Checklist de Estilo

- Estándar basado en las reglas de estilo de Airbnb JavaScript Style Guide
- Referencia : https://github.com/airbnb/javascript
- Link al repositorio principal: https://github.com/Izzi-Connect-Tec

Reglas Básicas

1. Un componente por archivo: cada archivo debe contener solo un componente de React a menos que sea un componente puro o sin estado.
2. Extensiones de archivo: utilice .jsx para archivos que contengan componentes de React.
3. Nombrar archivos: use PascalCase al nombrar sus archivos componentes.
4. Referencia de componentes: consulte los componentes que usan PascalCase y sus instancias usando camelCase.
5. Nomenclatura coherente: asegúrese de que el nombre del componente coincida con el nombre del archivo.
6. Nomenclatura de accesorios: utilice nombres de accesorios específicos en lugar de genéricos para evitar redundancias.
7. Alineación en la sintaxis JSX: alinee la sintaxis JSX en varias líneas para mejorar la legibilidad.
8. Uso de comillas:
9. Los atributos JSX deben estar entre comillas dobles (") y el resto del JS entre comillas simples ('). El espaciado es importante, especialmente para etiquetas autocerradas que deben incluir un espacio. Cuando utilice llaves JSX para parámetros, recuerde no use espacios dentro de ellos. Estas son algunas de las reglas clave a tener en cuenta al trabajar con la sintaxis JSX.

Paréntesis

25. **JSX en Múltiples Líneas**: Envuelve etiquetas JSX en paréntesis si ocupan más de una línea.

Etiquetas

26. **Etiquetas Autocerradas**: Siempre autocierra etiquetas que no tengan hijos.

Métodos

27. **Funciones Flecha en Eventos**: Usa funciones flecha para cerrar sobre variables locales.
28. **No Usar Prefijos de Guión Bajo**: Camel Case para métodos.

Estilo General

Types

1. **Usar `===` y `!==`**: Prefiere `===` y `!==` sobre `==` y `!=`.
2. **Strings**: Evita constructores `String`, usa `String(value)` o `value.toString()`.
3. **Numbers**: Usa `Number(value)` para coerción y `parseInt(value, radix)` para analizar cadenas.
4. **Booleans**: Evita constructores `Boolean`, usa `Boolean(value)` o `!!value`.

Variables

6. **Usar `const`/`let`**: Prefiere `const` para variables inmutables y `let` para variables mutables.

Objetos

7. **Object Literal Shorthand**: Utiliza shorthand en literales de objetos cuando sea posible.
8. **Object Destructuring**: Usa destructuring para acceder a propiedades de objetos.

Arrays

9. **Array Destructuring**: Usa destructuring para acceder a elementos de arrays.

Funciones

10. **Function Expressions**: Prefiere expresiones de función sobre declaraciones.
11. **Arrow Functions**: Utiliza arrow functions cuando no se necesita vinculación a `this`.
12. **Argumentos de Función**: Evita argumentos de función con valores predeterminados.

Funciones Flecha

13. **Paréntesis alrededor de Parámetros**: Usa paréntesis alrededor de parámetros, incluso si hay solo uno.

Módulos

16. **ES6 Modules**: Prefiere `import`/`export` sobre `require`/`module.exports`.

Iteradores y Generadores

17. **Iteradores**: Utiliza iteradores y `for...of` para recorrer colecciones.

Declaraciones de Control

18. **Usar Llaves**: Siempre utiliza llaves en declaraciones de control (`if`, `else`, `for`, `while`, etc.).

Manejo de Errores

19. **Lanzar Errores**: Utiliza `throw` para lanzar errores explícitamente.

Comentarios

20. **Comentarios Significativos**: Usa comentarios significativos que expliquen el por qué.
21. **Evitar Comentarios Redundantes**: Evita comentarios que repitan el código.

Espaciado

1. **Indentación**: Usa espacios en lugar de tabs con una configuración de 2 espacios.
2. **Espacio antes de Llaves**: Coloca un espacio antes de llaves de apertura.
3. **Espacio en Declaraciones de Control**: Coloca un espacio antes del paréntesis de apertura.
4. **Espacios alrededor de Operadores**: Coloca espacios alrededor de operadores.
5. **Nueva Línea al Final del Archivo**: Termina los archivos con una sola nueva línea.
6. **Indentación en Cadenas de Métodos**: Indenta cadenas de métodos largas.
7. **Línea en Blanco después de Bloques**: Deja una línea en blanco después de bloques.
8. **No Espacios en Blanco Internos**: Evita líneas en blanco dentro de bloques de código.
9. **No Múltiples Líneas en Blanco**: Permite solo una nueva línea al final del archivo.
10. **Sin Espacios dentro de Paréntesis**: Evita espacios dentro de paréntesis.
11. **Sin Espacios dentro de Corchetes**: Evita espacios dentro de corchetes.
12. **Espacios dentro de Llaves**: Usa espacios dentro de llaves.
13. **Longitud de Línea**: Evita líneas de más de 100 caracteres.
14. **Espaciado Consistente en Bloques**: Asegura espaciado consistente dentro de bloques.
15. **Sin Espacios antes de Comas y Espacio después**: Evita espacios antes de comas y requiere un espacio después..
16. **Sin Espacios al Final de Líneas**: Evita espacios al final de líneas.

Comas

1. **Evitar Comas Líderes**: Evita comas al principio de la línea.
2. **Coma Final**: Incluye comas adicionales al final de listas y objetos.

Punto y Coma

1. **Punto y Coma**: Termina todas las declaraciones con punto y coma.

Nombres de Variables

1. **Nombres Descriptivos**: Evita nombres de una sola letra. Usa nombres descriptivos.
2. **camelCase**: Usa camelCase para objetos, funciones e instancias.
3. **PascalCase**: Usa PascalCase solo para constructores o clases.
4. **Sin Guiones Bajos**: Evita guiones bajos al principio o al final de los nombres.
5. **Sin Guardar Referencias a `this`**: Evita guardar referencias a `this`. Usa funciones flecha.
6. **Nombre del Archivo Coincide con Exportación**: El nombre del archivo debe coincidir con su exportación por defecto.
7. **PascalCase para Constructores/Clases**: Usa PascalCase para constructores/clases.
8. **Acrónimos e Inicialismos**: Siempre en mayúsculas o minúsculas completas.
9. **Mayúsculas para Constantes**: Opcionalmente, usa mayúsculas para constantes exportadas que sean `const` y confiables para nunca cambiar.