KỸ THUẬTWEB VỚI ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG

Bài 3: ROUTING

NỘI DUNG CHÍNH

I $navigateTo

II $navigateBack

1. $ navigateTo

1. Cú pháp

$navigateTo(Component, options)

Chúng ta có thể gọi $navigateTo trong view hoặc trong methods

1.1 Gọi Trong View

Trong thành phần **Master**, sử dụng một thuộc tính data để hiển thị thành phần **Detail**. Gọi $navigateTo(<propertyName>) xem trực tiếp.

2. Ví dụ

Khi click vào nút trên thành phần Master nó sẽ điều hướng, tức là gọi đến thành phần Detail như sau:

1. $ navigateTo

```
import Vue from 'nativescript-vue';

const Master = {
  template: `
    <Page>
      <ActionBar title="Master" />
      <StackLayout>
        <Button text="To Details directly" @tap="$navigateTo(detailPage)" />
      </StackLayout>
    </Page>
  `,

  data() {
    return {
      detailPage: Detail
    }
  }
};

const Detail = {
  template: `
    <Page>
      <ActionBar title="Detail"/>
      <StackLayout>
        <Label text="Details.." />
      </StackLayout>
    </Page>
  `
};

new Vue({
  render: h => h('frame', [h(Master)])
}).$start()
```

1. $ navigateTo

1.2 Gọi Trong Method

```
const Master = {
  template: `
    <Page>
      <ActionBar title="Master" />
      <StackLayout>
        <Button text="To Details via method" @tap="goToDetailPage" />
      </StackLayout>
    </Page>
  `,

  methods: {
    goToDetailPage() {
      this.$navigateTo(Detail);
    }
  }
};

const Detail = {
  template: `
    <Page>
      <ActionBar title="Detail"/>
      <StackLayout>
        <Label text="Details.." />
      </StackLayout>
    </Page>
  `
};
```

1. $ navigateTo

2. Chuyển props đến thành phần đích

$navigateTo chấp nhận tham số options như sau:

- Thiết lập transition
- Dùng để chuyển 1 props của 1 đối tượng khi khởi tạo 1 component đích
- Ví dụ:

```
this.$navigateTo(Detail, {
  transition: {},
  transitioniOS: {},
  transitionAndroid: {},

  props: {
    foo: 'bar',
  }
});
```

Lưu ý:
Khi đi đến trang đích và kèm theo thuộc tính thì bên trang đích cũng phải nhận đúng thuộc tính đó thông qua props
props: [‘foo’],

1. $ navigateTo

Transition

- curl (same as curlUp) (iOS only)
- curlUp (iOS only)
- curlDown (iOS only)
- explode (Android Lollipop(21) and up only)
- fade
- flip (same as flipRight)
- flipRight
- flipLeft
- slide (same as slideLeft)
- slideLeft
- slideRight
- slideTop
- slideBottom

```
this.$navigateTo(Detail, {
  transition: {
    name: "slideLeft",
    duration: 300,
    curve: "easeIn"
  },
});
```

1. $ navigateTo

3. Điều hướng trong Frame

Mỗi phần tử trong <Frame> có điều hướng riêng. Nếu muốn sử dụng multiple Frame để điều hướng xảy ra.

```
this.$navigateTo(SomeComp, {
  frame: '<id, or ref, or instance>'
});
```

Giá trị trong Frame có thể là

- Các id của thành phần Frame
- Các ref

Ngoài ra còn có $showModal hoạt động như $navigateTo

$ showModal

Chức năng hoạt động giống $navigateTo. Để đóng phương thức ta dùng $modal.close

```
const Master = {
  template: `
    <Page>
      <ActionBar title="Master" />
      <StackLayout>
        <Button text="Show Details modally" @tap="showDetailPageModally" />
      </StackLayout>
    </Page>
  `,

  methods: {
    showDetailPageModally() {
      this.$showModal(Detail);
    }
  }
};

const Detail = {
  template: `
    <Frame>
      <Page>
        <ActionBar title="Detail"/>
        <StackLayout>
          <Button @tap="$modal.close" text="Close" />
        </StackLayout>
      </Page>
    </Frame>
  `
};
```

$ showModal

Tham số thứ hai của $showModal được sử dụng để chuyển props đối tượng của 1 thành phần

```
this.$showModal(Detail, { props: { id: 14 }});
```

```
const Detail = {
  props: ['id'],
  template: `
    <Page>
      <ActionBar title="Detail"/>
      <StackLayout>
        <Label :text="id" />
        <Button @tap="$modal.close" text="Close" />
      </StackLayout>
    </Page>
  `,
};
```

$ showModal

Buộc phương thức ở chế độ toàn màn hình

```
this.$showModal(Detail, { fullscreen: true, props: { id: 14 }});
```

Trả lại dữ liệu từ Modal

Khi gọi $showModal, được trả về để giải quyết với bất kỳ dữ liệu nào được truyền cho hàm $modal.close.

Trong ví dụ sau, đóng các đầu ra phương thức 'Foo' trong bảng điều khiển.

```
// ... inside Master
this.$showModal(Detail).then(data => console.log(data));
```

```
<!-- inside Detail -->
<Button @tap="$modal.close('Foo')" text="Close" />
```

2. $ navigateBack

1. Cú pháp

$navigateBack(options, backstackEntry = null)

2. Ví dụ:

```
const Detail = {
  template: `
    <Page>
      <ActionBar title="Detail"/>
      <StackLayout>
        <Button text="Back to Master" @tap="$navigateBack" />
      </StackLayout>
    </Page>
  `
};
```

3. CREATED VÀ MOUNTED

1. Created

Ngay khi component được tạo, hàm **created** có thể được sử dụng để thao tác với các dữ liệu trong **data** và các sự kiện mà các bạn thiết lập đã có thể được kích hoạt. Nhưng **template** và **DOM** ảo chưa được **mount** và **render**, tức là nếu các bạn truy cập đến các phần tử trong DOM lúc này sẽ không được và báo lỗi. Chúng ta sửa lại file ví dụ như sau:

3. CREATED VÀ MOUNTED

1. Created

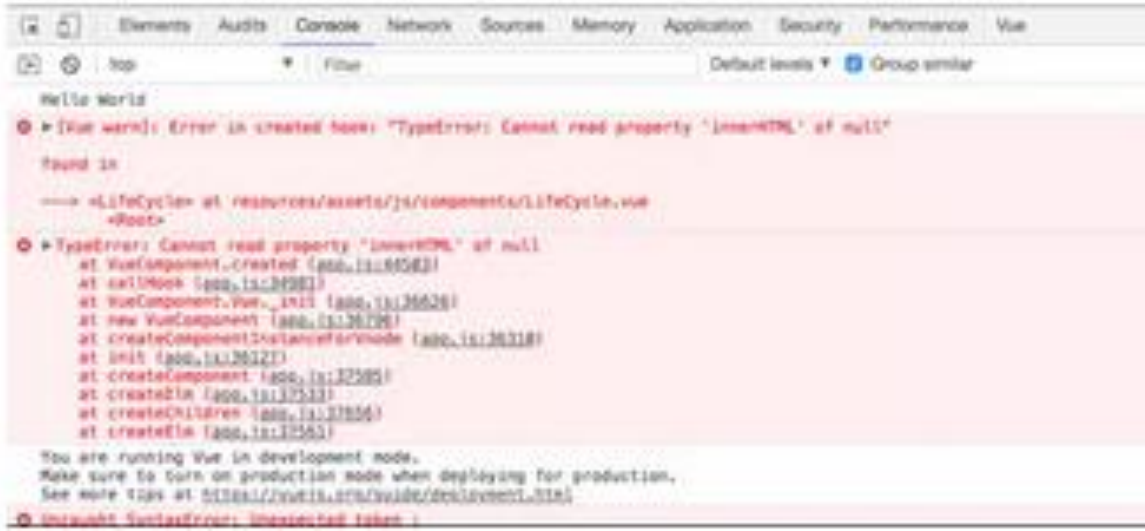
```
<template>
    <div id="my-text">
        This is my text
    </div>
</template>

<script>
    export default {
        data() {
            return {
                message: 'Hello World'
            }
        },
        created() {
            console.log(this.message)

            console.log(document.getElementById('my-text').innerHTML)
        }
    }
</script>

<style lang="scss" scoped>
</style>
```

Khi chạy sẽ báo lỗi sau:



3. CREATED VÀ MOUNTED

2. Mounting

Quá trình mounting xảy ra ngay trước và sau khi component của các bạn được khởi tạo. Thường được sử dụng khi các bạn cần truy cập vào các phần tử trong DOM

2.1 beforeMount

beforeMount được gọi sau khi component đã được compile và trước lần render đầu tiên. Ở giai đoạn này khi các bạn truy cập đến các phần tử trong DOM vẫn sẽ báo lỗi:

3. CREATED VÀ MOUNTED

2.1 beforeMount

```
<template>
    <div id="my-text">
        This is my text
    </div>
</template>

<script>
    export default {
        beforeMount() {
            console.log(this)
            console.log(document.getElementById('my-text').innerHTML)
         }
    }
</script>

<style lang="scss" scoped>
</style>
```

3. CREATED VÀ MOUNTED

2.1 Mounted

Ở quá trình này chúng ta đã có đầy đủ quyền truy cập vào **data**, **template**, DOM (bằng cách gọi this.$el). Mình thường dùng mounted khi dùng chung với Jquery để tác động vào các phần tử DOM, như ở ví dụ bên dưới ta đã có thể truy cập vào các phần tử trong DOM:

```
<template>
    <div id="my-text">
        This is my text
    </div>
</template>

<script>
    export default {
        mounted() {
            console.log(this.$el)
            console.log(document.getElementById('my-text').innerHTML)
         }
    }
</script>

<style lang="scss" scoped>
</style>
```


the
end