JS-08

Introducción a la programación orientada a objetos

Objetivos de aprendizaje

- Explicar los principios SOLID.
- Definir una clase con atributos y métodos.
- Instanciar una clase e invocar sus métodos.
- Explicar el formato JSON y su importancia.
- Definir objetos JavaScript usando el formato JSON.

S – Single Responsability Principle (SRP)
(Principio de responsabilidad única)

O – Open/Closed Principle (OCP)
(Principio abierto / cerrado)

L – Liskov Substitution Principle (LSP)
(Principio de substitución de Liskov)

I – Interface Segretation Principle (ISP)
(Principio de segregación de interfaz)

D – Dependency Inversion Principle (DIP)
(Principio de inversión de dependencias)

Cinco principios básicos de la programación orientada a objetos

Los cinco principios de SOLID para el diseño de aplicaciones de software

Aplicar los principios de SOLID nos permitirá crear software más ordenado, limpio y fácil de mantener

Cohesión y Acoplamiento

El acoplamiento y la cohesión en programación son dos conceptos muy importantes que tienes que tener en cuenta cuando desarrollas software.

La cohesión en programación está estrechamente relacionada con el primer principio SOLID: el principio de responsabilidad única.

Acoplamiento
El acoplamiento en programación orientada a objetos se refiere al grado de independencia que tienen dos piezas de software entre sí.

Programación orientada a objetos

Es un paradigma de programación que utiliza la abstracción para crear modelos de objetos basados en el mundo real. Aplicando este paradigma podemos conseguir desarrollos más robustos y organizados además de conseguir beneficios tales como.

- Código reutilizable
- Fácil de mantener
- Código fácil de entender
- Desarrollo sencillo

Programación orientada a objetos

El paradigma de POO es considerado como el diseño de Software a través de conjuntos que cooperan entre sí, lo cual es muy diferente al antiguo paradigma de programación estructurada.

La idea detrás de esto, es crear un plano como molde o plantilla que tiene el aspecto de un “objeto” y esto se llama una y otra vez para hacer lo que quieras con él.

Cada vez que se quiera usar un objeto, primero se debe crear para que exista, y luego configurar sus propiedades para usar las funcionalidades adjuntas.

Estas funcionalidades son conocidas como ‘métodos’.

Clases

Las clases son un pilar fundamental de la programación orientada a objetos. Permiten abstraer los datos y sus operaciones asociadas al modo de una caja negra. Los lenguajes de programación que soportan clases difieren sutilmente en su soporte para diversas características relacionadas con clases. La mayoría soportan diversas formas de herencia.

```
class User {

}

let user = new User()
```

Principios de POO - Abstracción

Hace pensar que los objetos son similares a cajas negras, ya que sabremos cómo interactuar con ellos, pero no conocemos su comportamiento interno, esto trae como consecuencia la capacidad de modificar el comportamiento de un objeto sin afectar a quienes lo utilicen.

Principios de POO - Encapsulamiento

Cada objeto es responsable de su propia información, estados los cuales solo pueden ser modificados por sus propios métodos, por lo cual sus atributos internos no tienen que ser accesibles desde fuera.

Principios de POO - Herencia

Es la capacidad que permite crear objetos a partir de otros ya existentes, los métodos y atributos del objeto padre (Super Clase) pasan a ser parte de los objetos hijos (Sub-Clase) los cuales son creados a partir de la Super Clase, la herencia se basa en la reutilización de código.

Principios de POO - Polimorfismo

Es la capacidad que tiene un objeto de presentar diferentes comportamientos al momento de realizar una acción, el polimorfismo se presenta cuando se aplica el principio de Herencia.