Navegación entre componentes

Componentes

En el contexto de un proyecto web que utiliza múltiples componentes, se explorará cómo simular la navegación mostrando **un componente a la vez.**

Conceptos clave

Componentes

Unidades modulares de la interfaz de usuario.

Router

Mecanismo para gestionar la **navegación entre componentes.**

Modelo

Representación de los datos que se manejan en la aplicación.

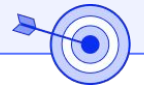
Clase *Router*

Se creará la clase que facilitará la **navegación fluida** entre diferentes componentes, dentro de la aplicación web. Esto permitirá una **gestión eficiente de las vistas y la lógica** asociada.

Características

- **Constructor versátil:** El constructor de `Router` acepta un elemento `parent` donde se renderizan los componentes y un `modelo` que puede contener datos compartidos entre componentes.
- **Método `navigateTo`**: Este método inicia la navegación hacia un nuevo componente especificado por `Componente` y opcionalmente un `model` adicional. Crea una instancia del componente proporcionado y lo renderiza en el `parent` definido en el constructor.
- **Flexibilidad de modelos:** La clase permite pasar modelos diferentes a cada componente. Esto facilita la personalización de datos según la vista requerida.

El objetivo principal de la clase `Router` es **facilitar la transición** de un componente a otro y **asegurar que el nuevo componente se renderice** correctamente en el contenedor adecuado.



Ejemplo:

```
class Router {
    constructor(parent, model){
        this.parent = parent;
        this.model = model;
    }

    navigateTo(Componente, model){
        let componente = new Componente(this.parent, this.model, this);
        componente.render();
    }
}
```

Clase *ApplicationComponent*

La clase **`ApplicationComponent`** actúa como el punto de entrada principal de la aplicación. Se encarga de **inicializar el entorno y gestionar el inicio de la navegación** hacia el primer componente de la aplicación.

```
class ApplicationComponent {
    constructor(parent){
        this.parent = parent;
        this.model =  new Model();
        this.model.crearCliente("Juan",
        this.router = new Router(this.parent, this.model)
        this.router.navigateTo(LoginComponent);
    }
}
```

Ejemplo de uso

Este ejemplo muestra cómo integrar las clases **`Router`** y **`ApplicationComponent`**, en una página HTML simple. La estructura del documento HTML incluye la **carga de *scripts* necesarios y la inicialización de la aplicación** una vez que la página se ha cargado completamente.

Este enfoque asegura que la aplicación se cargue y se inicie correctamente, permitiendo una **navegación fluida entre componentes** gracias al uso de **`Router`** y **`ApplicationComponent`**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="./modelo.js"></script>
    <script src="./componentes/Router.js"></script>
    <script src="./componentes/Applicacion.js"></script>


    <script>

        window.addEventListener("load", ()=>{
            let root = document.getElementById("root");
            let app = new ApplicationComponent(root);
        });
    </script>
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
    <div id="root">
</body>
</html>
```

Navegación entre componentes

Al desarrollar aplicaciones con componentes, es fundamental considerar cómo estos **interactúan entre sí** y cómo los **usuarios navegan** entre ellos.

- **Organización modular:** Los componentes permiten dividir la interfaz de usuario en partes más pequeñas y manejables. Esto facilita el desarrollo, la depuración y el mantenimiento del código.
- **Navegación estructurada:** Definir cómo los usuarios pueden moverse de un componente a otro garantiza una experiencia fluida y coherente dentro de la aplicación.
- **Componente principal:** Es común tener un componente principal que sirve como punto de entrada a la aplicación y gestiona la navegación inicial.

Ejemplo

En este código, se muestra cómo se puede utilizar un **componente de demostración** (**`DemoComponent`**) para gestionar la navegación dentro de una aplicación web.

El componente tiene un botón que, al hacer clic, utiliza el **`Router`** para navegar de regreso al **componente de inicio de sesión** (**`LoginComponent`**).

```
class DemoComponent {
    constructor(parent, model, router){
        this.parent = parent;
        this.model = model;
        this.router = router;
    }

    handleSalir(){
        this.router.navigateTo(LoginComponent);
    }

    render(){
        this.parent.innerHTML = `<button id="cuenta_btnSalir" class="btn transactions">Salir</button>
                </div>
            </div>
        `
        this.parent.querySelector("#cuenta_btnSalir").addEventListener("click",()=>{this.handleSalir()})
    }
}
```