

Voy a mostrarte cómo podrías modificar tu función en includes.js para que admita múltiples componentes y, al mismo tiempo, notifique a cada script asociado mediante eventos personalizados cuando su componente respectivo ya esté cargado. De este modo, cada script (por ejemplo, para el navbar o para el formulario de contacto) puede esperar a que su HTML se haya inyectado en el DOM antes de inicializar su funcionalidad.

Supongamos que tu estructura es la siguiente:

CLUB DE FACILITACION/

└─ frontend/

├─ index.html

└─ src/

├─ components/

│ ├─ navbar.html

│ └─ contactos.html

├─ js/

│ ├─ includes.js

│ ├─ menu-desplegable.js

│ └─ form-contactos.js

└─ css/

├─ styles-nav.css

└─ styles-contactos.css

**1. Modificar includes.js para cargar múltiples componentes**

Puedes generalizar la función de carga para que, además de inyectar el HTML en el contenedor correspondiente, despache un evento personalizado que identifique que ese componente ya está listo. Por ejemplo:

javascript

const loadHTMLComponent = (containerId, file, eventName) => {

fetch(file)

.then(response => {

if (!response.ok) {

throw new Error(`Error: ${response.status}`);

}

return response.text();

})

.then(data => {

const container = document.getElementById(containerId);

container.innerHTML = data;

// Despacha un evento personalizado que indica que el componente ya se cargó

document.dispatchEvent(new Event(eventName));

})

.catch(error => console.error(`Error al cargar ${file}:`, error));

};

window.addEventListener('DOMContentLoaded', () => {

// Cargar el navbar

loadHTMLComponent("container-nav", "src/components/navbar.html", "navbarLoaded");

// Cargar el formulario de contactos

loadHTMLComponent("container-contact", "src/components/contactos.html", "contactosLoaded");

});

Con este código, tan pronto inyectes cada componente en su contenedor (definido en el index.html), se manda a disparar el evento personalizado correspondiente.

**2. Ajustar tu index.html**

En tu index.html incluyes los contenedores para cada componente y vinculas los scripts (incluyendo el includes.js y los scripts propios de cada componente):

html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Mi Proyecto Modular</title>

<link rel="stylesheet" href="src/css/styles-nav.css">

<link rel="stylesheet" href="src/css/styles-contactos.css">

</head>

<body>

<!-- Contenedores para los componentes -->

<div id="container-nav"></div>

<div id="container-contact"></div>

<!-- Carga conjunta de los scripts -->

<script src="src/js/includes.js"></script>

<!-- Estos scripts se cargan de forma estática; su inicialización se hará cuando reciban cada evento -->

<script src="src/js/menu-desplegable.js" defer></script>

<script src="src/js/form-contactos.js" defer></script>

</body>

</html>

**3. Inicializar cada script mediante su respectivo evento personalizado**

Dentro de cada script asociado al componente, en lugar de depender solo de DOMContentLoaded, escuchas el evento que indica que el componente ya se ha cargado.

**Ejemplo para el menú desplegable (navbar):**

javascript

function initMenuDesplegable() {

const menuItems = document.querySelectorAll('#container-nav .menu-item');

// ... lógica de inicialización del menú ...

console.log("Menú desplegable inicializado");

}

// Escuchar cuando se haya cargado el navbar

document.addEventListener("navbarLoaded", initMenuDesplegable);

**Ejemplo para el formulario de contacto:**

javascript

function initFormContactos() {

const form = document.querySelector('#container-contact form');

if (!form) {

console.error("No se encontró el formulario de contacto");

return;

}

form.addEventListener('submit', (event) => {

event.preventDefault();

// Lógica para el envío de información del formulario

console.log("Formulario de contacto enviado");

});

console.log("Formulario de contacto inicializado");

}

// Escuchar cuando se haya cargado el componente de contactos

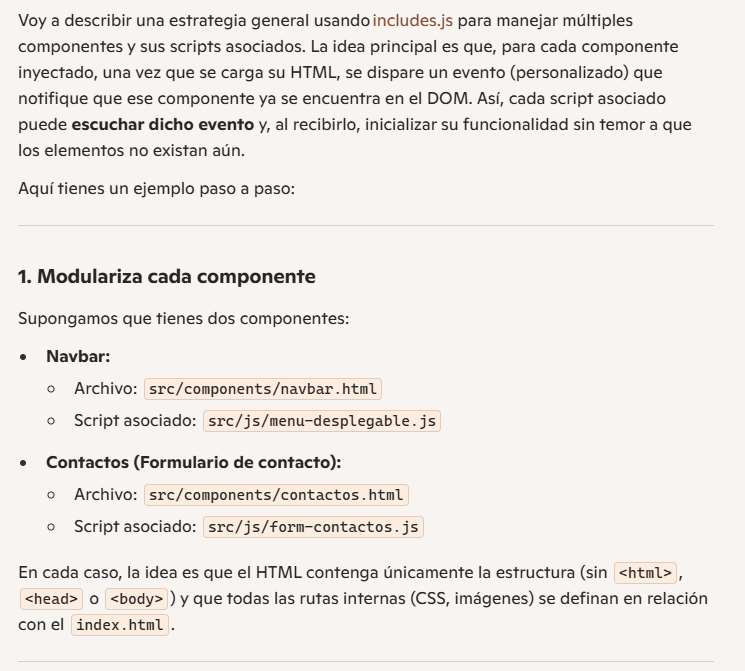
document.addEventListener("contactosLoaded", initFormContactos);

**Resumen**

1. **En** includes.js**:**
   * Se define la función loadHTMLComponent que inyecta el HTML en el contenedor indicado y, al finalizar, despacha un evento personalizado (como "navbarLoaded" o "contactosLoaded").
2. **En** index.html**:**
   * Se incluyen contenedores con identificadores para cada componente.
   * Se vinculan los scripts principales (incluyendo includes.js y cada script asociado) de forma estática.
3. **En los scripts de cada componente:**
   * Se escucha el evento personalizado para saber cuándo inicializar la funcionalidad del componente, asegurándose de que su HTML ya esté disponible en el DOM.

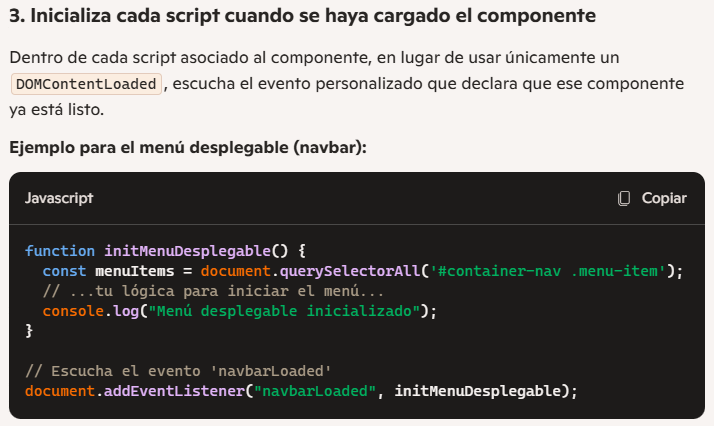
Con este enfoque, puedes escalar fácilmente y agregar más componentes y sus respectivos scripts sin preocuparte de que el código se ejecute antes de que el HTML se haya inyectado. Cada script queda vinculado a su evento personalizado, lo que garantiza una inicialización ordenada.

¿Te queda alguna duda o deseas profundizar en algún otro aspecto? Estoy aquí para ayudarte.

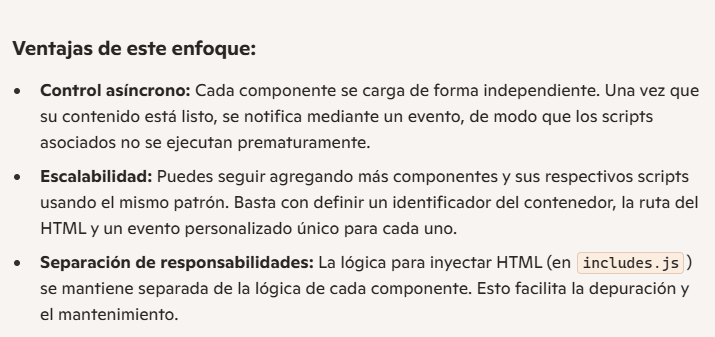


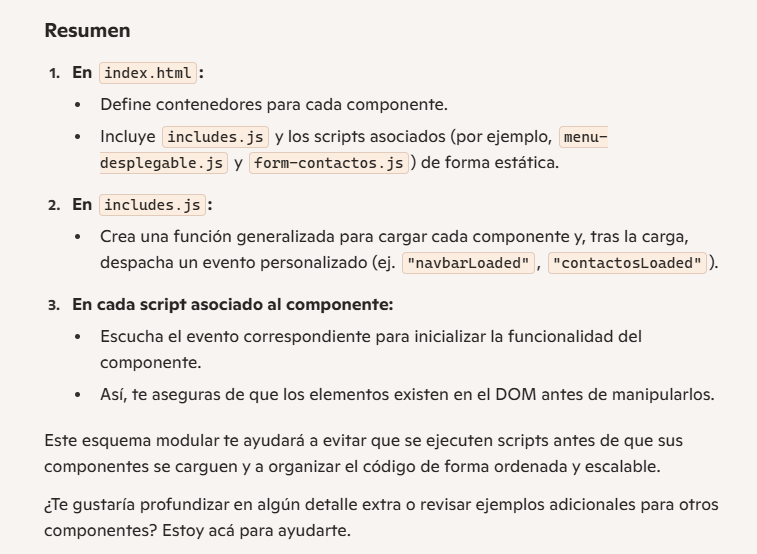












Modificar el includes.js para que funcione en cualquier componente

El problema es que al renderizarse la ruta se pone en función del lugar donde se renderiza el componente. Entonces podemos usar dos opciones, poner la ruta absoluta del componente, así sin importar donde nos encontremos vamos a poder acceder al componente. Aunque si cambiamos la organización de carpetas vamos a tener que modificar estas rutas. La otra opción es obtener la ruta actual con código y modificarle de a cuerdo nos conviene



