double curly braces)

```
<span>Message: {{ msg }}</span>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    msg: 'Hello Vue!'  
  }  
})
```

- **Raw HTML** → v-html directive

```
<p>Using mustaches: {{ rawHtml }}</p>
```

```
<p>Using v-html directive: <span v-html="rawHtml"></span></p>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    rawHtml: '<span style="color:red">This should be red</span>'  
  }  
})
```

BINDING

- ▶ v-model = 2 way binding
- ▶ v-bind = binding into html attribute

```
<h1>{{ greeting }}</h1>  
<input type="text" v-model="greeting">  
<button v-bind:disabled="isButtonDisabled">Button</button>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    greeting: 'Welcome to your Vue.js app!',  
    isButtonDisabled: false  
  }  
})
```

- ▶ v-bind shorthand

```
<!-- full syntax -->  
<a v-bind:href="url"> ... </a>  
<!-- shorthand -->  
<a :href="url"> ... </a>
```

BINDING - OBJECT

```
<div id="app">
  <h2>ID = {{customer.id}} </h2>
  <h2>Code = {{customer.code}} </h2>
  <h2>Name = {{customer.name}} </h2>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    customer: {
      id: 1,
      code: 'mju85',
      name: 'Maejo University',
      city: 'Chiangmai'
    }
  },
})
```


METHODS

```
<script>
  var app = new Vue({
    el: '#app',
    data: {

    },
    methods: {
      methodName: function (url) {
        //.....
      }
    }
  })
</script>
```

COMPUTED

- ▶ computed => ใช้เหมือน data มีเพียงการเรียกใช้ข้อมูลเท่านั้น
- ▶ ไม่สามารถทำการ binding กลับได้ **getter method**
- ▶ ไม่สามารถส่ง parameter เข้า function ได้
- ▶ คำนวณผลรวมเงินเดือนของพนักงานทั้งหมด, หาค่าเฉลี่ย, จัดรูปแบบการแสดงผลข้อมูล, ...

```
computed: {  
  // get only  
  aDouble: function () {  
    return this.a * 2  
  },  
  // both get and set  
  aPlus: {  
    get: function () {  
      return this.a + 1  
    },  
    set: function (v) {  
      this.a = v - 1  
    }  
  }  
},
```

```
{{aPlus=3}} <br>  
{{a}} <br>  
{{aDouble}} <br>
```

<https://vuejs.org/v2/api/#computed>

<https://vuejs.org/v2/guide/computed.html>

EVENT HANDLING

▶ HTML Event → https://www.w3schools.com/tags/ref_eventattributes.asp

➡ Window Event

➡ Form Events

➡ Keyboard Events

➡ Mouse Events

➡ Drag Events

➡ Clipboard Events

➡ Media Events

➡ Misc Events

EVENT HANDLING : V-ON

```
<div id="example">  
  <button v-on:click="greet">Greet</button>  
</div>
```

```
var vm = new Vue({  
  el: '#example',  
  data: {  
    name: 'Vue.js'  
  },  
  // define methods under the `methods` object  
  methods: {  
    greet: function (event) {  
      // `this` inside methods point to the Vue instance  
      alert('Hello ' + this.name + '!')  
      // `event` is the native DOM event  
      alert(event.target.tagName)  
    }  
  }  
})  
  
// you can invoke methods in JavaScript too  
vm.greet() // -> 'Hello Vue.js!'
```

EVENT HANDLING : V-ON +PARAMETER

```
<div id="example-2">
  <button v-on:click="say('hi')">Say Hi</button>
  <button v-on:click="say('what')">Say What</button>
</div>
```

```
new Vue({
  el: '#example-2',
  methods: {
    say: function (msg) {
      alert(msg)
    }
  }
})
```

► v-on Shorthand

```
<!-- full syntax -->
<a v-on:click="doSomething"> ... </a>
```

```
<!-- shorthand -->
<a @click="doSomething"> ... </a>
```

TIME TO PRACTICE

1. Icon Counter



2. เกมต่อสู้

- ▶ รูปภาพตัวละคร 2 รูป --> Player vs Monster ความสูง = 100 px
- ▶ button -> Start, Attack, Special Attack
- ▶ กดปุ่ม Start = เริ่มเกมส์ใหม่ ค่า health = 100
- ▶ ความสูงของรูปภาพมีขนาดตามค่า health
- ▶ กดปุ่ม Attack = random 3-10, กดปุ่ม Special Attack = random 10-20 เพื่อไปลดค่า health ของ monster
- ▶ แต่ละครั้งที่กดปุ่ม Attack/ Special Attack ให้ Monster โจมตีกลับ random 5-15 เพื่อไปลดค่า health ของ player
- ▶ ตรวจสอบผู้ชนะ health <= 0

```
randomDamage: function(min, max) {  
    return Math.max(Math.floor(Math.random() * max) + 1, min);  
},
```