Taller de Creación de Router para Single Page App con JavaScript

# La lógica detrás de nuestro enrutador

Los pasos que se van a seguir para llevar a cabo la creación del Router del lado del cliente para SPA son:

*Cargar la ruta*  
Identificar en donde nos encontramos en el sitio. (Carga inicial de la ruta).

* loadInitialRoute()  
  .

*Comparar la URL con una ruta*  
La URL a la que nos queremos mover, se debe comparar con las rutas que tenemos.

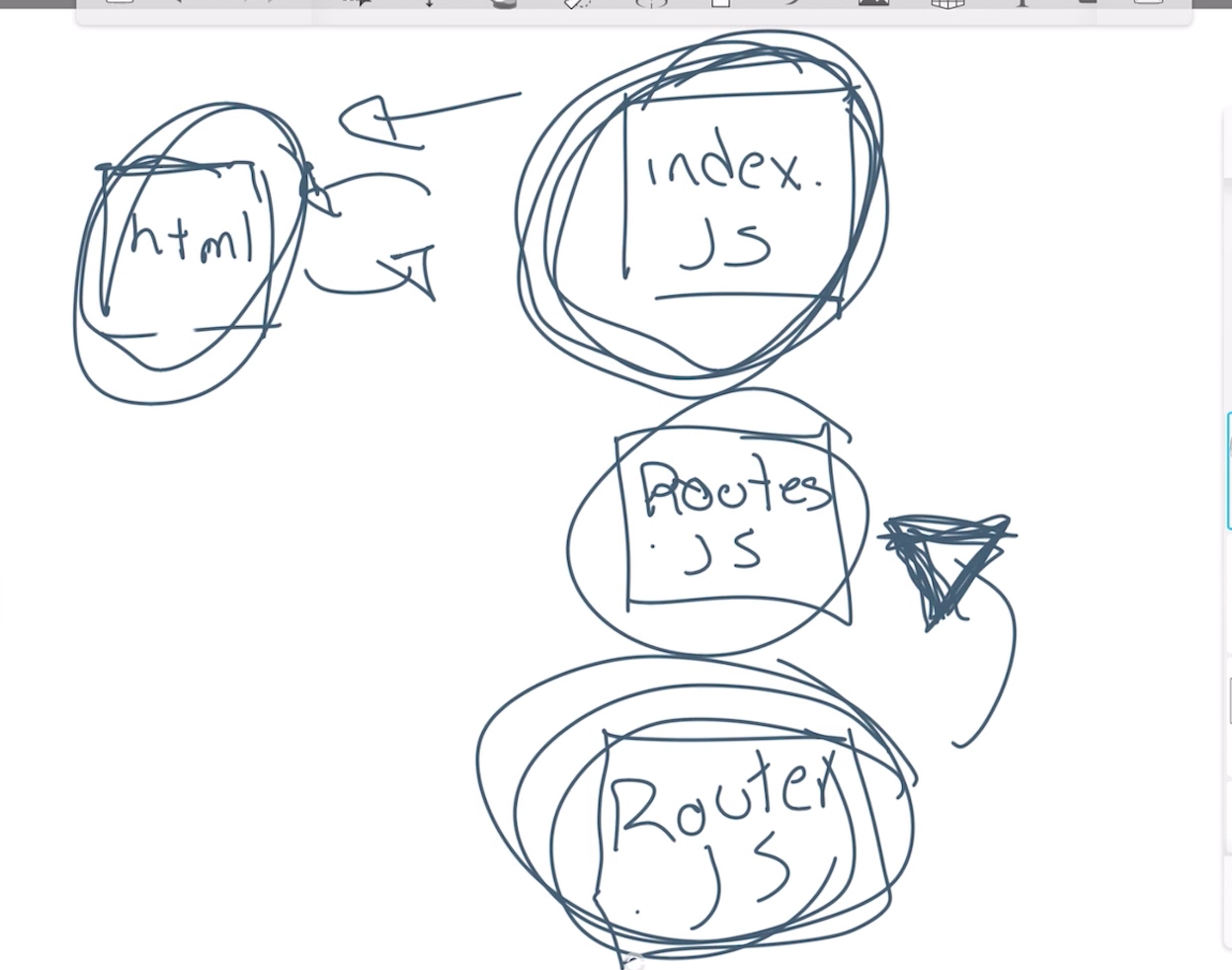
* matchUrlToRoute()  
  .

*Actualizar la URL en la barra de navegación*  
Para esto utilizaremos el método de HTML pushState. (windows.history.pushState).

* loadRoute()  
  .

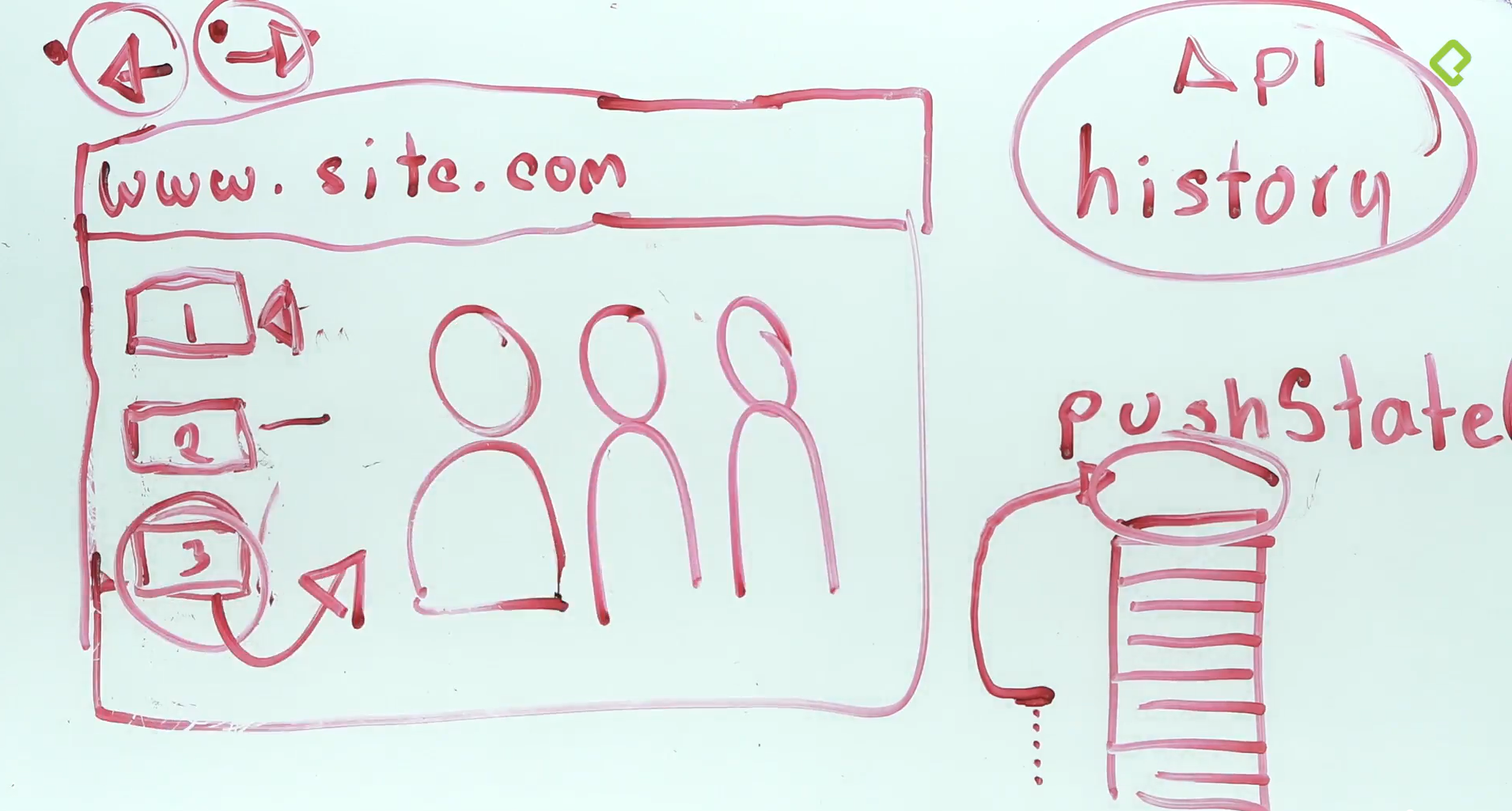
*Actualizar el DOM con el nuevo contenido*  
Para esto vamos a usar innerHTML.

* loadRoute()



Conceptos de SPA Routing y nuestro primer servidor

# Desglose del proyecto del curso y explicación del SPA Routing



# Implementando routing del lado del cliente

**ALGUNOS MÉTODOS DE HISTORY**

**window**.history.back();

Nos permite ir un paso hacia atrás en nuestro historial

**window**.history.forward();

Lo opuesto a back(), nos permite movernos hacia adelante en nuestro historial

**window**.history.go(**n**);

Nos permite movernos n páginas.  
Por ejemplo, si **n=-1** nos iremos *una página atrás*. Por otro lado, si **n=1**, iremos *una página hacia adelante*. Y si **n=0**, se *recargará la página actual*.

**window**.history.length;

Nos devolverá el número de páginas en nuestro stack de historial

**pushState**(estado,titulo,url)

Se encarga de añadir una entrada al stack de nuestro historial.  
**Estado** es el objeto al cual esta asociado con la nueva entrada al historial creada.  
**Título**, de momento es ignorado la mayoría de las veces.  
**Url** es la ruta que será añadida a la nueva entrada de nuestro historial

**replaceState**(estado,titulo,url)

Trabaja de la misma manera que **pushState()** Con la diferencia que modifica una entrada, en vez de crear una nueva.

history.**state**;

Nos da el estado actual de nuestro historial

Fuente: <https://developer.mozilla.org/es/docs/DOM/Manipulando_el_historial_del_navegador>

**window**.history.pushState({data:'Movimiento'},'Titulo','/path')

**El método pushState()**  
pushState()toma tres parámetros: un objeto estado, un título (el cual es normalmente ignorado) y (opcionalmente) una URL. Vamos a examinar cada uno de estos tres parametros en más detalle:

**##Object estado**  
El objeto estado es un objeto JavaScript el cual esta asociado con la nueva entrada al historial creada por pushState(). Cada vez que el usuario navega hacia un nuevo estado, un evento popstate event se dispara, y la propiedad state del evento contiene una copia del historial de entradas del objeto estado.

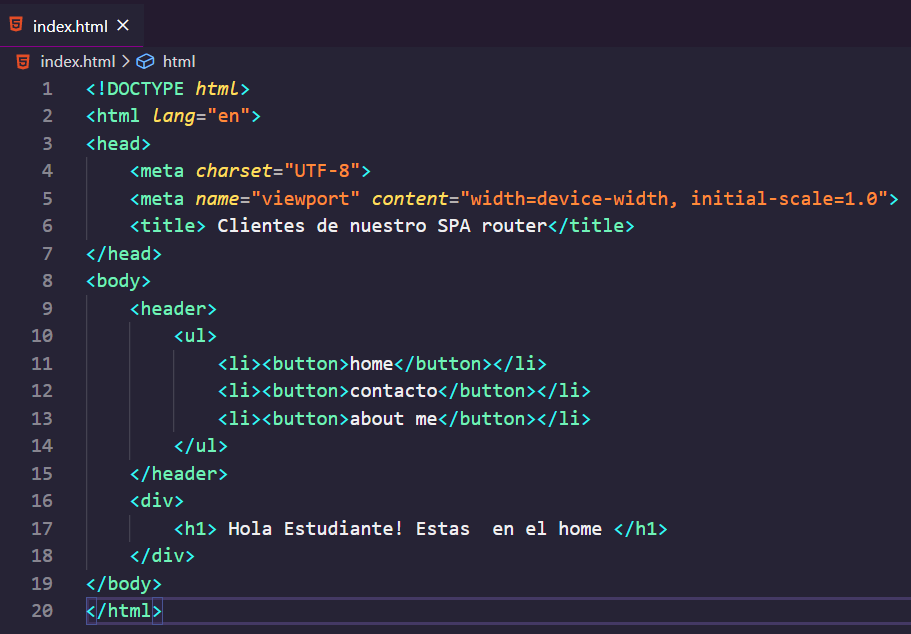
El objeto estado puede ser cualquier cosa que puedas pasar a JSON.stringify. Dado que Firefox guarda los objetos estado en el disco del usuario para que puedan ser restaurados después de que el usuario reinicie su navegador, se ha impuesto un tamaño límite de 640K caracteres en representación JSON de un objeto estado. Si pasas un objeto estado cuya representación es más larga que esto a pushState(), el método arrojará una excepción. Si necesitas más espacio, se recomienda usar sessionStorage y/o localStorage.

**##Título**  
Firefox actualmente ignora este parámetro, aunque podría usarse en el futuro. Pasar una cadena de caracteres vacia aquí podría asegurar estar a salvo de futuros cambios en este método. Alternativamente podrías pasar un título corto del estado hacia el cual te estás moviendo.

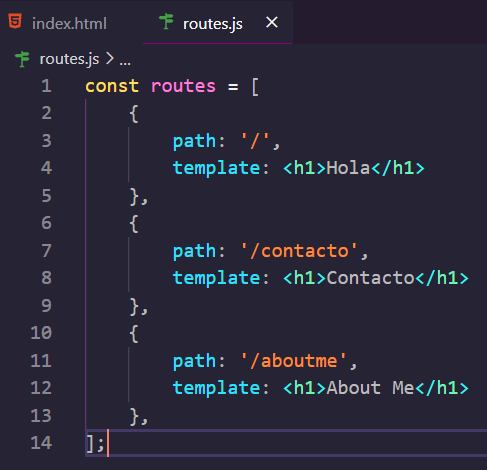
**##URL**  
La URL de la nueva entrada al historial está dada por este parámetro. Recuerda que el browser no intentará cargar esta URL después de llamar a pushState(), pero podría intentar cargar la URL más tarde, por ejemplo, después de que el usuario reinicie su navegador. La nueva URL no necesita ser absoluta; si es relativa, es resuelta relativamente a la actual URL. La nueva URL debe ser del mismo origen que la actual URL. Si no es así, pushState() arrojará una excepción. Este parámetro es opcional; si no se especifica, se tomará la URL actual del documento.

Puedes conseguir la guia completa [Aqui!](https://developer.mozilla.org/es/docs/DOM/Manipulando_el_historial_del_navegador" \t "_blank)

# Implementando routing del lado del cliente

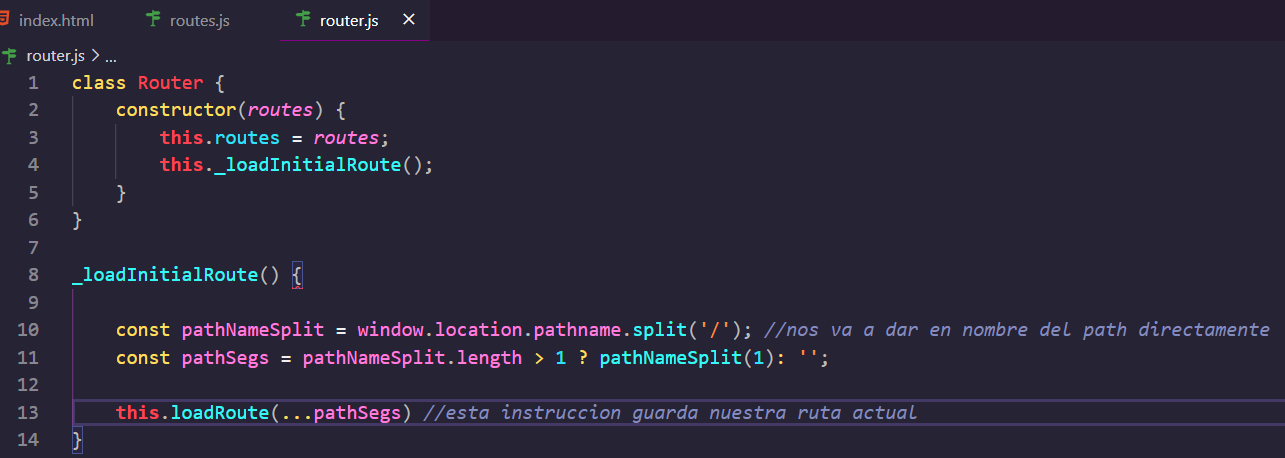


# Creando nuestro archivo de rutas

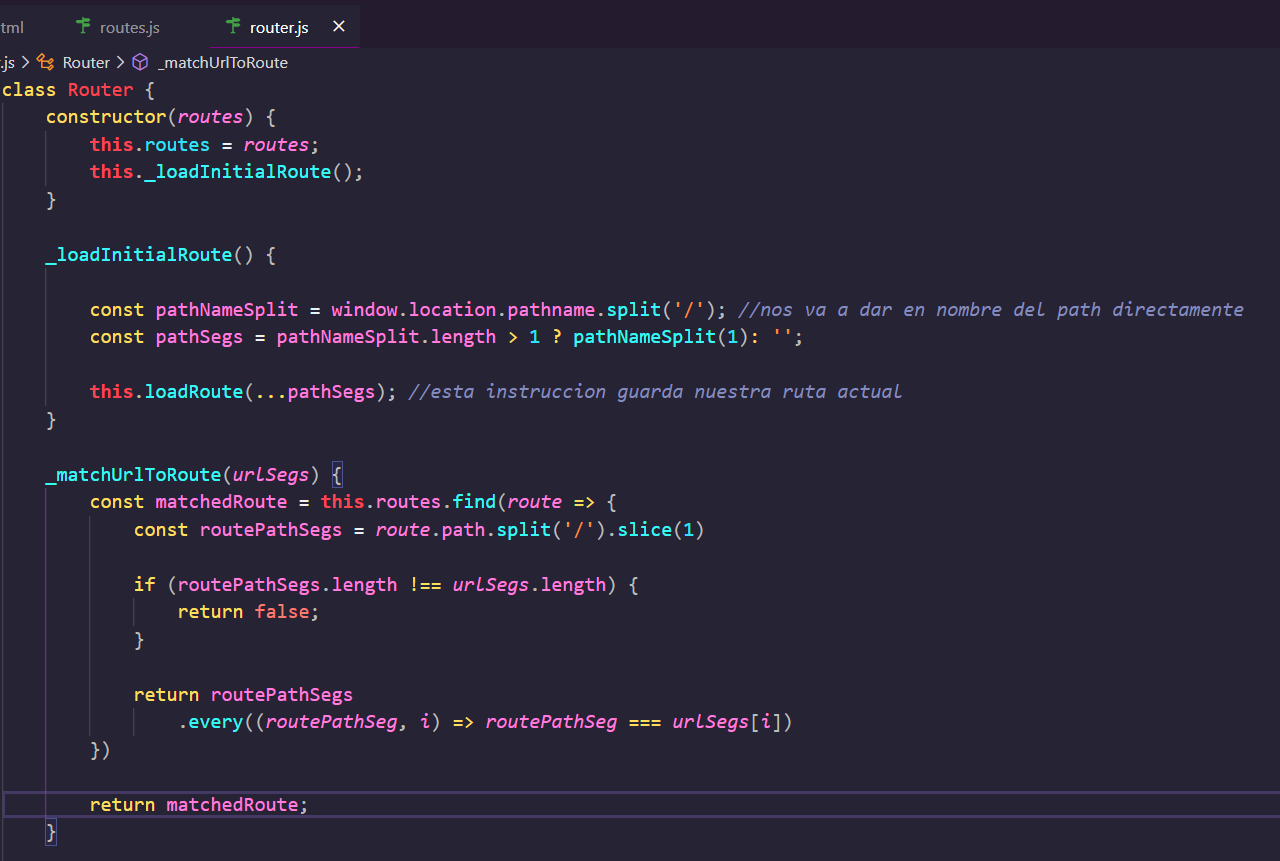


Lógica de nuestro ruteador

# Creación de la función loadInitialRoute



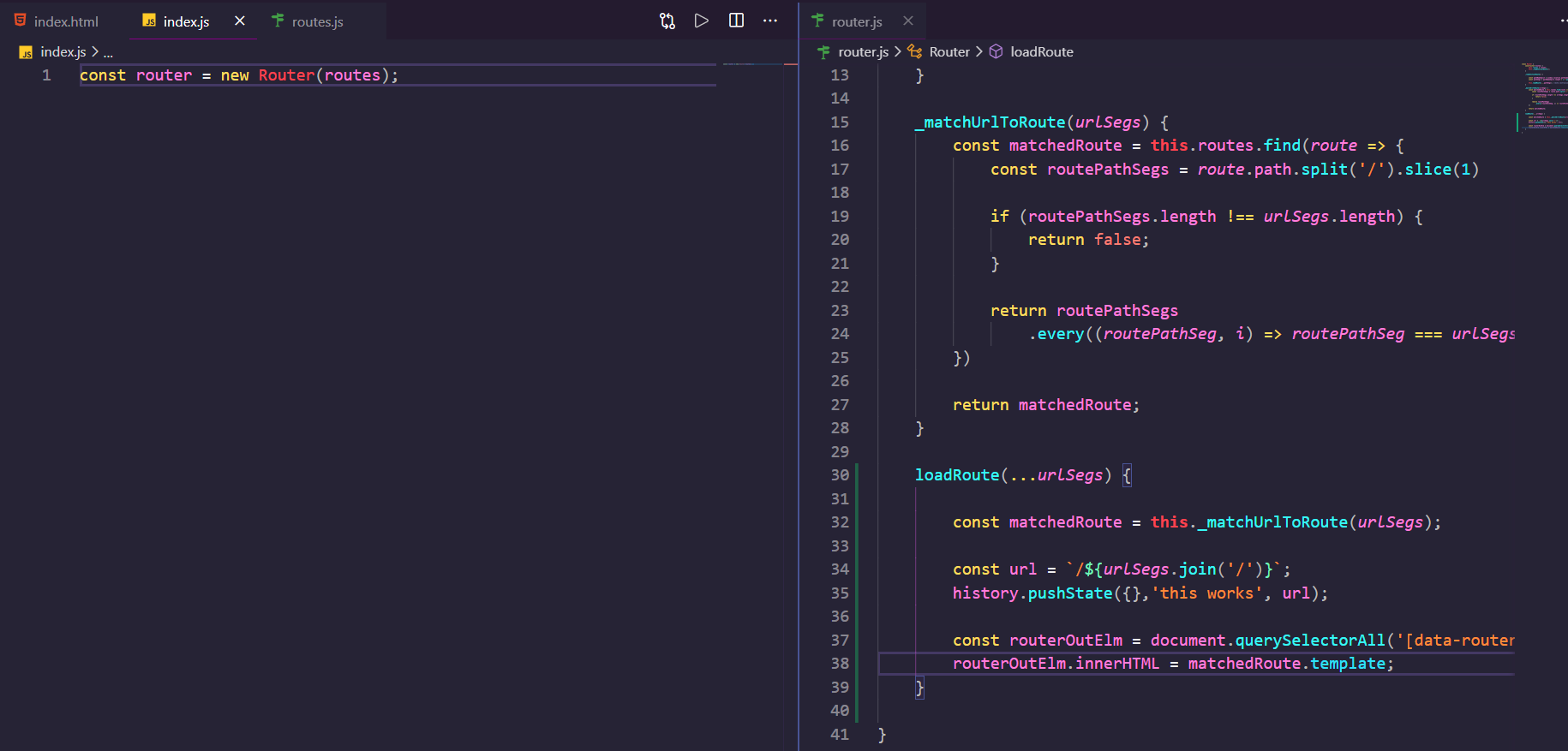
# Haciendo match entre la URL y una ruta



# Creando la función load routes

Hola, personalmente me cuesta seguir este curso no porque no entienda lo que se está haciendo, sino por la metodología de enseñanza. Creo que sería más fácil explicar:

1)Colocamos [data-router] en el elemento <div data-router>. 2)Creamos la constante “routerOutElm” […]. etc. Y no viceversa.



# Actualizando nuestro index.html para agregar la nueva funcionalidad de routing

Detalle importante: Eliminar /index.html del navegador (127.0.0.1:5500/index.html)

