SIT120 Practical

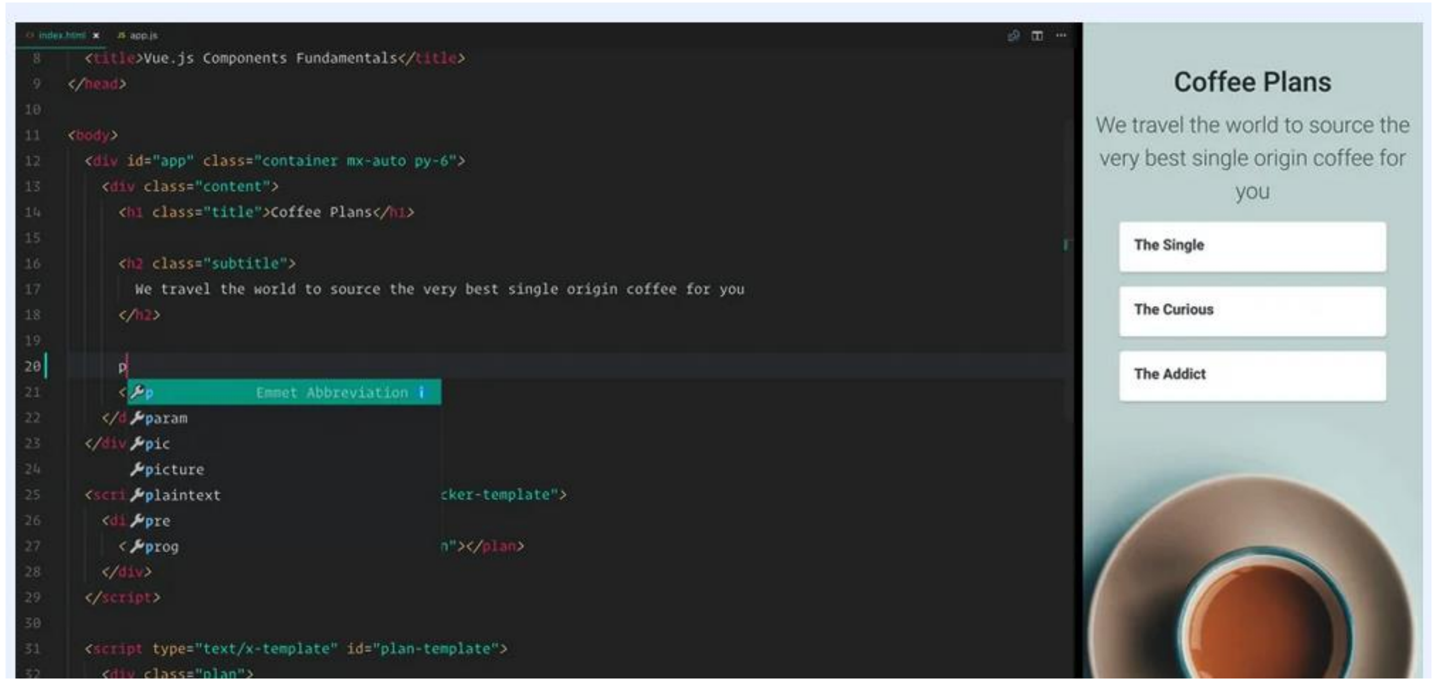
Week 7 – (Registration, Props, Custom Events, Dynamic/Async)

GOALS:

- Gain better understanding of Vue.js Framework
- How Component (Registration, Props, Custom Events, Dynamic/Async) works?
- Learn Coding of new Component concepts.

Watch Vue.js Components Video on Vue School

https://vueschool.io/lessons/global-vs-local-components?friend=vuejs


index.html
app.js
Coffee Plans
We travel the world to source the very best single origin coffee for you
The Single
The Curious
The Addict

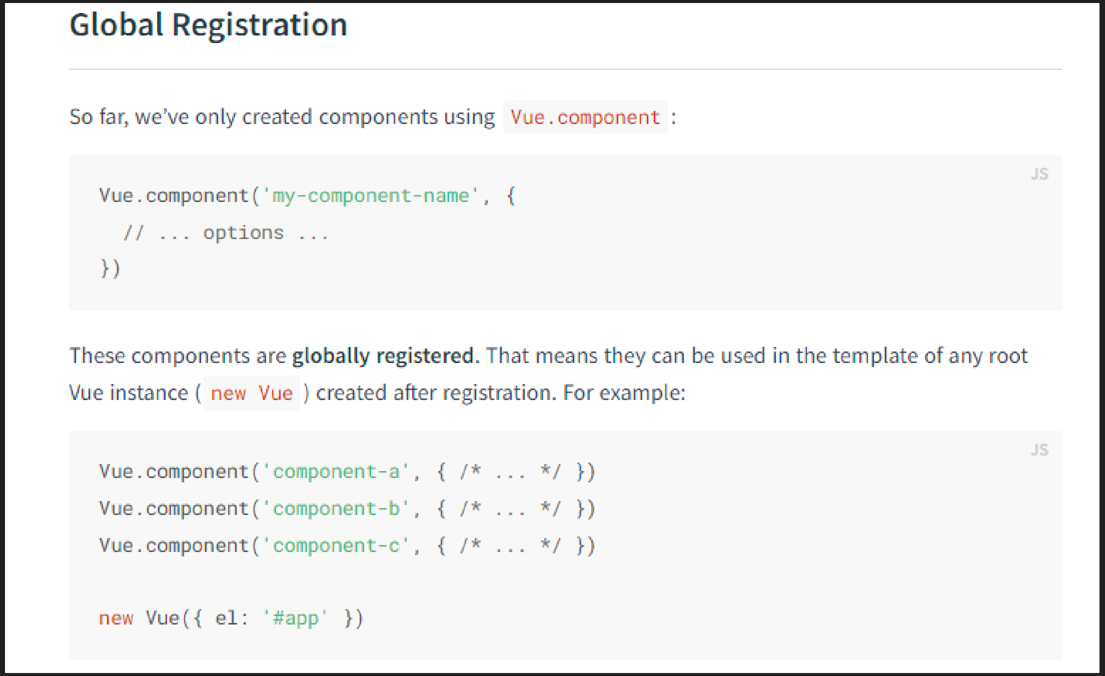
```
    <title>Vue.js Components Fundamentals</title>
  </head>

  <body>
    <div id="app" class="container mx-auto py-6">
      <div class="content">
        <h1 class="title">Coffee Plans</h1>

        <h2 class="subtitle">
          We travel the world to source the very best single origin coffee for you
        </h2>

        p
        <p                  Emmet Abbreviation
      </d param
    </div pic
          picture
    <scri plaintext                       cker-template">
      <di pre
        < prog                            n"></plan>
      </div>
    </script>

    <script type="text/x-template" id="plan-template">
      <div class="plan">
```

See boilerplate example: https://github.com/bencodezen/vue-enterprise-boilerplate/blob/main/src/components/_globals.js

Task 1: Local and Global Registration

(learn the codes with the help of your tutors and implement in your project)

Component Registration — Vue.js (vuejs.org)



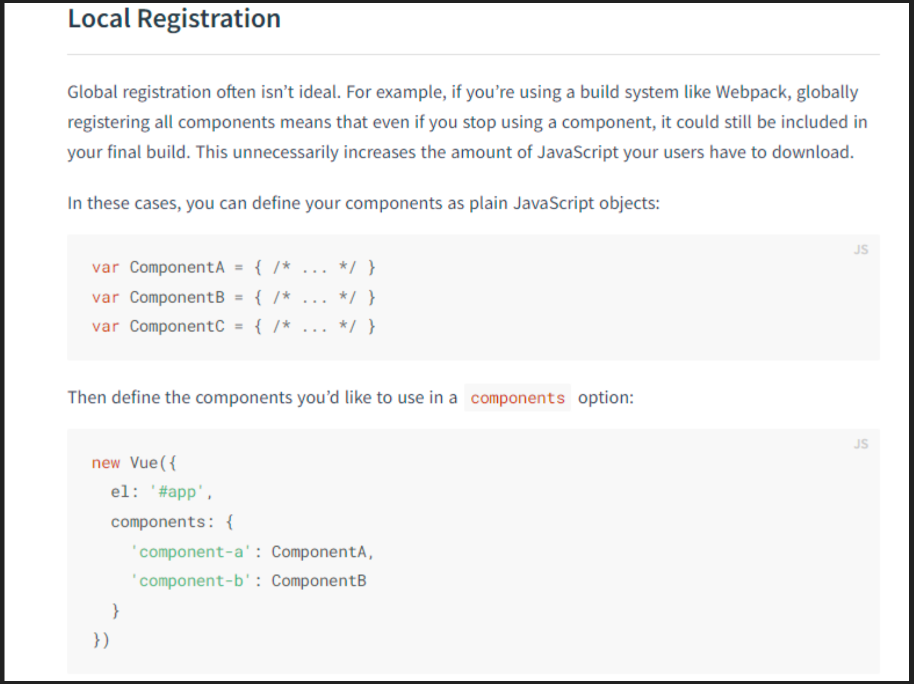
Global Registration

So far, we've only created components using `Vue.component`:

```js
Vue.component('my-component-name', {
  // ... options ...
})
```

These components are **globally registered**. That means they can be used in the template of any root Vue instance (`new Vue`) created after registration. For example:

```js
Vue.component('component-a', { /* ... */ })
Vue.component('component-b', { /* ... */ })
Vue.component('component-c', { /* ... */ })

new Vue({ el: '#app' })
```



Local Registration

Global registration often isn't ideal. For example, if you're using a build system like Webpack, globally registering all components means that even if you stop using a component, it could still be included in your final build. This unnecessarily increases the amount of JavaScript your users have to download.

In these cases, you can define your components as plain JavaScript objects:

```js
var ComponentA = { /* ... */ }
var ComponentB = { /* ... */ }
var ComponentC = { /* ... */ }
```

Then define the components you'd like to use in a `components` option:

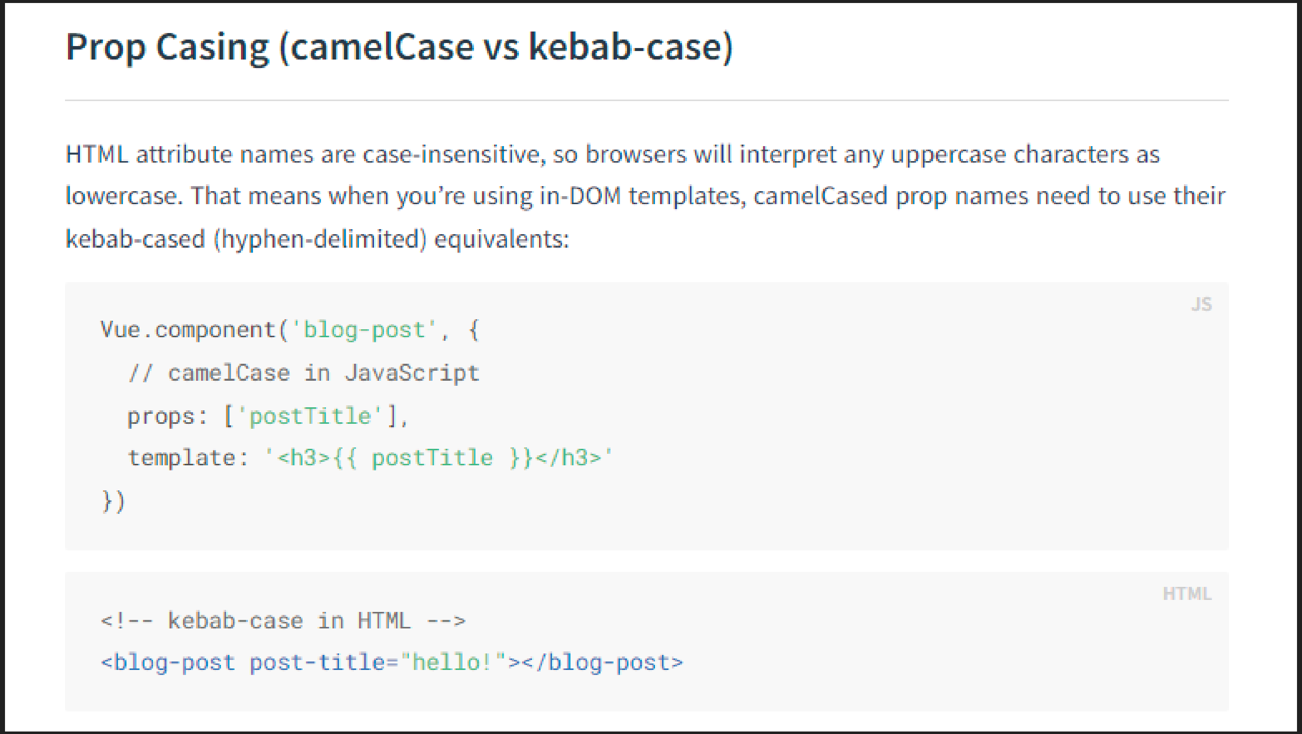
```js
new Vue({
  el: '#app',
  components: {
    'component-a': ComponentA,
    'component-b': ComponentB
  }
})
```

Apply and implement similar concepts in your project

Task 2: Coding Props

(learn the codes of Props with the help of your tutors and implement in your project)

Props — Vue.js (vuejs.org)



Prop Casing (camelCase vs kebab-case)

HTML attribute names are case-insensitive, so browsers will interpret any uppercase characters as lowercase. That means when you're using in-DOM templates, camelCased prop names need to use their kebab-cased (hyphen-delimited) equivalents:

```js
Vue.component('blog-post', {
  // camelCase in JavaScript
  props: ['postTitle'],
  template: '<h3>{{ postTitle }}</h3>'
})
```

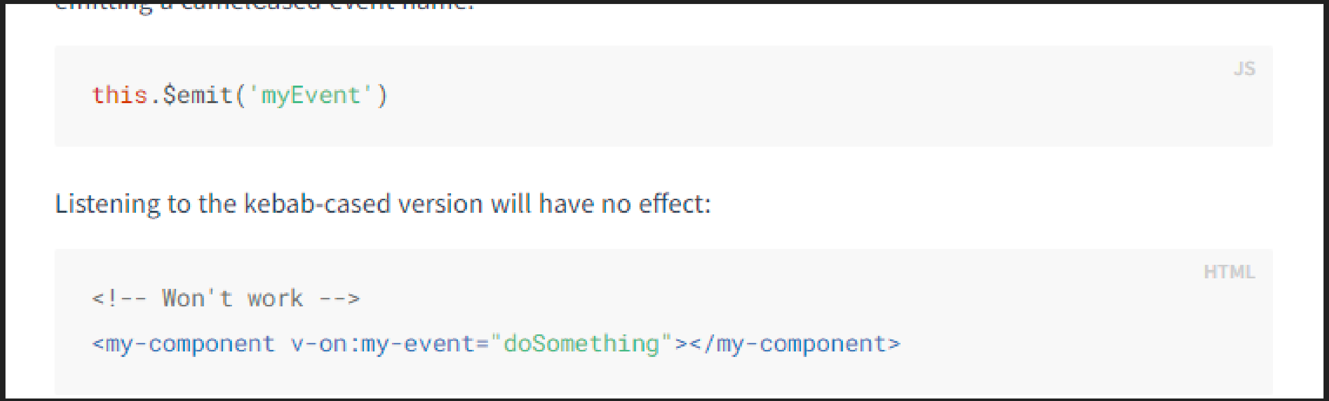
```html
<!-- kebab-case in HTML -->
<blog-post post-title="hello!"></blog-post>
```

How can we pass the properties of an Object?

Apply and implement similar concepts in your project

Task 3: Custom Events

Visit and understand [Custom Events — Vue.js (vuejs.org)]



emitting a camelcased event name:

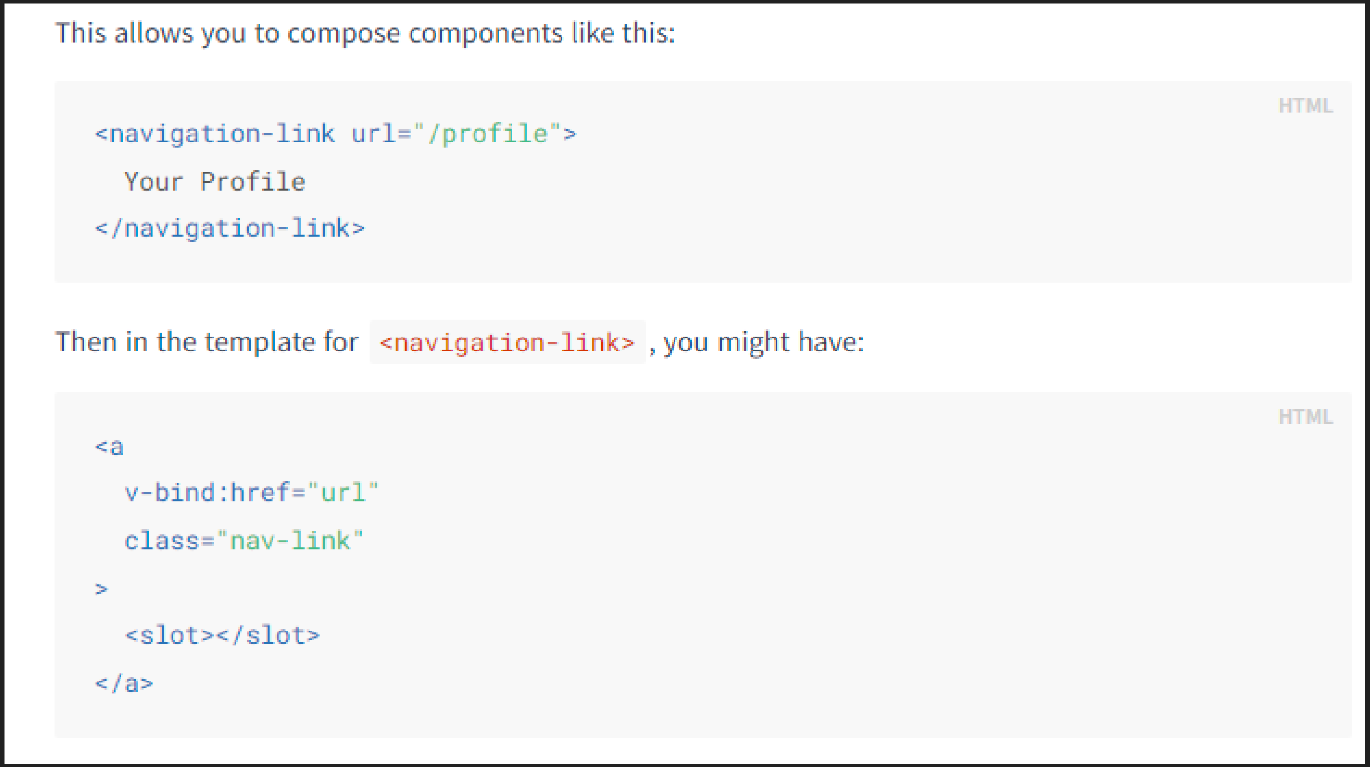
```js
this.$emit('myEvent')
```

Listening to the kebab-cased version will have no effect:

```html
<!-- Won't work -->
<my-component v-on:my-event="doSomething"></my-component>
```

Implement in your project.

Task 4: Vue Slots - coding Named Slots


This allows you to compose components like this:

HTML
```html
<navigation-link url="/profile">
  Your Profile
</navigation-link>
```

Then in the template for `<navigation-link>`, you might have:

HTML
```html
<a
  v-bind:href="url"
  class="nav-link"
>
  <slot></slot>
</a>
```

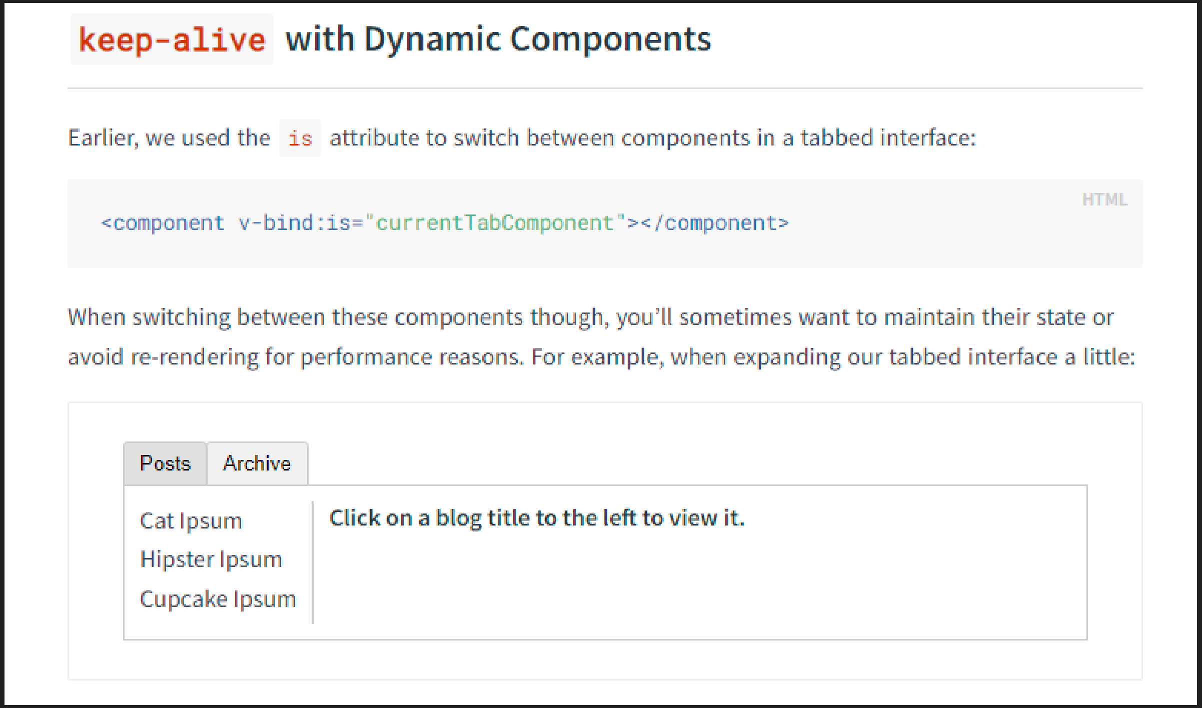
Slots — Vue.js (vuejs.org)

1. Copy the code and execute in your editor

2. Implement similar concept in your project

Before Next Week: Dynamic and Edge cases

Visit code at Dynamic & Async Components — Vue.js (vuejs.org)



keep-alive with Dynamic Components

Earlier, we used the `is` attribute to switch between components in a tabbed interface:

```html
<component v-bind:is="currentTabComponent"></component>
```

When switching between these components though, you'll sometimes want to maintain their state or avoid re-rendering for performance reasons. For example, when expanding our tabbed interface a little:

1. Learn the code with your tutor and execute it in your editor
2. Implement Dynamic Components in your project

Discuss with your tutor:
How can we handle edge cases?