DigitalHouse>
Coding School

Certified Tech Developer
The Ultimate Degree

Front End III

Glosario de React para repaso

1. **UI**: *User Interface* o interfaz de usuario. Una interfaz de usuario es el espacio donde ocurren las interacciones entre humanos y máquinas. No está limitada de ninguna manera a aplicaciones de *software* pues la necesidad de una interfaz de usuario existe en el campo del diseño industrial y donde sea necesaria la interacción humano-máquina. En *software*, la interfaz de usuario comprende los botones, cajas y áreas de texto, casillas de verificación (*checkboxes*), botones de radio (*radio-buttons*), y en general todas las formas de ingresar información. También incluye las partes donde se muestra información al usuario como etiquetas, gráficos, imágenes, iconos, etc.
2. **UX**: *User eXperience* o experiencia de usuario. Va más allá de la interfaz de usuario, y engloba los aspectos relacionados con la facilidad y rendimiento de la aplicación: los tiempos de respuesta; los mensajes al usuario; facilidad para encontrar la información; la cantidad de pasos para llegar a partes críticas de la aplicación; colores, tipo y tamaño de las fuentes, etc.
3. **JSX**: Javascript XML.
4. **Library**: se traduce como biblioteca y es el sitio donde se alojan colecciones de documentos para ser usados públicamente. En nuestro caso, una biblioteca de *software* es donde se aloja la funcionalidad común para que pueda ser compartida por múltiples aplicaciones, en lugar de estar repetida por cada aplicación. Por eso la traducción correcta es biblioteca y no librería (lugar donde se compran libros).

5. **Framework**: es una estructura básica de soporte para una construcción. En nuestro caso en particular, un framework de desarrollo es una estructura fija que sirve de marco para insertar la funcionalidad particular de la aplicación.
6. **Vanilla JavaScript**: el término *vanilla* (vainilla en castellano) se utiliza con mucha frecuencia en el mundo del software para referirse a algo puro sin agregados ni modificaciones.
7. **Componentes**: la programación orientada a componentes es una técnica de desarrollo de aplicaciones de software basada en la combinación de componentes nuevos y preexistentes, como en una fábrica. No es nueva en lo absoluto: similar a otros paradigmas modernos como la programación orientada a objetos (data de la década de los sesenta) y la programación funcional (data de la década de los cincuenta), la programación orientada a componentes data de la década de los sesenta. La idea esencial del componente es que este debe encapsular y controlar su propio estado.
8. **Estado de la aplicación**: es el conjunto de los valores de cada variable que maneja la aplicación en un momento dado. Por ejemplo, en un videojuego, el estado serían los valores de las variables que almacenan información como las vidas, la fuerza, las armas, las municiones, etc., tanto del personaje del jugador como de los enemigos, y también todo lo referente al nivel y al mapa.
9. **Front end**: es el código y el procesamiento de datos que ocurre en el navegador. No se limita a mostrar los datos en pantalla y a la parte visual, sino que también incluye llamados a APIs y el manejo del estado de toda la aplicación.
10. **Back end**: es el código y el procesamiento de datos que se hace fuera del navegador. El back end no es simplemente un servidor. Puede comprender varias computadoras o incluso máquinas virtuales que simulan computadoras reales.
11. **Arquitectura**: en front end en general, la arquitectura se refiere al modelo de la aplicación como un sistema. Incluye todos los componentes, cómo interactúan entre sí, el entorno en el que operan (contenedores, máquinas virtuales, plataformas en la nube) y los principios y decisiones utilizados para diseñarlos (OOP, FP, estándares). Normalmente, la arquitectura se separa en varias vistas o aspectos (concerns) destinadas a todos los stakeholders, desde los dueños del producto hasta los ingenieros.
12. **npm**: es, a la vez, un registro público y gratuito de código JavaScript y un manejador

de paquetes de Node.

13. **Yarn**: es un manejador de paquetes de Node que se puede utilizar para acceder al registro público de npm.
14. **Properties**: props de un componente. Son pares de clave/valor que se le pasan a los componentes. Son equivalentes los atributos que se le pasa a los componentes HTML, como por ejemplo, , la expresión *id="1"* es un par clave/valor, donde la clave es *id* y el valor es *1*.
15. **Boilerplate**: se trata de secciones de código (o cualquier texto en general) que se repiten en varios lugares sin cambios significativos, y que se pueden reutilizar en nuevos contextos o aplicaciones.
16. **ECMAScript**: es un lenguaje de programación de propósito general, estandarizado por Ecma International. Se trata de un estándar de JavaScript destinado a garantizar la interoperabilidad de las páginas web en diferentes navegadores web.
17. **DOM**: el modelo de objetos de documento es una interfaz multiplataforma e independiente de lenguaje de programación que trata a un documento XML o HTML como una estructura de árbol en la que cada nodo es un objeto que representa una parte del documento. El DOM representa un documento con un árbol lógico.
18. **Transpilador**: también llamado transcompilador, es un traductor que toma código fuente escrito en un lenguaje de programación como entrada y produce código fuente equivalente en el mismo lenguaje de programación o en uno diferente de salida.
19. **Compilador**: es un programa que traduce código de un lenguaje de programación a otro. A diferencia de un transpilador, el nombre compilador se utiliza principalmente para programas que traducen código fuente de un lenguaje de programación de alto nivel a uno de bajo nivel (comprendido por la máquina) para crear un programa ejecutable.
20. **Render**: la función render es donde React cambia el DOM. Para esto recopila la salida de las funciones createElement.

¡Hasta la próxima!