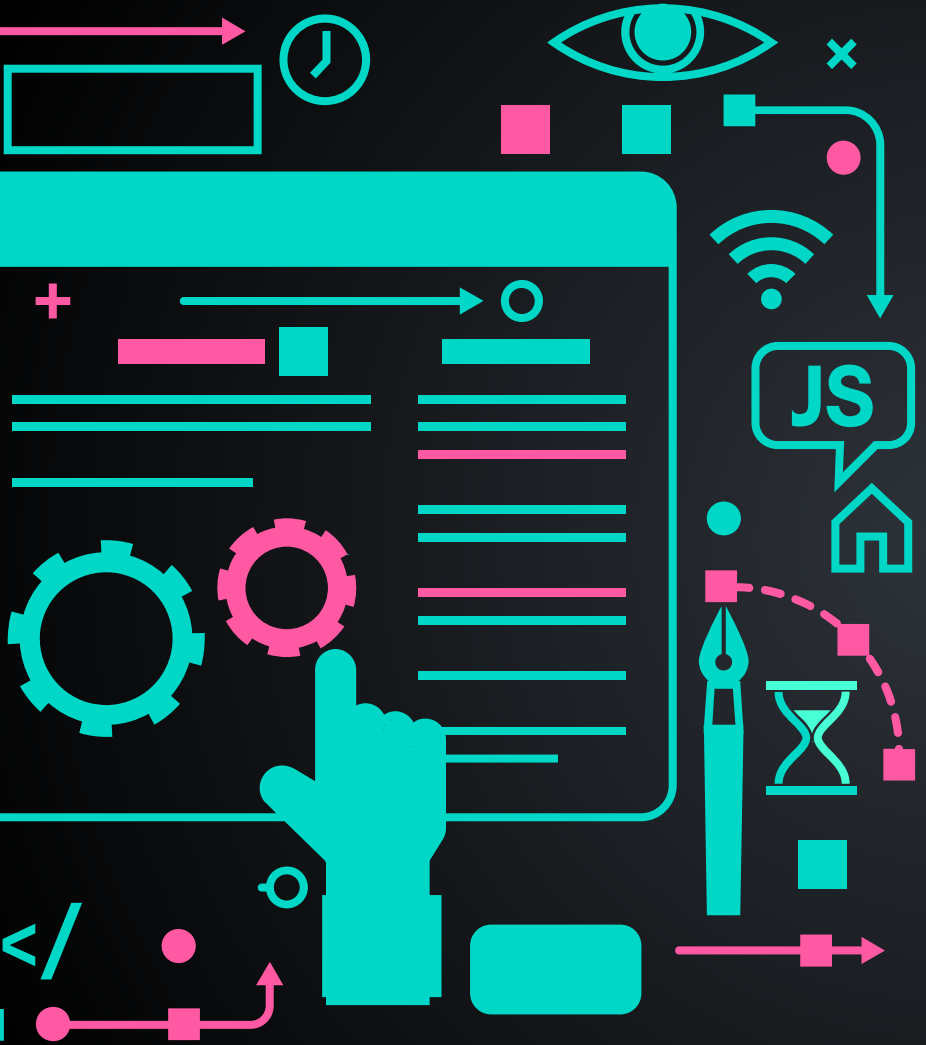

Javascript: Patrones de diseño

¿Que es un patrón de diseño?

Muchas veces escribimos código para resolver problemas.

Estos problemas suelen tener muchas similitudes y, al intentar solucionarlos, notamos varios patrones comunes.

Un patrón de diseño es un término utilizado en ingeniería de software para una solución general y reutilizable a un problema común en el diseño de software.

Porque utilizar patrones de diseño?

Los patrones de diseño son beneficiosos por varias razones:

- Son soluciones comprobadas que los veteranos de la industria han probado y probado.
- Son enfoques sólidos que resuelven problemas de una manera ampliamente aceptada y reflejan la experiencia y los conocimientos de los desarrolladores líderes en la industria que ayudaron a definirlos.
- Los patrones también hacen que su código sea más reutilizable y legible al tiempo que acelera enormemente el proceso de desarrollo.

Algo muy importante para tener en cuenta.

Los patrones de diseño no son de ninguna manera soluciones terminadas. Solo nos brindan enfoques o esquemas para resolver un problema.

Se pueden dividir en tres tipos.

1. Creacionales / creationals.
2. Estructurales / Estructurals.
3. Conductuales / Behaviorals.

Patrones Creacionales.

Estos patrones manejan la forma en la cual se construyen los objetos.

1. Ejemplos de estos:
2. Abstract Factory - Crea una instancia de varias familias de clases.
3. Builder - Separa la construcción de objetos de su representación.
4. Factory Method - Crea una instancia de varias clases derivadas.
5. Prototype - Una instancia completamente inicializada para ser copiada o clonada.
6. Singleton - Una clase de la que solo puede existir una instancia

Patrones Estructurales.

Ayudan a obtener nuevas funcionalidades sin alterar las existentes.

Ejemplos de estos:

- Adapter - Emparejar interfaces de diferentes clases.
- Bridge - Separa la interfaz de un objeto de su implementación.
- Composite - Una estructura de árbol de objetos simples y compuestos.
- Decorator - Agregue responsabilidades a los objetos de forma dinámica.
- Proxy - Un objeto que representa a otro objeto.

Patrones Conductuales.

Estos patrones están relacionados con la mejora de la comunicación entre objetos diferentes.

Ejemplos de estos:

- Iterator - Una forma de incluir elementos del lenguaje en un programa.
- Mediator - Define la comunicación simplificada entre clases.
- Observer - Una forma de notificar cambios a varias clases.
- Strategy - Encapsula un algoritmo dentro de una clase.
- Visitor - Define una nueva operación para una clase sin cambios.

Ejercicio grupal

Ver

¿Que son los web components?