TEXT CLASS REVIEW

TEMAS A TRATAR EN LA CUE:

- Introducción.
- Dónde y cómo usar los lifeCycles Hooks.
- Definiendo los ciclos de vida.

INTRODUCCIÓN

Los enlaces del ciclo de vida son funciones, que se ejecutan en un cierto estado de éste, en la instancia Vue; cada componente en Vue tiene su ciclo de vida, en el cual podemos ejecutar diferente lógica según nuestras necesidades. Para ello, vamos a estudiarlas una a una, y así entender cuando utilizarlas.

¿DÓNDE Y CÓMO USAR LOS LIFECYCLES HOOKS?

Los enlaces de ciclo de vida del componente son funciones, las cuales son ejecutadas en un cierto momento de la vida de éste. Para poder definirlas, crearemos funciones en la raíz del objeto. Se han comentado las distintas partes o secciones de un componente; la parte final es donde debemos posicionar los enlaces de ciclo de vida (lifecycle hook).

```
export default {
  name: 'component-name',
  // props: {},
  data: function(){
    return {}
  },
  // computed: {},
  methods: {
    // -- Metodos
  },
  // components: {},
  //lifecycle hooks

}
```

```
//lifecycle hooks
beforeCreate(){
  console.log("beforeCreate");
},
created(){
  console.log("created");
},
beforeMount(){
  console.log("beforeMount");
},
mounted(){
  console.log("mounted");
},
beforeUpdate(){
  console.log("beforeUpdate");
},
updated(){
  cosole.log("updated");
},
beforeDestroy(){
  cosole.log("beforeDestroy");
},
destroyed(){
  console.log("destroy");
}
```

Como podemos observar, los ciclos de vida son funciones con un nombre que las identifica. Cada una de ellas se ejecuta en un momento determinado en la vida del componente.

DEFINIENDO LOS CICLOS DE VIDA:

- **beforeCreate():**

  Es el primer Hook en ser llamado, y se emplea antes de crear el componente. En esta parte del ciclo de vida, no podemos acceder a los datos y métodos; por lo que es útil para generar alguna acción que sirva sin necesidad de dicho componente.

- **created():**

  Es el segundo hook en ser llamado, y es ideal para generar una llamada a un endpoint; pues nos da un tiempo de cargado antes de que se renderice. En esta fase, aún no podemos leer DOM, pero si acceder a la data, a propiedades computadas, métodos, entre otros.

- **beforeMount():**

  Esta hook se llama justo antes de que el componente sea montado en el DOM, y de que la función “render” sea llamada por primera vez. Se puede utilizar para inicializar variables.

- **mounted():**

  En esta fase ya se generó el DOM, pero fue reemplazado por el VDOM, y nos puede servir para inicializar librerías externas. Aquí ya podemos manipular el DOM.

- **beforeUpdate():**

  Este ciclo de vida es llamado justo después de mounted hook, cuando un cambio es hecho en la data, y requiere que el DOM sea actualizado.

- **Updated():**

  Este ciclo de vida es llamado solo cuando la actualización del DOM ya ha ocurrido, pero aún no es visible para el usuario.

- **beforeDestroy():**

  Este lifecycle hook, es llamado antes de destruir el componente. Nos puede servir para limpiar ciertos eventos generados por librerías externas.

- **Destroyed():**

  Es el último lifecycle hook en ser llamado. En esta fase, ya el elemento ha sido destruido.