Clases Abstractas en Javascript

Sí, en JavaScript puedes diseñar un objeto abstracto usando la idea de clases abstractas o funciones constructoras que no se pueden instanciar directamente, pero que proporcionan una estructura para las subclases. JavaScript no tiene clases abstractas como tal (como en otros lenguajes de programación orientados a objetos), pero se puede simular este comportamiento. A continuación te muestro cómo hacerlo:

Ejemplo de objeto abstracto usando una clase en ES6

```
class AbstractProduct {
  constructor(serial, brand, model) {
    if (new.target === AbstractProduct) {
      throw new TypeError("Cannot instantiate an abstract class directly");
    }

    if (this.calculatePrice === undefined) {
      throw new TypeError("Must override method calculatePrice");
    }

    this.serial = serial;
    this.brand = brand;
    this.model = model;
  }

  getDetails() {
    return Serial: ${this.serial}, Brand: ${this.brand}, Model: ${this.model};
  }

  // Método abstracto
  calculatePrice() {
    throw new Error("Abstract method calculatePrice must be implemented");
  }
}
```

Explicación del código:

1. new.target en el constructor:
- Esta verificación evita que la clase AbstractProduct sea instanciada directamente. Si alguien intenta hacer new AbstractProduct(), se lanzará un error con el mensaje "Cannot instantiate an abstract class directly".

2. Método abstracto calculatePrice:
- La clase define un método calculatePrice que lanza un error si no es implementado en las subclases. Esto simula el comportamiento de un método abstracto, que debe ser sobrescrito en cualquier clase derivada.

3. getDetails():
- Es un método concreto, que puede ser utilizado por las subclases sin necesidad de redefinirlo.

Subclase que implementa la clase abstracta

Para utilizar AbstractProduct, necesitas crear una clase concreta que herede de ella y proporcione una implementación para el método abstracto calculatePrice:

```
class Phone extends AbstractProduct {
  constructor(serial, brand, model, price, tax) {
    super(serial, brand, model);
    this.price = price;
    this.tax = tax;
  }

  // Implementación del método abstracto
  calculatePrice() {
    return this.price + (this.price * this.tax / 100);
  }
}

const myPhone = new Phone("A123", "Apple", "iPhone 14", 1000, 21);
console.log(myPhone.getDetails()); // "Serial: A123, Brand: Apple, Model: iPhone 14"
console.log("Price with taxes: " + myPhone.calculatePrice()); // "Price with taxes: 1210"
```

Resumen del comportamiento:

1. Clase abstracta AbstractProduct:
- No se puede instanciar directamente.
- Contiene un método abstracto calculatePrice que obliga a las subclases a implementarlo.

2. Subclase Phone:
- Hereda de AbstractProduct e implementa el método calculatePrice.

Conclusión:
En este ejemplo, AbstractProduct actúa como una clase abstracta que no puede ser instanciada directamente y define la estructura que sus subclases deben seguir. Este patrón es útil para organizar el código y forzar a que las clases concretas implementen ciertos métodos clave.