

คพ494

หัวข้อพิเศษทางวิทยาการคอมพิวเตอร์ 1

COMPONENT



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

Config <small>Global</small> silent optionMergeStrategies devtools errorHandler ignoredElements keyCodes performance <small>2.2</small> productionTip <small>2.2</small>	Lifecycle Hooks <small>Options</small> beforeCreate created beforeMount mounted beforeUpdate updated activated deactivated beforeDestroy destroyed	Composition <small>Options</small> parent mixins extends <small>2.2</small> provide / inject <small>2.2</small> Misc <small>Options</small> name delimiters functional model <small>2.2</small>	Properties <small>Instance</small> vm.\$data vm.\$props <small>2.2</small> vm.\$el vm.\$options vm.\$parent vm.\$root vm.\$children vm.\$slots vm.\$scopedSlots <small>2.1</small> vm.\$refs vm.\$isServer	Directives <small>Hooks</small> bind inserted update componentUpdated unbind Attributes <small>html</small> key ref slot	v-bind <small>Modifiers</small> v-bind .prop v-bind .camel <small>2.1</small> v-bind .sync <small>2.3</small>
API <small>Global</small> Vue.extend Vue.nextTick Vue.set Vue.delete Vue.directive Vue.filter Vue.component Vue.use Vue.mixin Vue.compile Vue.version	Data <small>Options</small> data props propsData computed methods watch	Data <small>Methods</small> vm.\$watch vm.\$set vm.\$delete	Directives v-text v-html v-show v-if v-else v-else-if v-for v-on v-bind v-model v-pre v-cloak v-once	Components <small>Built-In</small> component transition transition-group keep-alive slot v-model <small>Modifiers</small> v-model .lazy v-model .number v-model .trim Event <small>Modifiers</small> v-on:click .native v-on:click .stop v-on:click .prevent v-on:click .passive <small>2.3</small> v-on:click .capture v-on:click .self v-on:click .once <small>2.1.4</small>	Mouse <small>Modifiers</small> v-on:click .left <small>2.2</small> v-on:click .right <small>2.2</small> v-on:click .middle <small>2.2</small> Keyboard <small>Modifiers</small> v-on:keyup .ctrl <small>2.1</small> v-on:keyup .alt <small>2.1</small> v-on:keyup .shift <small>2.1</small> v-on:keyup .meta <small>2.1</small> 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
	DOM <small>Options</small> el template render renderError <small>2.2</small>	Events <small>Methods</small> vm.\$on vm.\$once vm.\$off vm.\$emit	Lifecycle <small>Methods</small> vm.\$mount vm.\$forceUpdate vm.\$nextTick vm.\$destroy		
	Assets <small>Options</small> directives filters components				

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

FORM INPUT BINDINGS

v-model

```
<input v-model="message" placeholder="edit me">
```

```
<textarea v-model="message"
  placeholder="add multiple lines"></textarea>
```

```
<input type="checkbox" id="checkbox" v-model="checked">
```

```
<input type="checkbox" id="jack" value="Jack" v-model="checkedNames">
<input type="checkbox" id="john" value="John" v-model="checkedNames">
<input type="checkbox" id="mike" value="Mike" v-model="checkedNames">
```

FORM INPUT BINDINGS

v-model

```
<input type="radio" id="one" value="One" v-model="picked">
```

```
<input type="radio" id="two" value="Two" v-model="picked">
```

```
<select v-model="selected">  
  <option disabled value="">Please select one</option>  
  <option>A</option>  
  <option>B</option>  
  <option>C</option>  
</select>
```

```
<select v-model="selected" multiple>
```

FORM INPUT BINDINGS

- ▶ dynamic options rendered with v-for

```
<select v-model="selected">
  <option v-for="option in options" v-bind:value="option.value">
    {{ option.text }}
  </option>
</select>
<span>Selected: {{ selected }}</span>
```

```
new Vue({
  el: '...',
  data: {
    selected: 'A',
    options: [
      { text: 'One', value: 'A' },
      { text: 'Two', value: 'B' },
      { text: 'Three', value: 'C' }
    ]
  }
})
```

FORM MODIFIERS

- ▶ .lazy = sync data after "change" events

```
<input v-model.lazy="msg" >
```

- ▶ .number = automatically type cast as a number

```
<input v-model.number="age" type="number">
```

- ▶ .trim = trimmed automatically

```
<input v-model.trim="msg">
```

CLASS & STYLE BINDING

Style Bindings

```
<h1 :style="{ color: color }">...</h1>  
                'red'
```

JS

```
data: {  
  color: 'red'  
}
```

Style Bindings

```
<p :style="{ fontSize: fontSize }">...</p>  
                '13px'
```

JS

```
data: {  
  color: 'red',  
  fontSize: '13px'  
}
```


CLASS & STYLE BINDING

Style Bindings

Objects

```
<span :style="style0object">...</span>
```

JS

```
data: {  
  style0object: {  
    color: 'red',  
    fontSize: '13px'  
  }  
}
```

Style Bindings

Arrays

```
<p :style="[style0object, style0object2]">...</p>
```

JS

```
data: {  
  style0object: {  
    color: 'red',  
    fontSize: '13px'  
  },  
  style0object2: {  
    margin: '5px',  
    padding: '20px'  
  }  
}
```

CLASS & STYLE BINDING

Class Bindings

```
<div class="color-box"  
  :class="{ active: activeClass, 'text-danger': errorClass }">  
...                                true                false  
</div>
```

JS

```
data: {  
  activeClass: true,  
  errorClass: false  
}
```

CLASS & STYLE BINDING

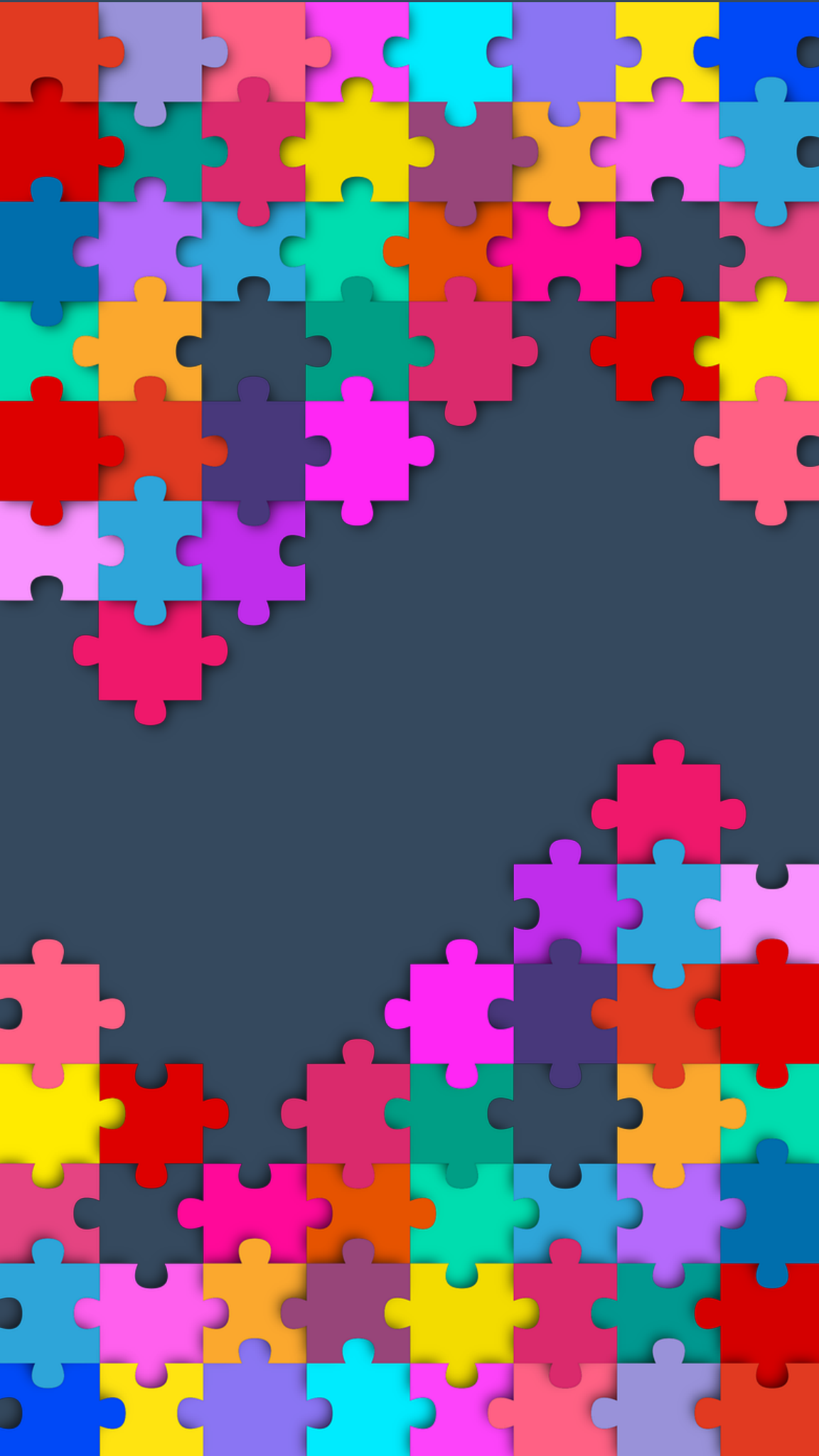
Class Bindings

Objects

```
<div :class="class0object">...</div>
```

JS

```
data: {  
  class0object: {  
    active: true,  
    'text-danger': false  
  }  
}
```

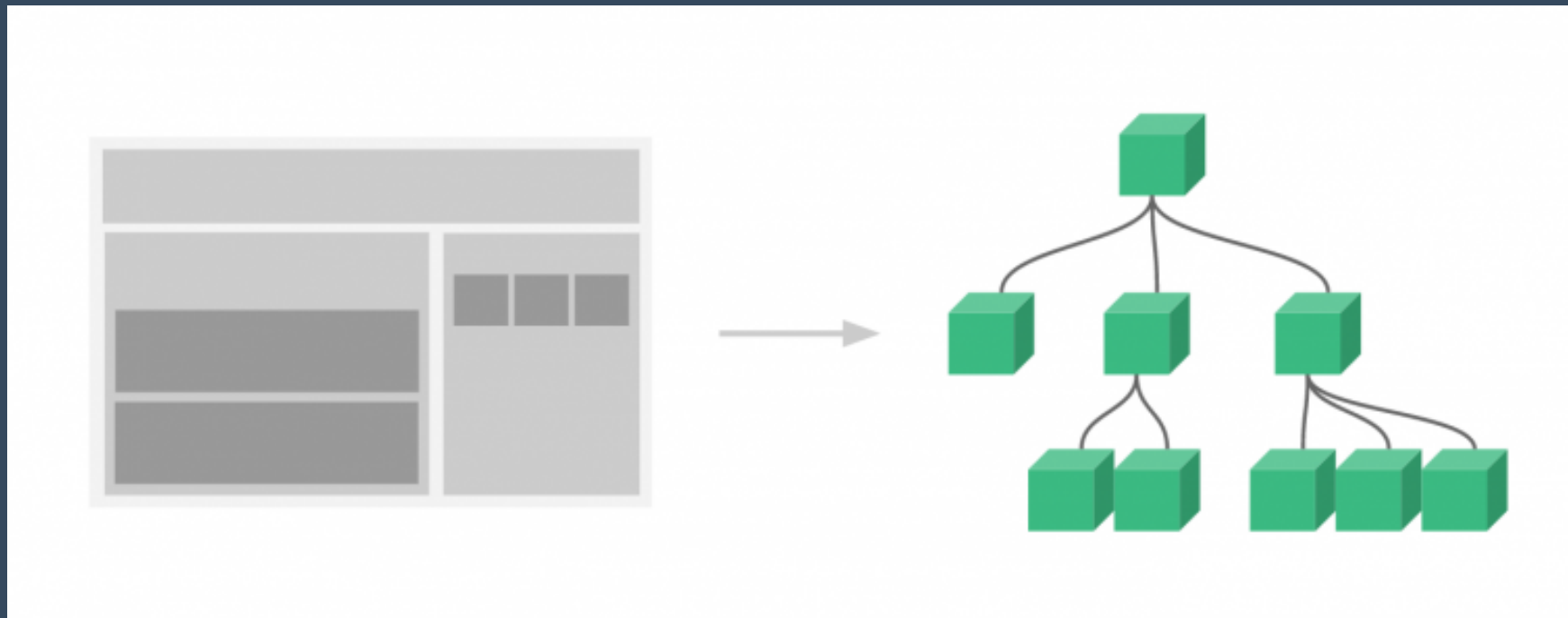


CREATE WEB WITH

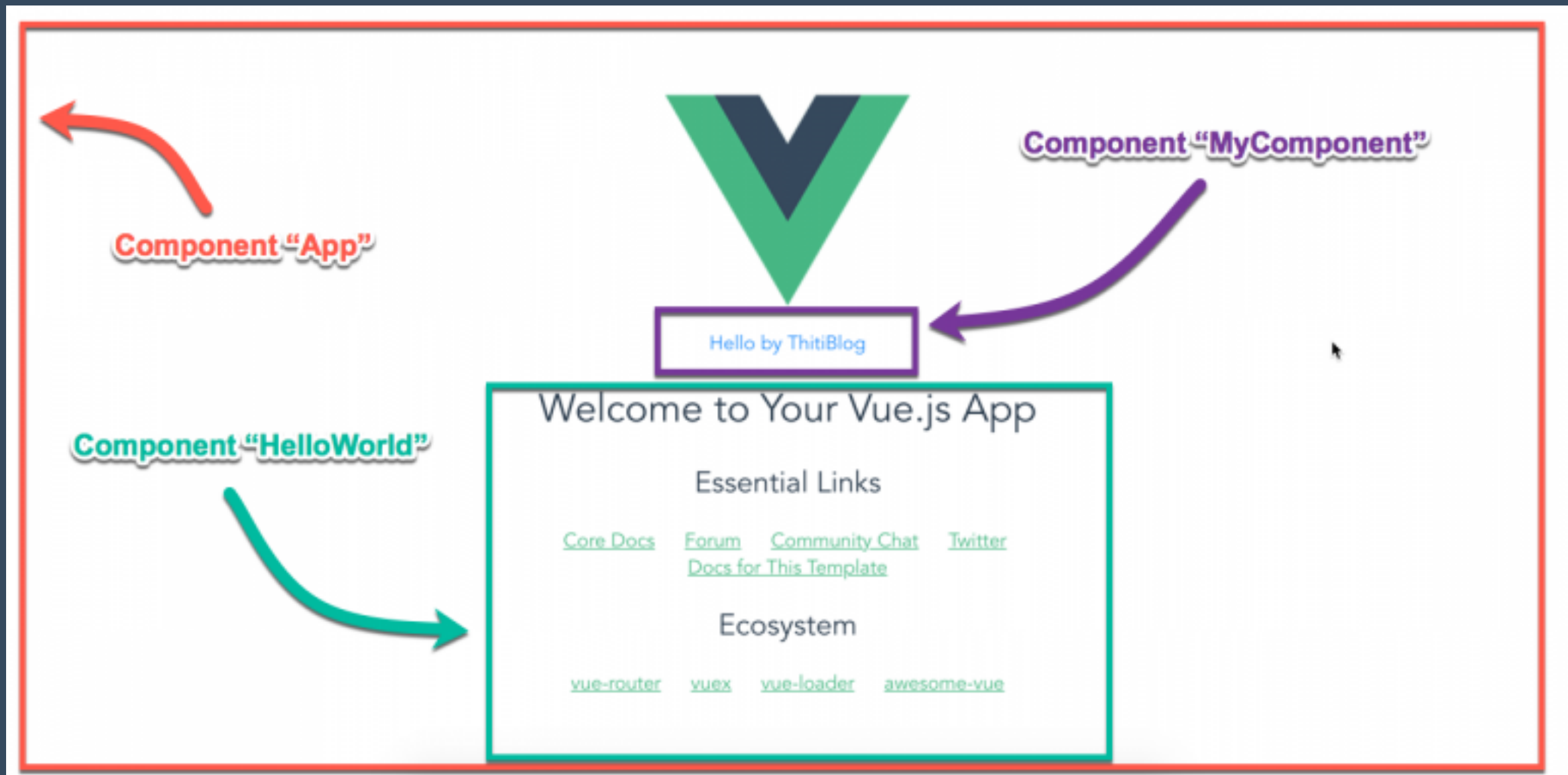
VUE
COMPONENT

COMPONENTS BASICS

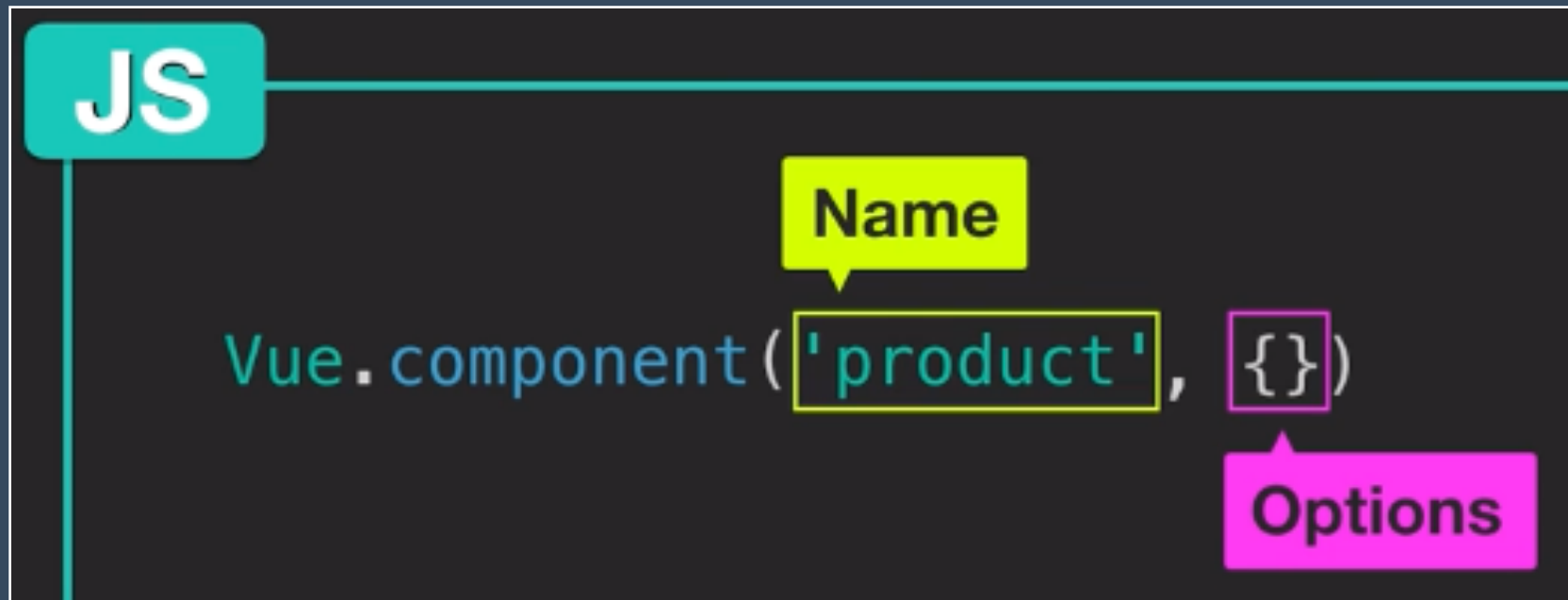
- ▶ Component = HTML + CSS + JS
- ▶ เป็นการกำหนดชิ้นส่วนการแสดงผลใน vue application ซึ่งสามารถแบ่งการทำงานให้เป็นอิสระต่อกันได้
- ▶ สามารถแยกเป็นไฟล์ของแต่ละ Component ได้ โดยใช้นามสกุล **.vue**
- ▶ มีองค์ประกอบ 3 ส่วน ได้แก่ `<template>` `<script>` `<style>`



COMPONENTS BASICS



COMPONENTS BASICS



- ▶ Name
- ▶ Option
 - template
 - `data()` *** function
 - props : pass data from parent to child (component can not access outside data)
 - methods...

COMPONENTS BASICS

JS

```
Vue.component('product', {  
  props: [message],  
  template: `<div>{{ message }}</div>`,  
  data() { ... }  
})
```

HTML

```
<product message="hello!"></product>
```


COMPONENTS BASICS

```
<div id="components-demo">
  <button-like></button-like>
</div>
```

```
<script>
  Vue.component(
    'button-like', {
      template: '<button >Like</button>'
    })

  new Vue({
    el: '#components-demo'
  })
</script>
```

COMPONENTS BASICS

```
<div id="components-demo">
  <button-like></button-like>
  <button-like></button-like>
  <button-like></button-like>
  <button-like></button-like>
</div>
```

```
<script>
  Vue.component(
    'button-like', {
      template: '<button >Like</button>'
    })

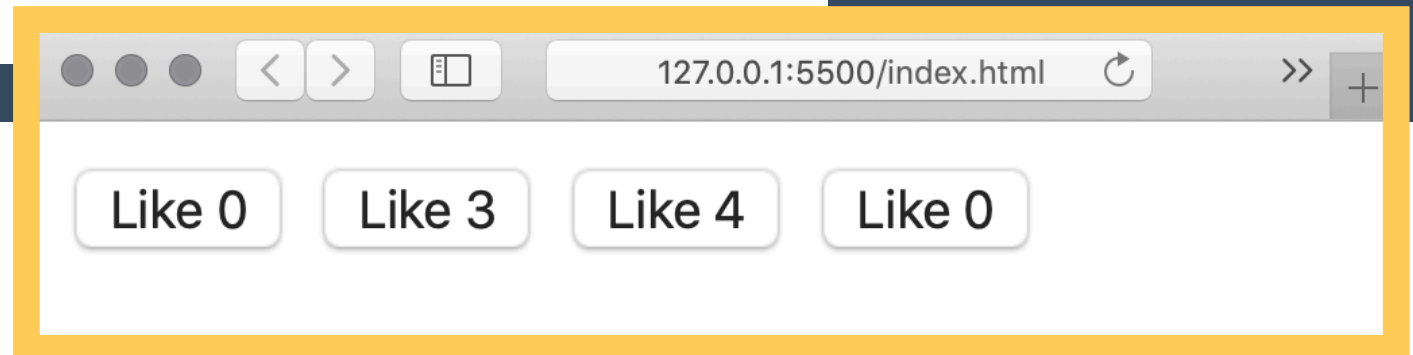
  new Vue({
    el: '#components-demo'
  })
</script>
```

COMPONENTS BASICS

```
<div id="components-demo">
  <button-like></button-like>
  <button-like></button-like>
  <button-like></button-like>
  <button-like></button-like>
</div>
```

```
<script>
  Vue.component(
    'button-like', {
      data: function () {
        return {
          count: 0
        }
      },
      template:
        '<button v-on:click="count++">Like {{ count }}</button>'
    })

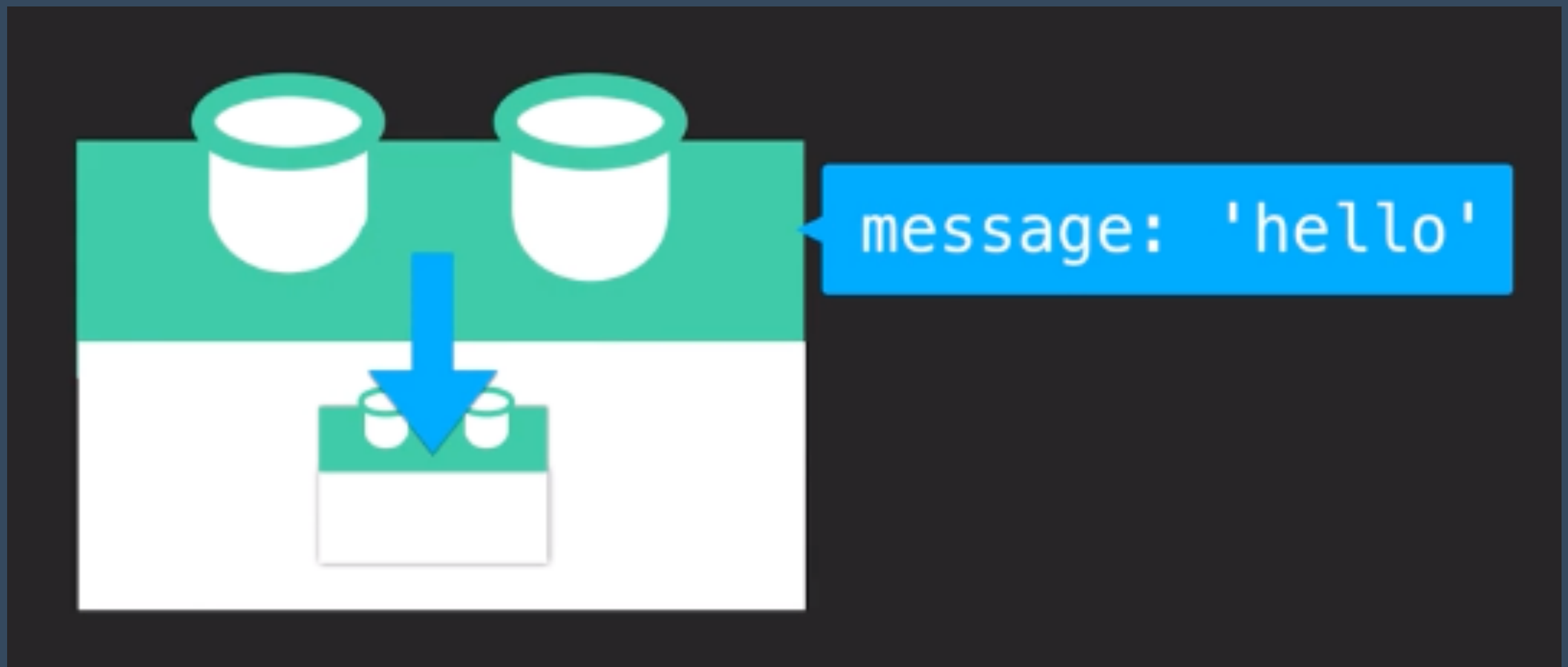
  new Vue({
    el: '#components-demo'
  })
</script>
```



data Must Be a Function !!!

COMPONENTS BASICS : PROPS

- ▶ "props" : essentially variables handle downward data sharing



PROPS

JS

```
Vue.component('product', {  
  props: [message],  
  template: `<div>{{ message }}</div>`,  
  data() {  
    .  
  }  
})
```

HTML

```
<product message="hello!"></product>
```

PROPS : OBJECT

JS

```
Vue.component('product', {  
  props: [message],  
  template: `<div>{{message}}</div>`  
})
```

JS

```
Vue.component('product', {  
  props: {  
    message: {  
      type: String,  
      required: true,  
      default: "Hi"  
    }  
  },  
  template: `<div> ... </div>`,  
  data() { ... }  
})
```

COMPONENTS BASICS

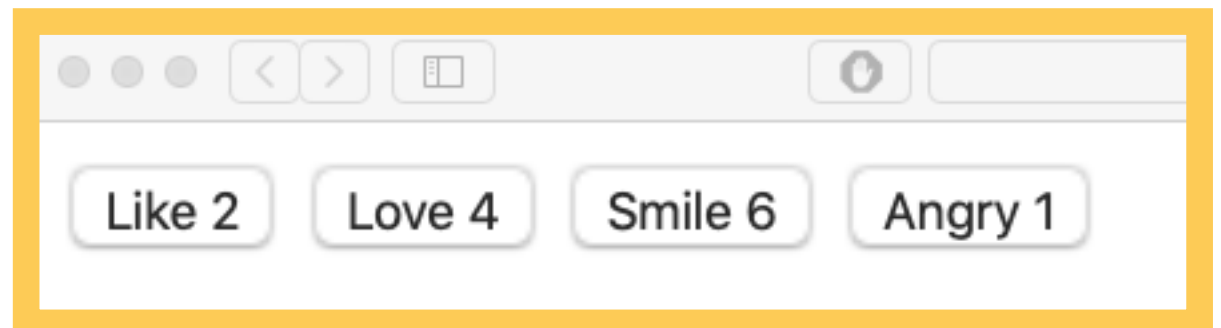
```
<div id="components-demo">  
  <button-like title="Like"></button-like>  
  <button-like title="Love"></button-like>  
  <button-like title="Smile"></button-like>  
  <button-like title="Angry"></button-like>  
</div>
```

1. ทำการ render การแสดงผลของ component จาก template

2. ส่งค่าชื่อปุ่ม [Like, Love, Smile, Angry] ผ่านตัวแปร title (props)

```
<script>  
  Vue.component(  
    'button-like', {  
      props: ['title'],  
      data: function () {  
        return {  
          count: 0  
        }  
      },  
      template: '<button v-on:click="count++">{{title}} {{ count }}</button>',  
    },  
  )  
  
  new Vue({  
    el: '#components-demo'  
  })  
</script>
```

3. binding แสดงผลค่าตัวแปร



COMPONENT COMMUNICATION EVENT

- ▶ แต่ละ component สามารถสื่อสารการทำงานด้วยคำสั่ง

\$emit

COMPONENT COMMUNICATION EVENT

- ▶ จาก Lab การกดปุ่ม Like, Love, Smile, Angry (แต่ละปุ่มเป็น component) หากต้องการนับจำนวนการ click รวมทั้งหมด สามารถสื่อสารมาที่ root component ได้โดยใช้คำสั่ง \$emit


```
Vue.component('button-like', {  
  props: ['title'],  
  data: function () {  
    return {  
      count: 0  
    }  
  },  
  template: '<button v-on:click="countUp">{{title}} {{ count }}</button>',  
  methods: {  
    countUp() {  
      this.count++;  
      this.$emit('sum-click');  
    }  
  }  
})
```

1. เรียก method "countUp" ใน component นี้

2. เตรียมชื่อ parameter ให้ root component เรียกใช้ "sum-click"


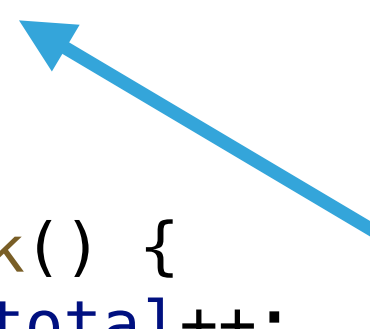
COMPONENT COMMUNICATION EVENT

```
<div id="components-demo">
  <button-like title="Like" @sum-click="totalClick"></button-like>
  <button-like title="Love" @sum-click="totalClick"></button-like>
  <button-like title="Smile" @sum-click="totalClick"></button-like>
  <button-like title="Angry" @sum-click="totalClick"></button-like>
  <hr>
  Total : {{total}}
</div>
```



3. กำหนดให้สื่อสารกับ root component ไปที่ method "totalClick"

```
new Vue({
  el: '#components-demo',
  data: {
    total: 0,
  },
  methods: {
    totalClick() {
      this.total++;
    }
  }
})
```



4. ทำงานที่ totalClick

5. นับจำนวนรวมทั้งหมด โดยใช้ตัวแปร total

COMPONENT COMMUNICATION EVENT

- ▶ ทั้งนี้ \$emit ยังสามารถรับส่งค่า parameter อื่นๆ เพิ่มเติมได้ เช่น หากต้องการเก็บชื่อปุ่มที่กดไปเก็บค่าไว้ใน Array

```
<div id="components-demo">
  <button-like title="Like" @sum-click="totalClick"></button-like>
  <button-like title="Love" @sum-click="totalClick"></button-like>
  <button-like title="Smile" @sum-click="totalClick"></button-like>
  <button-like title="Angry" @sum-click="totalClick"></button-like>
  <hr>
  Total : {{total}}<br>
  {{name}}
</div>
```

← สำหรับแสดงค่าที่เก็บไว้ใน array

```
methods: {
```

```
  countUp() {
    this.count++;
    this.$emit('sum-click', this.title);
  }
}
```

1. ส่งชื่อปุ่มที่กดออกไปที่ root component

```
new Vue({
  el: '#components-demo',
  data: {
    total: 0,
    name: [],
  },
  methods: {
    totalClick(n) {
      this.total++;
      this.name.push(n)
    }
  }
})
```

↓ เก็บชื่อไว้ใน array

Like 1 Love 1 Smile 1 Angry 4

Total : 7

["Smile", "Love", "Like", "Angry", "Angry", "Angry", "Angry"]



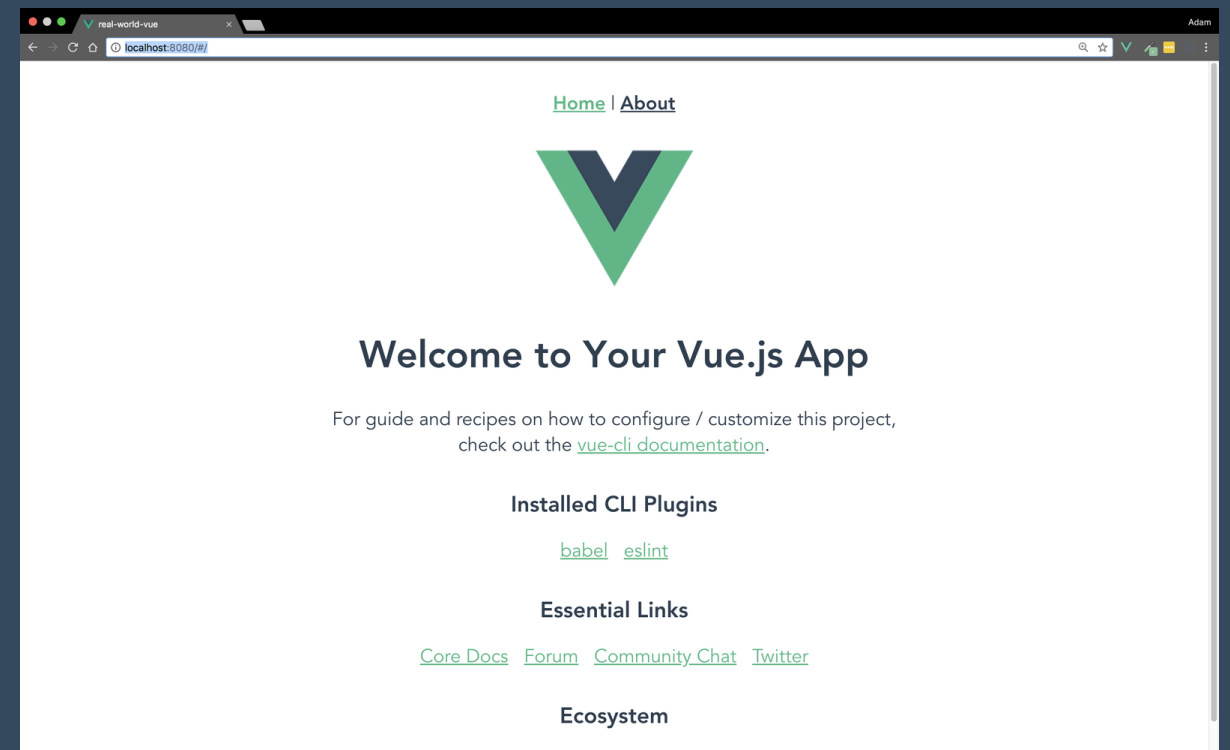
VUE

IN THE REAL WORLD

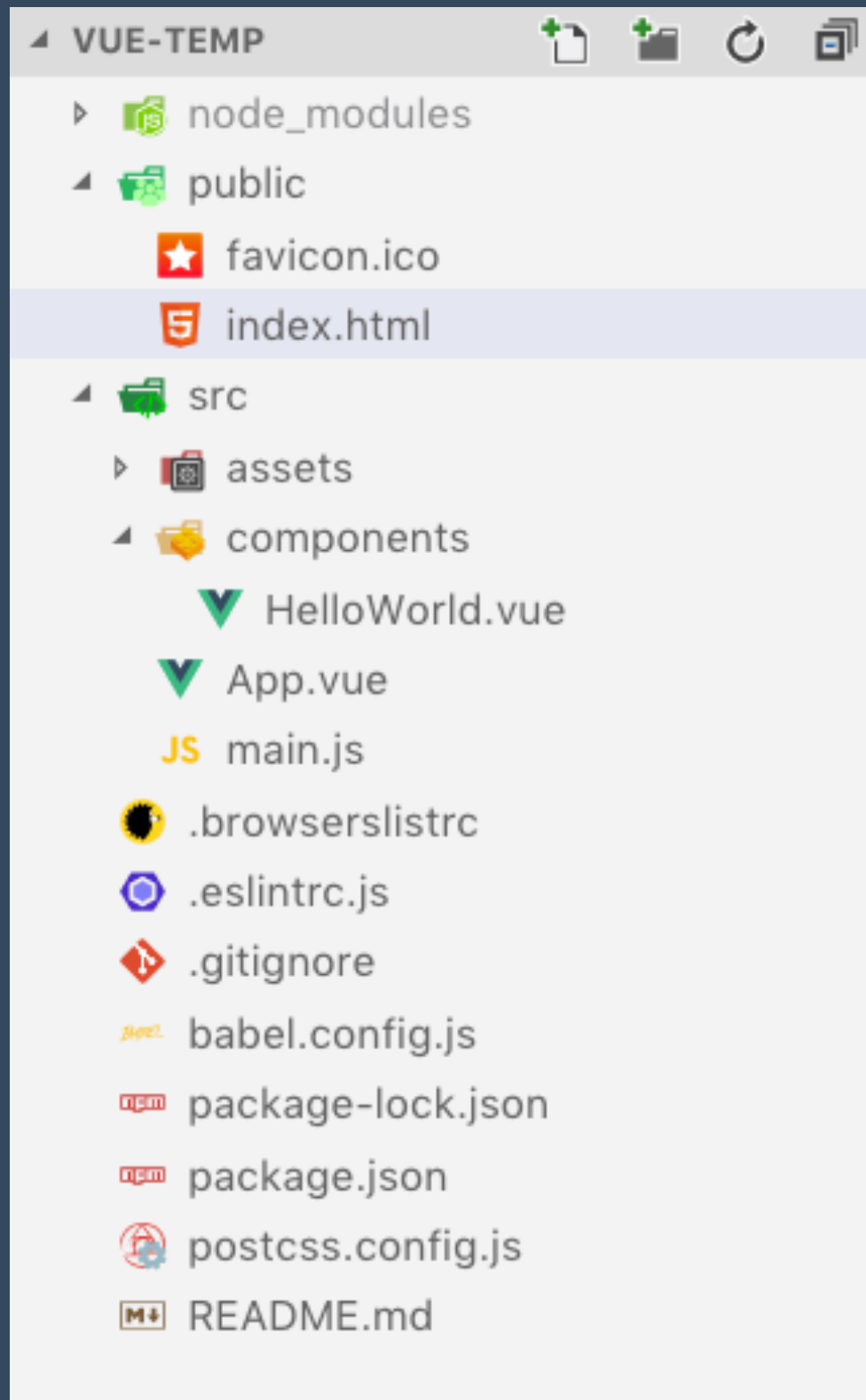
REAL DEVELOPMENT VUE-CLI

- ▶ `$ vue create real-world-vue`
- ▶ `$ cd real-world-vue`
- ▶ `$ npm run serve`
- ▶ `localhost:8080`

▶ `$ vue ui`



VUE PROJECT



- ▶ **node_modules** : libraries
- ▶ **public** : any static assets you don't want to be run through Webpack when we build our project
- ▶ **assets** : directory so they can be optimized by Webpack
- ▶ **components** : components, or building blocks, of our Vue app.
- ▶ **App.vue** : file is the root component that all other components are nested within
- ▶ **main.js** : file is what renders our App.vue component and mounts it to the DOM

VUE PROJECT



.VUE FILE

- ▶ .vue file is a Single File Component. It contains 3 chunks of code: HTML, CSS and JavaScript.

REAL DEVELOPMENT VUE-CLI

- ▶ `$ npm run build`
- ▶ This will build our project, and when it's complete, we see it says, "The **dist** directory is ready to be deployed".

```
vue-temp $ npm run build
```

```
> vue-temp@0.1.0 build /Volumes/Data/Vue/vue-temp  
> vue-cli-service build
```

```
⌚ Building for production...
```

```
DONE Compiled successfully in 25880ms
```

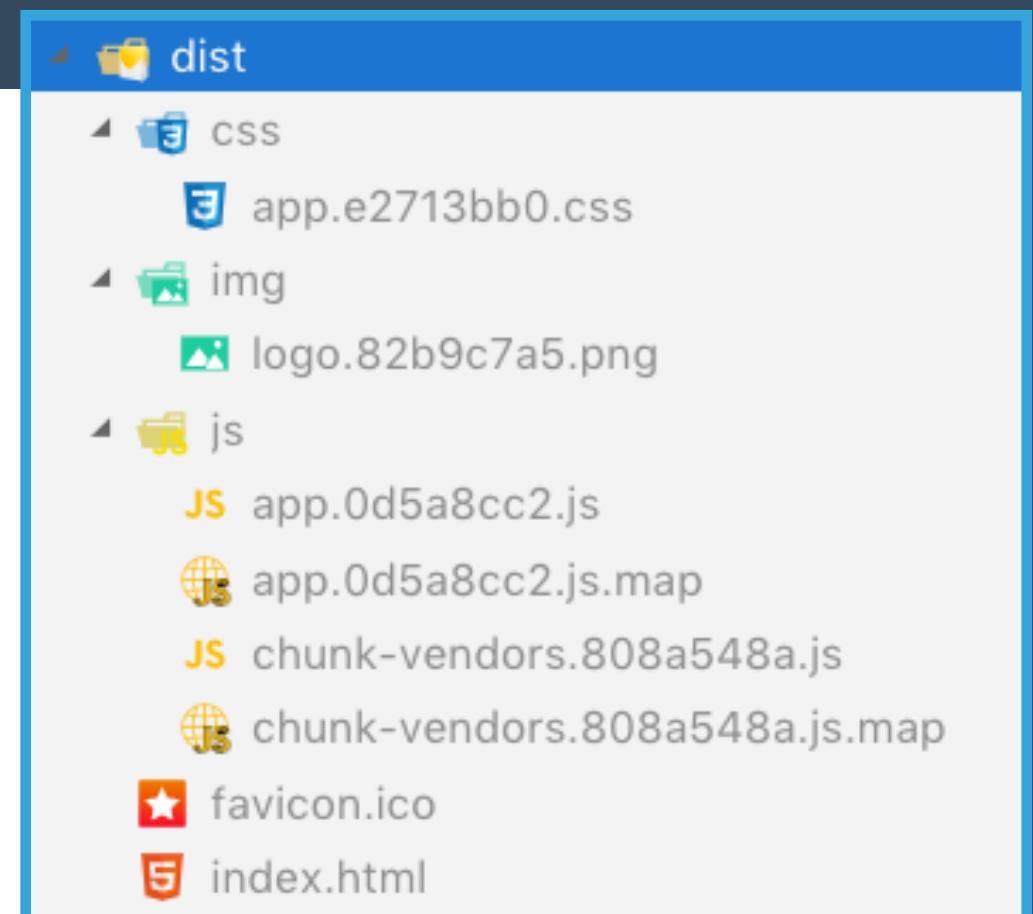
File	Size	Gzipped
<code>dist/js/chunk-vendors.808a548a.js</code>	79.30 kb	28.55 kb
<code>dist/js/app.0d5a8cc2.js</code>	4.66 kb	1.67 kb
<code>dist/css/app.e2713bb0.css</code>	0.33 kb	0.23 kb

Images and other types of assets omitted.

```
DONE Build complete. The dist directory is ready to be deployed.
```

```
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

```
vue-temp $
```



TIME TO PRACTICE
