Desarrollo de Aplicaciones Web

Tema Nº12:REACT JS – NAVEGABILIDAD I

Indicador de logro Nº12:Construye interfaces de usuario por medio de la librería de JavaScript a través de React JS desarrollada por Facebook para optimizar la perfomance de las aplicaciones web.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº12:**

REACT JS – NAVEGABILIDAD I

**Subtema 12.1:**

¿Qué es navegabilidad web?

Navegabilidad web es un término que se refiere a qué tan sencillo o complicado resulta para un usuario desplazarse y ubicarse entre las distintas páginas y elementos de un sitio web.

Recurriendo a la analogía que le da nombre al término, puede imaginarse que el sitio es una especie de océano y el usuario es el barco que navega en él. Sin importar el tamaño ni la cantidad de información del sitio, para que exista navegabilidad el internauta debe tener herramientas suficientes para determinar:

* En qué parte del sitio se encuentra
* De qué parte o partes del sitio viene
* Cómo llegar a la información a la que quiere ir

**Subtema 12.2:**

¿Para qué serve la navegabilidad web?

La navegabilidad web, junto con la usabilidad y la accesibilidad son tres criterios fundamentales en un sitio, pues están enfocados a mejorar la experiencia del usuario.

Mientras más contenido y diferentes categorías haya en un sitio web, más y más importante se vuelven las estrategias de navegabilidad para que el usuario no se “pierda” ni se desespere tratando de acceder a la información.

**Subtema 12.3:**

¿De qué depende la navegabilidad web?

Básicamente, la navegabilidad web depende de cómo esté organizado, categorizado y dividido el contenido de un sitio; a esto se le llama arquitectura de la información. Debe de ser sencillo e intuitivo localizar toda la información y los recursos disponibles, así como identificar las rutas y los atajos de navegación.

Normalmente, los sitios web resuelven la necesidad de navegación del usuario mediante el uso de menús, íconos y mapas.

**Subtema 12.4:**

¿Cuál es la diferencia entre navegabilidad, usabilidad y accesibilidad?

Mientras que la navegabilidad se refiere a la facilidad para desplazarse de manera voluntaria entre los contenidos, la usabilidad tiene más que ver con la claridad, simpleza y estilo de los componentes.

La accesibilidad, por su parte, tiene que ver con contenidos que sean accesibles para todas las personas, independientemente de que padezcan alguna discapacidad (visual, auditiva, intelectual, etc) o se enfrenten a circunstancias técnicas especiales.

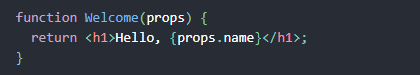
**Subtema 12.5:**

Componentes y propiedades

Los componentes permiten separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada. Esta página proporciona una introducción a la idea de los componentes. Puedes encontrar una API detallada sobre componentes aquí. Conceptualmente, los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas “props”) y devuelven a React elementos que describen lo que debe aparecer en la pantalla.

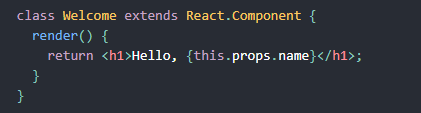
**Componentes funcionales y de clase**

La forma más sencilla de definir un componente es escribir una función de JavaScript:



Esta función es un componente de React válido porque acepta un solo argumento de objeto “props” (que proviene de propiedades) con datos y devuelve un elemento de React. Llamamos a dichos componentes “funcionales” porque literalmente son funciones JavaScript.

También puedes utilizar una clase de ES6 para definir un componente:



Los dos componentes anteriores son equivalentes desde el punto de vista de React. Tanto los componentes de función como de clase tienen algunas características adicionales que se verán en las próximas secciones.

**Renderizando un componente**

Anteriormente, sólo encontramos elementos de React que representan las etiquetas del DOM:



Sin embargo, los elementos también pueden representar componentes definidos por el usuario:



Cuando React ve un elemento representando un componente definido por el usuario, pasa atributos JSX e hijos a este componente como un solo objeto. Llamamos a este objeto “props”. Por ejemplo, este código muestra “Hello, Sara” en la página:





Recapitulemos lo que sucede en este ejemplo:

1. Llamamos a **ReactDOM.render()** con el elemento **<Welcome name="Sara" />**.
2. React llama al componente **Welcome** con **{name: 'Sara'}** como “props”.
3. Nuestro componente **Welcome** devuelve un elemento **<h1>Hello, Sara</h1>** como resultado.
4. React DOM actualiza eficientemente el DOM para que coincida con **<h1>Hello, Sara</h1>**.

**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* Visual Stude Code
* Node.js on WSL 2
* Npm
* Windows 10
* JavaScript

**2. PROCEDIMIENTO**

Navegación con React JS

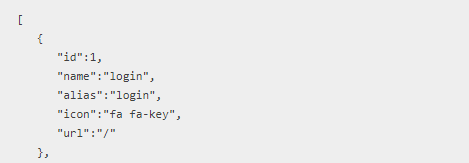
En esta entrada os quiero contar como podéis realizar una simple navegación en react js. Tendremos en cuenta objetivos como:

* Mostrar menú importado desde un file.json
* Estructura JSX para el contenido
* Estructura de las páginas a mostrar internas y externas
* Aplicar clase active
* Librería de iconos
* Estilos de css en línea y externo

Veremos todos estos pasos uno a uno y en el orden que hemos indicado.

**1.Mostrar menú importado desde un file.json**

En un fichero.json tendremos los elementos de nuestro menú con todas las características necesarias que queramos visualizar, veamos el código:

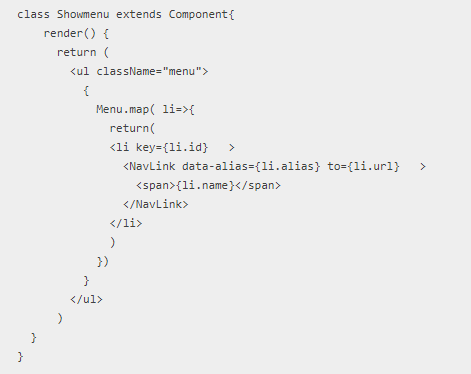




En este código podemos ver como el fichero json con el que vamos a trabajar tiene tres elementos principales con diferentes características como el id, nombre, alias, icono, ruta … Para poder trabajar con el fichero debemos importarlo en nuestro proyecto de la siguiente forma:



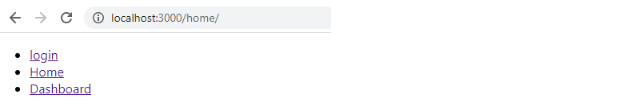
Una vez importado ya podemos hacer uso de ello ahora vamos a recorrer el fichero para poder listar los elementos de menú.



Usamos la función .map() para el tomar array que nos llegará a través del import del fichero.json.

**2. Estructura JSX para el contenido**

En el código anterior vemos como hemos incluido elementos de jsx necesarios para listar el menú. Actualmente este es el resultado y código de nuestro proyecto.



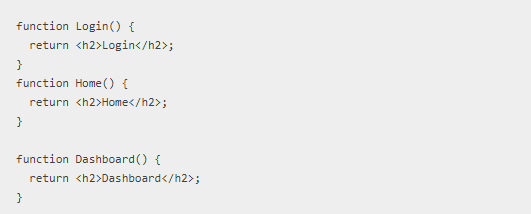


Mostramos a través de una lista el contenido del fichero json. La estructura del menú la tenemos guardada en una clase llamada Showmenú la cual estamos llamando en la función principal que va mostrar todo el contenido de nuestro proyecto.

A continuación, veremos también la estructura necesaria para completar el proceso de navegación que será el resultado que se visualice en los diferentes enlaces del menú

**3. Estructura de las páginas a mostrar internas y externas**

Para poder mostrar dicho resultado crearemos diferentes funciones, cada una de ellas contiene el resultado a mostrar cuando se selecciona esa opción, este sería el código.

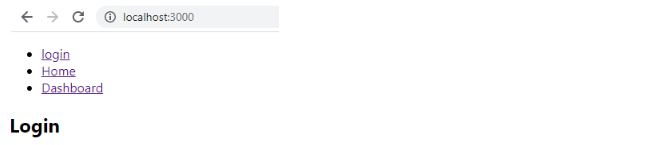


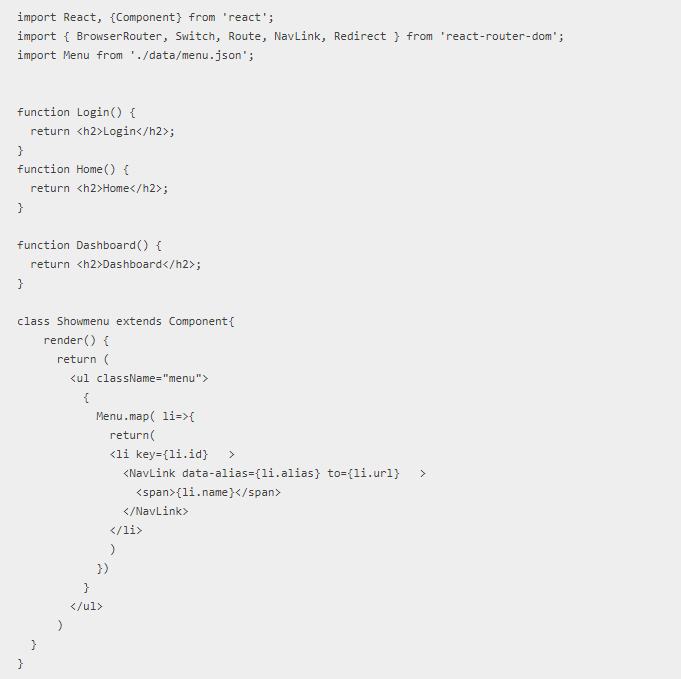
Dentro de cada función introduciremos el código que consideremos necesario, puede ser tan extenso como queráis eso si recordar devolverlo en un único elemento para no tener error de código.

Actualmente la visualización de nuestro proyecto sigue siendo la misma, ya que, aunque a pesar de tener la estructura para listar el menú y el contenido de cada sección no lo estamos vinculando. Para poder realizar esa vinculación cambiaremos la estructura de nuestra función principal App().



El uso de elementos como BrowserRouter, Switch , Route … son necesarios para el proceso de navegación y se declaran arriba del todo de nuestro fichero , nuevamente os dejo el código completo y la vista de nuestro proyecto actualmente así podéis ver la declaración de dichos elementos.

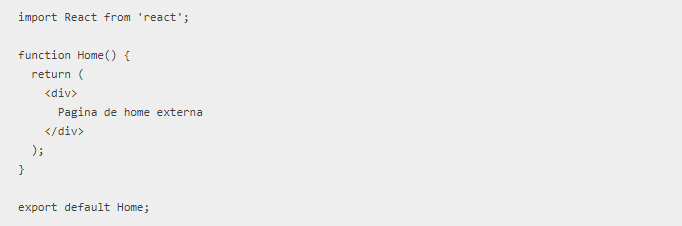






Como podéis comprobar actualmente tendríamos el proceso de navegación funcionando, pero si no queremos mezclar en el mismo fichero unas estructuras con otras podríamos tener el contenido de la navegación en ficheros a parte.

Veamos un ejemplo, para ello crearemos el fichero home.js que contendrá la información a mostrar en esa opción.



Lo incluiremos arriba en nuestro proyecto con el resto de los elementos que hemos incluido anteriormente

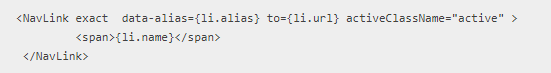


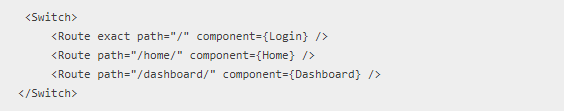
Esta es otra forma de contener estructura jsx en diferentes ficheros. Es una opción más limpia ya que podemos tener tanto ficheros como necesitemos y cada uno de ellos recogerá su información correspondiente. Ahora mismo se podría decir que tenemos nuestra navegación funcionando y en ficheros a parte, pero vamos a ver más cosas para completar este menú, como la funcionalidad de marcar como active el enlace del menú que estamos visualizando.

**4. Aplicar clase active**

Para poder destacar el enlace de menú que estamos visitando actualmente, usaremos la propiedad **exact** que es un booleano que nos ayudará ya que sirve para definir si la ruta será exacta o no.

Esta propiedad la colocaremos en dos lugares en los elementos navlink del menu y en los route que crean la vinculación. Si ambas son exactas pintaremos la clase active, veamos el ejemplo del código para dejarlo más claro.



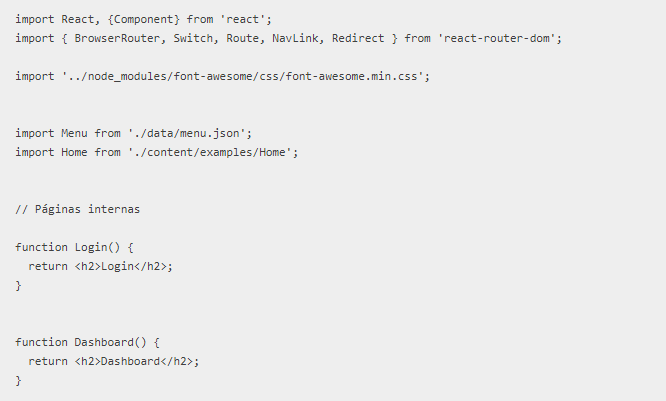


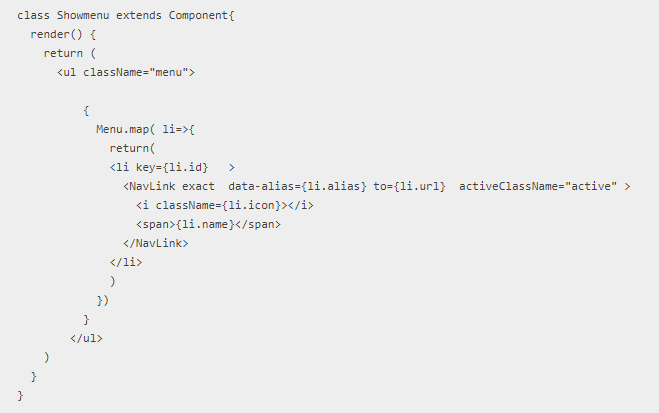
Si abrimos la consola de nuestro navegador y seleccionamos cualquier opción de menú veremos que la opción seleccionada contiene la clase active.

Continuemos completando el menú, para ello veremos cómo añadir iconos a los enlaces.

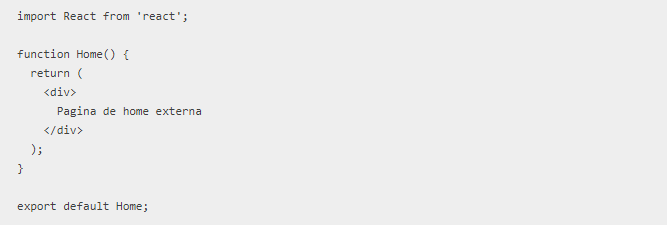
**5. Librería de iconos**

La librería de iconos que usaremos será font-awesome, para poder dar uso de la librería en nuestro proyecto debemos importarla.









Para completar nuestro tutorial sólo queda dar estilos al menú.

**6. Estilos de css en línea y externo**

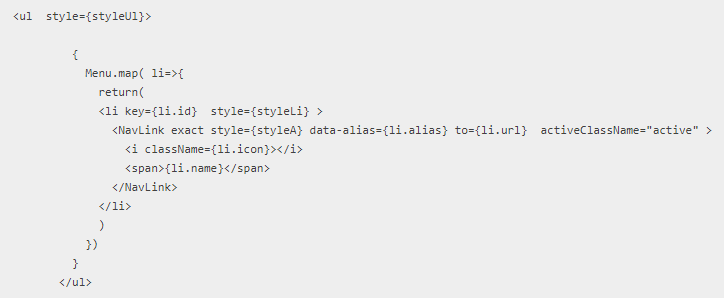
Podemos dar los estilos en línea o de una forma más limpia importando un fichero .css externo, veamos ambas:

CSS en línea: Para ello crearemos unas constantes dentro del render que contendrá el css que queremos añadir en línea a cada elemento.

Es un poco engorroso porque tenemos crear constantes diferente para cada elemento. Aquí algunos ejemplos.



Para usar esos estilos sólo necesitamos indicar que elemento llevará esas constantes



Si nos fijamos en el código, hemos añadido esas contantes a los elementos ul, li , NavLink. Como comentamos anteriormente también es posible hacerlo de forma externa.

CSS externo: Sólo necesitamos importar el fichero con el contenido css arriba del todo en el fichero principal del proyecto y añadir esas clases o estilos a los elementos con los que estamos trabajando.



Se ha llegado al final de esta guía, ha sido un poco extenso pero que servido de ayuda.

**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es navegabilidad web?
* ¿Quién se encarga de la navegabilidad de un sitio web?
* ¿Qué permiten los componentes en React JS? Dé una breve explicación
* Realice una App aplicando todo lo aprendido de esta guía

1. **ENLACES DE REFERENCIA**

* https://i.workana.com/glosario/que-es-navegabilidad-web/
* https://es.reactjs.org/docs/components-and-props.html
* https://www.nadiaprd.com/navegacion-con-react-js/

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_